



เอกสารประกอบการสอน  
การพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่

ณปภัช วรรณตรง

สาขาวิชาวิทยาการคอมพิวเตอร์  
คณะวิทยาศาสตร์ มหาวิทยาลัยราชภัฏบุรีรัมย์

## คำนำ

เอกสารประกอบการสอนรายวิชา การพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่ รหัสวิชา 4123305 เป็นรายวิชาหนึ่งที่สำคัญของหลักสูตรสาขาวิชาวิทยาการคอมพิวเตอร์ เนื่องจากปัจจุบันอุปกรณ์เคลื่อนที่ โดยเฉพาะโทรศัพท์มือถือกลายเป็นส่วนสำคัญในชีวิตประจำวันสำหรับผู้คนในปัจจุบันจนถูกกล่าวว่าเป็น อวัยวะที่ 33 ส่งผลให้มีความจำเป็นที่นักศึกษาจะต้องเรียนรู้เพื่อนำไปใช้ในการทำงานต่อไปในอนาคต ผู้จัดทำ จึงจัดทำเอกสารประกอบการสอนนี้ขึ้นเพื่อใช้ประกอบการเรียนการสอนรายวิชา การพัฒนาแอปพลิเคชันบน อุปกรณ์เคลื่อนที่ เพื่อให้นักศึกษาได้รับเนื้อหาที่เป็นประโยชน์ สามารถนำไปประยุกต์ใช้ในการทำงานใน อนาคต

เอกสารประกอบการสอนรายวิชา การพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่ ผู้เขียนได้เขียนเนื้อหา ครบตามหลักสูตรที่กำหนดไว้ และได้เพิ่มเติมเทคนิคและแหล่งค้นคว้าเพิ่มเติมที่ควรรู้ให้กับนักศึกษาที่สามารถ นำไปใช้ในการทำงานจริงในอนาคตได้ อาทิ เว็บไซต์ flutter.dev เว็บไซต์ Stack Overflow นอกจากนี้เนื้อหา ที่รวบรวม สัเคราะห์มาในเอกสารประกอบการสอน มาจากการศึกษา สัเคราะห์เนื้อหาจากเอกสาร ตำรา หนังสือ เว็บไซต์ ทั้งไทยและต่างประเทศเพื่อให้นักศึกษาได้ความรู้ที่ครอบคลุมจากแหล่งความรู้ที่หลากหลาย

การจัดทำเอกสารประกอบการสอนนี้สำเร็จลงได้ ต้องขอบพระคุณพ่อ แม่ ครูบาอาจารย์ที่เคยสิทธิ์ี ประสาทวิชา และเพื่อน ๆ พี่ ๆ อาจารย์มหาวิทยาลัยราชภัฏบุรีรัมย์ที่สนับสนุน เสริมแรงในการจัดทำเอกสาร ประกอบการสอน และท่านผู้ทรงคุณวุฒิที่ช่วยเสียสละเวลาอันมีค่าในการตรวจทาน ให้คำชี้แนะอันเป็น ประโยชน์ ทำให้เอกสารประกอบการสอน รายวิชาระบบสารสนเทศเพื่อการจัดการนี้ มีความสมบูรณ์ยิ่งขึ้น

ณปภัช วรรณตรง

# สารบัญ

เรื่อง	หน้า
คำนำ.....	(1)
สารบัญ.....	(3)
สารบัญภาพประกอบ.....	(9)
สารบัญตาราง.....	(38)
แผนบริหารการสอนรายวิชา.....	(39)
แผนบริหารการสอนประจำบทที่ 1.....	1
บทที่ 1 Introduction to Mobile Development.....	3
ทำความรู้จักกับ Flutter & Dart.....	3
การพัฒนาแอปพลิเคชันแบบ Cross Platform.....	4
ทำความรู้จักกับ Flutter.....	5
จุดเด่นของ Flutter.....	6
การเริ่มต้นสร้างแอปพลิเคชันด้วย Flutter.....	15
ลงโปรแกรมที่เกี่ยวข้อง.....	15
กำหนดค่าในการทำงานของโปรแกรม Set Path.....	21
ตรวจสอบการใช้งานด้วย Command Prompt.....	27
เริ่มสร้าง Project.....	29
เปิดไฟล์ใน VS Code.....	33
บทสรุป.....	38
เอกสารอ้างอิง.....	38
คำถามท้ายบทที่ 1.....	39
แผนบริหารการสอนประจำบทที่ 2.....	41
บทที่ 2 การใช้งานวิดเจ็ต (Widget).....	43
ทำความรู้จักวิดเจ็ตคืออะไร.....	43
วิดเจ็ต (Widget) คืออะไร.....	43
ประเภทของ Widget.....	43

## สารบัญ (ต่อ)

เรื่อง	หน้า
วิดเจ็ตสร้างขึ้นเอง (Custom Widget).....	44
วิดเจ็ตที่ใช้บ่อย.....	46
เริ่มขั้นตอนการสร้าง Project.....	51
จัด Layout หน้าแอปพลิเคชันด้วย Scaffold Widget.....	65
บทสรุป.....	77
เอกสารอ้างอิง.....	77
คำถามท้ายบทที่ 2.....	78
<b>แผนบริหารการสอนประจำบทที่ 3.....</b>	<b>80</b>
<b>บทที่ 3 ภาษา Dart (Dart Language).....</b>	<b>81</b>
ภาษา Dart คืออะไร.....	81
ประวัติภาษา Dart.....	81
ชนิดตัวแปรในภาษา Dart.....	82
ข้อแตกต่างระหว่าง Dynamic vs Var คือ.....	83
ข้อแตกต่างระหว่าง final vs const คือ.....	83
Operator.....	84
If/Else, Loop, Switch Case.....	89
Switch..Case.....	91
Loop.....	92
For Loop.....	92
Do While.....	93
Function.....	94
ประเภทของ Function.....	95
List.....	100
การจัดการข้อมูลในลิสต์.....	102
รายการลิสต์.....	103



## สารบัญ (ต่อ)

เรื่อง	หน้า
Null Safety.....	106
บทสรุป.....	109
เอกสารอ้างอิง.....	110
คำถามท้ายบทที่ 3.....	111
<b>แผนบริหารการสอนประจำบทที่ 4.....</b>	<b>112</b>
<b>บทที่ 4 การสร้างแอปพลิเคชันคำนวณ.....</b>	<b>114</b>
Layout Widget.....	114
กำหนด Layout.....	114
สร้างโปรเจกต์ Calculate.....	114
แอปบาร์ (AppBar).....	125
ใส่ AppBar Widget.....	126
Column และ Row Widget.....	144
ใส่รูปภาพ.....	149
SizedBox.....	163
เปลี่ยนรูปแบบฟอนต์ (Font).....	166
ตกแต่งแอปพลิเคชันด้วย Padding.....	173
TextField.....	175
ListView Widget.....	195
บทสรุป.....	223
เอกสารอ้างอิง.....	224
คำถามท้ายบทที่ 4.....	225

## สารบัญ (ต่อ)

เรื่อง	หน้า
แผนบริหารการสอนประจำบทที่ 5.....	226
<b>บทที่ 5 การสร้างหน้าแอปพลิเคชันและแท็บบาร์สำหรับผู้ใช้งาน.....</b>	<b>228</b>
การสร้างหน้าแอปพลิเคชันแยกแต่ละหน้า.....	228
BottomNavigationBar.....	260
การสร้างแท็บบาร์ที่ด้านล่างของแอปพลิเคชัน.....	281
การสร้างไอคอนของแอปพลิเคชัน.....	260
ทำการสร้างไฟล์สำหรับนำไปติดตั้งและใช้งานบนมือถือไฟล์ .apk.....	303
บทสรุป.....	306
เอกสารอ้างอิง.....	307
คำถามท้ายบทที่ 5.....	308
<b>แผนบริหารการสอนประจำบทที่ 6.....</b>	<b>309</b>
<b>บทที่ 6 การสร้างฐานข้อมูลสำหรับการใช้งานด้วย Python Django.....</b>	<b>311</b>
Django คืออะไร.....	311
ประวัติความเป็นมา Django.....	312
คุณลักษณะที่น่าสนใจของ Django.....	312
สถาปัตยกรรม Django.....	313
ติดตั้ง Django.....	315
สร้าง API เพื่อทำการส่ง json File ใน Django.....	324
API (Application Programming Interfaces) คืออะไร.....	324
เจสัน (JSON) คืออะไร.....	324
การใช้งาน ngrok Forward Port.....	333
ทดสอบระบบเดิมที่ได้ทำการพัฒนาไว้.....	335
บทสรุป.....	336
เอกสารอ้างอิง.....	337
คำถามท้ายบทที่ 6.....	338

## สารบัญ (ต่อ)

เรื่อง	หน้า
แผนบริหารการสอนประจำบทที่ 7 .....	339
บทที่ 7 การสร้าง RESTful APIs ด้วย Django REST Framework .....	341
Django REST Framework .....	341
การสร้างข้อมูลในฐานข้อมูล .....	342
ทดสอบเข้าดูฐานข้อมูลที่มีใน DB Browser for SQLite .....	348
เริ่มต้นติดตั้ง Django REST Framework .....	350
สร้างไฟล์ Serializers .....	352
โปรแกรม Postman .....	357
บทสรุป .....	363
เอกสารอ้างอิง .....	363
คำถามท้ายบทที่ 7 .....	364
แผนบริหารการสอนประจำบทที่ 8 .....	365
บทที่ 8 การใช้งาน Firebase Database ร่วมกับ Flutter .....	367
และตัวอย่างการประยุกต์ใช้	
Firebase .....	367
Firebase คืออะไร .....	367
บริการด้านฐานข้อมูลของ Firebase .....	368
โครงสร้างข้อมูลของ Firestore .....	368
ฐานข้อมูล NoSQL คืออะไร .....	368
การใช้งาน Flutter ร่วมกับ Firebase .....	370
การติดตั้ง Library ในโปรเจกต์ Flutter .....	377
การใช้งาน Cloud Firestore .....	380
การแสดงผลข้อมูลจาก Firestore Database .....	388
การใช้งาน Firebase Authentication .....	392
การใช้งาน image_picker ร่วมกับ Storage .....	402
การสร้างแอปพลิเคชัน Chat ด้วย Firebase .....	418
บทสรุป .....	426

## สารบัญ (ต่อ)

เรื่อง	หน้า
เอกสารอ้างอิง.....	426
คำถามท้ายบทที่ 8.....	427
บรรณานุกรม.....	428

## สารบัญภาพประกอบ

ภาพประกอบ	หน้า
1.1 การพัฒนาระบบในอดีต.....	4
1.2 Flutter.....	6
1.3 ภาษา Dart.....	6
1.4 หน้าเว็บไซต์ของ Flutter.....	7
1.5 ความต้องการของระบบ.....	8
1.6 เริ่มต้นการใช้งาน Flutter (1).....	8
1.7 เริ่มต้นการใช้งาน Flutter (2).....	9
1.8 เริ่มต้นการใช้งาน Flutter (3).....	9
1.9 เริ่มต้นการใช้งาน Flutter (4).....	10
1.10 เริ่มต้นการใช้งาน Flutter (5).....	10
1.11 เริ่มต้นการใช้งาน Flutter (6).....	11
1.12 เริ่มต้นการใช้งาน Flutter (7).....	11
1.13 เริ่มต้นการใช้งาน Flutter (8).....	12
1.14 เริ่มต้นการใช้งาน Flutter (9).....	13
1.15 เริ่มต้นการใช้งาน Flutter (10).....	14
1.16 Flutter สำหรับระบบปฏิบัติการ Windows.....	15
1.17 Flutter สำหรับระบบปฏิบัติการ macOS.....	15
1.18 Android Studio สำหรับระบบปฏิบัติการ Windows.....	16
1.19 Android Studio สำหรับระบบปฏิบัติการ macOS.....	16
1.20 ตั้งค่าเครื่องมือสำหรับการพัฒนา.....	17
1.21 ตั้งค่าเครื่องมือสำหรับการพัฒนา (ต่อ).....	17
1.22 ตั้งค่าเครื่องมือสำหรับการพัฒนา (ต่อ).....	18
1.23 ตั้งค่าเครื่องมือสำหรับการพัฒนา (ต่อ).....	18
1.24 Virtual Studio Code สำหรับทุกระบบปฏิบัติการ.....	19
1.25 ส่วนเสริมการใช้งานผ่าน Virtual Studio Code.....	20
1.26 กำหนดค่า ANDROID_HOME ใน Edit Environment.....	21

## สารบัญภาพประกอบ (ต่อ)

1.27	กำหนดค่า ANDROID_HOME ใน Edit Environment (ต่อ).....	21
1.28	กำหนดค่า ANDROID_HOME ใน Edit Environment (ต่อ).....	22
1.29	สร้างและกำหนดค่า ANDROID_HOME.....	22
1.30	สร้าง Path ใหม่.....	23
1.31	สร้าง Path ใหม่ (ต่อ).....	24
1.32	สร้าง Path ใหม่ (ต่อ).....	24
1.33	สร้าง Path ใหม่ (ต่อ).....	25
1.34	สร้าง Path ใหม่ (ต่อ).....	25
1.35	สร้าง Path ใหม่ (ต่อ).....	26
1.36	สร้าง Path ใหม่ (ต่อ).....	26
1.37	Flutter.....	27
1.38	Flutter Doctor.....	27
1.39	สร้าง Folder.....	29
1.40	cd เข้า Folder ที่สร้าง.....	29
1.41	สร้าง Project.....	30
1.42	สร้าง Project (ต่อ).....	30
1.43	ใช้คำสั่ง cd firstapp เพื่อเข้าสู่ Folder Project.....	31
1.44	ใช้คำสั่ง Flutter run เพื่อสั่ง run project.....	31
1.45	เลือก Device.....	32
1.46	เลือก Device (ต่อ).....	32
1.47	เปิด VS Code.....	33
1.48	เปิด VS Code (ต่อ).....	33
1.49	เปิด Folder lib.....	34
1.50	แก้ไขไฟล์ main.dart.....	34
1.51	Restart.....	35

## สารบัญภาพประกอบ (ต่อ)

1.52	Restart (ต่อ).....	35
1.53	ผลลัพธ์การแก้ไขโค้ด home:.....	36
1.54	ยกเลิกการทำงานเมื่อเกิดอาการค้าง.....	37
1.56	ยกเลิกการทำงานเมื่อเกิดอาการค้าง (ต่อ).....	37
2.1	ตัวอย่างโค้ดแบบ Stateless Widget.....	45
2.2	ตัวอย่างโค้ดแบบ Stateful Widget (ต่อ).....	45
2.3	ตัวอย่างโค้ดแบบ Scaffold Widget (ต่อ).....	47
2.4	ตัวอย่างโค้ดแบบ Center Widget.....	47
2.5	ตัวอย่างโค้ดแบบ Column Widget (ต่อ).....	48
2.6	ตัวอย่างโค้ดแบบ Row Widget.....	49
2.7	ตัวอย่างโค้ดแบบ Container Widget.....	50
2.8	เข้าหน้าต่าง cmd ผ่าน Path ของ Folder.....	51
2.9	เข้าหน้าต่าง cmd ผ่าน Path ของ Folder (ต่อ).....	51
2.10	เข้าหน้าต่าง cmd ผ่าน Path ของ Folder (ต่อ).....	51
2.11	พิมพ์คำสั่งสำหรับสร้างไฟล์ Flutter > flutter doctor -v.....	52
2.12	พิมพ์คำสั่งสำหรับสร้างไฟล์ Flutter > flutter doctor -v (ต่อ).....	52
2.13	พิมพ์คำสั่งสำหรับสร้างไฟล์ Second App.....	53
2.14	พิมพ์คำสั่งสำหรับสร้างไฟล์ Second App (ต่อ).....	53
2.15	ผลการสร้างไฟล์งาน Second App.....	53
2.16	เข้าไปยัง Folder งาน.....	54
2.17	เข้าไปยัง Folder งาน (ต่อ).....	54
2.18	สั่งการทำงานด้วย flutter run.....	54
2.19	สั่งการทำงานด้วย flutter run (ต่อ).....	55
2.20	เลือก Devices.....	55
2.21	เลือก Devices (ต่อ).....	56
2.22	ผลลัพธ์การสร้างไฟล์ Flutter Project.....	56
2.23	ดูแต่ละส่วนประกอบบนหน้าแอปพลิเคชัน.....	57

## สารบัญภาพประกอบ (ต่อ)

2.24	ทำความเข้าใจองค์ประกอบต่าง ๆ.....	58
2.25	ลบบอร์ดที่ไม่ได้ใช้ออก.....	59
2.26	แก้ไขโค้ดส่วน Title.....	59
2.27	แก้ไขโค้ดส่วน Title (ต่อ).....	60
2.28	แก้ไขจำนวนตัวเลขโค้ด int_counter.....	60
2.29	Restart Flutter.....	61
2.30	ผลลัพธ์การแก้ไขโค้ดเบื้องต้น.....	61
2.31	ครอบเลือกส่วน Scaffold(..).....	62
2.32	ครอบเลือกส่วน Scaffold(..) (ต่อ).....	63
2.33	ครอบเลือกส่วน Scaffold(..) (ต่อ).....	63
2.34	ลบส่วน Scaffold ออก.....	64
2.35	เพิ่มโค้ด Text(...) เข้าไป.....	64
2.36	เพิ่มข้อความเข้าไปใน Text(...).....	65
2.37	เพิ่มข้อความเข้าไปใน Text(...) (ต่อ).....	65
2.38	ลบส่วน appBar ออก.....	66
2.39	ลบส่วน appBar ออก (ต่อ).....	66
2.40	ส่วนที่เหลือจากการลบ appBar ออก.....	67
2.41	ลบส่วน Center(...) ออก.....	67
2.42	ลบส่วน Center(...) ออก (ต่อ).....	68
2.43	เพิ่มโค้ด Text(...).....	68
2.44	แก้ไขโค้ด floatingActionButton.....	69
2.45	ผลลัพธ์จากการแก้ไขโค้ด.....	69
2.46	กำหนด Style ให้กับ Text.....	70
2.47	กำหนด Style ให้กับ Text (ต่อ).....	70
2.48	เพิ่มโค้ด fontSize:.....	71
2.49	เพิ่มโค้ด fontWeight:.....	71
2.50	เพิ่มโค้ด fontWeight: (ต่อ).....	72
2.51	เพิ่มโค้ด color:.....	72



## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
2.52 เพิ่มโค้ด color: (ต่อ).....	72
2.53 กำหนด style ให้กับ text.....	72
2.54 ตัดส่วน Text(...) ไว้.....	73
2.55 เพิ่มโค้ด Center(...).....	73
2.56 เพิ่มโค้ดและวางโค้ดที่ตัดไว้เข้าไปใน Center(...).....	74
2.57 เพิ่มโค้ดและวางโค้ดที่ตัดไว้เข้าไปใน Center(...) (ต่อ).....	74
2.58 กำหนด text ให้อยู่ตรงกลาง.....	74
2.59 ดาวน์โหลด Font ที่ต้องการจะใช้งานในแอปพลิเคชัน.....	75
2.60 สร้าง Folder “font” สำหรับใช้เก็บ Font ที่ทำการดาวน์โหลดมา.....	75
2.61 แก้ไขไฟล์ที่มีชื่อว่า pubspec.yaml ตามภาพ.....	75
2.62 แก้ไขโค้ดโดยการใส่ fontFamily ที่สร้างใน main.dart ตามภาพ.....	76
2.63 ผลลัพธ์การสร้างแอปพลิเคชันจาก Flutter.....	76
3.1 ตัวอย่างการใช้งาน String methods.....	84
3.2 ผลลัพธ์การใช้งาน String methods.....	84
3.3 ตัวอย่างการใช้งาน Arithmetic Operators Operator.....	85
3.4 ผลลัพธ์การใช้งาน Arithmetic Operators Operator.....	85
3.5 ตัวอย่างการใช้งาน Type test Operators.....	85
3.6 ผลลัพธ์การใช้งาน Type test Operators.....	86
3.7 ตัวอย่างการใช้งาน Equality and Relational Operators.....	86
3.8 ผลลัพธ์การใช้งาน Equality and Relational Operators.....	86
3.9 ตัวอย่างการใช้งาน Assignment Operators.....	87
3.10 ผลลัพธ์การใช้งาน Assignment Operators.....	88
3.11 ตัวอย่างการใช้งาน Logical Operators.....	88
3.12 ผลลัพธ์การใช้งาน Logical Operators.....	89
3.13 ตัวอย่างการใช้งาน if แบบเงื่อนไขเดียว.....	89

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
3.14 ผลลัพธ์การใช้งาน if แบบเงื่อนไขเดียว.....	89
3.15 ตัวอย่างการใช้งาน if แบบ 2 เงื่อนไขเดียว.....	90
3.16 ผลลัพธ์การใช้งาน if แบบ 2 เงื่อนไขเดียว.....	90
3.17 ตัวอย่างการใช้งาน if แบบหลายเงื่อนไข.....	90
3.18 ผลลัพธ์การใช้งาน if แบบหลายเงื่อนไข.....	91
3.19 ตัวอย่างการใช้งาน Switch..Case.....	91
3.20 ผลลัพธ์การใช้งาน Switch..Case.....	91
3.21 ตัวอย่างการใช้งาน While.....	92
3.22 ผลลัพธ์การใช้งาน While.....	92
3.23 ตัวอย่างการใช้งาน For.....	93
3.24 ผลลัพธ์การใช้งาน For.....	93
3.25 ตัวอย่างการใช้งาน Do.While.....	94
3.26 ผลลัพธ์การใช้งาน Do.While.....	94
3.27 ตัวอย่างการใช้งาน Function.....	94
3.28 ตัวอย่างการใช้งาน Arrow Function.....	95
3.29 ตัวอย่างการใช้งาน Function : No return & No Args.....	95
3.30 ผลลัพธ์การใช้งาน Function : No return & No Args.....	95
3.31 ตัวอย่างการใช้งาน Function : No return & มี Args.....	96
3.32 ผลลัพธ์การใช้งาน Function : No return & มี Args.....	96
3.33 ตัวอย่างการใช้งาน Function : มี return & No Args.....	96
3.34 ผลลัพธ์การใช้งาน Function : มี return & No Args.....	97
3.35 ตัวอย่างการใช้งาน Function : มี return & มี Args.....	97
3.36 ผลลัพธ์การใช้งาน Function : มี return & มี Args.....	97
3.37 ตัวอย่างการใช้งาน Optional Positional Argument Optional.....	98
3.38 ผลลัพธ์การใช้งาน Optional Positional Argument Optional.....	98
3.39 ตัวอย่างการใช้งาน Optional Named Argument.....	99

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
3.40 ผลลัพธ์การใช้งาน Optional Named Argument.....	99
3.41 ตัวอย่างการใช้งาน Fixed-length (Array).....	100
3.42 ผลลัพธ์การใช้งาน Fixed-length (Array).....	100
3.43 ตัวอย่างการใช้งาน Growable (Link List).....	101
3.44 ผลลัพธ์การใช้งาน Growable (Link List).....	101
3.45 ตัวอย่างการเพิ่มรายการในลิสต์.....	102
3.46 ผลลัพธ์การเพิ่มรายการในลิสต์.....	102
3.47 ตัวอย่างการแทรกข้อมูลภายใน List.....	103
3.48 ผลลัพธ์ข้อมูลภายใน List.....	104
3.49 ตัวอย่างการจัดเรียงข้อมูลภายใน List.....	104
3.50 ผลลัพธ์การจัดเรียงข้อมูลภายใน List.....	104
3.51 ตัวอย่างการใช้งาน List for Each.....	105
3.52 ผลลัพธ์การใช้งาน List for Each.....	105
3.53 ตัวอย่างการใช้งาน List for Loop.....	106
3.54 ผลลัพธ์การใช้งาน List for Loop.....	106
3.55 ตัวอย่างการใช้งานการประกาศตัวแปรให้สามารถมีค่าเป็น Null ได้.....	107
3.56 ผลลัพธ์การใช้งานการประกาศตัวแปรให้สามารถมีค่าเป็น Null ได้.....	107
3.57 ตัวอย่างการประกาศตัวแปร List.....	108
3.58 ผลลัพธ์การใช้งานการประกาศตัวแปร List.....	108
3.59 ตัวอย่างการใช้งาน Null Safety เมื่อใช้งานกับฟังก์ชัน.....	109
3.60 ผลลัพธ์การใช้งาน Null Safety เมื่อใช้งานกับฟังก์ชัน.....	109
4.1 เปิดหน้าต่าง cmd และใช้คำสั่ง cd.....	115
4.2 เปิดหน้าต่าง cmd โดยการพิมพ์ cmd ในช่อง Path.....	115
4.3 เปิดหน้าต่าง cmd โดยการพิมพ์ cmd ในช่อง Path (ต่อ).....	115
4.4 พิมพ์คำสั่งสำหรับสร้างโปรเจกต์ Calculate.....	116
4.5 เมื่อสร้างไฟล์ Project Calculate เสร็จสิ้น.....	116

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
4.6 เมื่อสร้างไฟล์ Project Calculate เสร็จสิ้น (ต่อ).....	116
4.7 ไฟล์ใน Folder Calculate.....	117
4.8 เปิดโปรแกรม VSCode.....	118
4.9 เปิดไฟล์ Project Calculate.....	118
4.10 เปิดไฟล์ Project Calculate (ต่อ).....	119
4.11 เปิดไฟล์ Project Calculate (ต่อ).....	119
4.12 ลบโค้ดที่ไม่ได้ใช้ออกจากไฟล์ main.dart.....	120
4.13 ลบโค้ดที่ไม่ได้ใช้ออกจากไฟล์ main.dart (ต่อ).....	120
4.14 สร้างฟังก์ชัน void main(){}.....	121
4.15 สร้างฟังก์ชัน void main(){} (ต่อ).....	121
4.16 สร้าง class StatelessWidget(){}.....	122
4.17 สร้าง class StatelessWidget(){} (ต่อ).....	122
4.18 ตั้งชื่อของ class StatelessWidget(){}.....	123
4.19 ตั้งชื่อของ class StatelessWidget(){} (ต่อ).....	123
4.20 เปลี่ยนชื่อ class Container เป็น class MaterialApp.....	124
4.21 เปลี่ยนชื่อ class Container เป็น class MaterialApp.....	124
4.22 เพิ่มโค้ด home: Scaffold(),.....	124
4.23 เพิ่มโค้ด home: Scaffold(), (ต่อ).....	125
4.24 เพิ่มโค้ด home: Scaffold(), (ต่อ).....	125
4.25 เพิ่มโค้ด appBar: , .....	127
4.26 เพิ่ม AppBar() เข้าไปใน Tag appBar: ,.....	127
4.27 เพิ่ม AppBar() เข้าไปใน Tag appBar: , (ต่อ).....	128
4.28 เพิ่ม title: , เข้าไปใน Tag AppBar().....	128
4.29 เพิ่ม title: , เข้าไปใน Tag AppBar() (ต่อ).....	129
4.30 เพิ่ม Text(), เข้าไปใน Tag title: ,.....	129
4.31 เพิ่ม Text(), เข้าไปใน Tag title: , (ต่อ).....	130

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
4.32 เปลี่ยนข้อความใน Text(),.....	130
4.33 เปลี่ยนข้อความใน Text(), (ต่อ).....	130
4.34 เพิ่มโค้ดส่วนของ Scaffold(),.....	131
4.35 เพิ่มโค้ด body: ,.....	131
4.36 สร้าง class StatefulWidget.....	132
4.37 สร้าง class StatefulWidget (ต่อ).....	132
4.38 สร้าง class StatefulWidget (ต่อ).....	133
4.39 เพิ่มโค้ด Home().....	133
4.40 เรียกใช้งาน class MyApp(){}.....	134
4.41 เรียกใช้งาน class MyApp(){} (ต่อ).....	134
4.42 เรียกใช้งาน class MyApp(){} (ต่อ).....	135
4.43 เพิ่มโค้ด Text(“Home Page”);.....	135
4.44 เพิ่มโค้ด Text(“Home Page”); (ต่อ).....	135
4.45 เพิ่มโค้ด Text(“Home Page”); (ต่อ).....	136
4.46 เพิ่มโค้ด Text(“Home Page”); (ต่อ).....	136
4.47 สั่งการทำงาน Flutter.....	136
4.48 สั่งการทำงาน Flutter (ต่อ).....	137
4.49 สั่งการทำงาน Flutter (ต่อ).....	137
4.50 สั่งการทำงาน Flutter (ต่อ).....	138
4.51 เพิ่มโค้ด Center();.....	138
4.52 เพิ่มโค้ด Center(); (ต่อ).....	139
4.53 เพิ่มโค้ด Center(); (ต่อ).....	139
4.54 เพิ่มโค้ด child: ,.....	140
4.55 เพิ่มโค้ด child: , (ต่อ).....	140
4.56 เพิ่มโค้ด Column(...)......	141
4.57 เพิ่มโค้ด children: [].....	141

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
4.58 เพิ่มข้อความใน children: [].....	142
4.59 เพิ่มข้อความใน children: [] (ต่อ).....	142
4.60 เพิ่มข้อความใน children: [] (ต่อ).....	143
4.61 โครงสร้างของแอปพลิเคชัน.....	143
4.62 โครงสร้าง Column ของแอปพลิเคชัน (ต่อ).....	144
4.63 เพิ่ม TextField(...).....	145
4.64 เพิ่ม decoration:.....	145
4.65 เพิ่ม InputDecoration.....	146
4.66 เพิ่ม labelText:.....	146
4.67 เพิ่มข้อความใน labelText:.....	147
4.68 เพิ่มขอบให้กับ labText:.....	147
4.69 เพิ่มขอบให้กับ labText: (ต่อ).....	148
4.70 เพิ่ม TextField(...) (ต่อ).....	148
4.71 จัดเรียงโค้ดด้วย Prettier Extension.....	149
4.72 นำรูปภาพมาใช้งานบนหน้าเว็บ.....	149
4.73 สร้าง Folder assets ผ่าน VSCode.....	150
4.74 สร้าง Folder assets ผ่าน Desktop.....	150
4.75 เปลี่ยนชื่อไฟล์ภาพ.....	151
4.76 เปลี่ยนชื่อไฟล์ภาพ (ต่อ).....	151
4.77 ย้ายภาพไปยัง Folder assets.....	151
4.78 ย้ายภาพไปยัง Folder assets (ต่อ).....	152
4.79 ย้ายภาพไปยัง Folder assets (ต่อ).....	152
4.80 ย้ายภาพไปยัง Folder assets (ต่อ).....	153
4.81 ตั้งค่า assets ในไฟล์ pubspec.yaml.....	153
4.82 ตั้งค่า assets ในไฟล์ pubspec.yaml (ต่อ).....	154

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
4.83 ปิดคอมเม้นเพื่อให้โค้ดทำงานได้.....	154
4.84 ปิดคอมเม้นเพื่อให้โค้ดทำงานได้ (ต่อ).....	154
4.85 เปลี่ยนชื่อไฟล์ให้ตรงกัน.....	154
4.86 เปลี่ยนชื่อไฟล์ให้ตรงกัน (ต่อ).....	155
4.87 เปลี่ยนชื่อไฟล์ให้ตรงกัน (ต่อ).....	155
4.88 แก้ไขโค้ดเพื่อนำรูปภาพไปแสดงที่หน้าเว็บ.....	155
4.89 แก้ไขโค้ดเพื่อนำรูปภาพไปแสดงที่หน้าเว็บ (ต่อ).....	156
4.90 แก้ไขโค้ดเพื่อนำรูปภาพไปแสดงที่หน้าเว็บ (ต่อ).....	156
4.91 แก้ไขโค้ดเพื่อนำรูปภาพไปแสดงที่หน้าเว็บ (ต่อ).....	157
4.92 แก้ไขโค้ดเพื่อนำรูปภาพไปแสดงที่หน้าเว็บ (ต่อ).....	157
4.93 แก้ไขโค้ดเพื่อนำรูปภาพไปแสดงที่หน้าเว็บ (ต่อ).....	158
4.94 แก้ไขโค้ดที่ไฟล์ main.dart.....	158
4.95 เปิด New Window.....	159
4.96 เปิด New Window (ต่อ).....	159
4.97 ตกแต่งคุณลักษณะโค้ดของฟอนต์.....	160
4.98 เพิ่มโค้ดที่เป็นปุ่ม.....	160
4.99 เพิ่มโค้ดที่เป็นปุ่ม (ต่อ).....	161
4.100 เพิ่มโค้ดที่เป็นปุ่ม (ต่อ).....	161
4.101 เพิ่มโค้ดที่เป็นปุ่ม (ต่อ).....	161
4.102 สั่งการทำงาน Flutter หลังการแก้ไขไฟล์โค้ด.....	162
4.103 สั่งการทำงาน Flutter หลังการแก้ไขไฟล์โค้ด (ต่อ).....	162
4.104 ผลลัพธ์หลังการแก้ไขไฟล์โค้ด.....	163
4.105 เพิ่มกรอบการแสดงผล.....	164
4.106 เพิ่มกรอบการแสดงผล (ต่อ).....	164
4.107 เพิ่มกรอบการแสดงผล (ต่อ).....	164
4.108 เพิ่มกรอบการแสดงผล (ต่อ).....	165

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
4.109 ผลลัพธ์ของการสร้างแอปพลิเคชัน.....	165
4.110 ดาวน์โหลดฟอนต์ที่ต้องการ.....	166
4.111 ดาวน์โหลดฟอนต์ที่ต้องการ (ต่อ).....	166
4.112 แดกไฟล์ที่ได้ทำการดาวน์โหลดมา.....	167
4.113 เปลี่ยนชื่อไฟล์เพื่อให้ง่ายต่อการใช้งาน.....	167
4.114 สร้าง Folder fonts.....	168
4.115 สร้าง Folder fonts (ต่อ).....	168
4.116 ย้ายไฟล์ฟอนต์ไปยัง Folder fonts.....	169
4.117 ไปที่ไฟล์ pubspec.yaml.....	169
4.118 เปิดใช้งานโค้ด fonts.....	170
4.119 เปลี่ยนชื่อฟอนต์.....	170
4.120 อ้างอิงไฟล์ฟอนต์ให้ตรงกับไฟล์ที่นำมาใช้งาน.....	170
4.121 อ้างอิงไฟล์ฟอนต์ให้ตรงกับไฟล์ที่นำมาใช้งาน (ต่อ).....	171
4.122 กำหนดฟอนต์ให้กับข้อความที่ต้องการเปลี่ยนแปลงฟอนต์.....	171
4.123 กำหนดชื่อให้ตรงกับไฟล์ฟอนต์ที่นำมาใช้งาน.....	172
4.124 ผลลัพธ์การปรับแต่งฟอนต์ของข้อความ.....	172
4.125 Flutter Clean.....	173
4.126 เปิดไฟล์ในโปรแกรม VSCode.....	174
4.127 ใช้คำสั่ง flutter pub get แก้ไข Error.....	174
4.128 เปิดไฟล์งานและสั่งการทำงาน.....	175
4.129 เพิ่มโค้ดคำอธิบาย.....	176
4.130 เพิ่มโค้ดคำอธิบาย (ต่อ).....	176
4.131 เพิ่มโค้ดคำอธิบาย (ต่อ).....	176
4.132 เพิ่มโค้ดคำอธิบาย (ต่อ).....	177
4.133 ตกแต่งหน้าต่างของแอปพลิเคชันด้วย Padding.....	177
4.134 ตกแต่งหน้าต่างของแอปพลิเคชันด้วย Padding (ต่อ).....	178



## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
4.135 ตกแต่งหน้าต่างของแอปพลิเคชันด้วย Padding (ต่อ).....	179
4.136 ตกแต่งหน้าต่างของแอปพลิเคชันด้วย Padding (ต่อ).....	179
4.137 ตกแต่งหน้าต่างของแอปพลิเคชันด้วย Padding (ต่อ).....	180
4.138 ตกแต่งหน้าต่างของแอปพลิเคชันด้วย Padding.....	181
4.139 กำหนด Style ของปุ่ม.....	181
4.140 กำหนด Style ของปุ่ม (ต่อ).....	182
4.141 กำหนด Style ของปุ่ม (ต่อ).....	182
4.142 กำหนด Style ของปุ่ม (ต่อ).....	182
4.143 กำหนด Style ของปุ่ม (ต่อ).....	183
4.144 กำหนด Style ของปุ่ม (ต่อ).....	183
4.145 เลือกใช้โค้ดสี.....	184
4.146 กำหนด Style ของปุ่ม (ต่อ).....	184
4.147 กำหนด Style ของปุ่ม (ต่อ).....	185
4.148 กำหนด Style ของปุ่ม (ต่อ).....	185
4.149 จัดระยะของปุ่มด้วย Padding.....	185
4.150 จัดระยะของปุ่มด้วย Padding (ต่อ).....	186
4.151 จัดระยะของปุ่มด้วย Padding (ต่อ).....	186
4.152 จัดระยะของปุ่มด้วย Padding (ต่อ).....	187
4.153 จัดระยะของปุ่มด้วย Padding (ต่อ).....	187
4.154 จัดระยะของปุ่มด้วย Padding (ต่อ).....	187
4.155 จัดระยะของปุ่มด้วย Padding (ต่อ).....	188
4.156 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ).....	189
4.157 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ).....	189
4.158 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ).....	190
4.159 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ).....	190
4.160 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ).....	191

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
4.161 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ).....	191
4.162 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ).....	191
4.163 เพิ่มโค้ด textStyle ให้กับคำอธิบาย.....	192
4.164 เพิ่มโค้ด textStyle ให้กับคำอธิบาย (ต่อ).....	192
4.165 เพิ่มโค้ด textStyle ให้กับคำอธิบาย (ต่อ).....	193
4.166 เพิ่มโค้ด textStyle ให้กับคำอธิบาย (ต่อ).....	193
4.167 เพิ่มโค้ด textStyle ให้กับคำอธิบาย (ต่อ).....	193
4.168 เพิ่มโค้ด textStyle ให้กับคำอธิบาย.....	194
4.169 ผลลัพธ์การแก้ไขโค้ดแล้วข้อผิดพลาด.....	194
4.170 แก้ไข Scroll โดยการให้ชุดคำสั่งของ ListView.....	195
4.171 แก้ไข Scroll โดยการให้ชุดคำสั่งของ ListView (ต่อ).....	196
4.172 แก้ไข Scroll โดยการให้ชุดคำสั่งของ ListView (ต่อ).....	196
4.173 เปลี่ยนจาก Column เป็น ListView.....	197
4.174 เปลี่ยนจาก Column เป็น ListView (ต่อ).....	197
4.175 ผลลัพธ์ของแอปพลิเคชัน.....	198
4.176 สร้างโค้ดเพื่อใช้สำหรับคำนวณค่าที่ได้รับเข้ามา.....	199
4.177 สร้างโค้ดเพื่อใช้สำหรับคำนวณค่าที่ได้รับเข้ามา (ต่อ).....	199
4.178 สร้างโค้ดเพื่อใช้สำหรับคำนวณค่าที่ได้รับเข้ามา (ต่อ).....	199
4.179 กำหนดตัวแปรประเภท double.....	200
4.180 กำหนดตัวแปรประเภท double (ต่อ).....	200
4.181 สร้างโค้ดเพื่อใช้สำหรับคำนวณค่าที่ได้รับเข้ามา (ต่อ).....	200
4.182 เพิ่มโค้ดเพื่อรับค่าและส่งคำนวณ.....	201
4.183 เพิ่มโค้ดเพื่อรับค่าและส่งคำนวณ (ต่อ).....	201
4.184 เพิ่มโค้ดเพื่อรับค่าและส่งคำนวณ (ต่อ).....	201
4.185 เพิ่มโค้ด print(...).....	202
4.186 เพิ่มโค้ด print(...) (ต่อ).....	202

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
4.187 เพิ่มโค้ด print(...) (ต่อ).....	203
4.188 เพิ่มโค้ด print(...) (ต่อ).....	203
4.189 เพิ่มโค้ด print(...) (ต่อ).....	203
4.190 เพิ่มโค้ด print(...) (ต่อ).....	204
4.191 ทดสอบรับค่าและตรวจสอบค่าที่ได้รับมา.....	204
4.192 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ.....	205
4.193 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ).....	205
4.194 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ).....	206
4.195 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ).....	206
4.196 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ).....	207
4.197 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ).....	207
4.198 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ).....	207
4.199 เพิ่มโค้ดสำหรับการคำนวณและตัวแปรสำหรับรับค่าการคำนวณ.....	208
4.200 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ).....	208
4.201 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){}.....	209
4.202 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){} (ต่อ).....	209
4.203 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){} (ต่อ).....	209
4.204 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){} (ต่อ).....	210
4.205 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){} (ต่อ).....	210
4.206 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){} (ต่อ).....	211
4.207 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){} (ต่อ).....	211
4.208 กำหนดตัวแปรเริ่มต้นและสร้างฟังก์ชัน initState(){} (ต่อ).....	212
4.209 กำหนดตัวแปรเริ่มต้นและสร้างฟังก์ชัน initState(){} (ต่อ).....	212
4.210 เปลี่ยนข้อความของ Text(...) ให้เป็นตัวแปรข้อความ.....	213
4.211 เปลี่ยนข้อความของ Text(...) ให้เป็นตัวแปรข้อความ (ต่อ).....	213
4.212 เปลี่ยนข้อความของ Text(...) ให้เป็นตัวแปรข้อความ (ต่อ).....	213

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
4.213 เพิ่ม class setState(){...}	214
4.214 เพิ่ม class setState(){...} (ต่อ)	214
4.215 เพิ่ม class setState(){...} (ต่อ)	215
4.216 เปลี่ยนการแสดงผลการคำนวณ	215
4.217 เปลี่ยนการแสดงผลการคำนวณ (ต่อ)	215
4.218 เปลี่ยนการแสดงผลการคำนวณ (ต่อ)	215
4.219 เปลี่ยนการแสดงผลการคำนวณ (ต่อ)	216
4.220 สร้างฟังก์ชัน setState(){}	216
4.221 ส่งการทำงานเพื่อทดสอบแอปพลิเคชัน	217
4.222 เพิ่มโค้ดในส่วนของ price ที่สามารถแก้ไขค่าได้	217
4.223 เพิ่มโค้ดในส่วนของ price ที่สามารถแก้ไขค่าได้ (ต่อ)	218
4.224 เพิ่มโค้ดในส่วนของ price ที่สามารถแก้ไขค่าได้ (ต่อ)	218
4.225 เพิ่มโค้ดในส่วนของ price ที่สามารถแก้ไขค่าได้ (ต่อ)	219
4.226 แก้ไขโค้ดของผลการคำนวณ	219
4.227 แก้ไขโค้ดของผลการคำนวณ (ต่อ)	219
4.228 แก้ไขโค้ดของผลการคำนวณ (ต่อ)	219
4.229 แก้ไขโค้ดของผลการคำนวณ (ต่อ)	220
4.230 แก้ไขโค้ดของผลการคำนวณ (ต่อ)	220
4.231 เพิ่มโค้ด textField() สำหรับรับค่า price	221
4.232 เพิ่มโค้ด textField() สำหรับรับค่า price	221
4.233 เพิ่มโค้ด textField() สำหรับรับค่า price (ต่อ)	222
4.234 เพิ่มโค้ด textField() สำหรับรับค่า price (ต่อ)	222
4.235 ผลลัพธ์ของแอปพลิเคชันสำหรับคำนวณ	223

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
5.1 สร้าง Folder pages.....	229
5.2 สร้าง Folder pages (ต่อ).....	229
5.3 สร้าง Folder pages (ต่อ).....	230
5.4 สร้าง Folder pages (ต่อ).....	230
5.5 สร้างไฟล์งานใน Folder pages.....	231
5.6 สร้างไฟล์งานใน Folder pages (ต่อ).....	231
5.7 สร้างไฟล์งานใน Folder pages (ต่อ).....	232
5.8 สร้างไฟล์งานใน Folder pages (ต่อ).....	232
5.9 สร้าง class StatefulWidget.....	233
5.10 สร้าง class StatefulWidget (ต่อ).....	233
5.11 สร้าง class StatefulWidget (ต่อ).....	234
5.12 สร้าง class StatefulWidget (ต่อ).....	234
5.13 สร้าง class StatefulWidget (ต่อ).....	235
5.14 แก้ไขโค้ดที่ Error โดยการ import package.....	235
5.15 แก้ไขโค้ดที่ Error โดยการ import package (ต่อ).....	236
5.16 Comment ส่วนที่ไม่ต้องการใช้งาน.....	236
5.17 Comment ส่วนที่ไม่ต้องการใช้งาน (ต่อ).....	237
5.18 ตัด (Cut) โค้ดส่วนของ ListView();.....	237
5.19 ตัด (Cut) โค้ดส่วนของ ListView(); (ต่อ).....	238
5.20 ตัด (Cut) โค้ดส่วนของ ListView(); (ต่อ).....	238
5.21 ตัด (Cut) โค้ดส่วนของ ListView(); (ต่อ).....	239
5.22 นำโค้ดที่ตัดมาใส่แทนโค้ดส่วน Container();.....	239
5.23 นำโค้ดที่ตัดมาใส่แทนโค้ดส่วน Container(); (ต่อ).....	240
5.24 นำโค้ดที่ตัดมาใส่แทนโค้ดส่วน Container(); (ต่อ).....	240
5.25 ตัดโค้ดส่วนที่เหลือจากไฟล์ main.dart.....	241

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
5.26 ตัดโค้ดส่วนที่เหลือนอกจากไฟล์ main.dart (ต่อ).....	241
5.27 ตัดโค้ดส่วนที่เหลือนอกจากไฟล์ main.dart (ต่อ).....	242
5.28 นำโค้ดส่วนที่เหลือนำมาใส่ class State<CalculatePage>{}.....	242
5.29 นำโค้ดส่วนที่เหลือนำมาใส่ class State<CalculatePage>{} (ต่อ).....	243
5.30 นำโค้ดส่วนที่เหลือนำมาใส่ class State<CalculatePage>{} (ต่อ).....	243
5.31 ลบโค้ดส่วน class Home.....	244
5.32 ลบโค้ดส่วน class Home (ต่อ).....	244
5.33 ลบโค้ดส่วน class Home (ต่อ).....	245
5.34 ลบโค้ดส่วน Home(),.....	245
5.35 ลบโค้ดส่วน Home(), (ต่อ).....	245
5.36 ลบโค้ดส่วน Home(), (ต่อ).....	246
5.37 สร้าง class HomePage.....	246
5.38 สร้าง class HomePage (ต่อ).....	247
5.39 สร้าง class HomePage (ต่อ).....	247
5.40 แก้ไขส่วน return Container().....	248
5.41 แก้ไขส่วน return Container() (ต่อ).....	248
5.42 แก้ไขส่วน return Container() (ต่อ).....	249
5.43 แก้ไขส่วน return Container() (ต่อ).....	249
5.44 แก้ไขส่วน return Container() (ต่อ).....	250
5.45 แก้ไขส่วน return Container() (ต่อ).....	250
5.46 แก้ไขส่วน return Container() (ต่อ).....	251
5.47 แก้ไขส่วน return Container() (ต่อ).....	251
5.48 สร้าง class ContactPage.....	252
5.49 สร้าง class ContactPage (ต่อ).....	252
5.50 สร้าง class ContactPage (ต่อ).....	253
5.51 สร้าง class ContactPage (ต่อ).....	253

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
5.52 สร้าง class ContactPage (ต่อ).....	253
5.53 แก้ไขส่วน return Container().....	254
5.54 แก้ไขส่วน return Container() (ต่อ).....	255
5.55 แก้ไขส่วน return Container() (ต่อ).....	255
5.56 แก้ไขส่วน return Container() (ต่อ).....	256
5.57 แก้ไขส่วน return Container() (ต่อ).....	256
5.58 แก้ไขส่วน return Container() (ต่อ).....	257
5.59 แก้ไขส่วน return Container() (ต่อ).....	257
5.60 แก้ไขส่วน return Container() (ต่อ).....	258
5.61 แก้ไขส่วน return Container() (ต่อ).....	258
5.62 แก้ไขส่วน return Container() (ต่อ).....	259
5.63 แก้ไขส่วน return Container() (ต่อ).....	259
5.64 สร้าง class MainPage.....	260
5.65 สร้าง class MainPage (ต่อ).....	261
5.66 สร้าง class MainPage (ต่อ).....	261
5.67 สร้าง class MainPage (ต่อ).....	262
5.68 สร้าง class MainPage (ต่อ).....	262
5.69 ตัดส่วน Scaffold().....	263
5.70 ตัดส่วน Scaffold() (ต่อ).....	263
5.71 ตัดส่วน Scaffold() (ต่อ).....	264
5.72 นำส่วน Scaffold() มาใส่แทน Container().....	264
5.73 นำส่วน Scaffold() มาใส่แทน Container() (ต่อ).....	265
5.74 นำส่วน Scaffold() มาใส่แทน Container() (ต่อ).....	265
5.75 ใส่ MainPage() ในส่วน tag home:.....	266
5.76 ใส่ MainPage() ในส่วน tag home: (ต่อ).....	266

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
5.77 ใส่ MainPage() ในส่วน tag home: (ต่อ).....	266
5.77 ใส่ MainPage() ในส่วน tag home: (ต่อ).....	266
5.78 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage.....	267
5.79 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ).....	267
5.80 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ).....	268
5.81 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ).....	268
5.82 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ).....	269
5.83 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ).....	269
5.84 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ).....	269
5.85 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ).....	270
5.86 เพิ่มโค้ดในส่วนของ tag body:.....	270
5.87 เพิ่มโค้ดในส่วนของ tag body: (ต่อ).....	271
5.88 เพิ่มโค้ดในส่วนของ tag body: (ต่อ).....	271
5.89 เพิ่มโค้ดในส่วนของ tag body: (ต่อ).....	272
5.90 เพิ่มโค้ดในส่วนของ tag body: (ต่อ).....	272
5.91 เพิ่มโค้ดในส่วนของ tag body: (ต่อ).....	273
5.92 เพิ่มโค้ดในส่วนของ tag body: (ต่อ).....	273
5.93 เพิ่มโค้ดในส่วนของ tag body: (ต่อ).....	274
5.94 เพิ่มโค้ดในส่วนของ tag body: (ต่อ).....	274
5.95 เพิ่มโค้ดในส่วนของ tag body: (ต่อ).....	275
5.96 เพิ่มโค้ดในส่วนของ tag body: (ต่อ).....	275
5.97 เพิ่มโค้ดในส่วนของ tag body: (ต่อ).....	276
5.98 เพิ่มโค้ดในส่วนของ tag body: (ต่อ).....	276
5.99 เพิ่มโค้ดในส่วนของ tag body: (ต่อ).....	277
5.100 เพิ่มโค้ดในส่วนของ tag body: (ต่อ).....	277



## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
5.101 เพิ่มโค้ด onTab: (...){...}	278
5.102 เพิ่มโค้ด onTab: (...){...} (ต่อ)	278
5.103 เพิ่มโค้ด onTab: (...){...} (ต่อ)	279
5.104 เพิ่มโค้ด onTab: (...){...} (ต่อ)	279
5.105 เพิ่มโค้ด onTab: (...){...} (ต่อ)	280
5.106 เพิ่มโค้ด onTab: (...){...} (ต่อ)	280
5.107 เพิ่มโค้ด onTab: (...){...} (ต่อ)	281
5.108 ตัวอย่างไอคอนที่ติดมากับการสร้างไฟล์งาน	282
5.109 ตัวอย่างไอคอนที่ติดมากับการสร้างไฟล์งาน (ต่อ)	282
5.110 สร้าง Folder icon	283
5.111 สร้าง Folder icon (ต่อ)	283
5.112 สร้าง Folder icon (ต่อ)	284
5.113 ค้นหาไอคอนสำหรับนำมาใช้งาน	284
5.114 ค้นหาไอคอนสำหรับนำมาใช้งาน (ต่อ)	285
5.115 ค้นหาไอคอนสำหรับนำมาใช้งาน (ต่อ)	285
5.116 ค้นหาไอคอนสำหรับนำมาใช้งาน (ต่อ)	286
5.117 ค้นหาไอคอนสำหรับนำมาใช้งาน (ต่อ)	286
5.118 ค้นหาไอคอนสำหรับนำมาใช้งาน	287
5.119 ดาวน์โหลดไอคอนที่ต้องการแล้วนำไปไว้ใน Folder icon	288
5.120 ดาวน์โหลดไอคอนที่ต้องการแล้วนำไปไว้ใน Folder icon (ต่อ)	288
5.121 ค้นหา flutter launcher icon	289
5.122 ค้นหา flutter launcher icon (ต่อ)	289
5.123 คัดลอกโค้ดใน dependencies:	290
5.124 คัดลอกโค้ดใน dependencies: (ต่อ)	290
5.125 นำโค้ดที่คัดลอกมา มาไว้ในไฟล์ pubspec.yaml	291

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
5.126 นำโค้ดที่คัดลอกมา มาไว้ในไฟล์ pubspec.yaml (ต่อ).....	291
5.127 นำโค้ดที่คัดลอกมา มาไว้ในไฟล์ pubspec.yaml (ต่อ).....	292
5.128 นำโค้ดที่คัดลอกมา มาไว้ในไฟล์ pubspec.yaml (ต่อ).....	292
5.129 คัดลอกทั้งหมดของส่วน flutter_icons:.....	293
5.130 คัดลอกทั้งหมดของส่วน flutter_icons: (ต่อ).....	293
5.131 คัดลอกทั้งหมดของส่วน flutter_icons: (ต่อ).....	294
5.132 คัดลอกทั้งหมดของส่วน flutter_icons: (ต่อ).....	294
5.133 นำโค้ดที่คัดลอกมาจาก flutter_icons: มาวาง.....	295
5.134 นำโค้ดที่คัดลอกมาจาก flutter_icons: มาวาง (ต่อ).....	295
5.135 นำโค้ดที่คัดลอกมาจาก flutter_icons: มาวาง (ต่อ).....	296
5.136 แก้ไขโค้ดในส่วน flutter_icons:.....	296
5.137 แก้ไขโค้ดในส่วน flutter_icons: (ต่อ).....	297
5.138 แก้ไขโค้ดในส่วน flutter_icons: (ต่อ).....	297
5.139 แก้ไขโค้ดในส่วน flutter_icons: (ต่อ).....	298
5.140 แก้ไขโค้ดในส่วน flutter_icons: (ต่อ).....	298
5.141 คัดลอกโค้ด flutter pub run.....	299
5.142 คัดลอกโค้ด flutter pub run (ต่อ).....	299
5.143 วางโค้ด flutter pub run ในหน้าต่าง cmd.....	300
5.144 วางโค้ด flutter pub run ในหน้าต่าง cmd (ต่อ).....	300
5.145 ไอคอนได้ทำการเปลี่ยนแปลงเรียบร้อยแล้ว.....	301
5.146 เปลี่ยนชื่อไอคอนใน tag android:label.....	301
5.147 เปลี่ยนชื่อไอคอนใน tag android:label (ต่อ).....	302
5.148 เปลี่ยนชื่อไอคอนใน tag android:label (ต่อ).....	302
5.149 ใช้คำสั่ง flutter build apk.....	303
5.150 ใช้คำสั่ง flutter build apk (ต่อ).....	304

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
5.151 คัดลอก Path ที่ตัวหนังสือสีเขียว.....	304
5.152 นำ Path มาวางต่อ Path Project เพื่อค้นหาไฟล์ที่ได้ทำการสร้างมา.....	304
5.153 นำไฟล์ที่สร้างเสร็จแล้วไปติดตั้งที่มือถือ.....	305
5.154 เข้าทดสอบแอปพลิเคชัน.....	305
5.155 ผลลัพธ์การพัฒนาแอปพลิเคชัน.....	306
6.1 django.....	311
6.2 เปิด Python Shell.....	314
6.3 เปิด Python Shell.....	314
6.4 แก้ไขเมื่อไม่สามารถ Run ได้.....	315
6.5 ติดตั้ง virtualenv.....	315
6.6 สร้าง virtualenv.....	316
6.7 ได้โฟลเดอร์ venv.....	316
6.8 พิมพ์คำสั่งเพื่อเข้าสู่โหมด venv.....	317
6.9 ติดตั้ง django.....	317
6.10 ตรวจสอบการติดตั้ง.....	318
6.11 สร้าง Project ใน django.....	318
6.12 โฟลเดอร์ flutterapi.....	319
6.13 Run Server.....	319
6.14 Web API django.....	320
6.15 สั่งหยุดการทำงานของ Web Server.....	321
6.16 สร้างฐานข้อมูล.....	322
6.17 สร้างหน้าของผู้ดูแลระบบ.....	322
6.18 สร้างโฟลเดอร์แอปพลิเคชันย่อย.....	323
6.19 ตรวจสอบโฟลเดอร์.....	323

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
6.20 เปิด VSCode และทำการติดตั้งส่วนเสริม.....	325
6.21 ติดตั้งส่วนเสริม django.....	325
6.22 คัดลอกโค้ดที่ต้องใช้งาน.....	326
6.23 สร้างไฟล์ใหม่.....	326
6.24 วางโค้ดที่คัดลอกมาและเพิ่มโค้ด.....	327
6.25 ใส่ '*' ใน [ ] ของ tag ALLOWED_HOSTS.....	327
6.26 ใส่ 'myapp' ใน [ ] ของ tag INSTALLED_APPS.....	327
6.27 ใส่ include ใน tag django.urls ต่อจาก path.....	328
6.28 ให้เพิ่มโค้ดใน [ ] ของ tag urlpatterns ตามภาพ.....	328
6.29 เพิ่มโค้ดในไฟล์ views.py.....	329
6.30 ได้ผลลัพธ์ดังนี้.....	330
6.31 ค้นหาวิธีแก้ไข utf8 และคัดลอก return JsonResponse().....	331
6.32 วางโค้ดในส่วน return JsonResponse().....	332
6.33 ผลการวางโค้ดในส่วน return JsonResponse().....	332
6.34 ดาวน์โหลดและติดตั้ง ngrok.....	333
6.35 พิมพ์คำสั่ง ngrok http 8000.....	333
6.36 นำไปทดลองใช้งาน.....	334
6.37 ผลลัพธ์การพัฒนา ngrok.....	334
6.38 cd เข้าไปยังไฟล์งาน flutter Project.....	335
6.39 พิมพ์คำสั่งเปิดใช้งานโหมด venv.....	335
6.40 cd เข้าไปยังไฟล์ Project ที่ได้ทำการสร้างไว้.....	335
6.41 cd เข้าไปยังไฟล์ Project ที่ได้ทำการสร้างไว้.....	336
6.42 เข้าเว็บเซิร์ฟเวอร์โดยการพิมพ์ localhost:8000.....	332
6.42 เข้าเว็บเซิร์ฟเวอร์โดยการพิมพ์ localhost:8000.....	332
6.42 เข้าเว็บเซิร์ฟเวอร์โดยการพิมพ์ localhost:8000.....	332

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
7.1 เปิดไฟล์งาน Project ใน VSCode และเพิ่มโค้ด.....	342
7.2 เปิดไฟล์ admin.py และเพิ่มโค้ด.....	342
7.3 หยุด Run Server.....	343
7.4 ตรวจสอบอัปเดตการเปลี่ยนแปลง.....	343
7.5 อัปเดตการเปลี่ยนแปลง.....	343
7.6 สั่งการทำงาน Server.....	344
7.7 เข้าสู่หน้า Admin.....	344
7.8 หน้าการจัดการฐานข้อมูล.....	345
7.9 คลิก Add ที่ Todolist.....	345
7.10 กรอกข้อมูลที่ต้องการเพิ่มจากนั้นทำการกดบันทึก (Save).....	346
7.11 ได้ผลลัพธ์.....	346
7.12 เพิ่มข้อมูลให้ได้ 2 รายการ.....	347
7.13 เพิ่มโค้ดเพื่อแสดง Title ที่ต้องการ.....	347
7.14 ผลลัพธ์การแสดง Title.....	348
7.15 ดาวน์โหลดโปรแกรม DB Browser for SQLite.....	349
7.16 เปิดโครงสร้าง Database ในโปรแกรม.....	349
7.17 ดูข้อมูลหรือแสดงข้อมูลของตาราง.....	350
7.18 ค้นหา Django REST Framework.....	351
7.19 คัดลอกโค้ดสำหรับนำไปติดตั้ง.....	351
7.20 วางคำสั่ง php install djangrestframework.....	352
7.21 วางคำสั่ง rest_framework.....	352
7.22 สร้างไฟล์ serializers.py และทำการเพิ่มโค้ด.....	353
7.23 ไปที่ไฟล์ views.py และทำการเพิ่มโค้ด.....	353
7.24 ทำการเพิ่มโค้ดอีกส่วน.....	353
7.25 เพิ่มโค้ดที่ไฟล์ urls.py.....	354

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
7.26 สั่งการทำงาน Server.....	354
7.27 ผลลัพธ์การสร้างไฟล์ serializers.....	355
7.28 ค้นหา Postman.....	357
7.29 ดาวน์โหลดโปรแกรมสำหรับระบบปฏิบัติการที่ต้องการ.....	357
7.30 เปิดโปรแกรม Postman.....	358
7.31 ไปที่ Start with something new > Create New.....	358
7.32 เลือกประเภทงาน.....	359
7.33 สั่งการทำงานServer หากไม่ได้ Run Server ไว้.....	359
7.34 คัดลอก url จากหน้า Django REST Framework มาใส่ไว้ที่ช่อง GET.....	360
7.35 ผลลัพธ์การดึงข้อมูล.....	360
7.36 เปิดไฟล์ views.py และแก้ไขโค้ด.....	361
7.37 เปิดไฟล์ urls.py และแก้ไขโค้ด.....	361
7.38 แก้ไข Postman.....	362
7.39 มีข้อมูลที่เพิ่มเข้ามาใหม่.....	362
8.1 การเริ่มต้นใช้งาน Firebase.....	370
8.2 การเริ่มสร้างโปรเจกต์ Firebase.....	371
8.3 การสร้างโปรเจกต์ Firebase สำเร็จ.....	371
8.4 การสร้างโปรเจกต์ Firebase.....	372
8.5 Android->app->src->main->AndroidManifest.xml.....	372
8.6 การหา package name ในโปรเจกต์ Flutter.....	373
8.7 การเพิ่มแอปพลิเคชันฝั่ง Android.....	373
8.8 การดาวน์โหลดไฟล์ google-services.json.....	374
8.9 การวางไฟล์ google-services.json ใน Android/app.....	374
8.10 การวาง classpath 'com.google.gms:google-services: 4.3.15'.....	375
8.11 การ apply plugin.....	375

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
8.12 การวาง implementation platform.....	376
8.13 การเพิ่ม Firebase ในแอป Android เมื่อเพิ่มข้อมูลครบถ้วน.....	376
8.14 เว็บไซต์ pub.dev.....	377
8.15 ผลการค้นหา Library.....	378
8.16 Library ที่ต้องการใช้งาน.....	378
8.17 การติดตั้ง Library.....	379
8.18 การ run หลังการติดตั้ง Library สำเร็จ.....	380
8.19 การเพิ่ม multiDexEnabled true.....	381
8.20 Error:uses-sdk:minSdkVersion.....	382
8.21 การเพิ่ม minSdkVersion.....	383
8.22 การเริ่มต้นสร้าง Firestore Database.....	384
8.23 การตั้งค่าเริ่มต้นให้กับ Firestore Database.....	385
8.24 การเลือกที่ตั้งในการเก็บข้อมูล Firestore Database.....	386
8.25 การเพิ่ม collection.....	386
8.26 การเพิ่มข้อมูลลงไป collection.....	387
8.27 ผลที่เพิ่มลงไป collection.....	387
8.28 Extension Dart Data Class Generator.....	388
8.29 การสร้าง Model.....	388
8.30 Model ที่ Generate แล้ว.....	389
8.31 การอ่านค่าข้อมูลจาก Firestore Database มาเก็บไว้ใน Model.....	390
8.32 การข้อมูลของ userModel ที่ได้บน Terminal ใน VS Code.....	391
8.33 การเขียน Code ในการแสดงผลข้อมูลบน Imurater.....	391
8.34 การแสดงผลข้อมูลบน Imurater.....	392
และนำข้อมูลไปเปรียบเทียบกับ Firestore Database	
8.35 การเริ่มใช้งาน Firebase Authentication.....	393

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
8.36 การเปิดการใช้งาน Firebase Authentication.....	393
8.37 แพลตฟอร์มที่สามารถล็อกอินผ่าน Firebase Authentication.....	394
8.38 เปิดการใช้งาน Email/Password.....	394
8.39 User ที่ทำการเพิ่มข้อมูลเข้ามา.....	395
8.40 ตัวอย่างการออกแบบหน้าล็อกอิน.....	395
8.41 ตัวอย่างการใช้งาน TextField.....	396
8.42 ตัวอย่างการใช้งาน ElevatedButton.....	396
8.43 ตัวอย่างประกาศตัวแปร.....	397
8.44 ตัวอย่างการเพิ่มตัวแปรเพื่อรับค่าข้อมูลจากการกรอกบน TextField.....	397
8.45 การเช็คการล็อกอิน Firebase.....	398
8.46 การเพิ่ม Future Function checkAuthen ในปุ่ม Login.....	399
8.47 การ Copy UID User.....	399
8.48 ข้อมูลหลังการเพิ่ม Document.....	400
8.49 ข้อมูลหลังการลบ Document.....	400
8.50 หน้า Home Page.....	400
8.51 ปุ่มสำหรับการล็อกเอาท์.....	401
8.52 การเขียน Code เพิ่ม IconButton ไว้บน AppBar.....	401
8.53 การเขียน Code ล็อกเอาท์ หรือลงชื่อออกจากระบบ.....	401
8.54 เริ่มต้นใช้งาน Storage.....	402
8.55 การเปิดการใช้งาน Storage.....	403
8.56 การตั้งค่าเริ่มต้นให้กับ Storage.....	403
8.57 การกำหนดตำแหน่งในการเก็บข้อมูลให้กับ Cloud Storage.....	404
8.58 Cloud Storage สำหรับการใช้เก็บข้อมูลรูปภาพ Firebase.....	404
8.59 การดาวน์โหลดรูปภาพผ่านเว็บไซต์ Iconfinder.....	405
8.60 การสร้างโฟลเดอร์บน Cloud Storage.....	405



## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
8.61 ไฟล์รูปภาพที่ทำการอัปโหลดไปยัง Cloud Storage.....	406
8.62 การ Copy Access token.....	407
8.63 การเพิ่ม Field imageProfile.....	407
8.64 การเพิ่ม Icon Button.....	408
8.65 การเพิ่มไฟล์ add_picture.dart.....	408
8.66 การ Generate data Class ใหม่.....	409
8.67 ตัวอย่าง Code หาข้อมูลผู้ใช้งานมาเก็บไว้ใน userModel.....	410
8.68 ตัวอย่าง Code Widget showImage();.....	411
8.69 ผลการแสดงรูปภาพบน Emulator.....	412
8.70 ตัวอย่าง Code Future ในการเรียกใช้งาน image_picker.....	412
8.71 ตัวอย่าง Code เมื่อทำการกดจะเลือกจาก Camera.....	413
8.72 ตัวอย่าง Code เมื่อทำการกดจะเลือกจาก Gallery.....	414
8.73 ตัวอย่าง Code บันทึกข้อมูลไปยัง Storage และ Cloud Firestore.....	415
8.74 ตัวอย่าง Code ปุ่มในการกดบันทึกข้อมูล.....	416
8.75 ก่อนการเพิ่มรูปภาพ.....	416
8.76 หลังการเพิ่มรูปภาพ.....	417
8.77 รูปภาพที่อัปขึ้นไปใหม่.....	417
8.78 Code แก้ไขการ Route ไปยัง ChatPage.....	418
8.79 การประกาศตัวแปรที่ใช้ในหน้า ChatPage.....	418
8.80 Code การอ่านข้อมูลจาก Cloud Firestore Database.....	419
8.81 Code การออกแบบปุ่มออกจากระบบ.....	419
8.82 Code การออกจากระบบของ Fiebase.....	420
8.83 ตัวอย่าง Code ในส่วนของ Body.....	420
8.84 ตัวอย่าง Code Widget Message.....	421
8.85 ตัวอย่าง Code ในส่วนของการกรอกและส่งข้อความ.....	422

## สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ	หน้า
8.86 ตัวอย่าง Code void function onSendMessage().....	423
8.87 หน้า ChatPage.....	423
8.88 หน้า ChatPage ที่มีข้อความ.....	424
8.89 ฐานข้อมูล Cloud Firestore Database collection chatTable.....	424
8.90 การเพิ่ม Document ข้างใน Collection chatTable.....	425
8.91 หน้าแอปพลิเคชันหลังเพิ่ม Document ข้างใน Collection chatTable.....	425

## สารบัญตาราง

ตาราง	หน้า
3.1 ตารางชนิดตัวแปรในภาษา Dart.....	82
3.2 ตารางชนิดตัวแปรชนิดพิเศษในภาษา Dart.....	83
3.3 ตารางตัวดำเนินการทางตรรกศาสตร์.....	88
7.1 HTTP Methods ที่ควรรู้.....	356
7.2 Status Codes ที่ควรรู้.....	356
8.1 ความแตกต่างระหว่างฐานข้อมูลเชิงสัมพันธ์และฐานข้อมูล NoSQL.....	369

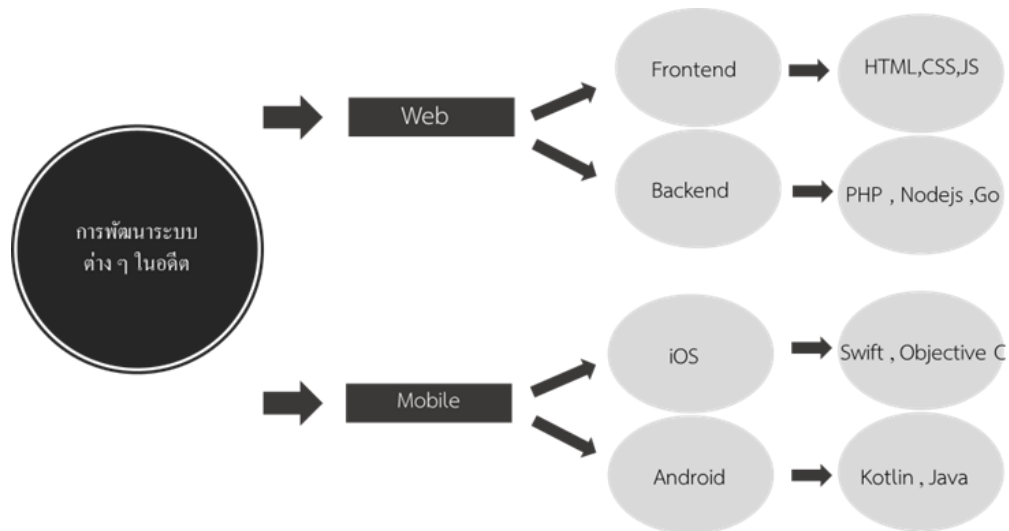
# บทที่ 1

## Introduction to Mobile Application Development

ปัจจุบันแอปพลิเคชันบนอุปกรณ์เคลื่อนที่เป็นที่นิยม ผลสืบเนื่องมาจากการเติบโตของอุปกรณ์เคลื่อนที่ โดยจากการสำรวจของสำนักงานสถิติแห่งชาติ (2564) พบว่าประชาชนใช้อุปกรณ์เคลื่อนที่แบบ สมาร์ทโฟน (Smart Phone) ถึงร้อยละ 96.4 ส่งผลให้แอปพลิเคชันบนอุปกรณ์เคลื่อนที่ที่มีอัตราการเติบโตตามไปด้วย ในการพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่ที่มีเครื่องมือในการพัฒนาหลากหลาย Flutter เป็นเครื่องมือหนึ่งที่เป็นที่นิยมในการพัฒนา Flutter เป็นเฟรมเวิร์คสำหรับการพัฒนาแอปพลิเคชันจากบริษัท กูเกิล (Google) ออกแบบมาเพื่อใช้ในการพัฒนาแอปพลิเคชันแบบข้ามแพลตฟอร์ม (Cross-Platform) ซึ่งก็คือ การที่แอปพลิเคชันสามารถทำงานได้ในหลายแพลตฟอร์ม โดย Flutter รองรับการใช้งานใน ทั้ง iOS และ Android จึงทำให้การพัฒนาแอปพลิเคชันทำได้ง่าย รวดเร็ว และมีประสิทธิภาพ ในบทนี้จะกล่าวถึง การพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่ในปัจจุบัน ความเป็นมาของ Flutter ภาษาที่ใช้ในการพัฒนาแอปพลิเคชันใน Flutter วิธีการดาวน์โหลด การติดตั้งเครื่องมือในการพัฒนา การตั้งค่าการใช้งานเครื่องมือต่าง ๆ การกำหนดค่าในการทำงานของโปรแกรม และการเริ่มต้นสร้างแอปพลิเคชันด้วย Flutter จากเนื้อหาข้างต้นที่กล่าวมาทั้งหมด มีรายละเอียดดังต่อไปนี้

### ทำความรู้จักกับ Flutter

การพัฒนาระบบต่าง ๆ ในอดีตนั้น มีลักษณะดังภาพต่อไปนี้



ภาพประกอบ 1.1 การพัฒนาระบบในอดีต

ที่มา : KongRuksiam. (2563)

ในอดีตจะมีการพัฒนาระบบในสองรูปแบบ ได้แก่ การพัฒนาในรูปแบบเว็บ และการพัฒนาในรูปแบบบนอุปกรณ์เคลื่อนที่ โดยการพัฒนาในรูปแบบเว็บนั้นจะแบ่งเป็น 2 ส่วน ได้แก่ ส่วนหน้าบ้าน (Frontend) ซึ่งเป็นส่วนที่ติดต่อกับผู้ใช้ และส่วนหลังบ้าน (Backend) ซึ่งเป็นส่วนที่ติดต่อกับฐานข้อมูล โปรแกรมภาษาที่ใช้พัฒนาส่วนหน้าบ้าน อาทิ HTML, CSS และ JavaScript (JS) โปรแกรมภาษาที่ใช้พัฒนาส่วนหลังบ้าน อาทิ PHP, NodeJS และ Go การพัฒนาในรูปแบบบนอุปกรณ์เคลื่อนที่แบ่งเป็น 2 ส่วนตามแพลตฟอร์มการใช้งานที่เป็นที่นิยม ได้แก่ iOS ซึ่งเป็นของบริษัทแอปเปิล และ Android ซึ่งเป็นของบริษัทกูเกิล โปรแกรมภาษาที่ใช้พัฒนาบนแพลตฟอร์ม iOS อาทิ Swift และ Object C โปรแกรมภาษาที่ใช้พัฒนาบนแพลตฟอร์ม Android อาทิ Kotlin และ JAVA จะเห็นว่าการพัฒนาโปรแกรมบนแพลตฟอร์มแต่ละชนิดใช้ภาษาที่แตกต่างกัน ซึ่งสร้างความยากลำบากให้กับผู้พัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่ที่ต้องพัฒนาแอปพลิเคชันในสองรูปแบบ จึงได้มีการพัฒนารูปแบบที่ทำให้สามารถพัฒนาแอปพลิเคชันครั้งเดียวแต่รองรับการทำงานทั้งสองแพลตฟอร์มที่เรียกกันว่า “การพัฒนาแอปพลิเคชันแบบ Cross Platform”

### 1. การพัฒนาแอปพลิเคชันแบบ Cross Platform

แพลตฟอร์ม คือ ชุดของส่วนประกอบที่มีความเสถียรซึ่งสนับสนุนความหลากหลายและการพัฒนาในระบบโดยจำกัดการเชื่อมโยงระหว่างส่วนประกอบอื่น ๆ (Baldwin C. Y. & Woodard C. J., 2009).

การพัฒนาแอปพลิเคชันอุปกรณ์เคลื่อนที่แบบข้ามแพลตฟอร์ม (Cross Platform Mobile Application Development) เป็นรูปแบบในการพัฒนาแอปพลิเคชันโดยใช้การเขียนโค้ดเพียงครั้งเดียวแต่สามารถใช้งานได้ในทุกแพลตฟอร์ม จึงสามารถรองรับการใช้งานระบบปฏิบัติการต่าง ๆ

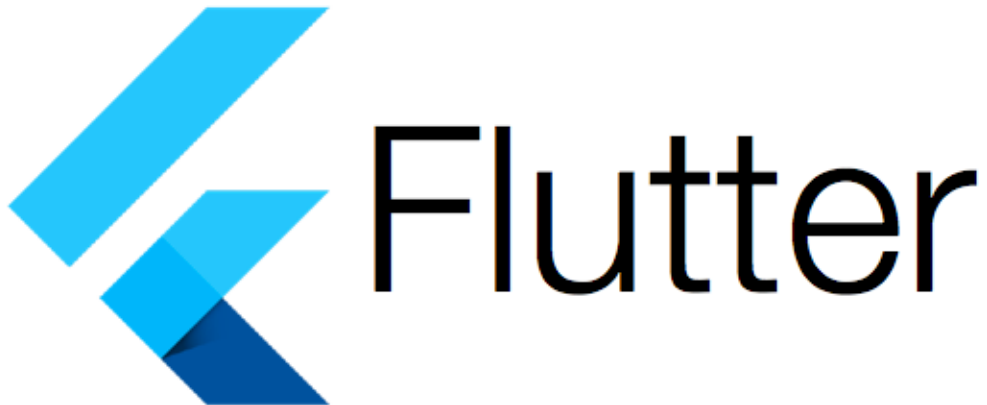
เช่น Android, iOS และ Windows Phone ได้ การพัฒนาแอปพลิเคชันเคลื่อนที่แบบข้ามแพลตฟอร์ม จะช่วยแก้ปัญหาเรื่องระยะเวลาที่ต้องใช้ในการพัฒนาแอปพลิเคชันในรูปแบบเนทีฟ (การเรียกใช้ความสามารถและเข้าถึงฮาร์ดแวร์ของอุปกรณ์ได้อย่างไม่จำกัด สนับสนุนส่วนติดต่อผู้ใช้และการโต้ตอบทั้งหมดที่มีอยู่ในสภาพแวดล้อมการทำงานของอุปกรณ์เคลื่อนที่ได้อย่างครอบคลุม) ซึ่งมีการใช้โปรแกรมภาษาและเครื่องมือที่แตกต่างกัน ดังนั้นระยะเวลาที่ใช้ในการพัฒนาแอปพลิเคชันในแต่ละแพลตฟอร์มจึงแปรผันโดยตรงกับจำนวนของแพลตฟอร์มที่ต้องการพัฒนา (เอกรินทร์ วทัญญูเลิศสกุล, 2564 : 19)

## 2. ทำความรู้จักกับ Flutter

Flutter เปลี่ยนกระบวนการพัฒนาแอปพลิเคชัน สร้าง ทดสอบ และการนำไปติดตั้งใช้งานบนเว็บเดสก์ท็อป และแอปพลิเคชันแบบฝัง (Embedded Apps) จากโค้ดเดียว เป็นเฟรมเวิร์คโอเพนซอร์สโดย กูเกิล (Google) สำหรับการสร้างแอปพลิเคชันหลายแพลตฟอร์ม (Platform) ที่สวยงาม (flutter.dev, 2566 : 1)

Flutter เป็นเฟรมเวิร์คที่พัฒนาโดยบริษัทกูเกิล เพื่อใช้ในการสร้างแอปพลิเคชันแบบ Multi-Platform จากโค้ดพื้นฐานเพียงชุดเดียวที่เขียนด้วยภาษา Dart โดย Flutter จะแปลงโค้ดดังกล่าวไปเป็นแอปพลิเคชันสำหรับใช้งานบนระบบต่าง ๆ ทั้ง Web, iOS และ Android ดังนั้นจึงไม่จำเป็นต้องสร้างแอปพลิเคชันแบบแยกระบบปฏิบัติการให้ยุ่งยากเหมือนที่ผ่านมา สร้างขึ้นโดย Google ซึ่งเป็นผู้พัฒนาระบบ Android ดังนั้นสามารถเชื่อถือได้ว่าแอปพลิเคชันที่ถูกแปลงเป็นแพลตฟอร์มต่าง ๆ จะสามารถทำงานได้อย่างถูกต้องทัดเทียมแอปพลิเคชันที่สร้างด้วยเครื่องมือแบบ Native โดยตรง เช่น Android Studio หรือ XCode เป็นต้น (ปัญญา ปะลีละเตสัง, 2566 : 111)

Flutter ใช้ภาษา Dart ในการเขียนโค้ดซึ่งเป็นภาษาที่เรียนรู้ได้ง่าย ไม่มีหลักเกณฑ์ที่ยุ่งยากซับซ้อน ดังนั้นการพัฒนาแอปพลิเคชันด้วย Flutter จึงทำได้ง่ายเช่นเดียวกัน (ปัญญา ปะลีละเตสัง, 2566: 111)



ภาพประกอบ 1.2 Flutter

ที่มา : Flutter. (2020 : 1)



ภาพประกอบ 1.3 ภาษา Dart

ที่มา : Dart. (2020 : 1)

Note
<p>สามารถนำแอปพลิเคชันที่พัฒนาด้วย Flutter เปลี่ยนเป็นรูปแบบเว็บแอปพลิเคชัน โดยการแปลงภาษา Dart เป็น JavaScript และสั่งการทำงานใน Web Browser สามารถเข้าดูรายละเอียดเพิ่มเติมได้ที่ <a href="https://flutter.dev/docs/get-started/web">https://flutter.dev/docs/get-started/web</a></p>

### 3. จุดเด่นของ Flutter

1. สามารถสร้างแอปพลิเคชันในแพลตฟอร์มต่าง ๆ ด้วยการเขียนโปรแกรมเพียงชุดเดียว (Single Codebase) เช่น เขียนแอปพลิเคชันเพียงชุดเดียว ก็สามารถทำงานได้กับอุปกรณ์ iOS และ อุปกรณ์ Android ได้พร้อม ๆ กัน (จีราวุธ วารินทร์, 2564 : 2)

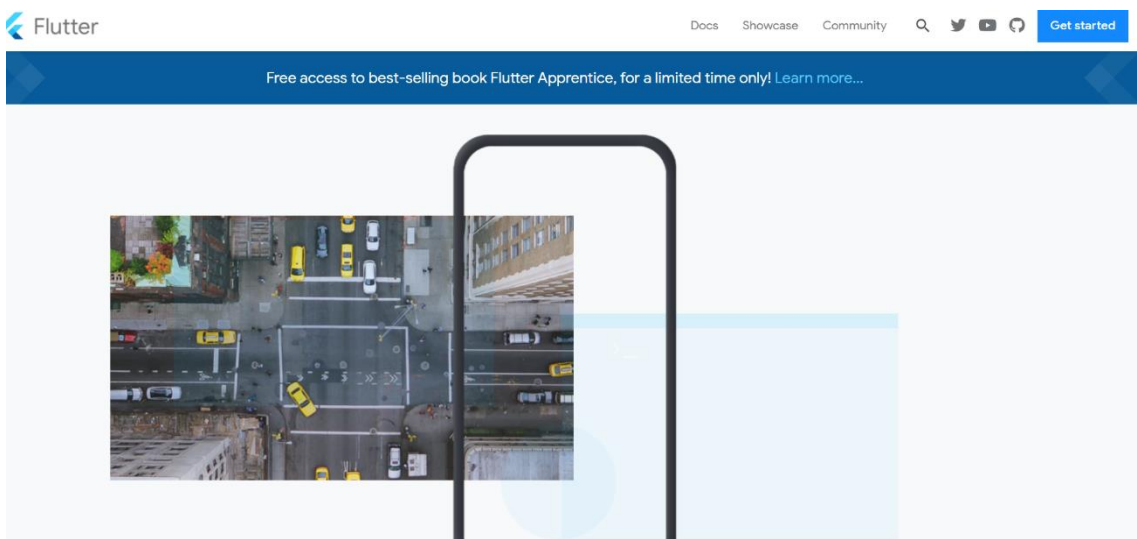
2. การสร้างแอปพลิเคชันใน Flutter จะใช้ภาษาคอมไพเลอร์ที่เรียกว่า Dart โดยผลลัพธ์แอปพลิเคชันที่ได้จาก Flutter นั้นจะเป็นแอปพลิเคชันในแบบ Native ซึ่งสามารถนำไปใช้กับอุปกรณ์ iOS หรือ Android ได้โดยตรง และสามารถอัปโหลดแอปพลิเคชัน ที่สร้างขึ้นไปยัง Apple App Store (สำหรับ iOS) หรืออัปโหลดไปยัง Google Play Store (สำหรับ Android) โดยไม่ต้องเสียเวลาแปลงโค้ด (จิราวุธ วารินทร์, 2564 : 1)

3. Flutter มีจำนวนผู้ใช้งานเพิ่มขึ้นอย่างรวดเร็ว ดังนั้นจึงมีชุมชนขนาดใหญ่ที่จะคอยช่วยเหลือเมื่อเจอปัญหา หรือต้องการคำแนะนำเพิ่มเติม (บัญชา ปะลีละเตสัง, 2566: 112)

4. Flutter มีส่วนเสริมหรือไลบรารีให้เลือกใช้งานค่อนข้างมากและมีจำนวนเพิ่มขึ้นเรื่อย ๆ จึงช่วยให้มีแนวทางการสร้างแอปพลิเคชันที่หลากหลายมากยิ่งขึ้น (บัญชา ปะลีละเตสัง, 2566: 112)

### การทดลองใช้งาน Flutter

1. เข้าไปที่เว็บไซต์ <https://flutter.dev/>

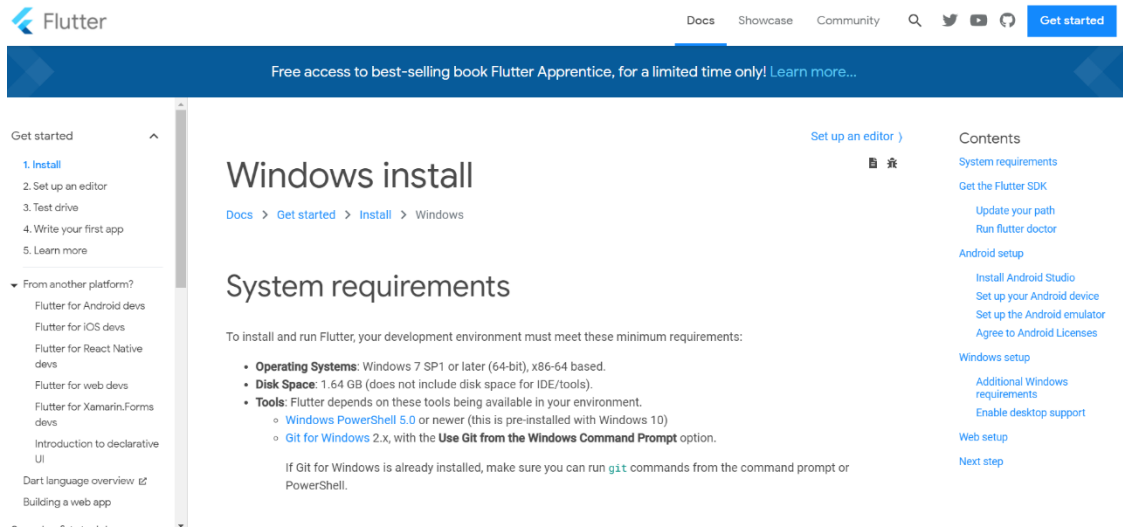


ภาพประกอบ 1.4 หน้าเว็บไซต์ของ Flutter

ที่มา : Flutter. (2020 : 1)



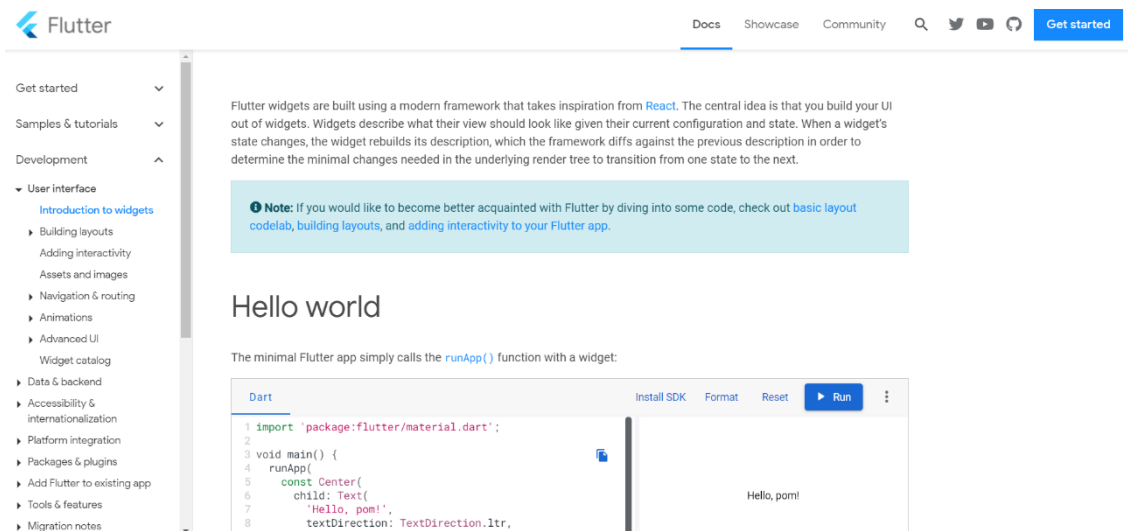
## 2. ความต้องการของระบบ (System Requirement) Flutter



ภาพประกอบ 1.5 ความต้องการของระบบ Flutter

ที่มา : Flutter. (2020 : 1)

## 3. เริ่มต้นทดลองการใช้งาน Flutter ด้วยการไปที่ Development > User Interface > Introduction to widgets



ภาพประกอบ 1.6 เริ่มต้นการใช้งาน Flutter (1)

ที่มา : Flutter. (2020 : 1)

4. ทดลองใช้งาน DartPad โดยไปที่ <https://dartpad.dev/> และทดลองสั่งทำงาน



The screenshot shows the DartPad web interface. The code editor contains the following Dart code:

```

1 void main() {
2   for (int i = 0; i < 4; i++) {
3     print('hello $i');
4   }
5 }
6

```

The console on the right displays the output of the program:

```

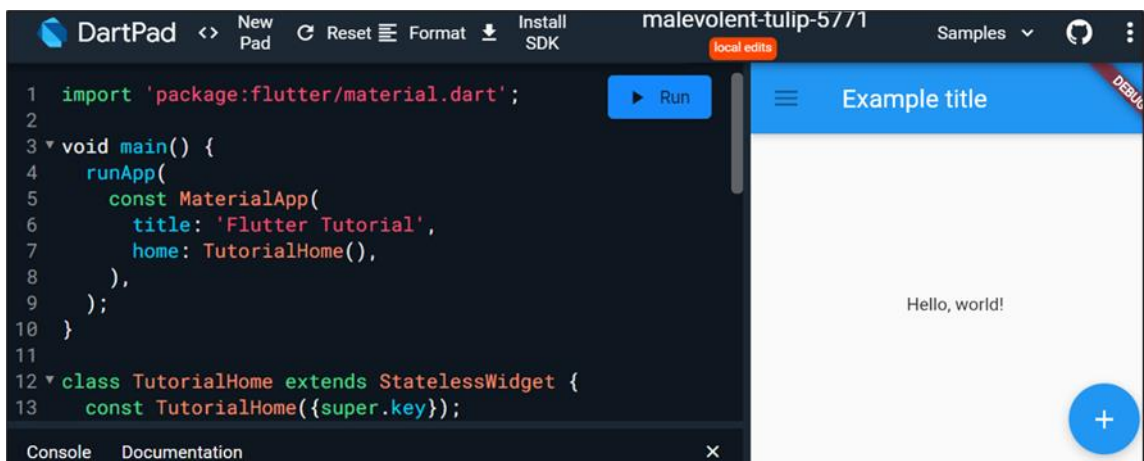
hello 0
hello 1
hello 2
hello 3

```

ภาพประกอบ 1.7 เริ่มต้นการใช้งาน Flutter (2)

ที่มา : DartPad. (2020 : 1)

5. นำโค้ดจาก Development > User Interface > Introduction to widgets ใน flutter.dev ของขั้นตอนที่ 3 มาทดลองสั่งทำงานเพื่อดูผลลัพธ์



The screenshot shows the DartPad web interface with a Flutter application. The code editor contains the following Dart code:

```

1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(
5     const MaterialApp(
6       title: 'Flutter Tutorial',
7       home: TutorialHome(),
8     ),
9   );
10 }
11
12 class TutorialHome extends StatelessWidget {
13   const TutorialHome({super.key});

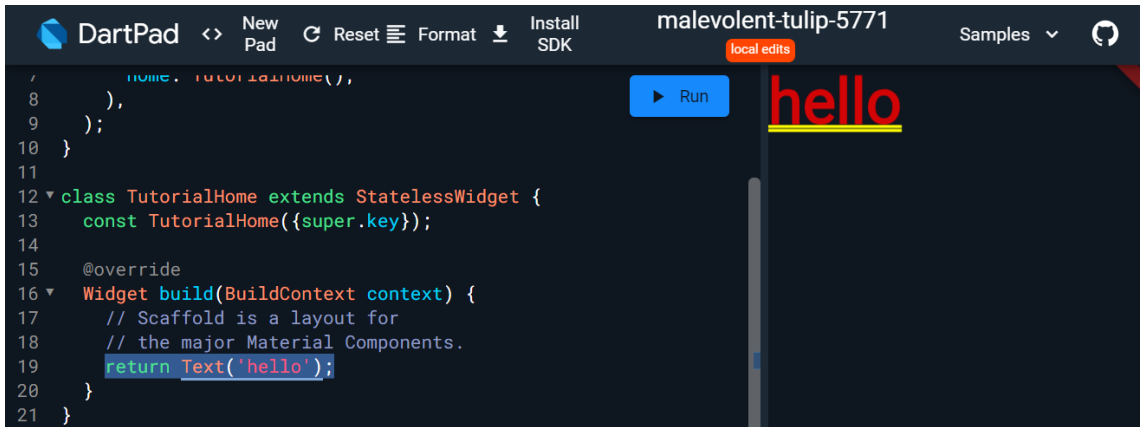
```

The rendered application on the right shows a blue header bar with the title "Example title" and a white body with the text "Hello, world!". A blue circular button with a white plus sign is visible in the bottom right corner.

ภาพประกอบ 1.8 ทดลองใช้งาน Flutter (3)

ที่มา : ฅนปลัซ วรณตรง (2564 : 18)

6. ทดลองการเอาส่วนที่ไม่ใช่ Return ออกและทำการใส่ข้อความอื่น ๆ เข้าไปแทน



```

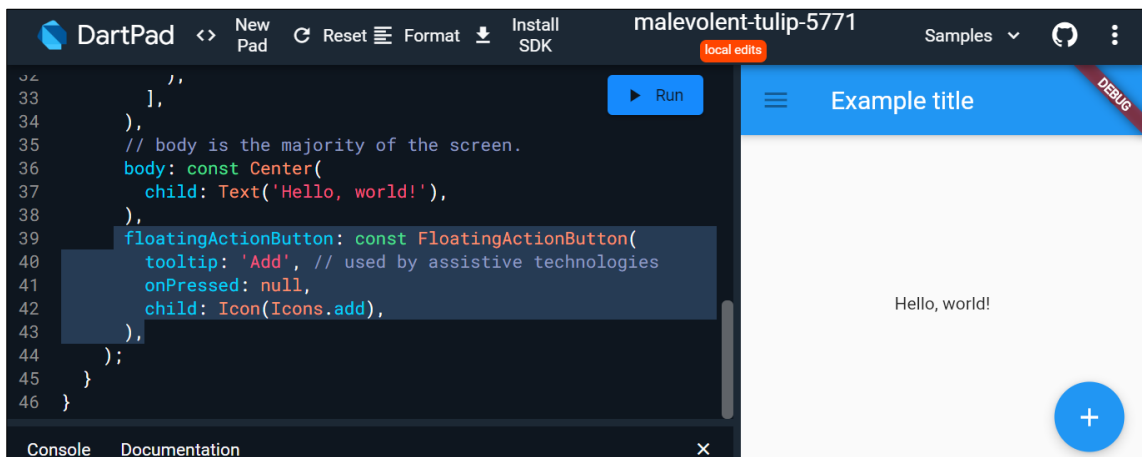
10 }
11
12 class TutorialHome extends StatelessWidget {
13   const TutorialHome({super.key});
14
15   @override
16   Widget build(BuildContext context) {
17     // Scaffold is a layout for
18     // the major Material Components.
19     return Text('hello');
20   }
21 }

```

ภาพประกอบ 1.9 ทดลองใช้งาน Flutter (4)

ที่มา : ณปภัช วรรณตรง (2564 : 20)

7. ย้อนกลับการทำงานโดยการกด Ctrl + Z และทำการเอา floatingActionButton ออก



```

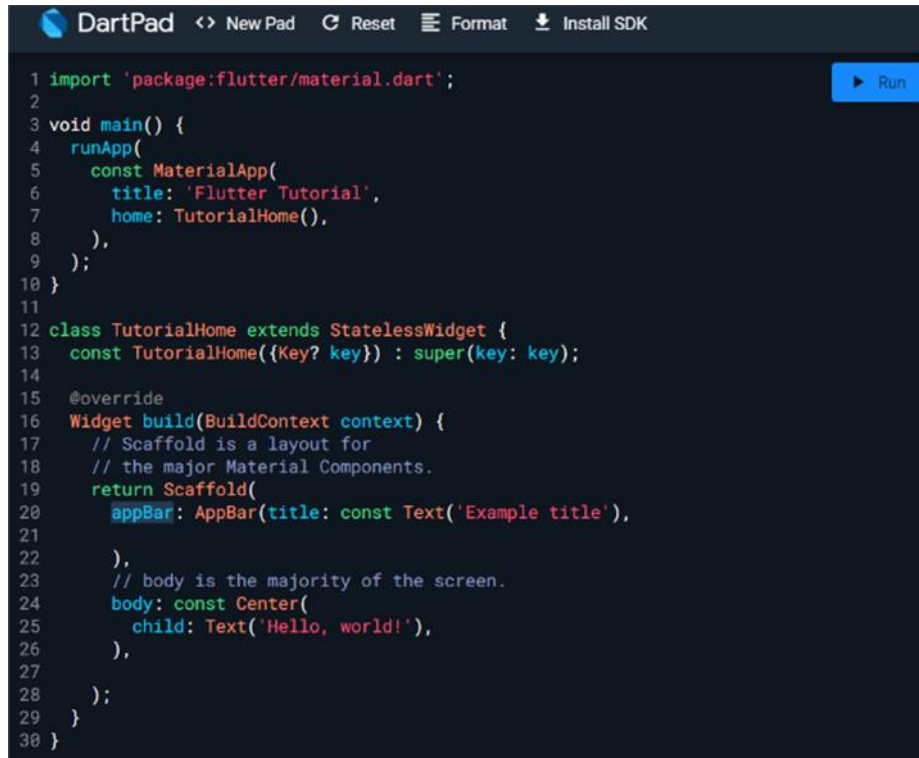
33   ],
34   ),
35   // body is the majority of the screen.
36   body: const Center(
37     child: Text('Hello, world!'),
38   ),
39   floatingActionButton: const FloatingActionButton(
40     tooltip: 'Add', // used by assistive technologies
41     onPressed: null,
42     child: Icon(Icons.add),
43   ),
44 );
45 }
46 }

```

ภาพประกอบ 1.10 ทดลองใช้งาน Flutter (5)

ที่มา : ณปภัช วรรณตรง (2564 : 17)

8. ทำการเอาโค้ดบางส่วนของ appBar ออก



```

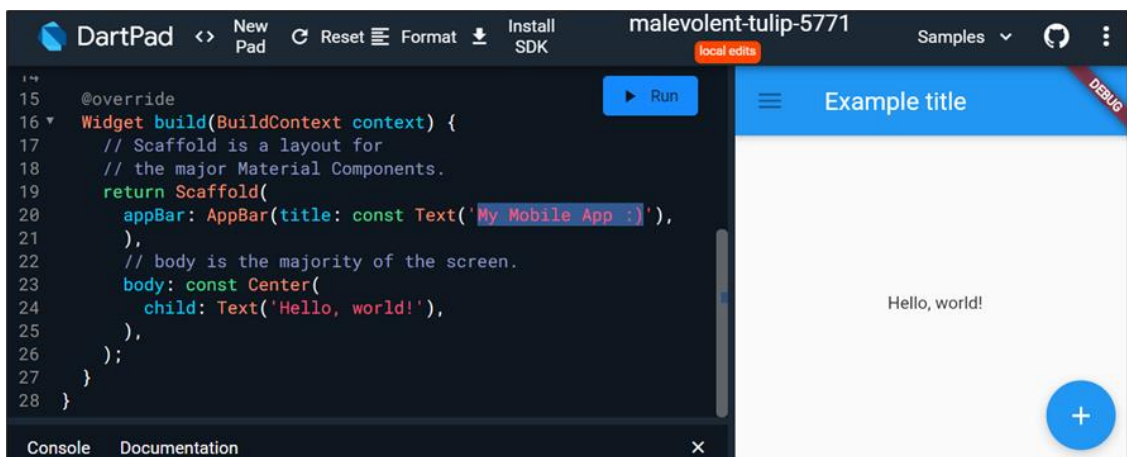
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(
5     const MaterialApp(
6       title: 'Flutter Tutorial',
7       home: TutorialHome(),
8     ),
9   );
10 }
11
12 class TutorialHome extends StatelessWidget {
13   const TutorialHome({Key? key}) : super(key: key);
14
15   @override
16   Widget build(BuildContext context) {
17     // Scaffold is a layout for
18     // the major Material Components.
19     return Scaffold(
20       appBar: AppBar(title: const Text('Example title'),
21       ),
22     // body is the majority of the screen.
23     body: const Center(
24       child: Text('Hello, world!'),
25     ),
26   );
27 }
28 }
29 }
30 }

```

ภาพประกอบ 1.11 ทดลองใช้งาน Flutter (6)

ที่มา : ฅปภัช วรรณตรง (2564 : 18)

9. ทำการเปลี่ยนข้อความของ title ในส่วนของ appBar



```

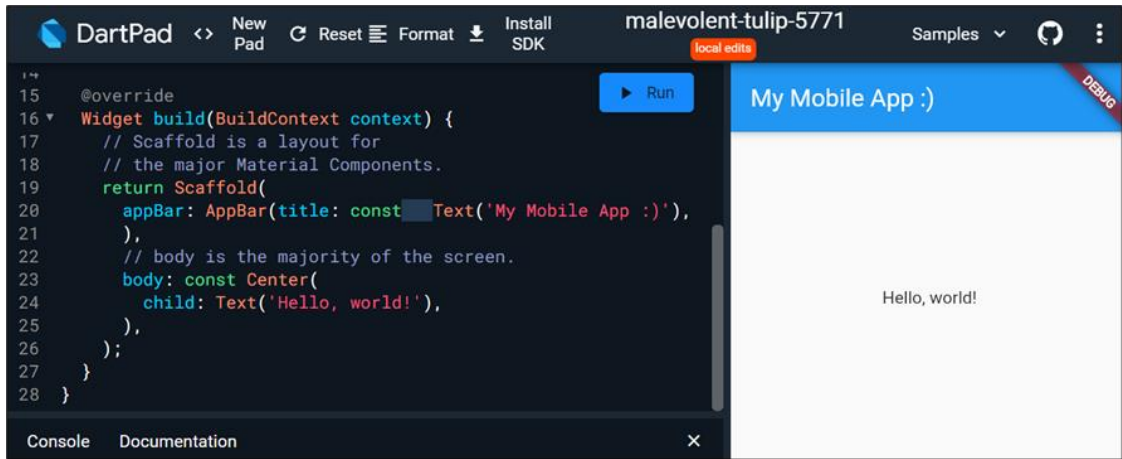
15 @override
16 Widget build(BuildContext context) {
17   // Scaffold is a layout for
18   // the major Material Components.
19   return Scaffold(
20     appBar: AppBar(title: const Text('My Mobile App :)'),
21     ),
22   // body is the majority of the screen.
23   body: const Center(
24     child: Text('Hello, world!'),
25   ),
26 );
27 }
28 }

```

ภาพประกอบ 1.12 ทดลองใช้งาน Flutter (7)

ที่มา : ฅปภัช วรรณตรง (2564 : 24)

10. ทำการทดสอบว่าการเว้นเคาะช่องของข้อความ ไม่มีผลเหมือนกันกับภาษา Python



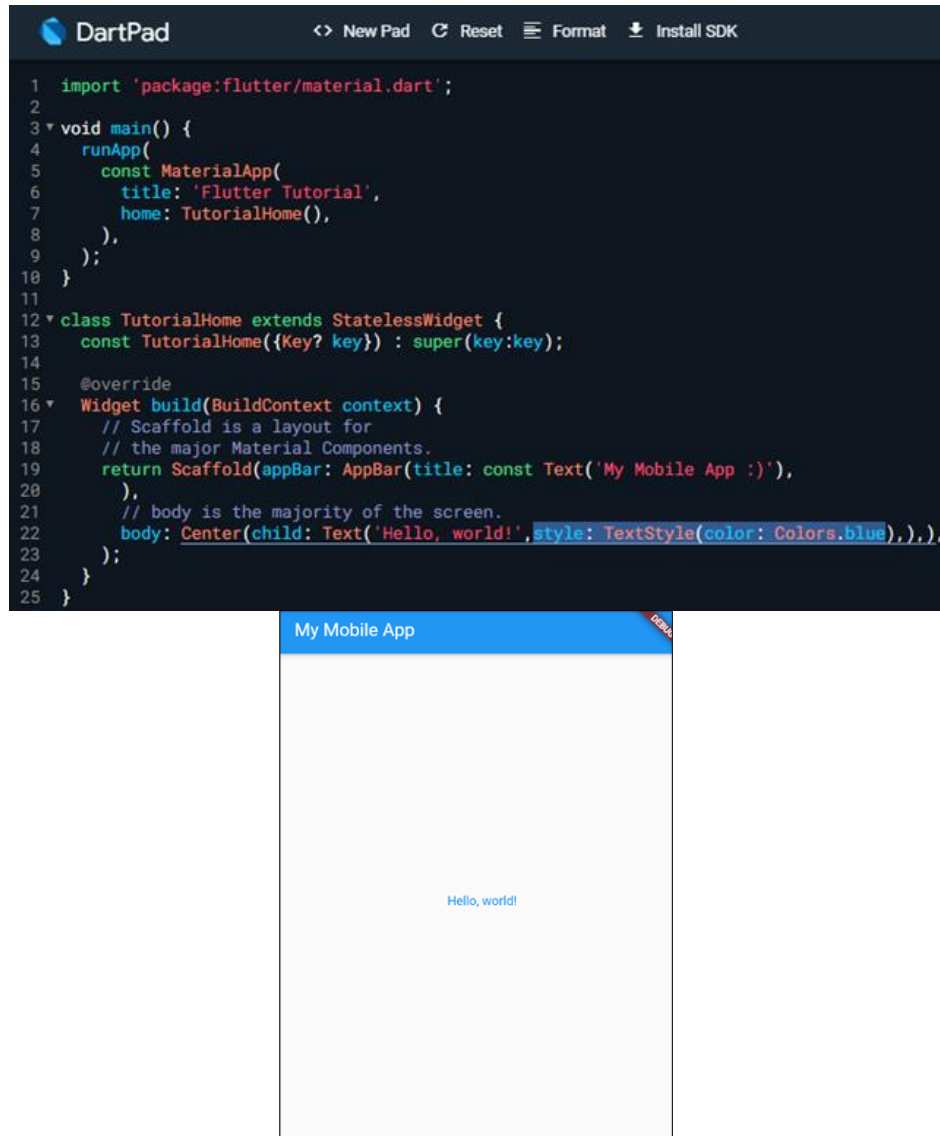
The screenshot shows the DartPad web IDE interface. The top bar includes the DartPad logo, navigation icons, and the user name 'malevolent-tulip-5771'. Below the top bar, there are tabs for 'New Pad', 'Reset', 'Format', and 'Install SDK'. The main editor area displays Dart code for a Flutter widget. The code defines a widget with an AppBar titled 'My Mobile App :)' and a body containing a centered text widget with the text 'Hello, world!'. A 'Run' button is visible next to the code. To the right of the code editor, a preview window shows the rendered mobile app interface, which matches the code: a blue AppBar with the title 'My Mobile App :)' and a white body with the text 'Hello, world!' centered. A 'DEBUG' button is visible in the top right corner of the preview window. At the bottom of the editor, there are tabs for 'Console' and 'Documentation'.

```
15 @override
16 Widget build(BuildContext context) {
17   // Scaffold is a layout for
18   // the major Material Components.
19   return Scaffold(
20     appBar: AppBar(title: const Text('My Mobile App :)'),
21   ),
22   // body is the majority of the screen.
23   body: const Center(
24     child: Text('Hello, world!'),
25   ),
26 );
27 }
28 }
```

ภาพประกอบ 1.13 ทดลองใช้งาน Flutter (8)

ที่มา : ฅนปลัช วรณตรง (2564 : 23)

## 11. ทำการเพิ่ม Style ให้กับ Font และสั่งทำงานเพื่อดูผลลัพธ์



The image shows a screenshot of the DartPad web editor. The top bar includes the DartPad logo and navigation options: '<> New Pad', 'Reset', 'Format', and 'Install SDK'. The main area contains the following Dart code:

```

1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(
5     const MaterialApp(
6       title: 'Flutter Tutorial',
7       home: TutorialHome(),
8     ),
9   );
10 }
11
12 class TutorialHome extends StatelessWidget {
13   const TutorialHome({Key? key}) : super(key:key);
14
15   @override
16   Widget build(BuildContext context) {
17     // Scaffold is a layout for
18     // the major Material Components.
19     return Scaffold(appBar: AppBar(title: const Text('My Mobile App :')),
20       ),
21     // body is the majority of the screen.
22     body: Center(child: Text('Hello, world!', style: TextStyle(color: Colors.blue)),),
23   );
24 }
25 }

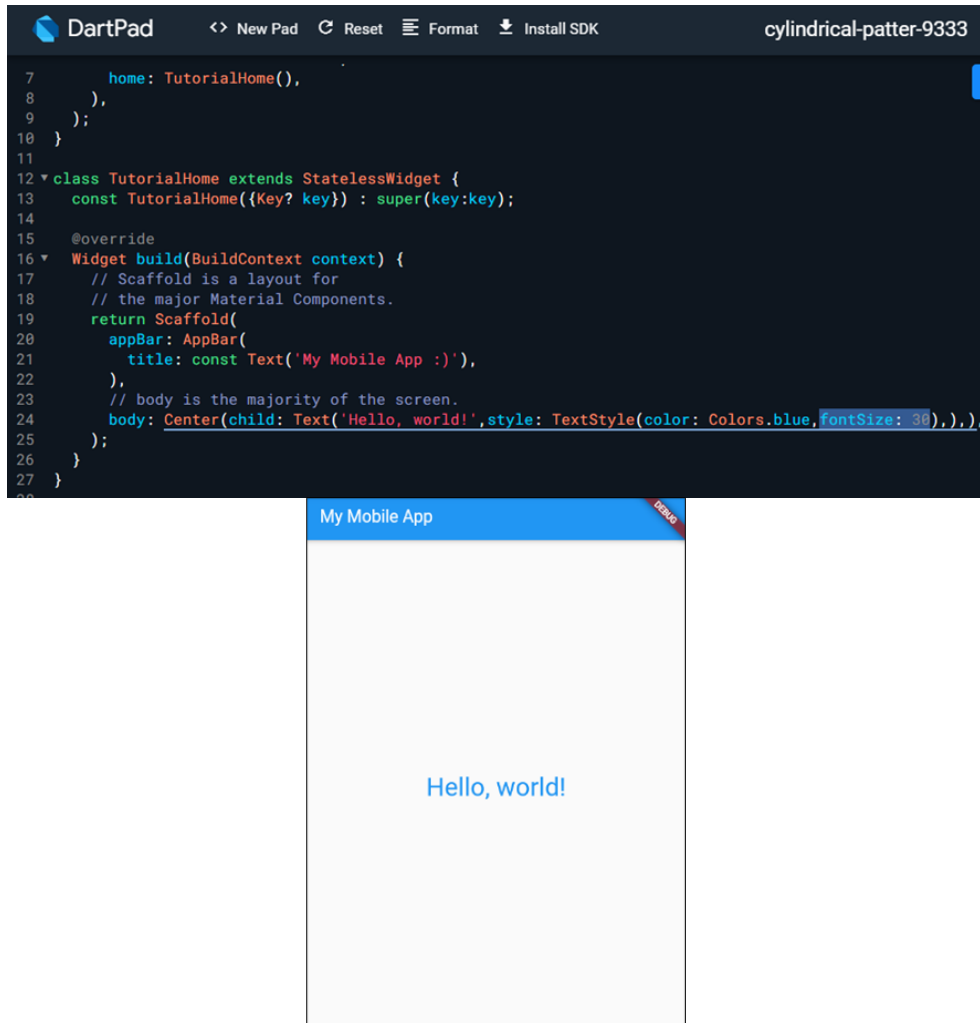
```

Below the code editor, the rendered application is shown. It features a blue header bar with the text 'My Mobile App' and a white body area with the text 'Hello, world!' centered in blue.

ภาพประกอบ 1.14 ทดลองใช้งาน Flutter (9)

ที่มา : ฌปภัช วรรณตรง (2564 : 25)

## 12. ทำการเพิ่มขนาดให้กับ Font



ภาพประกอบ 1.15 ทดลองใช้งาน Flutter (10)

ที่มา : ณปภัช วรณตรง (2564 : 26)

### สรุป Flutter

1) ชุดโปรแกรมด้าน User Interface ถูกสร้างโดย Google 2) จัดการส่วนต่าง ๆ ได้ง่ายและสะดวก 3) เป็น Open Source ใช้งานได้ฟรี 4) พัฒนา Mobile Application ให้สามารถใช้งานได้ทั้ง iOS และ Android

## การเริ่มต้นสร้างแอปพลิเคชันด้วย Flutter

### 1. ลงโปรแกรมที่เกี่ยวข้อง

ดาวน์โหลดโปรแกรม Android Studio สำหรับใช้ในการแสดงผลการทำงานแอปพลิเคชัน

#### 1.1 สำหรับระบบปฏิบัติการ Windows

## Get the Flutter SDK

1. Download the following installation bundle to get the latest stable release of the Flutter SDK:

`flutter_windows_2.2.3-stable.zip`

For other release channels, and older builds, see the [SDK releases](#) page.

### ภาพประกอบ 1.16 Flutter สำหรับระบบปฏิบัติการ Windows

ที่มา : Flutter. (2020 : 1)

1.1.1 ทำการดาวน์โหลด Flutter SDK ผ่านทางเว็บเบราว์เซอร์ของผู้พัฒนา

1.1.2 เมื่อทำการดาวน์โหลดเสร็จแล้ว ให้ทำการแยกไฟล์ที่ดาวน์โหลดมา ไปเก็บไว้ยัง

ตำแหน่งของโฟลเดอร์ที่ต้องการ เช่น ตำแหน่งโฟลเดอร์ที่ไดร์ฟ C:\

#### 1.2 สำหรับระบบปฏิบัติการ macOS

## Get the Flutter SDK

1. Download the following installation bundle to get the latest stable release of the Flutter SDK:

`flutter_macos_2.2.3-stable.zip`

For other release channels, and older builds, see the [SDK releases](#) page.

### ภาพประกอบ 1.17 Flutter สำหรับระบบปฏิบัติการ macOS

ที่มา : Flutter. (2020 : 1)

1.2.1 ทำการดาวน์โหลด Flutter SDK ผ่านทางเว็บเบราว์เซอร์ของผู้พัฒนา

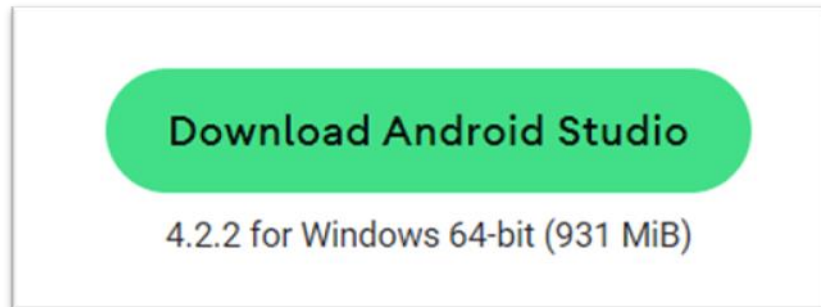
1.2.2 เมื่อทำการดาวน์โหลดเสร็จแล้ว ให้ทำการแยกไฟล์ที่ดาวน์โหลดมา ไปเก็บไว้ยัง

ตำแหน่งของโฟลเดอร์ที่ต้องการ เช่น ตำแหน่งโฟลเดอร์ที่ไดร์ฟ C:\



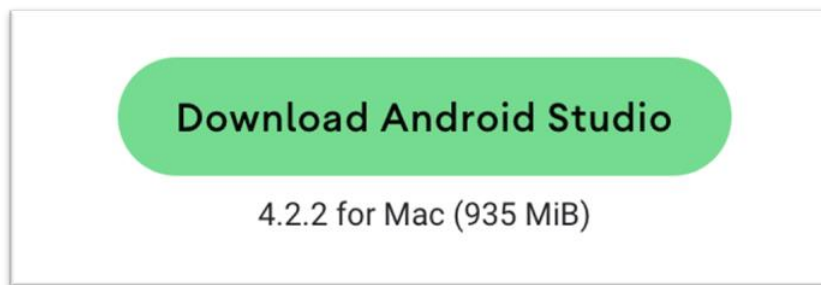
ดาวน์โหลดโปรแกรมพัฒนาแอปพลิเคชันบนมือถือ Android Studio

1.3 สามารถดาวน์โหลดได้ทั้งระบบปฏิบัติการ Windows และ macOS



ภาพประกอบ 1.18 Android Studio สำหรับระบบปฏิบัติการ Windows

ที่มา : Android Studio. (2020 : 1)



ภาพประกอบ 1.19 Android Studio สำหรับระบบปฏิบัติการ macOS

ที่มา : Android Studio. (2020 : 1)

1.3.1 ทำการดาวน์โหลด Android Studio ผ่านทางเว็บเบราว์เซอร์ของผู้พัฒนา

1.3.2 เมื่อดาวน์โหลดเสร็จแล้วให้ทำการเปิด Android Studio ที่ได้ทำการดาวน์โหลดมา

โหลดมา

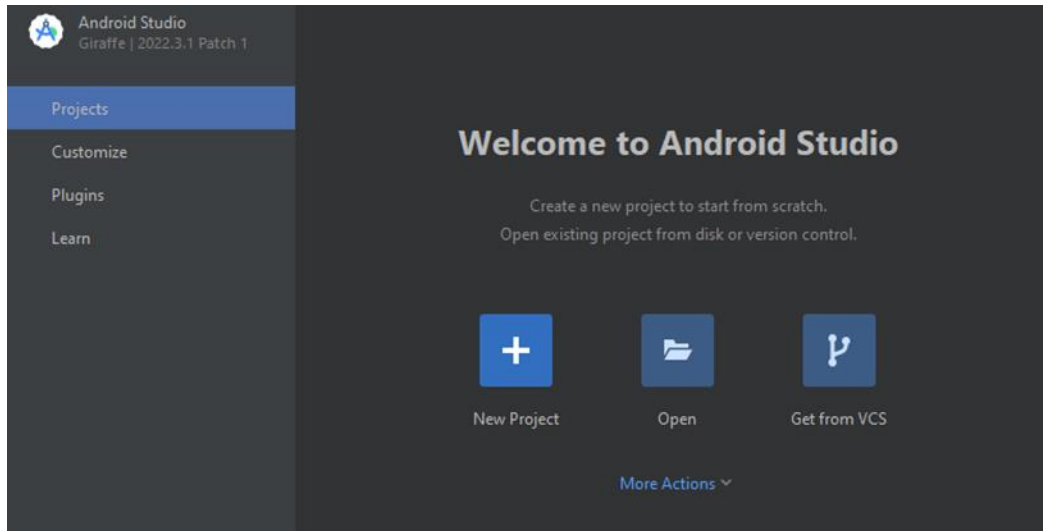
1.3.3 เลือกการติดตั้งเป็นหมวด Android Virtual Device

1.3.4 กดถัดไป (Next) จนถึงหน้าต่างตัวเลือก SDK จากนั้นทำการเลือกที่ SDK แล้วกดถัดไปจนถึงกระบวนการติดตั้ง หลังจากนั้นให้รอจนทำการติดตั้งเสร็จสิ้น

1.3.5 หลังจากทำการติดตั้งจนเสร็จสิ้นแล้วให้ทำการเปิดโปรแกรม Android Studio

ขึ้นมา

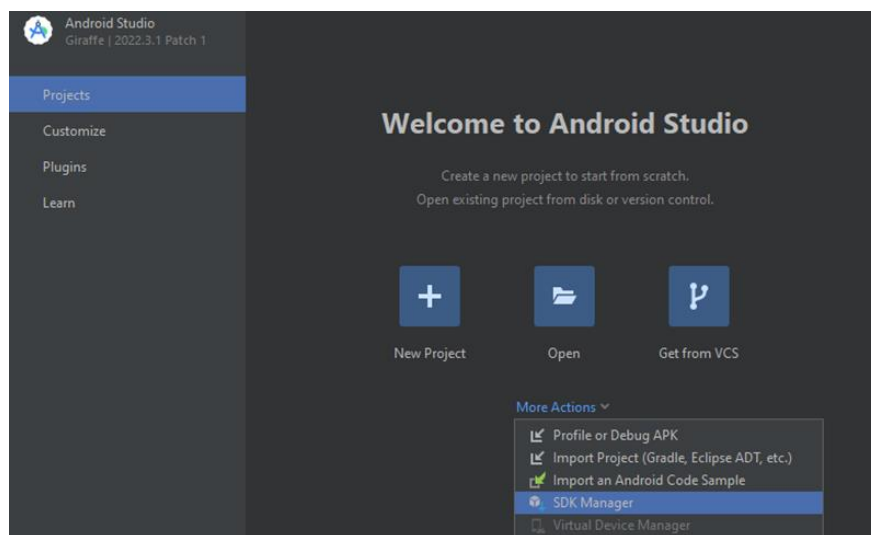
เมื่อเปิดโปรแกรม Android Studio ขึ้นมาแล้วให้ทำการตั้งค่าเครื่องมือสำหรับการพัฒนา



ภาพประกอบ 1.20 ตั้งค่าเครื่องมือสำหรับการพัฒนา

ที่มา : ฌปภัช วรรณตรง (2564 : 17)

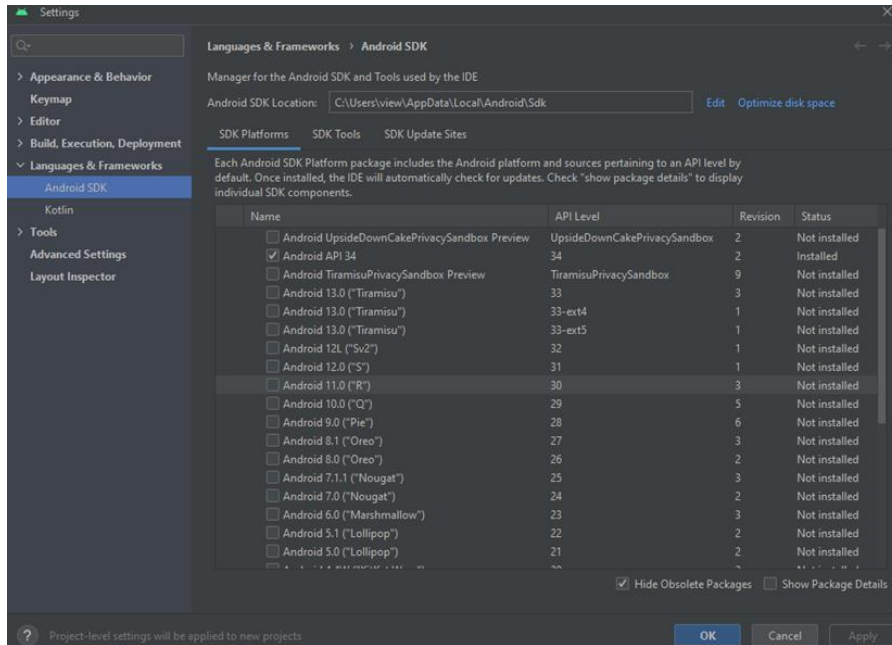
เลือกที่ More Action จากนั้นให้เลือกเป็น SDK Manager



ภาพประกอบ 1.21 ตั้งค่าเครื่องมือสำหรับการพัฒนา (ต่อ)

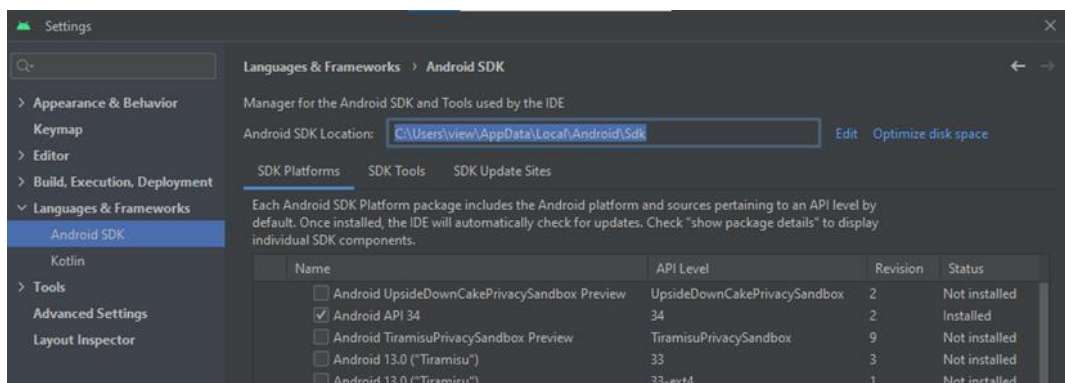
ที่มา : ฌปภัช วรรณตรง (2564 : 17)

เมื่อเลือก SDK Manager แล้วจะปรากฏหน้าต่างการตั้งค่าขึ้นมา



ภาพประกอบ 1.22 ตั้งค่าเครื่องมือสำหรับการพัฒนา (ต่อ)  
ที่มา : ฅนปลั๊ก วรรณตรง (2564 : 18)

จากนั้นให้ทำการคัดลอก Path เอาไว้ สำหรับนำไปกำหนด Path ใน Edit Environment



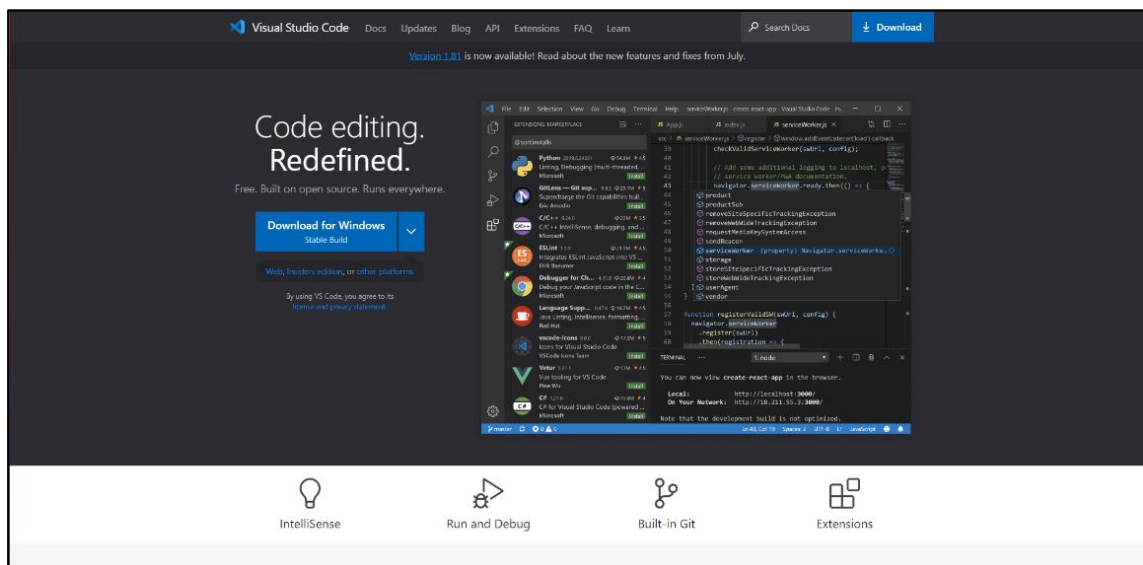
ภาพประกอบ 1.23 ตั้งค่าเครื่องมือสำหรับการพัฒนา (ต่อ)  
ที่มา : ฅนปลั๊ก วรรณตรง (2564 : 18)

## หมายเหตุ

การเรียนการสอนจริงอาจส่งการทำงานผ่านบราวเซอร์ เนื่องจาก Android Studio ใช้หน่วยความจำในการทำงานจำนวนมาก คอมพิวเตอร์ในห้องปฏิบัติคอมพิวเตอร์ที่ใช้เรียนอาจไม่สามารถรองรับการใช้งานได้

ดาวน์โหลดโปรแกรม Editor ที่ใช้ในการเขียน Flutter คือ Visual Studio Code

### 1.4 สามารถดาวน์โหลดได้ทั้งระบบปฏิบัติการ Windows และ macOS



ภาพประกอบ 1.24 Visual Studio Code สำหรับทุกระบบปฏิบัติการ

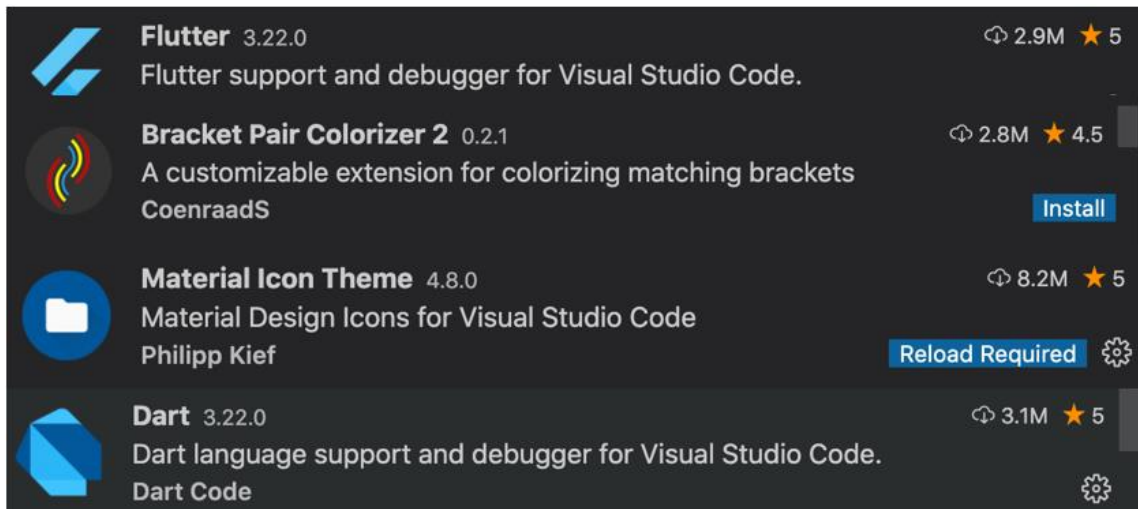
ที่มา : Visual Studio Code. (2020 : 1)

1.4.1 ทำการดาวน์โหลด Visual Studio Code ผ่านทางเว็บเบราว์เซอร์ของผู้พัฒนา

1.4.2 เมื่อดาวน์โหลดเสร็จแล้วให้ทำการเปิด Visual Studio Code ที่ได้ทำการดาวน์โหลดมา

1.4.3 ทำการติดตั้งโปรแกรม Visual Studio Code ให้เสร็จสมบูรณ์

1.4.4 หลังจากติดตั้งเสร็จสิ้นแล้วให้ทำการติดตั้งส่วนเสริมดังภาพ



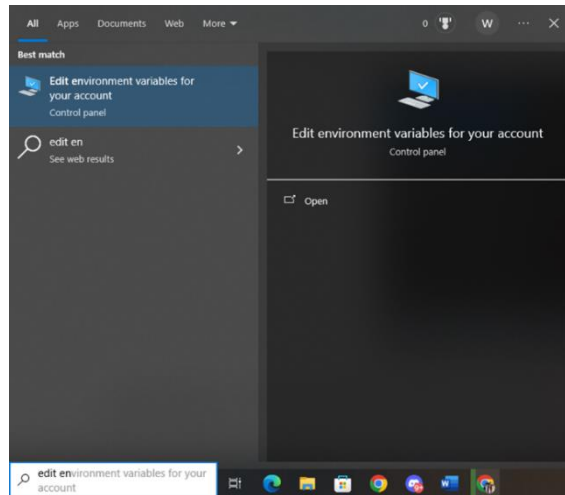
ภาพประกอบ 1.25 ส่วนเสริมการใช้งานผ่าน Virtual Studio Code

ที่มา : ณปภัช วรรณตรง (2564 : 10)

เพิ่มเติม
Dart SDK Version 2 ขึ้นไปจะมาพร้อมกับ Flutter อยู่แล้วในโฟลเดอร์ bin ไม่จำเป็นต้องดาวน์โหลดเพิ่มเติม แต่หากต้องการดาวน์โหลด สามารถดาวน์โหลดผ่านลิงค์ <a href="https://dart.dev/tools/sdk/archive">https://dart.dev/tools/sdk/archive</a>
Xcode สำหรับทดลองรันโปรแกรมบน iPhone รูปแบบจำลอง สามารถดาวน์โหลดได้ผ่านลิงค์ <a href="https://developer.apple.com/xcode/">https://developer.apple.com/xcode/</a>

## 2. กำหนดค่าในการทำงานของโปรแกรม Set Path

### 2.1 ไปที่ Edit Environment

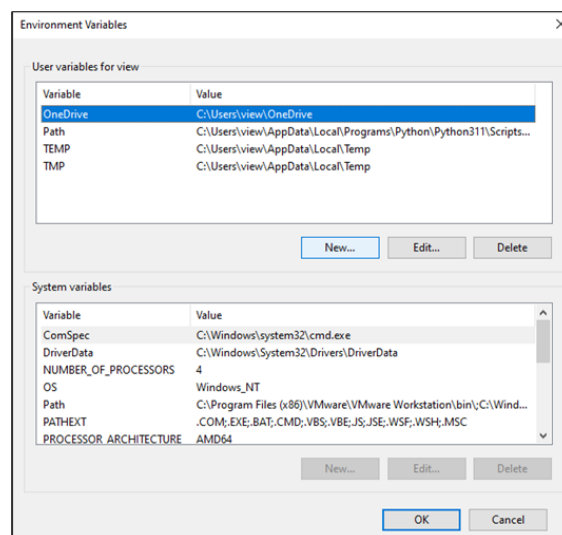


ภาพประกอบ 1.26 กำหนดค่า ANDROID\_HOME ใน Edit Environment

ที่มา : ฌปภัช วรรณตรง (2564 : 19)

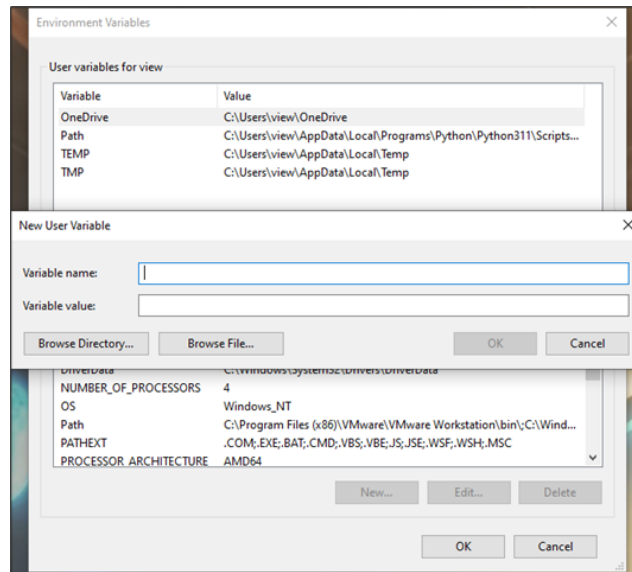
2.2 สร้างและกำหนดค่า ANDROID\_HOME ให้มี Path เป็น Path เดียวกับใน Android Studio ที่ได้ทำการคัดลอกไว้ก่อนหน้านี้

เมื่อเปิดหน้าต่าง Edit Environment แล้วให้ทำการไปที่ New เพิ่มทำการเพิ่ม Path

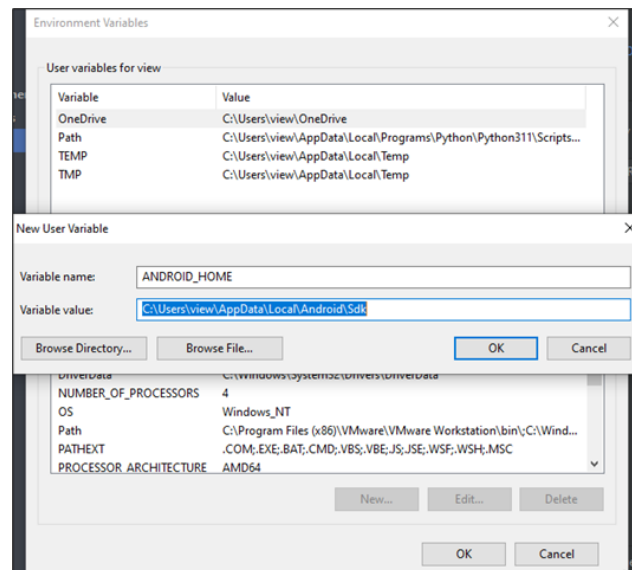


ภาพประกอบ 1.27 กำหนดค่า ANDROID\_HOME ใน Edit Environment (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 20)



ภาพประกอบ 1.28 กำหนดค่า ANDROID\_HOME ใน Edit Environment (ต่อ)  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 20)



ภาพประกอบ 1.29 สร้างและกำหนดค่า ANDROID\_HOME  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 20)

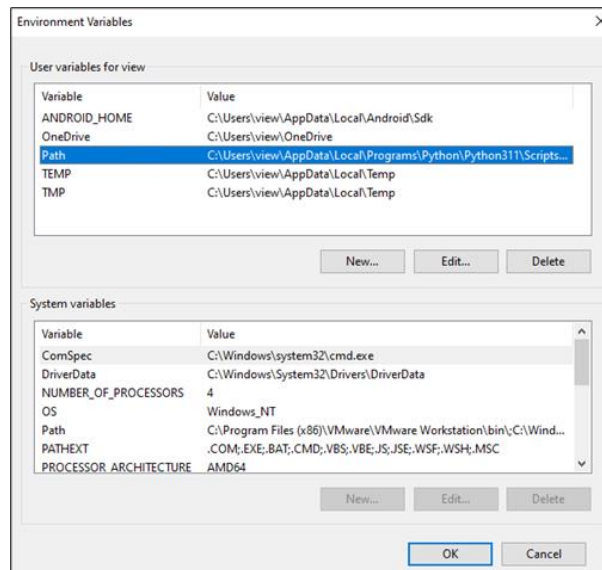
## 2.3 ไปที่ Path สร้าง Path ใหม่ ดังนี้

C:\flutter\bin

%ANDROID\_HOME%\tools

%ANDROID\_HOME%\platform-tools

### 2.3.1 ไปที่ Path

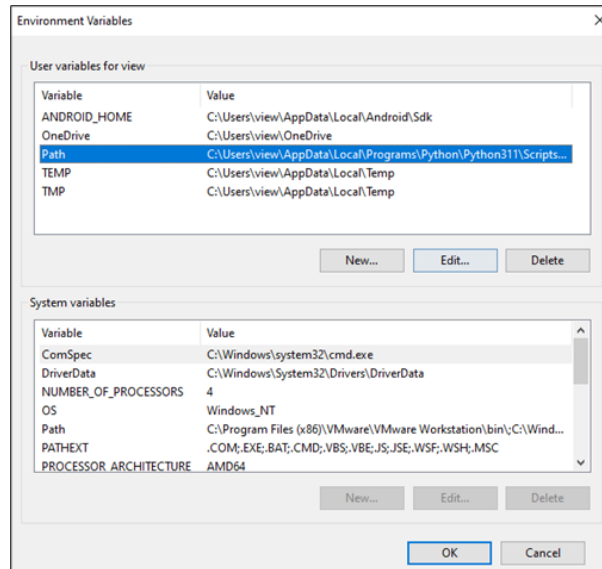


ภาพประกอบ 1.30 สร้าง Path ใหม่

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 21)

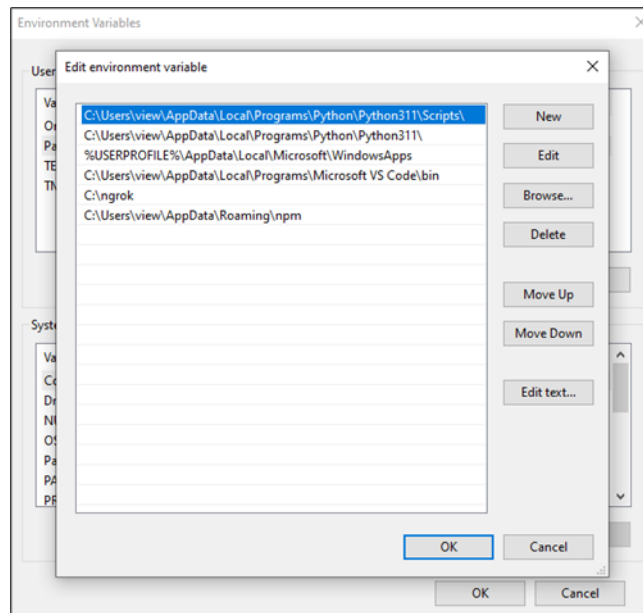


### 2.3.2 จากนั้นให้กดที่ Edit



ภาพประกอบ 1.31 สร้าง Path ใหม่ (ต่อ)

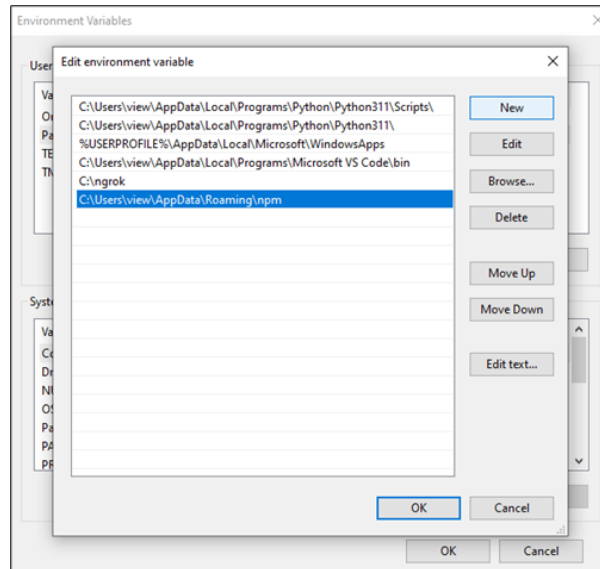
ที่มา : ฅนปลั๊ก วรณตรง (2564 : 21)



ภาพประกอบ 1.32 สร้าง Path ใหม่ (ต่อ)

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 21)

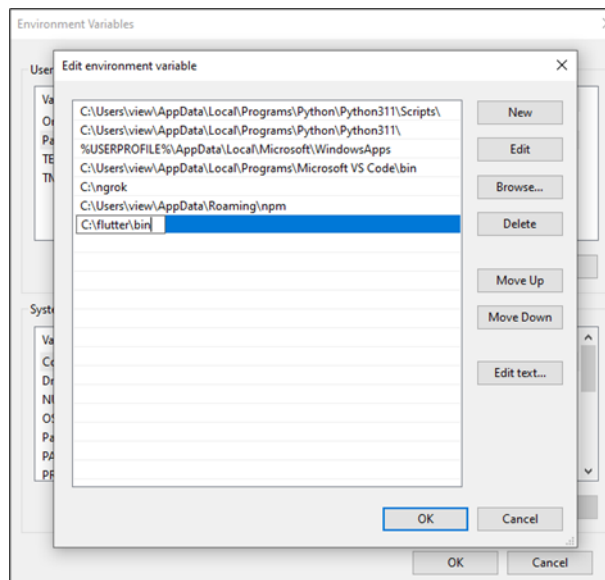
### 2.3.3 เลือก Path สุดท้ายจากนั้นกด New



ภาพประกอบ 1.33 สร้าง Path ใหม่ (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 21)

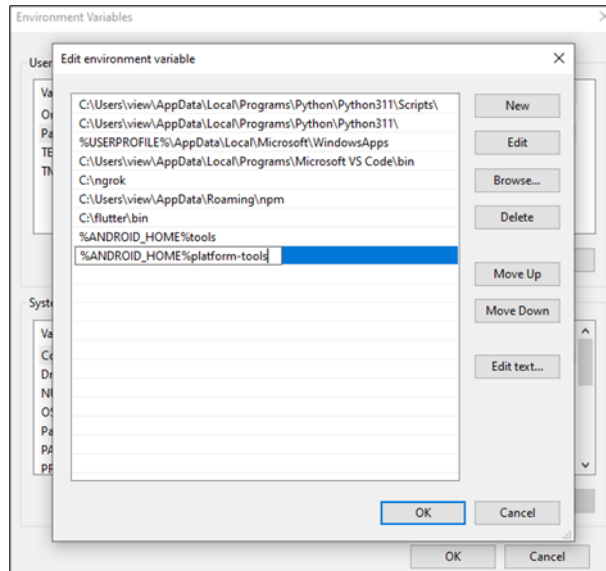
### 2.3.4 จากนั้นให้ทำการกำหนด Path ที่ได้กำหนดไว้



ภาพประกอบ 1.34 สร้าง Path ใหม่ (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 21)

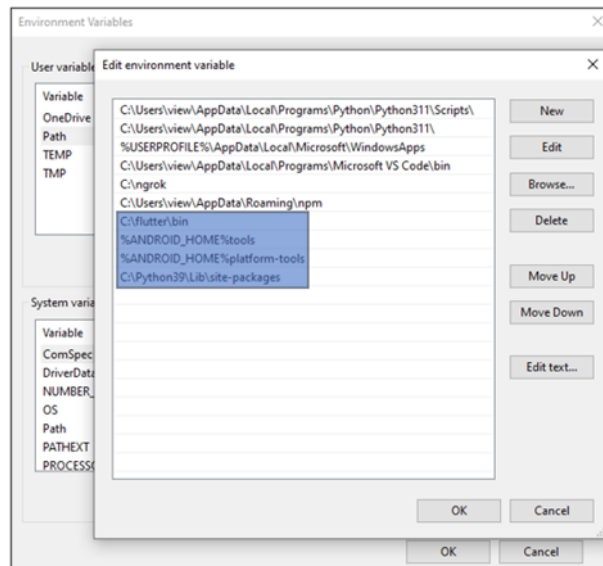
### 2.3.5 จากนั้นกำหนด Path ให้ครบและกด OK



ภาพประกอบ 1.35 สร้าง Path ใหม่ (ต่อ)

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 21)

### 2.3.6 ได้ผลลัพธ์การเพิ่ม Path ดังภาพ



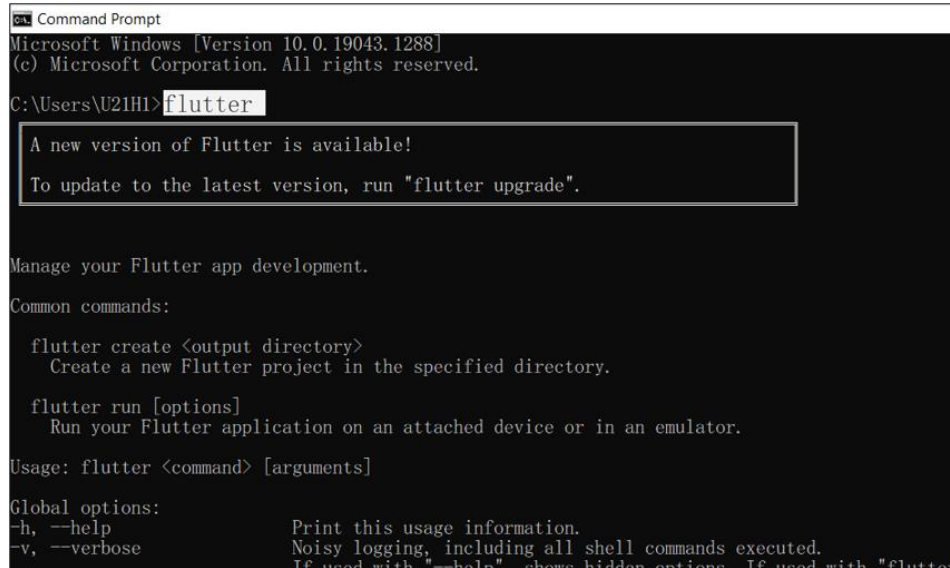
ภาพประกอบ 1.36 สร้าง Path ใหม่ (ต่อ)

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 21)

### 3. ตรวจสอบการใช้งานด้วย Command Prompt

ตรวจสอบการใช้งานด้วย Command Prompt ได้ดังนี้

#### 3.1 ตรวจสอบ Flutter ว่าสามารถใช้งานได้หรือไม่ ด้วยการพิมพ์ Flutter



```

Command Prompt
Microsoft Windows [Version 10.0.19043.1288]
(c) Microsoft Corporation. All rights reserved.

C:\Users\U21H1>flutter

A new version of Flutter is available!
To update to the latest version, run "flutter upgrade".

Manage your Flutter app development.

Common commands:

flutter create <output directory>
  Create a new Flutter project in the specified directory.

flutter run [options]
  Run your Flutter application on an attached device or in an emulator.

Usage: flutter <command> [arguments]

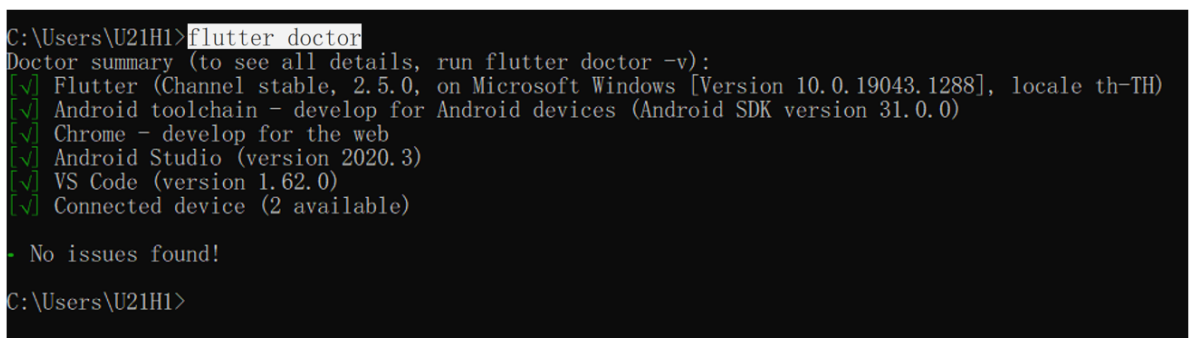
Global options:
-h, --help           Print this usage information.
-v, --verbose        Noisy logging, including all shell commands executed.
                    If used with "--help", shows hidden options. If used with "flutter
  
```

ภาพประกอบ 1.37 Flutter

ที่มา : ณปภัช วรรณตรง (2564 : 23)

#### 3.2 ตรวจสอบว่าโปรแกรมต่าง ๆ ที่ติดตั้งไว้สามารถใช้งานได้หรือไม่ ด้วยการพิมพ์ Flutter

Doctor กรณี “ผ่าน” จะได้ผลดังภาพ



```

C:\Users\U21H1>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 2.5.0, on Microsoft Windows [Version 10.0.19043.1288], locale th-TH)
[✓] Android toolchain - develop for Android devices (Android SDK version 31.0.0)
[✓] Chrome - develop for the web
[✓] Android Studio (version 2020.3)
[✓] VS Code (version 1.62.0)
[✓] Connected device (2 available)

- No issues found!

C:\Users\U21H1>
  
```

ภาพประกอบ 1.38 Flutter Doctor

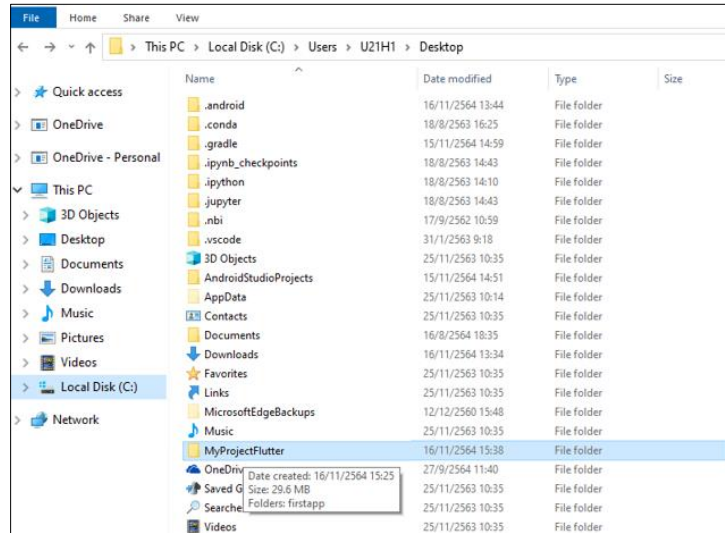
ที่มา : ณปภัช วรรณตรง (2564 : 23)

### 3.2 Flutter Doctor กรณี “ไม่ผ่าน” ให้ทำการคัดลอกส่วนที่ผิดพลาดหรือข้อผิดพลาด ไปค้นหาวิธีแก้ไขในอินเทอร์เน็ตเบราว์เซอร์

หมายเหตุ
<p>Flutter doctor กรณี “ไม่ผ่าน” ถ้าเจอ error ใช้คำสั่งต่อไปนี้</p> <ol style="list-style-type: none"> <li>1) ถ้าเจอคำว่า Android Studio not Installed flutter config --android-studio-dir = “C:\Program Files\Android\Android Studio” Path ใน “ ” ให้ใส่ Path ที่เก็บจริงของเครื่องผู้ใช้</li> <li>2) ถ้าเจอคำว่า Android Accept License flutter doctor --android-licenses ถ้าผ่าน Android Accept License จะขึ้นให้กด y enter เพื่อ Accept ตลอด</li> <li>3) ถ้าเจอคำว่า Android toolchain - develop for Android devices X ANDROID_HOME = C:\Users\User\AppData\Local\Android\Sdk but Android SDK not found at this location. แก้โดยพิมพ์ flutter config --android-sdk ใส่ path SDK ของผู้ใช้ตรงนี้</li> <li>4) ถ้าเจอคำว่า No devices available วิธีแก้ปัญหา No device รันคำสั่งต่อไปนี้ ทีละคำสั่ง flutter channel beta flutter upgrade flutter config --enable-web</li> </ol> <p>(ที่มา: cr. <a href="https://stackoverflow.com/questions/49175231/flutter-does-not-find-android-sdk">https://stackoverflow.com/questions/49175231/flutter-does-not-find-android-sdk</a>)</p>

## 4. เริ่มสร้าง Project

### 4.1 สร้าง Folder ตั้งชื่อตามต้องการ



ภาพประกอบ 1.39 สร้าง Folder

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 30)

### 4.2 cd เข้า Folder ที่สร้าง ใช้คำสั่ง cd

“C:\Users\NAPAPHAT\NapaphatFlutterProject” หมายถึง Path ใน “ ” เปลี่ยนตามที่ตั้ง

```

C:\Windows\System32\cmd.exe
C:\Users\U21H1>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 2.5.0, on Microsoft Windows [Version 10.0.19043.1288], locale th-TH)
[✓] Android toolchain - develop for Android devices (Android SDK version 31.0.0)
[✓] Chrome - develop for the web
[✓] Android Studio (version 2020.3)
[✓] VS Code (version 1.62.0)
[✓] Connected device (2 available)

• No issues found!

C:\Users\U21H1>cd "C:\Users\U21H1\Desktop\MyProjectFlutter"
C:\Users\U21H1\Desktop\MyProjectFlutter>

```

ภาพประกอบ 1.40 cd เข้า Folder ที่สร้าง

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 31)

4.3 สร้าง Project ด้วยคำสั่ง flutter create ตามด้วยชื่อที่ต้องการ (แต่ต้องเป็นตัวอักษรภาษาอังกฤษขนาดเล็กทั้งหมด)



```

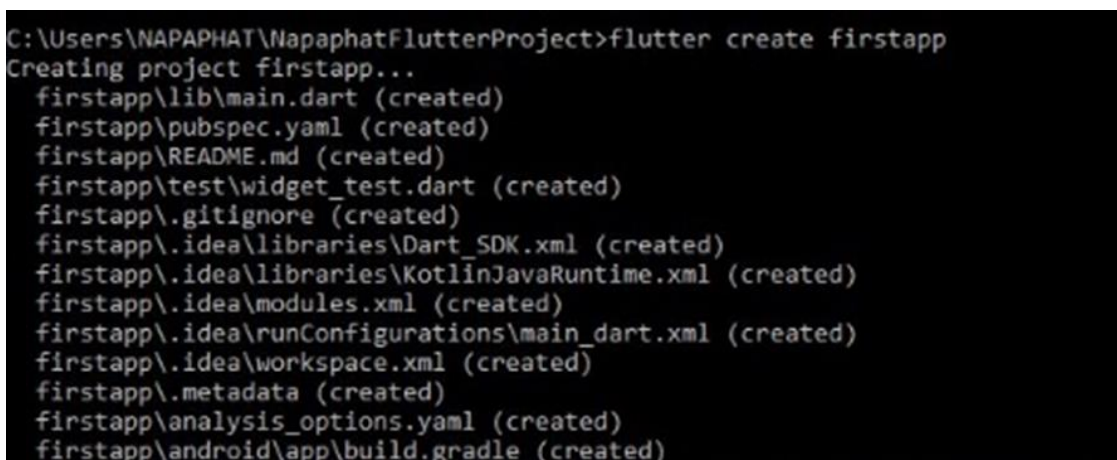
C:\Windows\System32\cmd.exe
C:\Users\U21H1>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 2.5.0, on Microsoft Windows [Version 10.0.19043.1288], locale th-TH)
[✓] Android toolchain - develop for Android devices (Android SDK version 31.0.0)
[✓] Chrome - develop for the web
[✓] Android Studio (version 2020.3)
[✓] VS Code (version 1.62.0)
[✓] Connected device (2 available)

- No issues found!
C:\Users\U21H1>cd "C:\Users\U21H1\Desktop\MyProjectFlutter"
C:\Users\U21H1\Desktop\MyProjectFlutter>flutter create firstapp

```

ภาพประกอบ 1.41 สร้าง Project

ที่มา : ฌปภัช วรรณตรง (2564 : 32)



```

C:\Users\NAPAPHAT\NapaphatFlutterProject>flutter create firstapp
Creating project firstapp...
  firstapp\lib\main.dart (created)
  firstapp\pubspec.yaml (created)
  firstapp\README.md (created)
  firstapp\test\widget_test.dart (created)
  firstapp\.gitignore (created)
  firstapp\.idea\libraries\Dart_SDK.xml (created)
  firstapp\.idea\libraries\KotlinJavaRuntime.xml (created)
  firstapp\.idea\modules.xml (created)
  firstapp\.idea\runConfigurations\main_dart.xml (created)
  firstapp\.idea\workspace.xml (created)
  firstapp\.metadata (created)
  firstapp\analysis_options.yaml (created)
  firstapp\android\app\build.gradle (created)

```

ภาพประกอบ 1.42 สร้าง Project (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 32)

4.4 ใช้คำสั่ง cd ชื่อ Project ที่สร้างไว้ที่ขั้นตอน 4.3 เพื่อเข้าสู่ Folder Project ที่สร้าง

```
C:\Users\U21H1\Desktop\myflutter>flutter create firstapp
Creating project firstapp...
Running "flutter pub get" in firstapp... 2,669ms
Wrote 127 files.

All done!
In order to run your application, type:

  $ cd firstapp
  $ flutter run

Your application code is in firstapp\lib\main.dart.
C:\Users\U21H1\Desktop\myflutter>cd firstapp
C:\Users\U21H1\Desktop\myflutter\firstapp>
```

ภาพประกอบ 1.43 ใช้คำสั่ง cd firstapp เพื่อเข้าสู่ Folder Project

ที่มา : ฌปภัช วรรณตรง (2564 : 32)

4.5 ใช้คำสั่ง Flutter run เพื่อสั่งการทำงาน project

```
C:\Users\U21H1\Desktop\myflutter\firstapp>flutter run
Multiple devices found:
Windows (desktop) • windows • windows-x64 • Microsoft Windows [Version 10.0.19045.2130]
Chrome (web)      • chrome • web-javascript • Google Chrome 107.0.5304.107
Edge (web)        • edge   • web-javascript • Microsoft Edge 107.0.1418.42
[1]: Windows (windows)
[2]: Chrome (chrome)
[3]: Edge (edge)
Please choose one (To quit, press "q/Q"):
```

ภาพประกอบ 1.44 ใช้คำสั่ง flutter run เพื่อสั่งทำงาน

ที่มา : ฌปภัช วรรณตรง (2564 : 33)



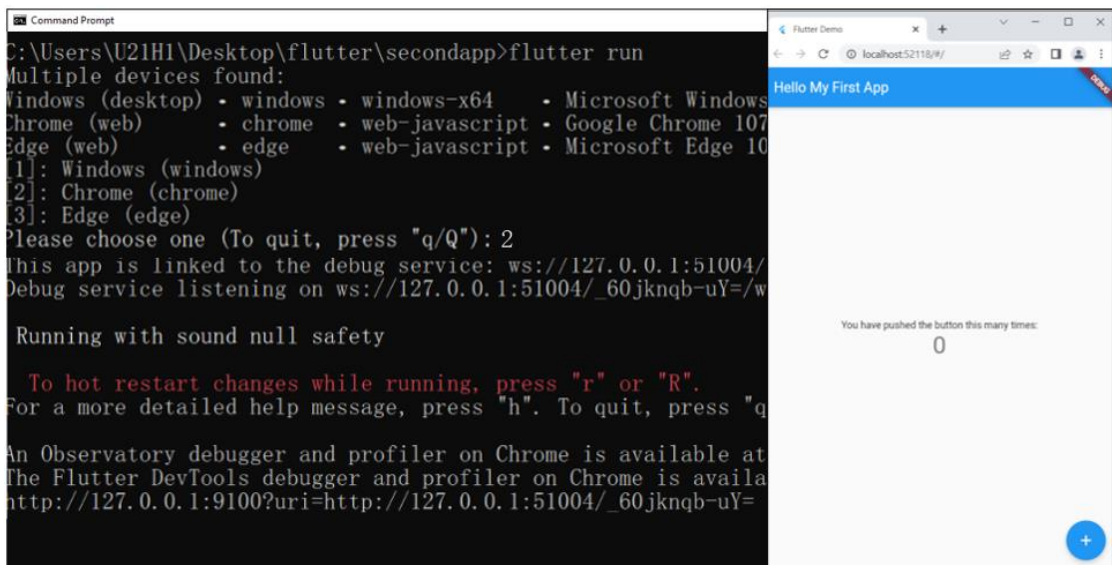
4.6 เมื่อทำการติดตั้งและสั่งทำการทำงาน Flutter เป็นครั้งแรก Flutter จะให้ทำการเลือก Device สำหรับเปิดหน้าเว็บเบราว์เซอร์

โดยจะมีตัวเลือกอยู่สองอย่าง 1) Chrome 2) Edge สามารถเลือกเว็บเบราว์เซอร์สำหรับการแสดงหน้าเว็บแอปฯ ได้ตามต้องการ

```
C:\Users\U21H1\Desktop\myflutter\firstapp>flutter run
Multiple devices found:
Windows (desktop) - windows - windows-x64 - Microsoft Windows [Version 10.0.19045.2130]
Chrome (web) - chrome - web-javascript - Google Chrome 107.0.5304.107
Edge (web) - edge - web-javascript - Microsoft Edge 107.0.1418.42
[1]: Windows (windows)
[2]: Chrome (chrome)
[3]: Edge (edge)
Please choose one (To quit, press "q/Q"):
```

ภาพประกอบ 1.45 เลือก Device

ที่มา : ฌปภัช วรรณตรง (2564 : 35)

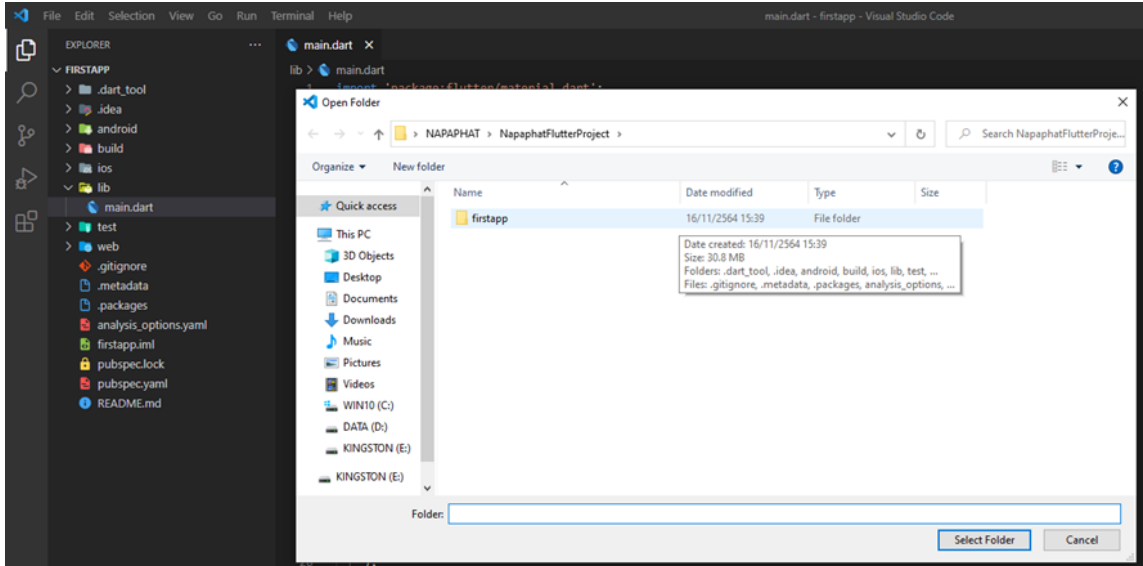


ภาพประกอบ 1.46 เลือก Device (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 36)

## 5. เปิดไฟล์ใน VS Code

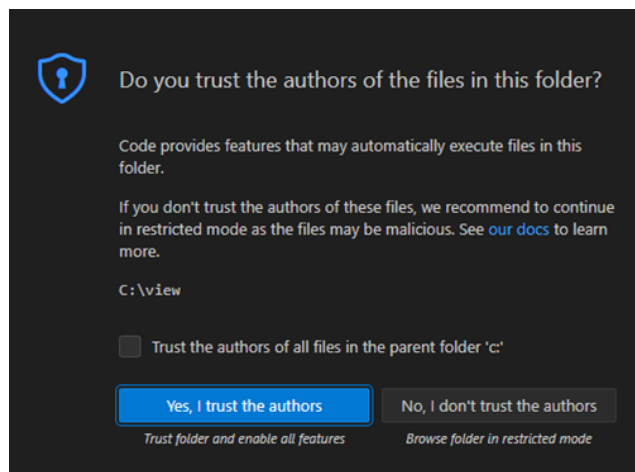
### 5.1 เปิด VS Code เลือก Folder Project



ภาพประกอบ 1.47 เปิด VS Code

ที่มา : ฌปภัช วรรณตรง (2564 : 38)

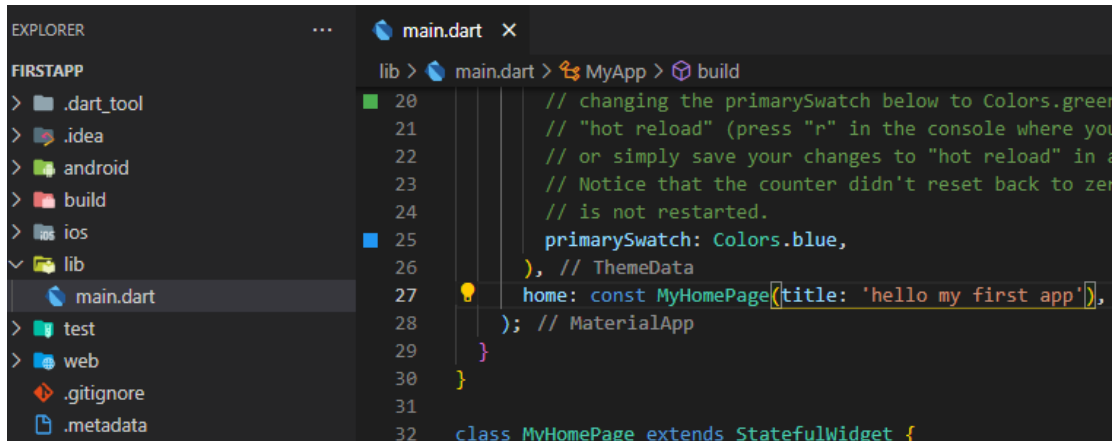
หากขึ้นหน้าต่างสอบถามความน่าเชื่อถือ ให้ตอบ Yes



ภาพประกอบ 1.48 เปิด VS Code (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 38)

## 5.2 เปิด Folder lib &gt; เลือกไฟล์ main.dart



```

EXPLORER
FIRSTAPP
> .dart_tool
> .idea
> android
> build
> ios
✓ lib
  main.dart
> test
> web
.gitignore
.metadata

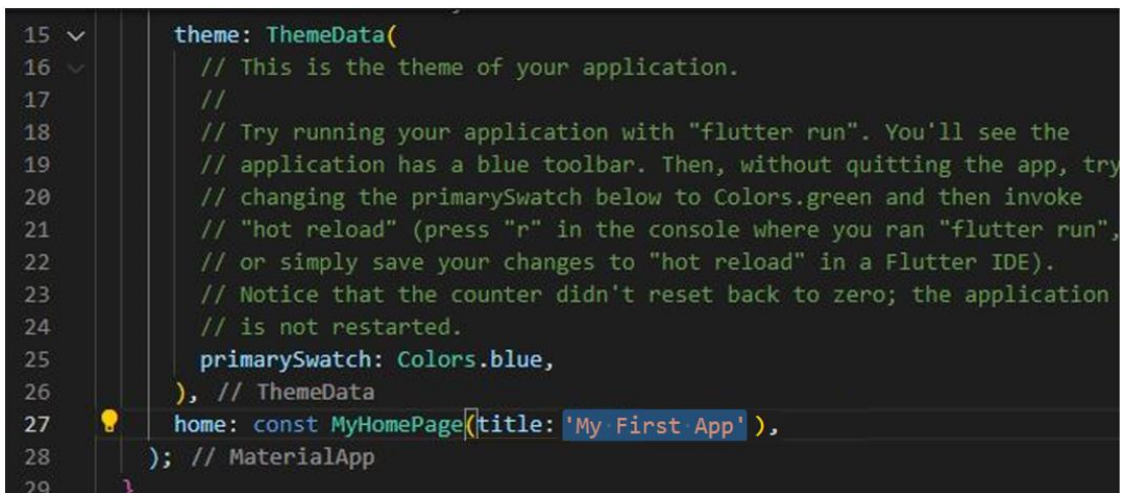
lib > main.dart > MyApp > build
20 // changing the primarySwatch below to Colors.green
21 // "hot reload" (press "r" in the console where you
22 // or simply save your changes to "hot reload" in a
23 // Notice that the counter didn't reset back to zero;
24 // is not restarted.
25 primarySwatch: Colors.blue,
26 ), // ThemeData
27 home: const MyHomePage(title: 'hello my first app'),
28 ); // MaterialApp
29 }
30 }
31 }
32 class MyHomePage extends StatefulWidget {

```

ภาพประกอบ 1.49 เปิด Folder lib

ที่มา : ฅนปักษ์ วรณตรง (2564 : 39)

## 5.3 แก้ไฟล์ main.dart ตรง home:



```

15 theme: ThemeData(
16 // This is the theme of your application.
17 //
18 // Try running your application with "flutter run". You'll see the
19 // application has a blue toolbar. Then, without quitting the app, try
20 // changing the primarySwatch below to Colors.green and then invoke
21 // "hot reload" (press "r" in the console where you ran "flutter run",
22 // or simply save your changes to "hot reload" in a Flutter IDE).
23 // Notice that the counter didn't reset back to zero; the application
24 // is not restarted.
25 primarySwatch: Colors.blue,
26 ), // ThemeData
27 home: const MyHomePage(title: 'My First App'),
28 ); // MaterialApp
29 }

```

ภาพประกอบ 1.50 แก้ไฟล์ main.dart

ที่มา : ฅนปักษ์ วรณตรง (2564 : 40)

## 5.4 Save &gt; แล้วกด r/Shift+R เพื่อ Restart

```

Command Prompt
Please choose one (To quit, press "q/Q"): 2
Launching lib\main.dart on Chrome in debug mode...
Waiting for connection from debug service on Chrome... 30.6s
This app is linked to the debug service: ws://127.0.0.1:51550/JuiM8UGIORo=/ws
Debug service listening on ws://127.0.0.1:51550/JuiM8UGIORo=/ws

Running with sound null safety

To hot restart changes while running, press "r" or "R".
For a more detailed help message, press "h". To quit, press "q".

An Observatory debugger and profiler on Chrome is available at: http://127.0.0.1:51550/
Flutter Web Bootstrap: Programmatic
The Flutter DevTools debugger and profiler on Chrome is available at:
http://127.0.0.1:9100?uri=http://127.0.0.1:51550/JuiM8UGIORo=
Performing hot restart....

```

ภาพประกอบ 1.51 Restart

ที่มา : ณปภัช วรรณตรง (2564 : 41)

```

To hot restart changes while running, press "r" or "R".
For a more detailed help message, press "h". To quit, press "q".

An Observatory debugger and profiler on Chrome is available at: http://127.0.0.1:51550/
Activating Dart DevTools... 18.2s
The Flutter DevTools debugger and profiler on Chrome is available at:
http://127.0.0.1:9101?uri=http://127.0.0.1:63488/1WZ8DEx8BCM=
r

Performing hot restart... 57.6s
Restarted application in 57,568ms.

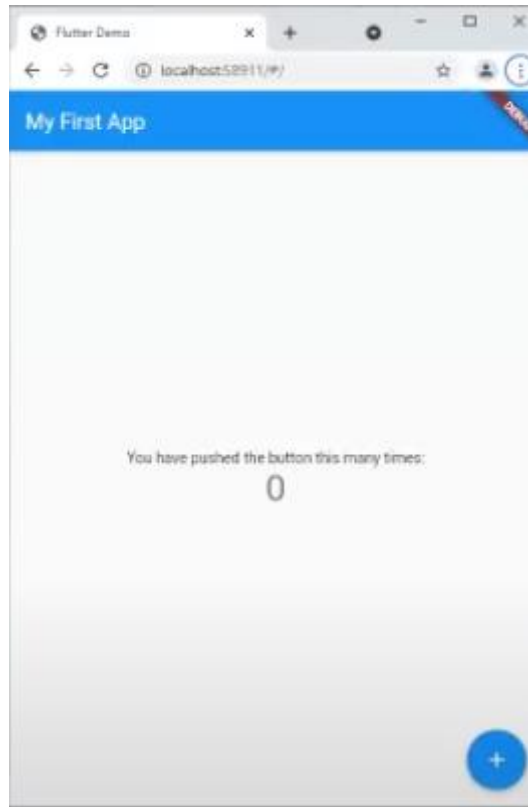
Performing hot restart... 152.7s
Restarted application in 152,709ms.

```

ภาพประกอบ 1.52 Restart (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 41)

5.5 จะได้ผลการสั่งการทำงานดังภาพ จะสังเกตเห็นว่าแถบข้อความด้านบนแอปพลิเคชันเปลี่ยนไปตามที่ได้แก้ไข



ภาพประกอบ 1.53 ผลลัพธ์การแก้ไขโค้ด home:

ที่มา : ฌปภัช วรณตรง (2564 : 42)

หมายเหตุ
หากพิมพ์ r จะทำการแก้ไขแค่เพียงบางส่วนใช้เวลาสั้น shift + r จะทำการแก้ไขทั้งหมดใช้เวลานาน

5.5 กด Ctrl+C ในหน้าต่าง cmd กรณีค้างหรือการสั่งการทำงานนานเกินไป

```

To hot restart changes while running, press "r" or "R".
For a more detailed help message, press "h". To quit, press "q".

An Observatory debugger and profiler on Chrome is available at: http://127.0.0.1:9101?uri=http://127.0.0.1:63488/1WZ8DEx8BCM=
Activating Dart DevTools... 18.2s
The Flutter DevTools debugger and profiler on Chrome is available at:
http://127.0.0.1:9101?uri=http://127.0.0.1:63488/1WZ8DEx8BCM=
r

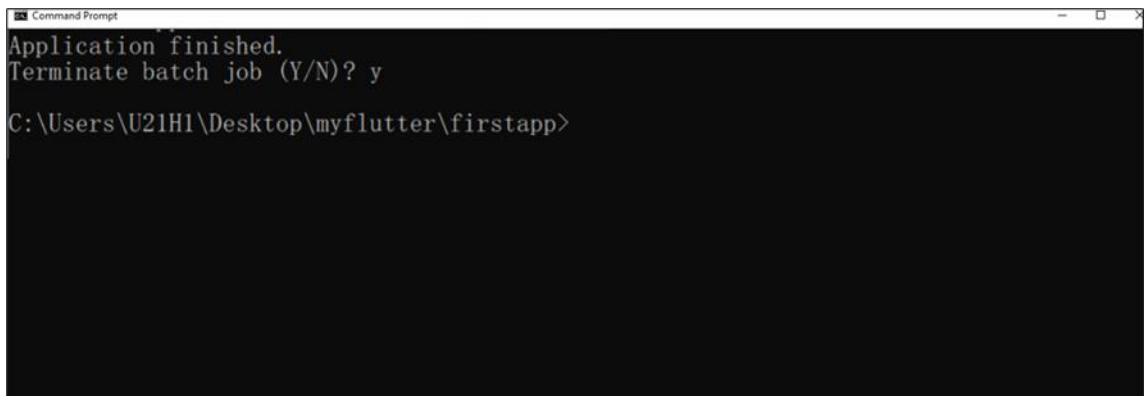
Performing hot restart... 57.6s
Restarted application in 57,568ms.

Performing hot restart... 152.7s
Restarted application in 152,709ms.

```

ภาพประกอบ 1.54 ยกเลิกการทำงานเมื่อเกิดอาการค้าง

ที่มา : ฌปภัช วรรณตรง (2564 : 43)



```

Command Prompt
Application finished.
Terminate batch job (Y/N)? y

C:\Users\U21H1\Desktop\myflutter\firstapp>

```

ภาพประกอบ 1.55 ยกเลิกการทำงานเมื่อเกิดอาการค้าง (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 43)

## บทสรุป

สำหรับเนื้อหาที่กล่าวมาทั้งหมดในบทนี้ จะพูดถึงการพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่ในปัจจุบัน ความเป็นมาของ Flutter ภาษาที่ใช้ในการพัฒนาแอปพลิเคชันใน Flutter วิธีการดาวน์โหลด การติดตั้งเครื่องมือในการพัฒนา และการตั้งค่าการใช้งานเครื่องมือต่าง ๆ การกำหนดค่าในการทำงานของโปรแกรมและการเริ่มต้นสร้างแอปพลิเคชันด้วย Flutter

## เอกสารอ้างอิง

จีราวุธ วารินทร์. (2564). **พัฒนาโมบายส์แอปด้วย Flutter + Dart**. กรุงเทพฯ : สำนักพิมพ์ชิมพลิฟาย บัญชา ปะลีละเตสัง. (2566). **พัฒนาแอปแบบ Multi-Platform ด้วย Flutter โดยใช้ภาษา Dart**.

กรุงเทพฯ : ซีเอ็ดดูเคชั่น.

สำนักงานสถิติแห่งชาติ. (2564). **สรุปผลที่สำคัญสำรวจการมี การใช้เทคโนโลยีสารสนเทศและการสื่อสารในครัวเรือน พ.ศ. 2563**. กรุงเทพฯ : สำนักงานสถิติแห่งชาติ กระทรวงดิจิทัลเพื่อเศรษฐกิจและสังคม.

เอกรินทร์ วทัญญูเลิศสกุล. (2564). **การพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่แบบข้ามแพลตฟอร์ม**. กรุงเทพฯ : สำนักพิมพ์จุฬาลงกรณ์มหาวิทยาลัย.

Baldwin C. Y. and Woodard C. J. (2009). The Architecture of Platforms: A Unified View. **SSRN Electronic Journal**. DOI:10.2139/ssrn.1265155

flutter.dev. (15 กันยายน 2566). Flutter. สืบค้นจาก, <https://flutter.dev/>.

KongRuksiam Official. (2563). **พัฒนาแอปด้วย Flutter สำหรับผู้เริ่มต้น 7 ชั่วโมงเต็ม [FULL COURSE]**. สืบค้นจาก, [https://www.youtube.com/watch?v=3jGj-1-m\\_zA&list=PLltVQYLz1BMBUgyhxZFA31of-EKjazC8G](https://www.youtube.com/watch?v=3jGj-1-m_zA&list=PLltVQYLz1BMBUgyhxZFA31of-EKjazC8G)

## บทที่ 2

### การใช้งานวิดเจ็ต (Widget)

ในการพัฒนาแอปพลิเคชันบนมือถือนั้น บนหน้าจอแอปพลิเคชันมักประกอบด้วยส่วนต่าง ๆ อาทิ เช่น ข้อความ ปุ่ม ภาพ ซึ่งส่วนต่าง ๆ เหล่านี้ Flutter มีแนวคิดในการแบ่งส่วนติดต่อผู้ใช้ (User Interface) ออกเป็นชิ้นส่วนต่าง ๆ แล้วนำมาแสดงผลบนหน้าจอ มีชื่อเรียกชิ้นส่วนต่าง ๆ ในหน้าจอนี้ว่า วิดเจ็ต (Widget) ในบทนี้ผู้เรียนจะได้เรียนรู้ว่า วิดเจ็ตคืออะไร ได้ใช้งานวิดเจ็ตพื้นฐานที่อยู่ในภาษา Dart ที่เป็นภาษาที่ใช้ในการพัฒนาแอปพลิเคชันด้วย Flutter ได้เรียนรู้ประเภทของวิดเจ็ตว่ามีประเภทใดบ้าง และแต่ละประเภทมีลักษณะอย่างไร วิธีการสร้างวิดเจ็ต การเริ่มสร้าง Flutter Project และขั้นตอนการสร้างวิดเจ็ตในโปรเจกต์ วิธีการตกแต่งตัวอักษร การดาวนโหลดรูปแบบตัวอักษรที่ต้องการ ตั้งค่ารูปแบบตัวอักษร กำหนดขนาด สี ลักษณะของตัวอักษร จากเนื้อหาข้างต้นที่กล่าวมาทั้งหมด มีรายละเอียดดังต่อไปนี้

#### ทำความเข้าใจกับวิดเจ็ตคืออะไร

##### 1. วิดเจ็ต (Widget) คืออะไร

วิดเจ็ต (Widget) เป็นส่วนประกอบของส่วนติดต่อผู้ใช้งานผ่านการแสดงผลแบบกราฟิก หรือวิธีการเฉพาะสำหรับอุปกรณ์สมาร์ต (Smart Device) เช่น โทรศัพท์ อุปกรณ์สวมใส่ (Wearable) และอุปกรณ์เคลื่อนที่ต่าง ๆ ซึ่งเป็นการควบคุมการทำงานจากระบบปฏิบัติการหรือแอปพลิเคชัน

วิดเจ็ตประกอบด้วยเครื่องมือและส่วนควบคุมต่าง ๆ ได้แก่ ปุ่มไอคอน เมนูแบบเลื่อน กล้อง ตัวเลือก สวิตช์ เป็นต้น สิ่งต่าง ๆ เหล่านี้ Flutter ได้จัดเตรียมไว้ โดยเรียกกลุ่มเครื่องมือนี้ว่า วิดเจ็ต แค็ตตาล็อก (วิดเจ็ตCatalog) (เอกรินทร์ วทัญญเลิศสกุล, 2563 : 67)

##### 2. ประเภทของ Widget

วิดเจ็ตที่ใช้ในแอปพลิเคชันสามารถแบ่งได้เป็น 3 ประเภท ดังนี้ (จิราวุธ วารินทร์, 2564 : 142)

2.1 วิดเจ็ตที่ Flutter เตรียมไว้ให้ (Standard Widget) เช่น วิดเจ็ตสำหรับแสดงข้อความ วิดเจ็ตสำหรับแสดงปุ่ม วิดเจ็ตสำหรับแสดงรูปภาพ วิดเจ็ตสำหรับจัดตำแหน่งในแนวคอลัมน์ ฯลฯ



ตัวอย่าง วิดเจ็ตที่ทาง Flutter มีให้ไว้สำหรับเลือกใช้งาน

2.1.1 Text คือ วิดเจ็ตสำหรับการแสดงข้อความใน Flutter (จีราวุธ วารินทร์, 2564 : 150)

2.1.2 Raised Button คือ วิดเจ็ตที่เป็นปุ่มกดในรูปแบบอย่างง่ายที่รองรับการโต้ตอบกับผู้ใช้งาน (เอกกรินทร์ วทัญญูเลิศสกุล, 2563 : 107)

2.1.3 Row, Column สำหรับสร้าง Layout (เค้าโครงหน้าแอปพลิเคชัน) Row เป็นการจัดเรียงวิดเจ็ตแบบซ้ายไปขวา ส่วน Column เป็นการจัดเรียงแบบบนลงล่าง (เอกกรินทร์ วทัญญูเลิศสกุล, 2563 : 107)

2.1.4 Stack เป็นวิดเจ็ตสำหรับการจัดวางวิดเจ็ตซ้อนทับกันเป็นชั้น ๆ (ปัญญา ปะสีละเตสัง, 2566 : 222)

2.1.5 Container เป็นวิดเจ็ตสำหรับการบรรจุหรือจัดวางวิดเจ็ตย่อย ๆ เพื่อกำหนดโครงสร้างตามต้องการ (ปัญญา ปะสีละเตสัง, 2566 : 201)

2.2 วิดเจ็ตสร้างขึ้นเอง (Custom Widget) เป็นการนำเอาวิดเจ็ตที่ Flutter เตรียมไว้นำมาจับกลุ่ม แก้ไข หรือสร้างใหม่ให้มีรูปแบบที่ต้องการ

2.3 วิดเจ็ตจากนักพัฒนาคนอื่น (Third Party Widget) หากมีผู้สร้างวิดเจ็ตซึ่งตรงกับความต้องการอยู่แล้ว ไม่ต้องเสียเวลาสร้างขึ้นเองใหม่ทั้งหมด เพราะเพียงแค่ติดตั้งไลบรารี (Liberally) และอิมพอร์ต (Import) วิดเจ็ตเหล่านั้นมาใช้ในโปรเจกต์ ซึ่งจะทำให้ได้วิดเจ็ตในรูปแบบที่ต้องการทันที

### 3. วิดเจ็ตสร้างขึ้นเอง (Custom Widget)

ใน Flutter ถ้าต้องการสร้างวิดเจ็ตขึ้นมาใช้งาน สามารถสร้างได้ 2 แบบ คือ Stateless Widget กับ Stateful Widget

3.1 Stateless Widget เป็นวิดเจ็ตใช้เพื่อแสดงผลเพียงอย่างเดียว และไม่มี การเปลี่ยนแปลงค่าในระหว่างใช้งาน เช่น วิดเจ็ตที่ใช้แสดงรูปโลโก้ของแอปพลิเคชัน หน้าจอหลัก วิดเจ็ตที่ใช้แสดงรูปสินค้า วิดเจ็ตที่ใช้แสดงรายชื่อนักเรียนแต่ละคน เป็นต้น (จีราวุธ วารินทร์, 2564 : 172)

เป็นเสมือนตัวควบคุมการโต้ตอบของวิดเจ็ตต่าง ๆ ที่มีในแต่ละฉาก เป็นวิดเจ็ตที่ไม่มีสถานะ ดังนั้นจึงไม่สามารถโต้ตอบหรือเปลี่ยนแปลงสถานะต่าง ๆ ของวิดเจ็ตที่อยู่ภายในนี้ได้ (เอกกรินทร์ วทัญญูเลิศสกุล, 2563 : 99)

ตัวอย่างโค้ดแบบ Stateless Widget ดังภาพ

```

1 class HelloWorldWidget extends StatelessWidget {
2   @override
3   Widget build(BuildContext context) {
4     return Container();
5   }
6 }

```

ภาพประกอบ 2.1 ตัวอย่างโค้ดแบบ Stateless Widget

ที่มา : mindphp. (2565)

3.2 Stateful Widget เป็นวิดเจ็ตมีลักษณะของการตอบสนองแบบไดนามิก ทำหน้าที่ในการจัดการวิดเจ็ตต่าง ๆ คล้ายกับ Stateless Widget แต่แตกต่างที่ Stateful Widget จะถูกนำมาใช้ในงานที่สามารถตอบสนองกับผู้ใช้งานได้ทั้งในรูปแบบที่ผู้ใช้สื่อสารมายังแอปพลิเคชันและการโต้ตอบหรือการเปลี่ยนแปลงสถานะตามสิ่งเร้าที่ผู้ใช้งานต้องการ Stateful Widget สามารถเก็บข้อมูลและติดตามสถานะของข้อมูล (State) เมื่อข้อมูลภายในวิดเจ็ตเปลี่ยน (เอกรินทร์ วทัญญเลิศสกุล, 2563 : 103)

ตัวอย่างโค้ดแบบ Stateful Widget ดังภาพ

```

1 class HelloWorldWidget extends StatefulWidget {
2   @override
3   _HelloWidgetState createState() => _HelloWidgetState();
4 }
5
6 class _HelloWidgetState extends State<HelloWorldWidget> {
7
8   @override
9   void initState() {
10    // init something.
11    super.initState();
12  }
13
14  @override
15  Widget build(BuildContext context) {
16    return Container();
17  }
18 }

```

ภาพประกอบ 2.2 ตัวอย่างโค้ดแบบ Stateful Widget (ต่อ)

ที่มา : mindphp. (2565)

## วิดเจ็ตที่ใช้บ่อย

Scaffold Widget เป็นวิดเจ็ตที่ใช้กำหนดโครงสร้างพื้นฐานสำหรับแอปพลิเคชัน เมื่อกำหนด Scaffold Widget จะได้เค้าโครงการออกแบบตามมาตรฐานของแอปพลิเคชันบนมือถือทั่วไป เช่น สามารถ กำหนดแถบด้านบน (เรียกว่า AppBar) สามารถกำหนดเนื้อหาไว้ในส่วนของ body สามารถ กำหนด ปุ่มที่เรียกว่า FloatingActionButton และสามารถกำหนดไอเทมสำหรับเปลี่ยนหน้าจอที่ด้านล่าง (bottomNavigationBar) เป็นต้น (จีราวุธ วารินทร์, 2564 : 152)

การใช้งาน Scaffold Widget เป็นฉากหลังของแอปพลิเคชันซึ่งประกอบด้วยส่วนต่าง ๆ (เอกรินทร์ วทัญญูเลิศสกุล, 2563 : 68) ที่น่าสนใจอีกหลายอย่างดังนี้

1.1 AppBar คือ แถบเครื่องมือที่แสดงด้านบนของหน้าจอ ใช้เพื่อแสดงหัวข้อสำคัญหรือเส้นทางการย้ายฉากหรือหน้าจอหรือเมนูการใช้งานต่าง ๆ (เอกรินทร์ วทัญญูเลิศสกุล, 2563 : 68)

1.2 body เป็นส่วนกำหนดเนื้อหาของการแสดงผลภายใน Scaffold สามารถเขียนเป็นแอตทริบิวต์ต่อจาก AppBar ได้โดยคั่นด้วยเครื่องหมายจุลภาค (เอกรินทร์ วทัญญูเลิศสกุล, 2563 : 68)

1.3 FloatingActionButton เป็นปุ่มคำสั่งโดยปุ่มนี้จะลอยอยู่บนหน้าจอ ปุ่มสามารถปรับตำแหน่งอัตโนมัติตามขนาดหน้าจอทั้งหน้า จอแนวตั้งและแนวนอน (จีราวุธ วารินทร์, 2564 : 155) สามารถปรับแต่งการตั้งค่าได้หลายอย่าง เช่น การใส่ Icon การใส่ข้อความ การจัดตำแหน่ง เป็นต้น (อนุชิต ชโลธร, 2565 : 28)

1.4 BottomNavigationBar เป็นแถบทูลบาร์ที่จะปรากฏอยู่ตรงขอบล่างของหน้าจอ สำหรับกำหนดปุ่มเพื่อเชื่อมโยงหรือเลื่อนไปยังหน้าจอต่าง ๆ (บัญชา ปะสีละเตสัง, 2566 : 339)

ตัวอย่างโค้ดแบบ Scaffold Widget ดังภาพ

```

1 void main() {
2   var app = MaterialApp(
3     title: "My App",
4     home: Scaffold(
5       appBar: AppBar(
6         title: Text("My App"),
7       ),
8       body: Text("สวัสดีครับ")
9     ),
10  );
11  runApp(app);
12 }

```

ภาพประกอบ 2.3 ตัวอย่างโค้ดแบบ Scaffold Widget (ต่อ)

ที่มา : mindphp. (2565)

การใช้งาน Center Widget เป็นวิดเจ็ตที่ทำหน้าที่ครอบคลุม วิดเจ็ตอื่น ๆ ให้อยู่ตรงกลางหน้าจอ (อนุชิต ชโลธร, 2565: 32)

ตัวอย่างโค้ดแบบ Center Widget ดังภาพ

```

1 void main() {
2   var app = MaterialApp(
3     title: "My App",
4     home: Scaffold(
5       appBar: AppBar(
6         title: Text("My App"),
7       ),
8       body: Center(child: Text("สวัสดีครับ"))
9     ),
10  );
11  runApp(app);
12 }

```

ภาพประกอบ 2.4 ตัวอย่างโค้ดแบบ Center Widget

ที่มา : mindphp. (2565)

การใช้งาน Column Widget เป็นวิดเจ็ตที่ใช้ในการจัดตำแหน่งการแสดงผลแบบคอลัมน์ เป็นวิดเจ็ตพื้นฐานที่เป็นประโยชน์ในการจัดโครงสร้างของหน้าจอแอปพลิเคชัน เพื่อแยกวัตถุให้เป็นสัดส่วน (เอกรินทร์ วทัญญูเลิศสกุล, 2563 : 76)

Column Widget เป็นวิดเจ็ตที่รับเอา วิดเจ็ตอื่น ๆ มาจัดเรียงให้อยู่ในแนวตั้งหรือแนวตั้ง ข้อดีของ Column Widget คือ สามารถนำอื่นมาใส่วิดเจ็ตได้ในคราวเดียว ซึ่งจะจัดให้อยู่ในรูปแบบแนวตั้ง ตัวอย่างโค้ดแบบ Column Widget ดังภาพ

```

1 import 'package:flutter/material.dart';
2
3 void main() {
4   var app = MaterialApp(
5     title: "My App",
6     home: Scaffold(
7       appBar: AppBar(
8         title: Text("My App"),
9       ),
10      body: Center(child: Column(
11        children: [
12          Text("สวัสดีครับ"),
13          Text("Thinnakorn"),
14        ],
15      ))
16    ),
17  );
18  runApp(app);
19 }

```

ภาพประกอบ 2.5 ตัวอย่างโค้ดแบบ Column Widget (ต่อ)

ที่มา : mindphp. (2565)

การใช้งาน Row Widget เป็นวิดเจ็ตที่ช่วยทำให้วิดเจ็ตต่าง ๆ สามารถจัดเรียงในทางแนวนอนได้ (เอกรินทร์ วทัญญูเลิศสกุล, 2563 : 79)

## ตัวอย่างโค้ดแบบ Row วิดเจ็ตตั้งภาพ

```

1 import 'package:flutter/material.dart';
2
3 void main() {
4   var app = MaterialApp(
5     title: "My App",
6     home: Scaffold(
7       appBar: AppBar(
8         title: Text("My App"),
9       ),
10      body: Center(child: Row(
11        children: [
12          Text("สวัสดีครับ"),
13          Text("Thinnakorn")
14        ],
15      ))
16    ),
17  );
18  runApp(app);
19 }

```

## ภาพประกอบ 2.6 ตัวอย่างโค้ดแบบ Row Widget

ที่มา : mindphp. (2565)

การใช้งาน Container Widget เป็นวิดเจ็ตแบบหนึ่งซึ่งทำหน้าที่จัดการความซับซ้อนของการแสดงผลภายในแอปพลิเคชันโดยสามารถกำหนดให้มีวิดเจ็ตลูกอยู่ภายในวิดเจ็ตหลักได้ ทำให้สามารถควบคุมการใช้งานเนื้อหาบนหน้าจอของแอปพลิเคชันได้อย่างทั่วถึง (เอกรินทร์ วทัญญูเลิศสกุล, 2563 : 70)

## ตัวอย่างโค้ดแบบ Container Widget ดังภาพ

```

1 import 'package:flutter/material.dart';
2
3 void main() {
4   var app = MaterialApp(
5     title: "My App",
6     home: Scaffold(
7       appBar: AppBar(
8         title: Text("My App"),
9       ),
10      body: Column(
11        children: [
12
13          Container(
14            margin: EdgeInsets.only(
15              top: 10, //กำหนดระยะห่างจากด้านบน
16              bottom: 10, //กำหนดระยะห่างจากด้านล่าง
17              left: 10, //กำหนดระยะห่างจากด้านซ้าย
18              right: 10, //กำหนดระยะห่างจากด้านขวา
19            ),
20            decoration: BoxDecoration(color: Colors.redAccent),
21            height: 100, //กำหนดความสูง
22            width: 200, //กำหนดความกว้าง
23          ),
24        ],
25      ),
26    );
27   runApp(app);
28 }

```

## ภาพประกอบ 2.7 ตัวอย่างโค้ดแบบ Container Widget

ที่มา : mindphp. (2565)

รูปแบบการแสดงผลแอปพลิเคชันที่สร้างจาก Flutter สามารถแบ่งออกได้เป็น 2 แนวทาง ดังนี้

1. แบบที่ 1 Material Design เป็นการออกแบบหน้าต่างในสไตล์ของ Google
2. แบบที่ 2 Cupertino Design เป็นการออกแบบหน้าต่างแอปพลิเคชันในสไตล์ของ iOS

(จิราวุธ วารินทร์, 2564 : 147)

หมายเหตุ
สามารถศึกษาเพิ่มเติมเกี่ยวกับวิดเจ็ตได้ที่เว็บไซต์ <a href="https://flutter.dev/widgets">https://flutter.dev/widgets</a>

## เริ่มขั้นตอนการสร้าง Project

1. สร้าง Folder ที่จะเอาไว้เก็บไฟล์งาน > เปิดหน้าต่าง cmd > cd Folder ที่ได้ทำการสร้างเอาไว้หรือเข้าไปยังตำแหน่งที่ต้องการเก็บ Folder งาน จากนั้นให้ทำการพิมพ์ cmd ใช้ช่องตำแหน่งของ Folder (Path)



ภาพประกอบ 2.8 เข้าหน้าต่าง cmd ผ่าน Path ของ Folder

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 3)



ภาพประกอบ 2.9 เข้าหน้าต่าง cmd ผ่าน Path ของ Folder (ต่อ)

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 3)

```
Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

C:\src\flutter>|
```

ภาพประกอบ 2.10 เข้าหน้าต่าง cmd ผ่าน Path ของ Folder (ต่อ)

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 3)



2. เมื่อเปิดหน้าต่าง cmd แล้ว ให้ทำการพิมพ์คำสั่งสำหรับตรวจสอบข้อมูลและเวอร์ชันของ Flutter โดยการพิมพ์ flutter doctor -v ในหน้าต่าง cmd ที่ได้ทำการเปิดขึ้นมา

```
Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

C:\src\flutter>flutter doctor -v|
```

ภาพประกอบ 2.11 พิมพ์คำสั่งสำหรับสร้างไฟล์ Flutter > flutter doctor -v  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 4)

```
C:\Users\U21H1\Desktop\flutter>flutter doctor -v

A new version of Flutter is available!
To update to the latest version, run "flutter upgrade".

[✓] Flutter (Channel stable, 2.5.0, on Microsoft Windows [Version 10.0.19043.1348], locale th-TH)
    • Flutter version 2.5.0 at C:\flutter
    • Upstream repository https://github.com/flutter/flutter.git
    • Framework revision 4cc385b4b8 (3 months ago), 2021-09-07 23:01:49 -0700
    • Engine revision f0826da7ef
    • Dart version 2.14.0

[✓] Android toolchain - develop for Android devices (Android SDK version 31.0.0)
    • Android SDK at C:\Users\U21H1\AppData\Local\Android\Sdk
    • Platform android-31, build-tools 31.0.0
    • ANDROID_HOME = C:\Users\U21H1\AppData\Local\Android\Sdk
    • Java binary at: C:\Program Files\Android\Android Studio\jre\bin\java
    • Java version OpenJDK Runtime Environment (build 11.0.8+10-b944.6842174)
    • All Android licenses accepted.

[✓] Chrome - develop for the web
    • Chrome at C:\Program Files\Google\Chrome\Application\chrome.exe

[✓] Android Studio (version 2020.3)
    • Android Studio at C:\Program Files\Android\Android Studio
    • Flutter plugin can be installed from:
      https://plugins.jetbrains.com/plugin/9212-flutter
    • Dart plugin can be installed from:
      https://plugins.jetbrains.com/plugin/6351-dart
    • Java version OpenJDK Runtime Environment (build 11.0.8+10-b944.6842174)
```

ภาพประกอบ 2.12 พิมพ์คำสั่งสำหรับสร้างไฟล์ Flutter > flutter doctor -v  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 4)

3. เมื่อรู้ข้อมูลและเวอร์ชันของ Flutter แล้ว ให้ทำการพิมพ์คำสั่งสำหรับสร้าง Flutter Project โดยใช้คำสั่ง create ตามด้วยชื่อ Project ที่ต้องการสร้าง

```
C:\src\flutter>flutter create secondapp|
```

ภาพประกอบ 2.13 พิมพ์คำสั่งสำหรับสร้างไฟล์ Project

ที่มา : ฅนปักษ์ วรณตรง (2564 : 5)

```
Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

C:\src\flutter>|
```

ภาพประกอบ 2.14 พิมพ์คำสั่งสำหรับสร้างไฟล์ Project (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 5)

เมื่อการทำงานของคำสั่งเสร็จสิ้นแล้ว Folder งานจะปรากฏขึ้นใน Folder ที่ได้ทำการเลือกไว้

Name	Type	Compressed size	Password p..	Size	Ratio	Date modified
.dart_tool	File folder					
idea	File folder					
android	File folder					
build	File folder					
fonts	File folder					
ios	File folder					
lib	File folder					
test	File folder					
web	File folder					
.gitignore	Git Ignore Source File	1 KB	No	1 KB	46%	11/24/2021 22:29
.metadata	METADATA File	1 KB	No	1 KB	33%	11/24/2021 22:29
.packages	PACKAGES File	1 KB	No	3 KB	76%	11/25/2021 0:39
analysis_options	Yaml Source File	1 KB	No	2 KB	53%	11/24/2021 22:29
pubspec	Yaml Source File	2 KB	No	4 KB	60%	11/25/2021 0:39
pubspec.lock	LOCK File	1 KB	No	4 KB	86%	11/25/2021 0:39
README	Markdown Source File	1 KB	No	1 KB	44%	11/24/2021 22:29
secondapp.iml	IML File	1 KB	No	1 KB	64%	11/24/2021 22:29

ภาพประกอบ 2.15 ผลการสร้างไฟล์งาน Project

ที่มา : ฅนปักษ์ วรณตรง (2564 : 5)

หมายเหตุ
ชื่อ Project ไม่สามารถเว้นเคาะว่างได้ต้องเขียนติดกัน

4. หลังจากสร้างไฟล์ Project เสร็จสิ้นแล้วให้ทำการเข้าไปในไฟล์งานโดยใช้คำสั่ง cd ตามด้วยชื่อ Project จากนั้นให้ทำการสั่งเปิดการทำงานของ Project > flutter run

4.1 ให้ทำการเข้าไปยัง Folder งานที่ได้ทำการสร้างขึ้นมา โดยใช้คำสั่ง cd secondapp

```
C:\Users\nan_s>cd secondapp
```

ภาพประกอบ 2.16 เข้าไปยัง Folder งาน

ที่มา : ฅนปักษ์ วรณตรง (2564 : 6)

```
C:\Users\nan_s>cd secondapp
C:\Users\nan_s\secondapp>
```

ภาพประกอบ 2.17 เข้าไปยัง Folder งาน (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 6)

4.2 จากนั้นให้ทำการพิมพ์คำสั่ง flutter run ลงไปเพื่อสั่งการทำงานของ flutter

```
C:\Users\nan_s>cd secondapp
C:\Users\nan_s\secondapp>flutter run|
```

ภาพประกอบ 2.18 สั่งการทำงานด้วย flutter run

ที่มา : ฅนปักษ์ วรณตรง (2564 : 7)

4.3 เมื่อสั่งการทำงานครั้งแรกหลังจากการติดตั้ง Flutter จะแจ้งให้ผู้ใช้งานเลือกเว็บเบราว์เซอร์ตั้งต้นสำหรับการแสดงหน้าเว็บ

```
C:\Users\nan_s\secondapp>flutter run
Connected devices:
Windows (desktop) • windows • windows-x64 • Microsoft Windows [Version 10.0.22621.2283]
Chrome (web) • chrome • web-javascript • Google Chrome 116.0.5845.188
Edge (web) • edge • web-javascript • Microsoft Edge 117.0.2045.31
[1]: Windows (windows)
[2]: Chrome (chrome)
[3]: Edge (edge)
Please choose one (or "q" to quit):
```

ภาพประกอบ 2.19 สั่งการทำงานด้วย flutter run (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 7)

5. เลือก Devices สำหรับการแสดงหน้าเว็บแอปฯ โดยจะมีให้เลือกอยู่ 3 ประเภท คือ

1) Windows 2) Google Chrome 3) Microsoft Edge

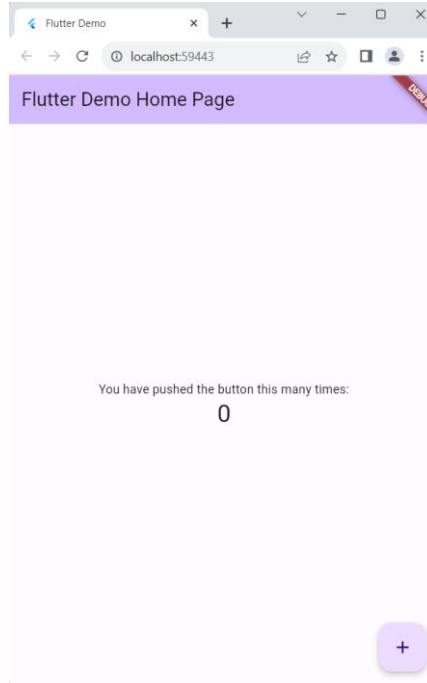
ให้ทำการกดหมายเลข Device ที่ต้องการ จากนั้นให้กดปุ่ม Enter

```
C:\Users\nan_s\secondapp>flutter run
Connected devices:
Windows (desktop) • windows • windows-x64 • Microsoft Windows [Version 10.0.22621.2283]
Chrome (web) • chrome • web-javascript • Google Chrome 116.0.5845.188
Edge (web) • edge • web-javascript • Microsoft Edge 117.0.2045.31
[1]: Windows (windows)
[2]: Chrome (chrome)
[3]: Edge (edge)
Please choose one (or "q" to quit): 2
Launching lib\main.dart on Chrome in debug mode...
Waiting for connection from debug service on Chrome...
```

ภาพประกอบ 2.20 เลือก Devices

ที่มา : ณปภัช วรรณตรง (2564 : 8)

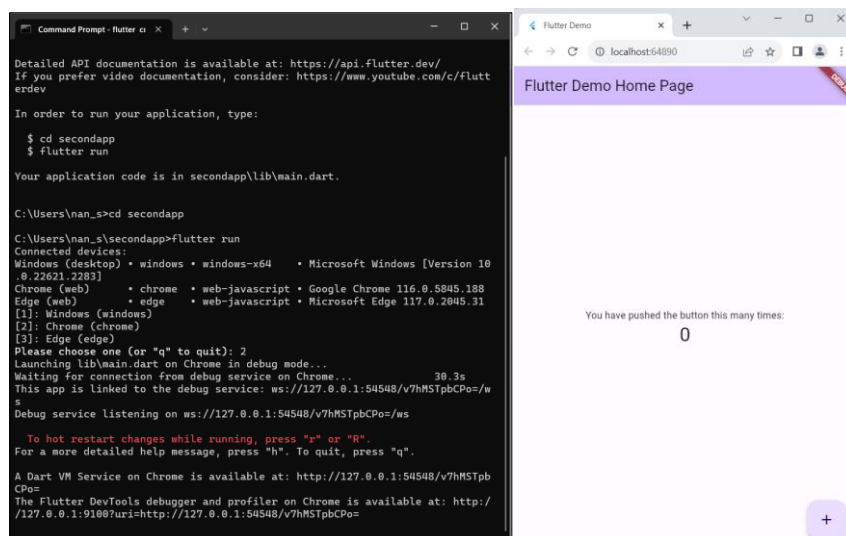
เมื่อคำสั่งทำงานเสร็จสิ้นจะปรากฏหน้าต่าง Device หรือเว็บเบราว์เซอร์ที่ได้ทำการเลือก



ภาพประกอบ 2.21 เลือก Devices (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 8)

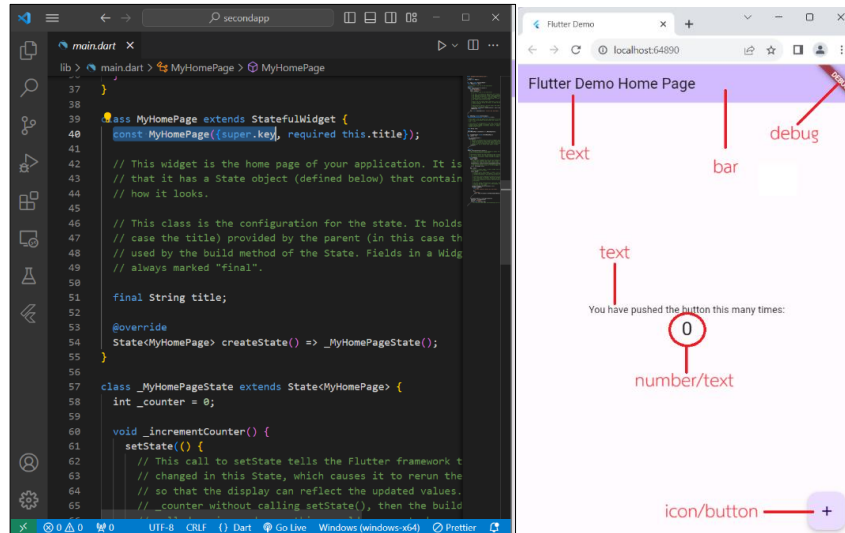
6. หลังจากเลือก Devices เสร็จสิ้น จะได้ผลลัพธ์ดังภาพ



ภาพประกอบ 2.22 ผลลัพธ์การสร้างไฟล์ Flutter Project

ที่มา : ฌปภัช วรณตรง (2564 : 9)

7. ให้ทำการเปิดโค้ดผ่าน VSCode เพื่อดูแต่ละส่วนประกอบบนหน้าแอปพลิเคชัน

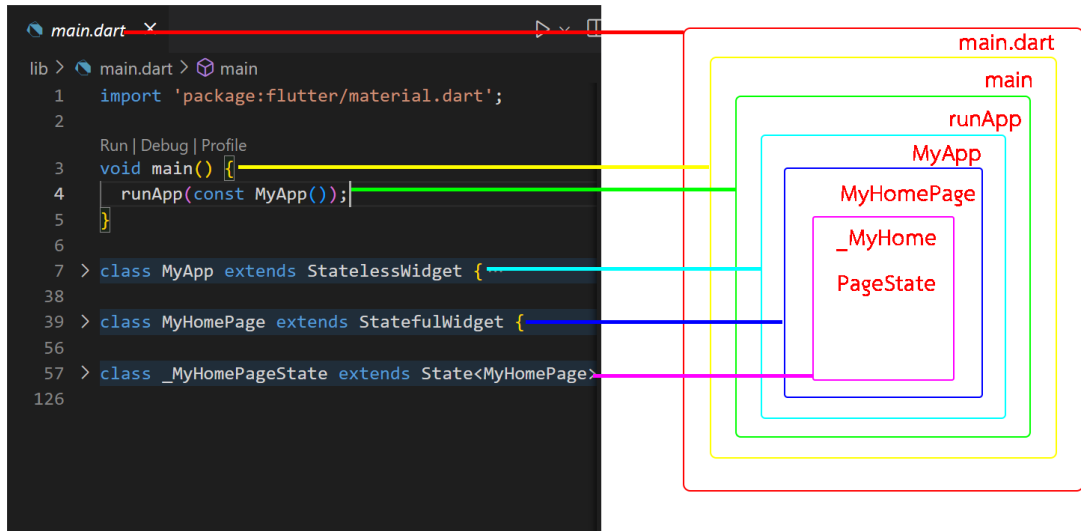


ภาพประกอบ 2.23 ดูแต่ละส่วนประกอบบนหน้าแอปพลิเคชัน

ที่มา : ฌปภัช วรรณตรง (2564 : 10)

หมายเหตุ
สามารถขยายขนาด Font ด้วยการกด ctrl และ + เพื่อทำการขยายขนาดหรือ ctrl และ - เพื่อทำการลดขนาดตัวอักษร

8. จากนั้นให้ลองทำความเข้าใจองค์ประกอบต่าง ๆ ของโครงสร้างแอปพลิเคชันที่พัฒนาโดยใช้ Flutter



ภาพประกอบ 2.24 ทำความเข้าใจองค์ประกอบต่าง ๆ

ที่มา : ฌปภัช วรณตรง (2564 : 11)

จากภาพ ภายในไฟล์ main.dart มีฟังก์ชัน main() เป็นฟังก์ชันหลักที่จะถูกเรียกใช้งานเป็นลำดับแรก ภายในฟังก์ชัน main จะมีคำสั่งเพื่อนำเอาวิดเจ็ตมาแสดงบนแอปพลิเคชัน

runApp เป็นฟังก์ชันที่ใช้สำหรับสร้างวิดเจ็ตหลัก โดยจะเรียกเมธอด build() ที่อยู่ในวิดเจ็ตหลักนั้น เพื่อนำผลลัพธ์มาแสดงบนหน้าจอต่อไป (จิราวุธ วารินทร์, 2564 : 144-145)

9. หลังจากที่ได้ทำความเข้าใจโครงสร้างของแอปพลิเคชันแล้ว ให้ลองทำการแก้ไขโค้ดในส่วน  
ของ Title โดยลบโค้ดที่ไม่ได้ใช้ออกเพื่อป้องกันการสับสนหรือการทำงานผิดพลาดของโค้ด

### 9.1 ลบโค้ด theme: ThemeData(...) ออก

```

11 @override
12 Widget build(BuildContext context) {
13   return MaterialApp(
14     title: 'Flutter Demo',
15     theme: ThemeData(
16       // This is the theme of your application.
17       // TRY THIS: Try running your application with "flutter run". You'll see
18       // the application has a blue toolbar. Then, without quitting the app,
19       // try changing the seedColor in the colorScheme below to Colors.green
20       // and then invoke "hot reload" (save your changes or press the "hot
21       // reload" button in a Flutter-supported IDE, or press "r" if you used
22       // the command line to start the app).
23       // Notice that the counter didn't reset back to zero; the application
24       // state is not lost during the reload. To reset the state, use hot
25       // restart instead.
26       // This works for code too, not just values: Most code changes can be
27       // tested with just a hot reload.
28       colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
29       useMaterial3: true,
30     ), // ThemeData
31     home: const MyHomePage(title: 'Flutter Demo Home Page'),
32   ); // MaterialApp
33 }
34 }
35 }
36 }
37 }

```

ภาพประกอบ 2.25 ลบโค้ดที่ไม่ได้ใช้ออก

ที่มา : ฌปภัช วรณตรง (2564 : 15)

9.2 เมื่อทำการลบโค้ดที่ไม่ได้ใช้ออกแล้ว ให้ทำการแก้ไขโค้ด title: “ ” เป็นข้อความที่  
ต้องการ

```

11 @override
12 Widget build(BuildContext context) {
13   return MaterialApp(
14     title: 'Flutter Demo',
15     home: const MyHomePage(title: 'Flutter Demo Home Page'),
16   ); // MaterialApp
17 }
18 }
19 }

```

ภาพประกอบ 2.26 แก้ไขโค้ดส่วน Title

ที่มา : ฌปภัช วรณตรง (2564 : 15)



```

6
7 class MyApp extends StatelessWidget {
8   const MyApp({Key? key}) : super(key: key);
9
10  // This widget is the root of your application.
11  @override
12  Widget build(BuildContext context) {
13    return MaterialApp(
14      title: 'Flutter Demo',
15
16      home: const MyHomePage(title: 'Hello Flutter'),
17    ); // MaterialApp
18  }
19 }
20

```

ภาพประกอบ 2.27 แก้ไขโค้ดส่วน Title (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 15)

10. ทำการแก้ไขจำนวนตัวเลขในโค้ด `int_counter` ของโครงสร้างแอปพลิเคชัน

```

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 10;

  void _incrementCounter() {
    setState(() {
      // This call to setState tells the Flutter framework that something has
      // changed in this State, which causes it to rerun the build method below
      // so that the display can reflect the updated values. If we changed
      // _counter without calling setState(), then the build method would not be
      // called again, and so nothing would appear to happen.
      _counter++;
    });
  }
}

```

ภาพประกอบ 2.28 แก้ไขจำนวนตัวเลขโค้ด `int_counter`

ที่มา : ณปภัช วรรณตรง (2564 : 17)

เมื่อทำการแก้ไขโค้ดเบื้องต้นแล้วให้ลองทำการ Restart Flutter โดยการกดปุ่ม r หรือ Ctrl+r จากนั้นให้กด y (ตกลง) แล้วกดปุ่ม Enter

```
To hot restart changes while running, press "r" or "R".
For a more detailed help message, press "h". To quit, press "q".

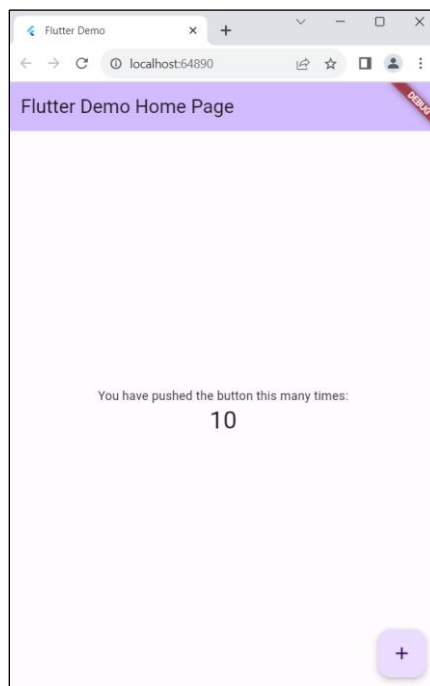
A Dart VM Service on Chrome is available at:
http://127.0.0.1:55317/8ZT0Jr1GauI=
The Flutter DevTools debugger and profiler on Chrome is available at:
http://127.0.0.1:9103?uri=http://127.0.0.1:55317/8ZT0Jr1GauI=
r

Performing hot restart... 181ms
Restarted application in 183ms.
```

ภาพประกอบ 2.29 Restart Flutter

ที่มา : ฌปภัช วรณตรง (2564 : 17)

จากนั้นให้ทำการสั่ง flutter run ใหม่อีกครั้ง จะพบผลลัพธ์ตัวนับมีค่าเริ่มต้นที่ 10 ตามที่ได้แก้ไขดั่งภาพ



ภาพประกอบ 2.30 ผลลัพธ์การแก้ไขโค้ดเบื้องต้น

ที่มา : ฌปภัช วรณตรง (2564 : 18)

## 11. ทำการเอาส่วน Scaffold ออกจากแอปพลิเคชัน

### 11.1 ทำการครอบเลือกส่วน Scaffold(..) ทั้งหมด จากนั้นลบโค้ดที่เลือกไว้ออก

```

79     return Scaffold(
80       appBar: AppBar(
81         // TRY THIS: Try changing the color here to a specific color (to...
82         // Here we take the value from the MyHomePage object that was created by
83         // the App.build method, and use it to set our appBar title.
84         backgroundColor: Theme.of(context).colorScheme.inversePrimary,
85         title: Text(widget.title),
86       ), // AppBar
87       body: Center(
88         // Center is a layout widget. It takes a single child and positions it
89         // in the middle of the parent.
90         child: Column(
91           // Column is also a layout widget. It takes a list of children and
92           // arranges them vertically. By default, it sizes itself to fit its
93           // children horizontally, and tries to be as tall as its parent.
94           //
95           // Column has various properties to control how it sizes itself and
96           // how it positions its children. Here we use mainAxisAlignment to
97           // center the children vertically; the main axis here is the vertical
98           // axis because Columns are vertical (the cross axis would be
99           // horizontal).
100          //
101          // TRY THIS: Invoke "debug painting" (choose the "Toggle Debug Paint"
102          // action in the IDE, or press "p" in the console), to see the
103          // wireframe for each widget.
104          mainAxisAlignment: MainAxisAlignment.center,
105          children: <Widget>[
106            const Text(
107              'You have pushed the button this many times:',
108            ),
109          ],

```

ภาพประกอบ 2.31 ครอบเลือกส่วน Scaffold(..)

ที่มา : ฌปภัช วรณตรง (2564 : 20)

```

105 ..... // wireframe for each widget.
106 ..... mainAxisAlignment: MainAxisAlignment.center,
107 ..... children: <Widget>[
108 .....   const Text(
109 .....     'You have pushed the button this many times:',
110 .....   ), // Text
111 .....   Text(
112 .....     '$_counter',
113 .....     style: Theme.of(context).textTheme.headlineMedium,
114 .....   ), // Text
115 ..... ], // <Widget>[]
116 ..... ), // Column
117 ..... ), // Center
118 ..... floatingActionButton: FloatingActionButton(
119 .....   onPressed: _incrementCounter,
120 .....   tooltip: 'Increment',
121 .....   child: const Icon(Icons.add),
122 ..... ), // This trailing comma makes auto-formatting nicer for build methods. //
123 ..... ); // Scaffold

```

ภาพประกอบ 2.32 ครอบเลือกส่วน Scaffold(..) (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 20)

```

57 class _MyHomePageState extends State<MyHomePage> {
58   int _counter = 10;
59
60   void _incrementCounter() {
61     setState(() {
62       // This call to setState tells the Flutter framework that something has
63       // changed in this State, which causes it to rerun the build method below
64       // so that the display can reflect the updated values. If we changed
65       // _counter without calling setState(), then the build method would not be
66       // called again, and so nothing would appear to happen.
67       _counter++;
68     });
69   }
70
71   @override
72   Widget build(BuildContext context) {
73     // This method is rerun every time setState is called, for instance as done
74     // by the _incrementCounter method above.
75     //
76     // The Flutter framework has been optimized to make rerunning build methods
77     // fast, so that you can just rebuild anything that needs updating rather
78     // than having to individually change instances of widgets.
79     return
80   }
81 }

```

ภาพประกอบ 2.33 ลบส่วน Scaffold(...) ออก

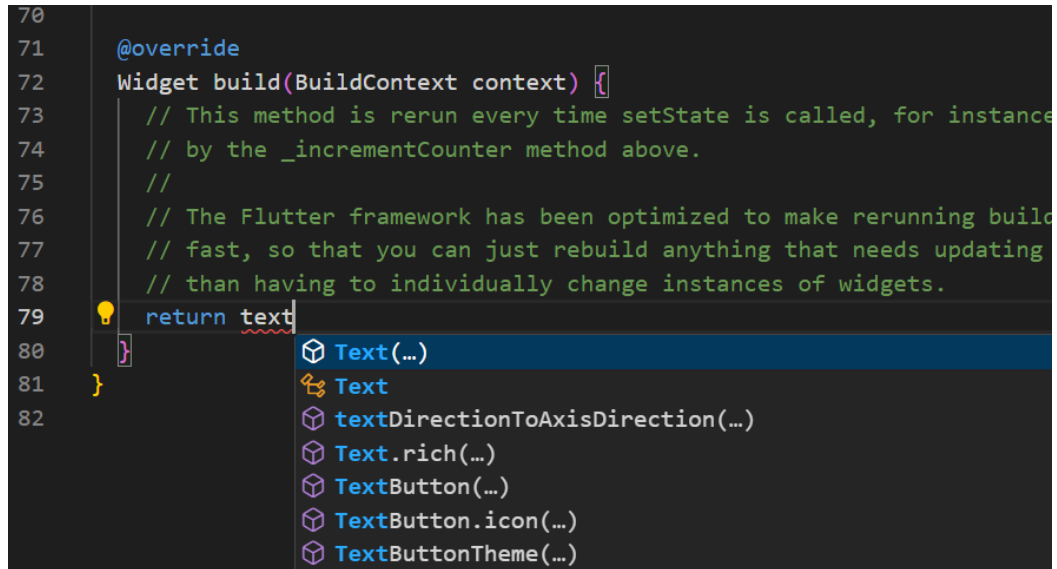
ที่มา : ฌปภัช วรณตรง (2564 : 20)

11.2 หลังจากที่ลบ Scaffold ออกแล้ว ให้ทำการเพิ่มโค้ด Text(...) เข้าไปแทนที่ Scaffold(...) โดยการพิมพ์ Te จากนั้นจะปรากฏหน้าต่างชุดคำสั่งขึ้นมาให้ทำการเลือก Text(...) ด้วยการคลิกที่คำสั่ง หรือ Enter

```

70
71  @override
72  Widget build(BuildContext context) {
73      // This method is rerun every time setState is called, for instance
74      // by the _incrementCounter method above.
75      //
76      // The Flutter framework has been optimized to make rerunning build
77      // fast, so that you can just rebuild anything that needs updating
78      // than having to individually change instances of widgets.
79      return text
80  }
81
82

```



ภาพประกอบ 2.34 เพิ่มโค้ด Text(...) เข้าไป

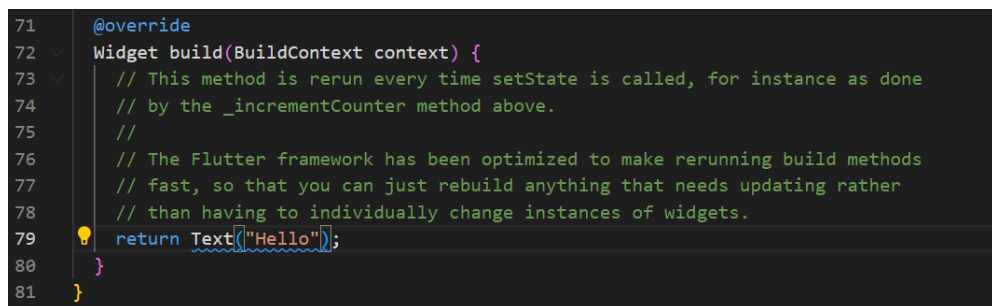
ที่มา : ฌปภัช วรรณตรง (2564 : 20)

11.3 จากนั้นเมื่อเพิ่มโค้ด Text(...) แล้ว ภายใน () ให้ทำการเพิ่มข้อความโดยใช้เครื่องหมาย “” เพื่อเก็บข้อความที่ได้ทำการเพิ่มเข้าไป

```

71  @override
72  Widget build(BuildContext context) {
73      // This method is rerun every time setState is called, for instance as done
74      // by the _incrementCounter method above.
75      //
76      // The Flutter framework has been optimized to make rerunning build methods
77      // fast, so that you can just rebuild anything that needs updating rather
78      // than having to individually change instances of widgets.
79      return Text("Hello");
80  }
81

```



ภาพประกอบ 2.35 เพิ่มข้อความเข้าไปใน Text(...)

ที่มา : ฌปภัช วรรณตรง (2564 : 20)

```

64  @override
65  Widget build(BuildContext context) {
66      // This method is rerun every time setState is called, for instance
67      // by the _incrementCounter method above.
68      //
69      // The Flutter framework has been optimized to make rerunning build
70      // fast, so that you can just rebuild anything that needs updating
71      // than having to individually change instances of widgets.
72      return Text("hello");
73  }
74  }

```

ภาพประกอบ 2.36 เพิ่มข้อความเข้าไปใน Text(...) (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 20)

#### หมายเหตุ

เมื่อทำการทดลองแก้ไขโค้ดดูแล้ว หากต้องการนำโค้ดเก่ากลับมาให้ทำการกดปุ่ม Ctrl+z เพื่อเป็นการย้อนคืน (Undo) การแก้ไขโค้ดได้ โดยให้ย้อนกลับไปทีก่อนจะลบ Scaffold(...) ออก จากนั้นจึงกดบันทึกโดยการกดปุ่ม Ctrl+s (Save)

## จัด Layout หน้าแอปพลิเคชันด้วย Scaffold Widget

Scaffold คือ วิวเจ็ตหน้าตาสำเร็จรูปสำหรับจัดการ Layout หรือโครงสร้างของหน้าแอปพลิเคชัน (มีการคำนวณระยะห่างของแอปพลิเคชันกับหน้าจอ Emulator ให้อัตโนมัติ) เป็นวิวเจ็ตสำหรับชั้นโครงหลักใน Material Design ซึ่งวิวเจ็ตนี้ทำหน้าที่เป็นกำหนดโครงสร้างและธีม ดังนั้นแต่ละหน้าจอก็จะควรขึ้นต้นด้วย Scaffold หากไม่ได้ใช้ Scaffold จะพบว่าหน้าจอจะไม่มี

การเรียกใช้ธีมและหน้าจอจะแสดงผลเป็นสีดำ (อนุชิต ชโลธร, 2565 : 6)

1.1 ให้ทำการลบ AppBar ออกจากภายใน Scaffold(...) โดยการครอบส่วนของ AppBar(...) จากนั้นให้ทำการลบออก

```

79  return Scaffold(
80      appBar: AppBar(
81          // TRY THIS: Try changing the color here to a specific color (to
82          // Colors.amber, perhaps?) and trigger a hot reload to see the AppBar
83          // change color while the other colors stay the same.
84          backgroundColor: Theme.of(context).colorScheme.inversePrimary,
85          // Here we take the value from the MyHomePage object that was created by
86          // the App.build method, and use it to set our appBar title.
87          title: Text(widget.title),
88      ), // AppBar

```

ภาพประกอบ 2.37 ลบส่วน AppBar ออก

ที่มา : ฌปภัช วรณตรง (2564 : 24)

```

71 @override
72 Widget build(BuildContext context) {
73   // This method is rerun every time setState is called, for instance as done
74   // by the _incrementCounter method above.
75   //
76   // The Flutter framework has been optimized to make rerunning build methods
77   // fast, so that you can just rebuild anything that needs updating rather
78   // than having to individually change instances of widgets.
79   return Scaffold(
80
81     body: Center(
82       // Center is a layout widget. It takes a single child and positions it
83       // in the middle of the parent.
84       child: Column(
85         // Column is also a layout widget. It takes a list of children and
86         // arranges them vertically. By default, it sizes itself to fit its
87         // children horizontally, and tries to be as tall as its parent.
88         //
89         // Column has various properties to control how it sizes itself and
90         // how it positions its children. Here we use mainAxisAlignment to

```

ภาพประกอบ 2.38 ลบส่วน AppBar ออก (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 24)

```

body: Center(
  // Center is a layout widget. It takes a single child and positions it
  // in the middle of the parent.
  child: Column(
    // Column is also a layout widget. It takes a list of children and
    // arranges them vertically. By default, it sizes itself to fit its
    // children horizontally, and tries to be as tall as its parent.
    //
    // Invoke "debug painting" (press "p" in the console, choose the
    // "Toggle Debug Paint" action from the Flutter Inspector in Android
    // Studio, or the "Toggle Debug Paint" command in Visual Studio Code)
    // to see the wireframe for each widget.
    //
    // Column has various properties to control how it sizes itself and
    // how it positions its children. Here we use mainAxisAlignment to
    // center the children vertically; the main axis here is the vertical
    // axis because Columns are vertical (the cross axis would be
    // horizontal).

```

ภาพประกอบ 2.39 ส่วนที่เหลือจากการลบ AppBar ออก

ที่มา : ฌปภัช วรรณตรง (2564 : 24)

1.2 เมื่อลองลบส่วน AppBar ออกแล้ว ให้ทำการเอา center หลัง tag body ออกและแก้ไขโค้ด

1.2.1 ให้ทำการครอบส่วน Center(...) เพื่อทำการลบออก

```

81  body: Center(
82  // Center is a layout widget. It takes a single child and positions it
83  // in the middle of the parent.
84  child: Column(
85  // Column is also a layout widget. It takes a list of children and
86  // arranges them vertically. By default, it sizes itself to fit its
87  // children horizontally, and tries to be as tall as its parent.
88  //
89  // Column has various properties to control how it sizes itself and
90  // how it positions its children. Here we use mainAxisAlignment to
91  // center the children vertically; the main axis here is the vertical
92  // axis because Columns are vertical (the cross axis would be
93  // horizontal).
94  //
95  // TRY THIS: Invoke "debug painting" (choose the "Toggle Debug Paint"
96  // action in the IDE, or press "p" in the console), to see the
97  // wireframe for each widget.
98  mainAxisAlignment: MainAxisAlignment.center,
99  children: <Widget>[
100  const Text(
101    'You have pushed the button this many times:',
102  ), // Text
103  Text(
104    '$_counter',
105    style: Theme.of(context).textTheme.headlineMedium,
106  ), // Text
107  ], // <Widget>[]
108  ), // Column
109  ), // Center

```

ภาพประกอบ 2.40 ลบส่วน Center(...) ออก

ที่มา : ฌปภัช วรณตรง (2564 : 24)

```

79  return Scaffold(
80
81  body: ,
82  floatingActionButton: FloatingActionButton(
83    onPressed: _incrementCounter,
84    tooltip: 'Increment',
85    child: const Icon(Icons.add),
86  ), // This trailing comma makes auto-formatting nicer for build methods. //
87  ); // Scaffold
88
89
90

```

ภาพประกอบ 2.41 ลบส่วน Center(...) ออก (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 24)



1.2.2 จากนั้นให้ทำการพิมพ์คำสั่ง Text(...) เข้าไปในส่วน body: และเพิ่มข้อความที่ต้องการเข้าไปใน “ ”

```

79     return Scaffold(
80
81         body: Text(""),
82         floatingActi
83             onPressed:
84             tooltip: '
85             child: con
86         ), // This t
87     ); // Scaffold
88 }
89
90

```

Text(String data, {Key? key, TextStyle? style, StrutStyle? strutStyle, TextAlign? textAlign, TextDirection? textDirection, Locale? locale, bool? softWrap, TextOverflow? overflow, double? textScaleFactor, int? maxLines, String? semanticsLabel, TextWidthBasis? textWidthBasis, TextHeightBehavior? textHeightBehavior, Color? selectionColor})

Creates a text widget.

ภาพประกอบ 2.42 เพิ่มโค้ด Text(...)

ที่มา : ฌปภัช วรณตรง (2564 : 24)

```

return Scaffold(
  body: Text("Hello"),
  floatingActionButton: FloatingActionButton(
    onPressed: _incrementCounter,
    tooltip: 'Increment',
    child: const Icon(Icons.add),
  ), // This trailing comma makes auto-formatting nicer for build methods. //
); // Scaffold
}

```

ภาพประกอบ 2.43 เพิ่มโค้ด Text(...) (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 24)

1.3 ทำการแก้ไขโค้ด floatingActionButton เพื่อทดลองจัดตำแหน่ง floatingActionButton โดยการนำส่วนของโค้ดที่ต้องการแก้ไขไปค้นหาหรือหาคำตอบผ่าน StackOverflow

เมื่อได้ชุดคำสั่งที่ต้องการแล้วให้ทำการคัดลอกมาใส่ต่อจาก floatingActionButton: FloatingActionButton(...),

```
floatingActionButton: FloatingActionButton(
  onPressed: _incrementCounter,
  tooltip: 'Increment',
  child: const Icon(Icons.add),
), // FloatingActionButton
floatingActionButtonLocation: FloatingActionButtonLocation.centerFloat//
```

ภาพประกอบ 2.44 แก้ไขโค้ด FloatingActionButton

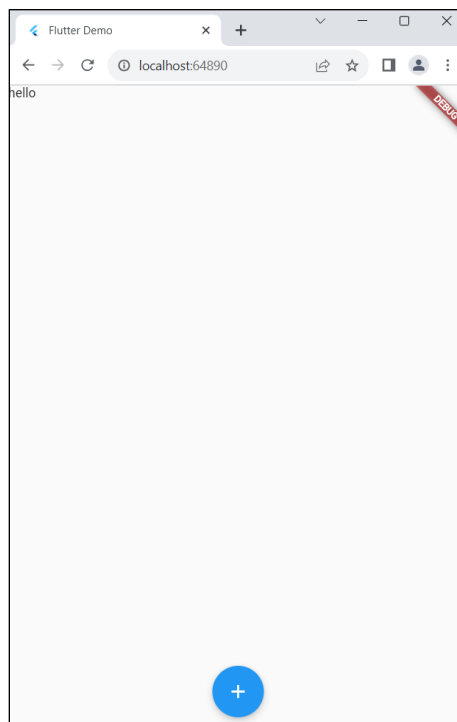
ที่มา : ฌปภัช วรรณตรง (2564 : 26)

ลิงก์ตัวอย่างโค้ดการจัดตำแหน่ง FloatingActionButton

<https://stackoverflow.com/questions/44713501/flutter-floatingactionbutton-in-the-center>

<https://stackoverflow.com/questions/57207469/how-can-i-move-floating-action-button-to-the-left-side-of-the-screen-in-flutter>

ผลลัพธ์ที่ได้จากการปรับปรุงโค้ด จะเห็นได้ว่าตำแหน่ง FloatingActionButton เปลี่ยนไปตามโค้ดที่ได้แก้ไข



ภาพประกอบ 2.45 ผลลัพธ์จากการแก้ไขโค้ด

ที่มา : ฌปภัช วรรณตรง (2564 : 27)

## 1.4 ทำการกำหนด Style ให้กับ Text

1.4.1 ให้ทำการเพิ่ม style: เข้าไปต่อจาก “”, เพื่อทำการกำหนด Style ให้กับ Text ที่ต้องการ โดยการพิมพ์ st จากนั้นจะปรากฏหน้าต่างชุดคำสั่งขึ้นมาให้ทำการเลือก style: ด้วยการคลิกที่คำสั่งหรือ Enter

```

80
81  body: Text("สวัสดี", s),
82  floatingActionButton: FloatingActionButton(
83    onPressed: _incrementCounter,
84    tooltip: 'Increment',
85    child: const Icon(Icons.add),
86  ), // This trailing comma makes auto-formatting nicer for Text
87 ); // Scaffold
88
89

```

### ภาพประกอบ 2.46 กำหนด Style ให้กับ Text

ที่มา : ฌปภัช วรรณตรง (2564 : 34)

1.4.2 จากนั้นภายใน style: ให้ทำการพิมพ์ TextStyle เข้าไป โดยการพิมพ์ TextSt จากนั้นจะปรากฏหน้าต่างชุดคำสั่งขึ้นมาให้ทำการเลือก TextStyle(...) ด้วยการคลิกที่คำสั่งหรือ Enter

```

80
81  body: Text("สวัสดี", style: TextStyle),
82  floatingActionButton: FloatingActionButton(
83    onPressed: _incrementCounter,
84    tooltip: 'Increment',
85    child: const Icon(Icons.add),
86  ), // This trailing comma makes auto-formatting nicer for Text
87 ); // Scaffold
88
89
90

```

### ภาพประกอบ 2.47 กำหนด Style ให้กับ Text (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 34)

1.4.3 ทุกชุดคำสั่งเหมือนเรียกใช้คำสั่งแล้วหากต้องการเพิ่มชุดคำสั่งอื่น ๆ ต้องคั่นด้วยเครื่องหมาย “,” (Comma) ทุกครั้ง

1.4.4 ภายใน TextStyle(...) ให้เพิ่มโค้ด fontSize: [ขนาดของฟอนต์ที่ต้องการ] เข้าไป

```

80
81  body: Text("สวัสดี", style: TextStyle(font)),
82  floatingActionButton: FloatingActionButton(
83    onPressed: _incrementCounter,
84    tooltip: 'Increment',
85    child: const Icon(Icons.add),
86  ), // This trailing comma makes auto-formatting nicer
87  ); // Scaffold
88  }
89  }
90

```

ภาพประกอบ 2.48 เพิ่มโค้ด fontSize:

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 34)

1.4.5 ต่อจาก fontSize: ให้ทำการเพิ่มโค้ด fontWeight: FontWeight.[ประเภทความหนาบางของฟอนต์]

```

body: Text("สวัสดี", style: TextStyle(fontSize: 70, fontWeight: FontWeight.bold)),
floatingActionButton: FloatingActionButton(
  onPressed: _incrementCounter,
  tooltip: 'Increment',
  child: const Icon(Icons.add),
), // This trailing comma makes auto-formatting nicer
); // Scaffold

```

ภาพประกอบ 2.49 เพิ่มโค้ด fontWeight:

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 34)

```
body: Text("สวีส์ดี",style: TextStyle(fontSize: 70,fontWeight: FontWeight.)),),
floatingActionButton: FloatingActionButton(
  onPressed: _incrementCounter,
  tooltip: 'Increment',
  child: const Icon(Icons.add),
), // This trailing comma makes auto-formatting nicer for build methods
); // Scaffold
```

ภาพประกอบ 2.50 เพิ่มโค้ด fontWeight: (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 34)

1.4.6 ต่อจาก fontWeight: ให้ทำการเพิ่มโค้ด color: Colors.[สีของฟอนต์]

```
body: Text("สวีส์ดี",style: TextStyle(fontSize: 70,fontWeight: FontWeight.bold ,c)),),
floatingActionButton: FloatingActionButton(
  onPressed: _incrementCounter,
  tooltip: 'Increment',
  child: const Icon(Icons.add),
), // This trailing comma makes auto-formatting nicer for build methods. // FloatingActionButton
); // Scaffold
```

ภาพประกอบ 2.51 เพิ่มโค้ด color:

ที่มา : ฅนปักษ์ วรณตรง (2564 : 34)

```
body: Text("สวีส์ดี",style: TextStyle(fontSize: 70,fontWeight: FontWeight.bold ,color: Colors.)),),
floatingActionButton: FloatingActionButton(
  onPressed: _incrementCounter,
  tooltip: 'Increment',
  child: const Icon(Icons.add),
), // This trailing comma makes auto-formatting nicer for build methods. // FloatingA
); // Scaffold
```

ภาพประกอบ 2.52 เพิ่มโค้ด color: (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 34)

1.4.7 ได้ผลลัพธ์โค้ดการกำหนด Style ให้กับ Text ทั้งหมดดังภาพ

```
body: Text("สวีส์ดี",
style: TextStyle (fontSize: 100, fontWeight: FontWeight.bold,
color: Colors.blue)), // TextStyle // Text
```

ภาพประกอบ 2.53 กำหนด Style ให้กับ Text

ที่มา : ฅนปักษ์ วรณตรง (2564 : 34)

หมายเหตุ
หากกดปุ่ม ctrl + spacebar จะทำการแสดงคำสั่งให้อัตโนมัติ

1.5 เมื่อกำหนด Style ให้กับ Text เรียบร้อยแล้วให้ทำการกำหนด Text ให้อยู่ตรงกลาง โดยใช้ Center

15.1 คัดลอกหรือตัดโค้ด Text(...) ออกมาเพื่อทำการเพิ่มโค้ด Center(...) จากนั้นให้ทำการวางโค้ด Text(...) เข้าไปไว้ใน Center(...) เพื่อกำหนดให้อยู่กึ่งกลาง

```

80
81 body: Text("สวัสดี", style: TextStyle(fontSize: 70, fontWeight: FontWeight.bold
82 , color: Colors.blue[200])), // TextStyle // Text
83 floatingActionButton: FloatingActionButton(
84   onPressed: _incrementCounter,
85   tooltip: 'Increment',
86   child: const Icon(Icons.add),
87 ), // This trailing comma makes auto-formatting nicer for build methods. // Flo
88 ); // Scaffold
89 }
90 }

```

ภาพประกอบ 2.54 ตัดส่วน Text(...) ไว้

ที่มา : ฅนปลัซ วรณตรง (2564 : 34)

15.2 เมื่อทำการตัดโค้ดส่วน Text(...) เอาไว้แล้ว ให้ทำการเพิ่มโค้ด Center(...) โดยการพิมพ์ Cente จากนั้นจะปรากฏหน้าต่างชุดคำสั่งขึ้นมาให้ทำการเลือก Center(...) ด้วยการคลิกที่คำสั่งหรือ Enter

```

81 body: Cen,
82 floatingA Center
83   onPress Center(...) ({Key? key, double? widthFactor, double? ...
84   tooltip FabCenterOffsetX
85   child: ChangeNotifier
86 ), // Thi ChangeNotifier() methods. //
87 ); // Scaff ClipPathEngineLayer
88 ClipRectEngineLayer
89 CalendarDatePicker

```

ภาพประกอบ 2.55 เพิ่มโค้ด Center(...)

ที่มา : ฅนปลัซ วรณตรง (2564 : 36)

1.5.3 ภายใน Center(...) จะต้องมี child: เพื่อเก็บวิดเจ็ตหรือชุดคำสั่งที่ใช้ใน Flutter จากนั้นเมื่อเพิ่ม child: เข้ามาแล้วให้ทำการวางโค้ด Text(...) ที่ได้ทำการตัดหรือคัดลอกเอาไว้เข้าไป

```

81   body: Center(child: ),
82   floatingActionButton: FloatingActionButton(
83     onPressed: _incrementCounter,
84     tooltip: 'Increment',
85     child: const Icon(Icons.add),
86   ), // This trailing comma makes auto-formatting nicer for build methods. //
87 ); // Scaffold
88 }
89 }

```

ภาพประกอบ 2.56 เพิ่มโค้ดและวางโค้ดที่ตัดไว้เข้าไปใน Center(...)

ที่มา : ฌปภัช วรรณตรง (2564 : 36)

```

80
81   body: Center(child: Text("สวัสดี", style: TextStyle(fontSize: 70, fontWeight: FontWeight.
82     ,color: Colors.blue[200]),),), // TextStyle // Text // Center
83   floatingActionButton: FloatingActionButton(
84     onPressed: _incrementCounter,
85     tooltip: 'Increment',
86     child: const Icon(Icons.add),
87   ), // This trailing comma makes auto-formatting nicer for build methods. // FloatingA
88 ); // Scaffold
89 }
90 }

```

ภาพประกอบ 2.57 เพิ่มโค้ดและวางโค้ดที่ตัดไว้เข้าไปใน Center(...) (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 36)

1.5.4 ได้ผลลัพธ์โค้ดการกำหนด Text ให้อยู่ตรงกลางทั้งหมดดังภาพ

```

body: Center(child: Text("สวัสดี",
style: TextStyle (fontSize: 100, fontWeight: FontWeight.bold,
color: Colors.blue),),),) // TextStyle // Text // Center

```

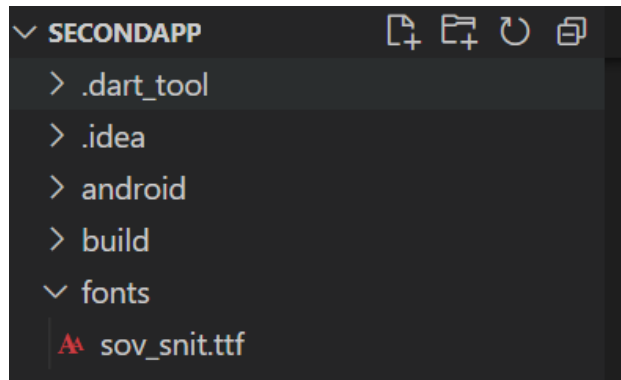
ภาพประกอบ 2.58 กำหนด Text ให้อยู่ตรงกลาง

ที่มา : ฌปภัช วรรณตรง (2564 : 36)

## 1.6 สามารถกำหนดรูปแบบ Font ให้กับ Text ได้



ภาพประกอบ 2.59 ดาวน์โหลด Font ที่ต้องการจะใช้งานในแอปพลิเคชัน  
ที่มา : ฅนปลั๊ก วรรณตรง (2564 : 39)



ภาพประกอบ 2.60 สร้าง Folder “font” สำหรับใช้เก็บ Font ที่ทำการดาวน์โหลดมา  
ที่มา : ฅนปลั๊ก วรรณตรง (2564 : 39)

```
fonts:
  - family: sovsnit
    fonts:
      - asset: fonts/sov_snit.ttf
```

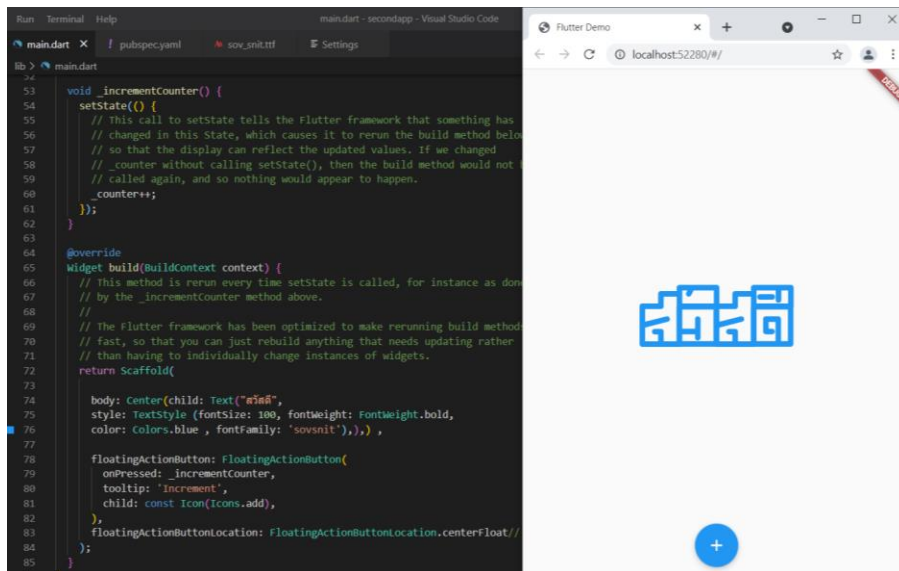
ภาพประกอบ 2.61 แก้ไขไฟล์ที่มีชื่อว่า pubspec.yaml ตามภาพ  
ที่มา : ฅนปลั๊ก วรรณตรง (2564 : 39)



```
body: Center(child: Text("สวัสดี",
style: TextStyle (fontSize: 100, fontWeight: FontWeight.bold,
color: Colors.blue , fontFamily: 'sovsnit'),),), // TextStyle // Te
```

ภาพประกอบ 2.62 แก้ไขโค้ดโดยการใส่ fontFamily ที่สร้างในไฟล์ main.dart ตามภาพ  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 39)

1.7 ผลลัพธ์ที่ได้ในการสร้างแอปพลิเคชันจาก Flutter จะเห็นได้ว่ารูปแบบ Font ของText  
เปลี่ยนไปตามที่ได้กำหนด



ภาพประกอบ 2.63 ผลลัพธ์การสร้างแอปพลิเคชันจาก Flutter  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 40)

## บทสรุป

สำหรับเนื้อหาที่กล่าวมาทั้งหมดในบทนี้ จะพูดถึงวิดเจ็ตคืออะไร วิดเจ็ตพื้นฐานที่อยู่ในภาษา Dart ซึ่งเป็นภาษาที่ใช้ในการพัฒนาแอปพลิเคชันด้วย Flutter ประเภทของวิดเจ็ตว่ามีประเภทใดบ้าง และแต่ละประเภทมีลักษณะอย่างไร วิธีการสร้างวิดเจ็ตการเริ่มสร้าง Flutter Project และขั้นตอนการสร้างวิดเจ็ตในโปรเจกต์ วิธีการตกแต่งตัวอักษร การดาวน์โหลดรูปแบบตัวอักษรที่ต้องการ ตั้งค่ารูปแบบตัวอักษร กำหนดขนาด สี ของตัวอักษร

## เอกสารอ้างอิง

จิราวุธ วารินทร์. (2564). **พัฒนาโมบายล์แอปด้วย Flutter + Dart**. กรุงเทพฯ : สำนักพิมพ์ชิมพลิฟาย.

บัญชา ปะลีละเตสัง. (2566). **พัฒนาแอปแบบ Multi-Platform ด้วย Flutter โดยใช้ภาษา Dart**.

กรุงเทพฯ : ซีเอ็ดดูเคชั่น.

อนุชิต ชโลธร. (2565). **สูตรลัด Flutter**. กรุงเทพฯ : สำนักพิมพ์ก๊อปวาง.

เอกรินทร์ วัญญูเลิศสกุล. (2563). **พัฒนา Mobile App ด้วย Flutter & Dart**. กรุงเทพฯ :

โปรวิชั่น.

## บทที่ 3

### ภาษา Dart (Dart Language)

ในการพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่ ภาษาที่ใช้ในการพัฒนาก็เป็นสิ่งจำเป็นที่ต้องเรียนรู้ สำหรับภาษาที่ใช้พัฒนาพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่ด้วย Flutter คือ การเขียนโปรแกรมภาษา Dart ดังนั้นในบทนี้จะเป็นการปูพื้นฐานภาษา Dart เพื่อให้ผู้เรียนมีพื้นฐานนำไปพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่ด้วย Flutter ได้ โดยมีเนื้อหาการเขียนโปรแกรมภาษา Dart ดังนี้

- 1) ภาษา Dart คืออะไร
- 2) ประวัติภาษา Dart
- 3) ชนิดตัวแปรในภาษา Dart
- 4) ตัวดำเนินการทางคณิตศาสตร์ (Operator) ที่ใช้ในภาษา Dart
- 5) การใช้ If/Else, Loop, Switch Case ในภาษา Dart
- 6) การเขียนฟังก์ชัน (Function) ในภาษา Dart
- 7) List คือ รายการข้อมูลที่สามารถจัดเรียงต่อเนื่องกัน อยู่ในรูปแบบของอาร์เรย์ และ
- 8) Null Safety คือ การตรวจสอบตัวแปรที่เป็นค่า null ในภาษา Dart

จากเนื้อหาข้างต้นที่กล่าวมาทั้งหมด มีรายละเอียดดังต่อไปนี้

#### ภาษา Dart คืออะไร

Dart เป็นภาษาที่ออกแบบมาให้พัฒนาแอปบนแพลตฟอร์มต่าง ๆ ให้รวดเร็ว ช่วยใช้เขียนโปรแกรมในหลาย ๆ แพลตฟอร์มได้ง่ายและสะดวกยิ่งขึ้น มีความยืดหยุ่นทำให้สามารถ Compile เป็นแอปพลิเคชันสำหรับการทำงานได้ในหลาย ๆ แพลตฟอร์ม

ภาษา Dart เหมาะสำหรับการพัฒนาแอปฯ ในฝั่ง Client โดยมีคุณสมบัติอย่างเช่น Hot reload เพื่อช่วยให้นักพัฒนาเห็นความเปลี่ยนแปลง ทันทีเมื่อแก้ไขโค้ดและยังรองรับการ Compile ไปยังแพลตฟอร์มต่าง ๆ ทั้ง Mobile, Desktop Web (อนุชิต ชโลธร, 2566 : 6)

#### ประวัติภาษา Dart

ภาษา Dart ได้รับการเปิดตัวครั้งแรกในการประชุม GOTO ที่จัดขึ้นในระหว่างวันที่ 10 - 12 เดือน ตุลาคม ค.ศ. 2010 ที่ประเทศเดนมาร์ก พัฒนาโดย Lars Bak โปรแกรมเมอร์สัญชาติเดนมาร์กที่มีชื่อเสียงในการทำงานกับเครื่องจักรเสมือนจริง (Virtual Machine) และมีส่วนร่วมในการสร้างเว็บเบราว์เซอร์ Google Chrome

Dart เป็นภาษาที่สร้างใหม่ ดังนั้นเพื่อสร้างมาตรฐานให้กับภาษาการเขียนโปรแกรมที่สร้างขึ้นใหม่นี้ ECMA (European Computer Manufacturers Association) หรือรู้จักกันในนาม ECMA International ซึ่งเป็นหน่วยงานที่ช่วยบริษัททางคอมพิวเตอร์ต่าง ๆ ในการออกแบบมาตรฐานของภาษาทางคอมพิวเตอร์ได้จัดตั้งคณะกรรมการด้านเทคนิคที่รู้จักกันในชื่อ TC52 และด้วยหลักการ

พื้นฐานของภาษา Dart ที่สามารถคอมไพล์เป็นจาวาสคริปต์ได้ จึงทำให้ภาษา Dart ได้รับการอนุมัติมาตรฐานภาษา Dart รุ่นแรกในเดือนกรกฎาคม ค.ศ. 2014 และจากนั้นอีก 6 เดือนต่อมาได้รับการอนุมัติจาก ECMA International ออกเป็นข้อกำหนดมาตรฐานภาษารุ่นที่ 2 ในเดือนธันวาคม ค.ศ. 2014 ซึ่งวัตถุประสงค์ของการสร้างภาษา Dart ถูกออกแบบมาเพื่อเป็นภาษาเชิงโครงสร้างที่มีความยืดหยุ่นสูง และได้มีการออกแบบโครงสร้างของภาษาไปพร้อมกับ Engine สำหรับการคอมไพล์โปรแกรม โดยมุ่งเน้นการแก้ปัญหการทำงานช้าของโปรแกรมหรือแอปพลิเคชันที่พัฒนาขึ้นมา ลดการใช้ทรัพยากรของระบบทั้งการประมวลผลและการใช้งานหน่วยความจำ แต่คงรูปแบบโครงสร้างความเป็นภาษาเชิงวัตถุแบบ C/C++ และ Java (เอกรินทร์ วทัญญูเลิศสกุล, 2563 : 36)

## ชนิดตัวแปรในภาษา Dart

ชนิดตัวแปรในภาษา Dart มีลักษณะดังนี้

ตาราง 3.1 ตารางชนิดตัวแปรในภาษา Dart

ตัวแปร	คำอธิบาย	ตัวอย่าง
Int	เลขจำนวนเต็ม	0, 1, -5, 86400
Double	เลขทศนิยม	0.0, 0.1, 0.14, -12.34
Number	เลขจำนวนเต็ม และเลขทศนิยม	123, 0.123
Bool	ค่าทางตรรกศาสตร์	True, False
String	ข้อความ หรือตัวอักษร	'Hello world!', "This is a book"
Map<index,value>	ชนิดข้อมูลที่เป็น array โดยมี key เป็นตัวเลขเรียงเป็นลำดับ เป็น base zero หรือเริ่มต้นที่ 0	["แอปเปิล", "มะนาว", "มันฝรั่ง"]
List<type>	ชนิดข้อมูลที่เป็น object โดยมาเป็นชุดข้อมูลที่มี key และ value จับคู่กัน	{firstName: "bas", lastName: "suksai", age: 28};
Dynamic	ตัวแปรที่สามารถเปลี่ยนแปลงได้	1, 0.14, true, "Hi!"

Dart ยังมีชนิดของตัวแปรแบบพิเศษ ซึ่งไม่จำเป็นต้องประกาศ Type เพราะตัวภาษาจะทำการ Auto Assign ชนิดของตัวแปรให้โดยอัตโนมัติ (เอกรินทร์ วทัญญูเลิศสกุล, 2563 : 42-44)

ตาราง 3.2 ตารางชนิดตัวแปรชนิดพิเศษในภาษา Dart

ตัวแปร	คำอธิบาย
Var	เป็นการละเว้น Type เอาไว้ให้โปรแกรมกำหนดให้ (ตาม Value)
Final	เหมือน Var แต่ไม่สามารถเปลี่ยนแปลงค่าได้
Const	ค่าคงที่

### 1. ข้อแตกต่างระหว่าง Dynamic vs Var คือ

**Dynamic** เป็นตัวแปรเก็บค่าชนิดไหนก็ได้ เปลี่ยนแปลงได้เรื่อย ๆ การใช้ Dynamic มีความเสี่ยงทำให้เกิด Runtime error ได้ เพราะ Compiler ไม่สามารถช่วยตรวจสอบชนิดของตัวแปรได้

**var** จะเป็นการกำหนดชนิดตัวแปร โดยดูชนิดตัวแปรจาก value หลังจากนั้นตัวแปรจะถูกกำหนดเป็น type นั้นไปตลอด ไม่สามารถเปลี่ยนแปลงได้แล้ว (ปัญหา ปะสีละเตสัง, 2566 : 30-33)

### 2. ข้อแตกต่างระหว่าง final vs const คือ

**final** เป็นการกำหนดว่าตัวแปรที่ไม่สามารถเปลี่ยนแปลงค่าได้ ซึ่งเป็นตัวแปรประเภท runtime ดังนั้นสามารถกำหนดค่า final จากตัวแปรหรือฟังก์ชันอื่นได้

**const** เป็นการประกาศตัวแปรที่ไม่ต้องการเปลี่ยนแปลงค่า หากประกาศไปแล้วจะไม่สามารถแก้ไขค่าตัวแปร นั้นซ้ำได้อีก (จิราวุธ วารินทร์, 2564 : 142)

#### String methods

- length นับจำนวน String
- subString(start, [end]) เลือกส่วนของ String โดยระบุตำแหน่งเริ่มต้นที่ต้องการตัด ส่วนตำแหน่งสิ้นสุด จะระบุหรือไม่ก็ได้
- split(object) แยก String ออกเป็นส่วน ๆ แล้วคืนค่าเป็น List โดยกำหนดว่าจะให้แยก String ด้วยอะไร
- toLowerCase() ปรับ String เป็นตัวพิมพ์เล็กทั้งหมด
- toUpperCase() ปรับ String เป็นตัวพิมพ์ใหญ่ทั้งหมด (ปัญหา ปะสีละเตสัง, 2566 : 46-47)

ตัวอย่างการใช้งาน String methods ดังภาพ

```

1 void main() {
2   String message = 'Hello Dart';
3   print(message.length);
4   print(message.substring(2));
5   print(message.substring(6));
6   print(message.split(' '));
7   print(message.split('!'));
8   print(message.toLowerCase());
9   print(message.toUpperCase());
10 }

```

ภาพประกอบ 3.1 ตัวอย่างการใช้งาน String methods

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

```

10
llo Dart
Dart
[H, e, l, l, o, , D, a, r, t]
[Hello Dart]
hello dart
HELLO DART

```

ภาพประกอบ 3.2 ผลลัพธ์การใช้งาน String methods

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

## Operator

Operator คือ ตัวดำเนินการทางคณิตศาสตร์ที่ใช้ประเมินผล (Evaluate) หรือกำหนดค่า (Assign) แบ่งออกเป็นกลุ่มตามการใช้งานได้ดังนี้

Arithmetic Operators Operator ใน Dart เป็นการเพิ่มและลดค่าทีละ 1 ให้กับตัวแปรที่ได้ประกาศไว้ (เอกรินทร์ วิทยุเลิศสกุล, 2563 : 45) ตัวอย่างการใช้งาน Arithmetic Operators Operator ดังภาพ

```

1 void main() {
2     int a = 5;
3     print(a++);
4     print(a--);
5     print(++a);
6     print(--a);
7 }

```

ภาพประกอบ 3.3 ตัวอย่างการใช้งาน Arithmetic Operators Operator  
ที่มา : ฌปภัช วรรณตรง (2564 : ...)

```

5
6
6
5

```

ภาพประกอบ 3.4 ผลลัพธ์การใช้งาน Arithmetic Operators Operator  
ที่มา : ฌปภัช วรรณตรง (2564 : ...)

Equality and Relational Operators เอาไว้กำหนดความสัมพันธ์ระหว่างตัวแปรสองตัว เช็คค่าตัวแปร ทำให้ได้คำตอบออกมาเป็น Boolean (True, False) ได้แก่ > (มากกว่า), < (น้อยกว่า), == (เท่ากับหรือมากกว่า), <= (เท่ากับหรือน้อยกว่า), == (เท่ากับ), != (ไม่เท่ากับ)

Type test Operators เอาไว้เช็ค Type ของตัวแปร ตอน runtime ได้แก่ is และ is! (เอกรินทร์ วัฒนฤเลิศสกุล, 2563 : 46) ตัวอย่างการใช้งาน Type test Operators ดังภาพ

```

1 void main() {
2     var e;
3     print(e is int);           // false
4     print(e is! int);        // true
5 }

```

ภาพประกอบ 3.5 ตัวอย่างการใช้งาน Type test Operators  
ที่มา : ฌปภัช วรรณตรง (2564 : ...)

```

false
true

```

ภาพประกอบ 3.6 ผลลัพธ์การใช้งาน Type test Operators

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

```

1 void main() {
2     int a = 5;
3     int b = 10;
4     print(a == b);
5     print(a != b);
6     print(a > b);
7     print(a < b);
8     print(a >= b);
9     print(a <= b);
10 }

```

ภาพประกอบ 3.7 ตัวอย่างการใช้งาน Equality and Relational Operators

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

```

false
true
false
true
false
true

```

ภาพประกอบ 3.8 ผลลัพธ์การใช้งาน Equality and Relational Operators

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

Bitwise Operators คือ ตัวดำเนินการแบบบิต เป็นตัวดำเนินการที่ใช้กับชนิดข้อมูลเลขจำนวนเต็ม (integer) เท่านั้น โดยจะแปลงเป็นเลขฐาน 2 (8-bit) ก่อนดำเนินการ หลังจากนั้น จะแปลงค่ากลับเป็นฐาน 10 กลับมา ซึ่งมีตัวดำเนินการได้แก่ & (AND) เป็นการหาเลข 1 ที่ตำแหน่ง (บิต) เดียวกัน มีเงื่อนไขดังนี้ ถ้า 1&1 เป็น 1, ถ้า 1&0 เป็น 0, ถ้า 0&1 เป็น 0, ถ้า 0&0 เป็น 0



Assignment Operators เป็นตัวดำเนินการที่เป็นการกำหนดค่าให้ตัวแปร มีดังนี้

= (Simple Assignment) คือ การกำหนดค่าตัวแปรให้เป็นค่าต่าง ๆ คืค่าเป็นค่าที่กำหนดให้

?? = กำหนดค่าให้ตัวแปร ถ้าตัวแปรนั้นเป็น null ถ้าตัวแปรนั้นไม่ใช่ null ค่าของตัวแปรนั้นจะคงเดิม

+ = (Add and Assignment) คือ การบวกค่าตัวแปรนั้นด้วยค่าที่กำหนด คืค่าเป็นตัวแปรที่ผ่านการบวกแล้ว

- = (Subtract and Assignment) คือ การบวกค่าตัวแปรนั้นด้วยค่าที่กำหนด คืค่าเป็นตัวแปรที่ผ่านการลบแล้ว

\* = (Multiply and Assignment) คือ การบวกค่าตัวแปรนั้นด้วยค่าที่กำหนด คืค่าเป็นตัวแปรที่ผ่านการคูณแล้ว

/ = (Divide and Assignment) คือ การบวกค่าตัวแปรนั้นด้วยค่าที่กำหนด คืค่าเป็นตัวแปรที่ผ่านการหารแล้ว

~/ (Floored Integer Division) คือ การหารค่าที่มีทศนิยมแล้วต้องแปลง cast เพื่อให้เป็น integer

% (Division Remainder) คือ การหารเพื่อเอาเศษที่เหลือจากการหารไปใช้งาน โดยที่ทั้งตัวตั้งและตัวหารต้องเป็นเลขจำนวนเต็ม (เอกรินทร์ วทัญญเลิศสกุล, 2563 : 46-47)

ตัวอย่างการใช้งาน Assignment Operators ดังภาพ

```

1 void main() {
2     int a = 5;
3     int b =10;
4     print(a + b);
5     print(a - b);
6     print(a * b);
7     print(a / b);
8     print(a ~/ b);
9     print(a % b);
10    print(a ?? b);
11    print(a = b);
12 }
```

ภาพประกอบ 3.9 ตัวอย่างการใช้งาน Assignment Operators

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

```

15
-5
50
0.5
0
5
5
10

```

ภาพประกอบ 3.10 ผลลัพธ์การใช้งาน Assignment Operators

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

Logical Operators คือตัวดำเนินการทางตรรกศาสตร์ เป็นองค์ประกอบที่ใช้ในการประมวลผล และจัดการกับค่าที่เป็นข้อมูลประเภทบูลีน (Boolean) ซึ่งมีค่าเป็นเพียงสองค่าเท่านั้นคือ True (จริง) และ False (เท็จ) โดยมักใช้ในการทำเงื่อนไขและการควบคุมกระบวนการทำงานของโปรแกรม (เอกรินทร์ วทัญญูเลิศสกุล, 2563 : 47)

ตาราง 3.3 ตารางตัวดำเนินการทางตรรกศาสตร์

Operator	ความหมาย
!	กลับค่าระหว่าง True หรือ False
	เหมือน OR ในตรรกศาสตร์
&&	เหมือน AND ในตรรกศาสตร์

```

1 void main() {
2     int a = 5;
3     int b = 10;
4     print(a > 0 && b > 0);
5     print(a < 5 || b < 5);
6     print(!(a > 10 || b > 10));
7 }

```

ภาพประกอบ 3.11 ตัวอย่างการใช้งาน Logical Operators

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

```

true
false
true

```

ภาพประกอบ 3.12 ผลลัพธ์การใช้งาน Logical Operators

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

### If/Else, Loop, Switch Case

กลุ่มคำสั่งที่ใช้ในการควบคุมการทำงานของโปรแกรม โดยในภาษา Dart จะประกอบไปด้วย 3 รูปแบบ

- แบบลำดับ (Sequence) คือ การทำงานจากบนลงล่าง ซ้ำๆ ไปขวา
- แบบมีเงื่อนไข (Condition) คือ กลุ่มคำสั่งที่ใช้ตัดสินใจในการเลือกเงื่อนไขต่าง ๆ ภายใน

โปรแกรมมาทำงาน ซึ่งจะประกอบไปด้วย 2 คำสั่ง คือ if else และ switch case

If เป็นคำสั่งที่ใช้กำหนดเงื่อนไขในการตัดสินใจทำงานของโปรแกรม ถ้าเงื่อนไขเป็นจริงก็จะทำตามคำสั่งต่าง ๆ ที่ได้กำหนดในเงื่อนไขนั้น ๆ (จิราวุธ วารินทร์, 2564 : 99) ตัวอย่างการใช้งาน If แบบเงื่อนไขเดียว ดังภาพ

```

1 void main(){
2   int x = 10,y=15;
3   if(x>y){
4     print("$x มากกว่า $y"); //ถ้าเงื่อนไขจริง
5   }
6   print("จบการทำงาน");
7 }

```

ภาพประกอบ 3.13 ตัวอย่างการใช้งาน If แบบเงื่อนไขเดียว

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

```

10 มากกว่า 8
จบการทำงาน

```

ภาพประกอบ 3.14 ผลลัพธ์การใช้งาน If แบบเงื่อนไขเดียว

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

```

1 void main(){
2     int x = 10;
3     if(x % 2 == 0){
4         print("$x เป็นเลขคู่");//ถ้าเงื่อนไขจริง
5     }else{
6         print("เป็นเลขคี่");
7     }
8 }

```

ภาพประกอบ 3.15 ตัวอย่างการใช้งาน if แบบ 2 เงื่อนไขเดียว

ที่มา : ฅนปักษ์ วรรณตรง (2564 : ...)

10 เป็นเลขคู่

ภาพประกอบ 3.16 ผลลัพธ์การใช้งาน if แบบ 2 เงื่อนไขเดียว

ที่มา : ฅนปักษ์ วรรณตรง (2564 : ...)

```

1 void main(){
2     int x = 10, y = 15;
3     if(x > y){
4         print("$x มากกว่า $y");
5     }else if(x < y){
6         print("$x น้อยกว่า $y");
7     }else{
8         print("$x เท่ากับ $y");
9     }
10 }

```

ภาพประกอบ 3.17 ตัวอย่างการใช้งาน if แบบหลายเงื่อนไข

ที่มา : ฅนปักษ์ วรรณตรง (2564 : ...)

10 น้อยกว่า 15

ภาพประกอบ 3.18 ผลลัพธ์การใช้งาน If แบบหลายเงื่อนไข

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

### 1. Switch..Case

Switch เป็นคำสั่งที่ใช้กำหนดเงื่อนไขคล้าย ๆ กับ If แต่จะเลือกเพียงหนึ่งทางเลือกออกมาทำงานโดยนำค่าในตัวแปรมากำหนดทางเลือกผ่านคำสั่ง Case ตัวอย่างเช่น เจอ Case ที่ 1 จะให้ทำอะไร Case ทำอะไร (จิราวุธ วารินทร์, 2564 : 101) ตัวอย่างการใช้งาน Switch..Case ดังภาพ

```

1 void main() {
2     var grade = "A";
3     switch(grade) {
4         case "A": { print("ยอดเยี่ยม"); }
5         break;
6
7         case "B": { print("ดี"); }
8         break;
9
10        case "C": { print("พอใช้"); }
11        break;
12
13        case "D": { print("ปรับปรุง"); }
14        break;
15    }
16 }
```

ภาพประกอบ 3.19 ตัวอย่างการใช้งาน Switch..Case

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

ยอดเยี่ยม

ภาพประกอบ 3.20 ผลลัพธ์การใช้งาน Switch..Case

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

## 2. Loop

กลุ่มคำสั่งที่ใช้ในการวนวนลูป โดยโปรแกรมจะทำงานไปเรื่อย ๆ จนกว่าเงื่อนไขที่กำหนดนั้นจะเป็นเท็จ โปรแกรมจึงจะหยุดทำงาน โดยในที่นี้จะมียู่ 3 คำสั่งคือ While, For และ Do While ซึ่งแต่ละตัวก็จะมีรูปแบบการใช้งานที่แตกต่างกันไป While Loop จะทำงานภายในคำสั่ง While ไปเรื่อย ๆ เมื่อเงื่อนไขที่กำหนดเป็นจริง เมื่อเงื่อนไขเป็นจริงก็จะทำซ้ำไปเรื่อย ๆ (เอกรินทร์ วทัญญูเลิศสกุล, 2563 : 51-53)

ตัวอย่างการใช้งาน While ดังภาพ

```

1 void main(){
2     int x=1;
3     while(x<=3){
4         print("Hello dart");
5         x++;
6     }
7 }
```

ภาพประกอบ 3.21 ตัวอย่างการใช้งาน While

ที่มา : ณปภัช วรรณตรง (2564 : ...)

```

Hello dart
Hello dart
Hello dart
```

ภาพประกอบ 3.22 ผลลัพธ์การใช้งาน While

ที่มา : ณปภัช วรรณตรง (2564 : ...)

## 3. For Loop

เป็นรูปแบบที่ใช้ตรวจสอบเงื่อนไข มีการกำหนดค่าเริ่มต้น และเปลี่ยนค่าไปพร้อม ๆ กัน เมื่อเงื่อนไขที่กำหนดเป็นจริง เมื่อเงื่อนไขเป็นจริงก็จะทำซ้ำไปเรื่อย ๆ

ตัวอย่างการใช้งาน For ดังภาพ

```

1 void main(){
2     for(var x=1;x<=5;x++){
3         print("$x");
4     }
5 }
6 }

```

ภาพประกอบ 3.23 ตัวอย่างการใช้งาน For

ที่มา : ฅนปักษ์ วรรณตรง (2564 : ...)

```

1
2
3
4
5

```

ภาพประกอบ 3.24 ผลลัพธ์การใช้งาน For

ที่มา : ฅนปักษ์ วรรณตรง (2564 : ...)

#### 4. Do While

โปรแกรมจะทำงานตามคำสั่งก่อนอย่างน้อย 1 รอบแล้วมาตรวจสอบเงื่อนไขที่คำสั่ง While ถ้าเป็นจริงก็จะวนกลับขึ้นไปทำงานอีกรอบแล้วถ้าเป็นเท็จก็จะหลุดออกจากลูป

ตัวอย่างการใช้งาน Do While ดังภาพ

```
1 void main(){
2   var x=10;
3   do{
4     print("hello dart");
5     x++;
6   } while(x<=3);
7 }
```

ภาพประกอบ 3.25 ตัวอย่างการใช้งาน Do While

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

hello dart

ภาพประกอบ 3.26 ผลลัพธ์การใช้งาน Do While

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

## Function

Function เป็นชุดของการทำงาน Function เป็นส่วนหนึ่งของ Code ซึ่งถูกเรียกโดยชื่อของฟังก์ชัน ฟังก์ชันสามารถส่งผ่านข้อมูล (Parameter) เพื่อทำอะไรบางอย่าง และส่งคืนค่า (จิราวุธ วารินทร์, 2564 : 77)

ตัวอย่างการใช้งาน Function ดังภาพ

```
1 void main() {
2   print('Hello Dart');
3 }
```

ภาพประกอบ 3.27 ตัวอย่างการใช้งาน Function

ที่มา : ฌปภัช วรรณตรง (2564 : ...)



Arrow Function คือ รูปแบบชนิดหนึ่ง ของ Function ที่เขียนให้สั้นลงและง่ายขึ้น

```
1 void main() => print('Hello dart');
```

ภาพประกอบ 3.28 ตัวอย่างการใช้งาน Arrow Function

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

### ประเภทของ Function

Function : No return & No Args

เป็น Function ที่ไม่ต้องรับค่าเข้ามาเพื่อประมวลผล ไม่คืนค่ากลับหลังจากทำงานเสร็จ

(void + No return) (จิราวุธ วารินทร์, 2564 : 79-81)

ตัวอย่างการใช้งาน Function : No return & No Args ดังภาพ

```
1 void main() {
2 test1();
3 }
4 void test1() {
5 print("No Return & No Args Function");
6 }
```

ภาพประกอบ 3.29 ตัวอย่างการใช้งาน Function : No return & No Args

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

No Return & No Args Function

ภาพประกอบ 3.30 ผลลัพธ์การใช้งาน Function : No return & No Args

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

Function : No return & Args

เป็น Function ที่ต้องรับค่าเข้ามาเพื่อประมวลผล ไม่คืนค่ากลับหลังจากทำงานเสร็จ (void + No return)

ตัวอย่างการใช้งาน Function : No return & Args ดังภาพ

```

1 void main() {
2     test2("Hello", "Dart");
3 }
4
5 void test2(String name, String surname) {
6     print("name is $name $surname");
7 }

```

ภาพประกอบ 3.31 ตัวอย่างการใช้งาน Function : No return & Args

ที่มา : ฅนปักษ์ วรรณตรง (2564 : ...)

name is Hello Dart

ภาพประกอบ 3.32 ผลลัพธ์การใช้งาน Function : No return & Args

ที่มา : ฅนปักษ์ วรรณตรง (2564 : ...)

Function : return & No Args

เป็น Function ที่ไม่ต้องรับค่าเข้ามาเพื่อประมวลผล แต่มีการคืนค่ากลับหลังจากทำงานเสร็จ (type + return)

ตัวอย่างการใช้งาน Function : return & No Args ดังภาพ


```

1 void main() {
2     print(test3());
3 }
4
5 String test3() {
6     String a = 'Hello Dart';
7     return a;
8 }

```

ภาพประกอบ 3.33 ตัวอย่างการใช้งาน Function : return & No Args

ที่มา : ฅนปักษ์ วรรณตรง (2564 : ...)



Hello Dart

ภาพประกอบ 3.34 ผลลัพธ์การใช้งาน Function : return & No Args

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

Function : return & Args

เป็น Function ที่มีการรับค่าเข้ามาเพื่อประมวลผลและมีการคืนค่ากลับหลังจากทำงานเสร็จ

ตัวอย่างการใช้งาน Function : return & Args ดังภาพ

```
1 void main() {
2   print(test4(5));
3 }
4 int test4(int a ) {
5   return a+50;
6 }
```

ภาพประกอบ 3.35 ตัวอย่างการใช้งาน Function : return & Args

ที่มา : ฌปภัช วรรณตรง (2564 : ...)



55

ภาพประกอบ 3.36 ผลลัพธ์การใช้งาน Function : return & Args

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

Optional Positional Argument Optional

คือ optional parameter ที่มีลักษณะดังนี้

- Optional Positional Argument ใช้ [ ] ครอบ Argument ที่ต้องการให้ เป็น Optional

- Optional Positional Argument การส่งค่าจะอิงกับตำแหน่งของ Argument ที่กำหนด

ไว้ ถ้าส่งค่าไม่ครบหรือไม่ตรงตำแหน่ง ค่าที่ออกมาจะมีโอกาสเพี้ยนได้ (ลุงวิศวกร สอนคำนวณ, 2564 :

66-75)

ตัวอย่างการใช้งาน Optional Positional Argument Optional ดังภาพ

```

1 void main() {
2     test5("Hello Dart", "M");
3     test5("Hello Dart", 23, "M");
4 }
5
6 void test5(name, [age, gender]) {
7     print("name is $name age is $age gender is $gender");
8 }

```

ภาพประกอบ 3.37 ตัวอย่างการใช้งาน Optional Positional Argument Optional

ที่มา : ฅนปักษ์ วรรณตรง (2564 : ...)

```

name is Hello Dart age is M gender is null
name is Hello Dart age is 23 gender is M

```

ภาพประกอบ 3.38 ผลลัพธ์การใช้งาน Optional Positional Argument Optional

ที่มา : ฅนปักษ์ วรรณตรง (2564 : ...)

Optional Named Argument

คือ optional parameter ที่มีลักษณะดังนี้

- Optional Named Argument ใช้ { } ครอบ Argument ที่ต้องการให้เป็น Optional Named Argument

- Optional Named Argument ให้เรียกชื่อ Argument พร้อมทั้งกำหนดค่าด้วย สามารถสลับตำแหน่งกันได้ จะต่างกับ Optional Positional Argument ที่ต้องส่งค่าในพารามิเตอร์ อิงตามตำแหน่งนั้น ๆ (ลุงวิศวกร สอนคำนวณ, 2564 : 66-75)

ตัวอย่างการใช้งาน Optional Named Argument ดังภาพ

```
1 void main() {  
2   test6("Hello dart", gender: "M");  
3   test6("Hello Dart", age: 22, gender: "M");  
4 }  
5  
6 void test6(name, {age, gender}) {  
7   print("name is $name age is $age gender is $gender");  
8 }
```

ภาพประกอบ 3.39 ตัวอย่างการใช้งาน Optional Named Argument

ที่มา : ณปภัช วรรณตรง (2564 : ...)

```
name is Hello dart age is null gender is M  
name is Hello Dart age is 22 gender is M
```

ภาพประกอบ 3.40 ผลลัพธ์การใช้งาน Optional Named Argument

ที่มา : ณปภัช วรรณตรง (2564 : ...)

## List

ลิสต์ (List) คือ รายการข้อมูลที่สามารถจัดเรียงต่อเนื่องกันอยู่ในรูปแบบของอาร์เรย์ สามารถกำหนดชนิดของข้อมูลที่จัดเก็บได้ทั้งแบบข้อความ ตัวเลข และอื่น ๆ ในการใช้งานพื้นฐานทั่วไปแบ่งออกได้ 2 แบบ Java (เอกรินทร์ วทัญญเลิศสกุล, 2563 : 129)

1. ลิสต์แบบขนาดคงที่ Fixed-length (Array) คือ ลิสต์ที่มีจำนวนของสมาชิกที่มีจำนวนคงที่แน่นอน มีรูปแบบที่ใช้ในการประกาศค่าดังนี้

```
var <ชื่อของลิสต์> = new List (ขนาด)
```

การกำหนดค่าให้แก่ตัวลิสต์สามารถทำได้โดยการระบุหมายเลขอินเด็กซ์ (Index) ประจำตำแหน่งของลิสต์ โดยมีรูปแบบดังนี้

```
<ชื่อของลิสต์> [เลขอินเด็กซ์] = ค่าที่ต้องการกำหนดให้
```

ตัวอย่างการใช้งาน Fixed-length (Array) ดังภาพ

```
1 void main() {
2 // ตัวเลข
3 List<int> list1 = [1,2,3,4,5,6,7,8,9];
4 // ตัวอักษร
5 List<String> list2 = ['q','e','t','u','o','a','d','g','j','l'];
6 print(list1);
7 print(list2);
8 }
```

ภาพประกอบ 3.41 ตัวอย่างการใช้งาน Fixed-length (Array)

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[q, e, t, u, o, a, d, g, j, l]
```

ภาพประกอบ 3.42 ผลลัพธ์การใช้งาน Fixed-length (Array)

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

2. ลิสต์แบบปรับเปลี่ยนขนาดได้ Growable (Link List) เป็นลิสต์ที่สามารถปรับเปลี่ยนขนาดหรือจำนวนของสมาชิกในลิสต์ได้ตลอดเวลา จะมีรูปแบบในการประกาศค่าดังนี้

```
var <ชื่อของลิสต์> = [ ]
```

สำหรับการกำหนดค่าและใช้งานสามารถใช้รูปแบบเดียวกันกับลิสต์แบบที่มีจำนวนสมาชิกคงที่ส่วนคุณสมบัติสำคัญที่ใช้งานเกี่ยวกับลิสต์มีรายละเอียดดังตาราง

ตาราง 3.3 ตารางคุณสมบัติสำคัญที่ใช้งานเกี่ยวกับลิสต์

คุณสมบัติ	คำอธิบาย	ชนิดข้อมูล
First	คืนค่าข้อมูลสมาชิกในลิสต์รายการลำดับแรก	ตามชนิดที่จัดเก็บ
isEmpty	คืนค่า จริง เมื่อไม่มีสมาชิกในลิสต์	ตรรกะ
isNotEmpty	คืนค่า จริง เมื่อมีสมาชิกในลิสต์ อย่างน้อย 1 ข้อมูล	ตรรกะ
length	คืนค่าขนาดหรือจำนวนสมาชิก	เลขจำนวนเต็ม
reversed	ชนิดข้อมูล ตามชนิดที่จัดเก็บ	ลิสต์

```
1 void main() {
2     List<String> growable = [];
3     print("growable: $growable");
4 }
```

ภาพประกอบ 3.43 ตัวอย่างการใช้งาน Growable (Link List)

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

```
growable: []
```

ภาพประกอบ 3.44 ผลลัพธ์การใช้งาน Growable (Link List)

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

## 1. การจัดการข้อมูลในลิสต์

สามารถใช้งานผ่านเมธอดสำคัญดังนี้ (เอกรินทร์ วทัญญูเลิศสกุล, 2563 : 130)

1.1 การเพิ่มรายการในลิสต์ โดยจะเพิ่มเข้าไปต่อในลำดับท้ายของลิสต์ สามารถใช้ได้เฉพาะลิสต์ แบบที่สามารถปรับเปลี่ยนขนาดได้เท่านั้น มีรูปแบบดังนี้

`<ชื่อลิสต์>.add(ข้อมูล)`

ตัวอย่างการเพิ่มรายการในลิสต์ ดังภาพ

```
1 void main() {
2     List<String> growable = [];
3     growable.add('s');
4     growable.add('g');
5     print("growable: $growable");
6 }
```

ภาพประกอบ 3.45 ตัวอย่างการเพิ่มรายการในลิสต์

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

growable: [s, g]

ภาพประกอบ 3.46 ผลลัพธ์การเพิ่มรายการในลิสต์

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

1.2 การปรับปรุงข้อมูลในลิสต์ สามารถทำได้ด้วยวิธีการพื้นฐานเหมือนอาร์เรย์ทั่วไป โดยการระบุ หมายเลขอินเด็กซ์ของลิสต์เช่นเดียวกับการกำหนดค่า โดยมีรูปแบบดังนี้

`<ชื่อของลิสต์> [เลขอินเด็กซ์] = ค่าที่ต้องการกำหนดให้`

สำหรับการปรับปรุงข้อมูลในลิสต์อีกรูปแบบหนึ่ง เป็นการกำหนดค่าเป็นย่านโดยใช้ฟังก์ชัน `replaceRange` เพื่อกำหนดค่าภายในช่วงที่ระบุ มีรูปแบบการใช้ฟังก์ชันดังนี้

`<ชื่อของลิสต์>.replaceRange (ตำแหน่งเริ่มต้น, ตำแหน่งสิ้นสุด, [รายการลิสต์])`

เมื่อตำแหน่งเริ่มต้น คือ ตำแหน่งของสมาชิกในลิสต์ที่จะเริ่มต้นแทนค่า ตำแหน่งสิ้นสุด คือ ตำแหน่งสิ้นสุดของสมาชิกในลิสต์ที่จะหยุดแทนค่า รายการลิสต์ คือ รายการลิสต์ที่จะแทนค่า ซึ่งจะต้องมีจำนวนสมาชิกตรงกับที่ระบุไว้ในตำแหน่ง เริ่มต้นและสิ้นสุด



1.3 การลบรายการในลิสต์ หมายถึง การลบสมาชิกในลิสต์ มีรูปแบบการลบโดยใช้ฟังก์ชัน 4 แบบ คือ

1.3.1 remove() คือ การลบสมาชิกในลิสต์ที่ละรายการโดยระบุข้อมูลที่ต้องการลบไว้ภายในวงเล็บ หากสมาชิกมีข้อมูลซ้ำหรือตรงกัน จะลบรายการแรกสุดที่มีข้อมูลตรงกันเท่านั้น

1.3.2 removeAt() คือ การลบสมาชิกในลิสต์ที่ละรายการโดยระบุหมายเลขอินเด็กซ์ที่ต้องการลบไว้ ภายในวงเล็บ

1.3.3 removeLast() คือ การลบสมาชิกรายการสุดท้ายของลิสต์ รูปแบบการใช้งานคือ ไม่มีการกำหนด พารามิเตอร์ในขณะใช้งานฟังก์ชันนี้

1.3.4 removeRange() คือ การลบสมาชิกด้วยการระบุช่วง โดยมีพารามิเตอร์ 2 ค่า ได้แก่ อินเด็กซ์ เริ่มต้น และอินเด็กซ์สุดท้าย

## 2. รายการลิสต์

รายการลิสต์ (List of Data) คือ การบริหารจัดการลิสต์ในการเขียนแอปพลิเคชันที่มีประโยชน์อย่างมาก ช่วยลดเวลาในการเขียนโปรแกรมและช่วยให้จัดการข้อมูลได้อย่างเป็นระบบ โดยมีพื้นฐานการใช้งานมาจากลิสต์ มีรูปแบบในการประกาศการใช้งานดังนี้ (เอกรินทร์ วัฑฒญเลิศสกุล, 2563 : 131)

List <ชนิดข้อมูล> <ชื่อรายการลิสต์>() - [รายการลิสต์]

List.insert คือ การแทรกข้อมูลภายใน List โดยการแทรกต้องระบุดังนี้

ชื่อตัวแปร.insert(index, ข้อมูลที่ต้องการแทรก)

ตัวอย่างการแทรกข้อมูลภายใน List ดังภาพ

```

1 void main() {
2     List<String> growable = [];
3     growable.add('s');
4     growable.add('g');
5     growable.insert(1, 'q');
6     print("growable: $growable");
7 }

```

ภาพประกอบ 3.47 ตัวอย่างการแทรกข้อมูลภายใน List

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

```
growable: [s, q, g]
```

ภาพประกอบ 3.48 ผลลัพธ์ข้อมูลภายใน List

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

List Sort คือ การจัดเรียงข้อมูลตัวเลขและตัวอักษรตามลำดับของข้อมูล

```
1 void main() {
2     var sort = ['B', 'C', 'A', 'D'];
3     sort.sort();
4     print("sort is: $sort");
5 }
```

ภาพประกอบ 3.49 ตัวอย่างการจัดเรียงข้อมูลภายใน List

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

```
sort is: [A, B, C, D]
```

ภาพประกอบ 3.50 ผลลัพธ์การจัดเรียงข้อมูลภายใน List

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

การ Loop ข้อมูลออกมาใช้ร่วมกับ Inline Function มี 2 รูปแบบ

1. List for Each

```
1 void main() {  
2     var sort = ['B', 'C', 'A', 'D'];  
3  
4     sort.sort();  
5     print("sort is: $sort");  
6     sort.forEach((String value) {  
7         print("value is $value");  
8     });  
9 }
```

ภาพประกอบ 3.51 ตัวอย่างการใช้งาน List for Each

ที่มา : ฅนปักษ์ วรณตรง (2564 : ...)

```
sort is: [A, B, C, D]  
value is A  
value is B  
value is C  
value is D
```

ภาพประกอบ 3.52 ผลลัพธ์การใช้งาน List for Each

ที่มา : ฅนปักษ์ วรณตรง (2564 : ...)

## 2. List for Loop

```

1 void main() {
2   var sort = ['B', 'C', 'A', 'D'];
3
4   sort.sort();
5   print("sort is: $sort");
6   for (String v in sort) {
7     print("v is $v");
8   }
9 }

```

ภาพประกอบ 3.53 ตัวอย่างการใช้งาน List for Loop

ที่มา : ณปภัช วรรณตรง (2564 : ...)

```

sort is: [A, B, C, D]
v is A
v is B
v is C
v is D

```

ภาพประกอบ 3.54 ผลลัพธ์การใช้งาน List for Loop

ที่มา : ณปภัช วรรณตรง (2564 : ...)

## Null Safety

Null Safety ฟีเจอร์ใหม่ในภาษา Dart ตั้งแต่เวอร์ชัน 2.12 ขึ้นไป Null Safety คือ การตรวจสอบตัวแปรที่เป็นค่า null ถ้าเอา ตัวแปร null ไปใช้งาน จะแสดง Error ตั้งแต่ตอนเขียนโค้ดทันที โดยที่ยังไม่ได้รันโปรแกรม เพราะฉะนั้นต้องกำหนดค่าให้ตัวแปรด้วย (ลุงวิศวกร สอนคำนวณ, 2564 : 34-40)

ถ้าจะอนุญาตให้ตัวแปรนั้น เป็นค่า null ได้ ให้ใส่เครื่องหมาย ? หลังชนิดของตัวแปร (Data Type)

ตัวอย่างการใช้งานการประกาศตัวแปรให้สามารถมีค่าเป็น Null ได้ ดังภาพ

```
1 void main() {
2     int? a;
3     String? b;
4     print(a);
5     print(b);
6 }
```

ภาพประกอบ 3.55 ตัวอย่างการใช้งานการประกาศตัวแปรให้สามารถมีค่าเป็น Null ได้  
ที่มา : ฌปภัช วรรณตรง (2564 : ...)

```
null
null
```

ภาพประกอบ 3.56 ผลลัพธ์การใช้งานการประกาศตัวแปรให้สามารถมีค่าเป็น Null ได้  
ที่มา : ฌปภัช วรรณตรง (2564 : ...)

Null Safety เมื่อใช้งานกับตัวแปร List มี 4 รูปแบบ

- ตัวแปรห้ามเป็น null และสมาชิกภายในตัวแปรห้ามมี null
- ตัวแปรห้ามเป็น null แต่สมาชิกภายในตัวแปรมี null ได้
- ตัวแปรเป็น null ได้ แต่สมาชิกภายในตัวแปรห้ามมี null
- ตัวแปรเป็น null ได้ และสมาชิกภายในตัวแปรมี null ได้

ตัวอย่างการประกาศตัวแปร List ดังภาพ

```

1 void main() {
2     List<String> list1; // ห้ามเป็น null และสมาชิกใน List ห้ามมีค่า null
3     List<String?> list2; // ห้ามเป็น null แต่สมาชิกใน List มีค่า null ได้
4     List<String>? list3; // null ได้ แต่สมาชิกใน List ห้ามมีค่า null
5     List<String?>? list4; // null ได้ และสมาชิกใน List มีค่า null ได้
6     list1 = [];
7     list2 = [null, 'Hello Dart'];
8     list3 = ['A'];
9     list4 = ['Hello Dart', null];
10    print(list1);
11    print(list2);
12    print(list3);
13    print(list4);
14 }

```

ภาพประกอบ 3.57 ตัวอย่างการประกาศตัวแปร List

ที่มา : ณปภัช วรรณตรง (2564 : ...)

```

[]
[null, Hello Dart]
[A]
[Hello Dart, null]

```

ภาพประกอบ 3.58 ผลลัพธ์การใช้งานการประกาศตัวแปร List

ที่มา : ณปภัช วรรณตรง (2564 : ...)

Null safety เมื่อใช้งานกับฟังก์ชัน

```

1 void main() {
2
3   int? name; // Nullable variable
4   if (name != null) {
5     print('name is test');
6   } else {
7     print('name is $name');
8   }
9
10 }
```

ภาพประกอบ 3.59 ตัวอย่างการใช้งาน Null safety เมื่อใช้งานกับฟังก์ชัน

ที่มา : ฅนปักษ์ วรรณตรง (2564 : ...)

name is null

ภาพประกอบ 3.60 ผลลัพธ์การใช้งาน Null Safety เมื่อใช้งานกับฟังก์ชัน

ที่มา : ฅนปักษ์ วรรณตรง (2564 : ...)

## บทสรุป

สำหรับเนื้อหาที่กล่าวมาทั้งหมดในบทนี้ จะพูดถึงการปูพื้นฐานภาษา Dart เพื่อให้ผู้เรียนมีพื้นฐานนำไปพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่ด้วย Flutter ได้ โดยมีเนื้อหาการเขียนโปรแกรมภาษา Dart ดังนี้

1) ภาษา Dart คืออะไร 2) ประวัติภาษา Dart 3) ชนิดตัวแปรในภาษา Dart 4) ตัวดำเนินการทางคณิตศาสตร์ (Operator) ที่ใช้ในภาษา Dart 5) การใช้ If/Else, Loop, Switch Case ในภาษา Dart 6) การเขียน Function ในภาษา Dart 7) List คือ รายการข้อมูลที่สามารถจัดเรียงต่อเนื่องกันอยู่ในรูปแบบของอาร์เรย์ และ 8) Null Safety คือ การตรวจสอบตัวแปรที่เป็นค่า null ในภาษา Dart

### เอกสารอ้างอิง

จิราวุธ วารินทร์. (2564). **พัฒนาโมบายล์แอปด้วย Flutter + Dart**. กรุงเทพฯ : สำนักพิมพ์ชิมพลิฟาย

ปัญญา ปะลีละเตสัง. (2566). **พัฒนาแอปแบบ Multi-Platform ด้วย Flutter โดยใช้ภาษา Dart**.

กรุงเทพฯ : ซีเอ็ดดูเคชั่น.

ลุงวิศวกร สอนคำนวณ. (2564). **Basic Dart Programming by Uncle Engineer**. Flutter 2

Bootcamp 2021 - Uncle Engineer.

อนุชิต ชโลธร. (2566). **สูตรลัด Dart 3**. กรุงเทพฯ : สำนักพิมพ์ก๊อปปวาง.

เอกรินทร์ วัญญูเลิศสกุล. (2563). **พัฒนา Mobile App ด้วย Flutter & Dart**. กรุงเทพฯ :

โปรวิชั่น.



## บทที่ 4

### การสร้างแอปพลิเคชันคำนวณ

หลังจากผู้เรียนได้เรียนรู้การสร้างโปรเจกต์ Flutter เบื้องต้น Widget เบื้องต้น เรียนรู้ภาษา Dart ที่ใช้พัฒนาพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่ด้วย Flutter ในบทนี้ผู้เรียนจะได้เรียนรู้ การเขียนคำสั่ง เพื่อลดขนาดและลบไฟล์ที่ไม่จำเป็นก่อนส่งไฟล์ให้ผู้อื่น การกำหนด Layout แบบ คอลัมน์ในแอปพลิเคชัน การสร้างแอปบาร์ (AppBar) และกำหนดค่าคุณสมบัติแอปบาร์ ขั้นตอน การใส่รูปภาพในแอปพลิเคชัน การสร้าง TextField กำหนดค่าคุณสมบัติใน TextField การตกแต่ง TextField การใส่ปุ่ม กำหนดค่าคุณสมบัติในปุ่ม การตกแต่งปุ่มด้วยการกำหนดสีปกติ และกำหนดด้วยการใช้ colorpicker การใส่ SizedBox กำหนดคุณสมบัติ SizedBox การกำหนดขอบในแอปพลิเคชัน ด้วย Padding การทำให้หน้าจอแอปพลิเคชันสามารถ Scroll ขึ้นลงด้วยการใช้ ListView การสร้าง ฟังก์ชันคำนวณในปุ่ม และเรียนรู้การกำหนดค่าและการใช้ฟังก์ชัน initState() และ setState() ดัง รายละเอียดต่อไปนี้

#### Layout Widget

Layout Widget เป็นวิดเจ็ตซึ่งทำหน้าที่จัดตำแหน่งวิดเจ็ตอื่น ๆ บนหน้าจอ สามารถแบ่งได้เป็น 2 ประเภท ดังนี้

ประเภทที่ 1 Single-child layout widgets คือ วิดเจ็ตที่ทำหน้าที่จัดตำแหน่งวิดเจ็ตอื่น ๆ โดยภายในวิดเจ็ตประเภทนี้จะมีเพียงวิดเจ็ตย่อยเพียงวิดเจ็ตเดียว เช่น Container, Padding และ Center เป็นต้น

ประเภทที่ 2 Multi-child layout widgets คือ วิดเจ็ตที่ทำหน้าที่จัดตำแหน่งวิดเจ็ตอื่น ๆ โดยภายในวิดเจ็ตจะประกอบไปด้วยหลาย ๆ วิดเจ็ตภายในได้ เช่น Row, Column และ Stack เป็นต้น (จีราวุธ วารินทร์, 2564 : 206)

#### กำหนด Layout

##### 1. สร้างโปรเจกต์ Calculate

1.1 เข้าไปยัง Folder ที่ได้สร้างไว้สำหรับเก็บไฟล์งาน > เปิดหน้าต่าง cmd > คัดลอก Path ของ Folder ที่ต้องการจะสร้างงาน หรือ พิมพ์ cmd ในช่อง Path ของ Folder เพื่อทำการเปิดหน้าต่างสำหรับใช้คำสั่งของ Flutter

เปิดหน้าต่าง cmd และใช้คำสั่ง cd ตามด้วย Path ที่ได้ทำการคัดลอกมา จากนั้นกด Enter

```
Microsoft Windows [Version 10.0.19043.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\U21H1>cd "C:\Users\U21H1\Desktop\flutter"
C:\Users\U21H1\Desktop\flutter>
```

ภาพประกอบ 4.1 เปิดหน้าต่าง cmd และใช้คำสั่ง cd

ที่มา : ณปภัช วรรณตรง (2564 : 3)

เปิด Folder ที่ต้องการจะสร้างงาน และให้ทำการพิมพ์ cmd ในช่อง Path ของ Folder



ภาพประกอบ 4.2 เปิดหน้าต่าง cmd โดยการพิมพ์ cmd ในช่อง Path

ที่มา : ณปภัช วรรณตรง (2564 : 3)

```
Microsoft Windows [Version 10.0.19043.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\U21H1\Desktop\flutter>
```

ภาพประกอบ 4.3 เปิดหน้าต่าง cmd โดยการพิมพ์ cmd ในช่อง Path (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 3)

1.2 เมื่อเข้าไปยัง Folder ที่ใช้สำหรับไว้เก็บไฟล์งานแล้ว ให้ทำการพิมพ์คำสั่ง flutter create calculate เพื่อทำการสร้างไฟล์ Project Calculate

```
C:\Users\U21H1>cd "C:\Users\U21H1\Desktop\flutter"
C:\Users\U21H1\Desktop\flutter>flutter create calculate
Creating project calculate...
```

ภาพประกอบ 4.4 พิมพ์คำสั่งสำหรับสร้างโปรเจกต์ Calculate

ที่มา : ฌปภัช วรรณตรง (2564 : 4)

1.3 เมื่อทำการสร้างไฟล์งานเสร็จแล้วจะมี Folder ปรากฏขึ้น

```
All done!
You can find general documentation for Flutter at: https://docs.flutter.dev/
Detailed API documentation is available at: https://api.flutter.dev/
If you prefer video documentation, consider: https://www.youtube.com/c/flutterdev

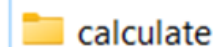
In order to run your application, type:

$ cd calculate
$ flutter run

Your application code is in calculate\lib\main.dart.
```

ภาพประกอบ 4.5 เมื่อสร้างไฟล์ Project Calculate เสร็จสิ้น

ที่มา : ฌปภัช วรรณตรง (2564 : 4)



ภาพประกอบ 4.6 เมื่อสร้างไฟล์ Project Calculate เสร็จสิ้น (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 4)

#### 1.4 ซึ่งใน Folder จะมีไฟล์ที่จำเป็นสำหรับการใช้งานไว้ให้ทั้งหมด

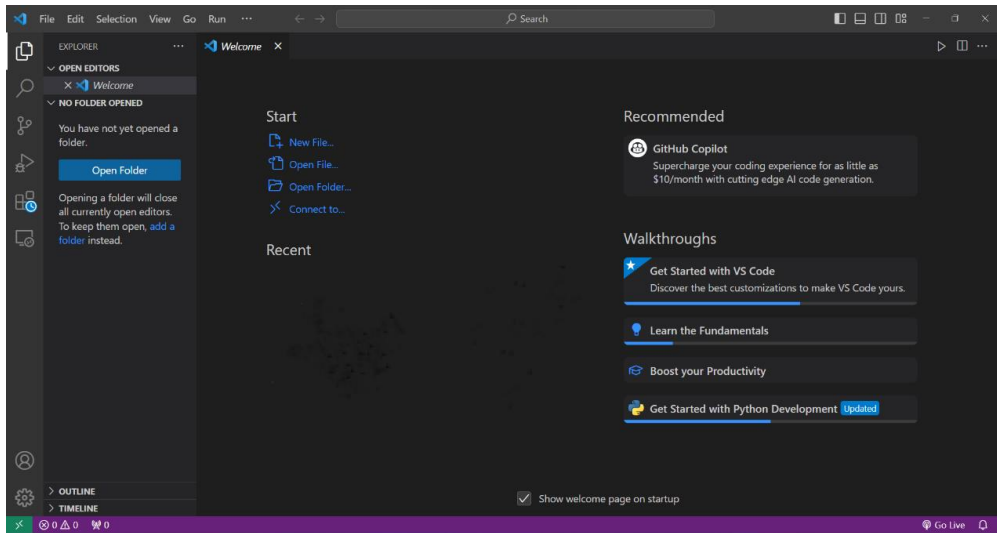
📁 .dart_tool	9/17/2023 16:14	File folder	
📁 .idea	9/17/2023 16:14	File folder	
📁 android	9/17/2023 16:15	File folder	
📁 assets	9/17/2023 16:15	File folder	
📁 build	9/17/2023 16:15	File folder	
📁 ios	9/17/2023 16:15	File folder	
📁 lib	9/17/2023 16:15	File folder	
📁 test	9/17/2023 16:15	File folder	
📁 web	9/17/2023 16:14	File folder	
📄 .gitignore	9/17/2023 16:14	Git Ignore Source ...	1 KB
📄 .metadata	9/17/2023 16:14	METADATA File	1 KB
📄 .packages	9/17/2023 16:14	PACKAGES File	3 KB
📄 calculate.iml	9/17/2023 16:14	IML File	1 KB
📄 pubspec.lock	9/17/2023 16:14	LOCK File	5 KB
📄 pubspec	9/17/2023 16:14	Yaml Source File	4 KB
📄 README	9/17/2023 16:14	Markdown Source ...	1 KB

#### ภาพประกอบ 4.7 ไฟล์ใน Folder Calculate

ที่มา : ฌปภัช วรณตรง (2564 : 4)

1.5 เมื่อทำการสร้างไฟล์ Project Calculate เสร็จสิ้นแล้ว ให้ทำการเปิดไฟล์งานใน VSCode จากนั้นให้ไปที่ main.dart แล้วทำการลบโค้ดที่ไม่ได้ใช้ออกให้หมด ให้เหลือไว้แค่ส่วนของ import

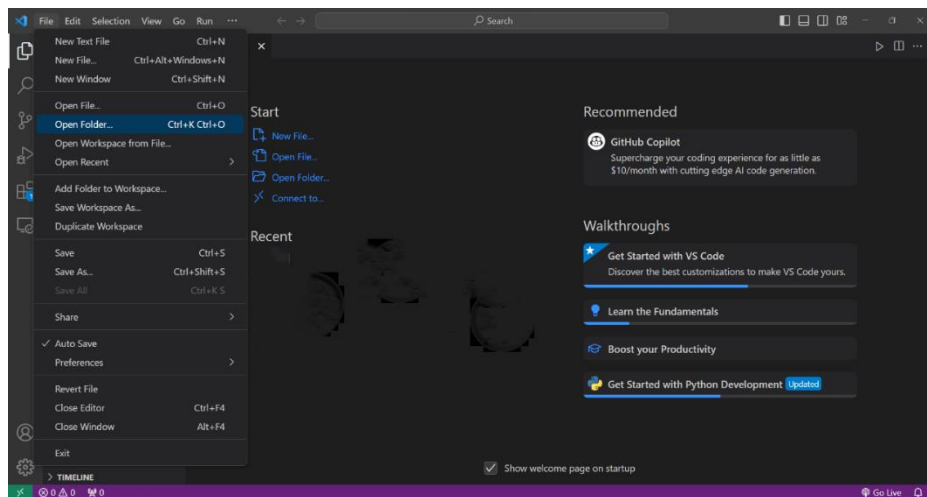
### 1.5.1 เปิดโปรแกรม VSCode



ภาพประกอบ 4.8 เปิดโปรแกรม VSCode

ที่มา : ฌปภัช วรณตรง (2564 : 5)

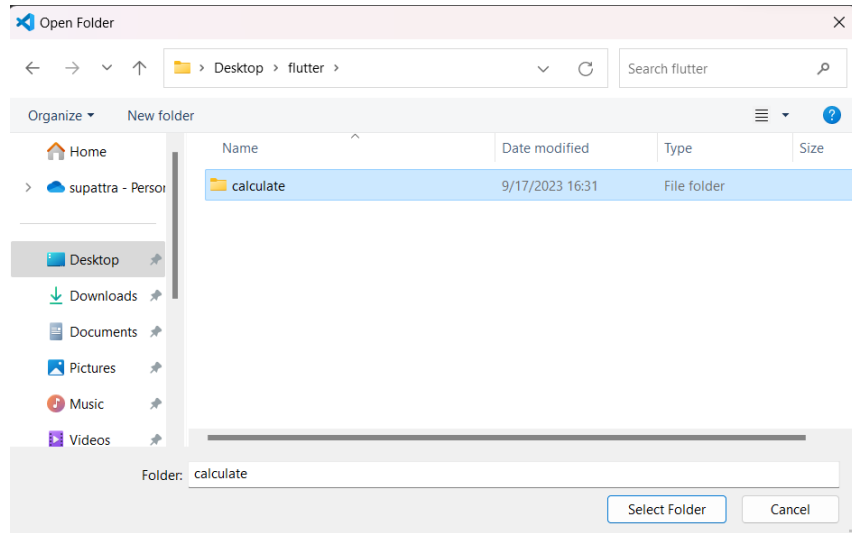
### 1.5.2 ไปที่ File > Open Folder



ภาพประกอบ 4.9 เปิดไฟล์ Project Calculate

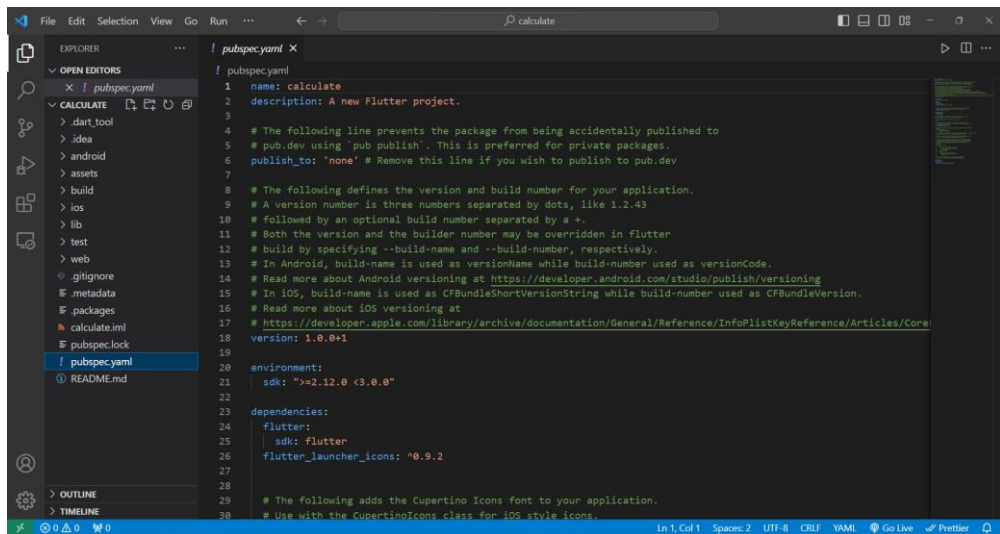
ที่มา : ฌปภัช วรณตรง (2564 : 5)

### 1.5.3 เลือกที่ Folder Calculate > จากนั้นให้กด Select Folder



ภาพประกอบ 4.10 เปิดไฟล์ Project Calculate (ต่อ)

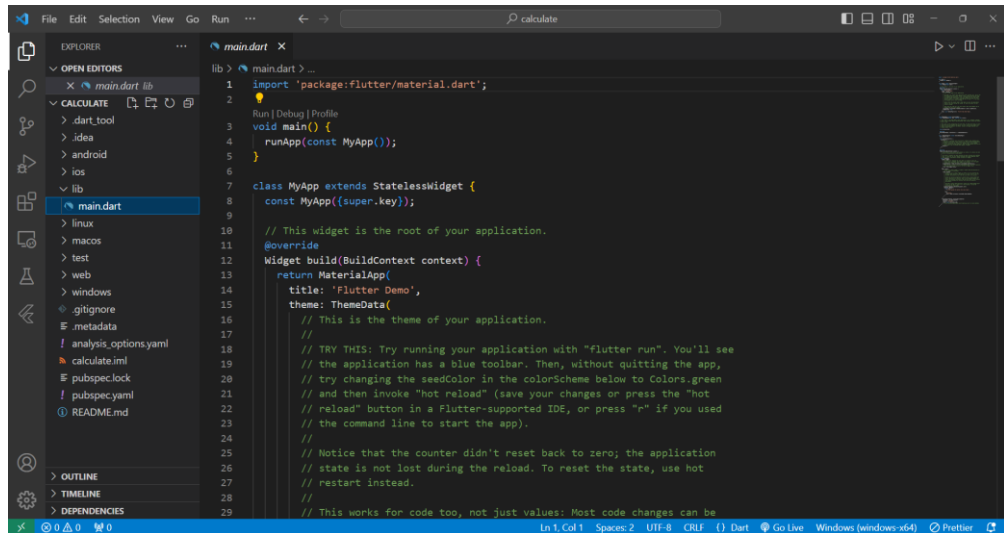
ที่มา : ฅปภัช วรรณตรง (2564 : 5)



ภาพประกอบ 4.11 เปิดไฟล์ Project Calculate (ต่อ)

ที่มา : ฅปภัช วรรณตรง (2564 : 5)

1.5.4 ไปที่ Folder main.dart เพื่อทำการลบโค้ดที่ไม่ได้ใช้ออก ให้เหลือเฉพาะโค้ดส่วน import



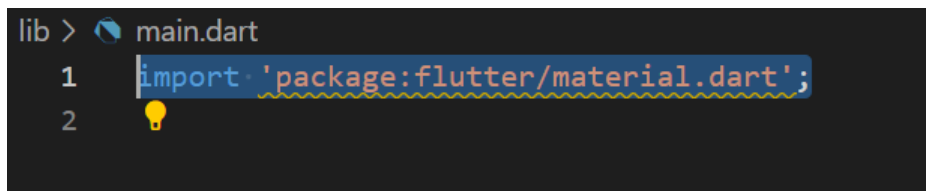
```

lib > main.dart > ...
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(const MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   const MyApp({super.key});
9
10  // This widget is the root of your application.
11  @override
12  Widget build(BuildContext context) {
13    return MaterialApp(
14      title: 'Flutter Demo',
15      theme: ThemeData(
16        // This is the theme of your application.
17        //
18        // TRY THIS: Try running your application with "Flutter run". You'll see
19        // the application has a blue toolbar. Then, without quitting the app,
20        // try changing the seedColor in the colorScheme below to Colors.green
21        // and then invoke "hot reload" (save your changes or press the "hot
22        // reload" button in a Flutter-supported IDE, or press "r" if you used
23        // the command line to start the app).
24        //
25        // Notice that the counter didn't reset back to zero; the application
26        // state is not lost during the reload. To reset the state, use hot
27        // restart instead.
28        //
29        // This works for code too, not just values: Most code changes can be

```

ภาพประกอบ 4.12 ลบโค้ดที่ไม่ได้ใช้ออกจากไฟล์ main.dart

ที่มา : ฅนปลัซ วรณตรง (2564 : 5)



```

lib > main.dart
1 import 'package:flutter/material.dart';
2

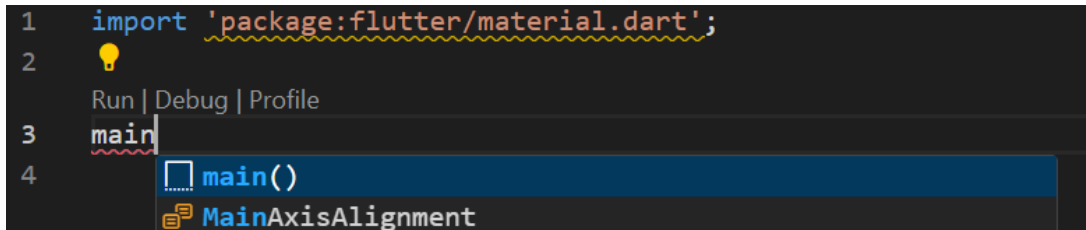
```

ภาพประกอบ 4.13 ลบโค้ดที่ไม่ได้ใช้ออกจากไฟล์ main.dart (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 5)

1.6 เมื่ออยู่ในไฟล์ main.dart ให้ทำการสร้างฟังก์ชัน void main(){} เข้าไปในไฟล์ main.dart

1.6.1 พิมพ์คำสั่ง main จากนั้นจะมีปรากฏหน้าต่างขึ้นมา ให้ทำการเลือกคำสั่ง main โดยการกด Ctrl+Space bar+Click



```

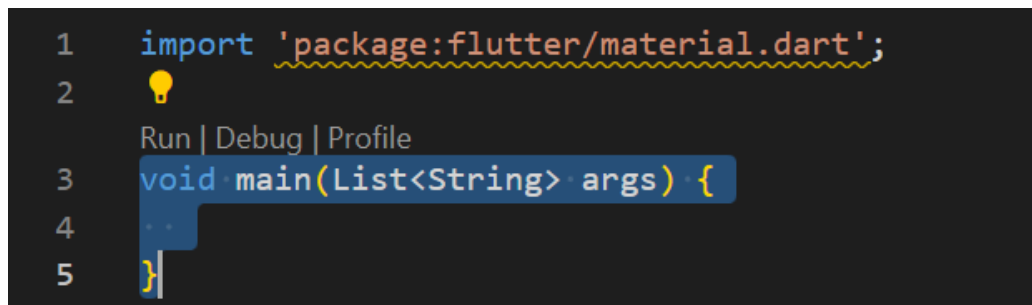
1  import 'package:flutter/material.dart';
2  ⚡
   Run | Debug | Profile
3  main
4  main()
   MainAxisAlignment

```

ภาพประกอบ 4.14 สร้างฟังก์ชัน void main(){}

ที่มา : ฌปภัช วรรณตรง (2564 : 6)

1.6.2 หลังจากทำการเลือกแล้วจะได้ชุดคำสั่งหรือฟังก์ชันดังนี้



```

1  import 'package:flutter/material.dart';
2  ⚡
   Run | Debug | Profile
3  void main(List<String> args) {
4
5  }

```

ภาพประกอบ 4.15 สร้างฟังก์ชัน void main(){} (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 6)

1.8 หลังจากที่ได้ทำการสร้างฟังก์ชัน void main(){} แล้วให้ทำการสร้าง class StatelessWidget(){}



พิมพ์คำสั่ง stl จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง Flutter stateless widget โดยการกด Click ที่คำสั่งหรือ Enter

```

1  import 'package:flutter/material.dart';
2
   Run | Debug | Profile
3  void main(List<String> args) {
4
5
6  stl
   Flutter Stateless Widget
   Flutter Stateful Widget

```

ภาพประกอบ 4.16 สร้าง class StatelessWidget(){}

ที่มา : ฌปภัช วรรณตรง (2564 : 7)

```

6  class MyWidget extends StatelessWidget {
7    const MyWidget({super.key});
8
9    @override
10   Widget build(BuildContext context) {
11     return Container();
12   }
13 }

```

ภาพประกอบ 4.17 สร้าง class StatelessWidget(){} (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 7)

1.9 เมื่อสร้าง class เสร็จแล้วให้ทำการตั้งชื่อ class ว่า MyApp ตามภาพ

```

6  class extends StatelessWidget {
7      const ({super.key});
8
9      @override
10     Widget build(BuildContext context) {
11         return Container();
12     }
13 }

```

ภาพประกอบ 4.18 ตั้งชื่อของ class StatelessWidget(){}

ที่มา : ฅนปักษ์ วรณตรง (2564 : 8)

```

6  class MyApp extends StatelessWidget {
7      const MyApp({super.key});
8
9      @override
10     Widget build(BuildContext context) {
11         return Container();
12     }
13 }

```

ภาพประกอบ 4.19 ตั้งชื่อของ class StatelessWidget(){} (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 8)

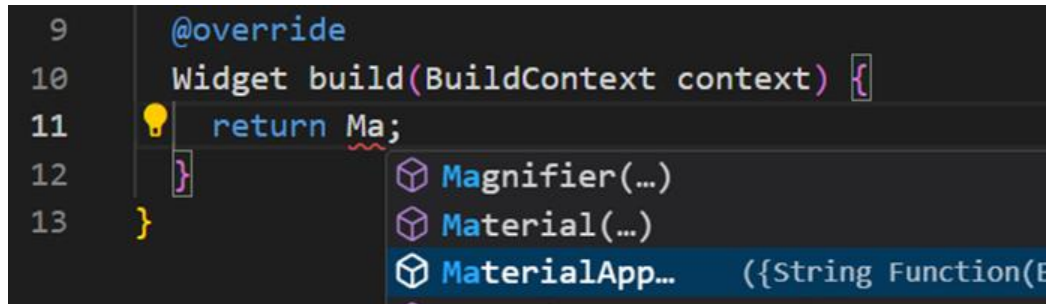
1.10 เมื่อทำการตั้งชื่อของ class แล้วให้ทำเปลี่ยนชื่อ Container เป็น MaterialApp

พิมพ์คำสั่ง mat จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง MaterialApp(...) โดยการกด Click ที่คำสั่งหรือ Enter

```

9      @override
10     Widget build(BuildContext context) {
11       return Ma;
12     }
13   }

```

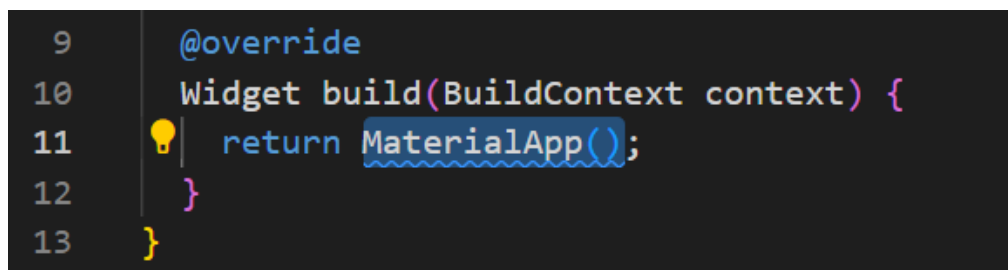


ภาพประกอบ 4.20 เปลี่ยนชื่อ class Container เป็น class MaterialApp  
ที่มา : ฅปภัช วรณตรง (2564 : 9)

```

9      @override
10     Widget build(BuildContext context) {
11       return MaterialApp();
12     }
13   }

```



ภาพประกอบ 4.21 เปลี่ยนชื่อ class Container เป็น class MaterialApp (ต่อ)  
ที่มา : ฅปภัช วรณตรง (2564 : 9)

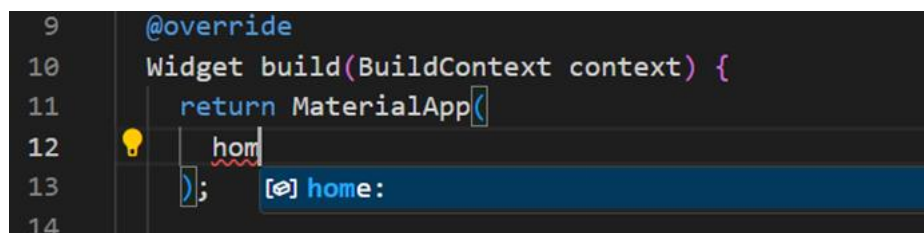
1.11 หลังจากที่ได้ทำการเปลี่ยนชื่อ class แล้ว ให้ทำการเพิ่มโค้ด home: Scaffold(), เข้าไปใน MaterialApp();

1.11.1 พิมพ์คำสั่ง home จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง home: โดยการกด Click ที่คำสั่งหรือ Enter

```

9      @override
10     Widget build(BuildContext context) {
11       return MaterialApp(
12         hom
13       );
14     }

```



ภาพประกอบ 4.22 เพิ่มโค้ด home: Scaffold(),  
ที่มา : ฅปภัช วรณตรง (2564 : 10)

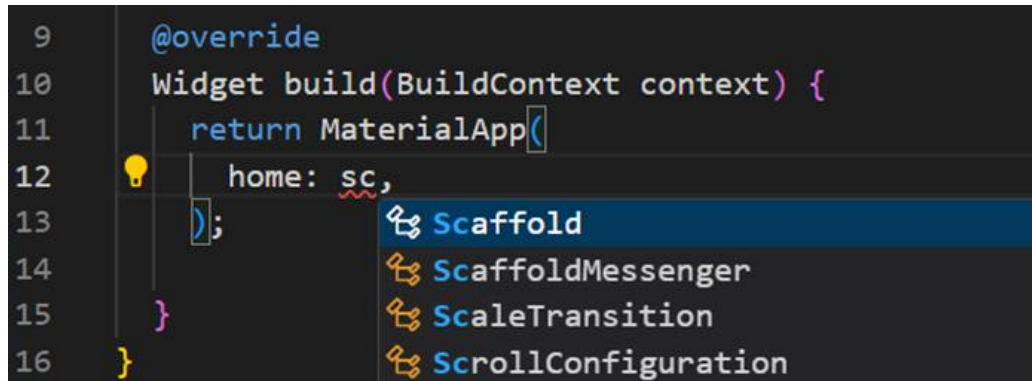
1.11.2 เมื่อได้ home: แล้วให้ทำการพิมพ์โค้ด Scaffold(), เข้าไปใน home:

1.11.3 พิมพ์คำสั่ง sc จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง Scaffold โดยการกด Click ที่คำสั่งหรือ Enter

```

9      @override
10     Widget build(BuildContext context) {
11         return MaterialApp(
12             home: sc,
13         );
14     }
15 }

```



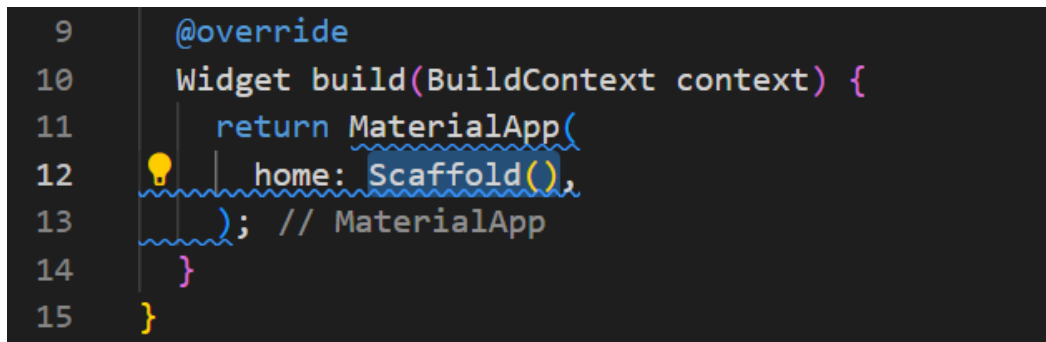
ภาพประกอบ 4.23 เพิ่มโค้ด home: Scaffold(), (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 10)

```

9      @override
10     Widget build(BuildContext context) {
11         return MaterialApp(
12             home: Scaffold(),
13         ); // MaterialApp
14     }
15 }

```



ภาพประกอบ 4.24 เพิ่มโค้ด home: Scaffold(), (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 10)

## 2. แอปบาร์ (AppBar)

แอปบาร์ (AppBar) ประกอบด้วยแถบเครื่องมือและวิดเจ็ตอื่น ๆ ที่อาจเกิดขึ้น เช่น TabBar และ FlexibleSpaceBar ซึ่งโดยปกติแล้วแอปบาร์จะทำหน้าที่เป็นที่อยู่ของวิดเจ็ตอื่น ๆ จึงตอบสนองการใช้งานมากกว่าหนึ่งอย่าง แอปบาร์จะถูกติดตั้งใช้งานไว้ใน Scaffold ด้วยคุณสมบัติ Scaffold.appBar โดยจะปรากฏเป็นแถบวิดเจ็ตที่ตำแหน่งด้านบนของหน้าจอ และภายในแอปบาร์ยังมีการจัดวางเครื่องมือต่าง ๆ

แอปบาร์สามารถกำหนดค่าคุณสมบัติต่าง ๆ ที่มีผลต่อการทำงานของแอปพลิเคชันที่สำคัญมี

2.1 Title เป็นการกำหนดข้อความในส่วนหัวข้อหลัก โดยสามารถใส่ข้อความหรือสัญลักษณ์ต่าง ๆ เพื่อบ่งบอกให้ทราบถึงหน้าที่หรือฟังก์ชันการทำงานของแอปพลิเคชันในหน้าจอนี้

2.2 Background Color เป็นการกำหนดสีของพื้นหลัง ซึ่งหมายถึงสีของแถบแอปบาร์

2.3 Actions เป็นตำแหน่งการแสดงผลที่ขอบทางด้านขวาของแอปบาร์ โดยปกติมักใช้เพื่อให้เกิดการทำสิ่งต่าง ๆ ตอบสนองตามผู้ใช้งาน เช่น การแสดงไอคอน ข้อความ สัญลักษณ์ ปุ่มกด และดรอปดาวน์ (Drop Down) เป็นต้น

2.4 Automatically imply Leading เป็นคุณสมบัติในการกำหนดการแสดงผลของตัวนำทางแบบอัตโนมัติจะปรากฏขึ้นเมื่อมีการเรียกใช้งานฉากแสดงผลต่อ ๆ กันไป โดยจะแสดงเป็นสัญลักษณ์ เพื่อให้สามารถย้อนกลับไปจากหรือหน้าจอใช้งานก่อนหน้าได้ ซึ่งสามารถกำหนดค่าคุณสมบัติเป็นแบบตรรกะ คือ จริงหรือเท็จ ค่าตั้งต้นจะมีค่าเป็น true คือ เปิดการแสดงผลของตัวนำทางและหากกำหนดค่าเป็น false จะเป็นการปิดการแสดงผลของตัวนำทางและส่งผลให้ไม่มีช่องว่างในส่วน Leading การแสดงผลของส่วนที่เป็น Title จึงมีพื้นที่ในการแสดงผลมากขึ้น แต่สำหรับหน้าแรกหรือหน้าจอหลักนั้นเป็นจุดเริ่มต้นของการเรียกใช้งานจะไม่มีสัญลักษณ์ของตัวนำทางขึ้น

2.5 Bottom คือ การแสดงผลที่ส่วนแฉกกลางของ Title ใช้เพื่อแสดงส่วนของคำอธิบายหรือรายละเอียดบางอย่างที่ต้องการแจ้งให้ผู้ใช้งานทราบ โดยต้องกำหนดค่าด้วยวิดเจ็ต PreferredSize เพื่อกำหนดขนาดของแถบที่เพิ่มขึ้นในแถวที่ 2 ของแอปบาร์ (เอกรินทร์ วัฑฒญเลิศสกุล, 2563 : 74-75)

### 3. ใส่ AppBar Widget

เมื่อทำการเพิ่มโค้ด `home: Scaffold()`, เสร็จสิ้นแล้ว ในส่วนของ `Scaffold()` ให้ใส่โค้ด `appBar: AppBar(title: Text("แอปพลิเคชันคำนวณ"),)`,

3.1 พิมพ์คำสั่ง `app` จากนั้นจะปรากฏหน้าต่างต่าง ให้ทำการเลือกคำสั่ง `AppBar`: โดยการกด `Click` ที่คำสั่งหรือ `Enter`

```

9      @override
10     Widget build(BuildContext context) {
11         return MaterialApp(
12             home: Scaffold(
13                 app
14             ),
15             appBar:
16             ); // M

```

ภาพประกอบ 4.25 เพิ่มโค้ด appBar: ,  
ที่มา : ฌปภัช วรรณตรง (2564 : 11)

3.2 เมื่อได้ appBar: , แล้ว ให้ทำการพิมพ์ AppBar เข้าไป โดยพิมพ์คำสั่ง app จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง AppBar(...) โดยการกด Click ที่คำสั่งหรือ Enter

```

9      @override
10     Widget build(BuildContext context) {
11         return MaterialApp(
12             home: Scaffold(
13                 appBar: App
14             ), // Scaffold
15             ); // MaterialA
16     }
17 }

```

ภาพประกอบ 4.26 เพิ่ม AppBar() เข้าไปใน Tag appBar: ,  
ที่มา : ฌปภัช วรรณตรง (2564 : 11)

```

9      @override
10     Widget build(BuildContext context) {
11         return MaterialApp(
12             home: Scaffold(
13                 appBar: AppBar(),
14             ), // Scaffold
15         ); // MaterialApp
16     }
17 }

```

ภาพประกอบ 4.27 เพิ่ม AppBar() เข้าไปใน Tag appBar: , (ต่อ)  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 11)

3.3 เมื่อทำการเพิ่ม AppBar() เรียบร้อยแล้ว ให้ทำการเพิ่มโค้ด title: , เข้าไป โดยพิมพ์คำสั่ง ti จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง title: โดยการกด Click ที่คำสั่งหรือ Enter

```

9      @override
10     Widget build(BuildContext context) {
11         return MaterialApp(
12             home: Scaffold(
13                 appBar: AppBar(ti),
14             ), // Scaffold
15         ); // MaterialApp
16     }
17 }

```

[x] title:  
[x] titleSpacing:  
[x] titleTextStyle:  
[x] centerTitle:

ภาพประกอบ 4.28 เพิ่ม title: , เข้าไปใน Tag AppBar()  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 11)

```

9      @override
10     Widget build(BuildContext context) {
11         return MaterialApp(
12             home: Scaffold(
13                 appBar: AppBar(title: ),),
14             ), // Scaffold
15         ); // MaterialApp
16     }
17 }

```

ภาพประกอบ 4.29 เพิ่ม title: , เข้าไปใน Tag AppBar() (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 11)

3.4 จากนั้นให้ทำการเพิ่ม Text เข้าไป โดยพิมพ์คำสั่ง Te จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง Text(...) โดยการกด Click ที่คำสั่งหรือ Enter

```

9      @override
10     Widget build(BuildContext context) {
11         return MaterialApp(
12             home: Scaffold(
13                 appBar: AppBar(title: Te),),
14             ), // Scaffold
15         ); // MaterialApp
16     }

```

Text  
TextButton  
TextButtonTheme

ภาพประกอบ 4.30 เพิ่ม Text(), เข้าไปใน Tag title: ,

ที่มา : ฌปภัช วรรณตรง (2564 : 11)



```

11     return MaterialApp(
12         home: Scaffold(
13             appBar: AppBar(title: Text(data)),
14         ), // Scaffold
15     ); // MaterialApp
16     }
17 }

```

ภาพประกอบ 4.31 เพิ่ม Text(), เข้าไปใน Tag title: , (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 11)

3.5 เมื่อทำการเพิ่ม Text() แล้ว ใน (...) ให้แก้ไขข้อความโดยใช้ Tag “...” เป็นข้อความที่ต้องการ

```

11     return MaterialApp(
12         home: Scaffold(
13             appBar: AppBar(title: Text("")),
14         ), // Scaffold
15     ); // MaterialApp
16     }
17 }

```

ภาพประกอบ 4.32 เปลี่ยนข้อความใน Text(),

ที่มา : ฌปภัช วรรณตรง (2564 : 11)

```

11     return MaterialApp(
12         home: Scaffold(
13             appBar: AppBar(title: Text("แอปคำนวณ")),
14         ), // Scaffold
15     ); // MaterialApp
16     }
17 }

```

ภาพประกอบ 4.33 เปลี่ยนข้อความใน Text(), (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 11)

```

10  @override
11  Widget build(BuildContext context) {
12    return MaterialApp(
13      home: Scaffold(
14        appBar: AppBar(title: Text("แอปคำนวณ"),),
15      ), // Scaffold
16
17    ); // MaterialApp

```

ภาพประกอบ 4.34 เพิ่มโค้ดส่วนของ Scaffold(),

ที่มา : ฅนปักษ์ วรณตรง (2564 : 11)

3.6 เมื่อได้ส่วน appBar: , แล้ว ให้ทำการเพิ่มโค้ด body: , เข้าไปใน Scaffold() ต่อจาก appBar: ,

```

11  return MaterialApp(
12    home: Scaffold(
13      appBar: AppBar(title: Text("แอปคำนวณ"),),
14      body: ,
15    ), // Scaffold
16  ); // MaterialApp
17  }
18  }

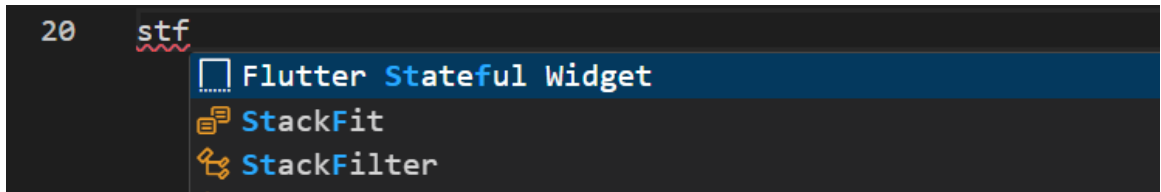
```

ภาพประกอบ 4.35 เพิ่มโค้ด body: ,

ที่มา : ฅนปักษ์ วรณตรง (2564 : 12)

3.7 เมื่อทำการสร้าง class StatelessWidget เสร็จสิ้นแล้วให้ทำการสร้าง class StatefulWidget ต่อจาก class StatelessWidget

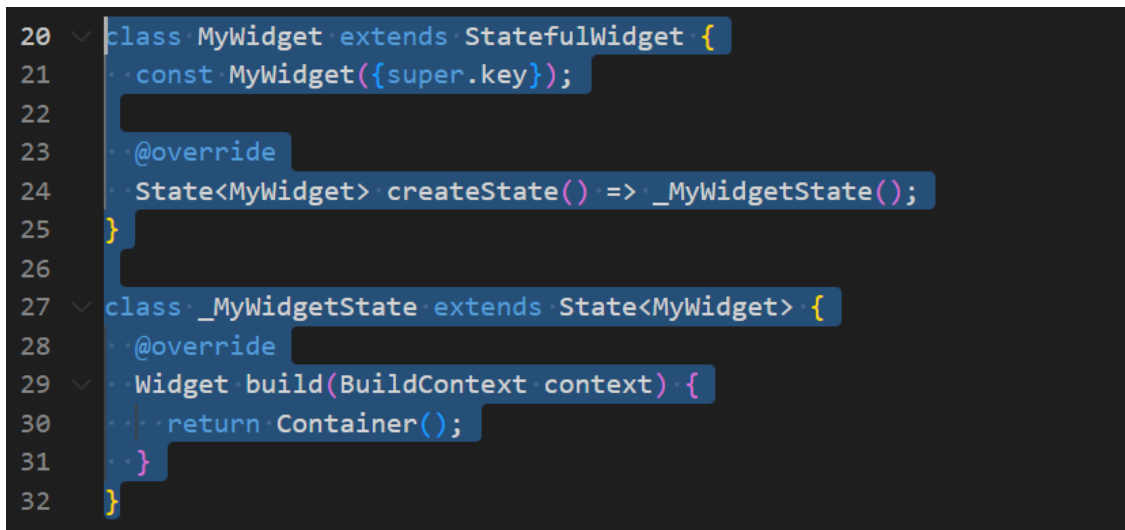
3.7.1 พิมพ์คำสั่ง stf จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง flutter: stateful widget โดยการกด Click ที่คำสั่งหรือ Enter



ภาพประกอบ 4.36 สร้าง class StatefulWidget

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 13)

3.7.2 เมื่อทำการสร้าง class StatefulWidget แล้วให้ทำการเปลี่ยนชื่อ class เป็น Home



ภาพประกอบ 4.37 สร้าง class StatefulWidget (ต่อ)

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 13)

```

20 class Home extends StatefulWidget {
21   const Home({super.key});
22
23   @override
24   State<Home> createState() => _HomeState();
25 }
26
27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Container();
31   }
32 }

```

ภาพประกอบ 4.38 สร้าง class StatefulWidget (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 13)

3.8 หลังจากนั้นให้กลับไป class StatelessWidget ในส่วนของ body: ให้ทำการเพิ่มโค้ด Home() เพื่อเรียกใช้ class Home ที่ได้ทำการสร้างไว้

```

9   @override
10  Widget build(BuildContext context) {
11    return MaterialApp(
12      home: Scaffold(
13        appBar: AppBar(title: Text("แอปคำนวณ")),
14        body: Home(),
15      ), // Scaffold
16    ); // MaterialApp
17  }
18 }
19

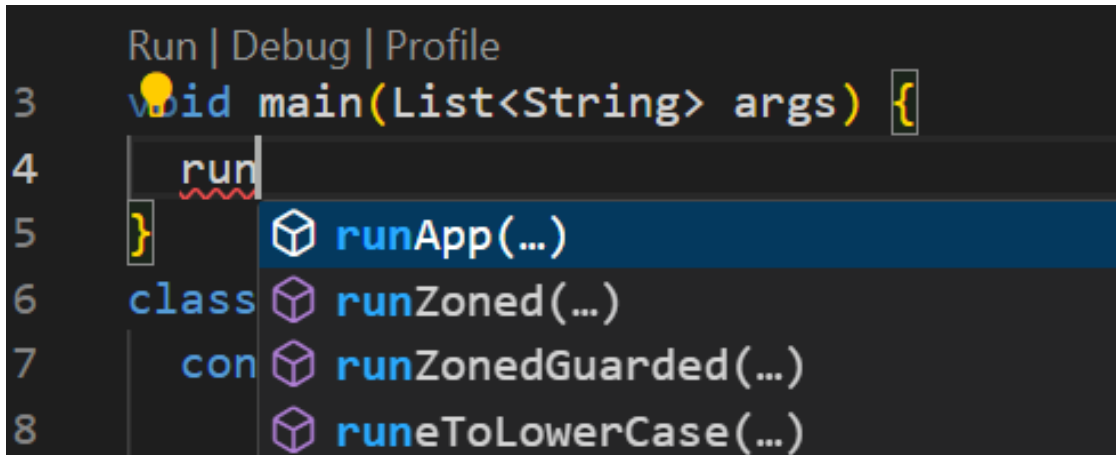
```

ภาพประกอบ 4.39 เพิ่มโค้ด Home()

ที่มา : ฌปภัช วรรณตรง (2564 : 14)

3.9 จากนั้นให้กลับไปส่วนของ class void main ให้ทำการเพิ่มโค้ด runApp(MyApp()); เพื่อเรียกใช้ class MyApp(); ที่ได้ทำการสร้างไว้ก่อนหน้า

3.9.1 พิมพ์คำสั่งการทำงานจากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง runApp(...) โดยการกด Click ที่คำสั่งหรือ Enter



```

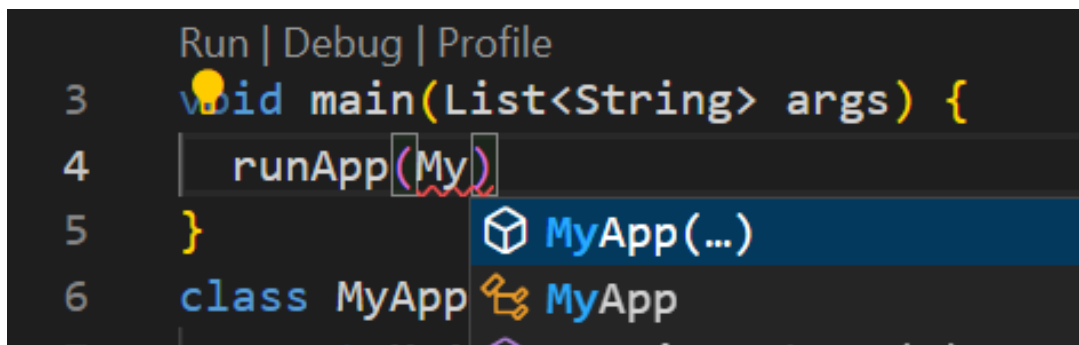
Run | Debug | Profile
3 void main(List<String> args) {
4   run
5   }
6 class
7   con
8   runApp(...)
   runZoned(...)
   runZonedGuarded(...)
   runToLowerCase(...)

```

ภาพประกอบ 4.40 เรียกใช้งาน class MyApp(){}

ที่มา : ฌปภัช วรรณตรง (2564 : 15)

3.9.2 จากนั้น ภายใน runApp(...) พิมพ์คำสั่ง My จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง MyApp(...) โดยการกด Click ที่คำสั่งหรือ Enter เพื่อเรียกใช้งาน class MyApp(){}



```

Run | Debug | Profile
3 void main(List<String> args) {
4   runApp(My)
5   }
6 class MyApp
7   MyApp
8   MyApp

```

ภาพประกอบ 4.41 เรียกใช้งาน class MyApp(){} (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 15)

```

Run | Debug | Profile
3  void main(List<String> args) {
4  |  runApp(Myapp());
5  }

```

ภาพประกอบ 4.42 เรียกใช้งาน class MyApp(){} (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 15)

3.10 กลับมาที่ class `_HomeState` ในส่วนของ return ให้ทำการแก้ไข Container เพิ่มโค้ด `Text("Home Page")`; เข้าไปแทน Container จากนั้นให้ลองทำการสั่งเปิดการทำงานของแอปพลิเคชัน

```

27  class _HomeState extends State<Home> {
28  |  @override
29  |  Widget build(BuildContext context) {
30  |  |  return Container();
31  |  }

```

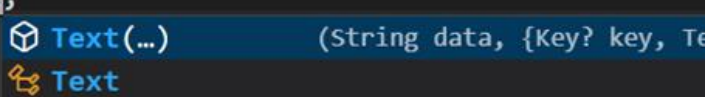
ภาพประกอบ 4.43 เพิ่มโค้ด `Text("Home Page")`;

ที่มา : ฌปภัช วรรณตรง (2564 : 16)

```

27  class _HomeState extends State<Home> {
28  |  @override
29  |  Widget build(BuildContext context) {
30  |  |  return Text;
31  |  |  }
32  |  }

```



ภาพประกอบ 4.44 เพิ่มโค้ด `Text("Home Page")`; (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 16)

เพิ่มข้อความใน Text(...) โดยใช้ Tag “ ” เป็น “Home Page”

```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Text("Home Page");
31   }
32 }

```

ภาพประกอบ 4.45 เพิ่มโค้ด Text(“Home Page”); (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 16)

```

class _HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return Text("Home Page");
  }
}

```

ภาพประกอบ 4.46 เพิ่มโค้ด Text(“Home Page”); (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 16)

3.11 เมื่อทำการเพิ่มโค้ด Text(“Home Page”); เสร็จสิ้นแล้ว ให้กลับไปหน้าต่าง cmd ทำการเข้าไปใน Folder calculate โดยการใส่คำสั่ง cd calculate

```

C:\Users\nan_s>cd calculate
C:\Users\nan_s\calculate>

```

ภาพประกอบ 4.47 สั่งการทำงาน Flutter

ที่มา : ฅนปักษ์ วรณตรง (2564 : 17)

เมื่อเข้าไปใน Folder calculate แล้วให้ทำการพิมพ์คำสั่ง flutter run จากนั้นกด Enter

```
C:\Users\nan_s>cd calculate
C:\Users\nan_s\calculate>flutter run|
```

ภาพประกอบ 4.48 สั่งการทำงาน Flutter (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 17)

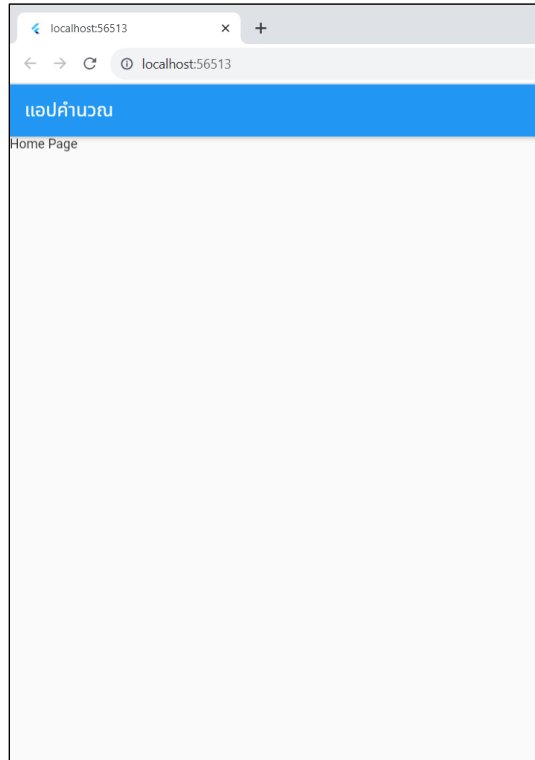
```
C:\Users\nan_s\calculate>flutter run
Connected devices:
Windows (desktop) • windows • windows-x64 • Microsoft Windows [Version 10.0.22621.2283]
Chrome (web)      • chrome • web-javascript • Google Chrome 116.0.5845.188
Edge (web)       • edge   • web-javascript • Microsoft Edge 117.0.2045.31
[1]: Windows (windows)
[2]: Chrome (chrome)
[3]: Edge (edge)
Please choose one (or "q" to quit): 2
Launching lib\main.dart on Chrome in debug mode...
Waiting for connection from debug service on Chrome... -|
```

ภาพประกอบ 4.49 สั่งการทำงาน Flutter (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 17)



เมื่อทำการสั่งการทำงานผ่านแล้ว จะปรากฏหน้าต่าง Web Browser ขึ้นมา



ภาพประกอบ 4.50 สั่งการทำงาน Flutter (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 17)

3.12 เมื่อทำการทดสอบการทำงานของ Flutter แล้ว ให้ทำการลบโค้ด Text("Home Page"); จากนั้นให้ทำการเปลี่ยนเป็น class Center(); และพิมพ์โค้ดใส่ Center();

```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Text("Home Page");
31   }
32 }
  
```

ภาพประกอบ 4.51 เพิ่มโค้ด Center();

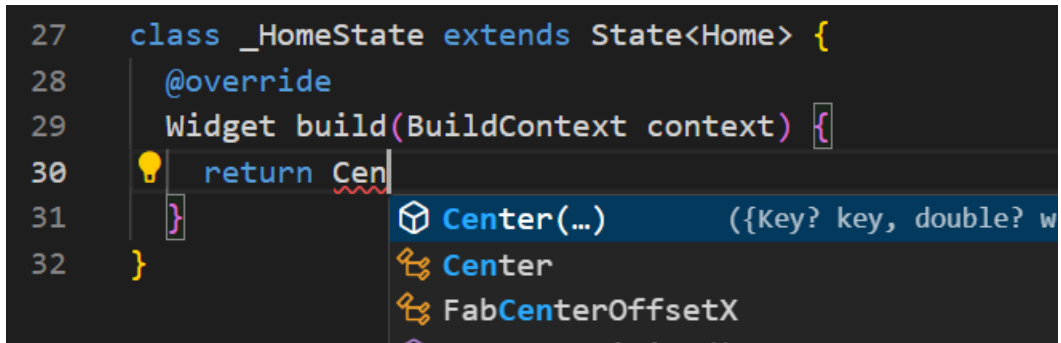
ที่มา : ณปภัช วรรณตรง (2564 : 18)

พิมพ์คำสั่ง Cen จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง Center(...) โดยการกด Click ที่คำสั่งหรือ Enter

```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Cen
31   }
32 }

```



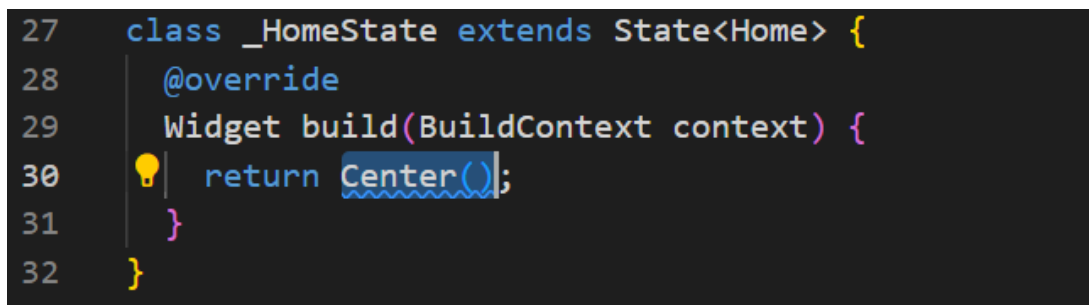
ภาพประกอบ 4.52 เพิ่มโค้ด Center(); (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 18)

```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Center();
31   }
32 }

```



ภาพประกอบ 4.53 เพิ่มโค้ด Center(); (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 18)

เมื่อได้ในส่วนของ Center(); แล้ว ให้ทำการเพิ่มโค้ดเข้าไป โดยใช้โค้ด child: ,

```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Center(
31       ch
32     ); [๑] child:
33   }
34 }

```

ภาพประกอบ 4.54 เพิ่มโค้ด child: ,  
ที่มา : ฌปภัช วรรณตรง (2564 : 18)

```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Center(
31       | child: ,
32     );
33   }
34 }

```

ภาพประกอบ 4.55 เพิ่มโค้ด child: , (ต่อ)  
ที่มา : ฌปภัช วรรณตรง (2564 : 18)

ใน child: , ให้ทำการเพิ่ม Column(...) โดยการพิมพ์คำสั่ง Col จากนั้นจะปรากฏหน้าต่างให้ทำการเลือกคำสั่ง Column(...) โดยการกด Click ที่คำสั่งหรือ Enter

```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Center(
31       child: Col,
32     );
33   }
34 }

```

ภาพประกอบ 4.56 เพิ่มโค้ด Column(...)

ที่มา : ณปภัช วรรณตรง (2564 : 18)

เมื่อทำการเพิ่ม Column เรียบร้อยแล้ว ต่อไปจะทำการเพิ่ม children: [] ที่สามารถเพิ่ม Widget ได้มากกว่า 1

```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Center(
31       child: Column(ch),
32     ); // Center
33   }
34 }

```

ภาพประกอบ 4.57 เพิ่มโค้ด children: []

ที่มา : ณปภัช วรรณตรง (2564 : 18)

ในส่วนของ children: [] ให้เพิ่มข้อความเพื่อนำมาแสดงผล โดยใช้โค้ด Text(...)

```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Center(
31       child: Column(children: [Text()],),
32     ); // Center
33   }
34 }

```

ภาพประกอบ 4.58 เพิ่มข้อความใน children: []

ที่มา : ณปภัช วรรณตรง (2564 : 18)

```

class _HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Column(children: [Text("-picture-"),Text("โปรแกรมคำนวณ")],),
    ); // Center
  }
}

```

ภาพประกอบ 4.59 เพิ่มข้อความใน children: [] (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 18)

เมื่อเพิ่มข้อความเสร็จแล้ว ให้ใส่ ; ปิด class Center() ด้วย

```
class _HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Column(children: [Text("-picture-"), Text("โปรแกรมคำนวณ"), ],),
    ); // Center
  }
}
```

ภาพประกอบ 4.60 เพิ่มข้อความใน children: [] (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 18)

จากโค้ดทั้งหมดจากขั้นตอนการทำที่ผ่านมา เมื่อแสดงเปรียบเทียบกับผลลัพธ์ที่ได้ Column ใน Flutter จะทำการเก็บข้อมูลที่ได้ทำการพิมพ์คำสั่งเอาไว้ในแนวตั้งเป็นชั้น ๆ ลงไปตามจำนวนโค้ดที่คั่นด้วยเครื่องหมาย Comma (,) ซึ่งในแต่ละ Column สามารถมี Widget ได้หลากหลาย ดังภาพ



ภาพประกอบ 4.61 โครงสร้างของแอปพลิเคชัน

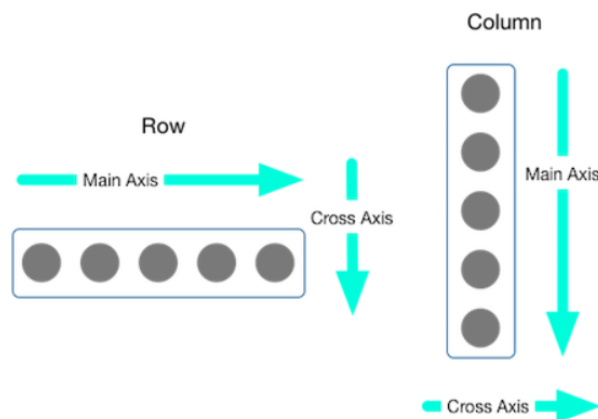
ที่มา : ฅนปักษ์ วรณตรง (2564 : 21)

#### 4. Column และ Row Widget

Column เป็นวิดเจ็ตในแบบ multi-child ใช้สำหรับจัดเรียงวิดเจ็ตในแนวตั้ง โดยวิดเจ็ตทั้งหมดที่อยู่ภายใต้ Column วิดเจ็ต จะถูกนำมากำหนดลงในพรีอเพอร์ตี้ children: (จีราวุธ วารินทร์, 2564 : 150)

Row เป็นวิดเจ็ตในแบบ multi-child ใช้สำหรับจัดเรียงวิดเจ็ตในแนวนอน การจัดรูปแบบสามารถทำได้เหมือนกับ Column แต่ต่างกันที่เป็นการแสดงผลแบบแนวนอน ขนาดของ Row จะมีขนาดเท่ากับขนาดของ Widget ที่อยู่ก่อนหน้า ยกตัวอย่างเช่น หาก Row อยู่ภายใต้หน้าจอหลักที่มีความกว้างเต็มขนาดหน้าจอ โดยปกติขนาดของ Row จะมีขนาดความกว้างเท่ากับหน้าจอ (อนุชิต ชโลธร, 2565: 54)

สามารถแสดงความแตกต่างของ Column และ Row ได้ดังภาพ



ภาพประกอบ 4.62 โครงสร้าง Column ของแอปพลิเคชัน (ต่อ)

ที่มา : Flutter (2020 : 20)

ศึกษาเพิ่มเติมเกี่ยวกับ Flutter Layout

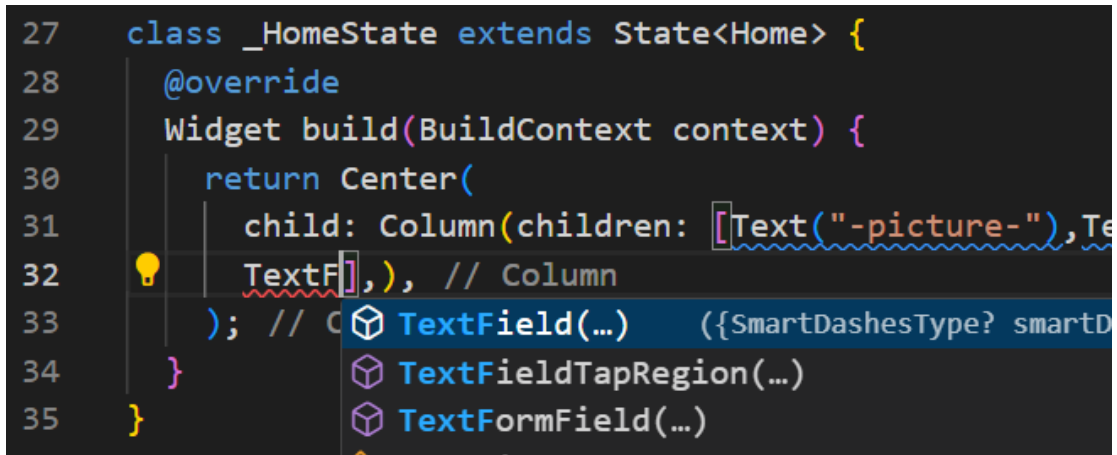
<https://docs.flutter.dev/development/ui/layout>

4.1 หลังจากทำความเข้าใจโครงสร้าง Column แล้ว ให้ทำการเพิ่มโค้ดช่องสำหรับกรอกข้อมูลโดยการใส่ Text Field ต่อจากโค้ดส่วน Text(), โดยการพิมพ์คำสั่ง TextField จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง TextField(...) โดยการกด Click ที่คำสั่งหรือ Enter

```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Center(
31       child: Column(children: [Text("-picture-"), Te
32       TextField]), // Column
33     ); // C
34   }
35 }

```



ภาพประกอบ 4.63 เพิ่ม TextField(...)

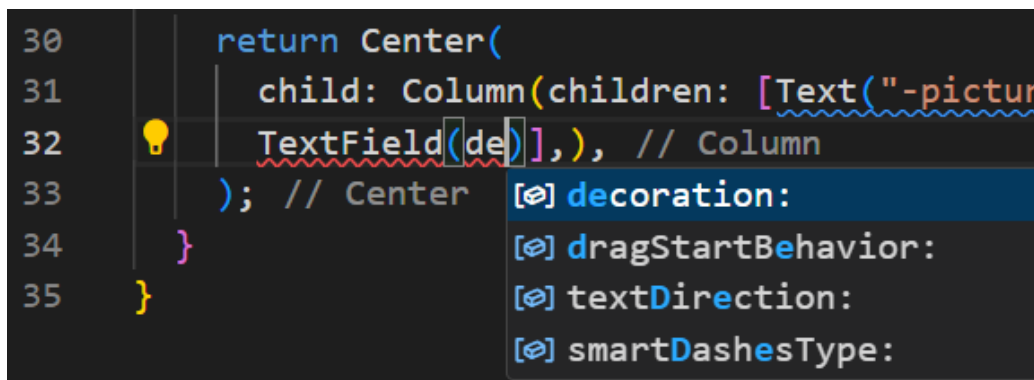
ที่มา : ฅนปักษ์ วรณตรง (2564 : 22)

ภายใน TextField(...) ให้ทำการใส่ decoration: โดยการพิมพ์คำสั่ง de จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง decoration: โดยการกด Click ที่คำสั่งหรือ Enter

```

30     return Center(
31       child: Column(children: [Text("-pictur
32       TextField(de)],), // Column
33     ); // Center
34   }
35 }

```



ภาพประกอบ 4.64 เพิ่ม decoration:

ที่มา : ฅนปักษ์ วรณตรง (2564 : 22)

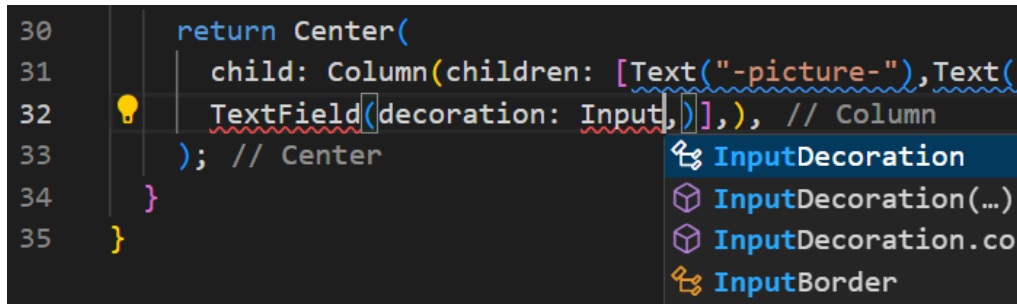


ภายใน decoration: จะทำการใส่โค้ด InputDecoration โดยการพิมพ์คำสั่ง Input จากนั้น จะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง InputDecoration(...) โดยการกด Click ที่คำสั่งหรือ Enter

```

30     return Center(
31       child: Column(children: [Text("-picture-"),Text(
32         TextField(decoration: Input,)],), // Column
33     ); // Center
34   }
35 }

```



ภาพประกอบ 4.65 เพิ่ม InputDecoration

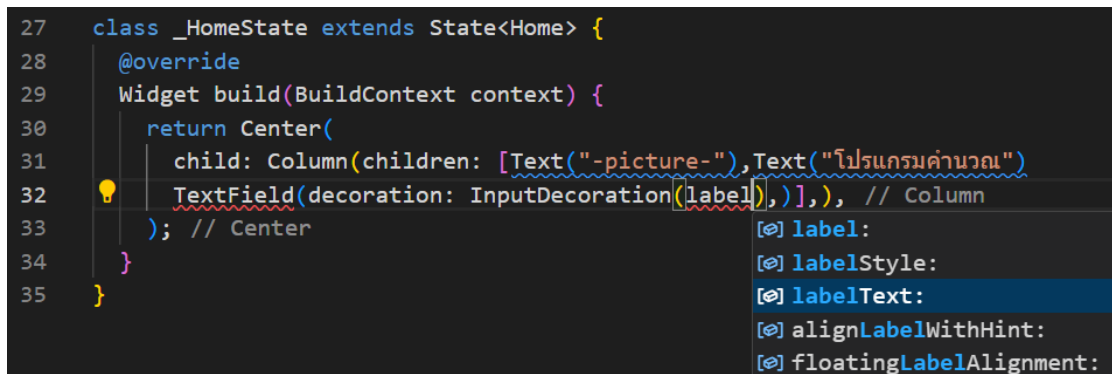
ที่มา : ฌปภัช วรรณตรง (2564 : 22)

ภายใน InputDecoration ให้ทำการใส่รอบโค้ด labelText: เพื่อแสดงข้อความ โดยการพิมพ์คำสั่ง label จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง labelText: โดยการกด Click ที่คำสั่งหรือ Enter

```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Center(
31       child: Column(children: [Text("-picture-"),Text("โปรแกรมคำนวณ")
32         TextField(decoration: InputDecoration(label)],),), // Column
33     ); // Center
34   }
35 }

```



ภาพประกอบ 4.66 เพิ่ม labelText:

ที่มา : ฌปภัช วรรณตรง (2564 : 22)

ภายใน labelText: ให้ใช้เครื่องหมาย “ ” เพื่อทำการเพิ่มข้อความที่ต้องการแสดง

```
class _HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Column(children: [Text("-picture-"),Text("โปรแกรมคำนวณ")
        TextField(decoration: InputDecoration(labelText: "จำนวนแอปเปิ้ล"),),), /
    ); // Center
  }
}
```

ภาพประกอบ 4.67 เพิ่มข้อความใน labelText:

ที่มา : ฅนปักษ์ วรณตรง (2564 : 22)

จากนั้นให้ทำการเพิ่มขอบให้กับ labelText: โดยใช้โค้ด border: โดยการพิมพ์คำสั่ง bor  
จากนั้นจะปรากฏหน้าต่างต่าง ให้ทำการเลือกคำสั่ง border: โดยการกด Click ที่คำสั่งหรือ Enter

```
@override
Widget build(BuildContext context) {
  return Center(
    child: Column(children: [Text("-picture-"),Text("โปรแกรมคำนวณ")
      TextField(decoration: InputDecoration(labelText: "จำนวนแอปเปิ้ล",bor],),), /
    ); // Center
  }
}
```

ภาพประกอบ 4.68 เพิ่มขอบให้กับ labText:

ที่มา : ฅนปักษ์ วรณตรง (2564 : 22)

เลือกขอบเป็น OutlineInputBorder() โดยการพิมพ์คำสั่ง out จากนั้นจะปรากฏหน้าต่างให้ทำการเลือกคำสั่ง OutlineInputBorder โดยการกด Click ที่คำสั่งหรือ Enter และเติม () ต่อท้าย

```
context) {
  children: [Text("-picture-"),Text("โปรแกรมคำนวณ")
  InputDecoration(labelText: "จำนวนแอปเปิ้ล",border: out)], // Column
}

OutlineInputBorder
OutlineInputBorder(...)
OutOfMemoryError
OutlinedBorder
OutlinedButton
```

ภาพประกอบ 4.69 เพิ่มขอบให้กับ labText: (ต่อ)

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 22)

```
27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Center(
31       child: Column(children: [Text("-picture-"),Text("โปรแกรมคำนวณ"),
32         TextField(decoration: InputDecoration(labelText: "จำนวนแอปเปิ้ล",
33         border: OutlineInputBorder()),)], // InputDecoration // TextField // Column
34     ); // Center
35   }
36 }
```

ภาพประกอบ 4.70 เพิ่ม TextField(...) (ต่อ)

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 22)

หากทำการติดตั้ง Prettier Extension สามารถทำการจัดเรียงโค้ดเพื่อตรวจสอบความถูกต้องของโค้ด และจัดเรียงโค้ดให้สวยงามได้ด้วยการกด Alt+Shift+F

```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Center(
31       child: Column(
32         children: [
33           Text("-picture-"),
34           Text("โปรแกรมคำนวณ"),
35           TextField(
36             decoration: InputDecoration(labelText: "จำนวนแอปเปิ้ล", border: OutlineInputBorder()),
37           ) // TextField
38         ],),); // Column // Center
39   }
40 }

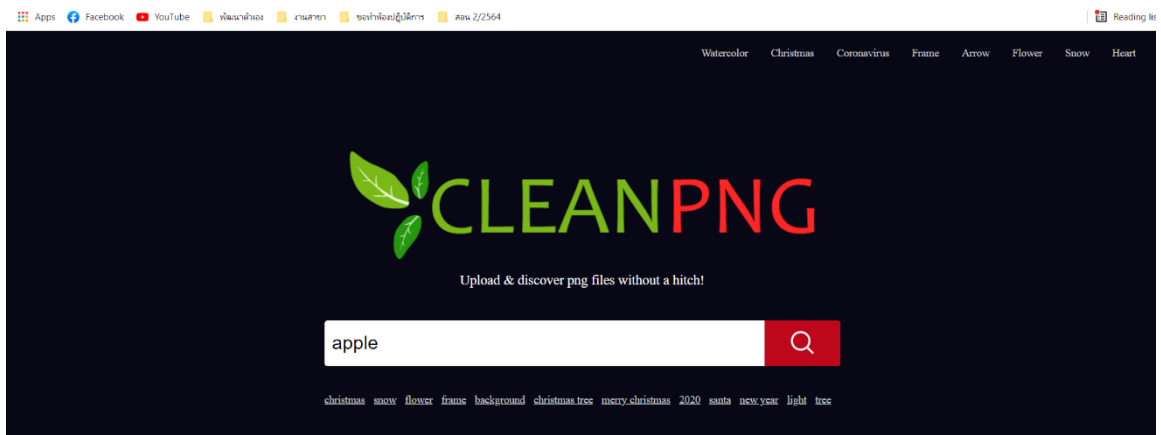
```

ภาพประกอบ 4.71 จัดเรียงโค้ดด้วย Prettier Extension

ที่มา : ณปภัช วรรณตรง (2564 : 23)

## 5. ใส่รูปภาพ

5.1 ให้ทำการใส่รูปภาพ โดยใช้เว็บไซต์ที่สามารถนำรูปภาพมาใช้แบบไม่ติดลิขสิทธิ์ได้ จากเว็บ cleanpng.com ซึ่งเป็นภาพประกอบไม่มีลิขสิทธิ์และพื้นหลังของรูปภาพโปร่งใส

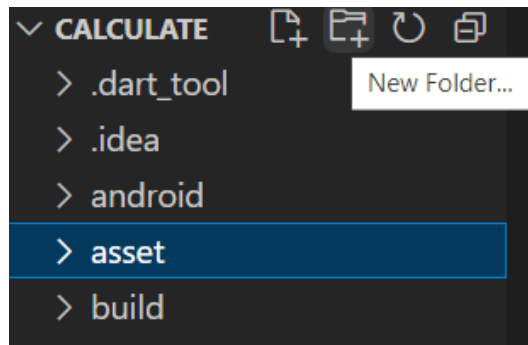


ภาพประกอบ 4.72 นำรูปภาพมาใช้งานบนหน้าเว็บ

ที่มา : CleanPNG. (2564 : 1)

5.2 เมื่อทำการดาวน์โหลดรูปภาพที่ต้องการใช้ได้แล้ว ให้ไปสร้าง Folder assets ที่ Folder Calculate

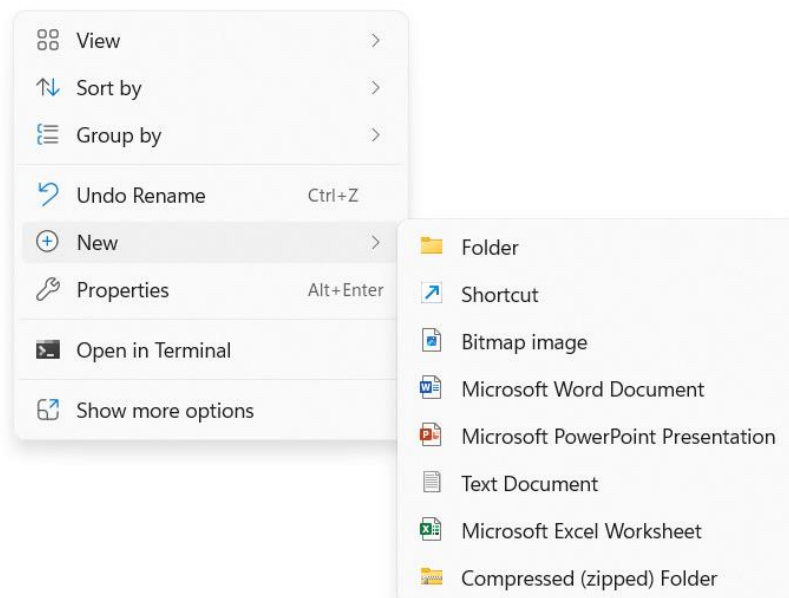
สร้าง Folder assets ผ่าน VSCode โดยทำการไปกดคลิกที่รูปภาพสร้างโฟลเดอร์ใหม่ (New Folder) จากนั้นโปรแกรมจะสร้าง Folder ใหม่ขึ้นมาให้ แล้วให้ตั้งชื่อเป็น assets



ภาพประกอบ 4.73 สร้าง Folder assets ผ่าน VSCode

ที่มา : ณปภัช วรรณตรง (2564 : 25)

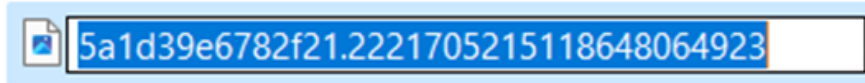
สร้าง Folder ผ่าน Desktop โดยให้เปิด Folder งานขึ้นมา จากนั้นให้คลิกขวาในพื้นที่ว่างของ Folder ไปที่ New > จากนั้นเลือกเป็น Folder เสร็จแล้วให้ทำการตั้งชื่อเป็น assets



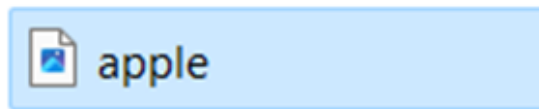
ภาพประกอบ 4.74 สร้าง Folder assets ผ่าน Desktop

ที่มา : ณปภัช วรรณตรง (2564 : 25)

จากนั้นให้ทำการเปลี่ยนชื่อไฟล์ภาพประกอบได้ทำการดาวน์โหลดมาก่อนหน้า เพื่อให้  
ง่ายต่อการค้นหาและใช้งาน

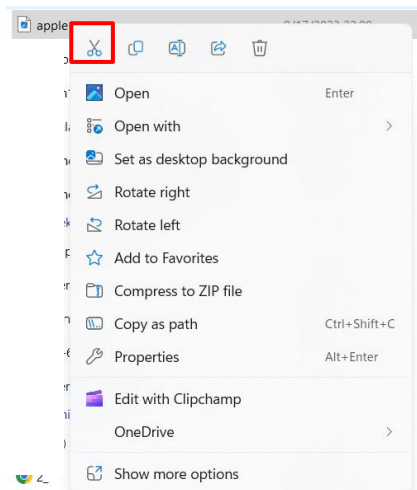


ภาพประกอบ 4.75 เปลี่ยนชื่อไฟล์ภาพ  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 26)



ภาพประกอบ 4.76 เปลี่ยนชื่อไฟล์ภาพ (ต่อ)  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 26)

เมื่อเปลี่ยนชื่อไฟล์เสร็จแล้ว ให้ทำการย้ายไฟล์รูปภาพไปยัง Folder assets ที่ได้  
ทำการสร้างไว้ โดยการกดคลิกขวาที่รูปภาพ เลือก Cut จากนั้นให้ไปที่ Folder assets แล้วคลิกขวาใน  
Folder เลือก Paste หรือ Ctrl+v



ภาพประกอบ 4.77 ย้ายภาพไปยัง Folder assets  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 26)

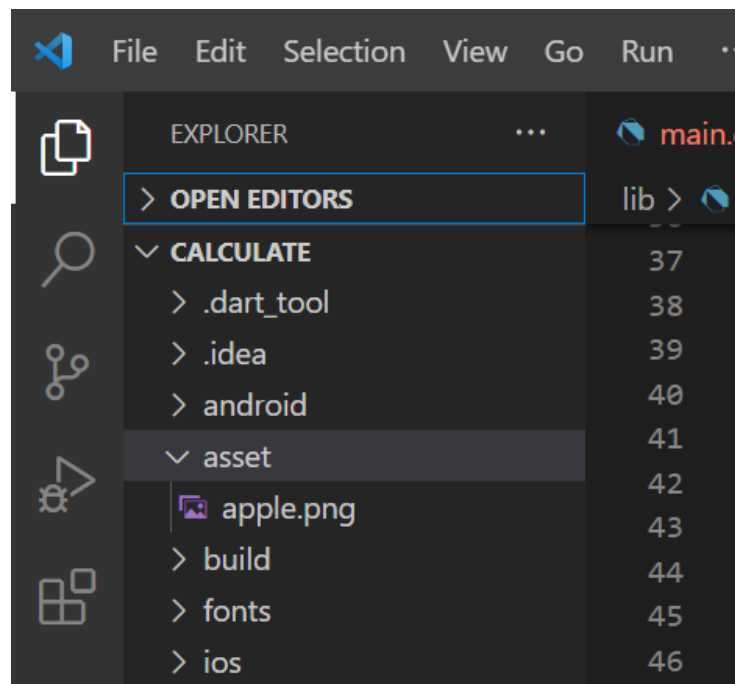


apple

ภาพประกอบ 4.78 ย้ายภาพไปยัง Folder assets (ต่อ)

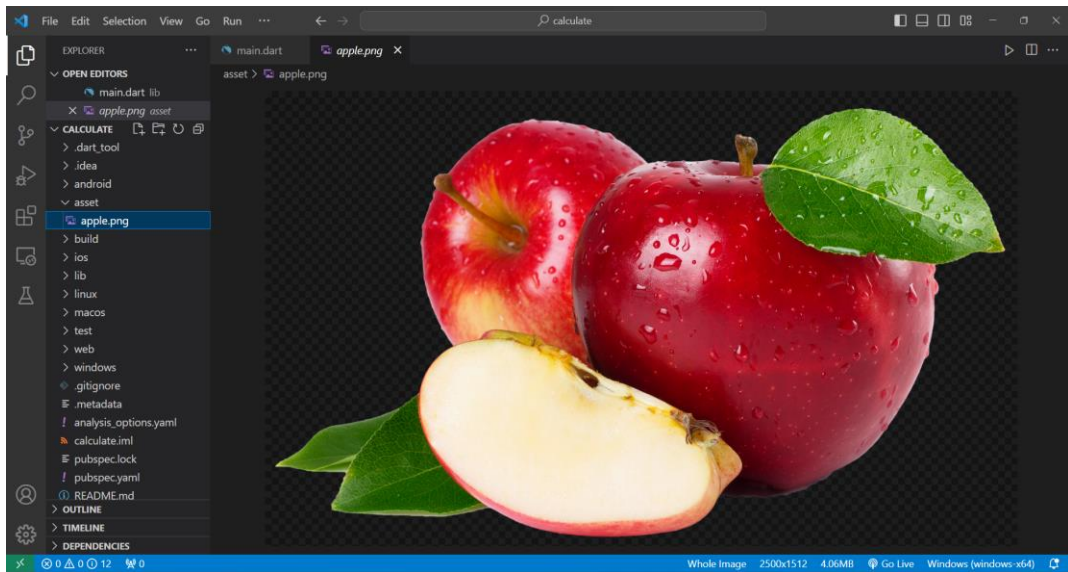
ที่มา : ฌปภัช วรรณตรง (2564 : 26)

เมื่อกลับไปโปรแกรม VSCode จะพบว่าไฟล์รูปภาพได้ปรากฏขึ้นในโปรแกรม VSCode สำหรับนำไปใช้งานเช่นเดียวกัน



ภาพประกอบ 4.79 ย้ายภาพไปยัง Folder assets (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 26)



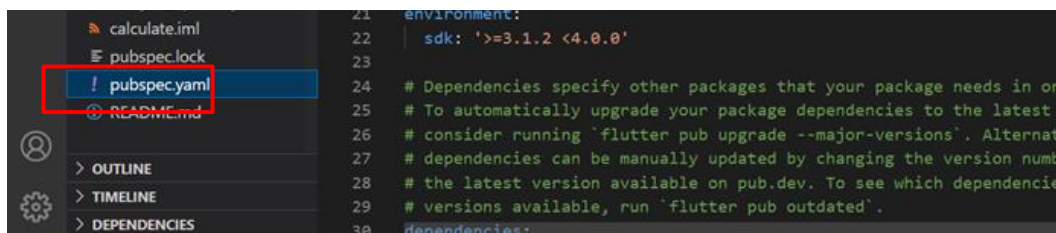
ภาพประกอบ 4.80 ย้ายภาพไปยัง Folder assets (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 26)

5.3 เมื่อทำการนำไฟล์ภาพเข้ามาเพื่อเรียกใช้แล้ว ให้ไปตั้งค่า assets ในไฟล์

pubspec.yaml

เข้าไปที่ไฟล์ pubspec.yaml จากนั้นให้ทำการเลื่อนไปที่ asset ซึ่งจะอยู่ที่ user-material-design: true



ภาพประกอบ 4.81 ตั้งค่า assets ในไฟล์ pubspec.yaml

ที่มา : ฅนปักษ์ วรณตรง (2564 : 27)



```

60
61 # To add assets to your application, add an assets section,
62 # assets:
63 #   - images/a_dot_burr.jpeg
64 #   - images/a_dot_ham.jpeg
65

```

ภาพประกอบ 4.82 ตั้งค่า assets ในไฟล์ pubspec.yaml (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 27)

เมื่อเจอตำแหน่งที่ต้องการแก้ไขแล้วให้ทำการเปิดคอมเมนต์ (Comment) โดยการครอบ  
โค้ดบรรทัดที่ต้องการ จากนั้นให้กด Ctrl+/

```

61 # To add assets to your application, add an assets section,
62 # assets:
63 #   - images/a_dot_burr.jpeg
64 #   - images/a_dot_ham.jpeg

```

ภาพประกอบ 4.83 เปิดคอมเมนต์เพื่อให้โค้ดทำงานได้

ที่มา : ฌปภัช วรรณตรง (2564 : 27)

```

61 # To add assets to your application, add an assets section,
62 assets:
63   - images/a_dot_burr.jpeg
64   # - images/a_dot_ham.jpeg

```

ภาพประกอบ 4.84 เปิดคอมเมนต์เพื่อให้โค้ดทำงานได้ (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 27)

จากนั้นให้ทำการเปลี่ยนชื่อไฟล์ให้ตรงกับชื่อภาพประกอบได้นำเข้ามาใช้งาน

```

61 # To add assets to your applic
62 assets:
63   - images/a_dot_burr.jpeg
64   # - images/a_dot_ham.jpeg

```

ภาพประกอบ 4.85 เปลี่ยนชื่อไฟล์ให้ตรงกัน

ที่มา : ฌปภัช วรรณตรง (2564 : 27)

```

62  assets:
63  |   - asset/apple.png
64  #   - images/a_dot_ham.jpeg
65

```

ภาพประกอบ 4.86 เปลี่ยนชื่อไฟล์ให้ตรงกัน (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 27)

```

46
47  # To add assets to your application, add an as
48  assets:
49  |   - assets/apple.png
50  #   - images/a_dot_ham.jpeg
51

```

ภาพประกอบ 4.87 เปลี่ยนชื่อไฟล์ให้ตรงกัน (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 27)

5.4 เมื่อทำการตั้งค่าไฟล์รูปภาพเรียบร้อยแล้ว ให้กลับไปแก้ไขโค้ดที่ไฟล์ main.dart โดยการใส่โค้ด Image.asset(),

เมื่อกลับมาที่ไฟล์ main.dart ให้ไปที่ Text("-picture-") จากนั้นแก้ไขโค้ดเป็น Image(...)

```

30  return Center(
31  |   child: Column(
32  |   |   children: [
33  |   |   |   Text("-picture-"),
34  |   |   |   Text("โปรแกรมคำนวณ"),
35  |   |   |   TextField(
36  |   |   |   |   decoration: InputDecoration(labelText: "จำนวนแอปเปิ้ล"),
37  |   |   |   |   ) // TextField
38  |   |   |   ],),); // Column // Center
39  }
40

```

ภาพประกอบ 4.88 แก้ไขโค้ดเพื่อนำรูปภาพไปแสดงที่หน้าเว็บ

ที่มา : ฌปภัช วรรณตรง (2564 : 28)

แก้ไขโค้ดโดยการพิมพ์คำสั่ง `Image.a` จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง `Image.assets(...)` โดยการกด `Click` ที่คำสั่งหรือ `Enter`

```

30     return Center(
31       child: Column(
32         children: [
33           Image.a,
34           Text("📄 asset(...) (String name, {Key? key, AssetBundle? bun...
35           TextField(
36             decoration: InputDecoration(labelText: "จำนวนแอปเปิ้ล",border
37           ) // TextField
38         ],),); // Column // Center
39       }
40     }

```

ภาพประกอบ 4.89 แก้ไขโค้ดเพื่อนำรูปภาพไปแสดงที่หน้าเว็บ (ต่อ)

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 28)

```

30     return Center(
31       child: Column(
32         children: [
33           Image.asset(name),
34           Text("โปรแกรมคำนวณ"),
35           TextField(
36             decoration: InputDecoration(labelText: "จำนวนแอปเปิ้ล",b
37           ) // TextField
38         ],),); // Column // Center
39       }
40     }

```

ภาพประกอบ 4.90 แก้ไขโค้ดเพื่อนำรูปภาพไปแสดงที่หน้าเว็บ (ต่อ)

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 28)

จากนั้นให้ทำการเปลี่ยนชื่อ name ให้เป็นตำแหน่งชื่อไฟล์ที่ได้ทำการเก็บเอาไว้

```

30     return Center(
31       child: Column(
32         children: [
33           Image.asset(name),
34           Text("โปรแกรมคำนวณ"),
35           TextField(
36             decoration: InputDecoration(labelText: "จำนวนแอปเปิ้ล", border
37           ) // TextField
38         ],),),); // Column // Center
39     }
40 }

```

ภาพประกอบ 4.91 แก้ไขโค้ดเพื่อนำรูปภาพไปแสดงที่หน้าเว็บ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 28)

```

30     return Center(
31       child: Column(
32         children: [
33           Image.asset("asset/apple.png"),
34           Text("โปรแกรมคำนวณ"),
35           TextField(
36             decoration: InputDecoration(labelText: "จำนวนแอปเปิ้ล", border
37           ) // TextField
38         ],),),); // Column // Center
39     }
40 }

```

ภาพประกอบ 4.92 แก้ไขโค้ดเพื่อนำรูปภาพไปแสดงที่หน้าเว็บ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 28)

เมื่อกำหนดตำแหน่งและชื่อไฟล์เรียบร้อยแล้ว ให้ทำการกำหนดขนาดของภาพ โดยการพิมพ์คำสั่ง `width : [ขนาดภาพที่ต้องการ] ต่อจาก "` โดยคั่นด้วยเครื่องหมาย ,

```

30     return Center(
31       child: Column(
32         children: [
33           Image.asset("asset/apple.png", width: 50),
34           Text("โปรแกรมคำนวณ"),
35           TextField(
36             decoration: InputDecoration(labelText: "จำนวนแอปเปิ้ล", border: OutlineInputBorder()),
37           ) // TextField
38         ],),); // Column // Center
39   }
40 }

```

ภาพประกอบ 4.93 แก้ไขโค้ดเพื่อนำรูปภาพไปแสดงที่หน้าเว็บ (ต่อ)  
ที่มา : ณปภัช วรรณตรง (2564 : 28)

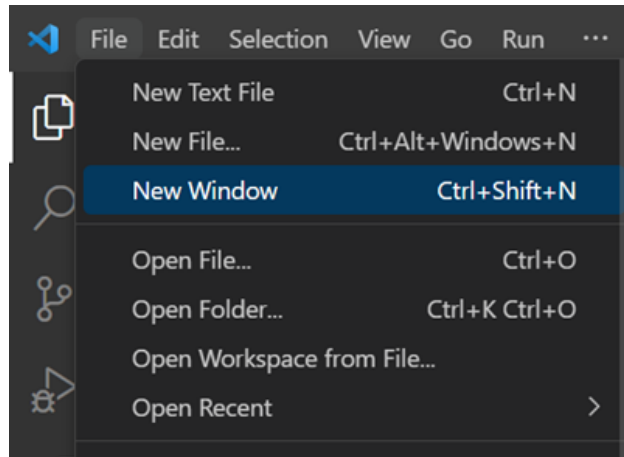
```

class _HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Column(
        children: [
          Image.asset('assets/apple.png', width: 100),
          Text("โปรแกรมคำนวณ"),
          TextField(
            decoration: InputDecoration(
              labelText: "จำนวนแอปเปิ้ล", border: OutlineInputBorder())
          ),
        ],
      ),
    );
  }
}

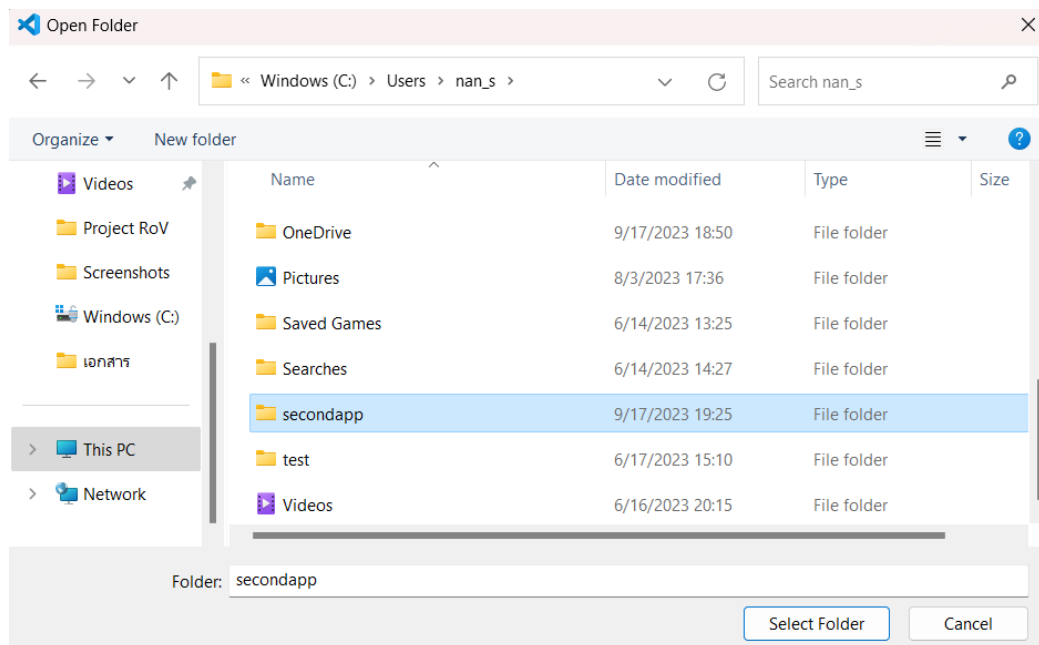
```

ภาพประกอบ 4.94 แก้ไขโค้ดที่ไฟล์ main.dart  
ที่มา : ณปภัช วรรณตรง (2564 : 28)

5.5 ให้ทำการเปิดหน้าต่างขึ้นมาใหม่เพื่อทำการดูโค้ดเก่าและนำโค้ดตกแต่งของงานเก่ามาแก้ไขโค้ดเพื่อตกแต่งฟอนต์ในไฟล์งานปัจจุบัน โดยการเปิดคล้ายกับการเปิด Folder calculate ไปที่ New Window และเลือก Folder secondapp



ภาพประกอบ 4.95 เปิด New Window  
ที่มา : ณปภัช วรณตรง (2564 : 30)



ภาพประกอบ 4.96 เปิด New Window (ต่อ)  
ที่มา : ณปภัช วรณตรง (2564 : 30)

5.6 หลังจากนั้นให้ทำการปรับคุณลักษณะโค้ดของฟอนต์ โดยการแก้ไขโค้ดต่อจาก “โปรแกรมคำนวณ”, ของ Text(...)

ควรตรวจสอบ () และ , ให้ถูกต้องเพื่อป้องกันการดำเนินงานผิดพลาดของโค้ด

```

30     return Center(
31       child: Column(
32         children: [
33           Image.asset('assets/app/png',width: 50,),
34           Text("โปรแกรมคำนวณ",
35             style: TextStyle(fontSize: 30,
36               fontWeight: FontWeight.bold,
37               color: Colors.blue[200]),), // TextStyle // Text
38           TextField(
39             decoration: InputDecoration(labelText: "จำนวนแอปเปิ้ล",border:
40           ); // Center
41         ]
42       );

```

ภาพประกอบ 4.97 ตกแต่งคุณลักษณะโค้ดของฟอนต์

ที่มา : ฅนปักษ์ วรณตรง (2564 : 31)

5.7 เมื่อทำการปรับคุณลักษณะของโค้ดเสร็จแล้ว ให้ทำการเพิ่มโค้ดที่เป็นปุ่มต่อจาก TextField(...), โดยการพิมพ์ Elev จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง ElevatedButton(...) โดยการกด Click ที่คำสั่งหรือ Enter

```

35     TextField(
36       decoration: InputDecoration(labelText: "จำนวนแอปเปิ้ล",border:
37     ); // TextField
38     Elev
39     ],),); ElevatedBut... ({required void Function
40     ElevatedButton.icon(...)
41     ElevatedButtonTheme( )

```

ภาพประกอบ 4.98 เพิ่มโค้ดที่เป็นปุ่ม

ที่มา : ฅนปักษ์ วรณตรง (2564 : 32)

```

35     TextField(
36       decoration: InputDecoration(labelText: "จำนวนแอปเปิ้ล", bord
37     ), // TextField
38     ElevatedButton(onPressed: onPressed, child: child)
39   ],),); // Column // Center
40 }
41 }

```

ภาพประกอบ 4.99 เพิ่มโค้ดที่เป็นปุ่ม (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 32)

จากนั้นให้ทำการแก้ไขโค้ดภายใน ElevatedButton(...) โดยแก้เป็น onPressed: () {}, child: Text("คำนวณ")

```

35     TextField(
36       decoration: InputDecoration(labelText: "จำนวนแอปเปิ้ล", border: Out
37     ), // TextField
38     ElevatedButton(onPressed: () {}, child: Text("คำนวณ")),
39   ],),); // Column // Center
40 }
41 }

```

ภาพประกอบ 4.100 เพิ่มโค้ดที่เป็นปุ่ม (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 32)

```

children: [
  Image.asset('assets/apple.png', width: 100),|
  Text("โปรแกรมคำนวณ", style: TextStyle(fontSize: 30,fontWeight: FontWeight.bold, color: C
  TextField(
    decoration: InputDecoration(labelText: "จำนวนแอปเปิ้ล", border: OutlineInputBorder()),),
    ElevatedButton(onPressed: () {}, child: Text("คำนวณ"))
  ],

```

ภาพประกอบ 4.101 เพิ่มโค้ดที่เป็นปุ่ม (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 32)



เมื่อเพิ่มปุ้มแล้ว ให้ทำการกดบันทึกไฟล์ จากนั้นให้กลับไปหน้าจอต่าง cmd แล้วทำการ  
สั่งการทำงานของ Flutter

```
C:\Users\nan_s\calculate>flutter run|
```

ภาพประกอบ 4.102 สั่งการทำงาน Flutter หลังการแก้ไขไฟล์โค้ด

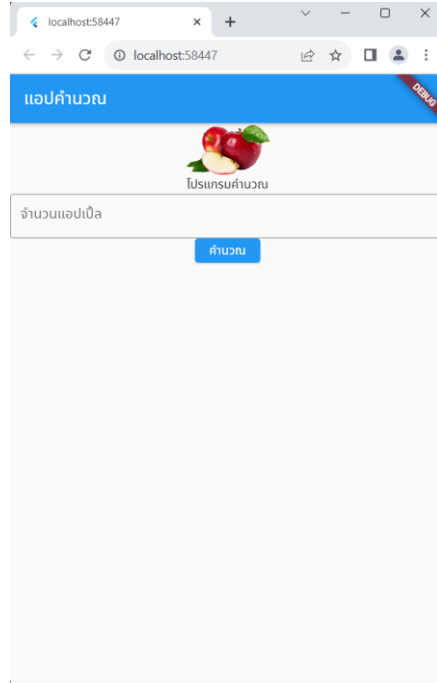
ที่มา : ฅนปักษ์ วรณตรง (2564 : 33)

```
C:\Users\nan_s\calculate>flutter run
Connected devices:
Windows (desktop) • windows • windows-x64 • Microsoft Windows [Version 10.0.22621.2283]
Chrome (web) • chrome • web-javascript • Google Chrome 116.0.5845.188
Edge (web) • edge • web-javascript • Microsoft Edge 117.0.2045.31
[1]: Windows (windows)
[2]: Chrome (chrome)
[3]: Edge (edge)
Please choose one (or "q" to quit): 2
Launching lib\main.dart on Chrome in debug mode...
Waiting for connection from debug service on Chrome... ||
```

ภาพประกอบ 4.103 สั่งการทำงาน Flutter หลังการแก้ไขไฟล์โค้ด (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 33)

ผลลัพธ์หลังการแก้ไขไฟล์โค้ด



ภาพประกอบ 4.104 ผลลัพธ์หลังการแก้ไขไฟล์โค้ด

ที่มา : ฌปภัช วรณตรง (2564 : 33)

## 6. SizedBox

เมื่อเรียงวิดเจ็ตในแนวตั้งด้วย Column วิดเจ็ต สามารถกำหนดระยะห่างระหว่างวิดเจ็ตในแต่ละคอลัมน์ด้วยการกำหนด SizedBox (จิราวุธ วารินทร์, 2564 : 150)

6.1 เนื่องจากกรอบการแสดงผลติดกันจนเกินไป สามารถเพิ่มโค้ด Sizebox ต่อจาก TextFeild(...) เพื่อเว้นระยะห่างระหว่างปุ่มและข้อความ โดยการพิมพ์ Siz จากนั้นจะปรากฏหน้าต่างให้ทำการเลือกคำสั่ง SizedBox(...) โดยการกด Click ที่คำสั่งหรือ Enter

```

35     TextField(
36       decoration: InputDecoration(
37         labelText: "จำนวนแอปเปิ้ล",
38         border: OutlineInputBorder()), // InputDecoration // TextF
39     Siz
40     Ele SizeChangedLayoutNotifier(...)
41     ],),) SizeTransition(...)
42   }   SizeBox(...) ({Key? key, double? width, double? height...
43   }   SizeBox.expand(...)
         SizeBox.fromSize(...)

```

ภาพประกอบ 4.105 เพิ่มกรอบการแสดงผล

ที่มา : ฅนปักษ์ วรณตรง (2564 : 34)

ภายใน SizeBox(...) ให้ทำการเพิ่มโค้ด height: [ความสูงที่ต้องการ] เพื่อกำหนดความสูงของ SizeBox

```

35     TextField(
36       decoration: InputDecoration(
37         labelText: "จำนวนแอปเปิ้ล",
38         border: OutlineInputBorder()), // InputDecoration // Te
39     SizeBox(h)
40     ElevatedBu [e] height:
41     ],),); // Column // Center
42   }
43   }

```

ภาพประกอบ 4.106 เพิ่มกรอบการแสดงผล (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 34)

```

35     TextField(
36       decoration: InputDecoration(
37         labelText: "จำนวนแอปเปิ้ล",
38         border: OutlineInputBorder()), // InputDecoration // TextF
39     SizeBox(height: 20.0),
40     ElevatedButton(onPressed: () {}, child: Text("คำนวณ")),
41     ],),); // Column // Center
42   }
43   }

```

ภาพประกอบ 4.107 เพิ่มกรอบการแสดงผล (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 34)

```

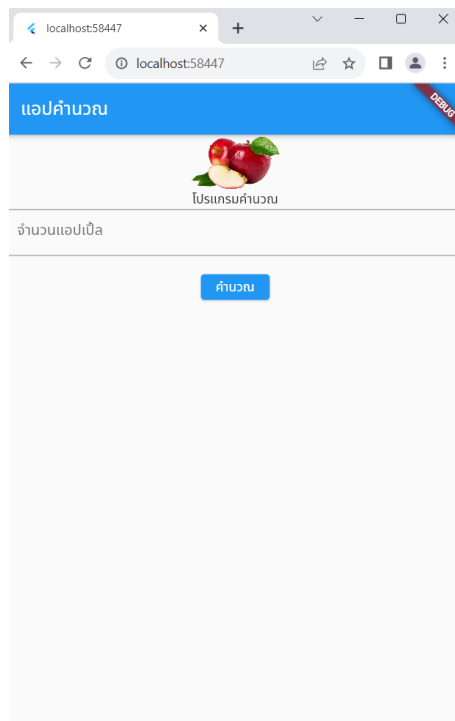
TextField(
  decoration: InputDecoration(labelText: "จำนวนแอปเปิ้ล", border: OutlineInputBorder()),
  SizedBox(height: 20.0),
  ElevatedButton(onPressed: () {}, child: Text("คำนวณ"))
],
), // Column
// Center

```

ภาพประกอบ 4.108 เพิ่มกรอบการแสดงผล (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 34)

6.2 หลังทำการสร้างไฟล์งานแก้ไขปรับปรุงโค้ดเรียบร้อยแล้ว จะได้ผลลัพธ์ตามภาพ



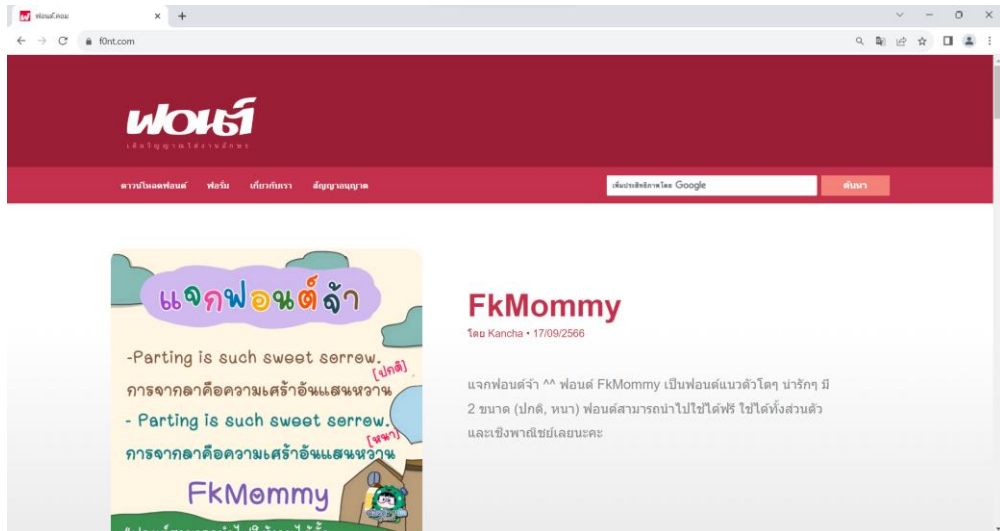
ภาพประกอบ 4.109 ผลลัพธ์ของการสร้างแอปพลิเคชัน

ที่มา : ฌปภัช วรรณตรง (2564 : 35)

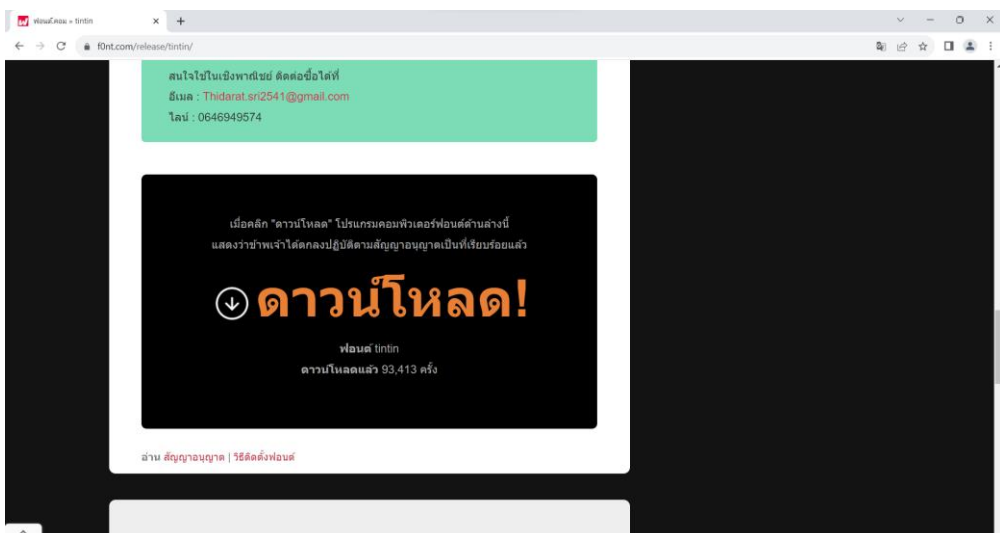
## 7. เปลี่ยนรูปแบบฟอนต์ (Font)

7.1 หากต้องการเปลี่ยนฟอนต์ ให้ทำการดาวน์โหลดฟอนต์มาใช้งาน ซึ่งมีวิธีการคล้ายกับการนำรูปเข้ามาใช้งาน

ให้ทำการเข้าไปดาวน์โหลดฟอนต์ที่ต้องการใช้งาน

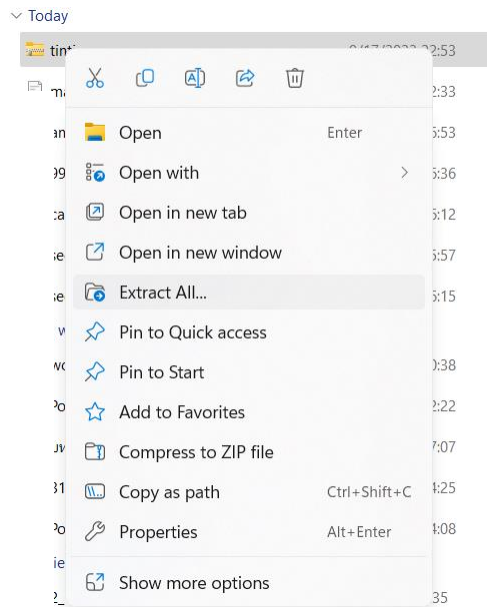


ภาพประกอบ 4.110 ดาวน์โหลดฟอนต์ที่ต้องการ  
ที่มา : ฌปภัช วรณตรง (2564 : 36)



ภาพประกอบ 4.111 ดาวน์โหลดฟอนต์ที่ต้องการ (ต่อ)  
ที่มา : ฌปภัช วรณตรง (2564 : 36)

เมื่อทำการดาวน์โหลดเรียบร้อยแล้วให้ทำการแตกไฟล์ที่ได้ทำการดาวน์โหลดมา



ภาพประกอบ 4.112 แยกไฟล์ที่ได้ทำการดาวน์โหลดมา

ที่มา : ฌปภัช วรรณตรง (2564 : 36)

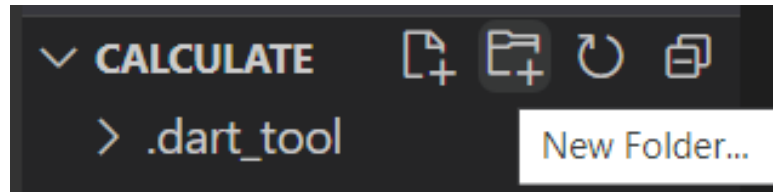
เมื่อได้ไฟล์ฟอนต์มาแล้ว ให้ทำการเปลี่ยนชื่อเพื่อให้งานต่อการนำไปใช้งาน



ภาพประกอบ 4.113 เปลี่ยนชื่อไฟล์เพื่อให้ง่ายต่อการใช้งาน

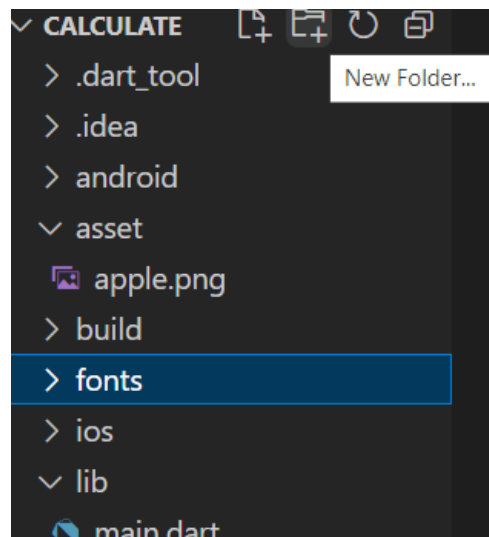
ที่มา : ฌปภัช วรรณตรง (2564 : 36)

เมื่อเปลี่ยนชื่อให้ง่ายต่อการใช้งานแล้ว กลับไปที่ VSCode และทำการสร้าง Folder fonts ขึ้นมาเพื่อเก็บไฟล์ฟอนต์



ภาพประกอบ 4.114 สร้าง Folder Fonts

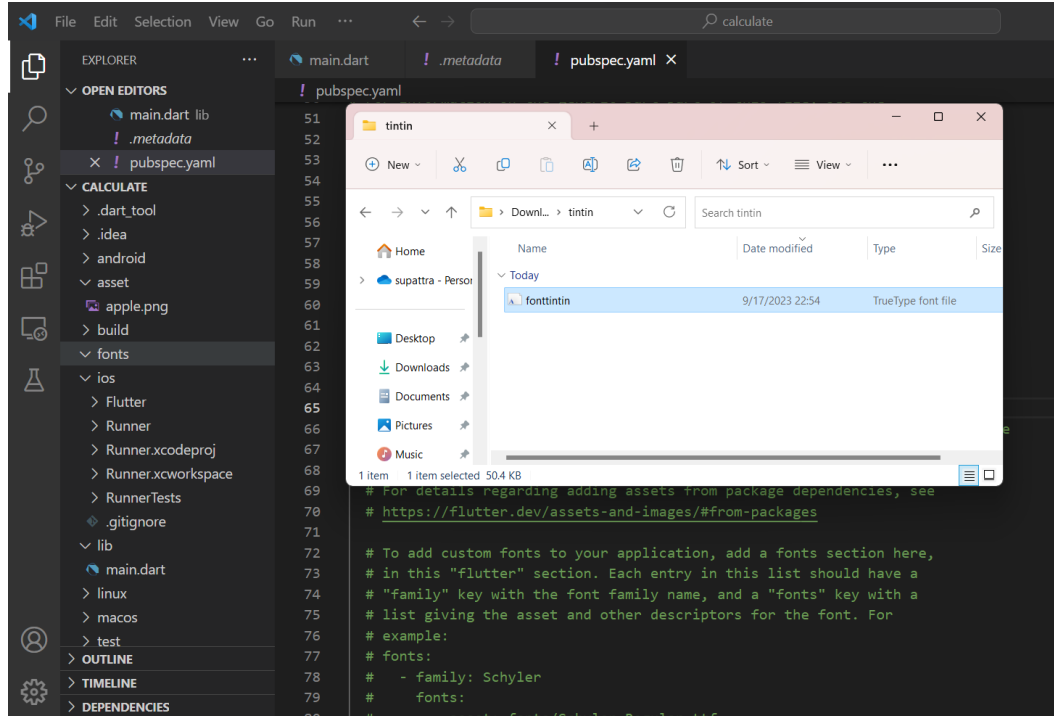
ที่มา : ฅนปักษ์ วรณตรง (2564 : 36)



ภาพประกอบ 4.115 สร้าง Folder fonts (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 36)

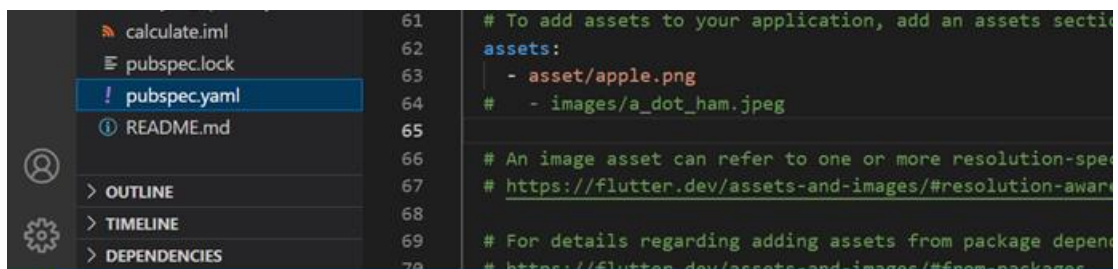
จากนั้นให้ทำการย้ายไฟล์ฟอนต์เข้าไปไว้ใน Folder fonts



ภาพประกอบ 4.116 ย้ายไฟล์ฟอนต์ไปยัง Folder fonts

ที่มา : ณปภัช วรรณตรง (2564 : 36)

เมื่อย้ายไฟล์ฟอนต์แล้ว ให้ไปที่ไฟล์ pubspec.yaml เพื่อแก้ไขโค้ดที่นำฟอนต์เข้ามาใช้งาน



ภาพประกอบ 4.117 ไปที่ไฟล์ pubspec.yaml

ที่มา : ณปภัช วรรณตรง (2564 : 36)



ค้นหา fonts จากนั้นให้ทำปิดคอมเมนต์เพื่อสั่งเปิดการใช้งานโค้ด

```

77  # fonts:
78  #   - family: Schyler
79  #   fonts:
80  #     - asset: fonts/Schyler-Regular.ttf
81  #     - asset: fonts/Schyler-Italic.ttf
82  #       style: italic

```

ภาพประกอบ 4.118 เปิดใช้งานโค้ด fonts

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 36)

จากนั้นให้ทำการแก้ไขชื่อฟอนต์เป็นชื่อไฟล์ฟอนต์ที่ได้ทำการดาวน์โหลดมา

```

77  fonts:
78  |   - family: Schyler
79  |   fonts:
80  |     - asset: fonts/Schyler-Regular.ttf
81  #     - asset: fonts/Schyler-Italic.ttf
82  #       style: italic

```

ภาพประกอบ 4.119 เปลี่ยนชื่อฟอนต์

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 36)

เมื่อเปลี่ยนชื่อฟอนต์แล้วให้ทำการอ้างอิงไฟล์ฟอนต์ให้ตรงกับไฟล์ที่นำมาใช้งาน

```

77  fonts:
78  |   - family: fonttintin
79  |   fonts:
80  |     - asset: fonts/Schyler-Regular.ttf

```

ภาพประกอบ 4.120 อ้างอิงไฟล์ฟอนต์ให้ตรงกับไฟล์ที่นำมาใช้งาน

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 36)

```

77   fonts:
78     - family: fonttintin
79     fonts:
80       - asset: fonts/fonttintin.ttf

```

ภาพประกอบ 4.121 อ้างอิงไฟล์ฟอนต์ให้ตรงกับไฟล์ที่นำมาใช้งาน (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 36)

จากนั้นให้ทำการกำหนดฟอนต์ให้กับข้อความที่ต้องการเปลี่ยนแปลงฟอนต์ โดยการพิมพ์ font จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง fontFamily: โดยการกด Click ที่คำสั่งหรือ Enter

```

child: Column(
  children: [
    Image.asset("asset/apple.png",width: 100,),
    Text("โปรแกรมคำนวณ",style: TextStyle(fontSize: 30,
      fontWeight: FontWeight.bold, color: Colors.blue[200],f)), // TextStyle
    TextField(
      decoration: InputDecoration(
        labelText: "จำนวนแอปเปิ้ล",
        border: OutlineInputBorder()), // InputDecoration //
      ],), // Column // Center

```

ภาพประกอบ 4.122 กำหนดฟอนต์ให้กับข้อความที่ต้องการเปลี่ยนแปลงฟอนต์

ที่มา : ฌปภัช วรรณตรง (2564 : 36)

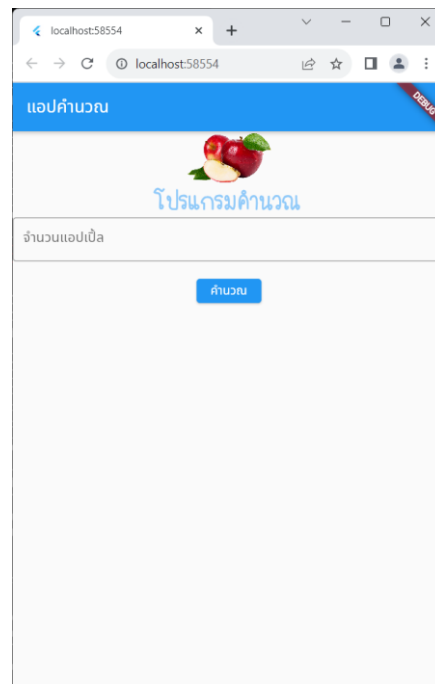
หลังจากนั้นให้ทำการกำหนดชื่อให้ตรงกับไฟล์ฟอนต์ที่นำมาใช้งาน

```
child: Column(
  children: [
    Image.asset("asset/apple.png",width: 100,),
    Text("โปรแกรมคำนวณ",style: TextStyle(fontSize: 30,
      fontWeight: FontWeight.bold, color: Colors.blue[200],fontFamily: 'fonttintin')),
    TextField(
      decoration: InputDecoration(
        labelText: "จำนวนแอปเปิ้ล",
        border: OutlineInputBorder()), // InputDecoration // TextField
      SizedBox(height: 20.0,),
      ElevatedButton(onPressed: () {}, child: Text("คำนวณ")),
```

ภาพประกอบ 4.123 กำหนดชื่อให้ตรงกับไฟล์ฟอนต์ที่นำมาใช้งาน

ที่มา : ณปภัช วรรณตรง (2564 : 36)

ผลลัพธ์การปรับแต่งฟอนต์ของข้อความ



ภาพประกอบ 4.124 ผลลัพธ์การปรับแต่งฟอนต์ของข้อความ

ที่มา : ณปภัช วรรณตรง (2564 : 36)

## ตกแต่งแอปพลิเคชันด้วย Padding

Padding เป็นวิดเจ็ตที่ทำหน้าที่สำหรับการแทรกวิดเจ็ตลูกโดยการเติมช่องว่างหรือเพิ่มระยะห่างไว้รอบ ๆ ภายในพื้นที่ที่กำหนดด้วยค่าของ EdgInsets ซึ่งมีการใช้งานหลัก 2 แบบ ดังนี้

1. EdgInsets.all คือ การเว้นระยะห่างจากขอบของวัตถุรอบทุกด้านทั้ง 4 ด้าน คือ ขอบด้านบน ซ้ายล่าง และขวา โดยกำหนดเป็นระยะด้วยค่าตัวเลขเป็นพิกเซล รองรับการกำหนดค่าเลขจำนวนเต็มและเลขจำนวนจริง เช่น EdgInsets.all (10)

2. EdgInsets.fromLTRB คือ การเว้นระยะโดยกำหนดระยะห่างจากขอบแบบที่เป็นค่าอิสระแยก จากกัน ค่าที่กำหนดให้เป็นพารามิเตอร์จะต้องกำหนดทั้งหมด 4 ค่า โดยเรียงลำดับจากขอบด้านบน ซ้าย บน ล่าง และขวา ตามลำดับ ตัวอย่างการใช้งานเช่น EdgInsets fromLTRB (10, 5, 7, 0) จะหมายถึงการกำหนดค่าเว้นระยะขอบด้านบนซ้าย 10 px, ด้านบน 5 px, ด้านขวา 7 px และ ด้านล่าง 0 px เป็นต้น (เอกรินทร์ วัฒนฤกษ์เลิศสกุล, 2563 : 91)

ในส่วนนี้จะทำการตกแต่งขอบของแอปพลิเคชันด้วย Padding

1. หากต้องการส่งให้ผู้ใช้คนอื่นที่ต้องการนำโค้ดไปพัฒนาต่อ ให้ทำการ cmd > cd [ไฟล์ Project] แล้วใช้คำสั่ง flutter clean เพื่อลดขนาดและลบไฟล์ที่ไม่จำเป็นออกจาก Folder งานก่อนทำการ ZIP ไฟล์แล้วส่งให้ผู้ใช้หรือผู้พัฒนาคคนอื่น ๆ นำไปพัฒนาต่อไป

```
Microsoft Windows [Version 10.0.19043.1415]
(c) Microsoft Corporation. All rights reserved.

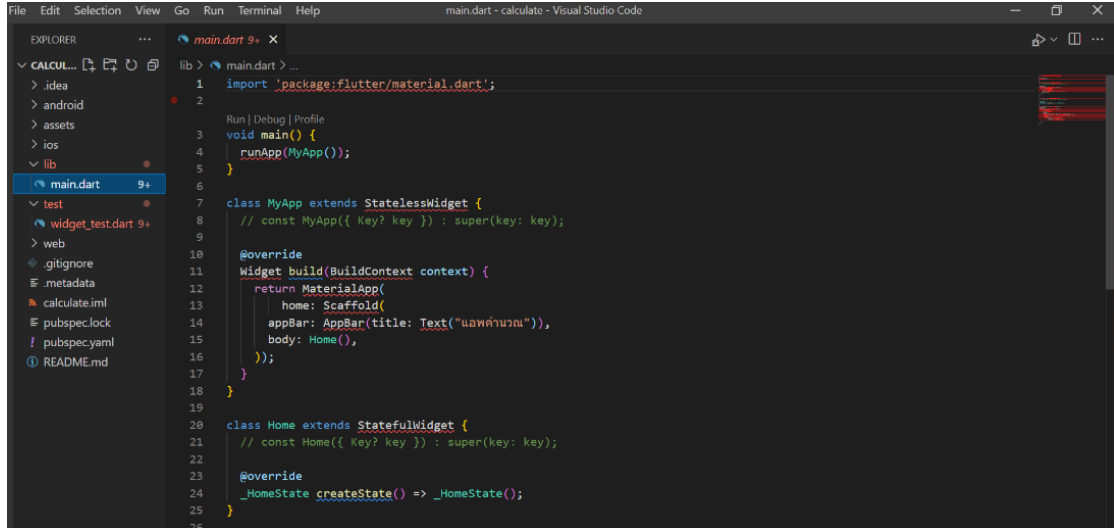
C:\Users\U21H1\Desktop\flutter\calculate>flutter clean
Deleting build... 12ms
Deleting .dart_tool... 58ms
Deleting .packages... 1ms
Deleting Generated.xcconfig... 0ms
Deleting flutter_export_environment.sh... 1ms

C:\Users\U21H1\Desktop\flutter\calculate>
```

ภาพประกอบ 4.125 Flutter Clean

ที่มา : ฌปภัช วรรณตรง (2564 : 3)

2. เมื่อได้ไฟล์มาแล้วให้ทำการแยกไฟล์ และเปิดไฟล์ในโปรแกรม VSCode หากพบ Error ให้ทำการแก้ไขภายหลัง



```

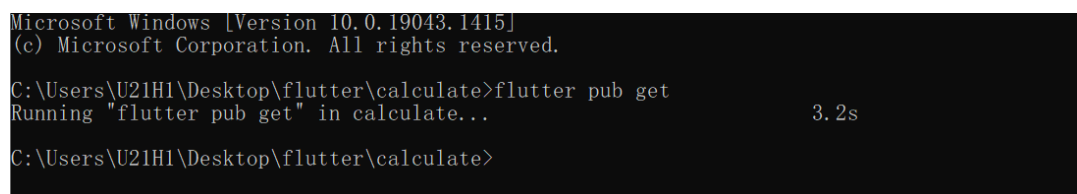
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   // const MyApp({ Key? key }) : super(key: key);
9
10  @override
11  Widget build(BuildContext context) {
12    return MaterialApp(
13      home: Scaffold(
14        appBar: AppBar(title: Text("แอปคำนวณ")),
15        body: Home(),
16      ));
17  }
18 }
19
20 class Home extends StatefulWidget {
21   // const Home({ Key? key }) : super(key: key);
22
23   @override
24   _HomeState createState() => _HomeState();
25 }
26

```

ภาพประกอบ 4.126 เปิดไฟล์ในโปรแกรม VSCode

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 4)

3. หากได้ไฟล์งานมาจากผู้พัฒนาคนอื่นแล้วพบ Error ให้ทำการแก้ไขโดยการเข้าไปยังโฟลเดอร์ไฟล์งาน Project ของผู้พัฒนาผ่าน cmd จากนั้นให้ใช้คำสั่ง flutter pub get เพื่อทำการแก้ไขข้อผิดพลาดของไฟล์งานที่ทำให้เกิด Error ตามภาพ



```

Microsoft Windows [Version 10.0.19043.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Users\U21H1\Desktop\flutter\calculate>flutter pub get
Running "flutter pub get" in calculate... 3.2s

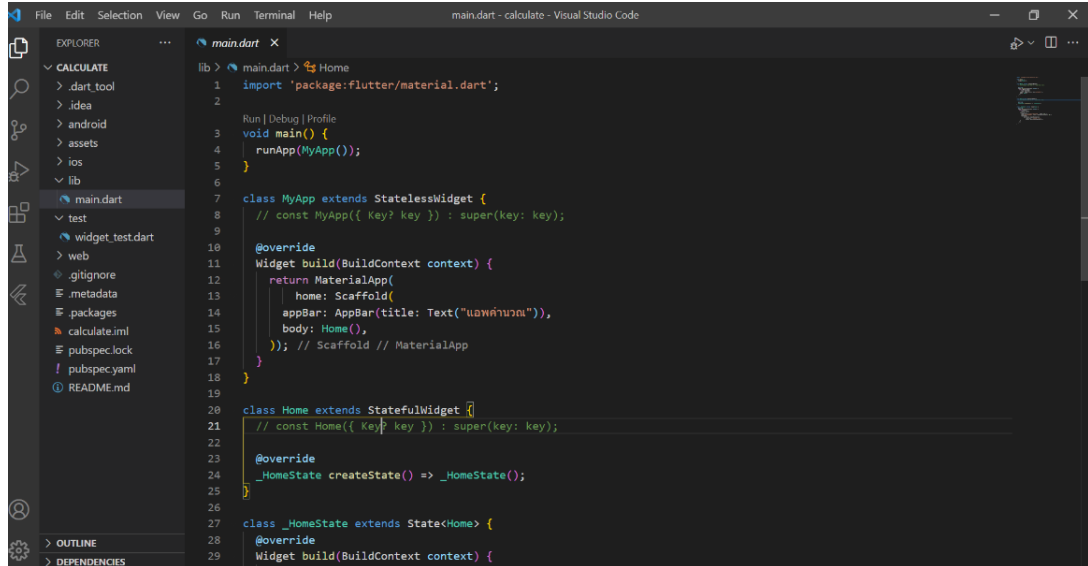
C:\Users\U21H1\Desktop\flutter\calculate>

```

ภาพประกอบ 4.127 ใช้คำสั่ง flutter pub get แก้ไข Error

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 5)

4. หลังจากทำการแก้ไขข้อผิดพลาดของไฟล์งานแล้ว ส่วนที่ Error จะหายไป ให้ทำการเปิดไฟล์งานขึ้นมาและสั่งการทำงานผ่าน flutter run



```

1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   // const MyApp({ Key? key }) : super(key: key);
9
10  @override
11  Widget build(BuildContext context) {
12    return MaterialApp(
13      home: Scaffold(
14        appBar: AppBar(title: Text("แอปคำนวณ")),
15        body: Home(),
16      )); // Scaffold // MaterialApp
17  }
18 }
19
20 class Home extends StatefulWidget {
21   // const Home({ Key? key }) : super(key: key);
22
23   @override
24   _HomeState createState() => _HomeState();
25 }
26
27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {

```

ภาพประกอบ 4.128 เปิดไฟล์งานและสั่งการทำงาน

ที่มา : ฌปภัช วรรณตรง (2564 : 6)

## 1. TextField

เท็กซ์ฟิลด์ (TextField) เป็นกล่องสำหรับรับข้อความที่ผู้ใช้งานแอปพลิเคชันจะกรอกเพื่อใช้เป็นข้อมูลรับเข้าของระบบ หรือบางครั้งสามารถใช้ในการแสดงผลได้ด้วยเช่นกัน โดยการใช้งานเท็กซ์ฟิลด์ จะต้องทำการตกแต่งให้มีความสวยงามและเหมาะต่อการแสดงผล จึงจำเป็นต้องใช้การกำหนดค่าต่าง ๆ เพื่อให้เท็กซ์ฟิลด์มีความน่าใช้งาน (เอกรินทร์ วฑัญญเลิศสกุล, 2563 : 110)

### 1.1 ทำการเขียนโค้ดเพิ่มคำอธิบายใต้ปุ่มของโค้ดที่ได้มา

ให้ทำการเปิดไฟล์ main.dart ขึ้นมา จากนั้นให้เลื่อนหาไปที่ ElevatedButton(...) ให้ทำการพิมพ์โค้ด Text(...) เพิ่มต่อจาก ElevatedButton (...),

```

TextField(
  decoration: InputDecoration(
    labelText: "จำนวนแอปเปิ้ล",
    border: OutlineInputBorder()), // InputDecoration // TextFie
  SizedBox(height: 20.0,),
  ElevatedButton(onPressed: () {}, child: Text("คำนวณ")),
],),); // Column // Center

```

ภาพประกอบ 4.129 เพิ่มโค้ดคำอธิบาย (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 7)

```

TextField(
  decoration: InputDecoration(
    labelText: "จำนวนแอปเปิ้ล",
    border: OutlineInputBorder()), // InputDecoration // TextFie
  SizedBox(height: 20.0,),
  ElevatedButton(onPressed: () {}, child: Text("คำนวณ")),
  Text(""),
],),); // Column // Center

```

ภาพประกอบ 4.130 เพิ่มโค้ดคำอธิบาย (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 7)

เมื่อเพิ่มโค้ด Text(...) แล้ว ให้ทำการเพิ่มข้อความเข้าไปใน (...) โดยใช้เครื่องหมาย “” เป็นตัวเก็บข้อความ

```

TextField(
  decoration: InputDecoration(
    labelText: "จำนวนแอปเปิ้ล",
    border: OutlineInputBorder()), // InputDecoration // Text
  SizedBox(height: 20.0,),
  ElevatedButton(onPressed: () {}, child: Text("คำนวณ")),
  Text("ชื่อแอปเปิ้ล"),
],),); // Column // Center

```

ภาพประกอบ 4.131 เพิ่มโค้ดคำอธิบาย (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 7)

```

30 class _HomeState extends State<Home> {
31   @override
32   Widget build(BuildContext context) {
33     return Center(
34       child: Column(
35         children: [
36           Image.asset('assets/apple.png', width: 50,),
37           Text("โปรแกรมคำนวณ", style: TextStyle(fontSize: 30,
38             fontWeight: FontWeight.bold, color: Colors.blue[300], fontFamily: 'fonttin')),
39           TextField(
40             decoration: InputDecoration(
41               labelText: "จำนวนแอปเปิ้ล", border: OutlineInputBorder(), // InputDecor
42             ), // TextField
43           SizedBox(height: 20.0,),
44           ElevatedButton(onPressed: () {}, child: Text("คำนวณ")),
45           Text("ชื่อแอปเปิ้ลจำนวน x ผล ราคาผล x บาท รวมลูกค้าต้องจ่ายเงิน x บาท")
46         ],
47       ); // Column // Center
48   }
49 }
50

```

ภาพประกอบ 4.132 เพิ่มโค้ดคำอธิบาย (ต่อ)

ที่มา : ฅปภัช วรรณตรง (2564 : 7)

1.2 สามารถทำการตกแต่งหน้าตาของแอปพลิเคชันด้วย Padding ได้โดยการคลิกเลือกที่ Center จากนั้นจะปรากฏดวงไฟสีเหลืองอยู่ทางด้านซ้าย ให้คลิกที่ดวงไฟ จากนั้นจะขึ้นชุดคำสั่ง ให้คลิกเลือกที่ชุดคำสั่ง wrap with padding

1.2.1 ให้ทำการไปที่ Center(...) จากนั้นเมื่อกดคลิกที่คำสั่งจะปรากฏหลอดไฟสีเหลืองขึ้นมาข้างหน้าบรรทัดคำสั่ง

```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Center(
31       child: Column(
32         children: [

```

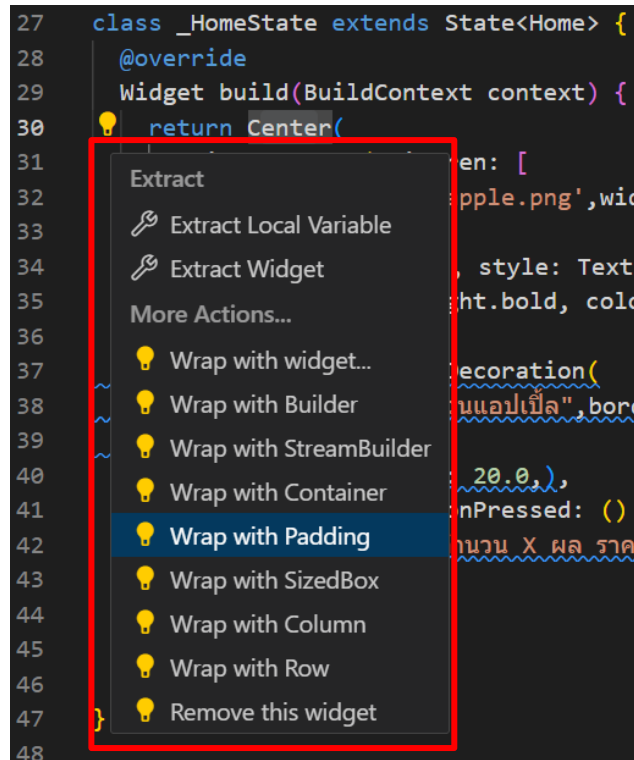
ภาพประกอบ 4.133 ตกแต่งหน้าตาของแอปพลิเคชันด้วย Padding

ที่มา : ฅปภัช วรรณตรง (2564 : 8)



1.2.2 จากนั้นให้ทำการคลิกที่หลอดไฟ จะปรากฏหน้าต่างชุดคำสั่งขึ้นมา ให้เลือกไปที่

Wrap with Padding



```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Center(
31       child: [
32         'apple.png', wid
33       ],
34       style: TextS
35       ht.bold, colo
36     ),
37     decoration(
38       BoxDecoration(
39         border: Border(
40           top: BorderSide(
41             color: Colors.red, width: 2.0, style: SolidStroke),
42           bottom: BorderSide(
43             color: Colors.red, width: 2.0, style: SolidStroke),
44           left: BorderSide(
45             color: Colors.red, width: 2.0, style: SolidStroke),
46           right: BorderSide(
47             color: Colors.red, width: 2.0, style: SolidStroke),
48         ),
49       ),
50     ),
51     onPressed: () {
52       // TODO: Implement this method.
53     },
54   ),
55 );
56 }

```

ภาพประกอบ 4.134 ตกแต่งหน้าต่างของแอปพลิเคชันด้วย Padding (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 8)

```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Center(
31       children: [
32         Image.asset('asset/apple.png', width: 100, height: 100),
33         Text("โปรแกรมคำนวณ", style: TextStyle(fontWeight: FontWeight.bold, color: Colors.blue[200], fontFamily: 'fonttisaeng-ae'),),
34         TextField(
35           decoration: InputDecoration(
36             labelText: "จำนวนแอปเปิ้ล",
37             border: OutlineInputBorder()), // InputDecoration // TextField
38         SizedBox(height: 20.0,),
39         ElevatedButton(onPressed: () {}, child: Text("คำนวณ")),
40         Text("ซื้อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท"),
41       ],
42     );
43   }
44 }
45
46
47
48
49

```

ภาพประกอบ 4.135 ตกแต่งหน้าต่างของแอปพลิเคชันด้วย Padding (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 8)

1.2.3 เมื่อเลือกที่ Warp with Padding แล้ว จะเพิ่มโค้ด Padding(...) มาครอบโค้ด Center(...) และมีโค้ด padding: เพิ่มเข้ามาใน Padding(...) ก่อนชุดคำสั่ง Center(...)

```

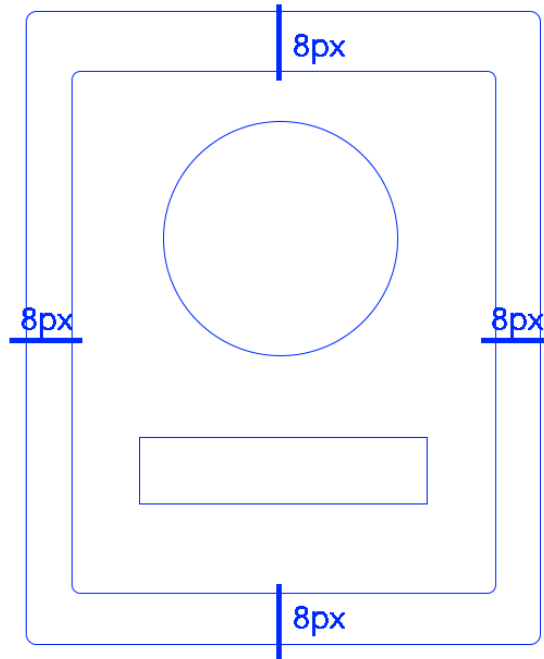
27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Padding(
31       padding: const EdgeInsets.all(8.0),
32       child: Center(
33         child: Column(
34           children: [
35             Image.asset("asset/apple.png", width: 100, height: 100),
36             Text("โปรแกรมคำนวณ", style: TextStyle(fontWeight: FontWeight.bold, color: Colors.blue[200], fontFamily: 'fonttisaeng-ae'),),
37             TextField(
38               decoration: InputDecoration(
39                 labelText: "จำนวนแอปเปิ้ล",
40                 border: OutlineInputBorder()), // InputDecoration // TextField
41             SizedBox(height: 20.0,),
42             ElevatedButton(onPressed: () {}, child: Text("คำนวณ")),
43             Text("ซื้อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท"),
44           ],
45         ),
46       ),
47     );
48   }
49 }

```

ภาพประกอบ 4.136 ตกแต่งหน้าต่างของแอปพลิเคชันด้วย Padding (ต่อ)

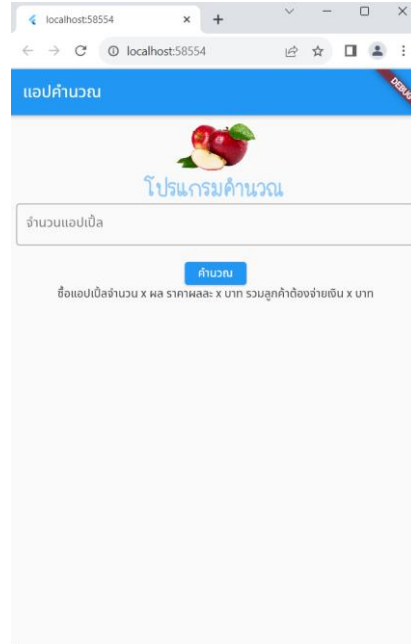
ที่มา : ณปภัช วรรณตรง (2564 : 8)

1.3 จากนั้นโค้ดจะทำการจัดโครงสร้างให้ โดยมีโค้ด `EdgeInsets.all(8.0)` เป็นตัวจัดโครงสร้างรอบ ๆ การแสดงผลของแอปพลิเคชันให้ ตามภาพ โดยหากต้องการกำหนดตัวเลขที่เป็นขนาดระยะขอบใหม่ สามารถแก้ไขตัวเลขให้วงเล็บ



ภาพประกอบ 4.137 ตกแต่งหน้าต่างของแอปพลิเคชันด้วย Padding (ต่อ)  
ที่มา : ฌปภัช วรรณตรง (2564 : 9)

1.4 ให้ทำการใช้คำสั่ง flutter run เพื่อดูผลลัพธ์ ซึ่งจะได้ผลลัพธ์ตามภาพ



ภาพประกอบ 4.138 ผลลัพธ์การตกแต่งหน้าต่างของแอปพลิเคชันด้วย Padding  
ที่มา : ฌปภัช วรรณตรง (2564 : 10)

1.5 ตกแต่งสีของปุ่มด้วยการใช้โค้ด style: ButtonStyle

1.5.1 โดยไปที่ ElevatedButton(...) จากนั้นให้ทำการเพิ่ม style: เข้าไปต่อจาก child: คั่นด้วยเครื่องหมาย ,

```

    SizedBox(height: 20.0,),
    ElevatedButton(
      onPressed: () {},
      child: Text("คำนวณ"),
      style:
    ),
    Text(
      setPluginHandler(...)
      showAboutDialog(...)
  
```

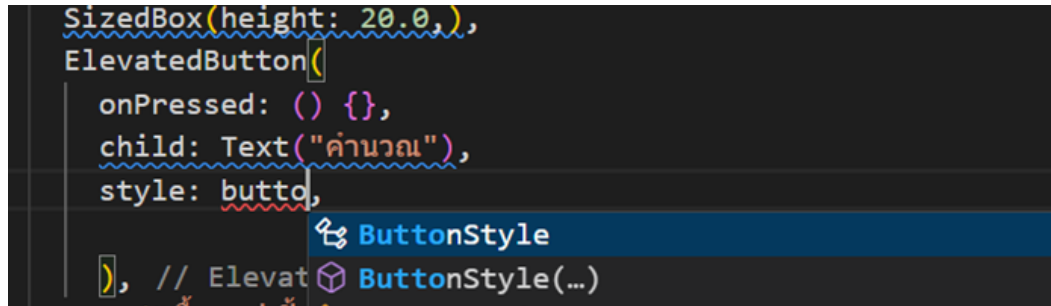
ภาพประกอบ 4.139 กำหนด Style ของปุ่ม

ที่มา : ฌปภัช วรรณตรง (2564 : 11)

1.5.2 เพิ่ม ButtonStyle(...) เข้าไปใน style:

```

    SizedBox(height: 20.0),
    ElevatedButton(
      onPressed: () {},
      child: Text("คำนวณ"),
      style: butt
    ), // Elevat
  
```



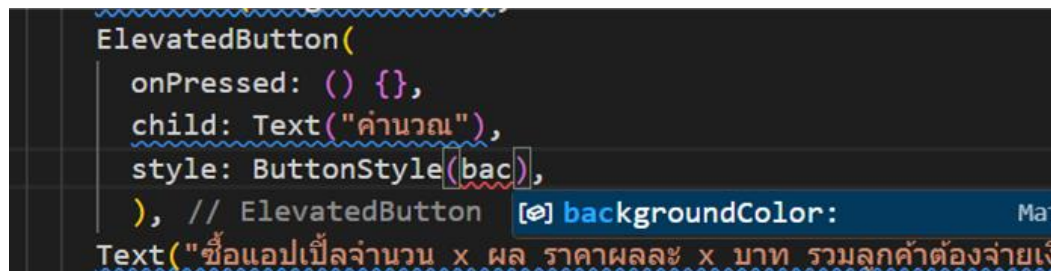
ภาพประกอบ 4.140 กำหนด Style ของปุ่ม (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 11)

1.5.3 ภายใน Button(...) ให้ทำการเพิ่มโค้ด backgroundColor:

```

    ElevatedButton(
      onPressed: () {},
      child: Text("คำนวณ"),
      style: ButtonStyle(bac),
    ), // ElevatedButton
  
```



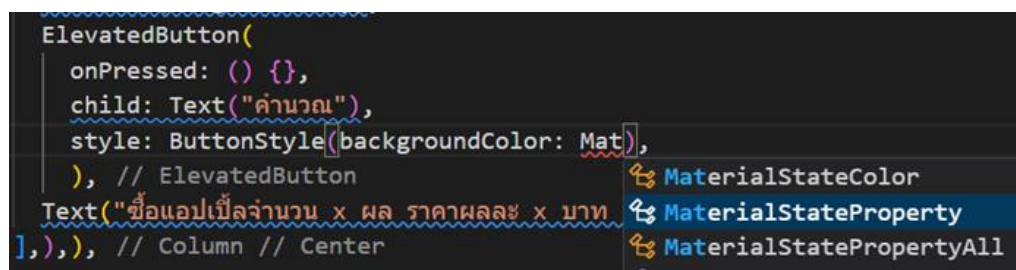
ภาพประกอบ 4.141 กำหนด Style ของปุ่ม (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 11)

1.5.4 จากนั้นเพิ่ม MaterialStateProperty.all(...) เข้าไปต่อจาก  
backgroundColor:

```

    ElevatedButton(
      onPressed: () {},
      child: Text("คำนวณ"),
      style: ButtonStyle(backgroundColor: Mat),
    ), // ElevatedButton
  
```



ภาพประกอบ 4.142 กำหนด Style ของปุ่ม (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 11)

### 1.5.5 ภายใน all(...) ให้ทำการเพิ่มโค้ด Color(...) เข้าไปเพื่อกำหนดสีของปุ่ม

```
ElevatedButton(
  onPressed: () {},
  child: Text("คำนวณ"),
  style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Color)),
), // ElevatedButton
Text("ชื่อแอปเปิ้ลจำนวน x ผล ราคาผล x บาท รวมลูกค้าต้องจ่าย"),
),), // Column // Center
Padding
```

ภาพประกอบ 4.143 กำหนด Style ของปุ่ม (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 11)

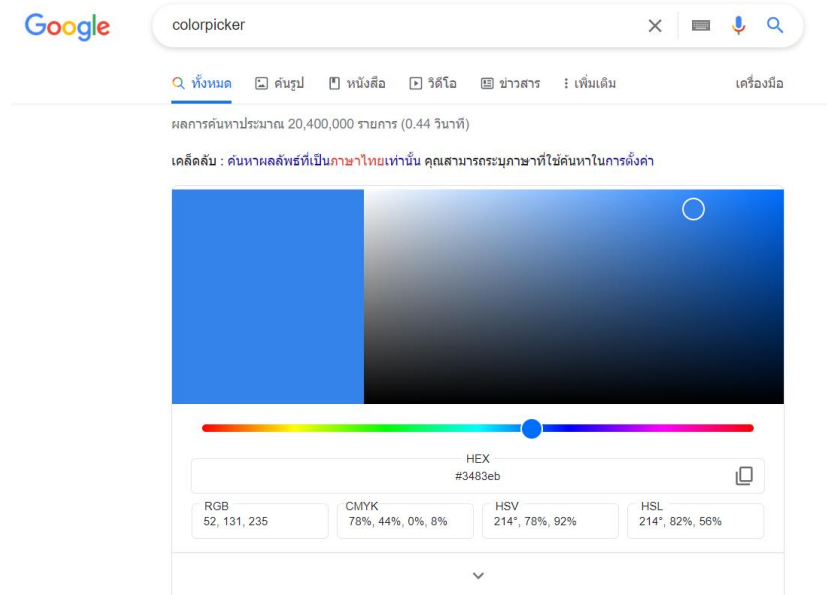
### 1.5.6 ก่อนที่จะเพิ่มรหัสสีเข้าไปใน Color(...) ให้ทำการเพิ่มโค้ด 0xff ก่อนจะเพิ่มรหัสสีเข้าไป

```
ElevatedButton(
  onPressed: () {},
  child: Text("คำนวณ"),
  style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Color(0xff))),
), // ElevatedButton
Text("ชื่อแอปเปิ้ลจำนวน x ผล ราคาผล x บาท รวมลูกค้าต้องจ่ายเงิน x บาท"),
),), // Column // Center
Padding
```

ภาพประกอบ 4.144 กำหนด Style ของปุ่ม (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 11)

1.6 สามารถเลือกใช้โค้ดสีในการตกแต่งได้ตามใจชอบ ซึ่งสามารถนำโค้ดสีมาได้จาก ColorPicker โดยใช้รหัสสีเป็นประเภท HEX



ภาพประกอบ 4.145 เลือกใช้โค้ดสี

ที่มา : ฅนปักษ์ วรณตรง (2564 : 12)

เมื่อได้รับรหัสสีที่ต้องการแล้ว ในนำรหัสมาใส่ต่อจาก 0xff ที่ได้ทำการกำหนดไว้ก่อนหน้านี้

```
ElevatedButton(
  onPressed: () {},
  child: Text("คำนวณ"),
  style: ButtonStyle(backgroundColor: MaterialStateProperty.all(□ Color(0xff))),
), // ElevatedButton
Text("ชื่อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท"),
],), // Column // Center
Padding
```

ภาพประกอบ 4.146 กำหนด Style ของปุ่ม (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 12)

```

ElevatedButton(
  onPressed: () {},
  child: Text("คำนวณ"),
  style: ButtonStyle(backgroundColor: MaterialStateProperty.all( Color(0xff8915b0))),
), // ElevatedButton
Text("ชื่อแอปเปิ้ลจำนวน x ผล ราคาผล x บาท รวมลูกค้าต้องจ่ายเงิน x บาท"),
],),), // Column // Center
Padding

```

ภาพประกอบ 4.147 กำหนด Style ของปุ่ม (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 12)

```

ElevatedButton(
  onPressed: () {},
  child: Text("คำนวณ"),
  style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Color(0xffe3d924))),
), // ElevatedButton
Text(
  "ชื่อแอปเปิ้ลจำนวน x ผล ราคาผล x บาท รวมลูกค้าต้องจ่ายเงิน x บาท") // Text

```

ภาพประกอบ 4.148 กำหนด Style ของปุ่ม (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 12)

1.7 เมื่อทำการกำหนดสีให้กับปุ่มได้แล้ว จากนั้นให้ทำการจัดระยะของปุ่มด้วย Padding

1.7.1 ภายใน ButtonStyle(...) ให้ทำการเพิ่มโค้ด padding: โดยการพิมพ์ , (Comma) เพื่อเป็นตัวคั่นระหว่างโค้ดต่อจาก backgroundColor: จากนั้นพิมพ์ padding:

```

ElevatedButton(
  onPressed: () {},
  child: Text("คำนวณ"),
  style: ButtonStyle(backgroundColor: MaterialStat
  pad
), [0] padding: MaterialStateProperty<EdgeInsets
), // ElevatedButton
Text("ชื่อแอปเปิ้ลจำนวน x ผล ราคาผล x บาท รวมลูกค้าต้อง
],),), // Column // Center

```

ภาพประกอบ 4.149 จัดระยะของปุ่มด้วย Padding

ที่มา : ฌปภัช วรรณตรง (2564 : 13)



1.7.2 จากนั้นให้ทำการเพิ่ม `MaterialStateProperty.all(...)` โดยการพิมพ์หรือคัดลอกโค้ดที่ได้พิมพ์ไว้ก่อนหน้าแล้วมาใส่ต่อจาก `padding`:

```

ElevatedButton(
  onPressed: () {},
  child: Text("คำนวณ"),
  style: ButtonStyle(backgroundColor: MaterialStateProperty.all(
    padding: ), // ButtonStyle
  ), // ElevatedButton
  Text("ชื่อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่าย"),
],),), // Column // Center
); // Padding
}

```

ภาพประกอบ 4.150 จักระยะของปุ่มด้วย Padding (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 13)

```

 SizedBox(height: 20.0,),
 ElevatedButton(
   onPressed: () {},
   child: Text("คำนวณ"),
   style: ButtonStyle(backgroundColor: MaterialStateProperty.all(
     padding: MaterialStateProperty.all]), // ButtonStyle
 ), // ElevatedButton
 Text("ชื่อแอปเปิ้ลจำนวน X ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท"),
 ],), // Column
 ), // Center

```

ภาพประกอบ 4.151 จักระยะของปุ่มด้วย Padding (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 13)

1.7.3 ภายใน all(...) ให้ทำการเพิ่ม EdgeInsets.fromLTRB(...) เข้าไป โดยการพิมพ์ ed จากนั้นจะปรากฏหน้าต่างชุดคำสั่งขึ้นมา ให้ทำการเลือกไปที่ EdgeInsets.fromLTRB(...) โดยการกดคลิกหรือกด Enter

```
SizedBox(height: 20.0,),
ElevatedButton(
  onPressed: (){},
  child: Text("คำนวณ"),
  style: ButtonStyle(backgroundColor: MaterialStateProperty.all( Color
padding: MaterialStateProperty.all(ed)), // ButtonStyle
), // ElevatedButton
Text("ซื้อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท
EdgeInsets.fromLTRB... (double
EdgeInsets.fromViewPadding(...
EdgeInsets.only(...
```

ภาพประกอบ 4.152 จัดระยะของปุ่มด้วย Padding (ต่อ)  
ที่มา : ฌปภัช วรรณตรง (2564 : 13)

```
height: 20.0,),
utton(
ed: (){},
Text("คำนวณ"),
ButtonStyle(backgroundColor: MaterialStateProperty.all( Color(0xff8915b0)),
: MaterialStateProperty.all(EdgeInsets.fromLTRB(left, top, right, bottom))), //
ElevatedButton
อปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท"), // Column
```

ภาพประกอบ 4.153 จัดระยะของปุ่มด้วย Padding (ต่อ)  
ที่มา : ฌปภัช วรรณตรง (2564 : 13)

1.7.4 จากนั้นให้ทำการกำหนดระยะห่างของปุ่ม โดยทำการกำหนดใน fromLTRB(...) หน่วยเป็น Pixel

```
ElevatedButton(
  onPressed: (){},
  child: Text("คำนวณ"),
  style: ButtonStyle(backgroundColor: MaterialStateProperty.all( Color(0xff8915b0
padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(50, 20, 50, 20))), // BU
), // ElevatedButton
Text("ซื้อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท"), // Column
```

ภาพประกอบ 4.154 จัดระยะของปุ่มด้วย Padding (ต่อ)  
ที่มา : ฌปภัช วรรณตรง (2564 : 13)

```

ElevatedButton(
  onPressed: () {},
  child: Text("คำนวณ"),
  style: ButtonStyle(
    backgroundColor: MaterialStateProperty.all(Color(0xffe3d924)),
    padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(50, 20, 50, 20)),
    textStyle: MaterialStateProperty.all(TextStyle(fontSize: 30))
  )), // ButtonStyle // ElevatedButton
  Text(
    "ชื่อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท" // Text
  ],
), // Column // Center
); // Padding
}
}

```

ภาพประกอบ 4.155 จัดระยะของปุ่มด้วย Padding (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 13)

หมายเหตุ
----------

สังเกตวงเล็บของโค้ดให้ดี การจัดระยะด้วย Padding ต้องอยู่ภายในโค้ดของ ButtonStyle
--

1.8 เมื่อทำการเพิ่มโค้ด padding: ได้แล้ว ให้ทำการเพิ่มโค้ด textStyle: ให้กับปุ่ม โดยทำตามขั้นตอนการเพิ่ม padding:

1.8.1 ทำการเพิ่มโค้ด textStyle: โดยการพิมพ์ text จากนั้นจะปรากฏหน้าต่างชุดคำสั่งขึ้นมา ให้ทำการเลือกไปที่ textStyle: โดยการกดคลิกหรือกด Enter

```

ElevatedButton(
  onPressed: (){},
  child: Text("คำนวณ"),
  style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Colors.red),
  padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(10, 10, 10, 10)), // ButtonStyle // ElevatedButton
  Text("๑) textStyle: MaterialStateProperty

```

ภาพประกอบ 4.156 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 14)

```

ElevatedButton(
  onPressed: (){},
  child: Text("คำนวณ"),
  style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Colors.red),
  padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(10, 10, 10, 10)), // ButtonStyle // ElevatedButton
  textStyle: TextStyle(color: Colors.red), // ButtonStyle // ElevatedButton
  Text("ชื่อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้่าด

```

ภาพประกอบ 4.157 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 14)

1.8.2 จากนั้นให้ทำการเพิ่ม `MaterialStateProperty.all(...)` โดยการพิมพ์หรือคัดลอกโค้ดก่อนหน้า เหมือนขั้นตอนของการเพิ่ม padding:

```
ElevatedButton(
  onPressed: () {},
  child: Text("คำนวณ"),
  style: ButtonStyle(backgroundColor: MaterialStateProperty.all(EdgeInsets.from
  padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(50, 20, 50, 20)), // Butt
  textStyle: MaterialStateProperty.all(TextStyle(...))), // ButtonStyle //
  Text("ชื่อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้อง
```

ภาพประกอบ 4.158 เพิ่มโค้ด `textStyle` ให้กับปุ่ม (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 14)

1.8.3 ภายใน `all(...)` ให้ทำการใส่โค้ด `TextStyle(...)` เข้าไป

```
ElevatedButton(
  onPressed: () {},
  child: Text("คำนวณ"),
  style: ButtonStyle(backgroundColor: MaterialStateProperty.all(□ Color
  padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(50, 20, 50, 20)
  textStyle: (MaterialStateProperty.all(TextStyle(...))), // ButtonStyle //
  Text("ชื่อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้อง TextStyle(...) ({bool inh
  TextStyle
```

ภาพประกอบ 4.159 เพิ่มโค้ด `textStyle` ให้กับปุ่ม (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 14)



#### 1.8.4 ภายใน TextStyle(...) ให้ทำการเพิ่ม fontSize: เข้าไป

```
ElevatedButton(
  onPressed: () {},
  child: Text("คำนวณ"),
  style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Color(0xff891
padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(50, 20, 50, 20)),
  textStyle: (MaterialStateProperty.all(TextStyle(fonts))), ), // ButtonStyle
Text("ซื้อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท"), // Colu
  [e] fontSize:
  [e] fontStyle:
```

ภาพประกอบ 4.160 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 14)

#### 1.8.5 สามารถกำหนดขนาดของ Font ได้โดยมีหน่วยเป็น pt (Point)

```
ElevatedButton(
  onPressed: () {},
  child: Text("คำนวณ"),
  style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Color(0xff891
padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(50, 20, 50, 20)),
  textStyle: (MaterialStateProperty.all(TextStyle(fontSize: 30))), ), // But
Text("ซื้อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท"), // Colu
```

ภาพประกอบ 4.161 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 14)

```
ElevatedButton(
  onPressed: () {},
  child: Text("คำนวณ"),
  style: ButtonStyle(
    backgroundColor: MaterialStateProperty.all(Color(0xffe3d924)),
    padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(50, 20, 50, 20)),
    textStyle: MaterialStateProperty.all(TextStyle(fontSize: 30))
  ), // ButtonStyle // ElevatedButton
Text(
  "ซื้อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท") // Text
```

ภาพประกอบ 4.162 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 14)

1.9 เมื่อทำการแก้ไขโค้ดให้กับปุ่มแล้ว จากนั้นให้ทำการเพิ่มโค้ด textStyle ให้กับคำอธิบาย

ภายใน Text(...) ต่อจากเครื่องหมาย “” ให้คั่นด้วย , เพื่อทำการเพิ่มโค้ดเข้าไป

```
onPressed: () {},
child: Text("คำนวณ"),
style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Color(0xff8959)),
padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(50, 20, 50, 20)),
textStyle: (MaterialStateProperty.all(TextStyle(fontSize: 30))), // B
Text("ซื้อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท" , )]
), // Column
/ Center
```

ภาพประกอบ 4.163 เพิ่มโค้ด textStyle ให้กับคำอธิบาย

ที่มา : ฅนปักษ์ วรณตรง (2564 : 15)

พิมพ์โค้ด style:

```
onPressed: () {},
child: Text("คำนวณ"),
style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Color(0xff8959)),
padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(50, 20, 50, 20)),
textStyle:MaterialStateProperty.all(TextStyle(fontSize: 30)) ), // ButtonSt
Text("ซื้อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท" , sty],
), // Column // Center
dding
```

ภาพประกอบ 4.164 เพิ่มโค้ด textStyle ให้กับคำอธิบาย (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 15)

จากนั้นเพิ่มโค้ด TextStyle(...) ต่อจาก style:

```
onPressed: () {},
child: Text("คำนวณ"),
style: ButtonStyle(backgroundColor: MaterialStateProperty.all(□ Color(0xff8915b0)),
padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(50, 20, 50, 20)),
textStyle: MaterialStateProperty.all(TextStyle(fontSize: 30))), // ButtonStyle // ElevatedButton
Text("ชื่อแอปเปิ้ลจำนวน x ผล ราคาผล x บาท รวมลูกค้าต้องจ่ายเงิน x บาท", style: TextStyle(
)), // Column // Center
padding
```

ภาพประกอบ 4.165 เพิ่มโค้ด textStyle ให้กับคำอธิบาย (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 15)

ภายใน TextStyle(...) ให้ทำการเพิ่มโค้ด fontSize: เข้าไป

```
onPressed: () {},
child: Text("คำนวณ"),
style: ButtonStyle(backgroundColor: MaterialStateProperty.all(□ Color(0xff8915b0)),
padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(50, 20, 50, 20)),
textStyle: MaterialStateProperty.all(TextStyle(fontSize: 30))), // ButtonStyle // ElevatedButton
Text("ชื่อแอปเปิ้ลจำนวน x ผล ราคาผล x บาท รวมลูกค้าต้องจ่ายเงิน x บาท", style: TextStyle(fontSize: ), ),
Column // Center
```

ภาพประกอบ 4.166 เพิ่มโค้ด textStyle ให้กับคำอธิบาย (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 15)

สามารถกำหนดขนาดของ Font ได้โดยมีหน่วยเป็น pt (Point) ได้เช่นเดียวกัน

```
onPressed: () {},
child: Text("คำนวณ"),
style: ButtonStyle(backgroundColor: MaterialStateProperty.all(□ Color(0xff8915b0)),
padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(50, 20, 50, 20)),
textStyle: MaterialStateProperty.all(TextStyle(fontSize: 30))), // ButtonStyle // ElevatedButton
Text("ชื่อแอปเปิ้ลจำนวน x ผล ราคาผล x บาท รวมลูกค้าต้องจ่ายเงิน x บาท", style: TextStyle(fontSize: 20)),
Column // Center
```

ภาพประกอบ 4.167 เพิ่มโค้ด textStyle ให้กับคำอธิบาย (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 15)



```

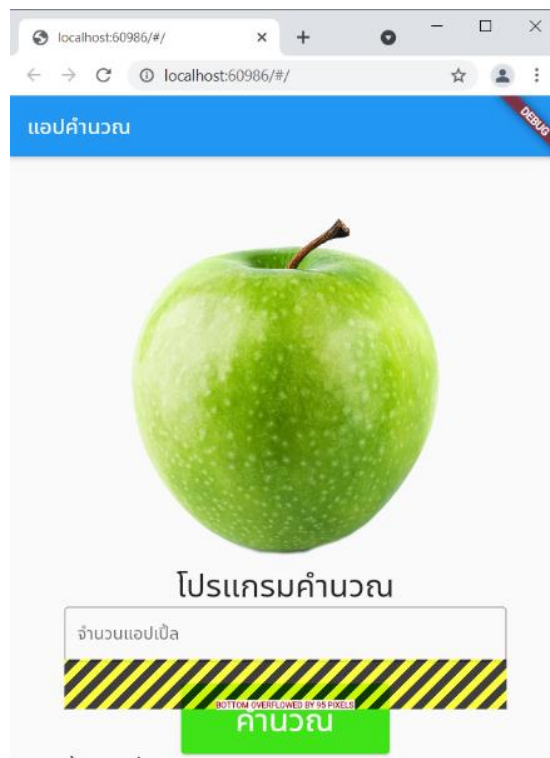
ElevatedButton(
  onPressed: () {},
  child: Text("คำนวณ"),
  style: ButtonStyle(
    backgroundColor: MaterialStateProperty.all(Color(0xffe3d924)),
    padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(50, 20, 50, 20)),
    textStyle: MaterialStateProperty.all(TextStyle(fontSize: 30))
  )), // ButtonStyle // ElevatedButton
  Text(
    "ชื่อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท", style: TextStyle(fontSize: 20),) /
],
// Column // Center
// Padding

```

ภาพประกอบ 4.168 เพิ่มโค้ด textStyle ให้กับคำอธิบาย

ที่มา : ณปภัช วรรณตรง (2564 : 15)

1.10 จากนั้นให้ทำการสั่งการทำงานของ Flutter แต่เมื่อย่อขนาดของหน้าแอปพลิเคชัน จะพบปัญหาตามภาพ



ภาพประกอบ 4.169 ผลลัพธ์สั่งการทำงานของ Flutter แล้วมีข้อผิดพลาด

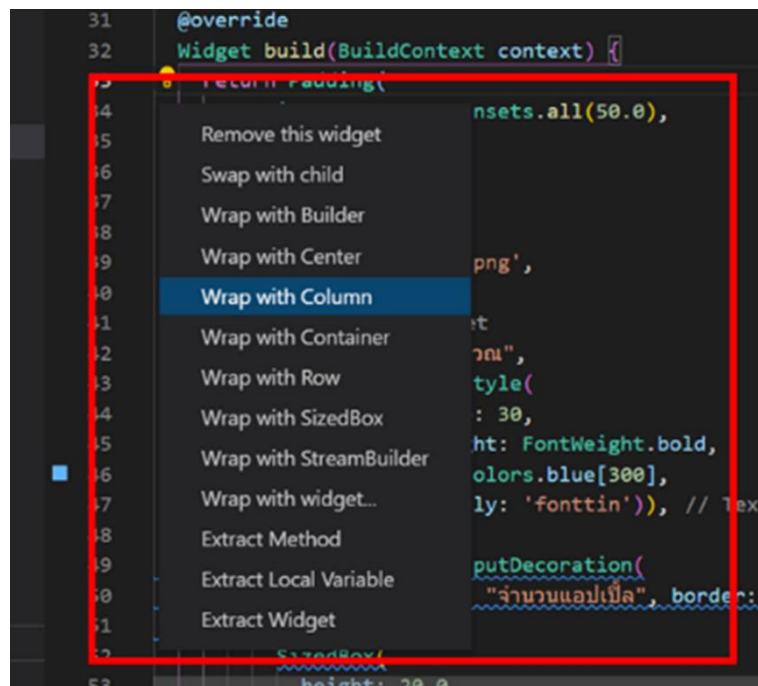
ที่มา : ณปภัช วรรณตรง (2564 : 16)

## 2. ListView Widget

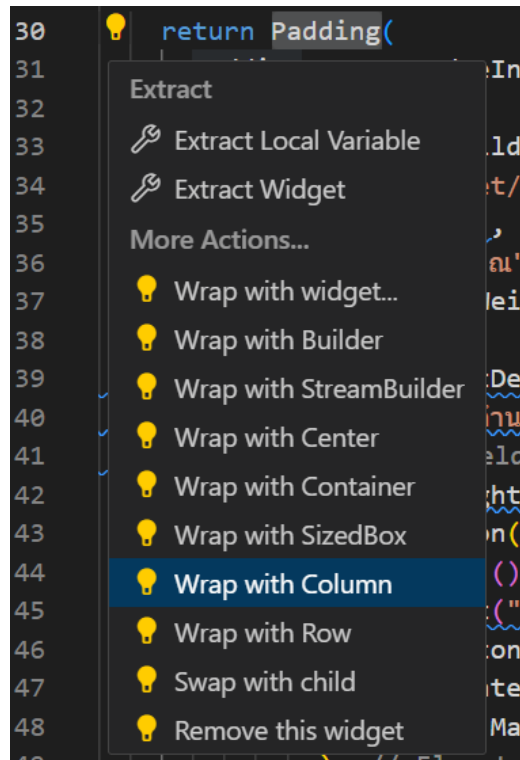
มีลักษณะคล้าย Column วิดเจ็ต และ Row วิดเจ็ต ใช้เพื่อแสดงวิดเจ็ตลูกเรียงลำดับในแนวตั้ง หรือในแนวนอนตามลำดับ โดย ListView สามารถ แสดงเนื้อหาที่มีขนาดใหญ่กว่าหน้าจอของอุปกรณ์ สามารถเลื่อนหน้าจอ (Scroll) ทั้งในแนวตั้งหรือแนวนอนก็ได้ (จีราวุธ วารินทร์, 2564 : 267)

2.1 ให้ทำการแก้ไข Scroll โดยการใช้ชุดคำสั่งของ ListView ซึ่งใช้วิธีเดียวกันกับขั้นตอนที่ 6 แต่ให้เลือกเป็น Warp with Column

2.1.1 คลิกที่ Padding(...) จากนั้นคลิกที่หลอดไฟ เลือกเป็น Warp with Column



ภาพประกอบ 4.170 แก้ไขการ Scroll โดยการใช้ชุดคำสั่งของ ListView  
ที่มา : ฅนปลัซ วรณตรง (2564 : 17)



ภาพประกอบ 4.171 แก้ไขการ Scroll โดยการใส่ชุดคำสั่งของ ListView (ต่อ)  
ที่มา : ฅนปักษ์ วรรณตรง (2564 : 17)

2.1.2 เมื่อทำการเลือก Warp with Column แล้ว จะมีชุดคำสั่ง Column(...) มาครอบ Padding(...) ไว้

```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Column(
31       children: [
32         Padding(
33           padding: const EdgeInsets.all(8.0),
34           child: Center(

```

ภาพประกอบ 4.172 แก้ไขการ Scroll โดยการใส่ชุดคำสั่งของ ListView (ต่อ)  
ที่มา : ฅนปักษ์ วรรณตรง (2564 : 17)

2.2 เมื่อได้ Column(...) มาแล้ว ให้ทำการเปลี่ยนจาก Column(...) เป็น ListView(...) แล้วทำการสั่งการทำงาน

```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return Column(
31       children: [
32         Padding(
33           padding: const EdgeInsets.all(8.0),

```

ภาพประกอบ 4.173 เปลี่ยนจาก Column เป็น ListView

ที่มา : ฌปภัช วรรณตรง (2564 : 18)

```

27 class _HomeState extends State<Home> {
28   @override
29   Widget build(BuildContext context) {
30     return ListView(
31       children:
32         Padding(
33           padding:
34             child:

```

ภาพประกอบ 4.174 เปลี่ยนจาก Column เป็น ListView (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 18)

## 2.3 ใช้คำสั่ง flutter run แล้วผลลัพธ์ที่ได้จากการแก้ไขการ Scroll จะได้ตามภาพ



ภาพประกอบ 4.175 ผลลัพธ์ของแอปพลิเคชัน

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 19)

2.3 ต่อไปคือการสร้างโค้ดเพื่อทำการคำนวณค่าที่ได้รับเข้ามา โดยจะใช้การกำหนดตัวแปรสำหรับรับค่า และใช้การสร้าง function class ที่ `_HomeState` สำหรับฟังก์ชันการคำนวณ

2.3.1 ภายใน `State<home>{}` ให้ทำการเพิ่ม `TextEditingController` `quantity = TextEditingController();` เข้าไปโดยการพิมพ์ `TextE` จากนั้นจะปรากฏหน้าต่างชุดโค้ดขึ้นมา ให้ทำการเลือก `TextEditingController` โดยการคลิกหรือกด `Enter` เพื่อเป็นการประกาศประเภทของข้อมูลที่ได้รับเข้ามา

### หมายเหตุ

`controller` เป็นการกำหนดคุณสมบัติเพื่อควบคุมข้อความหรือข้อมูลที่อยู่ในเท็กซ์ฟิลด์ โดยจะกำหนดข้อมูลในการใช้งานเป็นแบบ `TextEditingController` ประกอบด้วยคุณสมบัติสำคัญ คือ `text` และ `selection` ซึ่งทำหน้าที่ในการแสดงผลข้อความตามต้องการ และแสดงการเลือกข้อความ ตามลำดับ (เอกรินทร์ วัญญูเลิศสกุล, 2563 : 113)

```

27 class _HomeState extends State<Home> {
28
29   TextE
30   TextEditingController
31   TextEditingValue
32   @override TextEditingDelta

```

ภาพประกอบ 4.176 สร้างโค้ดเพื่อใช้สำหรับคำนวณค่าที่ได้รับเข้ามา  
 ที่มา : ฌปภัช วรรณตรง (2564 : 20)

2.3.2 เมื่อทำการเพิ่มโค้ด TextEditingController แล้ว ให้ทำการกำหนดชื่อของ  
 ตัวเก็บข้อมูลและโค้ด TextEditingController();

```

25
26 class _HomeState extends State<Home> {
27   TextEditingController quantity = t;
28   textDirectionToAxisDirecti...
29   toDiagnosticsNode(...)
30   @override toString()

```

ภาพประกอบ 4.177 สร้างโค้ดเพื่อใช้สำหรับคำนวณค่าที่ได้รับเข้ามา (ต่อ)  
 ที่มา : ฌปภัช วรรณตรง (2564 : 20)

```

25
26 class _HomeState extends State<Home> {
27   TextEditingController quantity = TextEditingController();
28
29

```

ภาพประกอบ 4.178 สร้างโค้ดเพื่อใช้สำหรับคำนวณค่าที่ได้รับเข้ามา (ต่อ)  
 ที่มา : ฌปภัช วรรณตรง (2564 : 20)

2.3.3 หลังจากนั้นให้ทำการกำหนดตัวแปรขึ้นมาหนึ่งตัว โดยใช้ประเภทของตัวแปรเป็น double

```

25
26 class _HomeState extends State<Home> {
27   TextEditingController quantity = TextEditingController();
28   dou
29   double
30   Double
31   @ov DoubleLinkedList

```

ภาพประกอบ 4.179 กำหนดตัวแปรประเภท double

ที่มา : ฌปภัช วรรณตรง (2564 : 20)

```

25
26 class _HomeState extends State<Home> {
27   TextEditingController quantity = TextEditingController();
28   double price = 10;
29
30

```

ภาพประกอบ 4.180 กำหนดตัวแปรประเภท double (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 20)

```

class _HomeState extends State<Home> {
  TextEditingController quantity = TextEditingController(); //เก็บค่าที่ผู้ใช้กรอก
  double price = 10; //กำหนดราคาสินค้า
}

```

ภาพประกอบ 4.181 สร้างโค้ดเพื่อใช้สำหรับคำนวณค่าที่ได้รับเข้ามา (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 20)

2.4 กำหนดโค้ดตัวแปรรับค่าใน TextField(),  
 เลื่อนหา TextField(...) จากนั้นภายใน (...) ให้ทำการเพิ่มโค้ด controller: เพื่อรับค่า  
 และส่งคำนวณ

```
TextField(  
  contr  
  decor [?] controller: TextEditingController?  
  lab [?] contentInsertionConfiguration:  
  bor [?] contextMenuBuilder:  
  bor [?] undoController:
```

ภาพประกอบ 4.182 เพิ่มโค้ดเพื่อรับค่าและส่งคำนวณ

ที่มา : ฌปภัช วรรณตรง (2564 : 21)

```
TextField(  
  controller: q,  
  decoration: I quantity  
  labelText: I querySelector(...)  
  labelText: querySelectorAll(...)
```

ภาพประกอบ 4.183 เพิ่มโค้ดเพื่อรับค่าและส่งคำนวณ (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 21)

```
Text("โปรแกรมคำนวณ",  
  style: TextStyle(  
    fontSize: 30,  
    fontWeight: FontWeight.bold,  
    color: Colors.blue[300],  
    fontFamily: 'fonttin')), // TextStyle // Text  
TextField(  
  controller: quantity,  
  decoration: InputDecoration(  
    labelText: "จำนวนแอปเปิ้ล", border: OutlineInputBorder()), // InputDecoration  
  ), // TextField
```

ภาพประกอบ 4.184 เพิ่มโค้ดเพื่อรับค่าและส่งคำนวณ (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 21)



2.5 หลังจากนั้นให้ทำการเพิ่มชุดคำสั่ง print เพื่อตรวจสอบค่าที่ได้รับมาจากหน้าเว็บ แอปพลิเคชันเข้าไปใน onPressed: () {}

2.5.1 ไปที่ onPressed: () {} จากนั้นภายใน {...} ให้ทำการเพิ่มโค้ด print(...) เข้าไป

```
ElevatedButton(
  onPressed: () {},
  child: Text("คำนวณ"),
  style: ButtonStyle(backgroundColor: MaterialStateProperty.all(
    padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(50, 20, 50,
  textStyle: MaterialStateProperty.all(TextStyle(fontSize: 30))), /
```

ภาพประกอบ 4.185 เพิ่มโค้ด print(...)

ที่มา : ฌปภัช วรรณตรง (2564 : 22)

```
ElevatedButton(
  onPressed: () {
    pri
  },
  child: primaryFocus
  style: PrimaryScrollController
  paddi: PrioritizedAction
  texts: PrioritizedIntents
  Text("ช PrimaryScrollController(...)
```

ภาพประกอบ 4.186 เพิ่มโค้ด print(...) (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 22)

2.5.2 ภายใน print(...) ให้ทำการเพิ่มข้อความเข้าไปโดยใช้เครื่องหมาย “” เป็นที่เก็บข้อความ

```
ElevatedButton(
  onPressed: () {
    print("Apple Quantity : ");
  },
```

ภาพประกอบ 4.187 เพิ่มโค้ด print(...) (ต่อ)  
ที่มา : ฌปภัช วรรณตรง (2564 : 22)

2.5.3 ต่อจาก “Apple Quantity:” ให้ทำการเพิ่มตัวแปรโดยใช้ \$ เป็นตัวเก็บค่าตัวแปร จากนั้นให้เพิ่มตัวแปรที่กำหนดมา

```
ElevatedButton(
  onPressed: () {
    print("Apple Quantity : {}");
  },
  child: Text("จำนวน"),
```

ภาพประกอบ 4.188 เพิ่มโค้ด print(...) (ต่อ)  
ที่มา : ฌปภัช วรรณตรง (2564 : 22)

2.5.4 โดยหามี .text ตัวแปรควรมี {} ครอบไว้ หากไม่มีไม่จำเป็นต้องใส่ {}

```
ElevatedButton(
  onPressed: () {
    print("Apple Quantity : ${quantity}");
  },
```

ภาพประกอบ 4.189 เพิ่มโค้ด print(...) (ต่อ)  
ที่มา : ฌปภัช วรรณตรง (2564 : 22)

```

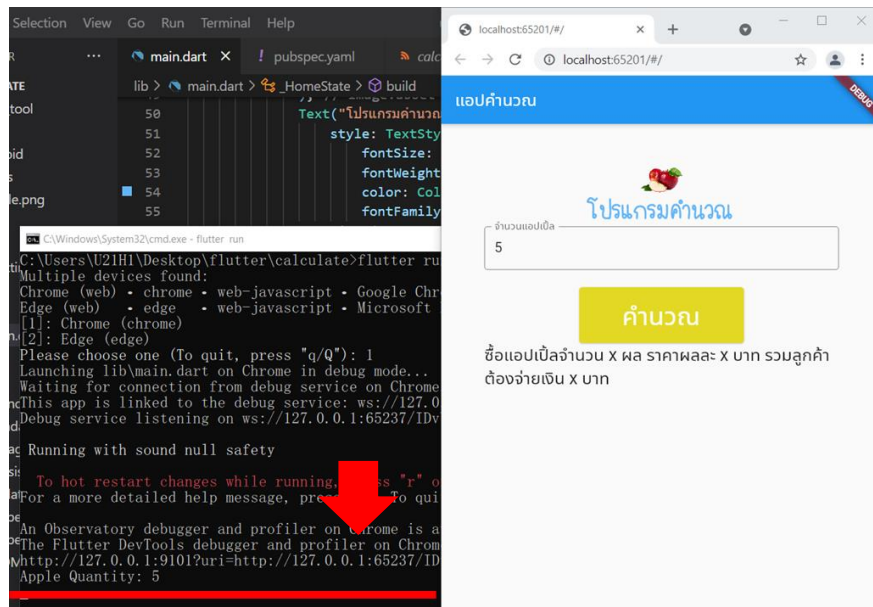
ElevatedButton(
  onPressed: () {
    print("Apple Quantity: ${quantity.text}");
  },
)

```

ภาพประกอบ 4.190 เพิ่มโค้ด print(...) (ต่อ)

ที่มา : ฅปภัช วรรณตรง (2564 : 22)

2.6 ทำการทดสอบโดยการใส่ค่าเข้าไปใน TextField และลองคลิกปุ่มคำนวณที่ได้กำหนดไว้ จากนั้นให้ไปดูผลลัพธ์ที่หน้าต่าง cmd



ภาพประกอบ 4.191 ทดสอบรับค่าและตรวจสอบค่าที่ได้รับมา

ที่มา : ฅปภัช วรรณตรง (2564 : 23)

2.7 เมื่อได้ผลลัพธ์ที่ถูกต้องแล้ว ให้ทำการเพิ่มโค้ดสำหรับเรียกใช้การคำนวณ จากนั้นให้ทำการเพิ่มตัวแปรสำหรับรับค่าการคำนวณ

2.7.1 ใน onPressed: () {...} ให้ทำการกำหนดตัวแปรสำหรับรับค่าผลการคำนวณ และนำค่าการคำนวณนั้นไปแสดง

```
ElevatedButton(
  onPressed: () {
    v
    var
    p [0] visibleForTesting
```

ภาพประกอบ 4.192 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ

ที่มา : ฌปภัช วรรณตรง (2564 : 24)

2.7.2 ให้ทำการกำหนดโค้ดเป็น var cal เพื่อเป็นตัวแปรสำหรับรับค่าผลการคำนวณ

```
ElevatedButton(
  onPressed: () {
    var cal =
    print("Apple Quantity: ${quantity.text}");
  },
```

ภาพประกอบ 4.193 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 24)

2.7.3 จากนั้นให้ทำการกำหนดสูตรคำนวณเพื่อนำไปใช้คำนวณค่าที่ได้รับมา แล้วเอาไปเก็บไว้ในค่าตัวแปรที่ได้ทำการกำหนดไว้

```
ElevatedButton(
  onPressed: () {
    var cal = q
    print("App1
  },
  child: Text("จำนวน"),
)
```

ภาพประกอบ 4.194 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 24)

2.7.4 กำหนดโค้ดรับค่าคำนวณ โดยพิมพ์โค้ด `int.parse(...)` เพื่อแปลงค่าที่ได้รับมาจาก text เป็น int ก่อนนำไปคำนวณ

```
ElevatedButton(
  onPressed: () {
    var cal = int.parse(quantity.text);
    print("Apple Quantity : ${quantity.text}");
  },
  child: Text("จำนวน"),
)
```

ภาพประกอบ 4.195 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 24)

2.7.5 ภายใน `int.parse(...)` ให้พิมพ์ตัวแปรรับค่าที่ได้กำหนดไว้ จากนั้นให้นำไปคูณด้วยตัวแปรราคาด้วยการพิมพ์ `*price`

```
ElevatedButton(
  onPressed: () {
    var cal = int.parse(quantity.text)*price ;
    print("Apple Quantity : ${quantity.text}");
  },
  child: Text("จำนวน"),
```

ภาพประกอบ 4.196 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 24)

2.7.6 เมื่อทำการเพิ่มตัวแปรสำหรับรับค่าและนำไปคำนวณแล้ว ใน `print(...)` ต่อจาก {} ให้ทำการเพิ่มโค้ดสำหรับแสดงผลการคำนวณ

```
ElevatedButton(
  onPressed: () {
    var cal = int.parse(quantity.text)*double.parse(price.text) ;
    print("Apple Quantity : ${quantity.text} Total : $ca");
  },
  child: Text("จำนวน"),
```

ภาพประกอบ 4.197 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 24)

```
ElevatedButton(
  onPressed: () {
    var cal = int.parse(quantity.text)*double.parse(price.text) ;
    print("Apple Quantity : ${quantity.text} Total : $cal path");
  },
```

ภาพประกอบ 4.198 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 24)

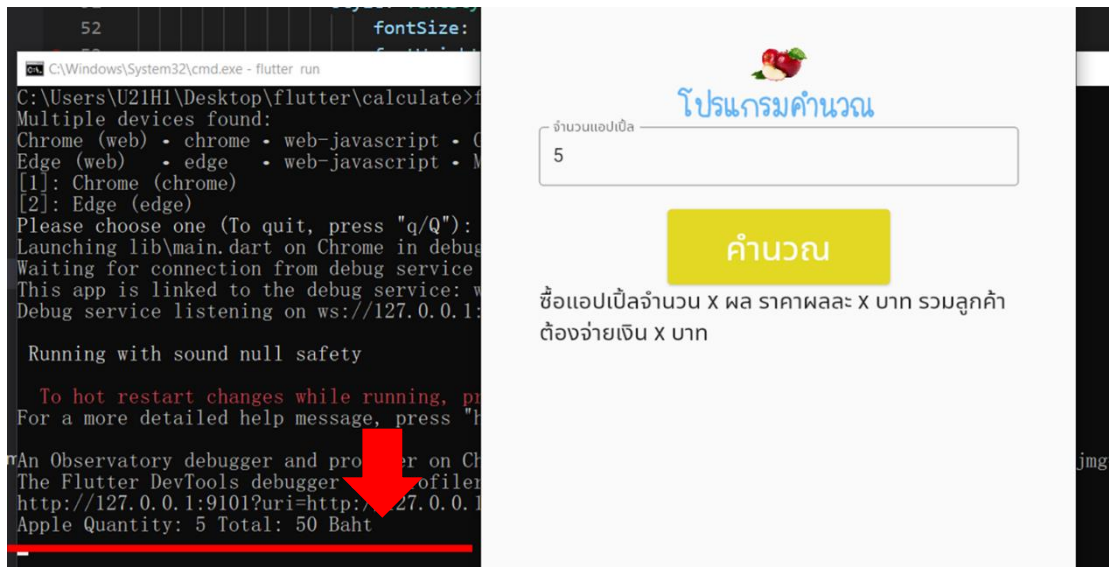
```

ElevatedButton(
  onPressed: () {
    var cal = int.parse(quantity.text)*price;
    print("Apple Quantity: ${quantity.text} Total: $cal Baht");
  },
)

```

ภาพประกอบ 4.199 เพิ่มโค้ดสำหรับการคำนวณและตัวแปรสำหรับรับค่าการคำนวณ (ต่อ)  
ที่มา : ฌปภัช วรรณตรง (2564 : 24)

2.8 เมื่อทำการเพิ่มโค้ดการคำนวณเสร็จแล้ว ให้ทำการใส่ค่าใน TextField จากนั้นให้คลิกปุ่มคำนวณแล้วไปดูผลลัพธ์ที่หน้าต่าง cmd อีกครั้งตามภาพ



ภาพประกอบ 4.200 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ)  
ที่มา : ฌปภัช วรรณตรง (2564 : 25)

2.8 กำหนดตัวแปรสำหรับเก็บค่าผลลัพธ์การคำนวณ และสร้างฟังก์ชัน initState() สำหรับกำหนดค่าเริ่มต้น

2.8.1 เมื่อทำการเพิ่มโค้ดสำหรับรับค่าและคำนวณแล้ว ให้ทำการเพิ่มโค้ดสำหรับรับค่าผลการคำนวณ โดยการคัดลอกโค้ดที่เคยพิมพ์ไว้แล้วมาเปลี่ยนชื่อของตัวแปร

```

27 class _HomeState extends State<Home> {
28
29   TextEditingController quantity = TextEditingController();
30   double price = 10;
31
32

```

ภาพประกอบ 4.201 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){}

ที่มา : ฌปภัช วรรณตรง (2564 : 26)

```

25
26 class _HomeState extends State<Home> {
27   TextEditingController quantity =TextEditingController();
28   double price = 10;
29
30   TextEditingController quantity =TextEditingController();
31

```

ภาพประกอบ 4.202 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){} (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 26)

```

25
26 class _HomeState extends State<Home> {
27   TextEditingController quantity =TextEditingController();
28   double price = 10;
29
30   TextEditingController result =TextEditingController();
31

```

ภาพประกอบ 4.203 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){} (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 26)



2.8.2 จากนั้นให้ทำการเพิ่ม initState() {...} โดยการพิมพ์ ini จากนั้นจะปรากฏหน้าต่างชุดคำสั่งขึ้นมาให้ทำการเลือกไปที่ initState() {...} โดยการคลิกหรือกด Enter

```

26 class _HomeState extends State<Home> {
27   TextEditingController quantity =TextEditingController();
28   double price = 10;
29
30   ⚠TextEditingController result =TextEditingController();
31   ini
32   initState() { ... } () → void
33   InitialRouteListFactory

```

ภาพประกอบ 4.204 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){} (ต่อ)  
ที่มา : ฌปภัช วรรณตรง (2564 : 26)

```

29
30   TextEditingController result =TextEditingController();
31   @override
32   void initState() {
33     // TODO: implement initState
34     super.initState();
35   }
36

```

ภาพประกอบ 4.205 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){} (ต่อ)  
ที่มา : ฌปภัช วรรณตรง (2564 : 26)

2.8.3 เมื่อทำการเพิ่ม initState(){...} แล้ว ภายใน {...} ให้ทำการเพิ่ม result.text ต่อจาก (); โดยการพิมพ์ res จากนั้นจะปรากฏหน้าต่างชุดคำสั่งขึ้นมาให้ทำการเลือกไปที่ result โดยการคลิกหรือกด Enter

```

31  @override
32  void initState() {
33    // TODO: implement initState
34    super.initState();
35
36    res
37  }
38

```

result TextEditingController  
ResizeImage

ภาพประกอบ 4.206 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){} (ต่อ)  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 26)

```

31  @override
32  void initState() {
33    // TODO: implement initState
34    super.initState();
35
36    result.text
37  }
38

```

ภาพประกอบ 4.207 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){} (ต่อ)  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 26)

2.8.4 จากนั้นให้ทำการเพิ่มข้อความ เพื่ออธิบายผลการคำนวณ ซึ่งจะเป็นข้อความที่แสดงออกทางหน้าจอแอปพลิเคชัน

```
@override
void initState() {
  // TODO: implement initState
  super.initState();

  result.text = "ซื้อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท";
}
```

ภาพประกอบ 4.208 กำหนดตัวแปรเริ่มต้นและสร้างฟังก์ชัน initState(){} (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 26)

```
class _HomeState extends State<Home> {
  TextEditingController quantity = TextEditingController(); //เก็บค่าที่ผู้ใช้กรอก
  double price = 10; //กำหนดราคาสินค้า

  TextEditingController result = TextEditingController();
  @override
  void initState() { //ฟังก์ชันพิเศษสำหรับกำหนดค่าในตอนแรก
    // TODO: implement initState
    super.initState();

    result.text = "ซื้อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท";
  }
}
```

ภาพประกอบ 4.209 กำหนดตัวแปรเริ่มต้นและสร้างฟังก์ชัน initState(){} (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 26)

2.9 ทำการแทนที่คำอธิบายเดิมด้านล่างของผลลัพธ์ ด้วยตัวแปรผลลัพธ์ที่ได้สร้างไว้เพื่อทำการแสดงผลการคำนวณผ่านหน้าต่างแอปพลิเคชัน ด้วยการเปลี่ยนโค้ดของ Text(...) จากข้อความให้เป็นตัวแปรข้อความที่ได้ทำการกำหนดไว้

```
child: Text("คำนวณ"),
style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Color(0xff8915b0)),
padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(50, 20, 50, 20)),
textStyle: MaterialStateProperty.all(TextStyle(fontSize: 30))), // ButtonStyle
), // ElevatedButton
Text("ซื้อแอปเปิ้ลจำนวน X ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท", style: TextStyle(for
]), // Column
), // Center
Padding
```

ภาพประกอบ 4.210 เปลี่ยนข้อความของ Text(...) ให้เป็นตัวแปรข้อความ  
ที่มา : ฅนปลัซ วรณตรง (2564 : 27)

```
style: ButtonStyle(backgroundColor: MaterialStateProperty.
padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(50,
textStyle:MaterialStateProperty.all(TextStyle(fontSize: 30
Text(result.text, style: TextStyle(fontSize: ), ),
]),),), // Column // Center
/ Padding
```

ภาพประกอบ 4.211 เปลี่ยนข้อความของ Text(...) ให้เป็นตัวแปรข้อความ (ต่อ)  
ที่มา : ฅนปลัซ วรณตรง (2564 : 27)

```
style: ButtonStyle(
backgroundColor: MaterialStateProperty.all(Color(0xffe3d924)),
padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(50, 20, 50, 20)),
textStyle: MaterialStateProperty.all(TextStyle(fontSize: 30))
)), // ButtonStyle // ElevatedButton
Text(
result.text, style: TextStyle(fontSize: 20),) // Text
],
// Column // Center
```

ภาพประกอบ 4.212 เปลี่ยนข้อความของ Text(...) ให้เป็นตัวแปรข้อความ (ต่อ)  
ที่มา : ฅนปลัซ วรณตรง (2564 : 27)

2.10 สร้างฟังก์ชัน `setState()` ขึ้นมาสำหรับใช้ในการแสดงผลการคำนวณที่ได้มีการเปลี่ยนแปลง

2.10.1 ไปที่ `onPressed: () { ... }` จากนั้นให้ทำการเพิ่ม `class setState() { ... }` ต่อจาก `print(...)`; โดยการพิมพ์ `set` จากนั้นจะปรากฏหน้าต่างคำสั่งขึ้นมาให้ทำการเลือกไปที่ `setState() { ... }` โดยการคลิกหรือกด Enter

```
ElevatedButton(
  onPressed: () {
    var cal = int.parse(quantity.text)*price;
    print("Apple Quantity: ${quantity.text} Total: $cal bath");

    set
      setState() {};
      setEquals(...)
  },
  child: Text(
    setPluginHandler(...)
```

ภาพประกอบ 4.213 เพิ่ม `class setState() { ... }`

ที่มา : ฅนปักษ์ วรณตรง (2564 : 28)

2.10.2 จากนั้นภายใน `{ ... }` ให้ทำการนำตัวแปรค่าการคำนวณมาใส่

```
ElevatedButton(
  onPressed: () {
    var cal = int.parse(quantity.text)*price;
    print("Apple Quantity: ${quantity.text} Total: $cal bath");

    setState() {
      result.t
    });
  },
  child: Text(
    toJSBox
```

ภาพประกอบ 4.214 เพิ่ม `class setState() { ... }` (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 28)

```

    setState(() {
      result.text = "ชื่อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมต้องจ่ายเงินทั้งหมด x บาท";
    });
  },

```

ภาพประกอบ 4.215 เพิ่ม class setState((){...}) (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 28)

2.10.3 ในข้อความอธิบายให้ทำการเปลี่ยนผลการคำนวณจากข้อความอธิบายเป็นตัวแปรเพื่อแสดงผลการคำนวณ โดยใช้ \${ตัวแปรที่กำหนด} ซึ่งสามารถคัดลอกตัวแปรจาก print(...) มาวางได้

```

    setState(() {
      result.text = "ชื่อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมต้องจ่ายเงินทั้งหมด x บาท";
    });
  },

```

ภาพประกอบ 4.216 เปลี่ยนการแสดงผลการคำนวณ

ที่มา : ฌปภัช วรรณตรง (2564 : 28)

```

"ชื่อแอปเปิ้ลจำนวน ${quantity.text} ผล

```

ภาพประกอบ 4.217 เปลี่ยนการแสดงผลการคำนวณ (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 28)

```

ราคาผลละ 10 บาท

```

ภาพประกอบ 4.218 เปลี่ยนการแสดงผลการคำนวณ (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 28)

รวมต้องจ่ายเงินทั้งหมด `$cal` บาท";

ภาพประกอบ 4.219 เปลี่ยนการแสดงผลการคำนวณ (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 28)

```
ElevatedButton(
  onPressed: () {
    var cal = int.parse(quantity.text) * price;
    print("Apple: ${quantity.text} Total $cal baht");

    setState(() {
      result.text =
        "ซื้อแอปเปิ้ลจำนวน ${quantity.text} ผล ราคาผลละ 10 บาท รวมต้องจ่ายทั้งหมด $cal บาท";
    });
  },
)
```

ภาพประกอบ 4.220 สร้างฟังก์ชัน setState(){} (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 28)

2.11 จากนั้นให้ทำการสั่งการทำงานเพื่อทดสอบว่าแอปพลิเคชันทำงานถูกต้องหรือไม่



ภาพประกอบ 4.221 สั่งการทำงานเพื่อทดสอบแอปพลิเคชัน

ที่มา : ฌปภัช วรรณตรง (2564 : 29)

2.12 เมื่อได้ผลลัพธ์ที่ถูกต้องแล้วให้ทำการเพิ่มโค้ดในส่วนของ price ให้เป็นค่าที่สามารถแก้ไขได้

2.12.1 ให้ทำการสร้างตัวแปรสำหรับรับค่า price โดยการคัดลอกโค้ดกำหนดตัวแปร

```

26 class _HomeState extends State<Home> {
27     TextEditingController quantity =TextEditingController();
28     // double price = 10;
29     TextEditingController quantity =TextEditingController();
30
31     TextEditingController result =TextEditingController();
32     @override

```

ภาพประกอบ 4.222 เพิ่มโค้ดในส่วนของ price ที่สามารถแก้ไขค่าได้ (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 30)



2.12.2 จากนั้นให้ทำการเปลี่ยนชื่อตัวแปรให้เป็น price และทำการคอมเมนต์หรือปิดการทำงานของโค้ดตัวแปร double price = 10;

```

25
26 class _HomeState extends State<Home> {
27     TextEditingController quantity =TextEditingController();
28     // double price = 10;
29     TextEditingController price =TextEditingController();
30
31     TextEditingController result =TextEditingController();
32     @override

```

ภาพประกอบ 4.223 เพิ่มโค้ดในส่วนของ price ที่สามารถแก้ไขค่าได้ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 30)

```

class _HomeState extends State<Home> {
    TextEditingController quantity = TextEditingController(); //เก็บค่าที่ผู้ใช้กรอก
    //double price = 10; //กำหนดราคาสินค้า
    TextEditingController price = TextEditingController();
    TextEditingController result = TextEditingController();
    @override
    void initState() { //ฟังก์ชันพิเศษสำหรับกำหนดค่าเริ่มต้น
        // TODO: implement initState
        super.initState();

        result.text = "ชื่อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท";
    }
}

```

ภาพประกอบ 4.224 เพิ่มโค้ดในส่วนของ price ที่สามารถแก้ไขค่าได้ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 30)

2.13 จากนั้นให้ทำการเปลี่ยนโค้ดของฟังก์ชัน setState จากตัวแปรที่ได้กำหนดไว้เป็นตัวแปรที่ได้รับมาจากค่า cal

2.13.1 ใน onPressed: () {...} ให้เปลี่ยนจาก \*price เป็น double.parse(price.text) เพื่อนำค่าที่ได้รับมาคำนวณแทนค่าคงที่ที่ได้กำหนดไว้ก่อนหน้านี้

```
onPressed: () {
  var cal = int.parse(quantity.text)*price;
  print("Apple Quantity: ${quantity.text} Total: $cal bath");
}
```

ภาพประกอบ 4.225 แก้ไขโค้ดของผลการคำนวณ

ที่มา : ฌปภัช วรณตรง (2564 : 31)

```
onPressed: () {
  var cal = int.parse(quantity.text)*double.parse(price.text);
  print("Apple Quantity: ${quantity.text} Total: $cal bath");
}
```

ภาพประกอบ 4.226 แก้ไขโค้ดของผลการคำนวณ (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 31)

2.13.2 จากนั้นให้ทำการเปลี่ยนข้อความอธิบายเป็นตัวแปร \${price.text}

```
setState(() {
  result.text = "ชื่อแอปเปิ้ลจำนวน ${quantity.text} ผล ราคาผลช 10 บาท
});
```

ภาพประกอบ 4.227 แก้ไขโค้ดของผลการคำนวณ (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 31)

ราคาผลช  บาท

ภาพประกอบ 4.228 แก้ไขโค้ดของผลการคำนวณ (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 31)

### 2.13.3 จะได้ผลลัพธ์โค้ดทั้งหมดดังภาพ

```
ElevatedButton(
  onPressed: () {
    var cal = int.parse(quantity.text) * double.parse(price.text);
    print("Apple: ${quantity.text} Total $cal baht");

    setState(() {
      result.text =
        "ชื่อแอปเปิ้ลจำนวน ${quantity.text} ผล ราคาผลละ ${price.text} บาท รวมต้องจ่ายทั้งหมด $cal บาท";
    });
  },
),
```

ภาพประกอบ 4.229 แก้ไขโค้ดของผลการคำนวณ (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 31)

2.14 เมื่อทำการแก้ไขและเพิ่มโค้ดกำหนดค่าต่าง ๆ เรียบร้อยแล้ว ให้ทำการเพิ่มโค้ด textField() เข้าไปในแอปพลิเคชันสำหรับค่าราคาหรือค่าที่ต้องการใส่เข้ามาเพื่อคำนวณ

2.14.1 ภายใน Padding(...) ให้ทำการเพิ่ม TextField(...) สำหรับการรับค่า price ต่อจาก Text(...) โดยการคัดลอก TextField(...) ที่ได้กำหนดไว้ก่อนหน้านี้มาวาง จากนั้นให้เปลี่ยนโค้ดตัวแปรภายใน TextField(...) ที่ได้ทำการคัดลอกมา

```
53   TextField(
54     controller: quantity,
55     decoration: InputDecoration(
56       labelText: "จำนวนแอปเปิ้ล", border: OutlineInputBorder()),
57   ), // TextField
58   SizedBox(
59     height: 20.0,
60   ), // SizedBox
61   ElevatedButton(
```

ภาพประกอบ 4.230 เพิ่มโค้ด textField() สำหรับรับค่า price

ที่มา : ฌปภัช วรณตรง (2564 : 32)

```

53     TextField(
54         controller: quantity,
55         decoration: InputDecoration(
56             labelText: "จำนวนแอปเปิ้ล",border: OutlineInputBorder())
57         ), // TextField
58     SizedBox(
59         height: 20.0,
60     ),TextField( // SizedBox
61         controller: quantity,
62         decoration: InputDecoration(
63             labelText: "จำนวนแอปเปิ้ล",border: OutlineInputBorder())
64         ), // TextField
65     SizedBox(
66         height: 20.0,
67     ), // SizedBox
68     ElevatedButton(
69         onPressed: () {

```

ภาพประกอบ 4.231 เพิ่มโค้ด textField() สำหรับปรับค่า price

ที่มา : ฅนปักษ์ วรณตรง (2564 : 32)

#### 2.14.2 ให้ทำการเปลี่ยนข้อความและตัวแปรภายใน TextField(...)

```

53     TextField(
54         controller: quantity,
55         decoration: InputDecoration(
56             labelText: "ราคาแอปเปิ้ล",border: OutlineInputBorder()) /
57         ), // TextField
58     SizedBox(
59         height: 20.0,
60     ),TextField( // SizedBox
61         controller: quantity,
62         decoration: InputDecoration(
63             labelText: "จำนวนแอปเปิ้ล",border: OutlineInputBorder())
64         ), // TextField
65     SizedBox(
66         height: 20.0,
67     ), // SizedBox

```

ภาพประกอบ 4.232 เพิ่มโค้ด textField() สำหรับปรับค่า price (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 32)

```

53     TextField(
54       controller: price,
55       decoration: InputDecoration(
56         labelText: "ราคาแอปเปิ้ล",border: OutlineInputBorder()) //
57     ), // TextField
58     SizedBox(
59       height: 20.0,
60     ),TextField( // SizedBox
61       controller: quantity,
62       decoration: InputDecoration(
63         labelText: "จำนวนแอปเปิ้ล",border: OutlineInputBorder()) //
64     ), // TextField
65     SizedBox(
66       height: 20.0,
67     ), // SizedBox

```

ภาพประกอบ 4.233 เพิ่มโค้ด textField() สำหรับรับค่า price (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 32)

2.14.3 จะได้ผลลัพธ์การเพิ่มโค้ด textField() สำหรับรับค่า price ทั้งหมด ดังภาพ

```

Text(
  "โปรแกรมคำนวณ",
  style: TextStyle(fontSize: 30),
), // Text
TextField(
  controller: price,
  decoration: InputDecoration(
    labelText: 'ราคาแอปเปิ้ล',
    border: OutlineInputBorder()), // InputDecoration // TextField
), // TextField
SizedBox(
  height: 20.0,
), // SizedBox
TextField(
  controller: quantity,
  decoration: InputDecoration(
    labelText: 'จำนวนแอปเปิ้ล',
    border: OutlineInputBorder()), // InputDecoration // TextField
), // TextField

```

ภาพประกอบ 4.234 เพิ่มโค้ด textField() สำหรับรับค่า price (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 32)

## 2.15 เมื่อเพิ่มโค้ดเสร็จสิ้นแล้วให้สั่งการทำงาน ซึ่งจะได้ผลลัพธ์ตามภาพ



ภาพประกอบ 4.235 ผลลัพธ์ของแอปพลิเคชันสำหรับคำนวณ

ที่มา : ฅนปักษ์ วรณตรง (2564 : 33)

### บทสรุป

สำหรับเนื้อหาที่กล่าวมาทั้งหมดในบทนี้ จะพูดถึงการเขียนคำสั่งเพื่อลดขนาดและลบไฟล์ที่ไม่จำเป็นก่อนส่งไฟล์ให้ผู้อื่น การกำหนด layout แบบคอลัมน์ในแอปพลิเคชัน การสร้างแอปบาร์ (AppBar) และกำหนดค่าคุณสมบัติแอปบาร์ ขั้นตอนการใส่รูปภาพในแอปพลิเคชัน การสร้าง TextField กำหนดค่าคุณสมบัติใน TextField การตกแต่ง TextField การใส่ปุ่ม กำหนดค่าคุณสมบัติในปุ่ม การตกแต่งปุ่มด้วยการกำหนดสีปกติ และกำหนดด้วยการใช้ colorpicker การใส่ SizedBox กำหนดคุณสมบัติ SizedBox การกำหนดขอบในแอปพลิเคชันด้วย Padding การทำให้หน้าจอแอปพลิเคชันสามารถ scroll ขึ้นลงด้วยการใช้ ListView การสร้างฟังก์ชันคำนวณในปุ่ม และเรียนรู้การกำหนดค่าและการใช้ฟังก์ชัน initState() และ setState()

### เอกสารอ้างอิง

- จีราวุธ วารินทร์. (2564). พัฒนาโมบายล์แอปด้วย Flutter + Dart. กรุงเทพฯ : สำนักพิมพ์ชิมพลีฟาย
- อนุชิต ชโลธร. (2565). สูตรลับ Flutter. กรุงเทพฯ : สำนักพิมพ์ก๊อปปวาง.
- เอกรินทร์ วทัญญูเลิศสกุล. (2563). พัฒนา Mobile App ด้วย Flutter & Dart. กรุงเทพฯ :  
โปรวิชั่น.

## บทที่ 5

### การสร้างหน้าแอปพลิเคชันและแท็บบาร์สำหรับผู้ใช้งาน

จากบทที่แล้วผู้เรียนได้เรียนรู้การสร้างแอปพลิเคชันที่สามารถคำนวณค่าได้ ซึ่งเป็นแอปพลิเคชันที่มีลักษณะใช้งานเพียงหน้าเดียว ในบทนี้ผู้เรียนจะได้เรียนรู้การสร้างแอปพลิเคชันมากกว่า 1 หน้า โดยจะเรียนรู้วิธีการสร้างหน้าแอปพลิเคชันแยกแต่ละหน้า จากนั้นเมื่อได้แอปพลิเคชันแต่ละหน้าแล้วผู้เรียนจะได้เรียนรู้วิธีการสร้างแท็บบาร์ที่ด้านล่างของแอปพลิเคชัน ซึ่งเป็นส่วนเมื่อคลิกที่แท็บบาร์แล้วจะไปยังหน้าจอที่เตรียมไว้ในขั้นตอนแรก เมื่อได้แอปพลิเคชันที่พร้อมใช้งานเรียบร้อยแล้วผู้เรียนจะได้เรียนรู้การสร้างไอคอนของแอปพลิเคชัน เพื่อเมื่อนำไปติดตั้งบนอุปกรณ์เคลื่อนที่จะแสดงเป็นไอคอนให้ผู้ใช้คลิกบนหน้าจอเพื่อเข้าสู่หน้าจอแอปพลิเคชัน และในเนื้อหาสุดท้ายในการเรียนบทนี้หลังจากได้สร้างไอคอนแอปพลิเคชันแล้ว จะทำการสร้างไฟล์แอปพลิเคชันเป็นไฟล์สำหรับติดตั้งและใช้งานบนระบบปฏิบัติการแอนดรอยด์ โดยจะเป็นไฟล์นามสกุล .apk ที่สามารถนำไปติดตั้งบนอุปกรณ์เคลื่อนที่ ให้สามารถเรียกใช้งานแอปพลิเคชันได้ ดังรายละเอียดต่อไปนี้

#### การสร้างหน้าแอปพลิเคชันแยกแต่ละหน้า

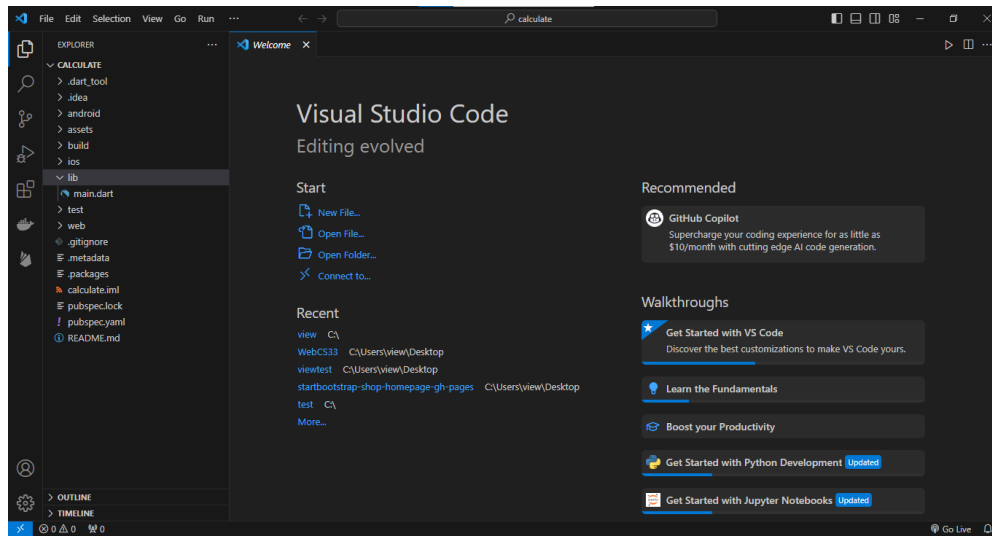
ในส่วนนี้จะเป็นการสร้างไฟล์แต่ละหน้าของแอปพลิเคชัน มีรายละเอียดขั้นตอนการทำดังนี้

1. ให้ทำการเปิดโปรแกรม VSCode ขึ้นมา จากนั้นให้ทำการเปิดไฟล์ Project ที่ได้

ทำการพัฒนาไว้ จากนั้นให้คลิกขวาที่ Folder lib เลือก New Folder ตั้งชื่อ Folder ว่า pages

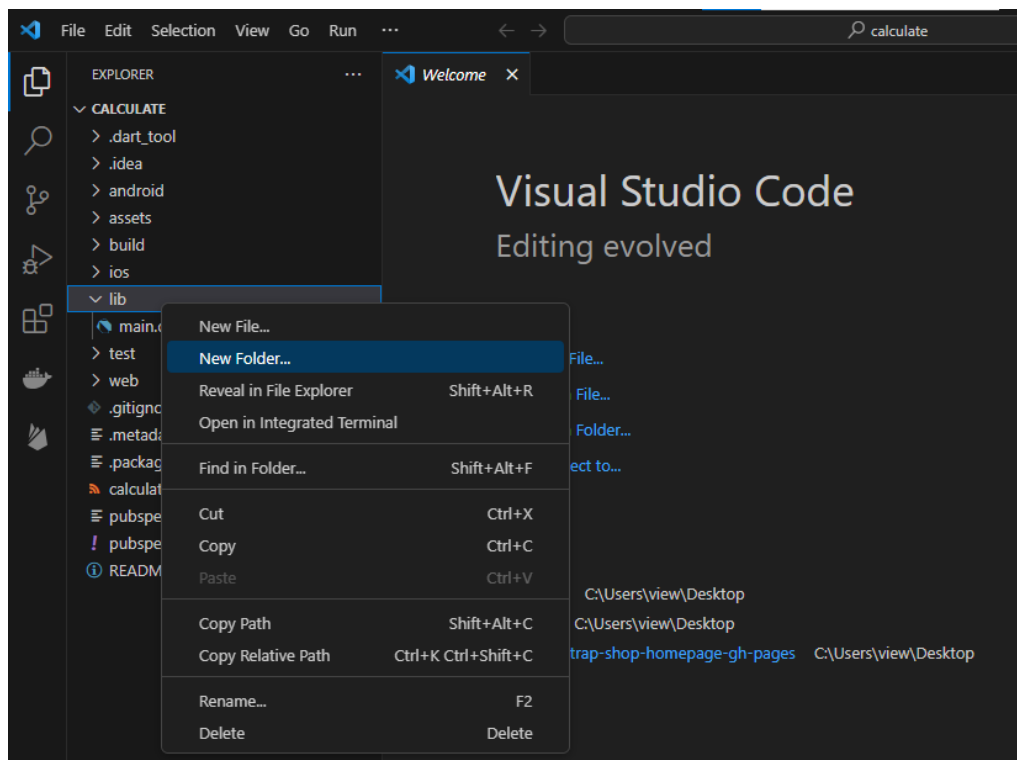
เมื่อเปิดไฟล์ Project แล้ว ให้ไปที่ Folder lib จากนั้นให้ทำการสร้าง Folder ขึ้นมาใหม่โดยตั้งชื่อว่า pages





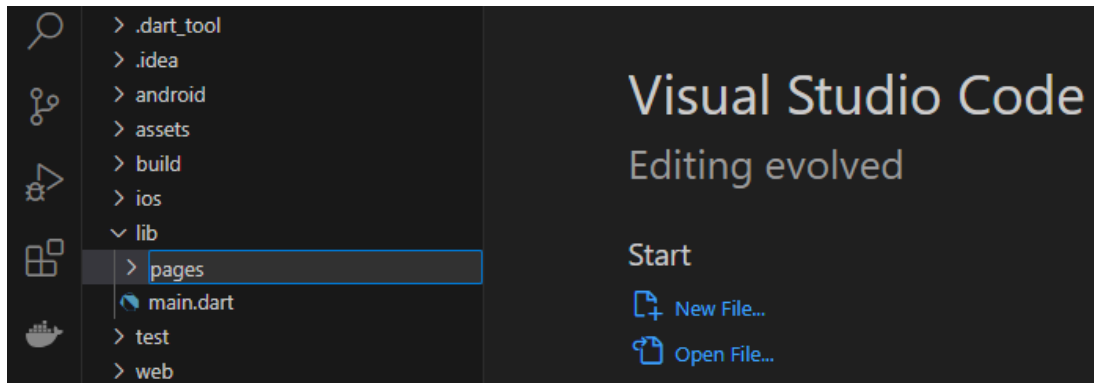
ภาพประกอบ 5.1 สร้าง Folder pages

ที่มา : ฅนปักษ์ วรณตรง (2564 : 4)



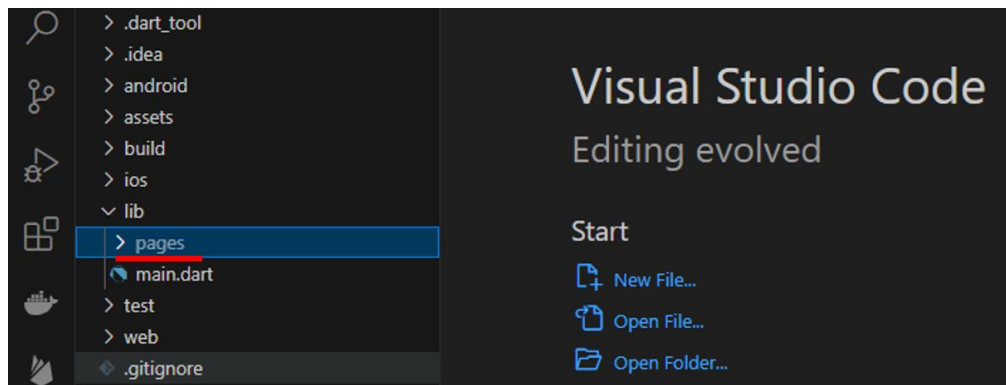
ภาพประกอบ 5.2 สร้าง Folder pages (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 4)



ภาพประกอบ 5.3 สร้าง Folder pages (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 4)

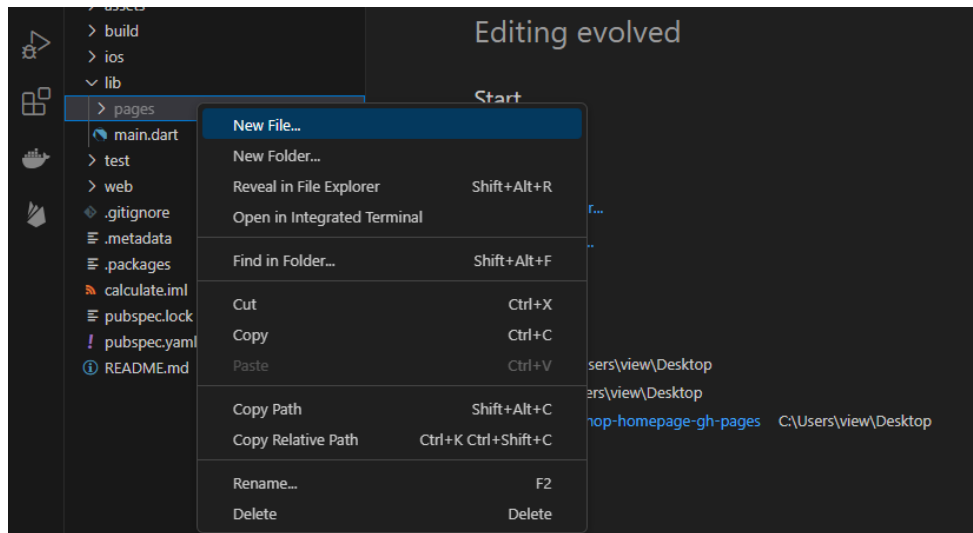


ภาพประกอบ 5.4 สร้าง Folder pages (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 4)

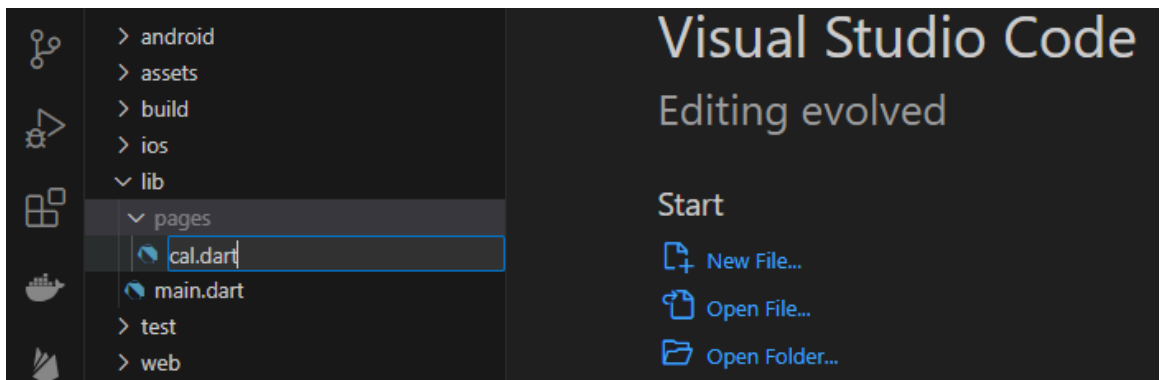
2. หลังจากทำการสร้าง Folder pages เสร็จสิ้นแล้ว ให้ทำการสร้างไฟล์ 3 ไฟล์ขึ้นมา ได้แก่ cal.dart, contact.dart และ home.dart

กดเลือกที่ Folder pages จากนั้นให้ทำการสร้างไฟล์งานขึ้นมาใหม่ 3 ไฟล์



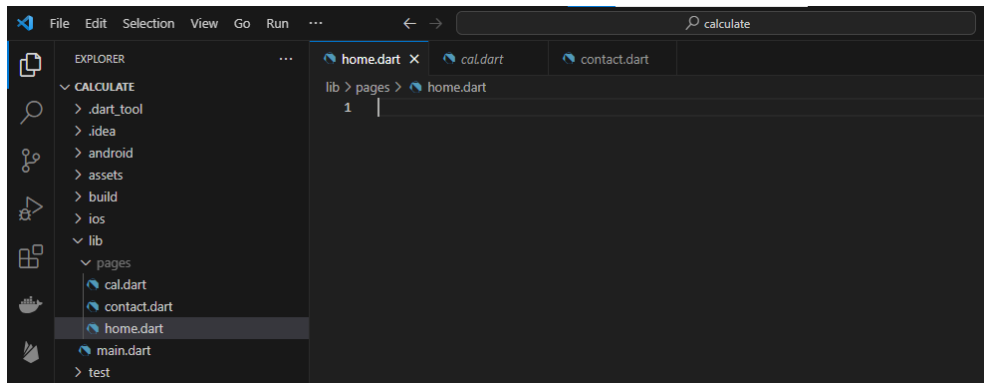
ภาพประกอบ 5.5 สร้างไฟล์งานใน Folder pages

ที่มา : ฌปภัช วรรณตรง (2564 : 5)



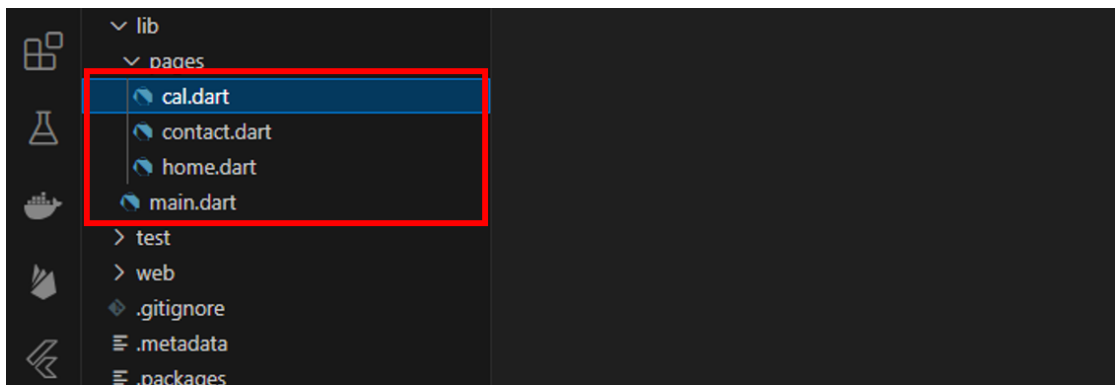
ภาพประกอบ 5.6 สร้างไฟล์งานใน Folder pages (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 5)



ภาพประกอบ 5.7 สร้างไฟล์งานใน Folder pages (ต่อ)

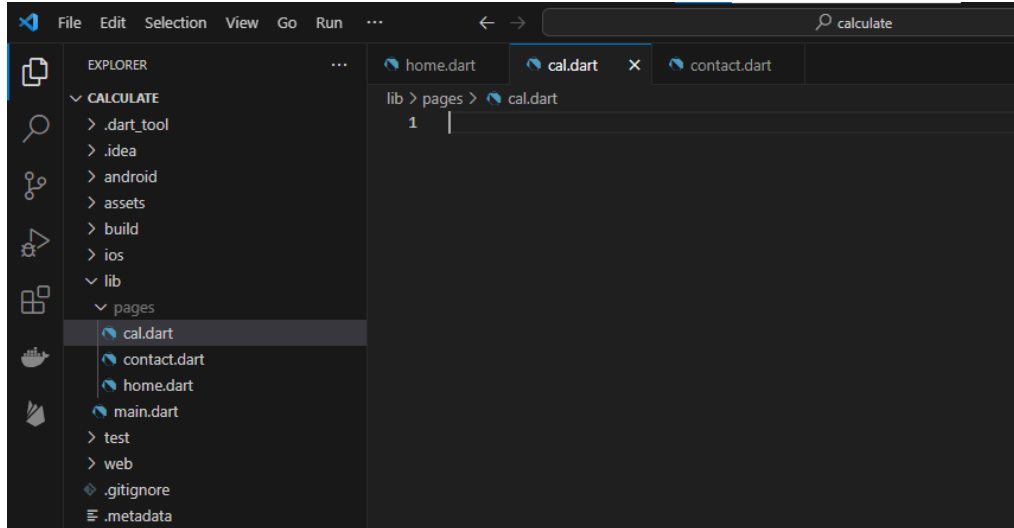
ที่มา : ฅนปักษ์ วรณตรง (2564 : 5)



ภาพประกอบ 5.8 สร้างไฟล์งานใน Folder pages (ต่อ)

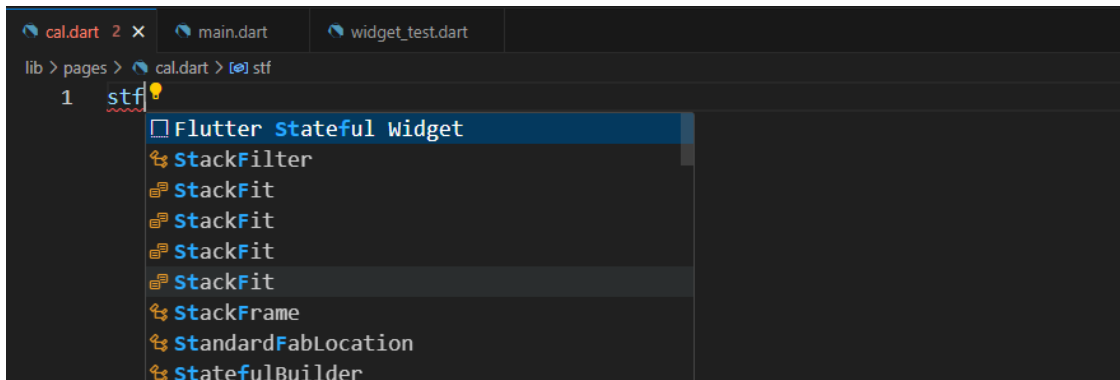
ที่มา : ฅนปักษ์ วรณตรง (2564 : 5)

3. เมื่อทำการสร้างไฟล์แล้ว ให้เปิดไฟล์ cal.dart จากนั้นให้ทำการสร้าง class StatefulWidget{} โดยตั้งชื่อ class ว่า CalculatePage ตามภาพ



ภาพประกอบ 5.9 สร้าง class StatefulWidget

ที่มา : ฌปภัช วรณตรง (2564 : 6)



ภาพประกอบ 5.10 สร้าง class StatefulWidget (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 6)

```

home.dart  cal.dart 9+ x  contact.dart
lib > pages > cal.dart > <unnamed>
1 class extends StatefulWidget {
2   const ({Key? key}) : super(key: key);
3
4   @override
5   State<> createState() => _State();
6 }
7
8 class _State extends State<> {
9   @override
10  Widget build(BuildContext context) {
11    return Container(
12
13    );
14  }
15 }

```

ภาพประกอบ 5.11 สร้าง class StatefulWidget (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 6)

จากนั้นเมื่อได้ class StatefulWidget เรียบร้อยแล้ว ให้ทำการตั้งชื่อ class เป็น CalculatePage

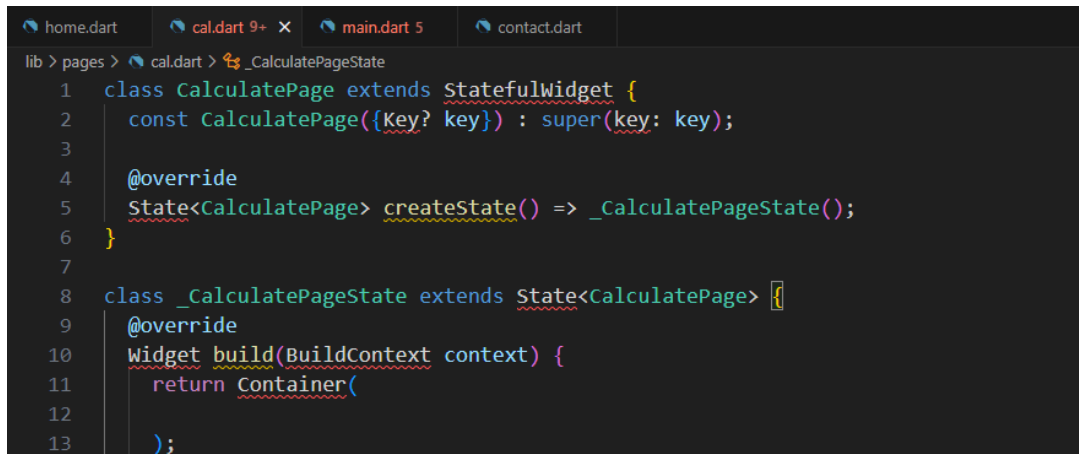
```

home.dart  cal.dart 9+ x  main.dart 5  contact.dart
lib > pages > cal.dart > CalculatePage
1 class CalculatePage extends StatefulWidget {
2   const CalculatePage({Key? key}) : super(key: key);
3
4   @override
5   State<CalculatePage> createState() => _CalculatePageState();
6 }
7
8 class _CalculatePageState extends State<CalculatePage> {
9   @override
10  Widget build(BuildContext context) {
11    return Container(
12
13    );

```

ภาพประกอบ 5.12 สร้าง class StatefulWidget (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 6)



```

lib > pages > cal.dart > _CalculatePageState
1 class CalculatePage extends StatefulWidget {
2   const CalculatePage({Key? key}) : super(key: key);
3
4   @override
5   State<CalculatePage> createState() => _CalculatePageState();
6 }
7
8 class _CalculatePageState extends State<CalculatePage> {
9   @override
10  Widget build(BuildContext context) {
11    return Container(
12
13  );

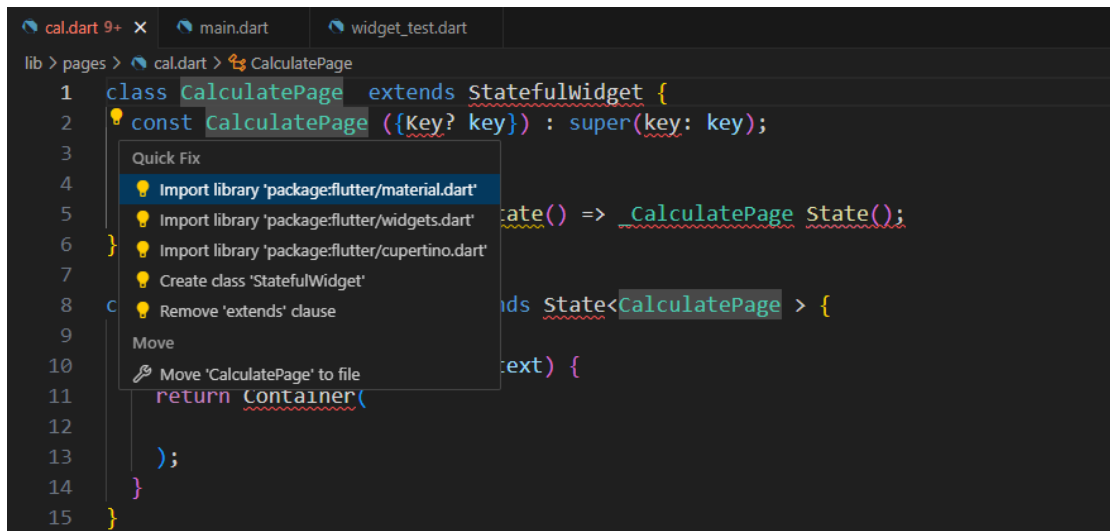
```

ภาพประกอบ 5.13 สร้าง class StatefulWidget (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 6)

4. หากพบ Error ที่ class CalculatePage ให้ทำการเพิ่มโค้ดไว้ส่วนบนของไฟล์งานเป็น import Library 'package:flutter/material.dart';

ให้ทำการคลิกโค้ด CalculatePage จากนั้นจะปรากฏหลอดไฟสีเหลือง คลิกที่หลอดไฟสีเหลืองจะมีหน้าต่างชุดคำสั่งขึ้นมา จากนั้นให้ทำการเลือกที่ import Library 'package:flutter/material.dart';



```

cal.dart 9+ x main.dart widget_test.dart
lib > pages > cal.dart > CalculatePage
1 class CalculatePage extends StatefulWidget {
2   const CalculatePage ({Key? key}) : super(key: key);
3
4   Quick Fix
5   Import library 'package:flutter/material.dart'
6   Import library 'package:flutter/widgets.dart'
7   Import library 'package:flutter/cupertino.dart'
8   Create class 'StatefulWidget'
9   Remove 'extends' clause
10  Move
11  Move 'CalculatePage' to file
12  return Container(
13
14  );
15 }

```

ภาพประกอบ 5.14 แก้ไขโค้ดที่ Error โดยการ import package

ที่มา : ฌปภัช วรรณตรง (2564 : 7)

```
lib > pages > cal.dart > ...
1  import 'package:flutter/material.dart';
2
3  class CalculatePage extends StatefulWidget {
4    const CalculatePage({ Key? key }) : super(key: key);
5
6    @override
7    _CalculatePageState createState() => _CalculatePageState();
8  }
9
10 class _CalculatePageState extends State<CalculatePage> {
```

ภาพประกอบ 5.15 แก้ไขโค้ดที่ Error โดยการ import package (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 7)

5. ในส่วนของ class ที่ได้ทำการเพิ่มขึ้นมาจะมีส่วนที่ไม่ได้ใช้งาน ให้ทำการ Comment ส่วน `const CalculatePage({ Key? key }) : super(key: key);` ไว้ โดยการพิมพ์ `//` ไว้ข้างหน้าโค้ด หรือจะใช้การคลิกที่ส่วนของโค้ดที่ต้องการ Comment และกด `Ctrl + /` ที่เป็นพิมพ์

```
home.dart x cal.dart x main.dart 5 contact.dart
lib > pages > cal.dart > CalculatePage > CalculatePage
1  import 'package:flutter/material.dart';
2
3  class CalculatePage extends StatefulWidget {
4    const CalculatePage({ Key? key }) : super(key: key);
5
6    @override
7    _CalculatePageState createState() => _CalculatePageState();
8  }
9
10 class _CalculatePageState extends State<CalculatePage> {
```

ภาพประกอบ 5.16 Comment ส่วนที่ไม่ต้องการใช้งาน

ที่มา : ฅนปักษ์ วรณตรง (2564 : 8)



```

lib > pages > cal.dart > CalculatePage
1  import 'package:flutter/material.dart';
2
3  class CalculatePage extends StatefulWidget {
4    //const CalculatePage({Key? key}) : super(key: key);
5
6    @override
7    _CalculatePageState createState() => _CalculatePageState();
8  }
9
10 class _CalculatePageState extends State<CalculatePage> {
11

```

ภาพประกอบ 5.17 Comment ส่วนที่ไม่ต้องการใช้งาน (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 8)

6. เปิดไฟล์ main.dart จากนั้นให้ไปตัด (Cut) โค้ดส่วนของ ListView(); ทั้งหมด

```

lib > main.dart > _HomeState > build
61
62 Widget build(BuildContext context) {
63   return ListView(
64     children: [
65       Padding(
66         padding: const EdgeInsets.all(100.0),
67         child: Center(
68           child: Column(
69             children: [
70               Image.asset("assets/cat11.png", width: 400),
71               Text("โปรแกรมคำนวณ", style: TextStyle(fontSize: 70, fontWeight: Fo

```

ภาพประกอบ 5.18 ตัด (Cut) โค้ดส่วนของ ListView();

ที่มา : ฌปภัช วรรณตรง (2564 : 9)

```

61
62 Widget build(BuildContext context) {
63   return ListView(
64     children: [
65       Padding(
66         padding: const EdgeInsets.all(100.0),
67         child: Center(
68           child: Column(
69             children: [
70               Image.asset("assets/cat11.png", width: 400),
71               Text("โปรแกรมคำนวณ", style: TextStyle(fontSize: 70, fontWeight: Font
72             ),
73             ),
74             TextField(
75               controller: rates,
76               decoration: InputDecoration(
77                 labelText: "ราคา", border: OutlineInputBorder()), // InputDeco
78             ),
79             TextField(
80               controller: number,
81             ),
82           ],
83         ),
84       ),
85     ],
86   );
87 }

```

ภาพประกอบ 5.19 ตัด (Cut) โค้ดส่วนของ ListView(); (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 9)

```

87 setState(() {
88   result.text =
89     "ชื่อแอปเปิ้ล ${number.text} ตัว ราคาถูกละ ${rates.text} บาท รวมค้
90 });
91   }, child: Text("คำนวณ"),
92   style: ButtonStyle(
93     backgroundColor: MaterialStateProperty.all(Color(0xff4a2424)),
94     padding: MaterialStateProperty.all(EdgeInsets.fromLTRB(25, 20, 25, 2
95     textStyle: MaterialStateProperty.all(TextStyle(fontSize: 18))), //
96     Text(result.text, style: TextStyle(fontSize: 18),)
97   ],
98   ),
99   // Column
100  ), // Center
101  ), // Padding
102  ],
103 ); // ListView
104

```

ภาพประกอบ 5.20 ตัด (Cut) โค้ดส่วนของ ListView(); (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 9)

```

lib > main.dart > _HomeState > build
55 @override
56 void initState() {
57     // TODO: implement initState
58     super.initState();
59     result.text = "ชื่อแอปเบิ้ลจำนวน X ลูก ราคาลูกละ X บาท รวมลูกค้ต้องจ่ายเงิน X บาท";
60 }
61
62 Widget build(BuildContext context) {}
63     return
64
65 }

```

ภาพประกอบ 5.21 ตัด (Cut) โค้ดส่วนของ ListView(); (ต่อ)

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 9)

7. เปิดไฟล์ cal.dart ในส่วนของ return ให้ทำการเปลี่ยนโค้ด Container(); เป็น ListView();  
ที่ได้ทำการตัดมาจากไฟล์ main.dart

เปลี่ยน Container(...) เป็น ListView(...) ที่ได้ทำการตัดมา

```

lib > pages > cal.dart > _CalculatePageState > build
1  import 'package:flutter/material.dart';
2
3  class CalculatePage extends StatefulWidget {
4      //const CalculatePage({Key? key}) : super(key: key);
5
6      @override
7      State<CalculatePage> createState() => _CalculatePageState();
8  }
9
10 class _CalculatePageState extends State<CalculatePage> {
11     @override
12     Widget build(BuildContext context) {
13         return Container(
14             ..
15         );
16     }

```

ภาพประกอบ 5.22 นำโค้ดที่ตัดมาใส่แทนโค้ดส่วน Container();

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 10)

```

home.dart  cal.dart 9 x  main.dart 5  contact.dart
lib > pages > cal.dart > _CalculatePageState > build
9
10 class _CalculatePageState extends State<CalculatePage> {
11   @override
12   Widget build(BuildContext context) {
13     return ListView(
14       children: [
15         Padding(
16           padding: const EdgeInsets.all(50.0),
17           child: Center(
18             child: Column(
19               children: [
20                 Image.asset('assets/apple.png',width: 300,),
21                 Text("โปรแกรมคำนวณ",style: TextStyle(fontSize: 30),),

```

ภาพประกอบ 5.23 นำโค้ดที่ตัดมาใส่แทนโค้ดส่วน Container(); (ต่อ)  
 ที่มา : ฅนปลัซ วรณตรง (2564 : 10)

```

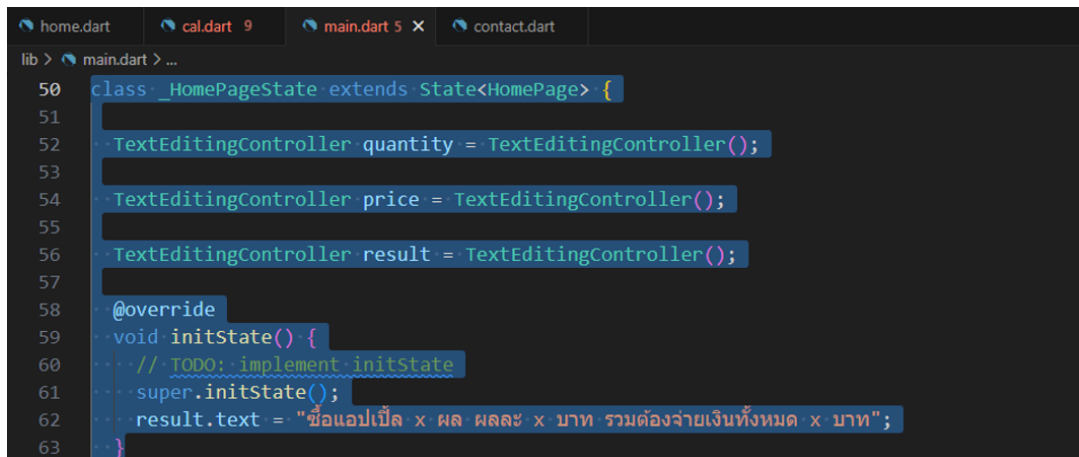
13   @override
14   Widget build(BuildContext context) {
15     return ListView(
16       children: [
17         Padding(
18           padding: const EdgeInsets.all(50.0),
19           child: Center(
20             child: Column(
21               children: [
22                 Image.asset('assets/apple.png',width: 300,),
23                 Text("โปรแกรมคำนวณ",style: TextStyle(fontSize: 30),),
24                 TextField(
25                   controller: price,
26                   decoration: InputDecoration(
27                     labelText: 'ราคาแอปเปิ้ล', border: OutlineInputBorder()), //
28                 SizedBox(
29                   height: 20.0,
30                 ), // SizedBox

```

ภาพประกอบ 5.24 นำโค้ดที่ตัดมาใส่แทนโค้ดส่วน Container(); (ต่อ)  
 ที่มา : ฅนปลัซ วรณตรง (2564 : 10)

8. จะเห็นว่าโค้ดยังมี Error ให้ทำการกลับไปไฟล์ main.dart จากนั้นทำการตัดโค้ดส่วนที่เหลือเข้ามาไว้ยังไฟล์ cal.dart

ภายใน class `_HomeState` ให้ทำการคัดลอกหรือตัดโค้ดที่ประกาศตัวแปรมาใส่ในไฟล์ cal.dart



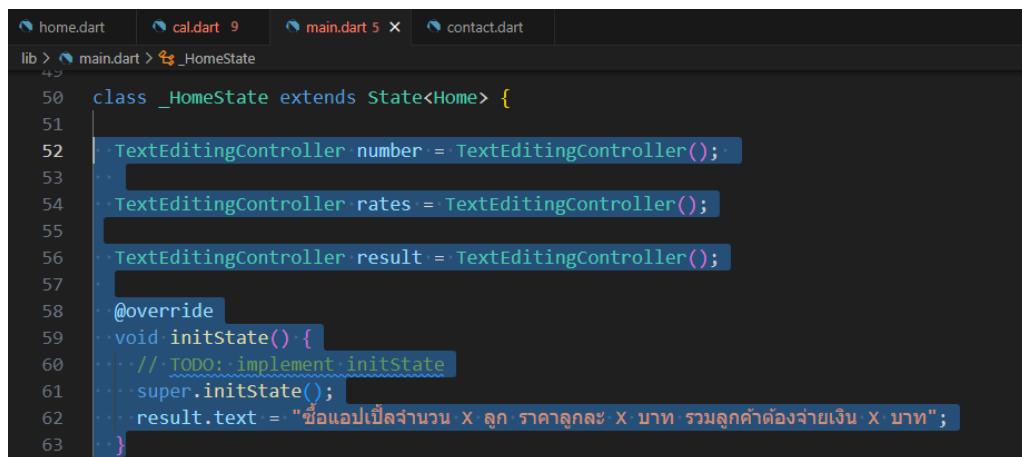
```

50 class _HomePageState extends State<HomePage> {
51
52   TextEditingController quantity = TextEditingController();
53
54   TextEditingController price = TextEditingController();
55
56   TextEditingController result = TextEditingController();
57
58   @override
59   void initState() {
60     // TODO: implement initState
61     super.initState();
62     result.text = "ชื่อแอปเบิ้ล x ผล ผลละ x บาท รวมต้องจ่ายเงินทั้งหมด x บาท";
63   }

```

ภาพประกอบ 5.25 ตัดโค้ดส่วนที่เหลือจากไฟล์ main.dart

ที่มา : ฌปภัช วรรณตรง (2564 : 11)



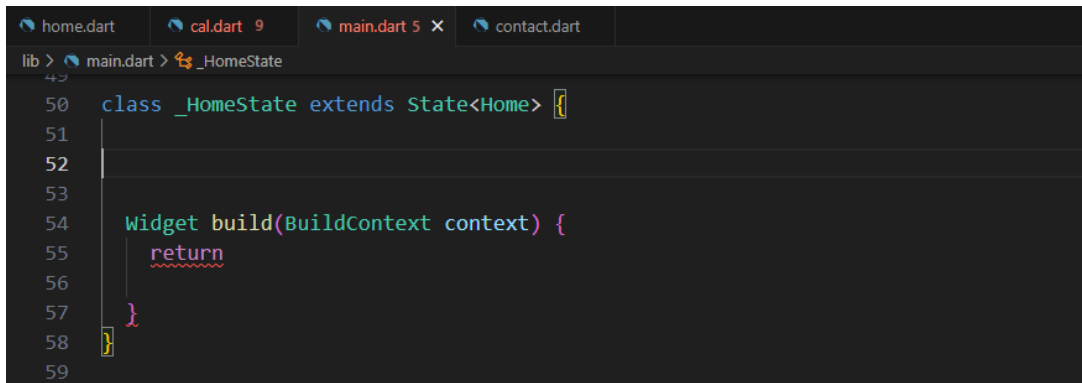
```

50 class _HomeState extends State<Home> {
51
52   TextEditingController number = TextEditingController();
53
54   TextEditingController rates = TextEditingController();
55
56   TextEditingController result = TextEditingController();
57
58   @override
59   void initState() {
60     // TODO: implement initState
61     super.initState();
62     result.text = "ชื่อแอปเบิ้ลจำนวน X ลูก ราคาลูกละ X บาท รวมลูกต้องจ่ายเงิน X บาท";
63   }

```

ภาพประกอบ 5.26 ตัดโค้ดส่วนที่เหลือจากไฟล์ main.dart (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 11)



```

home.dart  cal.dart 9  main.dart 5 x  contact.dart
lib > main.dart > _HomeState
49
50 class _HomeState extends State<Home> {}
51
52
53
54 widget build(BuildContext context) {
55   return
56
57 }
58
59

```

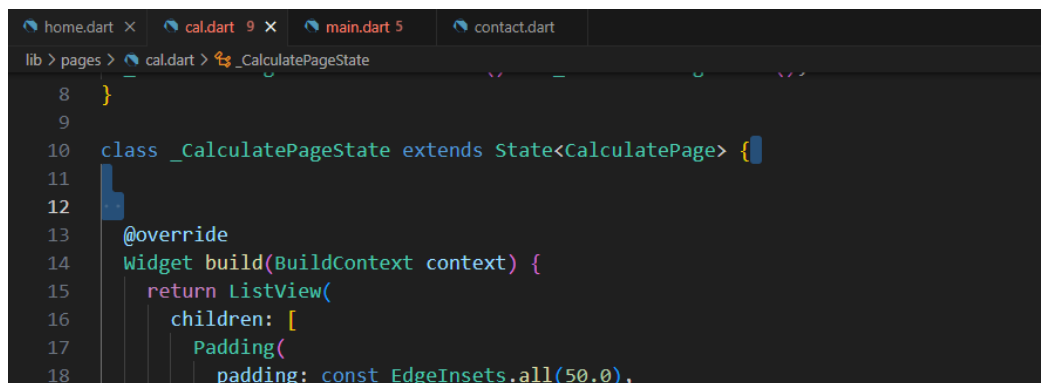
ภาพประกอบ 5.27 ตัดโค้ดส่วนที่เหลือจากไฟล์ main.dart (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 11)

9. นำโค้ดส่วนที่เหลือที่ตัดมาจากไฟล์ main.dart มาใส่ยัง {} ในส่วนของ class

State<CalculatePage>{} ตามภาพ

ไปที่ class \_CalculatePageState และนำโค้ดที่ได้ทำการคัดลอกมาหรือตัดมาไปวางไว้



```

home.dart x  cal.dart 9 x  main.dart 5  contact.dart
lib > pages > cal.dart > _CalculatePageState
8   }
9
10  class _CalculatePageState extends State<CalculatePage> {}
11
12
13  @override
14  widget build(BuildContext context) {
15    return ListView(
16      children: [
17        Padding(
18          padding: const EdgeInsets.all(50.0),

```

ภาพประกอบ 5.28 นำโค้ดส่วนที่เหลือมาใส่ class State<CalculatePage>{}

ที่มา : ฌปภัช วรณตรง (2564 : 12)

```

lib > pages > cal.dart > _CalculatePageState
10 class _CalculatePageState extends State<CalculatePage> {
11
12   TextEditingController quantity = TextEditingController();
13
14   TextEditingController price = TextEditingController();
15
16   TextEditingController result = TextEditingController();
17
18   @override
19   void initState() {
20     // TODO: implement initState
21     super.initState();
22     result.text = "ซื้อแอปเปิ้ล x ผล ผลละ x บาท รวมต้องจ่ายเงินทั้งหมด x บาท";
23   }
24   @override

```

ภาพประกอบ 5.29 นำโค้ดส่วนที่เหลือมาใส่ class State<CalculatePage>{} (ต่อ)  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 12)

```

lib > pages > cal.dart > _CalculatePageState
8 }
9
10 class _CalculatePageState extends State<CalculatePage> {
11
12   TextEditingController quantity = TextEditingController();
13
14   TextEditingController price = TextEditingController();
15
16   TextEditingController result = TextEditingController();
17
18   @override
19   void initState() {
20     // TODO: implement initState
21     super.initState();
22     result.text = "ซื้อแอปเปิ้ล x ผล ผลละ x บาท รวมต้องจ่ายเงินทั้งหมด x บาท";
23   }
24   @override
25   Widget build(BuildContext context) {
26     return ListView(

```

ภาพประกอบ 5.30 นำโค้ดส่วนที่เหลือมาใส่ class State<CalculatePage>{} (ต่อ)  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 12)

10. ไฟล์ main.dart ให้ทำการลบโค้ดส่วนของ class Home ออกทั้งหมด

```

lib > main.dart > Home
34 class Home extends StatefulWidget {
35   //const Home({super.key});
36
37   @override
38   State<Home> createState() => _HomeState();
39 }
40
41 class _HomeState extends State<Home> {
42   ...
43   Widget build(BuildContext context) {
44     return
45     ...
46   }
47 }

```

ภาพประกอบ 5.31 ลบโค้ดส่วน class Home

ที่มา : ฌปภัช วรรณตรง (2564 : 13)

```

lib > main.dart > Home
34 class Home extends StatefulWidget {
35   //const Home({super.key});
36
37   @override
38   State<Home> createState() => _HomeState();
39 }
40
41 class _HomeState extends State<Home> {
42   ...
43   Widget build(BuildContext context) {
44     return
45     ...
46   }
47 }

```

ภาพประกอบ 5.32 ลบโค้ดส่วน class Home (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 13)



```

lib > main.dart > ...
23 @override
24 Widget build(BuildContext context) {
25   return MaterialApp(
26     home: Scaffold (
27       appBar: AppBar(title: Text("แอปคำนวณ"), backgroundColor: Color.fromARGB(255,
28       body: Home(),
29     ), // Scaffold
30   ); // MaterialApp
31 }
32 }
33 |
34

```

ภาพประกอบ 5.33 ลบโค้ดส่วน class Home (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 13)

11. จากนั้นให้ไปลบโค้ด Home(), ที่ tag body ออก

```

lib > main.dart > MyApp > build
23 @override
24 Widget build(BuildContext context) {
25   return MaterialApp(
26     home: Scaffold (
27       appBar: AppBar(title: Text("แอปคำนวณ"), backgroundColor: Color.fromARGB(255,
28       body: Home(),
29     ), // Scaffold
30   ); // MaterialApp
31 }
32 }

```

ภาพประกอบ 5.34 ลบโค้ดส่วน Home(),

ที่มา : ฅนปลัซ วรณตรง (2564 : 14)

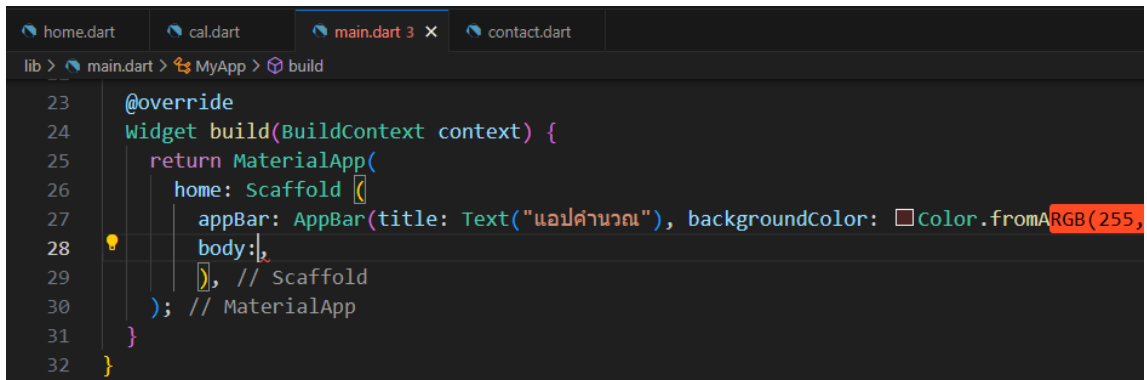
```

lib > main.dart > MyApp > build
23 @override
24 Widget build(BuildContext context) {
25   return MaterialApp(
26     home: Scaffold (
27       appBar: AppBar(title: Text("แอปคำนวณ"), backgroundColor: Color.fromARGB(
28       body: Home(),
29     ), // Scaffold
30   ); // MaterialApp
31 }

```

ภาพประกอบ 5.35 ลบโค้ดส่วน Home(), (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 14)



```

23  @override
24  Widget build(BuildContext context) {
25    return MaterialApp(
26      home: Scaffold (
27        appBar: AppBar(title: Text("แอปคำนวณ"), backgroundColor: Color.fromARGB(255,
28        body:
29      ), // Scaffold
30    ); // MaterialApp
31  }
32  }

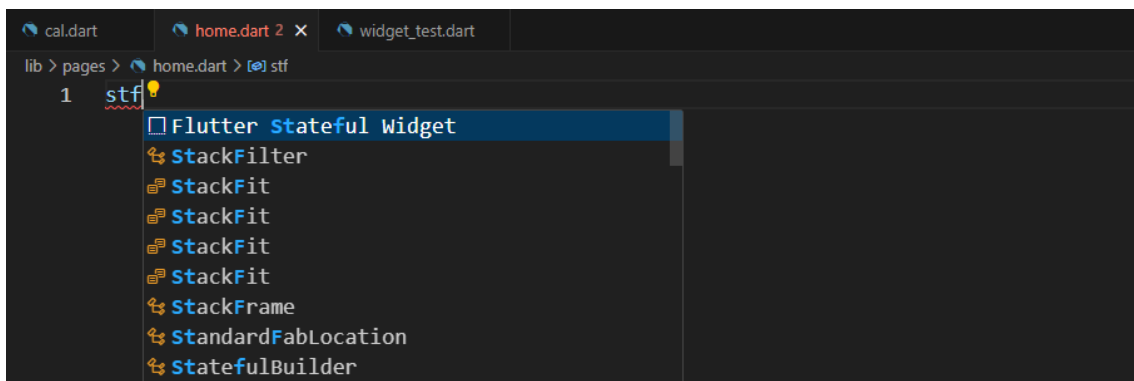
```

ภาพประกอบ 5.36 ลบโค้ดส่วน Home(), (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 14)

12. เปิดไฟล์ home.dart ทำการสร้าง class StatefulWidget{} ให้ตั้งชื่อ class เป็น HomePage จากนั้นให้ทำการ import package เข้ามา และ comment ส่วน const ใน class ไว้ตามภาพ

12.1 ให้ทำการเพิ่มโค้ด โดยใช้วิธีเดียวกันกับไฟล์ cal.dart แต่ให้ใช้ชื่อ class เป็น HomePage



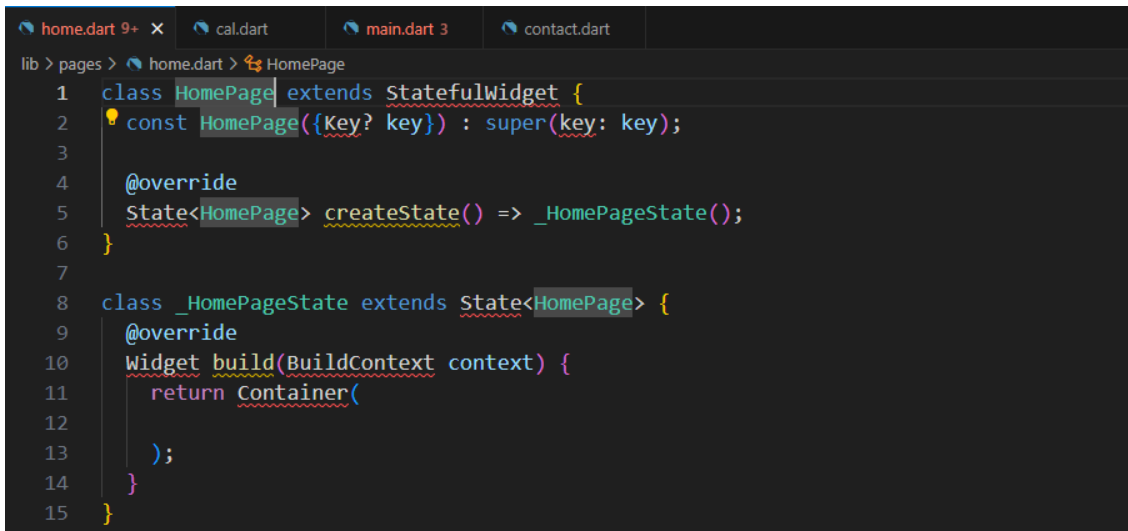
```

1  stf
   Flutter StatefulWidget
   StackFilter
   StackFit
   StackFit
   StackFit
   StackFit
   StackFrame
   StandardFabLocation
   StatefulWidget

```

ภาพประกอบ 5.37 สร้าง class HomePage

ที่มา : ฅนปลัซ วรณตรง (2564 : 15)



```

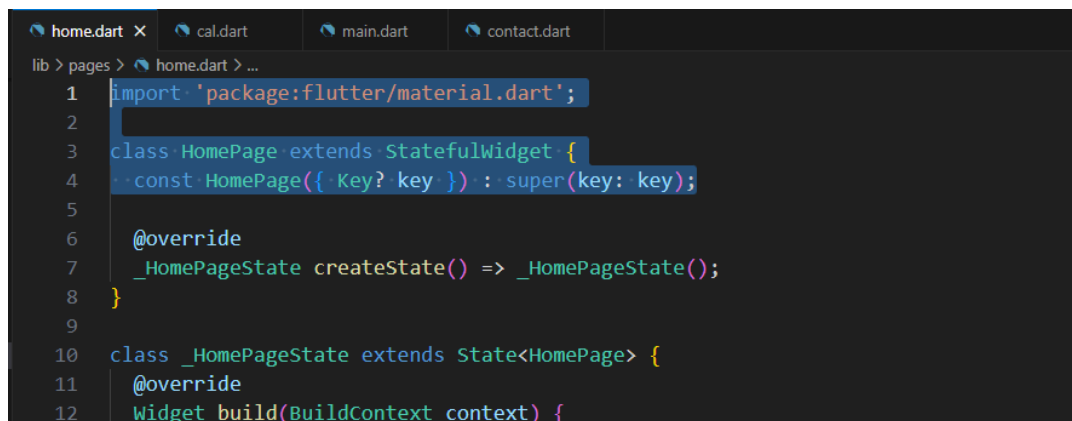
lib > pages > home.dart > HomePage
1 class HomePage extends StatefulWidget {
2   const HomePage({Key? key}) : super(key: key);
3
4   @override
5   State<HomePage> createState() => _HomePageState();
6 }
7
8 class _HomePageState extends State<HomePage> {
9   @override
10  Widget build(BuildContext context) {
11    return Container(
12
13    );
14  }
15 }

```

ภาพประกอบ 5.38 สร้าง class HomePage (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 15)

12.2 ให้ทำการ import package เข้ามาเพื่อแก้โค้ดที่ Error และปิดใช้งานโค้ด const โดยใช้วิธีเดียวกันกับไฟล์ cal.dart



```

lib > pages > home.dart > ...
1 import 'package:flutter/material.dart';
2
3 class HomePage extends StatefulWidget {
4   const HomePage({ Key? key }) : super(key: key);
5
6   @override
7   _HomePageState createState() => _HomePageState();
8 }
9
10 class _HomePageState extends State<HomePage> {
11   @override
12   Widget build(BuildContext context) {

```

ภาพประกอบ 5.39 สร้าง class HomePage (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 15)

13. แก้ไขโค้ดใน return Container() ให้เป็น ListView() และเพิ่มโค้ด  
13.1 ให้ทำการแก้ไขโค้ด Container() เป็น ListView()

```

lib > pages > home.dart > _HomePageState > build
1 import 'package:flutter/material.dart';
2
3 class HomePage extends StatefulWidget {
4   //const HomePage({Key? key}) : super(key: key);
5
6   @override
7   State<HomePage> createState() => _HomePageState();
8 }
9
10 class _HomePageState extends State<HomePage> {
11   @override
12   Widget build(BuildContext context) {
13     return Container(
14
15     );
16   }
17 }

```

ภาพประกอบ 5.40 แก้ไขส่วน return Container()

ที่มา : ฌปภัช วรณตรง (2564 : 16)

```

lib > pages > home.dart > _HomePageState > build
7   State<HomePage> createState() => _HomePageState();
8 }
9
10 class _HomePageState extends State<HomePage> {
11   @override
12   Widget build(BuildContext context) {
13     return ListView
14     [
15   ]

```

- ListView... ({Key? key, Axis scrollDirectio...
- ListView.builder(...)
- ListView.custom(...)
- ListView.separated(...)
- ListView
- ListViewWheelViewport(...)
- ListViewWheelViewport
- ListViewWheelScrollView(...)
- ListViewWheelScrollView.useDelegate(...)

ภาพประกอบ 5.41 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 16)

### 13.2 ภายใน ListView(...) ให้ทำการเพิ่มโค้ด children: []

```

lib > pages > home.dart > _HomePageState > build
7   State<HomePage> createState() => _HomePageState();
8   }
9
10  class _HomePageState extends State<HomePage> {
11    @override
12    Widget build(BuildContext context) {
13      return ListView(
14        children: []
15      );
16    }
17  }

```

ภาพประกอบ 5.42 แกะไขส่วน return Container() (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 16)

### 13.3 ภายใน [...] ให้ทำการเพิ่มโค้ด Center(...) เข้าไป

```

lib > pages > home.dart > _HomePageState > build
7   State<HomePage> createState() => _HomePageState();
8   }
9
10  class _HomePageState extends State<HomePage> {
11    @override
12    Widget build(BuildContext context) {
13      return ListView(
14        children: [
15          Center(
16            ),
17        ],
18      );
19    }
20  }

```

ภาพประกอบ 5.43 แกะไขส่วน return Container() (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 16)

13.4 ภายใน Center(...) ให้ทำการเพิ่ม child:

```

lib > pages > home.dart > _HomePageState > build
7   State<HomePage> createState() => _HomePageState();
8   }
9
10  class _HomePageState extends State<HomePage> {
11    @override
12    Widget build(BuildContext context) {
13      return ListView(
14        children: [
15          Center(
16            child:
17            Widget?
18          ),
19        ] // ListView
20      }
21    }

```

ภาพประกอบ 5.44 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 16)

13.5 ภายใน child: ให้ทำการเพิ่มโค้ดเพื่อแสดงข้อความเป็น Text(...)

```

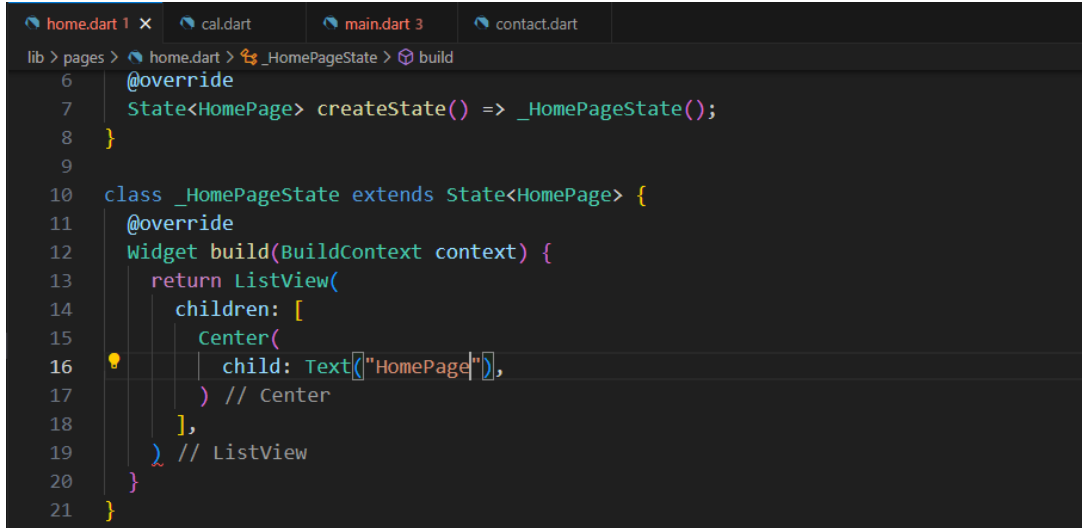
lib > pages > home.dart > _HomePageState > build
7   State<HomePage> createState() => _HomePageState();
8   }
9
10  class _HomePageState extends State<HomePage> {
11    @override
12    Widget build(BuildContext context) {
13      return ListView(
14        children: [
15          Center(
16            child: text,
17            // Center
18            Text
19            Text(...) (String data, {Key? key, TextS...}
20            textDirectionToAxisDirection(...)
21            TextButton
22            TextButtonTheme

```

ภาพประกอบ 5.45 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 16)

13.6 เมื่อทำการเพิ่มโค้ดเพื่อแสดงข้อความแล้ว ให้กำหนดข้อความที่จะแสดงเข้าไปภายในเครื่องหมาย “ ”



```

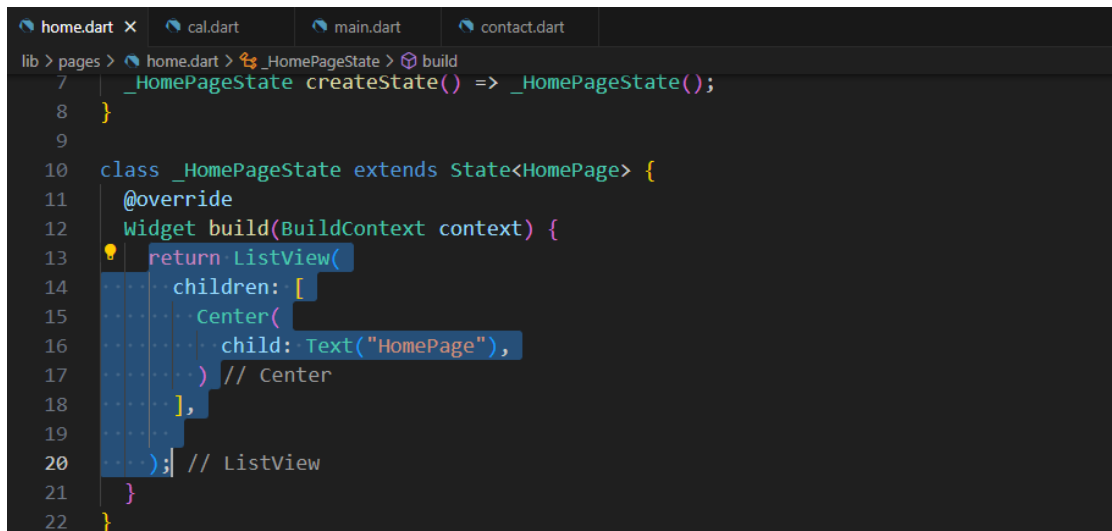
lib > pages > home.dart > _HomePageState > build
6  @override
7  State<HomePage> createState() => _HomePageState();
8  }
9
10 class _HomePageState extends State<HomePage> {
11  @override
12  Widget build(BuildContext context) {
13    return ListView(
14      children: [
15        Center(
16          child: Text("HomePage"),
17        ) // Center
18      ],
19    ) // ListView
20  }
21  }

```

ภาพประกอบ 5.46 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 16)

13.7 โค้ดทั้งหมดที่ได้แก้ไขดังภาพ



```

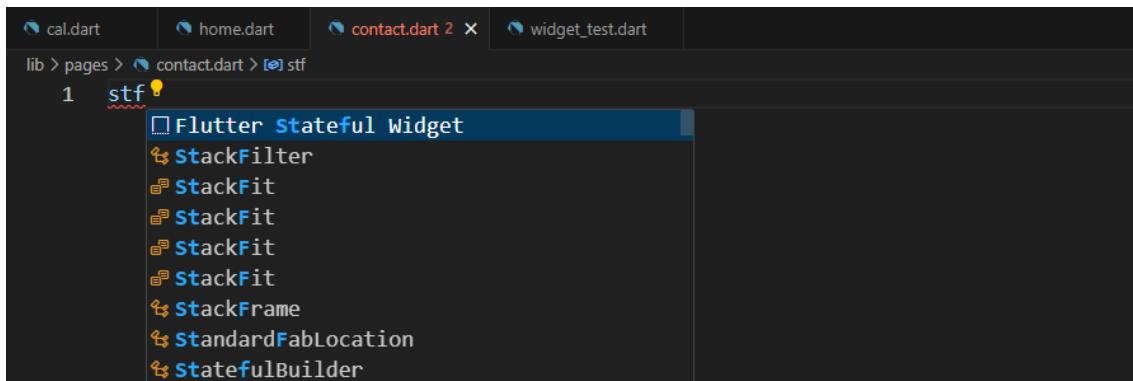
lib > pages > home.dart > _HomePageState > build
7  _HomePageState createState() => _HomePageState();
8  }
9
10 class _HomePageState extends State<HomePage> {
11  @override
12  Widget build(BuildContext context) {
13    return ListView(
14      children: [
15        Center(
16          child: Text("HomePage"),
17        ) // Center
18      ],
19    ); // ListView
20  }
21  }
22  }

```

ภาพประกอบ 5.47 แก้ไขส่วน return Container() (ต่อ)

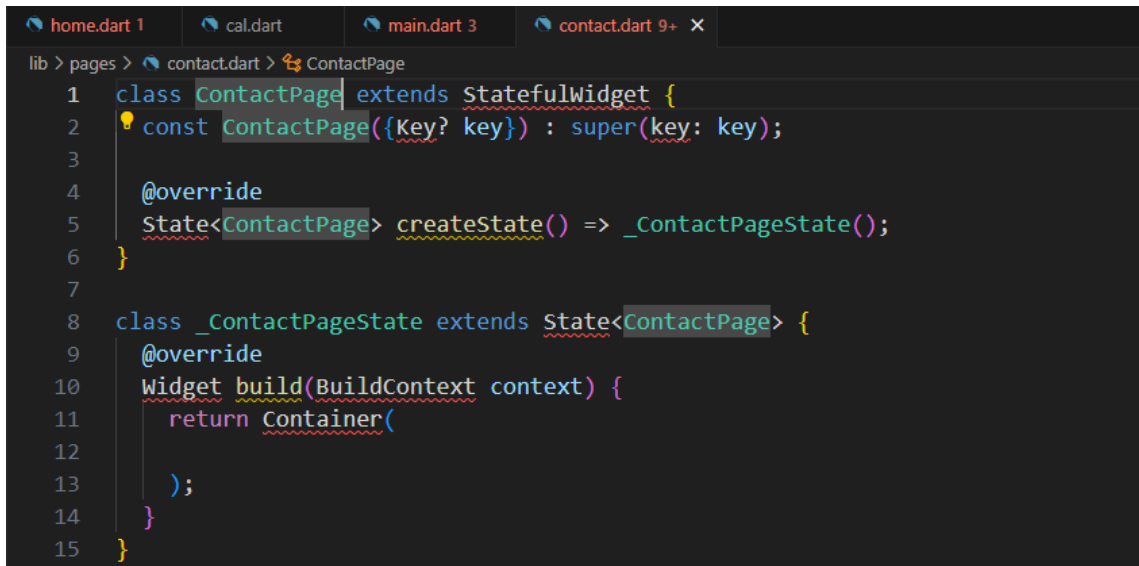
ที่มา : ฌปภัช วรณตรง (2564 : 16)

14. ไปที่ไฟล์ `contact.dart` จากนั้นให้ทำการสร้าง class `StatefulWidget` ตั้งชื่อ class ว่า `ContactPage` จากนั้นให้ทำการ import package เข้ามา และ comment ส่วน `const` ใน class ไว้  
ให้ทำการเปิดไฟล์ `contact.dart` จากนั้นให้ทำการเพิ่ม class `StatefulWidget` โดยใช้ขั้นตอนการเพิ่ม class เหมือนกันกับ `cal.dart` และ `home.dart`



ภาพประกอบ 5.48 สร้าง class `ContactPage`

ที่มา : ฌปภัช วรณตรง (2564 : 17)



ภาพประกอบ 5.49 สร้าง class `ContactPage` (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 17)



```

lib > pages > contact.dart > ContactPage
1 class ContactPage extends StatefulWidget {
2   const ContactPage ({Key? key}) : super(key: key);
3
4   Quick Fix
5   Import library 'package:flutter/material.dart'
6   Import library 'package:flutter/widgets.dart'
7   Import library 'package:flutter/cupertino.dart'
8   Create class 'StatefulWidget'
9   Remove 'extends' clause
10  Move
11  Move 'ContactPage' to file
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

ภาพประกอบ 5.50 สร้าง class ContactPage (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 17)

```

home.dart 1 | cal.dart | main.dart 3 | contact.dart x
lib > pages > contact.dart > ContactPage
1 import 'package:flutter/material.dart';
2
3 class ContactPage extends StatefulWidget {
4   const ContactPage({Key? key}) : super(key: key);
5
6   @override
7   State<ContactPage> createState() => _ContactPageState();
8 }
9
10 class _ContactPageState extends State<ContactPage> {
11   @override
12   Widget build(BuildContext context) {
13     return Container(
14
15     );
16   }
17 }

```

ภาพประกอบ 5.51 สร้าง class ContactPage (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 17)

```

lib > pages > contact.dart > ContactPage
1 import 'package:flutter/material.dart';
2
3 class ContactPage extends StatefulWidget {
4   //const ContactPage({Key? key}) : super(key: key);
5
6   @override
7   _ContactPageState createState() => _ContactPageState();
8 }
9
10 class _ContactPageState extends State<ContactPage> {
11   @override
12   Widget build(BuildContext context) {
13     return Container(
14

```

ภาพประกอบ 5.52 สร้าง class ContactPage (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 17)

15. แก้ไขโค้ดใน return Container() ให้เป็น ListView() และเพิ่มโค้ดในไฟล์ contact.dart ให้ทำการเปลี่ยน Container(...) เป็น ListView(...)

```

lib > pages > contact.dart > _ContactPageState > build
7   State<ContactPage> createState() => _ContactPageState();
8 }
9
10 class _ContactPageState extends State<ContactPage> {
11   @override
12   Widget build(BuildContext context) {
13     return Container(
14
15     );
16   }
17 }

```

ภาพประกอบ 5.53 แก้ไขส่วน return Container()

ที่มา : ฌปภัช วรณตรง (2564 : 18)

```

lib > pages > contact.dart > _ContactPageState > build
@override
7   State<ContactPage> createState() => _ContactPageState();
8 }
9
10 class _ContactPageState extends State<ContactPage> {
11   @override
12   Widget build(BuildContext context) {
13     return ListView
14   }
15 }

```

Dropdown menu options for ListView:

- ListView... ({Key? key, Axis scrollDirectio...
- ListView.builder(...)
- ListView.custom(...)
- ListView.separated(...)
- ListView
- ListWheelViewport(...)
- ListWheelViewport

ภาพประกอบ 5.54 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 18)

15.1 ภายใน ListView(...) ให้ทำการเพิ่มโค้ดเข้าไปเพื่อแสดงข้อมูลการติดต่อ โดยใช้ขั้นตอนเดียวกับ cal.dart และ home.dart

```

lib > pages > contact.dart > _ContactPageState > build
@override
7   State<ContactPage> createState() => _ContactPageState();
8 }
9
10 class _ContactPageState extends State<ContactPage> {
11   @override
12   Widget build(BuildContext context) {
13     return ListView(
14       ch
15     )
16   }
17 }

```

Dropdown menu options for ListView children:

- children: [] List<Widg...>
- cacheExtent:
- clipBehavior:
- semanticChildCount:

ภาพประกอบ 5.55 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 18)

```

lib > pages > contact.dart > _ContactPageState > build
7   State<ContactPage> createState() => _ContactPageState();
8   }
9
10  class _ContactPageState extends State<ContactPage> {
11    @override
12    Widget build(BuildContext context) {
13      return ListView(
14        children: [
15          Ce
16        ],
17      );
18    }
19  }

```

ภาพประกอบ 5.56 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 18)

```

lib > pages > contact.dart > _ContactPageState > build
7   State<ContactPage> createState() => _ContactPageState();
8   }
9
10  class _ContactPageState extends State<ContactPage> {
11    @override
12    Widget build(BuildContext context) {
13      return Container(
14        child:
15        Cent
16      );
17    }
18  }
19  }
20  }
21  }

```

ภาพประกอบ 5.57 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 18)

15.2 ภายใน child: ให้ทำการเพิ่ม Column(...) เข้าไป

```

lib > pages > contact.dart > _ContactPageState > build
7   @override
8   State<ContactPage> createState() => _ContactPageState();
9
10  class _ContactPageState extends State<ContactPage> {
11    @override
12    Widget build(BuildContext context) {
13      return ListView(
14        children: [
15          Center(
16            child: Colu,
17          ) // Center
18        ],
19      ) // ListView
20    }
21  }

```

Column(...) (Key? key, MainAxisAlignment m...)

- MaxColumnWidth
- MinColumnWidth
- MaxColumnWidth(...)
- MinColumnWidth(...)

ภาพประกอบ 5.58 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 18)

15.3 ภายใน Column(...) ให้ทำการเพิ่ม children: [...]

```

lib > pages > contact.dart > _ContactPageState > build
10  class _ContactPageState extends State<ContactPage> {
11    @override
12    Widget build(BuildContext context) {
13      return ListView(
14        children: [
15          Center(
16            child: Column(ch),
17          ) // Center
18        ],
19      ) // ListView
20    }
21  }

```

children: [] List<Widget>

ภาพประกอบ 5.59 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 18)

15.4 เมื่อได้ children: [...] แล้ว ให้ทำการเพิ่มโค้ดสำหรับแสดงข้อความ Text(...) เข้าไปใน [...]

```

10 class _ContactPageState extends State<ContactPage> {
11   @override
12   Widget build(BuildContext context) {
13     return ListView(
14       children: [
15         Center(
16           child: Column(children: [
17             text
18             ],), // Text(...) (String data, {Key? key, Text5...>
19           ) // Center
20         ],
21       ) // ListView
22     ) // TextButton(...)
23     ) // TextButton.icon(...)

```

ภาพประกอบ 5.60 แก๊ไขส่วน return Container() (ต่อ)

ที่มา : ฅปภัซ วรณตรง (2564 : 18)

15.5 ภายใน Text(...) ให้ทำการเพิ่มข้อความเข้าไปโดยใช้เครื่องหมาย “” สำหรับเก็บข้อความ

```

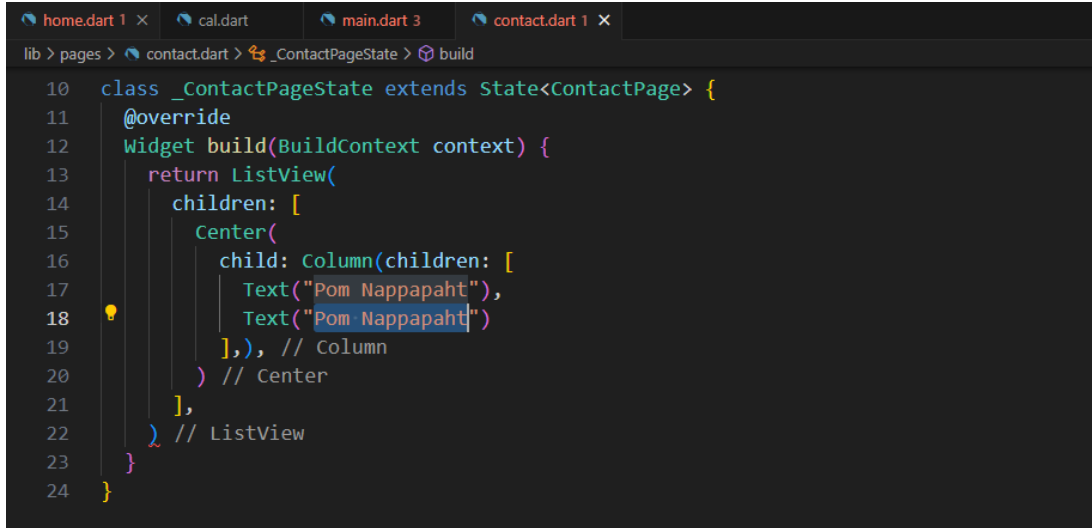
10 class _ContactPageState extends State<ContactPage> {
11   @override
12   Widget build(BuildContext context) {
13     return ListView(
14       children: [
15         Center(
16           child: Column(
17             children: [
18               Text("Pom Nappapaht")
19             ]
20           ) // Column
21         ) // Center

```

ภาพประกอบ 5.61 แก๊ไขส่วน return Container() (ต่อ)

ที่มา : ฅปภัซ วรณตรง (2564 : 18)

15.6 สามารถเพิ่ม Text(...) ได้ตามต้องการโดยการคัดลอกวางหรือพิมพ์โค้ดเพิ่มเข้ามาใหม่ จากนั้นให้เปลี่ยนข้อความภายใน “ ”



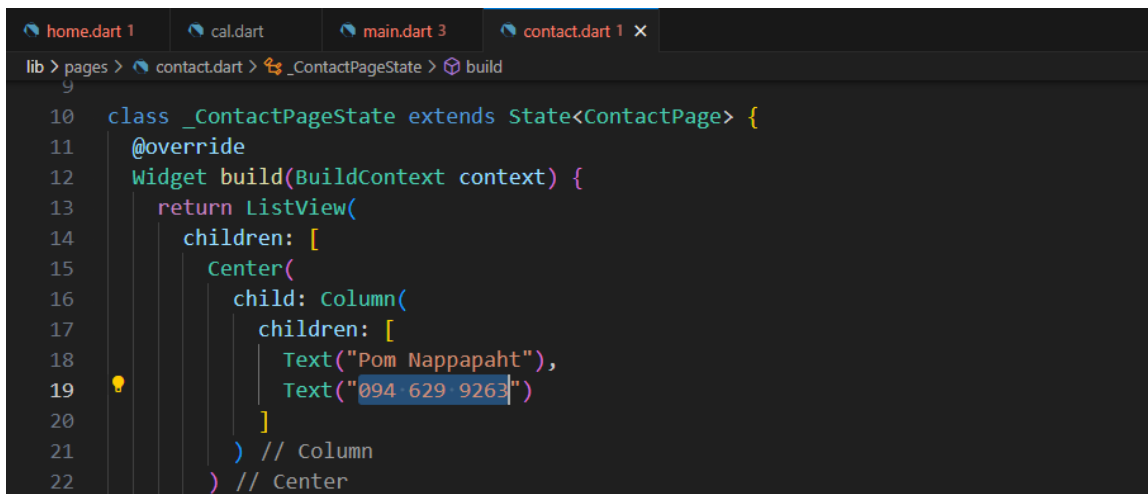
```

10 class _ContactPageState extends State<ContactPage> {
11   @override
12   Widget build(BuildContext context) {
13     return ListView(
14       children: [
15         Center(
16           child: Column(children: [
17             Text("Pom Nappapaht"),
18             Text("Pom Nappapaht")
19           ]), // Column
20         ) // Center
21       ],
22     ) // ListView
23   }
24 }

```

ภาพประกอบ 5.62 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 18)



```

10 class _ContactPageState extends State<ContactPage> {
11   @override
12   Widget build(BuildContext context) {
13     return ListView(
14       children: [
15         Center(
16           child: Column(
17             children: [
18               Text("Pom Nappapaht"),
19               Text("094 629 9263")
20             ]
21           ) // Column
22         ) // Center
23       ]
24     ) // ListView
25   }
26 }

```

ภาพประกอบ 5.63 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 18)

## 1. BottomNavigationBar

BottomNavigationBar เป็นแถบทูลบาร์ (Toolbar) ที่จะปรากฏอยู่ตรงขอบล่างของหน้าจอ สำหรับกำหนดปุ่มเพื่อเชื่อมโยงหรือเลื่อนไปยังหน้าจอต่าง ๆ ซึ่งเป็นรูปแบบที่นิยมใช้กันเป็นส่วนใหญ่ ในปัจจุบันทั้งบนระบบ iOS และ Android (ปัญญา ปะลีละเตสัง, 2566: 339)

การกำหนด BottomNavigationBar วิตเจ็ต เพื่อแสดงยังส่วนล่างของ Scaffold วิตเจ็ต จะต้องกำหนดค่าใน 3 ส่วน ดังต่อไปนี้ (จิราวุธ วารินทร์, 2564 : 288)

1.1 ส่วนแรก คือ ไอเทมต่าง ๆ ที่จะบรรจุไว้ใน BottomNavigationBar ซึ่งแต่ละไอเทมจะสร้างมาจากคลาส BottomNavigationBarItem

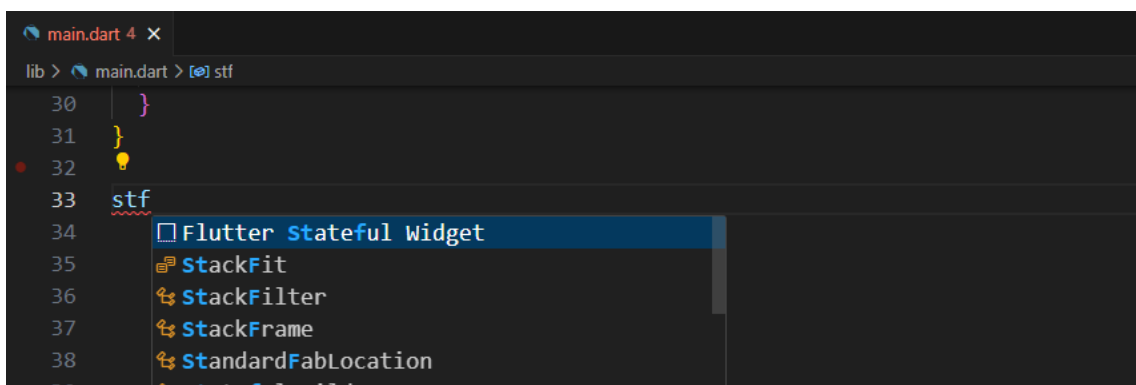
1.2 ส่วนที่สอง คือ ค่า index ที่บอกให้ทราบว่าปัจจุบันเลือกใช้อิเทมใดอยู่

1.3 ส่วนที่สาม คือ ฟังก์ชันสำหรับติดตามการเลือกไอเทมของผู้ใช้ เช่น เมื่อแตะไอเทมแรก ค่า index ก็จะถูกกำหนดให้เป็น 0 และไอเทมที่สองก็มีค่า index เป็น 1 และถ้าแตะที่ไอเทมที่สามก็จะมีค่า index เป็น 2 ตามลำดับ

ในลำดับถัดไปจะทำการสร้างแท็บบาร์ที่ด้านล่างของแอปพลิเคชันเพื่อสามารถคลิกไปยังหน้าแอปพลิเคชันแต่ละหน้าที่สร้างไว้ โดยมีรายละเอียดขั้นตอนการทำดังนี้

## 2. การสร้างแท็บบาร์ที่ด้านล่างของแอปพลิเคชัน

2.1 เปิดไฟล์ main.dart จากนั้นให้ทำการสร้าง class StatefulWidget{} ตั้งชื่อ class และ const ว่า MainPage



ภาพประกอบ 5.64 สร้าง class MainPage

ที่มา : ฅนปักษ์ วรณตรง (2564 : 20)



```

home.dart 1 | cal.dart | main.dart 8 x | contact.dart 1
lib > main.dart > <unnamed>
31   }
32 }
33 class | extends StatefulWidget {
34   const ({Key? key}) : super(key: key);
35
36   @override
37   State<> createState() => _State();
38 }
39
40 class _State extends State<> {
41   @override
42   Widget build(BuildContext context) {
43     return Container(
44
45   );
46 }
47 }

```

ภาพประกอบ 5.65 สร้าง class MainPage (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 20)

```

home.dart 1 | cal.dart | main.dart 3 x | contact.dart 1
lib > main.dart > MainPage
32 }
33 class MainPage extends StatefulWidget {
34   const MainPage({Key? key}) : super(key: key);
35
36   @override
37   State<MainPage> createState() => _MainPageState();
38 }
39
40 class _MainPageState extends State<MainPage> {
41   @override
42   Widget build(BuildContext context) {
43     return Container(
44
45   );
46 }
47 }

```

ภาพประกอบ 5.66 สร้าง class MainPage (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 20)

```

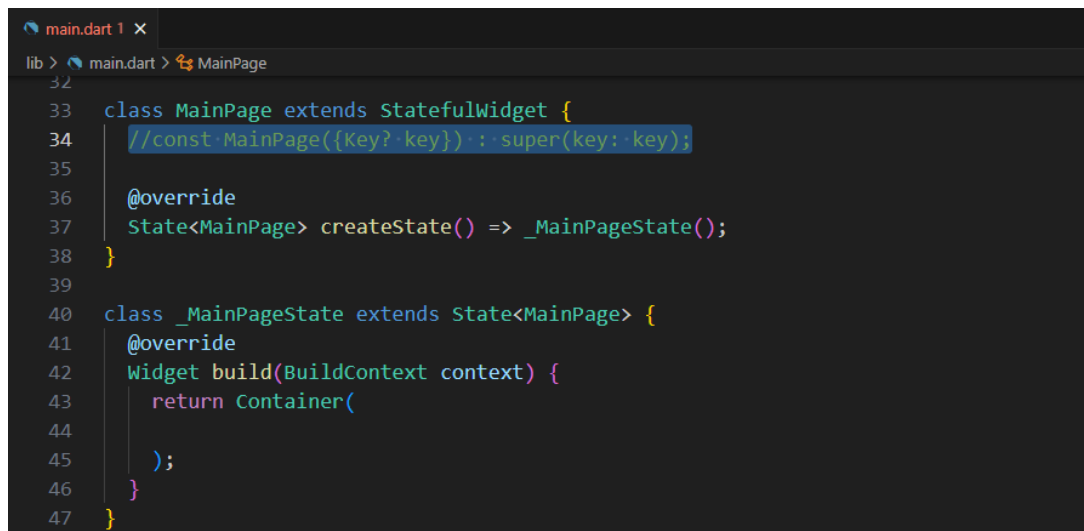
32
33 class MainPage extends StatefulWidget {
34   const MainPage({Key? key}) : super(key: key);
35
36   @override
37   State<MainPage> createState() => _MainPageState();
38 }
39
40 class _MainPageState extends State<MainPage> {
41   @override
42   Widget build(BuildContext context) {
43     return Container(
44       |
45     );
46   }
47 }

```

ภาพประกอบ 5.67 สร้าง class MainPage (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 20)

ให้ทำการปิดโค้ดที่ไม่ได้ใช้งานด้วยการ Comment โค้ดบรรทัดนั้น ๆ



```

main.dart 1 x
lib > main.dart > MainPage
32
33 class MainPage extends StatefulWidget {
34   //const MainPage({Key? key}) : super(key: key);
35
36   @override
37   State<MainPage> createState() => _MainPageState();
38 }
39
40 class _MainPageState extends State<MainPage> {
41   @override
42   Widget build(BuildContext context) {
43     return Container(
44       |
45     );
46   }
47 }

```

ภาพประกอบ 5.68 สร้าง class MainPage (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 20)

2.2 ในไฟล์ main.dart ให้ทำการตัดส่วน Scaffold() ใน tag home: ออกทั้งหมดตามภาพ  
ไปที่ MaterialApp(...) จากนั้นให้ทำการตัดโค้ดส่วน Scaffold(...) ไว้

```

13 @override
14 Widget build(BuildContext context) {
15   return MaterialApp(
16     home: Scaffold (
17       appBar: AppBar(title: Text("แอปคำนวณ"), backgroundColor:
18       body:
19     ), // Scaffold
20   ); // MaterialApp
21 }
22 }
23 class MainPage extends StatefulWidget {
24   //const MainPage({Key? key}) : super(key: key);
25
26 @override

```

ภาพประกอบ 5.69 ตัดส่วน Scaffold()

ที่มา : ฅนปลัซ วรณตรง (2564 : 21)

```

13 @override
14 Widget build(BuildContext context) {
15   return MaterialApp(
16     home: Scaffold (
17       appBar: AppBar(title: Text("แอปคำนวณ"), backgroundColor: Color.fromARGB(255, 255, 255, 255)
18       body:
19     ), // Scaffold
20   ); // MaterialApp
21 }

```

ภาพประกอบ 5.70 ตัดส่วน Scaffold() (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 21)



```

main.dart 1 x
lib > main.dart > MyApp > build
8   }
9
10  class MyApp extends StatelessWidget {
11    // const MyApp({ Key? key }) : super(key: key);
12
13    @override
14    Widget build(BuildContext context) {
15      return MaterialApp(
16        home: );
17    }
18  }

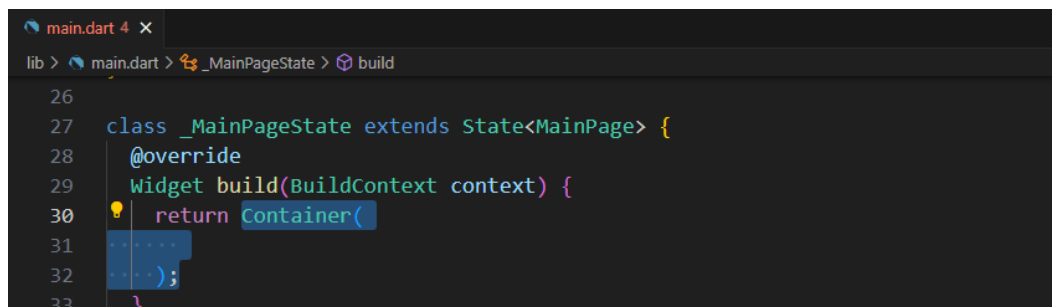
```

ภาพประกอบ 5.71 ตัดส่วน Scaffold() (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 21)

2.3 นำโค้ดที่ทำการตัดมา มาใส่แทนส่วน return Container(); ใน class MainPage ไปที่ class \_MainPageState จากนั้นให้ทำการวางโค้ดที่ได้ทำการตัด เข้าไปแทนที่

Container(...)



```

main.dart 4 x
lib > main.dart > _MainPageState > build
26
27  class _MainPageState extends State<MainPage> {
28    @override
29    Widget build(BuildContext context) {
30      return Container(
31        ...
32      );
33    }

```

ภาพประกอบ 5.72 นำส่วน Scaffold() มาใส่แทน Container()

ที่มา : ฅนปักษ์ วรณตรง (2564 : 22)

```

main.dart 7 x
lib > main.dart > _MainPageState > build
25 }
26
27 class _MainPageState extends State<MainPage> {
28   @override
29   Widget build(BuildContext context) {
30     return Scaffold (
31       appBar: AppBar(title: Text("แอปสำนวน"), backgroundColor: Color.fromARGB(
32         body: Home()),
33     ), // Scaffold
34   }

```

ภาพประกอบ 5.73 นำส่วน Scaffold() มาใส่แทน Container() (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 22)

```

main.dart 7 x
lib > main.dart > _MainPageState > build
25 }
26
27 class _MainPageState extends State<MainPage> {
28   @override
29   Widget build(BuildContext context) {
30     return Scaffold (
31       appBar: AppBar(title: Text("แอปสำนวน"), backgroundColor: Color.fromARGB(25
32         body: Home()),
33     ), // Scaffold
34   }

```

ภาพประกอบ 5.74 นำส่วน Scaffold() มาใส่แทน Container() (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 22)

2.4 ในส่วนโค้ด tag home: ให้ทำการใส่ MainPage() เข้าไปเพื่อทำการเรียกใช้งาน class MainPage ที่ได้ทำการเขียนไว้ ตามภาพ

ไปที่ MaterialApp(...) ต่อจาก home: ให้ทำการเพิ่มโค้ด MainPage() เพื่อกำหนดหน้าเริ่มต้นของแอปพลิเคชัน

```

main.dart 4 x
lib > main.dart > MyApp > build
11 // const MyApp({ Key? key }) : super(key: key);
12
13 @override
14 Widget build(BuildContext context) {
15   return MaterialApp(
16     home: Main);
17 }
18 }
19
20 class MainPage ext
21 //const MainPage

```

ภาพประกอบ 5.75 ใส่ MainPage() ในส่วน tag home:

ที่มา : ฅนปลัซ วรณตรง (2564 : 23)

```

main.dart 3 x
lib > main.dart > MyApp > build
11 // const MyApp({ Key? key }) : super(key: key);
12
13 @override
14 Widget build(BuildContext context) {
15   return MaterialApp(
16     home: MainPage()); // MaterialApp
17 }
18 }
19

```

ภาพประกอบ 5.76 ใส่ MainPage() ในส่วน tag home: (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 23)

```

main.dart 3 x
lib > main.dart > MyApp > build
11 // const MyApp({ Key? key }) : super(key: key);
12
13 @override
14 Widget build(BuildContext context) {
15   return MaterialApp(
16     home: MainPage()); // MaterialApp
17 }
18 }
19

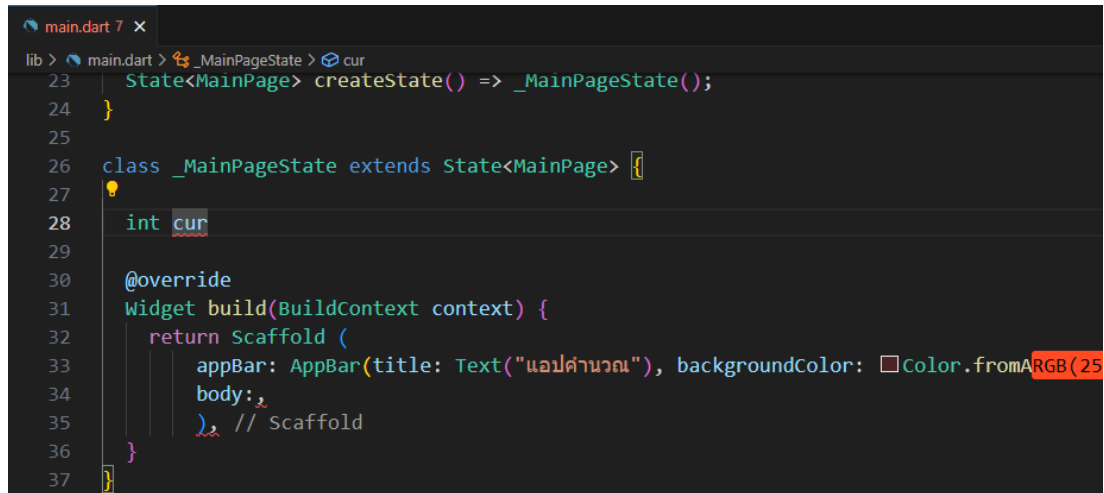
```

ภาพประกอบ 5.77 ใส่ MainPage() ในส่วน tag home: (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 23)

2.5 ทำการเพิ่มโค้ด `int currentIndex = 0;` และ `final tabs = [HomePage(), CalculatePage(), ContactPage()];` เข้าไปใน class `MainPage`

2.5.1 ไปที่ `MainPageState` ภายใน `{...}` ให้ทำการเพิ่มโค้ดเพื่อประกาศตัวแปรสำหรับรับค่าหน้าที่ผู้ใช้เปิดอยู่

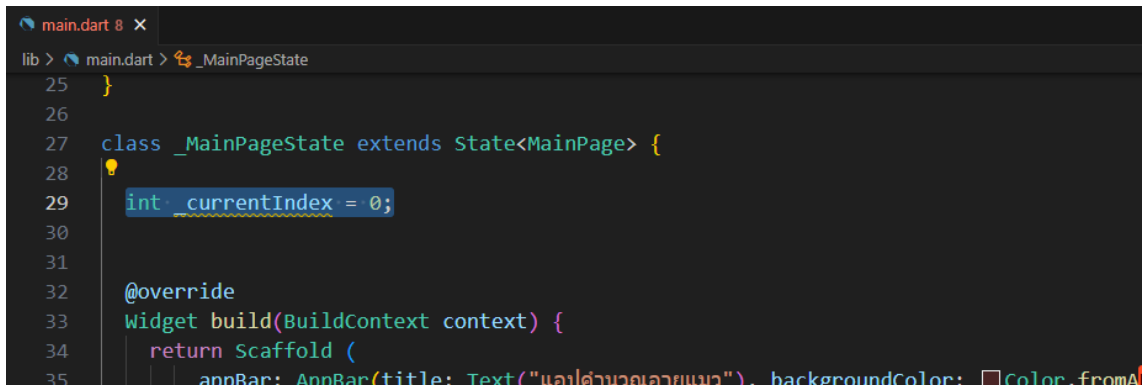


```

main.dart 7 x
lib > main.dart > _MainPageState > cur
23 State<MainPage> createState() => _MainPageState();
24 }
25
26 class _MainPageState extends State<MainPage> {
27
28   int cur
29
30   @override
31   Widget build(BuildContext context) {
32     return Scaffold (
33       appBar: AppBar(title: Text("แอปคำนวณ"), backgroundColor: Color.fromARGB(25
34       body:
35     ), // Scaffold
36   }
37 }

```

ภาพประกอบ 5.78 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class `MainPage`  
ที่มา : ฌปภัช วรรณตรง (2564 : 24)



```

main.dart 8 x
lib > main.dart > _MainPageState
25 }
26
27 class _MainPageState extends State<MainPage> {
28
29   int currentIndex = 0;
30
31
32   @override
33   Widget build(BuildContext context) {
34     return Scaffold (
35       appBar: AppBar(title: Text("แอปคำนวณเลขแนว"), backgroundColor: Color.fromA

```

ภาพประกอบ 5.79 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class `MainPage` (ต่อ)  
ที่มา : ฌปภัช วรรณตรง (2564 : 24)

### 2.5.2 เพิ่มตัวแปรที่ไม่สามารถเปลี่ยนแปลงได้สำหรับเก็บหน้าเว็บแต่ละหน้า โดยพิมพ์

final tabs = [หน้าเว็บที่ต้องการเก็บ]

```
main.dart 8 x
lib > main.dart > _MainPageState
25 }
26
27 class _MainPageState extends State<MainPage> {
28
29   int _currentIndex = 0;
30   final tabs = [];
31
32   @override
33   Widget build(BuildContext context) {
34     return Scaffold (
```

ภาพประกอบ 5.80 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ)

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 24)

### 2.5.3 ภายใน [...] ให้ทำการพิมพ์ชื่อ class ของแต่ละหน้าเว็บที่ต้องการจะเรียกใช้เข้า

ไป โดยที่เมื่อพิมพ์โค้ดไปบางส่วนแล้วจะมีหน้าต่างชุดคำสั่งขึ้นมาให้เลือก เมื่อเลือกแล้วจะทำการ import ไฟล์นั้น ๆ เข้ามาให้ในไฟล์ main.dart

```
main.dart 8 x
lib > main.dart > _MainPageState > tabs
27 class _MainPageState extends State<MainPage> {
28
29   int _currentIndex = 0;
30   final tabs = [Hom
31
32   @override
33   Widget build(Buil
34     return Scaffold
35     appBar: AppBa
36     body:
37
```

ภาพประกอบ 5.81 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ)

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 24)



```

main.dart 7 x
lib > main.dart > _MainPageState > tabs
25 }
26
27 class _MainPageState extends State<MainPage> {
28
29   int _currentIndex = 0;
30   final tabs = [HomePage()];
31
32   @override
33   Widget build(BuildContext context) {

```

ภาพประกอบ 5.82 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ)  
ที่มา : ฌปภัช วรณตรง (2564 : 24)

```

main.dart 7 x
lib > main.dart > _MainPageState > tabs
27 class _MainPageState extends State<MainPage> {
28
29   int _currentIndex = 0;
30   final tabs = [HomePage(), d]
31
32   @override
33   Widget build(BuildContext
34   return Scaffold(
35   appBar: AppBar(title:

```

ภาพประกอบ 5.83 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ)  
ที่มา : ฌปภัช วรณตรง (2564 : 24)

```

main.dart 6 x
lib > main.dart > _MainPageState > tabs
25 }
26
27 class _MainPageState extends State<MainPage> {
28
29   int _currentIndex = 0;
30   final tabs = [HomePage(), CalculatePage()];
31
32   @override
33   Widget build(BuildContext context) {
34   return Scaffold (

```

ภาพประกอบ 5.84 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ)  
ที่มา : ฌปภัช วรณตรง (2564 : 24)

```

main.dart 5 x
lib > main.dart > _MainPageState
25 }
26
27 class _MainPageState extends State<MainPage> {
28
29   int currentIndex = 0;
30   final tabs = [HomePage(), CalculatePage(), ContactPage()];
31
32   @override
33   Widget build(BuildContext context) {
34     return Scaffold (
35       appBar: AppBar(title: Text("แอปคำนวณอายุแมว"), backgroundColor: Color.fro

```

ภาพประกอบ 5.85 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 24)

2.6 ใน tag body: ให้ทำการเพิ่มโค้ดลงไปตามภาพ

2.6.1 ภายใน Scaffold(...) ต่อจาก body: ให้ทำการเพิ่มโค้ด tabs[...] เพื่อนำค่าตัวแปรที่ได้กำหนดมาใช้งาน

```

main.dart 2 x
lib > main.dart > _MainPageState > build
23 @override
24 _MainPageState createState() => _MainPageState();
25 }
26
27 class _MainPageState extends State<MainPage> {
28
29   int currentIndex = 0;
30   final tabs = [HomePage(), CalculatePage(), ContactPage()];
31
32   @override
33   Widget build(BuildContext context) {
34     return Scaffold (
35       appBar: AppBar(title: Text("แอปคำนวณอายุแมว"), backgroundColor: Color.fromARC
36       body: tabs[],
37
38     ); // Scaffold

```

ภาพประกอบ 5.86 เพิ่มโค้ดในส่วนของ tag body:

ที่มา : ฅนปักษ์ วรณตรง (2564 : 25)

2.6.2 ภายใน tabs[...] ให้ทำการพิมพ์ตัวแปรที่กำหนดไว้มาใส่ โดยการพิมพ์  
\_currentIndex

```
main.dart x
lib > main.dart > _MainPageState > build
25 }
26
27 class _MainPageState extends State<MainPage> {
28
29   int _currentIndex = 0;
30   final tabs = [HomePage(), CalculatePage(), ContactPage()];
31
32   @override
33   Widget build(BuildContext context) {
34     return Scaffold (
35       appBar: AppBar(title: Text("แอปคำนวณอายุแมว"), backgroundColor: Color.fromARGB(
36         body: tabs[_currentIndex],
37
38     ); // Scaffold
39   }
40 }
```

ภาพประกอบ 5.87 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 25)

2.6.3 จากนั้นให้ทำการเพิ่มโค้ด bottomNavigationBar: ซึ่งเป็นโค้ดส่วนสำหรับการ  
สร้างแท็บบาร์ด้านล่างแอปพลิเคชัน

```
main.dart 3 x
lib > main.dart > _MainPageState > build
29   int _currentIndex = 0;
30   final tabs = [HomePage(), CalculatePage(), ContactPage()];
31
32   @override
33   Widget build(BuildContext context) {
34     return Scaffold(
35       appBar: AppBar(title: Text("แอปคำนวณ"), backgroundColor: Color.fromARGB(
36       body: tabs[_currentIndex],
37       b
38     ) bottomNavigationBar: Widget?
39       bottomSheet:
40     } extendBody:
41   } extendBodyBehindAppBar:
```

ภาพประกอบ 5.88 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 25)

### 2.6.4 ต่อจาก bottomNavigationBar: ให้ทำการเพิ่ม BottomNavigationBar(...)

```

main.dart 1 x
lib > main.dart > _MainPageState > build
29   int _currentIndex = 0;
30   final tabs = [HomePage(), CalculatePage(), ContactPage()];
31
32   @override
33   Widget build(BuildContext context) {
34     return Scaffold(
35       appBar: AppBar(title: Text("แอปคำนวณ"), backgroundColor: Color.fromARGB(255, 255, 255, 255)),
36       body: tabs[_currentIndex],
37       bottomNavigationBar: BottomNavigationBar(
38     ); // Scaffold
39   );
40 }
41 }

```

ภาพประกอบ 5.89 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 25)

2.6.5 ภายใน BottomNavigationBar(...) ให้ทำการเพิ่มโค้ดเข้าไป โดยส่วนแรกจะเป็นการประกาศเพื่อเรียกตัวแปรมาใช้งาน

```

main.dart 3 x
lib > main.dart > _MainPageState > build
31
32   @override
33   Widget build(BuildContext context) {
34     return Scaffold(
35       appBar: AppBar(title: Text("แอปคำนวณ"), backgroundColor: Color.fromARGB(255, 255, 255, 255)),
36       body: tabs[_currentIndex],
37       bottomNavigationBar: BottomNavigationBar(
38     ),
39     );
40     );
41     );
42     );
43     );

```

ภาพประกอบ 5.90 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 25)

```

main.dart 1 x
lib > main.dart > _MainPageState > build
27 class _MainPageState extends State<MainPage> {
28
29   int _currentIndex = 0;
30   final tabs = [HomePage(), CalculatePage(), ContactPage()];
31
32   @override
33   Widget build(BuildContext context) {
34     return Scaffold (
35       appBar: AppBar(title: Text("แอปคำนวณอายุแมว"), backgroundColor: Color.from
36       body: tabs[_currentIndex],
37       bottomNavigationBar: BottomNavigationBar(
38         currentIndex: _currentIndex,
39       ) // BottomNavigationBar
40
41     ); // Scaffold
42   }
43 }

```

ภาพประกอบ 5.91 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 25)

2.6.6 จากนั้นให้ทำการเพิ่มโค้ด Item: [...] เข้าไปต่อจาก \_currentIndex

```

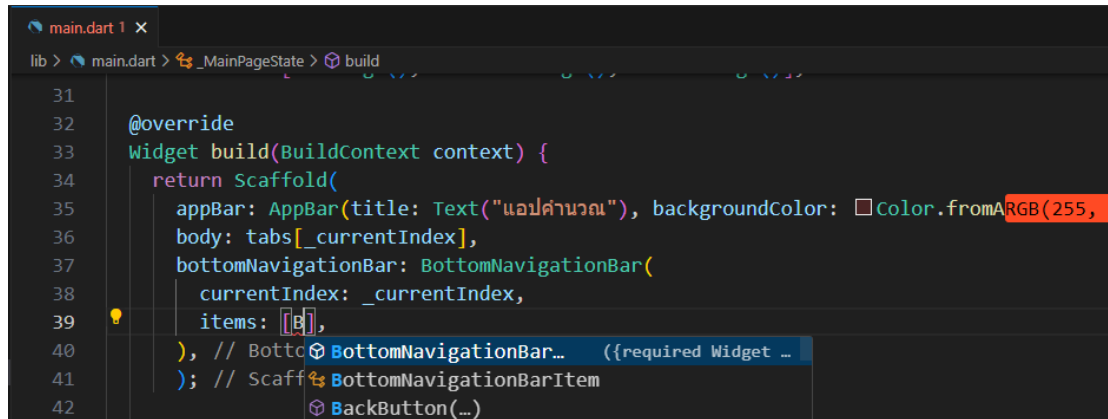
main.dart 4 x
lib > main.dart > _MainPageState > build
31
32   @override
33   Widget build(BuildContext context) {
34     return Scaffold(
35       appBar: AppBar(title: Text("แอปคำนวณ"), backgroundColor: Color.fromARGB(2
36       body: tabs[_currentIndex],
37       bottomNavigationBar: BottomNavigationBar(
38         currentIndex: _currentIndex,
39         items: [] List<BottomNavigationBarItem>
40       );
41     );
42   }
43 }
44

```

ภาพประกอบ 5.92 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 25)

### 2.6.7 ภายใน Item: [...] ให้ทำการเพิ่ม BottomNavigationBarItem(...)



```

31
32 @override
33 Widget build(BuildContext context) {
34   return Scaffold(
35     appBar: AppBar(title: Text("แอปคำนวณ"), backgroundColor: Color.fromARGB(255,
36     body: tabs[_currentIndex],
37     bottomNavigationBar: BottomNavigationBar(
38       currentIndex: _currentIndex,
39     items: []),
40   ), // BottomNavigationBar... ((required Widget ...
41   ); // ScaffoldBottomNavigationBarItem
42   BackButton(...)
```

ภาพประกอบ 5.93 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 25)

2.6.8 ภายใน icon: ให้ทำการเพิ่มโค้ด Icon(Icons.[ประเภทของ Icon]) ซึ่งเป็นการใส่ icon ที่จะแสดงบนแท็บบาร์



```

31
32 @override
33 Widget build(BuildContext context) {
34   return Scaffold(
35     appBar: AppBar(title: Text("แอปคำนวณ"), backgroundColor: Color.fromARGB(255,
36     body: tabs[_currentIndex],
37     bottomNavigationBar: BottomNavigationBar(
38       currentIndex: _currentIndex,
39     items: [BottomNavigationBarItem(icon: icon)],
40   ), // BottomNavigationBar
41   ); // Scaffold
42
```

ภาพประกอบ 5.94 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 25)

```

31
32 @override
33 Widget build(BuildContext context) {
34   return Scaffold(
35     appBar: AppBar(title: Text("แอปคำนวณ"), backgroundColor: Color.fromARGB(255, 255, 255, 255)),
36     body: tabs[_currentIndex],
37     bottomNavigationBar: BottomNavigationBar(
38       currentIndex: _currentIndex,
39       items: [BottomNavigationBarItem(icon: Icon(Icons.home))],
40     ), // BottomNavigationBar
41   ); // Scaffold
42
43 }

```

ภาพประกอบ 5.95 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 25)

2.6.9 จากนั้นให้เพิ่ม label: ตามด้วยข้อความที่เก็บด้วยเครื่องหมาย “ ” เพื่อแสดงผล ซึ่งเป็นการใส่ข้อความที่จะแสดงใต้ icon

```

29 int _currentIndex = 0;
30 final tabs = [HomePage(), CalculatePage(), ContactPage()];
31
32 @override
33 Widget build(BuildContext context) {
34   return Scaffold(
35     appBar: AppBar(title: Text("แอปคำนวณอายุแมว"), backgroundColor: Color.fromARGB(255, 255, 255, 255)),
36     body: tabs[_currentIndex],
37     bottomNavigationBar: BottomNavigationBar(
38       currentIndex: _currentIndex,
39       items: [
40         BottomNavigationBarItem(icon: Icon(Icons.home), label: "หน้าแรก"),
41       ]
42     )
43   );

```

ภาพประกอบ 5.96 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 25)

```

main.dart x
lib > main.dart > _MainPageState > build
31
32 @override
33 Widget build(BuildContext context) {
34   return Scaffold(
35     appBar: AppBar(title: Text("แอปคำนวณ"), backgroundColor: Color.fromARGB(255,
36     body: tabs[_currentIndex],
37     bottomNavigationBar: BottomNavigationBar(
38       currentIndex: _currentIndex,
39       items: [
40         BottomNavigationBarItem(icon: Icon(Icons.home), label: "หน้าแรก"),
41
42     ],
43   ), // BottomNavigationBar
44   ); // Scaffold
45

```

ภาพประกอบ 5.97 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 25)

2.6.10 จากนั้นให้ทำการคัดลอกโค้ดเพิ่มมาอีกสองบรรทัด ทำการเปลี่ยนชื่อ Icon และข้อความภายใน “ ”

```

main.dart x
lib > main.dart > _MainPageState > build
31
32 @override
33 Widget build(BuildContext context) {
34   return Scaffold(
35     appBar: AppBar(title: Text("แอปคำนวณอายุแมว"), backgroundColor: Color.fromARGB(
36     body: tabs[_currentIndex],
37     bottomNavigationBar: BottomNavigationBar(
38       currentIndex: _currentIndex,
39       items: [
40         BottomNavigationBarItem(icon: Icon(Icons.home), label: "หน้าแรก"),
41         BottomNavigationBarItem(icon: Icon(Icons.calculate), label: "หน้าคำนวณ")
42     ]
43   ), // BottomNavigationBar
44   ); // Scaffold
45

```

ภาพประกอบ 5.98 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 25)



```

main.dart x
lib > main.dart > _MainPageState > build
31
32 @override
33 Widget build(BuildContext context) {
34   return Scaffold(
35     appBar: AppBar(title: Text("แอปคำนวณอายุแมว"), backgroundColor: Color.fromARGB(
36     body: tabs[_currentIndex],
37     bottomNavigationBar: BottomNavigationBar(
38       currentIndex: _currentIndex,
39       items: [
40         BottomNavigationBarItem(icon: Icon(Icons.home), label: "หน้าแรก"),
41         BottomNavigationBarItem(icon: Icon(Icons.calculate), label: "หน้าคำนวณ"),
42         BottomNavigationBarItem(icon: Icon(Icons.contact_page), label: "หน้าติดต่อเรา")
43       ]
44     )
45   ) // BottomNavigationBar

```

ภาพประกอบ 5.99 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ฅนปลั๊ก วรรณตรง (2564 : 25)

```

main.dart x
lib > main.dart > _MainPageState > build
31
32 @override
33 Widget build(BuildContext context) {
34   return Scaffold(
35     appBar: AppBar(title: Text("แอปคำนวณอายุแมว"), backgroundColor: Color.fromARGB(
36     body: tabs[_currentIndex],
37     bottomNavigationBar: BottomNavigationBar(
38       currentIndex: _currentIndex,
39       items: [
40         BottomNavigationBarItem(icon: Icon(Icons.home), label: "หน้าแรก"),
41         BottomNavigationBarItem(icon: Icon(Icons.calculate), label: "หน้าคำนวณ"),
42         BottomNavigationBarItem(icon: Icon(Icons.contact_page), label: "หน้าติดต่อเรา")
43       ]
44     )
45   ) // BottomNavigationBar

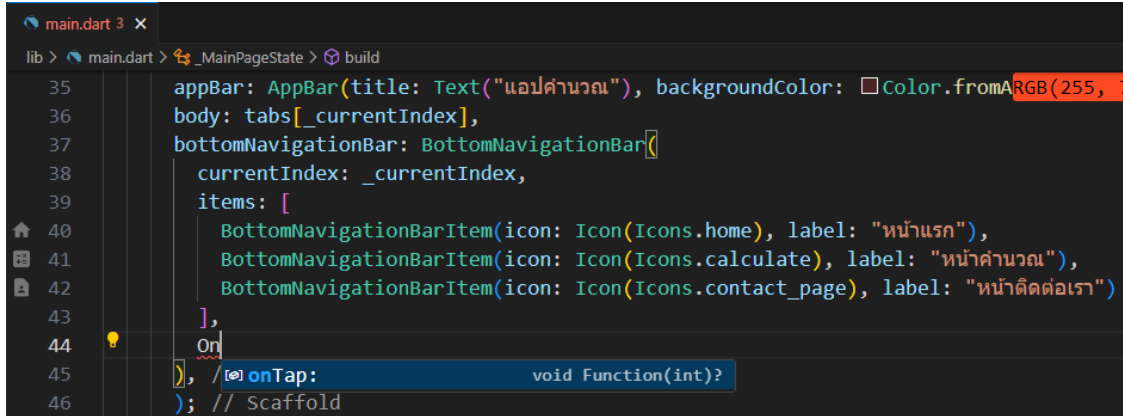
```

ภาพประกอบ 5.100 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ฅนปลั๊ก วรรณตรง (2564 : 25)

2.7 จากนั้นให้ทำการเพิ่มโค้ดต่อจาก item: [],

2.7.1 ให้ทำการเพิ่มโค้ด onTap: (...) {...} ต่อจาก Item: [...]



```

main.dart 3 x
lib > main.dart > _MainPageState > build
35 appBar: AppBar(title: Text("แอปคำนวณ"), backgroundColor: Color.fromARGB(255,
36 body: tabs[_currentIndex],
37 bottomNavigationBar: BottomNavigationBar(
38   currentIndex: _currentIndex,
39   items: [
40     BottomNavigationBarItem(icon: Icon(Icons.home), label: "หน้าแรก"),
41     BottomNavigationBarItem(icon: Icon(Icons.calculate), label: "หน้าคำนวณ"),
42     BottomNavigationBarItem(icon: Icon(Icons.contact_page), label: "หน้าติดต่อเรา")
43   ],
44   onTap: void Function(int)?
45 ); // Scaffold

```

ภาพประกอบ 5.101 เพิ่มโค้ด onTap: (...) {...}

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 26)



```

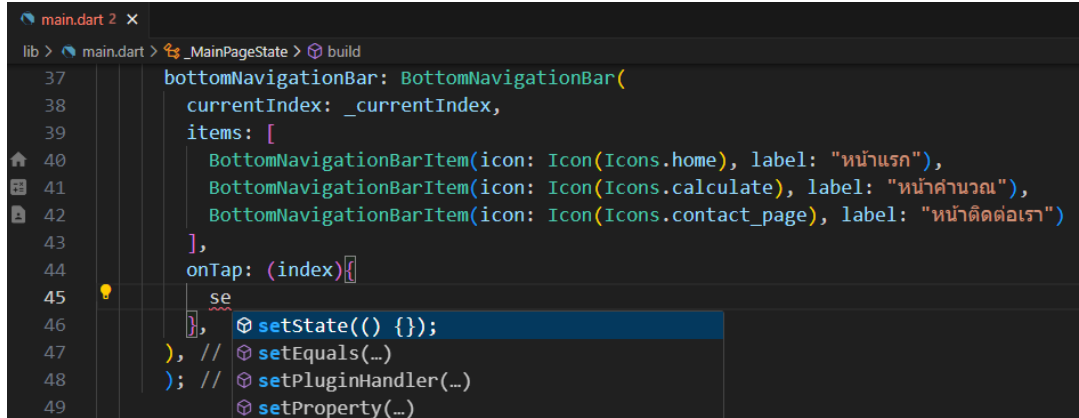
main.dart x
lib > main.dart > _MainPageState > build
36 appBar: AppBar(title: Text("แอปคำนวณ"), backgroundColor: Color.fromARGB(255,
37 body: tabs[_currentIndex],
38 bottomNavigationBar: BottomNavigationBar(
39   currentIndex: _currentIndex,
40   items: [
41     BottomNavigationBarItem(icon: Icon(Icons.home), label: "หน้าแรก"),
42     BottomNavigationBarItem(icon: Icon(Icons.calculate), label: "หน้าคำนวณ"),
43     BottomNavigationBarItem(icon: Icon(Icons.contact_page), label: "หน้าติดต่อเรา")
44     onTap: (index){},
45   ] // BottomNavigationBar
46
47

```

ภาพประกอบ 5.102 เพิ่มโค้ด onTap: (...) {...} (ต่อ)

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 26)

2.7.2 ภายใน {...} ให้ทำการเพิ่มโค้ด setState(...) {...};




```

main.dart 2 x
lib > main.dart > _MainPageState > build
37   bottomNavigationBar: BottomNavigationBar(
38     currentIndex: _currentIndex,
39     items: [
40       BottomNavigationBarItem(icon: Icon(Icons.home), label: "หน้าแรก"),
41       BottomNavigationBarItem(icon: Icon(Icons.calculate), label: "หน้าคำนวณ"),
42       BottomNavigationBarItem(icon: Icon(Icons.contact_page), label: "หน้าติดต่อเรา")
43     ],
44     onTap: (index){
45       se
46     }, // setState(() {});
47     // setEquals(...)
48     // setPluginHandler(...)
49     // setProperty(...)

```

ภาพประกอบ 5.103 เพิ่มโค้ด onTap: (...){...} (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 26)



```

main.dart x
lib > main.dart > _MainPageState > build
35   appBar: AppBar(title: Text("แอปพลิเคชัน"), backgroundColor: Colors.grey),
36   body: tabs[_currentIndex],
37   bottomNavigationBar: BottomNavigationBar(
38     currentIndex: _currentIndex,
39     items: [
40       BottomNavigationBarItem(icon: Icon(Icons.home), label: "หน้าแรก"),
41       BottomNavigationBarItem(icon: Icon(Icons.calculate), label: "หน้าคำนวณ"),
42       BottomNavigationBarItem(icon: Icon(Icons.contact_page), label: "หน้าติดต่อเร
43     ],
44     onTap: (index){
45       setState(() {
46
47       });
48     },

```

ภาพประกอบ 5.104 เพิ่มโค้ด onTap: (...){...} (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 26)

### 2.7.3 จากนั้นให้ทำการเพิ่มโค้ด print(...) สำหรับแสดงผล



```

main.dart x
lib > main.dart > _MainPageState > build
38     currentIndex: _currentIndex,
39     items: [
40       BottomNavigationBarItem(icon: Icon(Icons.home), label: "หน้าแรก"),
41       BottomNavigationBarItem(icon: Icon(Icons.calculate), label: "หน้าคำนวณ"),
42       BottomNavigationBarItem(icon: Icon(Icons.contact_page), label: "หน้าติดต่อ"),
43     ],
44     onTap: (index){
45       setState(() {
46         print(index);
47       });
48     },
49   ) // BottomNavigationBar

```

ภาพประกอบ 5.105 เพิ่มโค้ด onTap: (...){...} (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 26)

### 2.7.4 เพิ่มโค้ดกำหนดค่าตัวแปรให้กับ index



```

main.dart x
lib > main.dart > _MainPageState > build
40     BottomNavigationBarItem(icon: Icon(Icons.home), label: "หน้าแรก"),
41     BottomNavigationBarItem(icon: Icon(Icons.calculate), label: "หน้าคำนวณ"),
42     BottomNavigationBarItem(icon: Icon(Icons.contact_page), label: "หน้าติดต่อ"),
43   ],
44   onTap: (index){
45     setState(() {
46       print(index);
47       currentIndex = index;
48     });
49   },
50 ) // BottomNavigationBar
51

```

ภาพประกอบ 5.106 เพิ่มโค้ด onTap: (...){...} (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 26)

```

main.dart x
lib > main.dart > _MainPageState > build
31
32 @override
33 Widget build(BuildContext context) {
34   return Scaffold(
35     appBar: AppBar(title: Text("แอปคำนวณอายุแมว"), backgroundColor: Color.
36     body: tabs[_currentIndex],
37     bottomNavigationBar: BottomNavigationBar(
38       currentIndex: _currentIndex,
39       items: [
40         BottomNavigationBarItem(icon: Icon(Icons.home), label: "หน้าแรก"),
41         BottomNavigationBarItem(icon: Icon(Icons.calculate), label: "หน้าคำนวณ"),
42         BottomNavigationBarItem(icon: Icon(Icons.contact_page), label: "หน้าติดต่อ"),
43       ],
44       onTap: (index){
45         setState(() {
46           print(index);
47           _currentIndex = index;
48         });
49       },
50     ) // BottomNavigationBar
51
52
53 // Scaffold

```

ภาพประกอบ 5.107 เพิ่มโค้ด onTap: (...){...} (ต่อ)

ที่มา : ฌปภัช วรณตรง (2564 : 26)

#### ข้อควรระวัง

ควรตรวจสอบตัวแปรให้ตรงกัน เพื่อป้องกันความผิดพลาดของโค้ด

## การสร้างไอคอนของแอปพลิเคชัน

เนื้อหาส่วนนี้จะเป็นการสร้างไอคอนของแอปพลิเคชัน การสร้างไอคอนของแอปพลิเคชันเพื่อเมื่อนำไปติดตั้งบนอุปกรณ์เคลื่อนที่จะแสดงเป็นไอคอนบนหน้าจอให้ผู้ใช้คลิกเพื่อเข้าสู่หน้าจอแอปพลิเคชัน มีรายละเอียดขั้นตอนการทำงานดังนี้

1. ในขั้นตอนการสร้าง Project Flutter ตัวโปรเจกต์จะทำการสร้างไอคอนของตัวโปรเจกต์ติดมากับไฟล์งานอยู่แล้ว

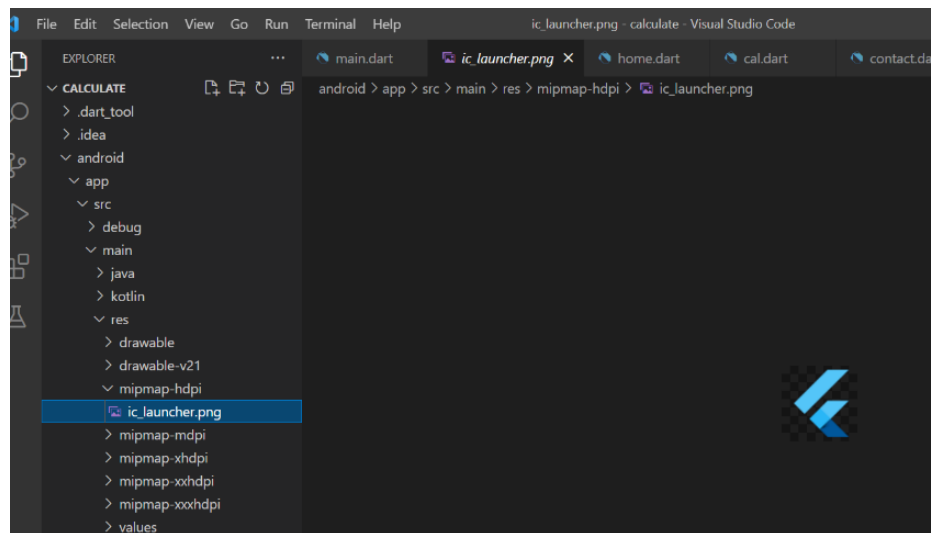
```

C:\Windows\System32\cmd.exe
test12\android\app\build.gradle (created)
test12\android\app\src\main\kotlin\com\example\test12\MainActivity.kt (created)
test12\android\build.gradle (created)
test12\android\test12_android.iml (created)
test12\android\.gitignore (created)
test12\android\app\src\debug\AndroidManifest.xml (created)
test12\android\app\src\main\AndroidManifest.xml (created)
test12\android\app\src\main\res\drawable\launch_background.xml (created)
test12\android\app\src\main\res\drawable-v21\launch_background.xml (created)
test12\android\app\src\main\res\mipmap-hdpi\ic_launcher.png (created)
test12\android\app\src\main\res\mipmap-mdpi\ic_launcher.png (created)
test12\android\app\src\main\res\mipmap-xhdpi\ic_launcher.png (created)
test12\android\app\src\main\res\mipmap-xxhdpi\ic_launcher.png (created)
test12\android\app\src\main\res\mipmap-xxxhdpi\ic_launcher.png (created)
test12\android\app\src\main\res\values\styles.xml (created)
test12\android\app\src\main\res\values-night\styles.xml (created)
test12\android\app\src\profile\AndroidManifest.xml (created)
test12\android\gradle\wrapper\gradle-wrapper.properties (created)

```

ภาพประกอบ 5.108 ตัวอย่างไอคอนที่ติดมากับการสร้างไฟล์งาน

ที่มา : ฌปภัช วรรณตรง (2564 : 4)

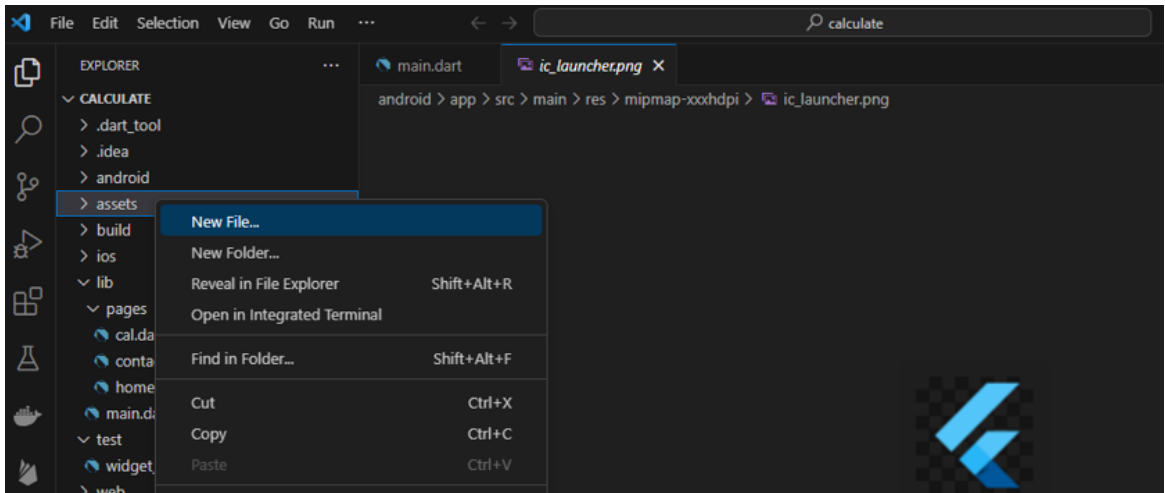


ภาพประกอบ 5.109 ตัวอย่างไอคอนที่ติดมากับการสร้างไฟล์งาน (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 5)

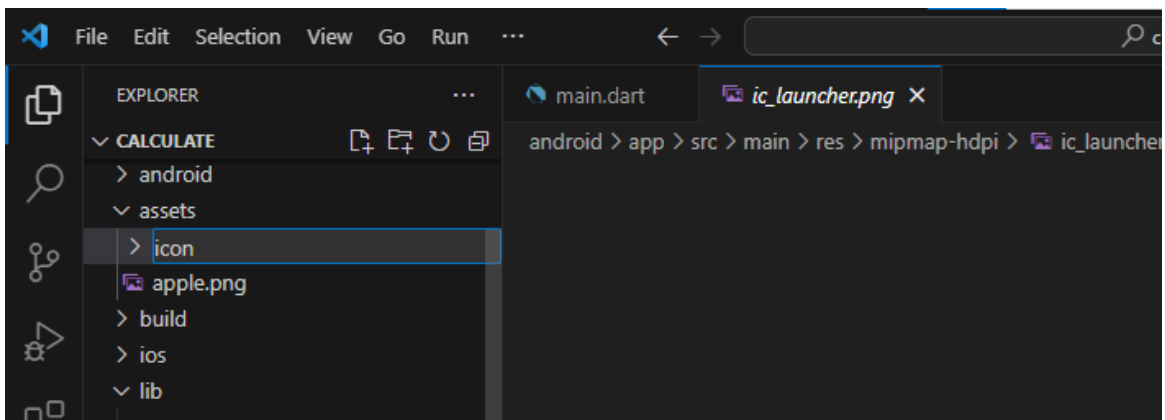
2. เริ่มทำการสร้างไอคอน ให้ทำการสร้างไฟล์ Folder icon สำหรับเก็บไอคอนไว้ใน Folder assets

ไปที่ asset จากนั้นให้ทำการสร้าง Folder icon



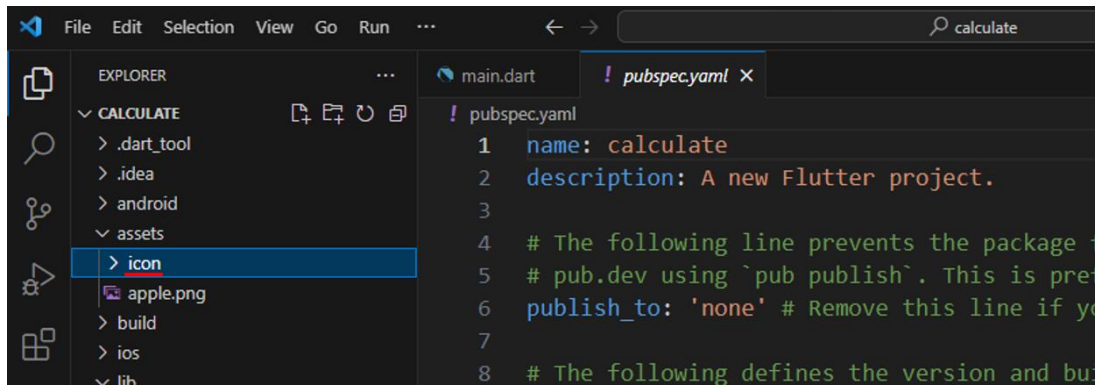
ภาพประกอบ 5.110 สร้าง Folder icon

ที่มา : ณปภัช วรรณตรง (2564 : 5)



ภาพประกอบ 5.111 สร้าง Folder icon (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 5)



ภาพประกอบ 5.112 สร้าง Folder icon (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 6)

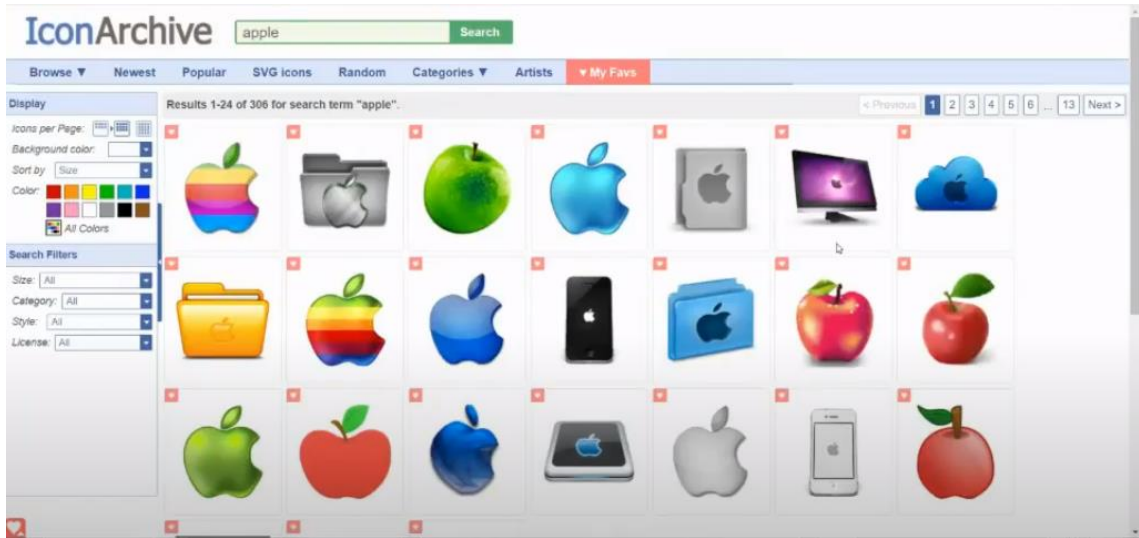
3. สามารถทำการค้นหาภาพไอคอนสำหรับนำมาใช้งานได้ฟรีจากเว็บไซต์ IconArchive



ภาพประกอบ 5.113 ค้นหาไอคอนสำหรับนำมาใช้งาน

ที่มา : ฅนปลัซ วรณตรง (2564 : 7)

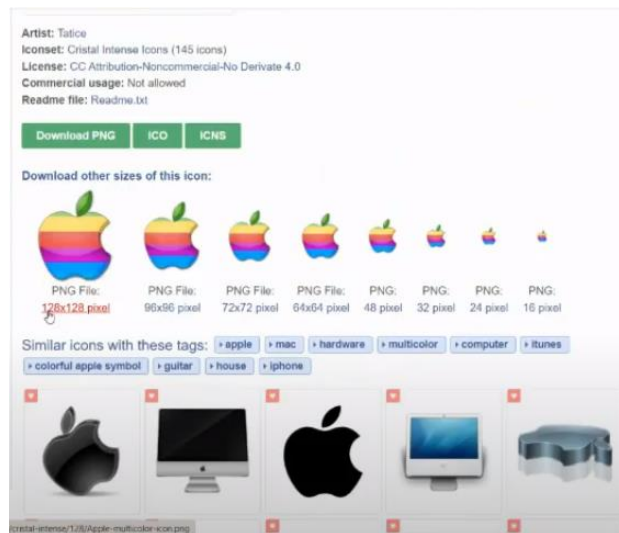




ภาพประกอบ 5.114 ค้นหาไอคอนสำหรับนำมาใช้งาน (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 7)

เมื่อได้ Icon ที่ต้องการแล้ว ให้ทำการเลือก Icon ที่มีขนาดใหญ่ที่สุด จากนั้นค่อยทำการดาวน์โหลด



ภาพประกอบ 5.114 ค้นหาไอคอนสำหรับนำมาใช้งาน (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 7)

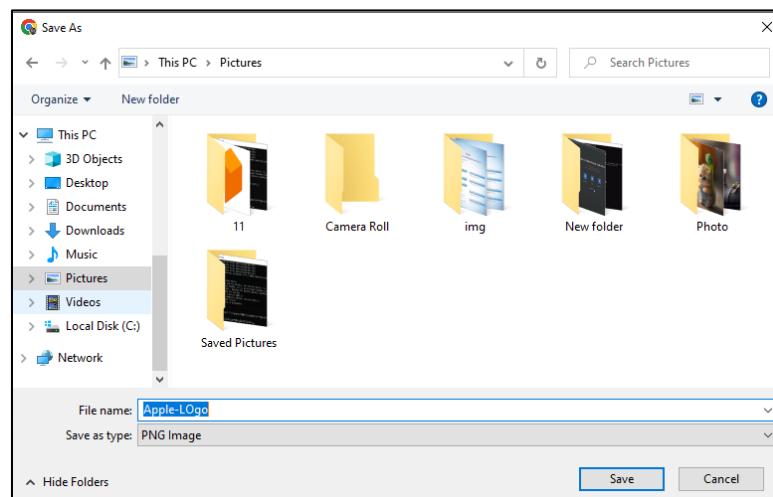
หลังจากเลือก Icon ได้แล้วให้ทำการดาวน์โหลดหรือบันทึกภาพ โดยการคลิกขวาที่ภาพ จากนั้นเลือกเป็น Save image as...



ภาพประกอบ 5.115 ค้นหาไอคอนสำหรับนำมาใช้งาน (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 7)

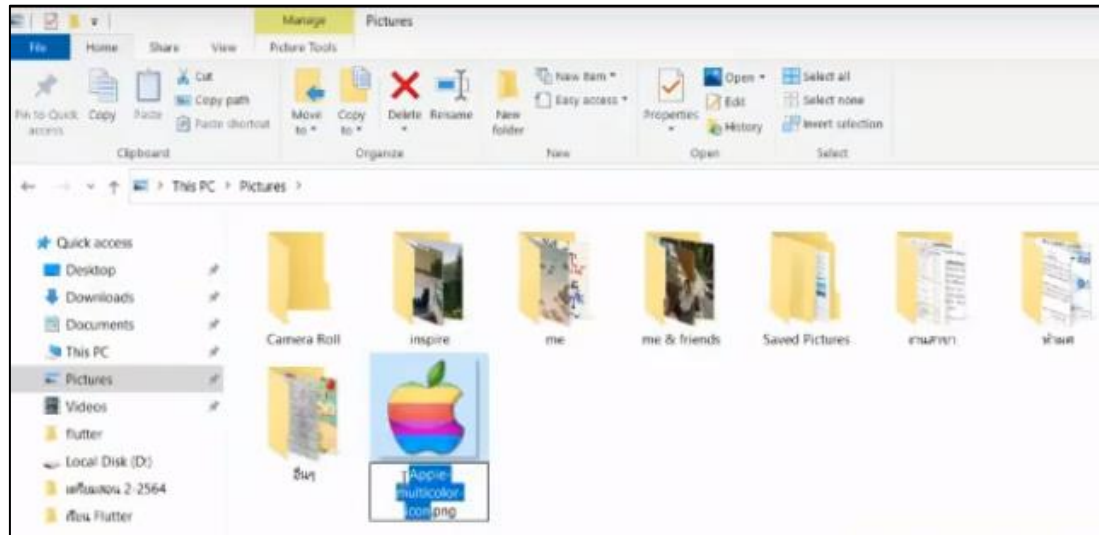
เลือกตำแหน่งสำหรับเก็บไฟล์รูปภาพ



ภาพประกอบ 5.116 ค้นหาไอคอนสำหรับนำมาใช้งาน (ต่อ)

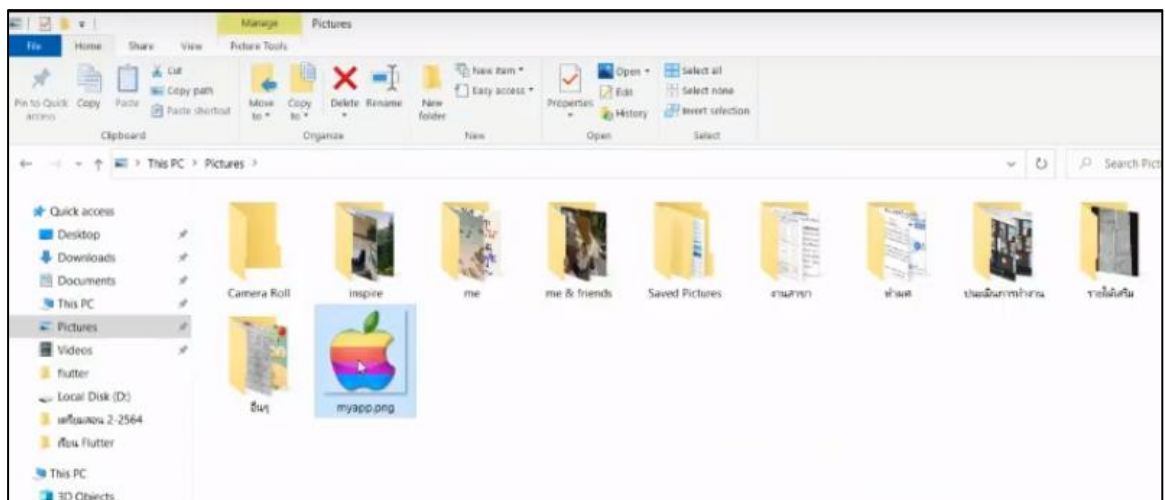
ที่มา : ฌปภัช วรรณตรง (2564 : 7)

ทำการเปลี่ยนชื่อให้สามารถเรียกใช้งานได้ง่าย



ภาพประกอบ 5.117 ค้นหาไอคอนสำหรับนำมาใช้งาน (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 7)

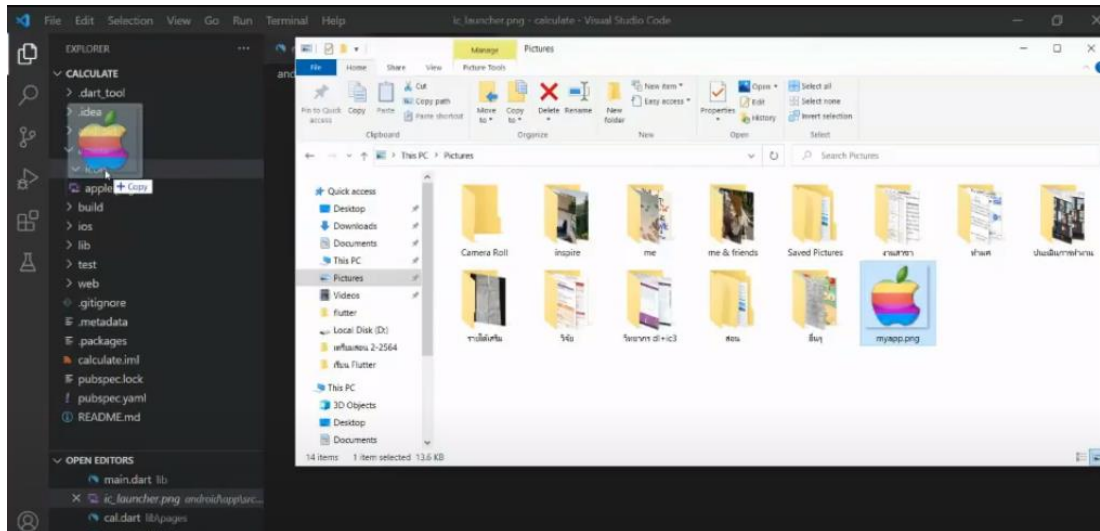


ภาพประกอบ 5.118 ค้นหาไอคอนสำหรับนำมาใช้งาน (ต่อ)

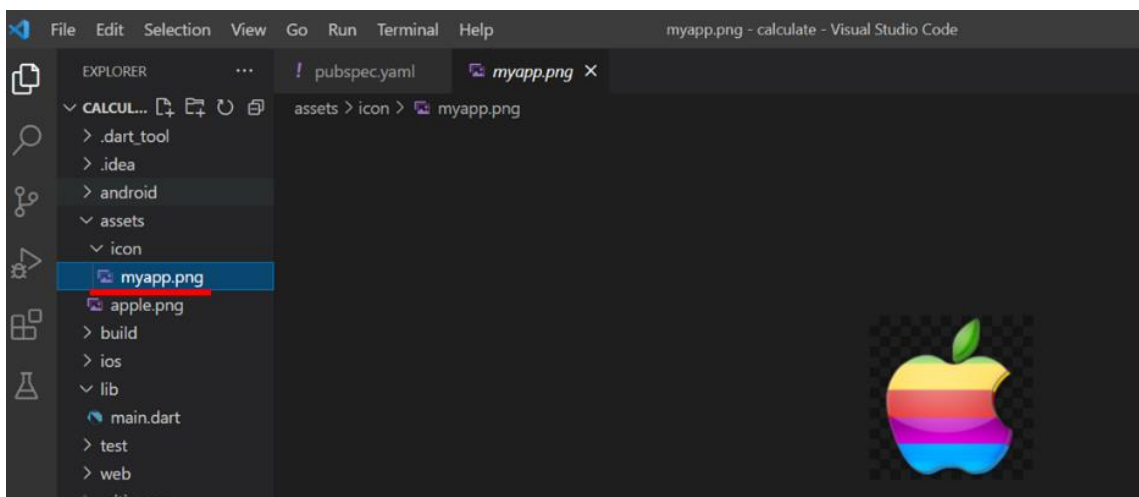
ที่มา : ฌปภัช วรรณตรง (2564 : 7)

4. เมื่อทำการเลือกไอคอนที่ต้องการได้แล้ว ให้ทำการดาวน์โหลดไอคอนดังกล่าวแล้วนำไปไว้ใน Folder icon

ให้นำไฟล์ภาพประกอบที่ได้ทำการดาวน์โหลดมาไปใส่ใน Folder icon ที่ได้ทำการสร้างไว้ โดยการคลิกลากไปวาง

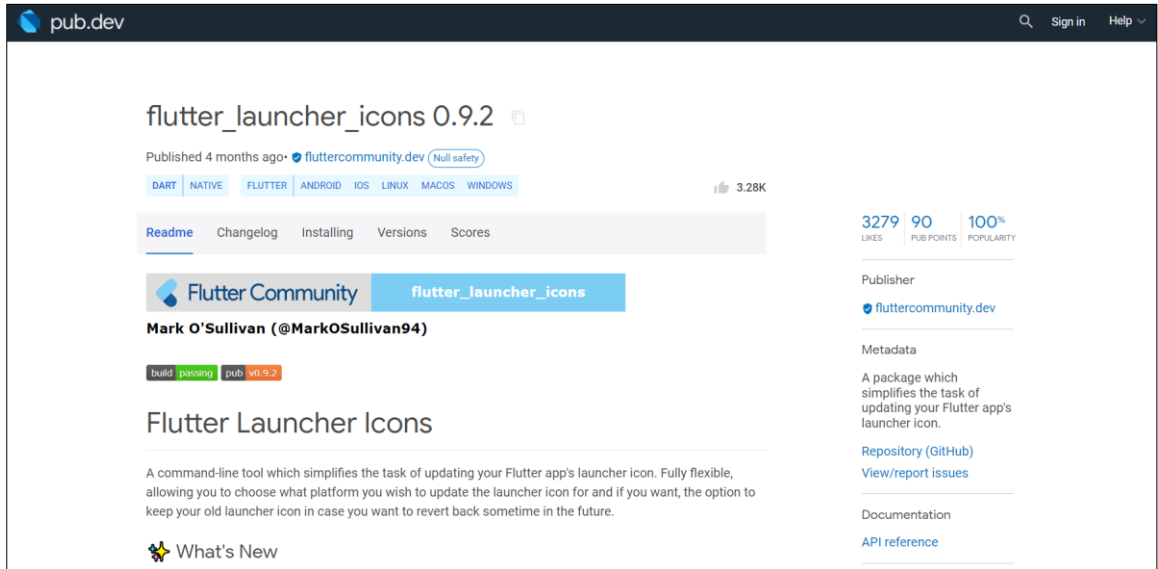


ภาพประกอบ 5.119 ดาวน์โหลดไอคอนที่ต้องการแล้วนำไปไว้ใน Folder icon  
ที่มา : ฌปภัช วรณตรง (2564 : 8)



ภาพประกอบ 5.120 ดาวน์โหลดไอคอนที่ต้องการแล้วนำไปไว้ใน Folder icon (ต่อ)  
ที่มา : ฌปภัช วรณตรง (2564 : 7)

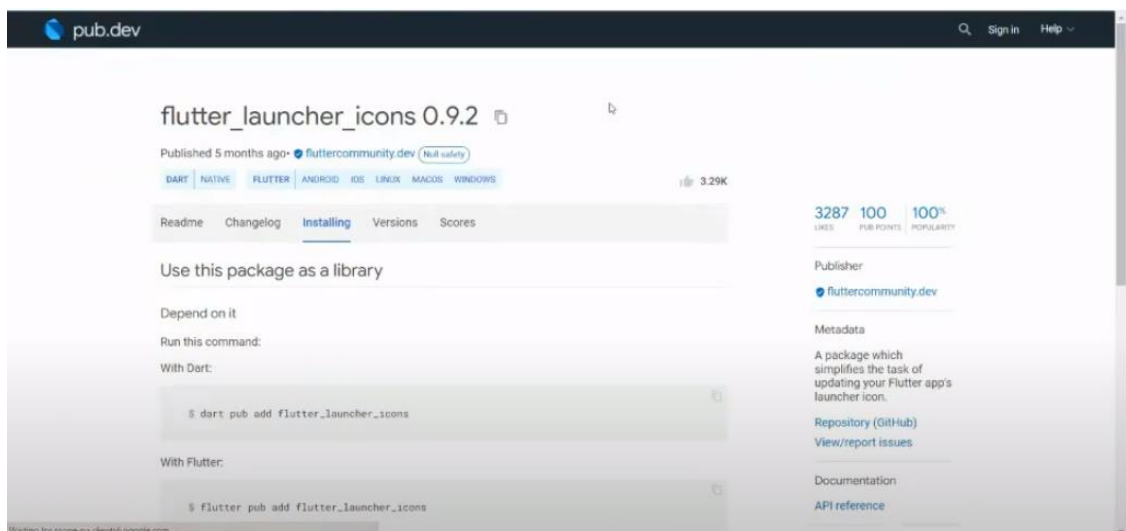
5. ให้ทำการค้นหา flutter launcher icon เพื่อนำโค้ดบางส่วนที่เกี่ยวข้องกับการเรียกใช้ไอคอนเข้ามาใช้งาน



ภาพประกอบ 5.121 ค้นหา flutter launcher icon

ที่มา : ฅนปลัซ วรณตรง (2564 : 9)

ไปที่ Installing เพื่อคัดลอกโค้ดสำหรับเปลี่ยน Icon



ภาพประกอบ 5.122 ค้นหา flutter launcher icon (ต่อ)

ที่มา : ฅนปลัซ วรณตรง (2564 : 9)

หมายเหตุ
----------

pub.dev คือ เว็บไซต์ที่รวม package เพิ่มเติมของ flutter
---

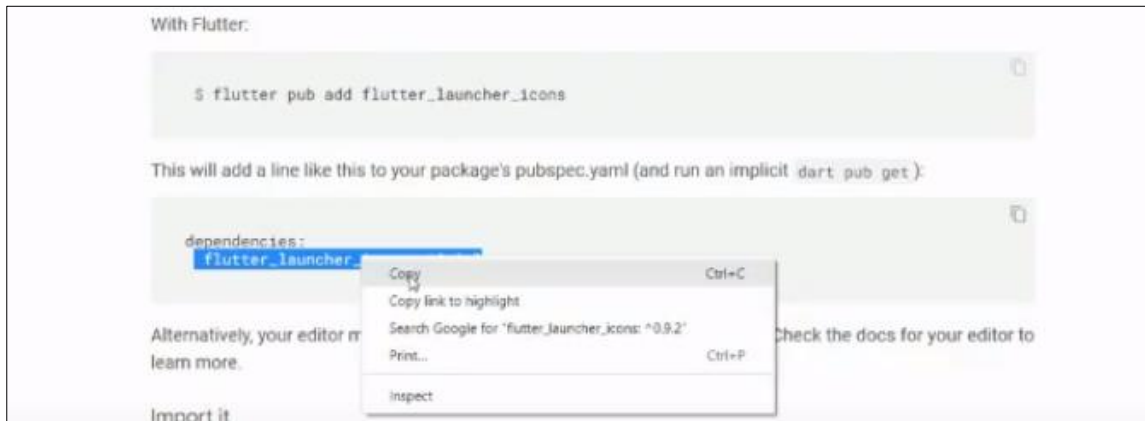
6. ในส่วนของ Installing ให้ทำการคัดลอกโค้ดใน dependencies:

This will add a line like this to your package's pubspec.yaml (and run an implicit `dart pub get`):

```
dependencies:
  flutter_launcher_icons: ^0.9.2
```

ภาพประกอบ 5.123 คัดลอกโค้ดใน dependencies:

ที่มา : ฌปภัช วรรณตรง (2564 : 10)

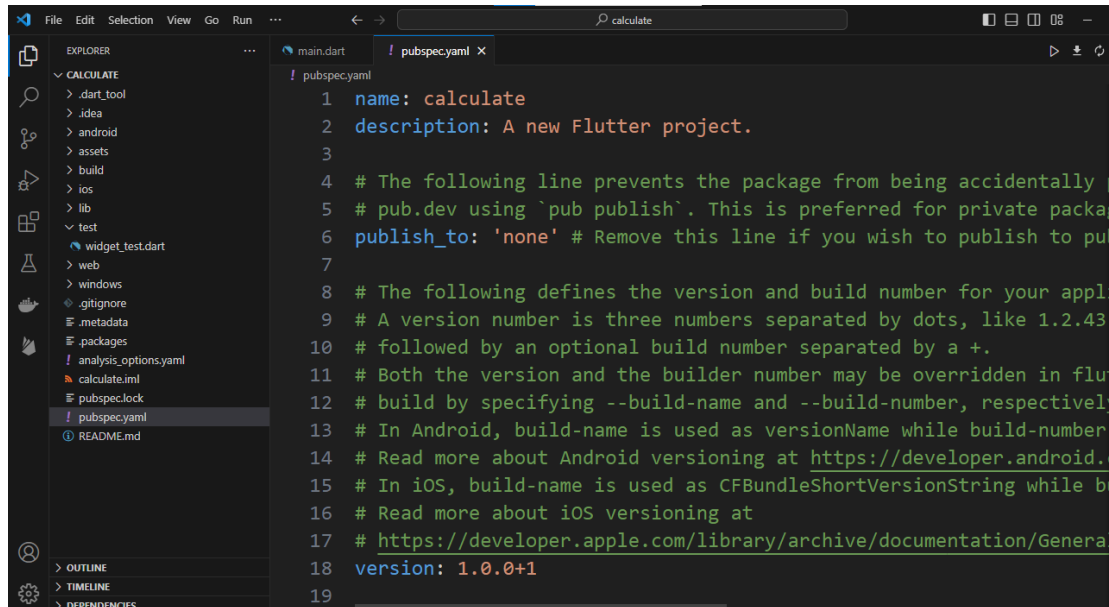


ภาพประกอบ 5.124 คัดลอกโค้ดใน dependencies: (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 10)

7. เปิดไฟล์ pubspec.yaml จากนั้นให้นำโค้ดที่คัดลอกมาจาก dependencies: มาวางในตำแหน่ง dependencies: เดียวกันในไฟล์ที่เปิดมาตามภาพ

### 7.1 ไปที่ไฟล์ pubspec.yaml



```

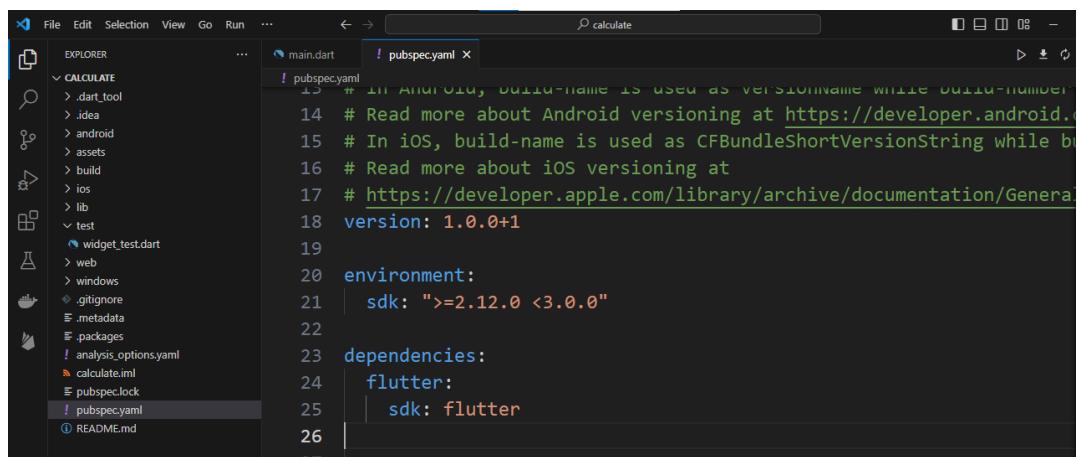
1 name: calculate
2 description: A new Flutter project.
3
4 # The following line prevents the package from being accidentally
5 # pub.dev using `pub publish`. This is preferred for private packa
6 publish_to: 'none' # Remove this line if you wish to publish to pu
7
8 # The following defines the version and build number for your appl
9 # A version number is three numbers separated by dots, like 1.2.43
10 # followed by an optional build number separated by a +.
11 # Both the version and the builder number may be overridden in flu
12 # build by specifying --build-name and --build-number, respectivel
13 # In Android, build-name is used as versionName while build-number
14 # Read more about Android versioning at https://developer.android.
15 # In iOS, build-name is used as CFBundleShortVersionString while b
16 # Read more about iOS versioning at
17 # https://developer.apple.com/library/archive/documentation/Genera
18 version: 1.0.0+1
19

```

ภาพประกอบ 5.125 นำโค้ดที่คัดลอกมา มาไว้ในไฟล์ pubspec.yaml

ที่มา : ฌปภัช วรรณตรง (2564 : 11)

### 7.2 ไปที่ dependencies: จากนั้นให้วางโค้ดต่อจาก sdk:



```

13 # In Android, build-name is used as versionName while build-number
14 # Read more about Android versioning at https://developer.android.
15 # In iOS, build-name is used as CFBundleShortVersionString while b
16 # Read more about iOS versioning at
17 # https://developer.apple.com/library/archive/documentation/Genera
18 version: 1.0.0+1
19
20 environment:
21 | sdk: ">=2.12.0 <3.0.0"
22
23 dependencies:
24 | flutter:
25 | | sdk: flutter
26
27

```

ภาพประกอบ 5.126 นำโค้ดที่คัดลอกมา มาไว้ในไฟล์ pubspec.yaml (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 11)

```

9 # The following defines the version and build number
10 # A version number is three numbers separated by dots
11 # followed by an optional build number separated by a
12 # Both the version and the builder number may be over
13 # build by specifying --build-name and --build-number
14 # In Android, build-name is used as versionName while
15 # Read more about Android versioning at https://deve
16 # In iOS, build-name is used as CFBundleShortVersionStr
17 # Read more about iOS versioning at
18 # https://developer.apple.com/library/archive/document
19 version: 1.0.0+1
20
21 environment:
22   sdk: ">=2.12.0 <3.0.0"
23
24 dependencies:
25   flutter:
26     sdk: flutter
27   flutter_launcher_icons: ^0.9.2

```

ภาพประกอบ 5.127 นำโค้ดที่คัดลอกมา มาไว้ในไฟล์ pubspec.yaml (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 11)

```

14 # Read more about Android versioning at https://develope
15 # In iOS, build-name is used as CFBundleShortVersionStr
16 # Read more about iOS versioning at
17 # https://developer.apple.com/library/archive/documentat
18 version: 1.0.0+1
19
20 environment:
21   sdk: ">=2.12.0 <3.0.0"
22
23 dependencies:
24   flutter:
25     sdk: flutter
26   flutter_launcher_icons: ^0.9.2
27

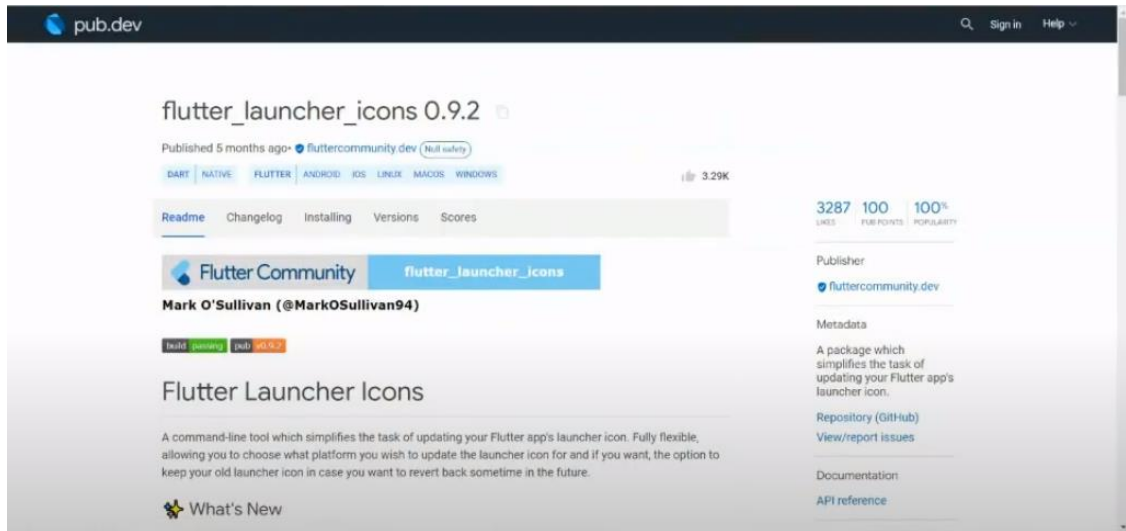
```

ภาพประกอบ 5.128 นำโค้ดที่คัดลอกมา มาไว้ในไฟล์ pubspec.yaml (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 11)



8. จากนั้นให้ไปที่ flutter\_icons: ให้ทำการคัดลอกมาทั้งหมดของส่วน flutter\_icons:  
8.1 ให้ทำการกลับไปหน้า Read me จากหน้า Installing



ภาพประกอบ 5.129 คัดลอกทั้งหมดของส่วน flutter\_icons:  
ที่มา : ณปภัช วรรณตรง (2564 : 12)

## 8.2 เลื่อนหา Setup the config file



ภาพประกอบ 5.130 คัดลอกทั้งหมดของส่วน flutter\_icons: (ต่อ)  
ที่มา : ณปภัช วรรณตรง (2564 : 12)

### 8.3 จากนั้นให้ทำการคัดลอกโค้ด flutter\_icon: ทั้งหมดไว้

:book: Guide

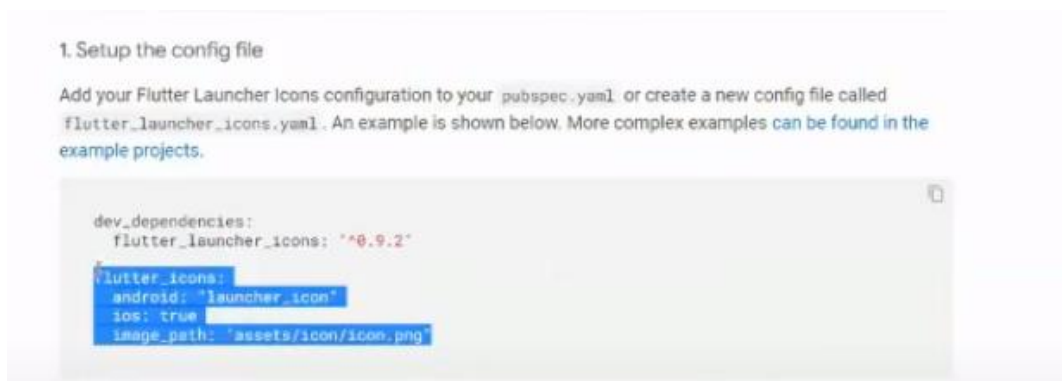
#### 1. Setup the config file

Add your Flutter Launcher Icons configuration to your `pubspec.yaml` or create a new config file called `flutter_launcher_icons.yaml`. An example is shown below. More complex examples [can be found in the example projects](#).

```
dev_dependencies:
  flutter_launcher_icons: "^0.9.2"

flutter_icons:
  android: "launcher_icon"
  ios: true
  image_path: "assets/icon/icon.png"
```

ภาพประกอบ 5.131 คัดลอกทั้งหมดของส่วน flutter\_icons: (ต่อ)  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 12)



ภาพประกอบ 5.132 คัดลอกทั้งหมดของส่วน flutter\_icons: (ต่อ)  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 12)

10. เปิดไฟล์ `pubspec.yaml` จากนั้นให้นำโค้ดที่คัดลอกมาจาก flutter\_icons: มาวางในไฟล์ที่เปิดมา

จากนั้นไปที่ไฟล์ `pubspec.yaml` เลื่อนลงมาข้างล่าง แล้วทำการวางโค้ดที่ได้คัดลอกไว้ จัดย่อหน้าให้ตรงตามภาพถัดไป (เนื่องจากการย่อหน้ามีผลต่อการทำงานของโค้ดไฟล์นี้)

```

61 # family key with the font family name, and a fonts key with a
62 # list giving the asset and other descriptors for the font. For
63 # example:
64 # fonts:
65 #   - family: Schyler
66 #     fonts:
67 #       - asset: fonts/Schyler-Regular.ttf
68 #       - asset: fonts/Schyler-Italic.ttf
69 #         style: italic
70 #   - family: Trajan Pro
71 #     fonts:
72 #       - asset: fonts/TrajanPro.ttf
73 #       - asset: fonts/TrajanPro_Bold.ttf
74 #         weight: 700
75 #
76 # For details regarding fonts from package dependencies,
77 # see https://flutter.dev/custom-fonts/#from-packages
78 #
79
80

```

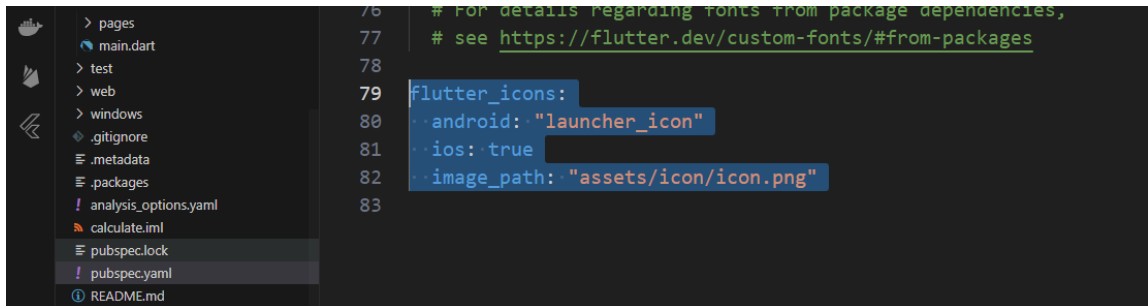
ภาพประกอบ 5.133 นำโค้ดที่คัดลอกมาจาก flutter\_icons: มาวาง  
 ที่มา : ฅนปักษ์ วรรณตรง (2564 : 13)

```

68 #   - asset: fonts/Schyler-Italic.ttf
69 #     style: italic
70 #   - family: Trajan Pro
71 #     fonts:
72 #       - asset: fonts/TrajanPro.ttf
73 #       - asset: fonts/TrajanPro_Bold.ttf
74 #         weight: 700
75 #
76 # For details regarding fonts from package dependencies,
77 # see https://flutter.dev/custom-fonts/#from-packages
78 #
79 flutter_icons:
80   android: "launcher_icon"
81   ios: true
82   image_path: "assets/icon/icon.png"
83

```

ภาพประกอบ 5.134 นำโค้ดที่คัดลอกมาจาก flutter\_icons: มาวาง (ต่อ)  
 ที่มา : ฅนปักษ์ วรรณตรง (2564 : 13)



```

76 # For details regarding fonts from package dependencies,
77 # see https://flutter.dev/custom-fonts/#from-packages
78
79 flutter_icons:
80   android: "launcher_icon"
81   ios: true
82   image_path: "assets/icon/icon.png"
83

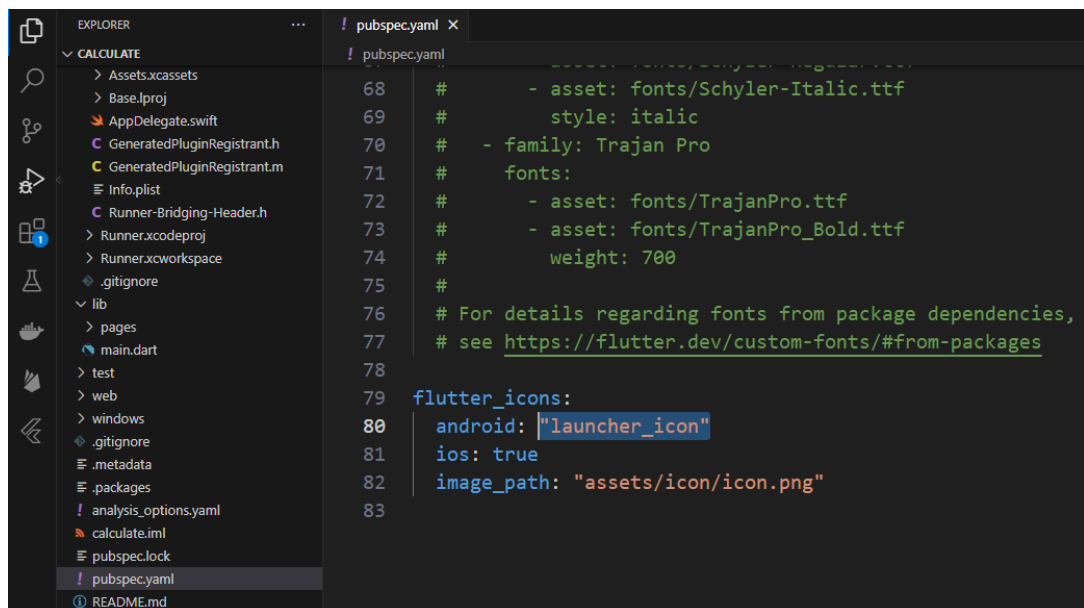
```

ภาพประกอบ 5.135 นำโค้ดที่คัดลอกมาจาก flutter\_icons: มาวาง (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 13)

11. ในไฟล์ pubspec.yaml ที่ได้ทำการคัดลอกโค้ดส่วน flutter\_icons: มาใส่ ให้ทำการแก้ไขโค้ด tag android: เป็น true และแก้ไขชื่อ icon.png ในส่วนของ tag image\_path: เป็นชื่อไอคอนที่ได้ทำการดาวน์โหลดมา

เปลี่ยนโค้ด flutter\_icons: android: จากข้อความให้เป็น true



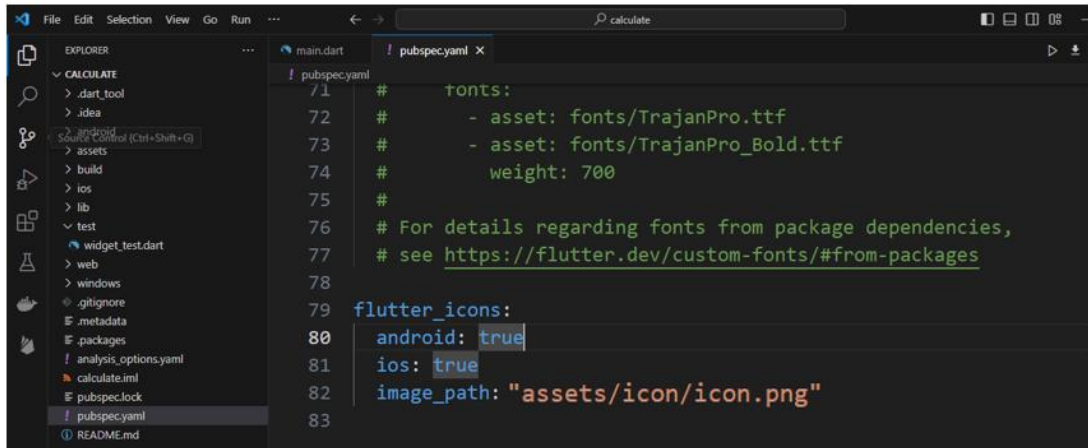
```

68 #
69 # - asset: fonts/Schlyer-Italic.ttf
70 #   style: italic
71 # - family: Trajan Pro
72 # fonts:
73 #   - asset: fonts/TrajanPro.ttf
74 #   - asset: fonts/TrajanPro_Bold.ttf
75 #     weight: 700
76 # For details regarding fonts from package dependencies,
77 # see https://flutter.dev/custom-fonts/#from-packages
78
79 flutter_icons:
80   android: "launcher_icon"
81   ios: true
82   image_path: "assets/icon/icon.png"
83

```

ภาพประกอบ 5.136 แก้ไขโค้ดใส่ส่วน flutter\_icons:

ที่มา : ฌปภัช วรรณตรง (2564 : 14)



```

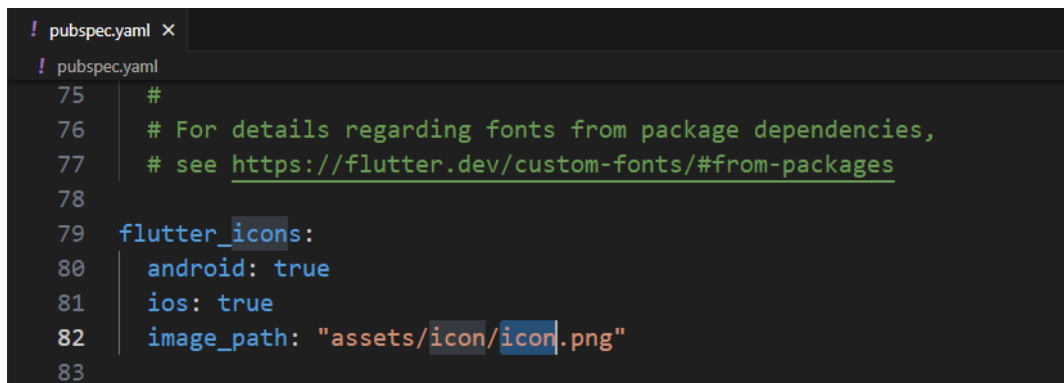
71 fonts:
72 #   - asset: fonts/TrajanPro.ttf
73 #   - asset: fonts/TrajanPro_Bold.ttf
74 #     weight: 700
75 #
76 # For details regarding fonts from package dependencies,
77 # see https://flutter.dev/custom-fonts/#from-packages
78
79 flutter_icons:
80   android: true
81   ios: true
82   image_path: "assets/icon/icon.png"
83

```

ภาพประกอบ 5.137 แก้ไขโค้ดใส่ส่วน flutter\_icons: (ต่อ)

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 14)

จากนั้นให้ทำการเปลี่ยนชื่อไฟล์รูปภาพสำหรับ Icon



```

75 #
76 # For details regarding fonts from package dependencies,
77 # see https://flutter.dev/custom-fonts/#from-packages
78
79 flutter_icons:
80   android: true
81   ios: true
82   image_path: "assets/icon/icon.png"
83

```

ภาพประกอบ 5.138 แก้ไขโค้ดใส่ส่วน flutter\_icons: (ต่อ)

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 14)

```
! pubspec.yaml x
! pubspec.yaml
75 #
76 # For details regarding fonts from package dependencies,
77 # see https://flutter.dev/custom-fonts/#from-packages
78
79 flutter_icons:
80   android: true
81   ios: true
82   image_path: "assets/icon/myapp.png"
83
```

ภาพประกอบ 5.139 แก้ไขโค้ดในส่วน flutter\_icons: (ต่อ)

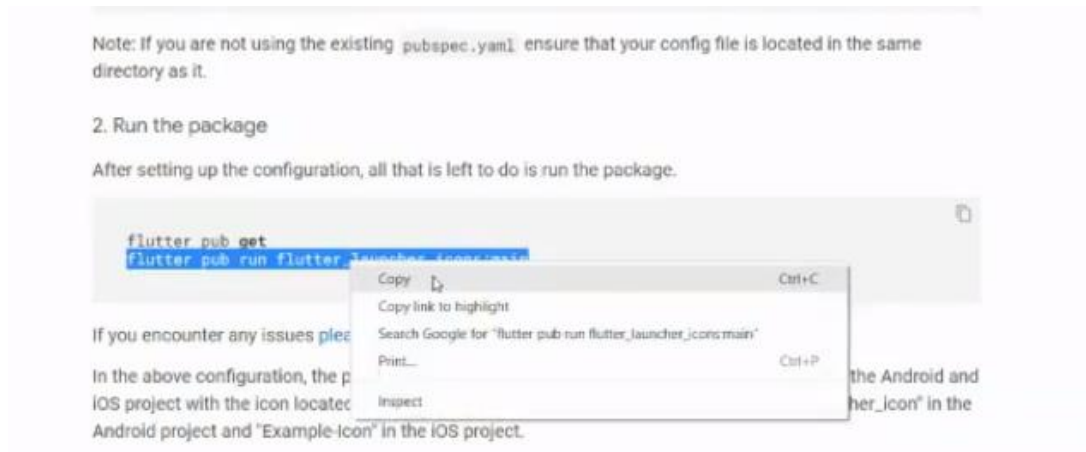
ที่มา : ณปภัช วรรณตรง (2564 : 14)

```
main.dart x ! pubspec.yaml x
! pubspec.yaml
75 #
76 # For details regarding fonts from package dependencies,
77 # see https://flutter.dev/custom-fonts/#from-packages
78
79 flutter_icons:
80   android: true
81   ios: true
82   image_path: "assets/icon/myapp.png"
83
```

ภาพประกอบ 5.140 แก้ไขโค้ดในส่วน flutter\_icons: (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 14)

12. ให้กลับไป flutter launcher icon แล้วไปทำการคัดลอกโค้ด flutter pub run ไปที่หน้า Read me จากนั้นเลือกไปที่ Run the package และทำการคัดลอก flutter pub run flutter\_launcher\_icons:main



ภาพประกอบ 5.141 คัดลอกโค้ด flutter pub run

ที่มา : ฅนปักษ์ วรณตรง (2564 : 15)

## 2. Run the package

After setting up the configuration, all that is left to do is run the package.

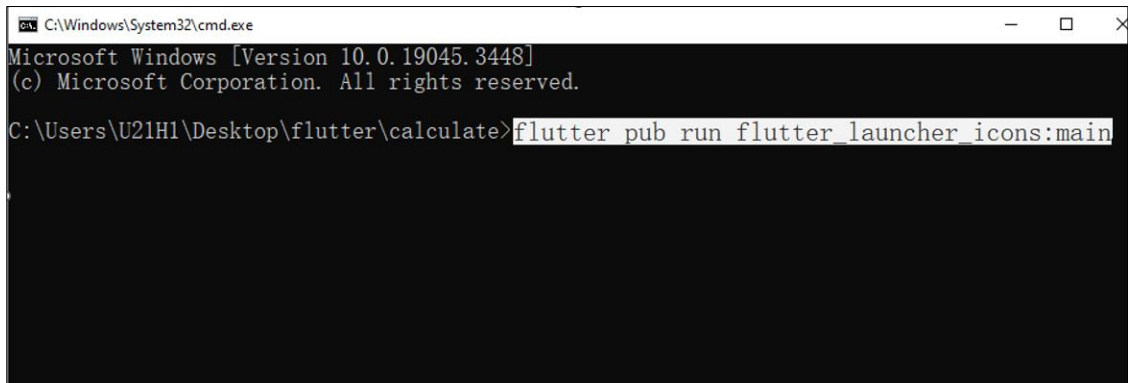
```
flutter pub get
flutter pub run flutter_launcher_icons:main
```

ภาพประกอบ 5.142 คัดลอกโค้ด flutter pub run (ต่อ)

ที่มา : ฅนปักษ์ วรณตรง (2564 : 15)

13. จากนั้นให้ทำการเปิดหน้าต่าง cmd แล้วทำการ `cd` เข้าไปในโฟลเดอร์โปรเจกต์ จากนั้นให้ทำการคลิกขวาที่หน้าต่าง cmd หรือ `Ctrl + v` เพื่อทำการวางโค้ด `flutter pub run` ที่ได้ทำการคัดลอกมาจากนั้นกด `Enter` แล้วจะได้ผลลัพธ์

เปิดหน้าต่าง cmd ของ Folder calculate จากนั้นให้ทำการวางโค้ดที่ได้คัดลอกมา แล้วกด Enter



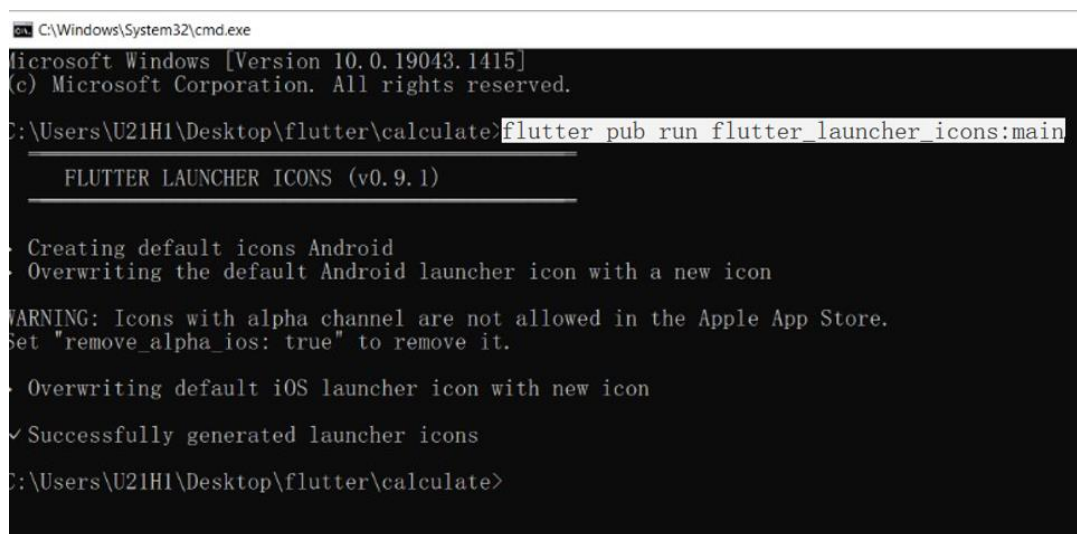
```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

C:\Users\U21H1\Desktop\flutter\calculate>flutter pub run flutter_launcher_icons:main

```

ภาพประกอบ 5.143 วางโค้ด flutter pub run ในหน้าต่าง cmd  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 16)



```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Users\U21H1\Desktop\flutter\calculate>flutter pub run flutter_launcher_icons:main

FLUTTER LAUNCHER ICONS (v0.9.1)

- Creating default icons Android
- Overwriting the default Android launcher icon with a new icon

WARNING: Icons with alpha channel are not allowed in the Apple App Store.
set "remove_alpha_ios: true" to remove it.

- Overwriting default iOS launcher icon with new icon
✓ Successfully generated launcher icons

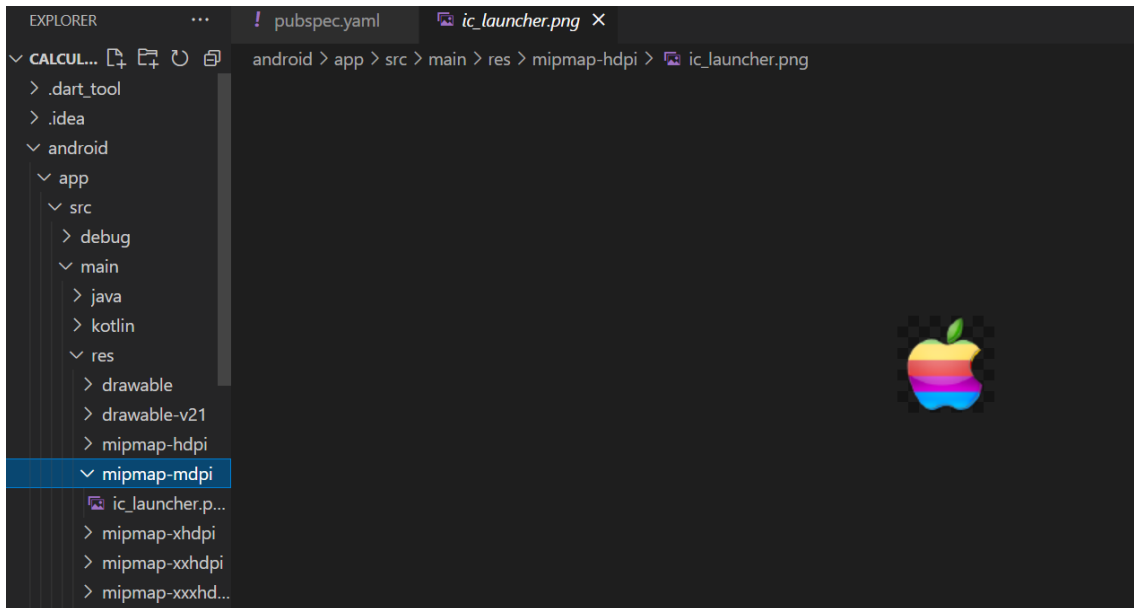
C:\Users\U21H1\Desktop\flutter\calculate>

```

ภาพประกอบ 5.144 วางโค้ด flutter run ในหน้าต่าง cmd (ต่อ)  
ที่มา : ฅนปักษ์ วรณตรง (2564 : 16)

14. ให้ทำการเปิดไฟล์งาน และเข้าไปที่ Folder res จะพบว่าไอคอนที่ Flutter สร้างมาได้ถูกเปลี่ยนเป็นไอคอนที่ได้ทำการเปลี่ยนแปลงเรียบร้อยแล้ว



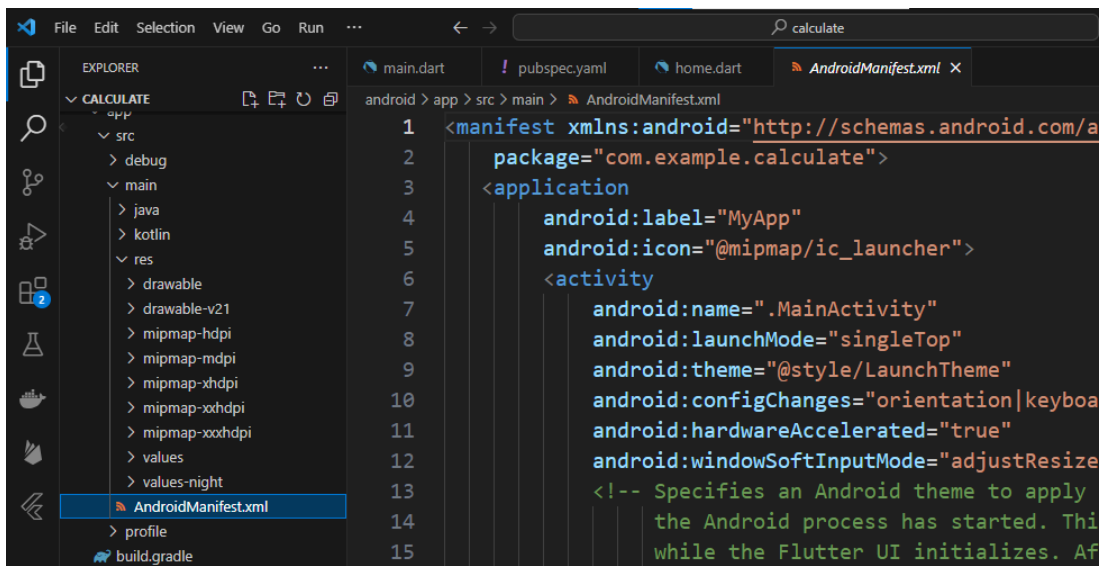


ภาพประกอบ 5.145 ไอคอนได้ทำการเปลี่ยนแปลงเรียบร้อยแล้ว

ที่มา : ฅปภัช วรรณตรง (2564 : 17)

15. จากนั้นให้เปิดไฟล์ AndroidManifest.xml ไปที่ tag android:label = “ ” ให้ทำการเปลี่ยนชื่อใน “ ” เป็นชื่อไอคอนที่ได้ทำการดาวน์โหลดมา

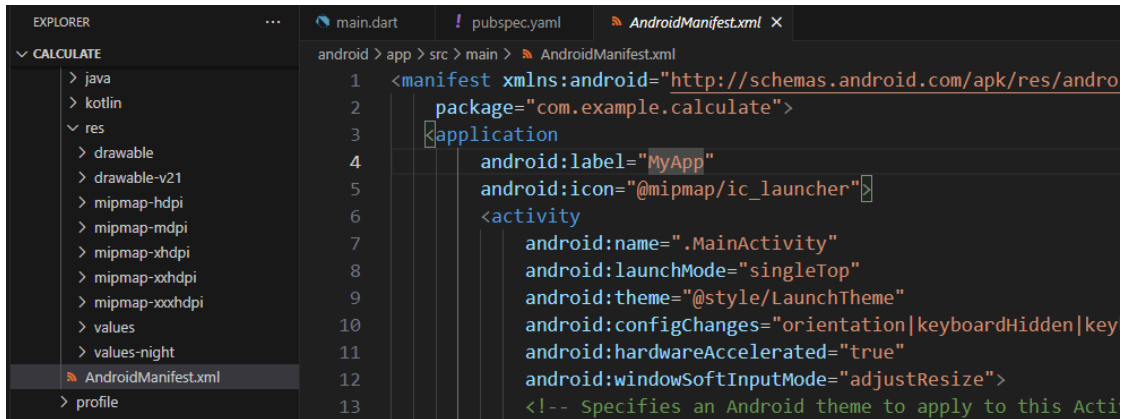
ไปที่ไฟล์ AndroidManifest.xml ใน Folder src



ภาพประกอบ 5.146 เปลี่ยนชื่อไอคอนใน tag android:label

ที่มา : ฅปภัช วรรณตรง (2564 : 18)

ทำการเปลี่ยนแอปที่ android:label ในเครื่องหมาย “ ”



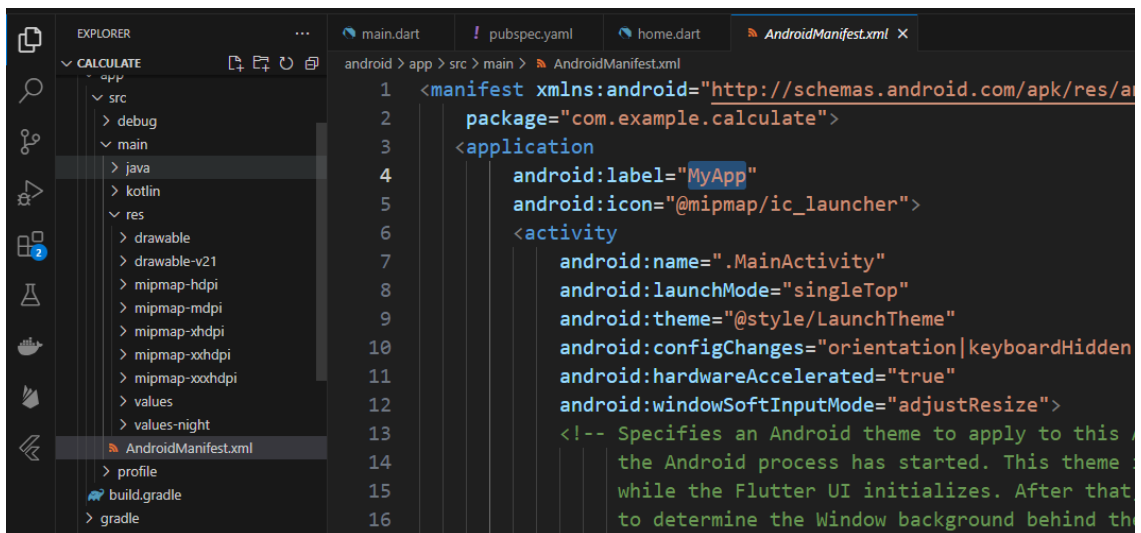
```

EXPLORER
main.dart pubspec.yaml AndroidManifest.xml X
android > app > src > main > AndroidManifest.xml
1 <manifest xmlns:android="http://schemas.android.com/apk/res/andro
2 package="com.example.calculate">
3 <application
4 android:label="MyApp"
5 android:icon="@mipmap/ic_launcher">
6 <activity
7 android:name=".MainActivity"
8 android:launchMode="singleTop"
9 android:theme="@style/LaunchTheme"
10 android:configChanges="orientation|keyboardHidden|key
11 android:hardwareAccelerated="true"
12 android:windowSoftInputMode="adjustResize">
13 <!-- Specifies an Android theme to apply to this Acti

```

ภาพประกอบ 5.147 เปลี่ยนชื่อไอคอนใน tag android:label (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 18)



```

EXPLORER
main.dart pubspec.yaml home.dart AndroidManifest.xml X
android > app > src > main > AndroidManifest.xml
1 <manifest xmlns:android="http://schemas.android.com/apk/res/a
2 package="com.example.calculate">
3 <application
4 android:label="MyApp"
5 android:icon="@mipmap/ic_launcher">
6 <activity
7 android:name=".MainActivity"
8 android:launchMode="singleTop"
9 android:theme="@style/LaunchTheme"
10 android:configChanges="orientation|keyboardHidden
11 android:hardwareAccelerated="true"
12 android:windowSoftInputMode="adjustResize">
13 <!-- Specifies an Android theme to apply to this
14 the Android process has started. This theme
15 while the Flutter UI initializes. After that
16 to determine the Window background behind th

```

ภาพประกอบ 5.148 เปลี่ยนชื่อไอคอนใน tag android:label (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 18)

## ทำการสร้างไฟล์สำหรับนำไปติดตั้งและใช้งานบนมือถือไฟล์ .apk

ไฟล์ .apk เป็นแพ็คเกจ Android คล้ายกับไฟล์ .exe บน Windows หรือไฟล์ .dmg บน macOS แพ็คเกจนี้ประกอบด้วยไฟล์ ทรัพยากร และข้อมูลเมตา (Metadata) ทั้งหมดที่จำเป็นสำหรับแอปพลิเคชันบนมือถือ

ในการทำงานบนอุปกรณ์ Android นักพัฒนาใช้ตัวติดตั้งแอปเพื่อติดตั้งแอปหลังจากที่ผู้ใช้ดาวน์โหลดไฟล์ .apk ที่เกี่ยวข้องลงในอุปกรณ์ของตน แพ็คเกจจะจัดเก็บข้อมูลเมตาที่จำเป็น เช่น ชื่อแอปฯ หมายเลขเวอร์ชัน และรายการสิทธิ์ ซึ่งระบบปฏิบัติการใช้ในการจัดการการควบคุมการเข้าถึง นอกจากนี้ ไฟล์ .apk ยังให้ความสะดวกในระหว่างขั้นตอนการทดสอบและแก้ไขจุดบกพร่องของการพัฒนา ตลอดจนเมื่อเผยแพร่แอปผ่านช่องทางสาธารณะ เช่น Google Play Store หรือร้านค้าบุคคลที่สาม (file.org, 2566 : 1)

ในขั้นตอนนี้จะทำการนำไฟล์โปรเจกต์ Flutter แปลงให้อยู่รูปแบบไฟล์ .apk เพื่อนำไปใช้ในการติดตั้งบนระบบปฏิบัติการ Android มีรายละเอียดขั้นตอนการทำดังนี้

1. ให้ทำการเปิดหน้าต่าง cmd แล้ว cd เข้าไปยังไฟล์งานเช่นขั้นตอนก่อนหน้านี้ จากนั้นให้ใช้คำสั่ง flutter build apk เพื่อทำการสร้างไฟล์ flutter สำหรับติดตั้งและใช้งานบนระบบปฏิบัติการแอนดรอยด์

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Users\U21H1\Desktop\flutter\calculate>flutter pub run flutter_launcher_icons:main

FLUTTER LAUNCHER ICONS (v0.9.1)

- Creating default icons Android
- Overwriting the default Android launcher icon with a new icon

WARNING: Icons with alpha channel are not allowed in the Apple App Store.
Set "remove_alpha_ios: true" to remove it.

- Overwriting default iOS launcher icon with new icon
✓ Successfully generated launcher icons

C:\Users\U21H1\Desktop\flutter\calculate>flutter build apk
```

ภาพประกอบ 5.149 ใช้คำสั่ง flutter build apk

ที่มา : ฌปภัช วรรณตรง (2564 : 20)

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Users\U21H1\Desktop\flutter\calculate> flutter build apk

Building with sound null safety

Running Gradle task 'assembleRelease'... 125.8s
√ Built build\app\outputs\flutter-apk\app-release.apk (23.2MB).

```

ภาพประกอบ 5.150 ใช้คำสั่ง flutter build apk (ต่อ)

ที่มา : ฌปภัช วรรณตรง (2564 : 20)

2. หลังจาก flutter ทำการสร้างไฟล์ .apk เสร็จเรียบร้อยแล้ว ให้ทำการคัดลอก Path Built build\... ไปต่อกับ Path Project ใน Folder ก็จะเจอไฟล์ที่ได้ทำการสร้างมา

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Users\U21H1\Desktop\flutter\calculate> flutter build apk

Building with sound null safety

Running Gradle task 'assembleRelease'... 125.8s
√ Built build\app\outputs\flutter-apk\app-release.apk (23.2MB).

```

ภาพประกอบ 5.151 คัดลอก Path ที่ตัวหนังสือสีเขียว

ที่มา : ฌปภัช วรรณตรง (2564 : 21)

PC > Desktop > flutter > calculate > build > app > outputs > flutter-apk

Name	Date modified	Type	Size
app.apk	19/1/2565 21:31	APK File	23,723 KB
app.apk.sha1	19/1/2565 21:31	SHA1 File	1 KB
app-debug.apk	19/1/2565 20:34	APK File	47,293 KB
app-release.apk	19/1/2565 21:31	APK File	23,723 KB

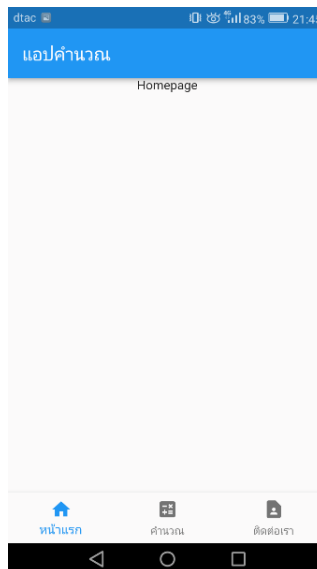
ภาพประกอบ 5.152 นำ Path มาวางต่อ Path Project เพื่อค้นหาไฟล์ที่ได้ทำการสร้างมา

ที่มา : ฌปภัช วรรณตรง (2564 : 21)

3. จากนั้นให้นำไฟล์ไปติดตั้งที่โทรศัพท์มือถือ เมื่อติดตั้งเสร็จเรียบร้อย และลองเข้าทดสอบแอปพลิเคชัน จะได้ผลลัพธ์ตามภาพ



ภาพประกอบ 5.153 นำไฟล์ที่สร้างเสร็จแล้วไปติดตั้งที่มือถือ  
ที่มา : ฌปภัช วรรณตรง (2564 : 22)



ภาพประกอบ 5.154 เข้าทดสอบแอปพลิเคชัน  
ที่มา : ฌปภัช วรรณตรง (2564 : 22)

4. หลังจากรอโหลดหน้าแอปพลิเคชันแล้ว จะได้ผลลัพธ์การพัฒนาตามภาพ



ภาพประกอบ 5.155 ผลลัพธ์การพัฒนาแอปพลิเคชัน

ที่มา : ฅนปลัซ วรณตรง (2564 : 22)

หากเกิดข้อผิดพลาด

สามารถนำโค้ดที่ผิดพลาดไปค้นหาใน StackOverFlow เพื่อหาคำตอบหรือการแก้ไขเบื้องต้นได้

## บทสรุป

สำหรับเนื้อหาที่กล่าวมาทั้งหมดในบทนี้ จะพูดถึงวิธีการสร้างหน้าแอปพลิเคชันแยกแต่ละหน้า จากนั้นเรียนรู้วิธีการสร้างแท็บบาร์ที่ด้านล่างของแอปพลิเคชัน ซึ่งเป็นส่วนเมื่อคลิกที่แท็บบาร์แล้ว จะไปยังหน้าจอที่เตรียมไว้ในขั้นตอนแรก เรียนรู้การสร้างไอคอนของแอปพลิเคชัน เพื่อเมื่อนำไปติดตั้งบนอุปกรณ์เคลื่อนที่จะแสดงเป็นไอคอนให้ผู้ใช้คลิกเพื่อเข้าสู่หน้าจอแอปพลิเคชัน และเรียนรู้การสร้างไฟล์แอปพลิเคชันเป็นไฟล์สำหรับติดตั้งและใช้งานบนระบบปฏิบัติการแอนดรอยด์ โดยจะเป็นไฟล์นามสกุล .apk ที่สามารถนำไปติดตั้งบนอุปกรณ์เคลื่อนที่ เพื่อให้สามารถเรียกใช้งานแอปพลิเคชันจากที่ได้ก็ไว้

### เอกสารอ้างอิง

จิราวุธ วารินทร์. (2564). **พัฒนาโมบายล์แอปด้วย Flutter + Dart**. กรุงเทพฯ : สำนักพิมพ์ชิมพลีฟาย.

ปัญญา ปะลีละเตสัง. (2566). **พัฒนาแอปแบบ Multi-Platform ด้วย Flutter โดยใช้ภาษา Dart**.

กรุงเทพฯ : ซีเอ็ดดูเคชั่น.

file.org. (16 กุมภาพันธ์ 2566). **APK File**. สืบค้นจาก, <https://file.org/extension/apk>

## บทที่ 6

# การสร้างฐานข้อมูลสำหรับการใช้งานด้วย Python Django

ในบทก่อนหน้าผู้เรียนได้เรียนรู้ถึงวิธีการสร้างหน้าแอปพลิเคชันแยกแต่ละหน้า จากนั้นเรียนรู้วิธีการสร้างแท็บบาร์ที่ด้านล่างของแอปพลิเคชัน ซึ่งเป็นส่วนเมื่อคลิกที่แท็บบาร์แล้ว จะไปยังหน้าจอที่เตรียมไว้ในขั้นตอนแรก เรียนรู้การสร้างไอคอนของแอปพลิเคชัน เพื่อเมื่อนำไปติดตั้งบนอุปกรณ์เคลื่อนที่จะแสดงเป็นไอคอนให้ผู้ใช้คลิกเพื่อเข้าสู่หน้าจอแอปพลิเคชัน และเรียนรู้การสร้างไฟล์แอปพลิเคชันเป็นไฟล์สำหรับติดตั้งและใช้งานบนระบบปฏิบัติการแอนดรอยด์ โดยจะเป็นไฟล์นามสกุล .apk ที่สามารถนำไปติดตั้งบนอุปกรณ์เคลื่อนที่ที่สามารถใช้งานแอปพลิเคชันได้ ซึ่งแอปพลิเคชันที่ได้ยังไม่สามารถเชื่อมต่อแหล่งข้อมูล ในบทนี้ผู้เรียนจะได้เรียนรู้การสร้างระบบฐานข้อมูลโดยใช้ Python Django เรียนรู้สถาปัตยกรรม Python Django การติดตั้งโปรแกรมที่ใช้ในการทำ Python Django จากนั้นเขียนคำสั่งเพื่อสร้างระบบฐานข้อมูล (Database) สำหรับใช้ในการเก็บข้อมูลจากแอปพลิเคชัน การใช้คำสั่งสร้างหน้าใช้งานของผู้ดูแลระบบ การสร้าง Web API ใน Python Django การทำ API ส่ง JSON File ใน Python Django การทดลองส่งข้อมูลออกไปภายนอกอินเทอร์เน็ตโดยการ forward port ผ่าน ngrok ซึ่งเป็นเครื่องมือที่อำนวยความสะดวกแก่ผู้พัฒนาระบบทำให้ระบบที่พัฒนาอยู่ในเครื่องสามารถออนไลน์ได้บนอินเทอร์เน็ต

### Django คืออะไร

Django เป็นเฟรมเวิร์กเว็บ Python ระดับสูงที่สนับสนุนการพัฒนาอย่างรวดเร็ว และการออกแบบที่สะอาดตาและใช้งานได้จริง ช่วยขจัดความยุ่งยากในการพัฒนาเว็บ สามารถใช้งานได้ฟรีและเป็นโอเพนซอร์ส (Open Source) (djangoproject, 2564 : 1).

Django คือ Python Web Framework คือ เครื่องมือที่ออกแบบมาเพื่อใช้สร้างเว็บไซต์ โดยเฉพาะเขียนด้วยภาษา Python ในการจัดการระบบหลังบ้านของเว็บไซต์ (Back End) โดยมีความสามารถที่พร้อมใช้งานหลายอย่างเช่น ระบบจัดการฐานข้อมูล (Database) ระบบผู้ดูแลระบบหลังบ้าน (Administration) ระบบจัดการสมาชิก (User Management) และอื่น ๆ ที่พร้อมสำหรับการสร้าง แอปพลิเคชันให้เสร็จในเวลาอันรวดเร็ว (ลุงวิศวะสอนคำนวณ, 2564 : 2)

โดยองค์กรที่ใช้ Django เป็นซอฟต์แวร์ภายในองค์กร ได้แก่ เว็บไซต์ Instagram, Spotify, YouTube, The Washington Post, NASA, Google, Venmo, Dropbox, Mozilla และ Pinterest (netguru, 2564 : 1)



## ประวัติความเป็นมา Django

Django อ่านออกเสียงเป็น จังโก้ คือ ชื่อของเฟรมเวิร์ก (Framework) สำหรับการพัฒนา Web Application ด้วยภาษาไพธอน ซึ่งโครงการนี้เริ่มต้นขึ้นในปี 2003 โดยนักพัฒนา 2 คนคือ Adrian Holovaty (ชาวอเมริกัน) และ Simon Willison (ชาวอังกฤษ) จนกระทั่งปี 2005 จึงได้เผยแพร่เวอร์ชัน 1.0 ออกสู่สาธารณชนเป็นครั้งแรก โดยใช้สโลแกนว่า Django : The web framework for perfectionists with deadlines.

สำหรับคำว่า Django ได้มาจากชื่อของนักกีตาร์ชาวเบลเยียมผู้โด่งดัง นั่นคือ Django Reinhardt และ ในปัจจุบัน Django Framework กำลังได้รับความนิยมอย่างมากจากนักพัฒนาเว็บไซต์ทั่วโลก (บัญชา ปะสีละเตสัง, 2563 : 25)

## คุณลักษณะที่น่าสนใจของ Django

1. Using Python Language Django ใช้ภาษา Python ในการพัฒนา เนื่องจาก Python นั้นเป็นภาษาที่ได้รับความนิยมสูงสุดในขณะนี้ เพราะมีรูปแบบการเขียนโค้ดที่ไม่ซับซ้อนและเข้าใจง่าย ดังนั้นจึงสามารถเรียนรู้ได้ในระยะเวลาอันรวดเร็ว
2. Fully Customizable: สามารถนำไปปรับปรุงแก้ไข หรือพัฒนาเพิ่มเติม หรือกำหนดวิธีการทำงานใหม่ได้ตามต้องการ
3. Don't Repeat Yourself (DRY) Django จะแยกการทำงานออกเป็นส่วนย่อย ๆ ให้สามารถนำมาใช้ร่วมกันได้ เพื่อลดขั้นตอนการเขียนโค้ดที่ซ้ำซ้อนกัน โดยไม่จำเป็นต้องคัดลอกโค้ดจากส่วนหนึ่งแล้วไปวางในอีกส่วนหนึ่ง (Copy & Paste) สำหรับการทำงานในลักษณะเดียวกัน
4. Model Template View (MTV) โครงสร้างการทำงานของ Django อยู่ในรูปแบบที่เรียกว่า Model Template View (MTV) โดยพัฒนาขึ้นจากแนวคิดพื้นฐานของ Model View Controller (MVC) ซึ่งเป็นสถาปัตยกรรมของ Web Framework ยุคใหม่ ดังนั้น Django จึงทันสมัยและมีประสิทธิภาพ เทียบเท่าเฟรมเวิร์กตัวอื่น ๆ ในปัจจุบัน
5. BSD License: Django อยู่ภายใต้ลิขสิทธิ์ของ BSD จึงมั่นใจได้ว่า สามารถนำไปใช้งานได้ฟรี โดยไม่มีค่าใช้จ่ายและเงื่อนไขใด ๆ
6. Support Community: มีกลุ่มผู้สนับสนุนการใช้งานจำนวนมากที่พร้อมให้ความช่วยเหลือ และแก้ไขปัญหาต่าง ๆ ที่เกิดขึ้น (บัญชา ปะสีละเตสัง, 2563 : 26)

# django

ภาพประกอบ 6.1 django

ที่มา : django (2020 : 1)

## สถาปัตยกรรม Django

Django มีรูปแบบสถาปัตยกรรมที่เรียกว่า MTV หรือ Model Template View เป็นคอนเซ็ปต์การเขียนเว็บไซต์ที่มีส่วนประกอบการทำงานที่เชื่อมโยงกันระหว่าง (บัญชา ปะสีละเตสัง, 2563 : 27)

M - Model มีหน้าที่ในการเชื่อมต่อฐานข้อมูล

T - Template สำหรับการกำหนดโครงสร้างของเพจและการแสดงผล

V- View คือ สำหรับควบคุมกระบวนการทำงานและเชื่อมโยงระหว่าง Model และ Template

### หมายเหตุ\*

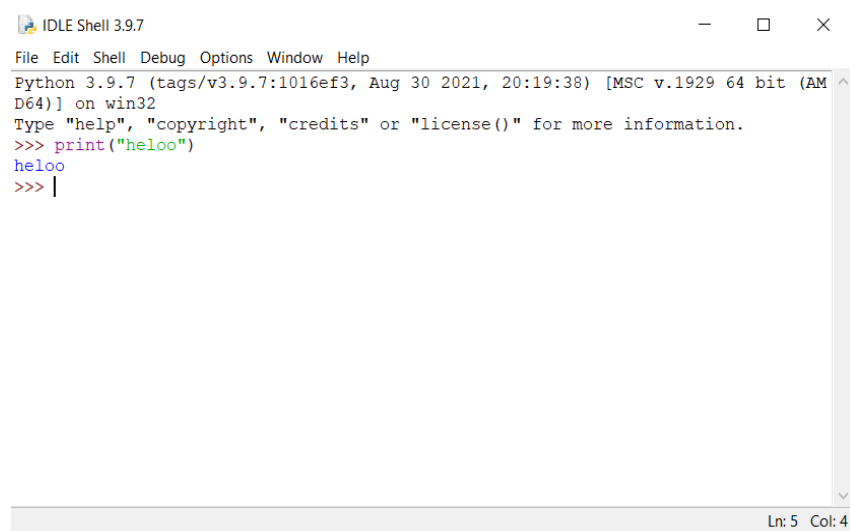
Open Source คือ การพัฒนาซอฟต์แวร์ตามแนวคิดที่อาศัยความร่วมมือของนักพัฒนาทั่วโลก เพื่อสร้างซอฟต์แวร์ที่ดีกว่าเดิม โดยซอฟต์แวร์ดังกล่าวสามารถเปิดเผยโค้ดต้นฉบับ (Source Code)

เริ่มการสร้าง Django ภาษาที่ใช้พัฒนา Python

1. ให้ทำการดาวน์โหลด Python เพื่อนำมาติดตั้งโดยการไปที่

<https://www.python.org/> และทำการกด Download เมื่อดาวน์โหลดเสร็จแล้ว ให้ทำการติดตั้ง โดยเลือก Path ไปที่ C:\Python39

2. เมื่อทำการติดตั้งเสร็จเรียบร้อยแล้วให้ทำการเปิด Python Shell จากนั้นให้พิมพ์ IDLE Shell ตามภาพ เพื่อเป็นการทดสอบว่า Python ที่ติดตั้งใช้งานได้หรือไม่



```

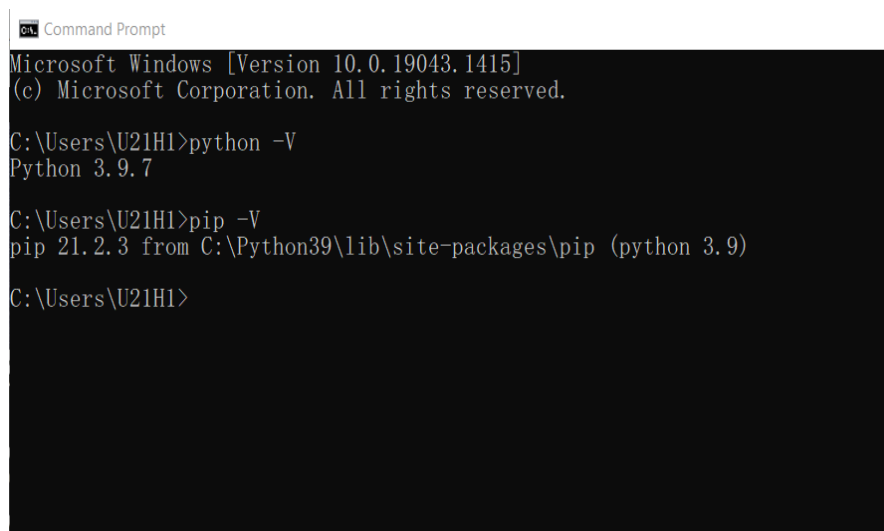
IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("heloo")
heloo
>>> |
Ln: 5 Col: 4

```

### ภาพประกอบ 6.2 เปิด Python Shell

ที่มา : ฅนปักษ์ วรณตรง (2564 : 7-8)

3. ทำการทดสอบ Python ด้วยการเปิดหน้าต่าง cmd จากนั้นให้ทำการพิมพ์ python -V และ pip -V ตามภาพ เพื่อเป็นการตรวจสอบเวอร์ชันของ Python ที่ติดตั้ง หากได้ผลดังภาพแสดงว่าการติดตั้ง Python เสร็จสิ้นเรียบร้อยแล้วพร้อมใช้งาน



```

Command Prompt
Microsoft Windows [Version 10.0.19043.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Users\U21H1>python -V
Python 3.9.7

C:\Users\U21H1>pip -V
pip 21.2.3 from C:\Python39\lib\site-packages\pip (python 3.9)

C:\Users\U21H1>

```

### ภาพประกอบ 6.3 เปิด Python Shell

ที่มา : ฅนปักษ์ วรณตรง (2564 : 9)

4. หากไม่สามารถ Run ได้ ให้ทำการเข้าไปตรวจสอบที่ Path โดยการไปที่ IDLE Shell > Path Browser จากนั้นให้พิมพ์ Environment Variables > Path และแก้ไขหรือเพิ่ม Path ตามภาพ

```
-C:\Python39\  
-C:\Python39\Scripts  
-C:\Python39\Lib\site-packages|
```

ภาพประกอบ 6.4 แก้ไขเมื่อไม่สามารถ Run ได้

ที่มา : ฅนปักษ์ วรณตรง (2564 : 10)

## ติดตั้ง Django

1. ให้ทำการสร้าง Virtual Environment โดยการเปิดหน้าต่าง cmd จากนั้นให้ทำการพิมพ์คำสั่ง pip3.9 install virtualenv ตามภาพ

```
C:\Windows\System32\cmd.exe  
Microsoft Windows [Version 10.0.19045.3570]  
(c) Microsoft Corporation. All rights reserved.  
C:\job\DJANGO WEBAPI>pip3.9 install virtualenv
```

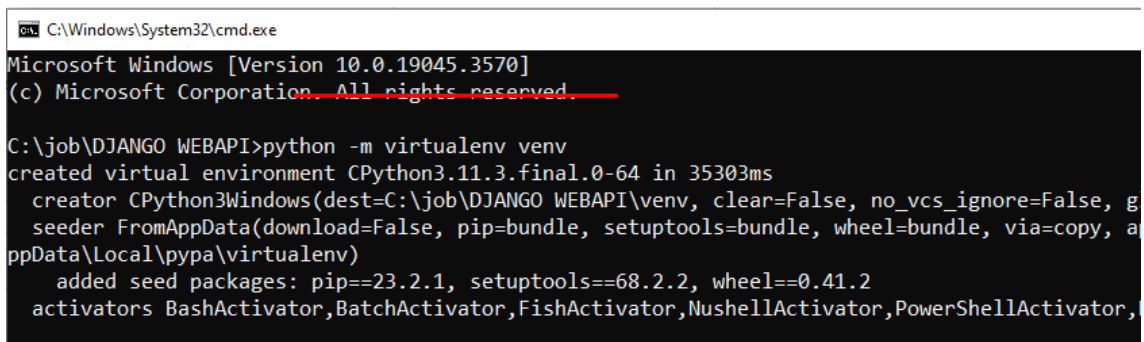
ภาพประกอบ 6.5 ติดตั้ง virtualenv

ที่มา : ฅนปักษ์ วรณตรง (2564 : 12)

หมายเหตุ\*

Virtual Environment คือ สิ่งแวดล้อมเสมือนที่สร้างขึ้นเพื่อติดตั้งโปรแกรมต่างๆ สำหรับการทดลองใช้งาน โดยจะทำให้การทำงานต่าง ๆ ของโปรแกรมที่ติดตั้งใหม่ ไม่ส่งผลกระทบต่อโปรแกรมหลักที่ได้ติดตั้งไว้ก่อนหน้านี้

2. เมื่อสร้างไฟล์งานเสร็จแล้ว ให้ทำการสร้าง New Folder โดยตั้งชื่อว่า DJANGO WEBAPI จากนั้นให้ทำการ cd เข้าไปในโฟลเดอร์ ดังกล่าว สร้าง virtualenv ด้วยการใช้คำสั่ง `python -m virtualenv venv` ตามภาพ



```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

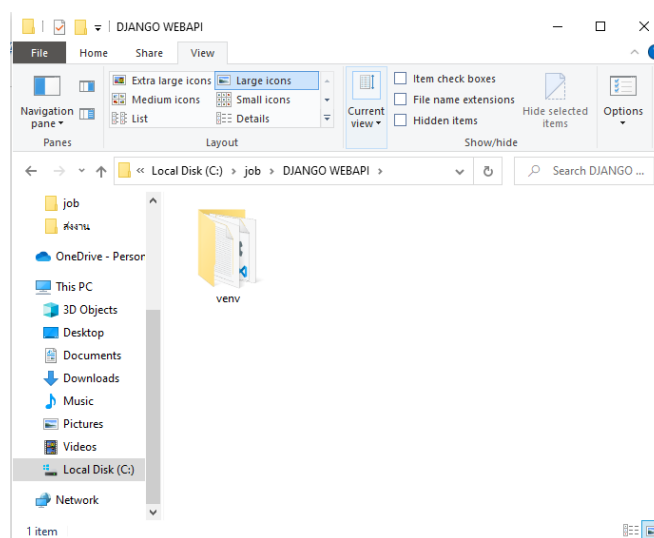
C:\job\DJANGO WEBAPI>python -m virtualenv venv
created virtual environment CPython3.11.3.final.0-64 in 35303ms
creator CPython3Windows(dest=C:\job\DJANGO WEBAPI\venv, clear=False, no_vcs_ignore=False, g
seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, a
ppData\Local\pypa\virtualenv)
added seed packages: pip==23.2.1, setuptools==68.2.2, wheel==0.41.2
activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,

```

ภาพประกอบ 6.6 สร้าง virtualenv

ที่มา : ฅนปลัซ วรณตรง (2564 : 13)

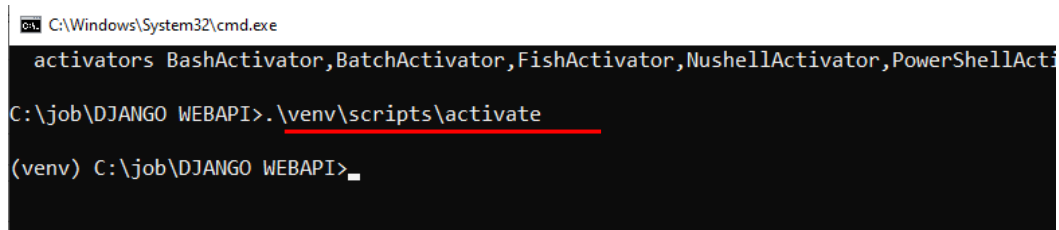
3. จากนั้นจะได้โฟลเดอร์ ขึ้นมาหนึ่งโฟลเดอร์ตามภาพ



ภาพประกอบ 6.7 ได้โฟลเดอร์ venv

ที่มา : ฅนปลัซ วรณตรง (2564 : 14)

4. เมื่อได้โฟลเดอร์ venv แล้วให้กลับไปหน้าจอต่าง cmd จากนั้นให้พิมพ์คำสั่ง  
 .\venv\scripts\activate เพื่อเข้าสู่โหมดการทำงานของ venv ตามภาพ



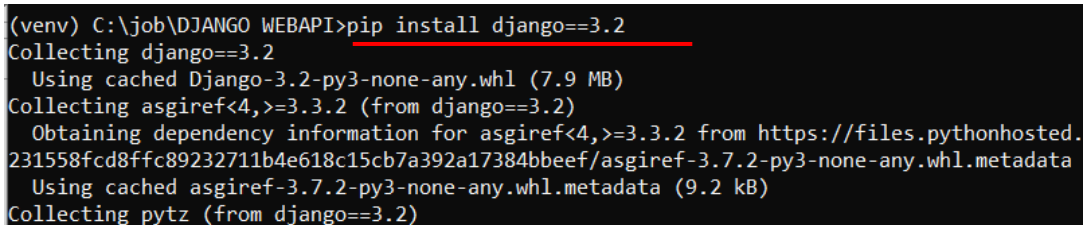
```

C:\Windows\System32\cmd.exe
activators BashActivator, BatchActivator, FishActivator, NushellActivator, PowerShellActi
C:\job\DJANGO WEBAPI> .\venv\scripts\activate
(venv) C:\job\DJANGO WEBAPI>
  
```

ภาพประกอบ 6.8 พิมพ์คำสั่งเพื่อเข้าสู่โหมด venv

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 15)

5. เมื่อเข้าสู่โหมด venv แล้ว ให้ทำการติดตั้ง Django ด้วยการใช้คำสั่ง pip install  
 django==3.2 ตามภาพ (3.2 เป็นเวอร์ชันของ Django ที่ต้องการติดตั้ง)



```

(venv) C:\job\DJANGO WEBAPI> pip install django==3.2
Collecting django==3.2
  Using cached Django-3.2-py3-none-any.whl (7.9 MB)
Collecting asgiref<4,>=3.3.2 (from django==3.2)
  Obtaining dependency information for asgiref<4,>=3.3.2 from https://files.pythonhosted.
  231558fcd8ffc89232711b4e618c15cb7a392a17384bbeef/asgiref-3.7.2-py3-none-any.whl.metadata
  Using cached asgiref-3.7.2-py3-none-any.whl.metadata (9.2 kB)
Collecting pytz (from django==3.2)
  
```

ภาพประกอบ 6.9 ติดตั้ง django

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 16)

6. หากไม่แน่ใจว่าทำการติดตั้งทั้งหมดที่กล่าวมาแล้วหรือยัง ให้ทำการตรวจสอบได้โดยการใช้คำสั่ง `pip list` ตามภาพ

```
(venv) C:\job\DJANGO WEBAPI>pip list
Package      Version
-----
asgiref      3.7.2
Django       3.2
pip          23.2.1
pytz         2023.3.post1
setuptools   68.2.2
sqlparse     0.4.4
wheel        0.41.2
```

ภาพประกอบ 6.10 ตรวจสอบการติดตั้ง

ที่มา : ฌปภัช วรรณตรง (2564 : 17)

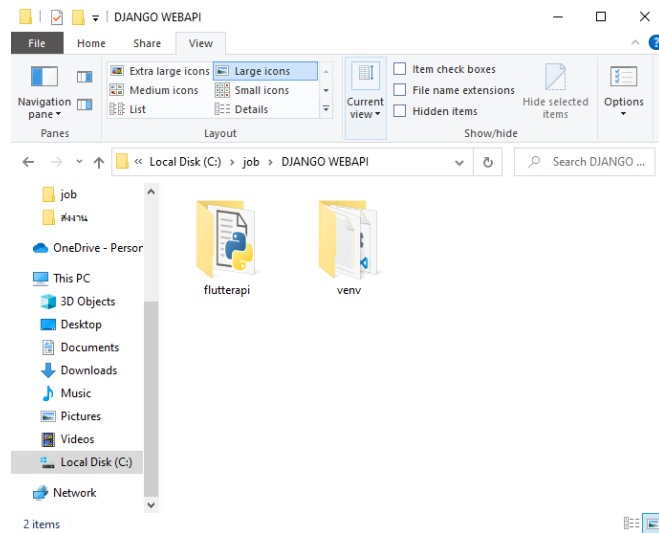
7. เมื่อตรวจสอบแล้ว ให้ทำการสร้าง Project ใน Django ด้วยการพิมพ์คำสั่ง `django-admin startproject` [ชื่อโปรเจกต์]

```
(venv) C:\job\DJANGO WEBAPI>django-admin startproject flutterapi
(venv) C:\job\DJANGO WEBAPI>_
```

ภาพประกอบ 6.11 สร้าง Project ใน django

ที่มา : ฌปภัช วรรณตรง (2564 : 18)

8. จากนั้นจะได้ไฟล์เตอร์โปรเจกต์ตามภาพ



ภาพประกอบ 6.12 โฟลเดอร์ flutterapi

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 19)

9. เปิดหน้าต่าง cmd และทำการ cd เข้าไปในโฟลเดอร์สร้างขึ้น จากนั้นให้ทำการ Run Server โดยการใช้คำสั่ง `python manage.py runserver` ตามภาพ

```
C:\job\DJANGO WEBAPI\flutterapi>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

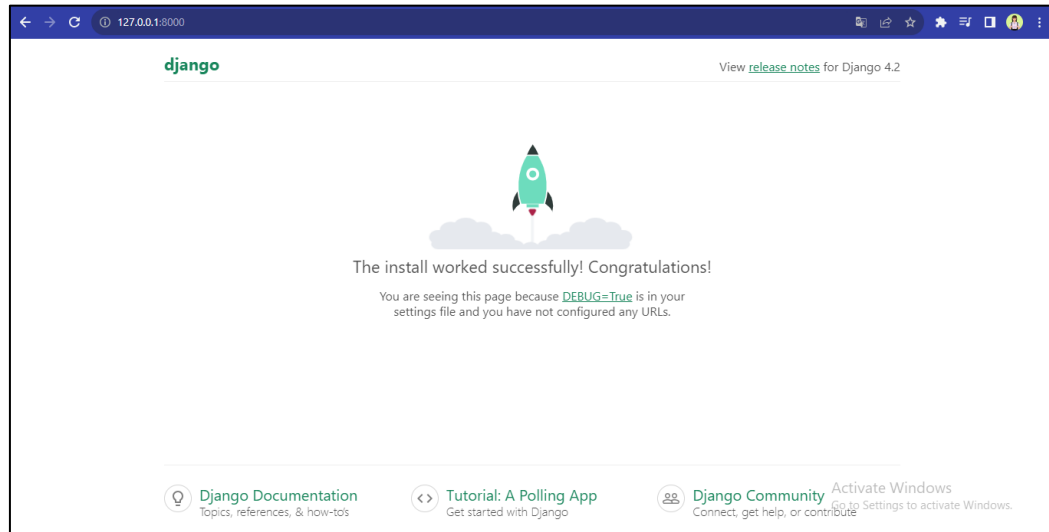
You have 18 unapplied migration(s). Your project may not work properly until you apply them.
Run 'python manage.py migrate' to apply them.
October 17, 2023 - 17:06:08
Django version 4.2.5, using settings 'flutterapi.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

ภาพประกอบ 6.13 Run Server

ที่มา : ฅนปักษ์ วรรณตรง (2564 : 20)



10. ให้ทำการทดสอบหน้าเว็บด้วยการเข้าไปที่ Web Browser จากนั้นให้ทำการพิมพ์ localhost:8000 ที่ Address Bar จะได้ผลลัพธ์ตามภาพ แสดงว่าการติดตั้ง Django เสร็จสมบูรณ์แล้ว



ภาพประกอบ 6.14 Web API django

ที่มา : ฅนปักษ์ วรณตรง (2564 : 21)

หมายเหตุ\*

localhost : การสั่งทำงานบนเครื่องที่ใช้งานอยู่

8000 : หมายเลขพอร์ตคอมพิวเตอร์ที่สั่งทำงาน

11. เมื่อได้ผลลัพธ์ที่ต้องการแล้ว ให้กลับไป cmd และทำการกด Ctrl + c เพื่อทำการสั่งหยุดการทำงานของ Web Server ตามภาพ

```
C:\job\DJANGO WEBAPI\flutterapi>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the
auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
October 17, 2023 - 17:06:08
Django version 4.2.5, using settings 'flutterapi.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

C:\job\DJANGO WEBAPI\flutterapi>_
```

ภาพประกอบ 6.15 สั่งหยุดการทำงานของ Web Server

ที่มา : ฅนปักษ์ วรณตรง (2564 : 22)

12. ให้ทำการสร้างฐานข้อมูลด้วยการพิมพ์คำสั่ง python manage.py migrate ในหน้าต่าง cmd ตามภาพ

```

C:\job\DJANGO WEBAPI\flutterapi>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
C:\job\DJANGO WEBAPI\flutterapi>

```

ภาพประกอบ 6.16 สร้างฐานข้อมูล

ที่มา : ฅปภัช วรรณตรง (2564 : 23)

13. เมื่อสร้างฐานข้อมูลเสร็จสิ้นแล้วให้ทำการสร้างหน้าของผู้ดูแลระบบด้วยการใช้คำสั่ง python manage.py createsuperuser จากนั้นจะขึ้นให้กรอกข้อมูลให้ครบ (ไม่จำเป็นต้องกรอก e-mail) เมื่อกรอกครบแล้วให้ทำการกดตกลง (y) ตามภาพ

```

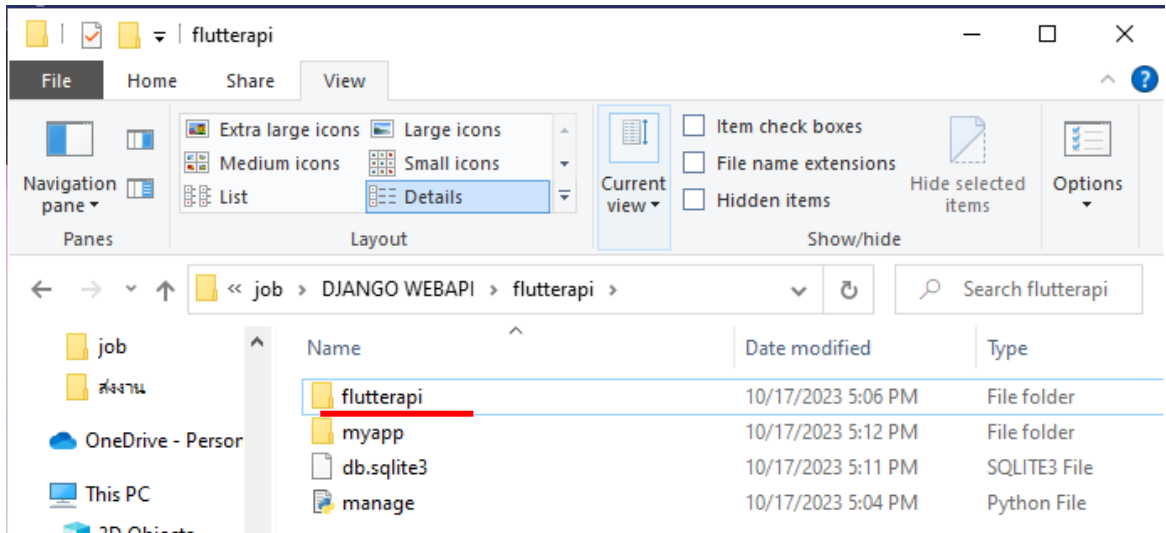
C:\job\DJANGO WEBAPI\flutterapi>python manage.py createsuperuser
Username (leave blank to use 'sawb'): admin
Email address:
Password:
Password (again):
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
C:\job\DJANGO WEBAPI\flutterapi>

```

ภาพประกอบ 6.17 สร้างหน้าของผู้ดูแลระบบ

ที่มา : ฅปภัช วรรณตรง (2564 : 24)

14. หลังจากที่ได้ทำการสร้างหน้าผู้ดูแลระบบเรียบร้อยแล้ว ให้ทำการพิมพ์คำสั่ง python manage.py startapp [ชื่อแอปพลิเคชันย่อยที่ต้องการสร้าง] ในหน้าต่าง cmd เพื่อสร้างแอปพลิเคชันย่อยขึ้นมาในโฟลเดอร์ หากสร้างเสร็จแล้วจะได้โฟลเดอร์ขึ้นมาตามภาพ



ภาพประกอบ 6.18 สร้างโฟลเดอร์แอปพลิเคชันย่อย

ที่มา : ฌปภัช วรรณตรง (2564 : 25)

15. สามารถตรวจสอบโฟลเดอร์ที่สร้างขึ้นมาได้ด้วยการเปิดหน้าต่าง cmd โฟลเดอร์ โปรเจกต์ และทำการใช้คำสั่ง dir ตามภาพ

```
C:\job\DJANGO WEBAPI\flutterapi>dir
Volume in drive C has no label.
Volume Serial Number is 886C-D918

Directory of C:\job\DJANGO WEBAPI\flutterapi
```

ภาพประกอบ 6.19 ตรวจสอบโฟลเดอร์

ที่มา : ฌปภัช วรรณตรง (2564 : 26)

## สร้าง API เพื่อทำการส่ง json File ใน Django

### 1. API (Application Programming Interfaces) คืออะไร

API (Application Programming Interfaces) คือ ชุดข้อมูลและฟังก์ชันเพื่ออำนวยความสะดวกในการติดต่อระหว่างโปรแกรมคอมพิวเตอร์ (Masse M., 2011)

การให้บริการข้อมูลบนเว็บเซิร์ฟเวอร์ที่สร้าง มาเพื่อรองรับความต้องการข้อมูลของเว็บไซต์หรือแอปพลิเคชันอื่น ๆ โดยโปรแกรมหรืออุปกรณ์ที่เป็นเครื่องลูกข่ายจะใช้งาน API เพื่อสื่อสารกันกับงานบริการเว็บ ซึ่งโดยทั่วไปแล้ว API จะเปิดเผย ชุดข้อมูลและฟังก์ชันการใช้งานสำหรับการติดต่อระหว่างเครื่องลูกข่ายที่ร้องขอข้อมูลบริการกับผู้ให้บริการ และจะอนุญาตให้เกิดการแลกเปลี่ยนข้อมูลกันได้ (เอกรินทร์ วทัญญูเลิศสกุล, 2563 : 188)

### 2. เจสัน (JSON) คืออะไร

เจสัน (JavaScript Object Notation : JSON) เป็นรูปแบบการแลกเปลี่ยนข้อมูลที่หลายระบบตกลงที่จะใช้ในการสื่อสารข้อมูล รูปแบบการแลกเปลี่ยนข้อมูล คือ รูปแบบข้อความที่ใช้ในการแลกเปลี่ยนข้อมูลระหว่างแพลตฟอร์ม (Bassett, L., 2015)

เป็นรูปแบบข้อมูลที่ช่วยให้แอปพลิเคชันสามารถสื่อสารผ่านเครือข่าย ซึ่งโดยทั่วไปจะผ่าน RESTful API ประกอบด้วยโครงสร้างที่เป็นมิตรต่อนักพัฒนา (Marrs, T., 2017)

เป็นรูปแบบการสื่อสารที่ช่วยอำนวยความสะดวกในการส่งข้อมูลจำนวนมากโดยอาศัยรูปแบบของข้อความปกติ ซึ่งเป็นพื้นฐานการสื่อสารที่ทำได้ง่าย รวดเร็ว และเป็นมาตรฐานที่ทุกแพลตฟอร์มทุกภาษาสามารถใช้ได้ แต่ด้วยเหตุผลของข้อมูลที่อาจจะต้องมีรายละเอียด หมวดย่อย หรือความหลากหลายมากมาย เจสันได้ออกแบบมาให้สามารถบรรจุข้อมูลที่มีความเป็นระเบียบเรียบร้อย และสามารถสกัดข้อมูลเหล่านั้นออกมาเพื่อนำไปใช้งานได้โดยง่าย รูปแบบการสื่อสารของเจสัน โดยทั่วไปจะเขียนอยู่ภายในวงเล็บ {} ได้ (เอกรินทร์ วทัญญูเลิศสกุล, 2563 : 134)

ดังนั้นโดยสรุป API คือ ชุดข้อมูลและฟังก์ชันเพื่ออำนวยความสะดวกในการติดต่อระหว่างโปรแกรมคอมพิวเตอร์ และ json คือ รูปแบบข้อมูลที่ช่วยให้แอปพลิเคชันสามารถสื่อสารผ่านเครือข่าย ซึ่งในส่วนนี้ผู้เรียนจะได้สร้าง API เพื่อทำการส่ง json File ใน Django ดังมีรายละเอียดขั้นตอนดังต่อไปนี้

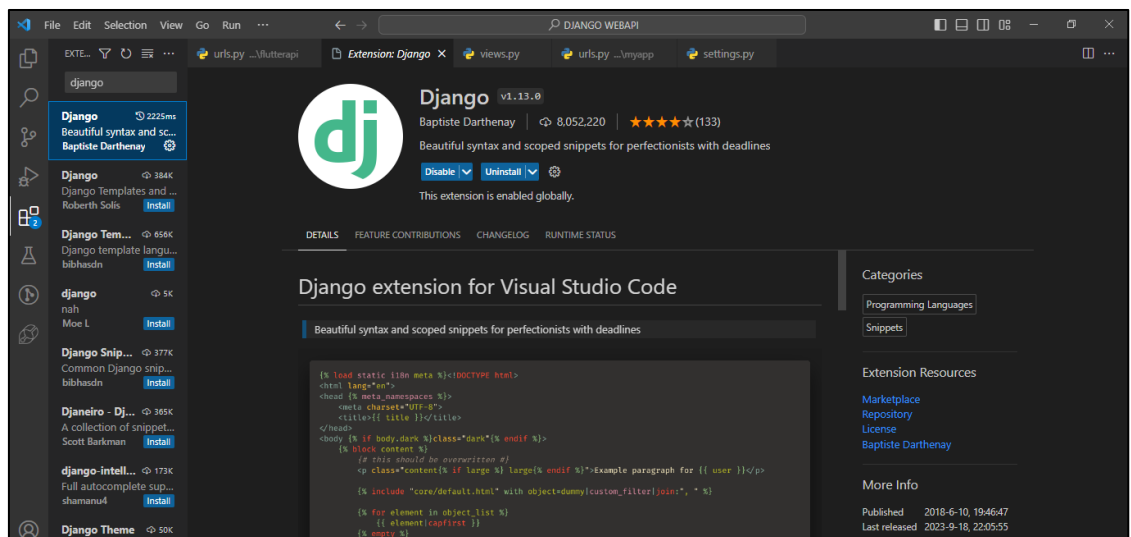
1. ให้ทำการเปิดโปรแกรม VSCode ขึ้นมาจากนั้นให้ทำการติดตั้งส่วนเสริม Python Extension ตามภาพ



ภาพประกอบ 6.20 เปิด VSCode และทำการติดตั้งส่วนเสริม

ที่มา : ฅนปักษ์ วรณตรง (2564 : 28)

2. จากนั้นให้ทำการลงส่วนเสริม Django



ภาพประกอบ 6.21 ติดตั้งส่วนเสริม django

ที่มา : ฅนปักษ์ วรณตรง (2564 : 29)

3. เมื่อทำการติดตั้งส่วนเสริมเสร็จสิ้นแล้ว ให้เปิดไฟล์แล้วไปที่ urls.py ในโฟลเดอร์ Project และทำการคัดลอกโค้ดตามภาพ

```

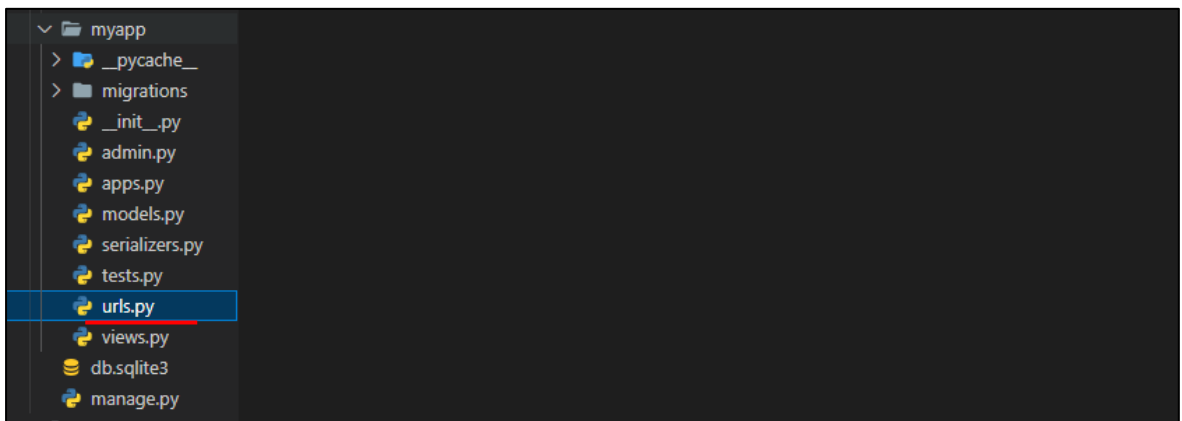
16 from django.contrib import admin
17 from django.urls import path
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     ...
22 ]
23

```

ภาพประกอบ 6.22 คัดลอกโค้ดที่ต้องใช้งาน

ที่มา : ฌปภัช วรรณตรง (2564 : 30)

4. ให้กลับไปโฟลเดอร์แอปพลิเคชันย่อยที่สร้าง และทำการสร้างไฟล์ใหม่โดยตั้งชื่อว่า urls.py



ภาพประกอบ 6.23 สร้างไฟล์ใหม่

ที่มา : ฌปภัช วรรณตรง (2564 : 31)

5. นำโค้ดที่คัดลอกมา มาวางไว้ในไฟล์ urls.py ของแอปพลิเคชันย่อยที่สร้าง และเพิ่มโค้ดตามภาพ

```

1  from django.urls import path
2  from .views import *
3  # * ดึงมาทุกฟังก์ชันใน views.py
4
5  urlpatterns = [
6      path('', Home),
7  ]

```

ภาพประกอบ 6.24 วางโค้ดที่คัดลอกมาและเพิ่มโค้ด

ที่มา : ฌปภัช วรรณตรง (2564 : 32)

6. ให้ทำการตั้งค่าโปรเจกต์งาน โดยการไปที่ไฟล์ settings.py และให้ทำการตั้งค่าโดยใส่ '\*' ใน [] ของ ALLOWED\_HOSTS ตามภาพ

```

26  DEBUG = True
27
28  ALLOWED_HOSTS = ['*']
29
30

```

ภาพประกอบ 6.25 ใส่ '\*' ใน [] ของ tag ALLOWED\_HOSTS

ที่มา : ฌปภัช วรรณตรง (2564 : 33)

จากนั้นไปที่ INSTALLED\_APPS และพิมพ์ชื่อแอปพลิเคชันย่อยที่สร้าง เพื่อเป็นลงทะเบียนแอปพลิเคชัน

```

33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      'myapp',
41      'rest_framework'
42  ]

```

ภาพประกอบ 6.26 ใส่ 'myapp' ใน [] ของ tag INSTALLED\_APPS

ที่มา : ฌปภัช วรรณตรง (2564 : 34)



7. จากนั้นให้กลับไปไฟล์ `urls.py` ของไฟล์ Project และทำการแก้ไขดังนี้

7.1 เพิ่ม `include` ตามภาพ เพื่อใส่คำสั่งการส่งต่อ url จากโปรเจกต์ไปยังแอปพลิเคชันย่อยที่สร้าง

```
16 from django.contrib import admin
17 from django.urls import path, include
18
```

ภาพประกอบ 6.27 ใส่ `include` ใน tag `django.urls` ต่อจาก `path`

ที่มา : ฌปภัช วรรณตรง (2564 : 35)

7.2 เพิ่ม `path` ตามภาพ

```
16 from django.contrib import admin
17 from django.urls import path, include
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('', include('myapp.urls')),
22 ]
23
```

ภาพประกอบ 6.28 ให้เพิ่มโค้ดใน `[]` ของ tag `urlpatterns` ตามภาพ

ที่มา : ฌปภัช วรรณตรง (2564 : 36)

8. เปิดไฟล์ views.py ของแอปพลิเคชันย่อยที่สร้างและทำการเพิ่มโค้ดตามภาพ ซึ่งเป็นใส่ข้อมูลที่จะส่งผ่าน API

```

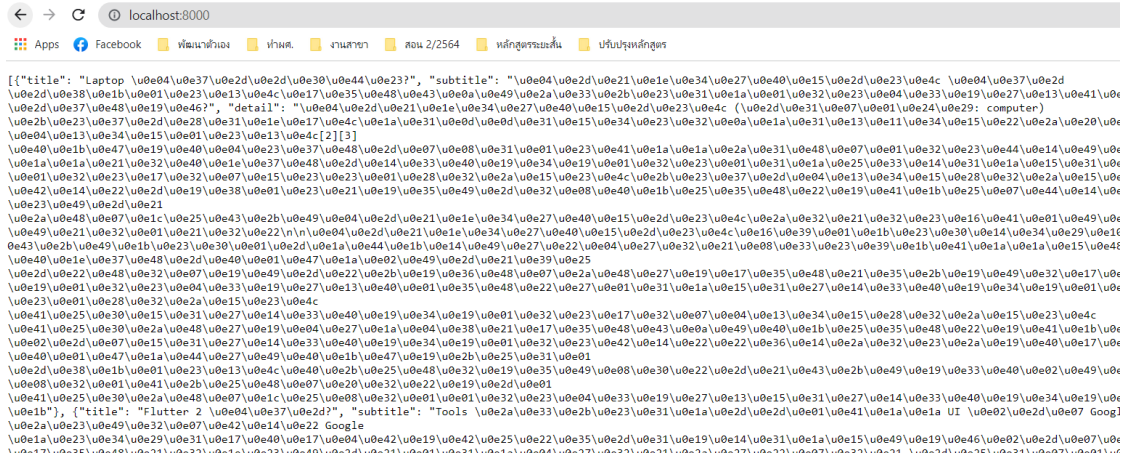
28 from django.http import JsonResponse
29 data = [
30     {
31         "title": "Laptop คืออะไร?",
32         "subtitle": "คอมพิวเตอร์ คือ อุปกรณ์ที่ใช้สำหรับการคำนวณและทำงานอื่นๆ?",
33         "image_url": "https://raw.githubusercontent.com/UncleEngineer/BasicAPI/main/compu",
34         "detail": "คอมพิวเตอร์ (อังกฤษ: computer) หรือศัพท์บัญญัติราชบัณฑิตยสถานว่า คณิตกรร[2][3]
35     },
36     {
37         "title": "Flutter 2 คือ?",
38         "subtitle": "Tools สำหรับออกแบบ UI ของ Google ",
39         "image_url": "https://raw.githubusercontent.com/UncleEngineer/BasicAPI/main/mobil",
40         "detail": "Flutter 2 สร้างโดย Google บริษัทเทคโนโลยีอันดับต้นๆของโลก ที่มาพร้อมกับความสวยงาม
41     },
42     {
43         "title": "Python คือ?",
44         "subtitle": "ภาษาเขียนโปรแกรมชนิดหนึ่ง สร้างขึ้นเมื่อ 1991",
45         "image_url": "https://raw.githubusercontent.com/UncleEngineer/BasicAPI/main/codin",
46         "detail": "ภาษาไพทอน (Python programming language) หรือที่มักเรียกกันว่าไพทอน เป็นภาษาร
47     }
48 ]
49
50
51
52 def Home(request):
53     return JsonResponse(data=data, safe=False, )

```

ภาพประกอบ 6.29 เพิ่มโค้ดในไฟล์ views.py

ที่มา : ฌปภัช วรรณตรง (2564 : 37)

9. บันทึกไฟล์ views.py ของแอปพลิเคชันย่อยที่สร้าง และเข้าไปที่ localhost:8080 ซึ่งจะได้ผลลัพธ์ตามภาพ



ภาพประกอบ 6.30 ได้ผลลัพธ์ดังนี้  
ที่มา : ฌปภัช วรรณตรง (2564 : 38)

10. จากภาพประกอบ 6.30 จะเห็นได้ว่าภาษาที่แสดงไม่สามารถอ่านเข้าใจได้ ให้ทำการแก้ไขเป็น utf8 ด้วยการใช้โค้ดจากผู้พัฒนาท่านอื่น โดยการไปที่ stackoverflow และคัดลอกโค้ดในส่วน return JsonResponse() ตามภาพ

## Creating UTF-8 JsonResponse in Django

Asked 6 years ago Active 11 months ago Viewed 14k times

▲ Is there any simple way to override `DjangoJSONEncoder.ensure_ascii` and set it to `False` or output non-ascii text in `django.http.JsonResponse` in any other way?

16

▼ json django utf-8



Share Follow

1



asked Jan 14 '16 at 20:05



int\_ua

1,334 ● 2 ● 14 ● 30

Add a comment

4 Answers

Active

Oldest

Votes

▲ As of Django 1.9 you can configure `JsonResponse` to disable the `ensure_ascii` switch, by passing in a value for the `json_dumps_params` argument:

18

▼ 

```
return JsonResponse(response_data, safe=False, json_dumps_params={'ensure_ascii': False})
```

ภาพประกอบ 6.31 ค้นหาวิธีแก้ไข utf8 และคัดลอก return JsonResponse()

ที่มา : ฌปภัช วรรณตรง (2564 : 39)

หมายเหตุ\* ลิงก์วิธีแก้ไข utf8 ข้างต้น

<https://stackoverflow.com/questions/34798703/creating-utf-8-jsonresponse-in-django>

11. จากนั้นให้วางโค้ดในส่วน return JsonResponse() แทน return JsonResponse() โค้ดเก่าในไฟล์ view.py ของแอปพลิเคชันย่อยที่สร้าง ซึ่งจะเห็นได้ว่าการเพิ่มส่วน json\_dumps\_params={'ensure\_ascii': False}

```

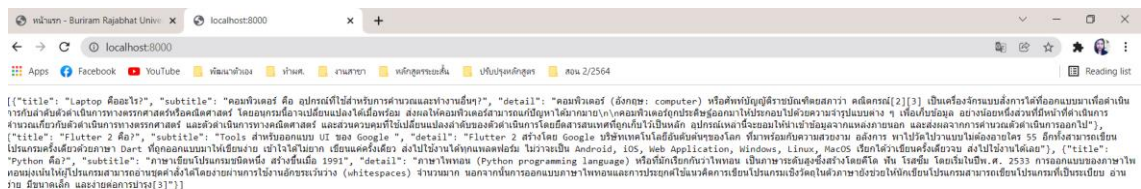
28 from django.http import JsonResponse
29 data = [
30     {
31         "title": "Laptop คืออะไร?",
32         "subtitle": "คอมพิวเตอร์ คือ อุปกรณ์ที่ใช้สำหรับการคำนวณและทำงานอื่นๆ",
33         "image_url": "https://raw.githubusercontent.com/UncleEngineer/BasicAPI/main/computer.jpg",
34         "detail": "คอมพิวเตอร์ (อังกฤษ: computer) หรือศัพท์บัญญัติราชบัณฑิตยสภาว่า คณิตกรณ์[2][3] เป็นเครื่องจักรนึ่งการได้ข้อมูลมาเพื่อคำนวณ
35     },
36 ],
37 [
38     {
39         "title": "Flutter 2 คือ?",
40         "subtitle": "Tools สำหรับออกแบบ UI ของ Google ",
41         "image_url": "https://raw.githubusercontent.com/UncleEngineer/BasicAPI/main/mobile.jpg",
42         "detail": "Flutter 2 สร้างโดย Google บริษัทเทคโนโลยีอันดับหนึ่งของโลก ที่มาพร้อมกับความสวยงาม
43     },
44 ],
45 [
46     {
47         "title": "Python คือ?",
48         "subtitle": "ภาษาเขียนโปรแกรมชนิดหนึ่ง สร้างขึ้นเมื่อ 1991",
49         "image_url": "https://raw.githubusercontent.com/UncleEngineer/BasicAPI/main/coding.jpg",
50         "detail": "ภาษาไพทอน (Python programming language) หรือที่มักเรียกกันว่าไพทอน เป็นภาษาร
51 ]
52 def Home(request):
53     return JsonResponse(data=data, safe=False, json_dumps_params={'ensure_ascii': False})

```

ภาพประกอบ 6.32 วางโค้ดในส่วน return JsonResponse()

ที่มา : ณปภัช วรรณตรง (2564 : 40)

12. จากนั้นให้บันทึกไฟล์และเข้าไปที่ localhost:8080 ซึ่งจะได้ผลลัพธ์ที่เปลี่ยนเป็นภาษาไทยตามภาพ



ภาพประกอบ 6.33 ผลการวางโค้ดในส่วน return JsonResponse()

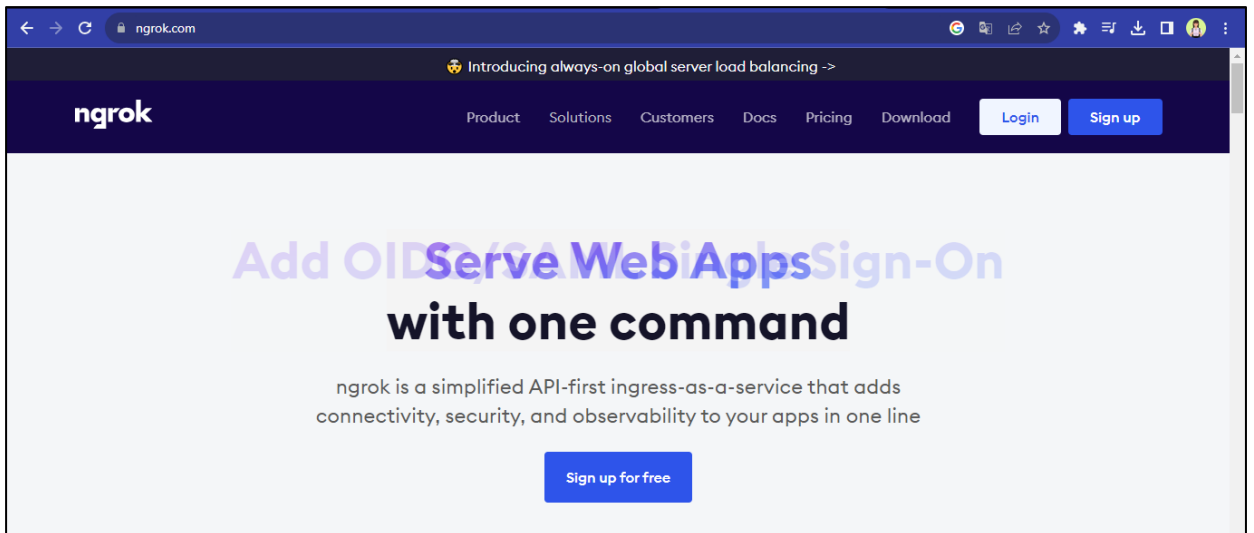
ที่มา : ณปภัช วรรณตรง (2564 : 40)

## การใช้งาน ngrok Forward Port

ngrok คือ ประตูหน้าเข้าสู่แอปพลิเคชัน เป็นส่วนสำหรับติดต่อกับเครือข่ายภายนอก (Reverse Proxy) ที่กระจายไปทั่วโลกซึ่งรักษาความปลอดภัย ปกป้อง และเร่งความเร็วแอปพลิเคชันและบริการเครือข่ายไม่ว่าจะเรียกใช้งานจากที่ใดก็ตาม ngrok รองรับการส่งแอปพลิเคชันที่ใช้ HTTP, TLS หรือ TCP (ngrok, 2023)

วิธีการดาวน์โหลด การติดตั้ง การใช้งานการใช้งาน ngrok สำหรับ Forward Port มีขั้นตอนดังต่อไปนี้

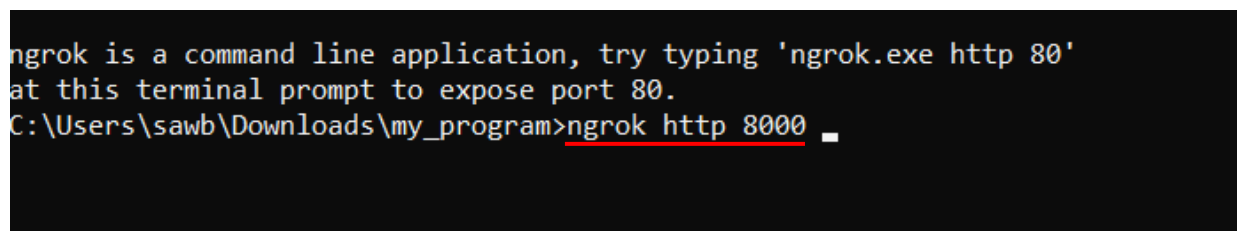
1. ทำการดาวน์โหลด ngrok และติดตั้ง โดยค้นหาจาก Web Browser ได้ตามภาพ



ภาพประกอบ 6.34 ดาวน์โหลดและติดตั้ง ngrok

ที่มา : ฌปภัช วรรณตรง (2564 : 42)

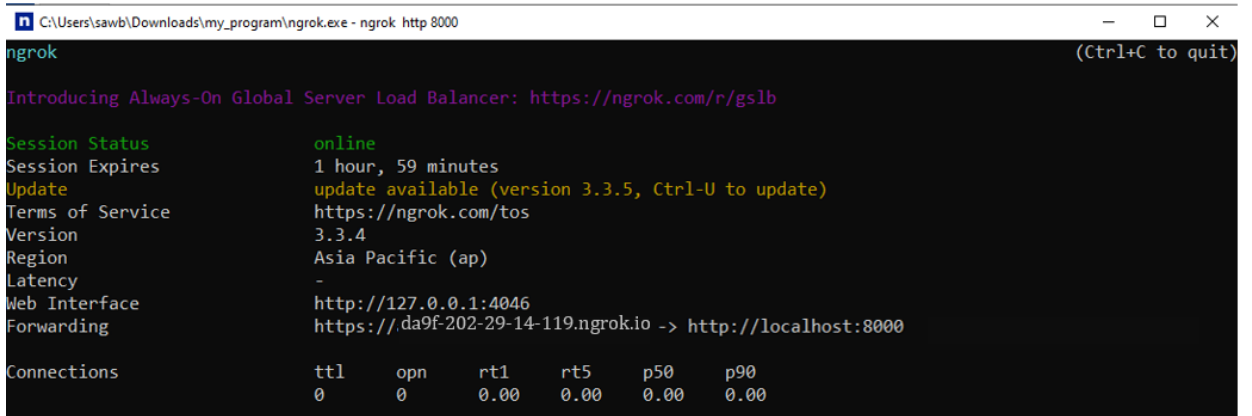
2. เปิดหน้าต่าง cmd ของไฟล์งานที่ได้ทำการสร้างเอาไว้ แล้วทำการพิมพ์คำสั่ง ngrok http 8000 ตามภาพ เพื่อให้ข้อมูลในพอร์ต 8000 ถูกส่งออกไปภายนอกได้



ภาพประกอบ 6.35 พิมพ์คำสั่ง ngrok http 8000

ที่มา : ฌปภัช วรรณตรง (2564 : 43)

3. เมื่อทำการเรียกใช้งาน ngrok Forward Port ได้แล้ว จะได้หมายเลข url ตามภาพ (สามารถส่ง url ให้ผู้พัฒนาท่านอื่นได้ทดลองใช้งาน ใช้งานได้ฟรี 2 ชั่วโมงนับตั้งแต่เปิดใช้งาน)



```

C:\Users\sawb\Downloads\my_program\ngrok.exe - ngrok http 8000
ngrok
Introducing Always-On Global Server Load Balancer: https://ngrok.com/r/gslb

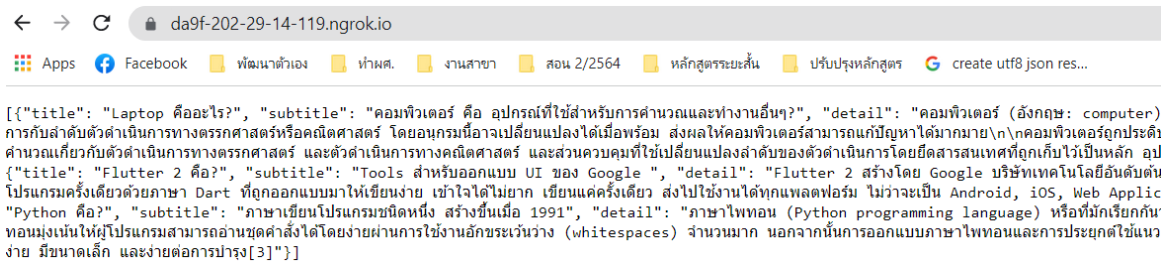
Session Status      online
Session Expires     1 hour, 59 minutes
Update              update available (version 3.3.5, Ctrl-U to update)
Terms of Service    https://ngrok.com/tos
Version             3.3.4
Region              Asia Pacific (ap)
Latency             -
Web Interface       http://127.0.0.1:4046
Forwarding           https://da9f-202-29-14-119.ngrok.io -> http://localhost:8000

Connections
  ttl   opn   rt1   rt5   p50   p90
   0     0    0.00  0.00  0.00  0.00
  
```

ภาพประกอบ 6.36 นำไปทดลองใช้งาน

ที่มา : ฌปภัช วรรณตรง (2564 : 44)

4. จากนั้นให้ทำการเข้า url ที่ได้จาก ngrok ข้างต้น จะได้ผลลัพธ์ตามภาพ



```

da9f-202-29-14-119.ngrok.io
Apps Facebook พัฒนาตัวเอง ทำมศ. งานสาขา สอน 2/2564 หลักสูตรระยะสั้น ปรับปรุงหลักสูตร create utf8 json res...

[{"title": "Laptop คืออะไร?", "subtitle": "คอมพิวเตอร์ คือ อุปกรณ์ที่ใช้สำหรับการคำนวณและทำงานอื่นๆ", "detail": "คอมพิวเตอร์ (อังกฤษ: computer) การกับลำดับตัวดำเนินการทางตรรกศาสตร์หรือคณิตศาสตร์ โดยอนุกรมนี้อาจเปลี่ยนแปลงได้เมื่อพร้อม ส่งผลให้คอมพิวเตอร์สามารถแก้ปัญหาได้มากมาย\nคอมพิวเตอร์ถูกประดิษฐ์ขึ้นโดยนักวิทยาศาสตร์และนักคณิตศาสตร์ และตัวดำเนินการทางคณิตศาสตร์ และส่วนควบคุมที่ใช้เปลี่ยนแปลงลำดับของตัวดำเนินการโดยยึดสารสนเทศที่ถูกเก็บไว้เป็นหลัก อุปกรณ์\nFlutter 2 คือ?", "subtitle": "Tools สำหรับออกแบบ UI ของ Google", "detail": "Flutter 2 สร้างโดย Google บริษัทเทคโนโลยีอันดับต้นโปรแกรมครั้งแรกด้วยภาษา Dart ที่ถูกออกแบบมาให้เขียนง่าย เข้าใจได้ไม่ยาก เขียนแค่ครั้งเดียว ส่งไปใช้งานได้ทุกแพลตฟอร์ม ไม่ว่าจะเป็น Android, iOS, Web Applic\nPython คือ?", "subtitle": "ภาษาเขียนโปรแกรมชนิดหนึ่ง สร้างขึ้นเมื่อ 1991", "detail": "ภาษาไพทอน (Python programming language) หรือที่มักเรียกกันท่อนหนึ่งให้ผู้โปรแกรมสามารถอ่านชุดคำสั่งได้โดยง่ายผ่านการใช้งานอักขระเว้นว่าง (whitespaces) จำนวนมาก นอกจากนั้นการออกแบบภาษาไพทอนและการประยุกต์ใช้แนวง่าย มีขนาดเล็ก และง่ายต่อการบำรุง[3]"}]
  
```

ภาพประกอบ 6.37 ผลลัพธ์การพัฒนา ngrok

ที่มา : ฌปภัช วรรณตรง (2564 : 45)

## ทดสอบระบบเดิมที่ได้ทำการพัฒนาไว้

1. ให้ทำการเปิดหน้าต่าง cmd และทำการ cd เข้าไปยังไฟล์ flutter Project ที่ได้ทำการสร้างไว้ตามภาพ

C:\Windows\System32\cmd.exe

```
C:\job\DJANGO WEBAPI>
```

ภาพประกอบ 6.38 cd เข้าไปยังไฟล์งาน flutter Project

ที่มา : ฅนปลัซ วรณตรง (2564 : 5)

2. จากนั้นให้ทำการพิมพ์คำสั่ง `.\venv\scripts\activate` เพื่อเปิดใช้งานโหมด venv ตามภาพ

```
C:\job\DJANGO WEBAPI>.\venv\scripts\activate
```

```
(venv) C:\job\DJANGO WEBAPI>
```

ภาพประกอบ 6.39 พิมพ์คำสั่งเปิดใช้งานโหมด venv

ที่มา : ฅนปลัซ วรณตรง (2564 : 6)

3. เมื่อทำการเปิดใช้งานโหมด venv ได้แล้ว ให้ทำการ cd เข้าไปในไฟล์ Project ที่ได้ทำการสร้างไว้ตามภาพ

```
C:\job\DJANGO WEBAPI>.\venv\scripts\activate
```

```
(venv) C:\job\DJANGO WEBAPI>cd flutterapi
```

```
(venv) C:\job\DJANGO WEBAPI\flutterapi>
```

ภาพประกอบ 6.40 cd เข้าไปยังไฟล์ Project ที่ได้ทำการสร้างไว้

ที่มา : ฅนปลัซ วรณตรง (2564 : 7)



4. จากนั้นให้ทำการพิมพ์คำสั่ง `python manage.py runserver` สำหรับ Run Server ตามภาพ

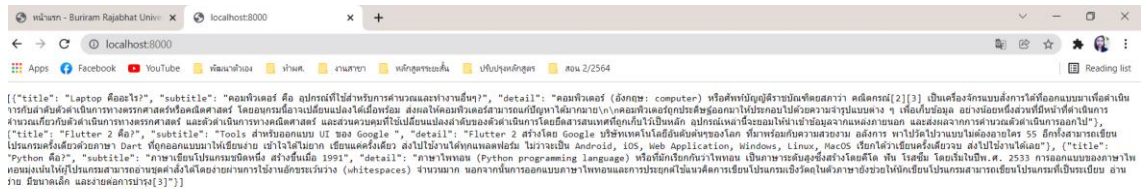
```
(venv) C:\job\DJANGO WEBAPI\flutterapi>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
October 17, 2023 - 17:28:43
Django version 3.2, using settings 'flutterapi.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

ภาพประกอบ 6.41 `cd` เข้าไปยังไฟล์ Project ที่ได้ทำการสร้างไว้

ที่มา : ฌปภัช วรรณตรง (2564 : 8)

5. เปิดหน้าต่าง Web Browser ขึ้นมาจากนั้นให้ทำการเข้าเว็บเซิร์ฟเวอร์โดยการพิมพ์ `localhost:8000` ที่ Address Bar จะได้ผลลัพธ์ตามภาพ



ภาพประกอบ 6.42 เข้าเว็บเซิร์ฟเวอร์โดยการพิมพ์ `localhost:8000`

ที่มา : ฌปภัช วรรณตรง (2564 : 9)

## บทสรุป

สำหรับเนื้อหาที่กล่าวมาทั้งหมดในบทนี้ จะพูดถึงการสร้างระบบฐานข้อมูลโดยใช้ Python Django เรียนรู้สถาปัตยกรรม Python Django การติดตั้งโปรแกรมที่ใช้ในการทำ Python Django การเขียนคำสั่งเพื่อสร้างระบบฐานข้อมูลสำหรับใช้ในการเก็บข้อมูลจากแอปพลิเคชัน การใช้คำสั่งสร้างหน้าใช้งานของผู้ดูแลระบบ การสร้าง Web API ใน Python Django การทำ API ส่ง JSON File ใน Python Django และการทดลองส่งข้อมูลออกไปภายนอกอินเทอร์เน็ตโดยการ forward port ผ่าน ngrok

## เอกสารอ้างอิง

- บัญชา ปะสีละเตสัง. (2563). **พัฒนา Web Application ด้วย Python Django**. กรุงเทพฯ : ซีเอ็ด  
ยูเคชั่น,
- ลุงวิศวสอนคำนวน. (11 กันยายน 2564). **สร้างเว็บไซต์ด้วย Python Django (แจงโก้) เฟรมเวิร์ก  
ใช้เขียนเว็บยอติตของ Python**. สืบค้นจาก,  
<https://www.facebook.com/UncleEngineer>
- เอกรินทร์ วัญญูเลิศสกุล. (2020). **พัฒนา Mobile App ด้วย Flutter & Dart**. กรุงเทพฯ :  
โปริวิชั่น, บจก.
- Bassett, L. (2015). **Bassett Introduction to JavaScript Object Notation**. O'Reilly Media,  
Inc.
- django project. (28 กันยายน 2564). **Django**. สืบค้นจาก, <https://www.djangoproject.com/>
- Marrs, T. (2017). **JSON at Work**. O'Reilly Media, Inc.
- Masse, M. (2011). **REST API Design Rulebook: Designing Consistent RESTful Web  
Service Interfaces**. O'Reilly Media.
- netguru. (28 กันยายน 2564). **Top 10 Django Apps Examples**. สืบค้นจาก,  
<https://www.netguru.com/blog/django-apps-examples>
- ngrok. (1 มกราคม 2566). **Overview**. สืบค้นจาก, <https://ngrok.com/docs>

## บทที่ 7

### การสร้าง RESTful APIs ด้วย Django REST Framework

ในบทนี้จะเป็นเนื้อหาต่อเนื่องจากบทที่ 6 ได้แก่ หลังจากผู้เรียนได้เรียนรู้การสร้างระบบฐานข้อมูลโดยใช้ Python Django เรียนรู้สถาปัตยกรรม Python Django การติดตั้งโปรแกรมที่ใช้ในการทำ Python Django การเขียนคำสั่งเพื่อสร้างระบบฐานข้อมูลสำหรับใช้ในการเก็บข้อมูลแอปพลิเคชันด้วย Python Django ทดลองสร้าง API เบื้องต้นด้วย Python Django และการใช้คำสั่งสร้างหน้าใช้งานของผู้ดูแลระบบ ในบทนี้ผู้เรียนจะได้นำฐานข้อมูลที่ได้สร้างไว้ในบทที่แล้ว มาทำการสร้างข้อมูลในฐานข้อมูล การทดสอบเข้าดูฐานข้อมูลที่มีใน DB Browser for SQLite ได้เรียนรู้ Django REST Framework Django ซึ่งเป็น toolkit หรือไลบรารีของ Python ในการสร้าง RESTful APIs ข้อดี Django REST Framework HTTP Methods ที่ควรรู้ Status Codes ที่ควรรู้ วิธีการติดตั้ง Django REST Framework การสร้างไฟล์ Serializers ซึ่งเป็นไฟล์ที่มีลักษณะคล้าย Model เป็นการระบุว่า จะให้แสดงข้อมูลไหนได้บ้างในหน้าต่าง APIs การติดตั้งโปรแกรม Postman และที่จะใช้ทดสอบการส่ง API ระหว่าง Server กับ Flutter โดรน Postman ซึ่งเป็นเครื่องมือสำหรับการพัฒนาและทดสอบ API service ซึ่งเป็นที่นิยม เนื่องจากใช้งานง่าย ใช้งานได้ฟรี และมีหน้าตาโปรแกรมสวยงาม จากเนื้อหาข้างต้นที่กล่าวมาทั้งหมด มีรายละเอียดดังต่อไปนี้

#### Django REST Framework

Django REST framework เป็นเฟรมเวิร์ก Django REST เป็นชุดเครื่องมือสำหรับการสร้าง Web API (django-rest-framework, 2566)

Django REST Framework คือ Toolkit หรือไลบรารีของ Python ในการสร้าง RESTful APIs โดย REST : REpresentational State Transfer คือ สถาปัตยกรรมอย่างหนึ่งของซอฟต์แวร์ที่อยู่บนพื้นฐานของ HTTP Protocol ที่เกี่ยวข้องกับการส่งข้อมูลระหว่าง Client และ Server (Stackpython, 2021)

ข้อดี Django REST Framework

1. ง่ายในการเรียนรู้เพราะว่า Base on Python ซึ่งก็ขึ้นชื่อในเรื่องของไวยากรณ์ที่อ่านเข้าใจง่าย สละสลวยสวยงามเหมือนภาษาอังกฤษ
2. สร้าง RESTful APIs ได้อย่างรวดเร็ว
3. มี Serializers ที่ช่วยในการแปลง Django, Python Object ไปเป็น JSON ได้อย่างง่ายดาย

4. Browsable API คือ มีหน้า Interface ในการเรียกดู ทดสอบ API ในตัว ทำให้ง่ายและสะดวกมากขึ้น

5. ได้รับความนิยมสูงและมีคอมมูนิตีขนาดใหญ่ มีแหล่งเรียนรู้ต่าง ๆ ไม่ว่าจะเป็นบทความ คลิป Tutorials และแหล่งเรียนรู้อื่น ๆ อีกมากมาย

6. มี Documentation ที่มีเนื้อหาครบถ้วนเขียนไว้ได้อย่างครอบคลุมและละเอียด ทำให้เป็น Reference ได้เป็นอย่างดี (Stackpython, 2021)

### การสร้างข้อมูลในฐานข้อมูล

เนื้อหาส่วนนี้เป็นการเขียนโค้ดเพื่อสร้างข้อมูลลงในฐานข้อมูลที่ได้ดำเนินการสร้างไว้ในบทที่ 5 มีรายละเอียดขั้นตอนดังนี้

1. เปิดไฟล์งาน Project ใน VSCode จากนั้นให้ไปที่ไฟล์ Models.py และให้ทำการเพิ่มโค้ดตามภาพ

```

1 from django.db import models
2
3 class Todolist(models.Model):
4     title = models.CharField(max_length=100)
5     detail = models.TextField(null=True, blank=True)
6

```

ภาพประกอบ 7.1 เปิดไฟล์งาน Project ใน VSCode และเพิ่มโค้ด

ที่มา : ณปภัช วรรณตรง (2564 : 10)

2. จากนั้นให้ไปที่ไฟล์ admin.py และทำการเพิ่มโค้ดตามภาพ ซึ่งส่วนนี้เป็นการเขียนโค้ดเพื่อลงทะเบียนแอปพลิเคชันที่จะสร้างในฐานข้อมูล

```

1 from django.contrib import admin
2 from .models import Todolist
3
4 admin.site.register(Todolist)
5
6

```

ภาพประกอบ 7.2 เปิดไฟล์ admin.py และเพิ่มโค้ด

ที่มา : ณปภัช วรรณตรง (2564 : 11)

3. ให้ทำการหยุด Run Server โดยการกด Ctrl + c ที่หน้าต่าง cmd ที่ได้ทำการสั่งรันไว้ตามภาพ

```
C:\job\DJANGO WEBAPI\flutterapi>
```

ภาพประกอบ 7.3 หยุด Run Server

ที่มา : ณปภัช วรรณตรง (2564 : 13)

4. เมื่อหยุดการ Run Server แล้วให้ทำการพิมพ์คำสั่ง python manage.py makemigrations เพื่อทำตรวจสอบการอัปเดตการเปลี่ยนแปลงว่ามีอะไรเปลี่ยนแปลงบ้างตามภาพ

```
Django version 4.2.5, using settings 'flutterapi.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

C:\job\DJANGO WEBAPI\flutterapi>python manage.py makemigrations
```

ภาพประกอบ 7.4 ตรวจสอบอัปเดตการเปลี่ยนแปลง

ที่มา : ณปภัช วรรณตรง (2564 : 14)

5. จากนั้นให้ทำการพิมพ์คำสั่ง python manage.py migrate เพื่อทำการอัปเดตการเปลี่ยนแปลงต่าง ๆ ที่มีการเปลี่ยนแปลงลงไปพื้นฐานข้อมูล

```
C:\job\DJANGO WEBAPI\flutterapi>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  No migrations to apply.
```

ภาพประกอบ 7.5 อัปเดตการเปลี่ยนแปลง

ที่มา : ณปภัช วรรณตรง (2564 : 15)

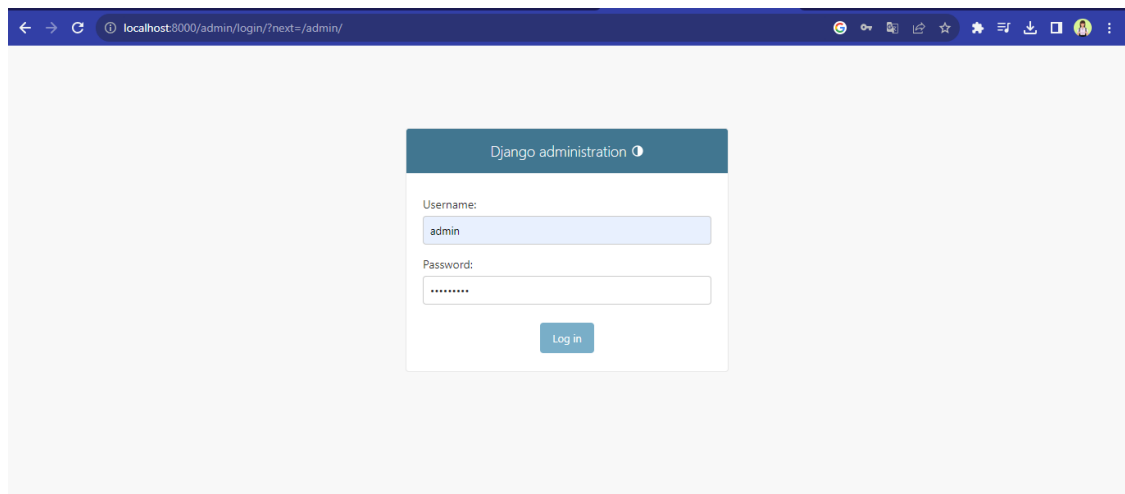
6. เมื่อทำการอัปเดตเสร็จแล้วให้ทำการสั่งการทำงาน Server ตามภาพ

```
C:\job\DJANGO WEBAPI\flutterapi>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
```

ภาพประกอบ 7.6 สั่งการทำงาน Server

ที่มา : ฌปภัช วรรณตรง (2564 : 16)

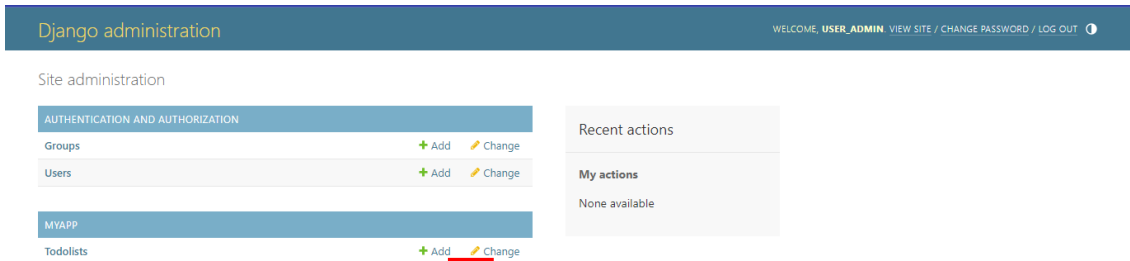
7. เปิด Web Browser และทำการพิมพ์ localhost:8000/admin/ ในชื่อ Address Bar เพื่อทำการเข้าสู่หน้า Admin โดยนำข้อมูลผู้ดูแลระบบที่สร้างใน cmd มากรอกเพื่อทำการเข้าสู่ระบบ ตามภาพ



ภาพประกอบ 7.7 เข้าสู่หน้า Admin

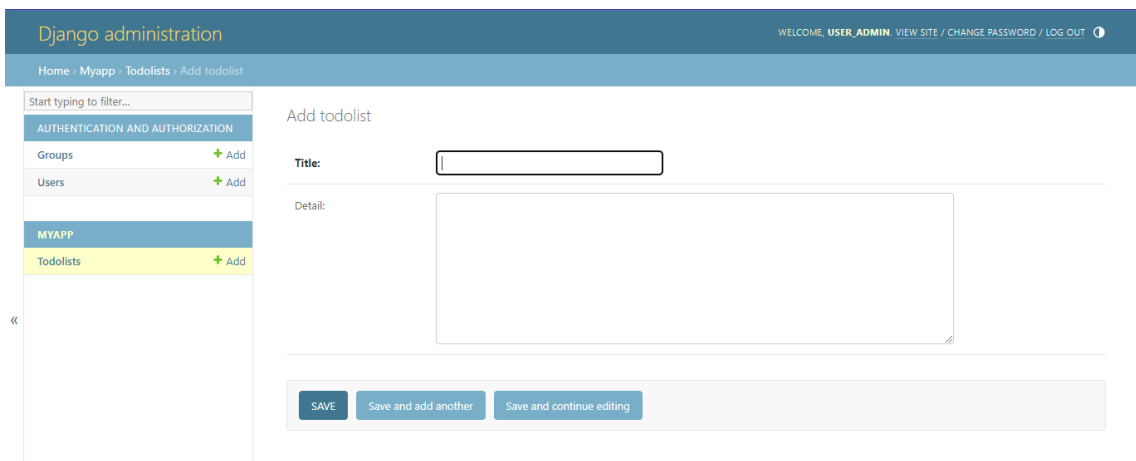
ที่มา : ฌปภัช วรรณตรง (2564 : 17)

8. จากนั้นให้ทำการเข้าสู่ระบบด้วยรหัสผ่านที่ได้ทำการตั้งค่าไว้ จะเข้าสู่หน้าการจัดการฐานข้อมูลตามภาพ



ภาพประกอบ 7.8 หน้าการจัดการฐานข้อมูล  
ที่มา : ณปภัช วรรณตรง (2564 : 18)

9. จากนั้นให้ทำการเข้าไปที่ Todolist > คลิก Add เพื่อทำการเพิ่มข้อมูลเข้าไปในฐานข้อมูล



ภาพประกอบ 7.9 คลิก Add ที่ Todolist  
ที่มา : ณปภัช วรรณตรง (2564 : 19)

Django administration

WELCOME, **USER ADMIN**, VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Myapp > Todolists > Add to do list

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

MYAPP

Todolists + Add

Add to do list

Title: ธนาคาร

Detail: เบิกเงิน 1000

SAVE Save and add another Save and continue editing

ภาพประกอบ 7.10 กรอกข้อมูลที่ต้องการเพิ่มจากนั้นทำการกดบันทึก (Save)  
ที่มา : ณปภัช วรรณตรง (2564 : 20)

Select to do list to change | Django

WELCOME, **ADMIN**, VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Myapp > Todolists

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

MYAPP

Todolists + Add

The to do list "Todolist object (2)" was added successfully.

Select to do list to change

ADD TODOLIST +

Action: [-----] Go 0 of 2 selected

TODOLIST

Todolist object (2)

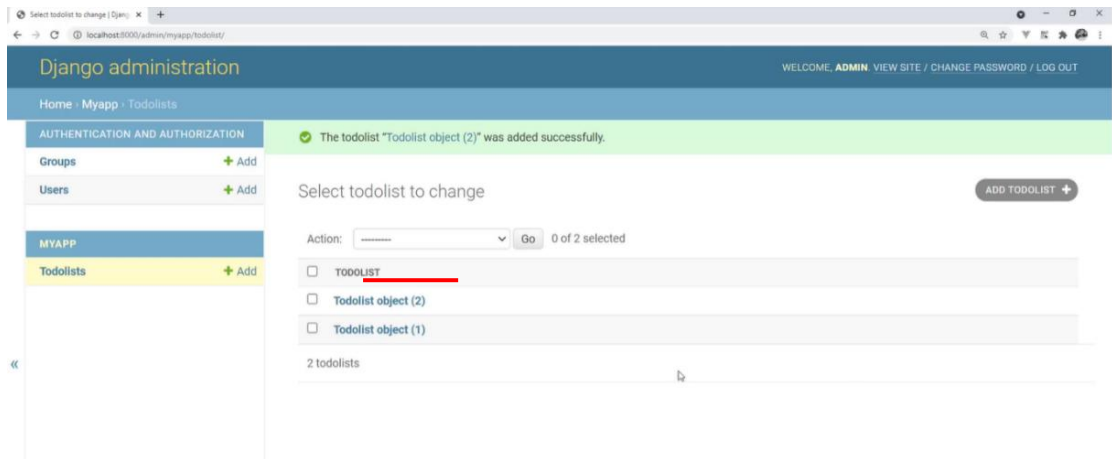
Todolist object (1)

2 to do lists

ภาพประกอบ 7.11 ได้ผลลัพธ์  
ที่มา : ณปภัช วรรณตรง (2564 : 20)



10. ให้ทำการเพิ่มข้อมูลให้ได้ทั้งหมดจำนวน 2 รายการตามภาพ



ภาพประกอบ 7.12 เพิ่มข้อมูลให้ได้ 2 รายการ

ที่มา : ฌปภัช วรรณตรง (2564 : 21)

11. จากภาพข้างต้นจะเห็นได้ว่าชื่อรายการที่สร้าง มีชื่อตามระบบที่ตั้งให้อัตโนมัติ สามารถเปลี่ยนชื่อให้เป็นไปตามที่บันทึกจริง ด้วยการเปิดไฟล์ Models.py เพื่อทำการแก้ไขให้สามารถแสดง Title ในแบบที่ต้องการ โดยการเพิ่มโค้ดตามภาพ

```

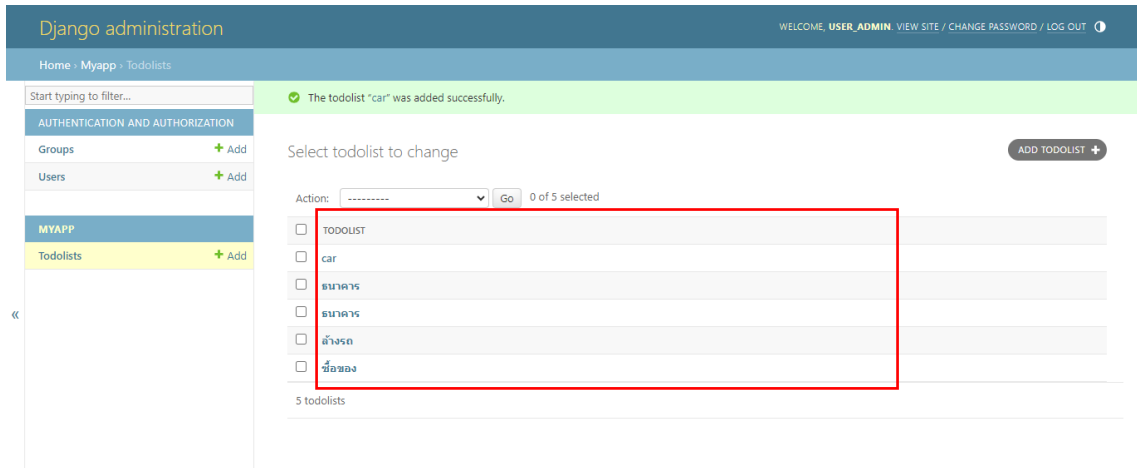
1  from django.db import models
2
3  class Todolist(models.Model):
4      title = models.CharField(max_length=100)
5      detail = models.TextField(null=True, blank=True)
6
7      def __str__(self):
8          return self.title

```

ภาพประกอบ 7.13 เพิ่มโค้ดเพื่อแสดง Title ที่ต้องการ

ที่มา : ฌปภัช วรรณตรง (2564 : 22)

12. จากนั้นให้ทำการกด Refresh Web Browser แล้วจะได้ผลลัพธ์ตามภาพ



ภาพประกอบ 7.14 ผลลัพธ์การแสดง Title

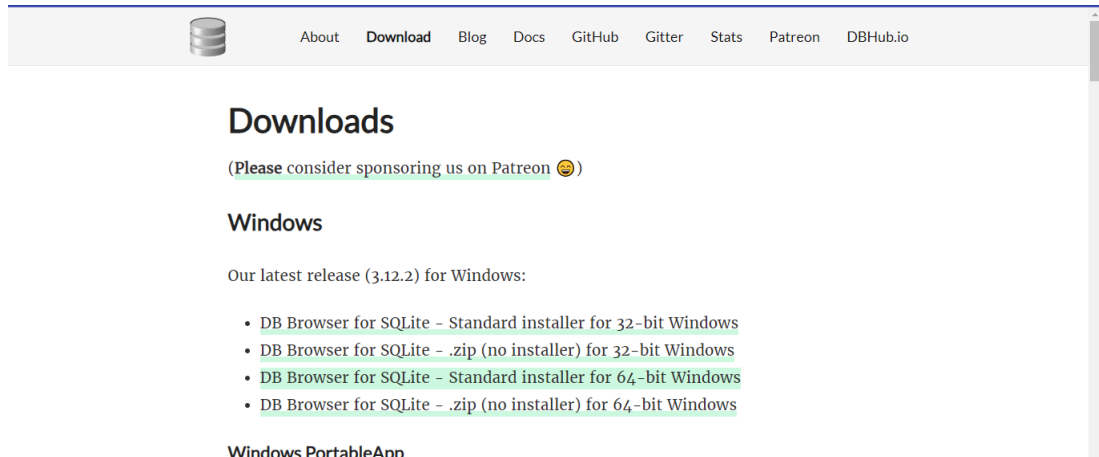
ที่มา : ฌปภัช วรรณตรง (2564 : 23)

### ทดสอบเข้าดูฐานข้อมูลที่มีใน DB Browser for SQLite

DB Browser สำหรับ SQLite เป็นเครื่องมือโอเพ่นซอร์สเชิงภาพคุณภาพสูงสำหรับสร้าง ออกแบบและแก้ไขไฟล์ฐานข้อมูลที่เข้ากันได้กับ SQLite โปรแกรมนี้ได้รับการพัฒนาโดย Mauricio Piacentini (sqlitebrowser, 2566)

เนื้อหาส่วนนี้ทดสอบเข้าดูฐานข้อมูลที่สร้างไว้ในขั้นตอนก่อนหน้านี้นี้ใน DB Browser for SQLite มีรายละเอียดขั้นตอนดังนี้

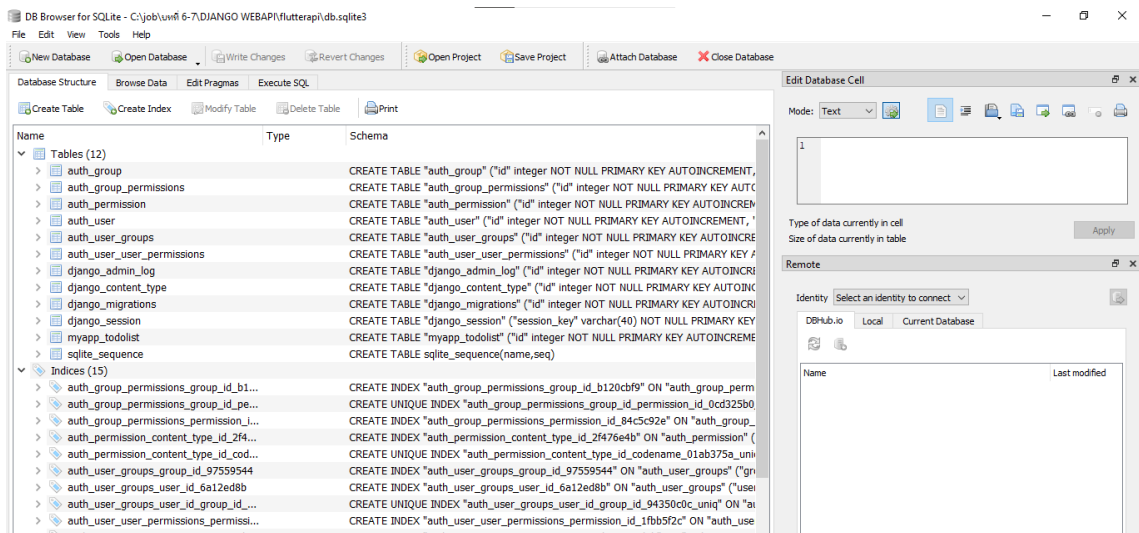
1. ให้ทำการดาวน์โหลดโปรแกรม DB Browser for SQLite ผ่าน Web Browser จากนั้นให้ทำการติดตั้ง



ภาพประกอบ 7.15 ดาวน์โหลดโปรแกรม DB Browser for SQLite

ที่มา : ฌปภัช วรรณตรง (2564 : 25)

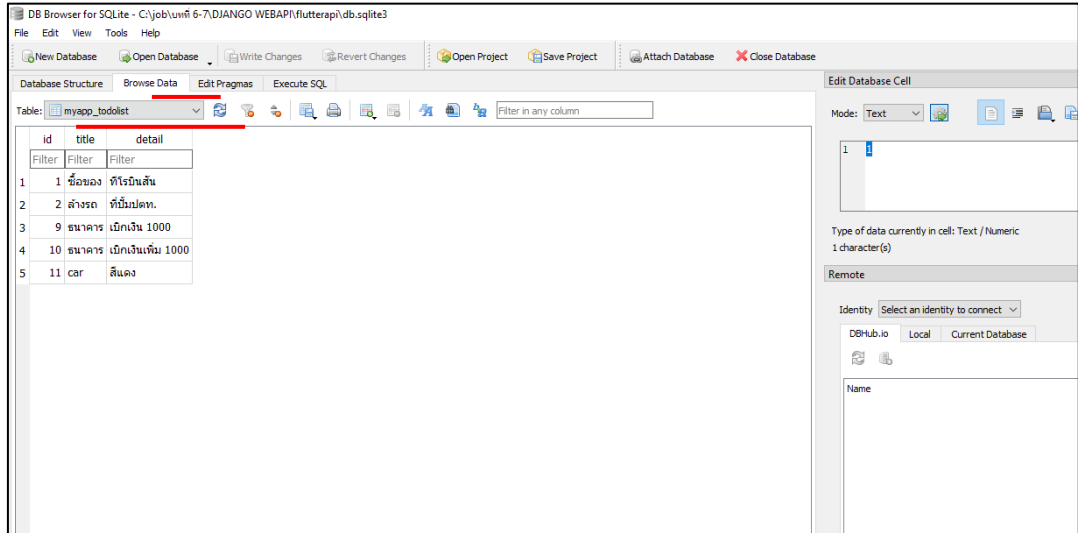
2. หลังจากที่ได้ทำการดาวน์โหลดและติดตั้งโปรแกรมเรียบร้อยแล้วให้ทำการเปิดโปรแกรม จากนั้นให้เลือก Open Database > เลือกไฟล์ Database ใน Project ที่ได้ทำการพัฒนาไว้ เมื่อทำการเลือกแล้วจะปรากฏโครงสร้าง Database ที่ได้ทำการสร้างไว้ตามภาพ



ภาพประกอบ 7.16 เปิดโครงสร้าง Database ในโปรแกรม

ที่มา : ฌปภัช วรรณตรง (2564 : 26)

3. สามารถดูข้อมูลหรือแสดงข้อมูลในตารางได้โดยการไปที่ Browser Data > เลือกตารางที่ต้องการดูหรือแสดง ซึ่งจะได้ผลลัพธ์ตามภาพ



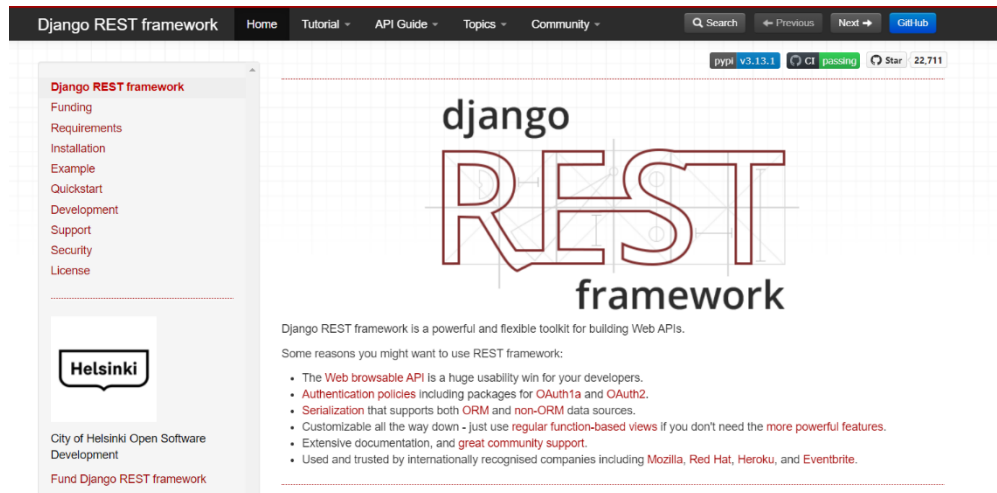
ภาพประกอบ 7.17 ดูข้อมูลหรือแสดงข้อมูลของตาราง

ที่มา : ฅปภัช วรรณตรง (2564 : 27)

## เริ่มต้นติดตั้ง Django REST Framework

เนื้อหาส่วนนี้เป็นการติดตั้ง Django REST Framework สำหรับนำไปใช้งาน มีรายละเอียดขั้นตอนดังนี้

1. ให้ทำการค้นหา Django REST Framework ใน Web Browser จากนั้นให้ไปที่เว็บไซต์เพื่อทำการเริ่มต้นการติดตั้ง



ภาพประกอบ 7.18 ค้นหา Django REST Framework

ที่มา : ฅนปักษ์ วรณตรง (2564 : 34)

2. ไปที่ Installation จากนั้นทำการคัดลอก `pip install djangorestframework` และ `rest_framework`

```

pip install djangorestframework
pip install markdown # Markdown support for the browsable API.
pip install django-filter # Filtering support

```

Add `'rest_framework'` to your `INSTALLED_APPS` setting.

```

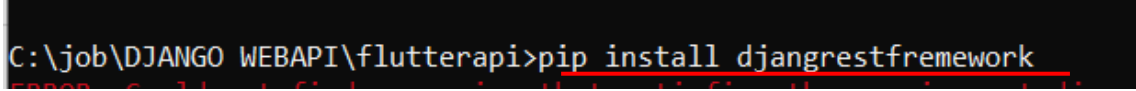
INSTALLED_APPS = [
    ...
    'rest_framework',
]

```

ภาพประกอบ 7.19 คัดลอกโค้ดสำหรับนำไปติดตั้ง

ที่มา : ฅนปักษ์ วรณตรง (2564 : 35)

3. ในหน้าต่าง cmd ให้ทำการหยุด Run Server แล้วทำการวางคำสั่ง pip install djangorestframework เพื่อติดตั้ง Django REST Framework

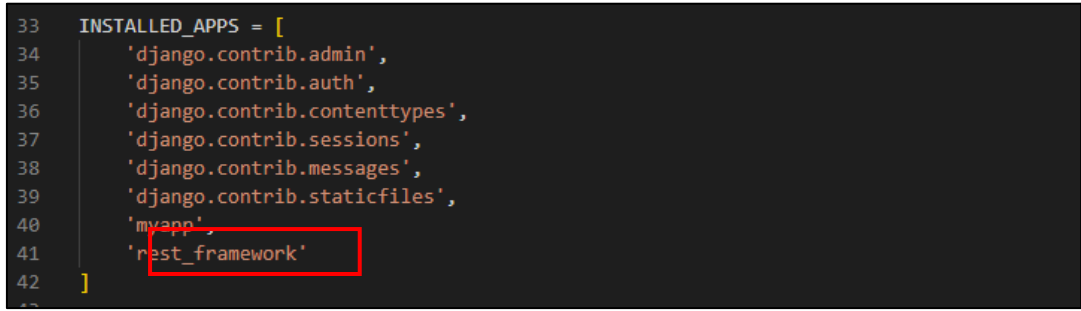


```
C:\job\DJANGO WEBAPI\flutterapi>pip install djangorestframework
```

ภาพประกอบ 7.20 วางคำสั่ง pip install djangorestframework

ที่มา : ฌปภัช วรรณตรง (2564 : 36)

4. เปิดไฟล์ setting.py จากนั้นให้ทำการวางโค้ด rest\_framework เข้าไปตามภาพ ซึ่งเป็น การลงทะเบียนสำหรับการนำไปใช้งาน



```
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'myapp',
41     'rest_framework'
42 ]
```

ภาพประกอบ 7.21 วางคำสั่ง rest\_framework

ที่มา : ฌปภัช วรรณตรง (2564 : 37)

## สร้างไฟล์ Serializers

ไฟล์ Serializers มีลักษณะคล้าย Model เป็นการระบุว่า จะแสดงข้อมูลใดหรือบันทึกข้อมูลใดได้บ้าง ในหน้าต่าง API เนื้อหาส่วนนี้จะเป็นการสร้างไฟล์ Serializers และเขียนโค้ดสำหรับการใช้งาน API ดังมีรายละเอียดขั้นตอนดังนี้

1. สร้างไฟล์ serializers.py ภายใต้โฟลเดอร์งานแอปพลิเคชันย่อย และทำการเพิ่มโค้ดตามภาพ

```

3 from rest_framework import serializers
4 from .models import Todolist
5
6 class TodolistSerializer(serializers.ModelSerializer):
7     class Meta:
8         model = Todolist
9         fields = ('id','title','detail') # '__all__'

```

ภาพประกอบ 7.22 สร้างไฟล์ serializers.py และทำการเพิ่มโค้ด

ที่มา : ฌปภัช วรรณตรง (2564 : 39)

2. จากนั้นไปที่ไฟล์ views.py และทำการเพิ่มโค้ดตามภาพ

```

4 from rest_framework.response import Response
5 from rest_framework.decorators import api_view
6 from rest_framework import status
7 from .serializers import TodolistSerializer
8 from .models import Todolist
9

```

ภาพประกอบ 7.23 ไปที่ไฟล์ views.py และทำการเพิ่มโค้ด

ที่มา : ฌปภัช วรรณตรง (2564 : 40)

3. หลังจากนั้นให้ทำการเพิ่มโค้ดอีกชุดตามภาพ ซึ่งเป็นโค้ดการกำหนดว่า API สามารถทำอะไรได้บ้างและสร้างฟังก์ชันสำหรับการดึงข้อมูล โดยในฟังก์ชันจะมีการส่งข้อมูลและสเตตัสที่จะส่งกลับไปให้ฝั่งผู้รับข้อมูลด้วย

```

4 from rest_framework.response import Response
5 from rest_framework.decorators import api_view
6 from rest_framework import status
7 from .serializers import TodolistSerializer
8 from .models import Todolist
9
10 # GET Data
11 @api_view(['GET'])
12 def all_todolist(request):
13     alltodolist = Todolist.objects.all() #ดึงข้อมูลจาก model Todolist
14     serializer = TodolistSerializer(alltodolist,many=True)
15     return Response(serializer.data, status=status.HTTP_200_OK)
16

```

ภาพประกอบ 7.24 ทำการเพิ่มโค้ดอีกส่วน

ที่มา : ฌปภัช วรรณตรง (2564 : 41)

4. ไปที่ไฟล์ urls.py ของแอปพลิเคชัน จากนั้นให้ทำการแก้ไขและเพิ่มโค้ดตามภาพ เพื่อสร้าง path url API สำหรับให้ผู้รับข้อมูลระบุ

```

1  from django.urls import path
2  from .views import *
3  # * ดึงมาทบทวนใน views.py
4
5  urlpatterns = [
6      path('', Home),
7      path('api/all-todolist/', all_todolist), # localhost:8000/api/all-todolist
8  ]

```

ภาพประกอบ 7.25 เพิ่มโค้ดที่ไฟล์ urls.py

ที่มา : ฌปภัช วรรณตรง (2564 : 42)

5. กลับไปที่หน้าต่าง cmd และทำการสั่งการทำงานServer

```

C:\job\DJANGO WEBAPI\flutterapi>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

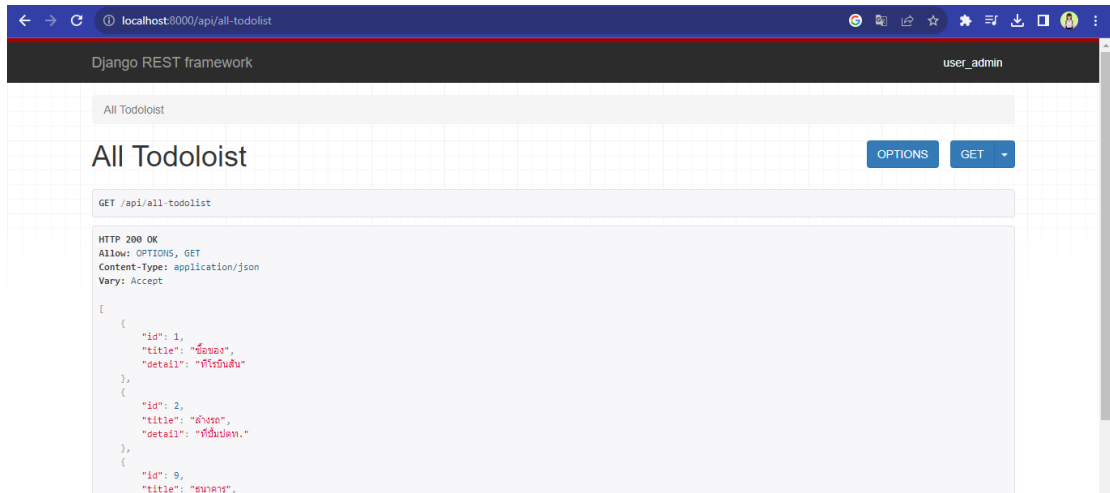
```

ภาพประกอบ 7.26 สั่งการทำงาน Server

ที่มา : ฌปภัช วรรณตรง (2564 : 41)



6. ทดสอบหน้าเว็บโดยการเข้าไปที่ Web Browser และทำการพิมพ์ localhost:8000/api/all-todolist/ ที่ Address Bar จะได้ผลลัพธ์ตามภาพ



ภาพประกอบ 7.27 ผลลัพธ์การสร้างไฟล์ serializers

ที่มา : ฌปภัช วรณตรง (2564 : 44)

ในเนื้อหาที่จะเรียนถัดไปจะเป็นการใช้โปรแกรม Postman ในการจำลองทดสอบการส่งข้อมูลระหว่าง Server กับ Flutter ซึ่งจะมีการเขียนโค้ดเพิ่มเติมและได้มีการกล่าวถึงและใช้งาน HTTP Methods และ Status Codes จึงขอกล่าวถึง HTTP Methods และ Status Codes ที่ผู้เรียนควรรู้จักก่อนเริ่มเรียนเนื้อหาถัดไปดังตาราง

ตาราง 7.1 HTTP Methods ที่ควรรู้ (StackPython, 2564 : 1)

ชื่อเมธอด	การใช้งาน
GET	เมธอดที่ใช้สำหรับดึงข้อมูล
POST	เมธอดที่ใช้สำหรับสร้างข้อมูลขึ้นมาใหม่
PUT	เมธอดที่ใช้สำหรับอัปเดตข้อมูลที่มีอยู่
DELETE	เมธอดที่ใช้สำหรับลบข้อมูล

ตาราง 7.2 Status Codes ที่ควรรู้

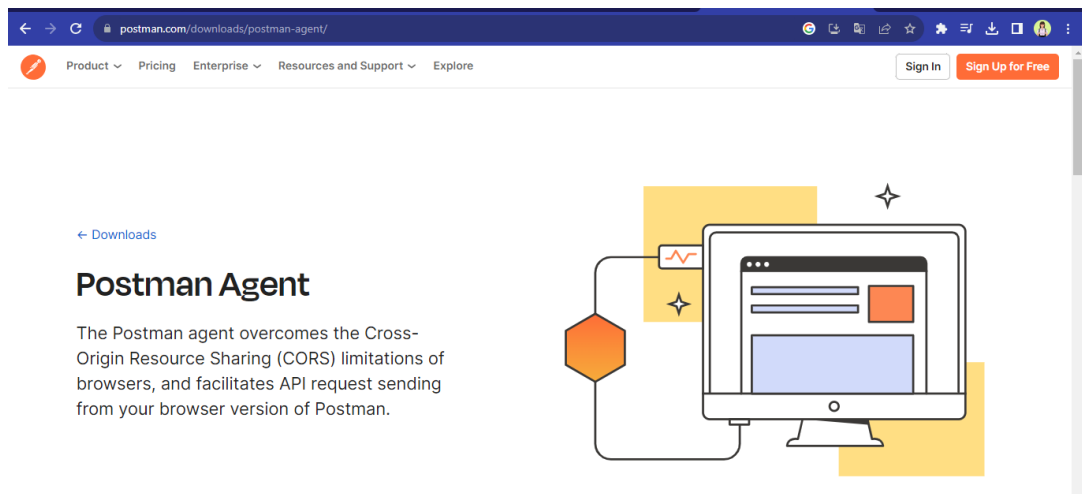
Status Codes	การบ่งบอก
200 – 299	Status ที่บ่งบอกว่า Response สำเร็จ (Successful Responses)
400 – 499	Status ที่บ่งบอกว่า Response ผิดพลาด (Client Error Responses)
500 – 599	Status ที่บ่งบอกว่า Server ผิดพลาด (Server Error Responses)
200 OK	Response สำเร็จ (เป็น Status Code ที่พบเจอบ่อยที่สุด เช่น เมื่อมีการค้นหาหน้าเว็บต่าง ๆ แล้วคลิกเข้าดูได้ Server ก็จะตอบกลับด้วย Status Code นี้)
201 Created	Response สำเร็จ เมื่อ Resource ใหม่ถูกสร้างขึ้น (พบเจอได้บ่อย ๆ เช่น มีการบันทึกข้อมูลผ่านหน้าฟอร์มและถูก Submit ไปที่ Server แล้ว Server ทำการตอบกลับด้วย Status Code นี้)
404 Not Found	Page Not Found (ไม่พบหน้าที่ต้องการหรือร้องขอ) หน้านี้ไม่มีอยู่อีกต่อไปหรือถูกลบไปแล้ว

## โปรแกรม Postman

โปรแกรม Postman เป็นแพลตฟอร์ม API สำหรับการสร้างและใช้งาน API โปรแกรม Postman ทำให้แต่ละขั้นตอนของวงจรการใช้งาน API ง่ายขึ้น และปรับปรุงการทำงานร่วมกันเพื่อให้สามารถสร้าง API ที่ดีขึ้นได้รวดเร็วยิ่งขึ้น (postman, 2566)

โดยในส่วนนี้จะนำโปรแกรม Postman มาใช้ในการทดสอบการส่ง API ระหว่าง Server กับ Flutter ดังมีขั้นตอนต่อไปนี้

1. ให้ทำการค้นหา Postman ที่ Web Browser จากนั้นเข้าไปที่เว็บไซต์และทำการดาวน์โหลด โดยการไปที่ Postman Desktop Agent > Download Postman Agent ตามภาพ



ภาพประกอบ 7.28 ค้นหา Postman

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 46)



### Postman Agent for Mac

For macOS 10.11 (El Capitan) and later

Mac Intel Chip

Mac Apple Chip



### Postman Agent for Windows

For Windows 7 and later

Windows 64-bit



### Postman Agent for Linux

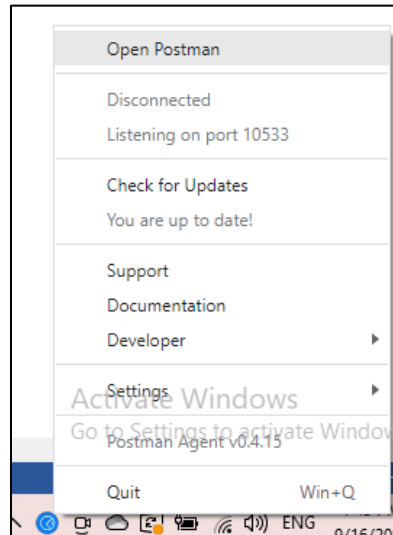
For Ubuntu 14.04 and later, Fedora 24, Debian 8 and later

Linux 64-bit

ภาพประกอบ 7.29 ดาวน์โหลดโปรแกรมสำหรับระบบปฏิบัติการที่ต้องการ

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 47)

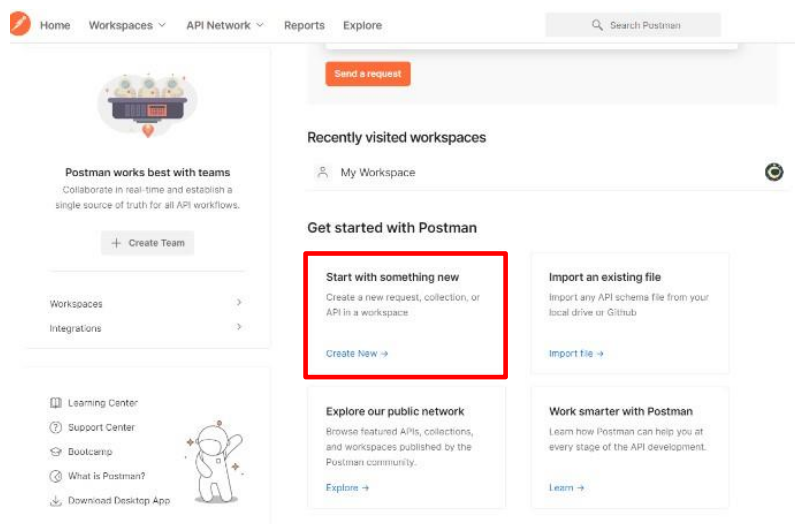
2. หลังจากทำการดาวน์โหลดและติดตั้งเสร็จสิ้นแล้ว ให้ทำการเปิดโปรแกรม Postman จากนั้นให้ไปที่ icon Postman > คลิกขวาเลือกที่ open Postman



ภาพประกอบ 7.30 เปิดโปรแกรม Postman

ที่มา : ฌปภัช วรณตรง (2564 : 34)

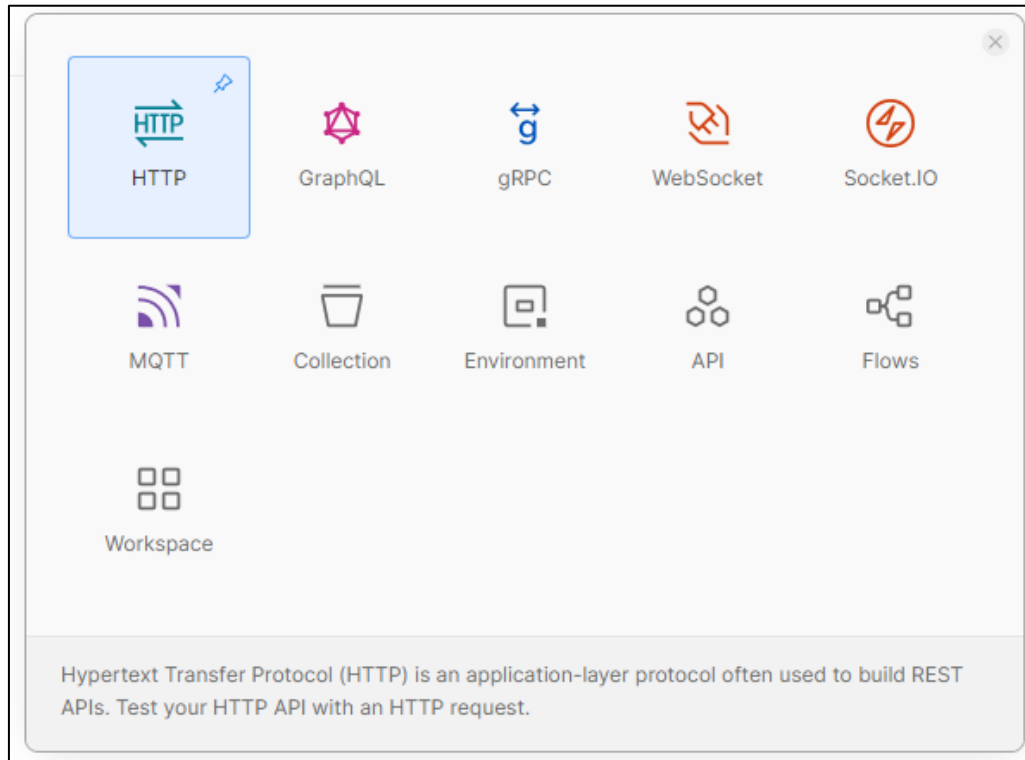
3. เมื่อทำการเปิดโปรแกรม Postman แล้วให้ทำการไปที่ Start with something new > Create New



ภาพประกอบ 7.31 ไปที่ Start with something new > Create New

ที่มา : ฌปภัช วรณตรง (2564 : 48)

4. จากนั้นให้ทำการเลือกเป็น HTTP Request



ภาพประกอบ 7.32 เลือกประเภทงาน

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 49)

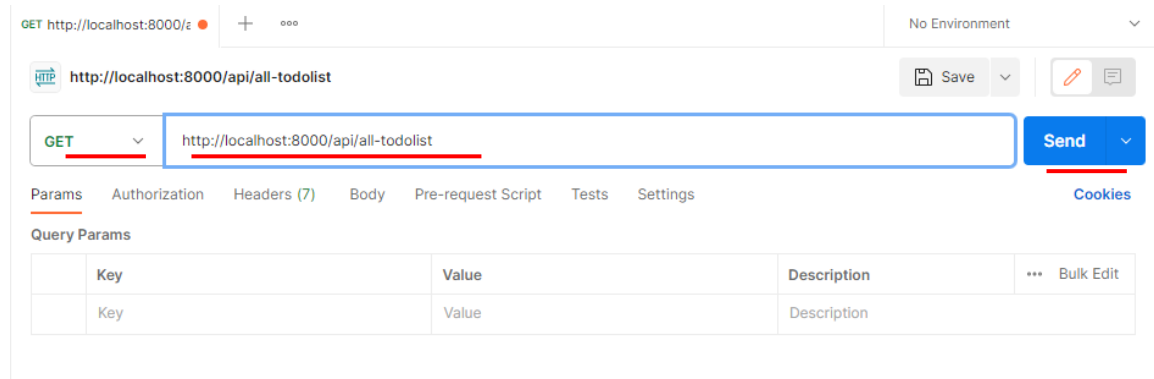
5. ให้ทำการตรวจสอบว่า Server ได้ทำการ Run Server อยู่หรือไม่ หากไม่ได้ Run Server ไว้ ให้ทำการสั่งการทำงานServer

```
C:\job\DJANGO WEBAPI\flutterapi>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
```

ภาพประกอบ 7.33 สั่งการทำงานServer หากไม่ได้ Run Server ไว้

ที่มา : ฅนปลั๊ก วรณตรง (2564 : 50)

6. เมื่อสร้าง Postman เรียบร้อยแล้วให้ทำการไปคัดลอก url จากหน้า Django REST Framework มาใส่ไว้ที่ช่อง GET ตามภาพ



ภาพประกอบ 7.34 คัดลอก url จากหน้า Django REST Framework มาใส่ไว้ที่ช่อง GET  
ที่มา : ฅนปักษ์ วรรณตรง (2564 : 51)

7. จากนั้นจะได้ผลลัพธ์การดึงข้อมูลจาก Server ตามภาพ



ภาพประกอบ 7.35 ผลลัพธ์การดึงข้อมูล  
ที่มา : ฅนปักษ์ วรรณตรง (2564 : 52)

8. เปิดไฟล์ views.py จากนั้นให้ทำการแก้ไขโค้ดตามภาพ เพื่อให้ฝั่ง Flutter สามารถส่งข้อมูลกลับมายัง Server ได้ และมีฟังก์ชันสำหรับรับข้อมูลเก็บลงฐานข้อมูลในฝั่ง Server นอกจากนี้ยังมีการตรวจสอบกรณีข้อมูลที่ส่งมาไม่ถูกต้อง หรือข้อมูลที่ส่งมาถูกต้องแล้วซึ่งได้บันทึกลงฐานข้อมูลแล้ว จะมีการแจ้งเตือนกลับไปฝั่ง Flutter

```

18 # POST Data (save data to database)
19 @api_view(['POST'])
20 def post_todoist(request):
21     if request.method == 'POST':
22         serializer = TodolistSerializer(data=request.data)
23         if serializer.is_valid():
24             serializer.save()
25             return Response(serializer.data, status=status.HTTP_201_CREATED)
26         return Response(serializer.errors, status=status.HTTP_404_NOT_FOUND)
27

```

ภาพประกอบ 7.36 เปิดไฟล์ views.py และแก้ไขโค้ด

ที่มา : ฅนปักษ์ วรณตรง (2564 : 53)

9. เปิดไฟล์ urls.py จากนั้นให้ทำการเพิ่มโค้ดตามภาพ

```

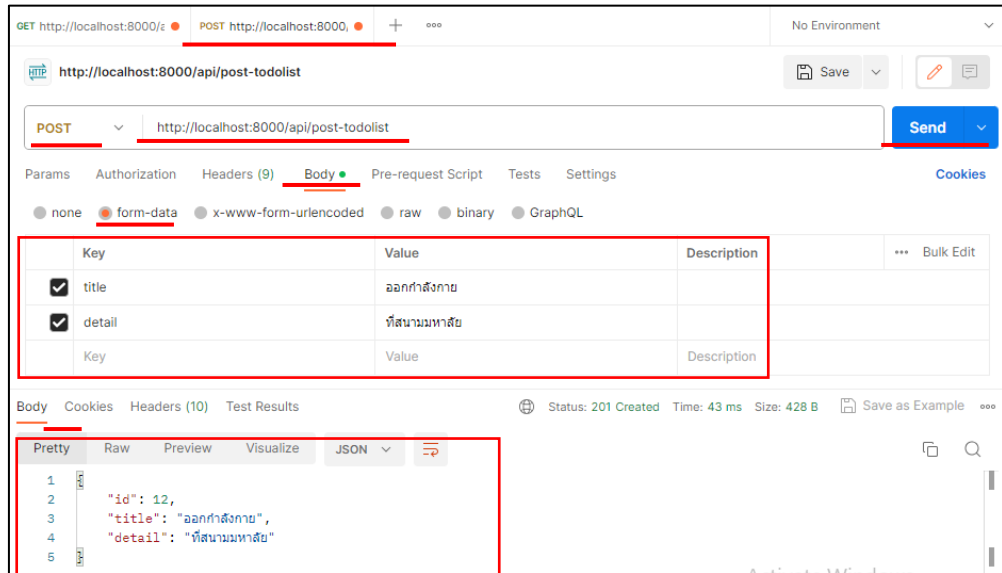
1 from django.urls import path
2 from .views import *
3 # * ดึงมาทุกฟังก์ชันใน views.py
4
5 urlpatterns = [
6     path('', Home),
7     path('api/all-todolist/', all_todoist), # localhost:8000/api/all-todolist
8     path('api/post-todolist', post_todoist),
9 ]

```

ภาพประกอบ 7.37 เปิดไฟล์ urls.py และแก้ไขโค้ด

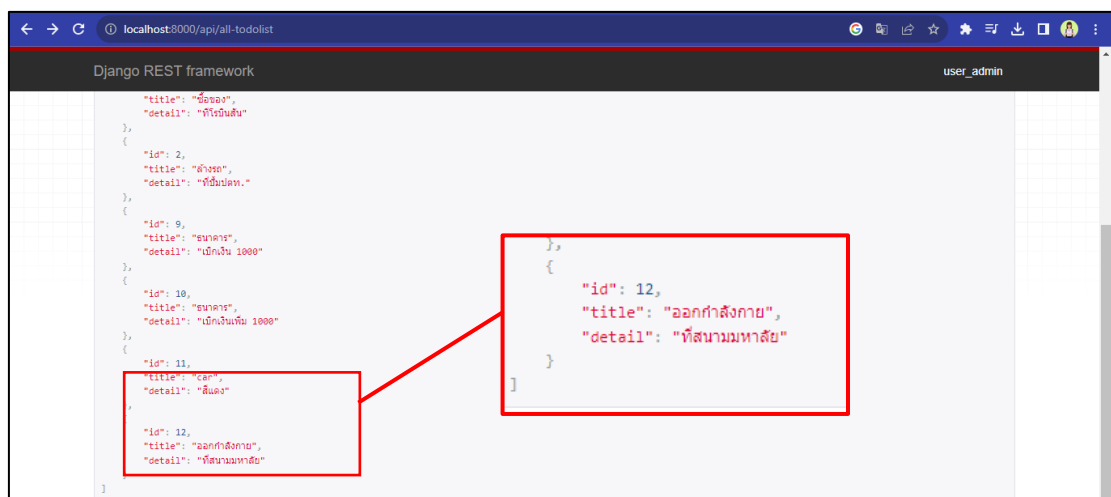
ที่มา : ฅนปักษ์ วรณตรง (2564 : 54)

10. กลับไปที่ Postman > กด + เพื่อเพิ่ม Tab ใหม่ > เลือก Post > ใส่ url > จากนั้นไปที่ Body > เลือก Form Data > กรอกข้อมูล > จากนั้นทำการกดส่ง Send ตามภาพ



ภาพประกอบ 7.38 แก้ไข Postman  
ที่มา : ณปภัช วรณตรง (2564 : 55)

11. เมื่อทำการกรอกข้อมูลและกดส่งเรียบร้อยแล้ว ให้กลับไป Django Rest Framework > กด Refresh จะพบว่า มีข้อมูลที่ได้ทำการเพิ่มเข้ามาใหม่ตามภาพ



ภาพประกอบ 7.39 มีข้อมูลที่เพิ่มเข้ามาใหม่  
ที่มา : ณปภัช วรณตรง (2564 : 56)



## บทสรุป

สำหรับเนื้อหาที่กล่าวมาทั้งหมดในบทนี้ จะพูดถึงการสร้างข้อมูลในฐานข้อมูล การทดสอบเข้าดูฐานข้อมูลที่มีใน DB Browser for SQLite รู้จัก Django REST Framework Django ซึ่งเป็น toolkit หรือไลบรารีของ Python ในการสร้าง RESTful APIs ข้อดี Django REST Framework HTTP Methods ที่ควรรู้ Status Codes ที่ควรรู้ วิธีการติดตั้ง Django REST Framework การสร้างไฟล์ Serializers เป็นไฟล์ที่มีลักษณะคล้าย model เป็นการระบุว่าจะให้แสดงข้อมูลไหนได้บ้างในหน้าต่าง API การติดตั้งโปรแกรม Postman และที่จะใช้ทดสอบการส่ง API ระหว่าง server กับ Flutter ด้วย Postman

## เอกสารอ้างอิง

django-rest-framework. (1 มกราคม 2566). **Django REST framework Overview**. สืบค้นจาก,  
<https://www.django-rest-framework.org/>

postman. (2 กุมภาพันธ์ 2566). **What is Postman?**. สืบค้นจาก,  
<https://www.postman.com/product/what-is-postman/>

sqlitebrowser. (10 กุมภาพันธ์ 2566). **DB Browser for SQLite**. สืบค้นจาก,  
<https://sqlitebrowser.org/>

## บทที่ 8

# การใช้งาน Firebase Database ร่วมกับ Flutter และตัวอย่างการประยุกต์ใช้

ในบทนี้ผู้เรียนจะได้เรียนรู้เพิ่มเติมในการใช้ Flutter ร่วมกับระบบฐานข้อมูลแบบอื่น ๆ ได้แก่ Firebase ซึ่งเป็นระบบฐานข้อมูลแบบ NoSQL เป็นแพลตฟอร์มของบริษัทกูเกิลที่มีเครื่องมือและบริการต่าง ๆ เพื่อช่วยให้นักพัฒนาสามารถสร้างแอปพลิเคชันแบบ Real-time ได้อย่างง่ายดายและรวดเร็ว โดยผู้เรียนจะได้เรียนรู้ตั้งแต่ นิยามของ Firebase ลักษณะของ Firebase Database การใช้งาน Flutter ร่วมกับ Firebase การติดตั้ง Library ในโปรเจกต์ Flutter การใช้งาน Cloud Firestore ซึ่งเป็นหนึ่งในบริการของ Firebase ช่วยจัดการเกี่ยวกับฐานข้อมูล การแสดงผลข้อมูลจาก Firestore Database การใช้งาน Firebase Authentication การใช้งาน image\_picker ร่วมกับ Storage และประยุกต์ใช้ความรู้ด้วยการสร้างแอปพลิเคชัน Chat ด้วย Flutter และ Firebase ดังรายละเอียดต่อไปนี้

## Firestore

### 1. Firestore คืออะไร

จิราวุธ วารินทร์ (2563 : 1) ได้อธิบายว่า Firestore เป็นบริการของ Google ที่เกี่ยวกับแบ็คเอนด์ (Backend Service) เช่น บริการด้านฐานข้อมูล (Cloud Firestore และ Realtime Database) บริการส่งการแจ้งเตือนไปยังผู้ใช้ (Cloud Message) บริการวิเคราะห์ประสิทธิภาพการทำงานของแอปพลิเคชัน (Performance Monitoring และ Google Analytics) บริการจัดเก็บและการเข้าใช้งานไฟล์ (Cloud Storage) และบริการอื่น ๆ

ศุภชัย สมพานิช (2563 : 177) ได้อธิบายว่า Firestore เป็นบริการประเภทหนึ่งที่ทำหน้าที่ทำงานอยู่ฝั่งหลังบ้าน (Backend) รองรับการพัฒนาทั้ง Mobile Apps และ Web Apps ภายในประกอบไปด้วยบริการต่าง ๆ มากมาย แต่จะขอให้บริการหลักเพียงตัวเดียวที่เรียกว่า Firestore เป็นระบบฐานข้อมูลที่เรียกว่า Realtime Database ส่งผลให้เป็นการแบ่งการทำงานออกเป็น 2 ฝั่ง คือ

- Frontend (หน้าบ้าน) คือ โปรเจกต์ Android Apps ที่สร้างขึ้น
- Backend (หลังบ้าน) คือ การขอใช้บริการ Firestore เข้ามาทำหน้าที่จัดเก็บข้อมูล

กล่าวได้อีกนัยหนึ่งว่า ฝั่งที่ส่วนนี้คือ APIs ที่ทำหน้าที่คอยให้บริการ Android Apps

กล่าวโดยสรุป Firestore คือ ชุดเครื่องมือและบริการที่ครอบคลุมการทำทั้งในส่วน Frontend และ Backend ซึ่งนำเสนอเป็นแพลตฟอร์ม Backend-as-aService (BaaS) ช่วยให้

นักพัฒนาสร้าง เปิดใช้และขยายการจัดการ ทั้งแอปพลิเคชันมือถือและเว็บไซต์ได้อย่างง่ายดาย มีฐานข้อมูลเรียลไทม์ การพิสูจน์ตัวตน พื้นที่เก็บข้อมูล Hosting และลักษณะอื่น ๆ อีกมากมาย พร้อมทั้งสามารถจัดการทั้งหมดได้จากแพลตฟอร์มเดียว

## 2. บริการด้านฐานข้อมูลของ Firebase

Firebase มีบริการเกี่ยวกับระบบ Backend อยู่หลายแบบ แต่บริการหลักของ Firebase คือ ระบบฐานข้อมูลแบบเรียลไทม์ โดยมี 2 รูปแบบ ให้เลือกใช้งาน ดังนี้ (จีราวุธ วารินทร์, 2563 : 1)

2.1 Realtime Database เป็นบริการฐานข้อมูล ที่สามารถจัดเก็บข้อมูลและซิงค์ข้อมูลทั้งหมดในแบบเรียลไทม์ Realtime Database จึงเหมาะสำหรับการใช้งานทั่วไป

2.2 Cloud Firestore หรือเรียกสั้น ๆ ว่า Firestore เป็นบริการฐานข้อมูลแบบใหม่ ใช้สำหรับจัดเก็บข้อมูลและอัปเดตข้อมูลแบบเรียลไทม์ ผนวกกับการค้นหาที่มีประสิทธิภาพ และสามารถปรับขนาดอัตโนมัติตามปริมาณข้อมูลที่ใช้งานที่เพิ่มขึ้น (Horizontal Scaling)

## 3. โครงสร้างข้อมูลของ Firestore

Cloud Firestore หรือเรียกสั้น ๆ ว่า Firestore เป็นบริการฐานข้อมูลในแบบ NOSQL ซึ่งพัฒนาโดย Google เป็นฐานข้อมูลในแบบเรียลไทม์ (Real Time) เมื่อข้อมูลในฐานข้อมูลมีการเปลี่ยนแปลงข้อมูลล่าสุด สามารถส่งไปยังไคลเอนต์เพื่ออัปเดตข้อมูลให้ใหม่โดยอัตโนมัติ โครงสร้างของ Firestore จะถูกแบ่งการจัดเก็บข้อมูลออกเป็น 3 ระดับ ได้แก่ ฐานข้อมูล (Database), คอลเล็กชัน (Collection) และเอกสาร (Document) (จีราวุธ วารินทร์, 2563 : 2)

จุดเด่นของ Firestore คือ เป็น ฐานข้อมูลในระบบคลาวด์จึงมีเสถียรภาพค่อนข้างสูง มีการใช้อินเด็กซ์ในการค้นข้อมูล นอกจากนั้นระบบคิวรีของ Firestore ยังถูกพัฒนาใหม่ ให้สามารถเลือกเฉพาะเอกสารที่ต้องการได้อย่างรวดเร็ว เป็นต้น (จีราวุธ วารินทร์, 2563 : 3)

## 4. ฐานข้อมูล NoSQL คืออะไร

ฐานข้อมูล NoSQL เป็นเทคโนโลยีที่เกิดขึ้นในปี ค.ศ. 1998 ปัจจุบันได้ถูกนำมาใช้ในแอปพลิเคชันต่าง ๆ แทนที่ฐานข้อมูลเชิงสัมพันธ์มากขึ้น เพื่อแก้ปัญหาและข้อจำกัดที่ไม่สามารถแก้ไขด้วยการใช้ฐานข้อมูลเชิงสัมพันธ์ เนื่องจากฐานข้อมูล NoSQL ถูกพัฒนาขึ้นเพื่อให้สามารถจัดการกับข้อมูลขนาดใหญ่ได้อย่างมีประสิทธิภาพและมีความยืดหยุ่น (เยาวภา, 2562 : 15)

ฐานข้อมูลเชิงสัมพันธ์ (Relational Database) เป็นฐานข้อมูลที่มีการจัดเก็บข้อมูลที่มีความสัมพันธ์กันไว้ด้วยกันในรูปของตาราง (Table) ตั้งแต่ 1 ตารางขึ้นไป ในแต่ละตารางจะ ประกอบด้วย แถว (Row) และคอลัมน์ (Column) โดยมีภาษาที่ใช้จัดการฐานข้อมูล คือ SQL (Structured Query Language) (เยาวภา, 2562 : 16)

ความแตกต่างระหว่างฐานข้อมูลเชิงสัมพันธ์และฐานข้อมูล NoSQL แสดงให้เห็นดังตารางต่อไป (เยาวภา, 2562 : 15-16)

ตาราง 8.1 ความแตกต่างระหว่างฐานข้อมูลเชิงสัมพันธ์และฐานข้อมูล NoSQL

ด้าน	ฐานข้อมูลเชิงสัมพันธ์	ฐานข้อมูล NoSQL
โครงสร้างฐานข้อมูล	เป็นแบบตาราง (Table)	เป็นได้ทั้งแบบคีย์ - ค่าของคีย์ (Key-value Store) - คอลัมน์ (Column-oriented Store) - เอกสาร (Document Database) - กราฟ (Graph Database)
การเก็บข้อมูล	เก็บในรูปแบบของตารางหลาย ๆ ตารางที่มีความสัมพันธ์กัน	เก็บในรูปแบบของ Key-value ซึ่งเป็นชุดของข้อมูลที่ประกอบด้วยคีย์ (Key) หรือชื่อที่ใช้ในการอ้างอิงถึงข้อมูลที่มีค่าไม่ซ้ำกัน และค่าของคีย์ (Value)
ภาษาที่ใช้จัดการฐานข้อมูล	SQL (Structured Query Language)	ไม่ใช่ภาษา SQL
การติดตั้งระบบบริหารจัดการฐานข้อมูล	ใช้เครื่องเซิร์ฟเวอร์ที่นำมาใช้ในการจัดเก็บข้อมูลที่มีราคาค่อนข้างแพง	ไม่จำเป็นต้องใช้เครื่องเซิร์ฟเวอร์ที่นำมาใช้ในการจัดเก็บข้อมูลที่มีราคาแพง
การขยายฐานข้อมูล เช่น การเพิ่มเครื่องเซิร์ฟเวอร์และการทำ Cluster	มีความยุ่งยาก	ทำได้ง่าย

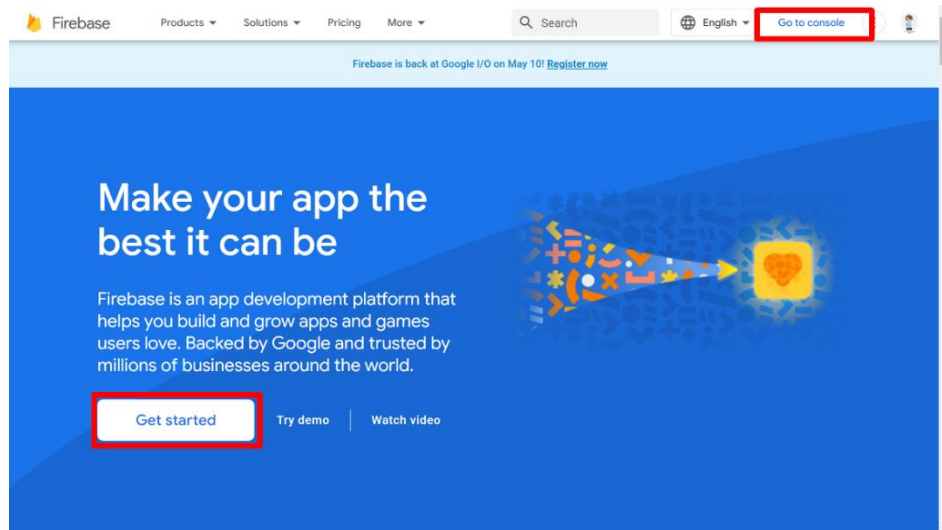
ตาราง 8.1 ความแตกต่างระหว่างฐานข้อมูลเชิงสัมพันธ์และฐานข้อมูล NoSQL (ต่อ)

ด้าน	ฐานข้อมูลเชิงสัมพันธ์	ฐานข้อมูล NoSQL
การเปลี่ยนแปลงใด ๆ กับโครงสร้างข้อมูล	มีผลกระทบต่อระบบงานทั้งระบบ	ไม่มีผลกระทบต่อระบบงานทั้งระบบ
การประมวลผลข้อมูลขนาดใหญ่ (Big Data)	ไม่ได้ถูกสร้างขึ้นมาเพื่อรองรับข้อมูลขนาดใหญ่	รองรับข้อมูลขนาดใหญ่ (Big Data) ข้อมูลขนาดใหญ่ ประมวลผลได้รวดเร็ว

## การใช้งาน Flutter ร่วมกับ Firebase

เนื้อหาส่วนนี้เป็นการเริ่มต้นใช้งาน Firebase สร้างโปรเจกต์ Firebase และการเพิ่ม Firebase ในโปรเจกต์ Flutter ดังมีรายละเอียดขั้นตอนดังนี้

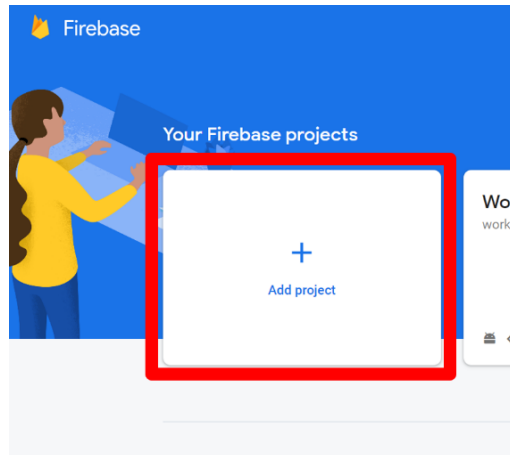
1. ให้ไปที่เว็บไซต์ <https://firebase.google.com/>
2. ทำการล็อกอินเข้าใช้งาน โดยใช้ e-mail ของ Google เมื่อทำการล็อกอินสำเร็จแล้ว ให้ทำการเริ่มต้นใช้งาน Firebase โดยกดที่ปุ่ม Get Started หรือปุ่ม Go to Console ดังภาพประกอบ 8.2



ภาพประกอบ 8.1 การเริ่มต้นใช้งาน Firebase

ที่มา : Firebase. (2020 : 1)

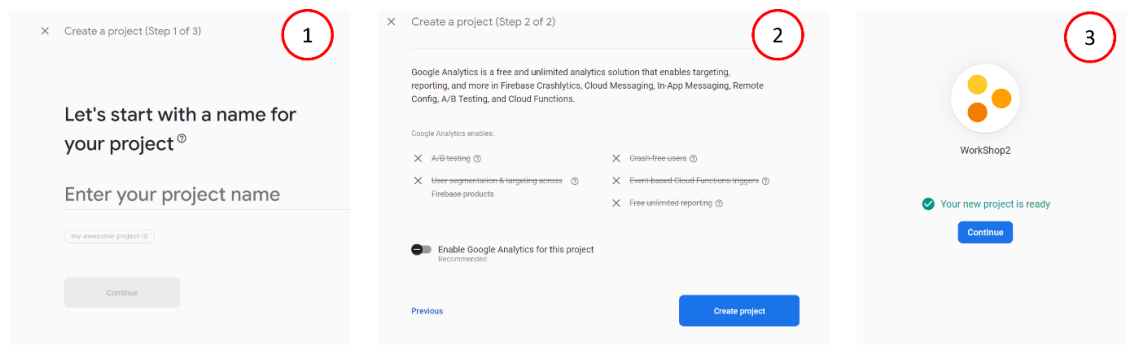
3. เริ่มต้นการสร้างโปรเจกต์ในการใช้งาน Firebase ดังภาพประกอบ 8.3
  - 3.1 กดที่ Add project หลังจากนั้นตั้งชื่อโปรเจกต์ของ Firebase
  - 3.2 สามารถเลือกเปิดหรือปิดการใช้งาน Google Analytics ของโปรเจกต์ Firebase



ภาพประกอบ 8.2 การเริ่มสร้างโปรเจกต์ Firebase

ที่มา : Firebase. (2020 : 1)

- 3.3 เมื่อตั้งชื่อ และตั้งค่าการใช้งาน Google Analytics เรียบร้อยแล้วรอเวลาสร้างโปรเจกต์ หากสร้างเสร็จจะขึ้นดังภาพประกอบ 8.4



ภาพประกอบ 8.3 การสร้างโปรเจกต์ Firebase สำเร็จ

ที่มา : Firebase. (2020 : 1)

3.4 เริ่มต้นการเพิ่ม Firebase ในโปรเจกต์ Flutter เนื่องจาก Flutter เป็น hybrid app สามารถเพิ่มได้ทั้ง Android และ iOS ดังภาพประกอบ 8.5

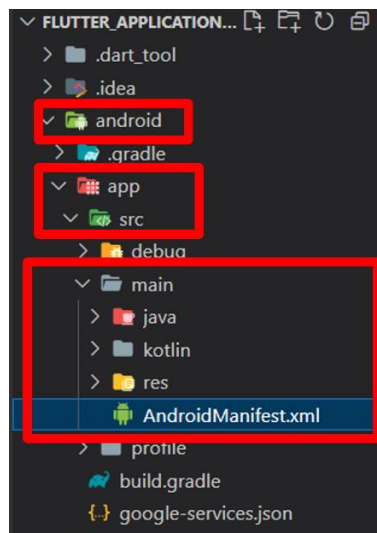


ภาพประกอบ 8.4 การสร้างโปรเจกต์ Firebase

ที่มา : Firebase. (2020 : 1)

โดยจะแสดงขั้นตอนการเพิ่มแอปพลิเคชันในฝั่ง Android การเพิ่มสามารถเลือกคลิกที่ไอคอนของแต่ละระบบปฏิบัติการที่ต้องการใช้งานดังขั้นตอนต่อไปนี้

3.4.1 การหา Package Name จำเป็นต้องสร้างโปรเจกต์ Flutter หรือโปรเจกต์ Flutter เดิมที่เคยสร้างไว้ เพื่อนำไปลงทะเบียนไว้บน Firebase โดยจะหา Package Name ได้ที่ไฟล์ Android/app/src/main/AndroidManifest.xml ของโปรเจกต์ Flutter ดังภาพประกอบ 8.6



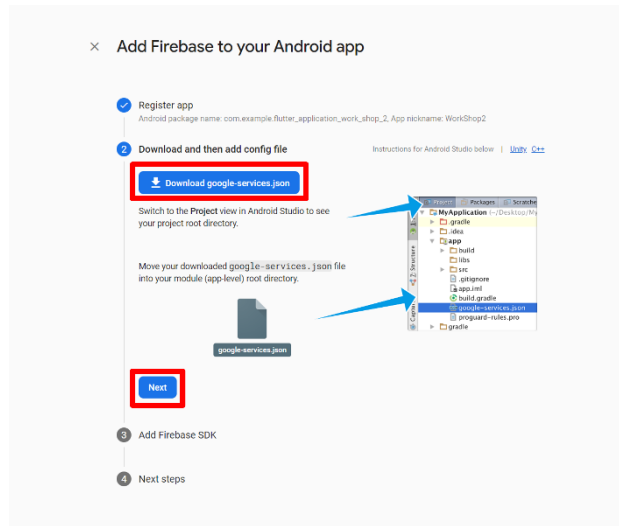
ภาพประกอบ 8.5 Android->app->src->main->AndroidManifest.xml

ที่มา : ณปภัช วรณตรง (2564 : ...)





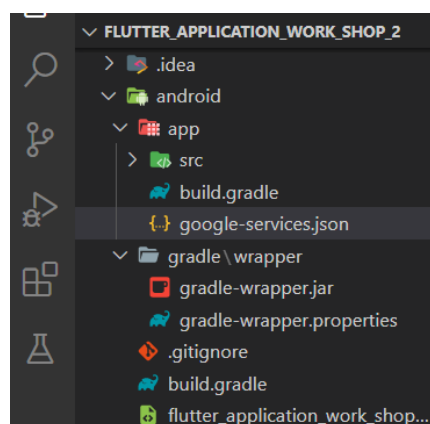
3.4.2 เมื่อลงทะเบียนแอปพลิเคชันเสร็จ ให้ดาวน์โหลดไฟล์ google-services.json เอาไว้ และกดถัดไป ดังภาพประกอบ 8.9



ภาพประกอบ 8.8 การดาวน์โหลดไฟล์ google-services.json  
ที่มา : Firebase. (2020 : 1)

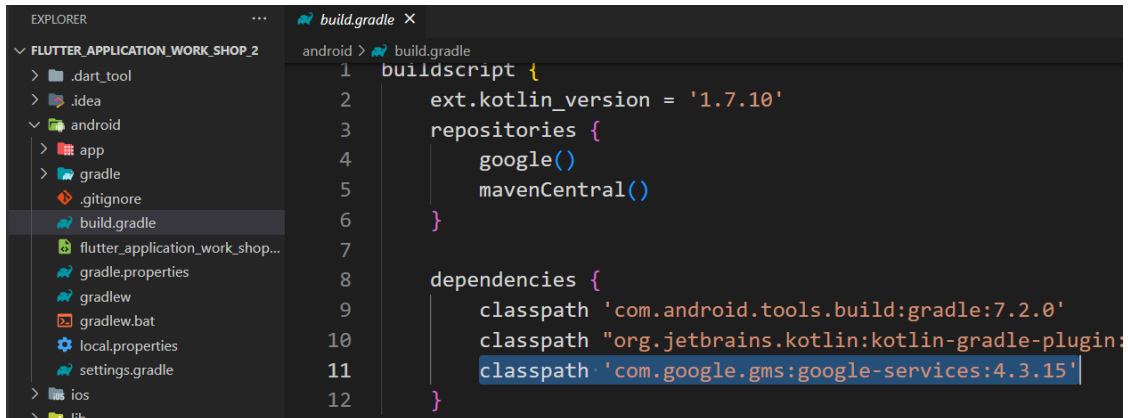
3.4.2.1 กดดาวน์โหลด google-service.json

3.4.2.2 กดถัดไป และลากไฟล์ google-services.json ที่ดาวน์โหลด ไปวางไว้ในโปรเจกต์ Flutter ในไฟล์ Android/app



ภาพประกอบ 8.9 การวางไฟล์ google-services.json ใน Android/app  
ที่มา : ฅนปักษ์ วรรณตรง (2564 : ...)

3.4.3 เมื่อวางไฟล์ google-services.json ใน Android/app เสร็จเรียบร้อยให้ไปที่ไฟล์ android\build.gradle หาแท็ก dependencies และวาง classpath 'com.google.gms:google-services:4.3.15' ลงไป ดังภาพประกอบ 8.11



```

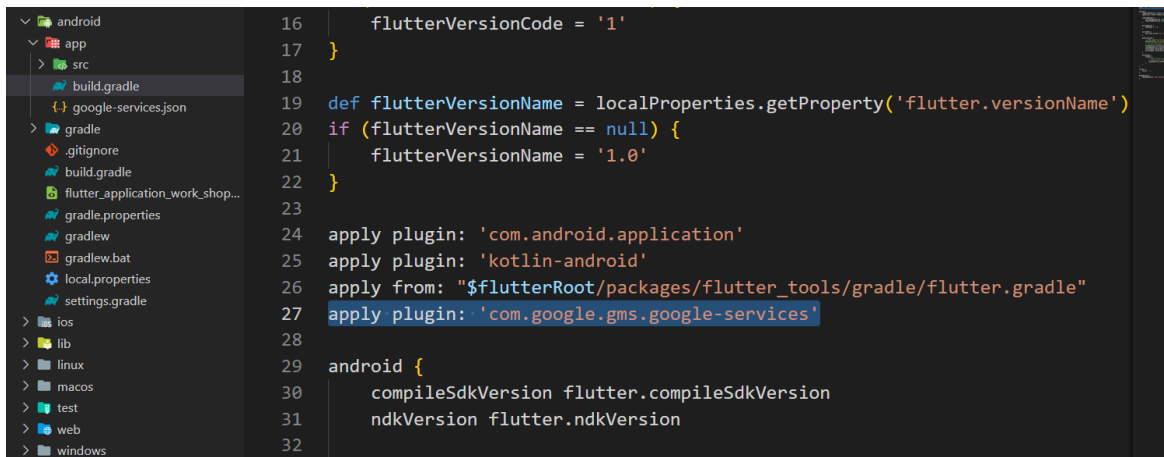
1  buildscript {
2      ext.kotlin_version = '1.7.10'
3      repositories {
4          google()
5          mavenCentral()
6      }
7
8      dependencies {
9          classpath 'com.android.tools.build:gradle:7.2.0'
10         classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
11         classpath 'com.google.gms:google-services:4.3.15'
12     }

```

ภาพประกอบ 8.10 การวาง classpath 'com.google.gms:google-services: 4.3.15'

ที่มา : ฌปภัช วรณตรง (2564 : ...)

จากนั้นต้อง apply plugin โดยไปที่ android\app\build.gradle เพิ่ม apply plugin: 'com.google.gms.google-services' ดังภาพ



```

16     flutterVersionCode = '1'
17 }
18
19 def flutterVersionName = localProperties.getProperty('flutter.versionName')
20 if (flutterVersionName == null) {
21     flutterVersionName = '1.0'
22 }
23
24 apply plugin: 'com.android.application'
25 apply plugin: 'kotlin-android'
26 apply from: "$flutterRoot/packages/flutter_tools/gradle/flutter.gradle"
27 apply plugin: 'com.google.gms.google-services'
28
29 android {
30     compileSdkVersion flutter.compileSdkVersion
31     ndkVersion flutter.ndkVersion
32

```

ภาพประกอบ 8.11 การ apply plugin

ที่มา : ฌปภัช วรณตรง (2564 : ...)

ไปที่ android\app\build.gradle หาแท็ก dependencies และวาง implementation platform('com.google.firebase:firebase-bom:31.2.3') ลงไปเพื่อเปิดการใช้งานปลั๊กอินและบริการต่าง ๆ ของ Firebase ดังภาพ

```

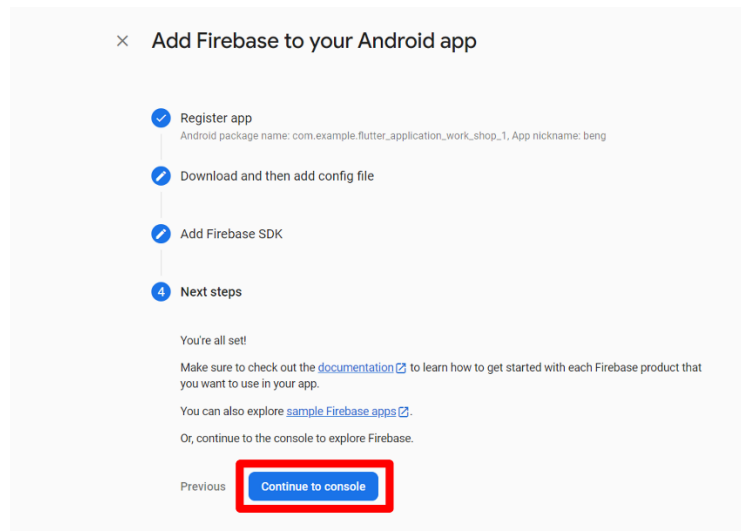
65
66 flutter {
67     source '../..'
68 }
69
70 dependencies {
71     implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
72     implementation platform('com.google.firebase:firebase-bom:31.2.3')
73 }

```

ภาพประกอบ 8.12 การวาง implementation platform

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

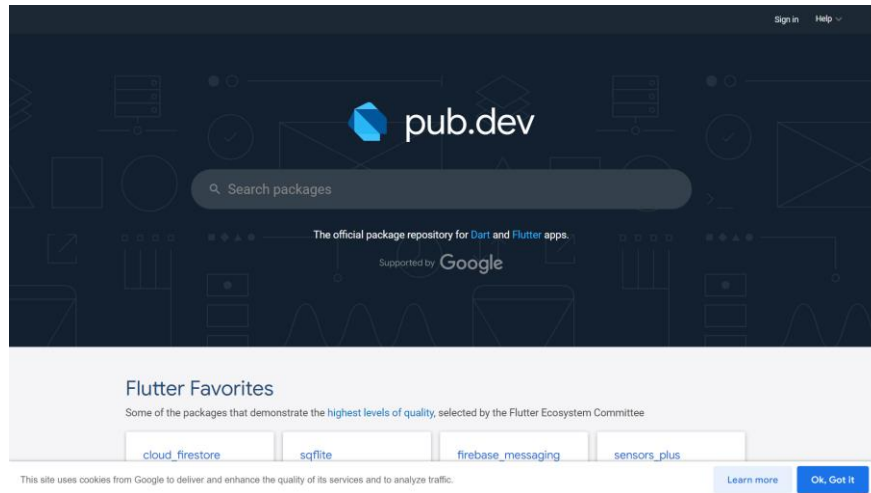
เมื่อกรอกข้อมูลครบแล้ว ให้กด Continue to console การเพิ่ม Firebase ในแอป Android ก็เสร็จเรียบร้อยแล้ว ดังภาพ



ภาพประกอบ 8.13 การเพิ่ม Firebase ในแอป Android เมื่อเพิ่มข้อมูลครบถ้วน

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

3.5 การใช้งาน Firebase มีบริการหลายอย่างมาก ๆ ซึ่งการใช้งานในบริการต่าง ๆ ของ Firebase จำเป็นต้องติดตั้ง Library ในการใช้งาน โดยหา Library ได้จาก <https://pub.dev/> ดังภาพ



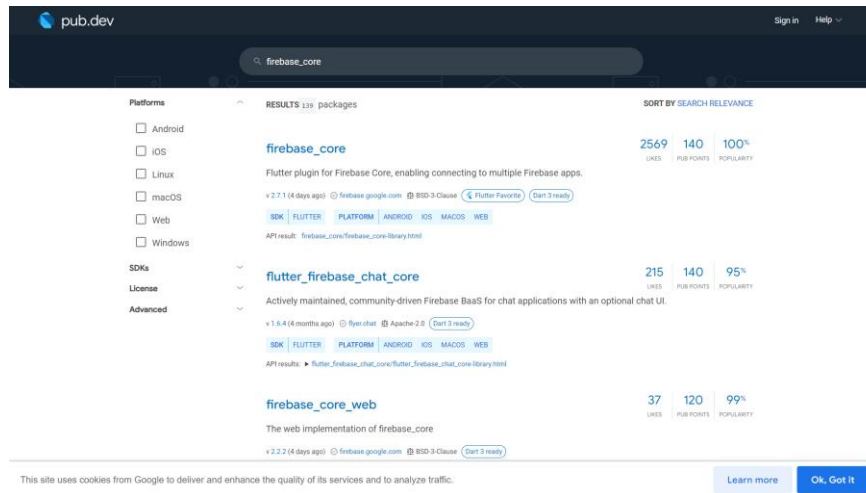
ภาพประกอบ 8.14 เว็บไซต์ pub.dev

ที่มา : Pub.Dev., (2020 : 1)

### การติดตั้ง Library ในโปรเจกต์ Flutter

เนื้อหาส่วนนี้จะเป็นการติดตั้ง Library ที่จะทำให้ Flutter และ Firebase สามารถทำงานร่วมกันได้ ดังมีรายละเอียดขั้นตอนดังนี้

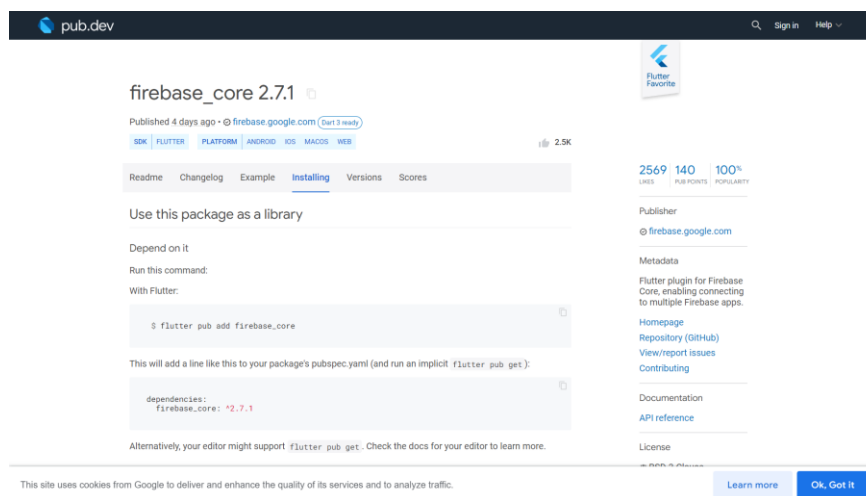
1. กรอกชื่อ Library ที่ต้องการใช้งานแล้วกดค้นหา ผลการค้นหา ซึ่งจะมี Library ให้เลือกใช้งานมากมาย ดังภาพ



ภาพประกอบ 8.15 ผลการค้นหา Library

ที่มา : Pub.Dev., (2020 : 1)

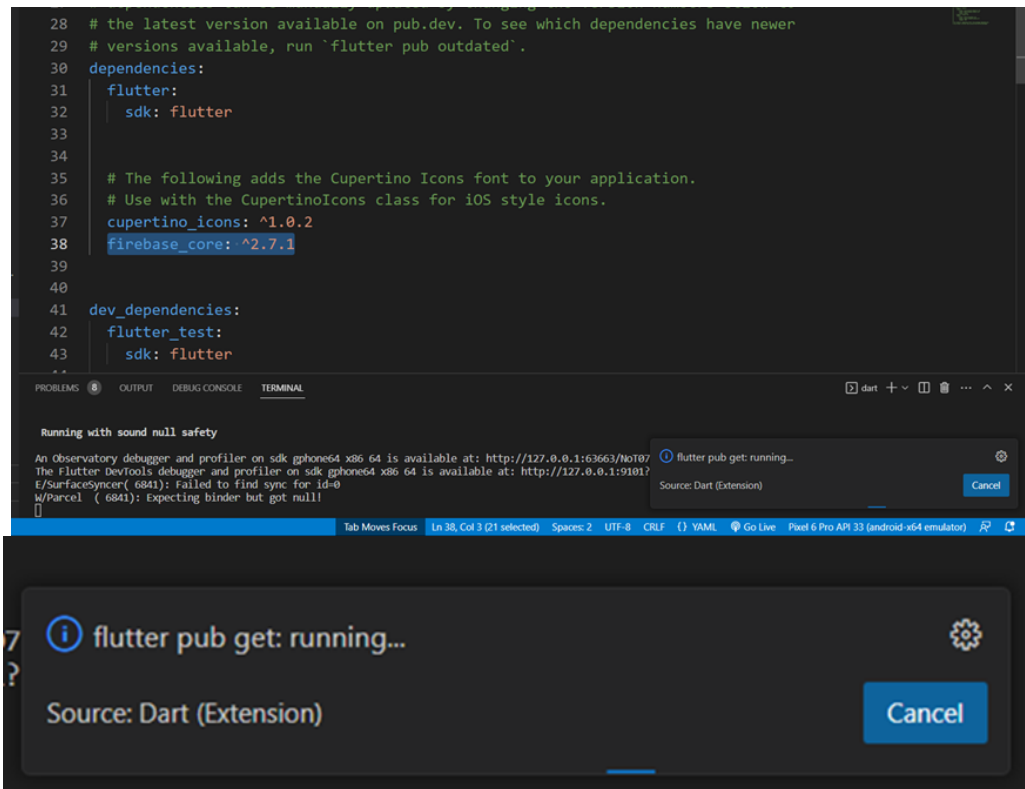
2. จากนั้นให้ทำการคลิกที่ Library ที่ต้องการ ไปที่แท็บเมนู Installing แล้วเลื่อนหา firebase\_core: ^2.7.1 แล้วทำการ Copy ไว้ ดังภาพ



ภาพประกอบ 8.16 Library ที่ต้องการใช้งาน

ที่มา : Pub.Dev., (2020 : 1)

3. ไปที่ไฟล์ pubspec.yaml แล้วนำ firebase\_core: ^2.7.1 ไปวางไว้ แล้วทำการกดบันทึก  
จะขึ้น flutter pub get :running ที่มุมขวาล่างของ VS Code แล้วทำการเปิด Imurater หรือ  
โปรแกรมจำลอง Android ทำการสั่งการทำงานโปรเจกต์เพื่อดูว่า Library ที่ติดตั้งไปสามารถใช้งานกับ  
ตัวโปรเจกต์ได้หรือไม่ เมื่อใช้งานได้แล้วการติดตั้ง Library ก็ถือว่าเสร็จสิ้น ดังภาพ



```

28 # the latest version available on pub.dev. To see which dependencies have newer
29 # versions available, run `flutter pub outdated`.
30 dependencies:
31   flutter:
32     sdk: flutter
33
34
35 # The following adds the Cupertino Icons font to your application.
36 # Use with the CupertinoIcons class for iOS style icons.
37 cupertino_icons: ^1.0.2
38 firebase_core: ^2.7.1
39
40
41 dev_dependencies:
42   flutter_test:
43     sdk: flutter

```

Running with sound null safety  
An Observatory debugger and profiler on sdk gphone64 x86 64 is available at: http://127.0.0.1:63663/No10?  
The Flutter DevTools debugger and profiler on sdk gphone64 x86 64 is available at: http://127.0.0.1:9101?  
E/SurfaceSynchronizer( 6841): Failed to find sync for id=0  
W/Parcel ( 6841): Expecting binder but got null!

flutter pub get running...  
Source: Dart (Extension)  
Cancel

ภาพประกอบ 8.17 การติดตั้ง Library  
ที่มา : ฅนปักษ์ วรณตรง (2564 : ...)

หากสั่งการทำงาน และสามารถเปิดหน้าแอปพลิเคชันได้ตามปกติ ไม่ error สามารถเขียน code เรียกใช้งาน Library ได้ ดังภาพ

```

lib > main.dart > MyApp
1 import 'package:flutter/material.dart';
2
3 Run | Debug | Profile
4 void main() {
5   runApp(const MyApp());
6 }
7
8 class MyApp extends StatelessWidget {
9   const MyApp({super.key});
10
11 // This widget is the root of your application.
12 @override
13 Widget build(BuildContext context) {

```

```

PS C:\Users\thinn\Music\Flutter Project\flutter_application_work_shop_2> flutter run
Using hardware rendering with device sdk gphone64 x86 64. If you notice graphics artifact
with "--enable-software-rendering".
Launching lib/main.dart on sdk gphone64 x86 64 in debug mode...
Parameter format not correct -
Running Gradle task 'assembleDebug'... 116.3s
✓ Built build/app/outputs/flutter-apk/app-debug.apk.
Installing build/app/outputs/flutter-apk/app-debug.apk... 1,811ms
Syncing files to device sdk gphone64 x86 64... 1,362ms

Flutter run key commands.
r Hot reload.
R Hot restart.
h list all available interactive commands.
d Detach (terminate "flutter run" but leave application running).
c Clear the screen.
q Quit (terminate the application on the device).

Running with sound null safety

An Observatory debugger and profiler on sdk gphone64 x86 64 is available at: http://127.0.0.1:56780/wT4xI05G9M=/
The Flutter DevTools debugger and profiler on sdk gphone64 x86 64 is available at:

```

ภาพประกอบ 8.18 การ run หลังการติดตั้ง Library สำเร็จ

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

หากมีการติดตั้ง Library เพิ่มเติมให้สั่งการทำงานแอปพลิเคชันใหม่ทุกครั้ง

## การใช้งาน Cloud Firestore

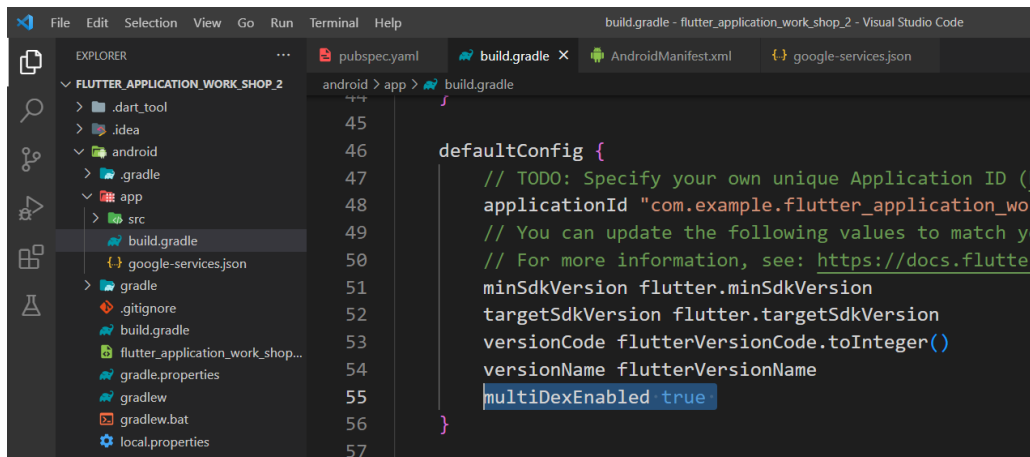
Cloud Firestore หรือเรียกสั้น ๆ ว่า Firestore เป็นบริการฐานข้อมูลแบบใหม่ ใช้สำหรับจัดเก็บข้อมูลและอัปเดตข้อมูลแบบเรียลไทม์ ผสมกับการค้นหาที่มีประสิทธิภาพ และสามารถปรับขนาดอัตโนมัติตามปริมาณข้อมูลที่ใช้งานที่เพิ่มขึ้น (Horizontal Scaling) (จิราวุธ วารินทร์, 2563 : 1) ในเนื้อหาส่วนนี้จะมาเรียนรู้การใช้งาน Cloud Firestore เพื่อสร้างฐานข้อมูลสำหรับเก็บข้อมูล มีรายละเอียดขั้นตอนดังนี้

1. เริ่มต้นด้วยการติดตั้ง Library ดังต่อไปนี้

1.1 firebase\_core

1.2 cloud\_firestore

การใช้งาน cloud\_firestore ถ้าเจอ ERROR: Cannot fit requested classed in a single dex file แก้ด้วยการเพิ่ม multiDexEnabled true ใน android -> app -> build.gradle ในแท็บ defaultConfig



```

45
46
47 defaultConfig {
48     // TODO: Specify your own unique Application ID (!)
49     applicationId "com.example.flutter_application_work_shop_2"
50     // You can update the following values to match your application
51     // For more information, see: https://docs.flutter.dev/development/platform-integration/android/build-configuration#application-id
52     minSdkVersion flutter.minSdkVersion
53     targetSdkVersion flutter.targetSdkVersion
54     versionCode flutterVersionCode.toInteger()
55     versionName flutterVersionName
56     multiDexEnabled true
57 }

```

ภาพประกอบ 8.19 การเพิ่ม multiDexEnabled true

ที่มา : ณปภัช วรณตรง (2564 : ...)



ถ้าเจอ Error:uses-sdk:minSdkVersion 16 cannot be smaller than version 19 declared in Library [:cloud\_firestore] ตั้งภาพ

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\thinn\Music\Flutter Project\Flutter_application_work_shop_2> flutter run
Using hardware rendering with device sdk gphone64 x86_64. If you notice graphics artifacts, consider enabling software rendering with
"--enable-software-rendering".
Launching lib/main.dart on sdk gphone64 x86_64 in debug mode...
Parameter format not correct -
C:\Users\thinn\Music\Flutter Project\Flutter_application_work_shop_2\android\app\src\debug\AndroidManifest.xml Error:
  uses-sdk:minSdkVersion 16 cannot be smaller than version 19 declared in library [:cloud_firestore] C:\Users\thinn\Music\Flutter Project\Flutter_application_work_shop_2\build\cloud_firestore\intermediates\merged_manifest\debug\AndroidManifest.xml as the library might be using APIs not available in 16
  Suggestion: use a compatible library with a minSdk of at most 16,
    or increase this project's minSdk version to at least 19,
    or use tools:overrideLibrary="io.flutter.plugins.firebase.firestore" to force usage (may lead to runtime failures)

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':app:processDebugMainManifest'.
> Manifest merger failed : uses-sdk:minSdkVersion 16 cannot be smaller than version 19 declared in library [:cloud_firestore] C:\Users\thinn\Music\Flutter Project\Flutter_application_work_shop_2\build\cloud_firestore\intermediates\merged_manifest\debug\AndroidManifest.xml as the library might be using APIs not available in 16
  Suggestion: use a compatible library with a minSdk of at most 16,
    or increase this project's minSdk version to at least 19,
    or use tools:overrideLibrary="io.flutter.plugins.firebase.firestore" to force usage (may lead to runtime failures)

* Try:
> Run with --stacktrace option to get the stack trace.
> Run with --info or --debug option to get more log output.
Fix this issue by adding the following to the file C:\Users\thinn\Music\Flutter Project\Flutter_application_work_shop_2\android\app\build.gradle:
    android {
        defaultconfig {
            minSdkVersion 19
        }
    }

Note that your app won't be available to users running Android SDKs below 19.
Alternatively, try to find a version of this plugin that supports these lower versions of the Android SDK.
For more information, see: https://docs.flutter.dev/deployment/android#reviewing-the-gradle-build-configuration

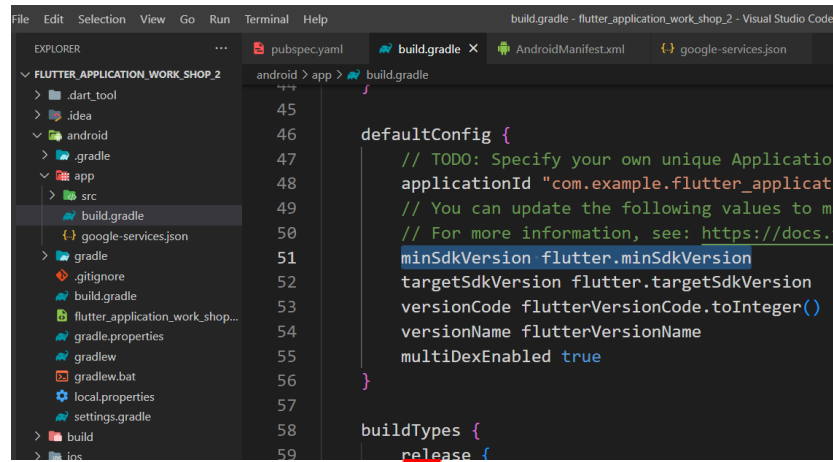
Exception: Gradle task assembleDebug failed with exit code 1
PS C:\Users\thinn\Music\Flutter Project\Flutter_application_work_shop_2>

```

ภาพประกอบ 8.20 Error:uses-sdk:minSdkVersion

ที่มา : ฅนปักษ์ วรณตรง (2564 : ...)

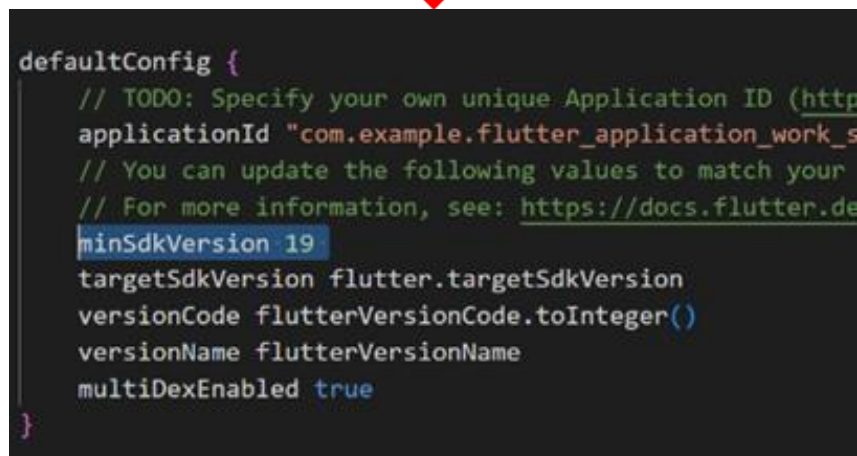
ให้ทำการไปแก้ไข minSdkVersion flutter.minSdkVersion ที่แท้ที่ defaultConfig ให้เป็น minSdkVersion 19 ดังภาพ



```

45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
defaultConfig {
    // TODO: Specify your own unique Application
    applicationId "com.example.flutter_applicat
    // You can update the following values to m
    // For more information, see: https://docs.
    minSdkVersion flutter.minSdkVersion
    targetSdkVersion flutter.targetSdkVersion
    versionCode flutterVersionCode.toInteger()
    versionName flutterVersionName
    multiDexEnabled true
}
buildTypes {
    release {

```

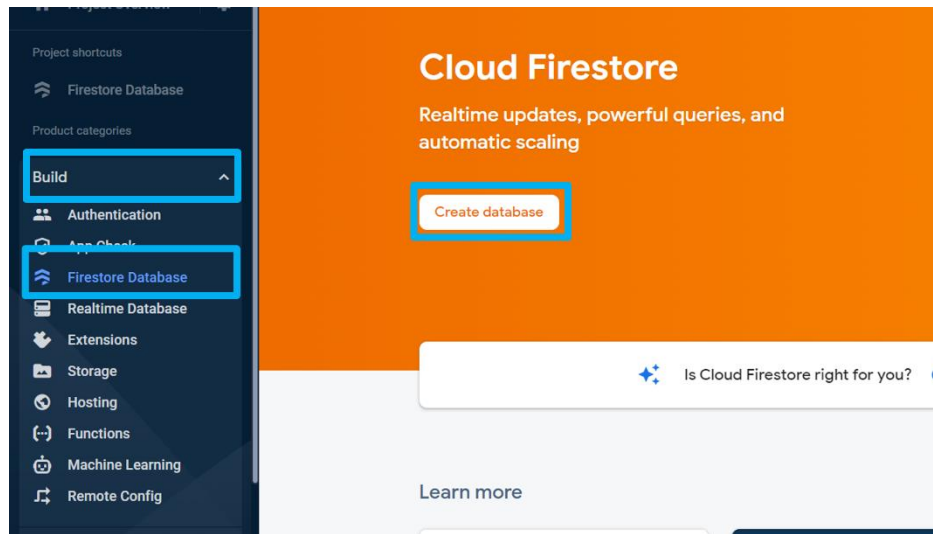
```

defaultConfig {
    // TODO: Specify your own unique Application ID (http
    applicationId "com.example.flutter_application_work_s
    // You can update the following values to match your
    // For more information, see: https://docs.flutter.de
    minSdkVersion 19
    targetSdkVersion flutter.targetSdkVersion
    versionCode flutterVersionCode.toInteger()
    versionName flutterVersionName
    multiDexEnabled true
}

```

ภาพประกอบ 8.21 การเพิ่ม minSdkVersion  
ที่มา : ฌปภัช วรณตรง (2564 : ...)

2. เมื่อติดตั้ง Library สำเร็จแล้ว ให้ไปที่ Firebase โพรเจกต์ เพื่อทำการสร้างฐานข้อมูล Firestore Database ดังภาพ
  - 2.1 ไปที่ build
  - 2.2 เลือก Firestore Database
  - 2.3 กด Create database

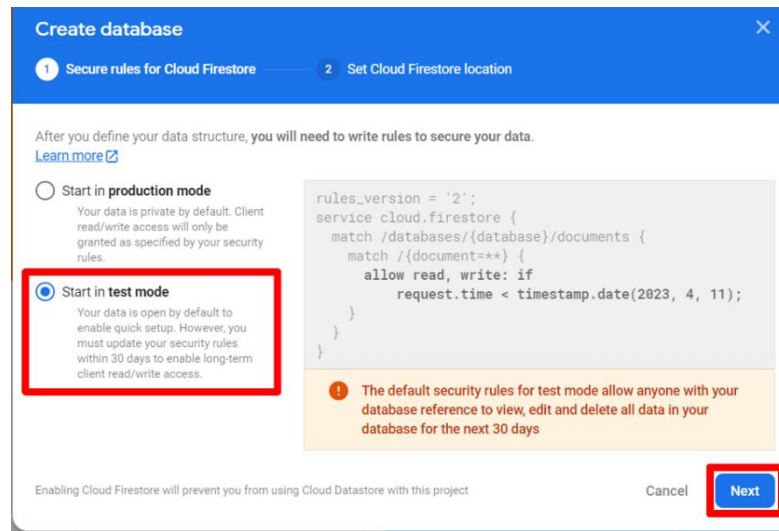


ภาพประกอบ 8.22 การเริ่มต้นสร้าง Firestore Database  
ที่มา : Firebase (2020 : 1)

3. หลังจากที่ทำกรสร้าง Firestore Database จะแสดงดั่งภาพประกอบ 8.24 - 8.25 ให้ทำตามขั้นตอนดั่งต่อไปนี้

3.1 เลือก test โหมด

3.2 กด Next ดั่งภาพ

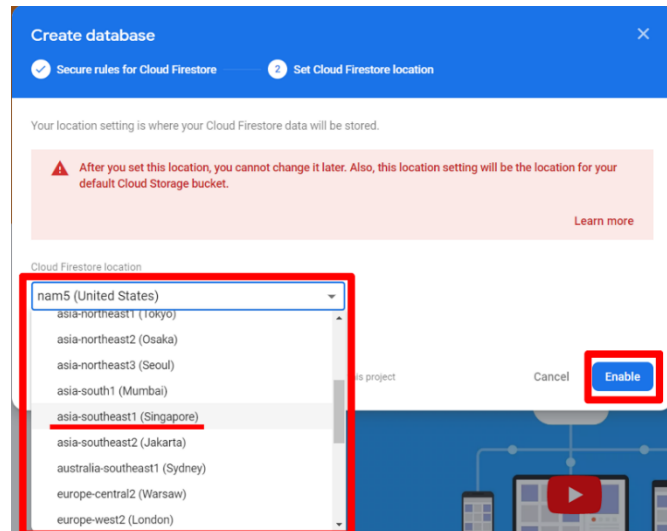


ภาพประกอบ 8.23 การตั้งค่าเริ่มต้นให้กับ Firestore Database

ที่มา : Firebase (2020 : 1)

3.3 เลือกที่ตั้งในการเก็บข้อมูล ของ Firestore Database ของโปรเจกต์

3.4 กด Enable ดั่งภาพ



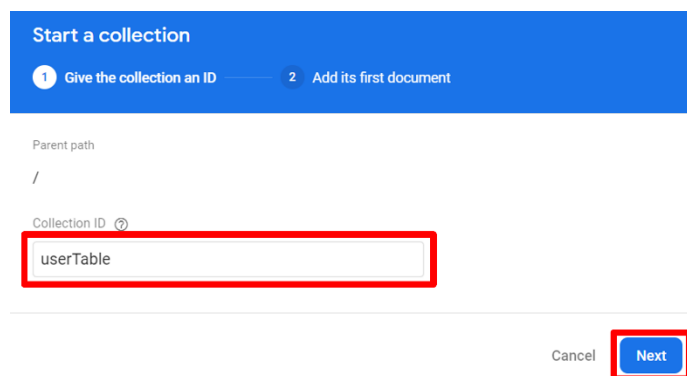
ภาพประกอบ 8.24 การเลือกที่ตั้งในการเก็บข้อมูล Firestore Database  
ที่มา : Firebase (2020 : 1)

### 3.5 ทำการเพิ่ม collection เพิ่มข้อมูลลงไป

#### 3.5.1 Start collection

#### 3.5.2 ใส่ชื่อ collection ลงไป

#### 3.5.3 กด Next ดังภาพ



ภาพประกอบ 8.25 การเพิ่ม collection

ที่มา : Firebase (2020 : 1)

### 3.6 ทำการเพิ่มข้อมูลลงไป ใน document

#### 3.6.1 ใส่ชื่อ Field ลงไป

#### 3.6.2 ระบุประเภทข้อมูลที่ต้องการเก็บ

#### 3.6.3 ระบุข้อมูลที่ต้องการเก็บ

#### 3.6.4 กด Save ดึงภาพ

Dialog: Add a document

Parent path: /userTable

Document ID: pyCoYgGF2KdsfgGgXGvZBtCnNBP2

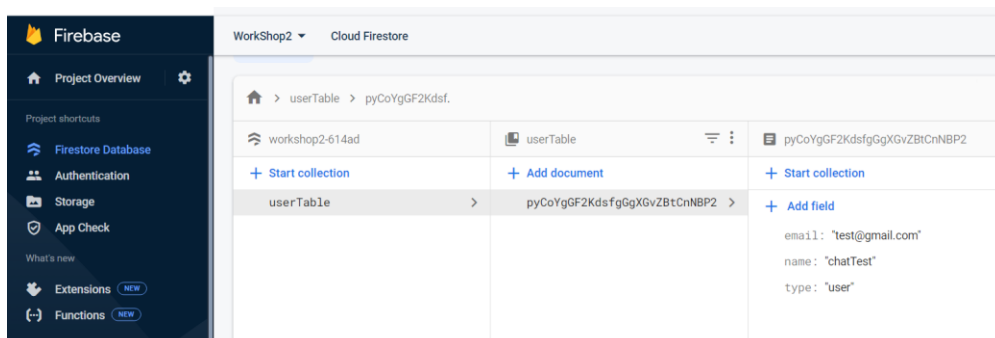
Field	Type	Value
name	string	chatTest
email	string	test@gmail.com
type	string	user

Buttons: Cancel, Save

### ภาพประกอบ 8.26 การเพิ่มข้อมูลลงไป ใน collection

ที่มา : Firebase (2020 : 1)

3.7 เมื่อกรอก Field, Type และ Value เสร็จแล้วจะได้ดึงภาพ จากนั้นกดถัดไป ซึ่งจะได้ข้อมูลดังภาพ



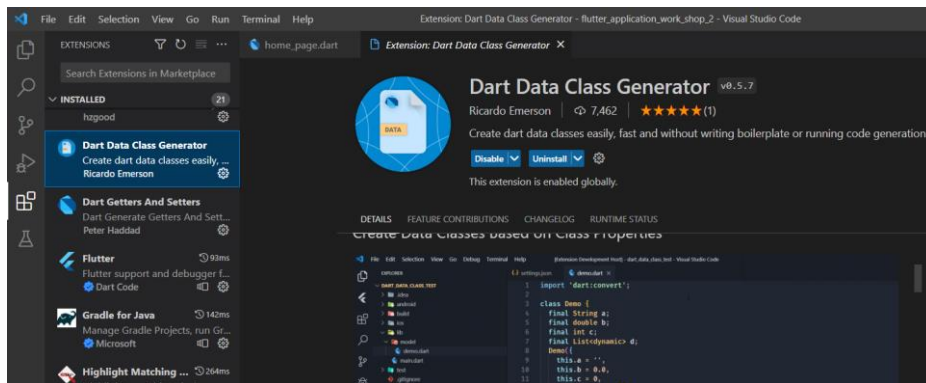
### ภาพประกอบ 8.27 ผลที่เพิ่มลงไป ใน collection

ที่มา : Firebase (2020 : 1)

## การแสดงผลข้อมูลจาก Firestore Database

เนื้อหาส่วนนี้จะเป็นการสร้าง Model ในการอ่านค่าข้อมูลที่ได้จาก Firestore Database มาเก็บไว้ใน Model ที่สร้างไว้ ดังมีรายละเอียดขั้นตอนดังนี้

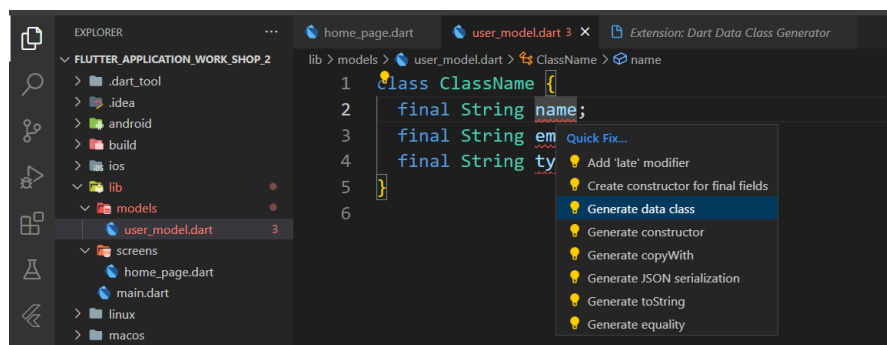
1. สิ่งแรกที่ต้องมีในการสร้าง Model คือ Extension Dart Data Class Generator ด้วยการค้นหาใน Vs Code และติดตั้ง



ภาพประกอบ 8.28 Extension Dart Data Class Generator

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

2. เมื่อลง Extension เรียบร้อยแล้ว ทำการปิดและเปิด Vs Code ใหม่ ทำการสร้างไฟล์เดอร์ Models ของในไฟล์เดอร์ lib เสร็จเรียบร้อยให้ทำการสร้างไฟล์ข้างในไฟล์เดอร์ Models ชื่อไฟล์ user\_Model.dart ทำการสร้าง Class ประกาศตัวแปรที่การอ่านข้อมูลมาเก็บไว้ แล้วทำการกดที่หลอดไฟ หรือนำเมาส์คลิกตัวแปรที่ประกาศที่ error กด Ctrl+จุด แล้วเลือก Generate data class



ภาพประกอบ 8.29 การสร้าง Model

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

เมื่อทำการ Generate data class เรียบร้อยแล้วจะแสดงดังภาพ

```

1 import 'dart:convert';
2
3 class ClassName {
4   final String name;
5   final String email;
6   final String type;
7   ClassName({
8     required this.name,
9     required this.email,
10    required this.type,
11  });
12
13  ClassName copyWith({
14    String? name,
15    String? email,
16    String? type,
17  }) {
18    return ClassName(
19      name: name ?? this.name,
20      email: email ?? this.email,
21      type: type ?? this.type,
22    );
23  }
24
25  Map<String, dynamic> toMap() {
26    final result = <String, dynamic>{};
27
28    result.addAll({'name': name});
29    result.addAll({'email': email});
30    result.addAll({'type': type});
31
32    return result;
33  }

```

ภาพประกอบ 8.30 Model ที่ Generate แล้ว

ที่มา : ณปภัช วรรณตรง (2564 : ...)

3. หลังจากที่เราสร้าง Model เสร็จเรียบร้อยแล้ว มาเริ่มต้นเขียน code ในการอ่านข้อมูลจาก Firestore Database มาเก็บไว้ใน Model ที่สร้างไว้ มีขั้นตอนการเขียน code ดังนี้

3.1 เริ่มต้นด้วยการประกาศตัวแปรในรูปแบบ List เพราะ Model ที่สร้างไว้ในการอ่านข้อมูลออกมาจาก Firestore Database มีข้อมูลมากกว่า 1 ตำแหน่ง

3.2 สร้าง initState method เป็น method ที่จะทำงานก่อนทุกครั้งที่เราเข้ามาถึงหน้าแอปพลิเคชันที่ทำการสร้าง method นี้ไว้

3.3 สร้าง Future Function คือ function app asynchronous จะดำเนินการสั่งการทำงานโปรแกรมทีละชุดคำสั่ง และจะส่งการทำงานชุดคำสั่งถัดไปทันทีโดยไม่ต้องรอชุดคำสั่งก่อนหน้าทำงานเสร็จ ตัวอย่างเช่น ถ้าโปรแกรมเรียกฟังก์ชัน A(); และ B(); ตามลำดับ โปรแกรมจะรัน



ฟังก์ชัน A()); และ B()); ตามลำดับโดยไม่สนใจว่าฟังก์ชัน A()); จะทำงานเสร็จหรือยัง จะไปเรียกฟังก์ชัน B()); ต่อเลยทันที

3.4 เขียน Code เรียกใช้งาน Library ของ Firebase Firestore ในการอ่านค่ามาเก็บไว้ใน Model ที่สร้างไว้ ดังภาพ

```

1 class _HomePageState extends State<HomePage> {
2   List<UserModel> userModel = [];
3
4   @override
5   void initState() {
6     // TODO: implement initState
7     super.initState();
8     readDataUserModel();
9   }
10
11   Future<Null> readDataUserModel() async {
12     Firebase.initializeApp().then((value) async {
13       FirebaseFirestore.instance
14         .collection('userModel')
15         .snapshots()
16         .listen((event) {
17           for (var element in event.docs) {
18             UserModel model = UserModel.fromMap(element.data());
19             setState(() {
20               userModel.add(model);
21             });
22             print('UserModel is ===>>>$userModel<<<===');
23           }
24         });
25     });
26   }
27
28   @override
29   Widget build(BuildContext context) {
30     return Scaffold(
31       appBar: AppBar(
32         leading: Icon(Icons.home),
33         title: Text('Home Page'),
34       ),
35     );
36   }
37 }

```

ภาพประกอบ 8.31 การอ่านค่าข้อมูลจาก Firestore Database มาเก็บไว้ใน Model  
ที่มา : ณปภัช วรรณตรง (2564 : ...)

3.5 นำ Future Function ที่สร้างไปใส่ไว้ใน initState method แล้วทำการ  
สั่งการทำงานโปรเจกต์ Flutter

3.6 เมื่อสั่งการทำงานโปรเจกต์สำเร็จแล้ว ให้ดูที่ console log terminal บน VS Code  
หาคำว่า UserModel is ===>>>\$userModel<<<=== ที่สั่งแสดงผลไว้ใน Future Function จะได้  
ผลลัพธ์ ดังภาพ

```

/mnt/expand:/data/user/0/com.google.android.gms:/product/lib64:/system/product/lib64
V/NativeCrypto( 9716): Registering com/google/android/gms/org/conscrypt/NativeCrypto's 295 native methods...
W/ion_work_shop_2( 9716): Accessing hidden method Ljava/security/spec/ECParameterSpec;->getCurveName()Ljava/lang/Strin
I/ProviderInstaller( 9716): Installed default security provider GmsCore_OpenSSL
I/flutter ( 9716): UserModel is ==>>>[UserModel(name: Thinnakorn, email: thinnakorn@gmail.com, type: user)]<<==
D/TrafficStats( 9716): tagSocket(78) with statsTag=0xffffffff, statsUid=-1
W/ion_work_shop_2( 9716): Accessing hidden field Ljava/net/Socket;->impl:Ljava/net/SocketImpl; (unsupported, reflecti
W/ion_work_shop_2( 9716): Accessing hidden method Ljava/security/spec/ECParameterSpec;->setCurveName(Ljava/lang/Strin

```

ภาพประกอบ 8.32 การข้อมูลของ userModel ที่ได้บน Terminal ใน VS Code

ที่มา : ณปภัช วรรณตรง (2564 : ...)

3.7 เมื่อได้ข้อมูลดังภาพประกอบ 8.33 แล้ว ให้เริ่มต้นเขียน Code แสดงข้อมูลที่จาก firestore database ไปแสดงผลบน Imurater หรือตัวจำลอง Android โดยจะเขียนในส่วนของ Body ดังภาพ

```

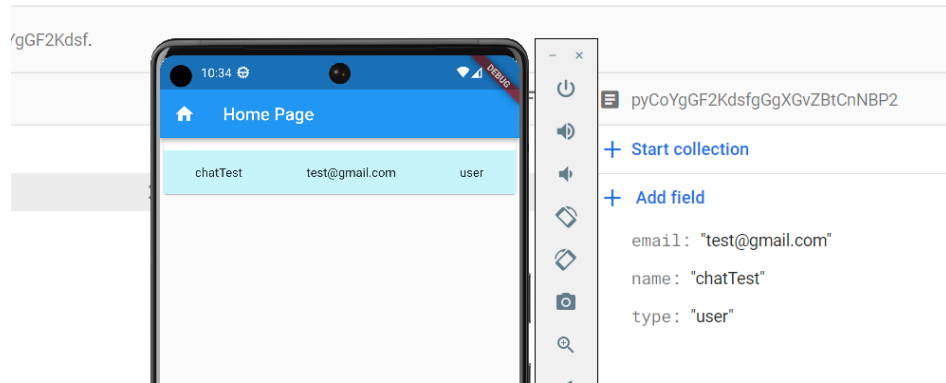
1 body: userModel.isEmpty
2   ? Text('ไม่พบข้อมูลใช้งาน')
3   : ListView.builder(
4     itemCount: userModel.length,
5     itemBuilder: (context, index) => Card(
6       child: Container(margin: EdgeInsets.only(top: 10),
7         color: Color.fromARGB(255, 198, 244, 248),
8         height: 50,
9         child: Row(
10          mainAxisAlignment: MainAxisAlignment.spaceAround,
11          children: [
12            Text(userModel[index].name),
13            Text(userModel[index].email),
14            Text(userModel[index].type),
15          ],
16        ),
17      ),
18    ),
19  ),

```

ภาพประกอบ 8.33 การเขียน Code ในการแสดงผลข้อมูลบน Imurater

ที่มา : ณปภัช วรรณตรง (2564 : ...)

3.8 เมื่อทำการเขียน Code เสร็จเรียบร้อยแล้วให้ทำการสั่งการทำงานโปรเจกต์และดูผลลัพธ์ หากเขียนถูกต้องจะแสดงผลดังภาพ



ภาพประกอบ 8.34 การแสดงผลข้อมูลบน Imurater และนำข้อมูลไปเปรียบเทียบกับ Firestore Database

ที่มา : ณปภัช วรรณตรง (2564 : ...)

## การใช้งาน Firebase Authentication

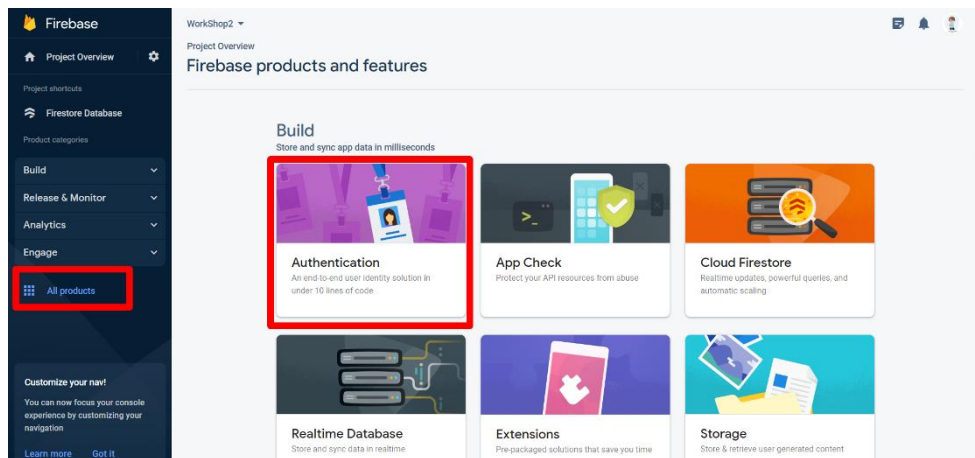
Firebase Authentication คือ การตรวจสอบสิทธิ์ Firebase ให้บริการแบ็กเอนด์ SDK ที่ใช้งานง่าย และไลบรารี UI สำเร็จรูปเพื่อตรวจสอบสิทธิ์ผู้ใช้ในแอปพลิเคชัน รองรับการตรวจสอบสิทธิ์โดยใช้รหัสผ่าน หมายเลขโทรศัพท์ และผู้ให้บริการข้อมูลประจำตัวส่วนกลางยอดนิยม เช่น Google, Facebook และ Twitter และอื่นๆ Firebase (Fiebase, 2566)

เนื้อหาส่วนนี้จะเป็นการติดตั้ง Library สำหรับการตรวจสอบสิทธิ์ผู้ใช้ใน Firebase และการเลือกแพลตฟอร์มที่สามารถล็อกอินผ่าน Firebase โดยในเนื้อหานี้จะใช้รูปแบบการเข้ารหัสผ่าน ดังมีรายละเอียดขั้นตอนดังนี้

เริ่มต้นด้วยการติดตั้ง Library ดังต่อไปนี้

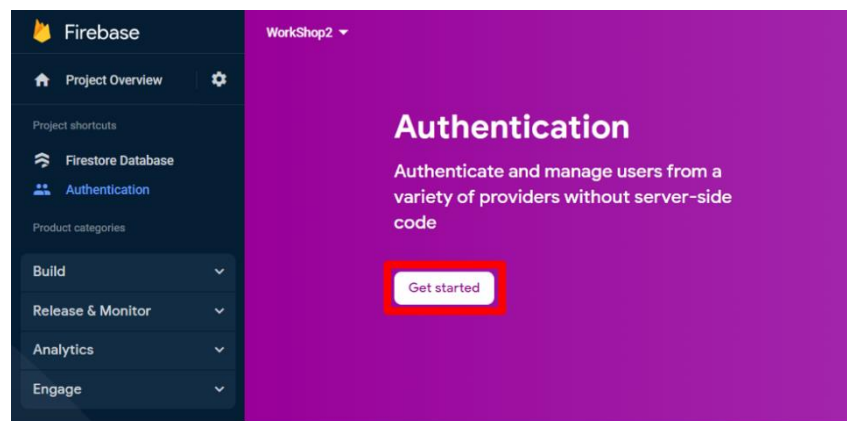
1. firebase\_auth

1.1 ไปที่ All Products เลือกที่ Authentication ดังภาพประกอบ 8.36



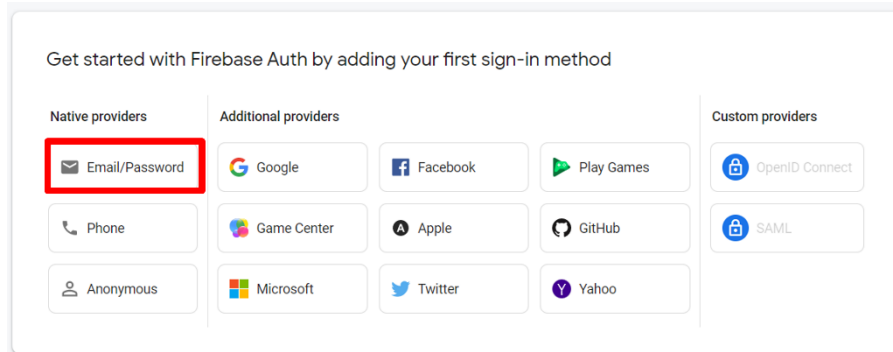
ภาพประกอบ 8.35 การเริ่มใช้งาน Firebase Authentication  
ที่มา : Firebase (2020 : 1)

### 1.2 ทำการกดที่ Get Started ดังภาพ



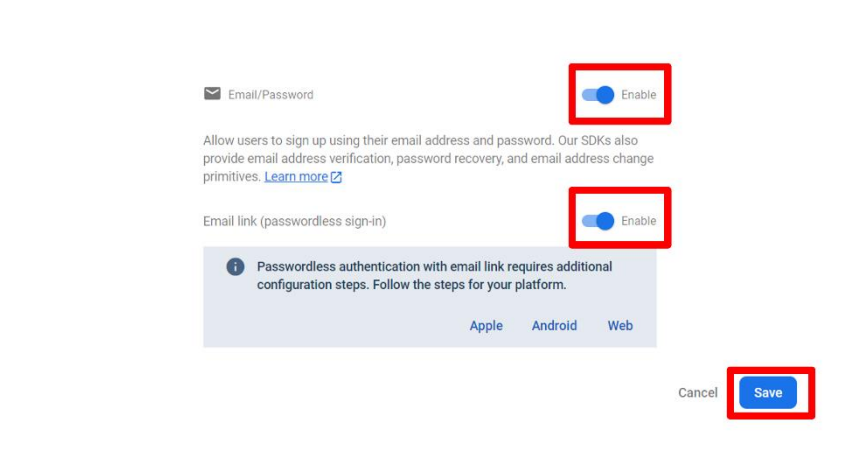
ภาพประกอบ 8.36 การเปิดการใช้งาน Firebase Authentication  
ที่มา : Firebase (2020 : 1)

### 1.3 เลือกแพลตฟอร์มที่ต้องการใช้งานในการล็อกอิน ดังภาพ



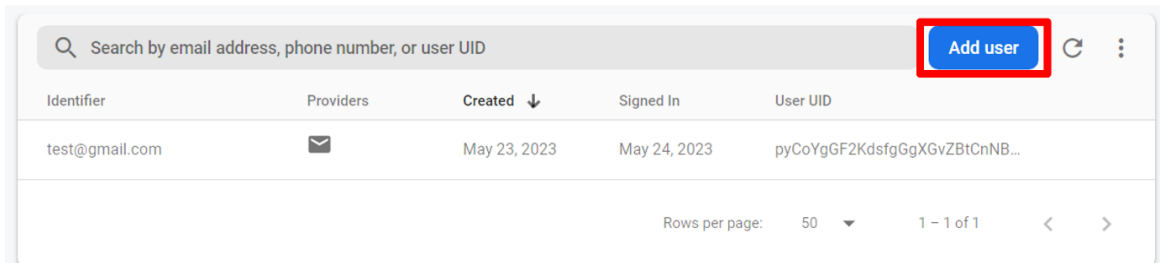
ภาพประกอบ 8.37 แพลตฟอร์มที่สามารถล็อกอินผ่าน Firebase Authentication  
ที่มา : Firebase (2020 : 1)

### 1.4 เลือกการล็อกอินด้วย Email/Password แล้วทำการเปิดการใช้งานและกด Save ให้เรียบร้อย การเปิดการใช้งาน Authentication ก็เป็นอันเสร็จสิ้น ดังภาพ



ภาพประกอบ 8.38 เปิดการใช้งาน Email/Password  
ที่มา : Firebase (2020 : 1)

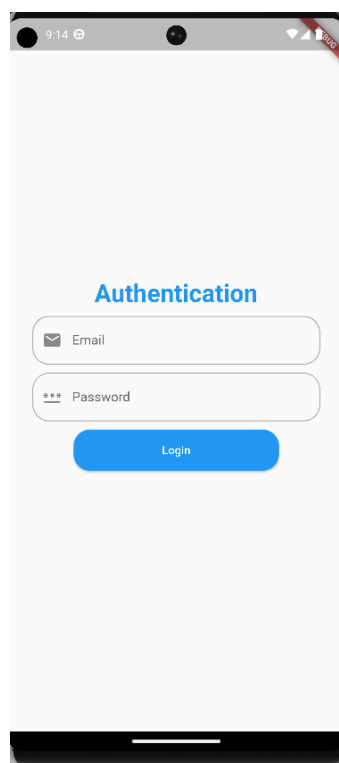
1.5 ทำการเพิ่ม Users โดยกดที่ปุ่ม Add user ใส่ Email/Password ให้ครบถ้วนทำการกดบันทึกให้เรียบร้อย หากเพิ่มข้อมูลสำเร็จจะแสดงดังภาพ



ภาพประกอบ 8.39 User ที่ทำการเพิ่มข้อมูลเข้ามา

ที่มา : Firebase (2020 : 1)

1.6 ออกแบบหน้าแอปพลิเคชันในการล็อกอินให้เรียบร้อย ดังภาพ



ภาพประกอบ 8.40 ตัวอย่างการออกแบบหน้าล็อกอิน

ที่มา : Firebase (2020 : 1)

จากภาพประกอบ 8.41 จะมี Widget ที่ใช้งานในหน้าแอปพลิเคชัน ได้แก่ Text, TextField และ ElevatedButton ดังภาพ

```

1  Widget emailTextField() {
2    return Container(
3      margin: EdgeInsets.only(top: 10),
4      width: 350,
5      child: TextField(
6        decoration: InputDecoration(
7          prefixIcon: Icon(Icons.email_rounded),
8          label: Text('Email'),
9          enabledBorder: OutlineInputBorder(
10         borderRadius: BorderRadius.circular(20),
11         borderSide: BorderSide(color: Colors.grey),
12       ),
13       focusedBorder: OutlineInputBorder(
14         borderRadius: BorderRadius.circular(20),
15         borderSide: BorderSide(color: Colors.blue),
16     ),
17   ),
18 ),
19 );
20 }

```

ภาพประกอบ 8.41 ตัวอย่างการใช้งาน TextField

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

```

1  Widget loginElevatedButton() {
2    return Container(
3      margin: EdgeInsets.only(top: 10),
4      width: 250,
5      height: 50,
6      child: ElevatedButton(
7        style: ElevatedButton.styleFrom(
8          shape: RoundedRectangleBorder(
9            borderRadius: BorderRadius.circular(20),
10         ),
11       ),
12       onPressed: () {},
13       child: Text('Login'),
14     ),
15   );
16 }

```

ภาพประกอบ 8.42 ตัวอย่างการใช้งาน ElevatedButton

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

1.7 เมื่อออกแบบหน้าแอปพลิเคชันเรียบร้อยแล้ว ให้ไปทำการประกาศตัวแปรที่ใช้ในการอ่านค่าจากฐานข้อมูลมาเก็บไว้ในไฟล์ authentication.dart ดังภาพ

```

1 class _AuthenticationState extends State<Authentication> {
2
3   String? email, password, type; // ใช้ในการรับข้อมูลจาก TextField และเช็คประเภทการล็อกอิน
4   List<UserModel> userModel = []; // model ในการรับข้อมูลจาก Firestore database
5
6   @override
7   Widget build(BuildContext context) {
8     return Scaffold();
9   }
10 }

```

ภาพประกอบ 8.43 ตัวอย่างประกาศตัวแปร

ที่มา : ณปภัช วรรณตรง (2564 : ...)

1.8 นำตัวแปรที่ประกาศในการใช้งานรับค่าจาก TextField ไปเพิ่มในแต่ละ TextField ที่ต้องการรับค่าและนำข้อมูลที่ได้มาใช้งานต่อ ดังภาพ

```

1 Widget emailTextField() {
2   return Container(
3     margin: EdgeInsets.only(top: 10),
4     width: 350,
5     child: TextField(
6       onChanged: (value) => email = value.trim(),
7       decoration: InputDecoration(
8         prefixIcon: Icon(Icons.email_rounded),
9         label: Text('Email'),
10        enabledBorder: OutlineInputBorder(
11          borderRadius: BorderRadius.circular(20),
12          borderSide: BorderSide(color: Colors.grey),
13        ),
14        focusedBorder: OutlineInputBorder(
15          borderRadius: BorderRadius.circular(20),
16          borderSide: BorderSide(color: Colors.blue),
17        ),
18      ),
19    );
20 };
21 }

```

ภาพประกอบ 8.44 ตัวอย่างการเพิ่มตัวแปรเพื่อรับค่าข้อมูลจากการกรอกบน TextField

ที่มา : ณปภัช วรรณตรง (2564 : ...)



1.9 เขียนเช็คข้อมูลที่ได้รับจากการกรอกบน TextField กับ Firebase Authentication และ Firestore Database โดยสร้าง Future Function ในการเช็คข้อมูลดังภาพ

```

1 Future<Null> checkAuthen() async {
2   await Firebase.initializeApp().then((value) async {
3     await FirebaseAuth.instance
4       .signInWithEmailAndPassword(email: email!, password: password!)
5       .then((value) async {
6         String? uid = value.user!.uid;
7         await FirebaseFirestore.instance
8           .collection('userTable')
9           .doc(uid)
10          .snapshots()
11          .listen((event) {
12            UserModel model = UserModel.fromMap(event.data()!);
13            switch (model.type) {
14              case 'user':
15                Navigator.pushAndRemoveUntil(
16                  context,
17                  MaterialPageRoute(
18                    builder: (context) => HomePage(),
19                  ),
20                  (route) => false);
21              break;
22              default:
23            }
24          });
25        });
26      });
27    }

```

ภาพประกอบ 8.45 การเช็คการล็อกอิน Firebase

ที่มา : ณปภัช วรณตรง (2564 : ...)

1.10 นำ Future Function checkAuthen ที่สร้างขึ้นไปใส่ในปุ่ม Login ในส่วนของ onPressed ดังภาพ

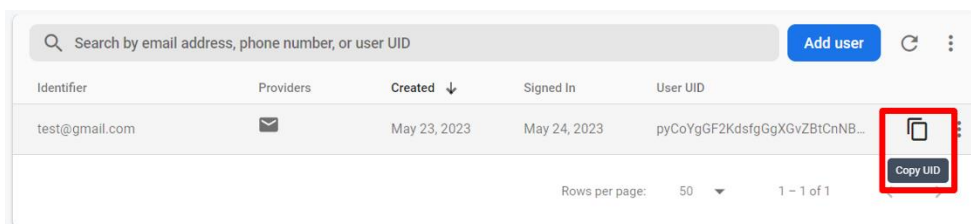
```

1
2 Widget loginElevatedButton() {
3   return Container(
4     margin: EdgeInsets.only(top: 10),
5     width: 250,
6     height: 50,
7     child: ElevatedButton(
8       style: ElevatedButton.styleFrom(
9         shape: RoundedRectangleBorder(
10          borderRadius: BorderRadius.circular(20),
11        ),
12      ),
13      onPressed: () {
14        checkAuthen();
15      },
16      child: Text('Login'),
17    ),
18  );
19 }

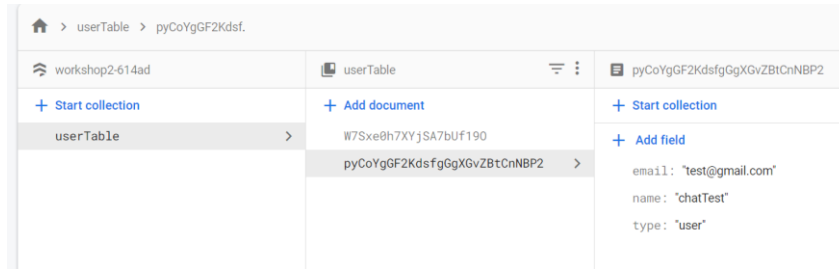
```

ภาพประกอบ 8.46 การเพิ่ม Future Function checkAuthen ในปุ่ม Login  
ที่มา : ฅนปักษ์ วรณตรง (2564 : ...)

1.11 เนื่องจาก UID ของ User ที่สมัครไว้ใน Authentication ไม่เหมือนกันกับ UID ของ Collection Firestore Database ให้ไปที่ Firebase Authentication ทำการ Copy UID ของ User ไว้ นำ UID ที่ได้ไปเพิ่ม Document ใหม่ใน Cloud Firestore ดังภาพ



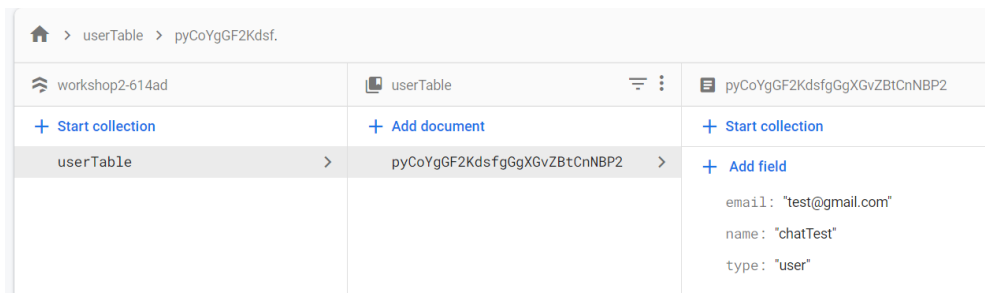
ภาพประกอบ 8.47 การ Copy UID User  
ที่มา : Firebase. (2020 : 1)



ภาพประกอบ 8.48 ข้อมูลหลังการเพิ่ม Document

ที่มา : Firebase. (2020 : 1)

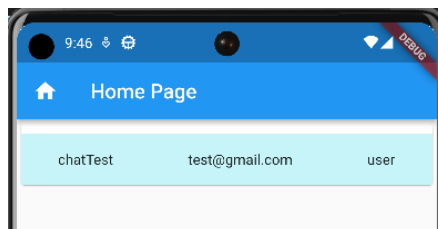
1.12 เมื่อได้ดั่งภาพประกอบ 8.49 แล้วให้ทำการลบ Document ตัวเดิมออกเนื่องจาก UID ของ Document ดังกล่าวไม่ตรงกับ UID User ที่ทำการสมัครไว้บน Firebase Authentication ลบ ข้อมูลเรียบร้อยแล้วจะได้ดั่งภาพ



ภาพประกอบ 8.49 ข้อมูลหลังการลบ Document

ที่มา : Firebase. (2020 : 1)

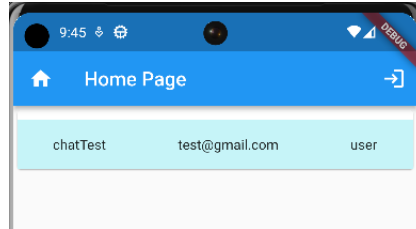
1.13 สั่งการทำงานแอปพลิเคชัน ทำการล็อกอินผ่านหน้าแอปพลิเคชัน โดยใช้ E-mail และ Password ที่ได้ทำการสมัครไว้บน Firebase Authentication หาก E-mail Password และ Type ถูกต้องจะล็อกอินไปยังหน้า Home Page ดั่งภาพ



ภาพประกอบ 8.50 หน้า Home Page

ที่มา : ฌปภัช วรณตรง (2564 : ...)

1.14 เมื่อสามารถล็อกอินเข้ามาใช้งานแอปพลิเคชันได้แล้ว ให้ทำการสร้างปุ่มสำหรับใช้ในการล็อกเอาต์ไว้บน AppBar และทำการเขียน code ในการล็อกเอาต์ ดังภาพ



ภาพประกอบ 8.51 ปุ่มสำหรับการล็อกเอาต์

ที่มา : ฅปภัช วรรณตรง (2564 : ...)

```

1  appBar: AppBar(
2    leading: Icon(Icons.home),
3    title: Text('Home Page'),
4    actions: [
5      IconButton(
6        onPressed: () {
7          signOut();
8        },
9        icon: Icon(Icons.login))
10   ],
11  ),

```

ภาพประกอบ 8.52 การเขียน Code เพิ่ม IconButton ไว้บน AppBar

ที่มา : ฅปภัช วรรณตรง (2564 : ...)

```

1  Future<Null> signOut() async {
2    await Firebase.initializeApp().then((value) async {
3      await FirebaseAuth.instance.signOut().then((value) {
4        Navigator.push(
5          context,
6          MaterialPageRoute(
7            builder: (context) => Authentication(),
8          ));
9    });
10  });
11  }

```

ภาพประกอบ 8.53 การเขียน Code ล็อกเอาต์ หรือลงชื่อออกจากระบบ

ที่มา : ฅปภัช วรรณตรง (2564 : ...)

## การใช้งาน image\_picker ร่วมกับ Storage

image\_picker เป็นปลั๊กอิน (Plugin) ของ Flutter สำหรับ iOS และ Android สำหรับเลือกภาพจากไลบรารีรูปภาพ และถ่ายภาพใหม่ด้วยกล้อง สำหรับการเพิ่มรูปภาพจากกล้อง แกลลอรี่ และวิดีโอ ในโปรเจกต์ Flutter (pub.dev, 2566)

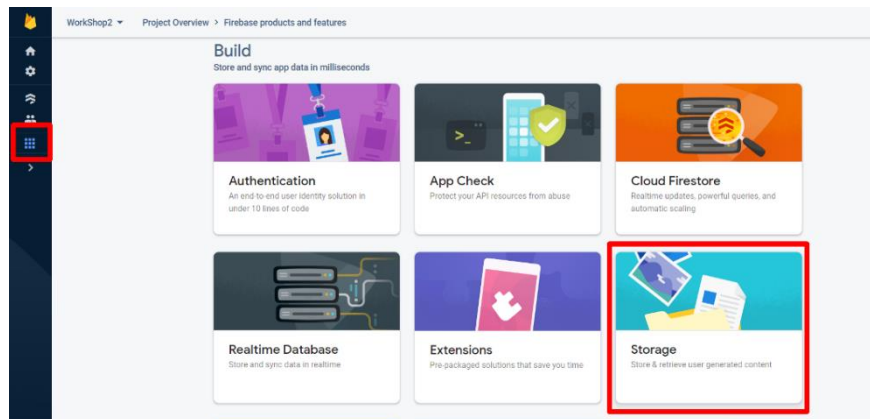
Cloud Storage for Firebase สร้างขึ้นบนโครงสร้างพื้นฐาน Google Cloud ที่รวดเร็วและปลอดภัยสำหรับนักพัฒนาแอปที่ต้องการจัดเก็บและให้บริการเนื้อหาที่ผู้ใช้สร้างขึ้น เช่น รูปภาพหรือวิดีโอ เป็นบริการที่ใช้สำหรับการเก็บข้อมูลรูปภาพของ Firebase (Fiebase, 2566)

เนื้อหาส่วนนี้จะเป็นการติดตั้ง Library สำหรับการใช้งาน image\_picker การเปิดการใช้งาน Storage การตั้งค่าเริ่มต้นให้กับ Storage การกำหนดตำแหน่งในการเก็บข้อมูลให้กับ Cloud Storage และการสร้างโค้ดเพิ่มรูปภาพใน Flutter ดังมีรายละเอียดขั้นตอนดังนี้

เริ่มต้นด้วยการติดตั้ง Library ดังต่อไปนี้

1. image\_picker
2. firebase\_storage

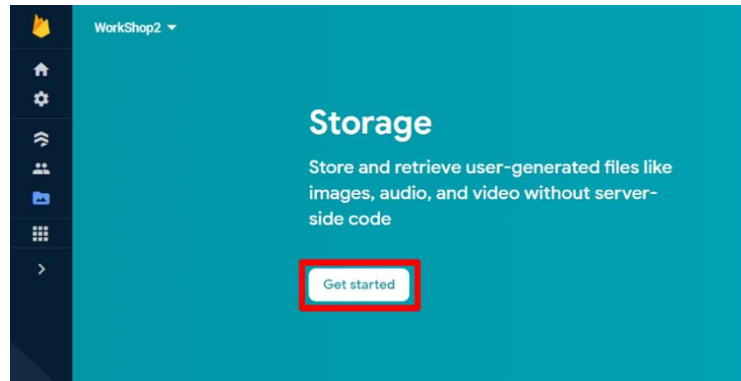
2.1 ไปที่ All products เลือกที่ Storage ดังภาพ



ภาพประกอบ 8.54 เริ่มต้นใช้งาน Storage

ที่มา : Firebase. (2020 : 1)

## 2.2 ทำการกดที่ Get Started ดังภาพ



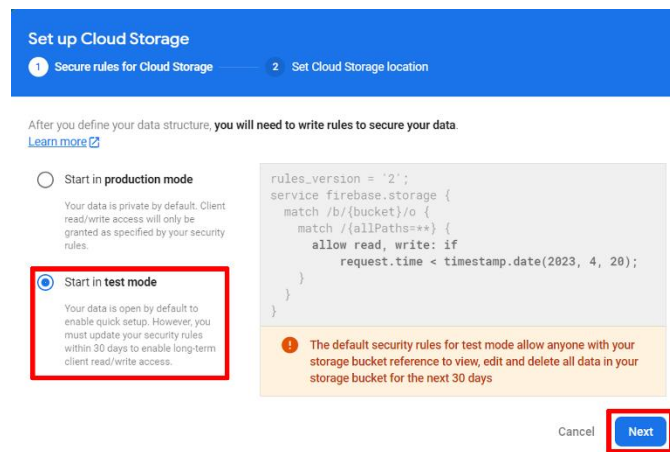
### ภาพประกอบ 8.55 การเปิดการใช้งาน Storage

ที่มา : Firebase. (2020 : 1)

## 2.3 หลังจากที่ทำกรกดสร้าง Cloud Storage จะแสดงดังภาพ

### 2.3.1 เลือก test mode

#### 2.3.1 กด Next

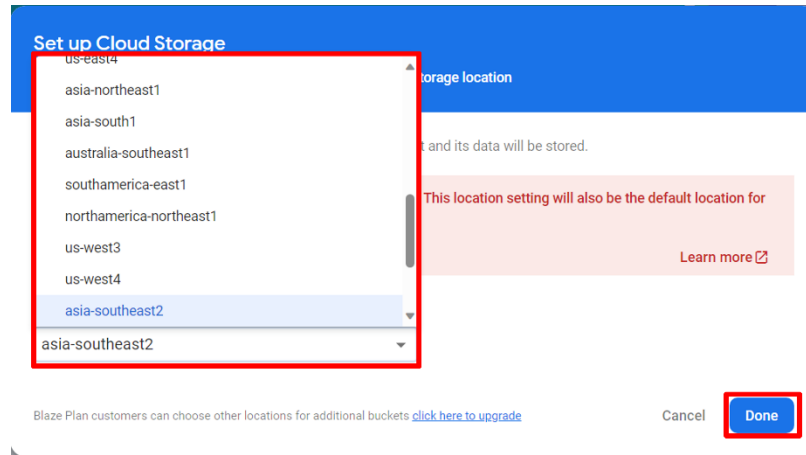


### ภาพประกอบ 8.56 การตั้งค่าเริ่มต้นให้กับ Storage

ที่มา : Firebase. (2020 : 1)

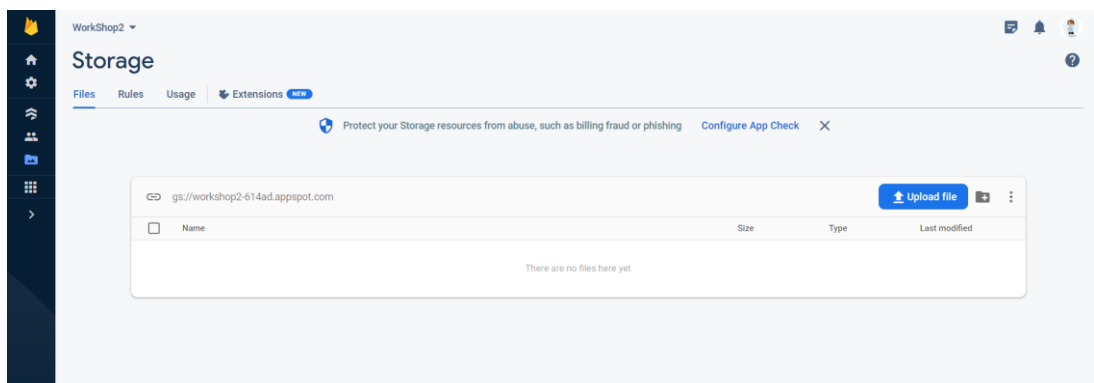
2.3.3 เลือกที่ตั้งในการเก็บข้อมูลของ Cloud Storage ของโปรเจกต์ หากทำการเลือกที่เก็บข้อมูลตั้งแต่ใช้ Firestore Database ไม่ต้องเลือกที่เก็บข้อมูลใหม่สามารถดสร้าง Cloud Storage ได้ทันที

2.3.4 กด Done ดั้งภาพ



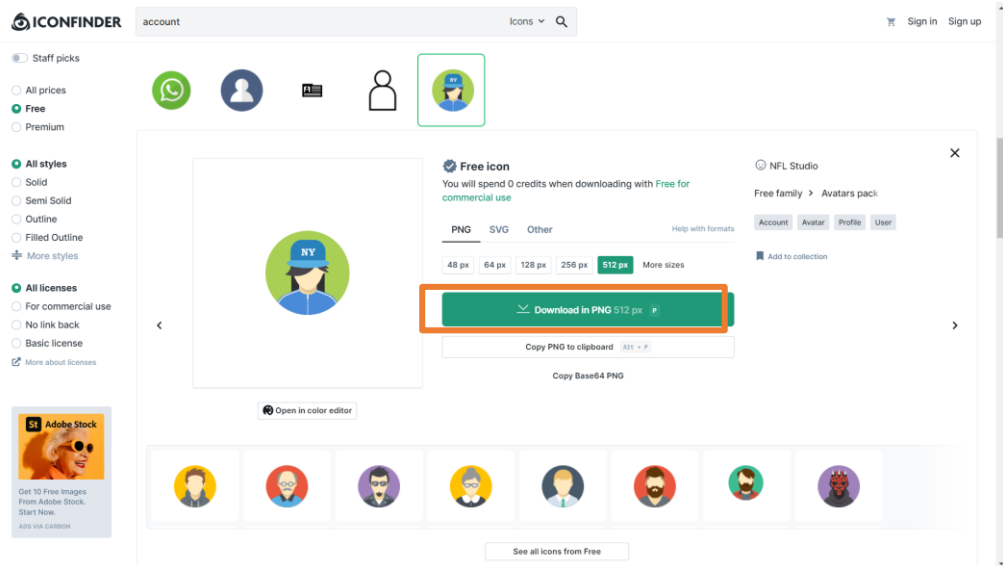
ภาพประกอบ 8.57 การกำหนดตำแหน่งในการเก็บข้อมูลให้กับ Cloud Storage  
ที่มา : Firebase. (2020 : 1)

2.4 หลังจากทำการสร้างเรียบร้อยแล้ว จะได้ดังภาพ



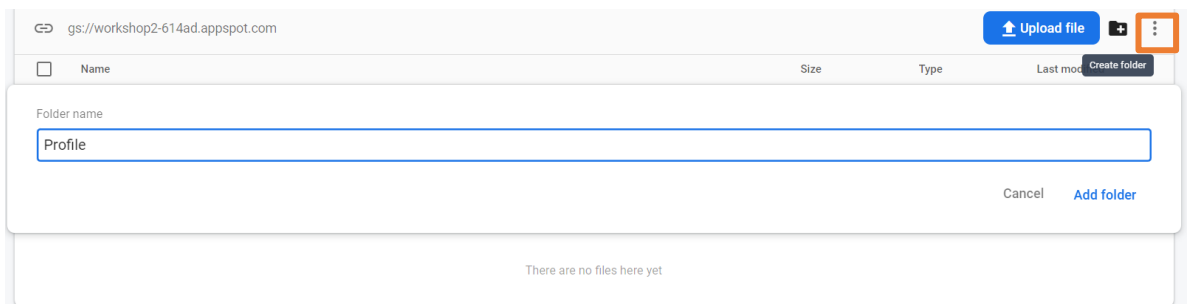
ภาพประกอบ 8.58 Cloud Storage สำหรับการใช้เก็บข้อมูลรูปภาพ Firebase  
ที่มา : Firebase. (2020 : 1)

2.5 ไปที่เว็บไซต์ <https://www.iconfinder.com> แล้วใส่คำค้นหาว่า Account จากนั้นให้เลือกรูปที่ชอบแล้วดาวน์โหลดเก็บไว้ ดึงภาพ



ภาพประกอบ 8.59 การดาวน์โหลดรูปภาพผ่านเว็บไซต์ Iconfinder  
ที่มา : Iconfinder. (2020 : 1)

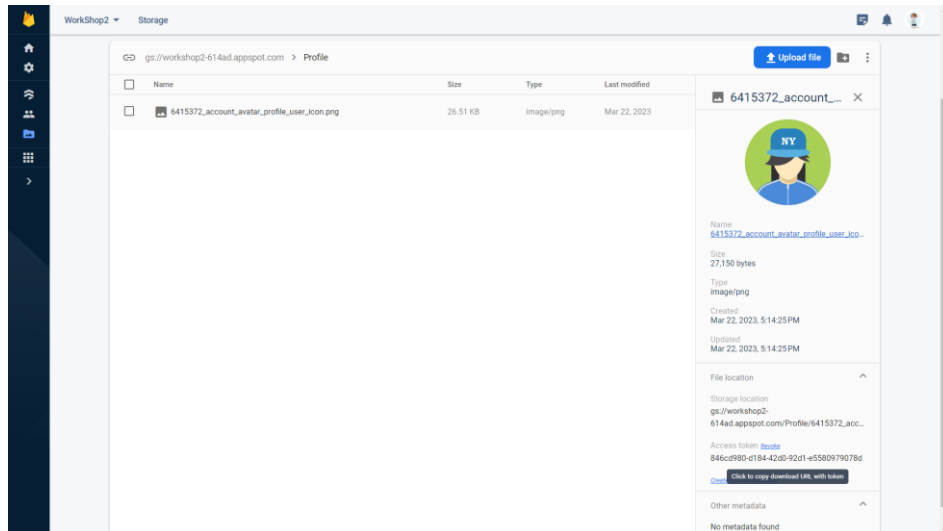
2.6 สร้างโฟลเดอร์ Profile ดึงภาพ



ภาพประกอบ 8.60 การสร้างโฟลเดอร์บน Cloud Storage  
ที่มา : Firebase. (2020 : 1)

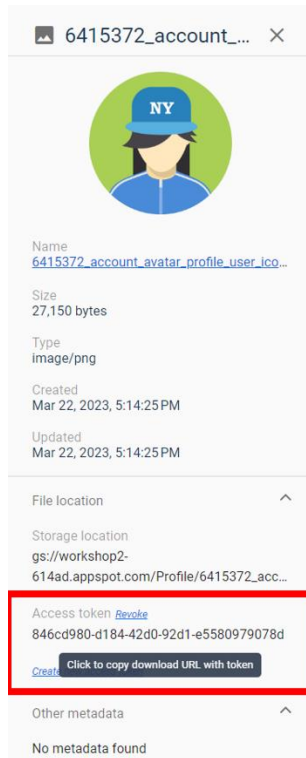


2.7 เมื่อสร้างโฟลเดอร์เสร็จเรียบร้อยแล้ว ให้กดเข้าไปที่โฟลเดอร์ Profile เพิ่มไฟล์รูปภาพที่ดาวน์โหลดไว้ ดังภาพ



ภาพประกอบ 8.61 ไฟล์รูปภาพที่ทำการอัปโหลดไปยัง Cloud Storage  
ที่มา : Firebase. (2020 : 1)

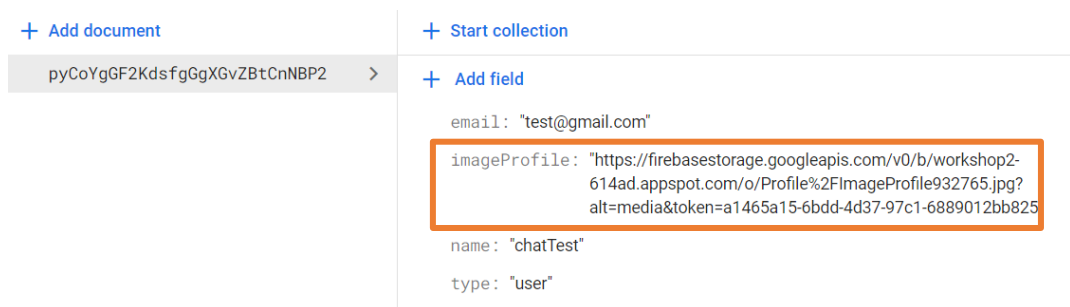
## 2.8 ไปที่ File location ทหา Access token แล้ว Copy ไว้ดั่งภาพ



ภาพประกอบ 8.62 การ Copy Access token

ที่มา : Firebase. (2020 : 1)

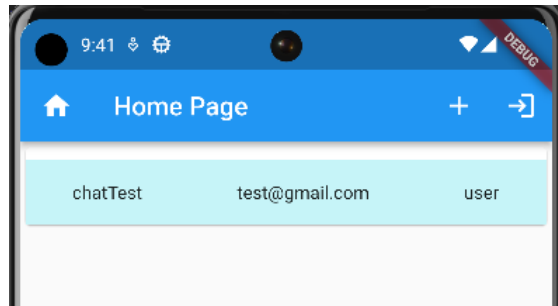
## 2.9 นำ Storage location ไปทำการสร้าง Field ใน Document ของ User โดยมีชื่อ Field ว่า imageProfile จะได้ดั่งภาพ



ภาพประกอบ 8.63 การเพิ่ม Field imageProfile

ที่มา : Firebase. (2020 : 1)

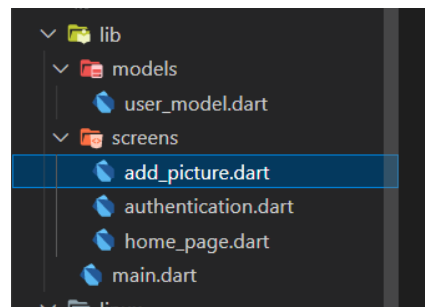
2.10 เพิ่ม Icon Button เพื่อกดไปยังหน้า AddPicturePage ใน AppBar ไว้ข้างหน้าปุ่ม  
ออกจากระบบ ดังภาพ



ภาพประกอบ 8.64 การเพิ่ม Icon Button

ที่มา : ณปภัช วรรณตรง (2564 : ...)

2.11 เพิ่มไฟล์ใหม่ ทำการเขียน Code ในไฟล์ให้เรียบร้อยโดยมีชื่อ Class ว่า  
AddPicturePage ดังภาพ



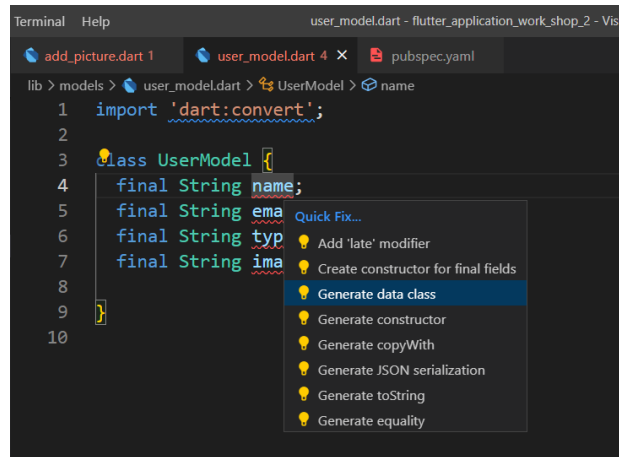
ภาพประกอบ 8.65 การเพิ่มไฟล์ add\_picture.dart

ที่มา : ณปภัช วรรณตรง (2564 : ...)

2.12 เขียนโค้ดเมื่อทำการกดไอคอนที่เพิ่มขึ้นมาใหม่สามารถไปยังหน้า AddPicturePage

2.13 แก้ไข UserModel เพิ่ม final String imageProfile; ให้ Generate data Class ใหม่

ดั่งภาพ



ภาพประกอบ 8.66 การ Generate data Class ใหม่

ที่มา : ฅนปักษ์ วรณตรง (2564 : ...)

2.14 เขียนโค้ดแสดงผลข้อมูลที่เก็บไว้บน Firestore Database และ Cloud Storage เริ่มด้วยการประกาศตัวแปร เขียน Future Function นำ Future ที่สร้างขึ้นไว้ใน initState(){} ให้เรียบร้อย ดังภาพ

```

1 String? urlImageProfile, uidUser, newUrlImageProfile;
2 UserModel? userModel;
3
4 @override
5 void initState() {
6   // TODO: implement initState
7   super.initState();
8   readDataUserLogin();
9 }
10
11 Future<Null> readDataUserLogin() async {
12   FirebaseAuth.instance.authStateChanges().listen(
13     (event) async {
14       uidUser = event!.uid;
15       FirebaseFirestore.instance
16         .collection('userTable')
17         .doc(uidUser)
18         .snapshots()
19         .listen(
20           (event) async {
21             setState(
22               () {
23                 userModel = UserModel.fromMap(
24                   event.data()!,
25                 );
26               },
27             );
28             urlImageProfile = userModel!.imageProfile;
29           },
30         );
31       },
32     );
33 }

```

ภาพประกอบ 8.67 ตัวอย่าง Code หาข้อมูลผู้ใช้งานมาเก็บไว้ใน userModel  
ที่มา : ฅนปักษ์ วรณตรง (2564 : ...)

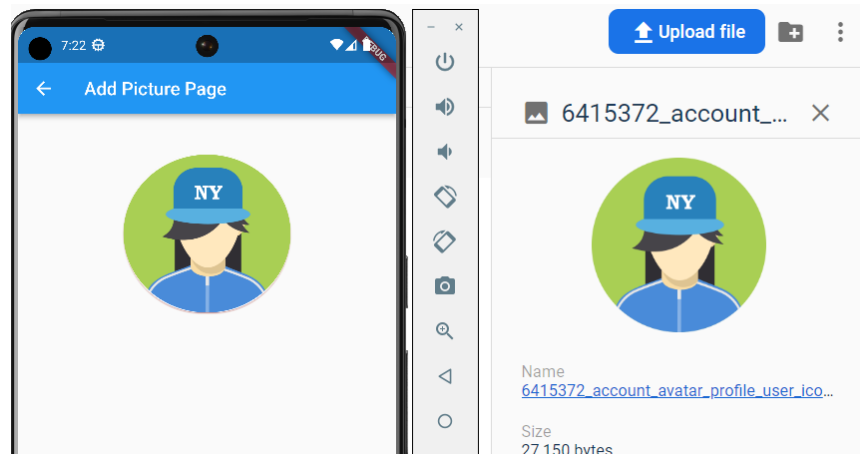
## 2.15 สร้าง Widget showImage แล้วนำไปใส่ในส่วนของ body ดังภาพ

```
1 Widget showImage() {
2   return Container(
3     margin: EdgeInsets.only(top: 40),
4     width: 190,
5     height: 180,
6     child: Card(
7       shape: RoundedRectangleBorder(
8         borderRadius: BorderRadius.all(
9           Radius.circular(100),
10        ),
11      ),
12      shadowColor: Colors.red,
13      child: Center(
14        child: ClipOval(
15          child: file == null
16            ? Image.network(
17              imageUrlProfile!,
18              width: 200,
19              height: 200,
20              fit: BoxFit.cover,
21            )
22            : Image.file(
23              file!,
24              width: 200,
25              height: 200,
26              fit: BoxFit.cover,
27            ),
28        ),
29      ),
30    );
31  };
32 }
33
```

ภาพประกอบ 8.68 ตัวอย่าง Code Widget showImage();

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

2.16 เมื่อนำ Widget `showImage` ไปใส่ในส่วนของ `Body` แล้ว ทำการสั่งการทำงานโปรเจกต์ แล้วดูผลลัพธ์ดังภาพ



ภาพประกอบ 8.69 ผลการแสดงผลภาพบน Emulator

ที่มา : ณปภัช วรรณตรง (2564 : ...)

2.16 สร้าง Future ในการเรียกใช้งาน `image_picker` ดังภาพ

```

1 Future<Null> chooseImage(ImageSource imageSource) async {
2   try {
3     var object = await ImagePicker().pickImage(
4       source: imageSource,
5       maxHeight: 500,
6       maxWidth: 500,
7     );
8     setState(() {
9       file = File(object!.path);
10    });
11  } catch (e) {}
12 }

```

ภาพประกอบ 8.70 ตัวอย่าง Code Future ในการเรียกใช้งาน `image_picker`

ที่มา : ณปภัช วรรณตรง (2564 : ...)

## 2.17 สร้างปุ่มในการกดเลือก gallery หรือ camera เมื่อทำการกดปุ่มให้เรียกใช้งาน Future chooseImage ดึงภาพ

```

1 Widget chooseImageCamera() {
2   return Container(
3     padding: EdgeInsets.all(10),
4     child: GestureDetector(
5       onTap: () => chooseImage(ImageSource.camera),
6       child: Container(
7         width: 200,
8         child: Card(
9           color: Colors.blue,
10          shape: RoundedRectangleBorder(
11            borderRadius: BorderRadius.all(
12              Radius.circular(20),
13            ),
14          ),
15          child: Row(
16            children: [
17              Container(
18                padding: EdgeInsets.all(10),
19                child: Icon(
20                  Icons.camera_alt_rounded,
21                  size: 30,
22                  color: Colors.white,
23                ),
24                Text(
25                  'เลือกจากกล้อง',
26                  style: TextStyle(
27                    fontSize: 18,
28                    color: Colors.white,
29                  ),
30                ),
31              ],
32            ),
33          ),
34        ),
35      ),
36    );
37  }

```

ภาพประกอบ 8.71 ตัวอย่าง Code เมื่อทำการกดจะเลือกจาก Camera  
ที่มา : ฅนปักษ์ วรณตรง (2564 : ...)



```

1 Widget chooseImageGallery() {
2   return Container(
3     padding: EdgeInsets.all(10),
4     child: GestureDetector(
5       onTap: () => chooseImage(ImageSource.gallery),
6       child: Container(
7         width: 200,
8         child: Card(
9           color: Colors.blue,
10          shape: RoundedRectangleBorder(
11            borderRadius: BorderRadius.all(
12              Radius.circular(20),
13            ),
14          ),
15          child: Row(
16            children: [
17              Container(
18                padding: EdgeInsets.all(10),
19                child: Icon(
20                  Icons.image,
21                  size: 30,
22                  color: Colors.white,
23                ),
24                Text(
25                  'เลือกจากแกลลอรี่',
26                  style: TextStyle(
27                    fontSize: 18,
28                    color: Colors.white,
29                  ),
30                ),
31              ],
32            ),
33          ),
34        ),
35      ),
36    );
37 }

```

ภาพประกอบ 8.72 ตัวอย่าง Code เมื่อทำการกดจะเลือกจาก Gallery  
ที่มา : ณปภัช วรรณตรง (2564 : ...)

## 2.18 ทำการเขียน Future ในการบันทึกข้อมูลไปยัง Storage และ Cloud Firestore

ดั่งภาพ

```

1 Future<Null> uploadPictureToStoage() async {
2   Random random = Random();
3   int i = random.nextInt(1000000);
4
5   Firebase.initializeApp().then((value) async {
6     FirebaseStorage storage = FirebaseStorage.instance;
7     Reference reference = storage.ref().child('Profile/ImageProfile$i.jpg');
8     UploadTask uploadTask = reference.putFile(file!);
9
10    newUrlImageProfile = await (await uploadTask).ref.getDownloadURL();
11    print('UrlImage ==>>> $newUrlImageProfile');
12
13    upDatePictureToCloudFriestore();
14  });
15 }
16
17 Future<Null> upDatePictureToCloudFriestore() async {
18   Firebase.initializeApp().then((value) async {
19     FirebaseFirestore.instance
20     .collection('userTable')
21     .doc(uidUser)
22     .update({"imageProfile": newUrlImageProfile}).then(
23     (value) async {},
24   );
25 });
26 }

```

ภาพประกอบ 8.73 ตัวอย่าง Code บันทึกข้อมูลไปยัง Storage และ Cloud Firestore  
ที่มา : ณปภัช วรรณตรง (2564 : ...)

จากภาพประกอบ 8.74 การทำงานของ Future จะเริ่มต้นทำงานที่ Future  
uploadPictureTostorage ให้เสร็จเรียบร้อยก่อน แล้วถึงจะไปเรียกใช้งาน Future  
upDatePictureToColundFirestore

## 2.19 สร้างปุ่มในการบันทึกข้อมูล เมื่อกดให้เรียกใช้งาน FutureupLoadPictureToStoage

ดั่งภาพ

```

1 appBar: AppBar(
2     actions: [
3         IconButton(
4             onPressed: () {
5                 upLoadPictureToStoage();
6             },
7             icon: Icon(Icons.save))
8     ],
9     title: Text('Add Picture Page'),
10 ),

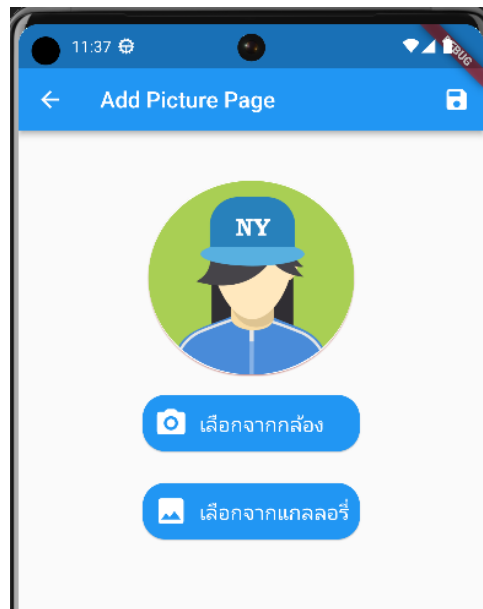
```

ภาพประกอบ 8.74 ตัวอย่าง Code ปุ่มในการกดบันทึกข้อมูล

ที่มา : ฅนปักษ์ วรรณตรง (2564 : ...)

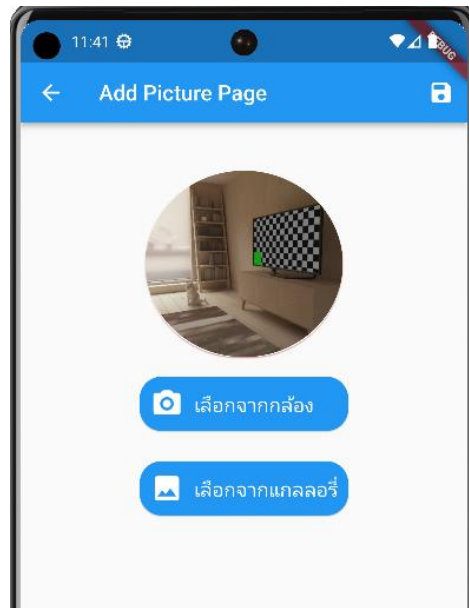
## 2.20 สิ่งการทำงานโปรเจกต์ แล้วไปยังหน้าที่จะทำการทดสอบอัปโหลดรูปภาพไปยัง

Storage และ Cloud Firestore ดั่งภาพ



ภาพประกอบ 8.75 ก่อนการเพิ่มรูปภาพ

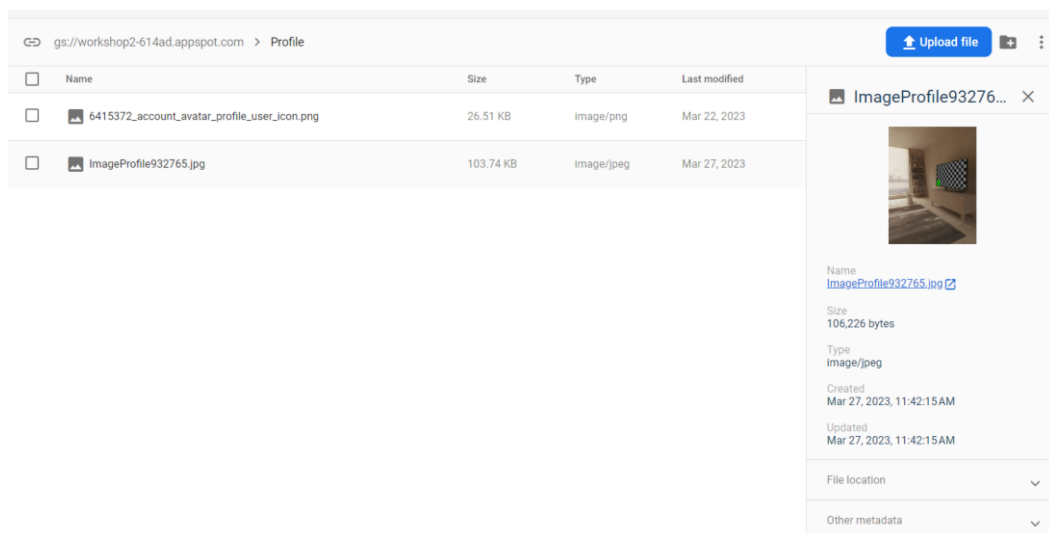
ที่มา : ฅนปักษ์ วรรณตรง (2564 : ...)



ภาพประกอบ 8.76 หลังการเพิ่มรูปภาพ

ที่มา : ฌปภัช วรรณตรง (2564 : ...)

หลังจากที่เพิ่มรูปภาพเสร็จแล้ว ทำการบันทึกรูปภาพไปยัง Storage และ Cloud Firestore ไปใช้รูปภาพที่ Storage และนำ Access token ของรูปภาพที่อัปโหลดขึ้นไปเช็คกับ Cloud Firestore ที่ Field : imageProfile ว่าตรงกันหรือไม่



ภาพประกอบ 8.77 รูปภาพที่อัปขึ้นไปใหม่

ที่มา : Firebase (2020 : 1)

## การสร้างแอปพลิเคชัน Chat ด้วย Firebase

เนื้อหาส่วนนี้เป็นประยุกต์ใช้งาน Firebase และ Firebase ในการสร้างแอปพลิเคชัน Chat ดังมีรายละเอียดขั้นตอนดังนี้

1. เพิ่มไฟล์ chat.dart ในโปรเจกต์เดิม แล้วตั้งชื่อ StatefulWidget Class ให้เรียบร้อย
2. ไปที่ไฟล์ authentication.dart แก้ไขเมื่อล็อกอินสำเร็จให้ไปที่ไฟล์ chat.dart

ดังภาพ

```

1 Future<Null> checkAuthen() async {
2   await Firebase.initializeApp().then((value) async {
3     await FirebaseAuth.instance
4       .signInWithEmailAndPassword(email: email!, password: password!)
5       .then((value) async {
6         String? uid = value.user!.uid;
7         await FirebaseFirestore.instance
8           .collection('userTable')
9           .doc(uid)
10          .snapshots()
11          .listen((event) {
12            UserModel model = UserModel.fromMap(event.data()!);
13            switch (model.type) {
14              case 'user':
15                Navigator.pushAndRemoveUntil(
16                  context,
17                  MaterialPageRoute(
18                    builder: (context) => ChatPage(),
19                  ),
20                  (route) => false);
21              break;
22              default:
23            }
24          });
25        });
26      });
27    }

```

ภาพประกอบ 8.78 Code แก้ไขการ Route ไปยัง ChatPage

ที่มา : ณปภัช วรรณตรง (2564 : ...)

3. ประกาศตัวแปรที่ใช้ในหน้า ChatPage ดังภาพประกอบ 8.80

```

1 TextEditingController msg = new TextEditingController();
2 String? title, body, nameUser;
3 FirebaseFirestore _firestore = FirebaseFirestore.instance;
4 UserModel? userModel;

```

ภาพประกอบ 8.79 การประกาศตัวแปรที่ใช้ในหน้า ChatPage

ที่มา : ณปภัช วรรณตรง (2564 : ...)

4. เขียน Future อ่านข้อมูลผู้ใช้งานจาก Firestore Database ไปเก็บไว้ในตัวแปร nameUser  
 ดังภาพ

```

1 @override
2 void initState() {
3   // TODO: implement initState
4   super.initState();
5   readDataUserLogin();
6 }
7
8 Future<Null> readDataUserLogin() async {
9   FirebaseAuth.instance.authStateChanges().listen(
10    (event) async {
11     String uidUser = event!.uid;
12     FirebaseFirestore.instance
13       .collection('userTable')
14       .doc(uidUser)
15       .snapshots()
16       .listen(
17         (event) async {
18           setState(
19             () {
20               userModel = UserModel.fromMap(
21                 event.data()!,
22               );
23             },
24           );
25           nameUser = userModel!.name;
26         },
27       );
28     },
29   );
30 }

```

ภาพประกอบ 8.80 Code การอ่านข้อมูลจาก Cloud Firestore Database  
 ที่มา : ณปภัช วรรณตรง (2564 : ...)

5. ออกแบบในส่วนของ AppBar โดยจะแสดงชื่อแล้วปุ่มออกจากระบบ ดังภาพ

```

1 return Scaffold(
2   appBar: AppBar(
3     actions: [
4       IconButton(
5         onPressed: () {
6           signOut();
7         },
8         icon: Icon(Icons.logout)),
9     ],
10    title: Text('คุณ $nameUser'),
11  ),

```

ภาพประกอบ 8.81 Code การออกแบบปุ่มออกจากระบบ  
 ที่มา : ณปภัช วรรณตรง (2564 : ...)

จากภาพประกอบ 8.82 จะเห็นได้ว่าเมื่อมีการกดปุ่มทุกครั้งจะเรียกใช้งาน Future signIn เป็น Future ที่เขียนขึ้นมาใช้สำหรับการออกจากระบบการใช้งานของผู้ใช้งาน ดังภาพ

```

1 Future<Null> signIn() async {
2   await Firebase.initializeApp().then((value) async {
3     await FirebaseAuth.instance.signIn().then((value) {
4       Navigator.push(
5         context,
6         MaterialPageRoute(
7           builder: (context) => Authentication(),
8         ));
9     });
10  });
11 }

```

ภาพประกอบ 8.82 Code การออกจากระบบของ Firebase

ที่มา : ณปภัช วรรณตรง (2564 : ...)

6. ออกแบบในส่วนของ Body จะ Query Snapshot จาก Cloud Firestore Database มาแสดงในรูปแบบ ListView และ Return ค่ากลับในรูปแบบของ Widget ที่มีชื่อว่า Message ดังภาพ

```

1 body: Container(
2   height: size.height / 1.25,
3   width: size.width,
4   child: StreamBuilder<QuerySnapshot>(
5     stream: _firestore
6       .collection('chatTable')
7       .orderBy('time', descending: false)
8       .snapshots(),
9     builder: (context, snapshot) {
10      if (snapshot.data != null) {
11        return ListView.builder(
12          itemCount: snapshot.data!.docs.length,
13          itemBuilder: (context, index) {
14            var map = snapshot.data!.docs[index].data();
15            return message(size, map);
16          },
17        );
18      } else {
19        return Container();
20      }
21    },
22  ),
23 ),

```

ภาพประกอบ 8.83 ตัวอย่าง Code ในส่วนของ Body

ที่มา : ณปภัช วรรณตรง (2564 : ...)

```
1 Widget message(Size size, var map) {  
2   return Container(  
3     width: size.width,  
4     alignment: map['sendby'] == nameUser  
5       ? Alignment.centerRight  
6       : Alignment.centerLeft,  
7     child: Container(  
8       padding: EdgeInsets.symmetric(vertical: 10, horizontal: 15),  
9       margin: EdgeInsets.symmetric(vertical: 10, horizontal: 15),  
10      decoration: BoxDecoration(  
11        color: Colors.blue,  
12        borderRadius: BorderRadius.circular(20),  
13      ),  
14      child: Text(  
15        map['message'],  
16        style: TextStyle(color: Colors.white, fontSize: 16),  
17      ),  
18    ),  
19  );  
20 }
```

ภาพประกอบ 8.84 ตัวอย่าง Code Widget Message

ที่มา : ฅนปักษ์ วรณตรง (2564 : ...)

จากภาพประกอบ 8.85 เป็นวิดเจ็ตที่เขียนเพื่อแสดงข้อความในหน้าแอปพลิเคชันโดยจะแยกข้อความจากชื่อผู้ใช้งานที่ทำการส่งข้อความ โดยใน Widget นี้จะมีการส่งค่ากลับไปยัง ListView ที่อยู่ในส่วนของ body



7. ส่วนด้านล่างสุดจะออกแบบสำหรับการกรอกข้อความ และปุ่มในการกดส่งข้อความ เมื่อทำการกดส่งข้อความจะเรียกใช้ void function ที่มีชื่อว่า onSendMessage() ดังภาพ

```

1  floatingActionButton: Container(
2    margin: EdgeInsets.only(left: 40, top: 100),
3    height: size.height / 8,
4    width: size.height,
5    alignment: Alignment.center,
6    child: Padding(
7      padding: const EdgeInsets.all(16.0),
8      child: Container(
9        height: size.height / 10,
10       width: size.height,
11       child: TextField(
12         controller: msg,
13         decoration: InputDecoration(
14           suffixIcon: IconButton(
15             onPressed: () {
16               onSendMessage();
17             },
18             icon: Icon(
19               Icons.send,
20               color: Colors.blue,
21             )),
22         hintText: 'กรุณากรอกข้อความที่ต้องการส่ง...',
23         border: InputBorder.none,
24         enabledBorder: OutlineInputBorder(
25           borderRadius: BorderRadius.circular(10),
26           borderSide: BorderSide(color: Colors.blue),
27         ),
28         focusedBorder: OutlineInputBorder(
29           borderRadius: BorderRadius.circular(10),
30           borderSide: BorderSide(color: Colors.blue),
31         ),
32       ),
33     ),
34   ),
35 ),
36 ),

```

ภาพประกอบ 8.85 ตัวอย่าง Code ในส่วนของการกรอกและส่งข้อความ  
ที่มา : ฌปภัช วรรณตรง (2564 : ...)

```

1 void onSendMessage() async {
2   if (msg.text.isNotEmpty) {
3     Map<String, dynamic> message = {
4       'sendby': nameUser,
5       'message': msg.text,
6       'time': FieldValue.serverTimestamp(),
7       // 'uidRest': queueModel.uidRest,
8     };
9
10    await _firestore.collection('chatTable').add(message);
11    msg.clear();
12  } else {
13    print("Enter Some Text");
14  }
15 }

```

ภาพประกอบ 8.86 ตัวอย่าง Code void function onSendMessage()

ที่มา : ฅนปักษ์ วรรณตรง (2564 : ...)

จากภาพประกอบ 8.87 เป็น void function ที่เขียนขึ้นมาเพื่อใช้รับค่าจาก TextField เมื่อมีการกดส่งข้อความและบันทึกไปยัง Cloud Firestore Database

8. สิ่งการทำงานโปรเจกต์ ทำการลือกอินเพื่อไปยังหน้า ChatPage ดังภาพ



ภาพประกอบ 8.87 หน้า ChatPage

ที่มา : ฅนปักษ์ วรรณตรง (2564 : ...)

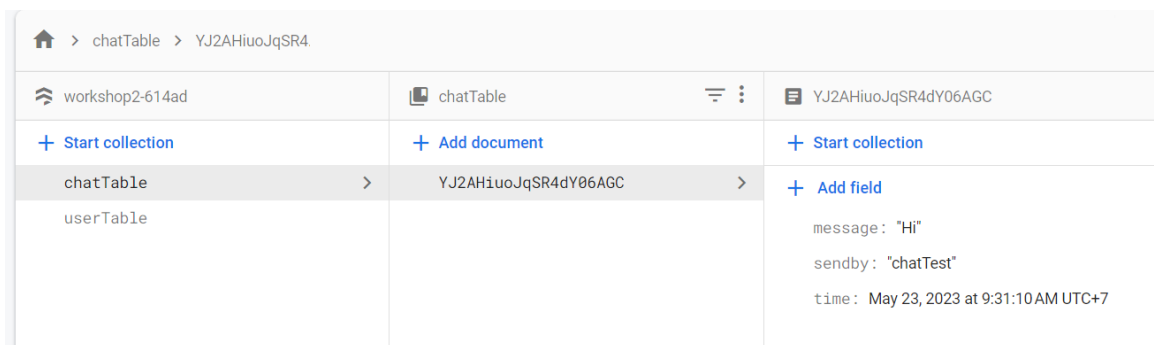
### 9. ทดสอบพิมพ์ข้อความแล้วทำการส่งข้อความจะได้ดังภาพ



ภาพประกอบ 8.88 หน้า ChatPage ที่มีข้อความ

ที่มา : ณปภัช วรรณตรง (2564 : ...)

จากภาพประกอบ 8.89 ที่ได้ส่งข้อความและแสดงผล เมื่อไปที่ Cloud Firestore Database จะพบข้อมูลที่ได้ทำการส่งโดยจะมี ข้อความที่ส่ง เวลาที่ส่ง ชื่อผู้ส่ง ดังภาพ



ภาพประกอบ 8.89 ฐานข้อมูล Cloud Firestore Database collection chatTable

ที่มา : Firebase (2020 : 1)

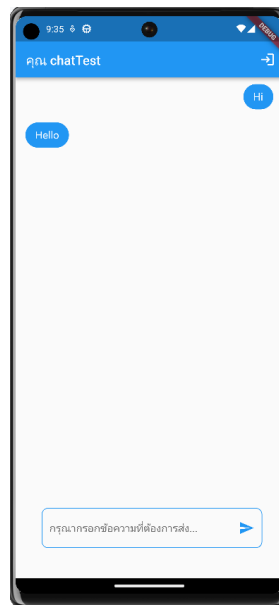
## 10. เพิ่ม Document ข้างใน Collection chatTable ดังภาพ

Document ID: v8d0pHPkZh49MJ7mDVkr

Field	Type	Value
message	string	Hello
sendy	string	test
time	timestamp	May 23, 2023

ภาพประกอบ 8.90 การเพิ่ม Document ข้างใน Collection chatTable  
ที่มา : Firebase (2020 : 1)

จากภาพประกอบ 8.91 เมื่อเพิ่มข้อความ ชื่อผู้ส่ง วันเวลา ให้ทำการกดบันทึกและไปดูที่หน้าแอปพลิเคชัน ดังภาพ



ภาพประกอบ 8.91 หน้าแอปพลิเคชันหลังเพิ่ม Document ข้างใน Collection chatTable  
ที่มา : ฌปภัช วรรณตรง (2564 : ...)

จากภาพประกอบ 8.92 เมื่อลองเพิ่ม Document ข้างใน Collection chatTableประกอบด้วยข้อความ วันเวลา ชื่อผู้ส่ง หากต้องการทดสอบพิมพ์ข้อความโต้ตอบให้ทำการสร้างบัญชีผู้ใช้เพิ่มขึ้นมา

## บทสรุป

สำหรับเนื้อหาที่กล่าวมาทั้งหมดในบทนี้ จะพูดถึงนิยามของ Firebase ลักษณะของ Firebase Database การใช้งาน Flutter ร่วมกับ Firebase การติดตั้ง Library ในโปรเจกต์ Flutter การใช้งาน Cloud Firestore การแสดงผลข้อมูลจาก Firestore Database การใช้งาน Firebase Authentication การใช้งาน image\_picker ร่วมกับ Storage และประยุกต์ใช้ความรู้ด้วยการสร้างแอปพลิเคชัน Chat ด้วย Firebase

## เอกสารอ้างอิง

จิราวุธ วารินทร์. (2563). การพัฒนาเว็บแอปพลิเคชันด้วย Firebase ร่วมกับ Vue.js. กรุงเทพฯ : ริโวว่า.

เยาวภา จรัสสันติจิต. (2562). จัดการข้อมูลด้วย MongoDB ฐานข้อมูล NoSQL. เชียงใหม่ : จรัสธุรกิจการพิมพ์.

ราวุธ วารินทร์. (2563). การพัฒนาเว็บแอปพลิเคชันด้วย Firebase ร่วมกับ React. กรุงเทพฯ : ริโวว่า.

ศุภชัย สมพานิช. (2563). การพัฒนาแอปพลิเคชันด้วย Firebase&Kotin. กรุงเทพฯ : โปรวิชั่น.

Firebase (1 มีนาคม 2566). Cloud Storage for Firebase. สืบค้นจาก,

<https://firebase.google.com/docs/storage>

Firebase (1 มีนาคม 2566). Firebase Authentication. สืบค้นจาก,

<https://firebase.google.com/docs/auth>

pub.dev. (1 มีนาคม 2566). Image Picker plugin for Flutter. สืบค้นจาก,

[https://pub.dev/packages/image\\_picker](https://pub.dev/packages/image_picker)

## บรรณานุกรม

- จิราวุธ วารินทร์. (2563). การพัฒนาเว็บแอปพลิเคชันด้วย Firebase ร่วมกับ React. กรุงเทพฯ: ริโวว่า.
- \_\_\_\_\_. (2563). การพัฒนาเว็บแอปพลิเคชันด้วย Firebase ร่วมกับ Vue.js. กรุงเทพฯ: สำนักพิมพ์ชิมพลิฟาย.
- \_\_\_\_\_. (2564). พัฒนาโมบายล์แอปด้วย Flutter + Dart. กรุงเทพฯ : สำนักพิมพ์ชิมพลิฟาย.
- บัญชา ปะลีละเตสัง. (2566). พัฒนาแอปแบบ Multi-Platform ด้วย Flutter โดยใช้ภาษา Dart. กรุงเทพฯ : ซีเอ็ดยูเคชั่น.
- \_\_\_\_\_. (2563). พัฒนา Web Application ด้วย Python Django. กรุงเทพฯ : ซีเอ็ดยูเคชั่น,
- เยาวภา จรัสสันติจิต. (2562). จัดการข้อมูลด้วย MongoDB ฐานข้อมูล NoSQL. เชียงใหม่ : จรัสธุรกิจการพิมพ์.
- ลุงวิศวกร สอนคำนวณ. (2564). Basic Dart Programming by Uncle Engineer. Flutter 2 Bootcamp 2021 - Uncle Engineer.
- ลุงวิศสอนคำนวณ. (11 กันยายน 2564). สร้างเว็บไซต์ด้วย Python Django (แจกโค้ด) เฟรมเวิร์กใช้เขียนเว็บยอดฮิตของ Python. สืบค้นจาก, <https://www.facebook.com/UncleEngineer>
- ศุภชัย สมพานิช. (2563). การพัฒนาแอปพลิเคชันด้วย Firebase&Kotin. กรุงเทพฯ: โปรวีชั่น.
- สำนักงานสถิติแห่งชาติ. (2564). สรุปผลที่สำคัญสำรวจการมี การใช้เทคโนโลยีสารสนเทศและการสื่อสารในครัวเรือน พ.ศ. 2563. กรุงเทพฯ: สำนักงานสถิติแห่งชาติ กระทรวงดิจิทัลเพื่อเศรษฐกิจและสังคม.
- อนุชิต ชโลธร. (2565). สูตรลัด Flutter. กรุงเทพฯ : สำนักพิมพ์ก๊อปวาง.
- \_\_\_\_\_. (2566). สูตรลัด Dart 3. กรุงเทพฯ : สำนักพิมพ์ก๊อปวาง.
- เอกรินทร์ วทัญญูเลิศสกุล. (2020). พัฒนา Mobile App ด้วย Flutter & Dart. กรุงเทพฯ : โปรวีชั่น, บจก.
- \_\_\_\_\_. (2564). การพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่แบบข้ามแพลตฟอร์ม. กรุงเทพฯ : สำนักพิมพ์จุฬาลงกรณ์มหาวิทยาลัย.
- Baldwin C. Y. and Woodard C. J. (2009). The Architecture of Platforms: A Unified View. SSRN Electronic Journal. DOI:10.2139/ssrn.1265155

- Bassett, L. (2015). **Bassett Introduction to JavaScript Object Notation**. O'Reilly Media, Inc.
- djangoproject. (28 กันยายน 2564). **Django**. สืบค้นจาก, <https://www.djangoproject.com/django-rest-framework>. (1 มกราคม 2566). **Django REST framework Overview**. สืบค้นจาก, <https://www.django-rest-framework.org/>
- Firebase. (1 มีนาคม 2566). **Cloud Storage for Firebase**. สืบค้นจาก, <https://firebase.google.com/docs/storage>
- \_\_\_\_\_ (1 มีนาคม 2566). **Firestore Authentication**. สืบค้นจาก, <https://firebase.google.com/docs/auth>
- file.org. (16 กุมภาพันธ์ 2566). **APK File**. สืบค้นจาก, <https://file.org/extension/apk>
- flutter.dev. (15 กันยายน 2566). **Flutter**. สืบค้นจาก, <https://flutter.dev/>.
- KongRuksiam Official. (2563). **พัฒนาแอปด้วย Flutter สำหรับผู้เริ่มต้น 7 ชั่วโมงเต็ม [FULL COURSE]**. สืบค้นจาก, [https://www.youtube.com/watch?v=3jGj-1-m\\_zA&list=PLltVQYLz1BMBUgyhxZFA31of-EKjazC8G](https://www.youtube.com/watch?v=3jGj-1-m_zA&list=PLltVQYLz1BMBUgyhxZFA31of-EKjazC8G)
- Marrs, T. (2017). **JSON at Work**. O'Reilly Media, Inc.
- Masse, M. (2011). **REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces**. O'Reilly Media.
- netguru. (28 กันยายน 2564). **Top 10 Django Apps Examples**. สืบค้นจาก, <https://www.netguru.com/blog/django-apps-examples>
- ngrok. (1 มกราคม 2566). **ngrok Overview**. สืบค้นจาก, <https://ngrok.com/docs>
- postman. (2 กุมภาพันธ์ 2566). **What is Postman?**. สืบค้นจาก, <https://www.postman.com/product/what-is-postman/>
- pub.dev. (1 มีนาคม 2566). **Image Picker plugin for Flutter**. สืบค้นจาก, [https://pub.dev/packages/image\\_picker](https://pub.dev/packages/image_picker)
- sqlitebrowser. (10 กุมภาพันธ์ 2566). **DB Browser for SQLite**. สืบค้นจาก, <https://sqlitebrowser.org/>