เอกสารประกอบการสอน
การพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่

ณปภัช วรรณตรง

สาขาวิชาวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ มหาวิทยาลัยราชภัฏบุรีรัมย์

คำนำ

เอกสารประกอบการสอนรายวิชา การพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่ รหัสวิชา 4123305 เป็นรายวิชาหนึ่งที่สำคัญของหลักสูตรสาขาวิชาวิทยาการคอมพิวเตอร์ เนื่องจากปัจจุบันอุปกรณ์เคลื่อนที่ โดยเฉพาะโทรศัพท์มือถือกลายเป็นส่วนสำคัญในชีวิตประจำวันสำหรับผู้คนในปัจจุบันจนถูกกล่าวว่าเป็นอวัยวะที่ 33 ส่งผลให้มีความจำเป็นที่นักศึกษาจะต้องเรียนรู้เพื่อนำไปใช้ในการทำงานต่อไปในอนาคต ผู้จัดทำจึงจัดทำเอกสารประกอบการสอนนี้ขึ้นเพื่อใช้ประกอบการเรียนการสอนรายวิชา การพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่ เพื่อให้นักศึกษาได้รับเนื้อหาที่เป็นประโยชน์ สามารถนำไปประยุกต์ใช้ในการทำงานในอนาคต

เอกสารประกอบการสอนรายวิชา การพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่ ผู้เขียนได้เขียนเนื้อหาครบตามหลักสูตรที่กำหนดไว้ และได้เพิ่มเติมเทคนิคและแหล่งค้นคว้าเพิ่มเติมที่ควรรู้ให้กับนักศึกษาที่สามารถนำไปใช้ในการทำงานจริงในอนาคตได้ อาทิ เว็บไซต์ flutter.dev เว็บไซต์ Stack Overflow นอกจากนี้เนื้อหาที่รวบรวม สังเคราะห์มาในเอกสารประกอบการสอน มาจากการศึกษา สังเคราะห์เนื้อหาจากเอกสาร ตำรา หนังสือ เว็บไซต์ ทั้งไทยและต่างประเทศเพื่อให้นักศึกษาได้ความรู้ที่ครอบคลุมจากแหล่งความรู้ที่หลากหลาย

การจัดทำเอกสารประกอบการสอนนี้สำเร็จลงได้ ต้องขอบพระคุณพ่อ แม่ ครูบาอาจารย์ที่เคยสิทธิ์ประสาทวิชา และเพื่อน ๆ พี่ ๆ อาจารย์มหาวิทยาลัยราชภัฏบุรีรัมย์ที่สนับสนุน เสริมแรงในการจัดทำเอกสารประกอบการสอน และท่านผู้ทรงคุณวุฒิที่ช่วยเสียสละเวลาอันมีค่าในการตรวจทาน ให้คำชี้แนะอันเป็นประโยชน์ ทำให้เอกสารประกอบการสอน รายวิชาระบบสารสนเทศเพื่อการจัดการนี้ มีความสมบูรณ์ยิ่งขึ้น

ณปภัช วรรณตรง

สารบัญ

เรื่อง **หน้า**

สารบัญ (ต่อ)

เรื่อง **หน้า**

สารบัญ (ต่อ)

เรื่อง **หน้า**

สารบัญ (ต่อ)

สารบัญ (ต่อ)

เรื่อง **หน้า**

สารบัญ (ต่อ)

เรื่อง **หน้า**

สารบัญภาพประกอบ

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

สารบัญภาพประกอบ (ต่อ)

สารบัญภาพประกอบ (ต่อ)

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สารบัญตาราง

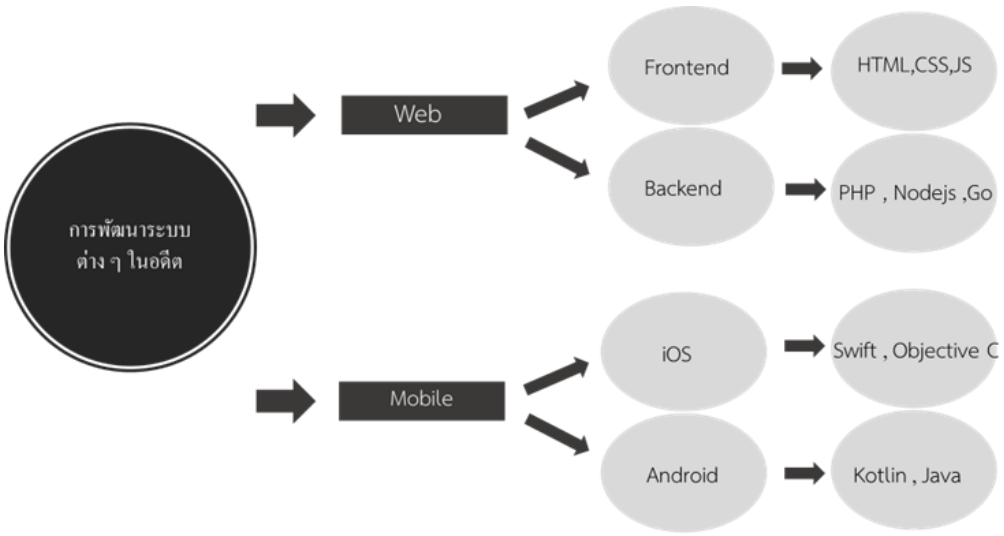
ตาราง **หน้า**

บทที่ 1
Introduction to Mobile Application Development

ปัจจุบันแอปพลิเคชันบนอุปกรณ์เคลื่อนที่เป็นที่นิยม ผลสืบเนื่องมาจากการเติบโตของอุปกรณ์เคลื่อนที่โดยจากการสำรวจของสำนักงานสถิติแห่งชาติ (2564) พบว่าประชาชนใช้อุปกรณ์เคลื่อนที่แบบ สมาร์ตโฟน (Smart Phone) ถึงร้อยละ 96.4 ส่งผลให้แอปพลิเคชันบนอุปกรณ์เคลื่อนที่มีอัตราการเติบโตตามไปด้วย ในการพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่มีเครื่องมือในการพัฒนาหลากหลาย Flutter เป็นเครื่องมือหนึ่งที่เป็นที่นิยมในการพัฒนา Flutter เป็นเฟรมเวิร์คสำหรับการพัฒนาแอปพลิเคชันจากบริษัท กูเกิล (Google) ออกแบบมาเพื่อใช้ในการพัฒนาแอปพลิเคชันแบบข้ามแพลตฟอร์ม (Cross-Platform) ซึ่งก็คือ การที่แอปพลิเคชันสามารถทำงานได้ในหลายแพลตฟอร์ม โดย Flutter รองรับการใช้งานใน ทั้ง iOS และ Android จึงทำให้การพัฒนาแอปพลิเคชันทำได้ง่าย รวดเร็ว และมีประสิทธิภาพ ในบทนี้จะกล่าวถึง การพัฒนาแอปพลิคชันบนอุปกรณ์เคลื่อนที่ในปัจจุบัน 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 Kotin และ JAVA จะเห็นว่าการพัฒนาโปรแกรมบนแพลตฟอร์มแต่ละชนิดใช้ภาษาที่แตกต่างกัน ซึ่งสร้างความยากลำบากให้กับผู้พัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่ต้องพัฒนาแอปพลิเคชันในสองรูปแบบ จึงได้มีผู้พัฒนารูปแบบที่ทำให้สามารถพัฒนาแอปพลิเคชันครั้งเดียวแต่รองรับการทำงานทั้งสองแพลตฟอร์มที่เรียกกันว่า “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

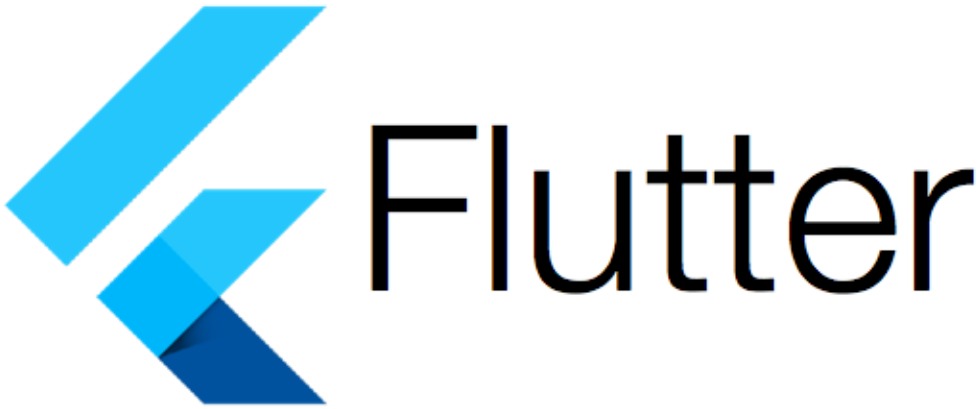
เช่น Android, iOS และ Windows Phone ได้ การพัฒนาแอปพลิเคชันเคลื่อนที่แบบข้ามแพลตฟอร์มจะช่วยแก้ปัญหาเรื่องระยะเวลาที่ต้องใช้ในการพัฒนาแอปพลิเคชันในรูปแบบเนทีฟ (การเรียกใช้ความสามารถและเข้าถึงฮาร์ดแวร์ของอุปกรณ์ได้อย่างไม่จำกัด สนับสนุนส่วนติดต่อผู้ใช้และการโต้ตอบทั้งหมดที่มีอยู่ในสภาพแวดล้อมการทำงานของอุปกรณ์เคลื่อนที่ได้อย่างครอบคลุม) 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 (บัญชา ปะลีละเตสัง, 2566 : 111)

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



ภาพประกอบ 1.2 Flutter

ที่มา : Flutter. (2020 : 1)

ภาพประกอบ 1.3 ภาษา Dart

ที่มา : Dart. (2020 : 1)

Note
สามารถนำแอปพลิเคชันที่พัฒนาด้วย Flutter เปลี่ยนเป็นรูปแบบเว็บแอปพลิเคชัน โดยการแปลงภาษา Dart เป็น JavaScript และสั่งการทำงานใน Web Browser สามารถเข้าดูรายละเอียดเพิ่มเติมได้ที่ https://flutter.dev/docs/get-started/web

3. จุดเด่นของ Flutter

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

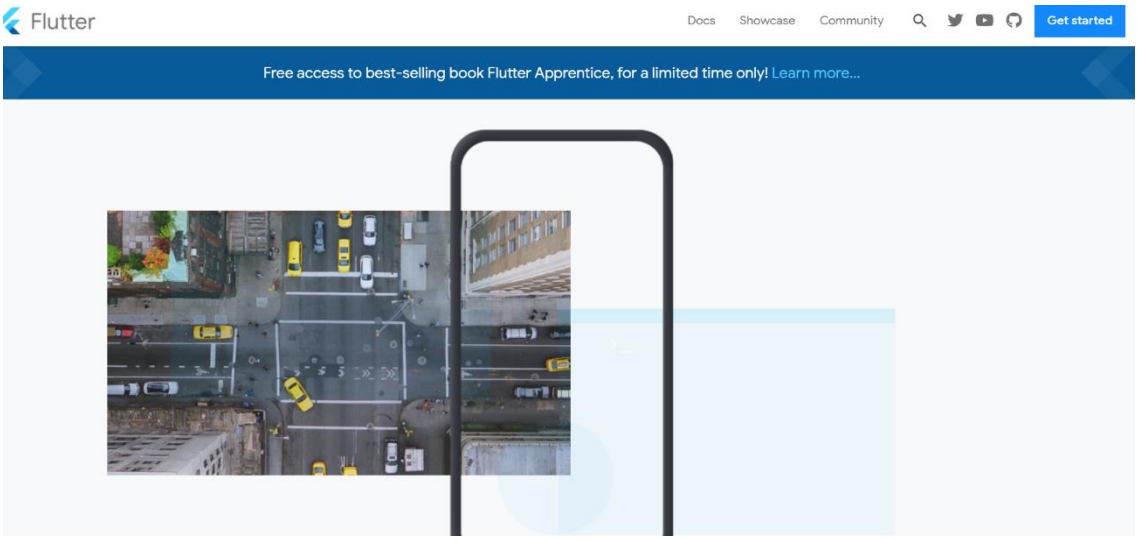
2. การสร้างแอปพลิเคชันใน Flutter จะใช้ภาษาคอมพิวเตอร์ที่เรียกว่า Dart โดยผลลัพธ์แอปพลิเคชันที่ได้จาก Flutter นั้นจะเป็นแอปพลิเคชันในแบบ Native ซึ่งสามารถนำไปใช้กับอุปกรณ์ iOS หรือ Android ได้โดยตรง และสามารถอัปโหลดแอปพลิเคชัน ที่สร้างขึ้นไปยัง Apple App Store (สำหรับ iOS) หรืออัปโหลดไปยัง Google Play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

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

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

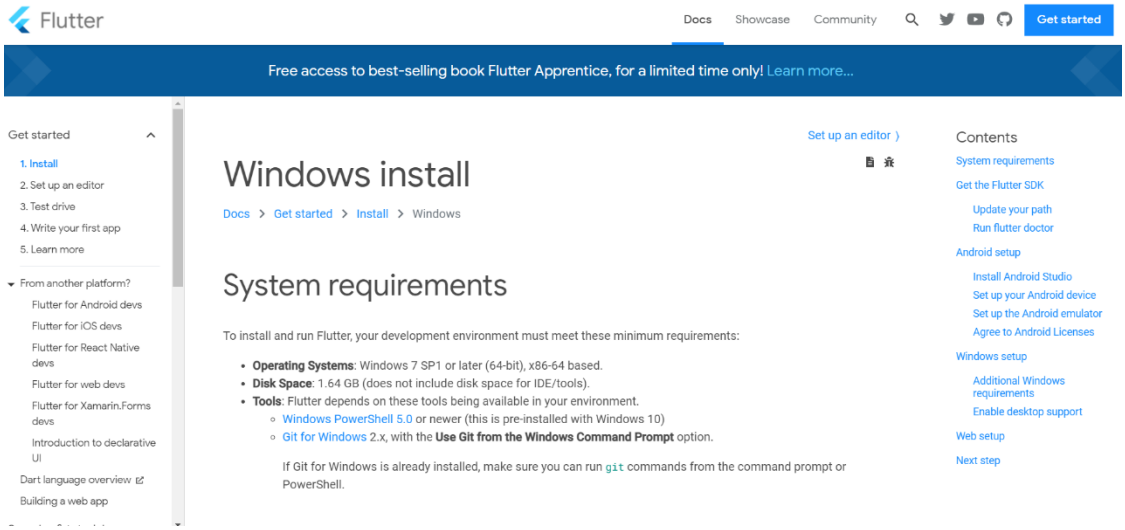
การทดลองใช้งาน Flutter

1. เข้าไปที่เว็บไซต์ https://flutter.dev/



ภาพประกอบ 1.4 หน้าเว็บไซต์ของ Flutter
ที่มา : Flutter. (2020 : 1)

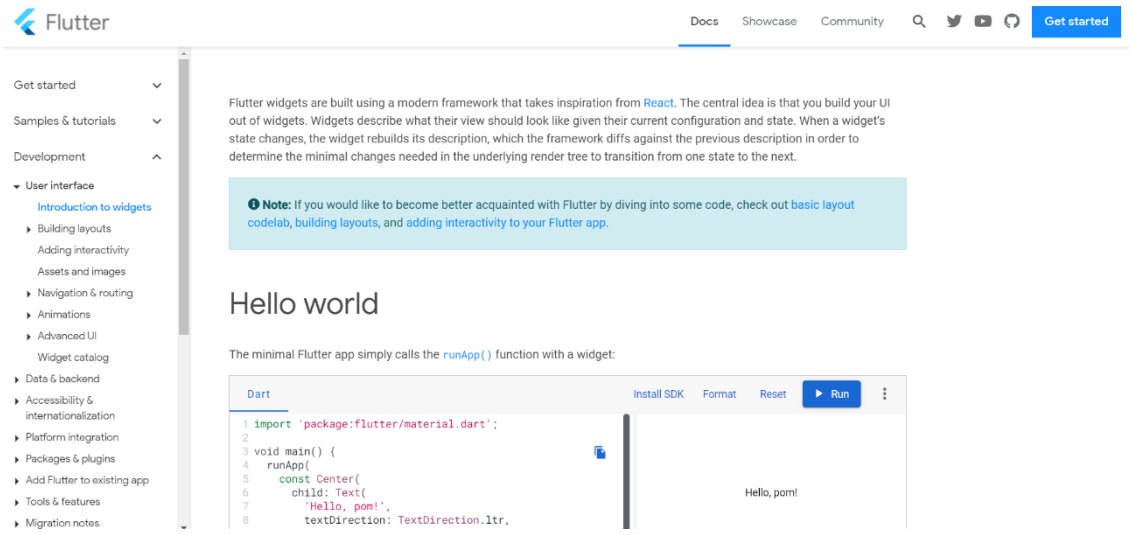
2. ดูความต้องการของระบบ (System Requirement) Flutter



ภาพประกอบ 1.5 ความต้องการของระบบ Flutter

ที่มา : Flutter. (2020 : 1)

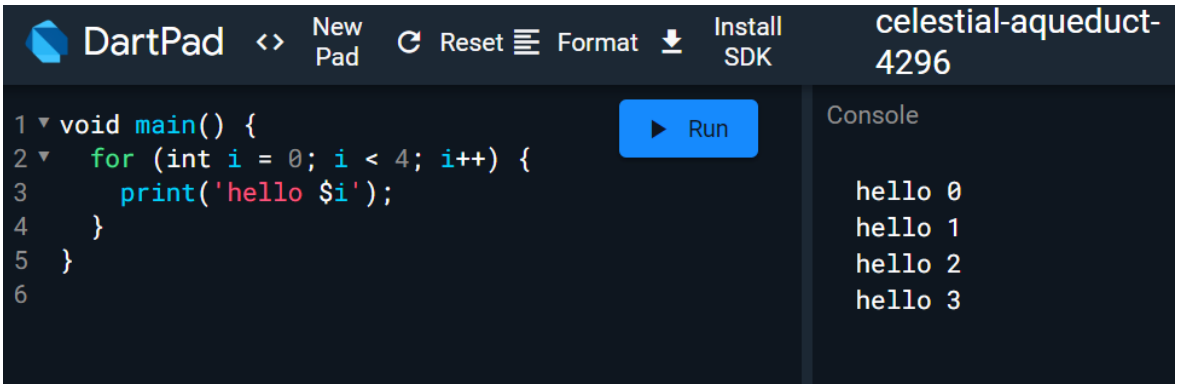
3. เริ่มต้นทดลองการใช้งาน Flutter ด้วยการไปที่ Development > User Interface > Introduction to widgets



ภาพประกอบ 1.6 เริ่มต้นการใช้งาน Flutter (1)

ที่มา : Flutter. (2020 : 1)

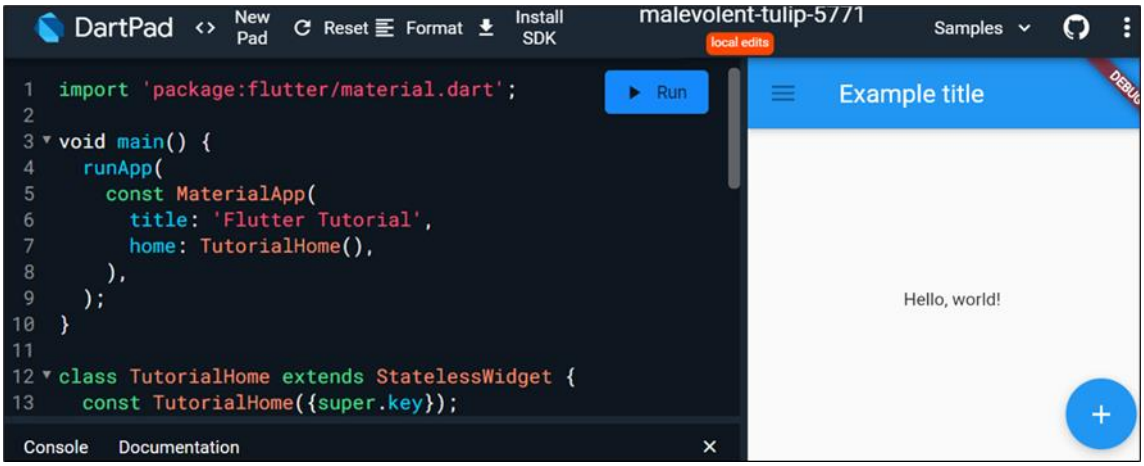
4. ทดลองใช้งาน DartPad โดยไปที่ https://dartpad.dev/ และทดลองสั่งทำงาน



ภาพประกอบ 1.7 เริ่มต้นการใช้งาน Flutter (2)

ที่มา : DardPad. (2020 : 1)

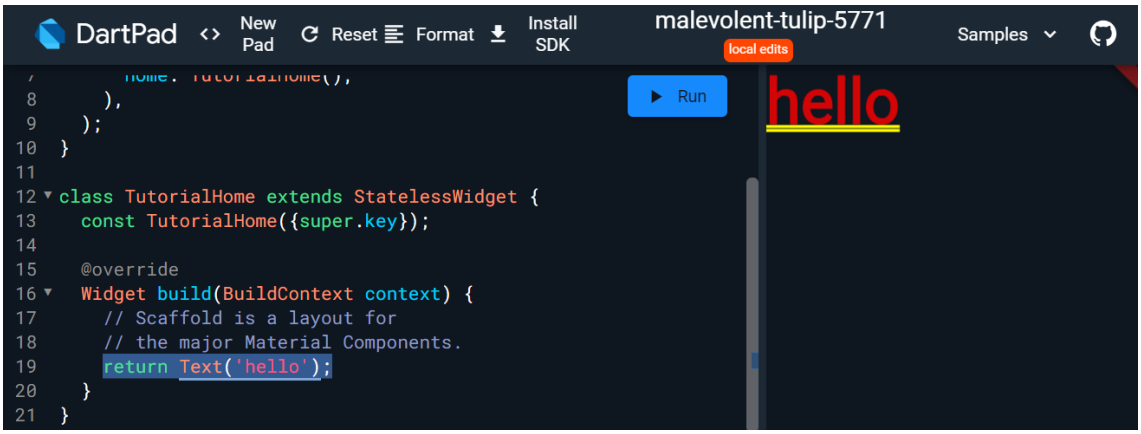
5. นำโค้ดจาก Development > User Interface > Introduction to widgets ใน flutter.dev ของขั้นตอนที่ 3 มาทดลองสั่งทำงานเพื่อดูผลลัพธ์



ภาพประกอบ 1.8 ทดลองใช้งาน Flutter (3)

ที่มา : ณปภัช วรรณตรง (2564 : 18)

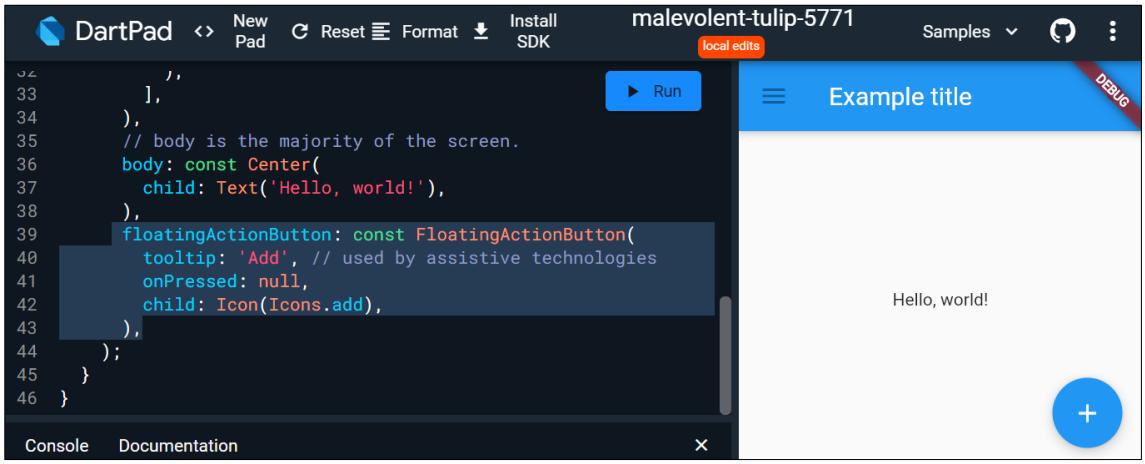
6. ทดลองการเอาส่วนที่ไม่ใช่ Return ออกและทำการใส่ข้อความอื่น ๆ เข้าไปแทน



ภาพประกอบ 1.9 ทดลองใช้งาน Flutter (4)

ที่มา : ณปภัช วรรณตรง (2564 : 20)

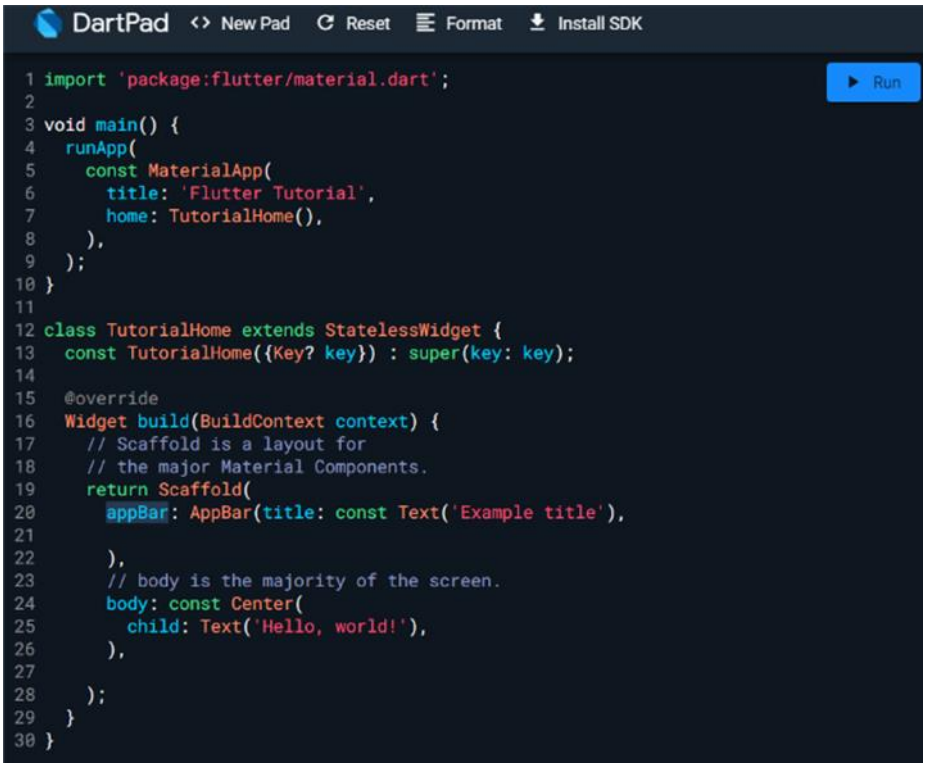
7. ย้อนกลับการทำงานโดยการกด Crtl + Z และทำการเอา floingActionButton ออก



ภาพประกอบ 1.10 ทดลองใช้งาน Flutter (5)

ที่มา : ณปภัช วรรณตรง (2564 : 17)

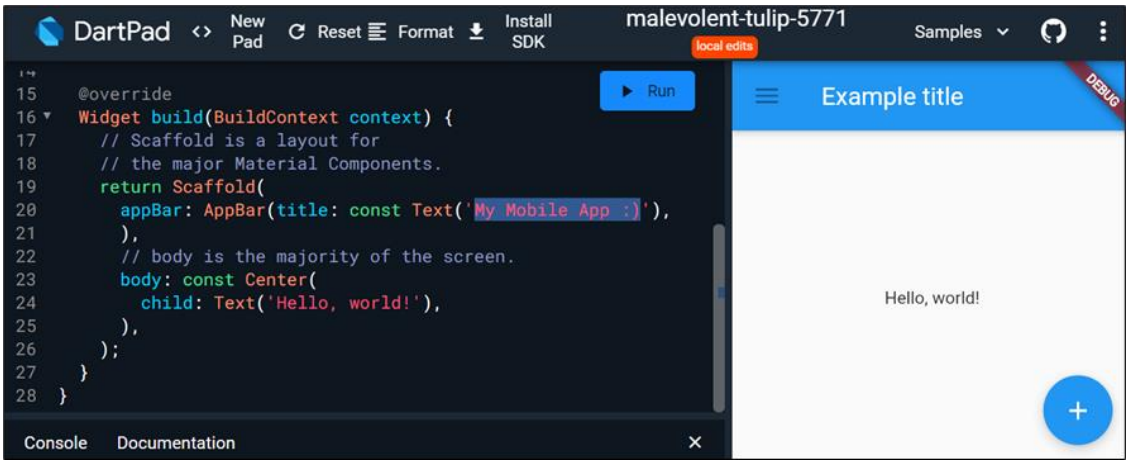
8. ทำการเอาโค้ดบางส่วนของ appBar ออก



ภาพประกอบ 1.11 ทดลองใช้งาน Flutter (6)

ที่มา : ณปภัช วรรณตรง (2564 : 18)

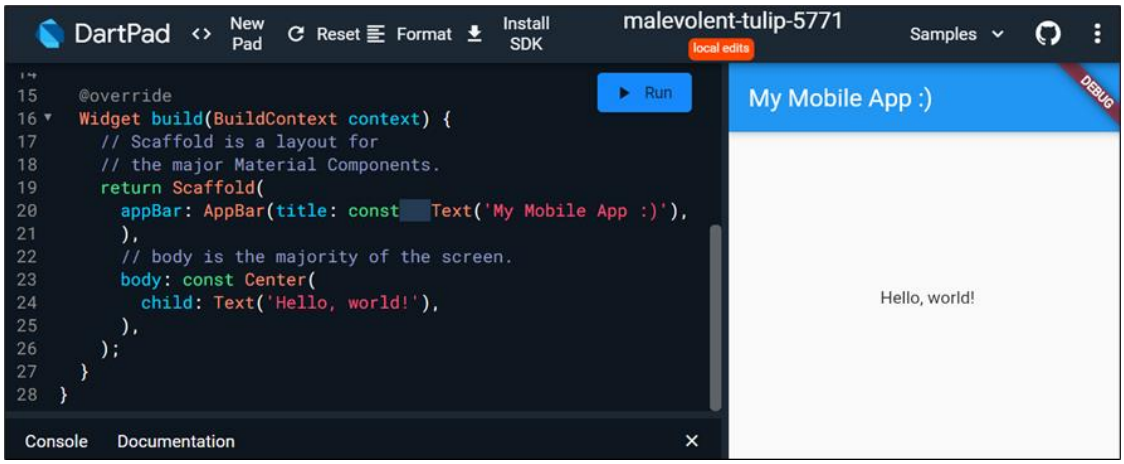
9. ทำการเปลี่ยนข้อความของ title ในส่วนของ appBar



ภาพประกอบ 1.12 ทดลองใช้งาน Flutter (7)

ที่มา : ณปภัช วรรณตรง (2564 : 24)

10. ทำการทดสอบว่าการเว้นเคาะช่องของข้อความ ไม่มีผลเหมือนกันกับภาษา Python



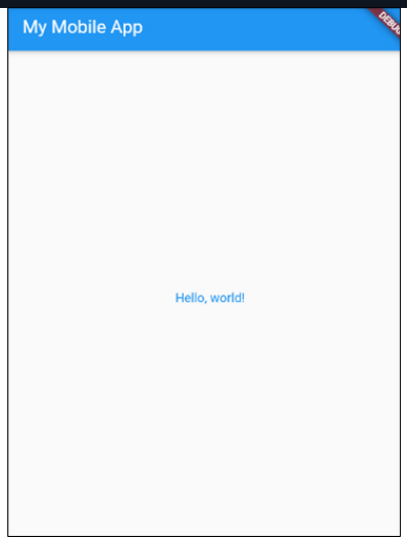
ภาพประกอบ 1.13 ทดลองใช้งาน Flutter (8)

ที่มา : ณปภัช วรรณตรง (2564 : 23)

11. ทำการเพิ่ม Style ให้กับ Font และสั่งทำงานเพื่อดูผลลัพธ์

DartPad New Pad Reset Format Install SDK

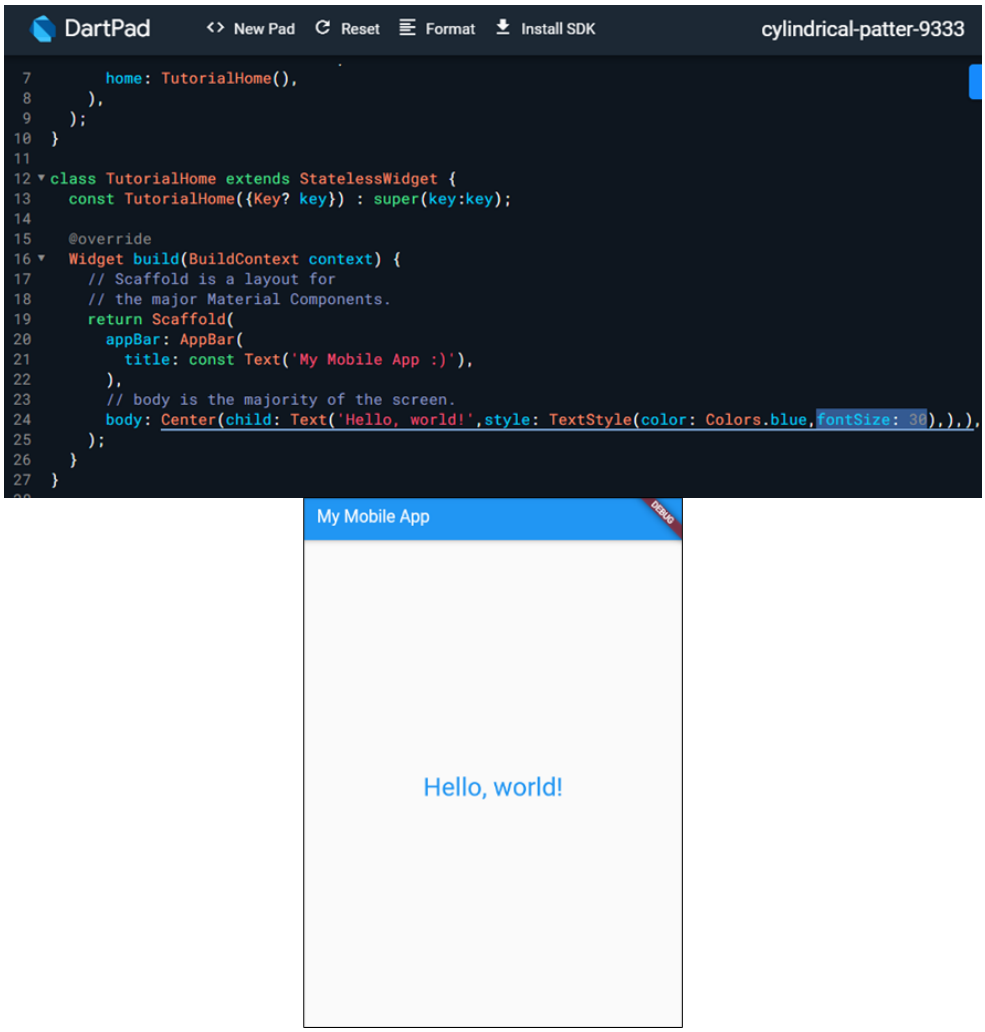
```dart
import 'package:flutter/material.dart';

void main() {
  runApp(
    const MaterialApp(
      title: 'Flutter Tutorial',
      home: TutorialHome(),
    ),
  );
}

class TutorialHome extends StatelessWidget {
  const TutorialHome({Key? key}) : super(key:key);

  @override
  Widget build(BuildContext context) {
    // Scaffold is a layout for
    // the major Material Components.
    return Scaffold(appBar: AppBar(title: const Text('My Mobile App :)'),
      ),
      // body is the majority of the screen.
      body: Center(child: Text('Hello, world!',style: TextStyle(color: Colors.blue),),),
    );
  }
}
```



ภาพประกอบ 1.14 ทดลองใช้งาน Flutter (9)

ที่มา : ณปภัช วรรณตรง (2564 : 25)

12. ทำการเพิ่มขนาดให้กับ Font



ภาพประกอบ 1.15 ทดลองใช้งาน Flutter (10)

ที่มา : ณปภัช วรรณตรง (2564 : 26)

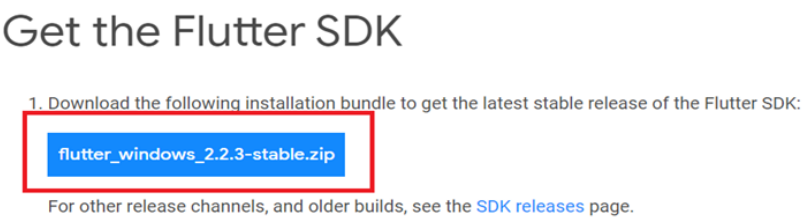
สรุป Flutter
1) ชุดโปรแกรมด้าน User Interface ถูกสร้างโดย Google 2) จัดการส่วนต่าง ๆ ได้ง่ายและสะดวก 3) เป็น Open Source ใช้งานได้ฟรี 4) พัฒนา Mobile Application ให้สามารถใช้งานได้ทั้ง iOS และ Android

การเริ่มต้นสร้างแอปพลิเคชันด้วย Flutter

1. ลงโปรแกรมที่เกี่ยวข้อง

ดาวน์โหลดโปรแกรม Android Studio สำหรับใช้ในการแสดงผลการทำงานแอปพลิเคชัน

1.1 สำหรับระบบปฏิบัติการ Windows



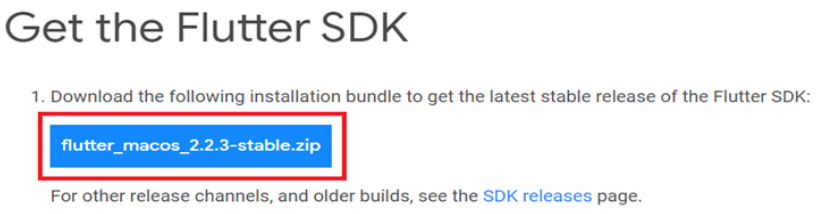
ภาพประกอบ 1.16 Flutter สำหรับระบบปฏิบัติการ Windows

ที่มา : Flutter. (2020 : 1)

1.1.1 ทำการดาวน์โหลด Flutter SDK ผ่านทางเว็บบราวน์เซอร์ของผู้พัฒนา

1.1.2 เมื่อทำการดาวน์โหลดเสร็จแล้ว ให้ทำการแยกไฟล์ที่ดาวน์โหลดมา ไปเก็บไว้ยังตำแหน่งของโฟลเดอร์ที่ต้องการ เช่น ตำแหน่งโฟลเดอร์ที่ไดร์ฟ C:\

1.2 สำหรับระบบปฏิบัติการ macOS



ภาพประกอบ 1.17 Flutter สำหรับระบบปฏิบัติการ macOS

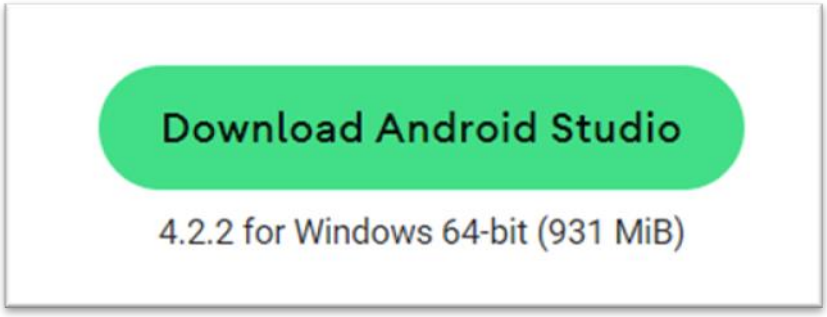
ที่มา : Flutter. (2020 : 1)

1.2.1 ทำการดาวน์โหลด Flutter SDK ผ่านทางเว็บบราวน์เซอร์ของผู้พัฒนา

1.2.2 เมื่อทำการดาวน์โหลดเสร็จแล้ว ให้ทำการแยกไฟล์ที่ดาวน์โหลดมา ไปเก็บไว้ยังตำแหน่งของโฟลเดอร์ที่ต้องการ เช่น ตำแหน่งโฟลเดอร์ที่ไดร์ฟ C:\

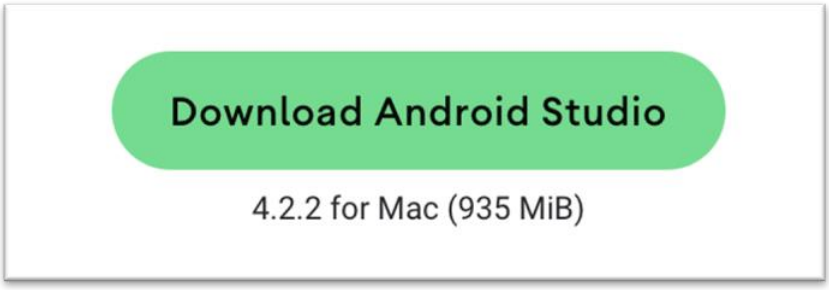
ดาวน์โหลดโปรแกรมพัฒนาแอปพลิเคชันบนมือถือ Android Studio

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 ผ่านทางเว็บบราวน์เซอร์ของผู้พัฒนา

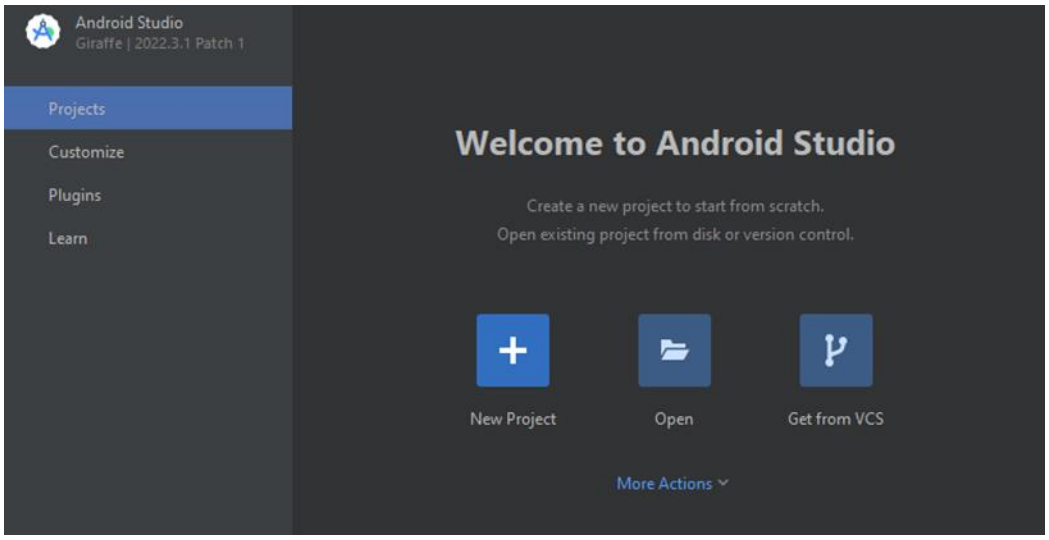
1.3.2 เมื่อดาวน์โหลดเสร็จแล้วให้ทำการเปิด Android Studio ที่ได้ทำการดาวน์โหลดมา

1.3.3 เลือกการติดตั้งเป็นหมวด Android Virtual Device

1.3.4 กดถัดไป (Next) จนถึงหน้าตัวเลือก SDK จากนั้นทำการเลือกที่ SDK แล้วกดถัดไปจนถึงกระบวนการติดตั้ง หลังจากนั้นให้รอจนทำการติดตั้งเสร็จสิ้น

1.3.5 หลังจากทำการติดตั้งจนเสร็จสิ้นแล้วให้ทำการเปิดโปรแกรม Android Studio ขึ้นมา

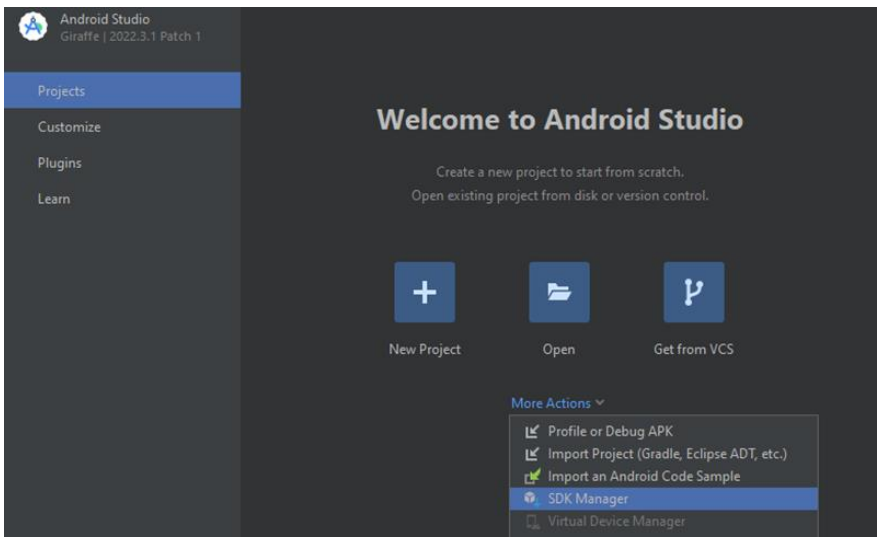
เมื่อเปิดโปรแกรม Android Studio ขึ้นมาแล้วให้ทำการตั้งค่าเครื่องมือสำหรับการพัฒนา



ภาพประกอบ 1.20 ตั้งค่าเครื่องมือสำหรับการพัฒนา

ที่มา : ณปภัช วรรณตรง (2564 : 17)

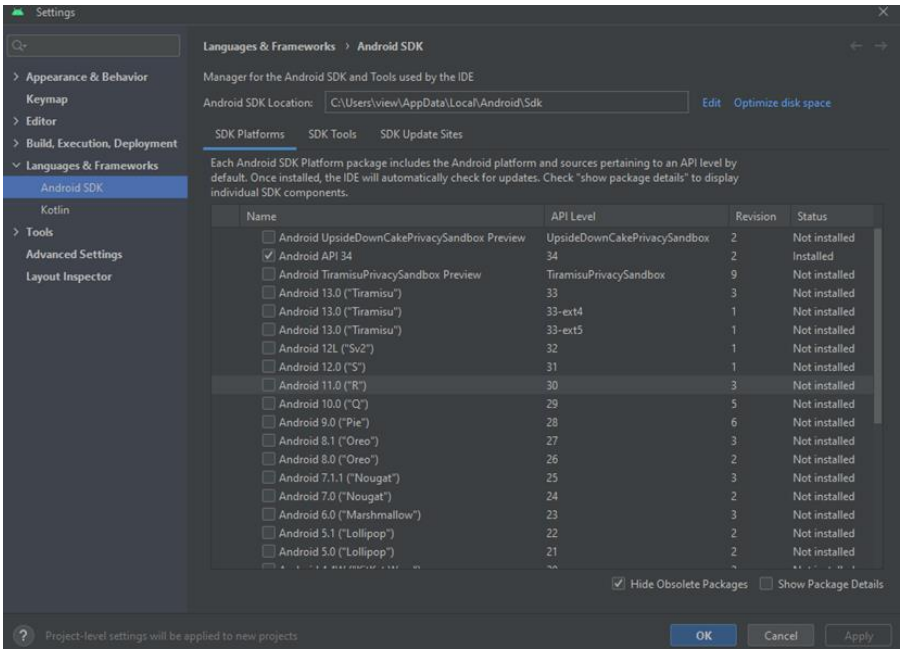
เลือกที่ More Action จากนั้นให้เลือกเป็น SDK Manager



ภาพประกอบ 1.21 ตั้งค่าเครื่องมือสำหรับการพัฒนา (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 17)

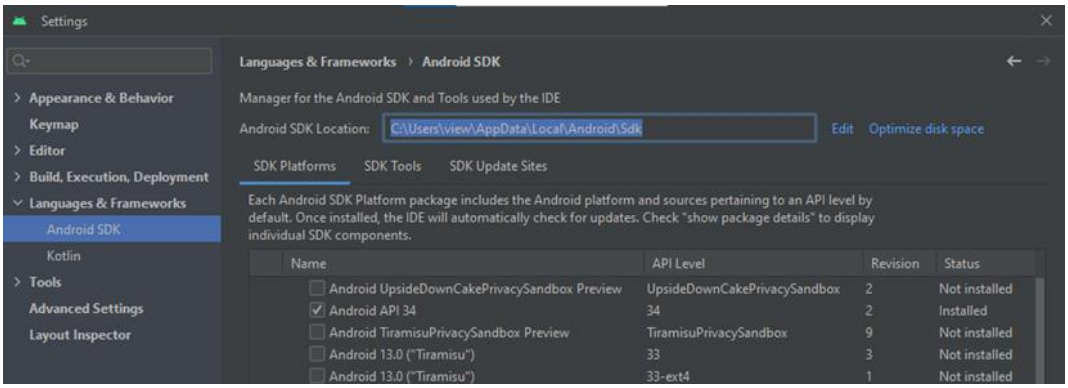
เมื่อเลือก SDk Manager แล้วจะปรากฏหน้าต่างการตั้งค่าขึ้นมา



ภาพประกอบ 1.22 ตั้งค่าเครื่องมือสำหรับการพัฒนา (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 18)

จากนั้นให้ทำการคัดลอก Path เอาไว้ สำหรับนำไปกำหนด Path ใน Edit Environment



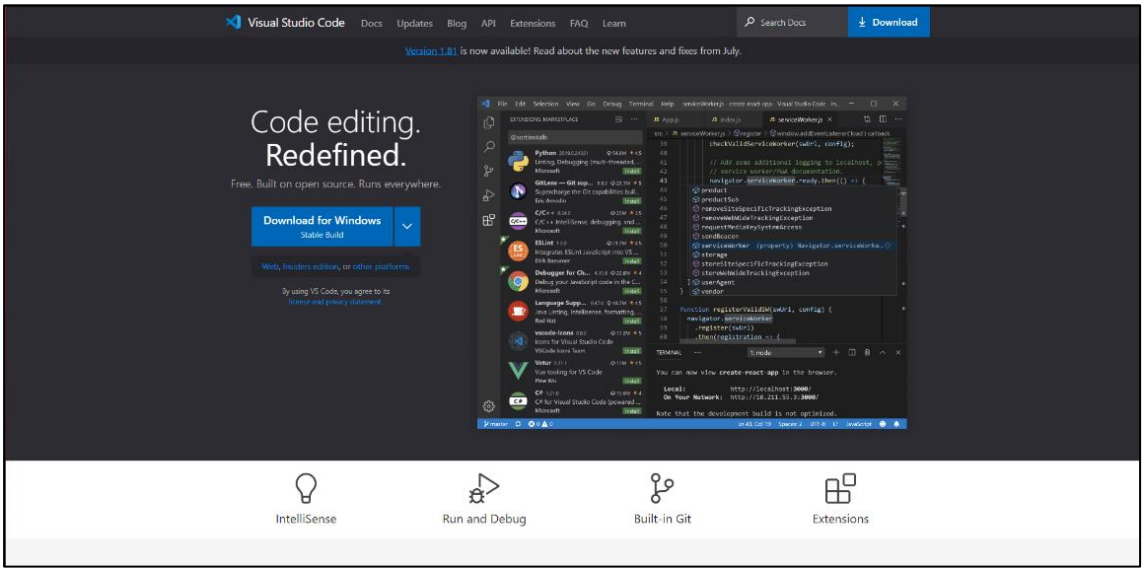
ภาพประกอบ 1.23 ตั้งค่าเครื่องมือสำหรับการพัฒนา (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 18)

หมายเหตุ
การเรียนการสอนจริงอาจสั่งการทำงานผ่านบราวเซอร์ เนื่องจาก Android Studio ใช้หน่วยความจำในการทำงานจำนวนมาก คอมพิวเตอร์ในห้องปฏิบัติคอมพิวเตอร์ที่ใช้เรียนอาจไม่สามารถรองรับการใช้งานได้

ดาวน์โหลดโปรแกรม Editor ที่ใช้ในการเขียน Flutter คือ Virtual Studio Code

1.4 สามารถดาวน์โหลดได้ทั้งระบบปฏิบัติการ Windows และ macOS



ภาพประกอบ 1.24 Virtual Studio Code สำหรับทุกระบบปฏิบัติการ

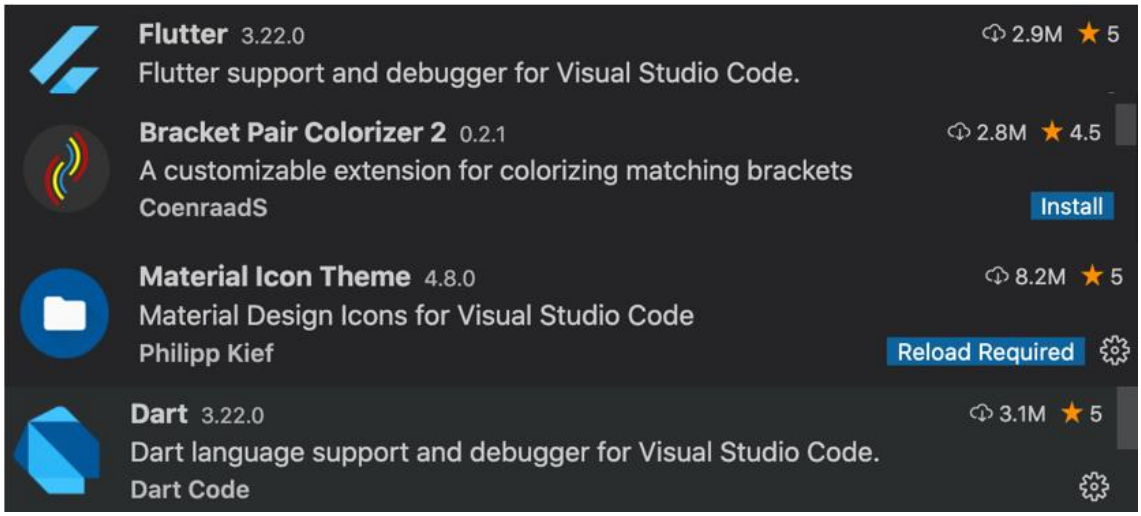
ที่มา : Visual Studio Code. (2020 : 1)

1.4.1 ทำการดาวน์โหลด Virtual Studio Code ผ่านทางเว็บบราวน์เซอร์ของผู้พัฒนา

1.4.2 เมื่อดาวน์โหลดเสร็จแล้วให้ทำการเปิด Virtual Studio Code ที่ได้ทำการดาวน์โหลดมา

1.4.3 ทำการติดตั้งโปรแกรม Virtual Studio Code ให้เสร็จสมบูรณ์

1.4.4 หลังจากติดตั้งเสร็จสิ้นแล้วให้ทำการติดตั้งส่วนเสริมดังภาพ



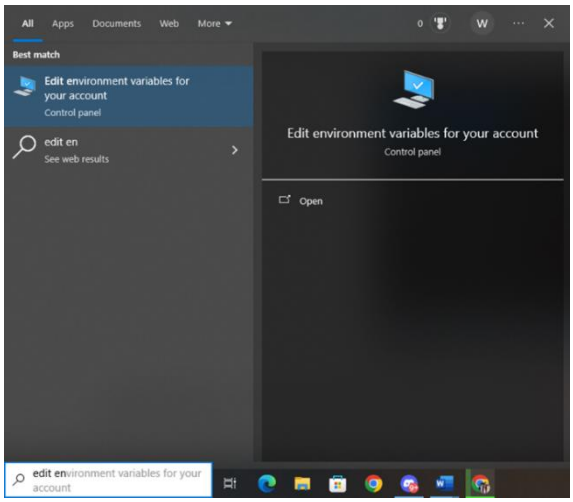
ภาพประกอบ 1.25 ส่วนเสริมการใช้งานผ่าน Virtual Studio Code

ที่มา : ณปภัช วรรณตรง (2564 : 10)

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

2. กำหนดค่าในการทำงานของโปรแกรม Set Path

2.1 ไปที่ Edit Environment

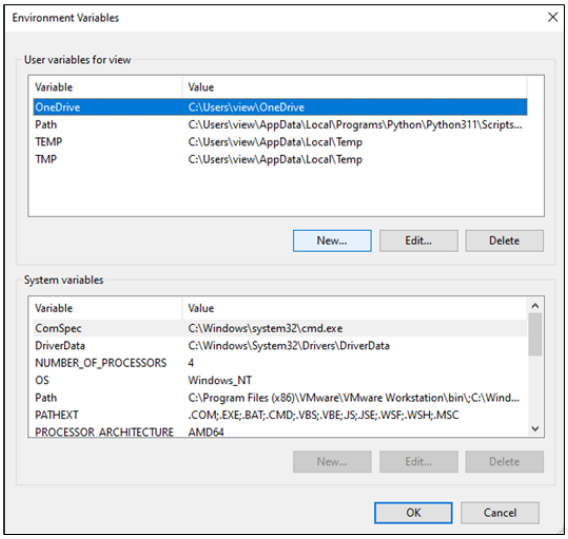


ภาพประกอบ 1.26 กำหนดค่า ANDROID_HOME ใน Edit Environment

ที่มา : ณปภัช วรรณตรง (2564 : 19)

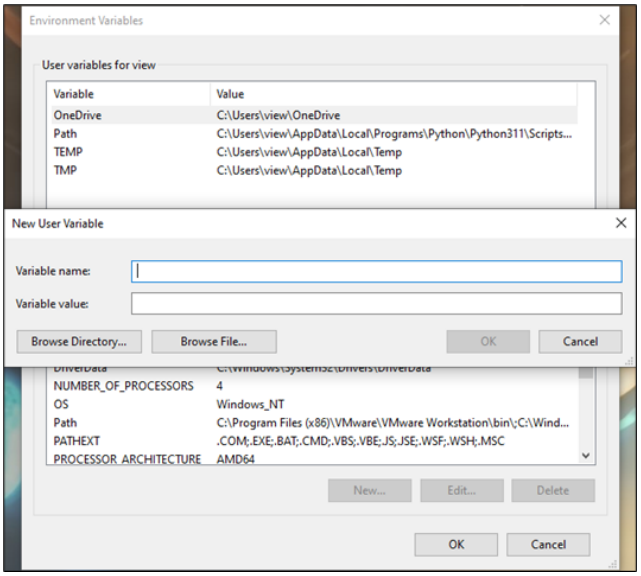
2.2 สร้างและกำหนดค่า ANDROID_HOME ให้มี Path เป็น Path เดียวกับใน Android Studio ที่ได้ทำการคัดลอกไว้ก่อนหน้า

เมื่อเปิดหน้าต่าง Edit Environment แล้วให้ทำการไปที่ New เพิ่มทำการเพิ่ม Path



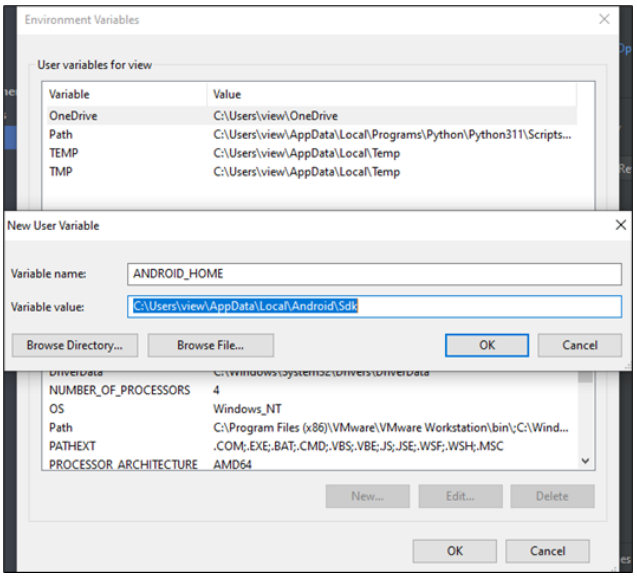
ภาพประกอบ 1.27 กำหนดค่า ANDROID_HOME ใน Edit Environment (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 20)



ภาพประกอบ 1.28 กำหนดค่า ANDROID_HOME ใน Edit Environment (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 20)



ภาพประกอบ 1.29 สร้างและกำหนดค่า ANDROID_HOME

ที่มา : ณปภัช วรรณตรง (2564 : 20)

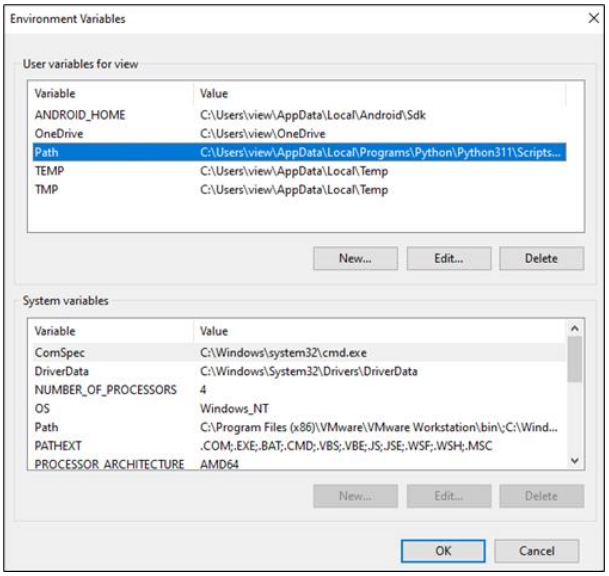
2.3 ไปที่ Path สร้าง Path ใหม่ ดังนี้

C:\flutter\bin

%ANDROID_HOME%tools

%ANDROID_HOME%platform-tools

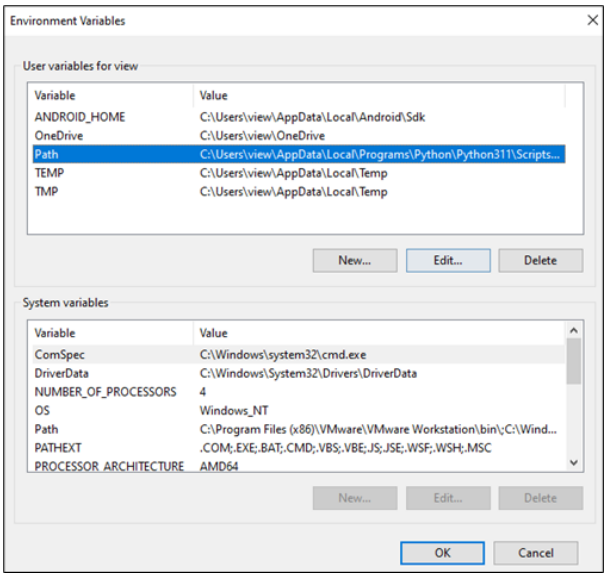
2.3.1 ไปที่ Path



ภาพประกอบ 1.30 สร้าง Path ใหม่

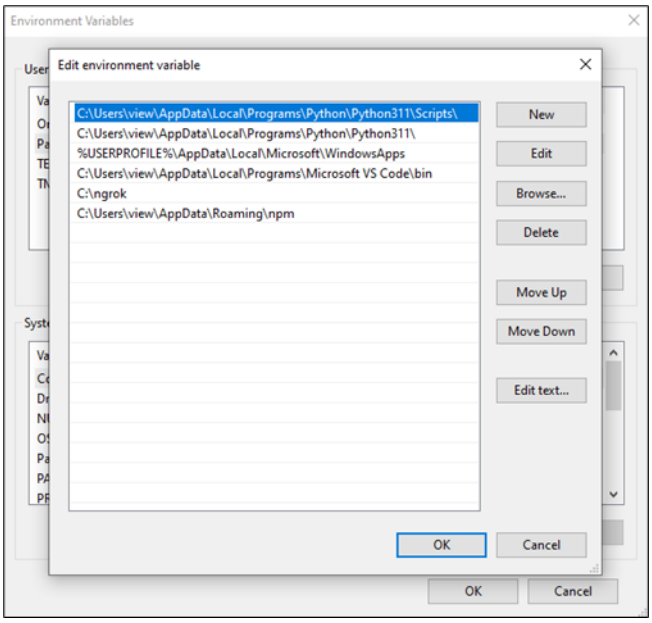
ที่มา : ณปภัช วรรณตรง (2564 : 21)

2.3.2 จากนั้นให้กดที่ Edit



ภาพประกอบ 1.31 สร้าง Path ใหม่ (ต่อ)

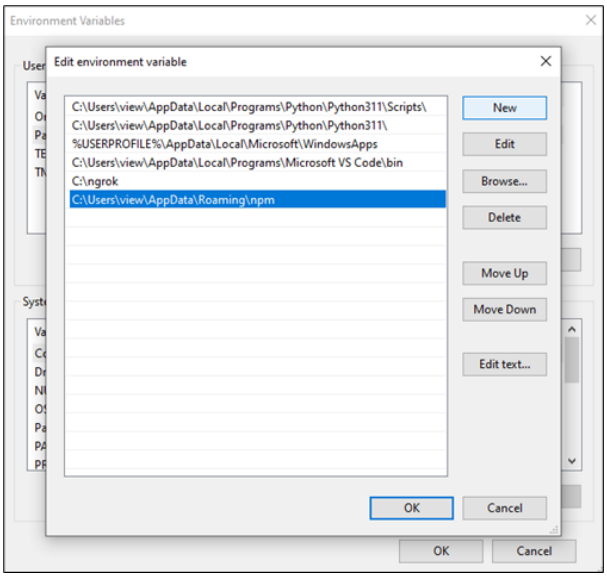
ที่มา : ณปภัช วรรณตรง (2564 : 21)



ภาพประกอบ 1.32 สร้าง Path ใหม่ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 21)

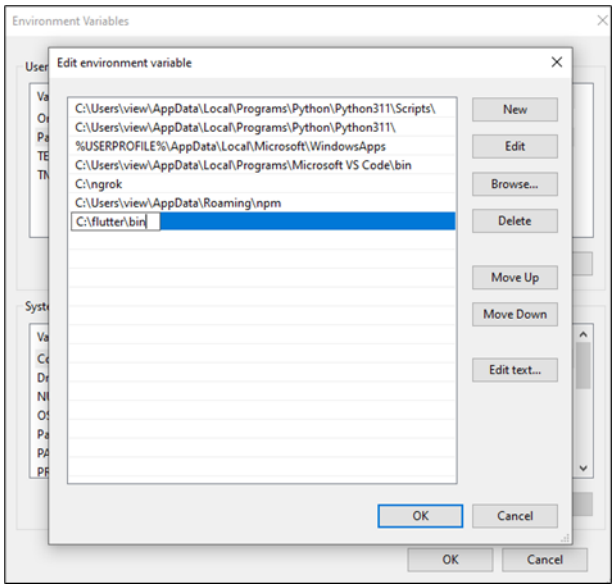
2.3.3 เลือก Path สุดท้ายจากนั้นกด New



ภาพประกอบ 1.33 สร้าง Path ใหม่ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 21)

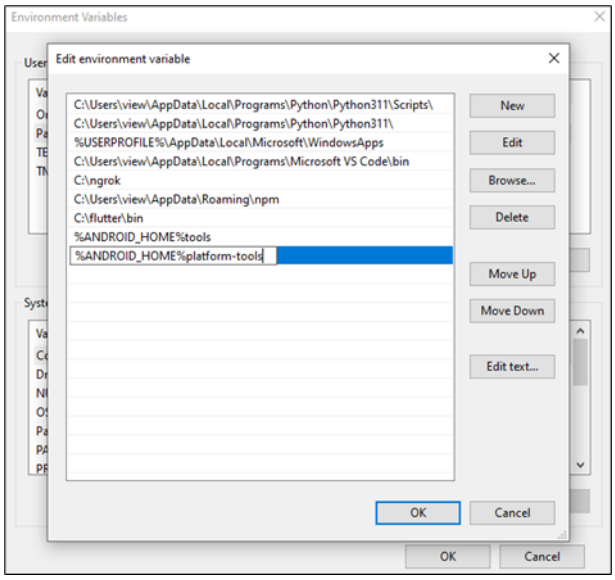
2.3.4 จากนั้นให้ทำการกำหนด Path ที่ได้กำหนดไว้



ภาพประกอบ 1.34 สร้าง Path ใหม่ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 21)

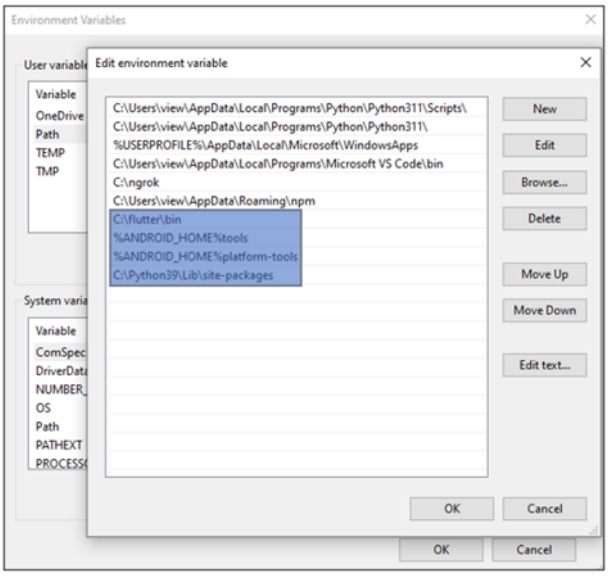
2.3.5 จากนั้นกำหนด Path ให้ครบและกด OK



ภาพประกอบ 1.35 สร้าง Path ใหม่ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 21)

2.3.6 ได้ผลลัพธ์การเพิ่ม Path ดังภาพ



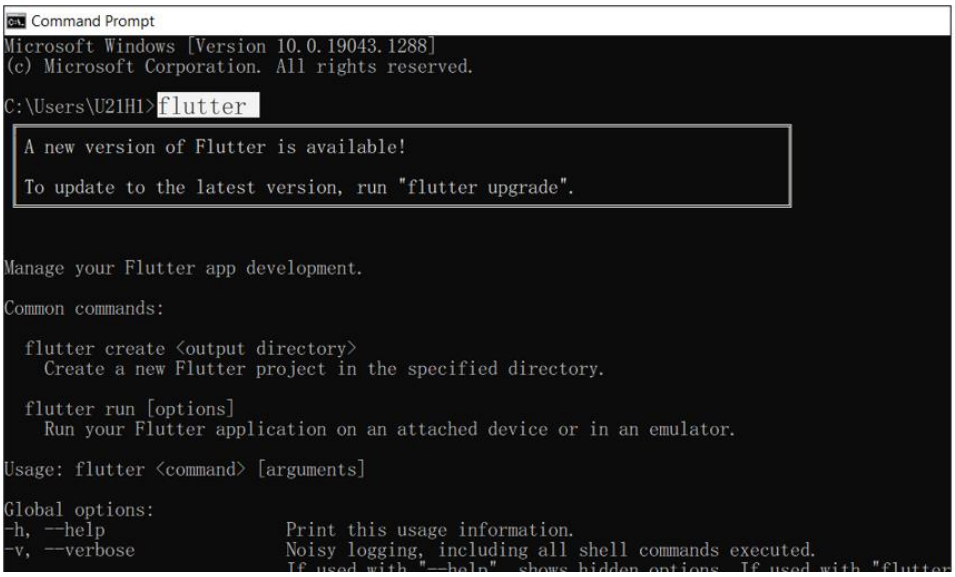
ภาพประกอบ 1.36 สร้าง Path ใหม่ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 21)

3. ตรวจสอบการใช้งานด้วย Command Prompt

ตรวจสอบการใช้งานด้วย Command Prompt ได้ดังนี้

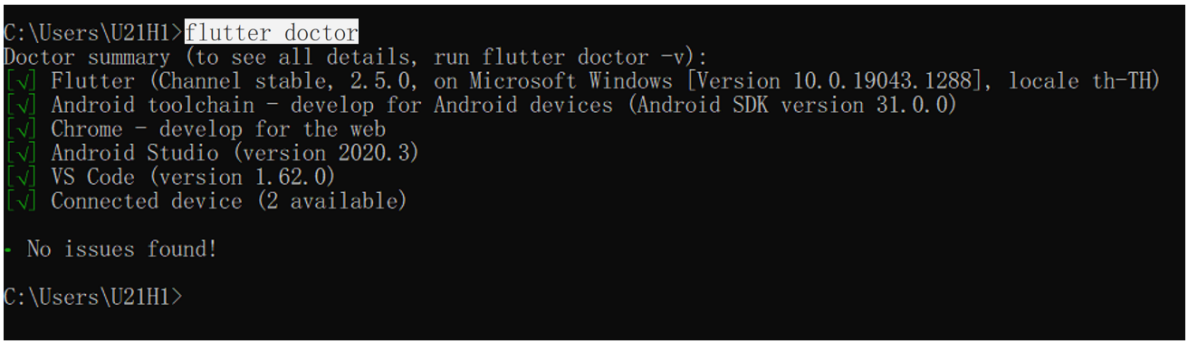
3.1 ตรวจสอบ Flutter ว่าสามารถใช้งานได้หรือไม่ ด้วยการพิมพ์ Flutter



ภาพประกอบ 1.37 Flutter

ที่มา : ณปภัช วรรณตรง (2564 : 23)

3.2 ตรวจสอบว่าโปรแกรมต่าง ๆ ที่ติดตั้งไว้สามารถใช้งานได้หรือไม่ ด้วยการพิมพ์ Flutter Doctor กรณี “ผ่าน” จะได้ผลดังภาพ



ภาพประกอบ 1.38 Flutter Doctor

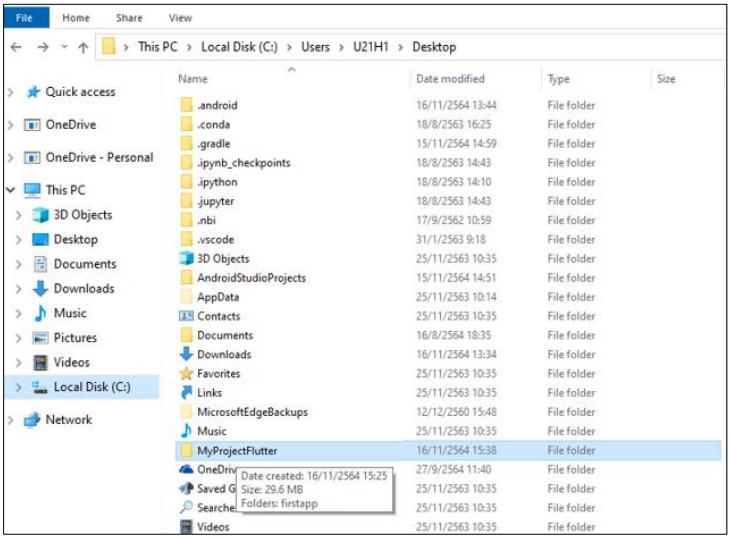
ที่มา : ณปภัช วรรณตรง (2564 : 23)

3.2 Flutter Doctor กรณี “ไม่ผ่าน” ให้ทำการคัดลอกส่วนที่ผิดพลาดหรือข้อผิดพลาดไปค้นหาวิธีแก้ไขในอินเทอร์เน็ตบราวน์เซอร์

หมายเหตุ
Flutter doctor กรณี “ไม่ผ่าน” ถ้าเจอ error ใช้คำสั่งต่อไปนี้ 1) ถ้าเจอคำว่า Android Studio not Installed flutter config --android-studio-dir = “C:\Program Files\Android\Android Studio” Path ใน “” ให้ใส่ Path ที่เก็บจริงของเครื่องผู้ใช้ 2) ถ้าเจอคำว่า Android Accept License flutter doctor --android-licenses ถ้าผ่าน Android Accept License จะขึ้นให้กด y enter เพื่อ Accept ตลอด 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 4) ถ้าเจอคำว่า No devices available วิธีแก้ปัญหา No device รันคำสั่งต่อไปนี้ ทีละคำสั่ง flutter channel beta flutter upgrade flutter config --enable-web (ที่มา: cr. https://stackoverflow.com/questions/49175231/flutter-does-not-find- android-sdk)

4. เริ่มสร้าง Project

4.1 สร้าง Folder ตั้งชื่อตามต้องการ



ภาพประกอบ 1.39 สร้าง Folder

ที่มา : ณปภัช วรรณตรง (2564 : 30)

4.2 cd เข้า Folder ที่สร้าง ใช้คำสั่ง cd

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

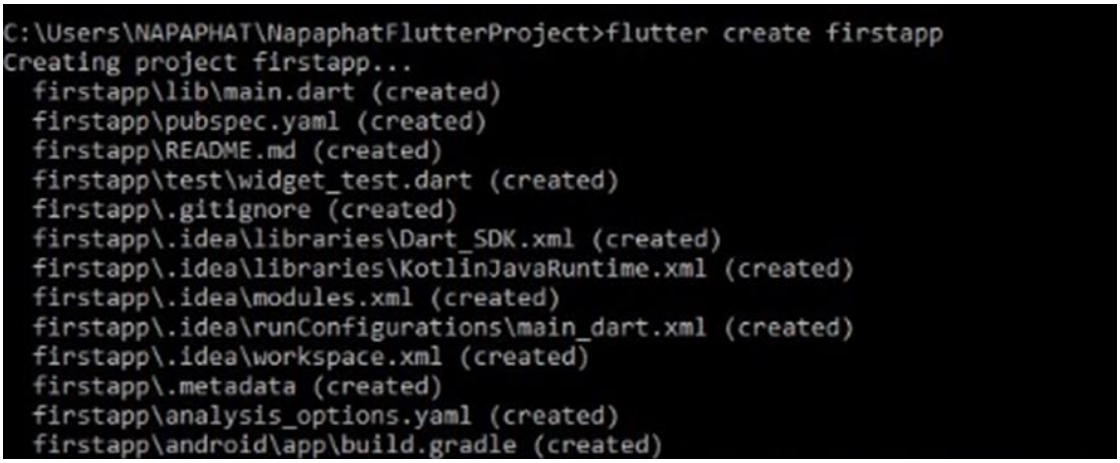
ภาพประกอบ 1.40 cd เข้า Folder ที่สร้าง

ที่มา : ณปภัช วรรณตรง (2564 : 31)

4.3 สร้าง Project ด้วยคำสั่ง flutter create ตามด้วยชื่อที่ต้องการ (แต่ต้องเป็นตัวอักษรภาษาอังกฤษขนาดเล็กทั้งหมด)

ภาพประกอบ 1.41 สร้าง Project

ที่มา : ณปภัช วรรณตรง (2564 : 32)



ภาพประกอบ 1.42 สร้าง Project (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 32)

4.4 ใช้คำสั่ง cd ชื่อ Project ที่สร้างไว้ที่ขั้นตอน 4.3 เพื่อเข้าสู่ Folder Project ที่สร้าง

ภาพประกอบ 1.43 ใช้คำสั่ง cd firstapp เพื่อเข้าสู่ Folder Project

ที่มา : ณปภัช วรรณตรง (2564 : 32)

4.5 ใช้คำสั่ง Flutter run เพื่อสั่งการทำงาน project

ภาพประกอบ 1.44 ใช้คำสั่ง flutter run เพื่อสั่งทำงาน

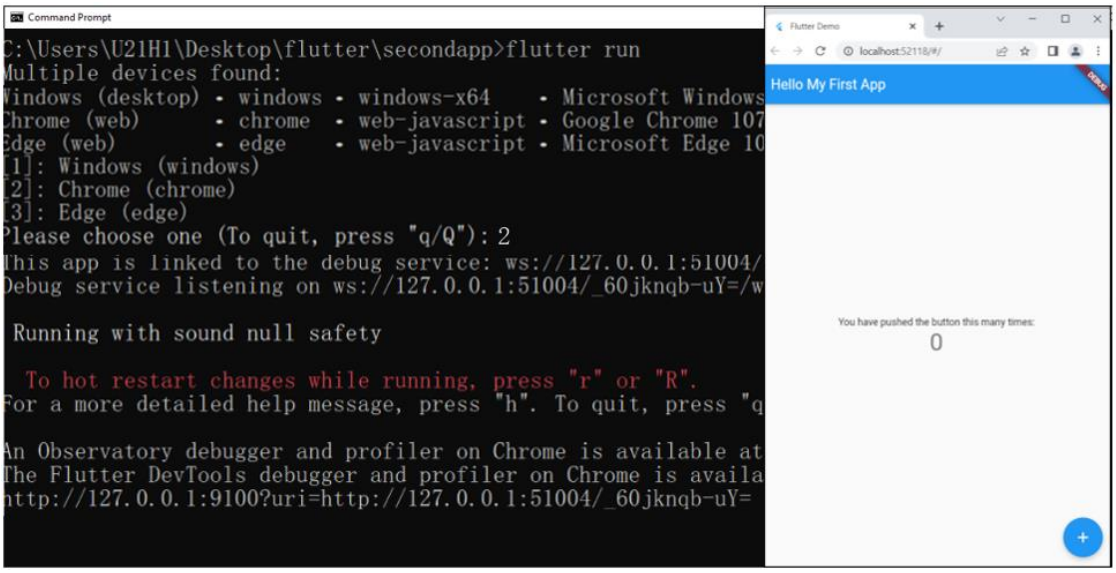
ที่มา : ณปภัช วรรณตรง (2564 : 33)

4.6 เมื่อทำการติดตั้งและสั่งทำการทำงาน Flutter เป็นครั้งแรก Flutter จะให้ทำการเลือก Device สำหรับเปิดหน้าเว็บบราวน์เซอร์

โดยจะมีตัวเลือกอยู่สองอย่าง 1) Chrome 2) Edge สามารถเลือกเว็บบราวน์เซอร์สำหรับการแสดงหน้าเว็บแอบฯ ได้ตามต้องการ

ภาพประกอบ 1.45 เลือก Device

ที่มา : ณปภัช วรรณตรง (2564 : 35)

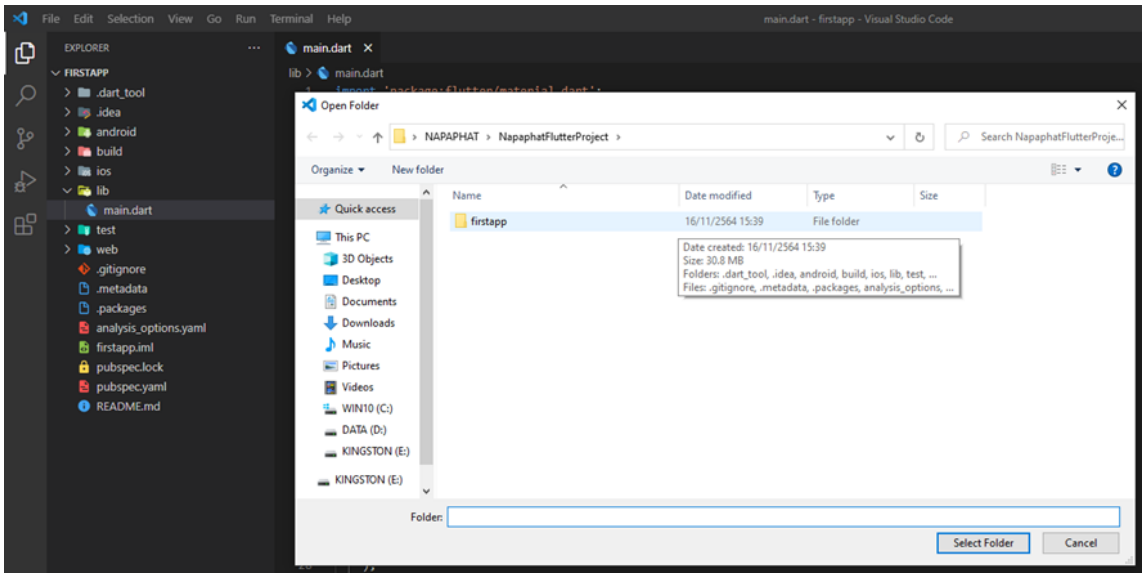


ภาพประกอบ 1.46 เลือก Device (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 36)

5. เปิดไฟล์ใน VS Code

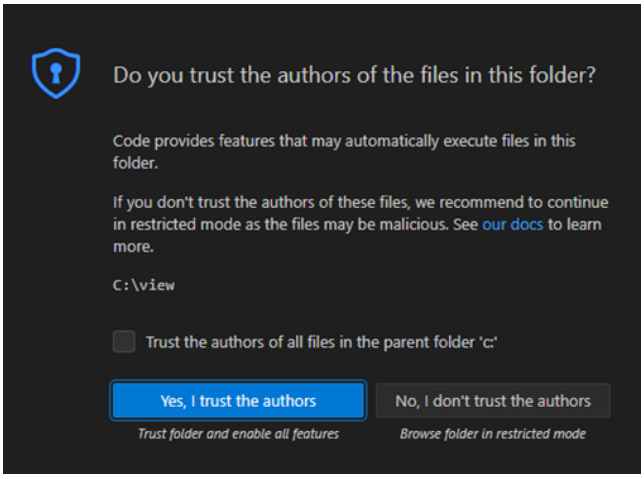
5.1 เปิด VS Code เลือก Folder Project



ภาพประกอบ 1.47 เปิด VS Code

ที่มา : ณปภัช วรรณตรง (2564 : 38)

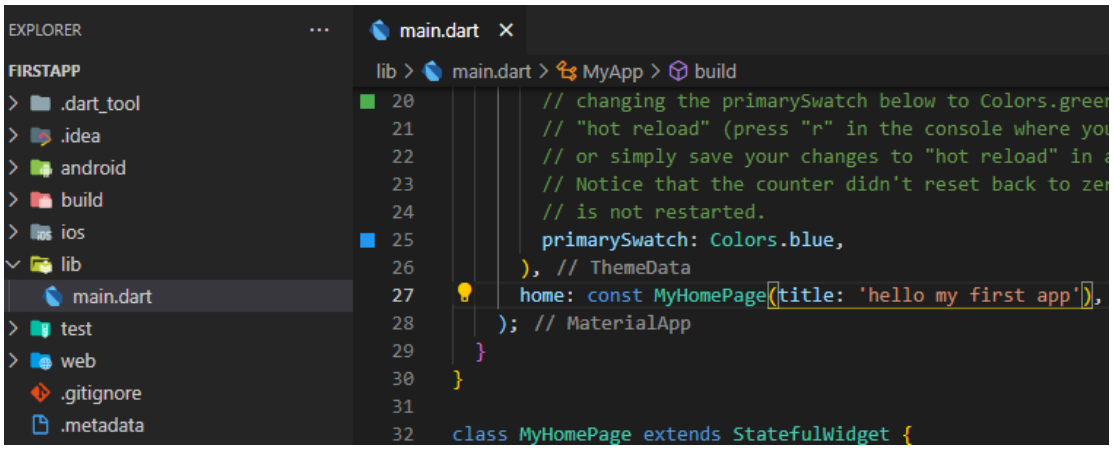
หากขึ้นหน้าต่างสอบถามความน่าเชื่อถือ ให้ตอบ Yes



ภาพประกอบ 1.48 เปิด VS Code (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 38)

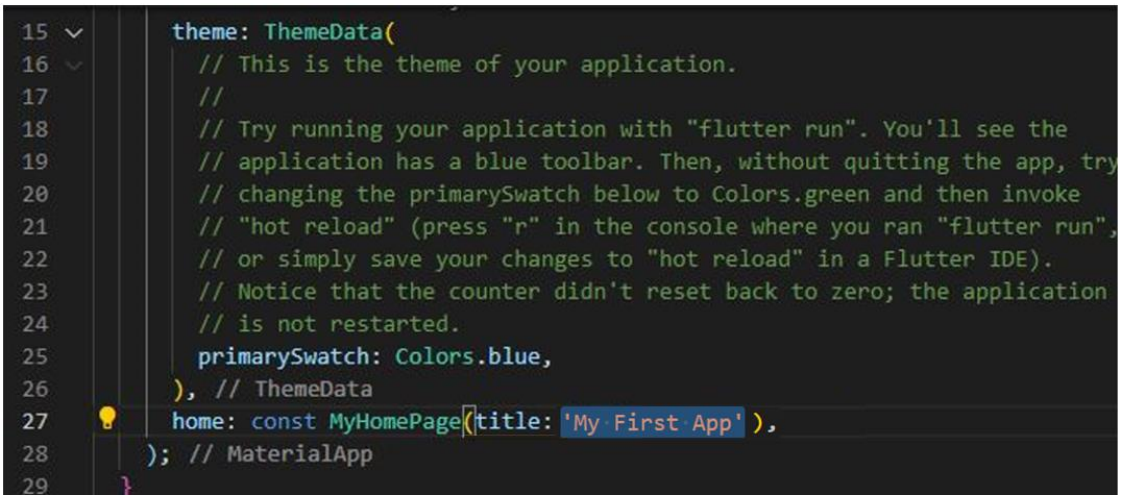
5.2 เปิด Folder lib > เลือกไฟล์ main.dart



ภาพประกอบ 1.49 เปิด Folder lib

ที่มา : ณปภัช วรรณตรง (2564 : 39)

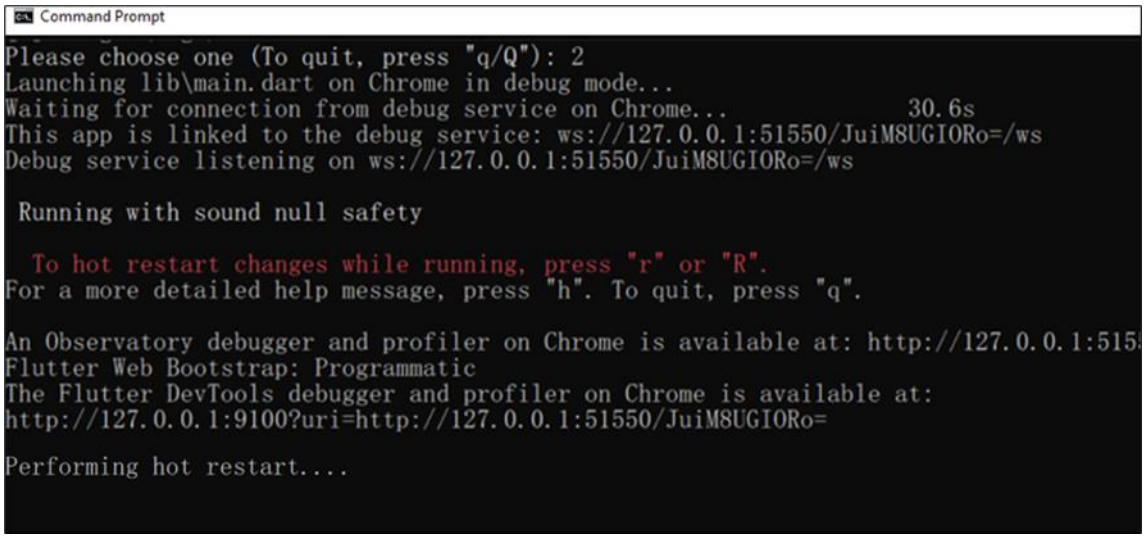
5.3 แก้ไฟล์ main.dart ตรง home:



ภาพประกอบ 1.50 แก้ไฟล์ main.dart

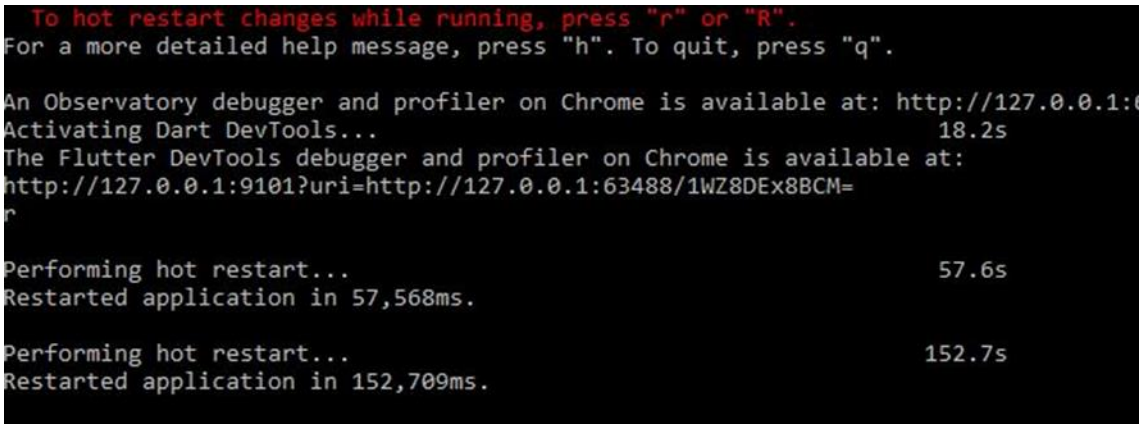
ที่มา : ณปภัช วรรณตรง (2564 : 40)

5.4 Save > แล้วกด r/Shift+R เพื่อ Restart



ภาพประกอบ 1.51 Restart

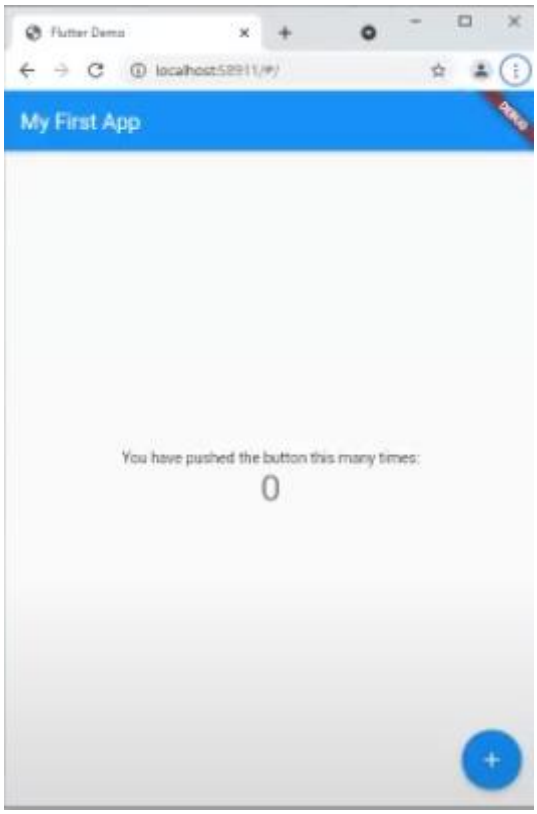
ที่มา : ณปภัช วรรณตรง (2564 : 41)



ภาพประกอบ 1.52 Restart (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 41)

5.5 จะได้ผลการสั่งการทำงานดังภาพ จะสังเกตเห็นว่าแถบข้อความด้านบนแอปพลิเคชันเปลี่ยนไปตามที่ได้แก้ไข

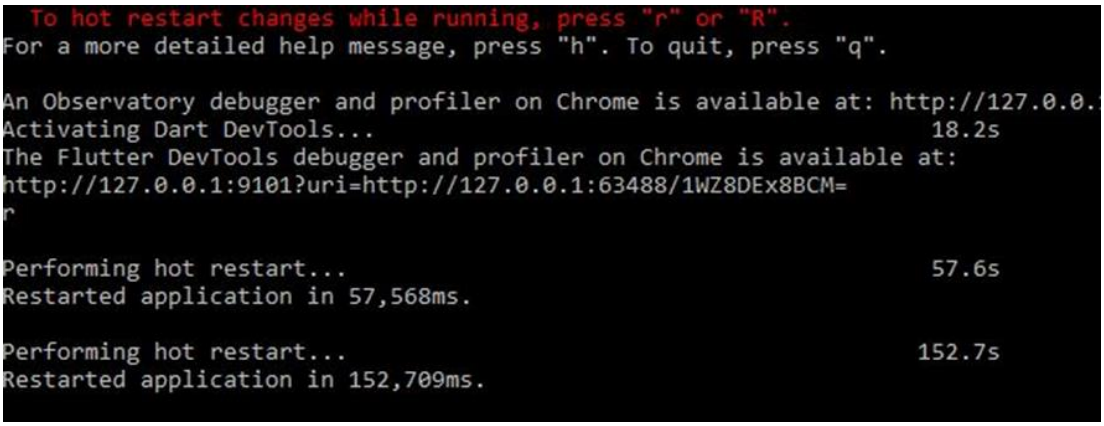


ภาพประกอบ 1.53 ผลลัพท์การแก้ไขโค้ด home:

ที่มา : ณปภัช วรรณตรง (2564 : 42)

หมายเหตุ
หากพิมพ์ r จะทำการแก้ไขแค่เพียงบางส่วนใช้เวลาสั้น shift + r จะทำการแก้ไขทั้งหมดใช้เวลานาน

5.5 กด Cltl+C ในหน้าต่าง cmd กรณีค้างหรือการสั่งการทำงานนานเกินไป



ภาพประกอบ 1.54 ยกเลิกการทำงานเมื่อเกิดอาการค้าง

ที่มา : ณปภัช วรรณตรง (2564 : 43)

ภาพประกอบ 1.55 ยกเลิกการทำงานเมื่อเกิดอาการค้าง (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 43)

บทสรุป

สำหรับเนื้อหาที่กล่าวมาทั้งหมดในบทนี้ จะพูดถึงการพัฒนาแอปพลิคชันบนอุปกรณ์เคลื่อนที่ในปัจจุบัน ความเป็นมาของ Flutter ภาษาที่ใช้ในการพัฒนาแอปพลิเคชันใน Flutter วิธีการดาวน์โหลดการติดตั้งเครื่องมือในการพัฒนา และการตั้งค่าการใช้งานเครื่องมือต่าง ๆ การกำหนดค่าในการทำงานของโปรแกรมและการเริ่มต้นสร้างแอปพลิเคชันด้วย Flutter

เอกสารอ้างอิง

จีราวุธ วารินทร์. (2564). **พัฒนาโมไบล์แอพด้วย Flutter + Dart.** กรุงเทพฯ : สำนักพิมพ์ซิมพลิฟาย

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

สำนักงานสถิติแห่งชาติ. (2564). **สรุปผลที่สำคัญสำรวจการมีการใช้เทคโนโลยีสารสนเทศและการสื่อสารในครัวเรือน 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). **พัฒนาแอพด้วย Flutter สำหรับผู้เริ่มต้น 7 ชั่วโมงเต็ม [FULL COURSE].** สืบค้นจาก, 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 แล้วทำมาแสดงผลบนหน้าจอ มีชื่อเรียกชิ้นส่วนต่าง ๆ ในหน้าจอนี้ว่า วิดเจ็ต (Widget) ในบทนี้ผู้เรียนจะได้เรียนรู้ว่า วิดเจ็ตคืออะไร ได้ใช้งานวิดเจ็ตพื้นฐานที่อยู่ในภาษา Dart ที่เป็นภาษาที่ใช้ในการพัฒนาแอปพลิคชันด้วย Flutter ได้เรียนรู้ประเภทของวิดเจ็ตว่ามีประเภทใดบ้าง และแต่ละประเภทมีลักษณะอย่างไร วิธีการสร้างวิดเจ็ต การเริ่มสร้าง Flutter Project และขั้นตอนการสร้างวิดเจ็ตในโปรเจกต์ วิธีการตกแต่งตัวอักษร การดาวน์โหลดรูปแบบตัวอักษรที่ต้องการ ตั้งค่ารูปแบบตัวอักษร กำหนดขนาด สี ลักษณะของตัวอักษร จากเนื้อหาข้างต้นที่กล่าวมาทั้งหมด มีรายละเอียดดังต่อไปนี้

ทำความรู้จักวิดเจ็ตคืออะไร

1. วิดเจ็ต (Widget) คืออะไร

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

วิดเจ็ตประกอบด้วยเครื่องมือและส่วนควบคุมต่าง ๆ ได้แก่ ปุ่มไอคอน เมนูแบบเลื่อน กล่องตัวเลือก สวิตช์ เป็นต้น สิ่งต่าง ๆ เหล่านี้ Flutter ได้จัดตรียมไว้ โดยเรียกกลุ่มเครื่องมือนี้ว่า วิดเจ็ต แค็ตตาล็อก (วิดเจ็ตCatalog) (เอกรินทร์ วทัญญูเลิศสกุล, 2563 : 67)

2. ประเภทของ Widget

วิดเจ็ตที่ใช้ในแอปพลิเคชันสามารถแบ่งได้เป็น 3 ประเภท ดังนี้ (จีราวุธ 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 (เอกรินทร์ วทัญญูเลิศสกุล, 2563 : 107)

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

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

2.1.5 Container เป็นวิดเจ็ตสำหรับการบรรจุหรือจัดวางวิดเจ็ตย่อย ๆ เพื่อกำหนดโครงร่างตามต้องการ (บัญชา 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 ซึ่งจะทำให้ได้วิดเจ็ตในแบบที่ต้องการทันที

3. วิดเจ็ตสร้างขึ้นเอง (Custom Widget)

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

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

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

ตัวอย่างโค้ดแบบ Stateless Widget ดังภาพ

```dart
class HelloWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

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 วทัญญูเลิศสกุล, 2563 : 103)

ตัวอย่างโค้ดแบบ Stateless Widget ดังภาพ

```dart
class HelloWidget extends StatefulWidget {
  @override
  _HelloWidgetState createState() => _HelloWidgetState();
}

class _HelloWidgetState extends State<HelloWidget> {

  @override
  void initState() {
    // init something.
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

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 สามารถกำหนด 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

```
void main() {
  var app = MaterialApp(
      title: "My App",
      home: Scaffold(
        appBar: AppBar(
          title: Text("My App"),
        ),
        body: Text("สวัสดีครับ")
      ),
    );
  runApp(app);
}
```

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

```
void main() {
  var app = MaterialApp(
      title: "My App",
      home: Scaffold(
        appBar: AppBar(
          title: Text("My App"),
        ),
        body: Center(child: Text("สวัสดีครับ"))
      ),
    );
  runApp(app);
}
```

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

```dart
import 'package:flutter/material.dart';

void main() {
  var app = MaterialApp(
      title: "My App",
      home: Scaffold(
        appBar: AppBar(
          title: Text("My App"),
        ),
        body: Center(child: Column(
          children: [
            Text("สวัสดีครับ"),
            Text("Thinnakorn"),
          ],
        ))
      ),
    );
  runApp(app);
}
```

ภาพประกอบ 2.5 ตัวอย่างโค้ดแบบ Column Widget (ต่อ)
ที่มา : mindphp. (2565)

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

ตัวอย่างโค้ดแบบ Row วิดเจ็ตดังภาพ

```dart
import 'package:flutter/material.dart';

void main() {
  var app = MaterialApp(
      title: "My App",
      home: Scaffold(
        appBar: AppBar(
          title: Text("My App"),
        ),
        body: Center(child: Row(
          children: [
            Text("สวัสดีครับ"),
            Text("Thinnakorn")
          ],
        ))
      ),
    );
  runApp(app);
}
```

ภาพประกอบ 2.6 ตัวอย่างโค้ดแบบ Row Widget
ที่มา : mindphp. (2565)

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

ตัวอย่างโค้ดแบบ Container Widget ดังภาพ

```dart
import 'package:flutter/material.dart';

void main() {
  var app = MaterialApp(
    title: "My App",
    home: Scaffold(
        appBar: AppBar(
          title: Text("My App"),
        ),
        body: Column(
          children: [

            Container(
              margin: EdgeInsets.only(
                top: 10, //กำหนดระยะห่างจากด้านบน
                bottom: 10, //กำหนดระยะห่างจากด้านล่าง
                left: 10, //กำหนดระยะห่างจากด้านซ้าย
                right: 10, //กำหนดระยะห่างจากด้านขวา
              ),
              decoration: BoxDecoration(color: Colors.redAccent),
              height: 100, //กำหนดความสูง
              width: 200, //กำหนดความกว้าง
            ),
          ],
        )),
  );
  runApp(app);
}
```

t

ภาพประกอบ 2.7 ตัวอย่างโค้ดแบบ Container Widget

ที่มา : mindphp. (2565)

รูปแบบการแสดงผลแอปพลิเคชันที่สร้างจาก Flutter สามารถแบ่งออกได้เป็น 2 แนวทาง ดังนี้

1. แบบที่ 1 Material Design เป็นการออกแบบหน้าตาในสไตล์ของ Google
2. แบบที่ 2 Cupertino Design เป็นการออกแบบหน้าตาแอพพลิเคชันในสไตล์ของ iOS

(จีราวุธ วารินทร์, 2564 : 147)

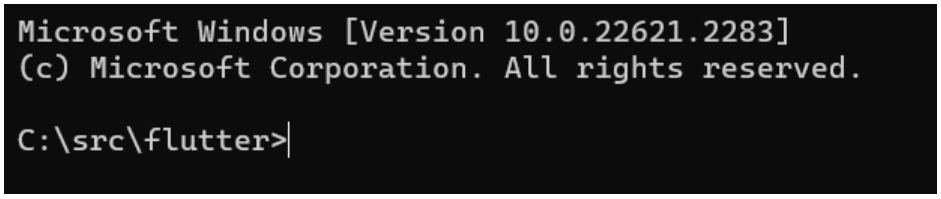
หมายเหตุ
สามารถศึกษาเพิ่มเติมเกี่ยวกับวิดเจ็ตได้ที่เว็บไซต์ https://flutter.dev/widgets

เริ่มขั้นตอนการสร้าง Project

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

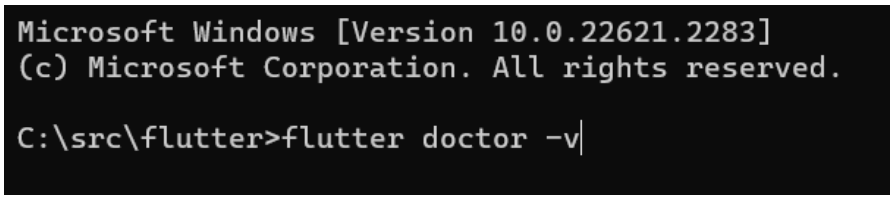
ภาพประกอบ 2.8 เข้าหน้าต่าง cmd ผ่าน Path ของ Folder
ที่มา : ณปภัช วรรณตรง (2564 : 3)

ภาพประกอบ 2.9 เข้าหน้าต่าง cmd ผ่าน Path ของ Folder (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 3)



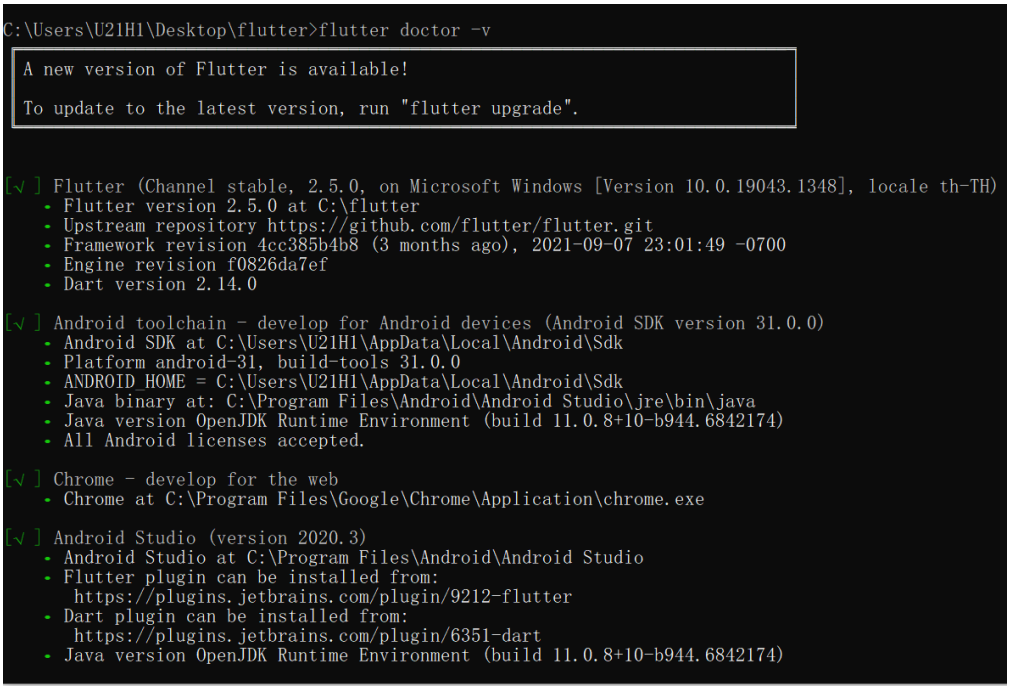
ภาพประกอบ 2.10 เข้าหน้าต่าง cmd ผ่าน Path ของ Folder (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 3)

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



ภาพประกอบ 2.11 พิมพ์คำสั่งสำหรับสร้างไฟล์ Flutter > flutter doctor -v

ที่มา : ณปภัช วรรณตรง (2564 : 4)



ภาพประกอบ 2.12 พิมพ์คำสั่งสำหรับสร้างไฟล์ Flutter > flutter doctor -v

ที่มา : ณปภัช วรรณตรง (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 : ณปภัช วรรณตรง (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 : ณปภัช วรรณตรง (2564 : 5)

เมื่อการทำงานของคำสั่งเสร็จสิ้นแล้ว Folder งานจะปรากฏขึ้นใน Folder ที่ได้ทำการเลือกไว้

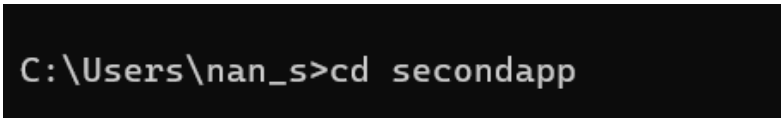
ภาพประกอบ 2.15 ผลการสร้างไฟล์งาน Project

ที่มา : ณปภัช วรรณตรง (2564 : 5)

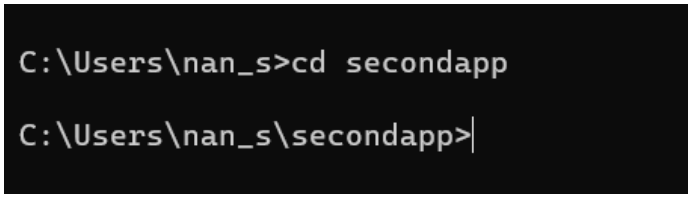
หมายเหตุ
ชื่อ Project ไม่สามารถเว้นเคาะว่างได้ต้องเขียนติดกัน

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

4.1 ให้ทำการเข้าไปยัง Folder งานที่ได้ทำการสร้างขึ้นมา โดยใช้คำสั่ง cd secondapp

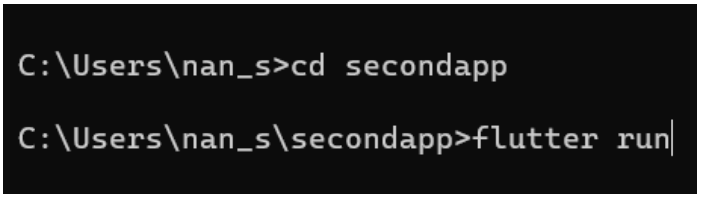


ภาพประกอบ 2.16 เข้าไปยัง Folder งาน
ที่มา : ณปภัช วรรณตรง (2564 : 6)



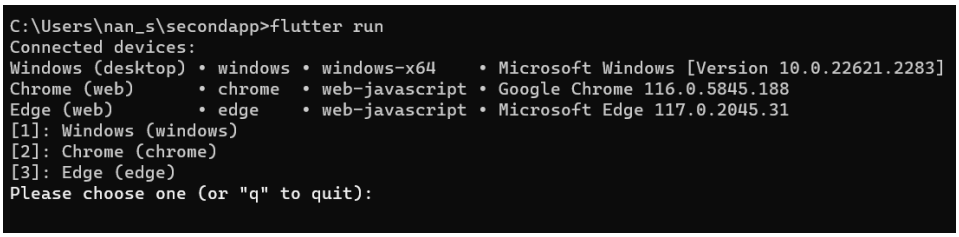
ภาพประกอบ 2.17 เข้าไปยัง Folder งาน (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 6)

4.2 จากนั้นให้ทำการพิมพ์คำสั่ง flutter run ลงไปเพื่อสั่งการทำงานของ flutter



ภาพประกอบ 2.18 สั่งการทำงานด้วย flutter run
ที่มา : ณปภัช วรรณตรง (2564 : 7)

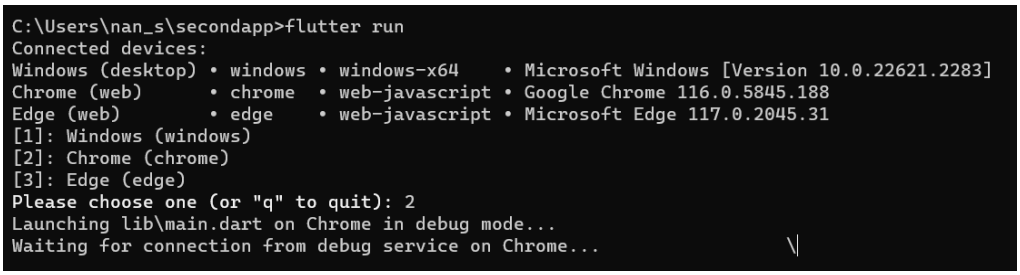
4.3 เมื่อสั่งการทำงานครั้งแรกหลังจากการติดตั้ง Flutter จะแจ้งให้ผู้ใช้งานเลือกเว็บบราวน์เซอร์ตั้งต้นสำหรับการแสดงหน้าเว็บ



ภาพประกอบ 2.19 สั่งการทำงานด้วย flutter run (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 7)

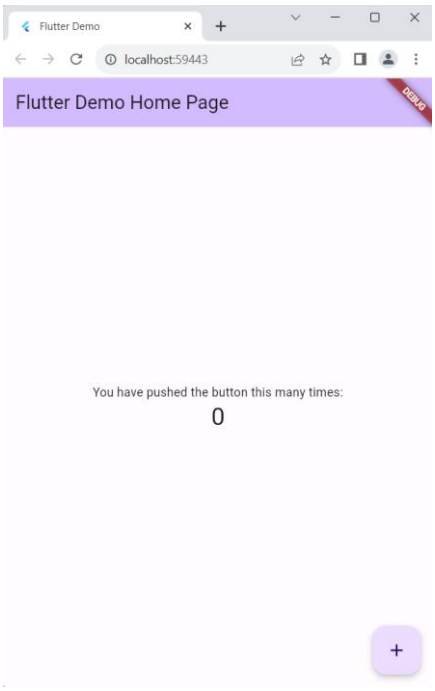
5. เลือก Devices สำหรับการแสดงหน้าเว็บแอปฯ โดยจะมีให้เลือกอยู่ 3 ประเภท คือ 1) Windows 2) Google Chrome 3) Microsoft Edge

ให้ทำการกดหมายเลข Device ที่ต้องการ จากนั้นให้กดปุ่ม Enter



ภาพประกอบ 2.20 เลือก Devices
ที่มา : ณปภัช วรรณตรง (2564 : 8)

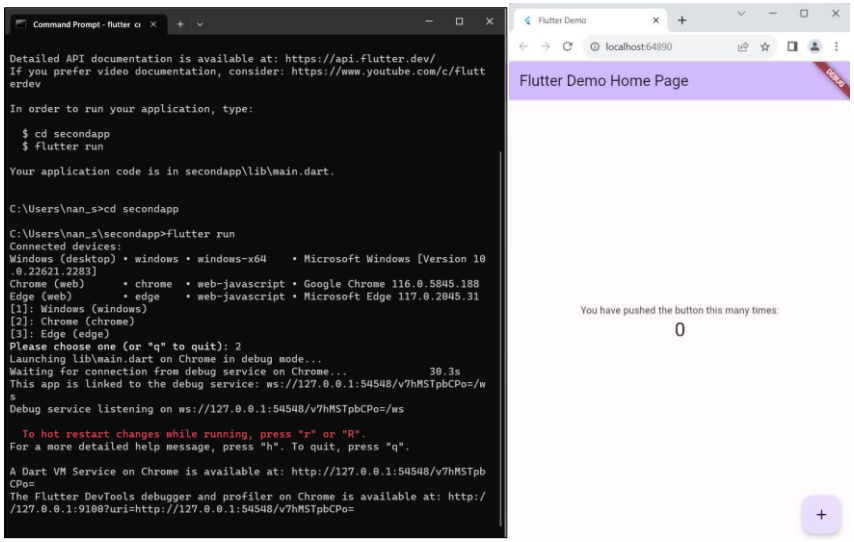
เมื่อคำสั่งทำงานเสร็จสิ้นจะปรากฏหน้าต่าง Device หรือเว็บบราวน์เซอร์ที่ได้ทำการเลือก



ภาพประกอบ 2.21 เลือก Devices (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 8)

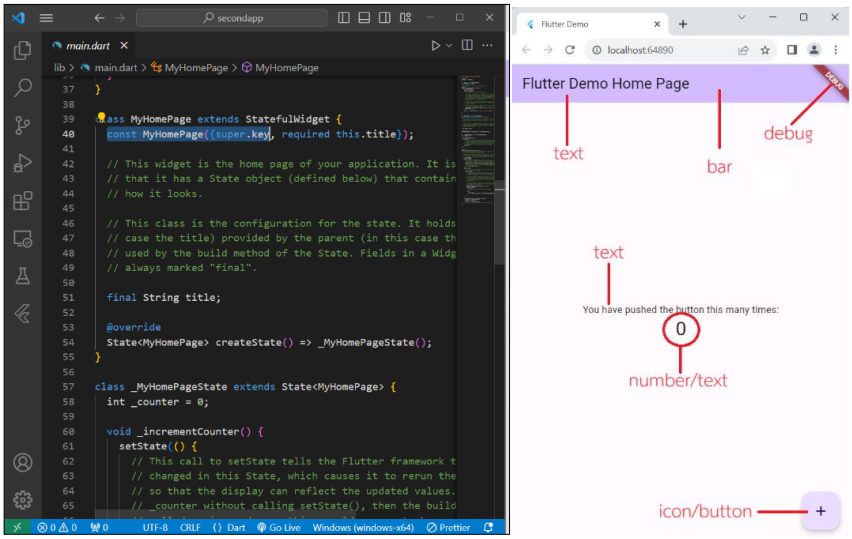
6. หลังจากเลือก Devices เสร็จสิ้น จะได้ผลลัพธ์ดังภาพ



ภาพประกอบ 2.22 ผลลัพธ์การสร้างไฟล์ Flutter Project

ที่มา : ณปภัช วรรณตรง (2564 : 9)

7. ให้ทำการเปิดโค้ดผ่าน VSCode เพื่อดูแต่ละส่วนประกอบบนหน้าแอปพลิเคชัน

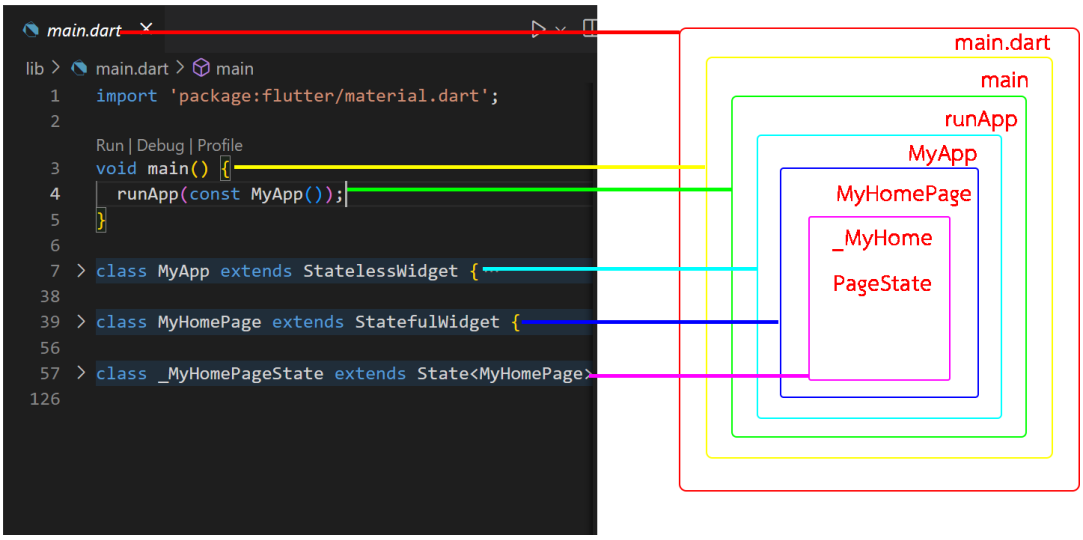


ภาพประกอบ 2.23 ดูแต่ละส่วนประกอบบนหน้าแอปพลิเคชัน

ที่มา : ณปภัช วรรณตรง (2564 : 10)

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

8. จากนั้นให้ลองทำความเข้าใจองค์ประกอบต่าง ๆ ของโครงสร้างแอปพลิเคชันที่พัฒนาโดยการใช้ Flutter



ภาพประกอบ 2.24 ทำความเข้าใจองค์ประกอบต่าง ๆ

ที่มา : ณปภัช วรรณตรง (2564 : 11)

จากภาพ ภายในไฟล์ main dart มีฟังก์ชัน main() เป็นฟังก์ชันหลักที่จะถูกเรียก ใช้งานเป็นลำดับแรก ภายในฟังก์ชัน main จะมีคำสั่งเพื่อนำเอาวิดเจ็ตมาแสดงบนแอปพลิเคชัน

runApp เป็นฟังก์ชันที่ใช้สำหรับสร้างวิดเจ็ตหลัก โดยจะเรียกเมธอด build() ที่อยู่ภายในวิดเจ็ตหลักนั้น เพื่อนำผลลัพธ์มาแสดงบนหน้าจอต่อไป (จีราวุธ วารินทร์, 2564 : 144-145)

9. หลังจากที่ได้ทำความเข้าใจโครงสร้างของแอปพลิเคชันแล้ว ให้ลองทำการแก้ไขโค้ดในส่วนของ 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          //
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
30          // tested with just a hot reload.
31          colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
32          useMaterial3: true,
33        ), // ThemeData
34        home: const MyHomePage(title: 'Flutter Demo Home Page'),
35      ); // MaterialApp
36    }
37  }
```

ภาพประกอบ 2.25 ลบโค้ดที่ไม่ได้ใช้ออก

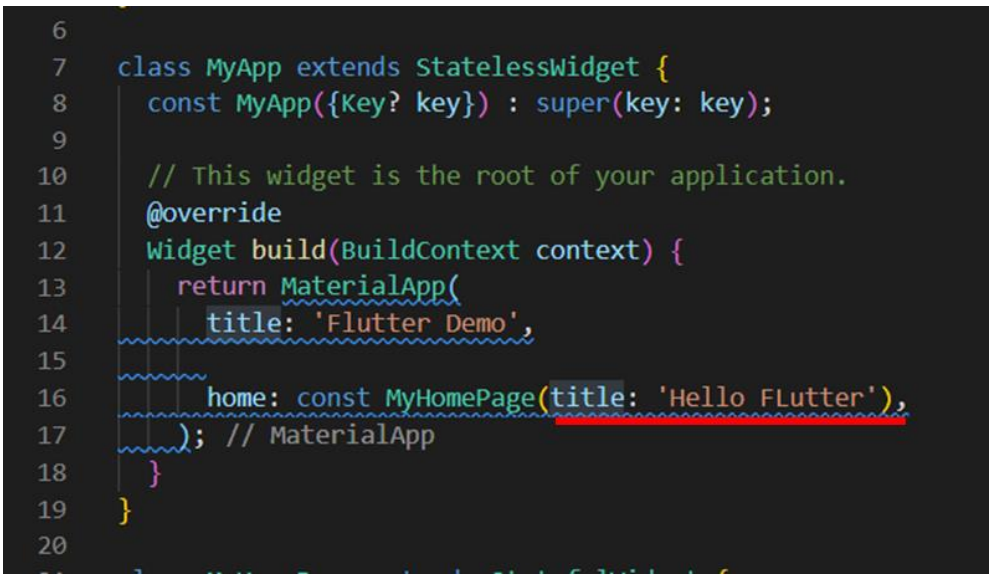
ที่มา : ณปภัช วรรณตรง (2564 : 15)

9.2 เมื่อทำการลบโค้ดที่ไม่ได้ใช้ออกแล้ว ให้ทำการแก้ไขโค้ด title: “ ” เป็นข้อความที่ต้องการ

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

ภาพประกอบ 2.26 แก้ไขโค้ดส่วน Title

ที่มา : ณปภัช วรรณตรง (2564 : 15)

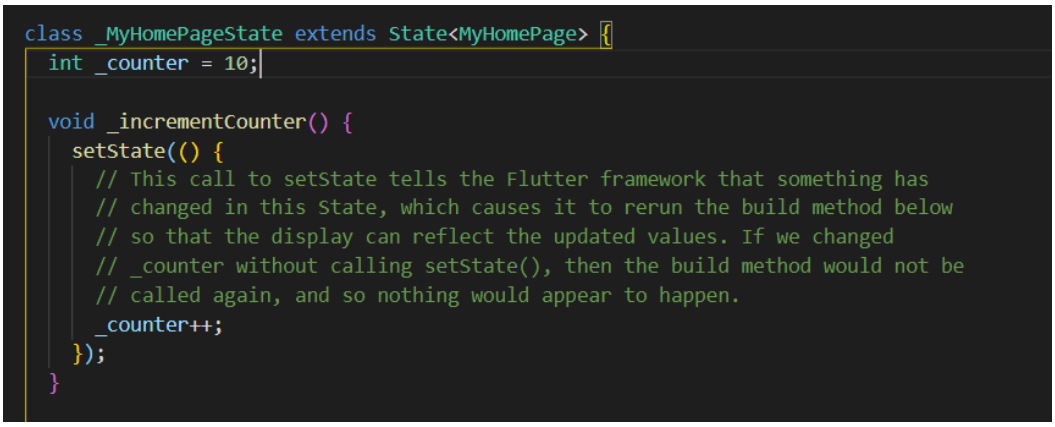


```
6
7   class MyApp extends StatelessWidget {
8     const MyApp({Key? key}) : super(key: key);
9
10    // This widget is the root of your application.
11    @override
12    Widget build(BuildContext context) {
13      return MaterialApp(
14        title: 'Flutter Demo',
15
16        home: const MyHomePage(title: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

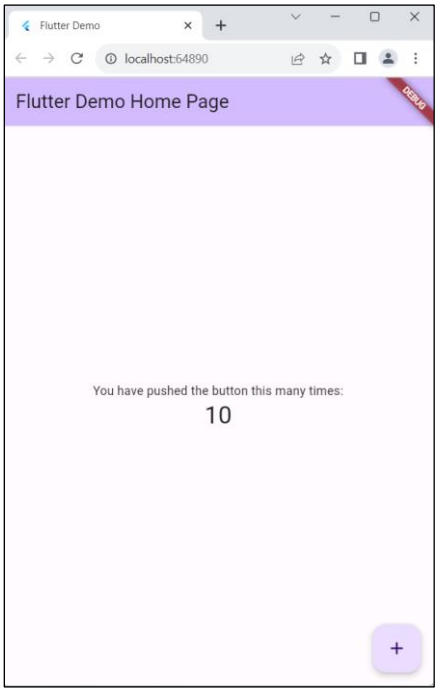
เมื่อทำการแก้ไขโค้ดเบื้องต้นแล้วให้ลองทำการ Restart Flutter โดยการกดปุ่ม r หรือ Ctrl+r จากนั้นให้กด y (ตกลง) แล้วกดปุ่ม Enter

```
  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://127.0.0.1:55317/8ZTOJr1GauI=
The Flutter DevTools debugger and profiler on Chrome is available at:
http://127.0.0.1:9103?uri=http://127.0.0.1:55317/8ZTOJr1GauI=
r

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

ภาพประกอบ 2.29 Restart Flutter

ที่มา : ณปภัช วรรณตรง (2564 : 17)

จากนั้นให้ทำการสั่ง flutter run ใหม่อีกครั้ง จะพบผลลัพธ์ตัวนับมีค่าเริ่มต้นที่ 10 ตามที่ได้แก้ไขดังภาพ



ภาพประกอบ 2.30 ผลลัพท์การแก้ไขโค้ดเบื้องต้น

ที่มา : ณปภัช วรรณตรง (2564 : 18)

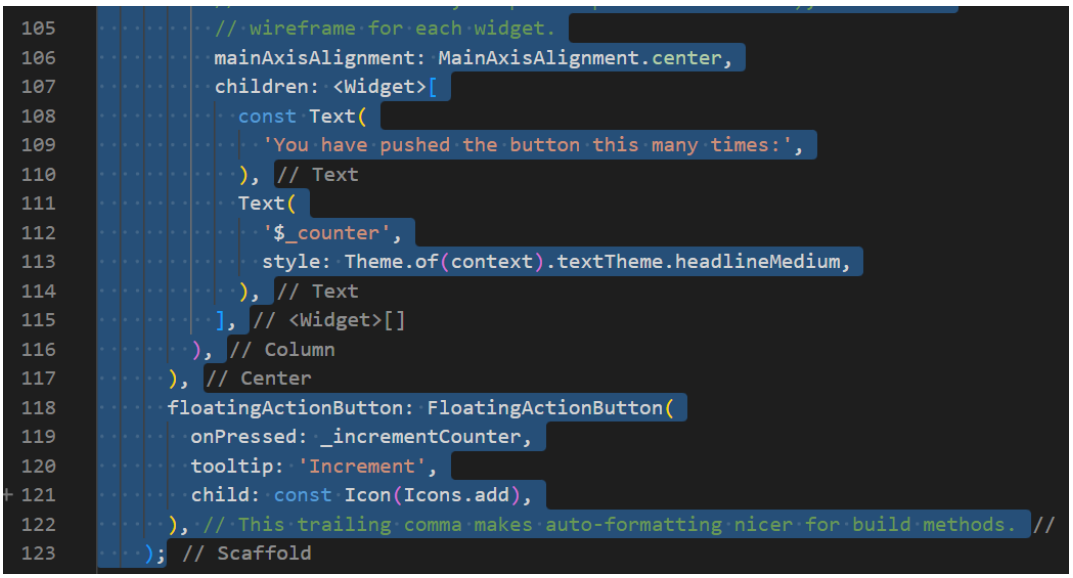
11. ทำการเอาส่วน Scaffold ออกจากแอปพลิเคชัน

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

ภาพประกอบ 2.31 ครอบเลือกส่วน Scaffold(..)

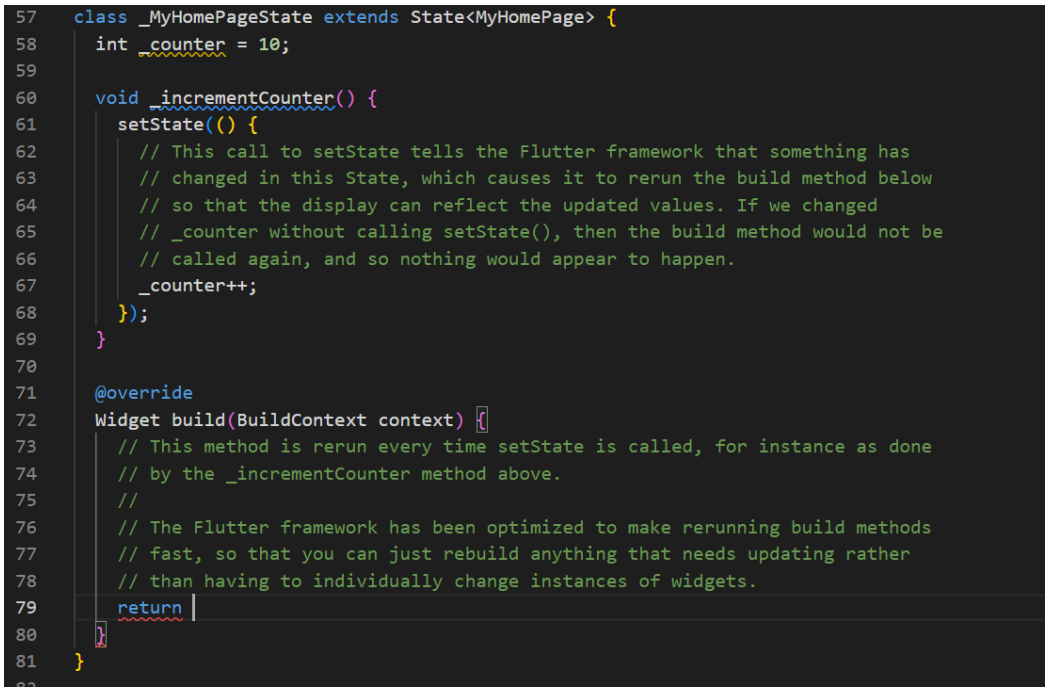
ที่มา : ณปภัช วรรณตรง (2564 : 20)



```
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 : ณปภัช วรรณตรง (2564 : 20)

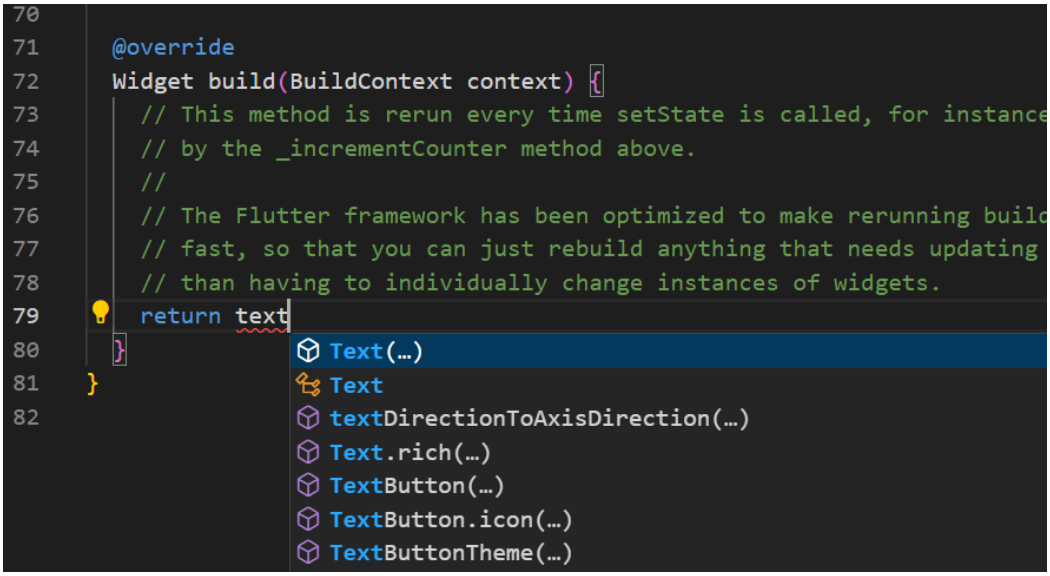


```
57  class _MyHomePageState extends State<MyHomePage> {
58    int _counter = 10;
59
60    void _incrementCounter() {
61      setState(() {
62        // This call to setState tells the Flutter framework that something has
63        // changed in this State, which causes it to rerun the build method 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 : ณปภัช วรรณตรง (2564 : 20)

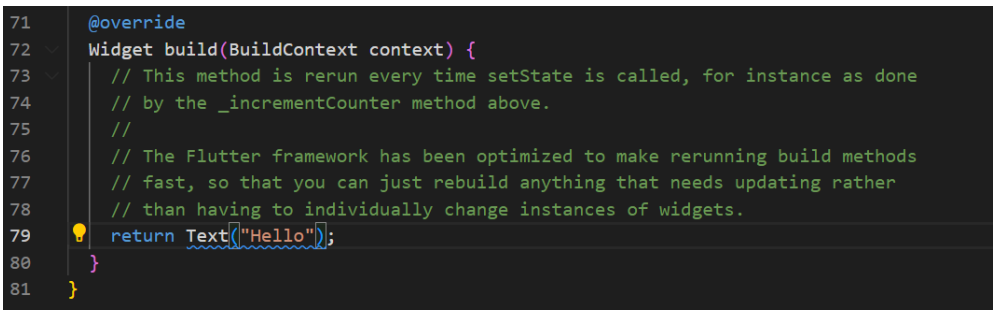
11.2 หลังที่ลบ Scaffold ออกแล้ว ให้ทำการเพิ่มโค้ด Text(...) เข้าไปแทนที่ Scaffold(...) โดยการพิมพ์ Te จากนั้นจะปรากฏหน้าต่างชุดคำสั่งขึ้นมาให้ทำการเลือก Text(...) ด้วยการคลิกที่คำสั่งหรือ Enter



ภาพประกอบ 2.34 เพิ่มโค้ด Text(...) เข้าไป

ที่มา : ณปภัช วรรณตรง (2564 : 20)

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



ภาพประกอบ 2.35 เพิ่มข้อความเข้าไปใน Text(...)

ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : 20)

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

จัด Layout หน้าแอปพลิเคชันด้วย Scaffold Widget

Scaffold คือ วิดเจ็ตหน้าต่างสำเร็จรูปสำหรับจัดการ Layout หรือโครงสร้างของหน้าแอปพลิเคชัน (มีการคำนวณระยะห่างของแอปพลิเคชันกับหน้าจอ Emulator ให้อัตโนมัติ) เป็นวิดเจ็ตสำหรับขึ้นโครงหลักใน Material Design ซึ่งวิดเจ็ตนี้ทำหน้าที่เป็นกำหนดโครงสร้างและธีม ดังนั้นแต่ละหน้าจอจะควรขึ้นต้นด้วย Scaffold หากไม่ได้ใช้ Scaffold จะพบว่าหน้าจอจะไม่มีการเรียกใช้ธีมและหน้าจอจะแสดงผลเป็นสีดำ (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 : ณปภัช วรรณตรง (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 : ณปภัช 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 : ณปภัช 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

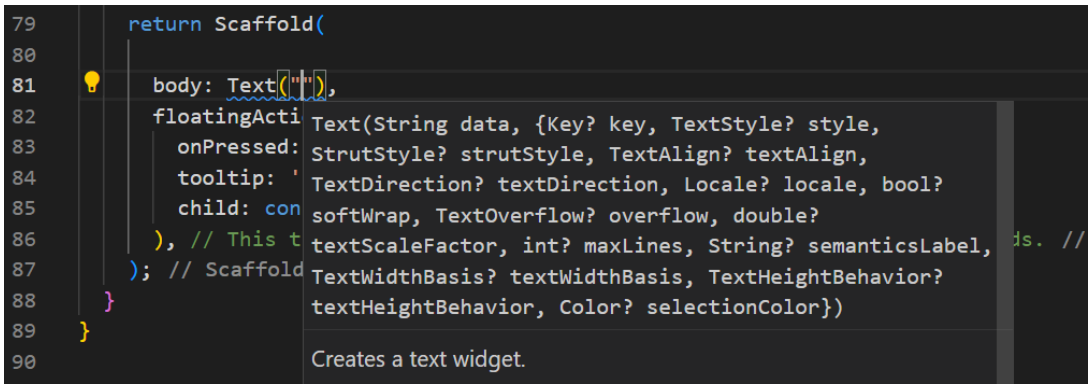
ที่มา : ณปภัช วรรณตรง (2564 : 24)

```
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)

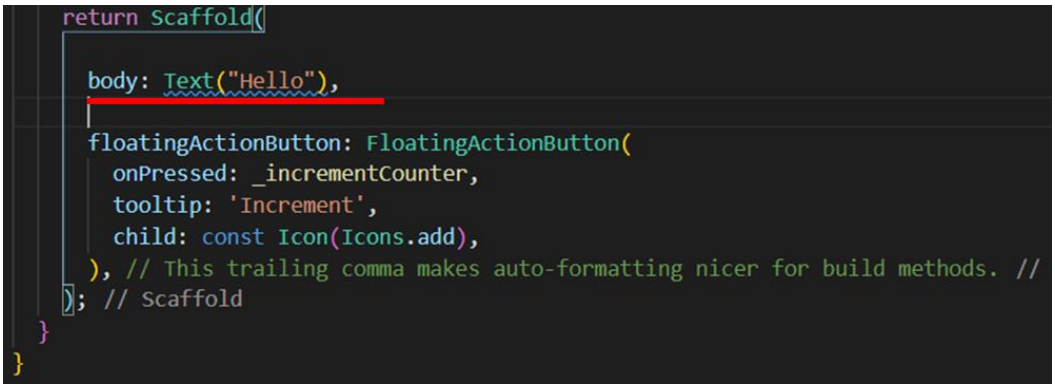
ที่มา : ณปภัช วรรณตรง (2564 : 24)

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



ภาพประกอบ 2.42 เพิ่มโค้ด Text(...)

ที่มา : ณปภัช วรรณตรง (2564 : 24)



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

ที่มา : ณปภัช วรรณตรง (2564 : 24)

1.3 ทำการแก้ไขโค้ด floatingActionButton เพื่อทดลองจัดตำแหน่ง floatingActionButton โดยการนำส่วนของโค้ดที่ต้องการแก้ไขไปค้นหาหรือหาคำตอบผ่าน StackOverflow

เมื่อได้ชุดคำสั่งที่ต้องการแล้วให้ทำการคัดลอกมาใส่ต่อจาก floatingActionButton: FloatingActionButton(...),

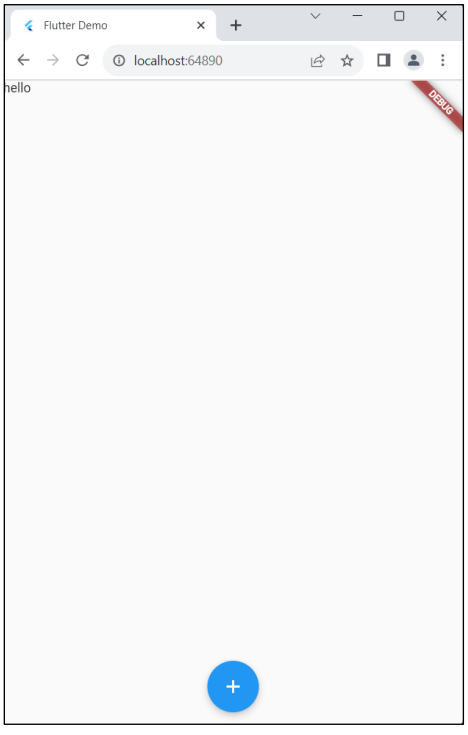
```
floatingActionButton: FloatingActionButton(
  onPressed: _incrementCounter,
  tooltip: 'Increment',
  child: const Icon(Icons.add),
),   // FloatingActionButton
floatingActionButtonLocation: FloatingActionButtonLocation.centerFloat//
```

ภาพประกอบ 2.44 แก้ไขโค้ด floatingActionButton

ที่มา : ณปภัช 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

ผลลัพท์ที่ได้จากการปรับปรุงโค้ด จะเห็นได้ว่าตำแหน่ง floatingActionButton เปลี่ยนไปตามโค้ดที่ได้แก้ไข

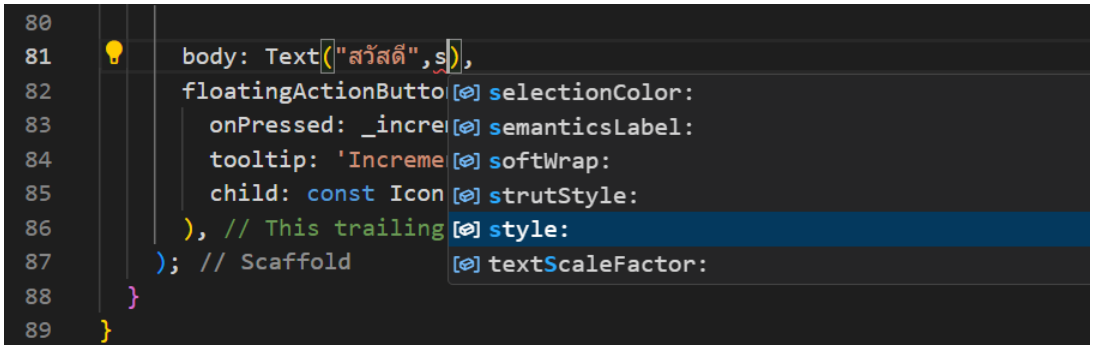


ภาพประกอบ 2.45 ผลลัพท์จากการแก้ไขโค้ด

ที่มา : ณปภัช วรรณตรง (2564 : 27)

1.4 ทำการกำหนด Style ให้กับ Text

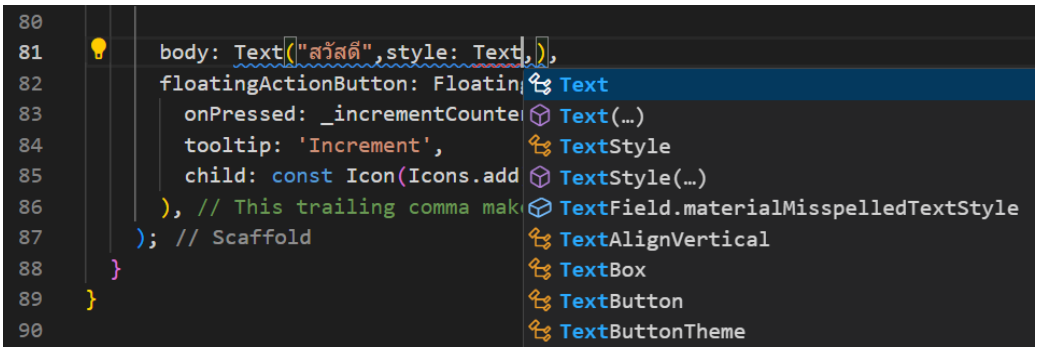
1.4.1 ให้ทำการเพิ่ม style: เข้าไปต่อจาก “ ”, เพื่อทำการกำหนด Style ให้กับ Text ที่ต้องการ โดยการพิมพ์ st จากนั้นจะปรากฏหน้าต่างชุดคำสั่งขึ้นมาให้ทำการเลือก style: ด้วยการคลิกที่คำสั่งหรือ Enter



ภาพประกอบ 2.46 กำหนด Style ให้กับ Text

ที่มา : ณปภัช วรรณตรง (2564 : 34)

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

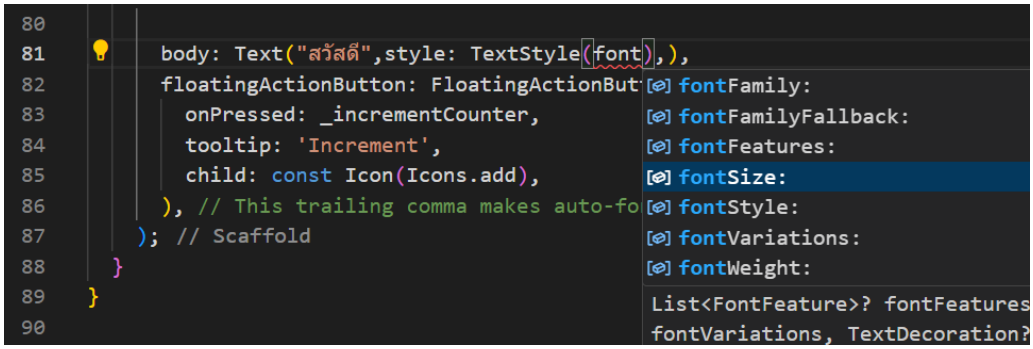


ภาพประกอบ 2.47 กำหนด Style ให้กับ Text (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 34)

1.4.3 ทุกชุดคำสั่งเหมือนเรียกใช้คำสั่งแล้วหากต้องการเพิ่มชุดคำสั่งอื่น ๆ ต้องคั่นด้วยเครื่องหมาย “,” (Comma) ทุกครั้ง

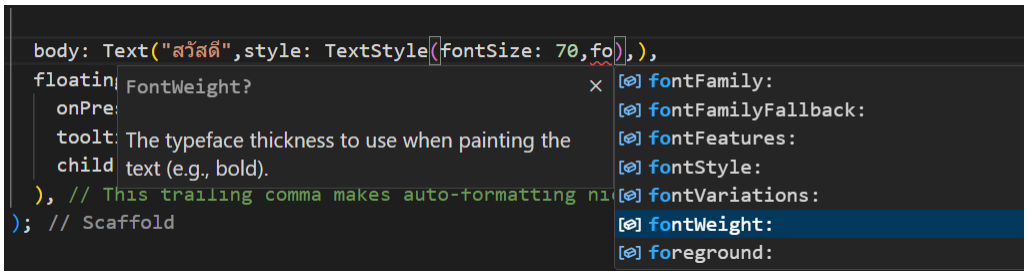
1.4.4 ภายใน TextStyle(...) ให้เพิ่มโค้ด fontSize: [ขนาดของฟอนต์ที่ต้องการ] เข้าไป



ภาพประกอบ 2.48 เพิ่มโค้ด fontSize:

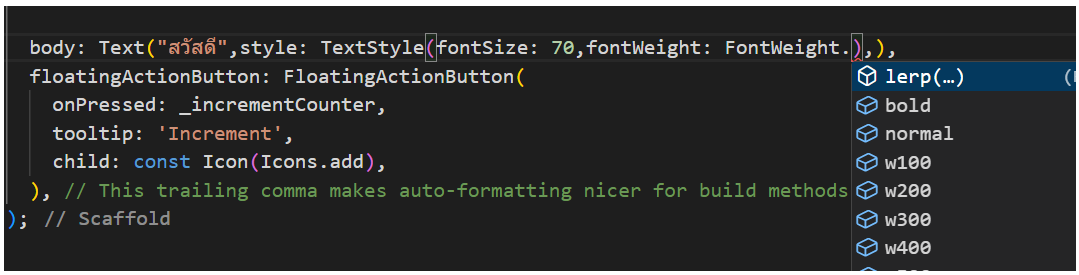
ที่มา : ณปภัช วรรณตรง (2564 : 34)

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



ภาพประกอบ 2.49 เพิ่มโค้ด fontWeight:

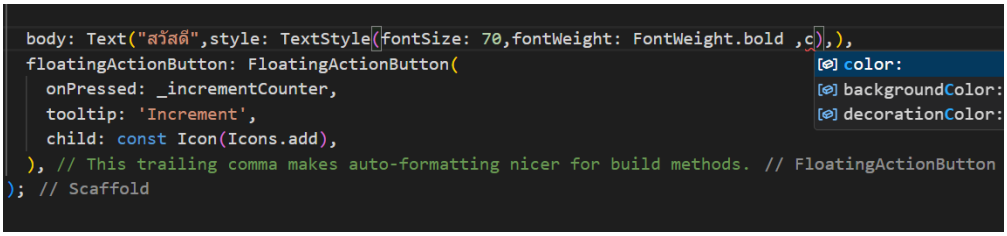
ที่มา : ณปภัช วรรณตรง (2564 : 34)



ภาพประกอบ 2.50 เพิ่มโค้ด fontWeight: (ต่อ)

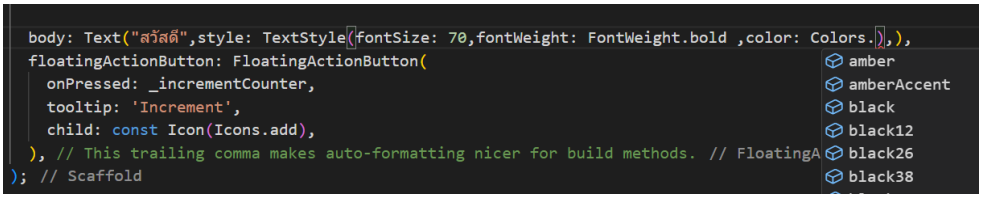
ที่มา : ณปภัช วรรณตรง (2564 : 34)

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



ภาพประกอบ 2.51 เพิ่มโค้ด color:

ที่มา : ณปภัช วรรณตรง (2564 : 34)



ภาพประกอบ 2.52 เพิ่มโค้ด color: (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 34)

1.4.7 ได้ผลลัพธ์โค้ดการกำหนด Style ให้กับ Text ทั้งหมดดังภาพ

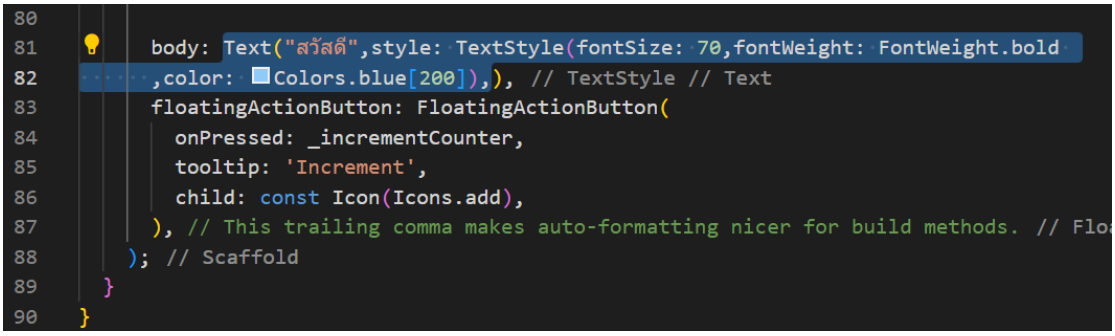
ภาพประกอบ 2.53 กำหนด Style ให้กับ Text

ที่มา : ณปภัช วรรณตรง (2564 : 34)

หมายเหตุ
หากกดปุ่ม ctrl + spacebar จะทำการแสดงคำสั่งให้อัตโนมัติ

1.5 เมื่อกำหนด Style ให้กับ Text เรียบร้อยแล้วให้ทำการกำหนด Text ให้อยู่ตรงกลางโดยใช้ Center

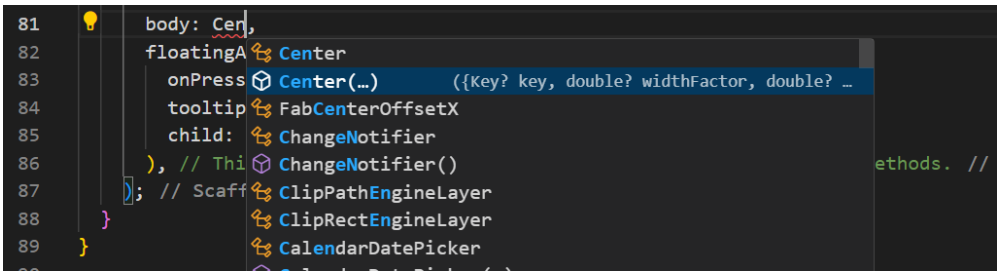
15.1 คัดลอกหรือตัดโค้ด Text(...) ออกมาเพื่อทำการเพิ่มโค้ด Center(...) จากนั้นให้ทำการวางโค้ด Text(...) เข้าไปไว้ใน Center(...) เพื่อกำหนดให้อยู่กึ่งกลาง



ภาพประกอบ 2.54 ตัดส่วน Text(...) ไว้

ที่มา : ณปภัช วรรณตรง (2564 : 34)

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



ภาพประกอบ 2.55 เพิ่มโค้ด Center(...)

ที่มา : ณปภัช วรรณตรง (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 : ณปภัช 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 : ณปภัช วรรณตรง (2564 : 36)

1.5.4 ได้ผลลัพธ์โค้ดการกำหนด Text ให้อยู่ตรงกลางทั้งหมดดังภาพ

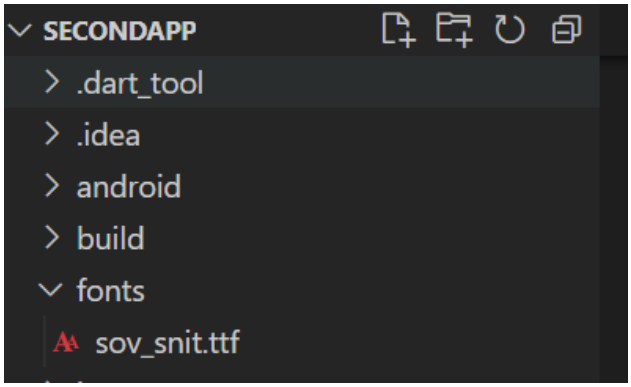
```
body: Center(child: Text("สวัสดี",
style: TextStyle (fontSize: 100, fontWeight: FontWeight.bold,
color: Colors.blue),),,) // TextStyle // Text // Center
```

ภาพประกอบ 2.58 กำหนด Text ให้อยู่ตรงกลาง

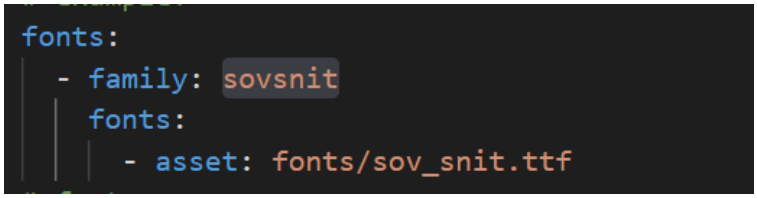
ที่มา : ณปภัช วรรณตรง (2564 : 36)

1.6 สามารถกำหนดรูปแบบ Font ให้กับ Text ได้

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



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

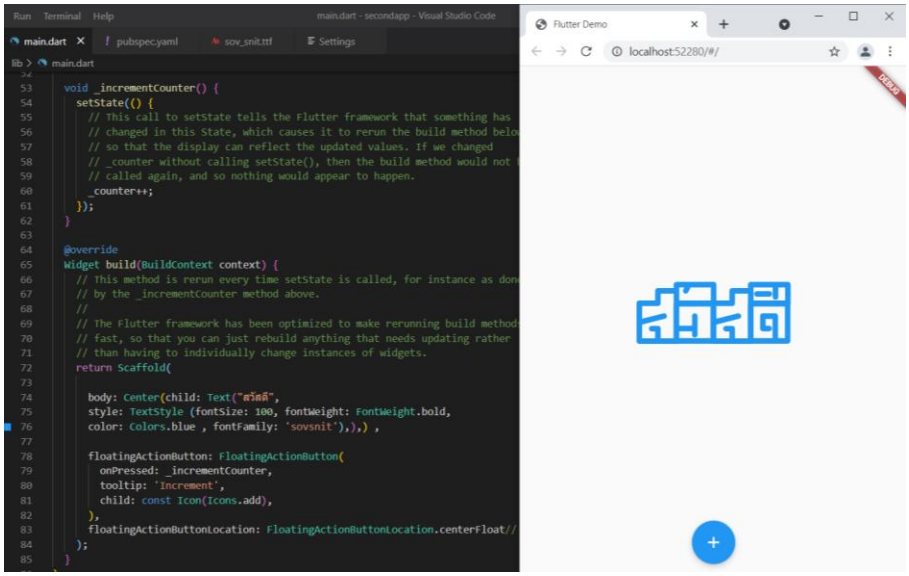


ภาพประกอบ 2.61 แก้ไขไฟล์ที่มีชื่อว่า pubspec.yaml ตามภาพ
ที่มา : ณปภัช วรรณตรง (2564 : 39)

```
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 : ณปภัช วรรณตรง (2564 : 39)

1.7 ผลลัพธ์ที่ได้ในการสร้างแอปพลิเคชันจาก Flutter จะเห็นได้ว่ารูปแบบ Font ของText เปลี่ยนไปตามที่ได้กำหนด



ภาพประกอบ 2.63 ผลลัพธ์การสร้างแอปพลิเคชันจาก Flutter
ที่มา : ณปภัช วรรณตรง (2564 : 40)

บทสรุป

สำหรับเนื้อหาที่กล่าวมาทั้งหมดในบทนี้ จะพูดถึงวิดเจ็ตว่าคืออะไร วิดเจ็ตพื้นฐานที่อยู่ในภาษา Dart ซึ่งเป็นภาษาที่ใช้ในการพัฒนาแอปพลิคชันด้วย Flutter ประเภทของวิดเจ็ตว่ามีประเภทใดบ้าง และแต่ละประเภทมีลักษณะอย่างไร วิธีการสร้างวิดเจ็ตการเริ่มสร้าง Flutter Project และขั้นตอนการสร้างวิดเจ็ตในโปรเจกต์ วิธีการตกแต่งตัวอักษร การดาวน์โหลดรูปแบบตัวอักษรที่ต้องการ ตั้งค่ารูปแบบตัวอักษร กำหนดขนาด สี ของตัวอักษร

เอกสารอ้างอิง

จีราวุธ วารินทร์. (2564). **พัฒนาโมไบล์แอพด้วย Flutter + Dart.** กรุงเทพฯ : สำนักพิมพ์ซิมพลิฟาย.

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

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

เอกรินทร์ วทัญญูเลิศสกุล. (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 รายการข้อมูลที่สามารถจัดเรียงต่อเนื่องกันอยู่ในรูปแบบของอาร์เรย์ 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) และมีส่วนร่วมในการสร้างเว็บบราวน์เซอร์ 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 เดือนต่อมาได้รับ การอนุมัติจาก ECMA International ออกเป็นข้อกำหนดมาตรฐานภาษารุ่นที่ 2 ในเดือนธันวาคม ค.ศ. 2014 ซึ่งวัตถุประสงค์ของการสร้างภาษา Dart ถูกออกแบบมาเพื่อเป็นภาษาเชิงโครงสร้างที่มีความยืดหยุ่นสูง และได้มีการออกแบบโครงสร้างของภาษาไปพร้อมกับ Engine สำหรับการคอมไพล์โปรแกรม โดยมุ่งเน้นการแก้ปัญหาการทำงานช้าของโปรแกรมหรือ
แอปพลิเคชันที่พัฒนาขึ้นมา 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 (บัญชา ปะสีละเตสัง, 2566 : 30-33)

2. ข้อแตกต่างระหว่าง final vs const คือ

final เป็นการกำหนดว่าตัวแปรที่ไม่สามารถเปลี่ยนแปลงค่าได้ ซึ่งเป็นตัวแปรประเภท runtime ดังนั้นสามารถกำหนดค่า final จากตัวแปรหรือฟังก์ชันอื่นได้

const เป็นการประกาศตัวแปรที่ไม่ต้องการเปลี่ยนแปลงค่า หากประกาศไปแล้วจะไม่สามารถแก้ไขค่าตัวแปร นั้นซ้ำได้อีก (จีราวุธ วารินทร์, 2564 : 142)

String methods

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

ตัวอย่างการใช้งาน String methods ดังภาพ

```dart
void main() {
  String message = 'Hello Dart';
  print(message.length);
  print(message.substring(2));
  print(message.substring(6));
  print(message.split(''));
  print(message.split('!'));
  print(message.toLowerCase());
  print(message.toUpperCase());
}
```

ภาพประกอบ 3.1 ตัวอย่างการใช้งาน String methods

ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : ...)

Operator

Operator คือ ตัวดำเนินการทางคณิตศาสตร์ที่ใช้ประเมินผล (Evaluate) หรือกำหนดค่า (Assign) แบ่งออกเป็นกลุ่มตามการใช้งานได้ดังนี้

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

```dart
void main() {
  int a = 5;
  print(a++);
  print(a--);
  print(++a);
  print(--a);
}
```

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

```
5
6
6
5
```

ภาพประกอบ 3.4 ผลลัพธ์การใช้งาน Arithmetic Operators Operator
ที่มา : ณปภัช 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 วทัญญูเลิศสกุล, 2563 : 46) ตัวอย่างการใช้งาน Type test Operators ดังภาพ

```dart
void main() {
var e;
print(e is int);      // false
print(e is! int);     // true
}
```

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

```
false
true
```

ภาพประกอบ 3.6 ผลลัพธ์การใช้งาน Type test Operators
ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : ...)

```
false
true
false
true
false
true
```

ภาพประกอบ 3.8 ผลลัพธ์การใช้งาน Equality and Relational Operators
ที่มา : ณปภัช วรรณตรง (2564 : ...)

Bitwise Operators คือ ตัวดำเนินการแบบบิต เป็นตัวดำเนินการที่ใช้กับชนิดข้อมูลเลขจำนวนเต็ม (integer) เท่านั้น โดยจะแปลงเป็นเลขฐาน 2 (8-bit) ก่อนดำเนินการ หลังจากนั้น จะแปลงค่ากลับเป็นฐาน 10 กลับมา ซึ่งมีตัวเนินการได้แก่ & (AND) เป็นการหาเลข 1 ที่ตำแหน่ง (บิด) 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 คืนค่าเป็นค่าที่กำหนดให้

?? = กำหนดค่าให้ตัวแปร ถ้าตัวแปรนั้นเป็น null ถ้าตัวแปรนั้นไม่ใช่ null ค่าของตัวแปรนั้นจะคงเดิม

+ = (Add and Assignment) คือ การบวกค่าตัวแปรนั้นด้วยค่าที่กำหนด คืนค่าเป็นตัวแปรที่ผ่านการบวกแล้ว

- = (Subtract and Assignment) คือ การบวกค่าตัวแปรนั้นด้วยค่าที่กำหนด คืนค่าเป็นตัวแปรที่ผ่านการลบแล้ว

* = (Multiply and Assignment) คือ การบวกค่าตัวแปรนั้นด้วยค่าที่กำหนด คืนค่าเป็นตัวแปรที่ผ่านการคูณแล้ว

/ = (Divide and Assignment) คือ การบวกค่าตัวแปรนั้นด้วยค่าที่กำหนด คืนค่าเป็นตัวแปรที่ผ่านการหารแล้ว

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

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

ตัวอย่างการใช้งาน Assignment Operators ดังภาพ

```dart
void main() {
  int a = 5;
  int b =10;
  print(a + b);
  print(a - b);
  print(a * b);
  print(a / b);
  print(a ~/ b);
  print(a % b);
  print(a ?? b);
  print(a = b);
}
```

ภาพประกอบ 3.9 ตัวอย่างการใช้งาน Assignment Operators

ที่มา : ณปภัช วรรณตรง (2564 : ...)

```
15
-5
50
0.5
0
5
5
10
```

ภาพประกอบ 3.10 ผลลัพธ์การใช้งาน Assignment Operators
ที่มา : ณปภัช วรรณตรง (2564 : ...)

Logical Operators คือตัวดำเนินการทางตรรกศาสตร์ เป็นองค์ประกอบที่ใช้ในการประมวลผลและจัดการกับค่าที่เป็นข้อมูลประเภทบูลีน (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 : ณปภัช วรรณตรง (2564 : ...)

```
true
false
true
```

ภาพประกอบ 3.12 ผลลัพธ์การใช้งาน Logical Operators

ที่มา : ณปภัช วรรณตรง (2564 : ...)

If/Else, Loop, Switch Case

กลุ่มคำสั่งที่ใช้ในการควบคุมการทำงานของโปรแกรม โดยในภาษา Dart จะประกอบไปด้วย 3 รูปแบบ

- แบบลำดับ (Sequence) คือ การทำงานจากบนลงล่าง ซ้ายไปขวา
- แบบมีเงื่อนไข (Condition) คือ กลุ่มคำสั่งที่ใช้ตัดสินใจในการเลือกเงื่อนไขต่าง ๆ ภายในโปรแกรมมาทำงาน ซึ่งจะประกอบไปด้วย 2 คำสั่ง คือ if else และ switch case

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

```dart
void main(){
  int x = 10,y=15;
  if(x>y){
    print("$x มากกว่า $y");//ถ้าเงื่อนไขจริง
  }
  print("จบการทำงาน");
}
```

ภาพประกอบ 3.13 ตัวอย่างการใช้งาน If แบบเงื่อนไขเดียว

ที่มา : ณปภัช วรรณตรง (2564 : ...)

```
10 มากกว่า 8
จบการทำงาน
```

ภาพประกอบ 3.14 ผลลัพธ์การใช้งาน If แบบเงื่อนไขเดียว

ที่มา : ณปภัช วรรณตรง (2564 : ...)

```
void main(){
  int x = 10;
  if(x % 2 == 0){
    print("$x เป็นเลขคู่");//ถ้าเงื่อนไขจริง
  }else{
  print("เป็นเลขขี้");
}
}
```

ภาพประกอบ 3.15 ตัวอย่างการใช้งาน if แบบ 2 เงื่อนไขเดียว

ที่มา : ณปภัช วรรณตรง (2564 : ...)

```
10 เป็นเลขคู่
```

ภาพประกอบ 3.16 ผลลัพธ์การใช้งาน if แบบ 2 เงื่อนไขเดียว

ที่มา : ณปภัช วรรณตรง (2564 : ...)

```
void main(){
  int x = 10, y = 15;
  if(x > y){
    print("$x มากกว่า $y");
  }else if(x<y){
  print("$x น้อยกว่า $y");
}else{
    print("$x เท่ากับ $y");
  }
}
```

ภาพประกอบ 3.17 ตัวอย่างการใช้งาน If แบบหลายเงื่อนไข

ที่มา : ณปภัช วรรณตรง (2564 : ...)

```
10 น้อยกว่า 15
```

ภาพประกอบ 3.18 ผลลัพธ์การใช้งาน If แบบหลายเงื่อนไข
ที่มา : ณปภัช วรรณตรง (2564 : ...)

1. **Switch..Case**

Switch เป็นคำสั่งที่ใช้กำหนดเงื่อนไขคล้าย ๆ กับ If แต่จะเลือกเพียงหนึ่งทางเลือกออกมาทำงานโดยนำค่าในตัวแปรมากำหนดทางเลือกผ่านคำสั่ง Case ตัวอย่างเช่น เจอ Case ที่ 1 จะให้ทำอะไร Case ทำอะไร (จีราวุธ 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 : ณปภัช วรรณตรง (2564 : ...)

```
ยอดเยี่ยม
```

ภาพประกอบ 3.20 ผลลัพธ์การใช้งาน Switch..Case
ที่มา : ณปภัช วรรณตรง (2564 : ...)

2. Loop

กลุ่มคำสั่งที่ใช้ในการวนวนลูป โดยโปรแกรมจะทำงานไปเรื่อย ๆ จนกว่าเงื่อนไขที่กำหนดนั้นจะเป็นเท็จ โปรแกรมจึงจะหยุดทำงาน โดยในที่นี้จะมีอยู่ 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

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

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 : ณปภัช วรรณตรง (2564 : ...)

```
1
2
3
4
5
```

ภาพประกอบ 3.24 ผลลัพธ์การใช้งาน For

ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : ...)

```
hello dart
```

ภาพประกอบ 3.26 ผลลัพธ์การใช้งาน Do While

ที่มา : ณปภัช วรรณตรง (2564 : ...)

Function

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

ตัวอย่างการใช้งาน Function ดังภาพ

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

ภาพประกอบ 3.27 ตัวอย่างการใช้งาน Function

ที่มา : ณปภัช วรรณตรง (2564 : ...)

Arrow Function คือ รูปย่อชนิดหนึ่ง ของ Function ที่เขียนให้สั้นลงและง่ายขึ้น

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

ภาพประกอบ 3.28 ตัวอย่างการใช้งาน Arrow Function
ที่มา : ณปภัช วรรณตรง (2564 : ...)

ประเภทของ Function

Function : No return & No Args

เป็น Function ที่ไม่ต้องรับค่าเข้ามาเพื่อประมวลผล ไม่คืนค่ากลับหลังจากทำงานเสร็จ (void + No return) (จีราวุธ วารินทร์, 2564 : 79-81)

ตัวอย่างการใช้งาน Function : No return & No Args ดังภาพ

```dart
void main() {
test1();
}
void test1() {
print("No Return & No Args Function");
}
```

ภาพประกอบ 3.29 ตัวอย่างการใช้งาน Function : No return & No Args
ที่มา : ณปภัช วรรณตรง (2564 : ...)

```
No Return & No Args Function
```

ภาพประกอบ 3.30 ผลลัพธ์การใช้งาน Function : No return & No Args
ที่มา : ณปภัช วรรณตรง (2564 : ...)

Function : No return & Args

เป็น Function ที่ต้องรับค่าเข้ามาเพื่อประมวลผล ไม่คืนค่ากลับหลังจากทำงานเสร็จ (void + No return)

ตัวอย่างการใช้งาน Function : No return & Args ดังภาพ

```dart
void main() {
  test2("Hello", "Dart");
}

void test2(String name, String surname) {
  print("name is $name $surname");
}
```

ภาพประกอบ 3.31 ตัวอย่างการใช้งาน Function : No return & Args
ที่มา : ณปภัช วรรณตรง (2564 : ...)

```
name is Hello Dart
```

ภาพประกอบ 3.32 ผลลัพธ์การใช้งาน Function : No return & Args
ที่มา : ณปภัช วรรณตรง (2564 : ...)

Function : return & No Args

เป็น Function ที่ไม่ต้องรับค่าเข้ามาเพื่อประมวลผล แต่มีการคืนค่ากลับหลังจากทำงานเสร็จ (type + return)

ตัวอย่างการใช้งาน Function : return & No Args ดังภาพ

```dart
void main() {
  print(test3());
}

String test3() {
  String a = 'Hello Dart';
  return a;
}
```

ภาพประกอบ 3.33 ตัวอย่างการใช้งาน Function : return & No Args
ที่มา : ณปภัช วรรณตรง (2564 : ...)

```
Hello Dart
```

ภาพประกอบ 3.34 ผลลัพธ์การใช้งาน Function : return & No Args
ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : ...)

```
55
```

ภาพประกอบ 3.36 ผลลัพธ์การใช้งาน Function : return & Args
ที่มา : ณปภัช วรรณตรง (2564 : ...)

Optional Positional Argument Optional
คือ optional parameter ที่มีลักษณะดังนี้
- Optional Positional Argument ใช้ [ ] ครอบ Argument ที่ต้องการให้ เป็น Optional
- Optional Positional Argument การส่งค่าจะอิงกับตำแหน่งของ Argument ที่กำหนดไว้ ถ้าส่งค่าไม่ครบหรือไม่ตรงตำแหน่ง ค่าที่ออกมาจะมีโอกาสเพี้ยนได้ (ลุงวิศวกร สอนคำนวณ, 2564 : 66-75)

ตัวอย่างการใช้งาน Optional Positional Argument Optional ดังภาพ

```dart
void main() {
  test5("Hello Dart", "M");
  test5("Hello Dart", 23, "M");
}

void test5(name, [age, gender]) {
  print("name is $name age is $age gender is $gender");
}
```

ภาพประกอบ 3.37 ตัวอย่างการใช้งาน Optional Positional Argument Optional
ที่มา : ณปภัช 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 : ณปภัช 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

```dart
void main() {
  test6("Hello dart", gender: "M");
  test6("Hello Dart", age: 22, gender: "M");
}

void test6(name, {age, gender}) {
  print("name is $name age is $age gender is $gender");
}
```

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 วทัญญูเลิศสกุล, 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 : ณปภัช 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 : ณปภัช 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 : ณปภัช วรรณตรง (2564 : ...)

```
growable: []
```

ภาพประกอบ 3.44 ผลลัพธ์การใช้งาน Growable (Link List)

ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : ...)

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

ภาพประกอบ 3.46 ผลลัพธ์การเพิ่มรายการในลิสต์

ที่มา : ณปภัช 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 รูปแบบการใช้งาน คือ 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 วทัญญูเลิศสกุล, 2563 : 131)

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

List.insert คือ การแทรกข้อมูลภายยใน 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 : ณปภัช วรรณตรง (2564 : ...)

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

ภาพประกอบ 3.48 ผลลัพธ์ข้อมูลภายใน List

ที่มา : ณปภัช วรรณตรง (2564 : ...)

List Sort คือ การจัดเรียงข้อมูลตัวเลขและตัวอักษรตำลำดับของข้อมูล

```
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 : ณปภัช วรรณตรง (2564 : ...)

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

ภาพประกอบ 3.50 ผลลัพธ์การจัดเรียงข้อมูลภายใน List

ที่มา : ณปภัช วรรณตรง (2564 : ...)

การ Loop ข้อมูลออกมาใช้ร่วมกับ Inline Function มี 2 รูปแบบ

1. List for Each

```dart
void main() {
  var sort = ['B', 'C', 'A', 'D'];

  sort.sort();
  print("sort is: $sort");
  sort.forEach((String value) {
    print("value is $value");
  });
}
```

ภาพประกอบ 3.51 ตัวอย่างการใช้งาน List for Each

ที่มา : ณปภัช วรรณตรง (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 : ณปภัช วรรณตรง (2564 : ...)

2. List for Loop

```dart
void main() {
  var sort = ['B', 'C', 'A', 'D'];

  sort.sort();
  print("sort is: $sort");
  for (String v in sort) {
    print("v is $v");
  }
}
```

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)

ตัวอย่างการใช้งานการประกาศตัวแปรให้สามารมีค่าเป็น Null ได้ ดังภาพ

```dart
void main() {
  int? a;
  String? b;
  print(a);
  print(b);
}
```

ภาพประกอบ 3.55 ตัวอย่างการใช้งานการประกาศตัวแปรให้สามารมีค่าเป็น Null ได้
ที่มา : ณปภัช วรรณตรง (2564 : ...)

```
null
null
```

ภาพประกอบ 3.56 ผลลัพธ์การใช้งานการประกาศตัวแปรให้สามารมีค่าเป็น Null ได้
ที่มา : ณปภัช 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

```dart
void main() {
  List<String> list1; // ห้ามเป็น null และสมาชิกใน List ห้ามมีค่า null
  List<String?> list2; // ห้ามเป็น null แต่สมาชิกใน List มีค่า null ได้
  List<String>? list3; // null ได้ แต่สมาชิกใน List ห้ามมีค่า null
  List<String?>? list4; // null ได้ และสมาชิกใน List มีค่า null ได้
  list1 = [];
  list2 = [null, 'Hello Dart'];
  list3 = ['A'];
  list4 = ['Hello Dart', null];
  print(list1);
  print(list2);
  print(list3);
  print(list4);
}
```

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

```dart
void main() {

  int? name; // Nullable variable
  if (name != null) {
    print('name is test');
  } else {
    print('name is $name');
  }

}
```

ภาพประกอบ 3.59 ตัวอย่างการใช้งาน Null safety เมื่อใช้งานกับฟังก์ชัน
ที่มา : ณปภัช วรรณตรง (2564 : ...)

```
name is null
```

ภาพประกอบ 3.60 ผลลัพธ์การใช้งาน Null Safety เมื่อใช้งานกับฟังก์ชัน
ที่มา : ณปภัช 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

จีราวุธ วารินทร์. (2564). **พัฒนาโมไบล์แอพด้วย Flutter + Dart.** กรุงเทพฯ : สำนักพิมพ์ซิมพลิฟาย

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

ลุงวิศวกร สอนคำนวณ. (2564). **Basic Dart Programming by Uncle Engineer**. Flutter 2 Bootcamp 2021 - Uncle Engineer.

อนุชิต ชโลธร. (2566). สูตรลัด Dart 3. กรุงเทพฯ : สำนักพิมพ์ก๊อปวาง.

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

บทที่ 4
การสร้างแอปพลิเคชันคำนวณ

หลังจากผู้เรียนได้เรียนรู้การสร้างโปรเจกต์ Flutter เบื้องต้น Widget เบื้องต้น เรียนรู้ภาษา Dart ที่ใช้พัฒนาพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่ด้วย Flutter ในบทนี้ผู้เรียนจะได้เรียนรู้การเขียนคำสั่ง เพื่อลดขนาดและลบไฟล์ที่ไม่จำเป็นก่อนส่งไฟล์ให้ผู้อื่น การกำหนด Layout แบบคอลัมภ์ในแอปพลิเคชัน 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() ดังรายละเอียดต่อไปนี้

Layout Widget

Layout Widget เป็นวิดเจ็ตซึ่งทำหน้าที่จัดตำแหน่งวิดเจ็ตอื่น ๆ บนหน้าจอ สามารถแบ่งได้เป็น 2 ประเภท ดังนี้

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

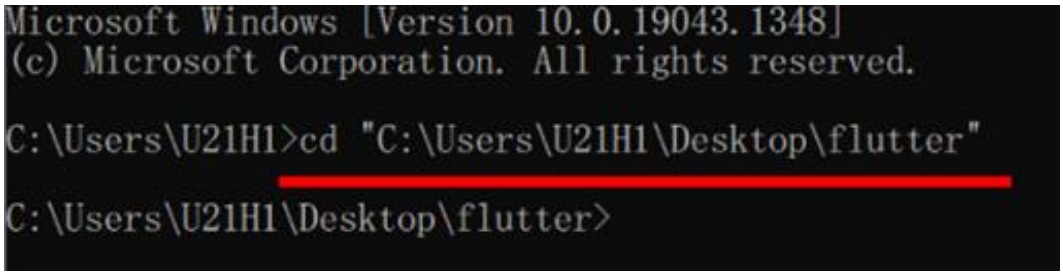
ประเภทที่ 2 Multi-child layout widgets คือ วิดเจ็ตที่ทำหน้าจัดตำแหน่งวิดเจ็ตอื่น ๆ โดยภายในวิดเจ็ตจะประกอบไปด้วยหลาย ๆ วิดเจ็ตภายใต้ เช่น Row, Coloumn และ Stack เป็นต้น (จีราวุธ วารินทร์, 2564 : 206)

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



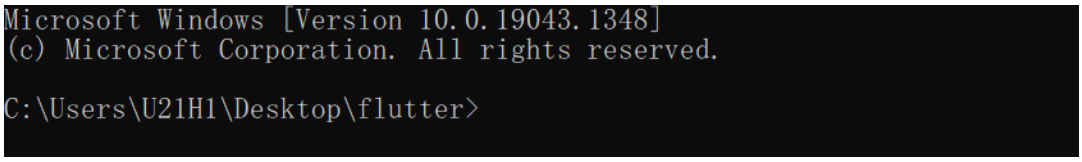
ภาพประกอบ 4.1 เปิดหน้าต่าง cmd และใช้คำสั่ง cd

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)



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

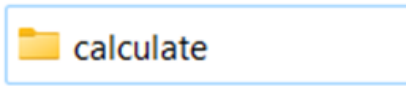
ภาพประกอบ 4.4 พิมพ์คำสั่งสำหรับสร้างโปรเจกต์ Calculate

ที่มา : ณปภัช วรรณตรง (2564 : 4)

1.3 เมื่อทำการสร้างไฟล์งานเสร็จแล้วจะมี Folder ปรากฏขึ้น

ภาพประกอบ 4.5 เมื่อสร้างไฟล์ Project Calculate เสร็จสิ้น

ที่มา : ณปภัช วรรณตรง (2564 : 4)



ภาพประกอบ 4.6 เมื่อสร้างไฟล์ Project Calculate เสร็จสิ้น (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 4)

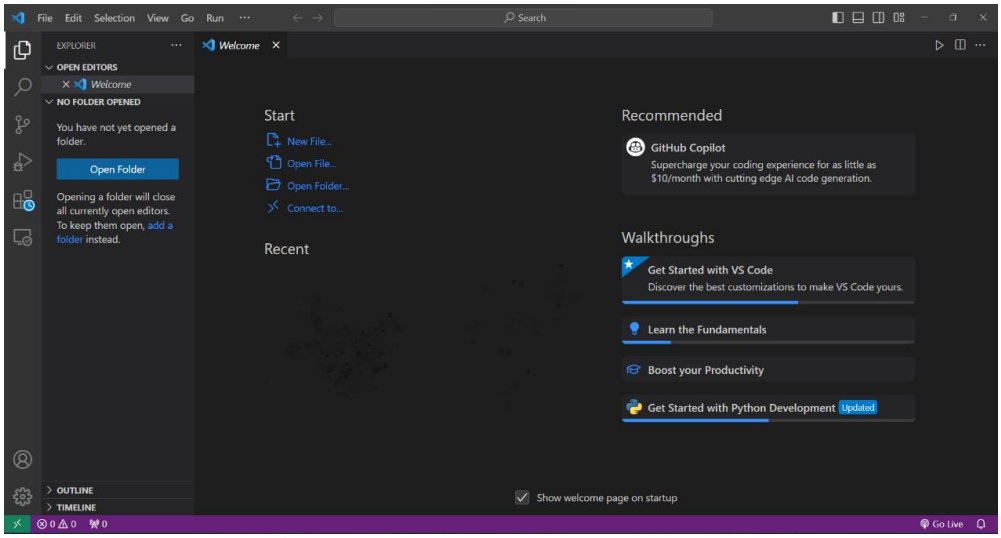
1.4 ซึ่งใน Folder จะมีไฟล์ที่จำเป็นสำหรับการใช้งานไว้ให้ทั้งหมด

ภาพประกอบ 4.7 ไฟล์ใน Folder Calculate

ที่มา : ณปภัช วรรณตรง (2564 : 4)

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

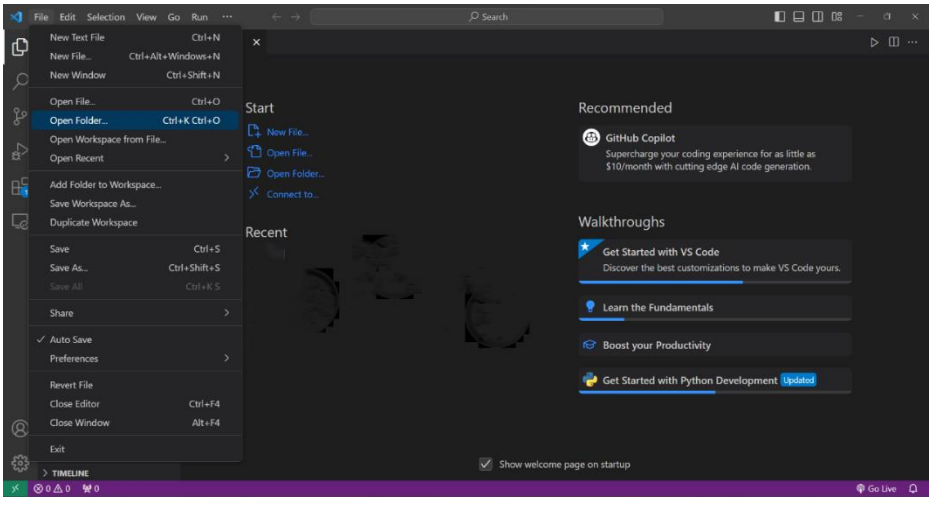
1.5.1 เปิดโปรแกรม VSCode



ภาพประกอบ 4.8 เปิดโปรแกรม VSCode

ที่มา : ณปภัช วรรณตรง (2564 : 5)

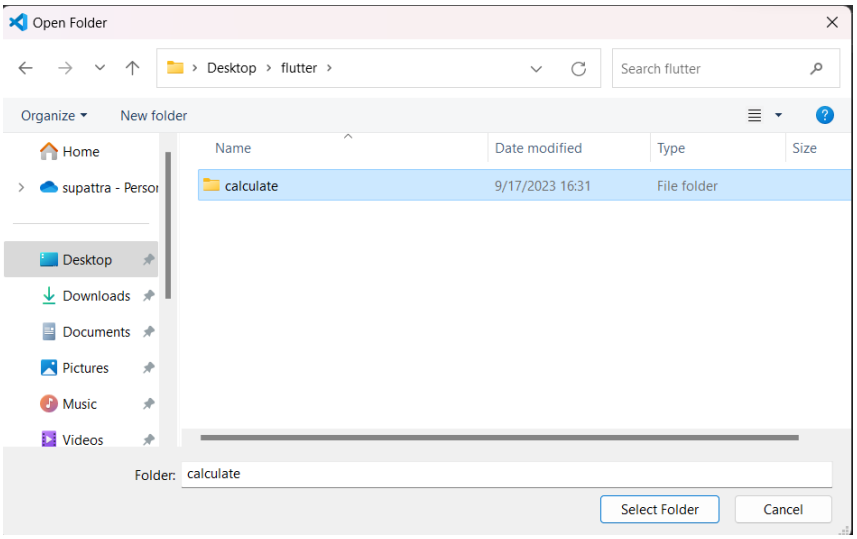
1.5.2 ไปที่ File > Open Folder



ภาพประกอบ 4.9 เปิดไฟล์ Project Calculate

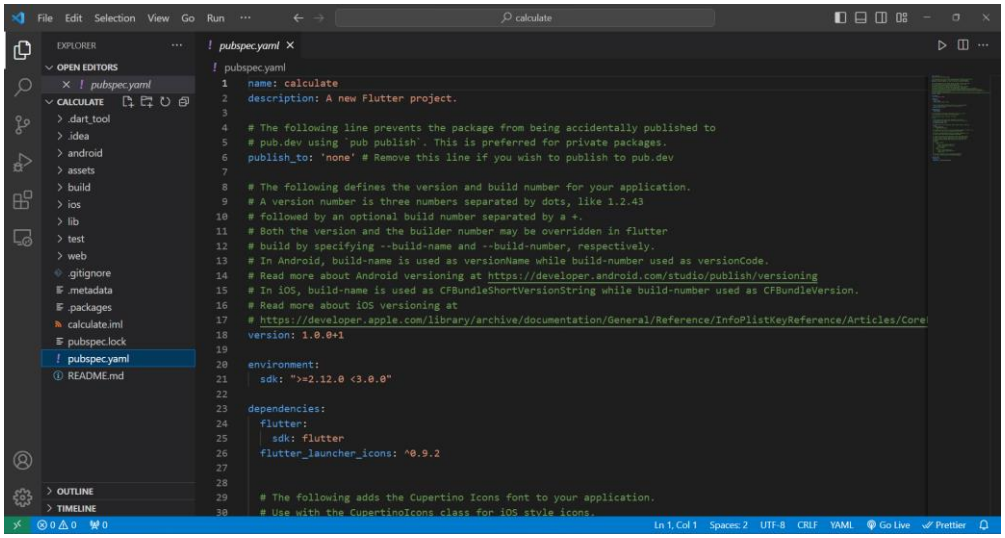
ที่มา : ณปภัช วรรณตรง (2564 : 5)

1.5.3 เลือกที่ Folder Calculate > จากนั้นให้กด Select Folder



ภาพประกอบ 4.10 เปิดไฟล์ Project Calculate (ต่อ)

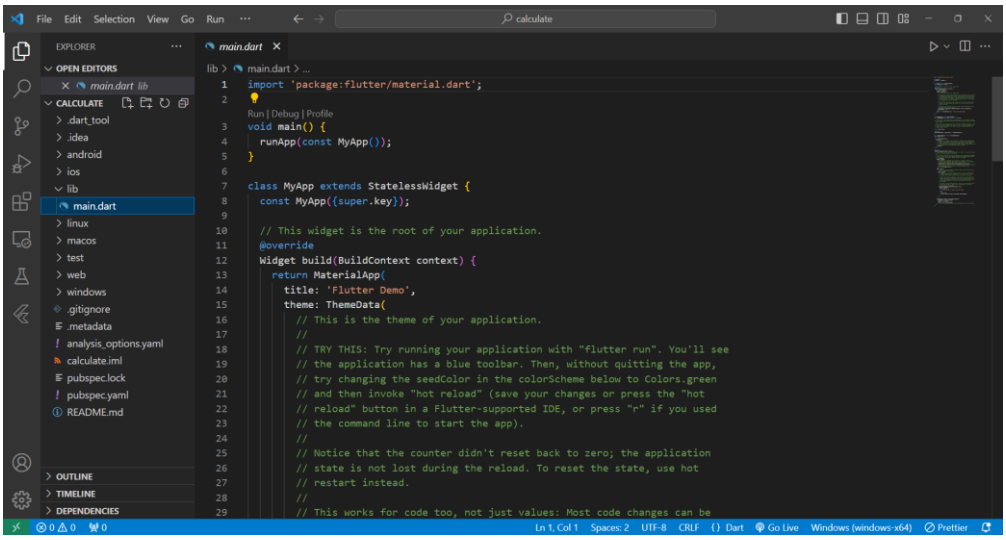
ที่มา : ณปภัช วรรณตรง (2564 : 5)



ภาพประกอบ 4.11 เปิดไฟล์ Project Calculate (ต่อ)

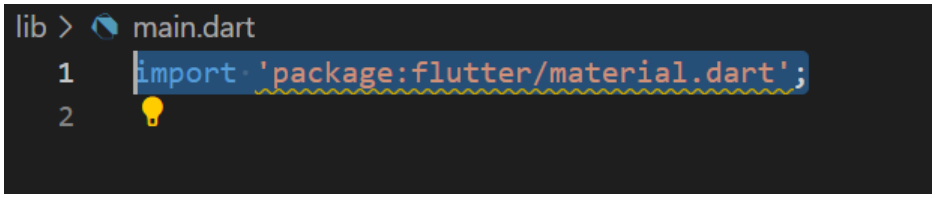
ที่มา : ณปภัช วรรณตรง (2564 : 5)

1.5.4 ไปที่ Folder main.dart เพื่อทำการลบโค้ดที่ไม่ได้ใช้ออก ให้เหลือเฉพาะโค้ดส่วน import



ภาพประกอบ 4.12 ลบโค้ดที่ไม่ได้ใช้ออกจากไฟล์ main.dart

ที่มา : ณปภัช วรรณตรง (2564 : 5)

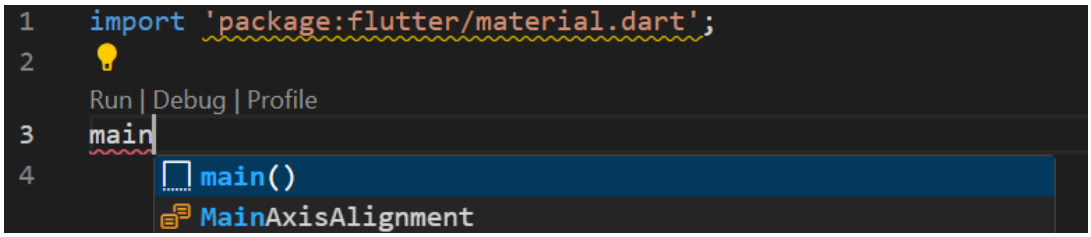


ภาพประกอบ 4.13 ลบโค้ดที่ไม่ได้ใช้ออกจากไฟล์ main.dart (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 5)

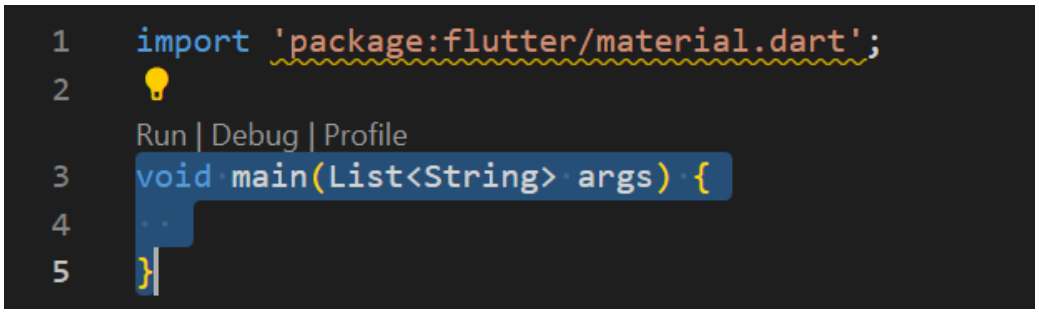
1.6 เมื่ออยู่ในไฟล์ main.dart ให้ทำการสร้างฟังก์ชัน void main(){} เข้าไปในไฟล์ main.dart

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



ภาพประกอบ 4.14 สร้างฟังก์ชัน void main(){}
ที่มา : ณปภัช วรรณตรง (2564 : 6)

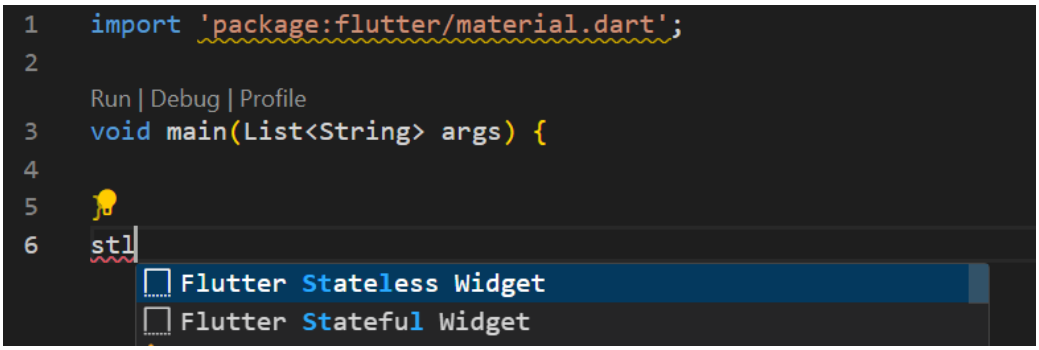
1.6.2 หลังจากทำการเลือกแล้วจะได้ชุดคำสั่งหรือฟังก์ชันดังนี้



ภาพประกอบ 4.15 สร้างฟังก์ชัน void main(){} (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 6)

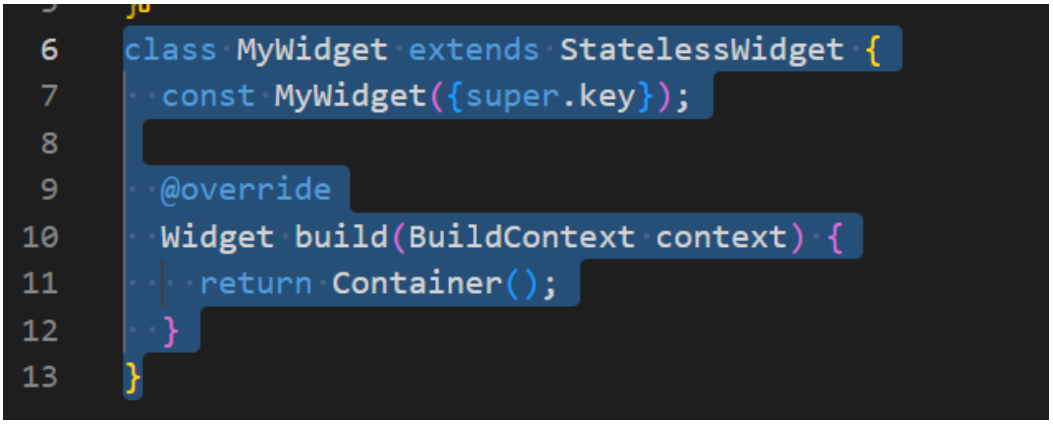
1.8 หลังจากที่ได้ทำการสร้างฟังก์ชัน void main(){} แล้วให้ทำการสร้าง class StatelessWidget(){}

พิมพ์คำสั่ง stl จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง Flutter stateless widget โดยการกด Click ที่คำสั่งหรือ Enter



ภาพประกอบ 4.16 สร้าง class StatelessWidget(){}

ที่มา : ณปภัช วรรณตรง (2564 : 7)



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

ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : 8)

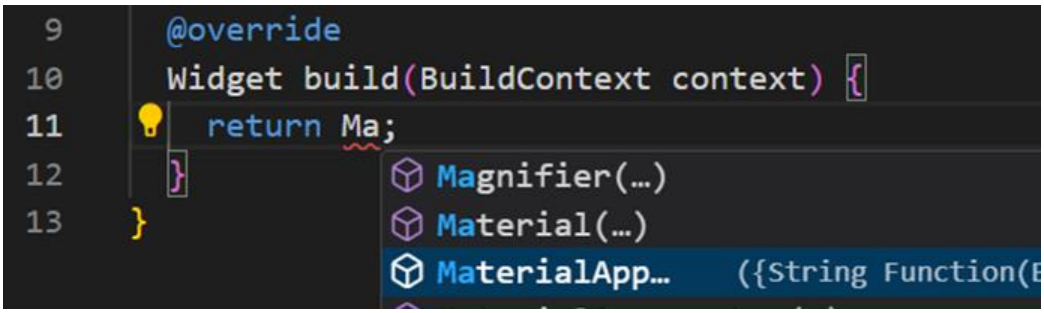
```
6  class Myapp extends StatelessWidget {
7    const Myapp({super.key});
8
9    @override
10   Widget build(BuildContext context) {
11     return Container();
12   }
13 }
```

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

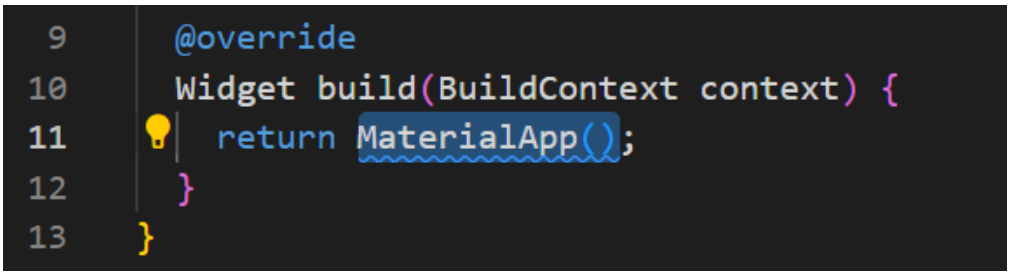
ที่มา : ณปภัช วรรณตรง (2564 : 8)

1.10 เมื่อทำการตั้งชื่อของ class แล้วให้ทำเปลี่ยนชื่อ Container เป็น MaterialApp

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



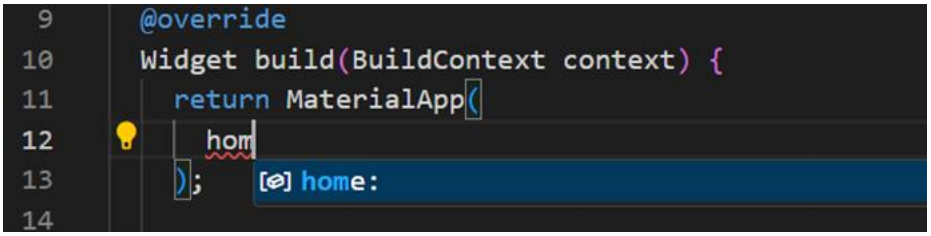
ภาพประกอบ 4.20 เปลี่ยนชื่อ class Container เป็น class MaterialApp
ที่มา : ณปภัช วรรณตรง (2564 : 9)



ภาพประกอบ 4.21 เปลี่ยนชื่อ class Container เป็น class MaterialApp (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 9)

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

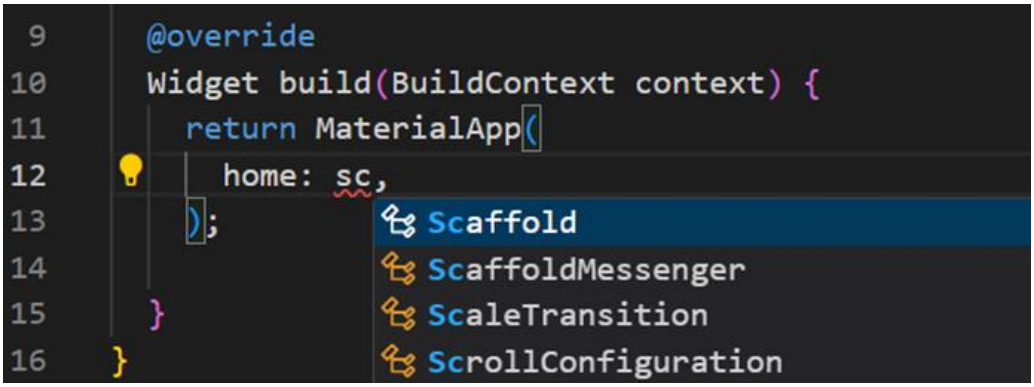
1.11.1 พิมพ์คำสั่ง home จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง home: โดยการกด Click ที่คำสั่งหรือ Enter



ภาพประกอบ 4.22 เพิ่มโค้ด home: Scaffold(),
ที่มา : ณปภัช วรรณตรง (2564 : 10)

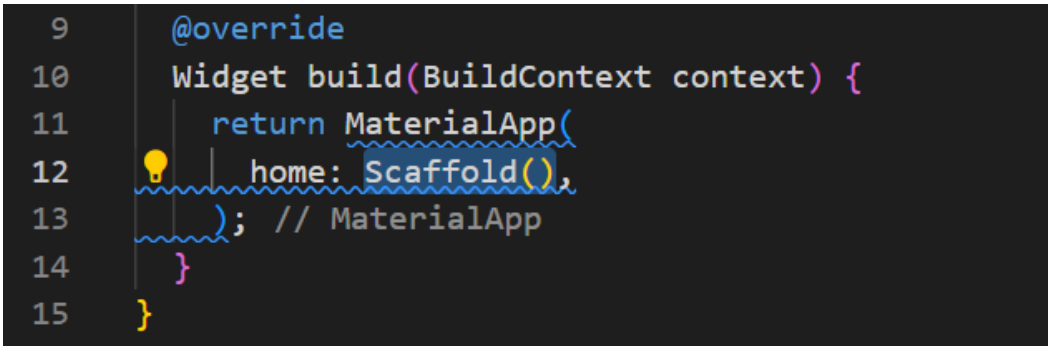
1.11.2 เมื่อได้ home: แล้วให้ทำการพิมพ์โค้ด Scaffold(), เข้าไปใน home:

1.11.3 พิมพ์คำสั่ง sc จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง Scaffold โดยการกด Click ที่คำสั่งหรือ Enter



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

ที่มา : ณปภัช วรรณตรง (2564 : 10)



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

ที่มา : ณปภัช 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 และดร็อปดาวน์ (Drop Down) เป็นต้น

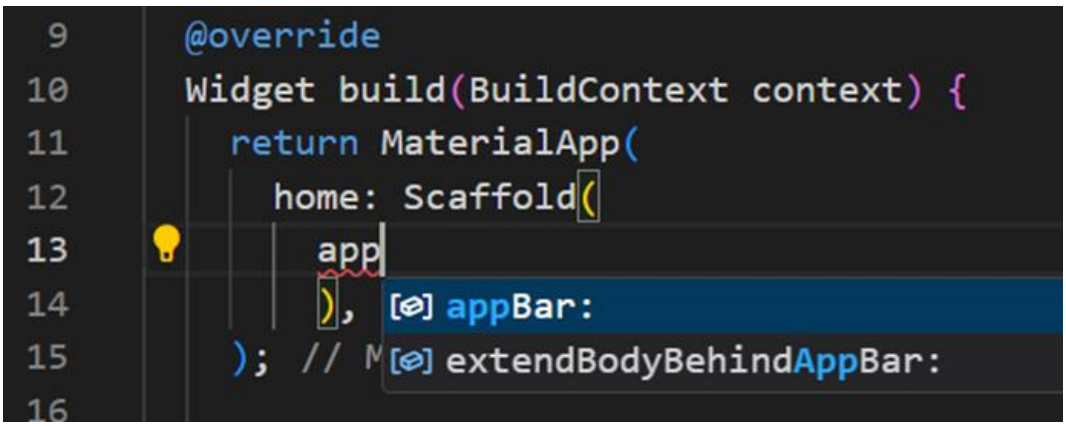
2.4 Automaticallyimply Leading เป็นคุณสมบัติในการกำหนดการแสดงผลของตัวนำทางแบบอัตโนมัติจะปรากฏขึ้นเมื่อมีการเรียกใช้งาน**ฉาก**แสดงผลต่อ ๆ กันไป โดยจะแสดงเป็นสัญลักษณ์ เพื่อให้สามารถย้อนกลับไปฉากหรือหน้าจอใช้งานก่อนหน้าได้ ซึ่งสามารถกำหนดค่าคุณสมบัติเป็นแบบตรรกะ คือ จริงหรือเท็จ ค่าตั้งต้นจะมีค่าเป็น true คือ เปิดการแสดงผลของตัวนำทางและหากกำหนดค่าเป็น false จะเป็นการปิดการแสดงผลของตัวนำทางและส่งผลให้ไม่มีช่องว่างในส่วน Leading การแสดงผลของส่วนที่เป็น Title จึงมีพื้นที่ในการแสดงผลมากขึ้น แต่สำหรับหน้าแรกหรือหน้าจอหลักนั้นเป็นจุดเริ่มต้นของการเรียกใช้งานจะไม่มีสัญลักษณ์ของตัวนำทางขึ้น

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

3. ใส่ AppBar Widget

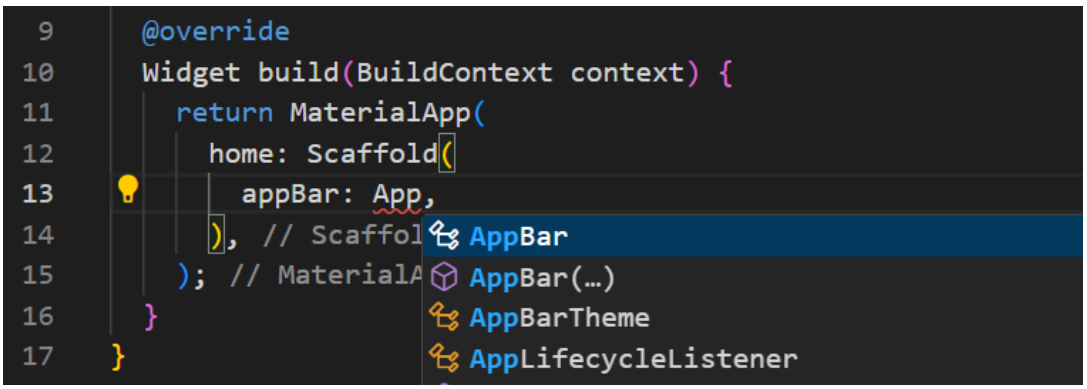
เมื่อทำการเพิ่มโค้ด home: Scaffold(), เสร็จสิ้นแล้ว ในส่วนของ Scaffold() ให้ใส่โค้ด appBar: AppBar(title: Text(“แอปพลิเคชันคำนวน”),),

3.1 พิมพ์คำสั่ง app จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง appBar: โดยการกด Click ที่คำสั่งหรือ Enter



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

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



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

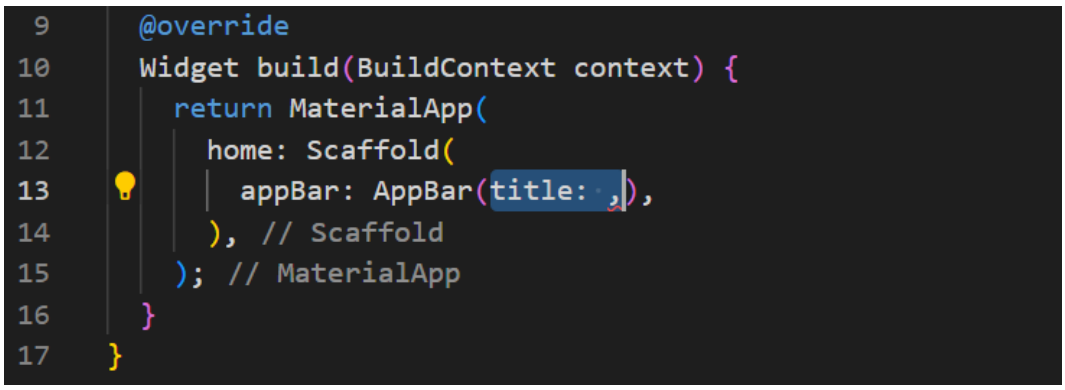
3.3 เมื่อทำการเพิ่ม AppBar() เรียบร้อยแล้ว ให้ทำการเพิ่มโค้ด title: , เข้าไป โดยพิมพ์คำสั่ง ti จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง title: โดยการกด Click ที่คำสั่งหรือ Enter

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

title:
titleSpacing:
titleTextStyle:
centerTitle:

ภาพประกอบ 4.28 เพิ่ม title: , เข้าไปใน Tag AppBar()

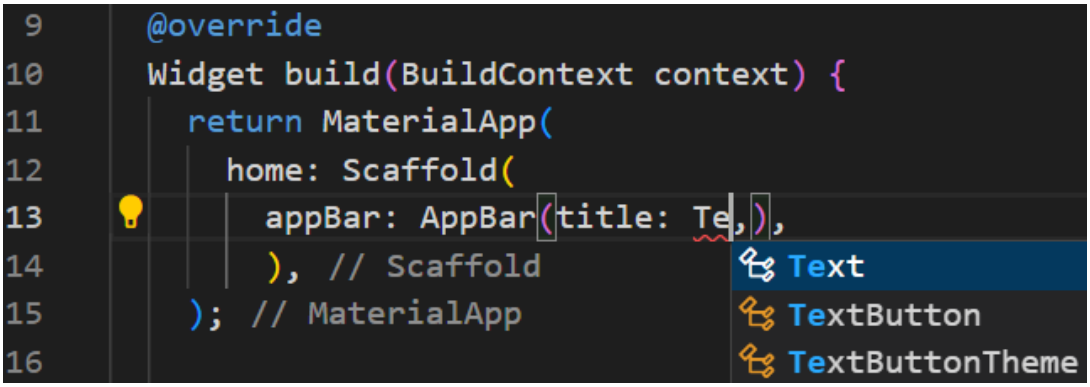
ที่มา : ณปภัช วรรณตรง (2564 : 11)



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

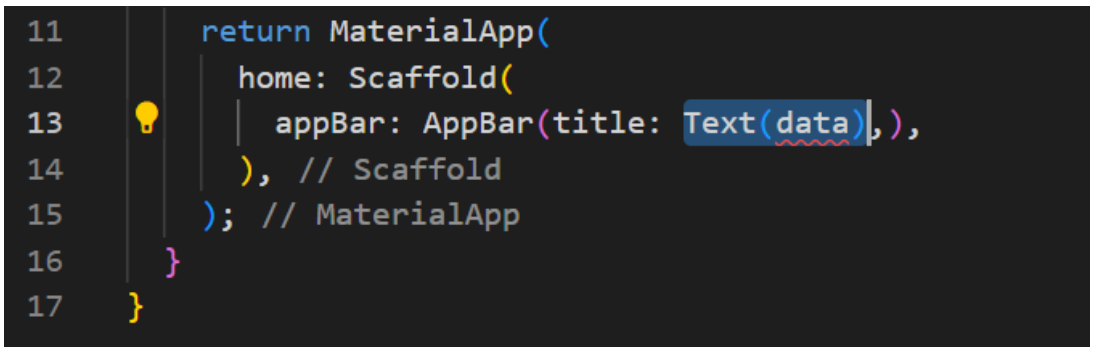
ที่มา : ณปภัช วรรณตรง (2564 : 11)

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



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

ที่มา : ณปภัช วรรณตรง (2564 : 11)



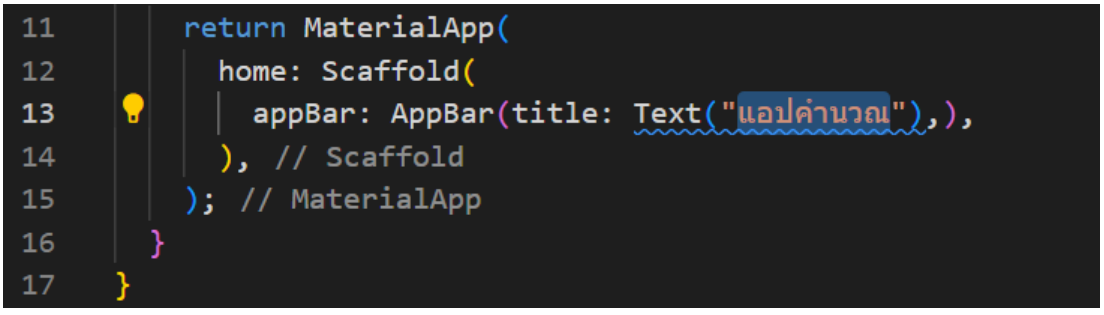
ภาพประกอบ 4.31 เพิ่ม Text(), เข้าไปใน Tag title: , (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 11)

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

ภาพประกอบ 4.32 เปลี่ยนข้อความใน Text(),

ที่มา : ณปภัช วรรณตรง (2564 : 11)



ภาพประกอบ 4.33 เปลี่ยนข้อความใน Text(), (ต่อ)

ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : 11)

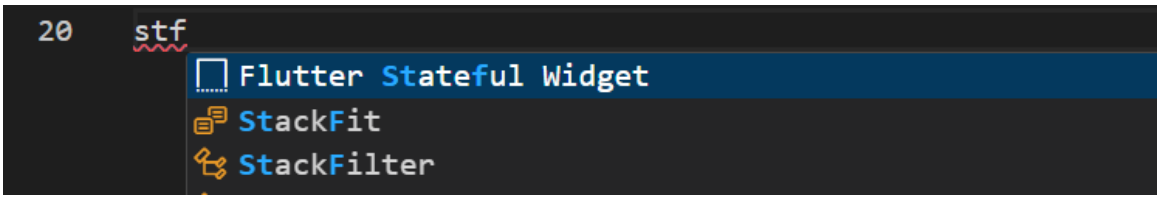
3.6 เมื่อได้ส่วน appBar: , แล้ว ให้ทำการเพิ่มโค้ด body: , เข้าไปใน Scaffold() ต่อจาก appBar: ,

```
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 : ณปภัช วรรณตรง (2564 : 12)

3.7 เมื่อทำการสร้าง class StatelessWidget เสร็จสิ้นแล้วให้ทำการสร้าง class StatefulWidget ต่อจาก class StatelessWidget

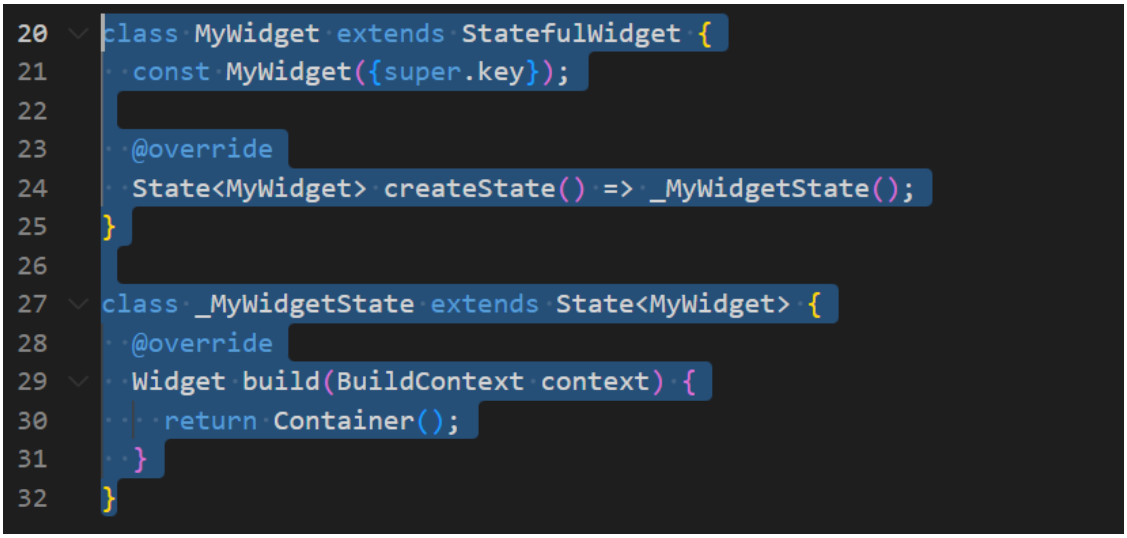
3.7.1 พิมพ์คำสั่ง stf จากนั้นจะปรากฏหน้าต่าง ให้ทำการเลือกคำสั่ง flutter: stateful widget โดยการกด Click ที่คำสั่งหรือ Enter



ภาพประกอบ 4.36 สร้าง class StatefulWidget

ที่มา : ณปภัช วรรณตรง (2564 : 13)

3.7.2 เมื่อทำการสร้าง class StatefulWidget แล้วให้ทำการเปลี่ยนชื่อ class เป็น Home



ภาพประกอบ 4.37 สร้าง class StatefulWidget (ต่อ)

ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : 13)

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

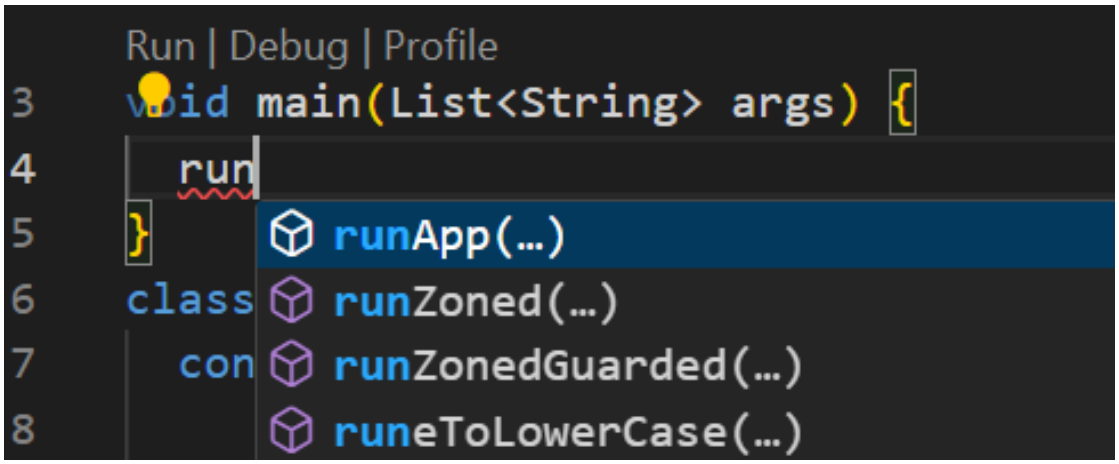
```
 9    @override
10    Widget build(BuildContext context) {
11      return MaterialApp(
12        home: Scaffold(
13          appBar: AppBar(title: Text("แอปคำนวณ"),),
14          body: Home(),
15        ), // Scaffold
16      ); // MaterialApp
17    }
18  }
19
```

ภาพประกอบ 4.39 เพิ่มโค้ด Home()

ที่มา : ณปภัช วรรณตรง (2564 : 14)

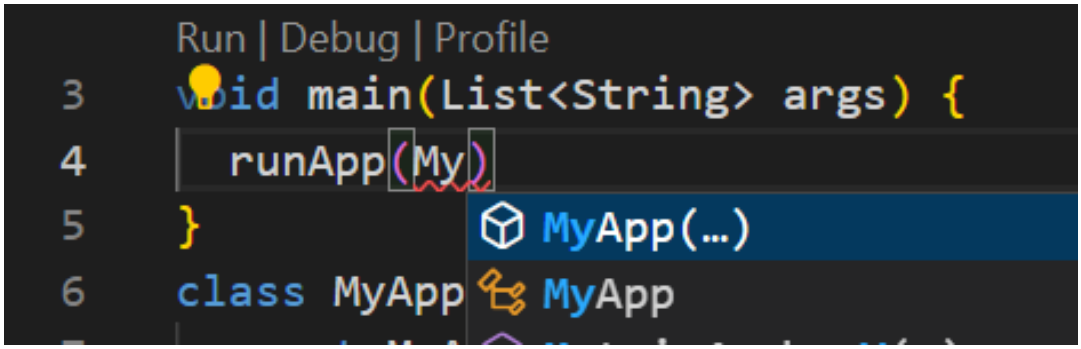
3.9 จากนั้นให้กลับไปที่ส่วนของ class void main ให้ทำการเพิ่มโค้ด runApp (MyApp()); เพื่อเรียกใช้ class MyApp(); ที่ได้ทำการสร้างไว้ก่อนหน้า

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



ภาพประกอบ 4.40 เรียกใช้งาน class MyApp(){}
ที่มา : ณปภัช วรรณตรง (2564 : 15)

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



ภาพประกอบ 4.41 เรียกใช้งาน class MyApp(){} (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 15)

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

ภาพประกอบ 4.42 เรียกใช้งาน class MyApp(){} (ต่อ)
ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : 16)

```
27  class _HomeState extends State<Home> {
28    @override
29    Widget build(BuildContext context) {
30      return Text;
31    }                Text(…)        (String data, {Key? key, Te
32  }                  Text
```

ภาพประกอบ 4.44 เพิ่มโค้ด Text(“Home Page”); (ต่อ)
ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (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 : ณปภัช วรรณตรง (2564 : 16)

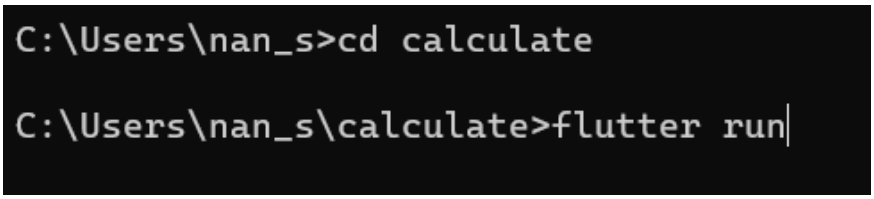
3.11 เมื่อทำการเพิ่มโค้ด Text(“Home Page”); เสร็จสิ้นแล้ว ให้กลับไปที่หน้าต่าง cmd ทำการเข้าไปใน Folder calculate โดยการใช้คำสั่ง cd calculate

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

C:\Users\nan_s\calculate>
```

ภาพประกอบ 4.47 สั่งการทำงาน Flutter

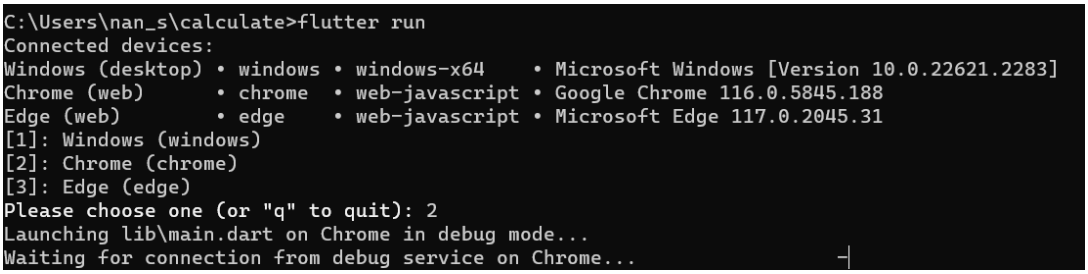
ที่มา : ณปภัช วรรณตรง (2564 : 17)

เมื่อเข้าไปใน Folder calculate แล้วให้ทำการพิมพ์คำสั่ง flutter run จากนั้นกด Enter



ภาพประกอบ 4.48 สั่งการทำงาน Flutter (ต่อ)

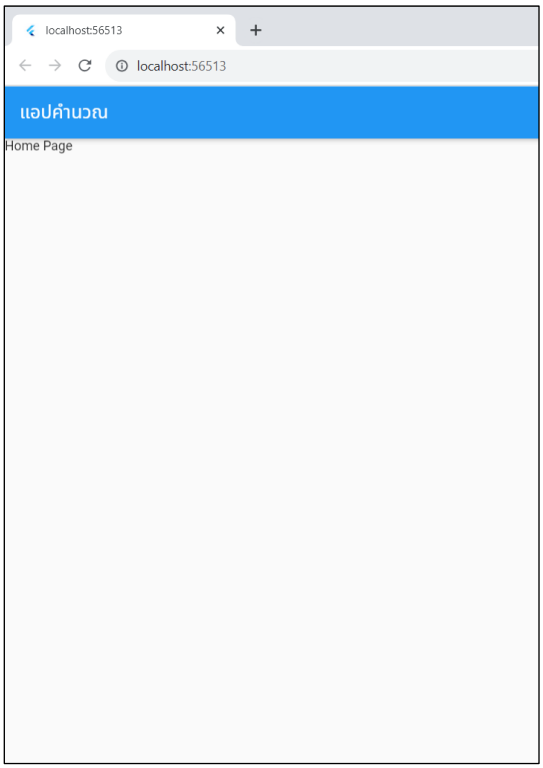
ที่มา : ณปภัช วรรณตรง (2564 : 17)



ภาพประกอบ 4.49 สั่งการทำงาน Flutter (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 17)

เมื่อทำการสั่งการทำงานผ่านแล้ว จะปรากฏหน้าต่าง Web Browser ขึ้นมา



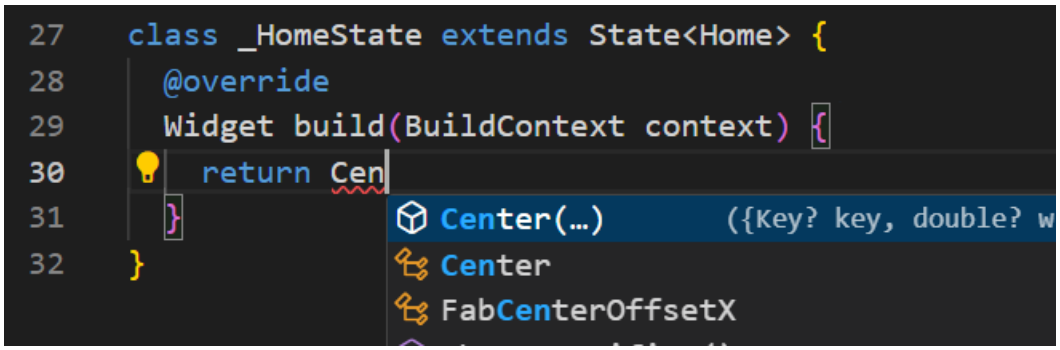
ภาพประกอบ 4.50 สั่งการทำงาน Flutter (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 17)

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  }
```

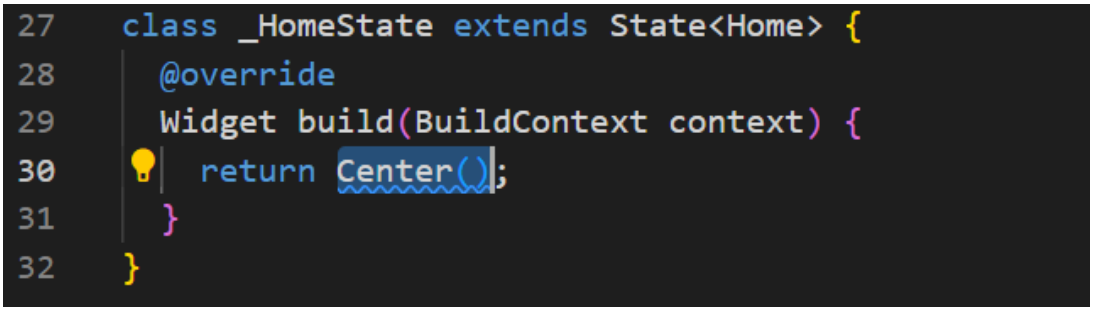
ภาพประกอบ 4.51 เพิ่มโค้ด Center();
ที่มา : ณปภัช วรรณตรง (2564 : 18)

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



ภาพประกอบ 4.52 เพิ่มโค้ด Center(); (ต่อ)

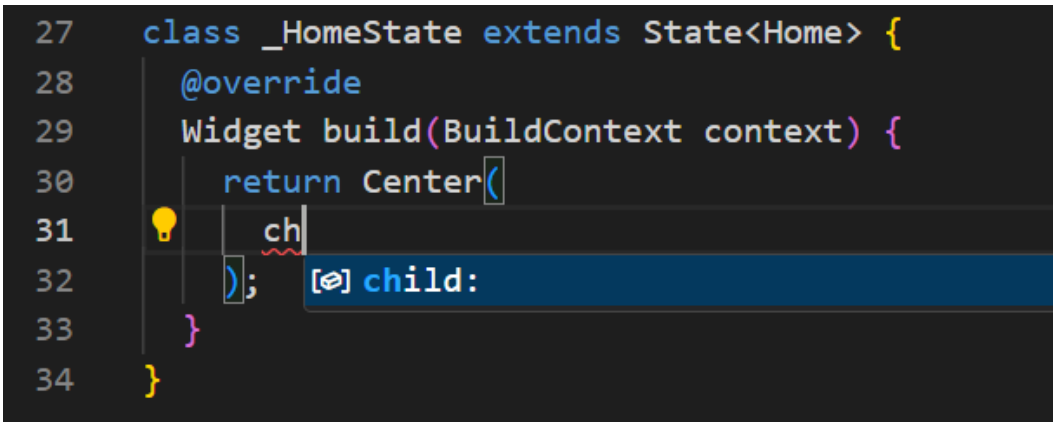
ที่มา : ณปภัช วรรณตรง (2564 : 18)



ภาพประกอบ 4.53 เพิ่มโค้ด Center(); (ต่อ)

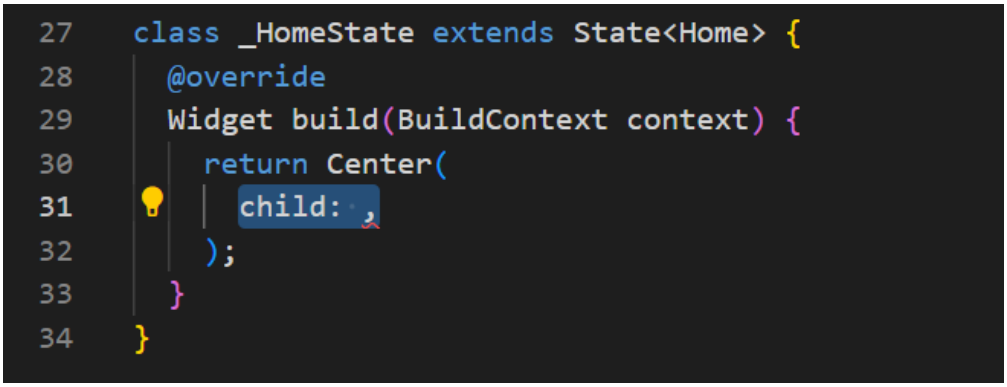
ที่มา : ณปภัช วรรณตรง (2564 : 18)

เมื่อได้ในส่วนของ Center(); แล้ว ให้ทำการเพิ่มโค้ดเข้าไป โดยใช้โค้ด child: ,



ภาพประกอบ 4.54 เพิ่มโค้ด child: ,

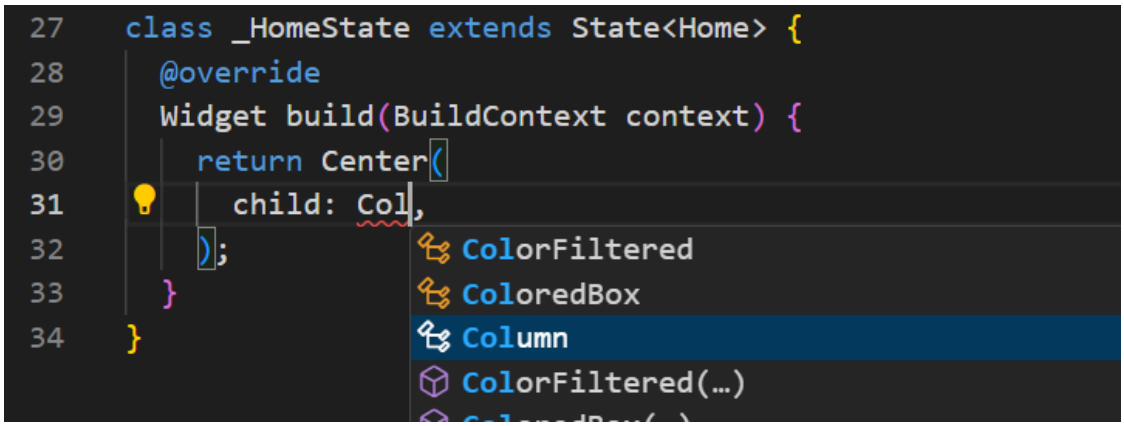
ที่มา : ณปภัช วรรณตรง (2564 : 18)



ภาพประกอบ 4.55 เพิ่มโค้ด child: , (ต่อ)

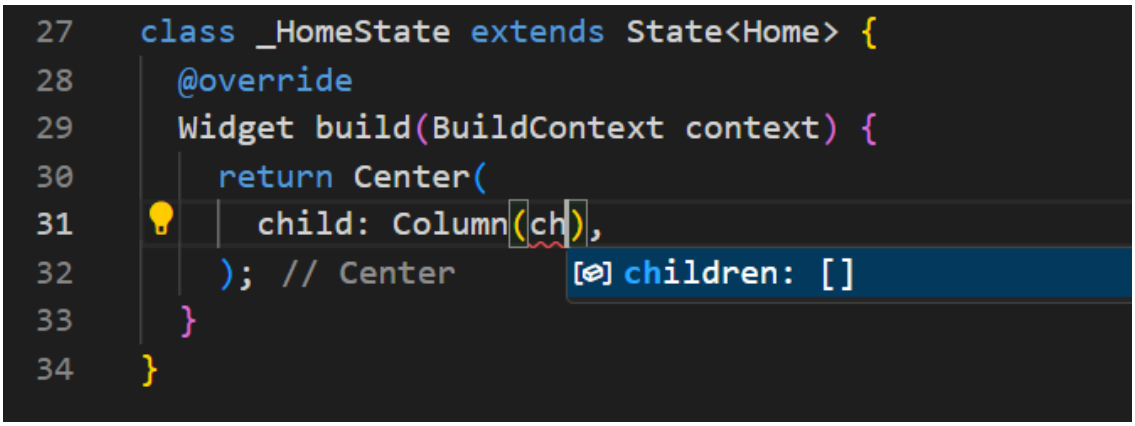
ที่มา : ณปภัช วรรณตรง (2564 : 18)

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



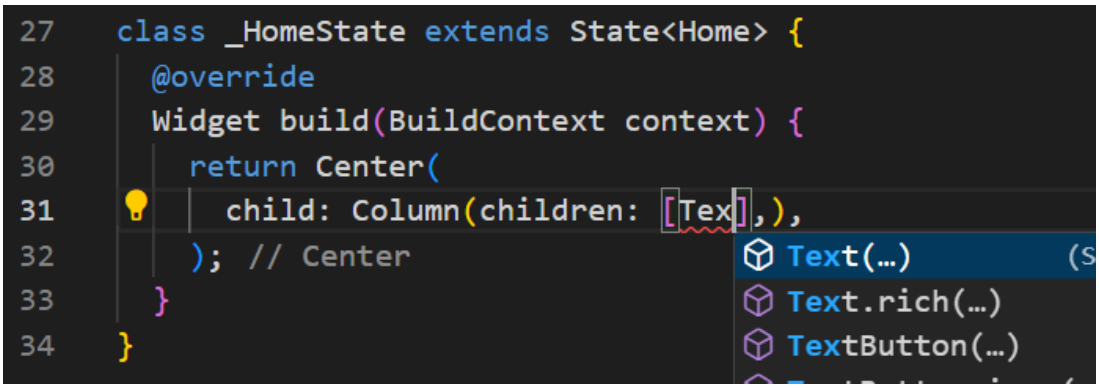
ภาพประกอบ 4.56 เพิ่มโค้ด Column(...)
ที่มา : ณปภัช วรรณตรง (2564 : 18)

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



ภาพประกอบ 4.57 เพิ่มโค้ด children: []
ที่มา : ณปภัช วรรณตรง (2564 : 18)

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



ภาพประกอบ 4.58 เพิ่มข้อความใน children: []

ที่มา : ณปภัช วรรณตรง (2564 : 18)

ภาพประกอบ 4.59 เพิ่มข้อความใน children: [] (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 18)

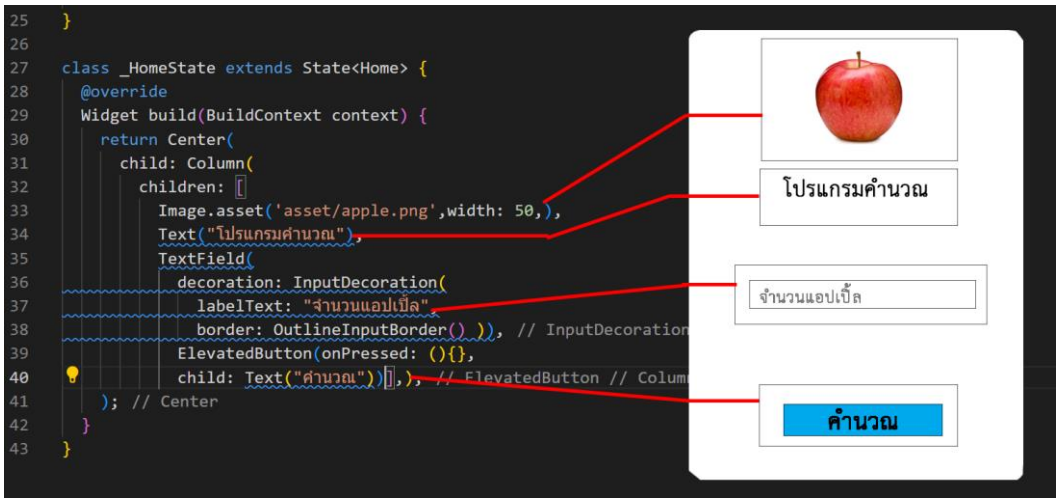
เมื่อเพิ่มข้อความเสร็จแล้ว ให้ใส่ ; ปิด class Center() ด้วย

```
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 : ณปภัช วรรณตรง (2564 : 18)

จากโค้ดทั้งหมดจากขั้นตอนการทำที่ผ่านมา เมื่อแสดงเปรียบเทียบกับผลลัพธ์ที่ได้ Column ใน Flutter จะทำการเก็บข้อมูลที่ได้ทำการพิมพ์คำสั่งเอาไว้ในแนวตั้งเป็นขั้น ๆ ลงไปตามจำนวนโค้ดที่คั้นด้วยเครื่องหมาย Comma (,) ซึ่งในแต่ละ Column สามารถมี Widget ได้หลากหลาย ดังภาพ



ภาพประกอบ 4.61 โครงสร้างของแอปพลิเคชัน

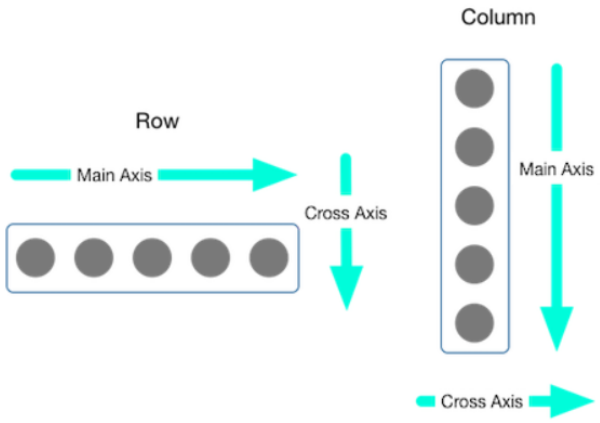
ที่มา : ณปภัช วรรณตรง (2564 : 21)

4. Column และ Row Widget

Column เป็นวิดเจ็ตในแบบ multi-child ใช้สำหรับจัดเรียงวิดเจ็ตในแนวตั้ง โดยวิดเจ็ตทั้งหมดที่อยู่ภายใต้ Column วิดเจ็ต จะถูกนำมากำหนดลงในพร็อพเพอร์ตี้ 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)

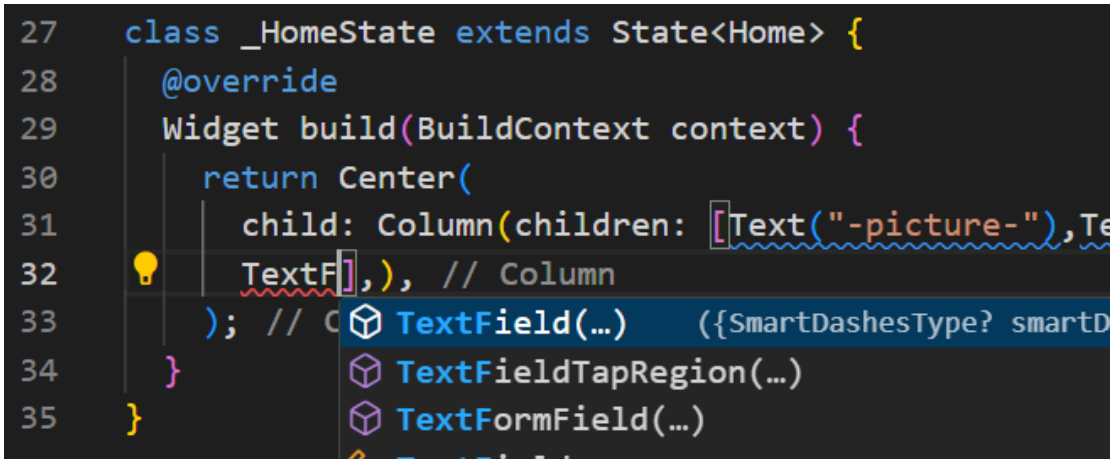
สามารถแสดงความแตกต่างของ Column และ Row ได้ดังภาพ



ภาพประกอบ 4.62 โครงสร้าง Column ของแอปพลิเคชัน (ต่อ)
ที่มา : Flutter (2020 : 20)

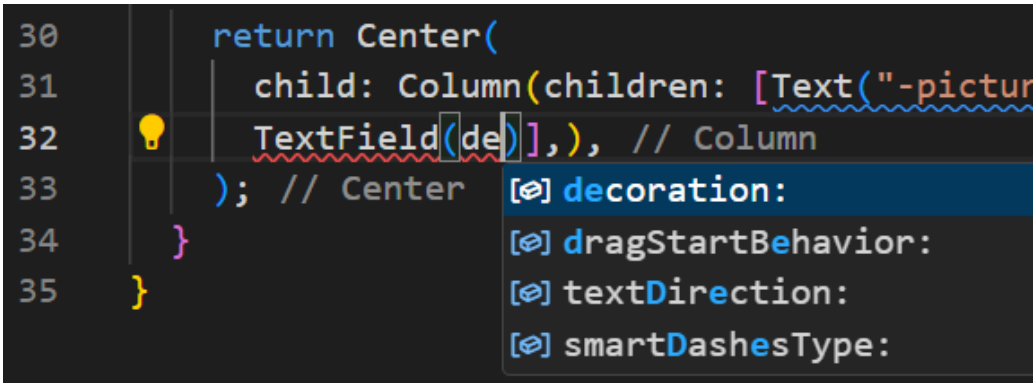
ศึกษาเพิ่มเติมเกี่ยวกับ Flutter Layout
https://docs.flutter.dev/development/ui/layout

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



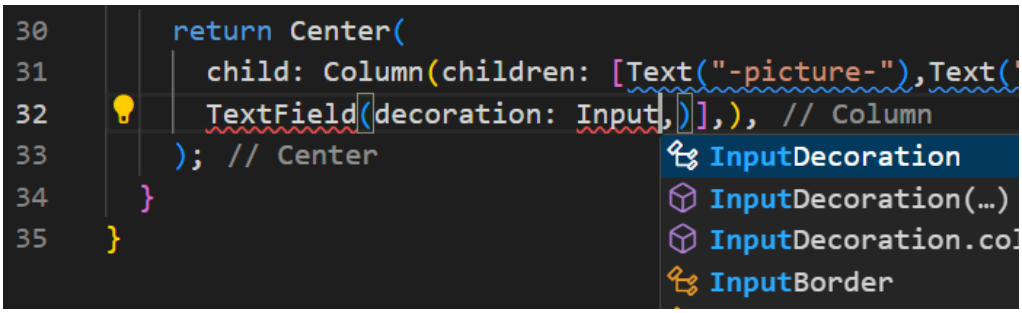
ภาพประกอบ 4.63 เพิ่ม TextField(...)
ที่มา : ณปภัช วรรณตรง (2564 : 22)

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



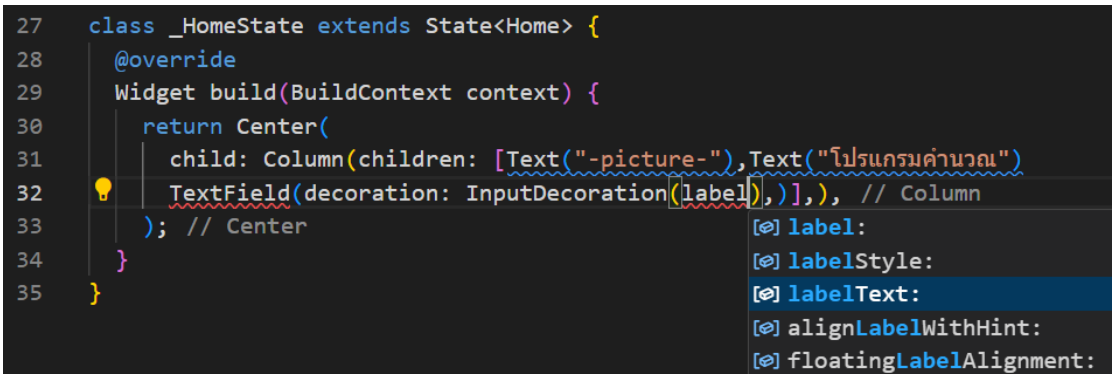
ภาพประกอบ 4.64 เพิ่ม decoration:
ที่มา : ณปภัช วรรณตรง (2564 : 22)

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



ภาพประกอบ 4.65 เพิ่ม InputDecoration
ที่มา : ณปภัช วรรณตรง (2564 : 22)

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

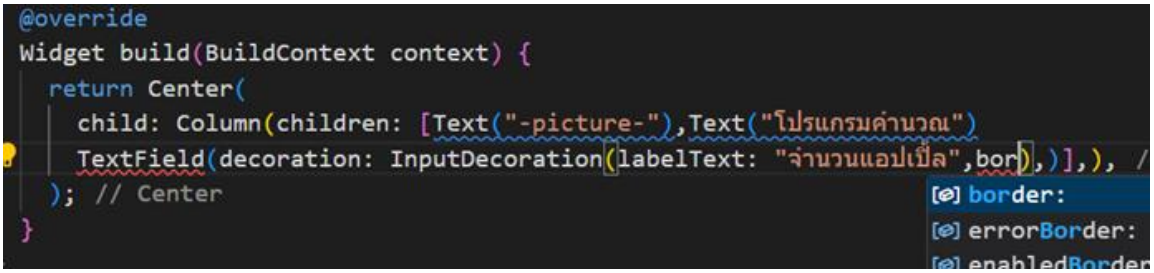


ภาพประกอบ 4.66 เพิ่ม labelText:
ที่มา : ณปภัช วรรณตรง (2564 : 22)

ภายใน labelText: ให้ใช้เครื่องหมาย “ ” เพื่อทำการเพิ่มข้อความที่ต้องการแสดง

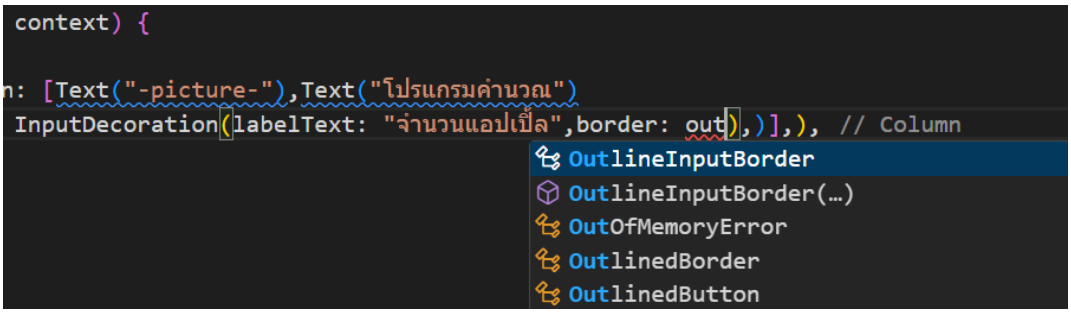
ภาพประกอบ 4.67 เพิ่มข้อความใน labelText:
ที่มา : ณปภัช วรรณตรง (2564 : 22)

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



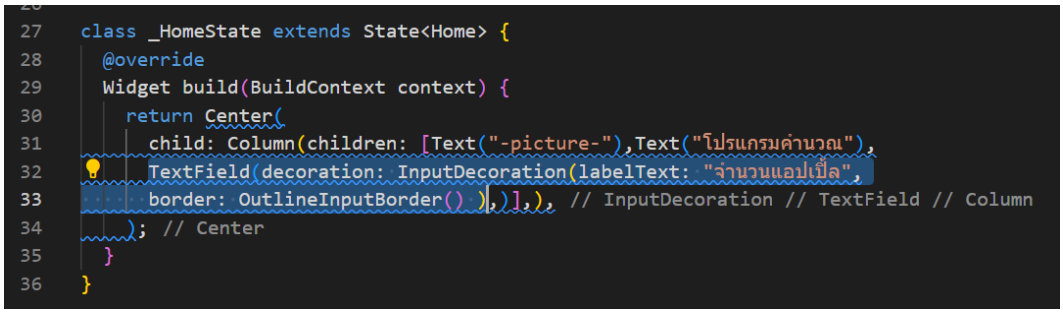
ภาพประกอบ 4.68 เพิ่มขอบให้กับ labText:
ที่มา : ณปภัช วรรณตรง (2564 : 22)

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



ภาพประกอบ 4.69 เพิ่มขอบให้กับ labText: (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 22)



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

ที่มา : ณปภัช วรรณตรง (2564 : 22)

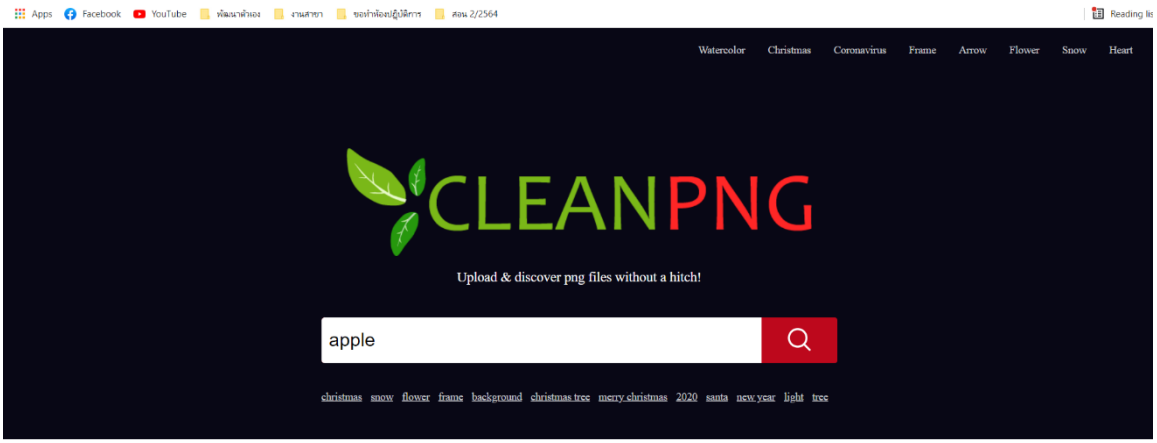
หากทำการติดตั้ง Prettier Extension สามารถทำการจัดเรียงโค้ดเพื่อตรวจสอบความถูกต้องของโค้ด และจัดเรียงโค้ดให้สวยงามได้ด้วยการกด Alt+Shift+F

ภาพประกอบ 4.71 จัดเรียงโค้ดด้วย Prettier Extension

ที่มา : ณปภัช วรรณตรง (2564 : 23)

5. ใส่รูปภาพ

5.1 ให้ทำการใส่รูปภาพ โดยใช้เว็บไซต์ที่สามารถนำรูปภาพมาใช้แบบไม่ติดลิขสิทธิ์ได้ จากเว็บ cleanpng.com ซึ่งเป็นภาพประกอบไม่มีลิขสิทธิ์และพื้นหลังของรูปภาพโปร่งใส

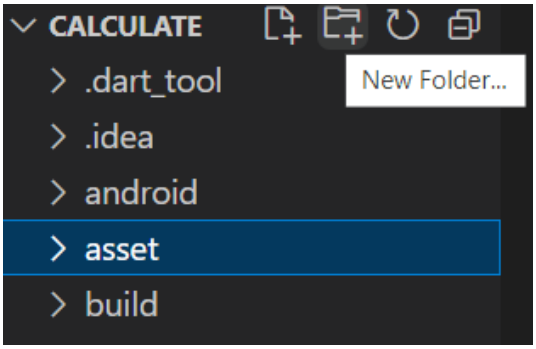


ภาพประกอบ 4.72 นำรูปภาพมาใช้งานบนหน้าเว็บ

ที่มา : CleanPNG. (2564 : 1)

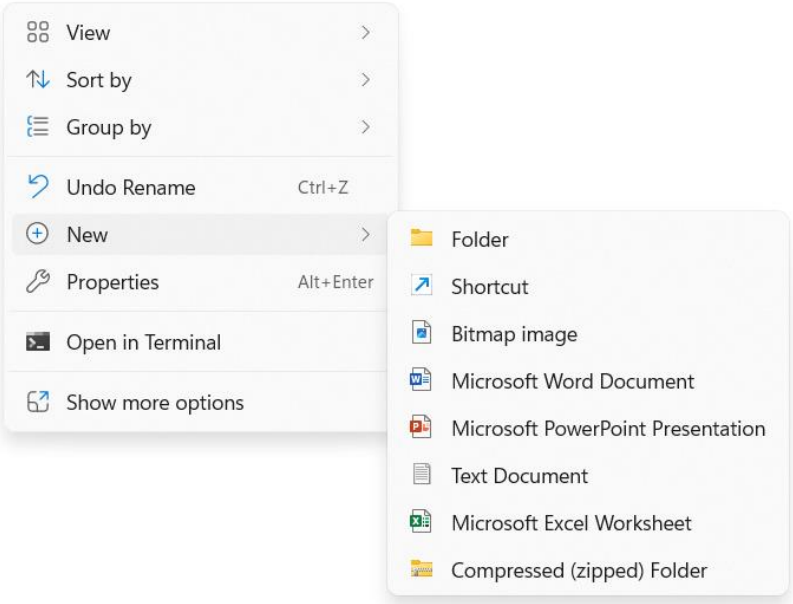
5.2 เมื่อทำการดาวน์โหลดรูปภาพที่ต้องการใช้ได้แล้ว ให้ไปสร้าง Folder assets ที่ Folder Calculate

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)

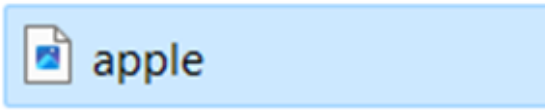
สร้าง Folder ผ่าน Desktop โดยให้เปิด Folder งานขึ้นมา จากนั้นให้คลิกขวาในพื้นที่ว่างของ Folder ไปที่ New > จากนั้นเลือกเป็น Folder เสร็จแล้วให้ทำการตั้งชื่อเป็น assets



ภาพประกอบ 4.74 สร้าง Folder assets ผ่าน Desktop
ที่มา : ณปภัช วรรณตรง (2564 : 25)

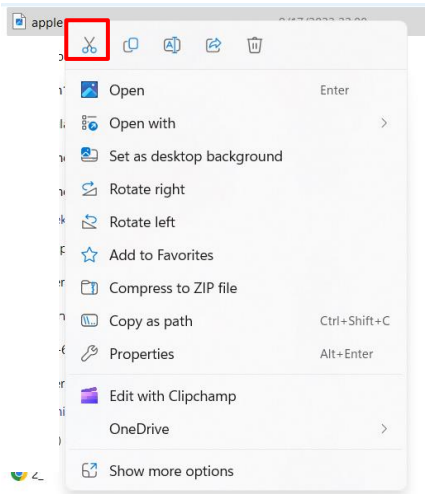
จากนั้นให้ทำการเปลี่ยนชื่อไฟล์ภาพประกอบได้ทำการดาวน์โหลดมาก่อนหน้า เพื่อให้ง่ายต่อการค้นหาและใช้งาน

ภาพประกอบ 4.75 เปลี่ยนชื่อไฟล์ภาพ
ที่มา : ณปภัช วรรณตรง (2564 : 26)



ภาพประกอบ 4.76 เปลี่ยนชื่อไฟล์ภาพ (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 26)

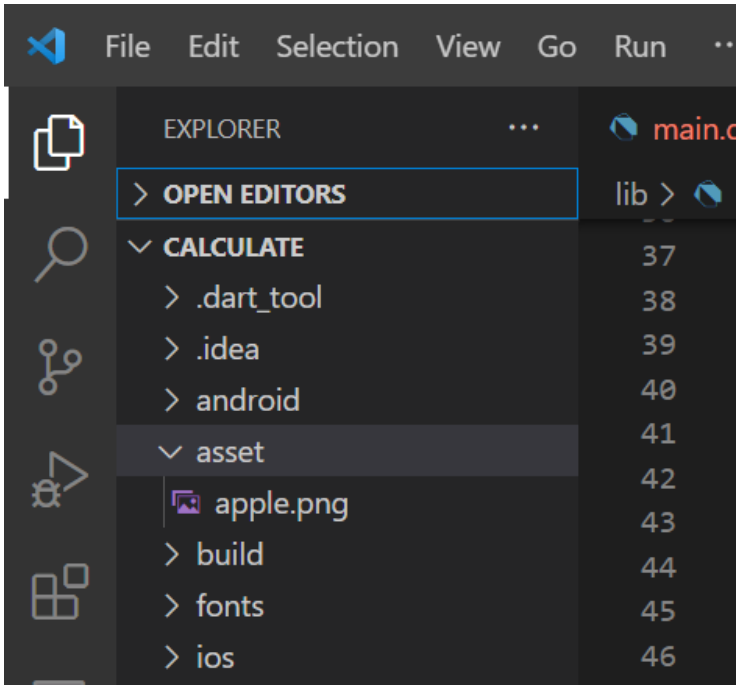
เมื่อเปลี่ยนชื่อไฟล์เสร็จแล้ว ให้ทำการย้ายไฟล์รูปภาพไปยัง Folder assets ที่ได้ทำการสร้างไว้ โดยการกดคลิกขวาที่รูปภาพ เลือก Cut จากนั้นให้ไปที่ Folder assets แล้วคลิกขวาใน Folder เลือก Paste หรือ Ctrl+v



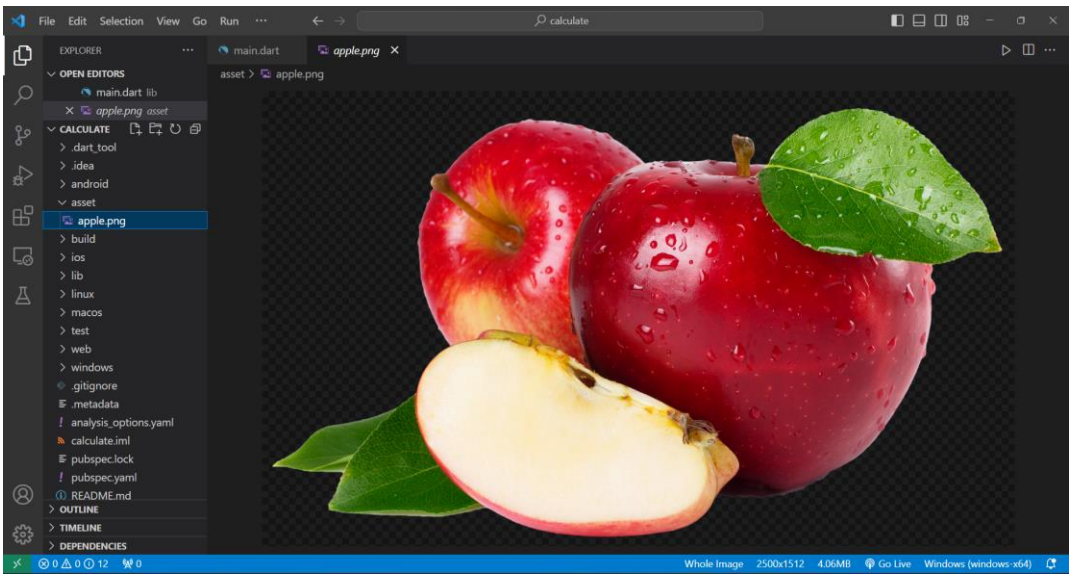
ภาพประกอบ 4.77 ย้ายภาพไปยัง Folder assets
ที่มา : ณปภัช วรรณตรง (2564 : 26)

ภาพประกอบ 4.78 ย้ายภาพไปยัง Folder assets (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 26)

เมื่อกลับไปที่โปรแกรม VSCode จะพบว่าไฟล์รูปภาพได้ปรากฏขึ้นในโปรแกรม VSCode สำหรับนำไปใช้งานเช่นเดียวกัน



ภาพประกอบ 4.79 ย้ายภาพไปยัง Folder assets (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 26)

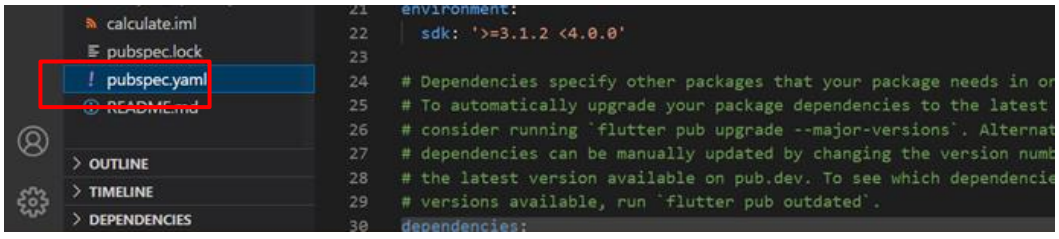


ภาพประกอบ 4.80 ย้ายภาพไปยัง Folder assets (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 26)

5.3 เมื่อทำการนำไฟล์ภาพเข้ามาเพื่อเรียกใช้แล้ว ให้ไปตั้งค่า assets ในไฟล์ pubspec.yaml

เข้าไปที่ไฟล์ pubspec.yaml จากนั้นให้ทำการเลื่อนไปที่ asset ซึ่งจะอยู่ใต้ user-material-design: ture



ภาพประกอบ 4.81 ตั้งค่า assets ในไฟล์ pubspec.yaml

ที่มา : ณปภัช วรรณตรง (2564 : 27)

```
60
61  # To add assets to your application, add an assets sectio
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 : ณปภัช วรรณตรง (2564 : 27)

เมื่อเจอตำแหน่งที่ต้องการแก้ไขแล้วให้ทำการเปิดคอมเม้น (Comment) โดยการครอบโค้ดบรรทัดที่ต้องการ 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 เปิดคอมเม้นเพื่อให้โค้ดทำงานได้

ที่มา : ณปภัช 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 เปิดดคอมเม้นเพื่อให้โค้ดทำงานได้ (ต่อ)

ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : 27)

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

ภาพประกอบ 4.86 เปลี่ยนชื่อไฟล์ให้ตรงกัน (ต่อ)

ที่มา : ณปภัช 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 : ณปภัช 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 : ณปภัช 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("โ 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 : ณปภัช 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 : ณปภัช 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",bo
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",borde
37          ) // TextField
38          ],),); // Column // Center
39      }
40    }
```

ภาพประกอบ 4.92 แก้ไขโค้ดเพื่อนำรูปภาพไปแสดงที่หน้าเว็บ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 28)

เมื่อกำหนดตำแหน่งและชื่อไฟล์เรียบร้อยแล้ว ให้ทำการกำหนดขนาดของภาพ โดยการพิมพ์คำสั้ง 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",bor
37            ) // TextField
38          ],),); // Column // Center
39      }
40    }
```

ภาพประกอบ 4.93 แก้ไขโค้ดเพื่อนำรูปภาพไปแสดงที่หน้าเว็บ (ต่อ)

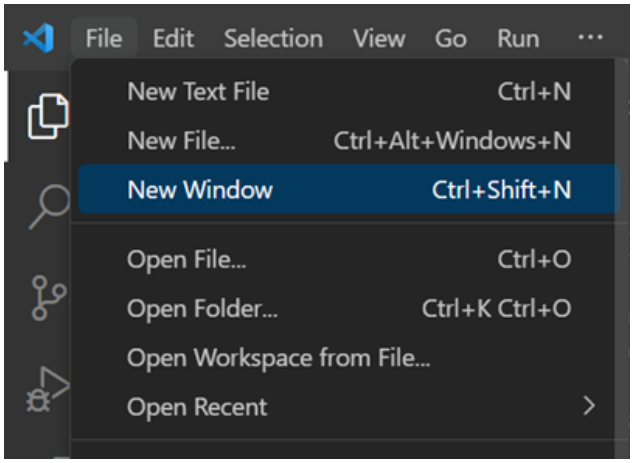
ที่มา : ณปภัช วรรณตรง (2564 : 28)

```
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

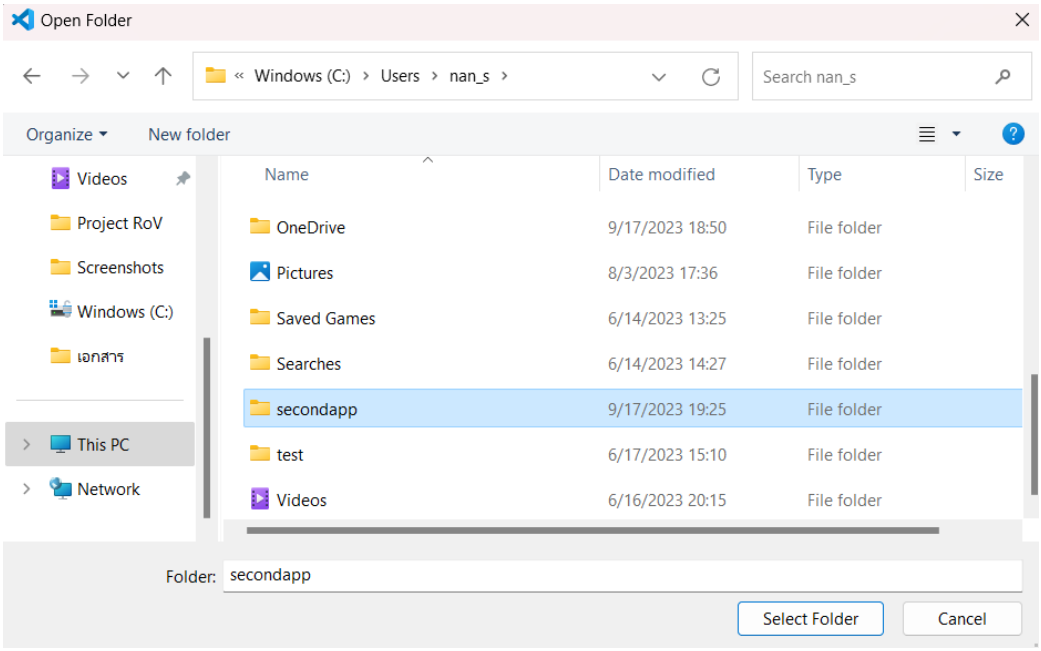
ที่มา : ณปภัช วรรณตรง (2564 : 28)

5.5 ให้ทำการเปิดหน้าต่างขึ้นมาใหม่เพื่อทำการดูโค้ดเก่าและนำโค้ดตกแต่งของงานเก่ามาแก้ไขโค้ดเพื่อตกแต่งฟอนต์ในไฟล์งานปัจจุบัน โดยการเปิดคล้ายกับการเปิด Folder calculate ไปที่ New Window และเลือก Folder secondapp



ภาพประกอบ 4.95 เปิด New Window

ที่มา : ณปภัช วรรณตรง (2564 : 30)

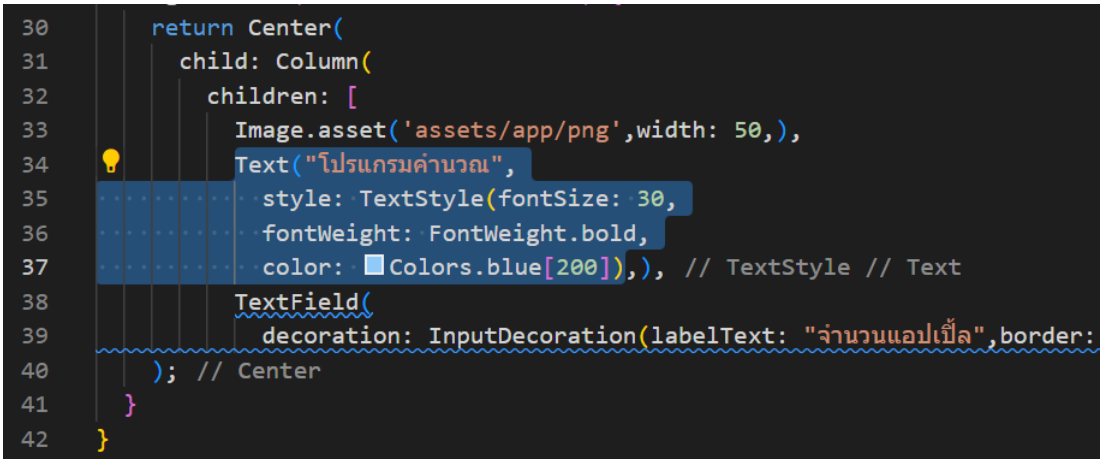


ภาพประกอบ 4.96 เปิด New Window (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 30)

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

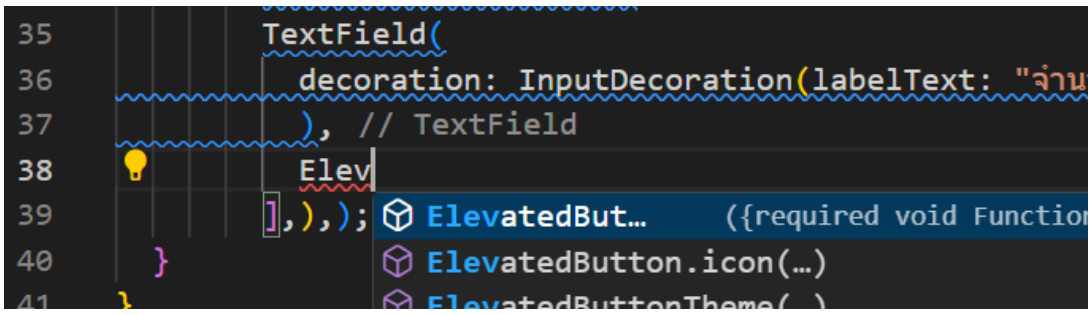
ควรตรวจสอบ () และ , ให้ถูกต้องเพื่อป้องกันการทำงานผิดพลาดของโค้ด



ภาพประกอบ 4.97 ตกแต่งคุณลักษณะโค้ดของฟอนต์

ที่มา : ณปภัช วรรณตรง (2564 : 31)

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



ภาพประกอบ 4.98 เพิ่มโค้ดที่เป็นปุ่ม

ที่มา : ณปภัช วรรณตรง (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: Ou
37                ), // TextField
38                ElevatedButton(onPressed: () {}, child: Text("คำนวณ")),
39              ],),); // Column // Center
40      }
41  }
```

ภาพประกอบ 4.100 เพิ่มโค้ดที่เป็นปุ่ม (ต่อ)

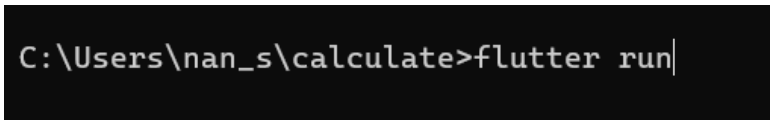
ที่มา : ณปภัช วรรณตรง (2564 : 32)

```
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)

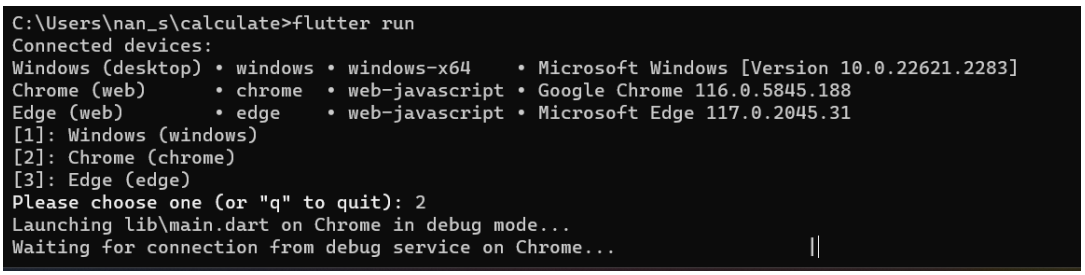
ที่มา : ณปภัช วรรณตรง (2564 : 32)

เมื่อเพิ่มปุ่มแล้ว ให้ทำการกดบันทึกไฟล์ จากนั้นให้กลับไปที่หน้าต่าง cmd แล้วทำการสั่งการทำงานของ Flutter



ภาพประกอบ 4.102 สั่งการทำงาน Flutter หลังการแก้ไขไฟล์โค้ด

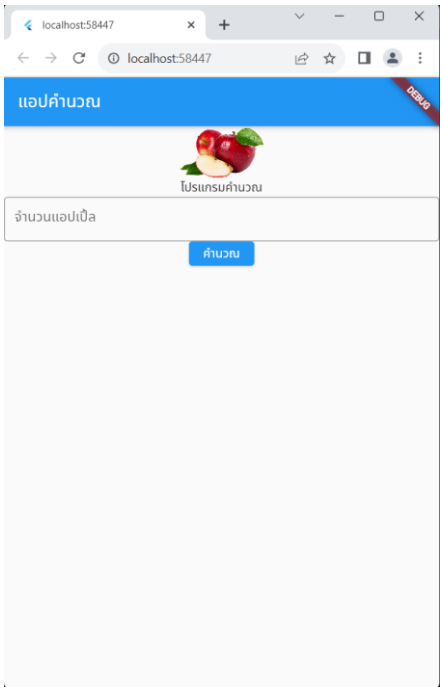
ที่มา : ณปภัช วรรณตรง (2564 : 33)



ภาพประกอบ 4.103 สั่งการทำงาน Flutter หลังการแก้ไขไฟล์โค้ด (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 33)

ผลลัพธ์หลังการแก้ไขไฟล์โค้ด

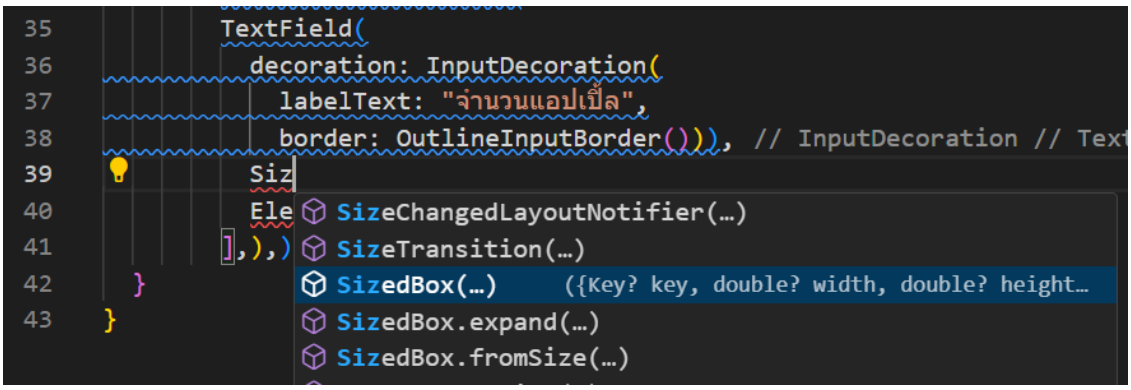


ภาพประกอบ 4.104 ผลลัพธ์หลังการแก้ไขไฟล์โค้ด
ที่มา : ณปภัช วรรณตรง (2564 : 33)

6. SizedBox

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

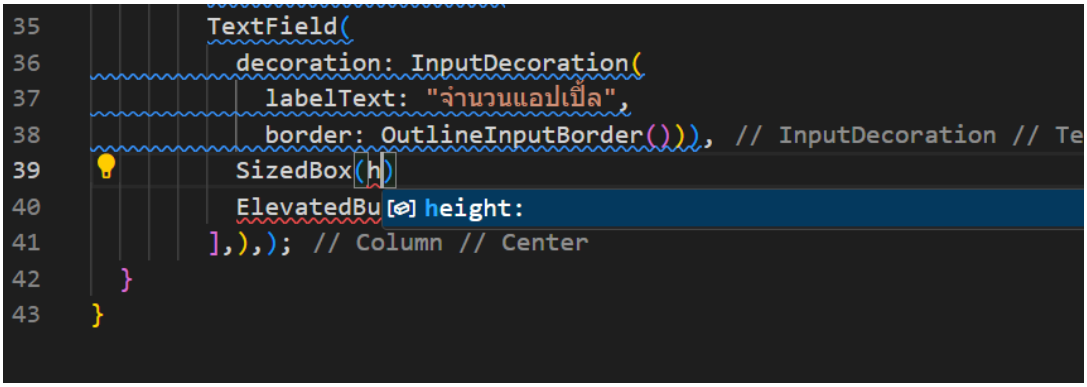
6.1 เนื่องจากกรอบการแสดงผลติดกันจนเกินไป สามารถเพิ่มโค้ด Sizebox ต่อจาก TextFeild(...) เพื่อเว้นระยะห่างระหว่างปุ่มและข้อความ โดยการพิมพ์ Siz จากนั้นจะปรากฏหน้าต่างให้ทำการเลือกคำสั่ง SizeBox(...) โดยการกด Click ที่คำสั่งหรือ Enter



ภาพประกอบ 4.105 เพิ่มกรอบการแสดงผล

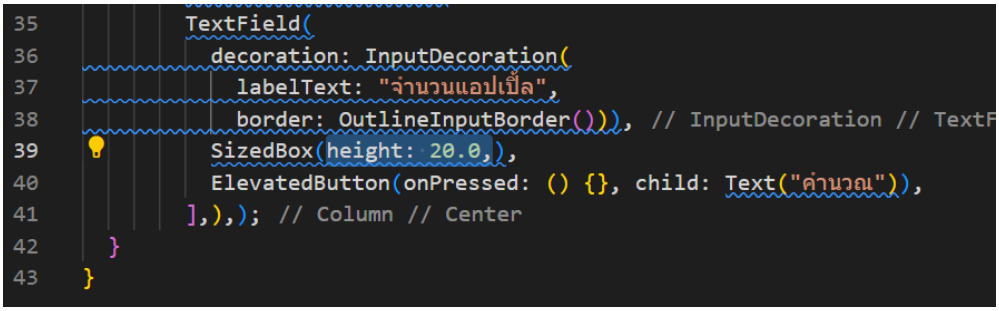
ที่มา : ณปภัช วรรณตรง (2564 : 34)

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



ภาพประกอบ 4.106 เพิ่มกรอบการแสดงผล (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 34)



ภาพประกอบ 4.107 เพิ่มกรอบการแสดงผล (ต่อ)

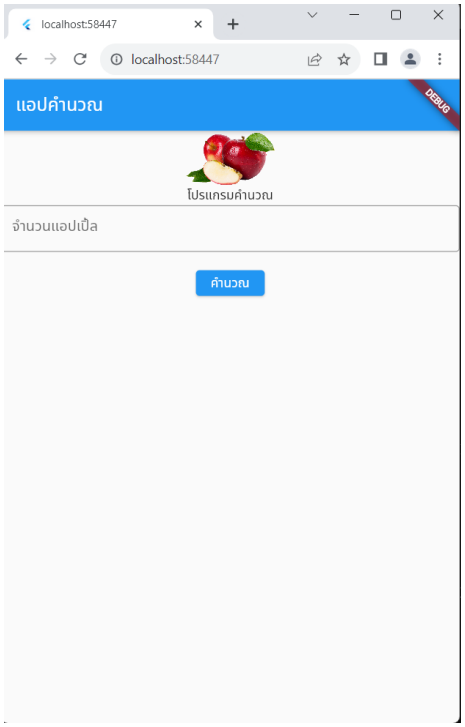
ที่มา : ณปภัช วรรณตรง (2564 : 34)

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

ภาพประกอบ 4.108 เพิ่มกรอบการแสดงผล (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 34)

6.2 หลังทำการสร้างไฟล์งานแก้ไขปรับปรุงโค้ดเรียบร้อยแล้ว จะได้ผลลัพธ์ตามภาพ



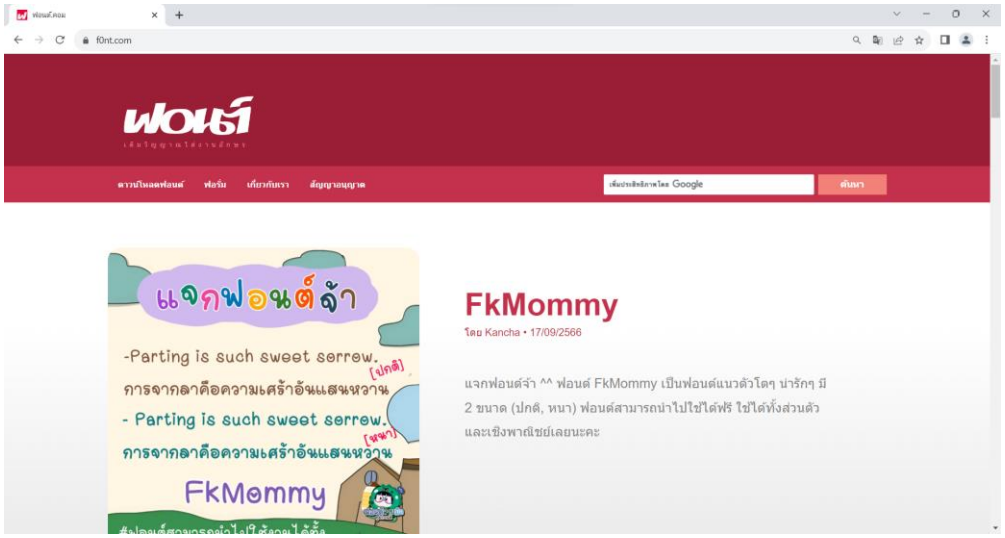
ภาพประกอบ 4.109 ผลลัพธ์ของการสร้างแอปพลิเคชัน

ที่มา : ณปภัช วรรณตรง (2564 : 35)

7. เปลี่ยนรูปแบบฟอนต์ (Font)

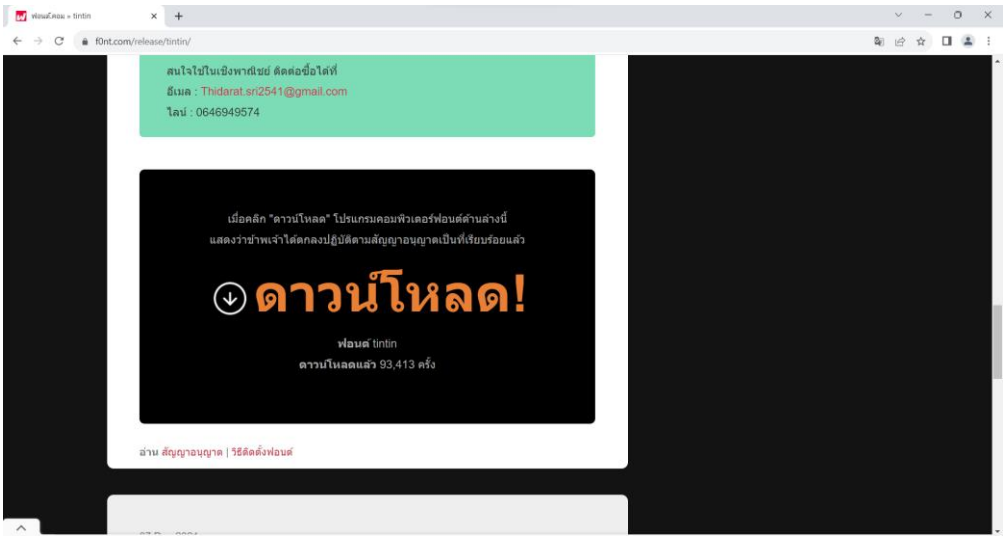
7.1 หากต้องการเปลี่ยนฟอนต์ ให้ทำการดาวน์โหลดฟอนต์มาใช้งาน ซึ่งมีวิธีการคล้ายกับการนำรูปเข้ามาใช้งาน

ให้ทำการเข้าไปดาวน์โหลดฟอนต์ที่ต้องการใช้งาน



ภาพประกอบ 4.110 ดาวน์โหลดฟอนต์ที่ต้องการ

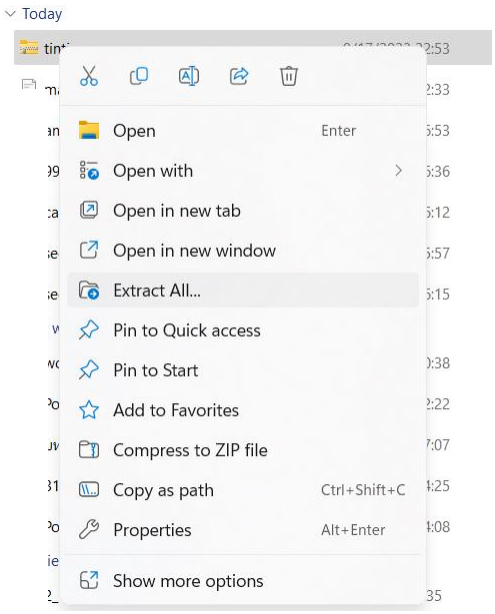
ที่มา : ณปภัช วรรณตรง (2564 : 36)



ภาพประกอบ 4.111 ดาวน์โหลดฟอนต์ที่ต้องการ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 36)

เมื่อทำการดาวน์โหลดเรียบร้อยแล้วให้ทำการแตกไฟล์ที่ได้ทำการดาวน์โหลดมา



ภาพประกอบ 4.112 แตกไฟล์ที่ได้ทำการดาวน์โหลดมา

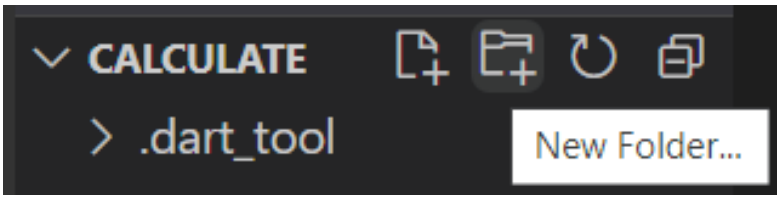
ที่มา : ณปภัช วรรณตรง (2564 : 36)

เมื่อได้ไฟล์ฟอนต์มาแล้ว ให้ทำการเปลี่ยนชื่อเพื่อให้งานต่อการนำไปใช้งาน

ภาพประกอบ 4.113 เปลี่ยนชื่อไฟล์เพื่อให้ง่ายต่อการใช้งาน

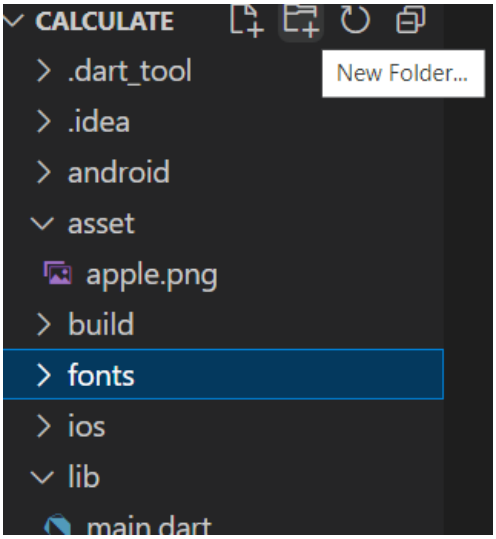
ที่มา : ณปภัช วรรณตรง (2564 : 36)

เมื่อเปลี่ยนชื่อให้ง่ายต่อการใช้งานแล้ว กลับไปที่ VSCode และทำการสร้าง Folder fonts ขึ้นมาเพื่อเก็บไฟล์ฟอนต์



ภาพประกอบ 4.114 สร้าง Folder Fonts

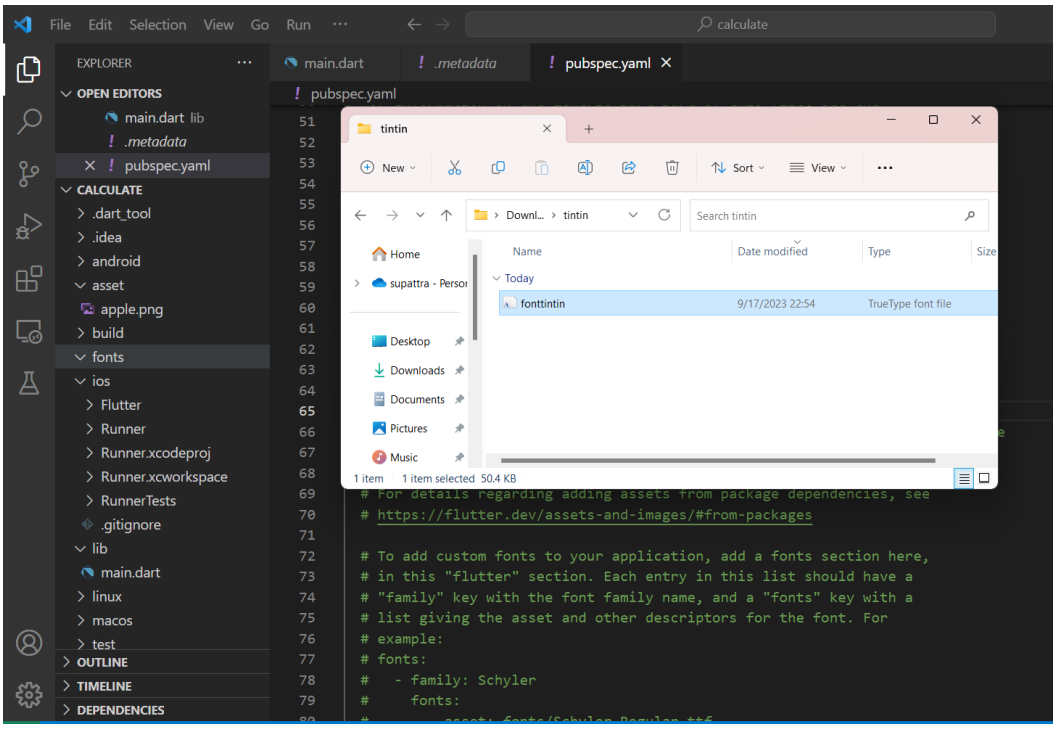
ที่มา : ณปภัช วรรณตรง (2564 : 36)



ภาพประกอบ 4.115 สร้าง Folder fonts (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 36)

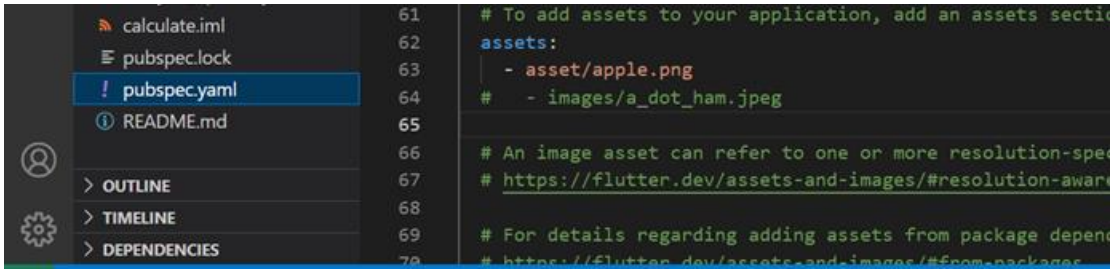
จากนั้นให้ทำการย้ายไฟล์ฟอนต์เข้าไปไว้ใน Folder fonts



ภาพประกอบ 4.116 ย้ายไฟล์ฟอนต์ไปยัง Folder fonts

ที่มา : ณปภัช วรรณตรง (2564 : 36)

เมื่อย้ายไฟล์ฟอนต์แล้ว ให้ไปที่ไฟล์ pubspec.yaml เพื่อแก้โค้ดที่นำฟอนต์เข้ามาใช้งาน



ภาพประกอบ 4.117 ไปที่ไฟล์ pubspec.yaml

ที่มา : ณปภัช วรรณตรง (2564 : 36)

ค้นหา fonts จากนั้นให้ทำปิดคอมเม้นเพื่อสั่งเปิดการใช้งานโค้ด

```
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 : ณปภัช 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 : ณปภัช วรรณตรง (2564 : 36)

เมื่อเปลี่ยนชื่อฟอนต์แล้วให้ทำการอ้างอิงไฟล์ฟอนต์ให้ตรงกับไฟล์ที่นำมาใช้งาน

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

ภาพประกอบ 4.120 อ้างอิงไฟล์ฟอนต์ให้ตรงกับไฟล์ที่นำมาใช้งาน

ที่มา : ณปภัช วรรณตรง (2564 : 36)

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

ภาพประกอบ 4.121 อ้างอิงไฟล์ฟอนต์ให้ตรงกับไฟล์ที่นำมาใช้งาน (ต่อ)
ที่มา : ณปภัช 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 //
    SizedBox(height: 20.0,),
    ElevatedButton(onPressed: () {}, child: Text("คำนวณ")),
  ],),); // Column // Center
```

```
fontFamily:
fontFamilyFallbac
fontFeatures:
fontStyle:
fontVariations:
foreground:
```

ภาพประกอบ 4.122 กำหนดฟอนต์ให้กับข้อความที่ต้องการเปลี่ยนแปลงฟอนต์
ที่มา : ณปภัช วรรณตรง (2564 : 36)

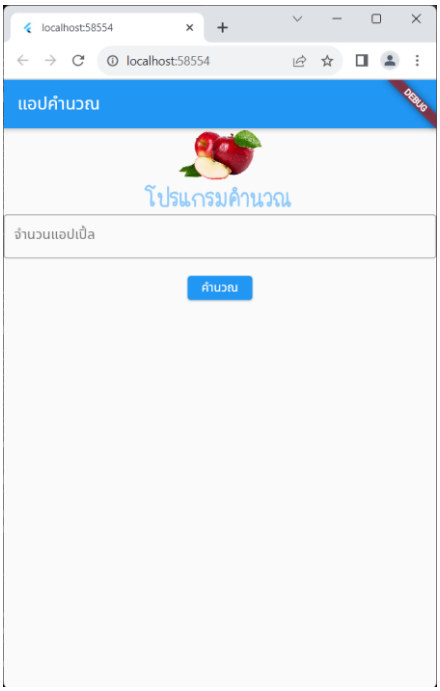
หลังจากนั้นให้ทำการกำหนดชื่อให้ตรงกับไฟล์ฟอนต์ที่นำมาใช้งาน

```
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)

ผลลัพธ์การปรังแต่งฟอนต์ของข้อความ



ภาพประกอบ 4.124 ผลลัพธ์การปรังแต่งฟอนต์ของข้อความ

ที่มา : ณปภัช วรรณตรง (2564 : 36)

ตกแต่งแอปพลิเคชันด้วย Padding

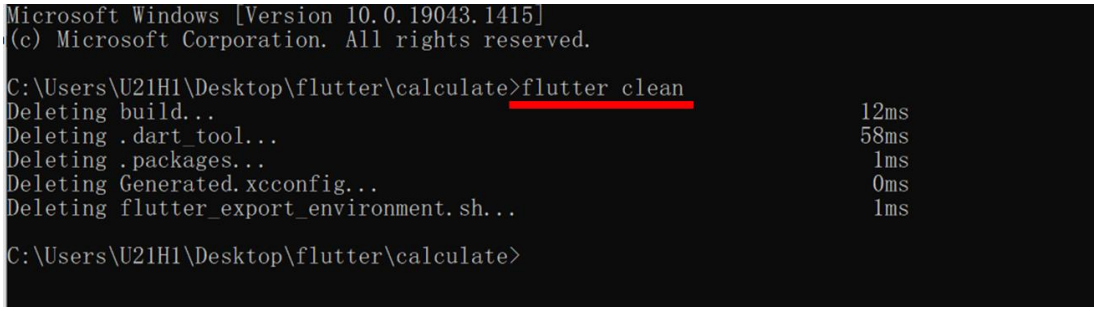
Padding เป็นวิดเจ็ตที่ทำหน้าที่สำหรับการแทรกวิดเจ็ตลูกโดยการเติมช่องว่างหรือเพิ่มระยะห่างไว้รอบ ๆ ภายในพื้นที่ที่กำหนดด้วยค่าของ EdgeInsets ซึ่งมีการใช้งานหลัก 2 แบบ ดังนี้

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

2. EdgeInsets.fromLTRB คือ การเว้นระยะโดยกำหนดระยะห่างจากขอบแบบที่เป็นค่าอิสระแยก จากกัน ค่าที่กำหนดให้เป็นพารามิเตอร์จะต้องกำหนดทั้งหมด 4 ค่า โดยเรียงลำดับจากขอบด้านซ้าย บน ล่าง และขวา ตามลำดับ ตัวอย่างการใช้งานเช่น EdgeInsets fromLTRB (10, 5, 7, 0) จะหมายถึงการกำหนดค่าเว้นระยะขอบด้านซ้าย 10 px, ด้านบน 5 px, ด้านขวา 7 px และ ด้านล่าง 0 px เป็นต้น (เอกรินทร์ วทัญญูเลิศสกุล, 2563 : 91)

ในส่วนนี้จะทำการตกแต่งขอบของแอปพลิเคชันด้วย Padding

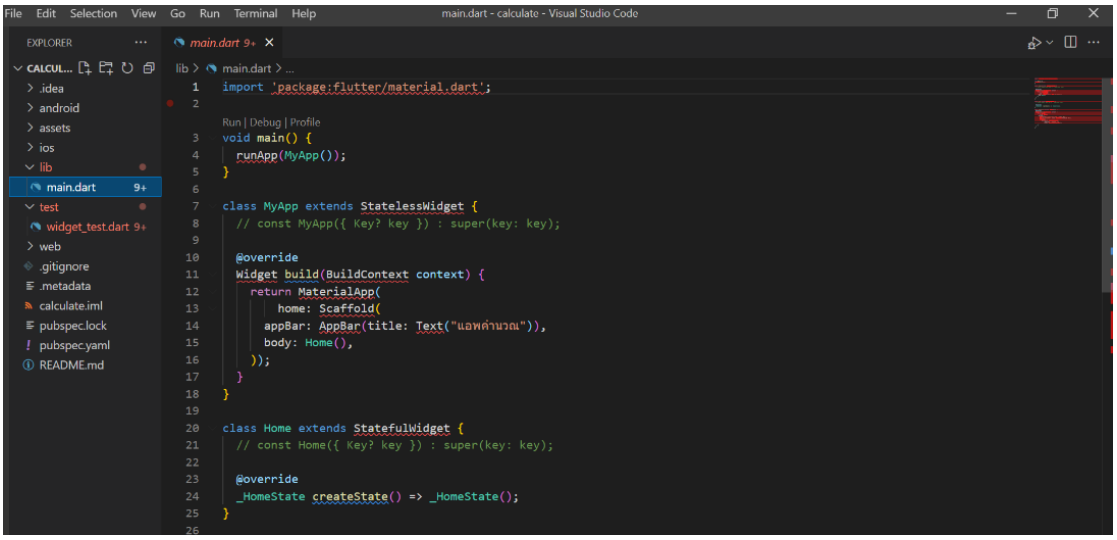
1. หากต้องการส่งให้ผู้ใช้คนอื่นที่ต้องการนำโค้ดไปพัฒนาต่อ ให้ทำการ cmd > cd [ไฟล์ Project] แล้วใช้คำสั่ง flutter clean เพื่อลดขนาดและลบไฟล์ที่ไม่จำเป็นออกจาก Folder งานก่อนทำการ ZIP ไฟล์แล้วส่งให้ผู้ใช้หรือผู้พัฒนาคนอื่น ๆ นำไปพัฒนาต่อไป



ภาพประกอบ 4.125 Flutter Clean

ที่มา : ณปภัช วรรณตรง (2564 : 3)

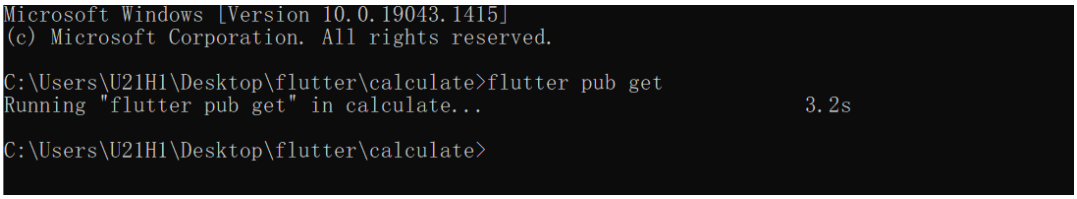
2. เมื่อได้ไฟล์มาแล้วให้ทำการแยกไฟล์ และเปิดไฟล์ในโปรแกรม VSCode หากพบ Error ให้ทำการแก้ไขภายหลัง



ภาพประกอบ 4.126 เปิดไฟล์ในโปรแกรม VSCode

ที่มา : ณปภัช วรรณตรง (2564 : 4)

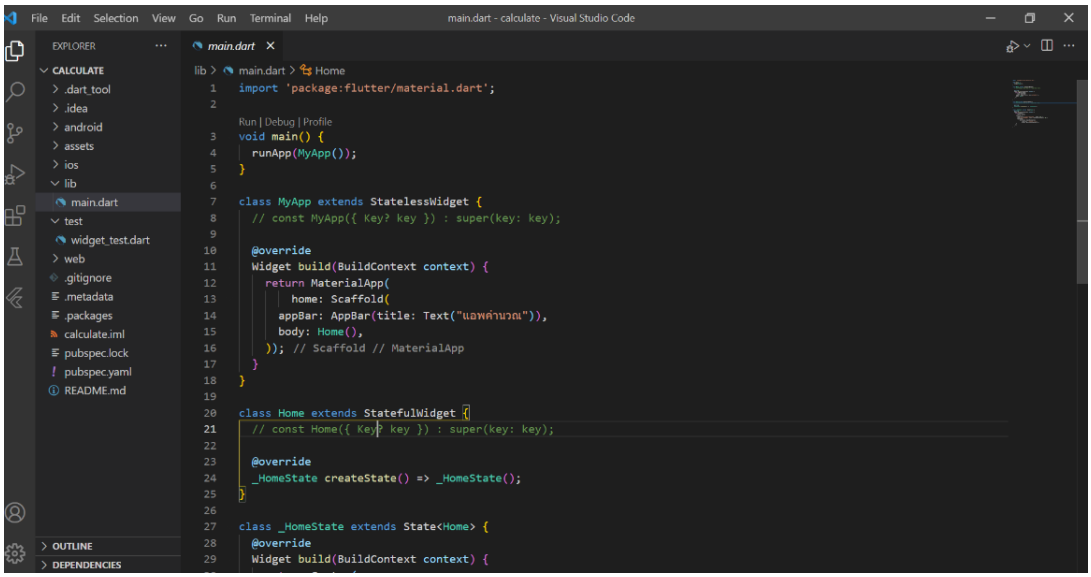
3. หากได้ไฟล์งานมาจากผู้พัฒนาคนอื่นแล้วพบ Error ให้ทำการแก้ไขโดยการเข้าไปยังโฟลเดอร์ไฟล์งาน Project ของผู้พัฒนาผ่าน cmd จากนั้นให้ใช้คำสั่ง flutter pub get เพื่อทำการแก้ไขข้อผิดพลาดของไฟล์งานที่ทำให้เกิด Error ตามภาพ



ภาพประกอบ 4.127 ใช้คำสั่ง flutter pub get แก้ไข Error

ที่มา : ณปภัช วรรณตรง (2564 : 5)

4. หลังจากทำการแก้ไขข้อผิดพลาดของไฟล์งานแล้ว ส่วนที่ Error จะหายไป ให้ทำการเปิดไฟล์งานขึ้นมาและสั่งการทำงานผ่าน flutter run



ภาพประกอบ 4.128 เปิดไฟล์งานและสั่งการทำงาน
ที่มา : ณปภัช วรรณตรง (2564 : 6)

1. TextField

เท็กซ์ฟิลด์ (TextField) เป็นกล่องสำหรับรับข้อความที่ผู้ใช้งานแอพพลิเคชันจะกรอกเพื่อใช้เป็นข้อมูลรับเข้าของระบบ หรือบางครั้งสามารถใช้เพื่อการแสดงผลได้ด้วยเช่นกัน โดยการใช้งานเท็กซ์ฟิลด์ จะต้องทำการตกแต่งให้มีความสวยงามและเหมาะต่อการแสดงผล จึงจำเป็นต้องใช้การกำหนดค่าต่าง ๆ เพื่อให้เท็กซ์ฟิลด์มีความน่าใช้งาน (เอกรินทร์ วทัญญูเลิศสกุล, 2563 : 110)

1.1 ทำการเขียนโค้ดเพิ่มคำอธิบายใต้ปุ่มของโค้ดที่ได้มา

ให้ทำการเปิดไฟล์ main.dart ขึ้นมาจากนั้นให้เลื่อนหาไปที่ 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 // TextFi
  SizedBox(height: 20.0,),
  ElevatedButton(onPressed: () {}, child: Text("คำนวณ")),
  Text(""),
],),); // Column // Center
```

ภาพประกอบ 4.130 เพิ่มโค้ดคำอธิบาย (ต่อ)

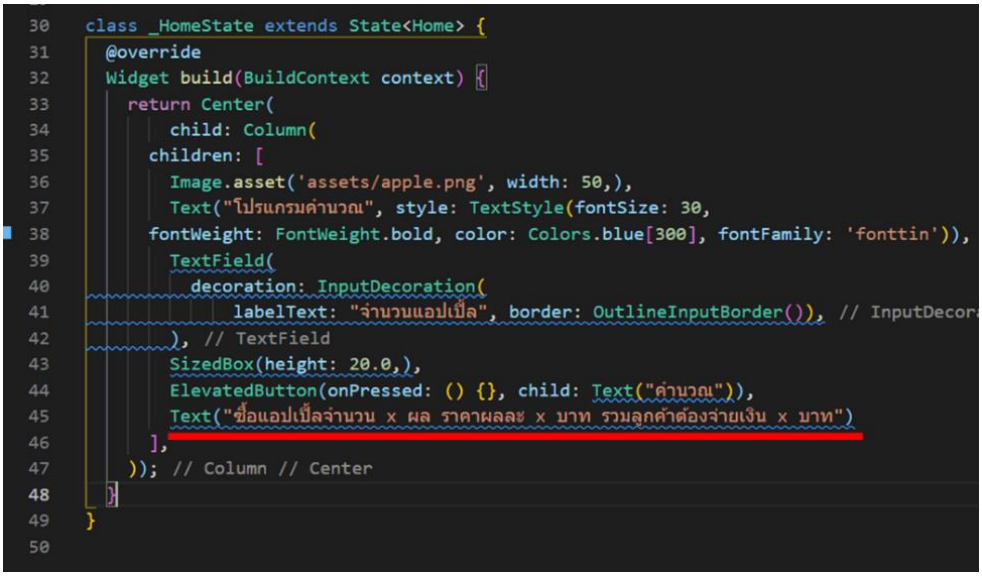
ที่มา : ณปภัช วรรณตรง (2564 : 7)

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("ซื้อแอปเปิ้ล"),
],),); // Column // Center
```

ภาพประกอบ 4.131 เพิ่มโค้ดคำอธิบาย (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 7)



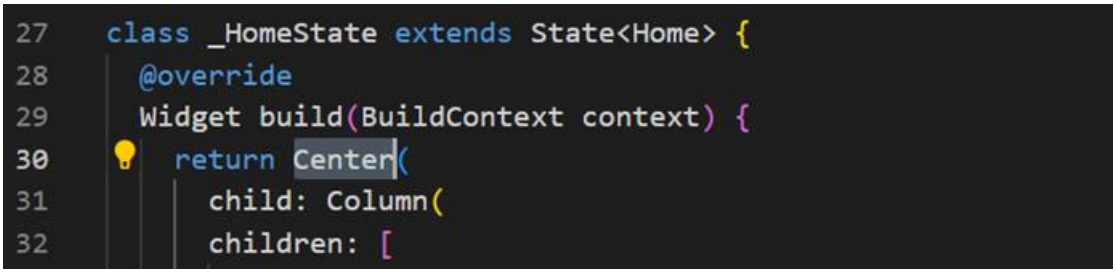
```
30  class _HomeState extends State<Home> {
31    @override
32    Widget build(BuildContext context) {
33      return Center(
34          child: Column(
35        children: [
36          Image.asset('assets/apple.png', width: 50,),
37          Text("โปรแกรมคำนวณ", style: TextStyle(fontSize: 30,
38        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("ซื้อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท")
46        ],
47      )); // Column // Center
48    }
49  }
50
```

ภาพประกอบ 4.132 เพิ่มโค้ดคำอธิบาย (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 7)

1.2 สามารถทำการตกแต่งหน้าต่างของแอปพลิเคชันด้วย Padding ได้โดยการคลิกเลือกที่ Center จากนั้นจะปรากฏวงไฟสีเหลืองอยู่ทางด้านซ้าย ให้คลิกที่ดวงไฟ จากนั้นจะขึ้นชุดคำสั่ง ให้คลิกเลือกที่ชุดคำสั่ง wrap with padding

1.2.1 ให้ทำการไปที่ Center(...) จากนั้นเมื่อกดคลิกที่คำสั่งจะปรากฏหลอดไฟสีเหลืองขึ้นมาข้างหน้าบรรทัดคำสั่ง

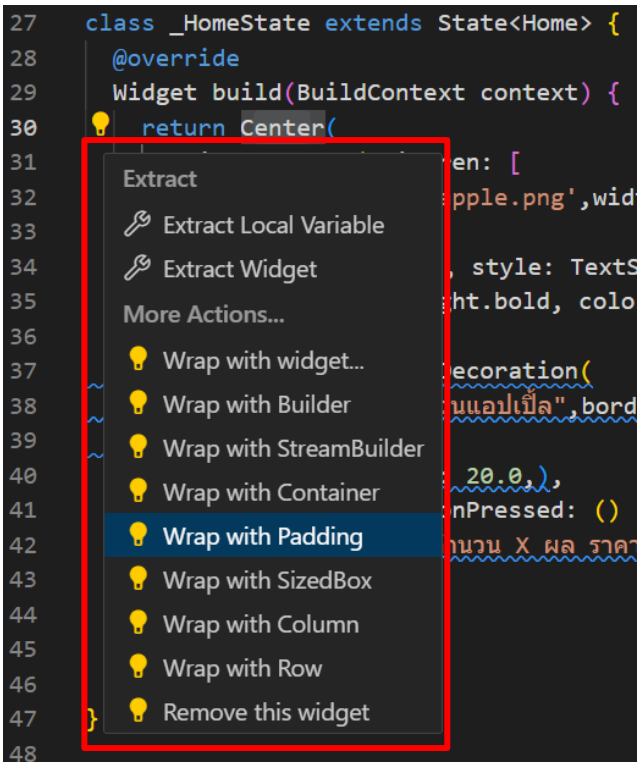


```
27  class _HomeState extends State<Home> {
28    @override
29    Widget build(BuildContext context) {
30      return Center(
31        child: Column(
32        children: [
```

ภาพประกอบ 4.133 ตกแต่งหน้าต่างของแอปพลิเคชันด้วย Padding

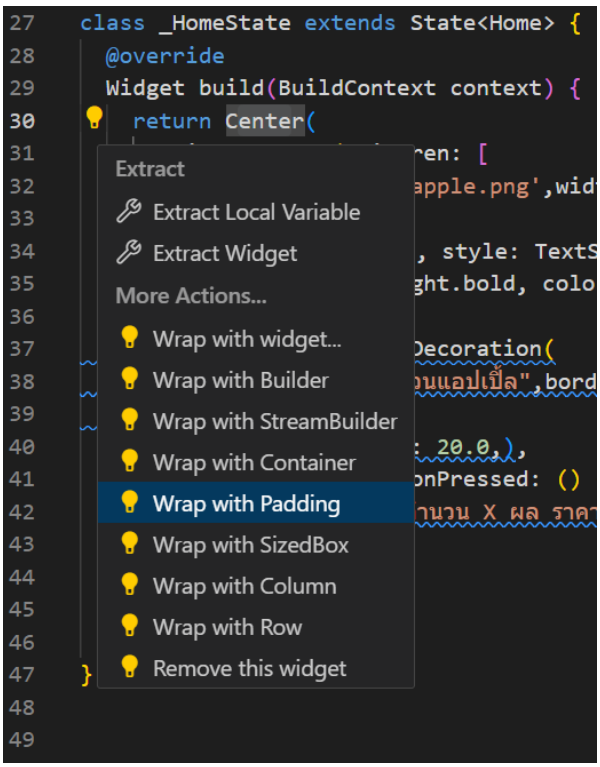
ที่มา : ณปภัช วรรณตรง (2564 : 8)

1.2.2 จากนั้นให้ทำการคลิกที่หลอดไฟ จะปรากฏหน้าต่างชุดคำสั่งขึ้นมา ให้เลือกไปที่ Warp with Padding



ภาพประกอบ 4.134 ตกแต่งหน้าต่างของแอปพลิเคชันด้วย Padding (ต่อ)

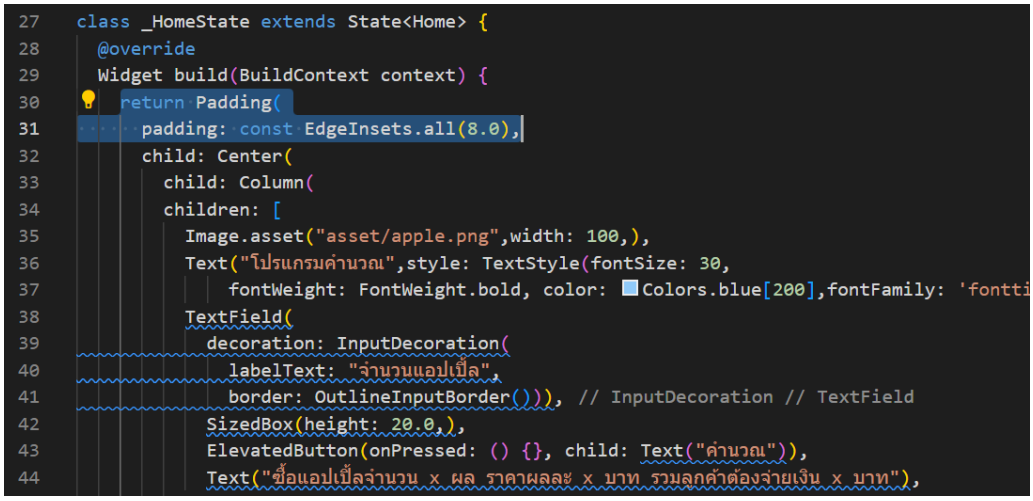
ที่มา : ณปภัช วรรณตรง (2564 : 8)



ภาพประกอบ 4.135 ตกแต่งหน้าต่างของแอปพลิเคชันด้วย Padding (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 8)

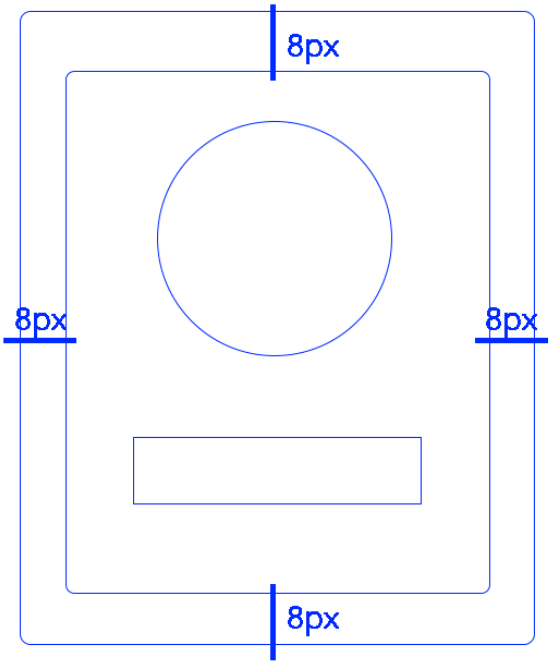
1.2.3 เมื่อเลือกที่ Warp with Padding แล้ว จะเพิ่มโค้ด Padding(...) มาครอบโค้ด Center(...) และมีโค้ด padding: เพิ่มเข้ามาใน Padding(...) ก่อนชุดคำสั่ง Center(...)



ภาพประกอบ 4.136 ตกแต่งหน้าต่างของแอปพลิเคชันด้วย Padding (ต่อ)

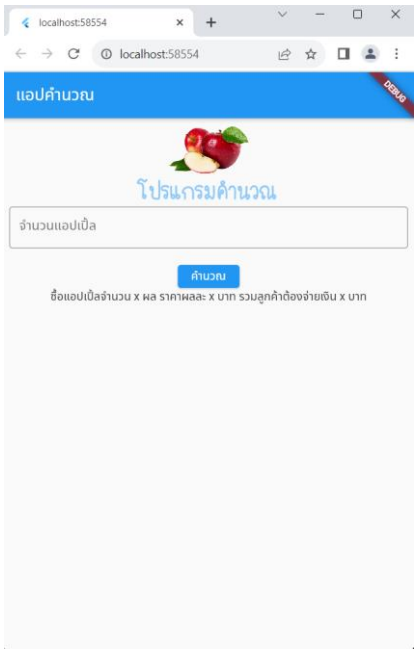
ที่มา : ณปภัช วรรณตรง (2564 : 8)

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



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

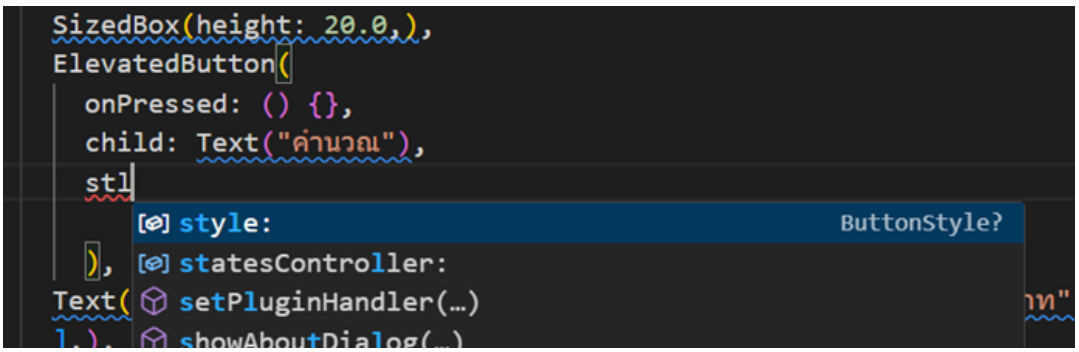
1.4 ให้ทำการใช้คำสั่ง flutter run เพื่อดูผลลัพธ์ ซึ่งจะได้ผลลัพธ์ตามภาพ



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

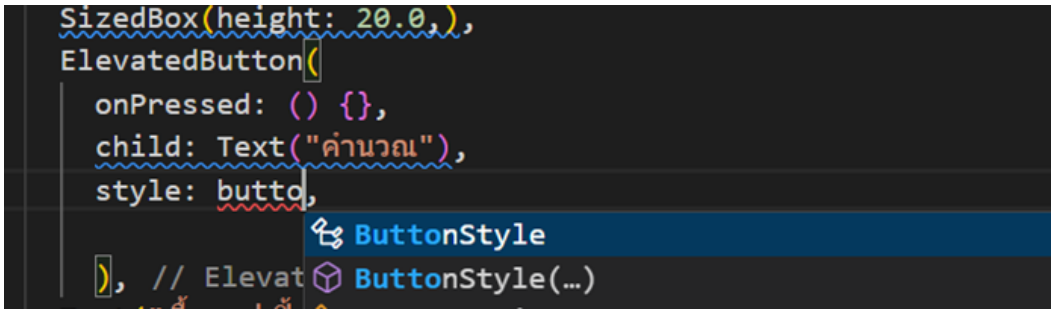
1.5 ตกแต่งสีของปุ่มด้วยการใช้โค้ด style: ButtonStyle

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



ภาพประกอบ 4.139 กำหนด Style ของปุ่ม
ที่มา : ณปภัช วรรณตรง (2564 : 11)

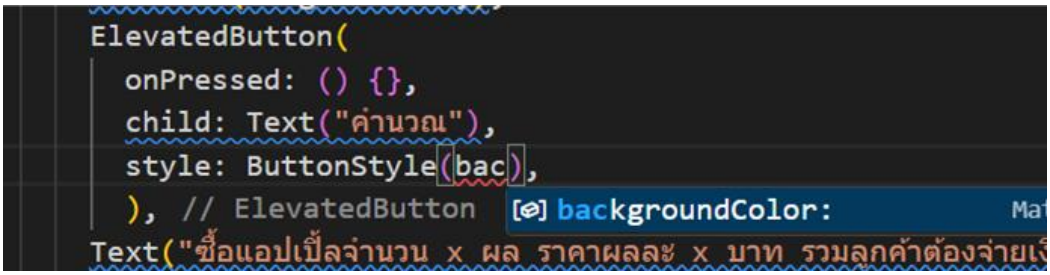
1.5.2 เพิ่ม ButttonStyle(...) เข้าไปใน style:



ภาพประกอบ 4.140 กำหนด Style ของปุ่ม (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 11)

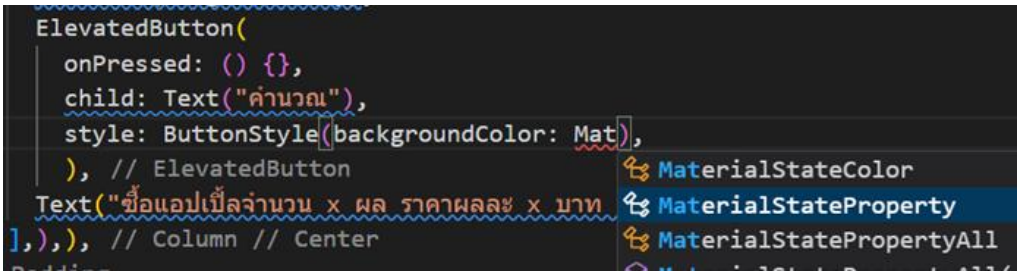
1.5.3 ภายใน Buttun(...) ให้ทำการเพิ่มโค้ด backgroundColor:



ภาพประกอบ 4.141 กำหนด Style ของปุ่ม (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 11)

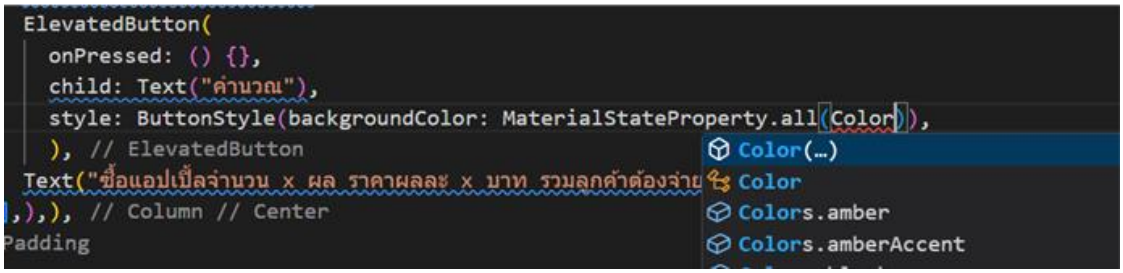
1.5.4 จากนั้นเพิ่ม MaterialStateProperty.all(...) เข้าไปต่อจาก backgroundColor:



ภาพประกอบ 4.142 กำหนด Style ของปุ่ม (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 11)

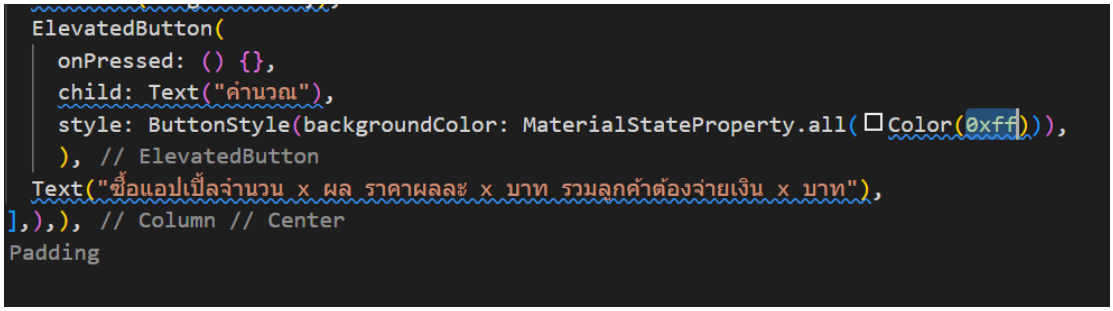
1.5.5 ภายใน all(...) ให้ทำการเพิ่มโค้ด Color(...) เข้าไปเพื่อกำหนดสีของปุ่ม



ภาพประกอบ 4.143 กำหนด Style ของปุ่ม (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 11)

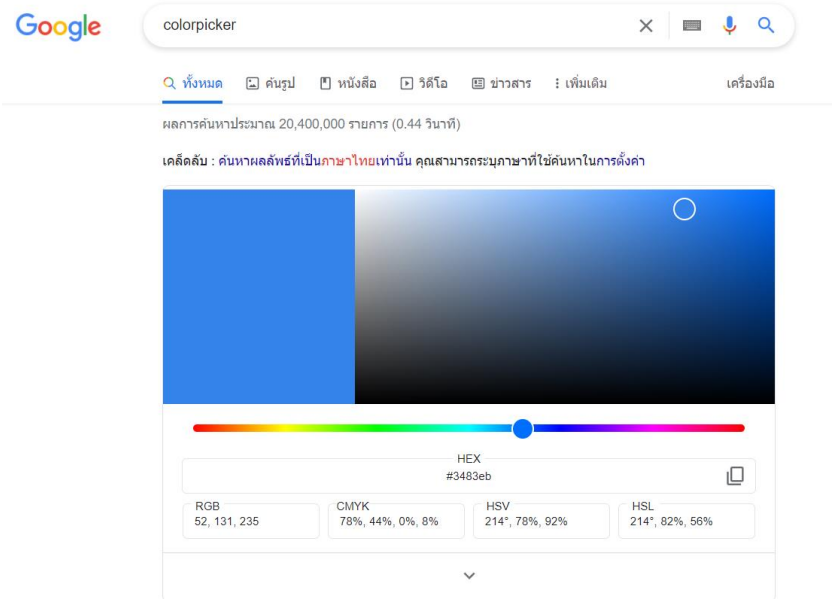
1.5.6 ก่อนที่จะเพิ่มรหัสสีเข้าไปใน Color(...) ให้ทำการเพิ่มโค้ด 0xff ก่อนจะเพิ่มรหัสสีเข้าไป



ภาพประกอบ 4.144 กำหนด Style ของปุ่ม (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 11)

1.6 สามารถเลือกใช้โค้ดสีในการตกแต่งได้ตามใจชอบ ซึ่งสามารถนำโค้ดสีมาได้จาก ColorPicker โดยใช้รหัสสีเป็นประเภท HEX



ภาพประกอบ 4.145 เลือกใช้โค้ดสี

ที่มา : ณปภัช วรรณตรง (2564 : 12)

เมื่อได้รหัสสีที่ต้องการแล้ว ในนำรหัสมาใส่ต่อจาก 0xff ที่ได้ทำการกำหนดไว้ก่อนหน้า

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

ภาพประกอบ 4.146 กำหนด Style ของปุ่ม (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 12)

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

ภาพประกอบ 4.147 กำหนด Style ของปุ่ม (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 12)

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

ภาพประกอบ 4.148 กำหนด Style ของปุ่ม (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 12)

1.7 เมื่อทำการกำหนดสีให้กับปุ่มได้แล้ว จากนั้นให้ทำการจัดระยะของปุ่มด้วย Padding

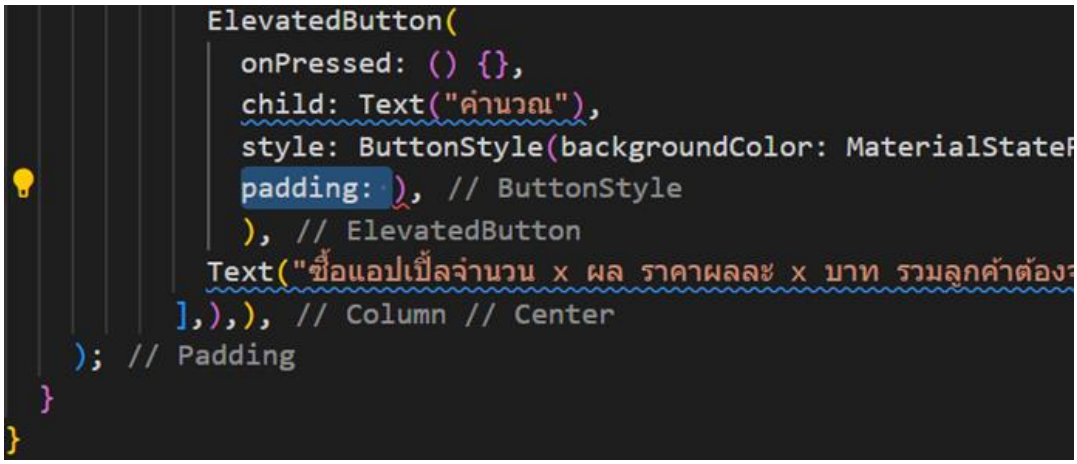
1.7.1 ภายใน ButtonStyle(...) ให้ทำการเพิ่มโค้ด padding: โดยการพิมพ์ , (Comma) เพื่อเป็นตัวคั่นระหว่างโค้ดต่อจาก backgroundColor: จากนั้นพิมพ์ padding:

```
ElevatedButton(
  onPressed: () {},
  child: Text("คำนวณ"),
  style: ButtonStyle(backgroundColor: MaterialStat
  pad
  ), padding:          MaterialStateProperty<EdgeInsets
  ), // ElevatedButton
Text("ซื้อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้อ
],),), // Column // Center
```

ภาพประกอบ 4.149 จัดระยะของปุ่มด้วย Padding

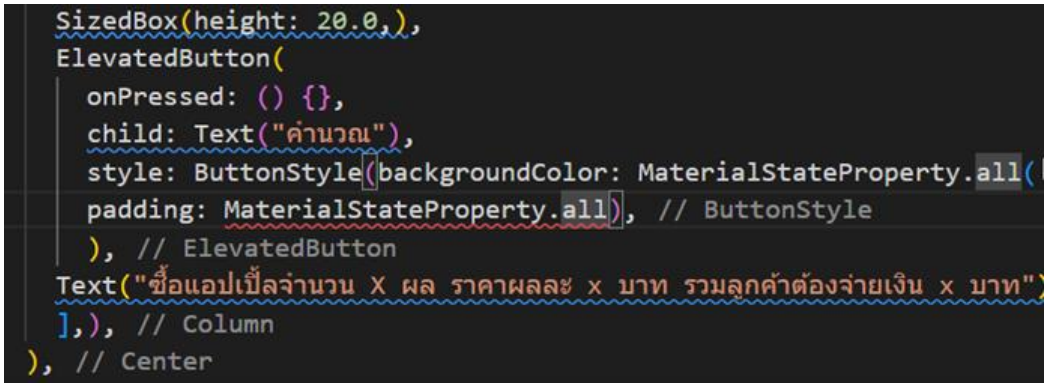
ที่มา : ณปภัช วรรณตรง (2564 : 13)

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



ภาพประกอบ 4.150 จัดระยะของปุ่มด้วย Padding (ต่อ)

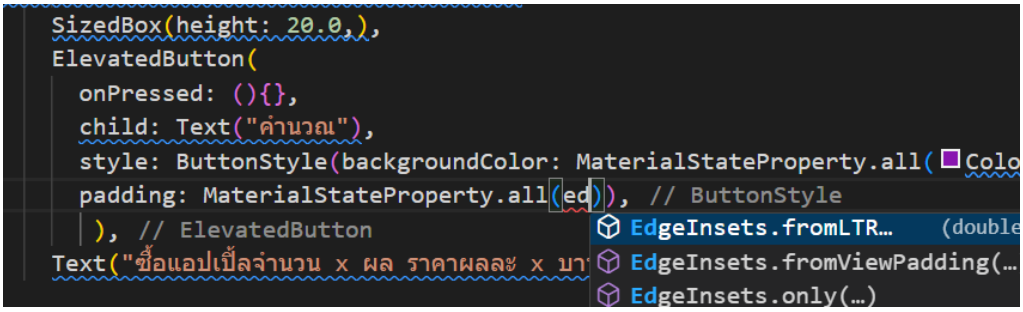
ที่มา : ณปภัช วรรณตรง (2564 : 13)



ภาพประกอบ 4.151 จัดระยะของปุ่มด้วย Padding (ต่อ)

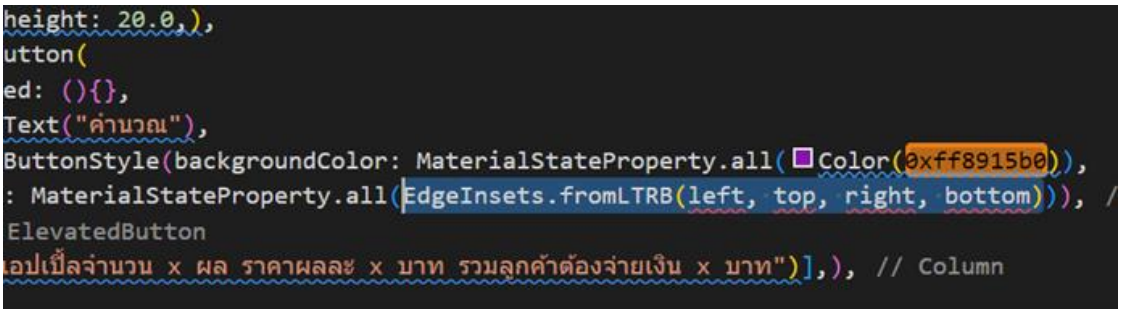
ที่มา : ณปภัช วรรณตรง (2564 : 13)

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



ภาพประกอบ 4.152 จัดระยะของปุ่มด้วย Padding (ต่อ)

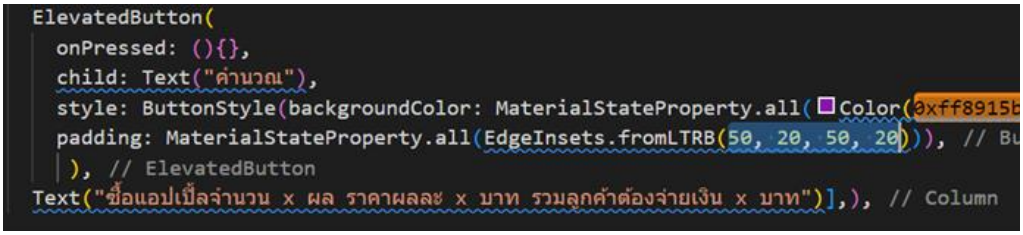
ที่มา : ณปภัช วรรณตรง (2564 : 13)



ภาพประกอบ 4.153 จัดระยะของปุ่มด้วย Padding (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 13)

1.7.4 จากนั้นให้ทำการกำหนดระยะห่างของปุ่ม โดยทำการกำหนดใน fromLTRB(...) หน่วยเป็น Pixel



ภาพประกอบ 4.154 จัดระยะของปุ่มด้วย Padding (ต่อ)

ที่มา : ณปภัช 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(
                "ซื้อแอปเปิ้ลจำนวน x ผล ราคาผลละ x บาท รวมลูกค้าต้องจ่ายเงิน x บาท") // Text
          ],
        )), // Column // Center
      ); // Padding
    }
}
```

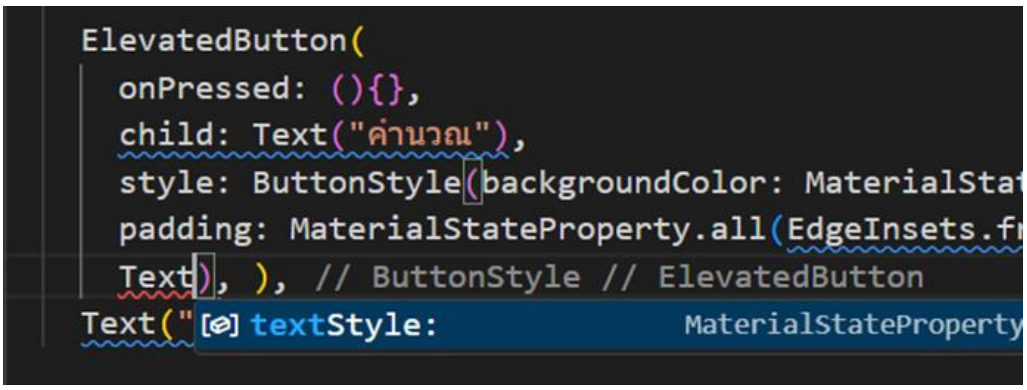
ภาพประกอบ 4.155 จัดระยะของปุ่มด้วย Padding (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 13)

หมายเหตุ
สังเกตวงเล็บของโค้ดให้ดี การจัดระยะด้วย Padding ต้องอยู่ภายในโค้ดของ ButtonStyle

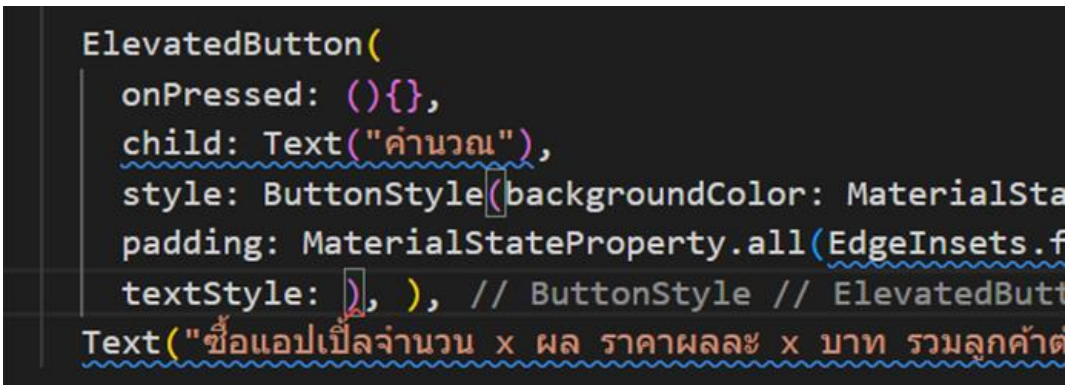
1.8 เมื่อทำการเพิ่มโค้ด padding: ได้แล้ว ให้ทำการเพิ่มโค้ด textStyle: ให้กับปุ่ม โดยทำตามขั้นตอนการเพิ่ม padding:

1.8.1 ทำการเพิ่มโค้ด textStyle: โดยการพิมพ์ text จากนั้นจะปรากฏหน้าต่าง ชุดคำสั่งขึ้นมา ให้ทำการเลือกไปที่ textStyle: โดยการกดคลิกหรือกด Enter



ภาพประกอบ 4.156 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ)

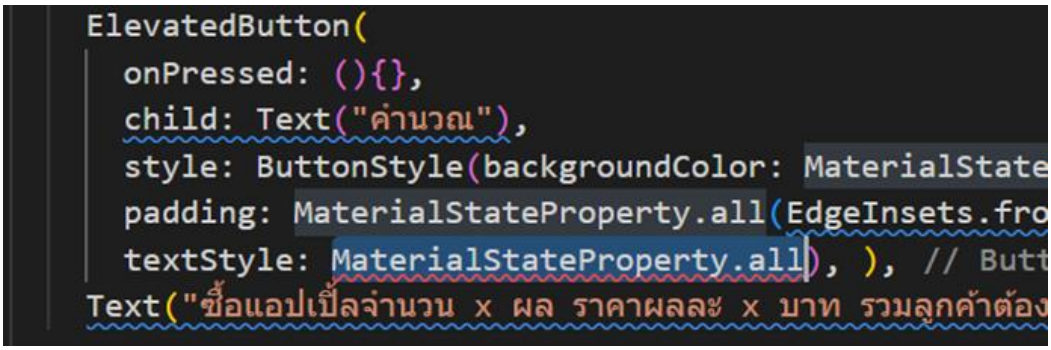
ที่มา : ณปภัช วรรณตรง (2564 : 14)



ภาพประกอบ 4.157 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 14)

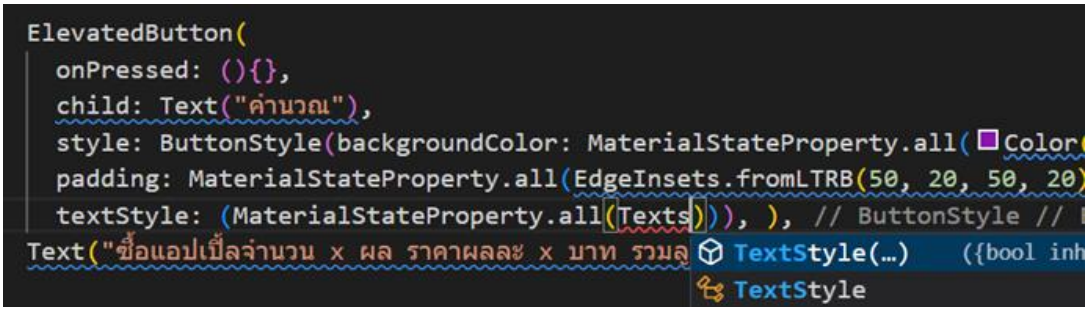
1.8.2 จากนั้นให้ทำการเพิ่ม MaterialStateProperty.all(...) โดยการพิมพ์หรือคัดลอกโค้ดก่อนหน้า เหมือนขั้นตอนของการเพิ่ม padding:



ภาพประกอบ 4.158 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 14)

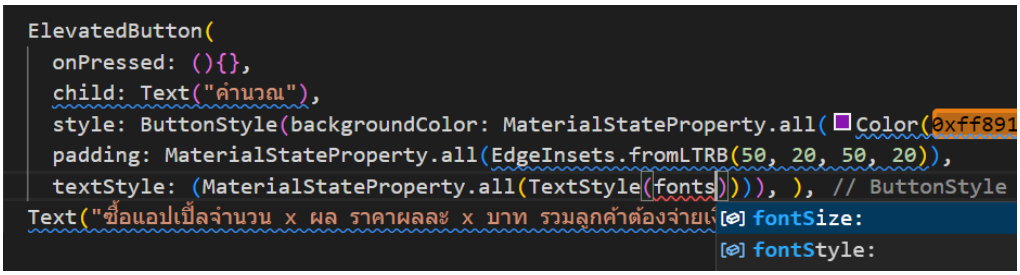
1.8.3 ภายใน all(...) ให้ทำการใส่โค้ด TextStyle(...) เข้าไป



ภาพประกอบ 4.159 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 14)

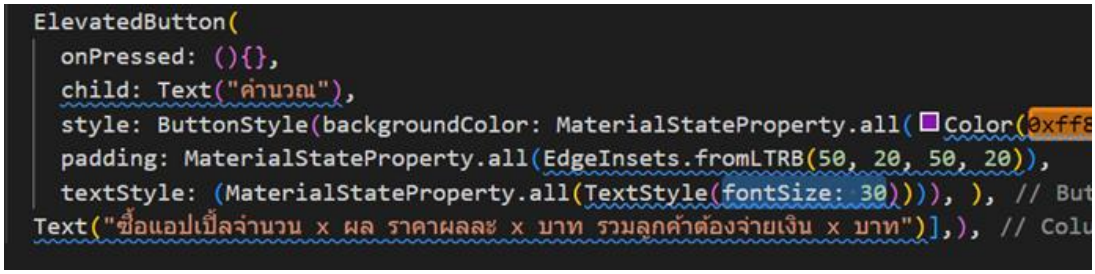
1.8.4 ภายใน TextStyle(...) ให้ทำการเพิ่ม fontSize: เข้าไป



ภาพประกอบ 4.160 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ)

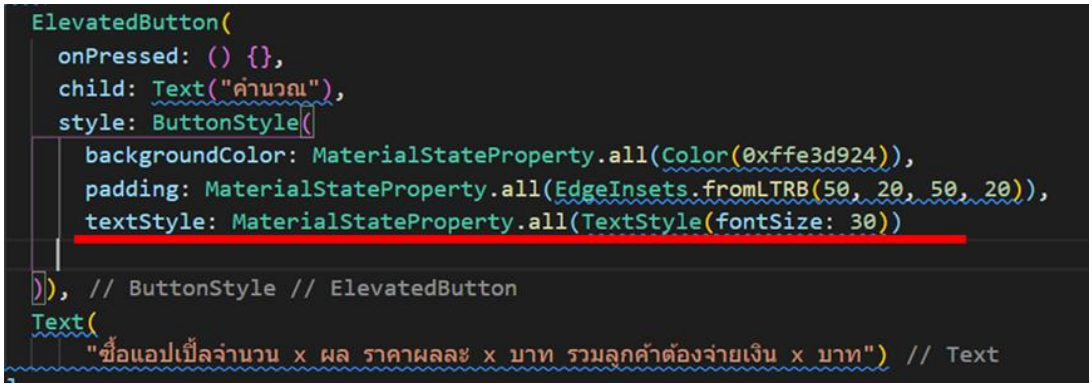
ที่มา : ณปภัช วรรณตรง (2564 : 14)

1.8.5 สามารถกำหนดขนาดของ Font ได้โดยมีหน่วยเป็น pt (Point)



ภาพประกอบ 4.161 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 14)

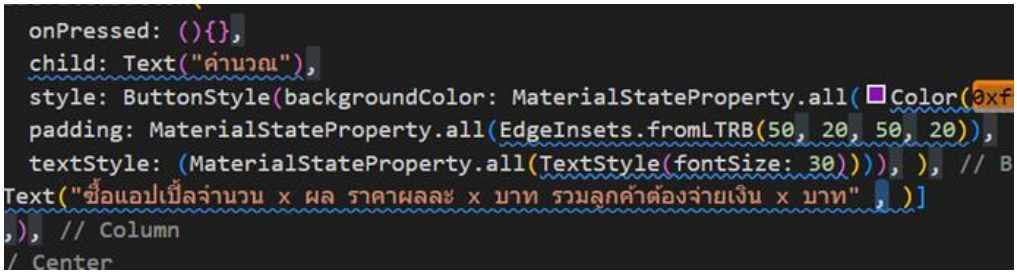


ภาพประกอบ 4.162 เพิ่มโค้ด textStyle ให้กับปุ่ม (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 14)

1.9 เมื่อทำการแก้ไขโค้ดให้กับปุ่มแล้ว จากนั้นให้ทำการเพิ่มโค้ด textStyle ให้กับคำอธิบาย

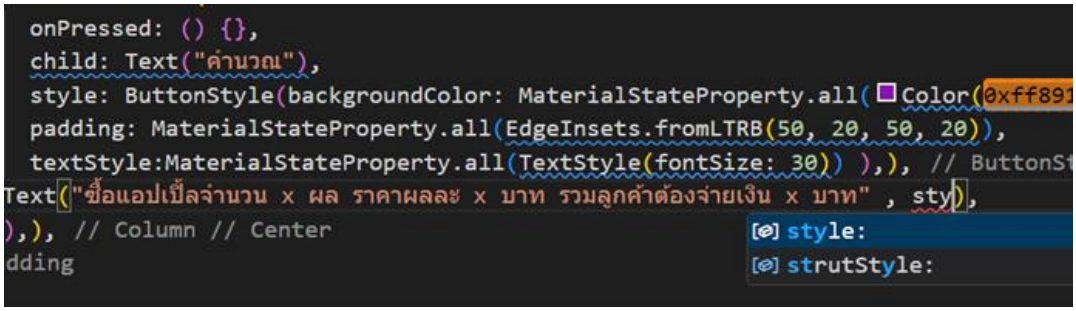
ภายใน Text(...) ต่อจากเครื่องหมาย “ ” ให้คั่นด้วย , เพื่อทำการเพิ่มโค้ดเข้าไป



ภาพประกอบ 4.163 เพิ่มโค้ด textStyle ให้กับคำอธิบาย

ที่มา : ณปภัช วรรณตรง (2564 : 15)

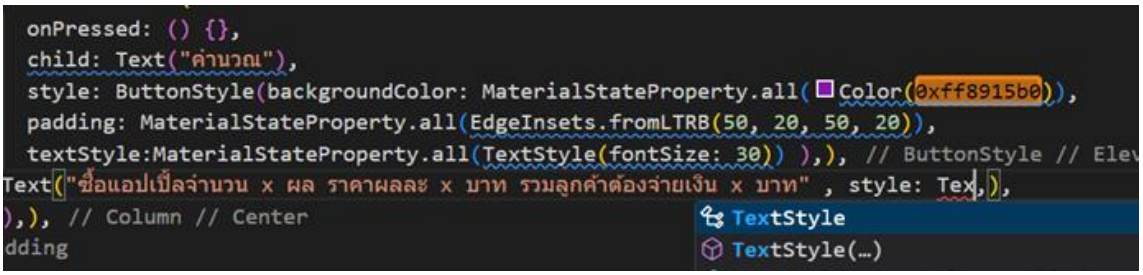
พิมพ์โค้ด styte:



ภาพประกอบ 4.164 เพิ่มโค้ด textStyle ให้กับคำอธิบาย (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 15)

จากนั้นเพิ่มโค้ด TextStyle(...) ต่อจาก style:



ภาพประกอบ 4.165 เพิ่มโค้ด textStyle ให้กับคำอธิบาย (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 15)

ภายใน TextStyle(...) ให้ทำการเพิ่มโค้ด fontSize: เข้าไป

ภาพประกอบ 4.166 เพิ่มโค้ด textStyle ให้กับคำอธิบาย (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 15)

สามารถกำหนดขนาดของ Font ได้โดยมีหน่วยเป็น pt (Point) ได้เช่นเดียวกัน

ภาพประกอบ 4.167 เพิ่มโค้ด textStyle ให้กับคำอธิบาย (ต่อ)

ที่มา : ณปภัช 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(
      "ซื้อแอปเปิ้ลจำนวน 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 แต่เมื่อย่อขนาดของหน้าแอปพลิเคชันจะพบปัญหาตามภาพ

ภาพประกอบ 4.169 ผลลัพท์สั่งการทำงานของ Flutter แล้วมีข้อผิดพลาด

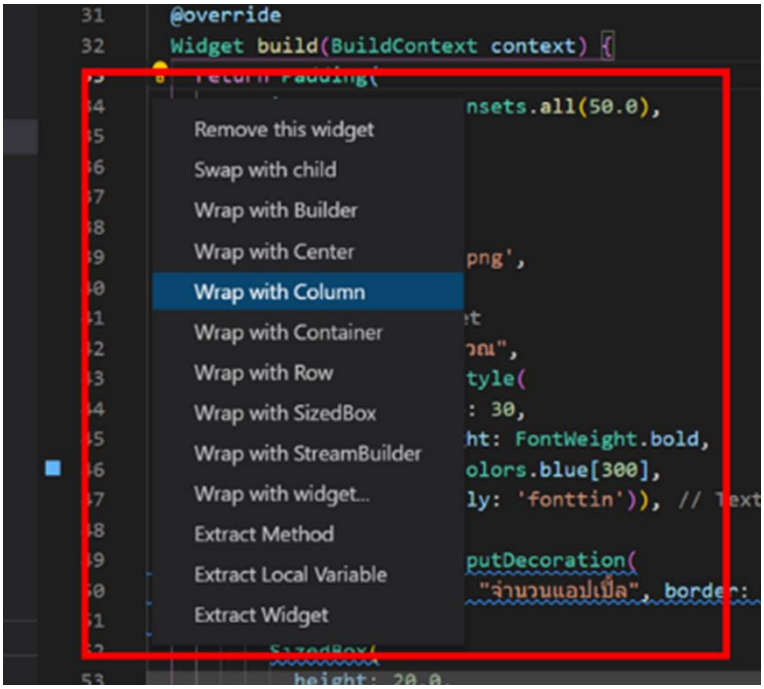
ที่มา : ณปภัช วรรณตรง (2564 : 16)

2. ListView Widget

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

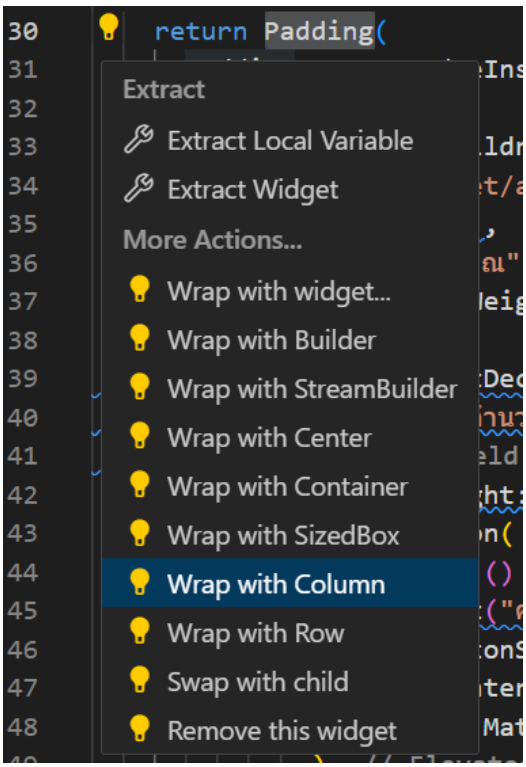
2.1 ให้ทำการแก้ไข Scroll โดยการใช้ชุดคำสั่งของ ListView ซึ่งใช้วิธีเดียวกันกับขั้นตอนที่ 6 แต่ให้เลือกเป็น Warp with Column

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



ภาพประกอบ 4.170 แก้ไขการ Scroll โดยการใช้ชุดคำสั่งของ ListView

ที่มา : ณปภัช วรรณตรง (2564 : 17)



ภาพประกอบ 4.171 แก้ไขการ Scroll โดยการใช้ชุดคำสั่งของ ListView (ต่อ)
ที่มา : ณปภัช 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 โดยการใช้ชุดคำสั่งของ ListView (ต่อ)
ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : 18)

```
27  class _HomeState extends State<Home> {
28    @override
29    Widget build(BuildContext context) {
30      return Listv(
31        children:  ListView(...)      ({Key? key, Axis scrollDire
32          Padding(  ListView.builder(...)
33            paddin  ListView.custom(...)
34            child:  ListView.separated(...)
```

ภาพประกอบ 4.174 เปลี่ยนจาก Column เป็น ListView (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 18)

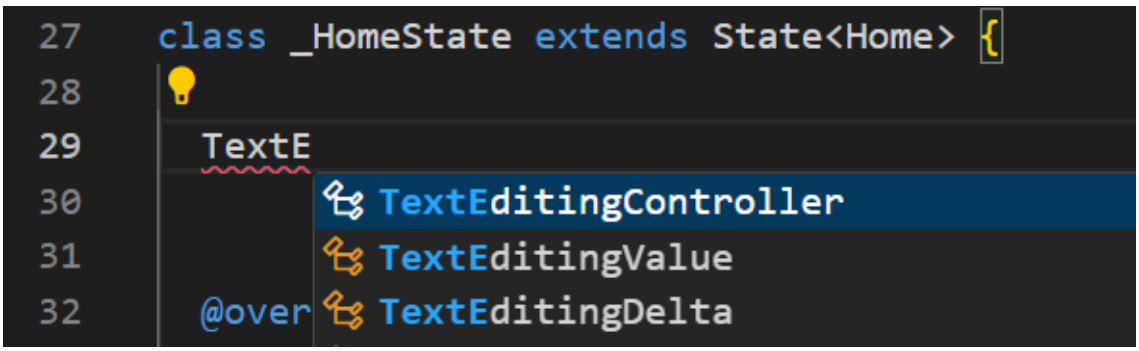
2.3 ใช้คำสั่ง flutter run แล้วผลลัพธ์ที่ได้จากการแก้ไขการ Scroll จะได้ตามภาพ

ภาพประกอบ 4.175 ผลลัพธ์ของแอปพลิเคชัน
ที่มา : ณปภัช 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 จากนั้นจะปรากฏหน้าต่างชุดโค้ดขึ้นมาให้ทำการเลือก TextEditingController โดยการคลิกหรือกด Enter เพื่อเป็นการประกาศประเภทของข้อมูลที่ได้รับเข้ามา

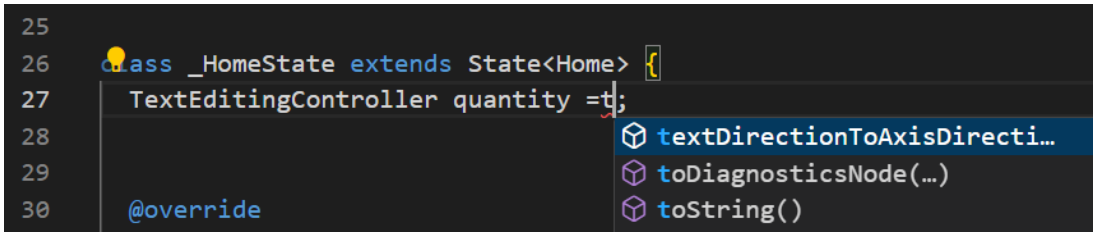
หมายเหตุ
controller เป็นการกำหนดคุณสมบัติเพื่อควบคุมข้อความหรือข้อมูลที่อยู่ในเท็กซ์ฟิลด์ โดยจะกำหนดข้อมูลในการใช้งานเป็นแบบ TextEditingController ประกอบด้วยคุณสมบัติสำคัญ คือ text และ selection ซึ่งทำหน้าที่ในการแสดงผลข้อความตามต้องการ และแสดงการเลือกข้อความ ตามลำดับ (เอกรินทร์ วทัญญูเลิศสกุล, 2563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



ภาพประกอบ 4.176 สร้างโค้ดเพื่อใช้สำหรับคำนวณค่าที่ได้รับเข้ามา

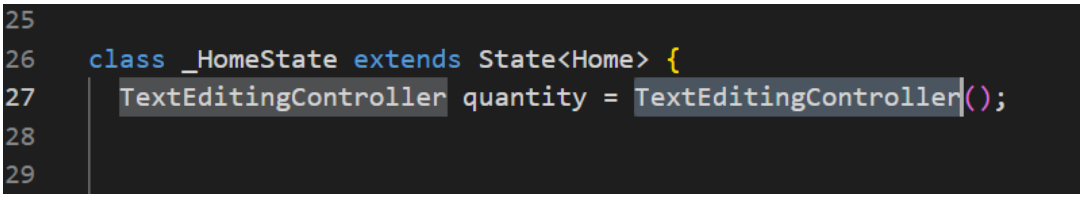
ที่มา : ณปภัช วรรณตรง (2564 : 20)

2.3.2 เมื่อทำการเพิ่มโค้ด TextEditingController แล้ว ให้ทำการกำหนดชื่อของตัวเก็บข้อมูลและโค้ด TextEditingController();



ภาพประกอบ 4.177 สร้างโค้ดเพื่อใช้สำหรับคำนวณค่าที่ได้รับเข้ามา (ต่อ)

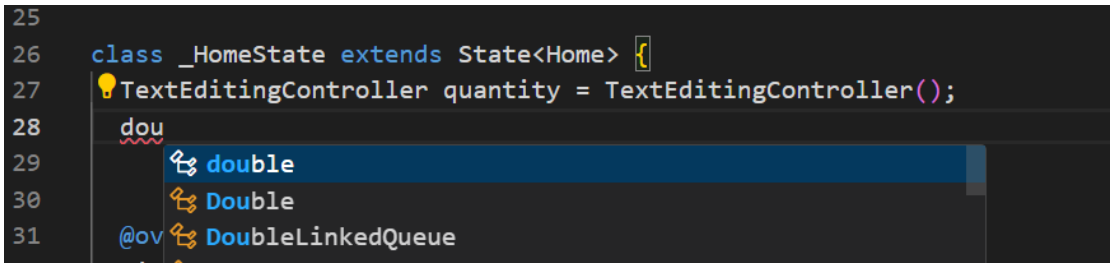
ที่มา : ณปภัช วรรณตรง (2564 : 20)



ภาพประกอบ 4.178 สร้างโค้ดเพื่อใช้สำหรับคำนวณค่าที่ได้รับเข้ามา (ต่อ)

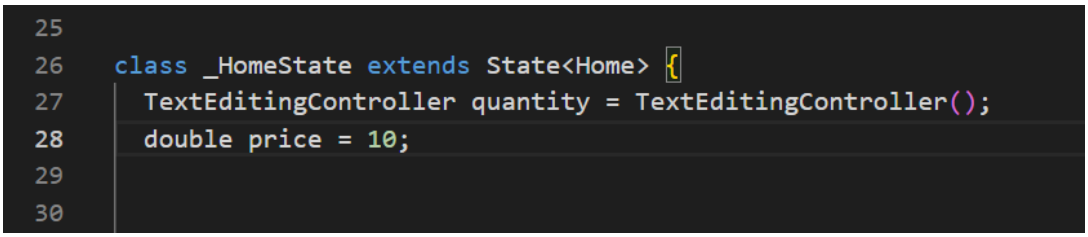
ที่มา : ณปภัช วรรณตรง (2564 : 20)

2.3.3 หลังจากนั้นให้ทำการกำหนดตัวแปรขึ้นมาหนึ่งตัว โดยใช้ประเภทของตัวแปรเป็น double



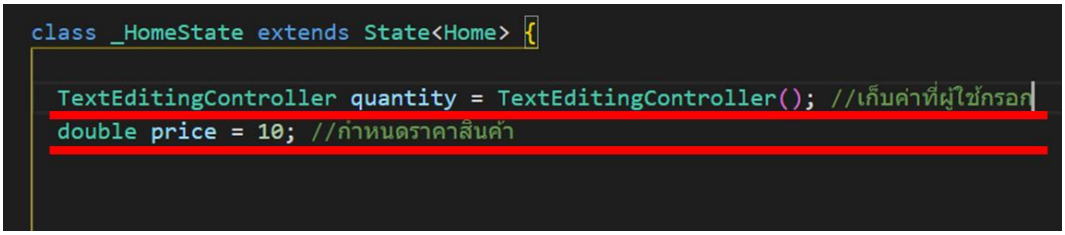
ภาพประกอบ 4.179 กำหนดตัวแปรประเภท double

ที่มา : ณปภัช วรรณตรง (2564 : 20)



ภาพประกอบ 4.180 กำหนดตัวแปรประเภท double (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 20)

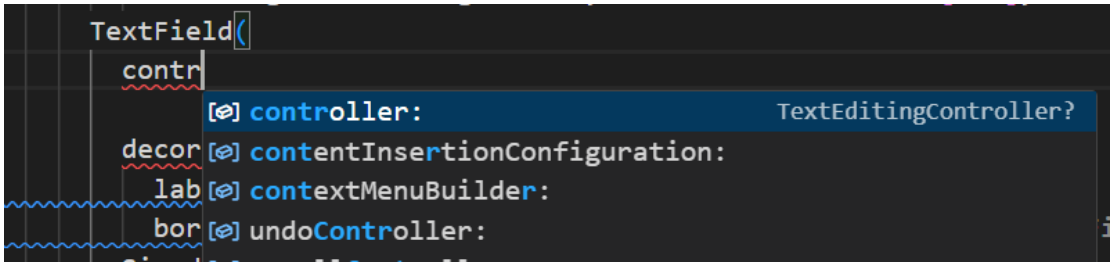


ภาพประกอบ 4.181 สร้างโค้ดเพื่อใช้สำหรับคำนวณค่าที่ได้รับเข้ามา (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 20)

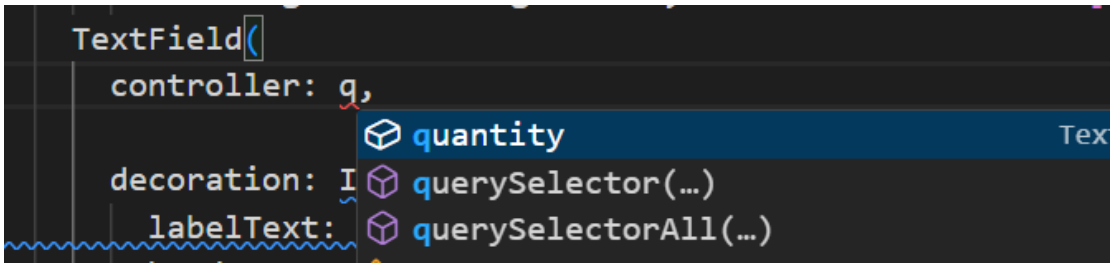
2.4 กำหนดโค้ดตัวแปรรับค่าใน TextField(),

เลื่อนหา TextField(...) จากนั้นภายใน (...) ให้ทำการเพิ่มโค้ด controller: เพื่อรับค่าและสั่งคำนวณ



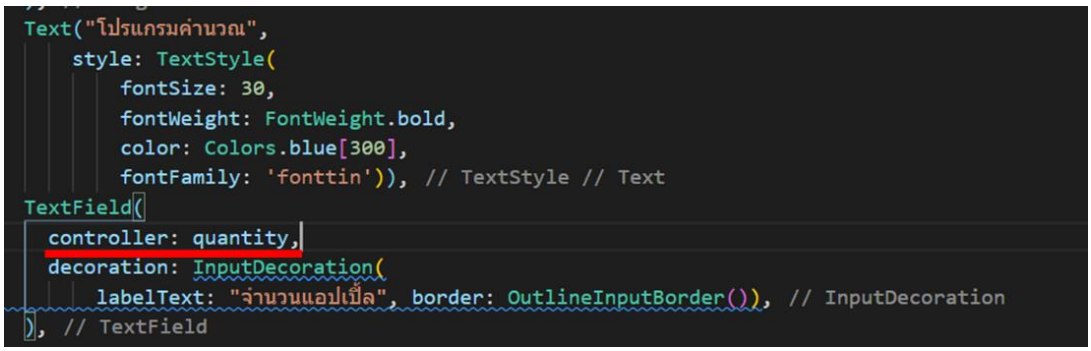
ภาพประกอบ 4.182 เพิ่มโค้ดเพื่อรับค่าและสั่งคำนวณ

ที่มา : ณปภัช วรรณตรง (2564 : 21)



ภาพประกอบ 4.183 เพิ่มโค้ดเพื่อรับค่าและสั่งคำนวณ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 21)

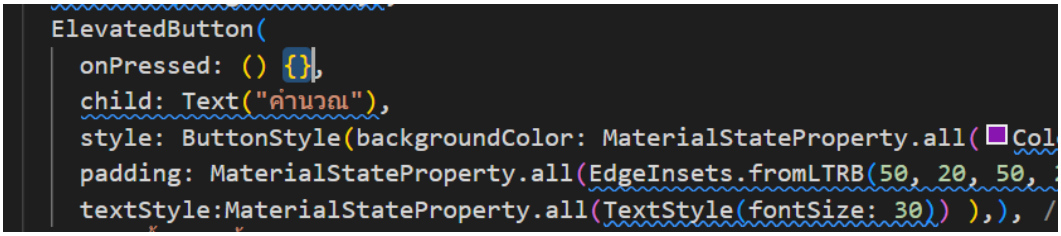


ภาพประกอบ 4.184 เพิ่มโค้ดเพื่อรับค่าและสั่งคำนวณ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 21)

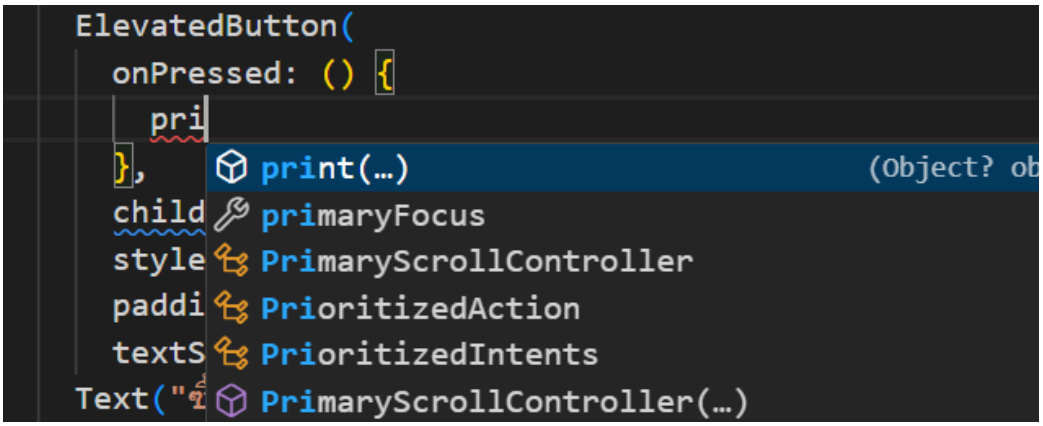
2.5 หลังจากนั้นให้ทำการเพิ่มชุดคำสั่ง print เพื่อตรวจสอบค่าที่ได้รับมาจากหน้าเว็บแอปพลิเคชันเข้าไปใน onPress: (){}

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



ภาพประกอบ 4.185 เพิ่มโค้ด print(...)

ที่มา : ณปภัช วรรณตรง (2564 : 22)



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

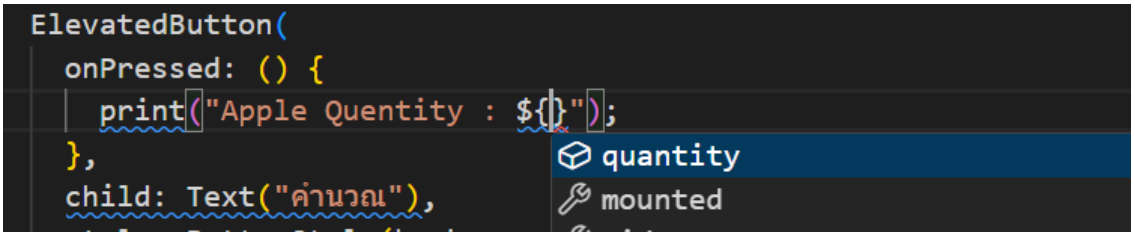
ที่มา : ณปภัช วรรณตรง (2564 : 22)

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

ภาพประกอบ 4.187 เพิ่มโค้ด print(...) (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 22)

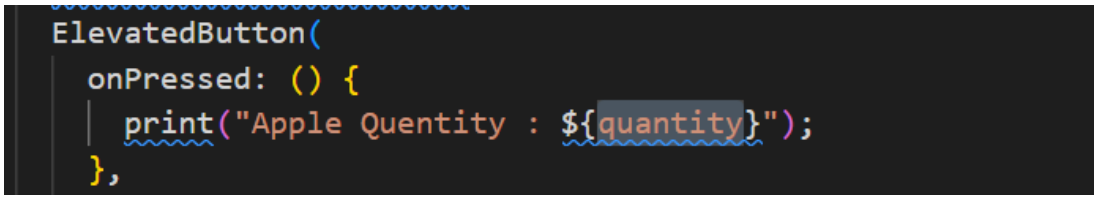
2.5.3 ต่อจาก “Apple Quentity:” ให้ทำการเพิ่มตัวแปรโดยใช้ $ เป็นตัวเก็บค่าตัวแปร จากนั้นให้เพิ่มตัวแปรที่กำหนดมา



ภาพประกอบ 4.188 เพิ่มโค้ด print(...) (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 22)

2.5.4 โดยหามี .text ตัวแปรควรมี {} ครอบไว้ หากไม่มีไม่จำเป็นต้องใส่ {}



ภาพประกอบ 4.189 เพิ่มโค้ด print(...) (ต่อ)

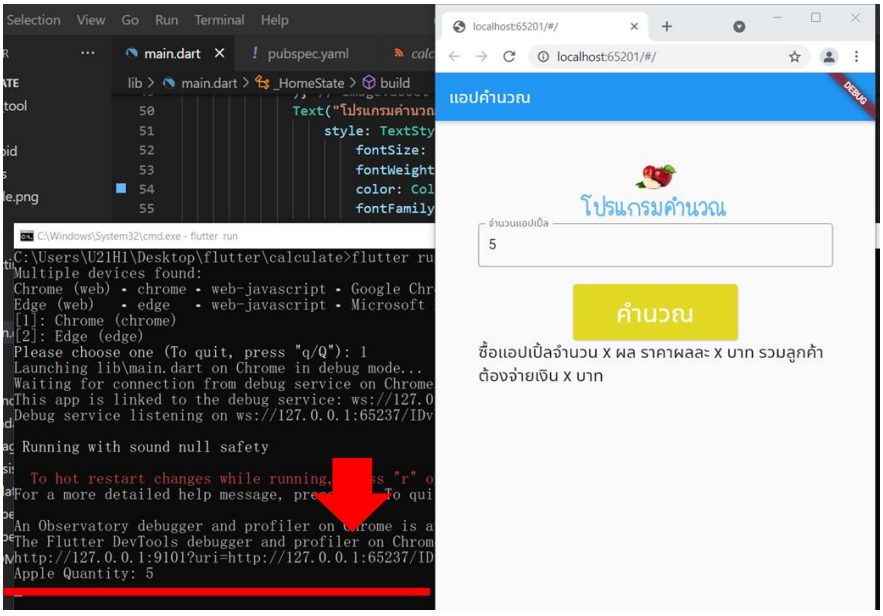
ที่มา : ณปภัช วรรณตรง (2564 : 22)

```
ElevatedButton(
  onPressed: () {

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


  },
```

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

ที่มา : ณปภัช วรรณตรง (2564 : 22)

2.6 ทำการทดสอบโดยการใส่ค่าเข้าไปใน TextField และลองคลิกปุ่มคำนวณที่ได้กำหนดไว้ จากนั้นให้ไปดูผลลัพธ์ที่หน้าต่าง cmd

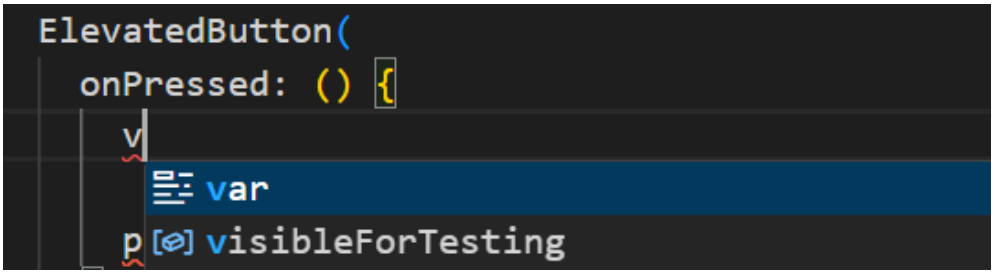


ภาพประกอบ 4.191 ทดสอบรับค่าและตรวจสอบค่าที่ได้รับมา

ที่มา : ณปภัช วรรณตรง (2564 : 23)

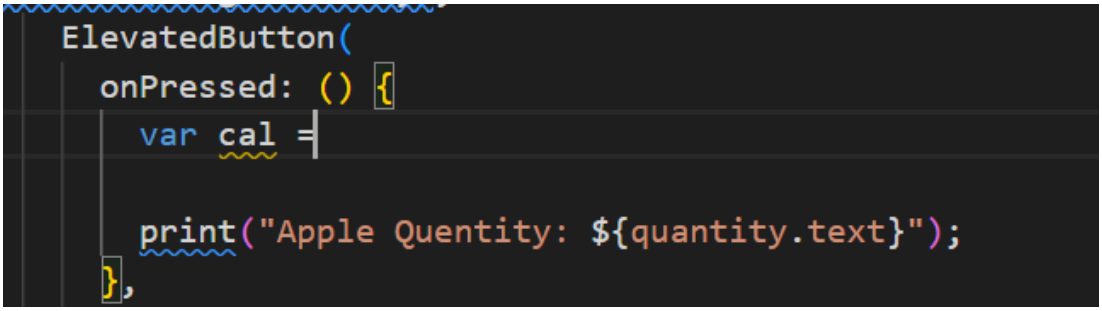
2.7 เมื่อได้ผลลัพธ์ที่ถูกต้องแล้ว ให้ทำการเพิ่มโค้ดสำหรับเรียกใช้การคำนวณ จากนั้นให้ทำการเพิ่มตัวแปรสำหรับรับค่าการคำนวณ

2.7.1 ใน onPressed: (){...} ให้ทำการกำหนดตัวแปรสำหรับรับค่าผลการคำนวณและนำค่าการคำนวณนั้นไปแสดง



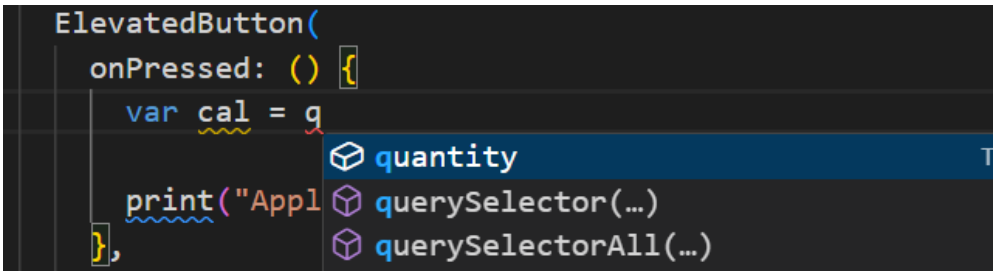
ภาพประกอบ 4.192 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ
ที่มา : ณปภัช วรรณตรง (2564 : 24)

2.7.2 ให้ทำการกำหนดโค้ดเป็น var cal เพื่อเป็นตัวแปรสำหรับรับค่าผลการคำนวณ



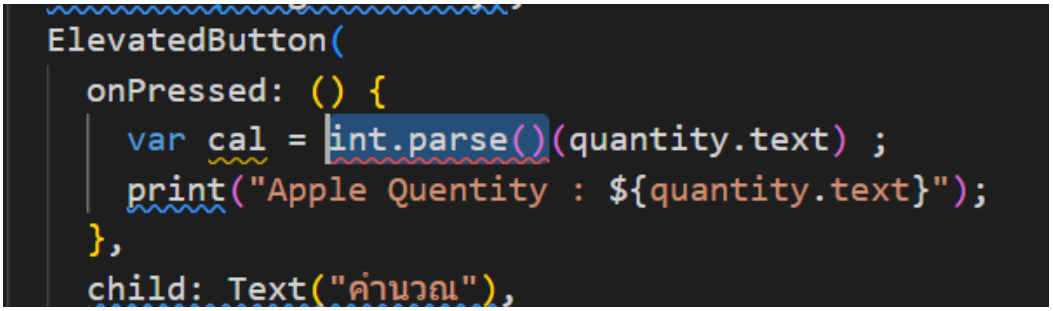
ภาพประกอบ 4.193 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 24)

2.7.3 จากนั้นให้ทำการกำหนดสูตรคำนวณเพื่อนำไปใช้คำนวณค่าที่ได้รับมา แล้วเอาไปเก็บไว้ในค่าตัวแปรที่ได้ทำการกำหนดไว้



ภาพประกอบ 4.194 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 24)

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



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 Quentity : ${quantity.text}");
  },
  child: Text("คำนวณ"),
```

ภาพประกอบ 4.196 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ)

ที่มา : ณปภัช 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 Quentity : ${quantity.text} Total : $ca");
  },
  child: Text("คำนวณ"),
```

cal
callConstructor(…)

ภาพประกอบ 4.197 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 24)

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

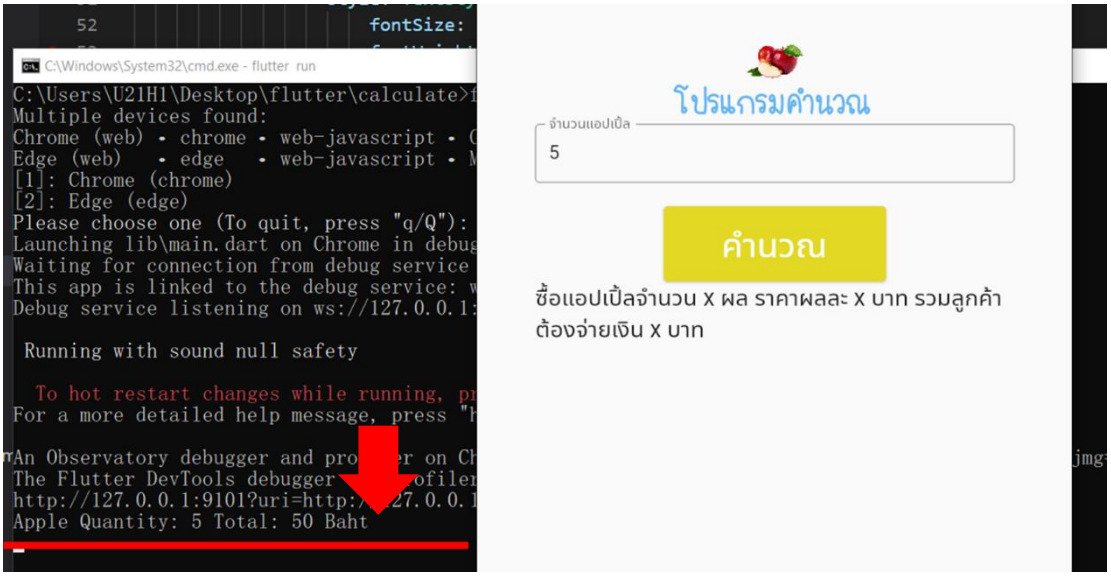
ภาพประกอบ 4.198 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ)

ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : 24)

2.8 เมื่อทำการเพิ่มโค้ดการคำนวณเสร็จแล้ว ให้ทำการใส่ค่าใน TextField จากนั้นให้คลิกปุ่มคำนวณแล้วไปดูผลลัพธ์ที่หน้าต่าง cmd อีกครั้งตามภาพ



ภาพประกอบ 4.200 เพิ่มโค้ดสำหรับการคำนวณและแสดงผลการคำนวณ (ต่อ)
ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : 26)

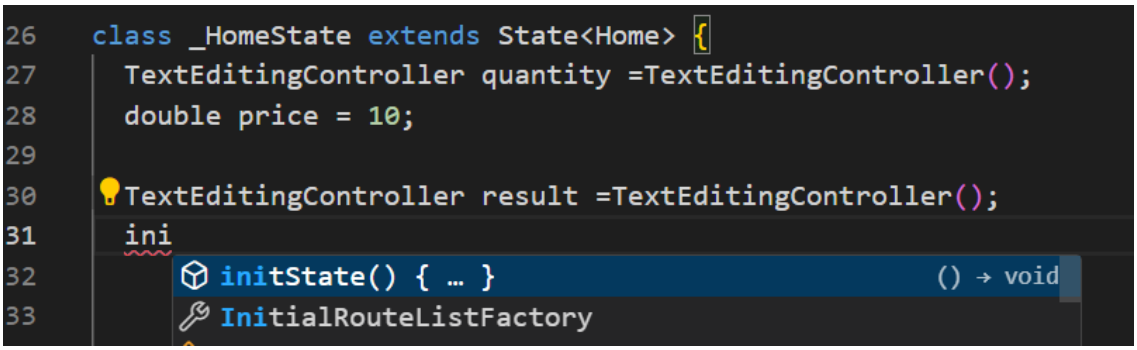
```
25
26   class _HomeState extends State<Home> {
27     TextEditingController quantity =TextEditingController();
28     double price = 10;
29
30     TextEditingController quantity =TextEditingController();
31
```

ภาพประกอบ 4.202 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){} (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 26)

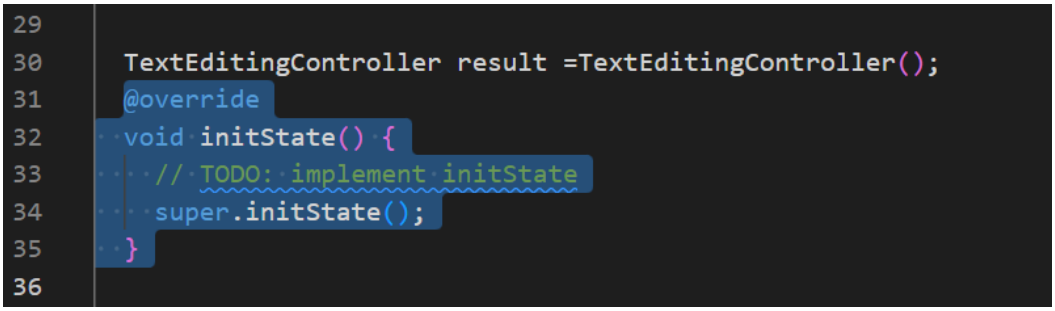
```
25
26  class _HomeState extends State<Home> {
27    TextEditingController quantity =TextEditingController();
28    double price = 10;
29
30    TextEditingController result =TextEditingController();
31
```

ภาพประกอบ 4.203 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){} (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 26)

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

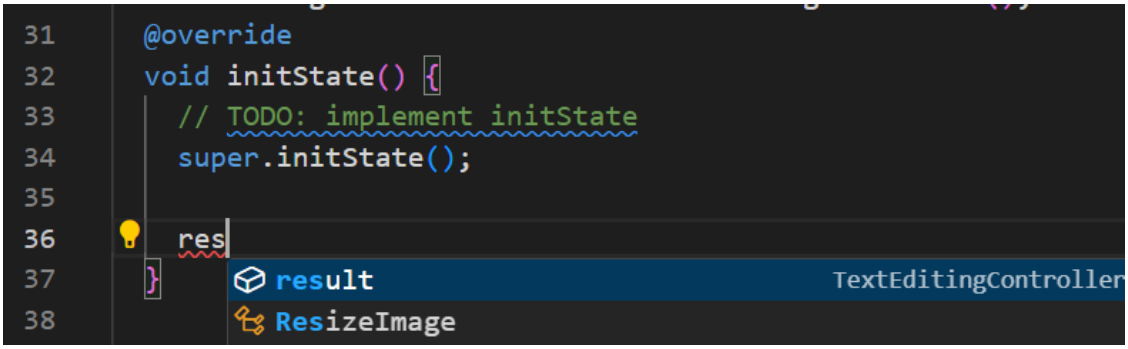


ภาพประกอบ 4.204 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){} (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 26)



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

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



ภาพประกอบ 4.206 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){} (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 26)

ภาพประกอบ 4.207 กำหนดตัวแปรรับผลคำนวณและสร้างฟังก์ชัน initState(){} (ต่อ)
ที่มา : ณปภัช วรรณตรง (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)

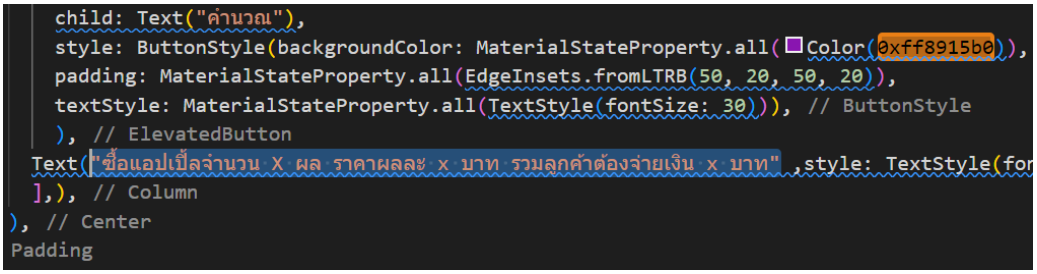
ที่มา : ณปภัช วรรณตรง (2564 : 26)

```
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)

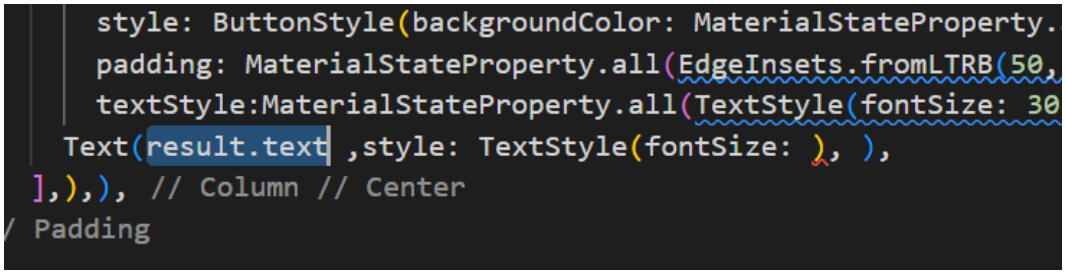
ที่มา : ณปภัช วรรณตรง (2564 : 26)

2.9 ทำการแทนที่คำอธิบายเดิมด้านล่างของผลลัพธ์ ด้วยตัวแปรผลลัพธ์ที่ได้สร้างไว้เพื่อทำการแสดงผลการคำนวณผ่านหน้าต่างแอปพลิเคชัน ด้วยการเปลี่ยนโค้ดของ Text(...) จากข้อความให้เป็นตัวแปรข้อความที่ได้ทำการกำหนดไว้



ภาพประกอบ 4.210 เปลี่ยนข้อความของ Text(...) ให้เป็นตัวแปรข้อความ

ที่มา : ณปภัช วรรณตรง (2564 : 27)



ภาพประกอบ 4.211 เปลี่ยนข้อความของ Text(...) ให้เป็นตัวแปรข้อความ (ต่อ)

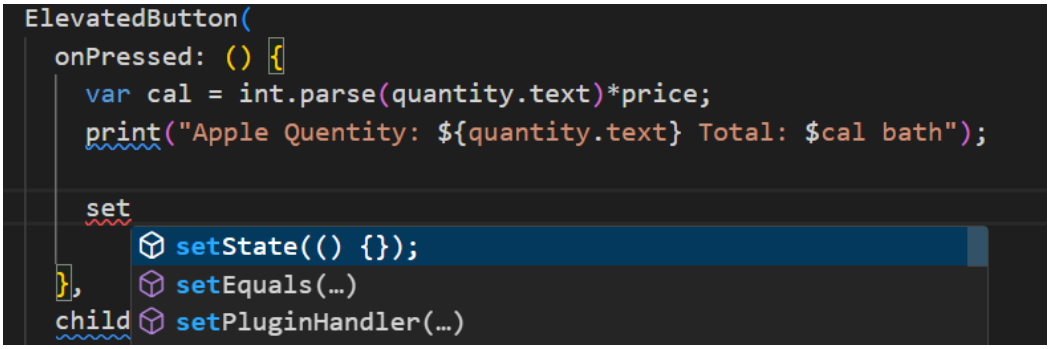
ที่มา : ณปภัช วรรณตรง (2564 : 27)

ภาพประกอบ 4.212 เปลี่ยนข้อความของ Text(...) ให้เป็นตัวแปรข้อความ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 27)

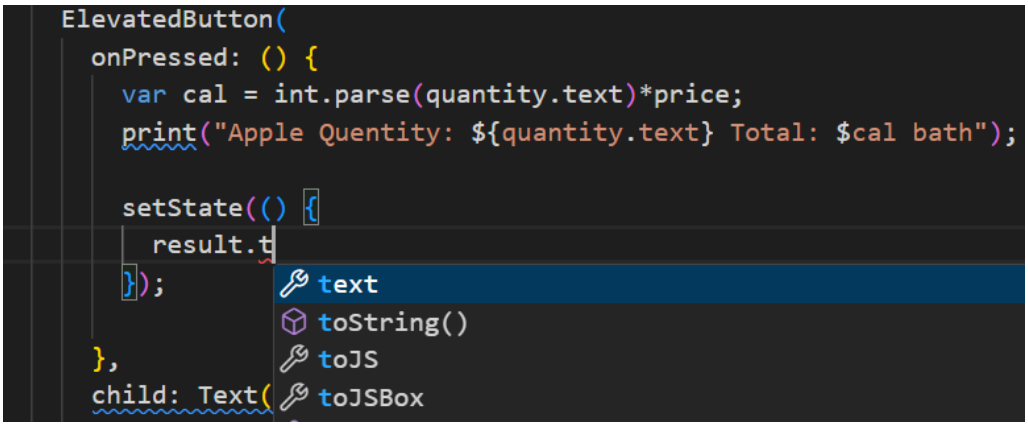
2.10 สร้างฟังก์ชัน setState(){} ขึ้นมาสำหรับใช้ในการแสดงผลลัพธ์การคำนวณที่ได้มีการเปลี่ยนแปลง

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

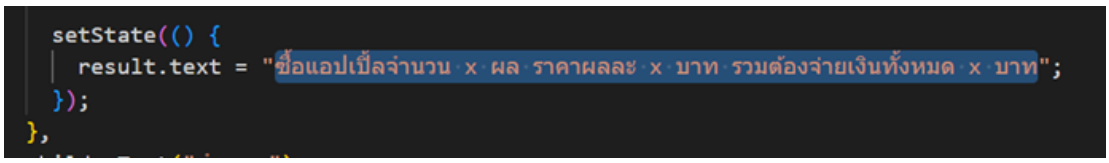


ภาพประกอบ 4.213 เพิ่ม class setState((){...})
ที่มา : ณปภัช วรรณตรง (2564 : 28)

2.10.2 จากนั้นภายใน {...} ให้ทำการนำตัวแปรค่าการคำนวณมาใส่



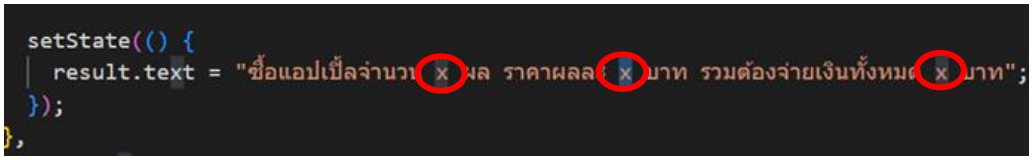
ภาพประกอบ 4.214 เพิ่ม class setState((){...}) (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 28)



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

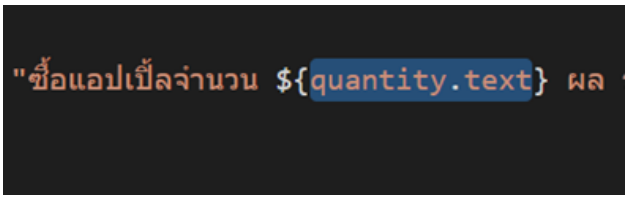
ที่มา : ณปภัช วรรณตรง (2564 : 28)

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



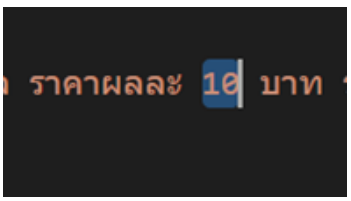
ภาพประกอบ 4.216 เปลี่ยนการแสดงผลการคำนวณ

ที่มา : ณปภัช วรรณตรง (2564 : 28)



ภาพประกอบ 4.217 เปลี่ยนการแสดงผลการคำนวณ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 28)



ภาพประกอบ 4.218 เปลี่ยนการแสดงผลการคำนวณ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 28)

```
รวมต้องจ่ายเงินทั้งหมด $cal บาท";
```

ภาพประกอบ 4.219 เปลี่ยนการแสดงผลการคำนวณ (ต่อ)

ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : 28)

2.11 จากนั้นให้ทำการสั่งการทำงานเพื่อทดสอบว่าแอปพลิเคชันทำงานถูกต้องหรือไม่

ภาพประกอบ 4.221 สั่งการทำงานเพื่อทดสอบแอปพลิเคชัน

ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : 30)

2.12.2 จากนั้นให้ทำการเปลี่ยนชื่อตัวแปรให้เป็น price และทำการคอมเม้นท์หรือปิดการทำงานของโค้ดตัวแปร 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 = "ซื้อแอปเปิ้ลจำนวน 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) เพื่อนำค่าที่ได้รับมามาคำนวณแทนค่าคงที่ที่ได้กำหนดไว้ก่อนหน้า

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

ภาพประกอบ 4.225 แก้ไขโค้ดของผลการคำนวณ

ที่มา : ณปภัช วรรณตรง (2564 : 31)

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

ภาพประกอบ 4.226 แก้ไขโค้ดของผลการคำนวณ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 31)

2.13.2 จากนั้นให้ทำการเปลี่ยนข้อความอธิบายเป็นตัวแปร ${price.text}

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

ภาพประกอบ 4.227 แก้ไขโค้ดของผลการคำนวณ (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 31)

```
ราคาผลละ ${price.text} บาท
```

ภาพประกอบ 4.228 แก้ไขโค้ดของผลการคำนวณ (ต่อ)

ที่มา : ณปภัช วรรณตรง (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 =
          "ซื้อแอปเปิ้ลจำนวน ${quantity.text} ผล ราคาผลละ ${price.text} บาท รวมต้องจ่ายทั้งหมด $cal บาท";
    });
  },
```

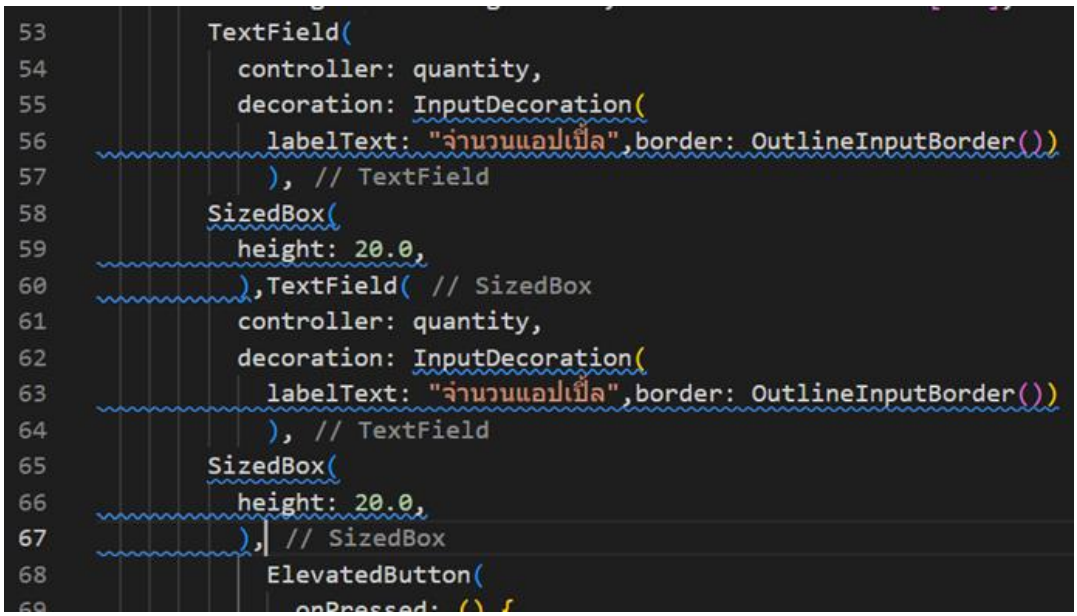
ภาพประกอบ 4.229 แก้ไขโค้ดของผลการคำนวณ (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 31)

2.14 เมื่อทำการแก้ไขและเพิ่มโค้ดกำหนดค่าต่าง ๆ เรียบร้อยแล้ว ให้ทำการเพิ่มโค้ด textField() เข้าไปในแอปพลิเคชันสำหรับค่าราคาหรือค่าที่ต้องการใส่เข้ามาเพื่อคำนวณ

2.14.1 ภายใน Padding(...) ให้ทำการเพิ่ม TextField(...) สำหรับการรับค่า price ต่อจาก Text(...) โดยการคัดลอก TextField(...) ที่ได้กำหนดไว้ก่อนหน้ามาวาง 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(
```

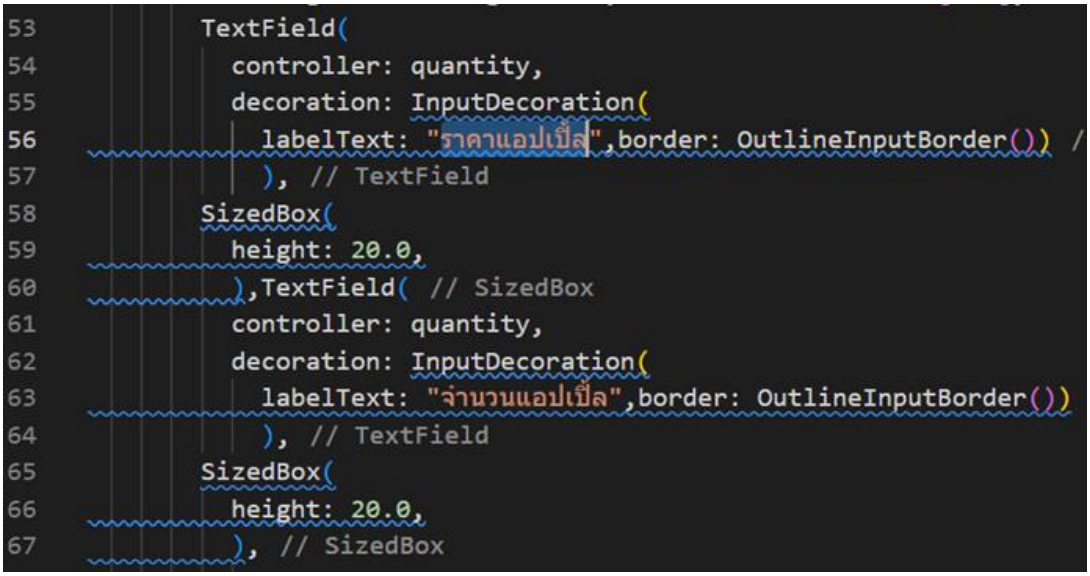
ภาพประกอบ 4.230 เพิ่มโค้ด textField() สำหรับรับค่า price
ที่มา : ณปภัช วรรณตรง (2564 : 32)



ภาพประกอบ 4.231 เพิ่มโค้ด textField() สำหรับรับค่า price

ที่มา : ณปภัช วรรณตรง (2564 : 32)

2.14.2 ให้ทำการเปลี่ยนข้อความและตัวแปรภายใน TextField(...)



ภาพประกอบ 4.232 เพิ่มโค้ด textField() สำหรับรับค่า price (ต่อ)

ที่มา : ณปภัช วรรณตรง (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 : ณปภัช วรรณตรง (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
SizedBox(
  height: 20.0,
), // SizedBox
TextField(
    controller: quantity,
    decoration: InputDecoration(
        labelText: 'จำนวนแอปเปิ้ล',
        border: OutlineInputBorder())), // InputDecoration // TextField
```

ภาพประกอบ 4.234 เพิ่มโค้ด textField() สำหรับรับค่า price (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 32)

2.15 เมื่อเพิ่มโค้ดเสร็จสิ้นแล้วให้สั่งการทำงาน ซึ่งจะได้ผลลัพธ์ตามภาพ

ภาพประกอบ 4.235 ผลลัพธ์ของแอปพลิเคชันสำหรับคำนวณ
ที่มา : ณปภัช วรรณตรง (2564 : 33)

บทสรุป

สำหรับเนื้อหาที่กลาวมาทั้งหมดในบทนี้ จะพูดถึงการเขียนคำสั่งเพื่อลดขนาดและลบไฟล์ที่ไม่จำเป็นก่อนส่งไฟล์ให้ผู้อื่น การกำหนด layout แบบคอลัมภ์ในแอปพลิเคชัน 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). **พัฒนาโมไบล์แอพด้วย Flutter + Dart.** กรุงเทพฯ : สำนักพิมพ์ซิมพลิฟาย

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

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

บทที่ 5
การสร้างหน้าแอปพลิเคชันและแท็บบาร์สำหรับผู้ใช้งาน

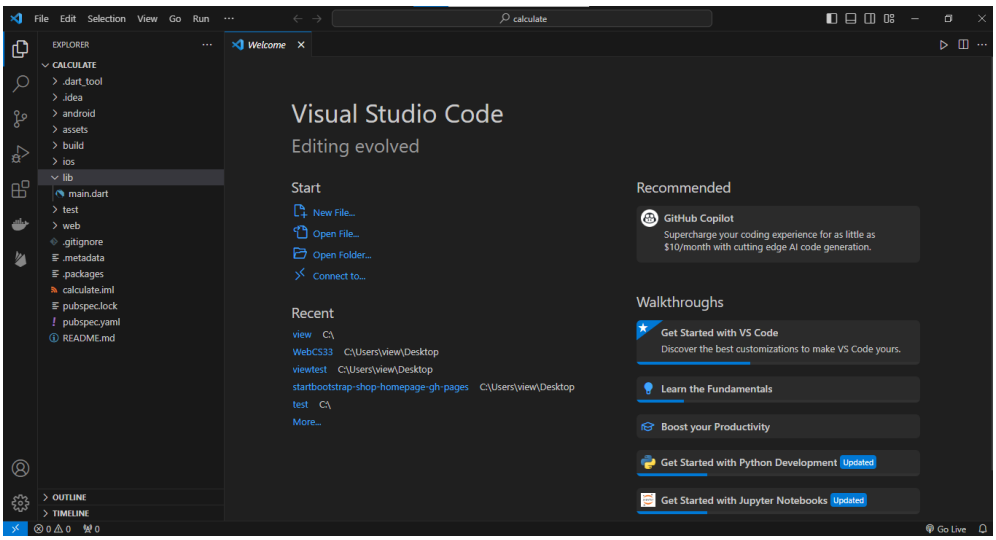
จากบทที่แล้วผู้เรียนได้เรียนรู้การสร้างแอปพลิเคชันที่สามารถคำนวณค่าได้ ซึ่งเป็นแอปพลิเค-ชันที่มีลักษณะใช้งานเพียงหน้าเดียว ในบทนี้ผู้เรียนจะได้เรียนรู้การสร้างแอปพลิเคชันมากกว่า 1 หน้า โดยจะเรียนรู้วิธีการสร้างหน้าแอปพลิเคชันแยกแต่ละหน้า จากนั้นเมื่อได้แอปพลิเคชันแต่ละหน้าแล้ว ผู้เรียนจะได้เรียนรู้วิธีการสร้างแท็บบาร์ที่ด้านล่างของแอปพลิเคชัน ซึ่งเป็นส่วนเมื่อคลิกที่แท็บบาร์แล้วจะไปยังหน้าจอที่เตรียมไว้ในขั้นตอนแรก เมื่อได้แอปพลิเคชันที่พร้อมใช้งานเรียบร้อย ผู้เรียนจะได้เรียนรู้การสร้างไอคอนของแอปพลิเคชัน เพื่อเมื่อนำไปติดตั้งบนอุปกรณ์เคลื่อนที่จะแสดงเป็นไอคอนให้ผู้ใช้คลิกบนหน้าจอเพื่อเข้าสู่หน้าจอแอปพลิชัน และในเนื้อหาสุดท้ายในการเรียนบทนี้หลังจากได้สร้างไอคอนแอปพลิเคชันแล้ว จะทำการสร้างไฟล์แอปพลิเคชันเป็นไฟล์สำหรับติดตั้งและใช้งานบนระบบปฏิบัติการแอนดรอยด์ โดยจะเป็นไฟล์นามสกุล .apk ที่สามารถนำไปติดตั้งบนอุปกรณ์เคลื่อนที่ให้สามารถเรียกใช้งานแอปพลิคชันได้ ดังรายละเอียดต่อไปนี้

การสร้างหน้าแอปพลิเคชันแยกแต่ละหน้า

ในส่วนนี้จะเป็นการสร้างไฟล์แต่ละหน้าของแอปพลิเคชัน มีรายละเอียดขั้นตอนการทำดังนี้

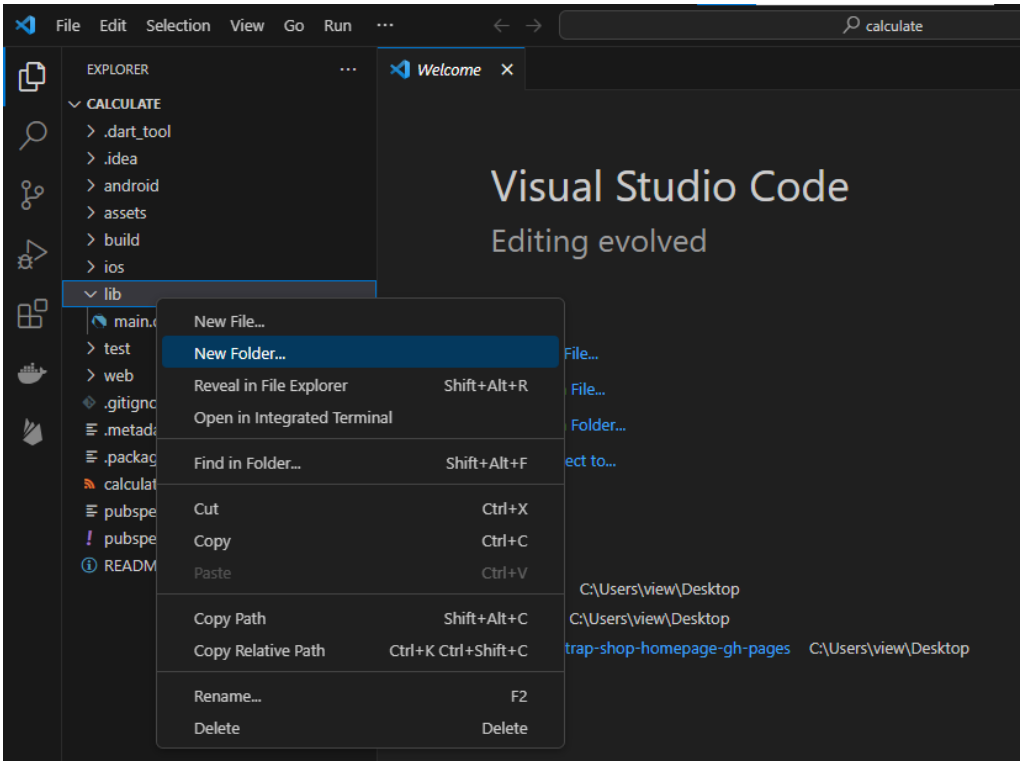
1. ให้ทำการเปิดโปรแกรม VSCode ขึ้นมาจากนั้นให้ทำการเปิดไฟล์ Project ที่ได้ทำการพัฒนาไว้ จากนั้นให้คลิกขวาที่ Folder lib เลือก New Folder ตั้งชื่อ Folder ว่า pages

เมื่อเปิดไฟล์ Project แล้ว ให้ไปที่ Folder lib จากนั้นให้ทำการสร้าง Folder ขึ้นมาใหม่โดยตั้งชื่อว่า pages



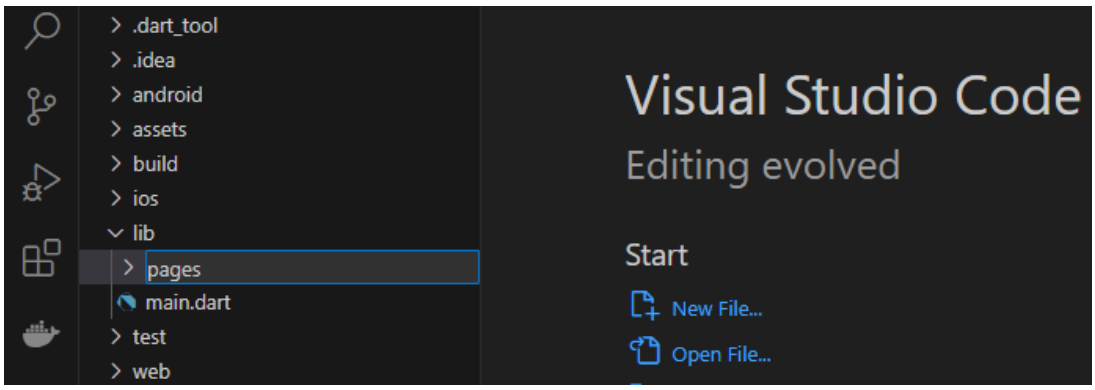
ภาพประกอบ 5.1 สร้าง Folder pages

ที่มา : ณปภัช วรรณตรง (2564 : 4)



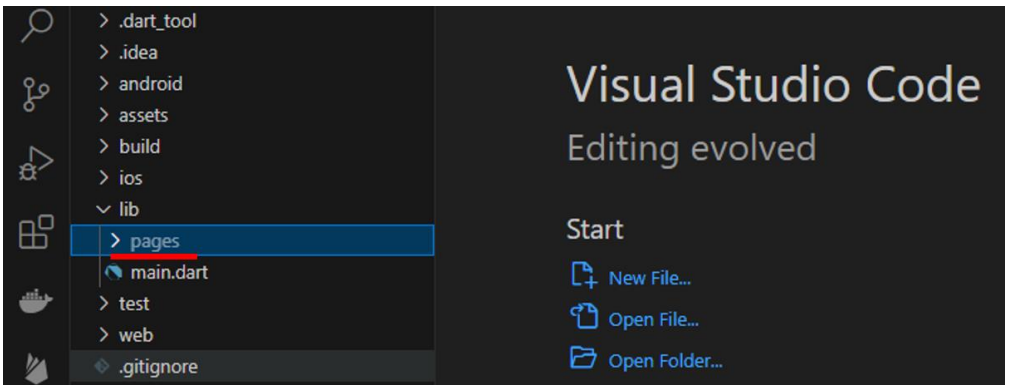
ภาพประกอบ 5.2 สร้าง Folder pages (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 4)



ภาพประกอบ 5.3 สร้าง Folder pages (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 4)

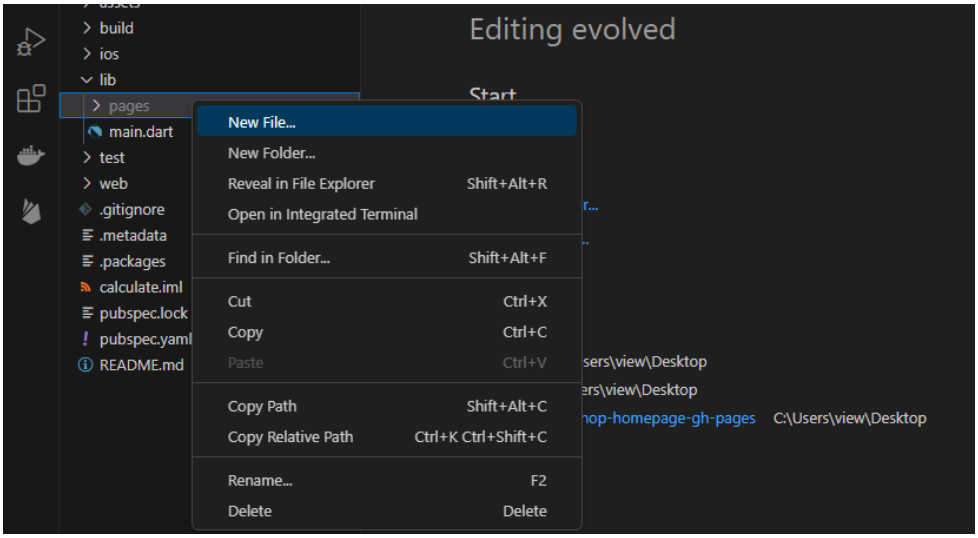


ภาพประกอบ 5.4 สร้าง Folder pages (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 4)

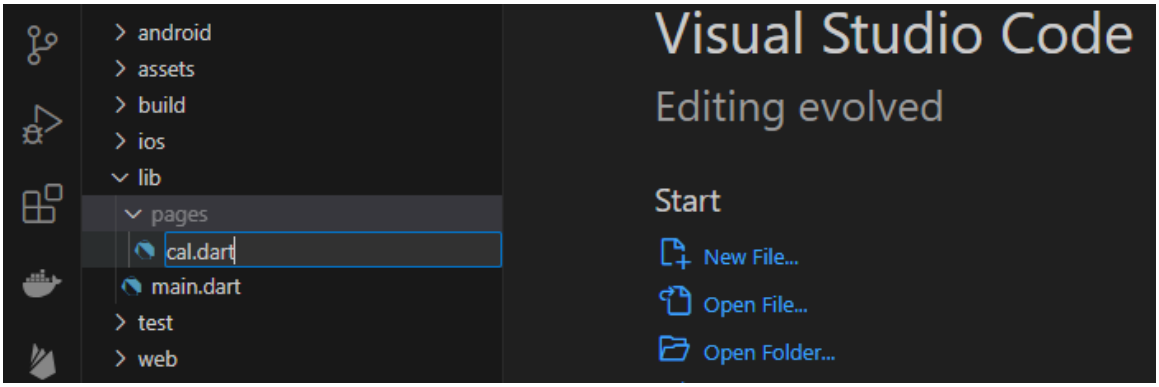
2. หลังจากที่ทำการสร้าง Folder pages เสร็จสิ้นแล้ว ให้ทำการสร้างไฟล์ 3 ไฟล์ขึ้นมา ได้แก่ cal.dart, contact.dart และ home.dart

กดเลือกที่ Folder pages จากนั้นให้ทำการสร้างไฟล์งานขึ้นมาใหม่ 3 ไฟล์



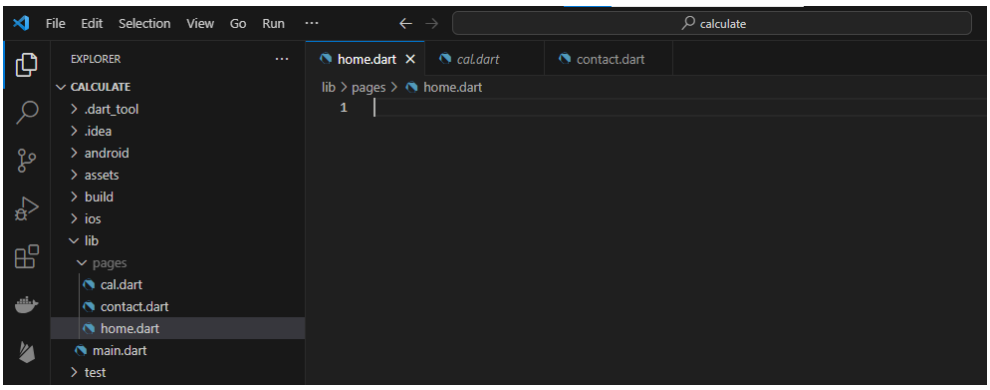
ภาพประกอบ 5.5 สร้างไฟล์งานใน Folder pages

ที่มา : ณปภัช วรรณตรง (2564 : 5)



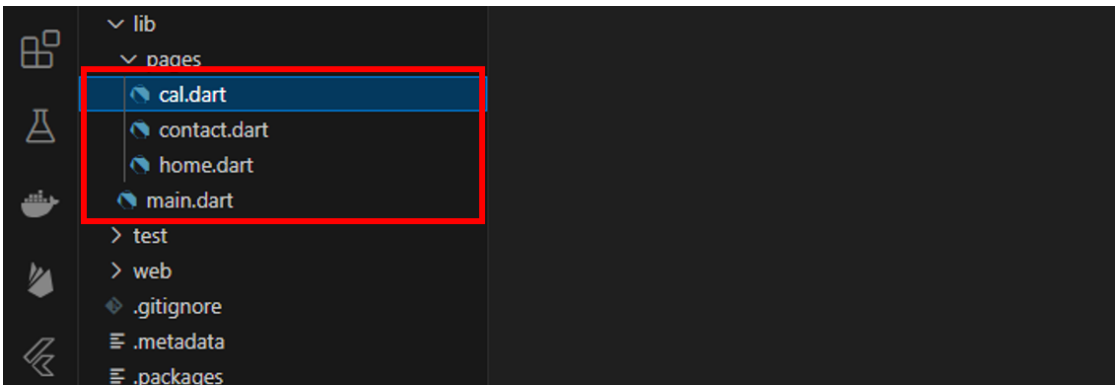
ภาพประกอบ 5.6 สร้างไฟล์งานใน Folder pages (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 5)



ภาพประกอบ 5.7 สร้างไฟล์งานใน Folder pages (ต่อ)

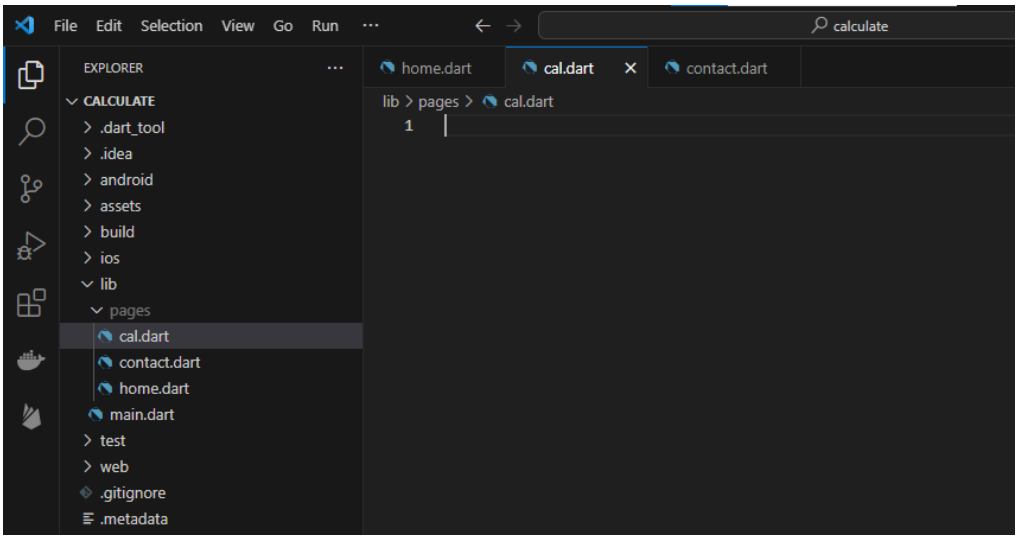
ที่มา : ณปภัช วรรณตรง (2564 : 5)



ภาพประกอบ 5.8 สร้างไฟล์งานใน Folder pages (ต่อ)

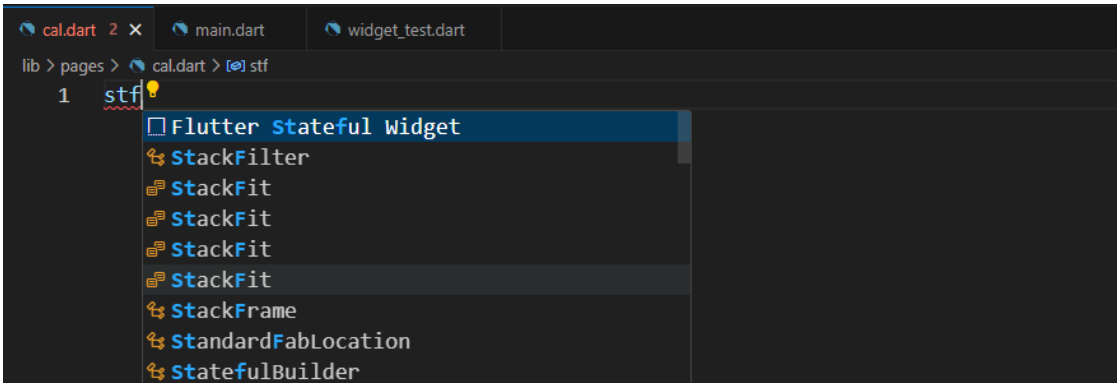
ที่มา : ณปภัช วรรณตรง (2564 : 5)

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



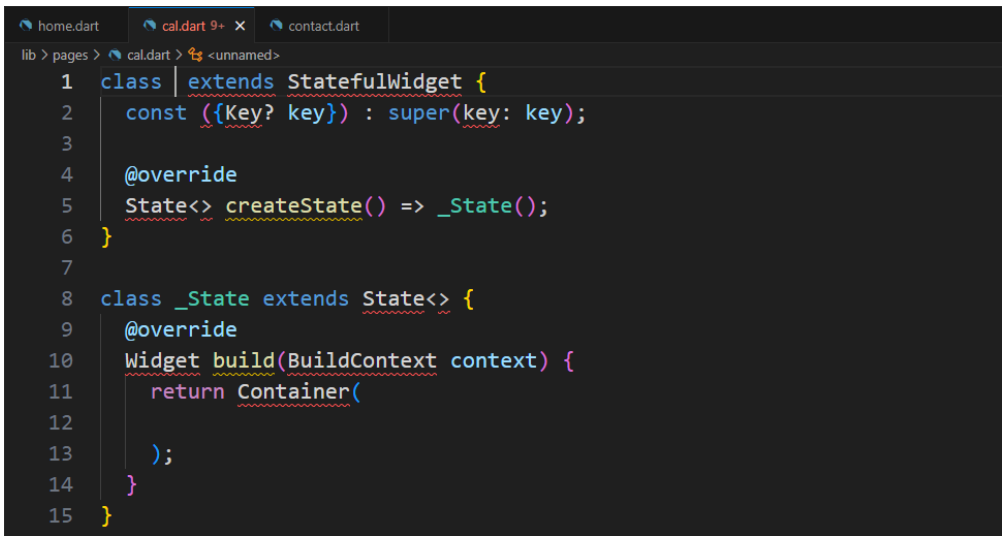
ภาพประกอบ 5.9 สร้าง class StatefulWidget

ที่มา : ณปภัช วรรณตรง (2564 : 6)



ภาพประกอบ 5.10 สร้าง class StatefulWidget (ต่อ)

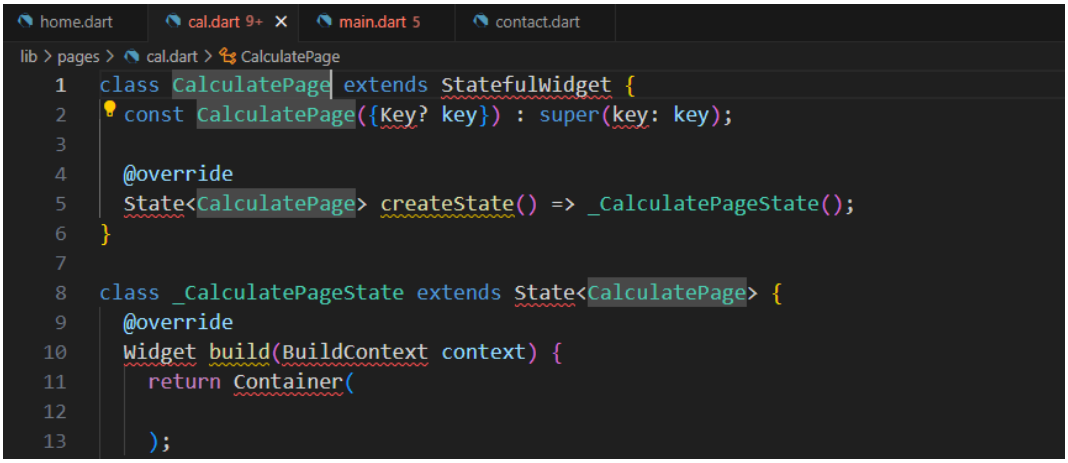
ที่มา : ณปภัช วรรณตรง (2564 : 6)



ภาพประกอบ 5.11 สร้าง class StatefulWidget (ต่อ)

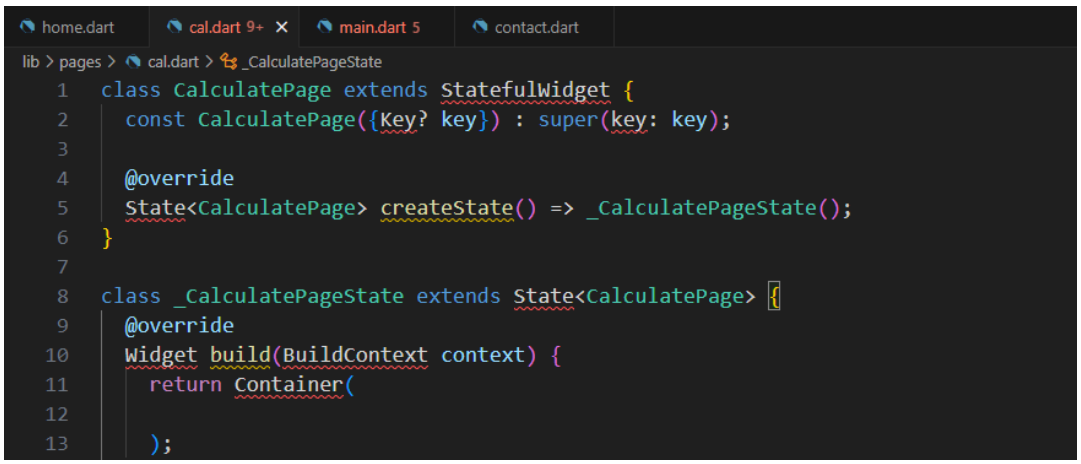
ที่มา : ณปภัช วรรณตรง (2564 : 6)

จากนั้นเมื่อได้ class StatefulWidget เรียบร้อยแล้ว ให้ทำการตั้งชื่อ class เป็น CalculatePage



ภาพประกอบ 5.12 สร้าง class StatefulWidget (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 6)

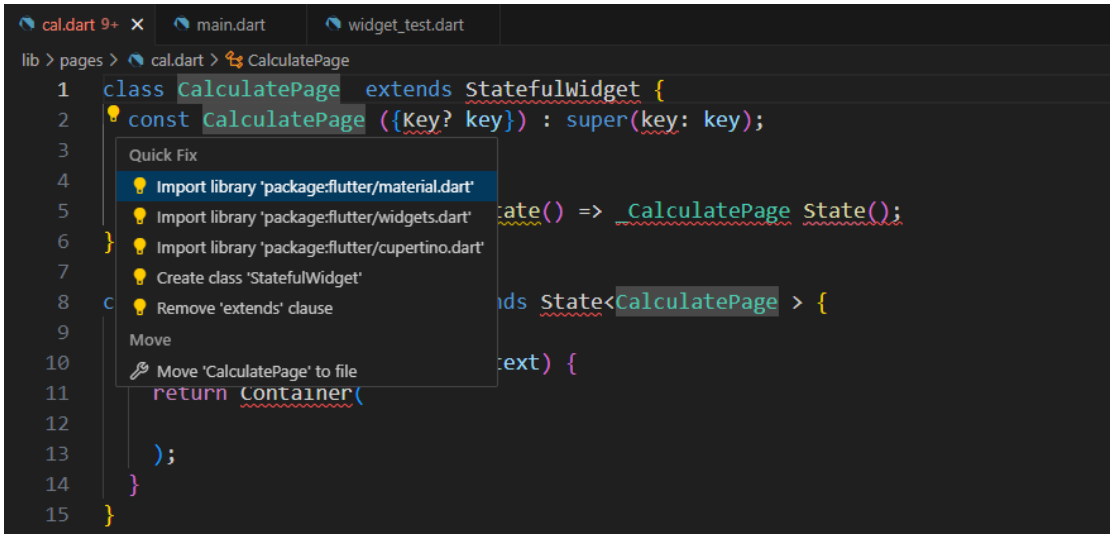


ภาพประกอบ 5.13 สร้าง class StatefulWidget (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 6)

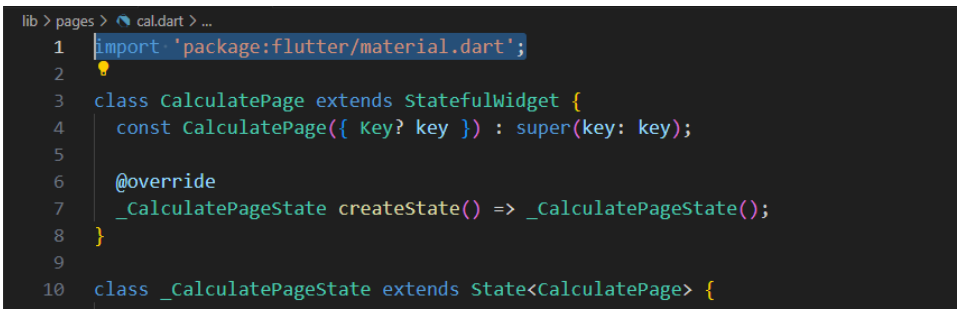
4. หากพบ Error ที่ class CalculatePage ให้ทำการเพิ่มโค้ดไว้ส่วนบนของไฟล์งานเป็น import Library 'package:flutter/material.dart';

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



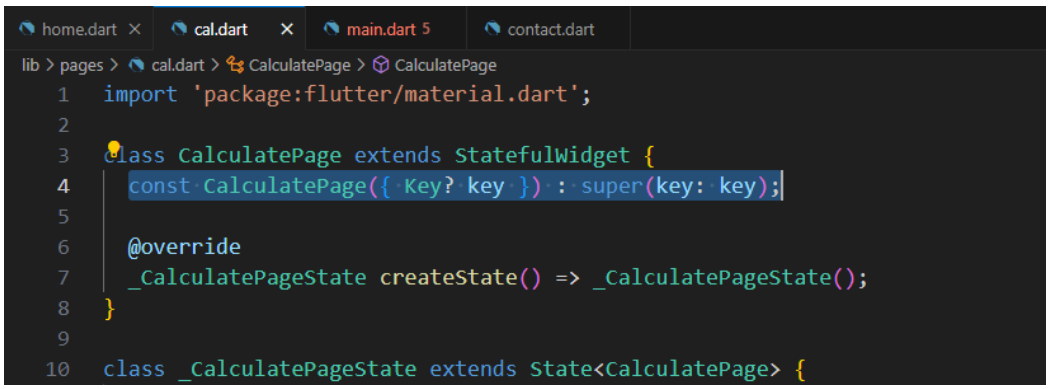
ภาพประกอบ 5.14 แก้ไขโค้ดที่ Error โดยการ import package

ที่มา : ณปภัช วรรณตรง (2564 : 7)



ภาพประกอบ 5.15 แก้ไขโค้ดที่ Error โดยการ import package (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 7)

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

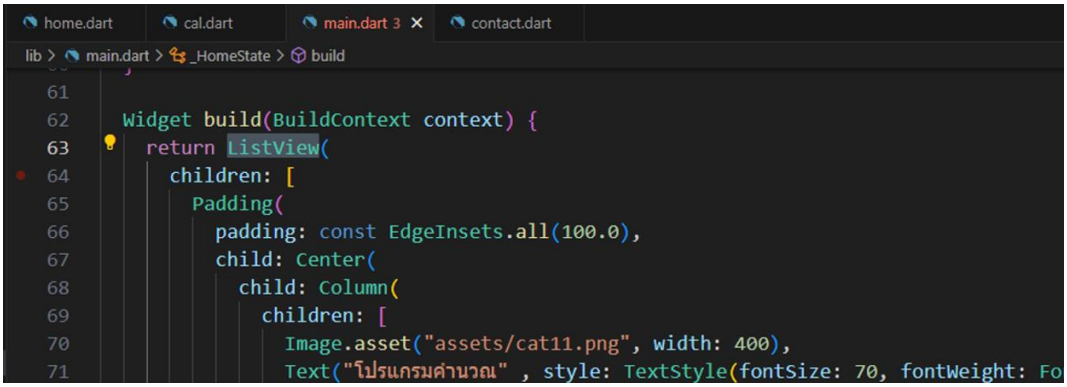


ภาพประกอบ 5.16 Comment ส่วนที่ไม่ต้องการใช้งาน
ที่มา : ณปภัช วรรณตรง (2564 : 8)

ภาพประกอบ 5.17 Comment ส่วนที่ไม่ต้องการใช้งาน (ต่อ)

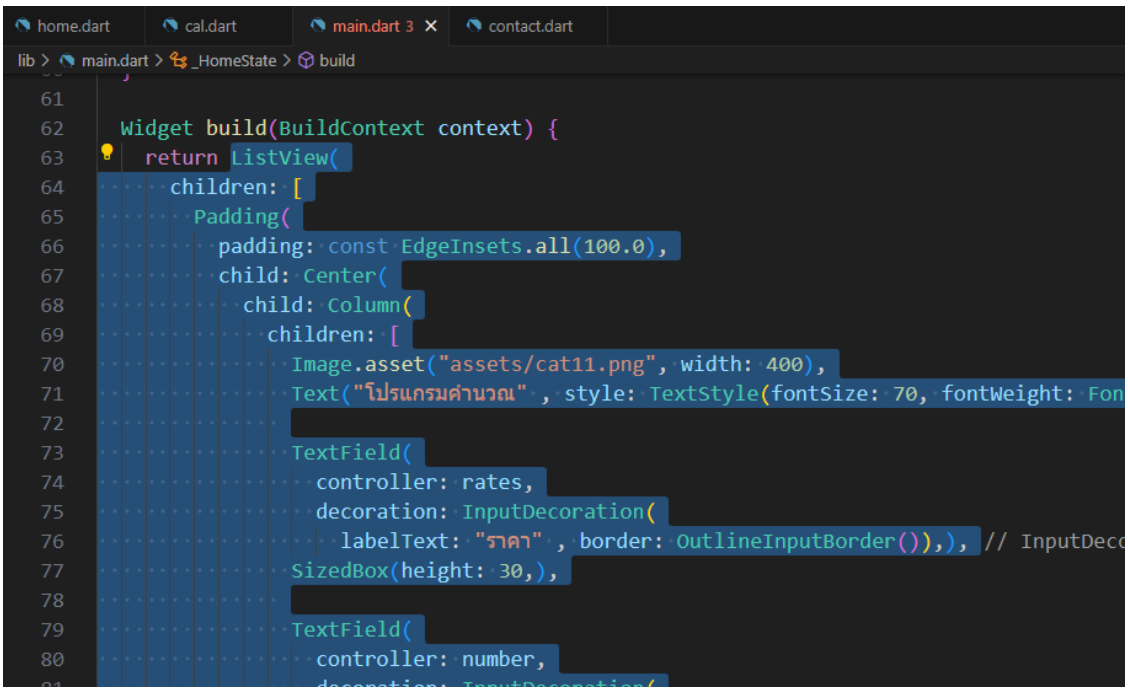
ที่มา : ณปภัช วรรณตรง (2564 : 8)

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



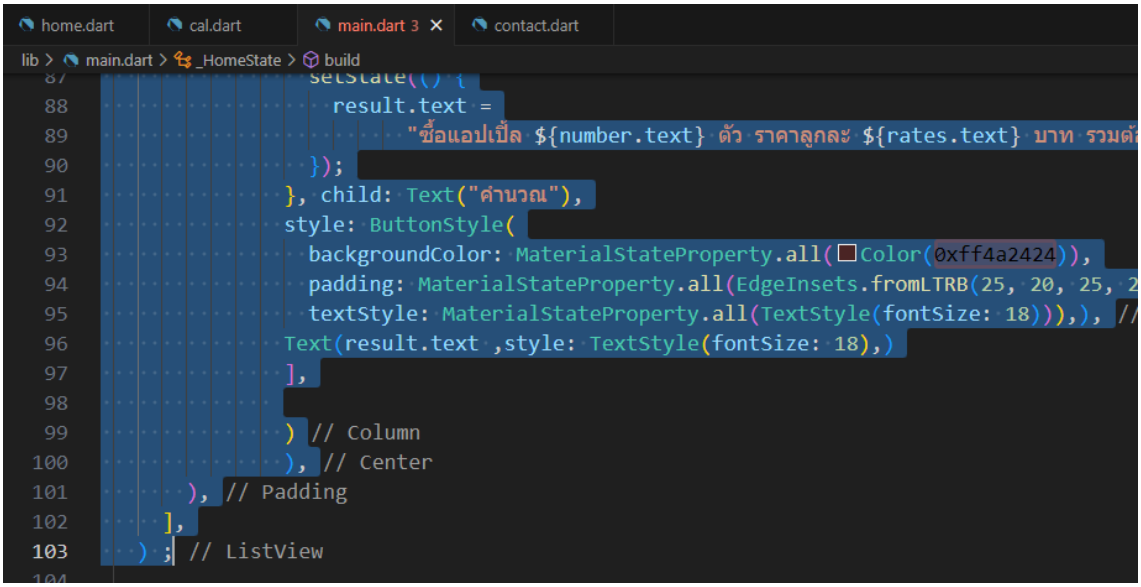
ภาพประกอบ 5.18 ตัด (Cut) โค้ดส่วนของ ListView();

ที่มา : ณปภัช วรรณตรง (2564 : 9)



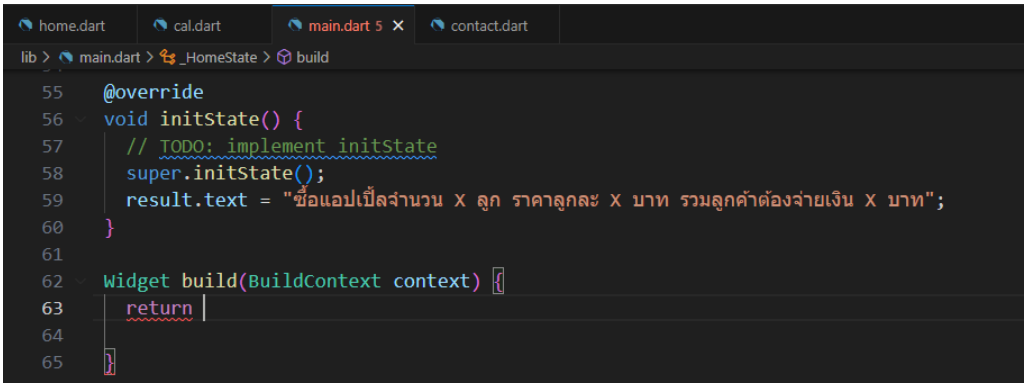
ภาพประกอบ 5.19 ตัด (Cut) โค้ดส่วนของ ListView(); (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 9)



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

ที่มา : ณปภัช วรรณตรง (2564 : 9)



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

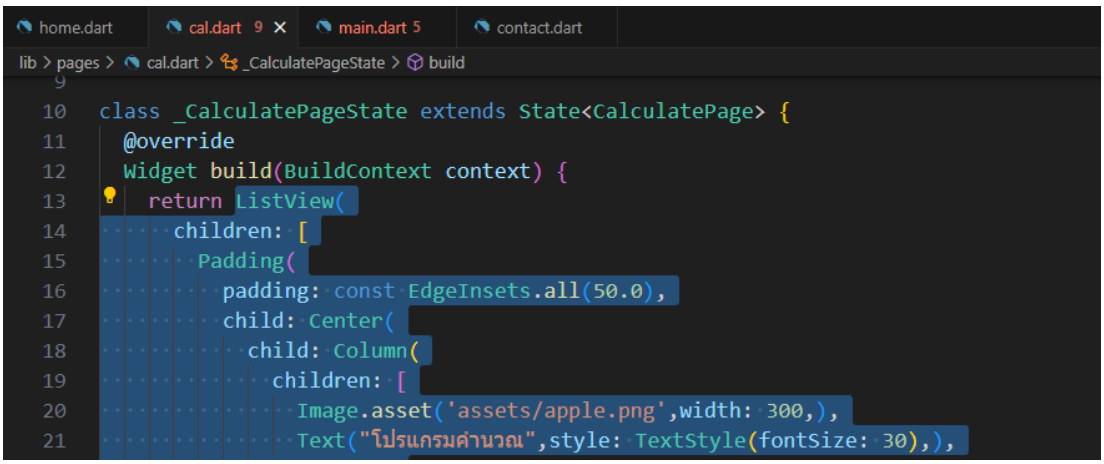
ที่มา : ณปภัช วรรณตรง (2564 : 9)

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

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

ภาพประกอบ 5.22 นำโค้ดที่ตัดมาใส่แทนโค้ดส่วน Container();

ที่มา : ณปภัช วรรณตรง (2564 : 10)



ภาพประกอบ 5.23 นำโค้ดที่ตัดมาใส่แทนโค้ดส่วน Container(); (ต่อ)

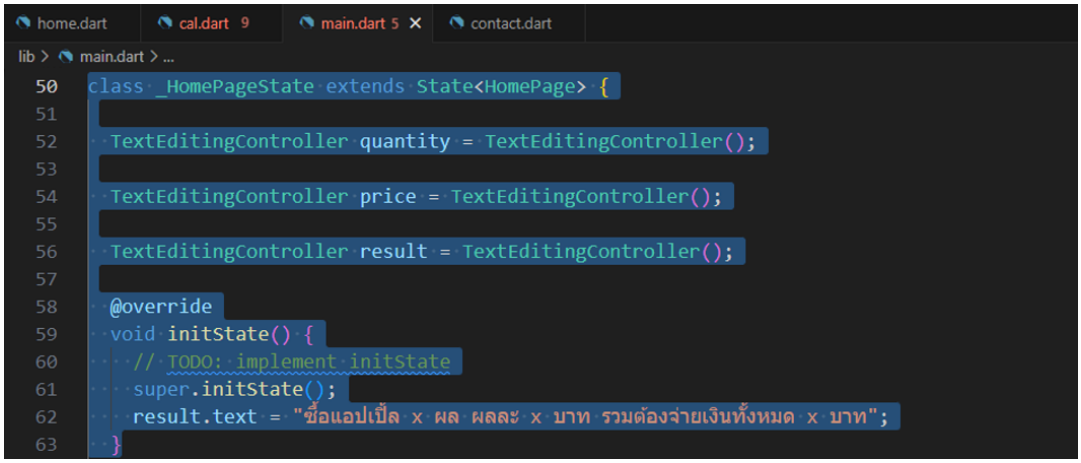
ที่มา : ณปภัช วรรณตรง (2564 : 10)

ภาพประกอบ 5.24 นำโค้ดที่ตัดมาใส่แทนโค้ดส่วน Container(); (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 10)

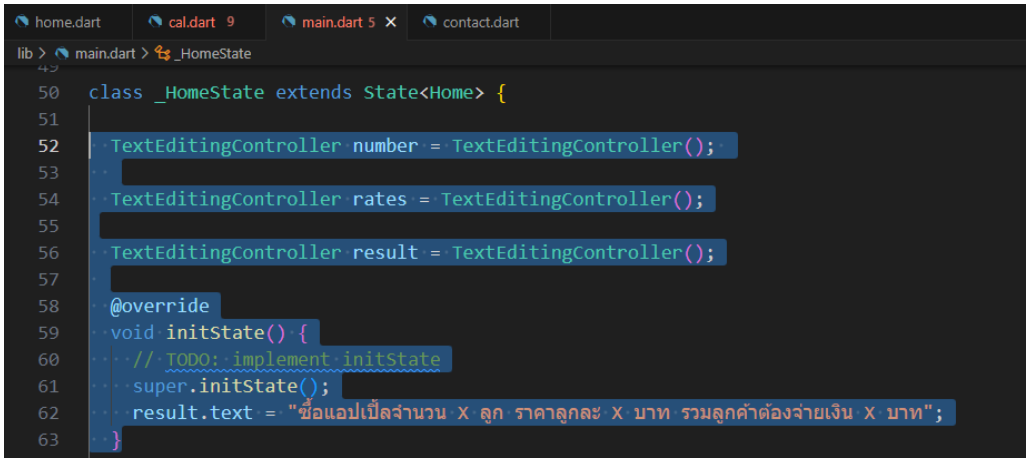
8. จะเห็นว่าโค้ดยังมี Error ให้ทำการกลับไปที่ไฟล์ main.dart จากนั้นทำการตัดโค้ดส่วนที่เหลือเข้ามาไว้ยังไฟล์ cal.dart

ภายใน class _HomeState ให้ทำการคัดลอกหรือตัดโค้ดที่ประกาศตัวแปรมาใส่ในไฟล์ cal.dart



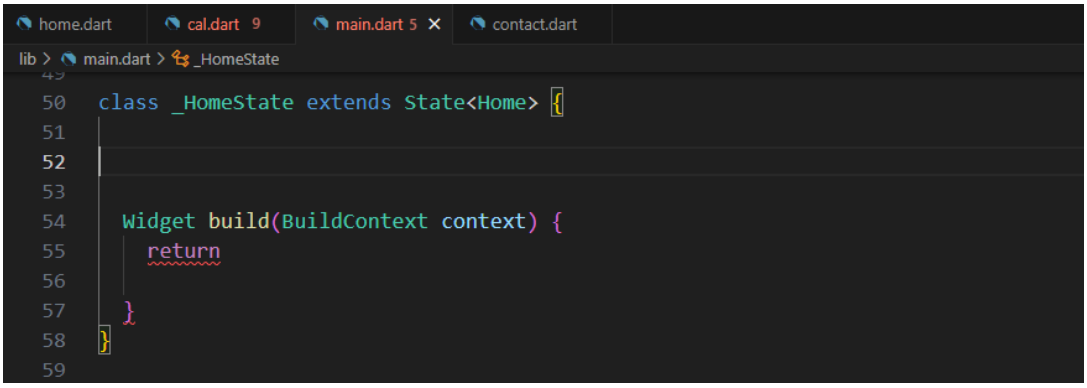
ภาพประกอบ 5.25 ตัดโค้ดส่วนที่เหลือจากไฟล์ main.dart

ที่มา : ณปภัช วรรณตรง (2564 : 11)



ภาพประกอบ 5.26 ตัดโค้ดส่วนที่เหลือจากไฟล์ main.dart (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 11)

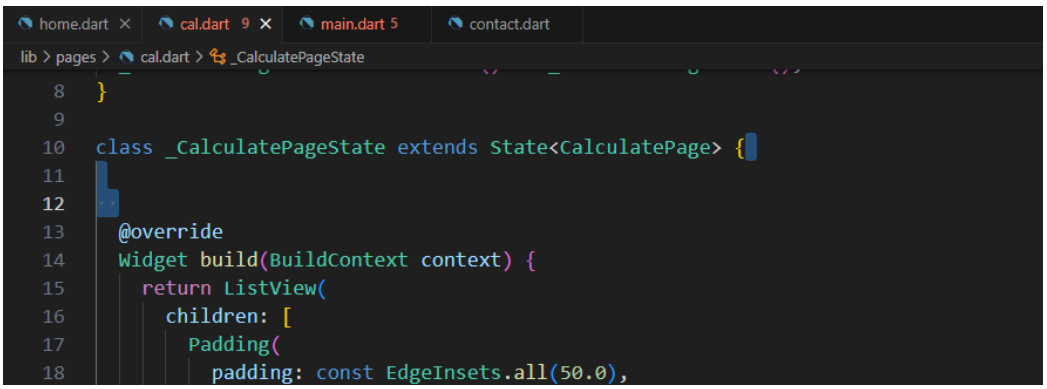


ภาพประกอบ 5.27 ตัดโค้ดส่วนที่เหลือจากไฟล์ main.dart (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 11)

9. นำโค้ดส่วนที่เหลือที่ตัดมาจากไฟล์ main.dart มาใส่ยัง {} ในส่วนของ class State<CalculatePage>{} ตามภาพ

ไปที่ class _CalculatePageState และนำโค้ดที่ได้ทำการคัดลอกมาหรือตัดมาไปวางไว้

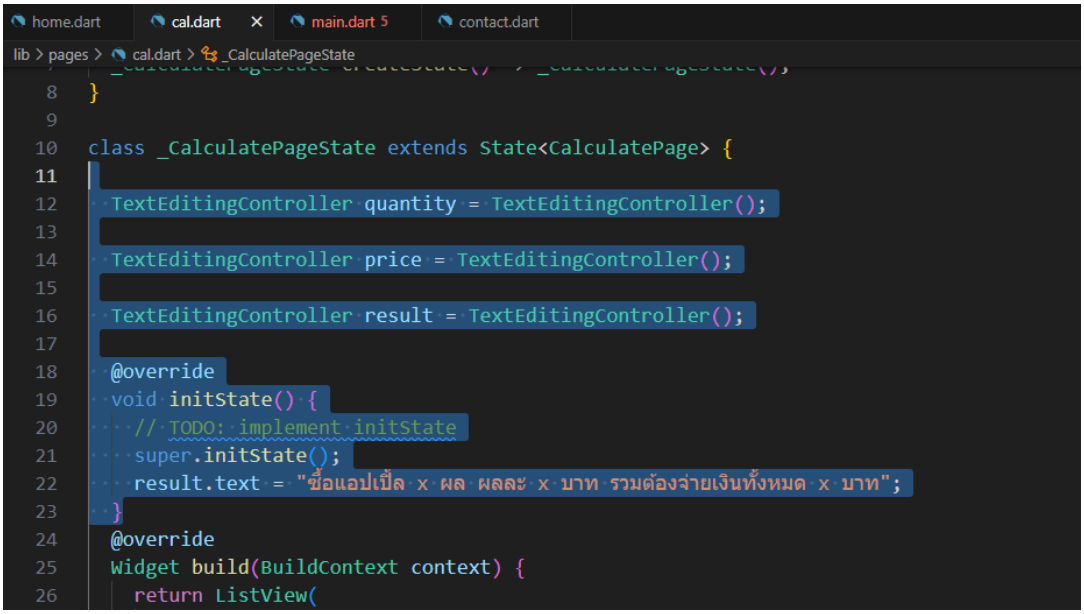


ภาพประกอบ 5.28 นำโค้ดส่วนที่เหลือมาใส่ class State<CalculatePage>{}

ที่มา : ณปภัช วรรณตรง (2564 : 12)

ภาพประกอบ 5.29 นำโค้ดส่วนที่เหลือมาใส่ class State<CalculatePage>{} (ต่อ)

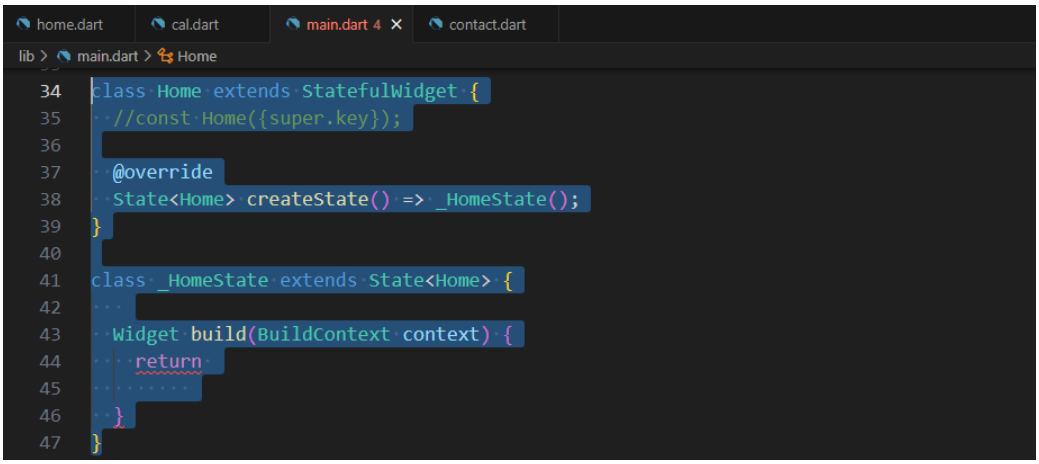
ที่มา : ณปภัช วรรณตรง (2564 : 12)



ภาพประกอบ 5.30 นำโค้ดส่วนที่เหลือมาใส่ class State<CalculatePage>{} (ต่อ)

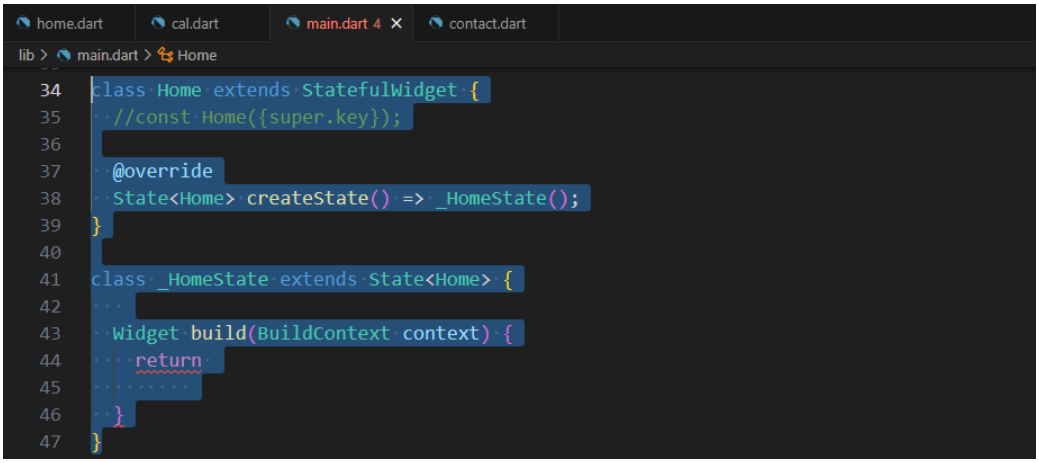
ที่มา : ณปภัช วรรณตรง (2564 : 12)

10. ไฟล์ main.dart ให้ทำการลบโค้ดส่วนของ class Home ออกทั้งหมด



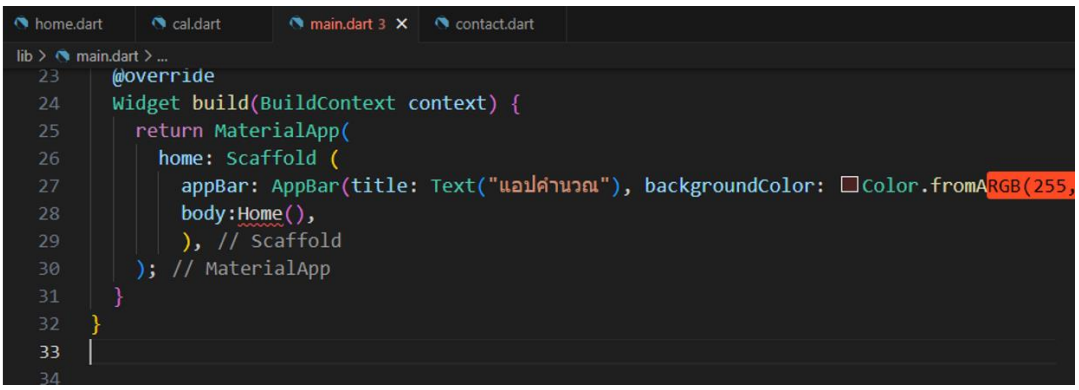
ภาพประกอบ 5.31 ลบโค้ดส่วน class Home

ที่มา : ณปภัช วรรณตรง (2564 : 13)



ภาพประกอบ 5.32 ลบโค้ดส่วน class Home (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 13)



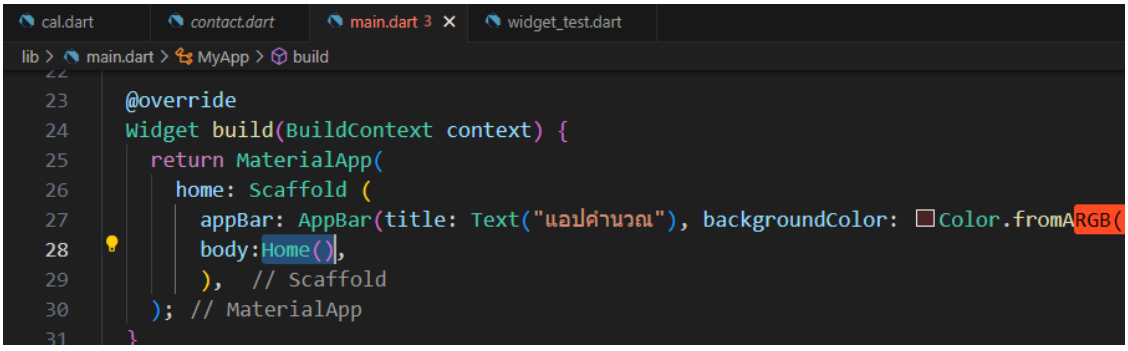
ภาพประกอบ 5.33 ลบโค้ดส่วน class Home (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 13)

11. จากนั้นให้ไปลบโค้ด Home(), ที่ tag body ออก

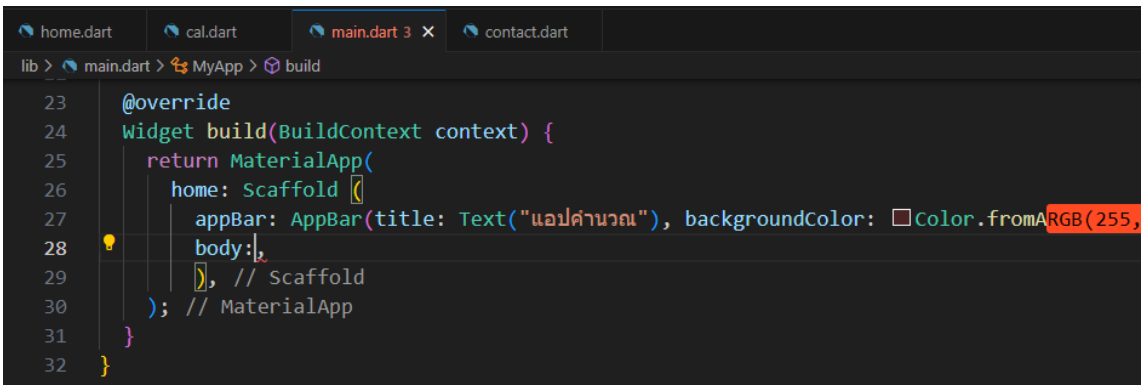
ภาพประกอบ 5.34 ลบโค้ดส่วน Home(),

ที่มา : ณปภัช วรรณตรง (2564 : 14)



ภาพประกอบ 5.35 ลบโค้ดส่วน Home(), (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 14)

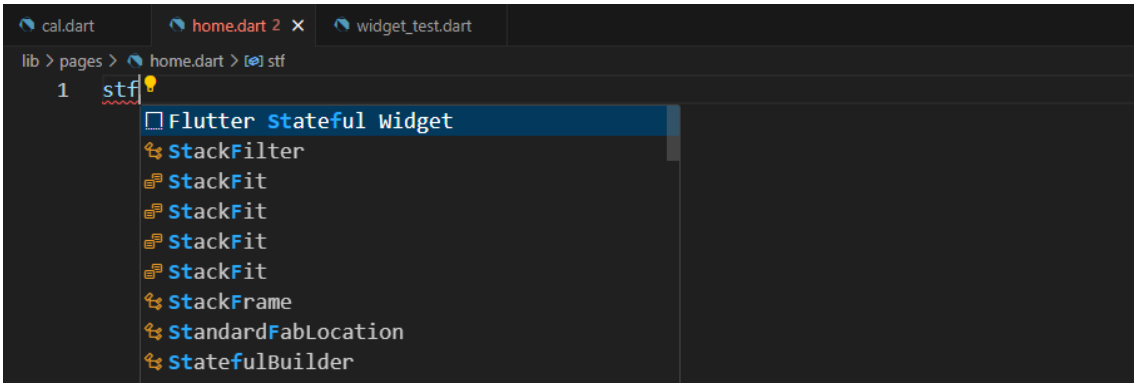


ภาพประกอบ 5.36 ลบโค้ดส่วน Home(), (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 14)

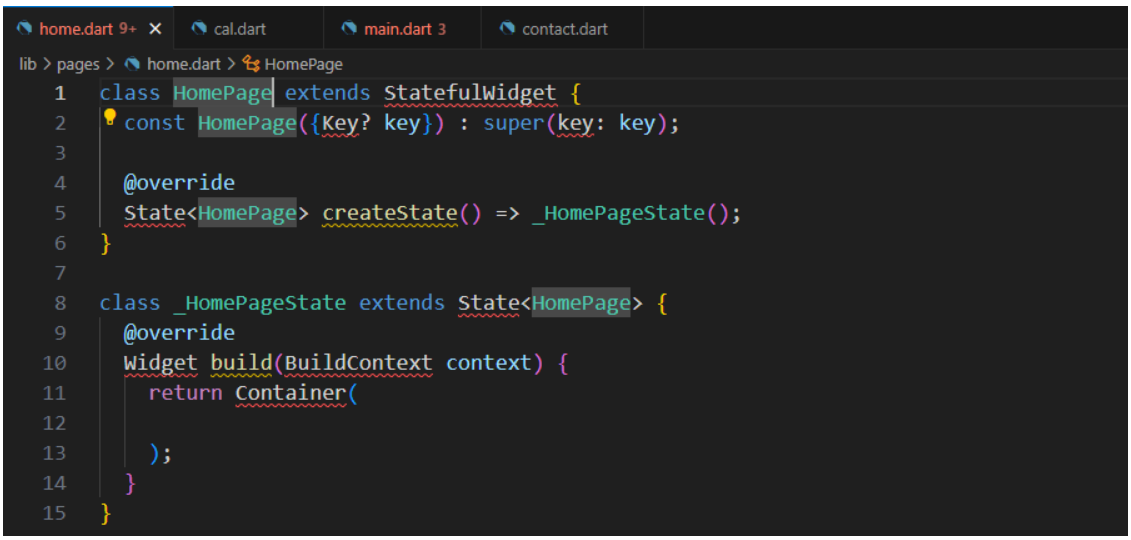
12. เปิดไฟล์ home.dart ทำการสร้าง class StatefulWidget{} ให้ตั้งชื่อ class เป็น HomePage จากนั้นให้ทำการ import package เข้ามา และ comment ส่วน const ใน class ไว้ตามภาพ

12.1 ให้ทำการเพิ่มโค้ด โดยใช้วิธีเดียวกันกับไฟล์ cal.dart แต่ให้ใช้ชื่อ class เป็น HomePage



ภาพประกอบ 5.37 สร้าง class HomePage

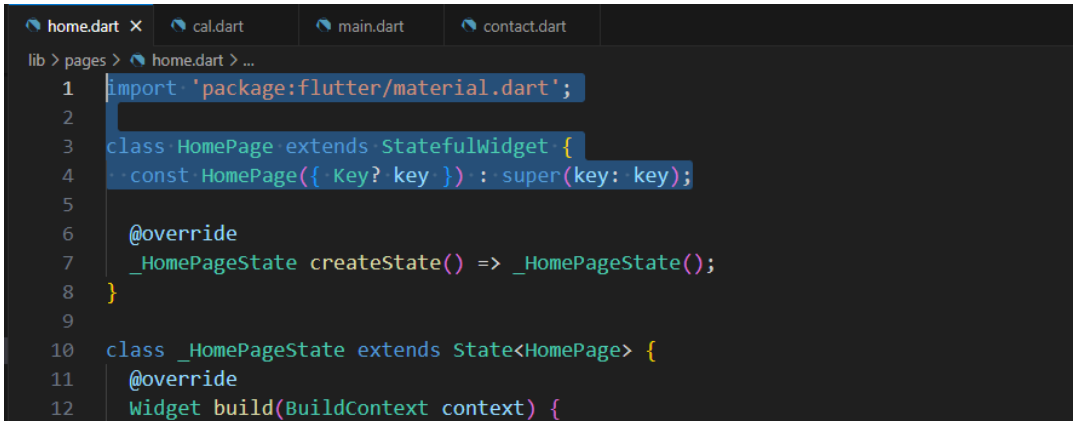
ที่มา : ณปภัช วรรณตรง (2564 : 15)



ภาพประกอบ 5.38 สร้าง class HomePage (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 15)

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

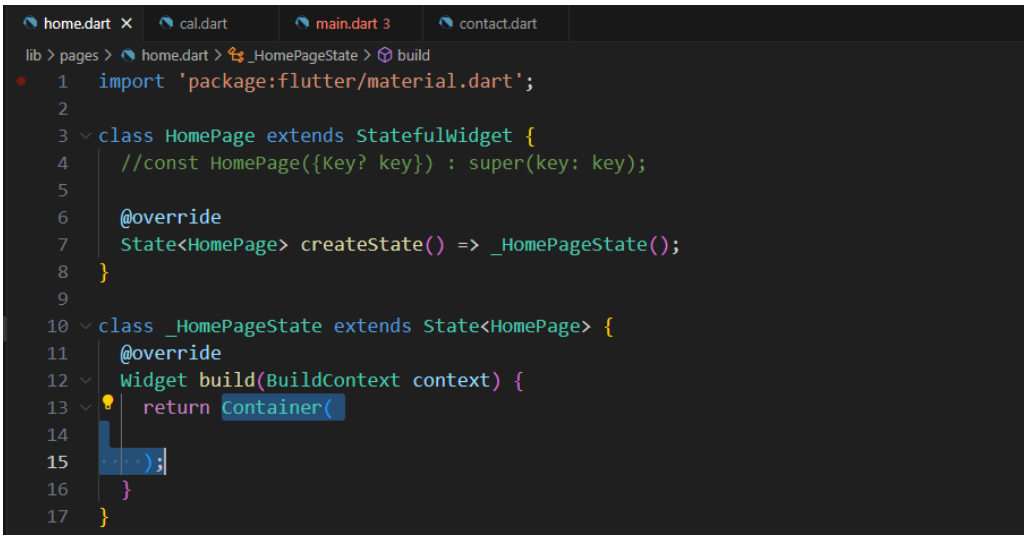


ภาพประกอบ 5.39 สร้าง class HomePage (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 15)

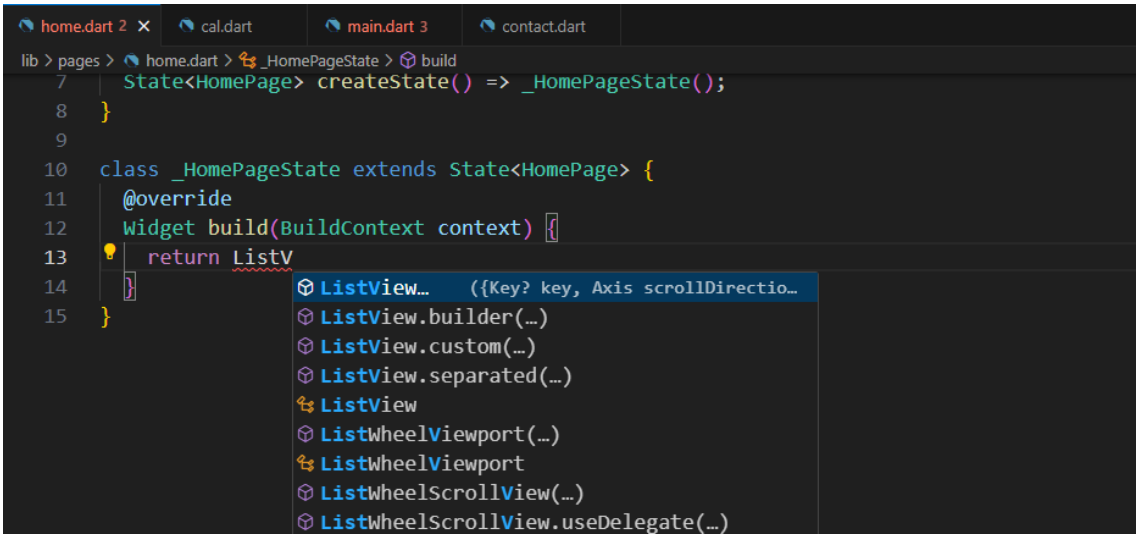
13. แก้ไขโค้ดใน return Container() ให้เป็น ListView() และเพิ่มโค้ด

13.1 ให้ทำการแก้ไขโค้ด Container() เป็น ListView()



ภาพประกอบ 5.40 แก้ไขส่วน return Container()

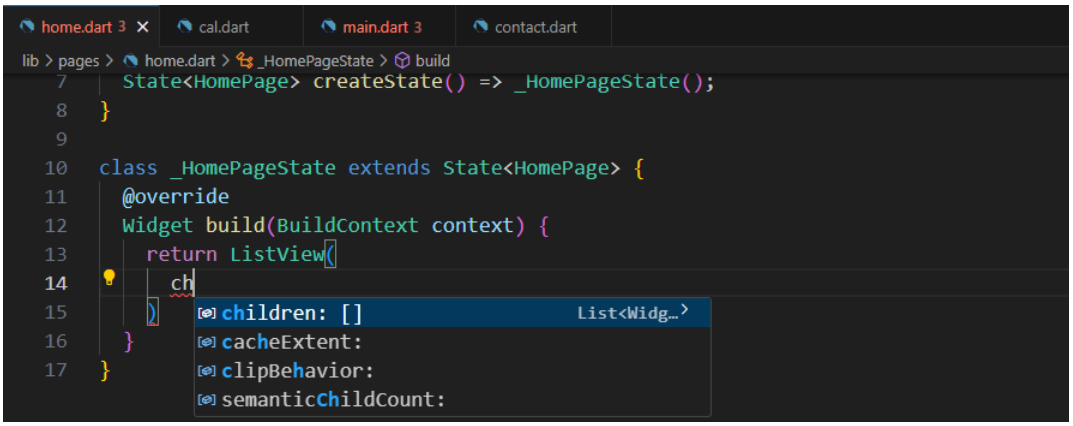
ที่มา : ณปภัช วรรณตรง (2564 : 16)



ภาพประกอบ 5.41 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 16)

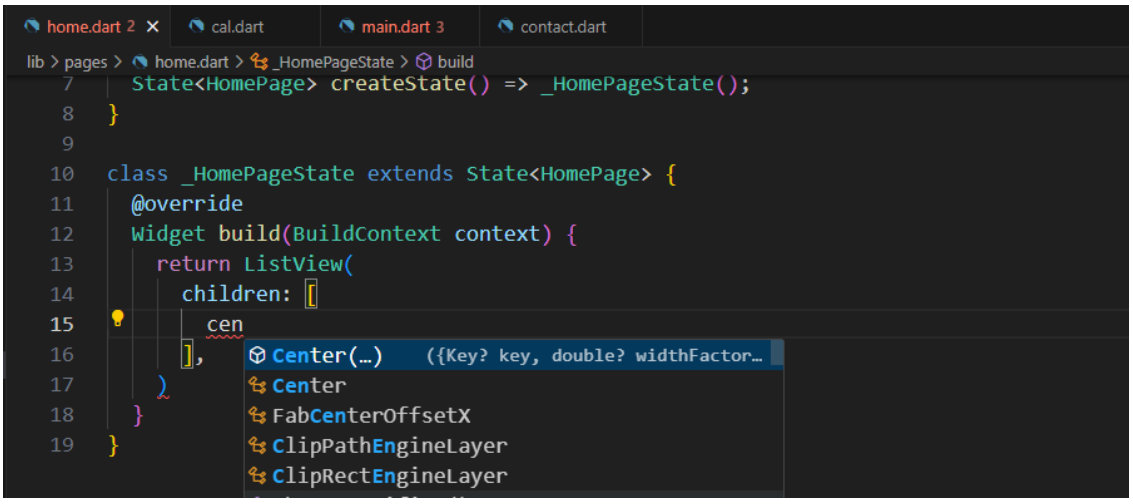
13.2 ภายใน ListViwe(...) ให้ทำการเพิ่มโค้ด children: []



ภาพประกอบ 5.42 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 16)

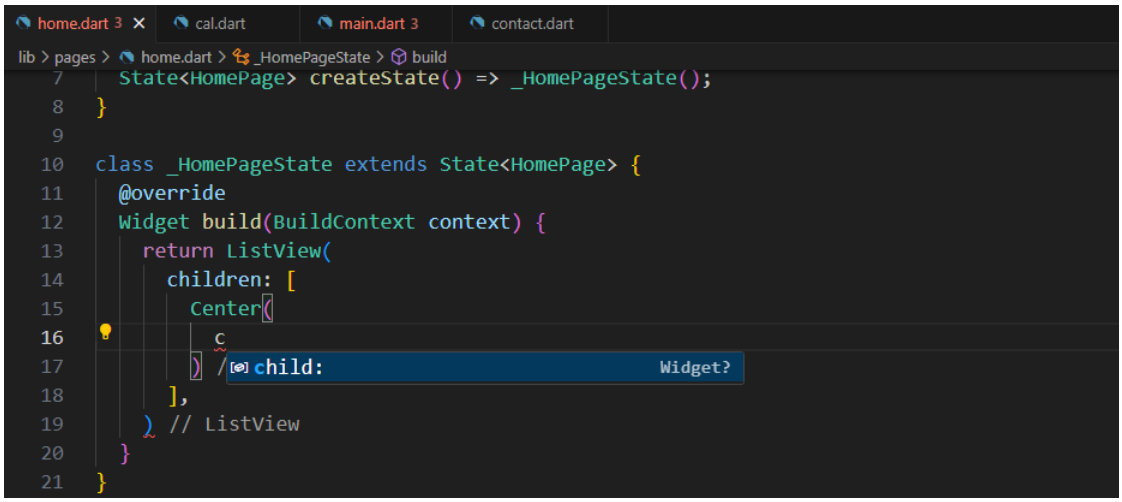
13.3 ภายใน [...] ให้ทำการเพิ่มโค้ด Center(...) เข้าไป



ภาพประกอบ 5.43 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 16)

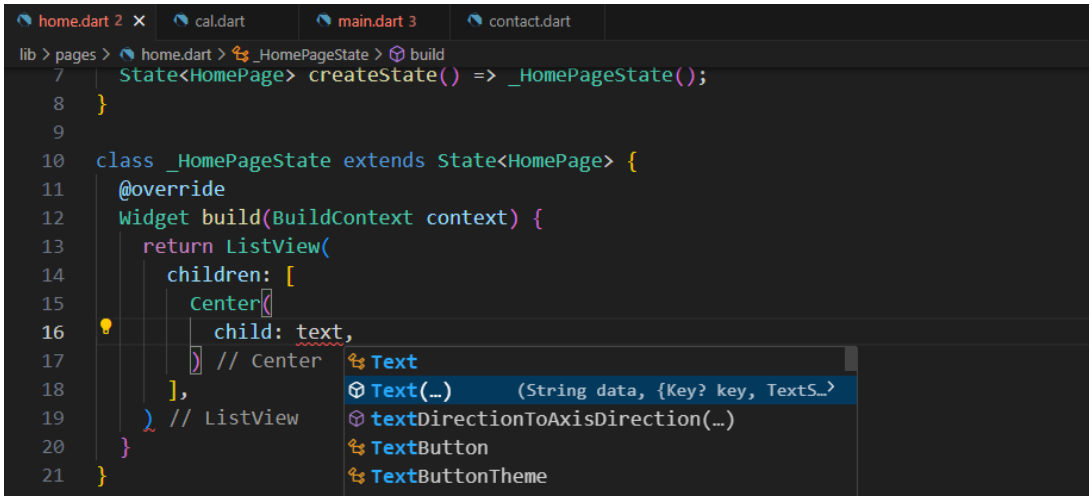
13.4 ภายใน Center(...) ให้ทำการเพิ่ม child:



ภาพประกอบ 5.44 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 16)

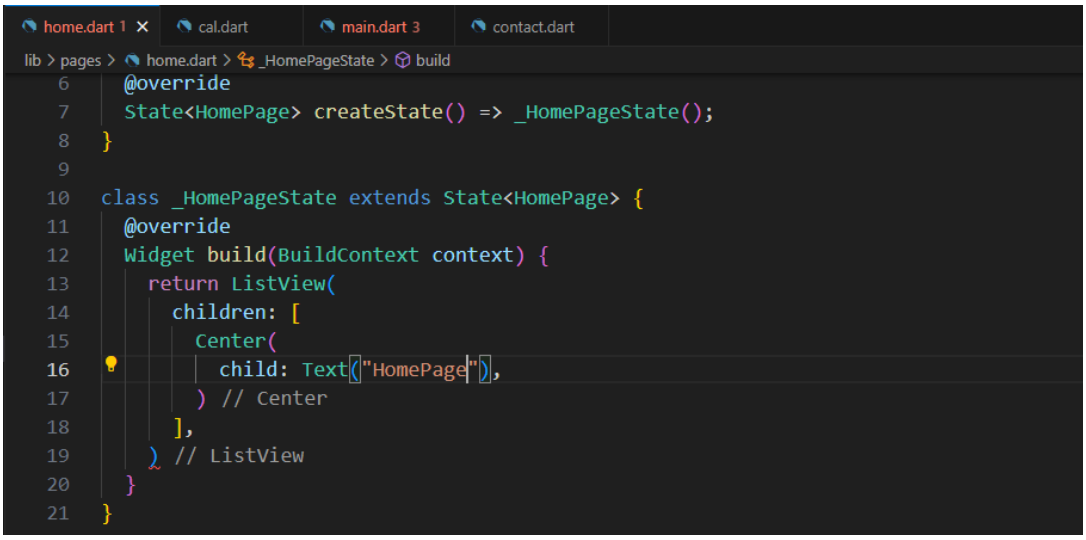
13.5 ภายใน child: ให้ทำการเพิ่มโค้ดเพื่อแสดงข้อความเป็น Text(...)



ภาพประกอบ 5.45 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 16)

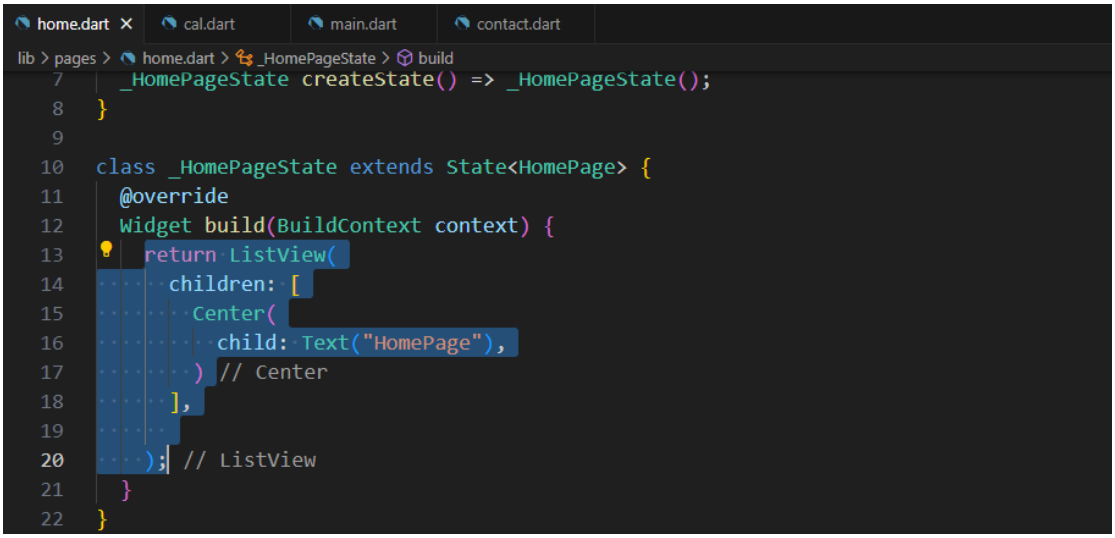
13.6 เมื่อทำการเพิ่มโค้ดเพื่อแสดงข้อความแล้ว ให้กำหนดข้อความที่จะแสดงเข้าไปภายในเครื่องหมาย “ ”



ภาพประกอบ 5.46 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 16)

13.7 โค้ดทั้งหมดที่ได้แก้ไขดังภาพ

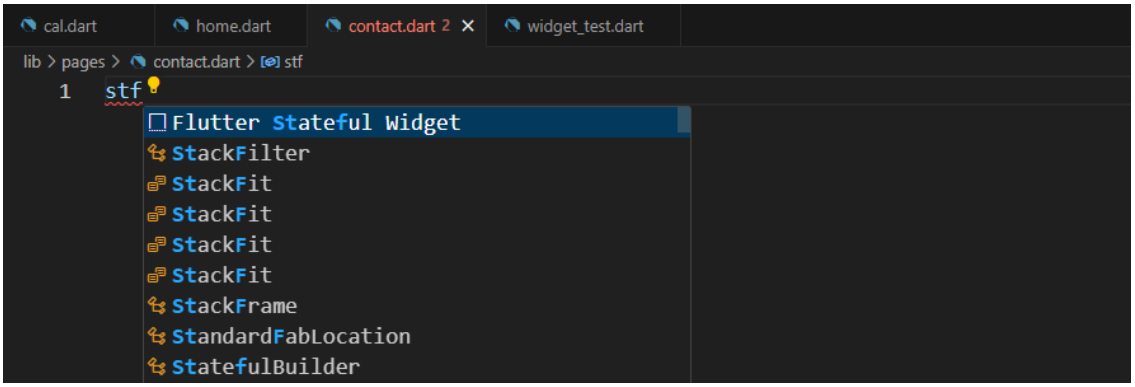


ภาพประกอบ 5.47 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 16)

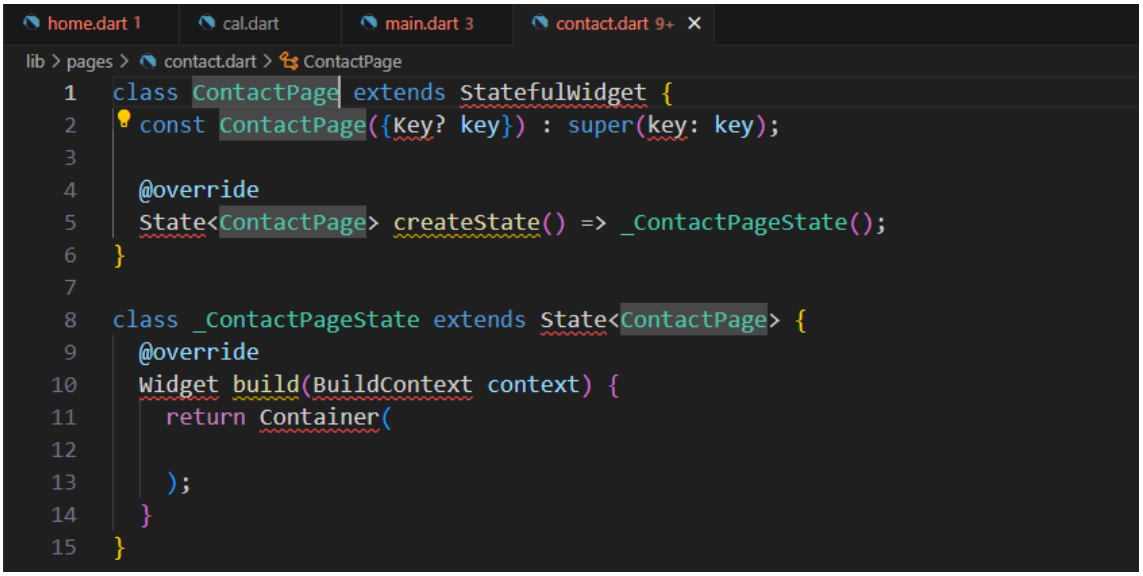
14. ไปที่ไฟล์ contact.dart จากนั้นให้ทำการสร้าง class StatefulWidget{} ตั้งชื่อ class ว่า ContactPage จากนั้นให้ทำการ import package เข้ามา และ comment ส่วน const ใน class ไว้

ให้ทำการเปิดไฟล์ contact.dart จากนั้นให้ทำการเพิ่ม class StatefulWidget{...} โดยใช้ขั้นตอนการเพิ่ม class เหมือนกันกับ cal.dart และ home.dart



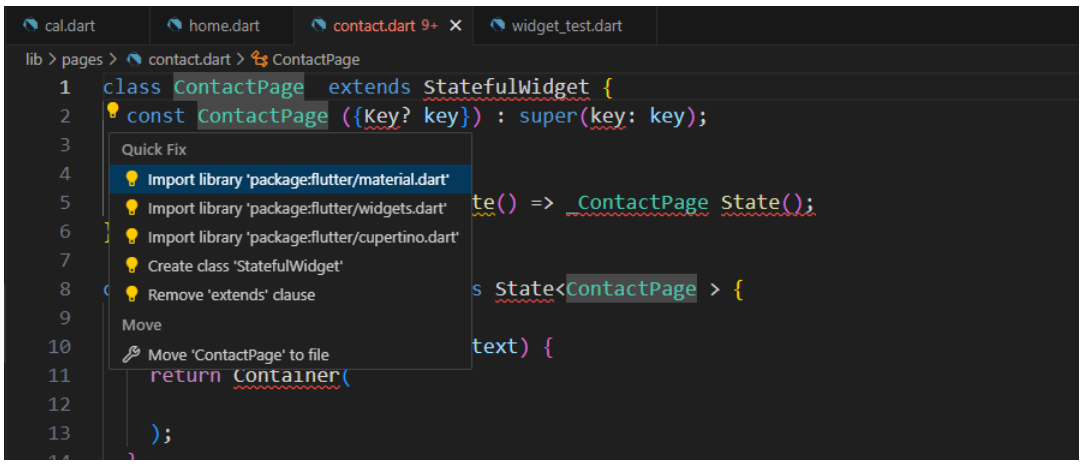
ภาพประกอบ 5.48 สร้าง class ContactPage

ที่มา : ณปภัช วรรณตรง (2564 : 17)



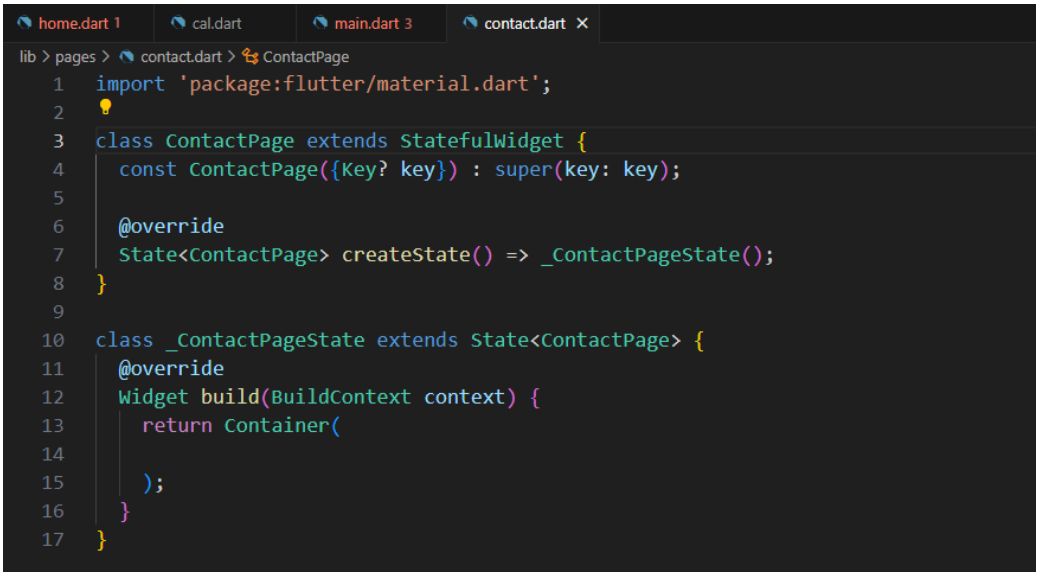
ภาพประกอบ 5.49 สร้าง class ContactPage (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 17)



ภาพประกอบ 5.50 สร้าง class ContactPage (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 17)



ภาพประกอบ 5.51 สร้าง class ContactPage (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 17)

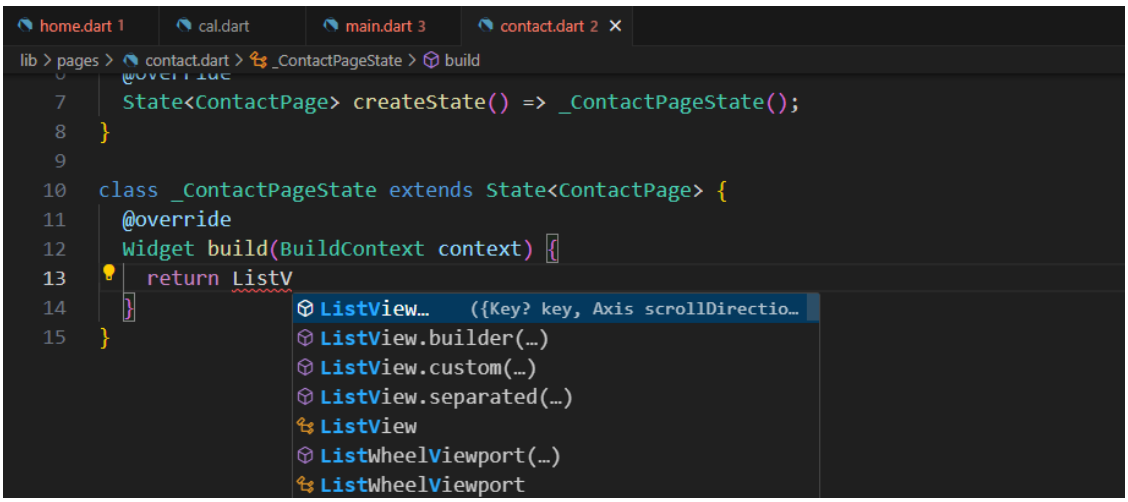
ภาพประกอบ 5.52 สร้าง class ContactPage (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 17)

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

ภาพประกอบ 5.53 แก้ไขส่วน return Container()

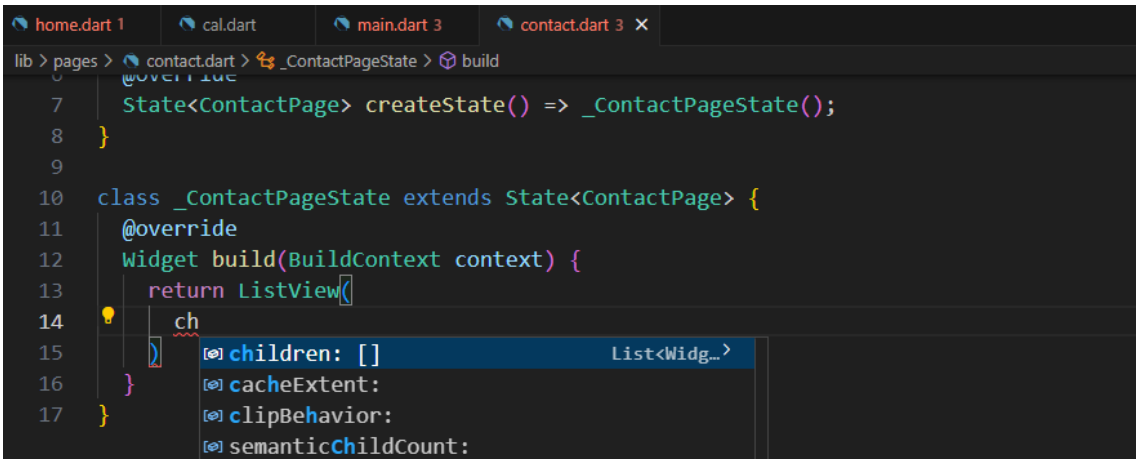
ที่มา : ณปภัช วรรณตรง (2564 : 18)



ภาพประกอบ 5.54 แก้ไขส่วน return Container() (ต่อ)

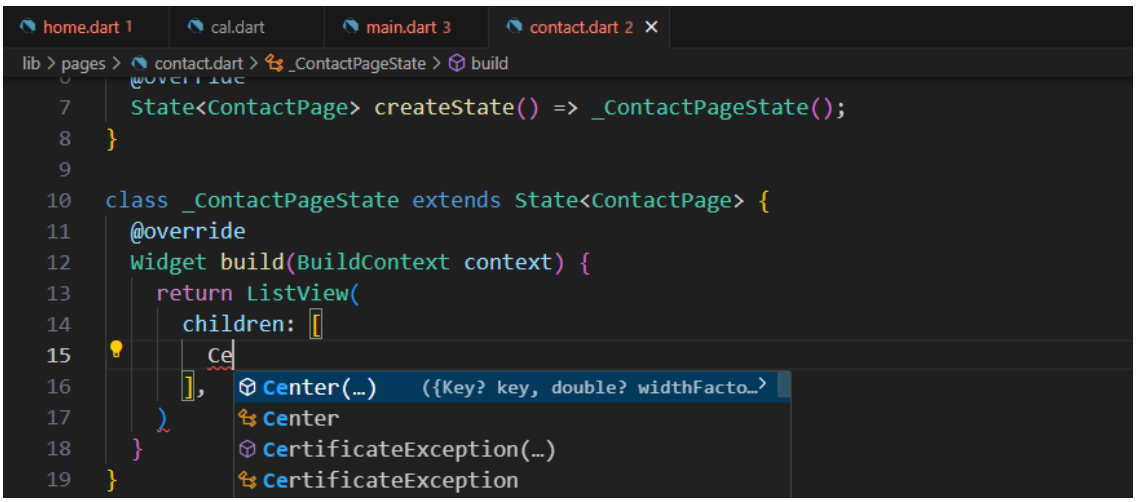
ที่มา : ณปภัช วรรณตรง (2564 : 18)

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



ภาพประกอบ 5.55 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 18)



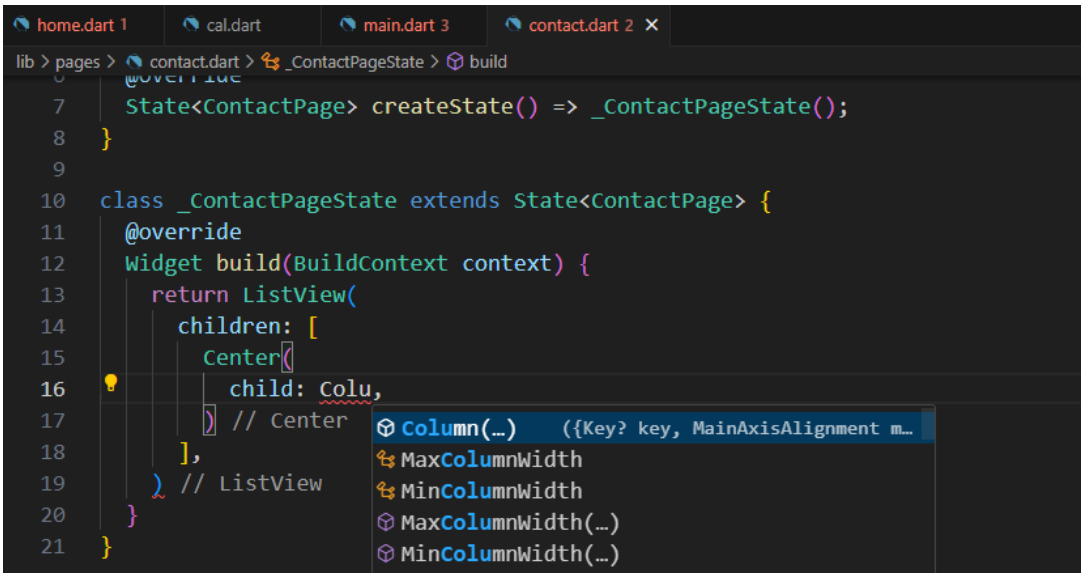
ภาพประกอบ 5.56 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 18)

ภาพประกอบ 5.57 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 18)

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



ภาพประกอบ 5.58 แก้ไขส่วน return Container() (ต่อ)

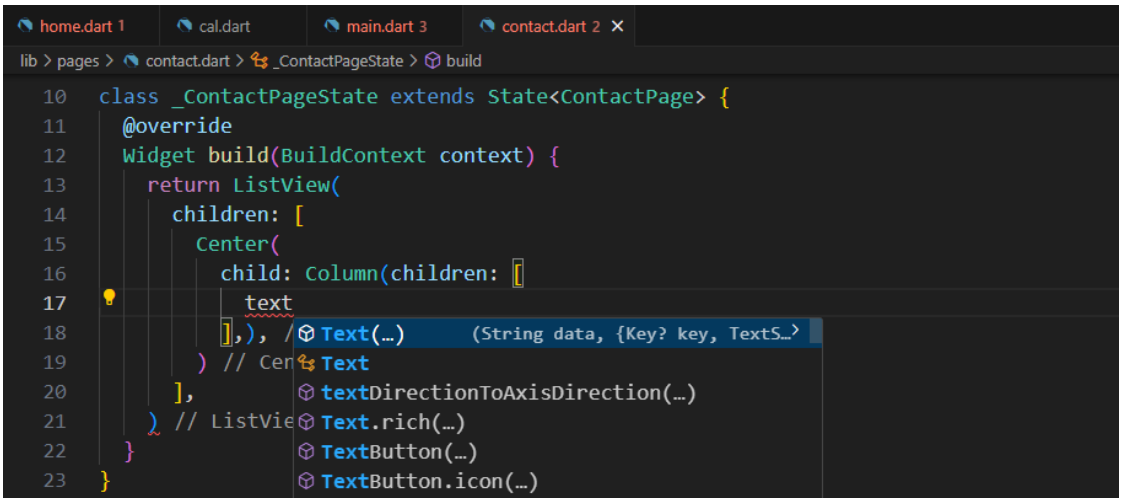
ที่มา : ณปภัช วรรณตรง (2564 : 18)

15.3 ภายใน Column(...) ให้ทำการเพิ่ม children: [...]

ภาพประกอบ 5.59 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 18)

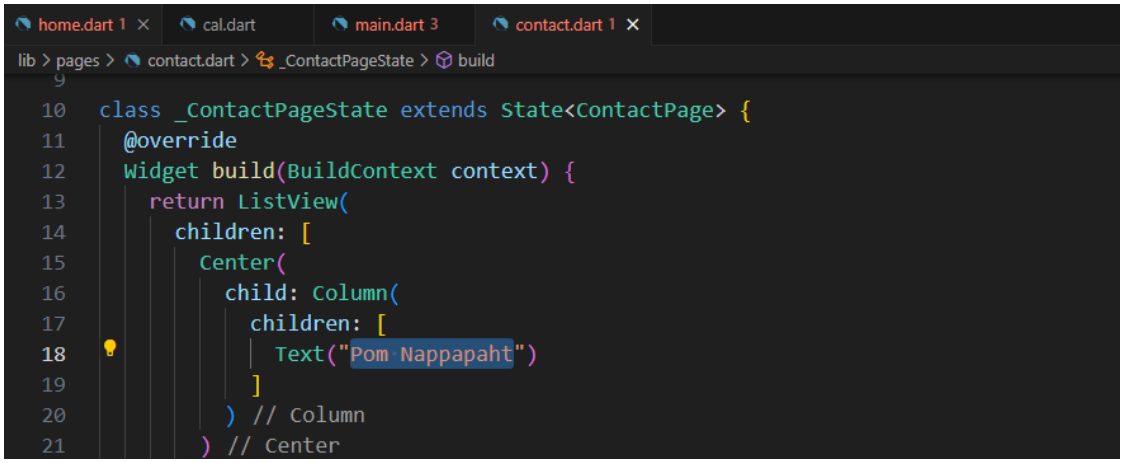
15.4 เมื่อได้ children: [...] แล้ว ให้ทำการเพิ่มโค้ดสำหรับแสดงข้อความ Text(...) เข้าไปใน [...]



ภาพประกอบ 5.60 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 18)

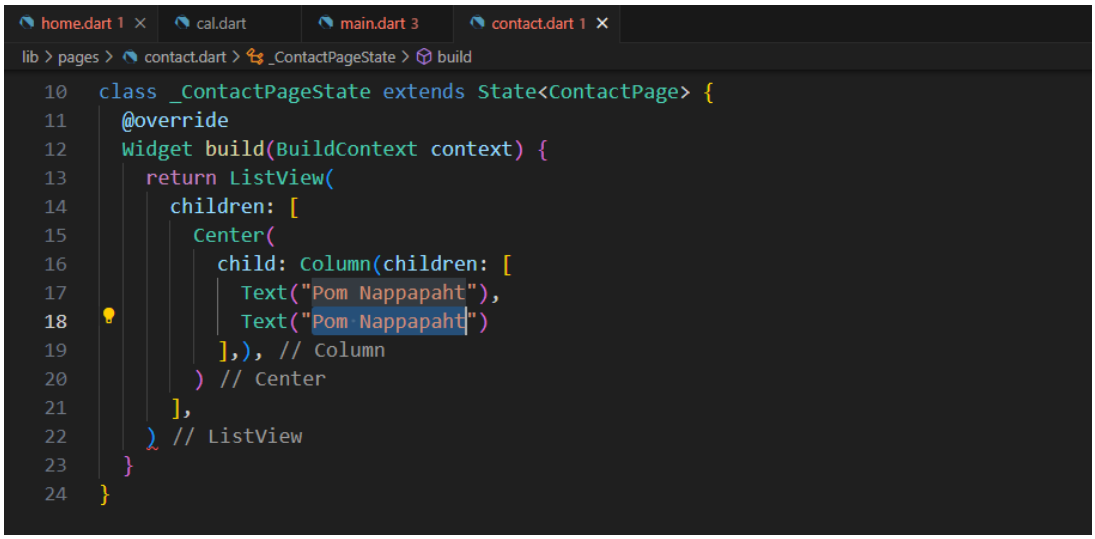
15.5 ภายใน Text(...) ให้ทำการเพิ่มข้อความเข้าไปโดยใช้เครื่องหมาย “” สำหรับเก็บข้อความ



ภาพประกอบ 5.61 แก้ไขส่วน return Container() (ต่อ)

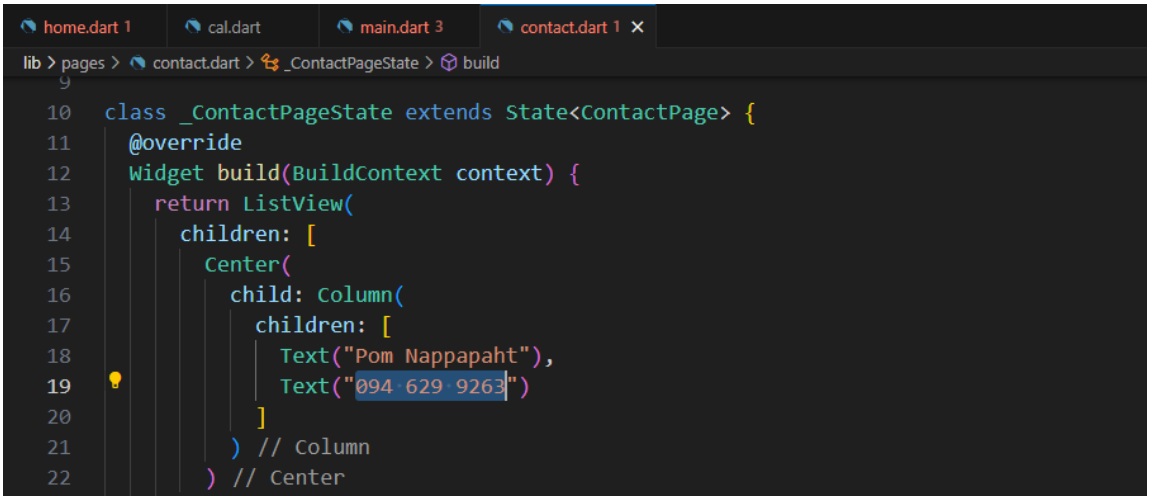
ที่มา : ณปภัช วรรณตรง (2564 : 18)

15.6 สามารถเพิ่ม Text(...) ได้ตามต้องการโดยการคัดลอกวางหรือพิมพ์โค้ดเพิ่มเข้ามาใหม่ จากนั้นให้เปลี่ยนข้อความภายใน “ ”



ภาพประกอบ 5.62 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 18)



ภาพประกอบ 5.63 แก้ไขส่วน return Container() (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 18)

1. BottomNavigationBar

BottomNavigationBar เป็นแถบทูลบาร์ (Toolbar) ที่จะปรากฏอยู่ตรงขอบล่างของหน้าจอ สำหรับกำหนดปุ่มเพื่อเชื่อมโยงหรือเลื่อนไปยังหน้าจอต่าง ๆ ซึ่งเป็นรูปแบบที่นิยมใช้กันเป็นส่วนใหญ่ ในปัจจุบันทั้งบนระบบ iOS และ Android (บัญชา ปะลีละเตสัง, 2566: 339)

การกำหนด BottomNavigationBar วิดเจ็ต เพื่อแสดงยังส่วนล่างของ Scaffold วิดเจ็ตจะต้องกำหนดค่าใน 3 ส่วน ดังต่อไปนี้ (จีราวุธ วารินทร์, 2564 : 288)

1.1 ส่วนแรก คือ ไอเทมต่าง ๆ ที่จะบรรจุไว้ใน BottomNavigationBar ซึ่งแต่ละไอเทมจะสร้างมาจากคลาส BottomNavigationBarItem

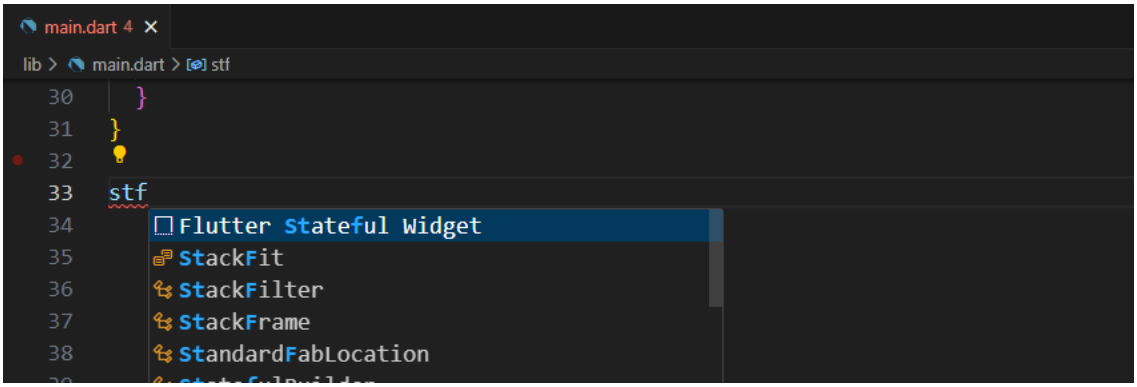
1.2 ส่วนที่สอง คือ ค่า index ที่บอกให้ทราบว่าปัจจุบันเลือกใช้ไอเทมใดอยู่

1.3 ส่วนที่สาม คือ ฟังก์ชันสำหรับติดตามการเลือกไอเทมของผู้ใช้ เช่น เมื่อแตะไอเทมแรก ค่า index ก็จะถูกกำหนดให้เป็น 0 แตะไอเทมที่สองก็มีค่า index เป็น 1 และถ้าแตะที่ไอเทมที่สามก็จะมีค่า index เป็น 2 ตามลำดับ

ในลำดับถัดไปจะทำการสร้างแท็บบาร์ที่ด้านล่างของแอปพลิเคชันเพื่อสามารถคลิกไปยังหน้าแอปพลิเคชันแต่ละหน้าที่สร้างไว้ โดยมีรายละเอียดขั้นตอนการทำดังนี้

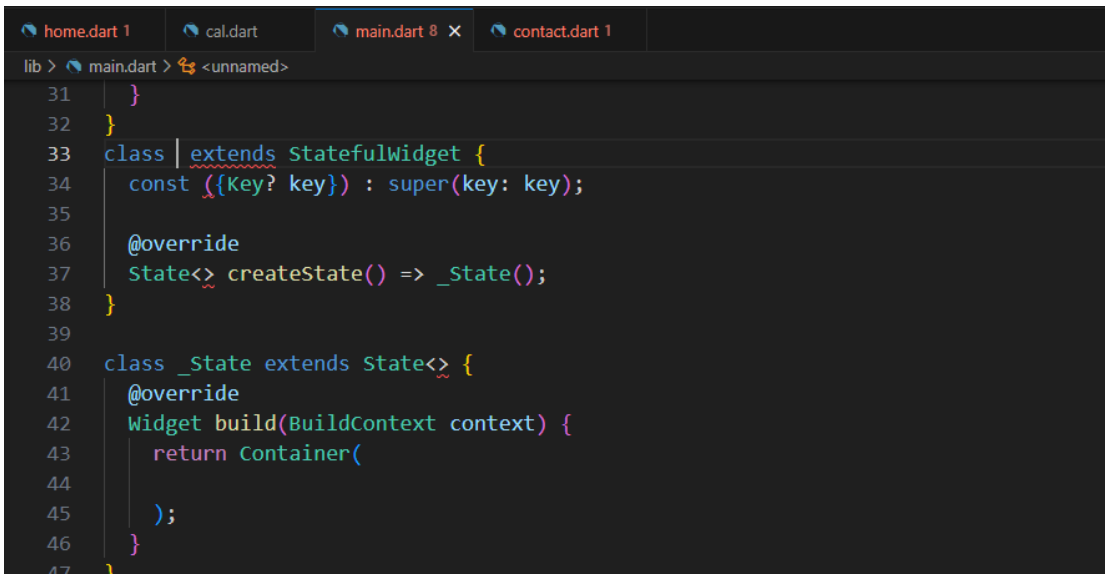
2. การสร้างแท็บบาร์ที่ด้านล่างของแอปพลิเคชัน

2.1 เปิดไฟล์ main.dart จากนั้นให้ทำการสร้าง class StatefulWidget{} ตั้งชื่อ class และ const ว่า MainPage



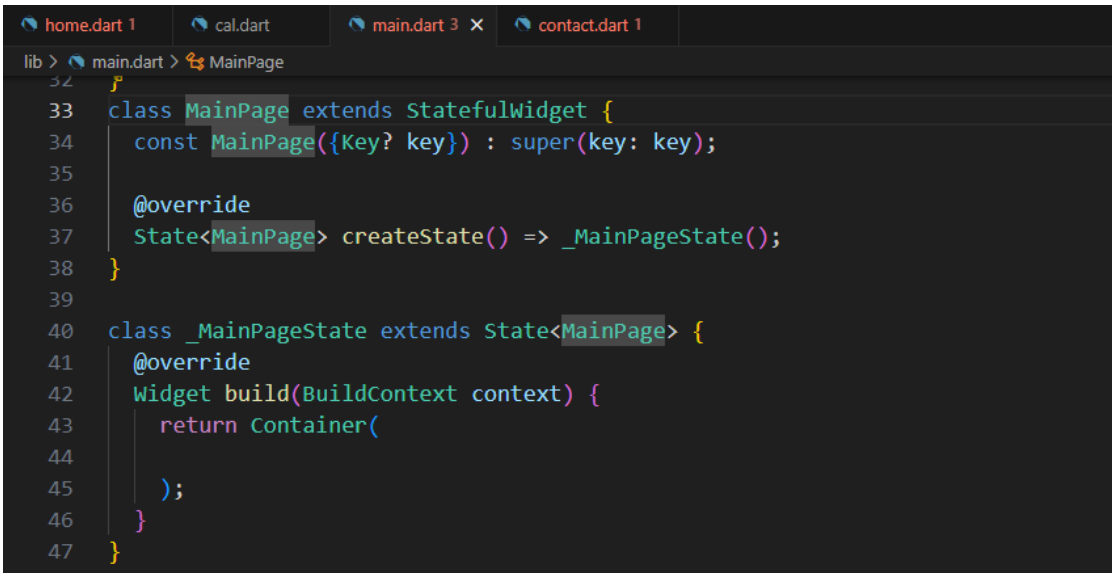
ภาพประกอบ 5.64 สร้าง class MainPage

ที่มา : ณปภัช วรรณตรง (2564 : 20)



ภาพประกอบ 5.65 สร้าง class MainPage (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 20)



ภาพประกอบ 5.66 สร้าง class MainPage (ต่อ)

ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : 20)

ให้ทำการปิดโค้ดที่ไม่ได้ใช้งานด้วยการ Comment โค้ดบรรทัดนั้น ๆ

main.dart 1 ×

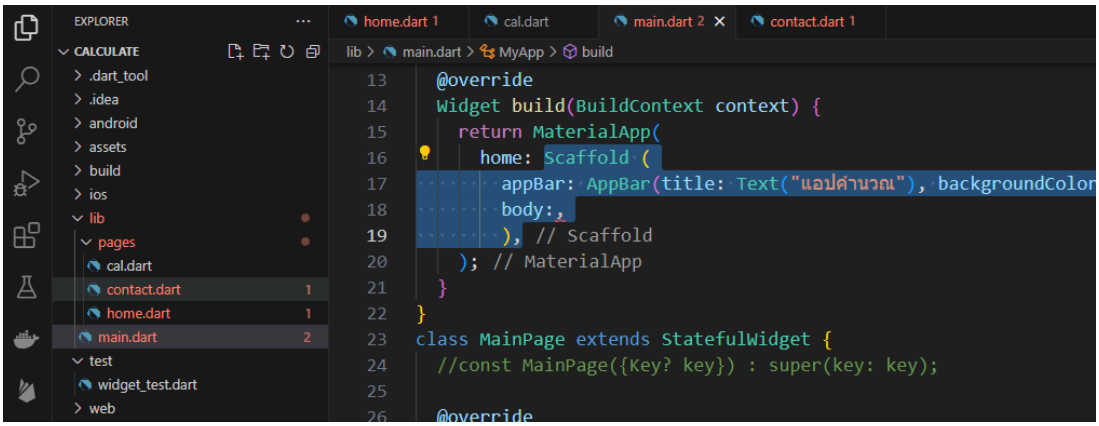
lib > main.dart > MainPage

```
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 : ณปภัช วรรณตรง (2564 : 20)

2.2 ในไฟล์ main.dart ให้ทำการตัดส่วน Scaffold() ใน tag home: ออกทั้งหมดตามภาพ ไปที่ MaterialApp(...) จากนั้นให้ทำการตัดโค้ดส่วน Scaffold(...) ไว้



ภาพประกอบ 5.69 ตัดส่วน Scaffold()

ที่มา : ณปภัช วรรณตรง (2564 : 21)

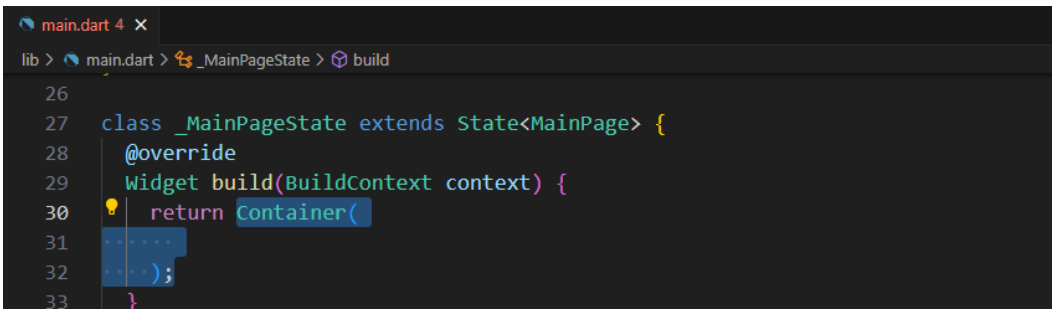
ภาพประกอบ 5.70 ตัดส่วน Scaffold() (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 21)

ภาพประกอบ 5.71 ตัดส่วน Scaffold() (ต่อ)

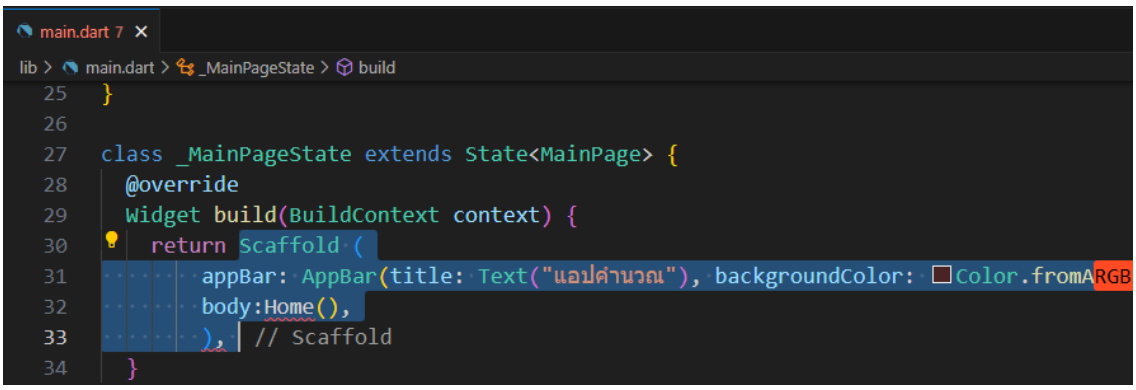
ที่มา : ณปภัช วรรณตรง (2564 : 21)

2.3 นำโค้ดที่ทำการตัดมา มาใส่แทนส่วน return Container(); ใน class MainPage ไปที่ class _MainPageState จากนั้นให้ทำการวางโค้ดที่ได้ทำการตัด เข้าไปแทนที่ Container(...)



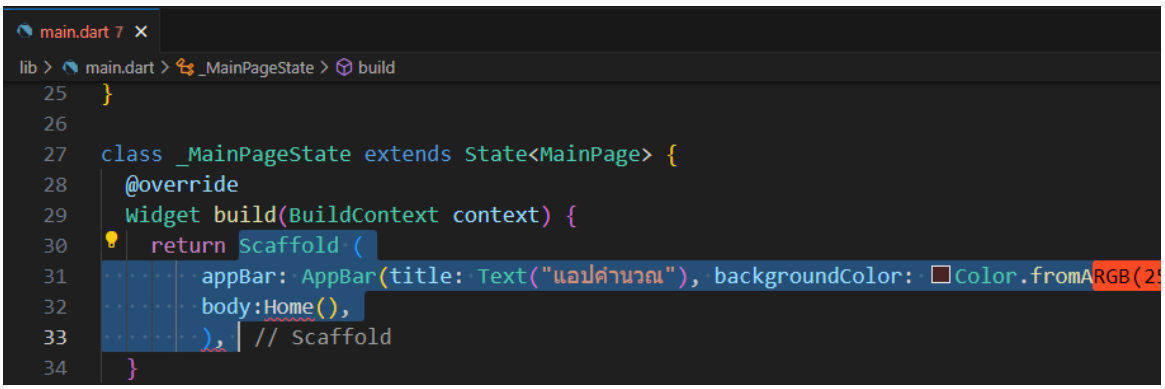
ภาพประกอบ 5.72 นำส่วน Scaffold() มาใส่แทน Container()

ที่มา : ณปภัช วรรณตรง (2564 : 22)



ภาพประกอบ 5.73 นำส่วน Scaffold() มาใส่แทน Container() (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 22)

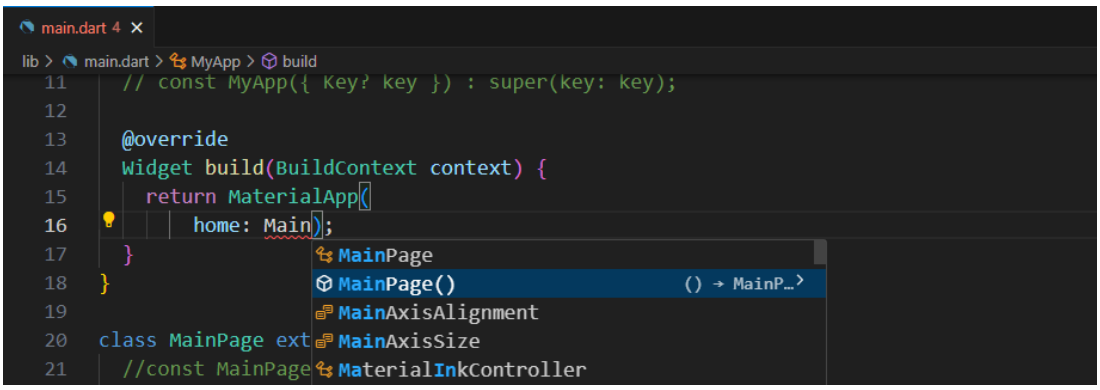


ภาพประกอบ 5.74 นำส่วน Scaffold() มาใส่แทน Container() (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 22)

2.4 ในส่วนโค้ด tag home: ให้ทำการใส่ MainPage() เข้าไปเพื่อทำการเรียกใช้งาน class MainPage ที่ได้ทำการเขียนไว้ ตามภาพ

ไปที่ MaterialApp(...) ต่อจาก home: ให้ทำการเพิ่มโค้ด MainPage() เพื่อกำหนดหน้าเริ่มต้นของแอปพลิเคชัน



ภาพประกอบ 5.75 ใส่ MainPage() ในส่วน tag home:

ที่มา : ณปภัช วรรณตรง (2564 : 23)

ภาพประกอบ 5.76 ใส่ MainPage() ในส่วน tag home: (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 23)

ภาพประกอบ 5.77 ใส่ MainPage() ในส่วน tag home: (ต่อ)

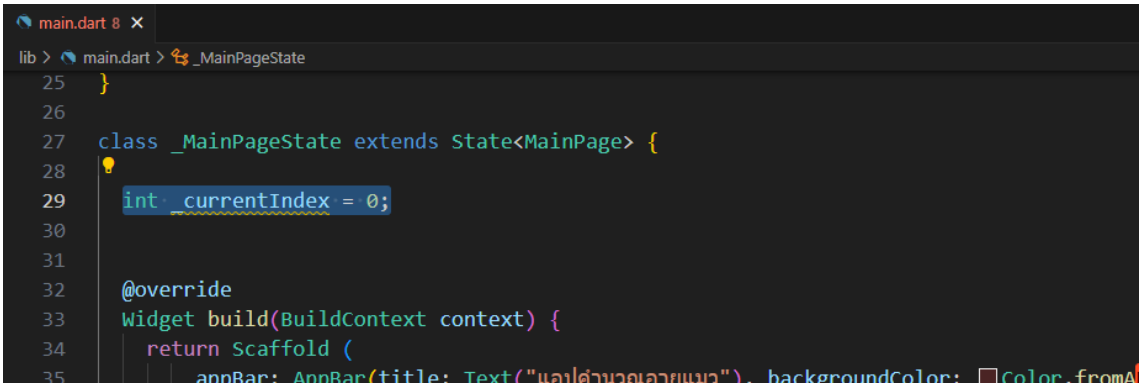
ที่มา : ณปภัช วรรณตรง (2564 : 23)

2.5 ทำการเพิ่มโค้ด int currrentIndex = 0; และ final tabs = [HomePage(), CalculatePage(), ContactPage()]; เข้าไปใน class MainPage

2.5.1 ไปที่ MainPageState ภายใน {...} ให้ทำการเพิ่มโค้ดเพื่อประกาศตัวแปรสำหรับรับค่าหน้าที่ผู้ใช้เปิดอยู่

ภาพประกอบ 5.78 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage

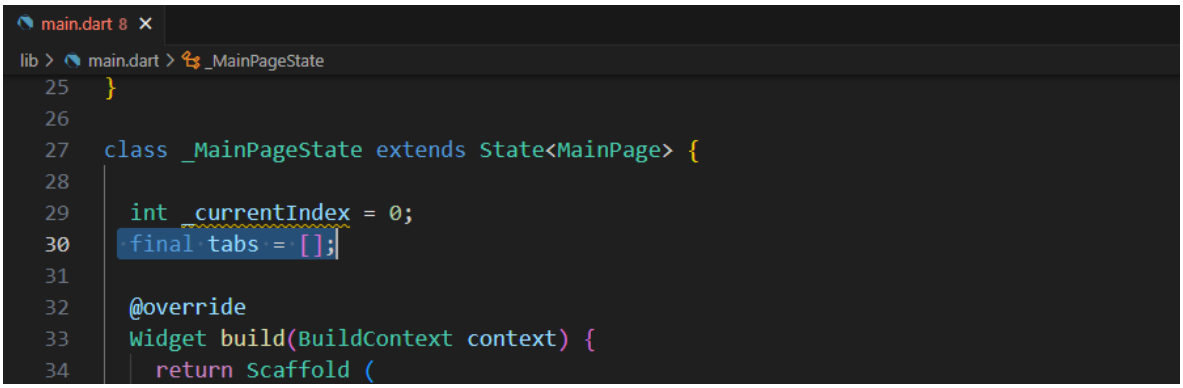
ที่มา : ณปภัช วรรณตรง (2564 : 24)



ภาพประกอบ 5.79 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ)

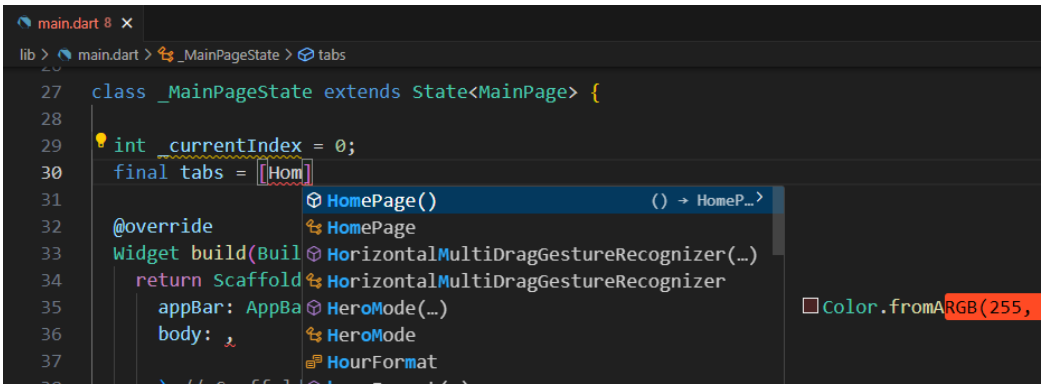
ที่มา : ณปภัช วรรณตรง (2564 : 24)

2.5.2 เพิ่มตัวแปรที่ไม่สามารถเปลี่ยนแปลงได้สำหรับเก็บหน้าเว็บแต่ละหน้า โดยพิมพ์ final tabs = [หน้าเว็บที่ต้องการเก็บ]

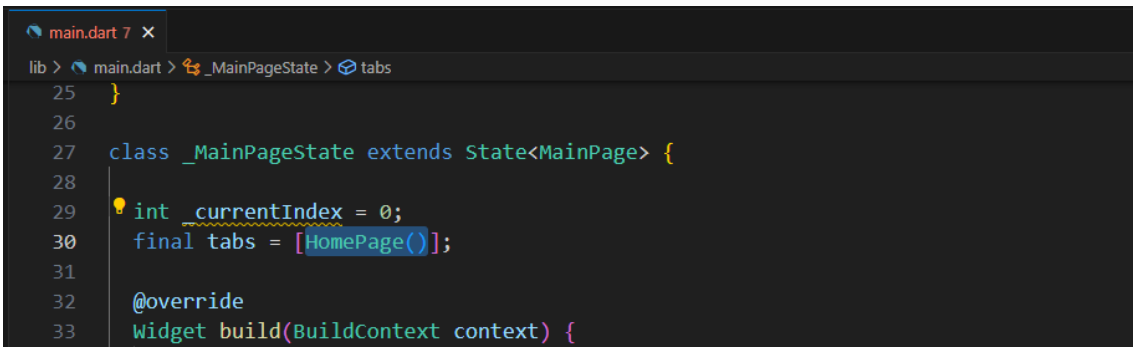


ภาพประกอบ 5.80 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 24)

2.5.3 ภายใน [...] ให้ทำการพิมพ์ชื่อ class ของแต่ละหน้าเว็บที่ต้องการจะเรียกใช้เข้าไป โดยที่เมื่อพิมพ์โค้ดไปบางส่วนแล้วจะมีหน้าต่างชุดคำสั่งขึ้นมาให้เลือก เมื่อเลือกแล้วจะทำการ import ไฟล์นั้น ๆ เข้ามาให้ในไฟล์ main.dart

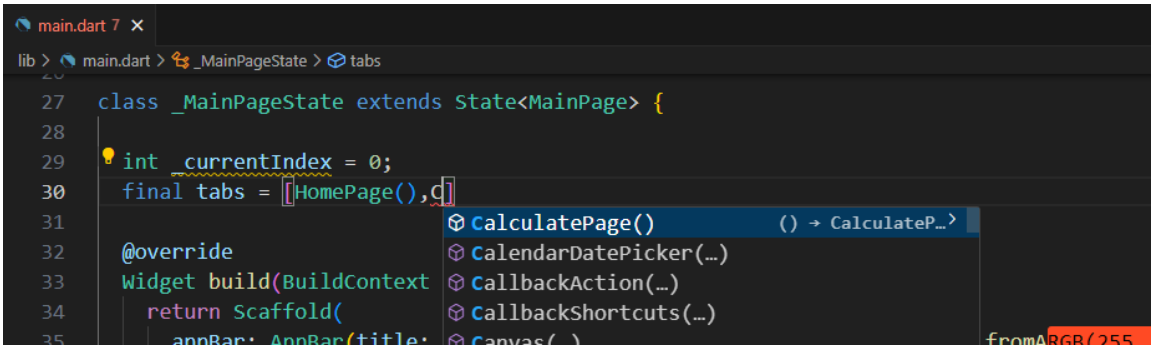


ภาพประกอบ 5.81 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 24)



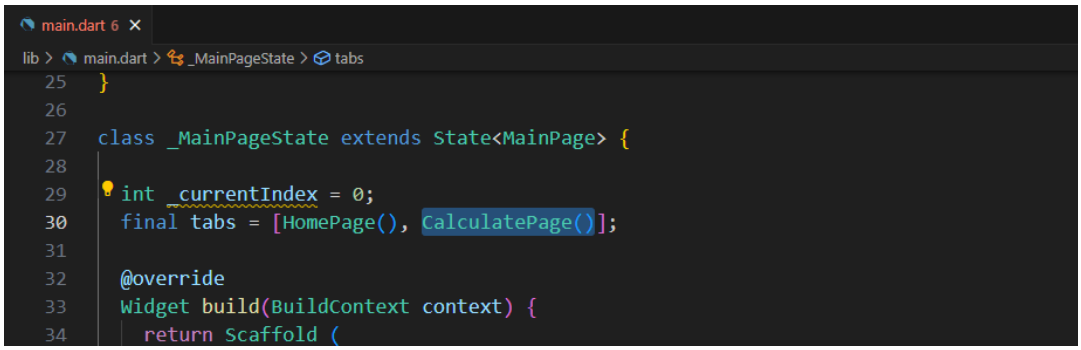
ภาพประกอบ 5.82 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 24)



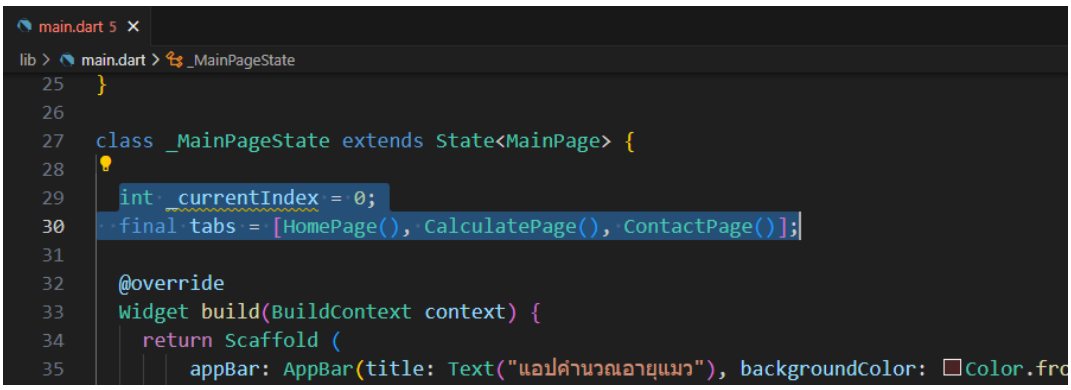
ภาพประกอบ 5.83 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 24)



ภาพประกอบ 5.84 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 24)

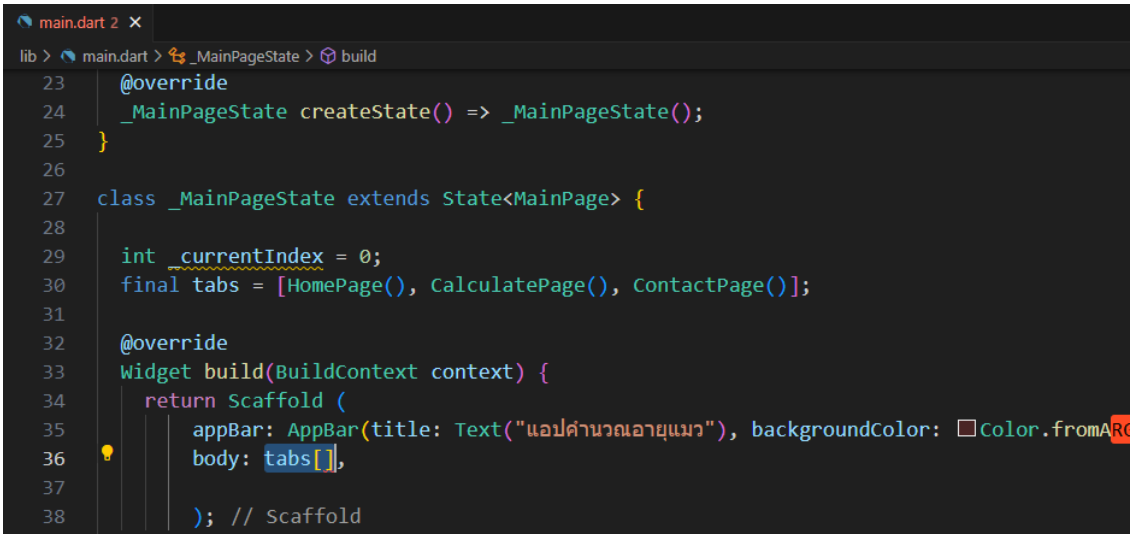


ภาพประกอบ 5.85 ทำการเพิ่มโค้ดตัวแปรรับค่าเข้าไปใน class MainPage (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 24)

2.6 ใน tag body: ให้ทำการเพิ่มโค้ดลงไปตามภาพ

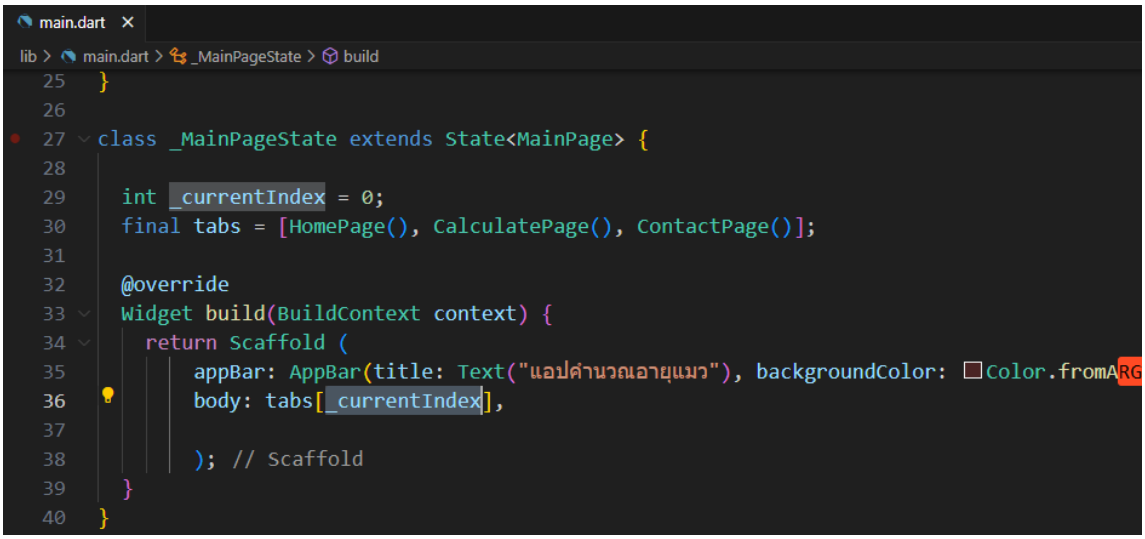
2.6.1 ภายใน Scaffold(...) ต่อจาก body: ให้ทำการเพิ่มโค้ด tabs[...] เพื่อนำค่าตัวแปรที่ได้กำหนดมาใช้งาน



ภาพประกอบ 5.86 เพิ่มโค้ดในส่วนของ tag body:

ที่มา : ณปภัช วรรณตรง (2564 : 25)

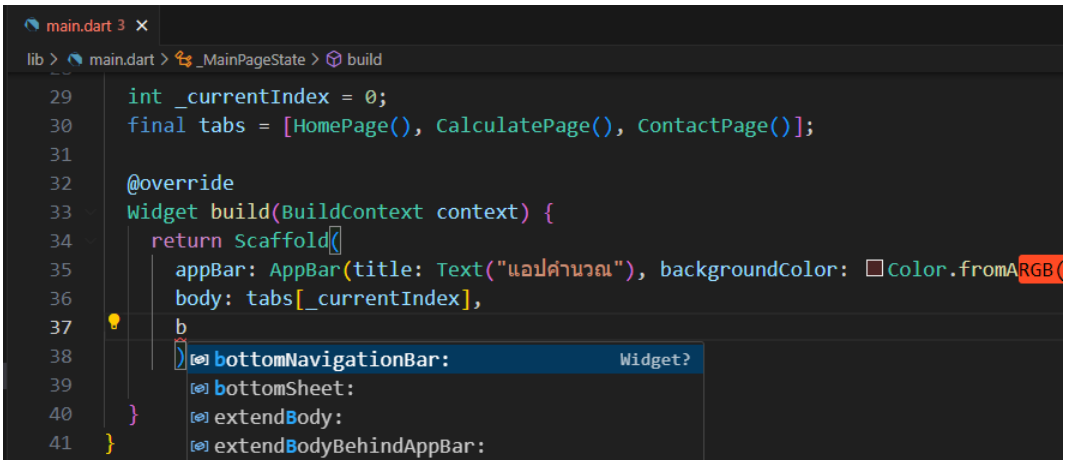
2.6.2 ภายใน tabs[...] ให้ทำการพิมพ์ตัวแปรที่กำหนดไว้มาใส่ โดยการพิมพ์ _currentIndex



ภาพประกอบ 5.87 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 25)

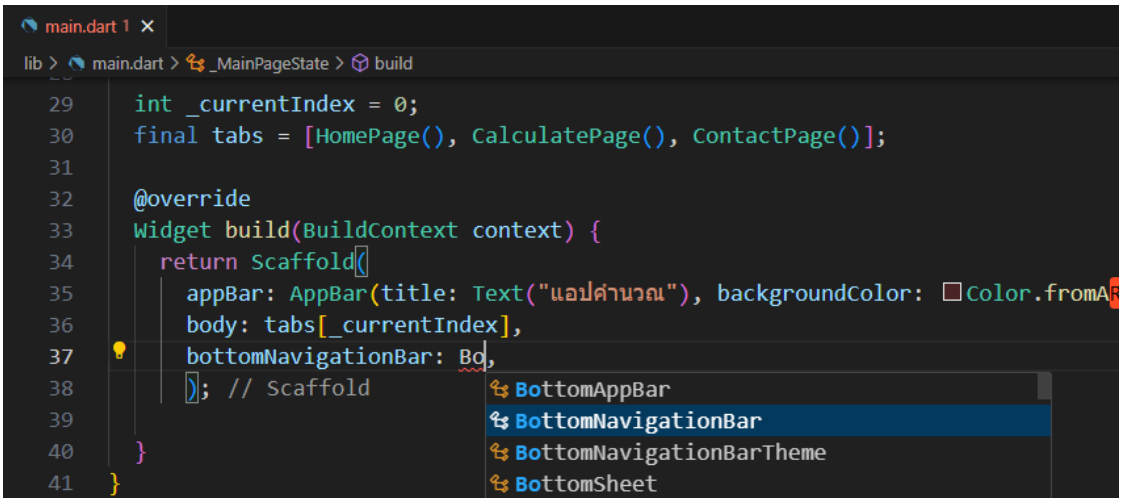
2.6.3 จากนั้นให้ทำการเพิ่มโค้ด bottomNavigationBar: ซึ่งเป็นโค้ดส่วนสำหรับการสร้างแท็บบาร์ด้านล่างแอปพลิเคชัน



ภาพประกอบ 5.88 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 25)

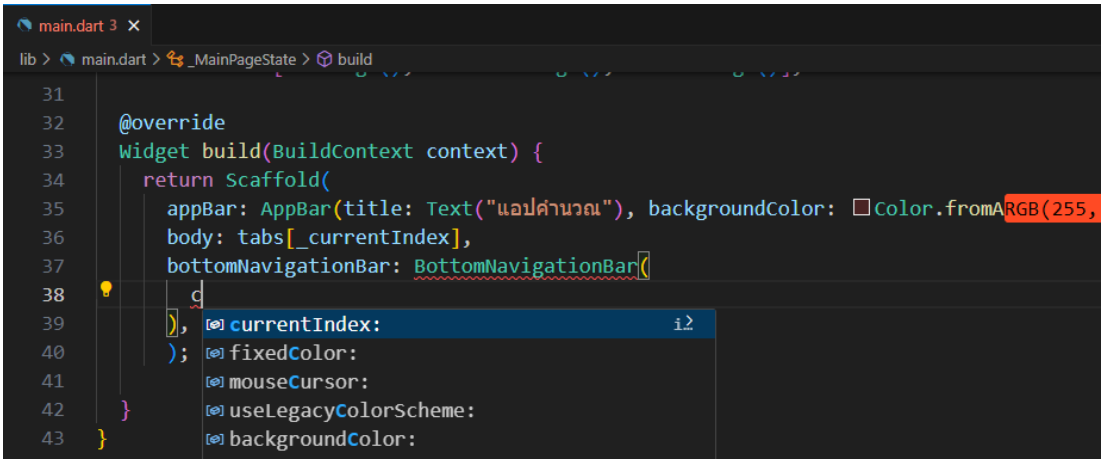
2.6.4 ต่อจาก bottomNavigationBar: ให้ทำการเพิ่ม BottomNavigationBar(...)



ภาพประกอบ 5.89 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

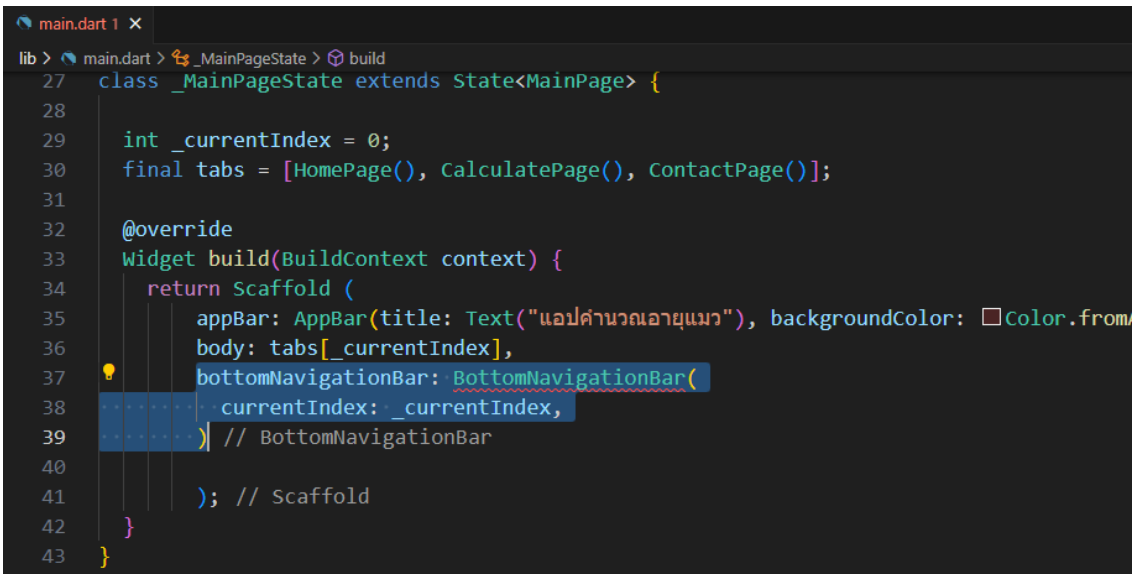
ที่มา : ณปภัช วรรณตรง (2564 : 25)

2.6.5 ภายใน BottomNavigationBar(...) ให้ทำการเพิ่มโค้ดเข้าไป โดยส่วนแรกจะเป็นการประกาศเพื่อเรียกตัวแปรมาใช้งาน



ภาพประกอบ 5.90 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

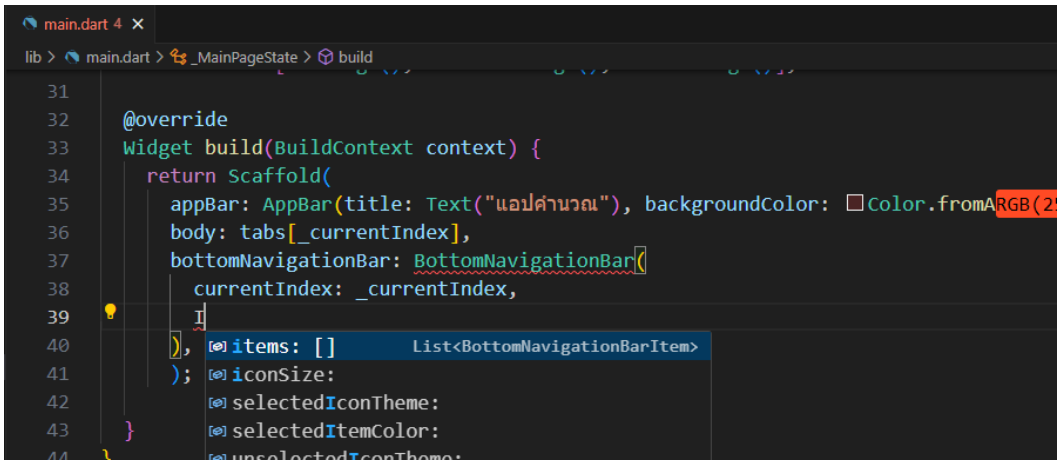
ที่มา : ณปภัช วรรณตรง (2564 : 25)



ภาพประกอบ 5.91 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 25)

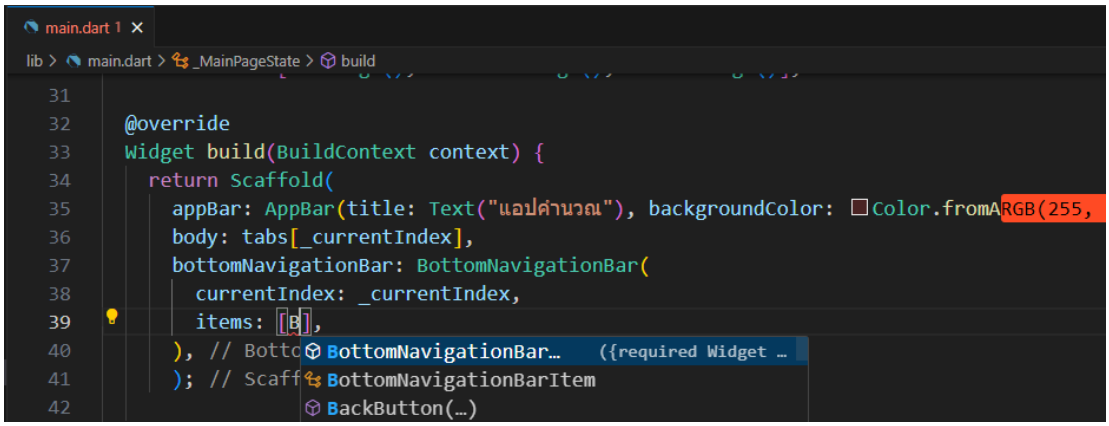
2.6.6 จากนั้นให้ทำการเพิ่มโค้ด Item: [...] เข้าไปต่อจาก _currentIndex



ภาพประกอบ 5.92 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 25)

2.6.7 ภายใน Item: [...] ให้ทำการเพิ่ม BottomNavigationBarItem(...)



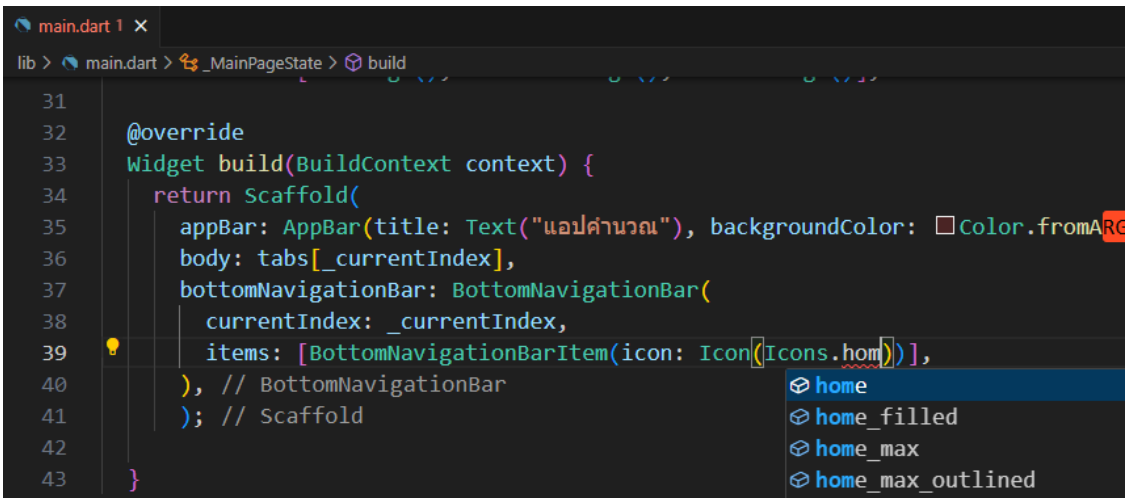
ภาพประกอบ 5.93 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 25)

2.6.8 ภายใน icon: ให้ทำการเพิ่มโค้ด Icon(Icons.[ประเภทของ Icon]) ซึ่งเป็นการใส่ icon ที่จะแสดงบนแท็บบาร์

ภาพประกอบ 5.94 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

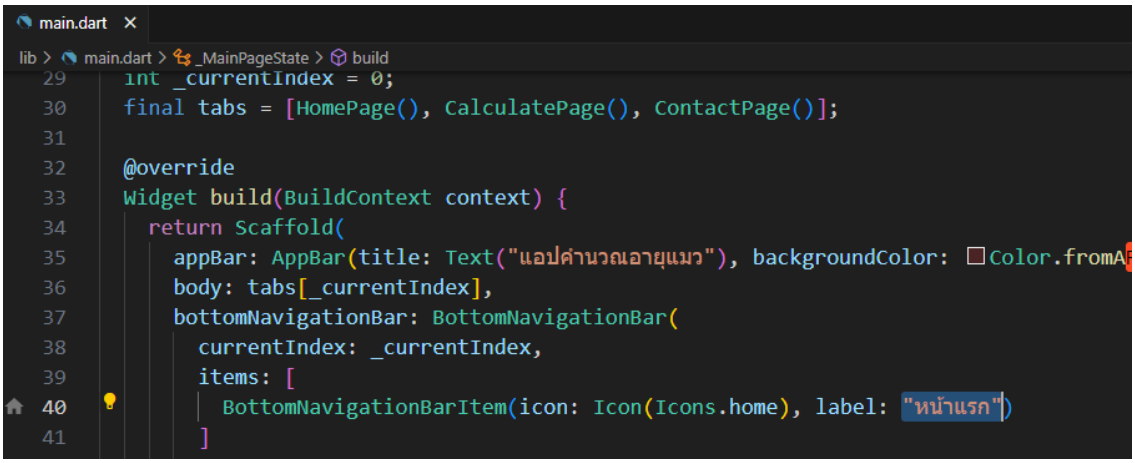
ที่มา : ณปภัช วรรณตรง (2564 : 25)



ภาพประกอบ 5.95 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

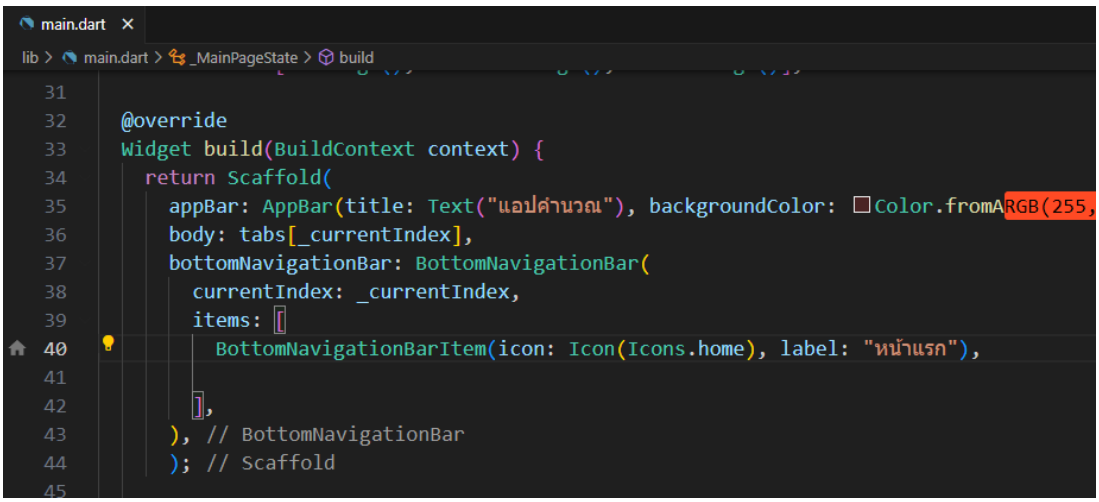
ที่มา : ณปภัช วรรณตรง (2564 : 25)

2.6.9 จากนั้นให้เพิ่ม label: ตามด้วยข้อความที่เก็บด้วยเครื่องหมาย “ ” เพื่อแสดงผล ซึ่งเป็นการใส่ข้อความที่จะแสดงใต้ icon



ภาพประกอบ 5.96 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

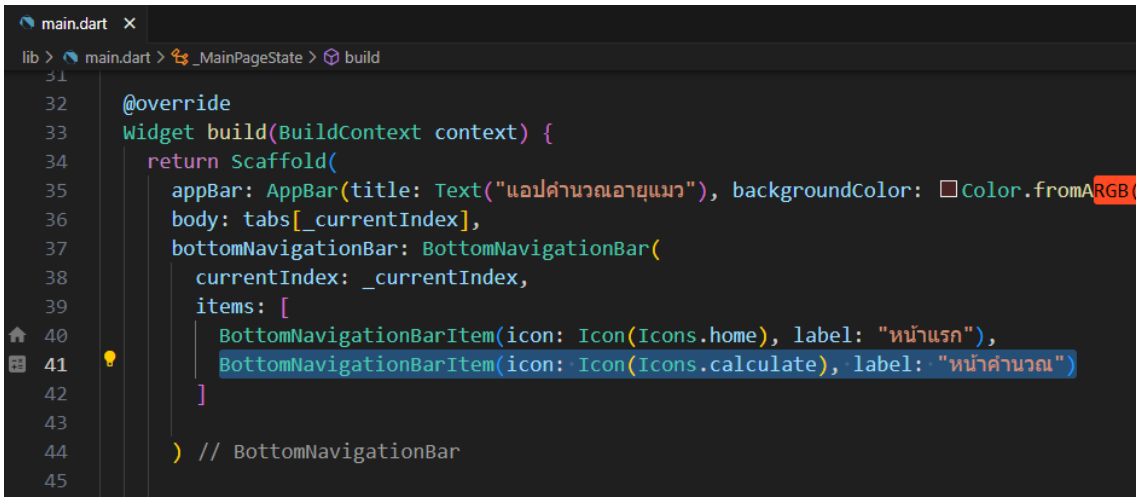
ที่มา : ณปภัช วรรณตรง (2564 : 25)



ภาพประกอบ 5.97 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

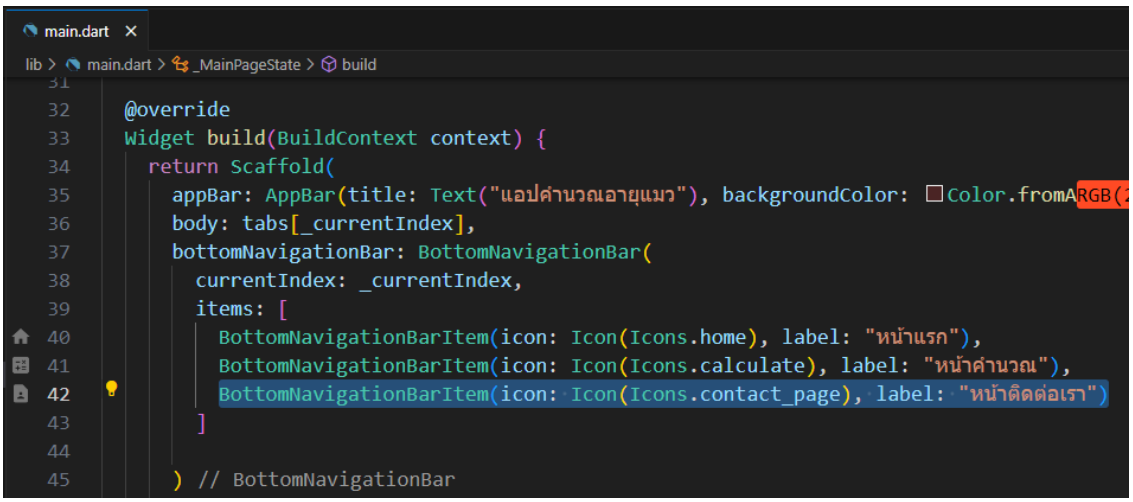
ที่มา : ณปภัช วรรณตรง (2564 : 25)

2.6.10 จากนั้นให้ทำการคัดลอกโค้ดเพิ่มมาอีกสองบรรทัด ทำการเปลี่ยนชื่อ Icon และข้อความภายใน “ ”



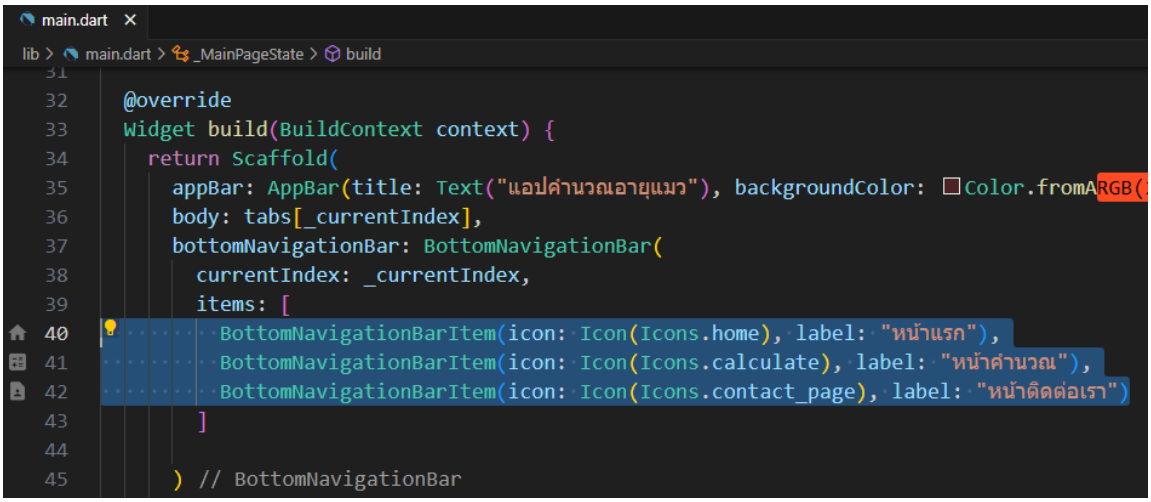
ภาพประกอบ 5.98 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 25)



ภาพประกอบ 5.99 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 25)

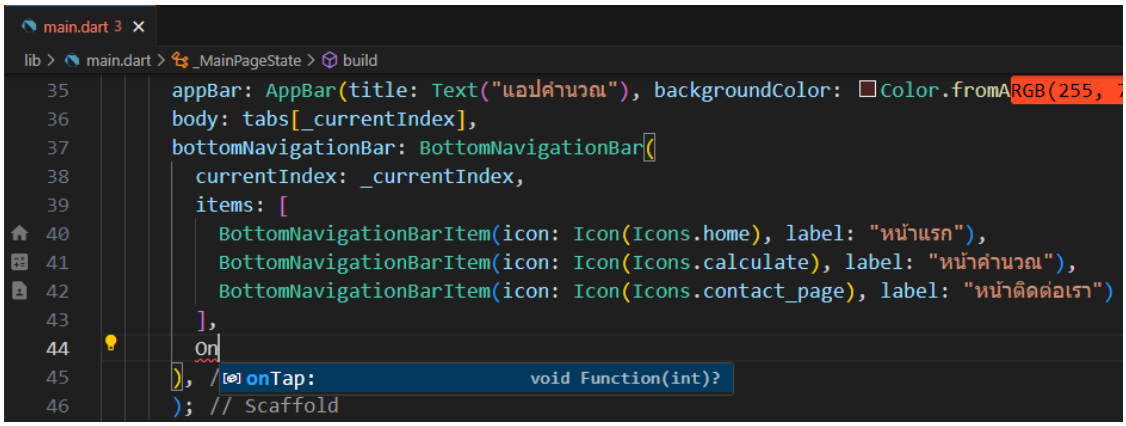


ภาพประกอบ 5.100 เพิ่มโค้ดในส่วนของ tag body: (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 25)

2.7 จากนั้นให้ทำการเพิ่มโค้ดต่อจาก item: [],

2.7.1 ให้ทำการเพิ่มโค้ด onTap: (...){...} ต่อจาก Item: [...]



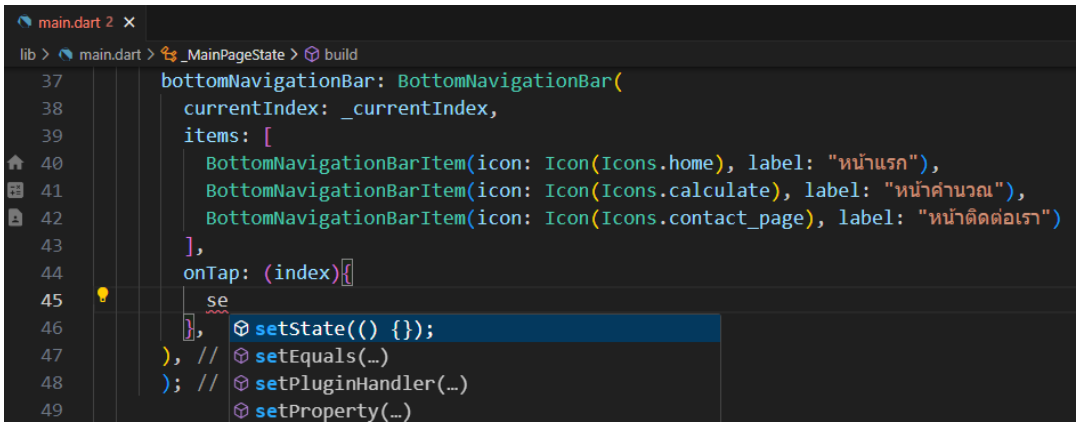
ภาพประกอบ 5.101 เพิ่มโค้ด onTab: (...){...}

ที่มา : ณปภัช วรรณตรง (2564 : 26)

ภาพประกอบ 5.102 เพิ่มโค้ด onTab: (...){...} (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 26)

2.7.2 ภายใน {...} ให้ทำการเพิ่มโค้ด setState((...) {...});



ภาพประกอบ 5.103 เพิ่มโค้ด onTab: (...){...} (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 26)

ภาพประกอบ 5.104 เพิ่มโค้ด onTab: (...){...} (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 26)

2.7.3 จากนั้นให้ทำการเพิ่มโค้ด print(...) สำหรับแสดงผล

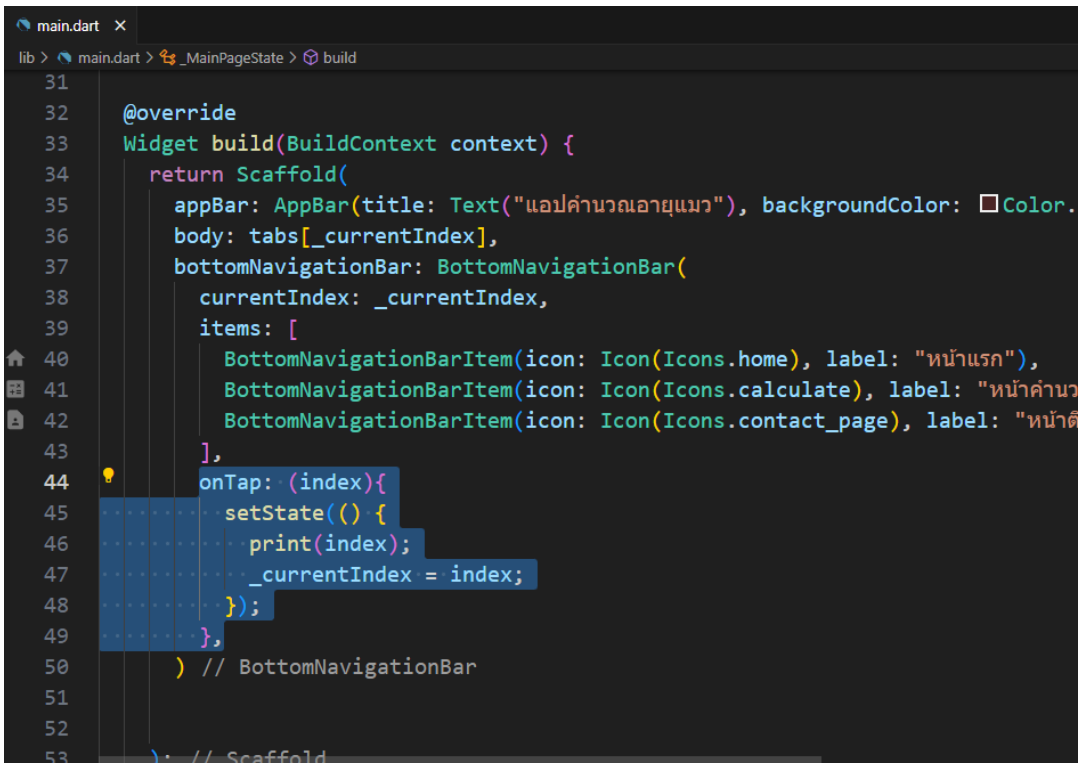
ภาพประกอบ 5.105 เพิ่มโค้ด onTab: (...){...} (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 26)

2.7.4 เพิ่มโค้ดกำหนดค่าตัวแปรให้กับ index

ภาพประกอบ 5.106 เพิ่มโค้ด onTab: (...){...} (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 26)



ภาพประกอบ 5.107 เพิ่มโค้ด onTab: (...){...} (ต่อ)

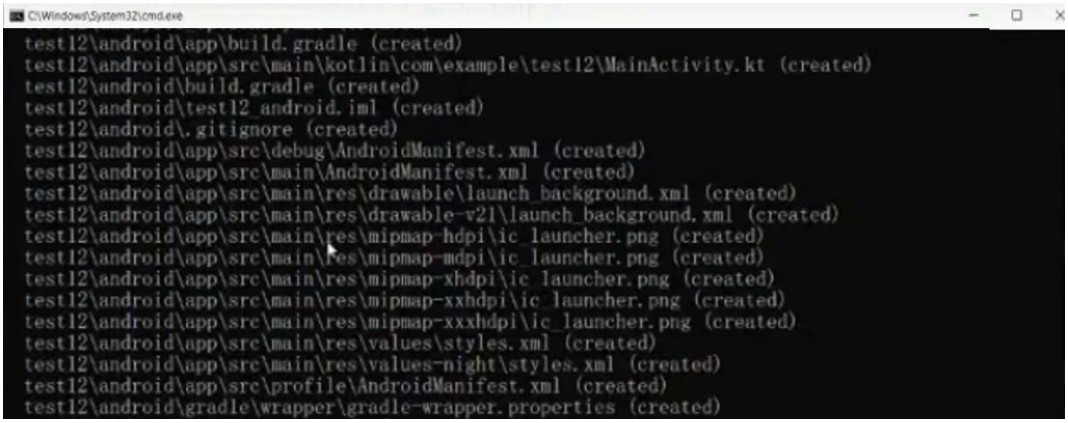
ที่มา : ณปภัช วรรณตรง (2564 : 26)

ข้อควรระวัง
ควรตรวจสอบตัวแปรให้ตรงกัน เพื่อป้องกันความผิดพลาดของโค้ด

การสร้างไอคอนของแอปพลิเคชัน

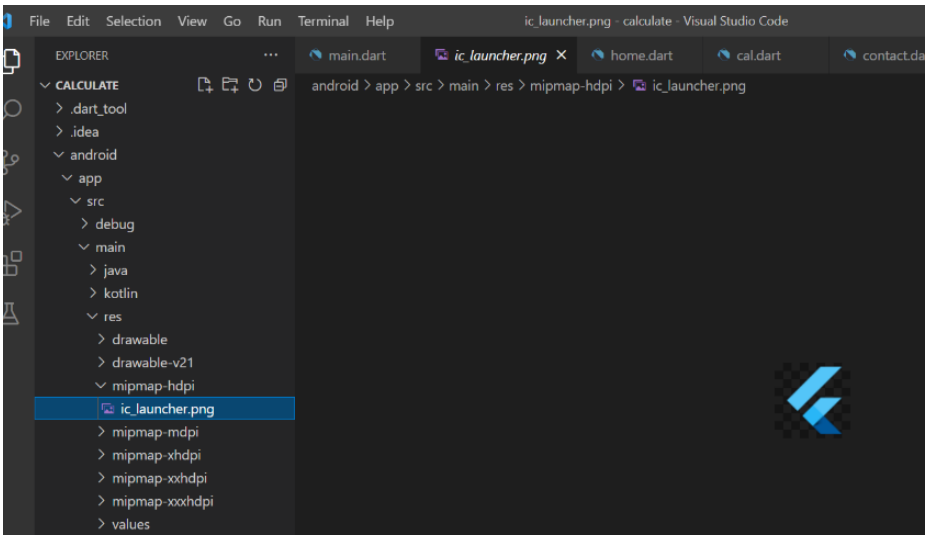
เนื้อหาส่วนนี้จะเป็นการสร้างไอคอนของแอปพลิเคชัน การสร้างไอคอนของแอปพลิเคชันเพื่อเมื่อนำไปติดตั้งบนอุปกรณ์เคลื่อนที่จะแสดงเป็นไอคอนบนหน้าจอให้ผู้ใช้คลิกเพื่อเข้าสู่หน้าจอแอปพลิชัน มีรายละเอียดขั้นตอนการทำดังนี้

1. ในขั้นตอนการสร้าง Project Flutter ตัวโปรเจกต์จะทำการสร้างไอคอนของตัวโปรเจกต์ติดมากับไฟล์งานอยู่แล้ว



ภาพประกอบ 5.108 ตัวอย่างไอคอนที่ติดมากับการสร้างไฟล์งาน

ที่มา : ณปภัช วรรณตรง (2564 : 4)

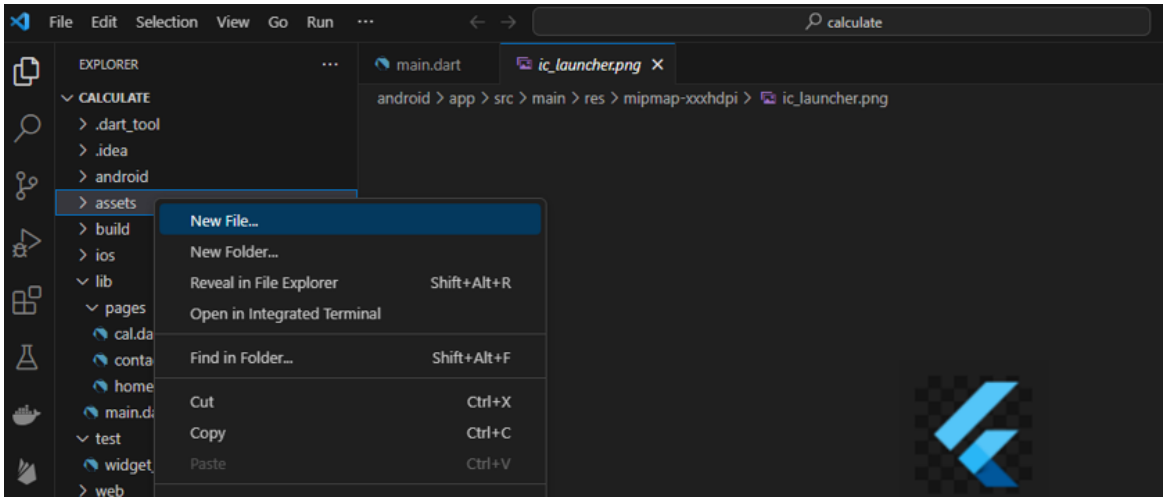


ภาพประกอบ 5.109 ตัวอย่างไอคอนที่ติดมากับการสร้างไฟล์งาน (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 5)

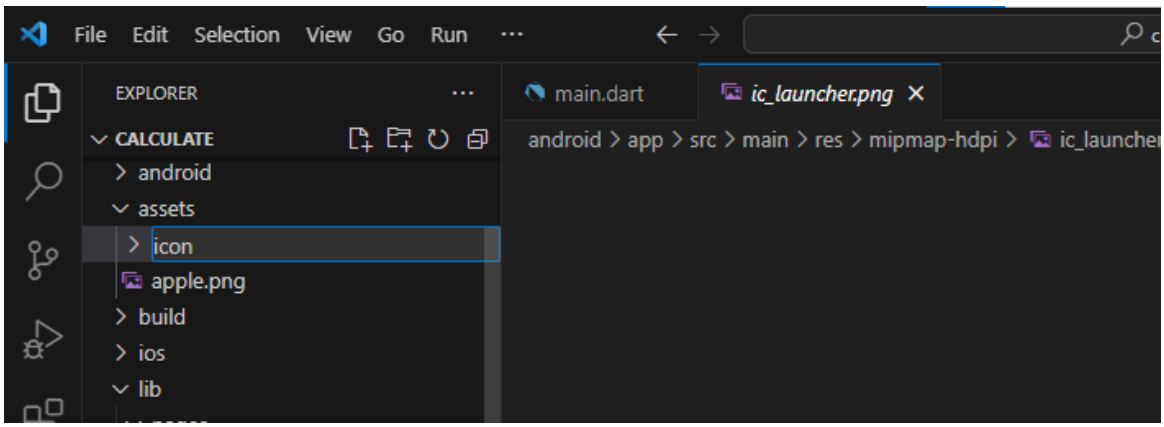
2. เริ่มทำการสร้างไอคอน ให้ทำการสร้างไฟล์ Folder icon สำหรับเก็บไอคอนไว้ใน Folder assets

ไปที่ asset จากนั้นให้ทำการสร้าง Folder icon



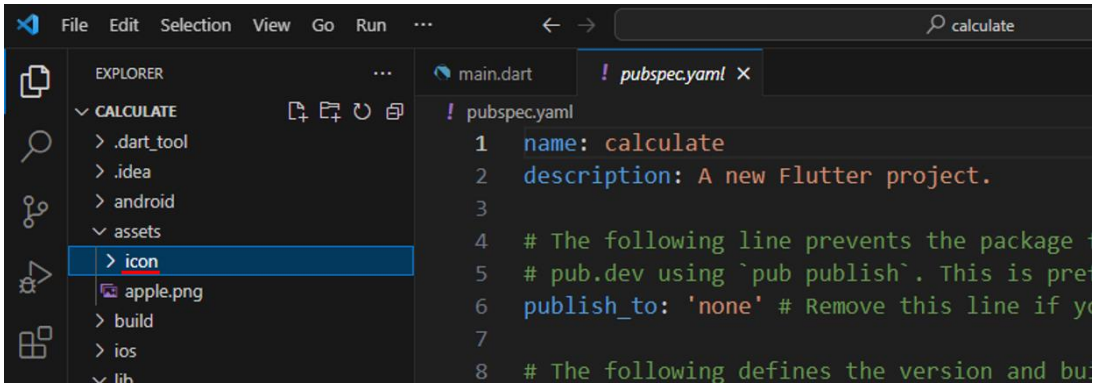
ภาพประกอบ 5.110 สร้าง Folder icon

ที่มา : ณปภัช วรรณตรง (2564 : 5)



ภาพประกอบ 5.111 สร้าง Folder icon (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 5)



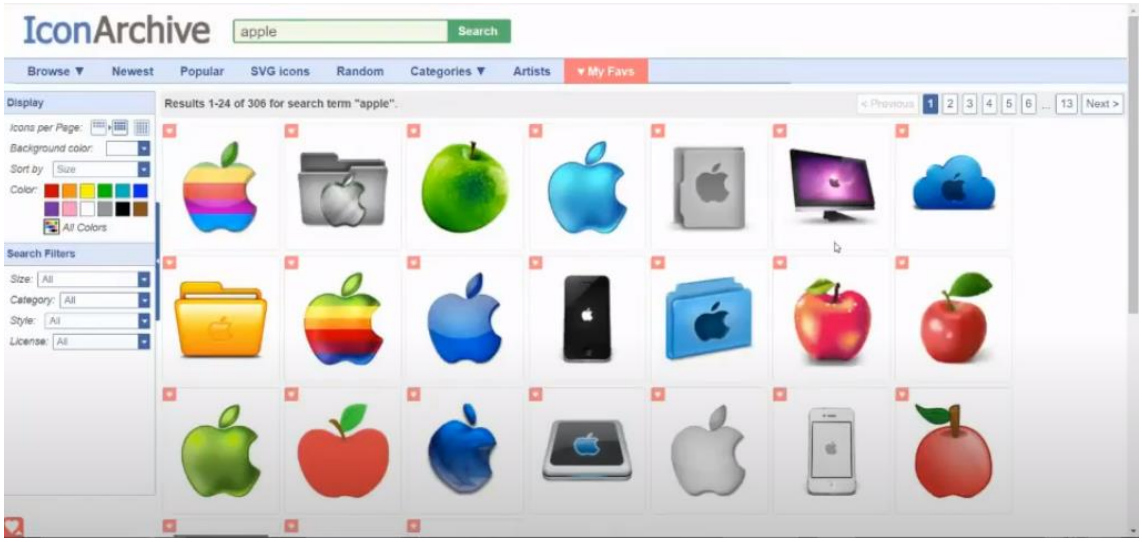
ภาพประกอบ 5.112 สร้าง Folder icon (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 6)

3. สามารถทำการค้นหาภาพไอคอนสำหรับนำมาใช้งานได้ฟรีจากเว็บฯ IconArchive

ภาพประกอบ 5.113 ค้นหาไอคอนสำหรับนำมาใช้งาน

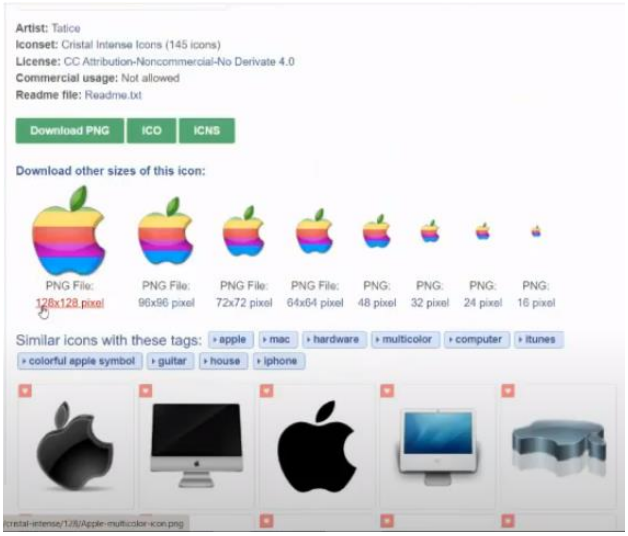
ที่มา : ณปภัช วรรณตรง (2564 : 7)



ภาพประกอบ 5.114 ค้นหาไอคอนสำหรับนำมาใช้งาน (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 7)

เมื่อได้ Icon ที่ต้องการแล้ว ให้ทำการเลือก Icon ที่มีขนาดใหญ่ที่สุด จากนั้นค่อยทำการดาวน์โหลด



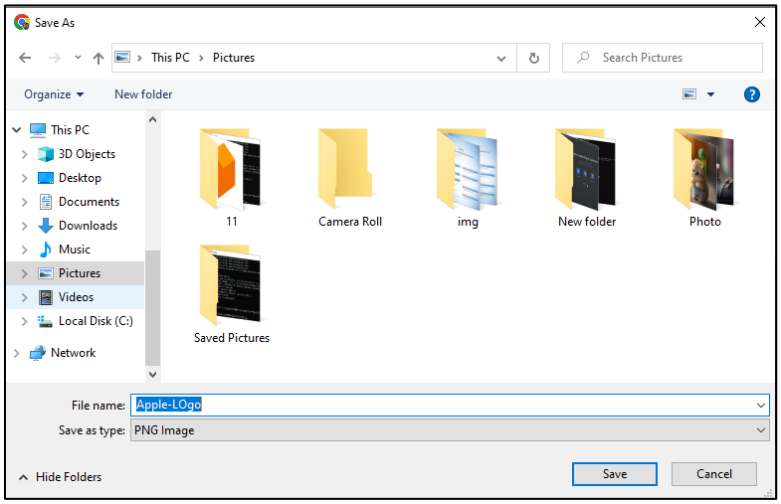
ภาพประกอบ 5.114 ค้นหาไอคอนสำหรับนำมาใช้งาน (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 7)

หลังจากเลือก Icon ได้แล้วให้ทำการดาวน์โหลดหรือบันทึกภาพ โดยการคลิกขวาที่ภาพ จากนั้นเลือกเป็น Save image as...

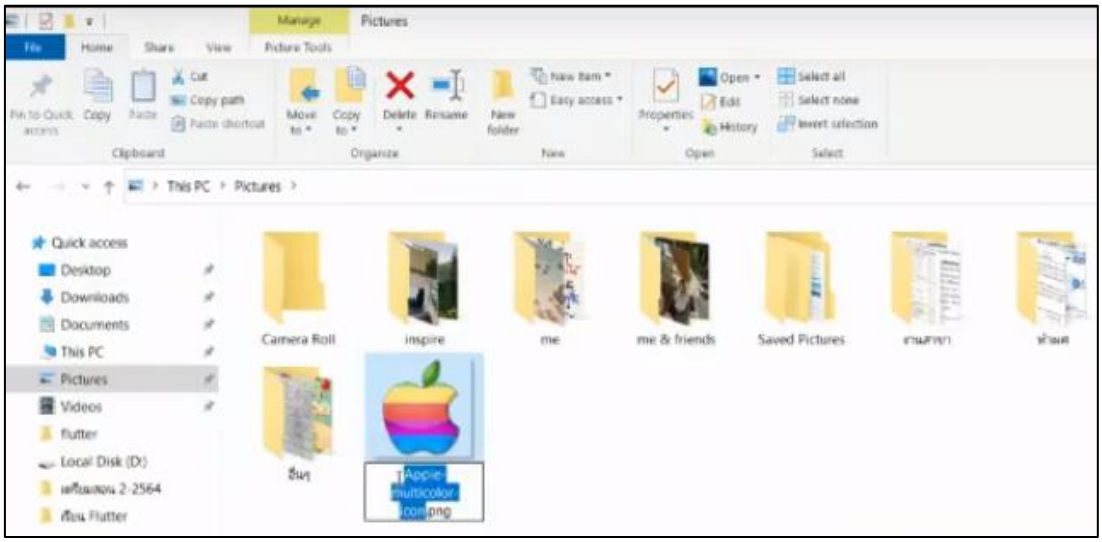
ภาพประกอบ 5.115 ค้นหาไอคอนสำหรับนำมาใช้งาน (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 7)

เลือกตำแหน่งสำหรับเก็บไฟล์รูปภาพ



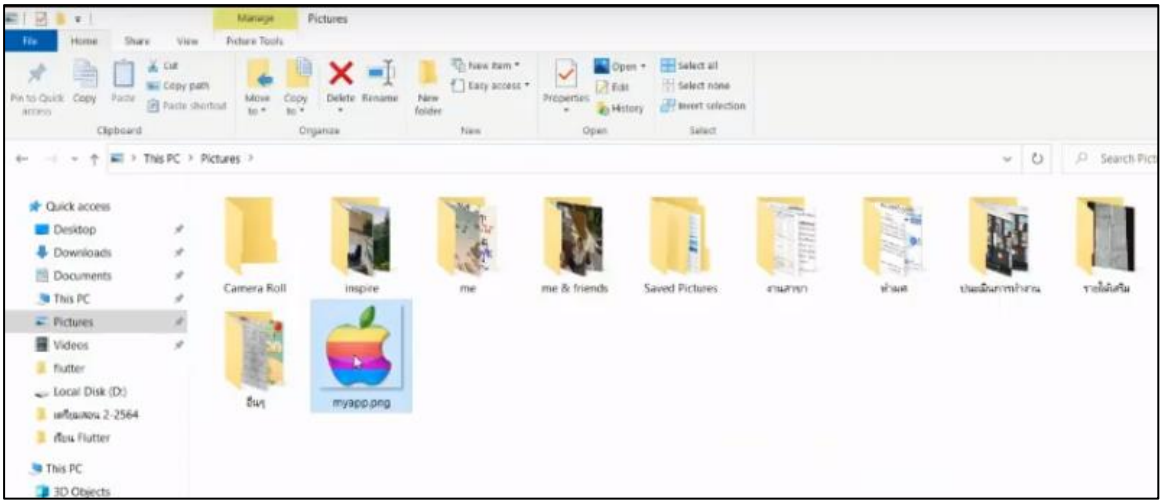
ภาพประกอบ 5.116 ค้นหาไอคอนสำหรับนำมาใช้งาน (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 7)

ทำการเปลี่ยนชื่อให้สามารถเรียกใช้งานได้ง่าย



ภาพประกอบ 5.117 ค้นหาไอคอนสำหรับนำมาใช้งาน (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 7)

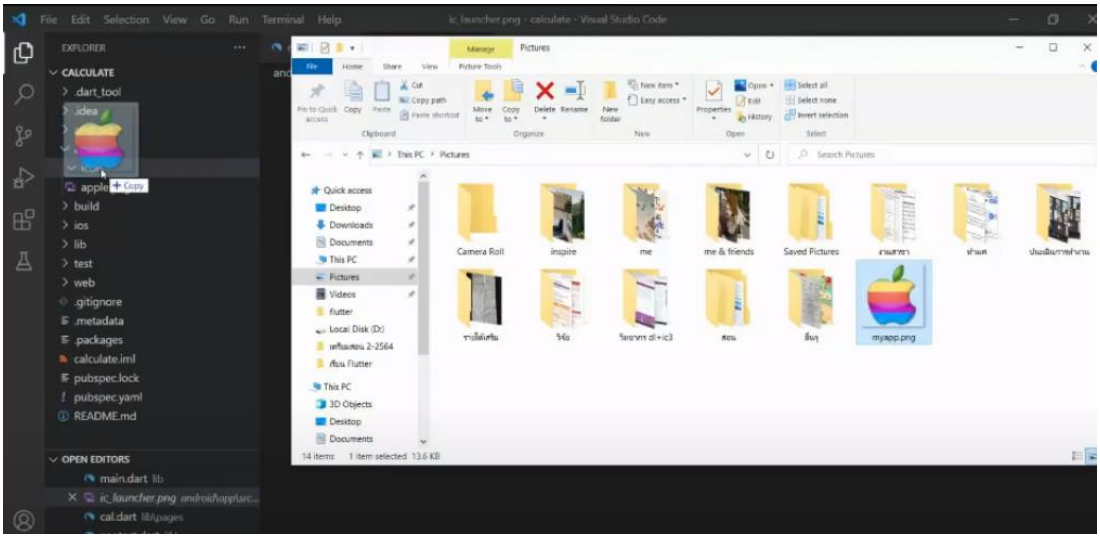


ภาพประกอบ 5.118 ค้นหาไอคอนสำหรับนำมาใช้งาน (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 7)

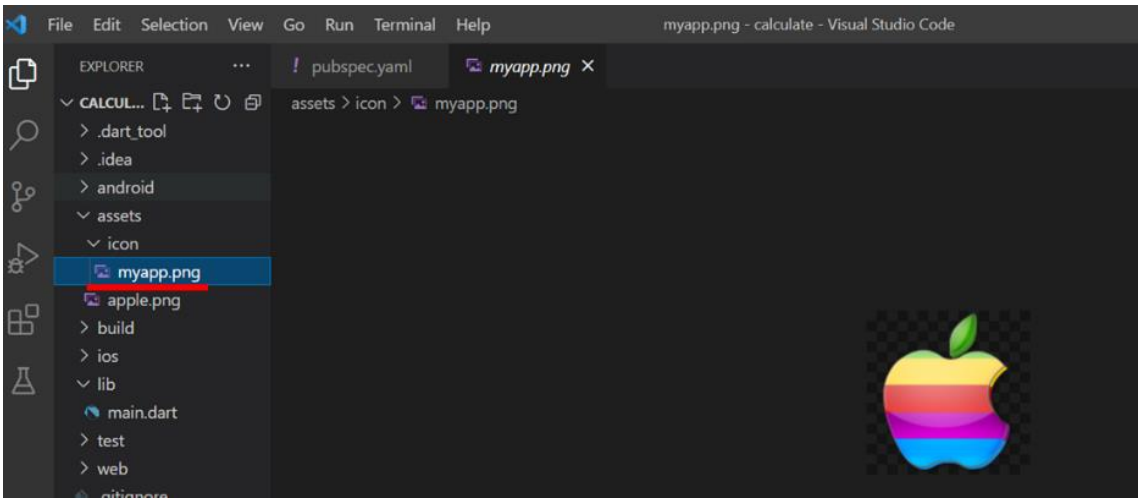
4. เมื่อทำการเลือกไอคอนที่ต้องการได้แล้ว ให้ทำการดาวน์โหลดไอคอนดังกล่าวแล้วนำไปไว้ใน Folder icon

ให้นำไฟล์ภาพประกอบที่ได้ทำการดาวน์โหลดมาไปใส่ใน Folder icon ที่ได้ทำการสร้างไว้ โดยการคลิกลากไปวาง



ภาพประกอบ 5.119 ดาวน์โหลดไอคอนที่ต้องการแล้วนำไปไว้ใน Folder icon

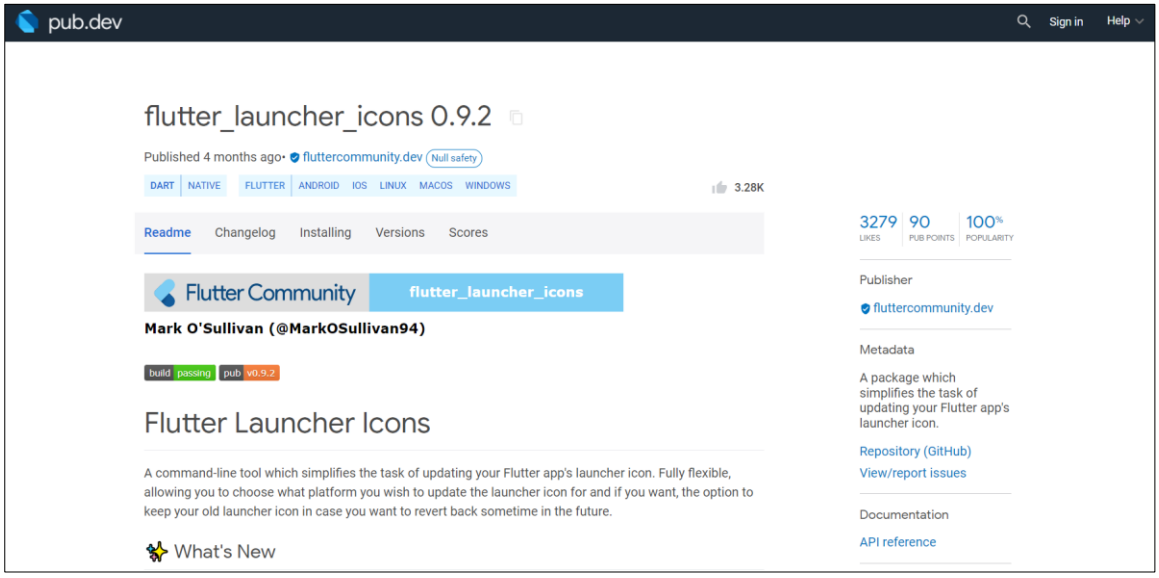
ที่มา : ณปภัช วรรณตรง (2564 : 8)



ภาพประกอบ 5.120 ดาวน์โหลดไอคอนที่ต้องการแล้วนำไปไว้ใน Folder icon (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 7)

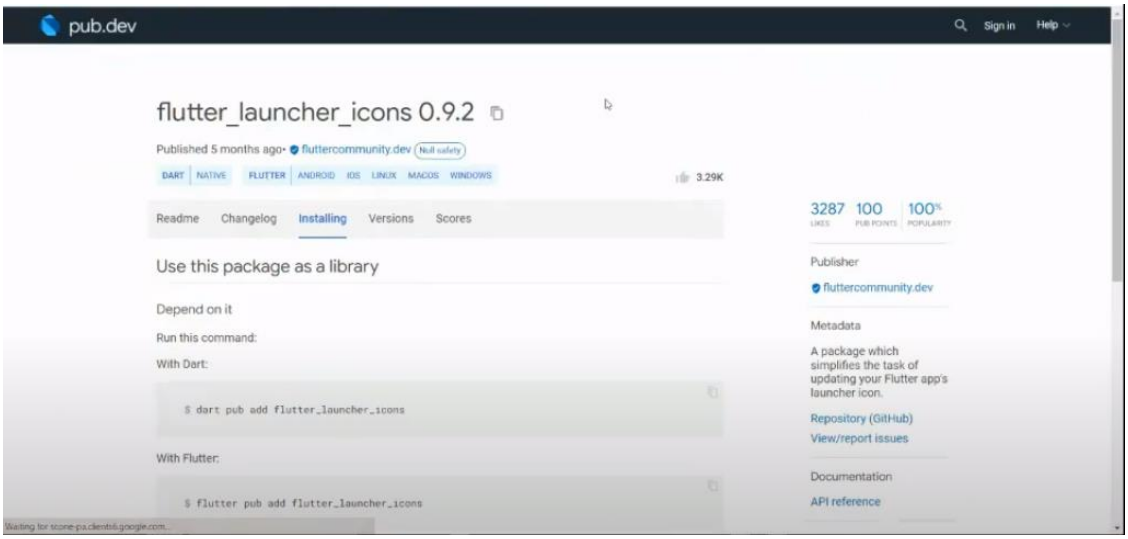
5. ให้ทำการค้นหา flutter launcher icon เพื่อนำโค้ดบางส่วนที่เกี่ยวข้องกับการเรียกใช้ไอคอนเข้ามาใช้งาน



ภาพประกอบ 5.121 ค้นหา flutter launcher icon

ที่มา : ณปภัช วรรณตรง (2564 : 9)

ไปที่ Installing เพื่อคัดลอกโค้ดสำหรับเปลี่ยน Icon



ภาพประกอบ 5.122 ค้นหา flutter launcher icon (ต่อ)

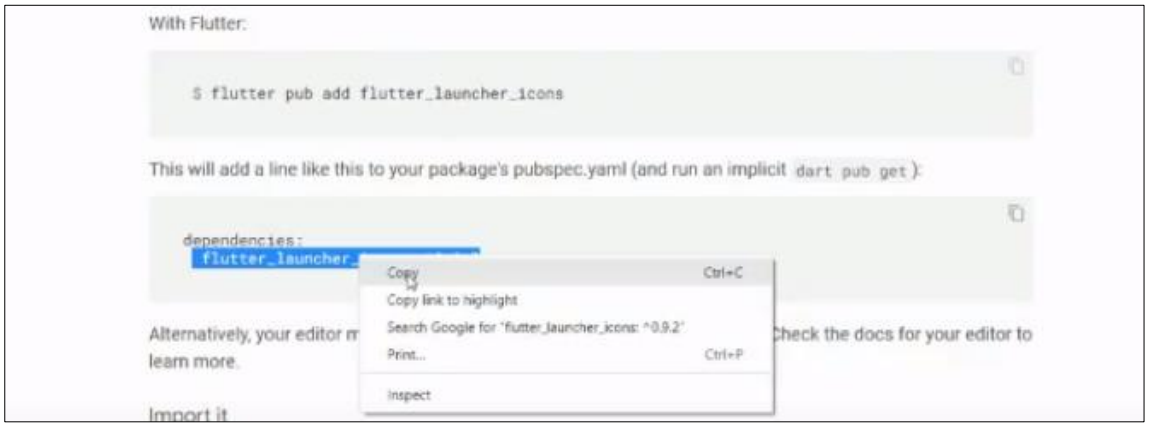
ที่มา : ณปภัช วรรณตรง (2564 : 9)

หมายเหตุ
pub.dev คือ เว็บไซต์ที่รวม package เพิ่มเติมของ flutter

6. ในส่วนของ Installing ให้ทำการคัดลอกโค้ดใน dependencies:

ภาพประกอบ 5.123 คัดลอกโค้ดใน dependencies:

ที่มา : ณปภัช วรรณตรง (2564 : 10)

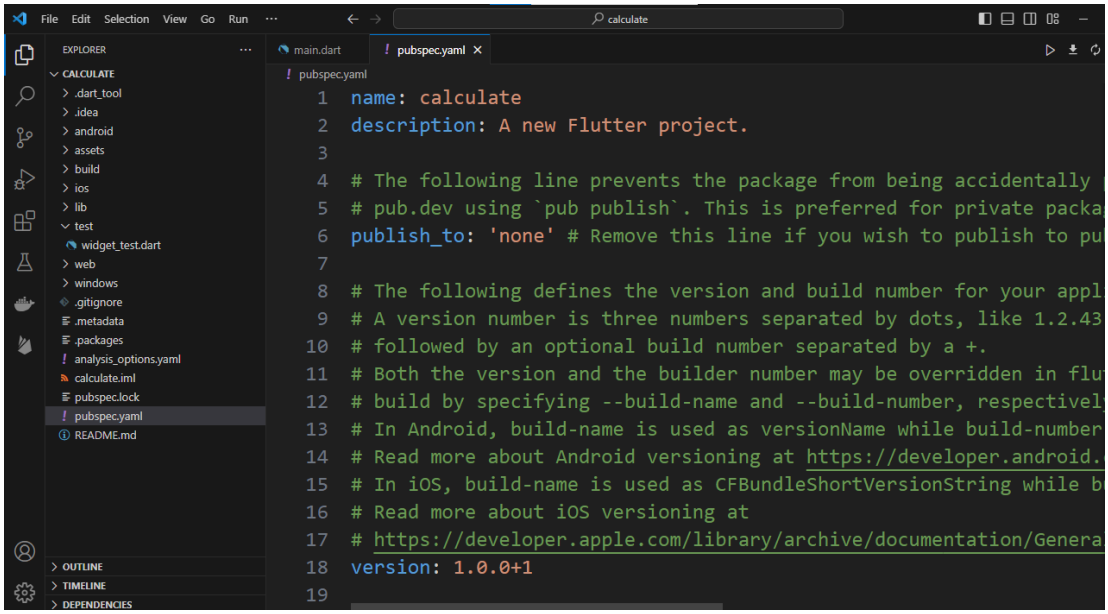


ภาพประกอบ 5.124 คัดลอกโค้ดใน dependencies: (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 10)

7. เปิดไฟล์ pubspec.yaml จากนั้นให้นำโค้ดที่คัดลอกมาจาก dependencies: มาวางในตำแหน่ง dependencies: เดียวกันในไฟล์ที่เปิดมาตามภาพ

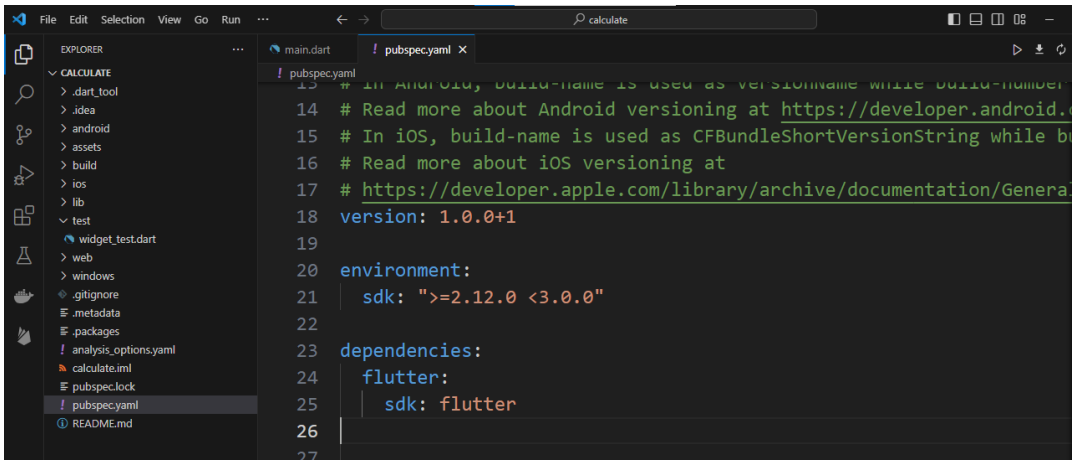
7.1 ไปที่ไฟล์ pubspec.yaml



ภาพประกอบ 5.125 นำโค้ดที่คัดลอกมา มาไว้ในไฟล์ pubspec.yaml

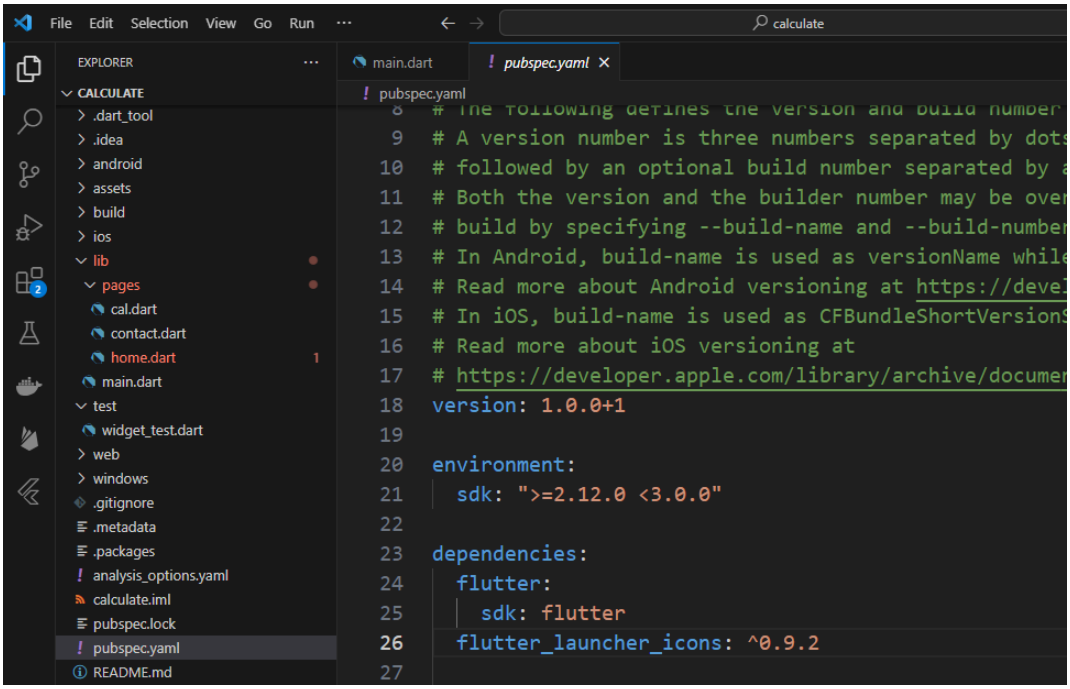
ที่มา : ณปภัช วรรณตรง (2564 : 11)

7.2 ไปที่ dependencies: จากนั้นให้วางโค้ดต่อจาก sdk:



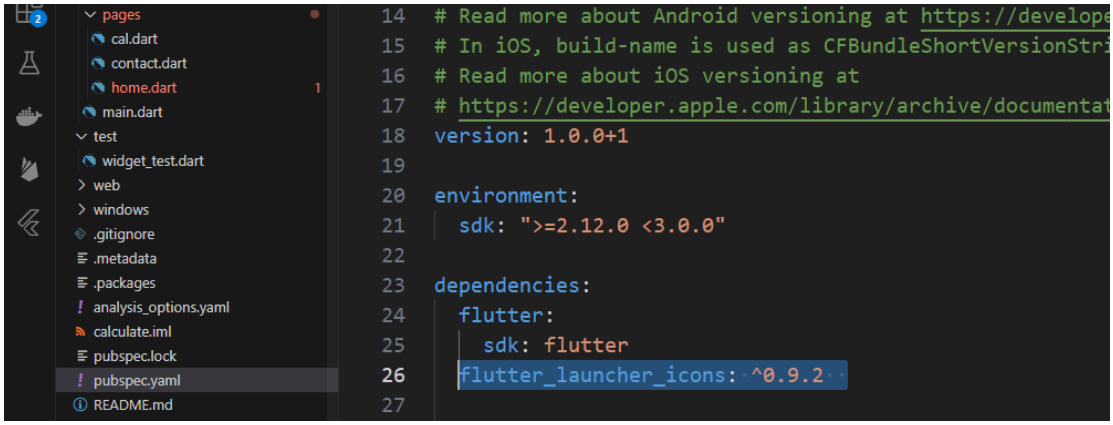
ภาพประกอบ 5.126 นำโค้ดที่คัดลอกมา มาไว้ในไฟล์ pubspec.yaml (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 11)



ภาพประกอบ 5.127 นำโค้ดที่คัดลอกมา มาไว้ในไฟล์ pubspec.yaml (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 11)

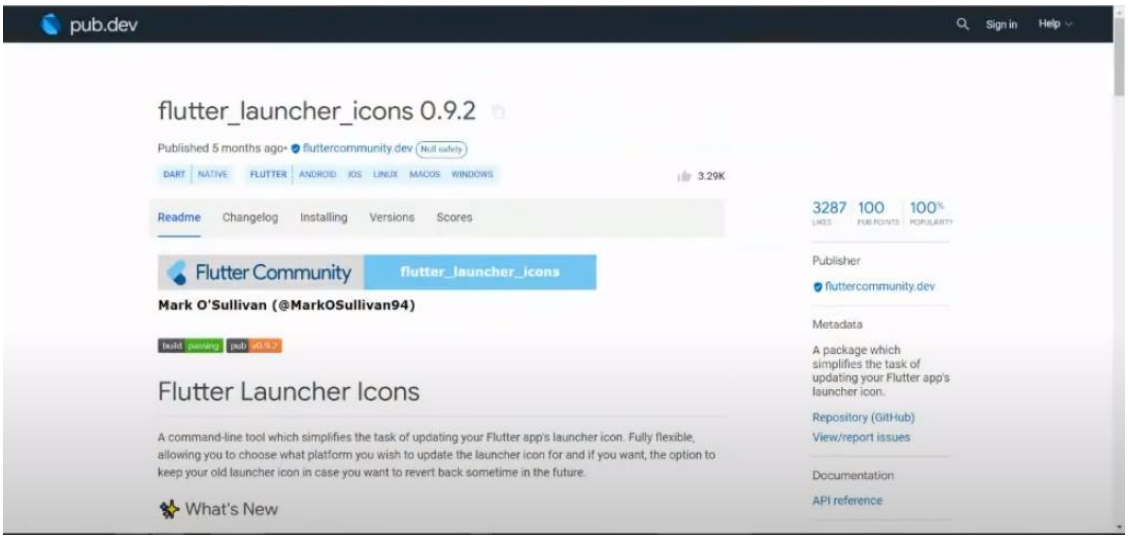


ภาพประกอบ 5.128 นำโค้ดที่คัดลอกมา มาไว้ในไฟล์ pubspec.yaml (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 11)

8. จากนั้นให้ไปที่ flutter_icons: ให้ทำการคัดลอกมาทั้งหมดของส่วน flutter_icons:

8.1 ให้ทำการกลับไปที่หน้า Read me จากหน้า Installing



ภาพประกอบ 5.129 คัดลอกทั้งหมดของส่วน flutter_icons:

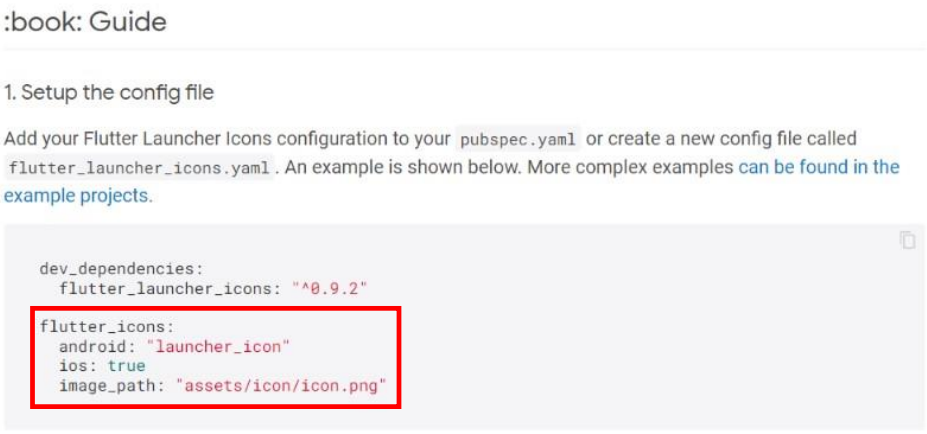
ที่มา : ณปภัช วรรณตรง (2564 : 12)

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

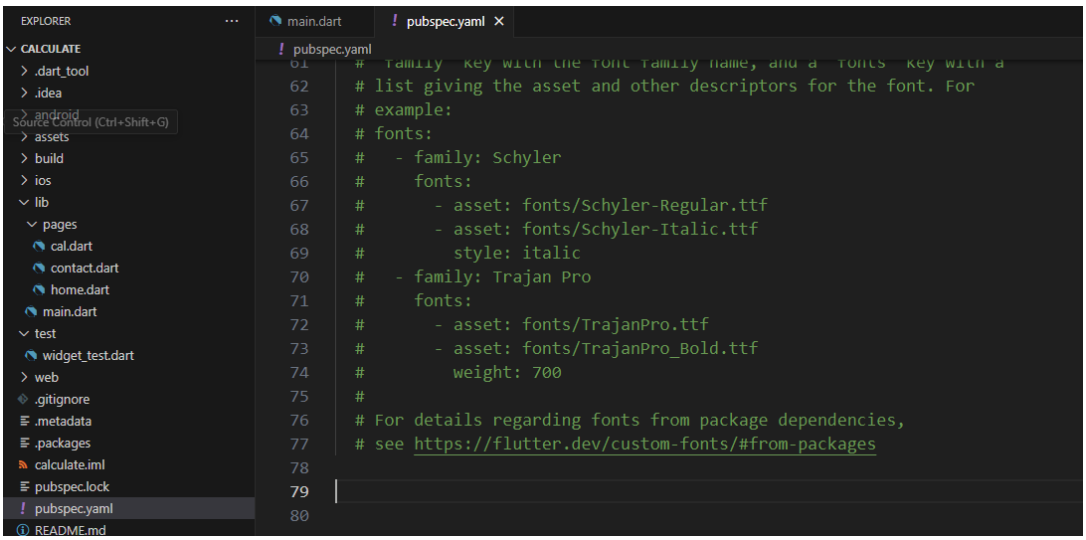


ภาพประกอบ 5.131 คัดลอกทั้งหมดของส่วน flutter_icons: (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 12)

ภาพประกอบ 5.132 คัดลอกทั้งหมดของส่วน flutter_icons: (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 12)

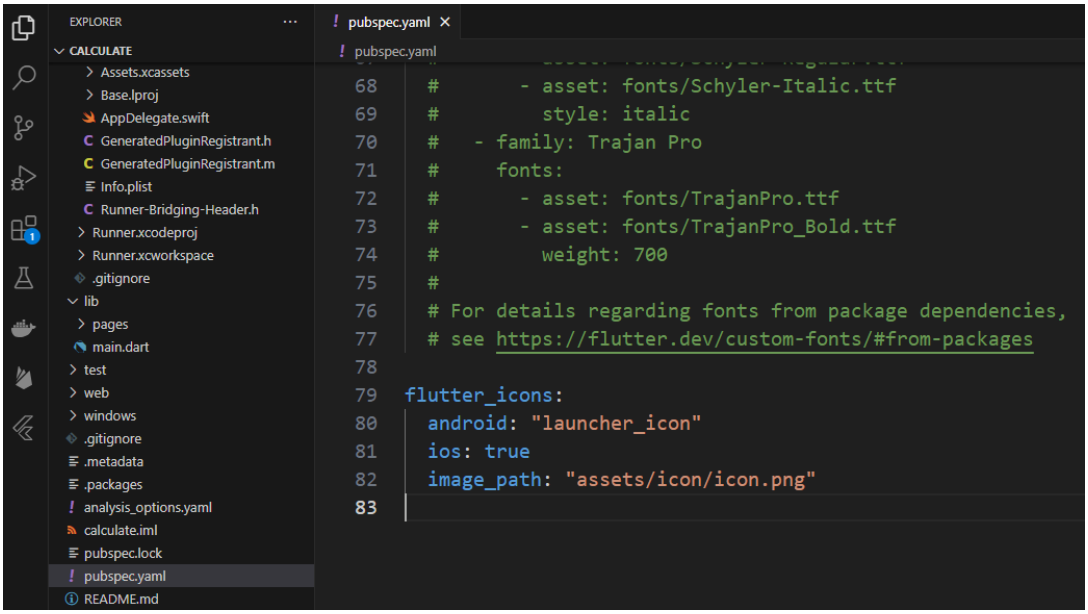
10. เปิดไฟล์ pubspec.yaml จากนั้นให้นำโค้ดที่คัดลอกมาจาก flutter_icons: มาวางในไฟล์ที่เปิดมา

จากนั้นไปที่ไฟล์ pubspec.yaml เลื่อนลงมาข้างล่าง แล้วทำการวางโค้ดที่ได้คัดลอกไว้ จัดย่อหน้าให้ตรงตามภาพถัดไป (เนื่องจากการย่อหน้ามีผลต่อการทำงานของโค้ดไฟล์นี้)



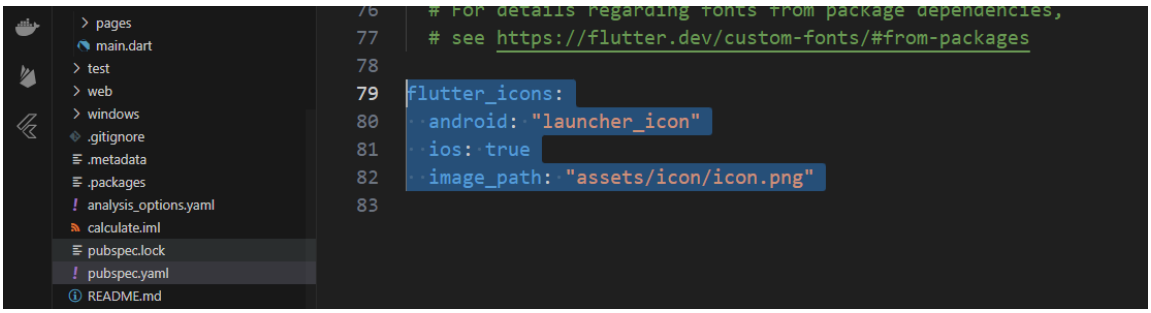
ภาพประกอบ 5.133 นำโค้ดที่คัดลอกมาจาก flutter_icons: มาวาง

ที่มา : ณปภัช วรรณตรง (2564 : 13)



ภาพประกอบ 5.134 นำโค้ดที่คัดลอกมาจาก flutter_icons: มาวาง (ต่อ)

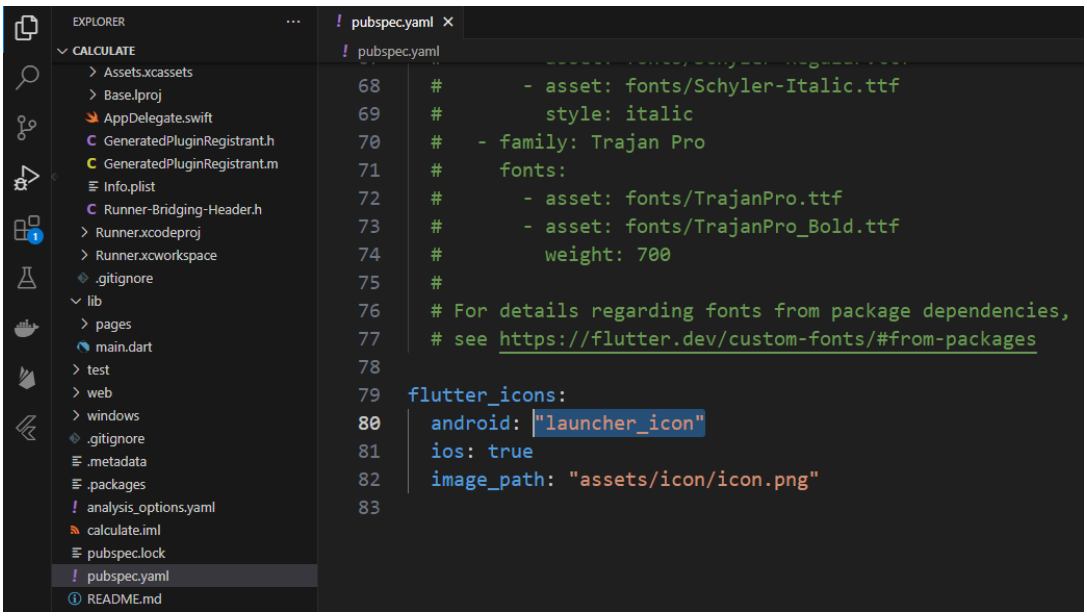
ที่มา : ณปภัช วรรณตรง (2564 : 13)



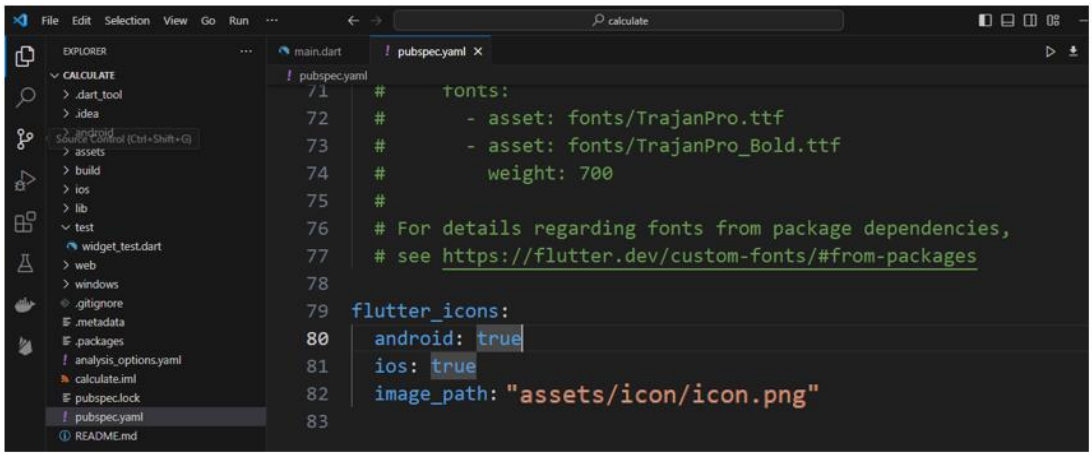
ภาพประกอบ 5.135 นำโค้ดที่คัดลอกมาจาก flutter_icons: มาวาง (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 13)

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



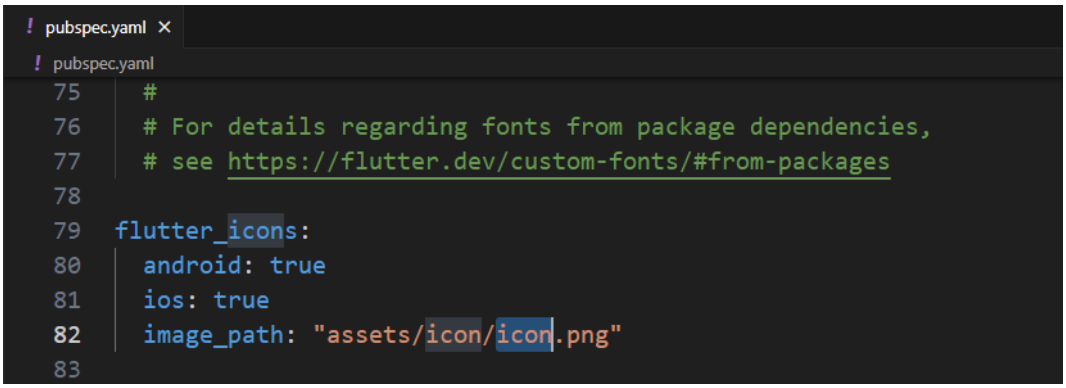
ภาพประกอบ 5.136 แก้ไขโค้ดใส่ส่วน flutter_icons:
ที่มา : ณปภัช วรรณตรง (2564 : 14)



ภาพประกอบ 5.137 แก้ไขโค้ดใส่ส่วน flutter_icons: (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 14)

จากนั้นให้ทำการเปลี่ยนชื่อไฟล์รูปภาพสำหรับ Icon



ภาพประกอบ 5.138 แก้ไขโค้ดใส่ส่วน flutter_icons: (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 14)

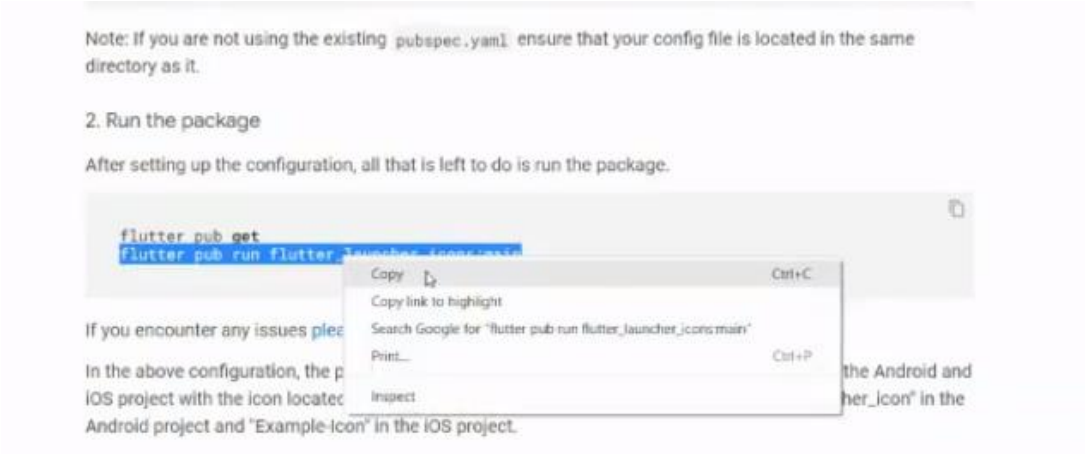
ภาพประกอบ 5.139 แก้ไขโค้ดใส่ส่วน flutter_icons: (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 14)

ภาพประกอบ 5.140 แก้ไขโค้ดใส่ส่วน flutter_icons: (ต่อ)

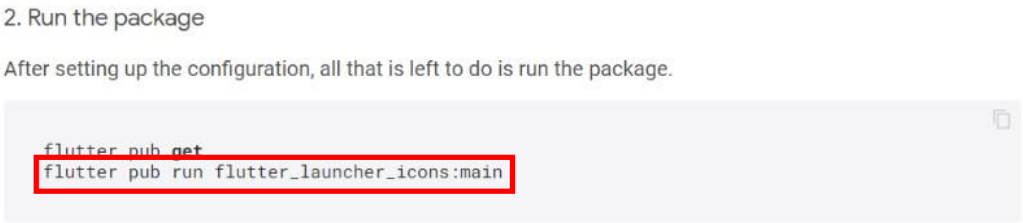
ที่มา : ณปภัช วรรณตรง (2564 : 14)

12. ให้กลับไปที่ 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 : ณปภัช วรรณตรง (2564 : 15)

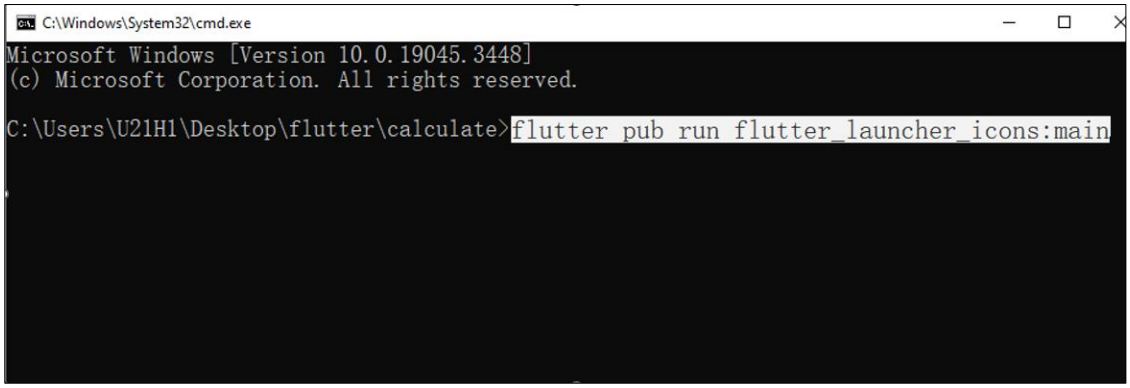


ภาพประกอบ 5.142 คัดลอกโค้ด flutter pub run (ต่อ)

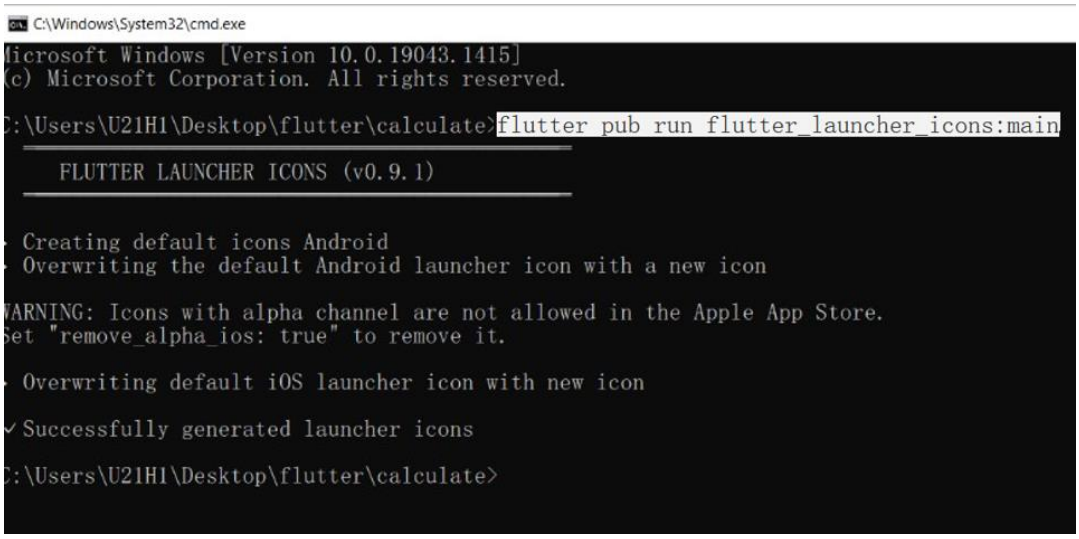
ที่มา : ณปภัช วรรณตรง (2564 : 15)

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

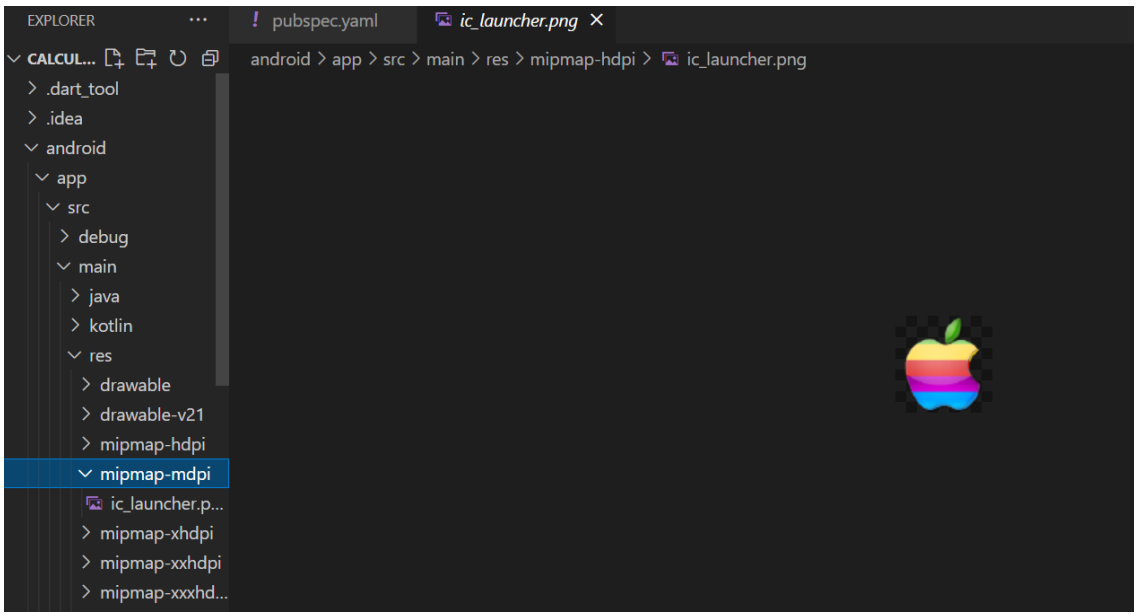


ภาพประกอบ 5.143 วางโค้ด flutter pub run ในหน้าต่าง cmd
ที่มา : ณปภัช วรรณตรง (2564 : 16)



ภาพประกอบ 5.144 วางโค้ด flutter run ในหน้าต่าง cmd (ต่อ)
ที่มา : ณปภัช วรรณตรง (2564 : 16)

14. ให้ทำการเปิดไฟล์งาน และเข้าไปที่ Folder res จะพบว่าไอคอนที่ Flutter สร้างมาได้ถูกเปลี่ยนเป็นไอคอนที่ได้ทำการเปลี่ยนแปลงเรียบร้อยแล้ว

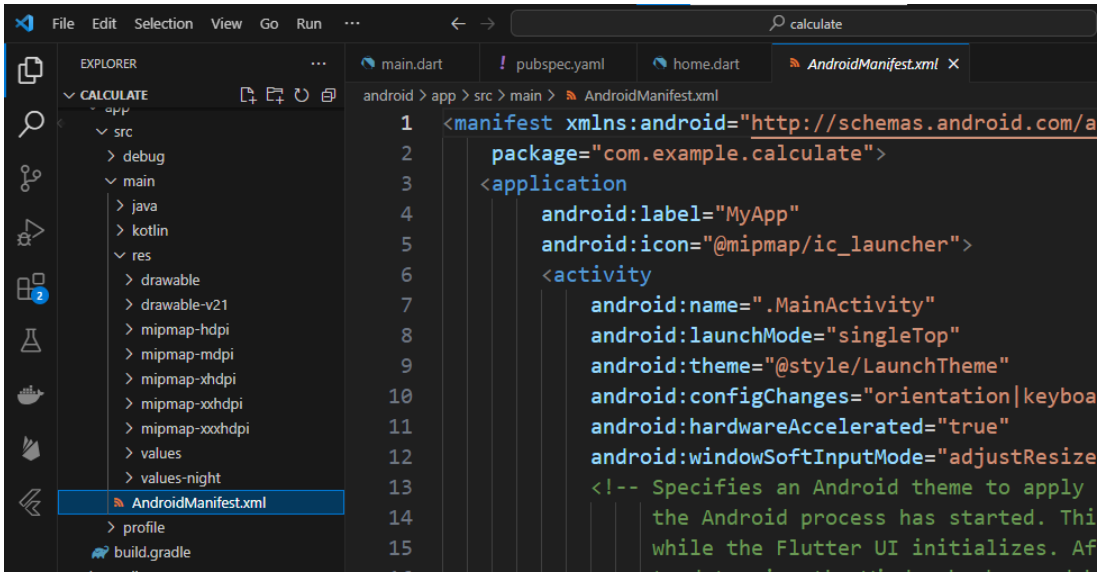


ภาพประกอบ 5.145 ไอคอนได้ทำการเปลี่ยนแปลงเรียบร้อยแล้ว

ที่มา : ณปภัช วรรณตรง (2564 : 17)

15. จากนั้นให้เปิดไฟล์ AndroidManifest.xml ไปที่ tag android:label = “” ให้ทำการเปลี่ยนชื่อใน “” เป็นชื่อไอคอนที่ได้ทำการดาวน์โหลดมา

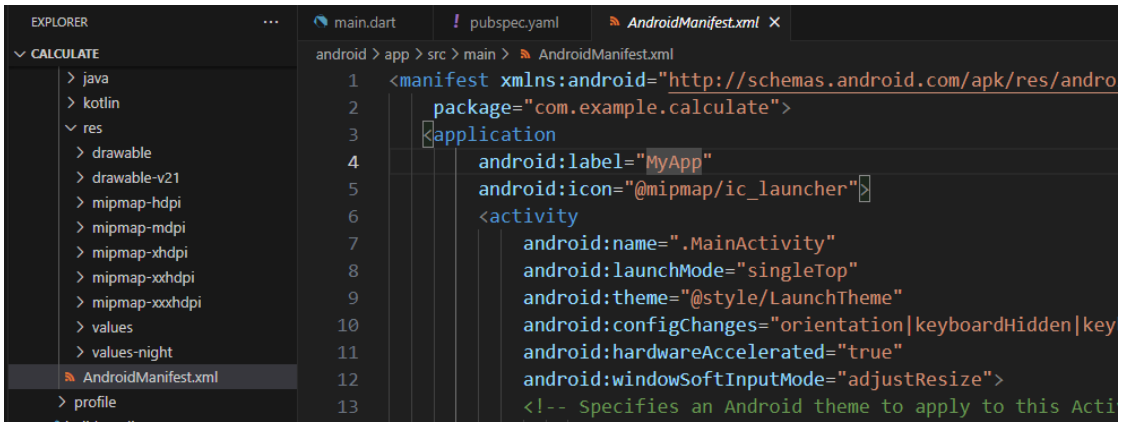
ไปที่ไฟล์ AndroidManifest.xml ใน Folder src



ภาพประกอบ 5.146 เปลี่ยนชื่อไอคอนใน tag android:label

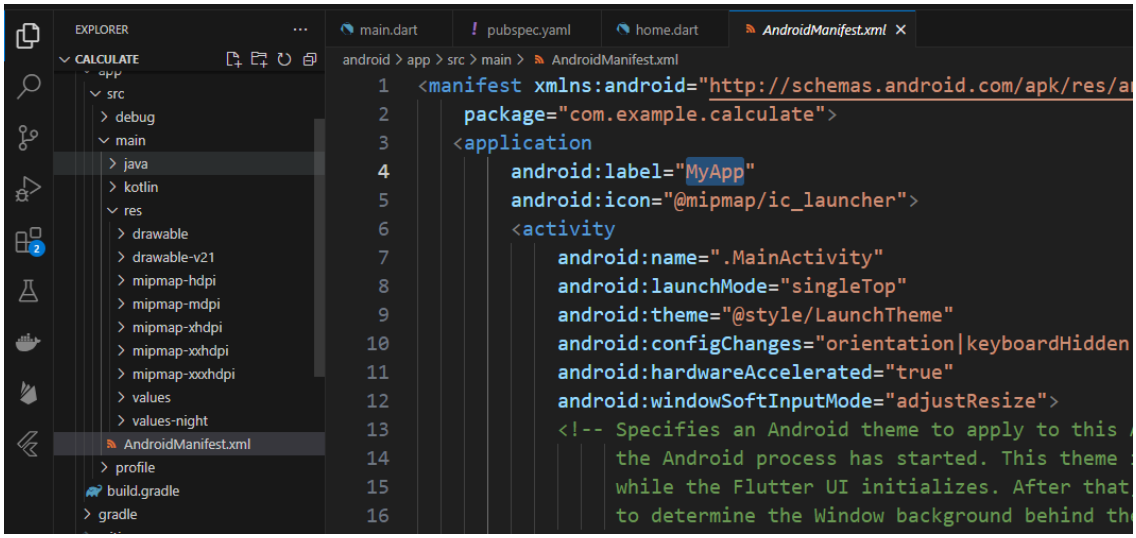
ที่มา : ณปภัช วรรณตรง (2564 : 18)

ทำการเปลี่ยนแอปที่ android:label ในเครื่องหมาย “ ”



ภาพประกอบ 5.147 เปลี่ยนชื่อไอคอนใน tag android:label (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 18)



ภาพประกอบ 5.148 เปลี่ยนชื่อไอคอนใน tag android:label (ต่อ)

ที่มา : ณปภัช วรรณตรง (2564 : 18)

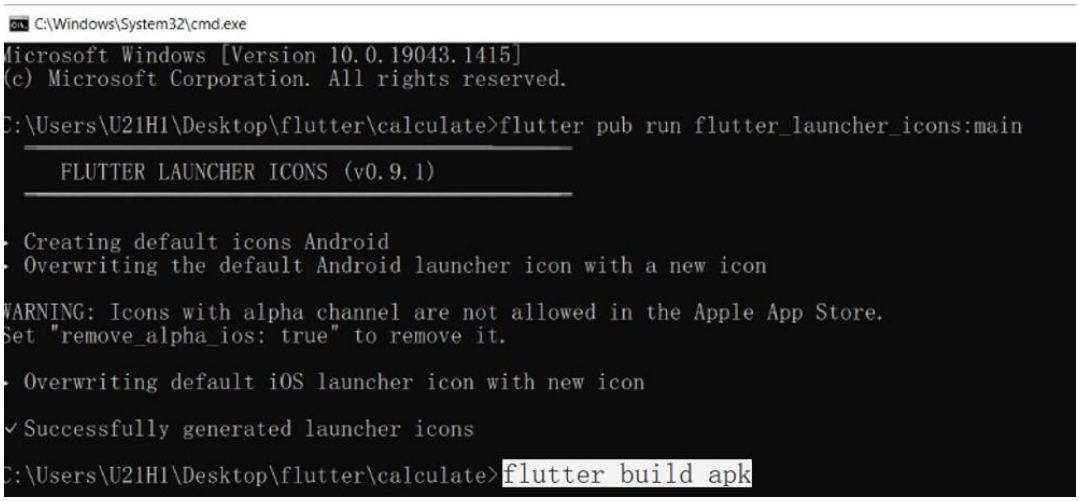
ทำการสร้างไฟล์สำหรับนำไปติดตั้งและใช้งานบนมือถือไฟล์ .apk

ไฟล์ .apk เป็นแพ็คเกจ Android คล้ายกับไฟล์ .exe บน Windows หรือไฟล์ .dmg บน macOS แพ็คเกจนี้ประกอบด้วยไฟล์ ทรัพยากร และข้อมูลเมตา (Metadata) ทั้งหมดที่จำเป็นสำหรับแอปพลิเคชันบนมือถือ

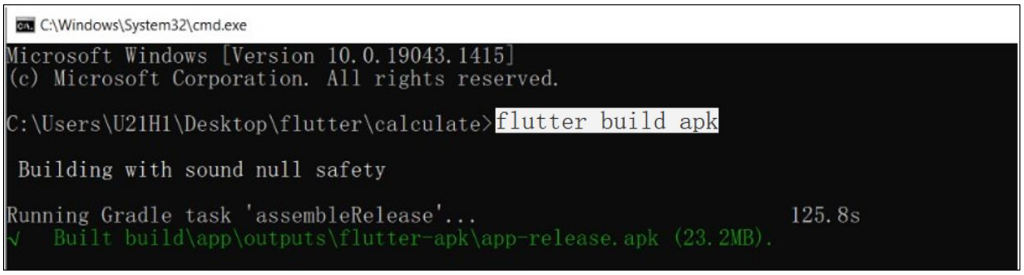
ในการทำงานบนอุปกรณ์ Android นักพัฒนาใช้ตัวติดตั้งแอปเพื่อติดตั้งแอปหลังจากที่ผู้ใช้ดาวน์โหลดไฟล์ .apk ที่เกี่ยวข้องลงในอุปกรณ์ของตน แพ็คเกจจะจัดเก็บข้อมูลเมตาที่จำเป็น เช่น ชื่อแอปฯ หมายเลขเวอร์ชัน และรายการสิทธิ์ ซึ่งระบบปฏิบัติการใช้ในการจัดการการควบคุมการเข้าถึง นอกจากนี้ ไฟล์ .apk ยังให้ความสะดวกในระหว่างขั้นตอนการทดสอบและแก้ไขจุดบกพร่องของการพัฒนา 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



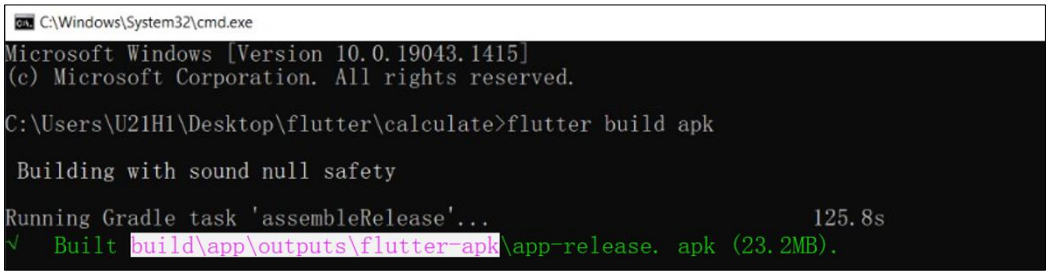
ภาพประกอบ 5.149 ใช้คำสั่ง flutter build apk
ที่มา : ณปภัช วรรณตรง (2564 : 20)



ภาพประกอบ 5.150 ใช้คำสั่ง flutter build apk (ต่อ)

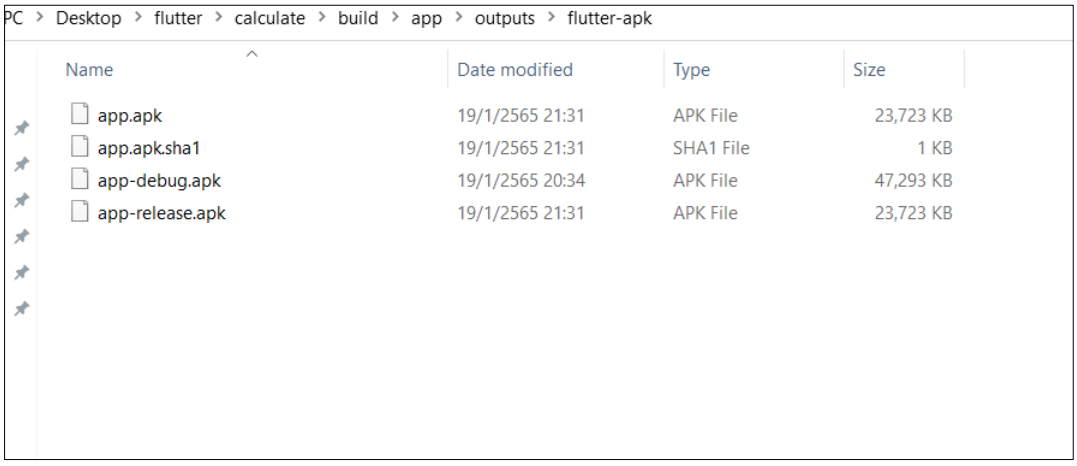
ที่มา : ณปภัช วรรณตรง (2564 : 20)

2. หลังจากที่ flutter ทำการสร้างไฟล์ .apk เสร็จเรียบร้อยแล้ว ให้ทำการคัดลอก Path Built build\.... ไปต่อกับ Path Project ใน Folder ก็จะเจอไฟล์ที่ได้ทำการสร้างมา



ภาพประกอบ 5.151 คัดลอก Path ที่ตัวหนังสือสีเขียว

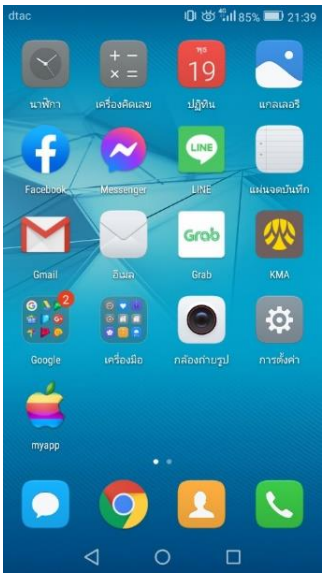
ที่มา : ณปภัช วรรณตรง (2564 : 21)



ภาพประกอบ 5.152 นำ Path มาวางต่อ Path Project เพื่อค้นหาไฟล์ที่ได้ทำการสร้างมา

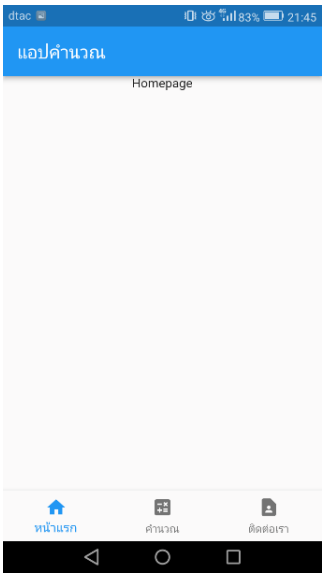
ที่มา : ณปภัช วรรณตรง (2564 : 21)

3. จากนั้นให้นำไฟล์ไปติดตั้งที่โทรศัพท์มือถือ เมื่อติดตั้งเสร็จเรียบร้อย และลองเข้าทดสอบแอปพลิเคชัน จะได้ผลลัพธ์ตามภาพ



ภาพประกอบ 5.153 นำไฟล์ที่สร้างเสร็จแล้วไปติดตั้งที่มือถือ

ที่มา : ณปภัช วรรณตรง (2564 : 22)



ภาพประกอบ 5.154 เข้าทดสอบแอปพลิเคชัน

ที่มา : ณปภัช วรรณตรง (2564 : 22)

4. หลังจากรอโหลดหน้าแอปพลิเคชันแล้ว จะได้ผลลัพธ์การพัฒนาตามภาพ

ภาพประกอบ 5.155 ผลลัพธ์การพัฒนาแอปพลิเคชัน
ที่มา : ณปภัช วรรณตรง (2564 : 22)

หากเกิดข้อผิดพลาด
สามารถนำโค้ดที่ผิดพลาดไปค้นหาใน StackOverFlow เพื่อหาคำตอบหรือการแก้ไขเบื้องต้นได้

บทสรุป

สำหรับเนื้อหาที่กล่าวมาทั้งหมดในบทนี้ จะพูดถึงวิธีการสร้างหน้าแอปพลิเคชันแยกแต่ละหน้า จากนั้นเรียนรู้วิธีการสร้างแท็บบาร์ที่ด้านล่างของแอปพลิเคชัน ซึ่งเป็นส่วนเมื่อคลิกที่แท็บบาร์แล้วจะไปยังหน้าจอที่เตรียมไว้ในขั้นตอนแรก เรียนรู้การสร้างไอคอนของแอปพลิเคชัน เพื่อเมื่อนำไปติดตั้งบนอุปกรณ์เคลื่อนที่จะแสดงเป็นไอคอนให้ผู้ใช้คลิกเพื่อเข้าสู่หน้าจอแอปพลิชัน และเรียนรู้การสร้างไฟล์แอปพลิเคชันเป็นไฟล์สำหรับติดตั้งและใช้งานบนระบบปฏิบัติการแอนดรอยด์ โดยจะเป็นไฟล์นามสกุล .apk ที่สามารถนำไปติดตั้งบนอุปกรณ์เคลื่อนที่ เพื่อให้สามารถเรียกใช้งานแอปพลิคชันจากที่ใดก็ได้

เอกสารอ้างอิง

จีราวุธ วารินทร์. (2564). **พัฒนาโมไบล์แอพด้วย Flutter + Dart.** กรุงเทพฯ : สำนักพิมพ์ซิมพลิฟาย.

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

file.org. (16 กุมภาพันธ์ 2566). **APK File.** สืบค้นจาก, https://file.org/extension/apk

บทที่ 6
การสร้างฐานข้อมูลสำหรับการใช้งานด้วย Python Django

ในบทก่อนหน้านี้ผู้เรียนได้เรียนรู้ถึงวิธีการสร้างหน้าแอปพลิเคชันแยกแต่ละหน้า จากนั้นเรียนรู้วิธีการสร้างแท็บบาร์ที่ด้านล่างของแอปพลิเคชัน ซึ่งเป็นส่วนเมื่อคลิกที่แท็บบาร์แล้ว จะไปยังหน้าจอที่เตรียมไว้ในขั้นตอนแรก เรียนรู้การสร้างไอคอนของแอปพลิเคชัน เพื่อเมื่อนำไปติดตั้งบนอุปกรณ์เคลื่อนที่จะแสดงเป็นไอคอนให้ผู้ใช้คลิกเพื่อเข้าสู่หน้าจอแอปพลิชัน และเรียนรู้การสร้างไฟล์แอปพลิเคชันเป็นไฟล์สำหรับติดตั้งและใช้งานบนระบบปฏิบัติการแอนดรอยด์ โดยจะเป็นไฟล์นามสกุล .apk ที่สามารถนำไปติดตั้งบนอุปกรณ์เคลื่อนที่ให้สามารถใช้งานแอปพลิคชันได้ 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 (ลุงวิศวสอนคำนวณ, 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 ให้สามารถนำมา ใช้ร่วมกันได้ เพื่อลดขั้นตอนการเขียนโค้ดที่ซ้ำซ้อนกัน โดยไม่จำเป็นต้องคัดลอกโค้ดจากส่วนหนึ่งแล้วไปวางในอีกส่วนหนึ่ง (Copy & Paste) สำหรับการทำงานในลักษณะเดียวกัน

4. Model Template View (MTV) โครงสร้างการทำงานของ Django อยู่ในรูปแบบที่เรียกกว่า 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 สามารถนำไปใช้งานได้อย่างอิสระ โดยไม่มีค่าใช้จ่ายและเงื่อนไขใด ๆ

6. Support Community: มีกลุ่มผู้สนับสนุนการใช้งานจำนวนมากที่พร้อมให้ความช่วยเหลือและแก้ไขปัญหาต่าง ๆ ที่เกิดขึ้น (บัญชา ปะสีละเตสัง, 2563 : 26)

ภาพประกอบ 6.1 django
ที่มา : django (2020 : 1)

สถาปัตยกรรม Django

Django มีรูปแบบสถาปัตยกรรมที่เรียกว่า MTV หรือ Model Template View เป็นคอนเซปต์การเขียนเว็บไซต์ที่มีส่วนประกอบการทำงานที่เชื่อมโยงกันระหว่าง (บัญชา ปะสีละเตสัง, 2563 : 27)

M - Model มีหน้าที่ในการเชื่อมต่อฐานข้อมูล

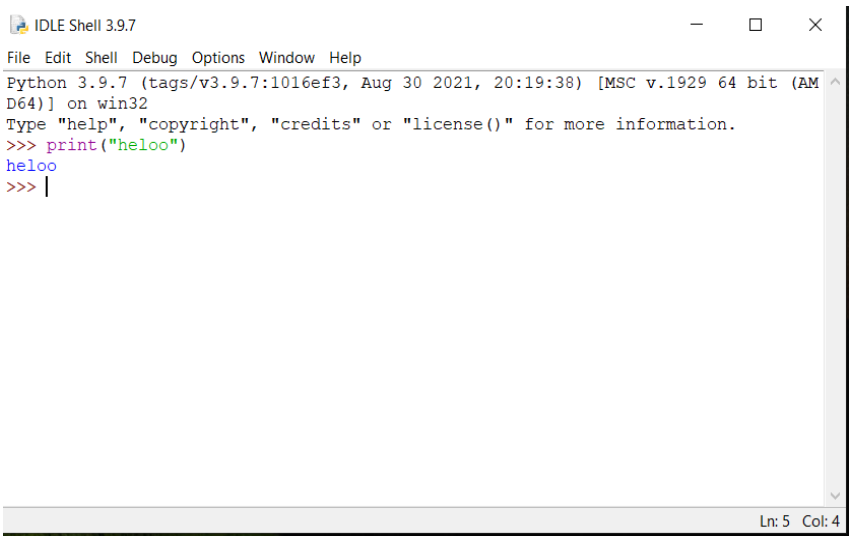
T - Template สำหรับการกำหนดโครงสร้างของเพจและการแสดงผล

V- View คือ สำหรับควบคุมกระบวนการทำงานและเชื่อมโยงระหว่าง Model และ Template

หมายเหตุ*
Open Source คือ การพัฒนาซอฟต์แวร์ตามแนวคิดที่อาศัยความร่วมมือของนักพัฒนาทั่วโลก เพื่อสร้างซอฟต์แวร์ที่ดีกว่าเดิม โดยซอฟต์แวร์ดังกล่าวสามารถเปิดเผยโค้ดต้นฉบับ (Source Code)

เริ่มการสร้าง Django ภาษาที่ใช้พัฒนา Python

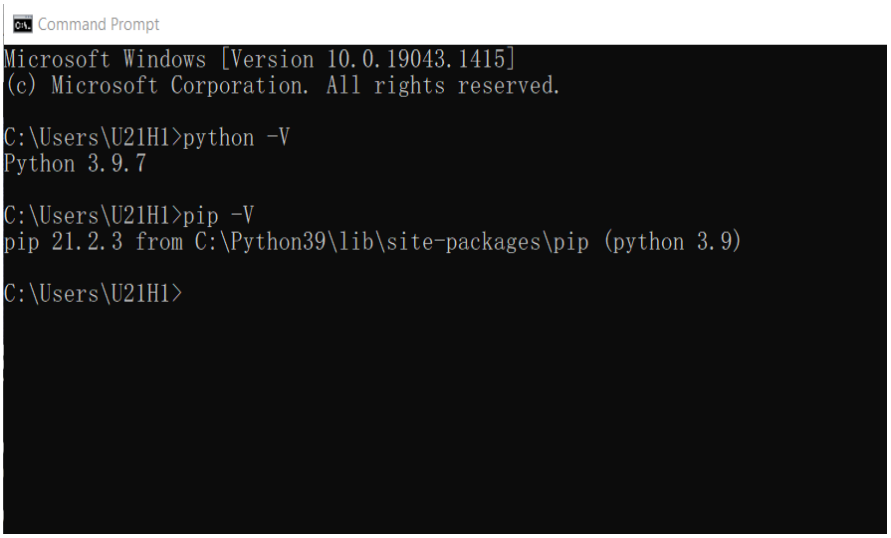
1. ให้ทำการดาวน์โหลด Python เพื่อนำมาติดตั้งโดยการไปที่ https://www.python.org/ และทำการกด Download เมื่อดาวน์โหลดเสร็จแล้ว ให้ทำการติดตั้ง โดยเลือก Path ไปที่ C:\Python39
2. เมื่อทำการติดตั้งเสร็จเรียบร้อยแล้วให้ทำการเปิด Python Shell จากนั้นให้พิมพ์ IDLE Shell ตามภาพ เพื่อเป็นการทดสอบว่า Python ที่ติดตั้งใช้งานได้หรือไม่



ภาพประกอบ 6.2 เปิด Python Shell

ที่มา : ณปภัช วรรณตรง (2564 : 7-8)

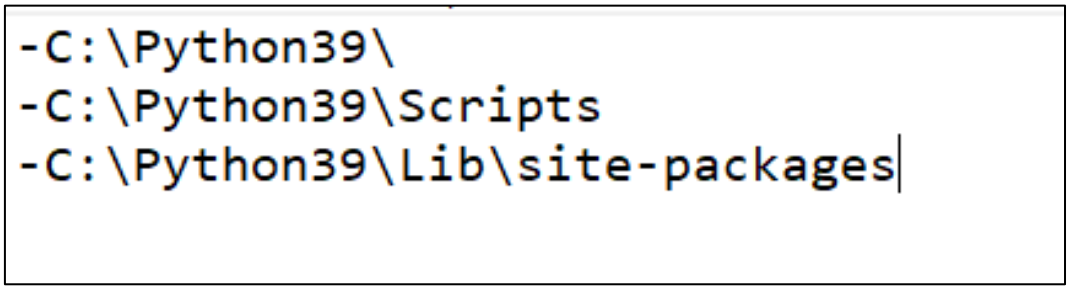
3. ทำการทดสอบ Python ด้วยการเปิดหน้าต่าง cmd จากนั้นให้ทำการพิมพ์ python -V และ pip -V ตามภาพ เพื่อเป็นการตรวจสอบเวอร์ชันของ Python ที่ติดตั้ง หากได้ผลดังภาพแสดงว่าการติดตั้ง Python เสร็จสิ้นเรียบร้อยพร้อมใช้งาน



ภาพประกอบ 6.3 เปิด Python Shell

ที่มา : ณปภัช วรรณตรง (2564 : 9)

4. หากไม่สามารถ Run ได้ ให้ทำการเข้าไปตรวจสอบที่ Path โดยการไปที่ IDLE Shell> Path Browser จากนั้นให้พิมพ์ Environment Variables > Path และแก้ไขหรือเพิ่ม Path ตามภาพ

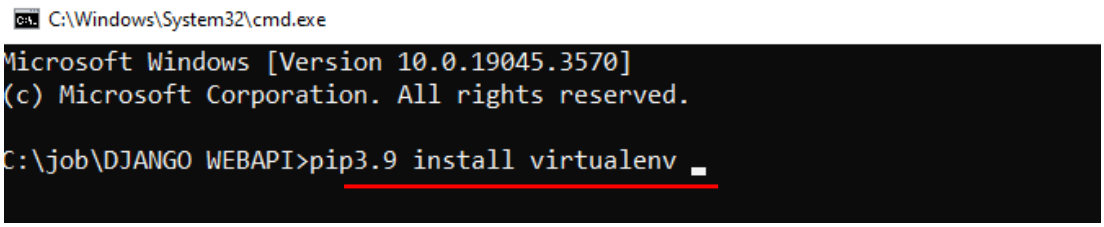


ภาพประกอบ 6.4 แก้ไขเมื่อไม่สามารถ Run ได้

ที่มา : ณปภัช วรรณตรง (2564 : 10)

ติดตั้ง Django

1. ให้ทำการสร้าง Virtual Environment โดยการเปิดหน้าต่าง cmd จากนั้นให้ทำการพิมพ์คำสั่ง pip3.9 install virtualenv ตามภาพ

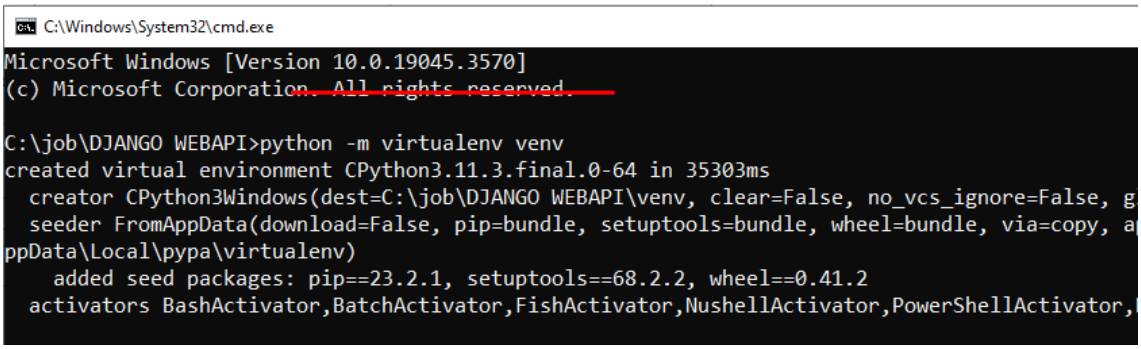


ภาพประกอบ 6.5 ติดตั้ง virtualenv

ที่มา : ณปภัช วรรณตรง (2564 : 12)

หมายเหตุ*
Virtual Environment คือ สิ่งแวดล้อมเสมือนที่สร้างขึ้นเพื่อติดตั้งโปรแกรมต่างๆ สำหรับการทดลองใช้งาน โดยจะทำให้การทำงานต่าง ๆ ของโปรแกรมที่ติดตั้งใหม่ ไม่ส่งผลต่อโปรแกรมหลักที่ได้ติดตั้งไว้ก่อนหน้านี้

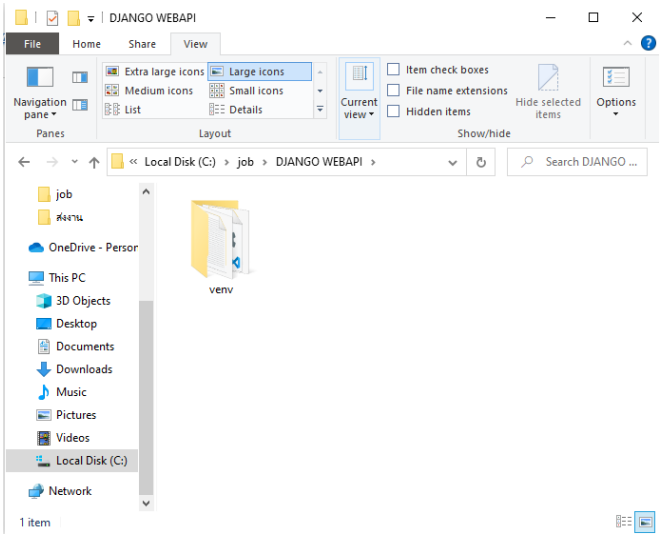
2. เมื่อสร้างไฟล์งานเสร็จแล้ว ให้ทำการสร้าง New Folder โดยตั้งชื่อว่า DJANGO WEBAPI จากนั้นให้ทำการ cd เข้าไปในโฟลเดอร์ ดังกล่าว สร้าง virtualenv ด้วยการใช้คำสั่ง python -m virtualenv venv ตามภาพ



ภาพประกอบ 6.6 สร้าง virtualenv

ที่มา : ณปภัช วรรณตรง (2564 : 13)

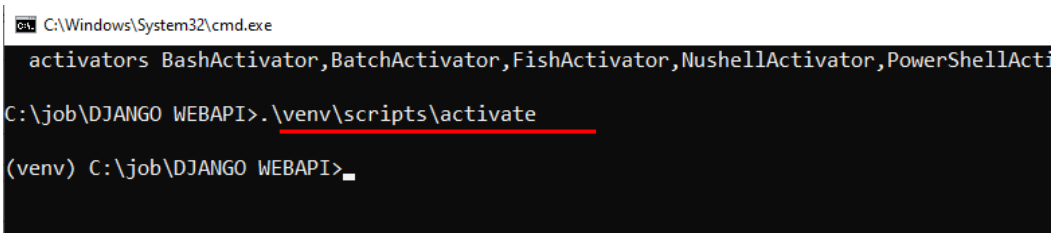
3. จากนั้นจะได้โฟลเดอร์ ขึ้นมาหนึ่งโฟลเดอร์ตามภาพ



ภาพประกอบ 6.7 ได้โฟลเดอร์ venv

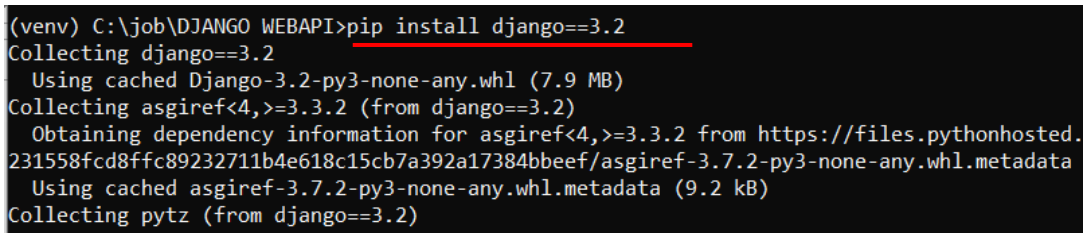
ที่มา : ณปภัช วรรณตรง (2564 : 14)

4. เมื่อได้โฟลเดอร์ venv แล้วให้กลับไปที่หน้าต่าง cmd จากนั้นให้พิมพ์คำสั่ง .\venv\scripts\activate เพื่อเข้าสู่โหมดการทำงานของ venv ตามภาพ



ภาพประกอบ 6.8 พิมพ์คำสั่งเพื่อเข้าสู่โหมด venv
ที่มา : ณปภัช วรรณตรง (2564 : 15)

5. เมื่อเข้าสู่โหมด venv แล้ว ให้ทำการติดตั้ง Django ด้วยการใช้คำสั่ง pip install django==3.2 ตามภาพ (3.2 เป็นเวอร์ชันของ Django ที่ต้องการติดตั้ง)



ภาพประกอบ 6.9 ติดตั้ง django
ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : 17)

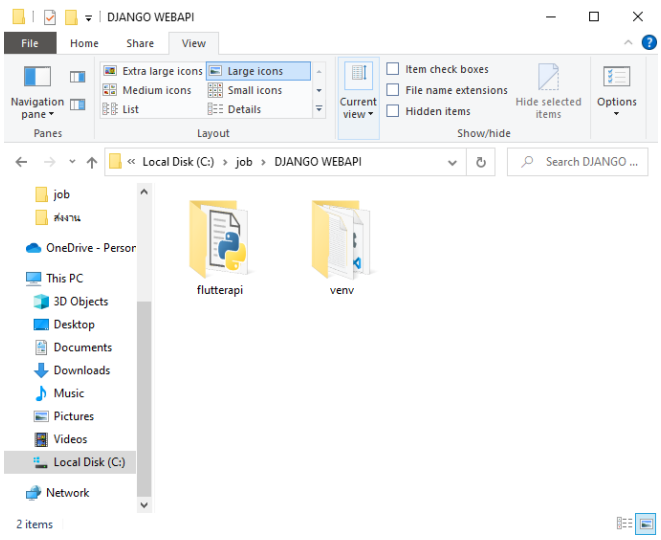
7. เมื่อตรวจสอบแล้ว ให้ทำการสร้าง Project ใน Django ด้วยการพิมพ์คำสั่ง django-admin startproject [ชื่อโปรเจกต์]

```
(venv) C:\job\DJANGO WEBAPI>django-admin startproject flutterapi

(venv) C:\job\DJANGO WEBAPI>
```

ภาพประกอบ 6.11 สร้าง Project ใน django

ที่มา : ณปภัช วรรณตรง (2564 : 18)

8. จากนั้นได้จะได้โฟลเดอร์โปรเจกต์ตามภาพ



ภาพประกอบ 6.12 โฟลเดอร์ flutterapi

ที่มา : ณปภัช วรรณตรง (2564 : 19)

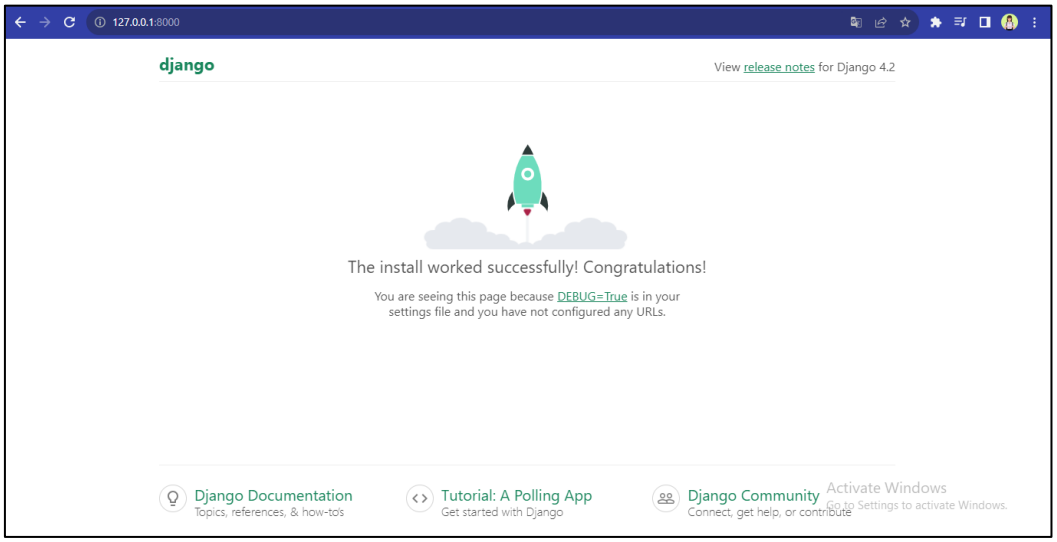
9. เปิดหน้าต่าง cmd และทำการ cd เข้าไปในโฟลเดอร์สร้างขึ้น จากนั้นให้ทำการ Run Server โดยการใช้คำสั่ง python manage.py runserver ตามภาพ

```
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 appl
 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.
```

ภาพประกอบ 6.13 Run Server

ที่มา : ณปภัช วรรณตรง (2564 : 20)

10. ให้ทำการทดสอบหน้าเว็บด้วยการเข้าไปที่ Web Browser จากนั้นให้ทำการพิมพ์ localhost:8000 ที่ Address Bar จะได้ผลลัพธ์ตามภาพ แสดงว่าการติดตั้ง Django เสร็จสมบูรณ์แล้ว

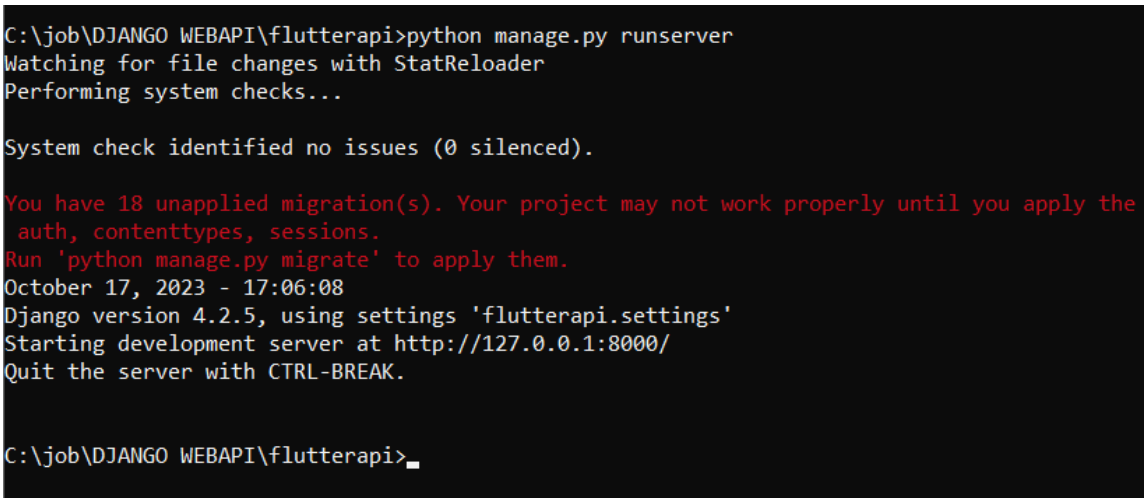


ภาพประกอบ 6.14 Web API django

ที่มา : ณปภัช วรรณตรง (2564 : 21)

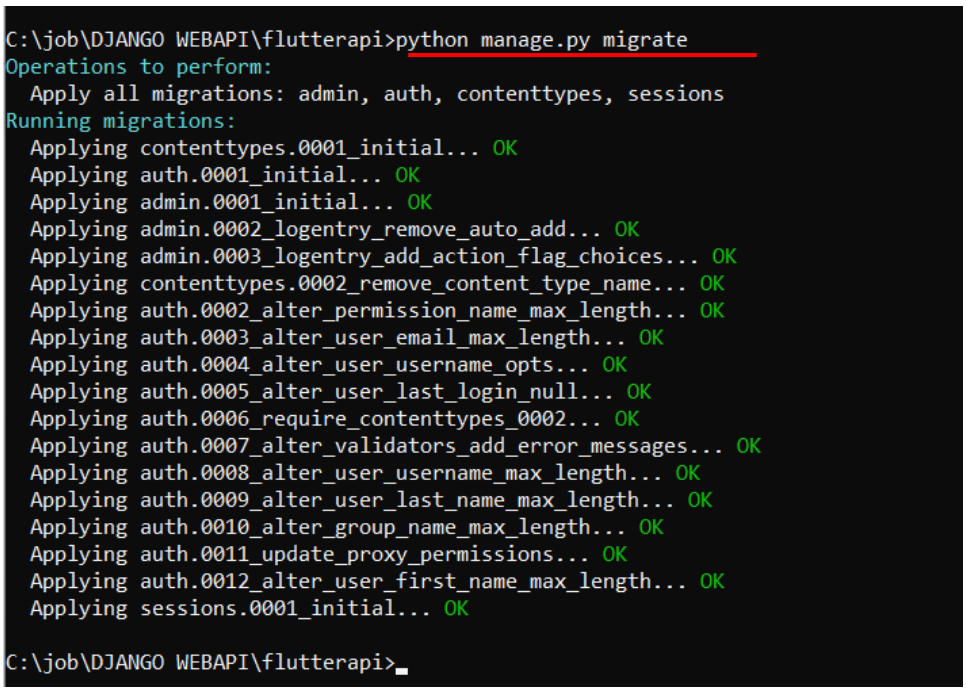
หมายเหตุ*
localhost : การสั่งทำงานบนเครื่องที่ใช้งานอยู่ 8000 : หมายเลขพอร์ตคอมพิวเตอร์ที่สั่งทำงาน

11. เมื่อได้ผลลัพธ์ที่ต้องการแล้ว ให้กลับไปที่ cmd และทำการกด Ctrl + c เพื่อทำการสั่งหยุดการทำงานของ Web Server ตามภาพ



ภาพประกอบ 6.15 สั่งหยุดการทำงานของ Web Server
ที่มา : ณปภัช วรรณตรง (2564 : 22)

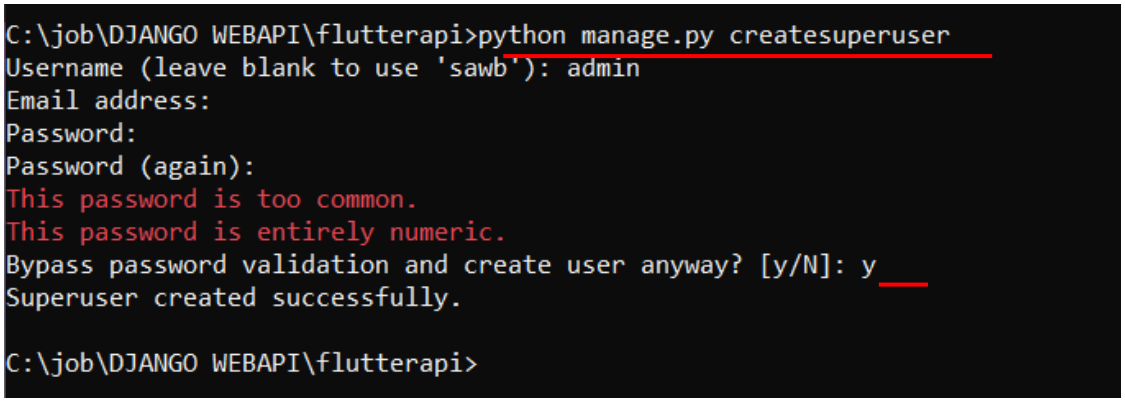
12. ให้ทำการสร้างฐานข้อมูลด้วยการพิมพ์คำสั่ง python manage.py migrate ในหน้าต่าง cmd ตามภาพ



ภาพประกอบ 6.16 สร้างฐานข้อมูล

ที่มา : ณปภัช วรรณตรง (2564 : 23)

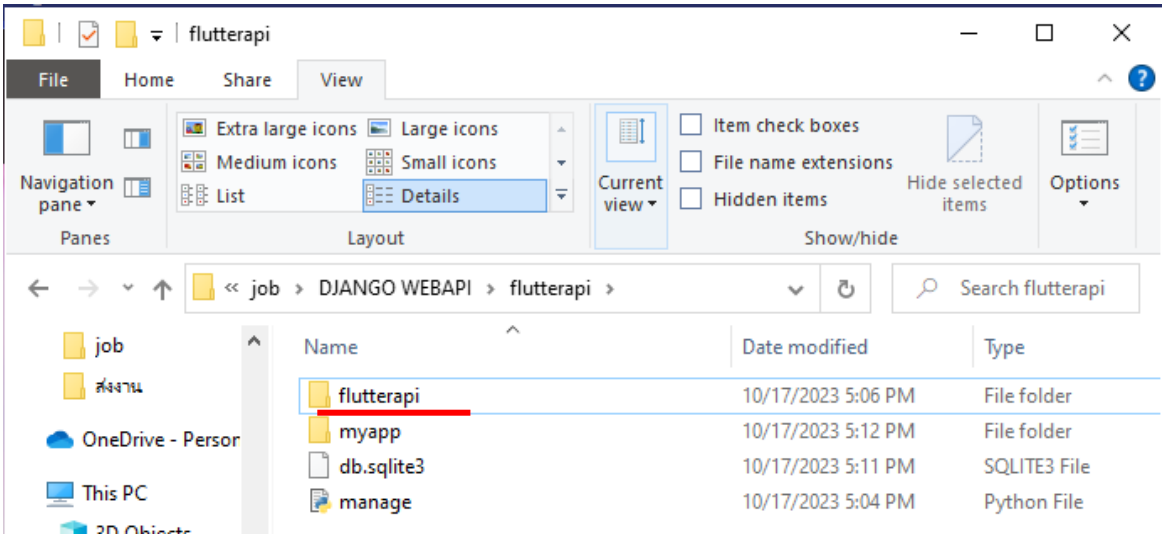
13. เมื่อสร้างฐานข้อมูลเสร็จสิ้นแล้วให้ทำการสร้างหน้าของผู้ดูแลระบบด้วยการใช้คำสั่ง python manage.py createsuperuser จากนั้นจะขึ้นให้กรอกข้อมูลให้ครบ (ไม่จำเป็นต้องกรอก e-mail) เมื่อกรอกครบแล้วให้ทำการกดตกลง (y) ตามภาพ



ภาพประกอบ 6.17 สร้างหน้าของผู้ดูแลระบบ

ที่มา : ณปภัช วรรณตรง (2564 : 24)

14. หลังจากที่ได้ทำการสร้างหน้าผู้ดูแลระบบเรียบร้อยแล้ว ให้ทำการพิมพ์คำสั่ง python manage.py startapp [ชื่อแอปพลิเคชันย่อยที่ต้องการสร้าง] ในหน้าต่าง cmd เพื่อสร้างแอปพลิเคชันย่อยขึ้นมาในโฟลเดอร์ หากสร้างเสร็จแล้วจะได้โฟลเดอร์ขึ้นมาตามภาพ



ภาพประกอบ 6.18 สร้างโฟลเดอร์แอปพลิเคชันย่อย

ที่มา : ณปภัช 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 : ณปภัช วรรณตรง (2564 : 26)

สร้าง API เพื่อทำการส่ง json File ใน Django

1. API (Application Programming Interfaces) คืออะไร

API (Application Programming Interfaces) คือ ชุดข้อมูลและฟังก์ชันเพื่ออำนวยความสะดวกในการโต้ตอบระหว่างโปรแกรมคอมพิวเตอร์ (Masse M., 2011)

การให้บริการข้อมูลบนเว็บเซิร์ฟเวอร์ที่สร้าง มาเพื่อรองรับความต้องการข้อมูลของเว็บไซต์หรือแอปพลิเคชันอื่น ๆ โดยโปรแกรมหรืออุปกรณ์ที่ เป็นเครื่องลูกข่ายจะใช้งาน API เพื่อสื่อสารกันกับงานบริการเว็บ ซึ่งโดยทั่วไปแล้ว API จะเปิดเผย ชุดข้อมูลและฟังก์ชันการใช้งานสำหรับการโต้ตอบระหว่างเครื่องลูกข่ายที่ร้องขอข้อมูลบริการกับผู้ให้บริการ และจะอนุญาตให้เกิดการแลกเปลี่ยนข้อมูลกันได้ (เอกรินทร์ วทัญญูญูเลิศสกุล, 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)

เป็นรูปแบบการสื่อสารที่ช่วยอำนวยประโยชน์ ในการส่งข้อมูลจำนวนมากโดยอาศัยรูปแบบของข้อความปกติ ซึ่งเป็นพื้นฐานการสื่อสารที่ทำได้ง่าย รวดเร็ว และเป็นมาตรฐานที่ทุกแพลตฟอร์ม ทุกภาษาสามารถใช้ได้ แต่ด้วยเหตุผลของข้อมูลที่อาจจะต้องมีรายละเอียด หมวดหมู่ หรือความหลากหลายมากมาย เจสันได้ออกแบบมาให้สามารถบรรจุข้อมูลที่มีความเป็นระเบียบเรียบร้อย และสามารถสกัดข้อมูลเหล่านั้นออกมาเพื่อนำไปใช้งานได้โดยง่าย รูปแบบการสื่อสารของเจสันโดยทั่วไปจะเขียนอยู่ภายในวงเล็บ {} ได้ (เอกรินทร์ วทัญญูญูเลิศสกุล, 2563 : 134)

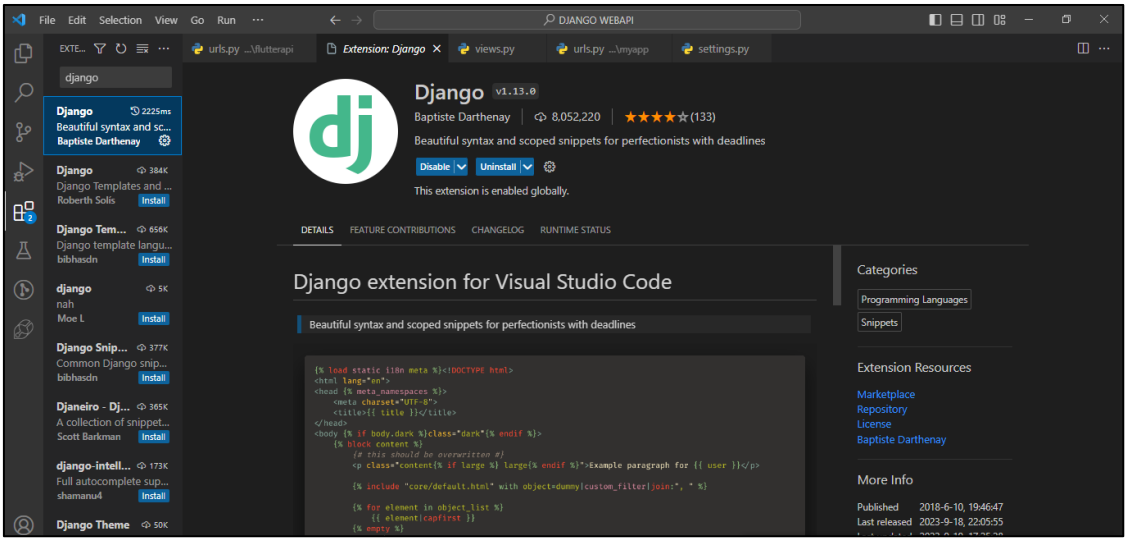
ดังนั้นโดยสรุป API คือ ชุดข้อมูลและฟังก์ชันเพื่ออำนวยความสะดวกในการโต้ตอบระหว่างโปรแกรมคอมพิวเตอร์ และ json คือ รูปแบบข้อมูลที่ช่วยให้แอปพลิเคชันสามารถสื่อสารผ่านเครือข่าย ซึ่งในส่วนนี้ผู้เรียนจะได้สร้าง API เพื่อทำการส่ง json File ใน Django ดังมีรายละเอียดขั้นตอนดังต่อไปนี้

1. ให้ทำการเปิดโปรแกรม VSCode ขึ้นมาจากนั้นให้ทำการติดตั้งส่วนเสริม Python Extension ตามภาพ

ภาพประกอบ 6.20 เปิด VSCode และทำการติดตั้งส่วนเสริม

ที่มา : ณปภัช วรรณตรง (2564 : 28)

2. จากนั้นให้ทำการลงส่วนเสริม Django



ภาพประกอบ 6.21 ติดตั้งส่วนเสริม django

ที่มา : ณปภัช วรรณตรง (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 : ณปภัช วรรณตรง (2564 : 30)

4. ให้กลับไปที่โฟลเดอร์แอปพลิเคชันย่อยที่สร้าง และทำการสร้างไฟล์ใหม่โดยตั้งชื่อว่า urls.py

ภาพประกอบ 6.23 สร้างไฟล์ใหม่

ที่มา : ณปภัช วรรณตรง (2564 : 31)

5. นำโค้ดที่คัดลอกมา มาวางไว้ในไฟล์ urls.py ของแอปพลิเคชันย่อยที่สร้าง และเพิ่มโค้ดตามภาพ

```python
from django.urls import path
from .views import *
# * ดึงมาทุกฟังชั่นใน views.py

urlpatterns = [
    path('', Home),
]
```

ภาพประกอบ 6.24 วางโค้ดที่คัดลอกมาและเพิ่มโค้ด
ที่มา : ณปภัช วรรณตรง (2564 : 32)

6. ให้ทำการตั้งค่าโปรเจกต์งาน โดยการไปที่ไฟล์ settings.py และให้ทำการตั้งค่าโดยใส่ '*' ใน [ ] ของ ALLOWED_HOSTS ตามภาพ

```python
DEBUG = True

ALLOWED_HOSTS = ['*']

```

ภาพประกอบ 6.25 ใส่ '*' ใน [ ] ของ tag ALLOWED_HOSTS
ที่มา : ณปภัช วรรณตรง (2564 : 33)

จากนั้นไปที่ INSTALLED_APPS และพิมพ์ชื่อแอปพลิเคชันย่อยที่สร้าง เพื่อเป็นลงทะเบียนแอปพลิชัน

```python
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'myapp',
    'rest_framework'
]
```

ภาพประกอบ 6.26 ใส่ 'myapp' ใน [ ] ของ tag INSTALLED_APPS
ที่มา : ณปภัช วรรณตรง (2564 : 34)

7. จากนั้นให้กลับไปที่ไฟล์ 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 : ณปภัช 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 : ณปภัช วรรณตรง (2564 : 36)

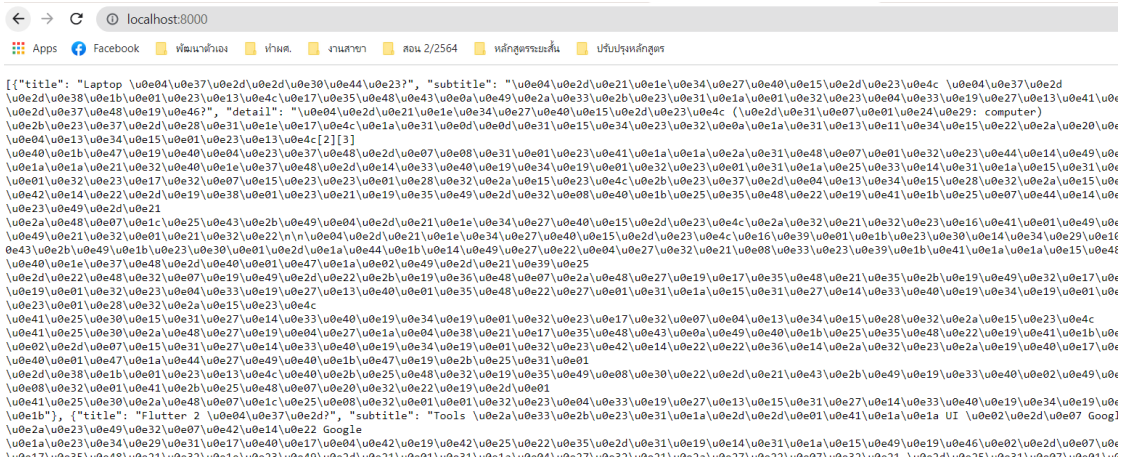
8. เปิดไฟล์ views.py ของแอปพลิเคชันย่อยที่สร้างและทำการเพิ่มโค้ดตามภาพ ซึ่งเป็นใส่ข้อมูลที่จะส่งผ่าน API

```
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) หรือศัพท์บัญญัติราชบัณฑิตยสภาว่า คณิตกรณ์[2][3]
35
36      },
37      {
38          "title":"Flutter 2 คือ?",
39          "subtitle":"Tools สำหรับออกแบบ UI ของ Google ",
40          "image_url":"https://raw.githubusercontent.com/UncleEngineer/BasicAPI/main/mobil
41          "detail":"Flutter 2 สร้างโดย Google บริษัทเทคโนโลยีอันดับต้นๆของโลก ที่มาพร้อมกับความสวย
42      },
43      {
44          "title":"Python คือ?",
45          "subtitle":"ภาษาเขียนโปรแกรมชนิดหนึ่ง สร้างขึ้นเมื่อ 1991",
46          "image_url":"https://raw.githubusercontent.com/UncleEngineer/BasicAPI/main/codin
47          "detail":"ภาษาไพทอน (Python programming language) หรือที่มักเรียกกันว่าไพทอน เป็นภาษาร
48      }
49
50  ]
51
52  def Home(request):
53      return JsonResponse(data=data,safe=False, )
```

ภาพประกอบ 6.29 เพิ่มโค้ดในไฟล์ views.py

ที่มา : ณปภัช วรรณตรง (2564 : 37)

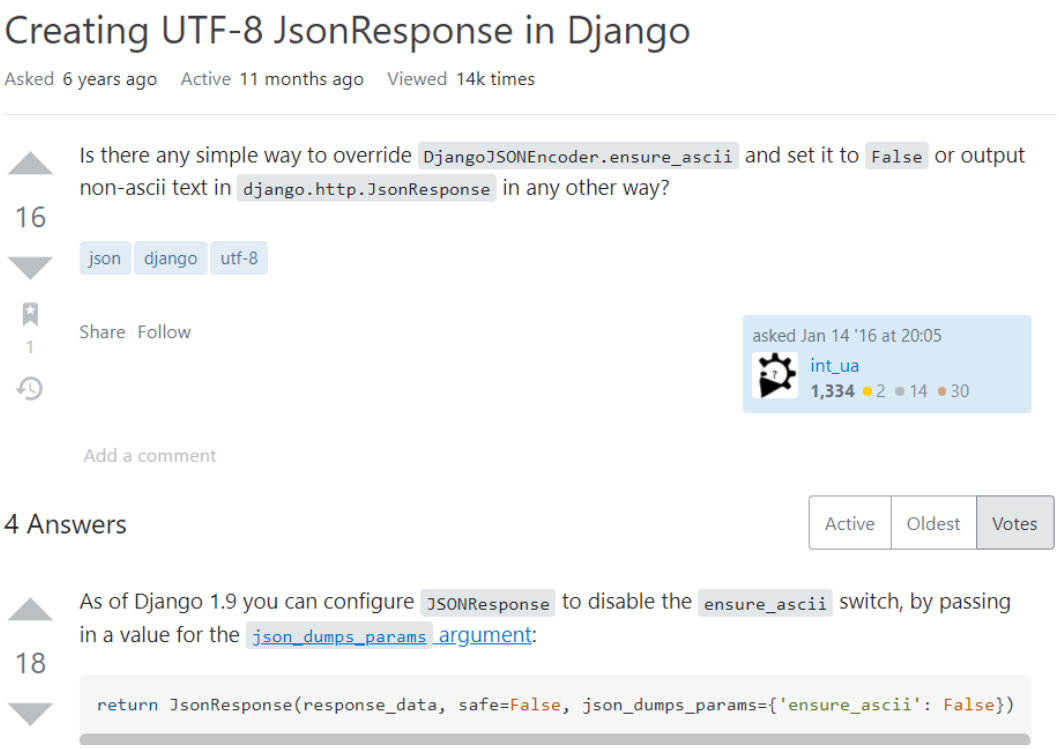
9. บันทึกไฟล์ views.py ของแอปพลิเคชันย่อยที่สร้าง และเข้าไปที่ localhost:8080 ซึ่งจะได้ผลลัพธ์ตามภาพ



ภาพประกอบ 6.30 ได้ผลลัพธ์ดังนี้

ที่มา : ณปภัช วรรณตรง (2564 : 38)

10. จากภาพประกอบ 6.30 จะเห็นได้ว่าภาษาที่แสดงไม่สามารถอ่านเข้าใจได้ ให้ทำการแก้ไขเป็น utf8 ด้วยการใช้โค้ดจากผู้พัฒนาท่านอื่น โดยการไปที่ stackoverflow และคัดลอกโค้ดในส่วน return JsonRespone() ตามภาพ



Creating UTF-8 JsonResponse in Django

Asked 6 years ago Active 11 months ago Viewed 14k times

16

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?

json django utf-8

1

Share Follow

asked Jan 14 '16 at 20:05

int_ua

1,334 ● 2 ● 14 ● 30

Add a comment

4 Answers

Active Oldest Votes

18

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:

```
return JsonResponse(response_data, safe=False, json_dumps_params={'ensure_ascii': False})
```

ภาพประกอบ 6.31 ค้นหาวิธีแก้ไข utf8 และคัดลอก return JsonRespone()

ที่มา : ณปภัช วรรณตรง (2564 : 39)

หมายเหตุ* ลิงก์วิธีแก้ไข utf8 ข้างต้น
https://stackoverflow.com/questions/34798703/creating-utf-8-jsonresponse-in-django

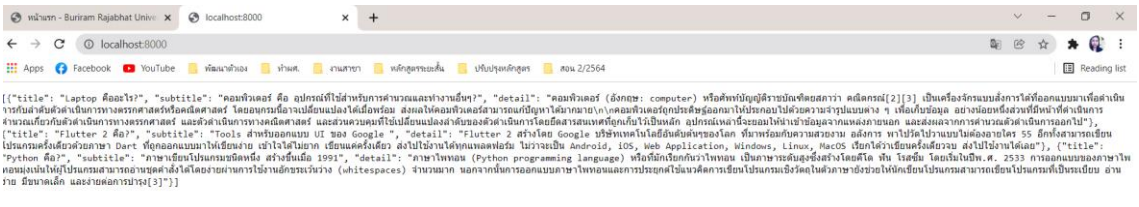
11. จากนั้นให้วางโค้ดในส่วน return JsonRespone() แทน return JsonRespone() โค้ดเก่าในไฟล์ view.py ของแอปพลิเคชันย่อยที่สร้าง ซึ่งจะเห็นได้ว่ามีการเพิ่มส่วน json_dumps_params={'ensure_ascii': False}

```
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/compu
34          "detail":"คอมพิวเตอร์ (อังกฤษ: computer) หรือศัพท์บัญญัติราชบัณฑิตยสภาว่า คณิตกรณ์[2][3]
35
36      },
37      {
38          "title":"Flutter 2 คือ?",
39          "subtitle":"Tools สำหรับออกแบบ UI ของ Google ",
40          "image_url":"https://raw.githubusercontent.com/UncleEngineer/BasicAPI/main/mobil
41          "detail":"Flutter 2 สร้างโดย Google บริษัทเทคโนโลยีอันดับต้นๆของโลก ที่มาพร้อมกับความสวยง
42      },
43      {
44          "title":"Python คือ?",
45          "subtitle":"ภาษาเขียนโปรแกรมชนิดหนึ่ง สร้างขึ้นเมื่อ 1991",
46          "image_url":"https://raw.githubusercontent.com/UncleEngineer/BasicAPI/main/codin
47          "detail":"ภาษาไพทอน (Python programming language) หรือที่มักเรียกกันว่าไพทอน เป็นภาษาร
48      }
49
50  ]
51
52  def Home(request):
53      return JsonResponse(data=data,safe=False,json_dumps_params={'ensure_ascii': False})
```

ภาพประกอบ 6.32 วางโค้ดในส่วน return JsonRespone()

ที่มา : ณปภัช วรรณตรง (2564 : 40)

12. จากนั้นให้บันทึกไฟล์และเข้าไปที่ localhost:8080 ซึ่งจะได้ผลลัพธ์ที่เปลี่ยนเป็นภาษาไทยตามภาพ



ภาพประกอบ 6.33 ผลการวางโค้ดในส่วน return JsonRespone()

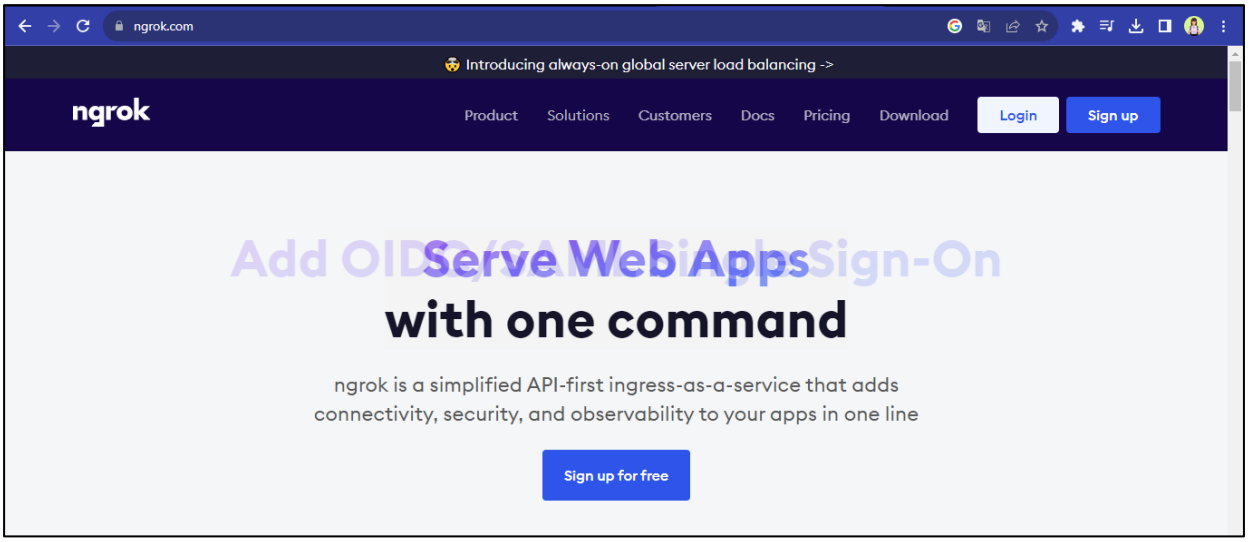
ที่มา : ณปภัช วรรณตรง (2564 : 40)

การใช้งาน ngrok Forward Port

ngrok คือ ประตูหน้าเข้าสู่แอปพลิเคชัน เป็นส่วนสำหรับติดต่อกับเครือข่ายภายนอก (Reverse Proxy) ที่กระจายไปทั่วโลกซึ่งรักษาความปลอดภัย ปกป้อง และเร่งความเร็วแอปพลิเคชันและบริการเครือข่ายไม่ว่าจะเรียกใช้งานจากที่ใดก็ตาม ngrok รองรับการส่งแอปพลิเคชันที่ใช้ HTTP, TLS หรือ TCP (ngrok, 2023)

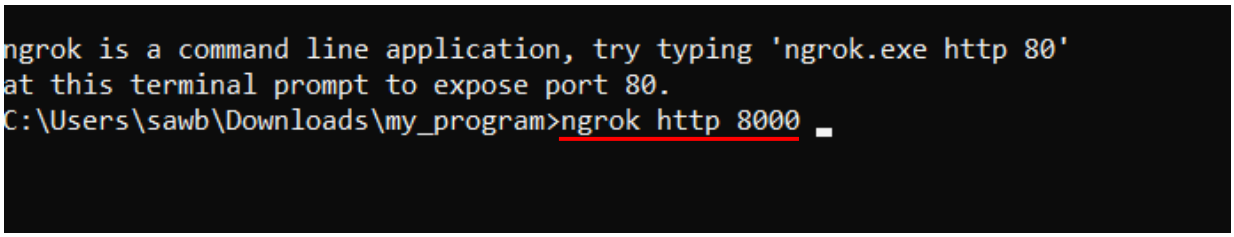
วิธีการดาวน์โหลด การติดตั้ง การใช้งานการใช้งาน ngrok สำหรับ Forward Port มีขั้นตอนดังต่อไปนี้

1. ทำการดาวน์โหลด ngrok และติดตั้ง โดยค้นหาจาก Web Browser ได้ตามภาพ



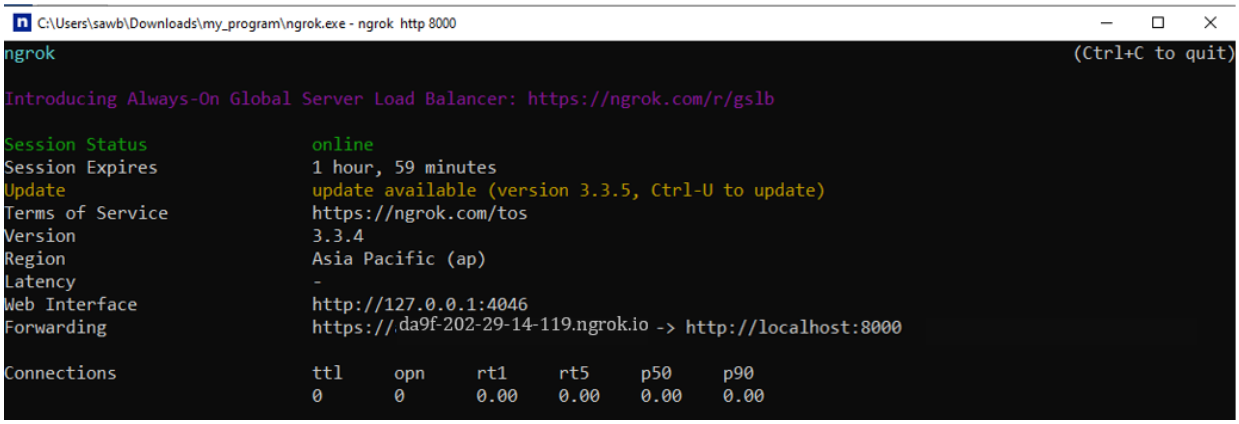
ภาพประกอบ 6.34 ดาวน์โหลดและติดตั้ง ngrok
ที่มา : ณปภัช วรรณตรง (2564 : 42)

2. เปิดหน้าต่าง cmd ของไฟล์งานที่ได้ทำการสร้างเอาไว้ แล้วทำการพิมพ์คำสั่ง ngrok http 8000 ตามภาพ เพื่อทำให้ข้อมูลในพอร์ต 8000 ถูกส่งออกไปภายนอกได้



ภาพประกอบ 6.35 พิมพ์คำสั่ง ngrok http 8000
ที่มา : ณปภัช วรรณตรง (2564 : 43)

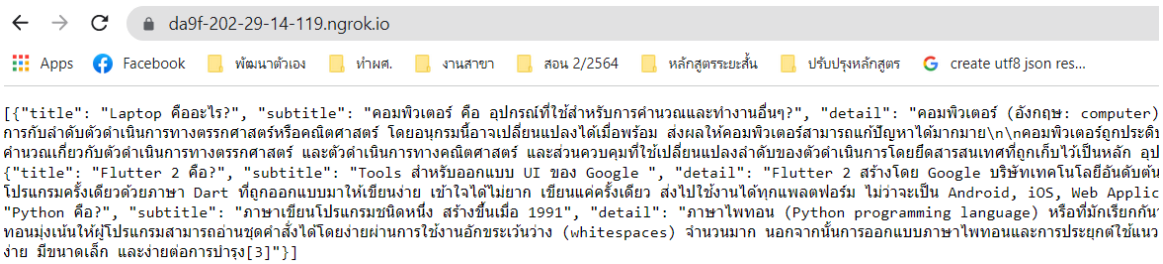
3. เมื่อทำการเรียกใช้งาน ngrok Forward Port ได้แล้ว จะได้หมายเลข url ตามภาพ (สามารถส่ง url ให้ผู้พัฒนาท่านอื่นได้ทดลองใช้งาน ใช้งานได้ฟรี 2 ชั่วโมงนับตั้งแต่เปิดใช้งาน)



ภาพประกอบ 6.36 นำไปทดลองใช้งาน

ที่มา : ณปภัช วรรณตรง (2564 : 44)

4. จากนั้นให้ทำการเข้า url ที่ได้จาก ngrok ข้างต้น จะได้ผลลัพธ์ตามภาพ



ภาพประกอบ 6.37 ผลลัพธ์การพัฒนา ngrok

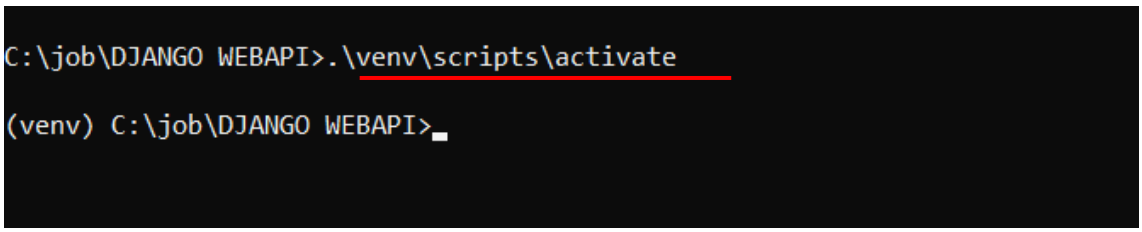
ที่มา : ณปภัช วรรณตรง (2564 : 45)

ทดสอบระบบเดิมที่ได้ทำการพัฒนาไว้

1. ให้ทำการเปิดหน้าต่าง cmd และทำการ cd เข้าไปยังไฟล์ flutter Project ที่ได้ทำการสร้างไว้ตามภาพ

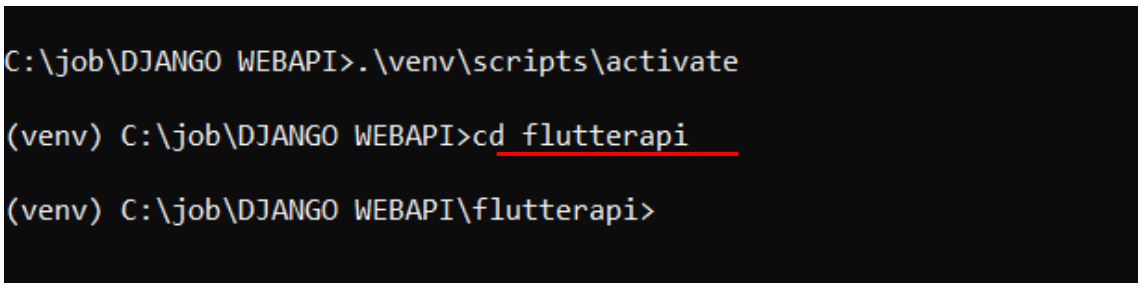
ภาพประกอบ 6.38 cd เข้าไปยังไฟล์งาน flutter Project
ที่มา : ณปภัช วรรณตรง (2564 : 5)

2. จากนั้นให้ทำการพิมพ์คำสั่ง .\venv\scripts\activate เพื่อเปิดใช้งานโหมด venv ตามภาพ



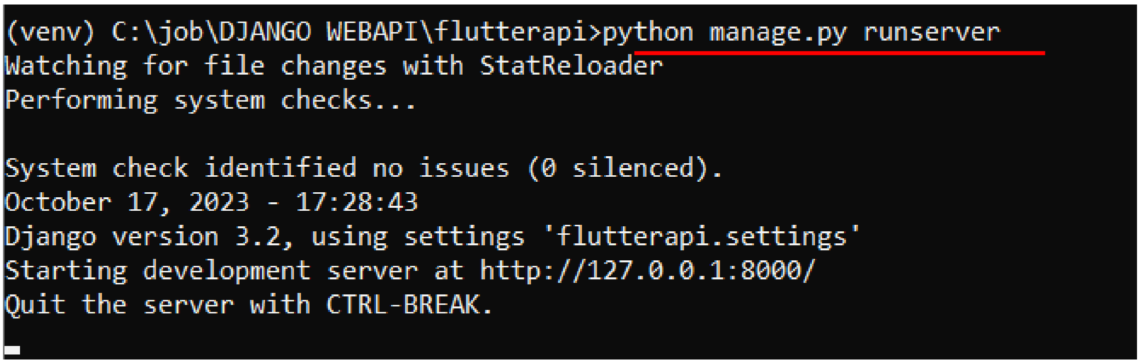
ภาพประกอบ 6.39 พิมพ์คำสั่งเปิดใช้งานโหมด venv
ที่มา : ณปภัช วรรณตรง (2564 : 6)

3. เมื่อทำการเปิดใช้งานโหมด venv ได้แล้ว ให้ทำการ cd เข้าไปในไฟล์ Project ที่ได้ทำการสร้างไว้ตามภาพ



ภาพประกอบ 6.40 cd เข้าไปยังไฟล์ Project ที่ได้ทำการสร้างไว้
ที่มา : ณปภัช วรรณตรง (2564 : 7)

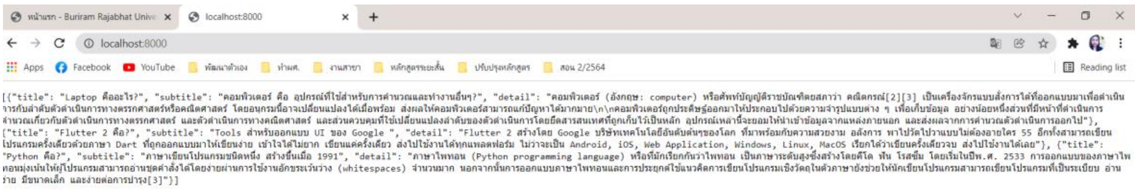
4. จากนั้นให้ทำการพิมพ์คำสั่ง python manage.py runserver สำหรับ Run Server ตามภาพ



ภาพประกอบ 6.41 cd เข้าไปยังไฟล์ Project ที่ได้ทำการสร้างไว้

ที่มา : ณปภัช วรรณตรง (2564 : 8)

5. เปิดหน้าต่าง Web Browser ขึ้นมาจากนั้นให้ทำการเข้าเว็บเซิร์ฟเวอร์โดยการพิมพ์ localhost:8000 ที่ Address Bar จะได้ผลลัพธ์ตามภาพ



ภาพประกอบ 6.42 เข้าเว็บเซิร์ฟเวอร์โดยการพิมพ์ localhost:8000

ที่มา : ณปภัช วรรณตรง (2564 : 9)

บทสรุป

สำหรับเนื้อหาที่กลาวมาทั้งหมดในบทนี้ 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 : ซีเอ็ดยูเคชั่น,

ลุงวิศวสอนคำนวณ. (11 กันยายน 2564). **สร้างเว็บไซต์ด้วย Python Django (แจงโก้) เฟรมเวิร์กใช้เขียนเว็บยอดฮิตของ Python.** สืบค้นจาก, https://www.facebook.com/UncleEngineer

เอกรินทร์ วทัญญูเลิศสกุล. (2020). พัฒนา Mobile App ด้วย Flutter & Dart. กรุงเทพฯ : โปรวิชั่น, บจก.

Bassett, L. (2015). **Bassett Introduction to JavaScript Object Notation.** O'Reilly Media, Inc.

djangoproject. (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 เป็นการระบุว่าจะให้แสดงข้อมูลไหนได้บ้างในหน้าต่าง APIs การติดตั้งโปรแกรม Postman และที่จะใช้ทดสอบการส่ง API ระหว่าง Server กับ Flutter โดน 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. ได้รับความนิยมสูงและมีคอมมิวนิตี้ขนาดใหญ่ 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

```python
from django.db import models

class Todolist(models.Model):
    title = models.CharField(max_length=100)
    detail = models.TextField(null=True, blank=True)

```

ภาพประกอบ 7.1 เปิดไฟล์งาน Project ใน VSCode และเพิ่มโค้ด

ที่มา : ณปภัช วรรณตรง (2564 : 10)

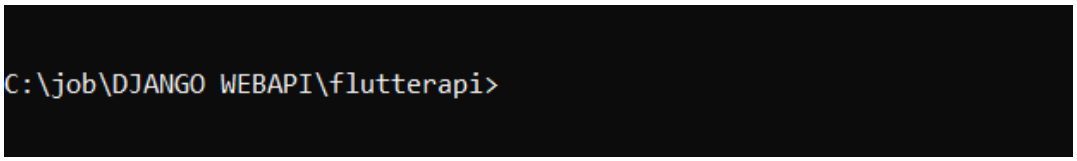
2. จากนั้นให้ไปที่ไฟล์ admin.py และทำการเพิ่มโค้ดตามภาพ ซึ่งส่วนนี้เป็นการเขียนโค้ดเพื่อลงทะเบียนแอปพลิเคชันที่จะสร้างในฐานข้อมูล

```python
from django.contrib import admin
from .models import Todolist

admin.site.register(Todolist)

```

ภาพประกอบ 7.2 เปิดไฟล์ admin.py และเพิ่มโค้ด

ที่มา : ณปภัช วรรณตรง (2564 : 11)

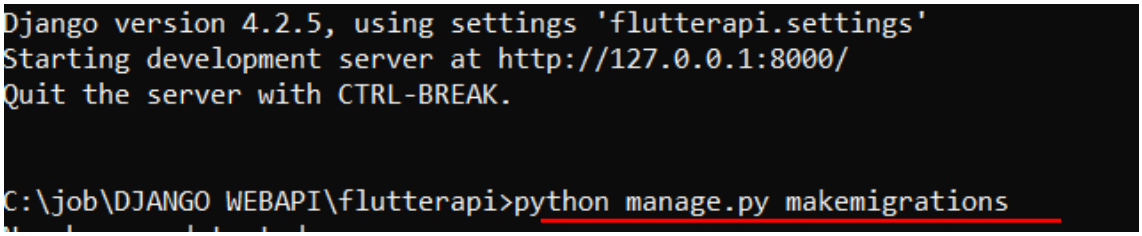
3. ให้ทำการหยุด Run Server โดยการกด Ctrl + c ที่หน้าต่าง cmd ที่ได้ทำการสั่งรันไว้ตามภาพ



ภาพประกอบ 7.3 หยุด Run Server

ที่มา : ณปภัช วรรณตรง (2564 : 13)

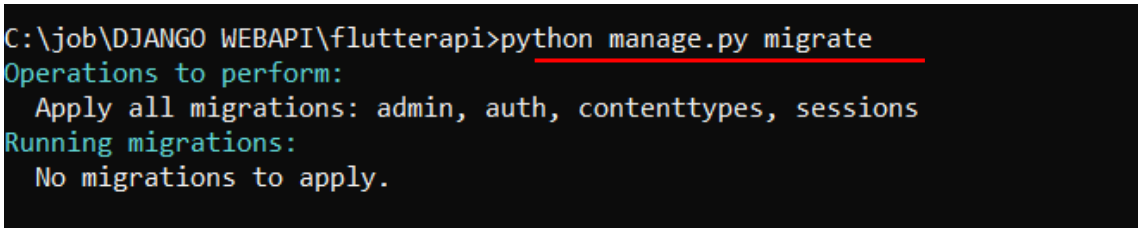
4. เมื่อหยุดการ Run Server แล้วให้ทำการพิมพ์คำสั่ง python manage.py makemigrations เพื่อทำตรวจสอบการอัพเดตการเปลี่ยนแปลงว่ามีอะไรเปลี่ยนแปลงบ้างตามภาพ



ภาพประกอบ 7.4 ตรวจสอบอัพเดตการเปลี่ยนแปลง

ที่มา : ณปภัช วรรณตรง (2564 : 14)

5. จากนั้นให้ทำการพิมพ์คำสั่ง python manage.py migrate เพื่อทำการอัพเดตการเปลี่ยนแปลงต่าง ๆ ที่มีการเปลี่ยนแปลงลงไปในฐานข้อมูล



ภาพประกอบ 7.5 อัพเดตการเปลี่ยนแปลง

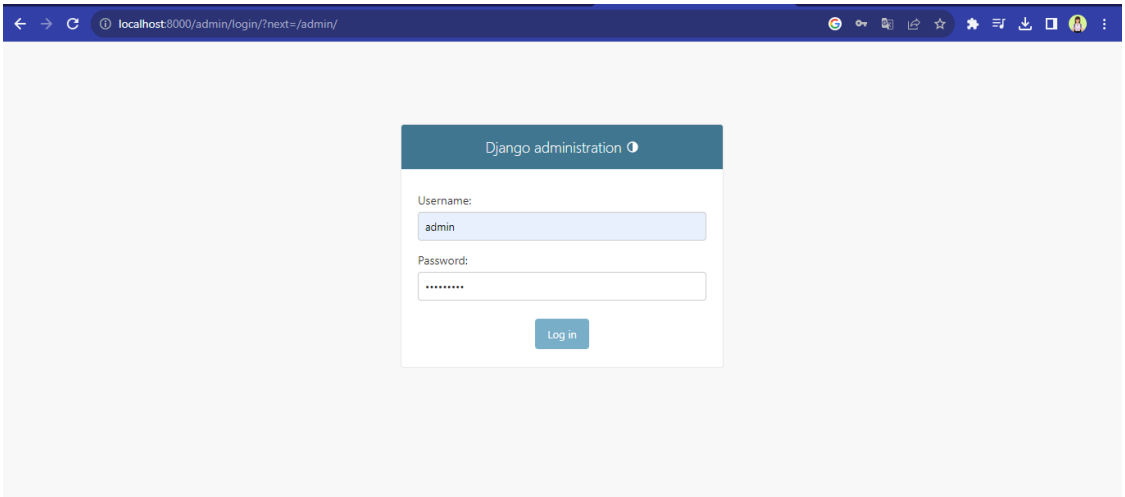
ที่มา : ณปภัช วรรณตรง (2564 : 15)

6. เมื่อทำการอัพเดตเสร็จแล้วให้ทำการสั่งการทำงาน Server ตามภาพ

```
C:\job\DJANGO WEBAPI\flutterapi>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
```

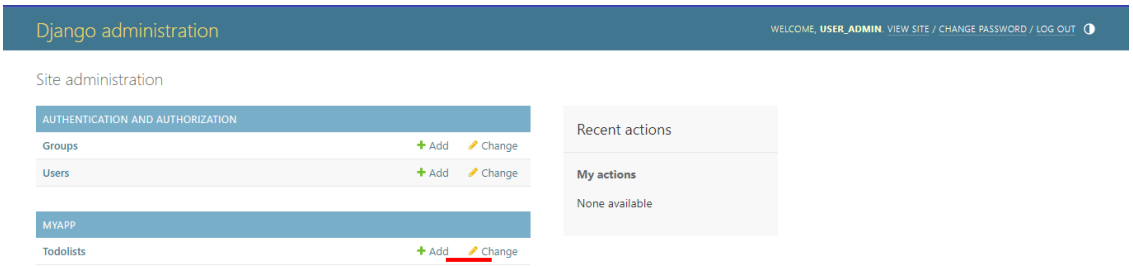
ภาพประกอบ 7.6 สั่งการทำงาน Server
ที่มา : ณปภัช วรรณตรง (2564 : 16)

7. เปิด Web Browser และทำการพิมพ์ localhost:8000/admin/ ในชื่อ Address Bar เพื่อทำการเข้าสู่หน้า Admin โดยนำข้อมูลผู้ดูแลระบบที่สร้างใน cmd มากรอกเพื่อทำการเข้าสู่ระบบ ตามภาพ



ภาพประกอบ 7.7 เข้าสู่หน้า Admin
ที่มา : ณปภัช วรรณตรง (2564 : 17)

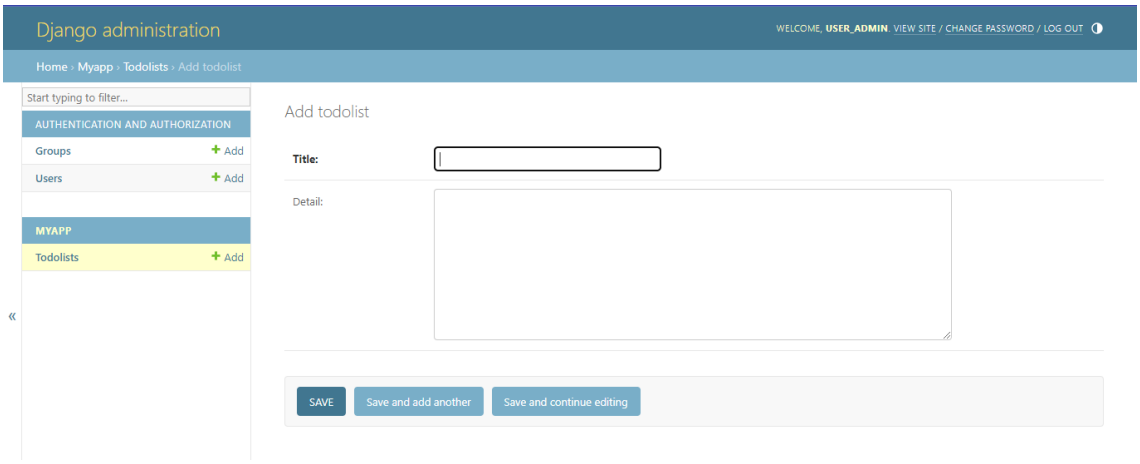
8. จากนั้นให้ทำการเข้าสู่ระบบด้วยรหัสผ่านที่ได้ทำการตั้งค่าไว้ จะเข้าสู่หน้าการจัดการฐานข้อมูลตามภาพ



ภาพประกอบ 7.8 หน้าการจัดการฐานข้อมูล

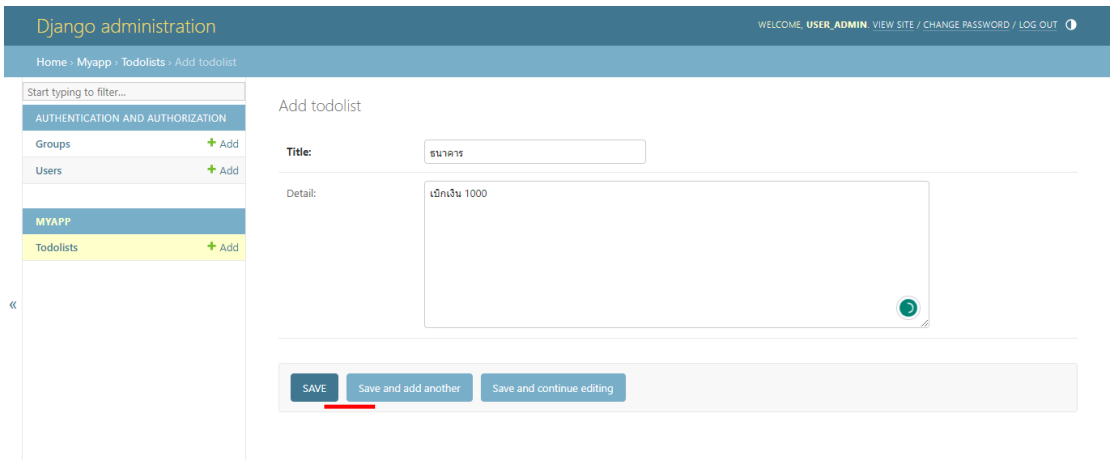
ที่มา : ณปภัช วรรณตรง (2564 : 18)

9. จากนั้นให้ทำการเข้าไปที่ Todolist > คลิก Add เพื่อทำการเพิ่มข้อมูลเข้าไปในฐานข้อมูล



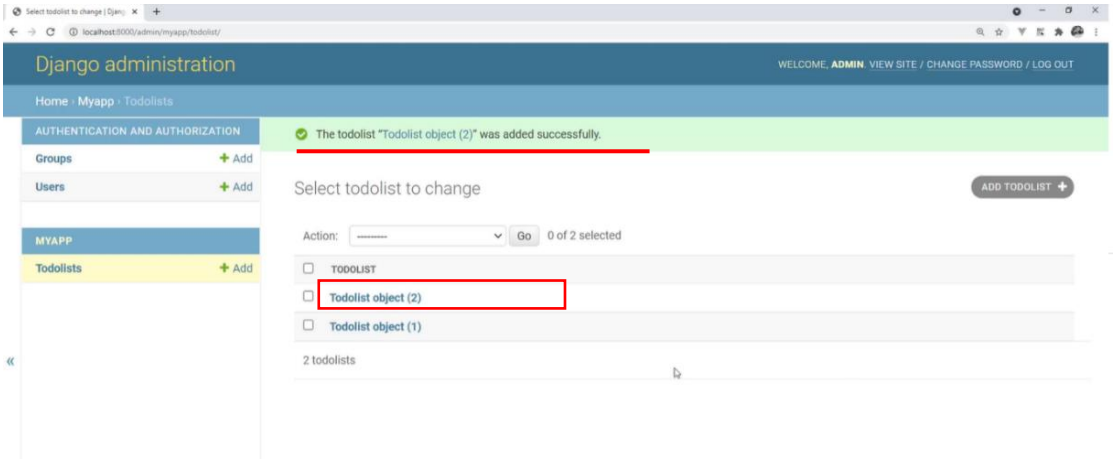
ภาพประกอบ 7.9 คลิก Add ที่ Todolist

ที่มา : ณปภัช วรรณตรง (2564 : 19)



ภาพประกอบ 7.10 กรอกข้อมูลที่ต้องการเพิ่มจากนั้นทำการกดบันทึก (Save)

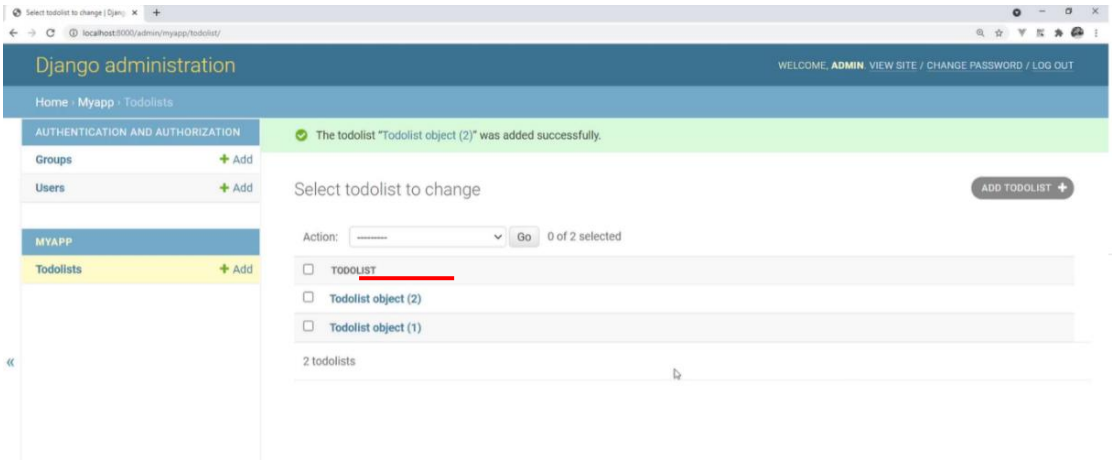
ที่มา : ณปภัช วรรณตรง (2564 : 20)



ภาพประกอบ 7.11 ได้ผลลัพธ์

ที่มา : ณปภัช วรรณตรง (2564 : 20)

10. ให้ทำการเพิ่มข้อมูลให้ได้ทั้งหมดจำนวน 2 รายการตามภาพ



ภาพประกอบ 7.12 เพิ่มข้อมูลให้ได้ 2 รายการ

ที่มา : ณปภัช วรรณตรง (2564 : 21)

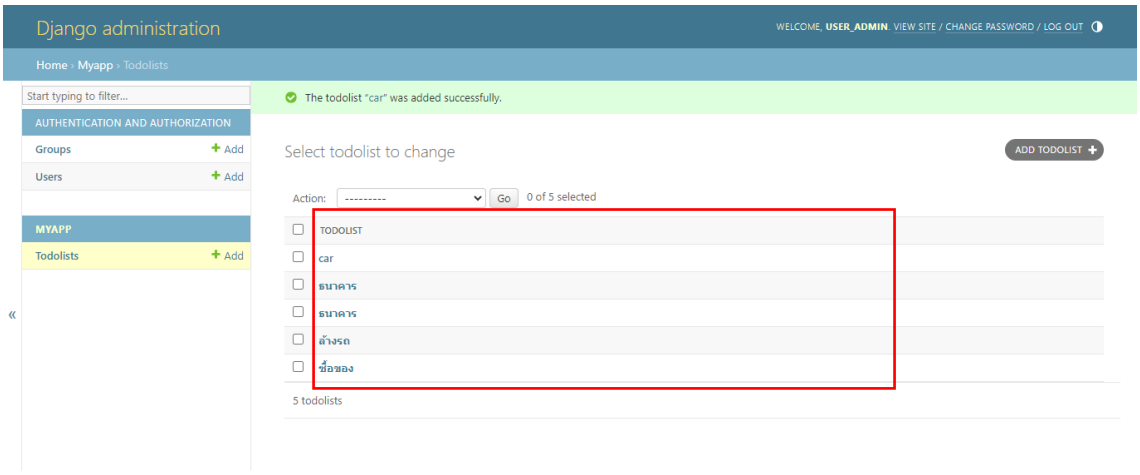
11. จากภาพข้างต้นจะเห็นได้ว่าชื่อรายการที่สร้าง มีชื่อตามระบบที่ตั้งให้อัตโนมัติ สามารถเปลี่ยนชื่อให้เป็นไปตามที่บันทึกจริง ด้วยการเปิดไฟล์ Models.py เพื่อทำการแก้ไขให้สามารถแสดง Title ในแบบที่ต้องการ โดยการเพิ่มโค้ดตามภาพ

```python
from django.db import models

class Todolist(models.Model):
    title = models.CharField(max_length=100)
    detail = models.TextField(null=True, blank=True)

    def __str__(self):
        return self.title
```

ภาพประกอบ 7.13 เพิ่มโค้ดเพื่อแสดง Title ที่ต้องการ

ที่มา : ณปภัช วรรณตรง (2564 : 22)

12. จากนั้นให้ทำการกด Refresh Web Browser แล้วจะได้ผลลัพธ์ตามภาพ



ภาพประกอบ 7.14 ผลลัพธ์การแสดง Title

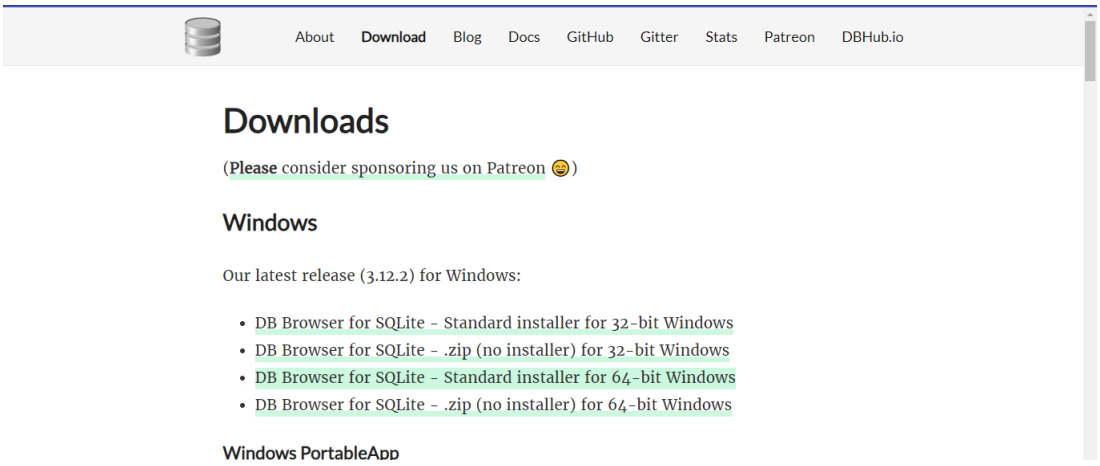
ที่มา : ณปภัช วรรณตรง (2564 : 23)

ทดสอบเข้าดูฐานข้อมูลที่มีใน DB Browser for SQLite

DB Browser สำหรับ SQLite เป็นเครื่องมือโอเพ่นซอร์สเชิงภาพคุณภาพสูงสำหรับสร้างออกแบบและแก้ไขไฟล์ฐานข้อมูลที่เข้ากันได้กับ SQLite โปรแกรมนี้ได้รับการพัฒนาโดย Mauricio Piacentini (sqlitebrowser, 2566)

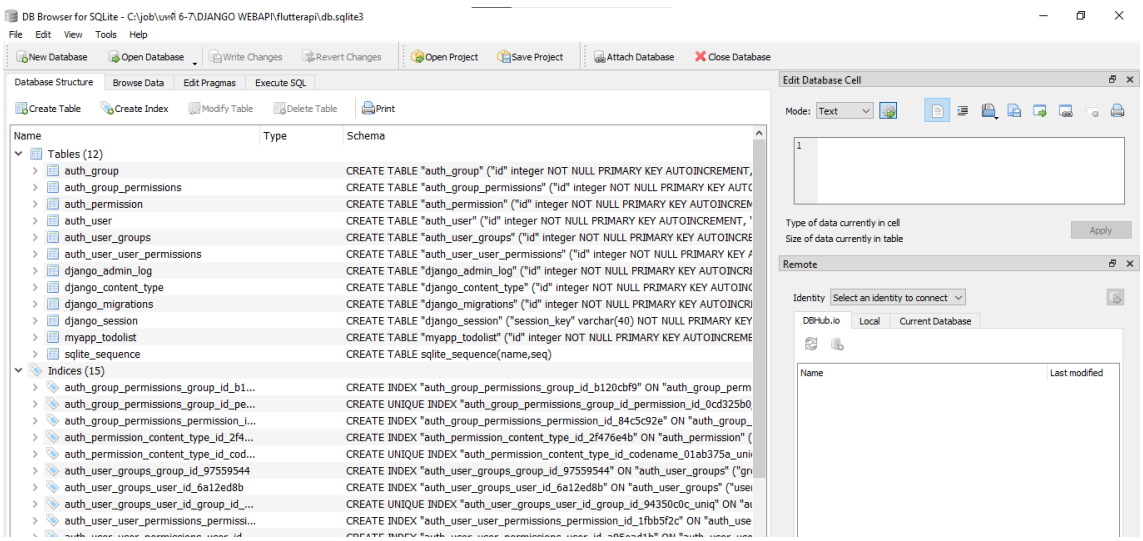
เนื้อหาส่วนนี้ทดสอบเข้าดูฐานข้อมูลที่สร้างไว้ในขั้นตอนก่อนหน้านี้ใน DB Browser for SQLite มีรายละเอียดขั้นตอนดังนี้

1. ให้ทำการดาวน์โหลดโปรแกรม DB Browser for SQLite ผ่าน Web Browser จากนั้นให้ทำการติดตั้ง



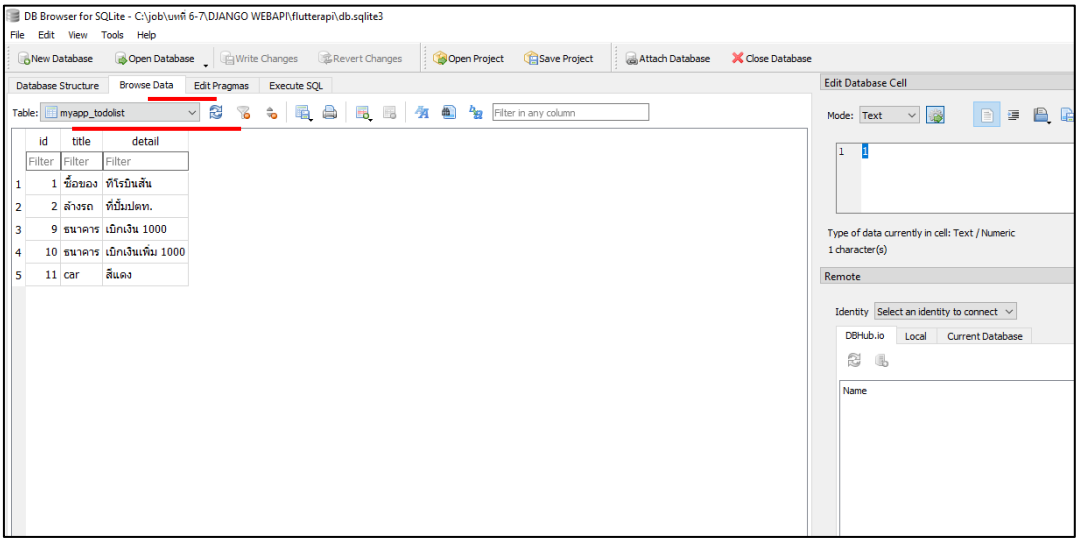
ภาพประกอบ 7.15 ดาวน์โหลดโปรแกรม DB Browser for SQLite
ที่มา : ณปภัช วรรณตรง (2564 : 25)

2. หลังจากที่ได้ทำการดาวน์โหลดและติดตั้งโปรแกรมเรียบร้อยแล้วให้ทำการเปิดโปรแกรม จากนั้นให้เลือก Open Database > เลือกไฟล์ Database ใน Project ที่ได้ทำการพัฒนาไว้ เมื่อทำการเลือกแล้วจะปรากฏโครงสร้าง Database ที่ได้ทำการสร้างไว้ตามภาพ



ภาพประกอบ 7.16 เปิดโครงสร้าง Database ในโปรแกรม
ที่มา : ณปภัช วรรณตรง (2564 : 26)

3. สามารถดูข้อมูลหรือแสดงข้อมูลในตารางได้โดยการไปที่ Browser Data > เลือกตารางที่ต้องการดูหรือแสดง ซึ่งจะได้ผลลัพธ์ตามภาพ

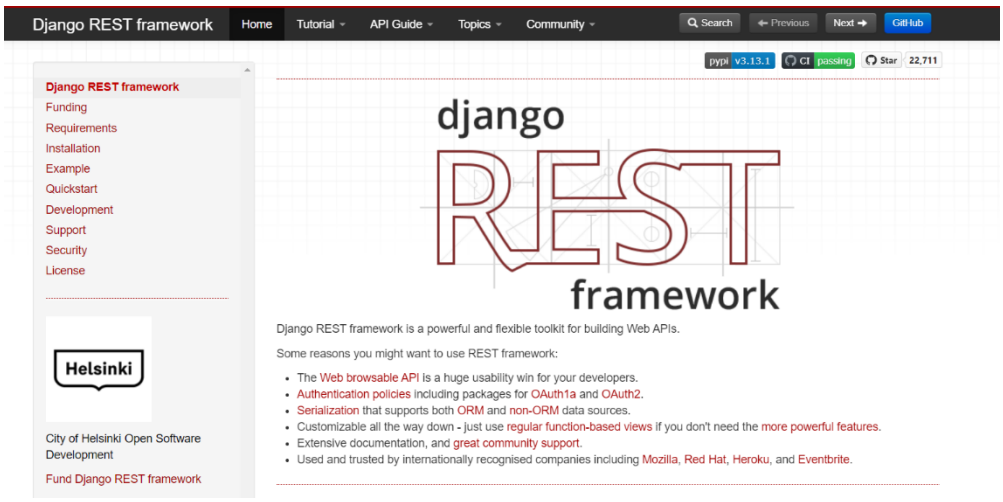


ภาพประกอบ 7.17 ดูข้อมูลหรือแสดงข้อมูลของตาราง
ที่มา : ณปภัช วรรณตรง (2564 : 27)

เริ่มต้นติดตั้ง Django REST Framework

เนื้อหาส่วนนี้เป็นการติดตั้ง Django REST Framework สำหรับนำไปใช้งาน มีรายละเอียดขั้นตอนดังนี้

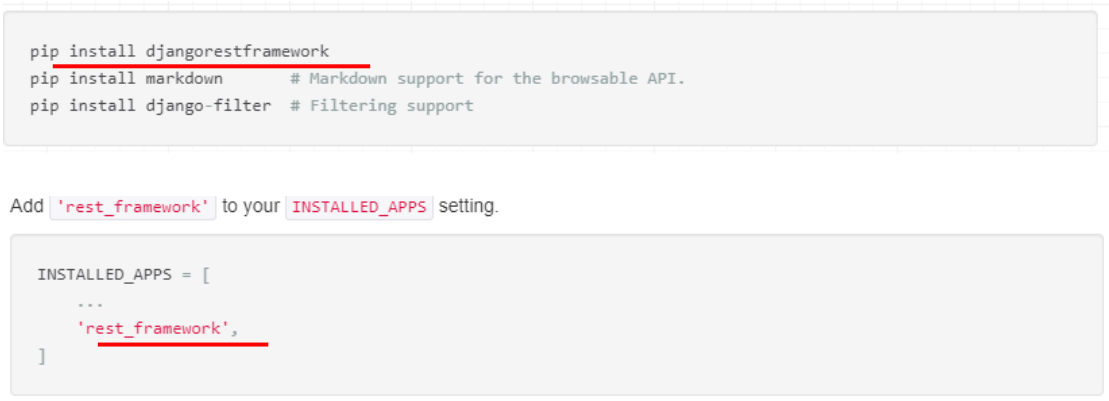
1. ให้ทำการค้นหา Django REST Framework ใน Web Browser จากนั้นให้ไปที่เว็บไซต์เพื่อทำการเริ่มต้นการติดตั้ง



ภาพประกอบ 7.18 ค้นหา Django REST Framework

ที่มา : ณปภัช วรรณตรง (2564 : 34)

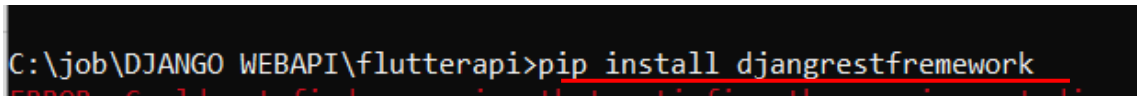
2. ไปที่ Installation จากนั้นทำการคัดลอก php install djangrestfremework และ rest_fremework



ภาพประกอบ 7.19 คัดลอกโค้ดสำหรับนำไปติดตั้ง

ที่มา : ณปภัช วรรณตรง (2564 : 35)

3. ในหน้าต่าง cmd ให้ทำการหยุด Run Server แล้วทำการวางคำสั่ง pip install djangrestfremework เพื่อติดตั้ง Django REST Framework



ภาพประกอบ 7.20 วางคำสั่ง php install djangrestfremework
ที่มา : ณปภัช วรรณตรง (2564 : 36)

4. เปิดไฟล์ setting.py จากนั้นให้ทำการวางโค้ด rest_fremework เข้าไปตามภาพ ซึ่งเป็นการลงทะเบียนสำหรับการนำไปใช้งาน

ภาพประกอบ 7.21 วางคำสั่ง rest_fremework
ที่มา : ณปภัช วรรณตรง (2564 : 37)

สร้างไฟล์ Serializers

ไฟล์ Serializers มีลักษณะคล้าย Model เป็นการระบุว่าจะแสดงข้อมูลใดหรือบันทึกข้อมูลใดได้บ้างในหน้าต่าง API เนื้อหาส่วนนี้จะเป็นการสร้างไฟล์ Serializers และเขียนโค้ดสำหรับการใช้งาน API ดังมีรายละเอียดขั้นตอนดังนี้

1. สร้างไฟล์ serializers.py ภายใต้โฟลเดอร์งานแอปพลิเคชันย่อย และทำการเพิ่มโค้ดตามภาพ

```python
from rest_framework import serializers
from .models import Todolist

class TodolistSerializer(serializers.ModelSerializer):
    class Meta:
        model = Todolist
        fields = ('id','title','detail') # '__all__'
```

ภาพประกอบ 7.22 สร้างไฟล์ serializers.py และทำการเพิ่มโค้ด

ที่มา : ณปภัช วรรณตรง (2564 : 39)

2. จากนั้นไปที่ไฟล์ views.py และทำการเพิ่มโค้ดตามภาพ

```python
from rest_framework.response import Response
from rest_framework.decorators import api_view
from rest_framework import status
from .serializers import TodolistSerializer
from .models import Todolist

```

ภาพประกอบ 7.23 ไปที่ไฟล์ views.py และทำการเพิ่มโค้ด

ที่มา : ณปภัช วรรณตรง (2564 : 40)

3. หลังจากนั้นให้ทำการเพิ่มโค้ดอีกชุดตามภาพ ซึ่งเป็นโค้ดการกำหนดว่า API สามารถทำอะไรได้บ้างและสร้างฟังก์ชันสำหรับการดึงข้อมูล โดยในฟังก์ชันจะมีการส่งข้อมูลและสเตตัสที่จะส่งกลับไปให้ฝั่งผู้รับข้อมูลด้วย

```python
from rest_framework.response import Response
from rest_framework.decorators import api_view
from rest_framework import status
from .serializers import TodolistSerializer
from .models import Todolist

# GET Data
@api_view(['GET'])
def all_todolist(request):
    alltodolist = Todolist.objects.all() #ดึงข้อมูลจาก model Todolist
    serializer = TodolistSerializer(alltodolist,many=True)
    return Response(serializer.data, status=status.HTTP_200_OK)

```

ภาพประกอบ 7.24 ทำการเพิ่มโค้ดอีกส่วน

ที่มา : ณปภัช วรรณตรง (2564 : 41)

4. ไปที่ไฟล์ urls.py ของแอปพลิเคชัน จากนั้นให้ทำการแก้ไขและเพิ่มโค้ดตามภาพ เพื่อสร้าง path url API สำหรับให้ผู้รับข้อมูลระบุ

```python
from django.urls import path
from .views import *
# * ดึงมาทุกฟังชั่นใน views.py

urlpatterns = [
    path('', Home),
    path('api/all-todolist/',all_todolist), # localhost:8000/api/all-todolist
]
```

ภาพประกอบ 7.25 เพิ่มโค้ดที่ไฟล์ urls.py

ที่มา : ณปภัช วรรณตรง (2564 : 42)

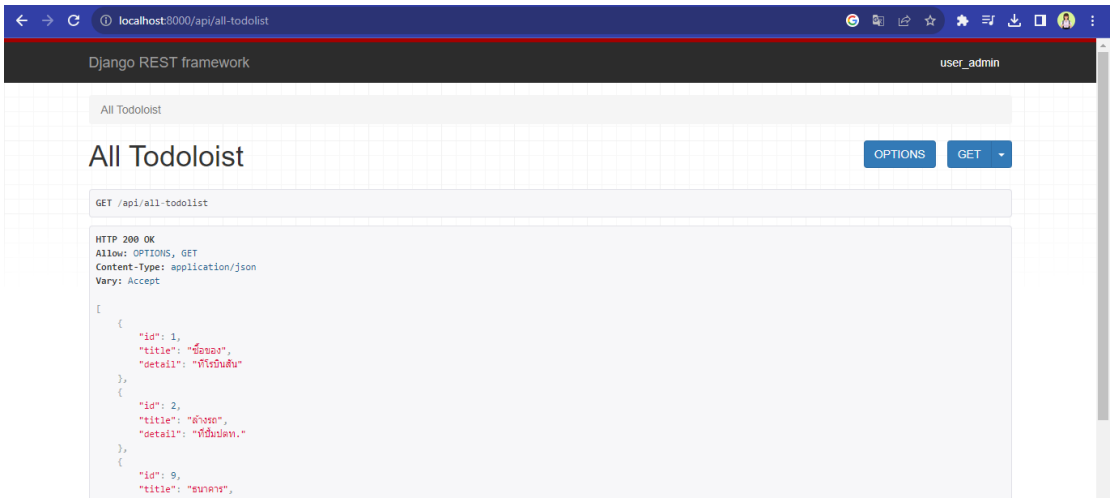
5. กลับไปที่หน้าต่าง cmd และทำการสั่งการทำงานServer

```
C:\job\DJANGO WEBAPI\flutterapi>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
```

ภาพประกอบ 7.26 สั่งการทำงาน Server

ที่มา : ณปภัช วรรณตรง (2564 : 41)

6. ทดสอบหน้าเว็บโดยการเข้าไปที่ Web Browser และทำการพิมพ์ localhost:8000/api/all-todolist/ ที่ Address Bar จะได้ผลลัพธ์ตามภาพ



ภาพประกอบ 7.27 ผลลัพธ์การสร้างไฟล์ serializers
ที่มา : ณปภัช วรรณตรง (2564 : 44)

ในเนื้อหาที่จะเรียนถัดไปจะเป็นการใช้โปรแกรม Postman ในการจำลองทดสอบการส่งข้อมูลระหว่าง Server กับ Flutter ซึ่งจะมีการเขียนโค้ดเพิ่มเติมและได้มีการกล่าวถึงและใช้งาน HTTP Methods และ Status Codes จึงขอกล่าวถึง HTTP Methods และ Status Codes ที่ผู้เรียนควรรู้ก่อนเริ่มเรียนเนื้อหาถัดไปดังตาราง

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

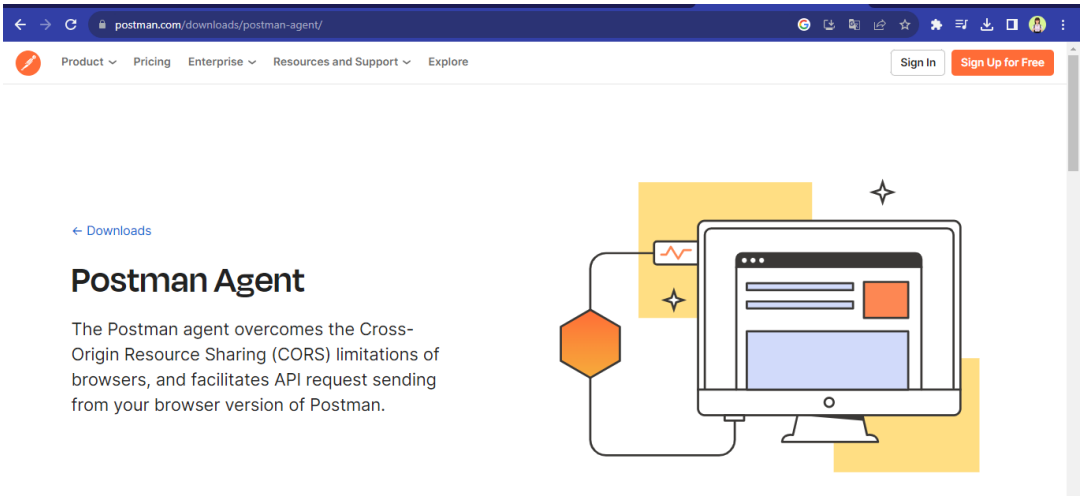
Status Codes	การบ่งบอก
200 – 299	Status ที่บ่งบอกว่า Response สำเร็จ (Successful Responses)
400 – 499	Status ที่บ่งบอกว่า Response ผิดพลาด (Client Error Responses)
500 – 599	Status ที่บ่งบอกว่า Server ผิดพลาด (Server Error Responses)
200 OK	Response สำเร็จ (เป็น Status Code ที่พบเจอบ่อยที่สุด เช่น เมื่อมีการค้นหาหน้าเว็บต่าง ๆ แล้วคลิกเข้าดูได้ Server ก็จะตอบกลับด้วย Status Code นี้)
201 Created	Response สำเร็จ เมื่อ Resource ใหม่ถูกสร้างขึ้นมา (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)

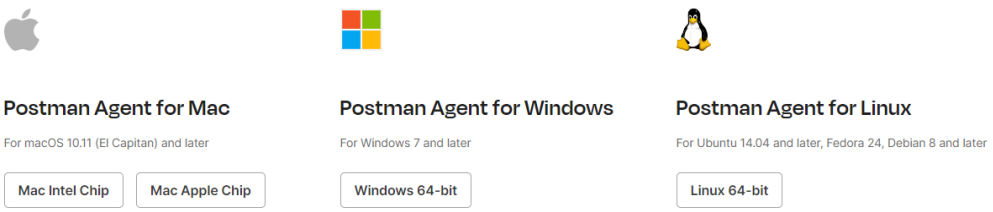
โดยในส่วนนี้จะนำโปรแกรม Postman มาใช้ในการทดสอบการส่ง API ระหว่าง Server กับ Flutter ดังมีขั้นตอนต่อไปนี้

1. ให้ทำการค้นหา Postman ที่ Web Browser จากนั้นเข้าไปที่เว็บไซต์และทำการดาวน์โหลด โดยการไปที่ Postman Desktop Agent > Download Postman Agent ตามภาพ



ภาพประกอบ 7.28 ค้นหา Postman

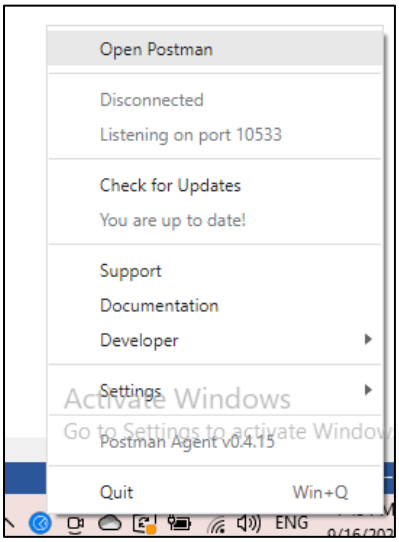
ที่มา : ณปภัช วรรณตรง (2564 : 46)



ภาพประกอบ 7.29 ดาวน์โหลดโปรแกรมสำหรับระบบปฏิบัติการที่ต้องการ

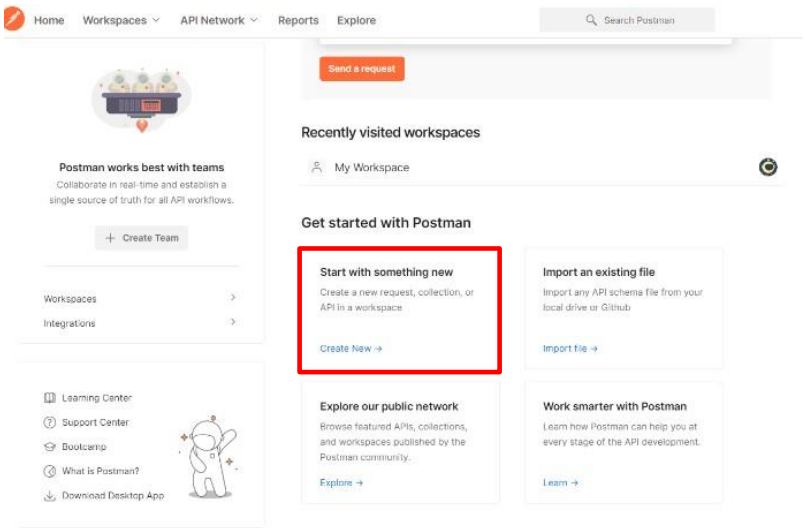
ที่มา : ณปภัช วรรณตรง (2564 : 47)

2. หลังจากทำการดาวน์โหลดและติดตั้งเสร็จสิ้นแล้ว ให้ทำการเปิดโปรแกรม Postman จากนั้นให้ไปที่ icon Postman > คลิกขวาเลือกที่ open Postman



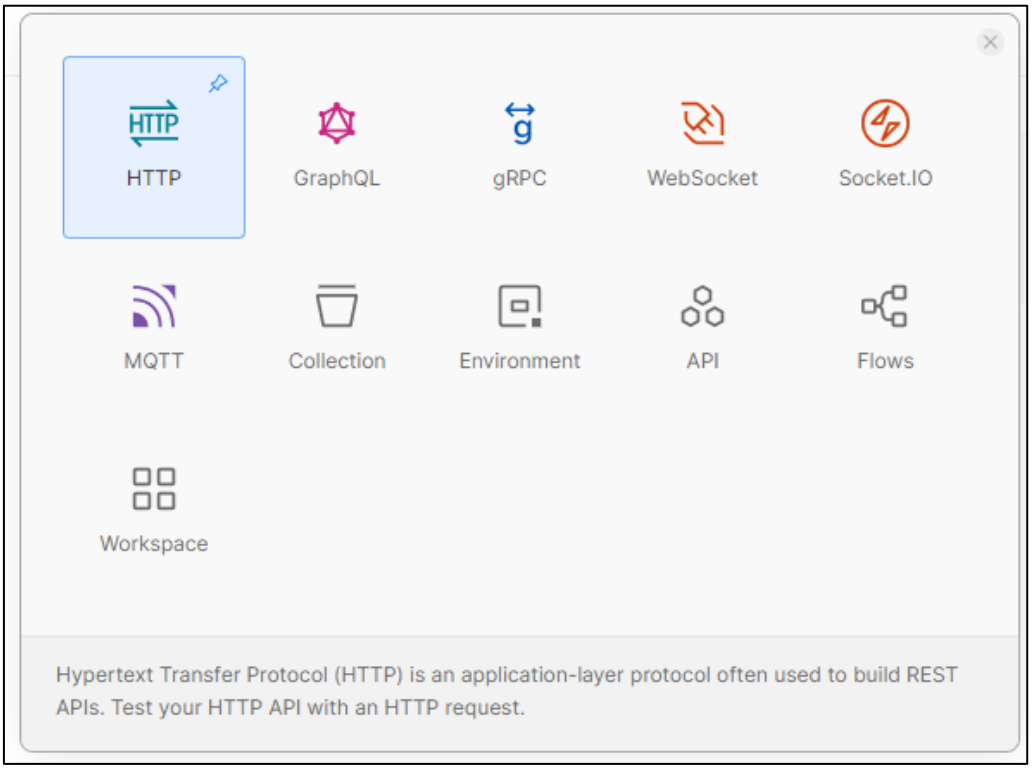
ภาพประกอบ 7.30 เปิดโปรแกรม Postman
ที่มา : ณปภัช วรรณตรง (2564 : 34)

3. เมื่อทำการเปิดโปรแกรม Postman แล้วให้ทำการไปที่ Start with something new > Create New



ภาพประกอบ 7.31 ไปที่ Start with something new > Create New
ที่มา : ณปภัช วรรณตรง (2564 : 48)

4. จากนั้นให้ทำการเลือกเป็น HTTP Request



ภาพประกอบ 7.32 เลือกประเภทงาน

ที่มา : ณปภัช วรรณตรง (2564 : 49)

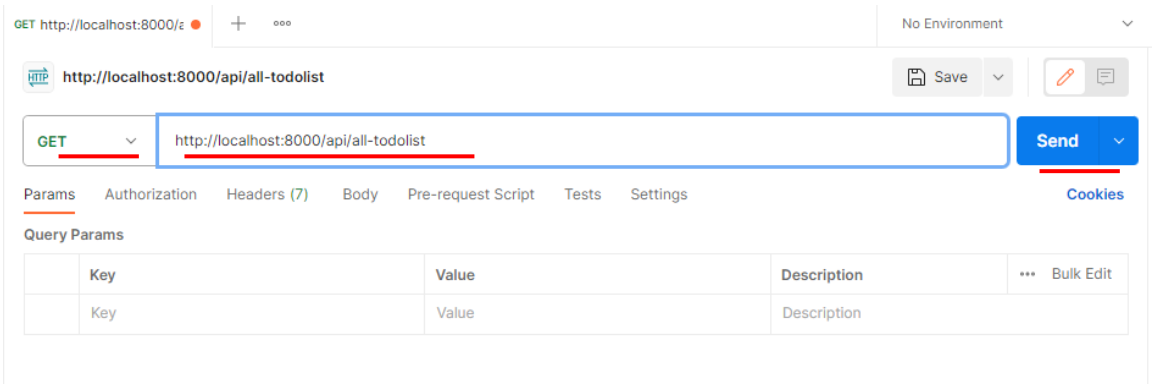
5. ให้ทำการตรวจสอบว่า Server ได้ทำการ Run Server อยู่หรือไม่ หากไม่ได้ Run Server ไว้ให้ทำการสั่งการทำงานServer

```
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 : ณปภัช วรรณตรง (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 : ณปภัช วรรณตรง (2564 : 51)

7. จากนั้นจะได้ผลลัพธ์การดึงข้อมูลจาก Server ตามภาพ

ภาพประกอบ 7.35 ผลลัพธ์การดึงข้อมูล
ที่มา : ณปภัช วรรณตรง (2564 : 52)

8. เปิดไฟล์ views.py จากนั้นให้ทำการแก้ไขโค้ดตามภาพ เพื่อให้ฝั่ง Flutter สามารถส่งข้อมูลกลับมายัง Server ได้ และมีฟังก์ชันสำหรับรับข้อมูลเก็บลงฐานข้อมูลในฝั่ง Server นอกจากนี้ยังมีการตรวจสอบกรณีข้อมูลที่ส่งมาไม่ถูกต้อง หรือข้อมูลที่ส่งมาถูกต้องแล้วซึ่งได้บันทึกลงฐานข้อมูลแล้วจะมีการแจ้งสเตตัสกลับไปยังฝั่ง Flutter

```python
# POST Data (save data to database)
@api_view(['POST'])
def post_todolist(request):
    if request.method == 'POST':
        serializer = TodolistSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data,status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_404_NOT_FOUND)

```

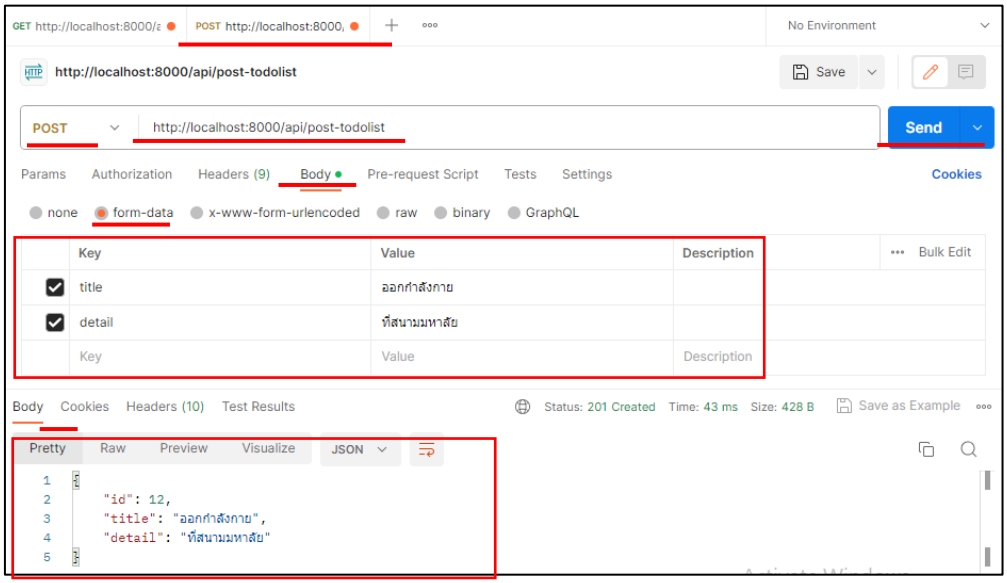
ภาพประกอบ 7.36 เปิดไฟล์ views.py และแก้ไขโค้ด
ที่มา : ณปภัช วรรณตรง (2564 : 53)

9. เปิดไฟล์ urls.py จากนั้นให้ทำการเพิ่มโค้ดตามภาพ

```python
from django.urls import path
from .views import *
# * ดึงมาทุกฟังชั่นใน views.py

urlpatterns = [
    path('', Home),
    path('api/all-todolist/',all_todolist), # localhost:8000/api/all-todolist
    path('api/post-todolist', post_todolist),
]
```

ภาพประกอบ 7.37 เปิดไฟล์ urls.py และแก้ไขโค้ด
ที่มา : ณปภัช วรรณตรง (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 วรรณตรง (2564 : 55)

11. เมื่อทำการกรอกข้อมูลและกดส่งเรียบร้อยแล้ว ให้กลับไปที่ Django Rest Framework > กด Refresh จะพบว่ามีข้อมูลที่ได้ทำการเพิ่มเข้ามาใหม่ตามภาพ

ภาพประกอบ 7.39 มีข้อมูลที่เพิ่มเข้ามาใหม่

ที่มา : ณปภัช วรรณตรง (2564 : 56)

บทสรุป

สำหรับเนื้อหาที่กลาวมาทั้งหมดในบทนี้ 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 ซึ่งเป็นคือหนึ่งในบริการของ Firebase ช่วยจัดการเกี่ยวกับฐานข้อมูล การแสดงผลข้อมูลจาก Firestore Database การใช้งาน Firebase Authenication การใช้งาน image_picker ร่วมกับ Storage และประยุกต์ใช้ความรู้ด้วยการสร้างแอปพลิเคชัน Chat ด้วย Flutter และ Firebase ดังรายละเอียดต่อไปนี้

Firebase

1. Firebase คืออะไร

จีราวุธ วารินทร์ (2563 : 1) ได้อธิบายว่า Firebase 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 Firebase เป็นบริการประเภทหนึ่งที่ทำงานอยู่ฝั่งหลังบ้าน (Backend) รองรับการพัฒนาทั้ง Mobile Apps และ Web Apps ภายในประกอบไปด้วยบริการต่าง ๆ มากมาย แต่จะขอให้บริการหลักเพียงตัวเดียวที่เรียกว่า Firebase เป็นระบบฐานข้อมูลที่เรียกว่า Realtime Database ส่งผลให้เป็นการแบ่งการทำงานออกเป็น 2 ฝั่ง คือ

- Frontend (หน้าบ้าน) คือ โปรเจกต์ Android Apps ที่สร้างขึ้นมา
- Backend (หลังบ้าน) คือ การขอใช้บริการ Firebase เข้ามาทำหน้าที่จัดเก็บข้อมูล กล่าวได้อีกนัยหนึ่งว่า พื้นที่ส่วนนี้คือ APIs ที่ทำหน้าที่คอยให้บริการ Android Apps

กล่าวโดยสรุป Firebase คือ ชุดเครื่องมือและบริการที่ครอบคลุมการทำทั้งในส่วนของ 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 ใช้สำหรับจัดเก็บข้อมูลและอัพเดตข้อมูลแบบเรียลไทม์ ผนวกกับการค้นหาที่มีประสิทธิภาพ และสามารถปรับขนาดอัตโนมัติตามปริมาณข้อมูลที่ใช้งานที่เพิ่มขึ้น (Horizontal Scaling)

3. โครงสร้างข้อมูลของ Firestore

Cloud Firestore หรือเรียกสั้น ๆ ว่า Firestore เป็นบริการฐานข้อมูลในแบบ NOSQL ซึ่งพัฒนาโดย Google เป็นฐานข้อมูลในแบบเรียลไทม์ (Real Time) เมื่อข้อมูลในฐานข้อมูลมีการเปลี่ยนแปลงข้อมูลล่าสุด สามารถส่งไปยังไคลเอนต์เพื่ออัพเดตข้อมูลให้ใหม่โดยอัตโนมัติ 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 ปัจจุบันได้ถูกนำมาใช้ในแอปพลิชันต่าง 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 แสดงให้เห็นดังตารางต่อไปนี้ (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	เก็บในรูปของตารางหลาย ๆ ตารางที่มีความสัมพันธ์กัน	เก็บในรูปของ Key-value ซึ่งเป็นชุดของข้อมูลที่ประกอบด้วยคีย์ (Key) หรือชื่อที่ใช้ในการอ้างอิงถึงข้อมูลที่มีค่าไม่ซ้ำกัน และค่าของคีย์ (Value)
ภาษาที่ใช้จัดการฐานข้อมูล	SQL (Structured Query Language)	ไม่ใช้ภาษา SQL
การติดตั้งระบบบริหารจัดการฐานข้อมูล	ใช้เครื่องเซิร์ฟเวอร์ที่นำมาใช้ ในการจัดเก็บข้อมูลที่มีราคาค่อนข้างแพง	ไม่จำเป็นต้องใช้เครื่องเซิร์ฟเวอร์ที่นำมาใช้ในการจัดเก็บข้อมูลที่มีราคาแพง
การขยายฐานข้อมูล เช่น การเพิ่มเครื่องเซิร์เวอร์และการทำ Cluster	มีความยุ่งยาก	ทำได้ง่าย

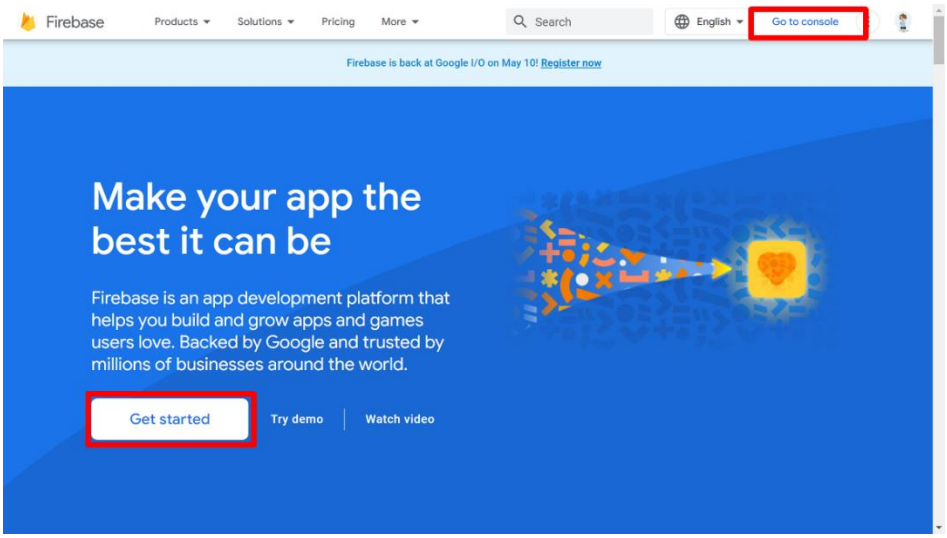
ตาราง 8.1 ความแตกต่างระหว่างฐานข้อมูลเชิงสัมพันธ์และฐานข้อมูล NoSQL (ต่อ)

ด้าน	ฐานข้อมูลเชิงสัมพันธ์	ฐานข้อมูล NoSQL
การเปลี่ยนแปลงใด ๆ กับโครงสร้างข้อมูล	มีผลกระทบกับระบบงานทั้งระบบ	ไม่มีผลกระทบกับระบบงานทั้งระบบ
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



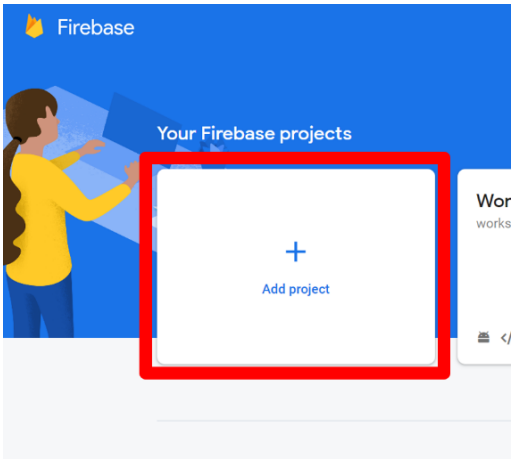
ภาพประกอบ 8.1 การเริ่มต้นใช้งาน Firebase

ที่มา : Firebase. (2020 : 1)

3. เริ่มต้นการสร้างโปรเจกต์ในการใช้งาน Firebase ดังภาพประกอบ 8.3

3.1 กดที่ Add project หลังจากนั้นตั้งชื่อโปรเจกต์ของ Firebase

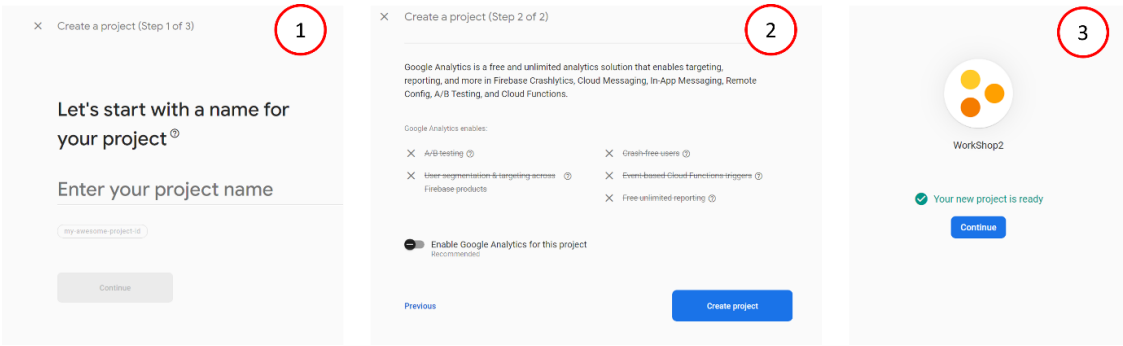
3.2 สามารถเลือกเปิดหรือปิดการใช้งาน Google Analytics ของโปรเจกต์ Firebase



ภาพประกอบ 8.2 การเริ่มสร้างโปรเจกต์ Firebase

ที่มา : Firebase. (2020 : 1)

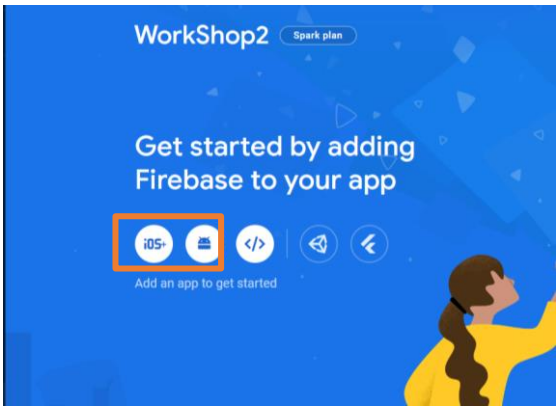
3.3 เมื่อตั้งชื่อ และตั้งค่าการใช้งาน Google Analytics เรียบร้อยแล้วรอเวลาสร้างโปรเจกต์ หากสร้างสำเร็จจะขึ้นดังภาพประกอบ 8.4



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

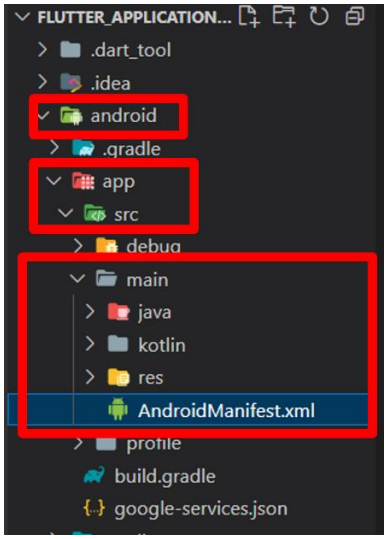


ภาพประกอบ 8.4 การสร้างโปรเจกต์ Firebase

ที่มา : Firebase. (2020 : 1)

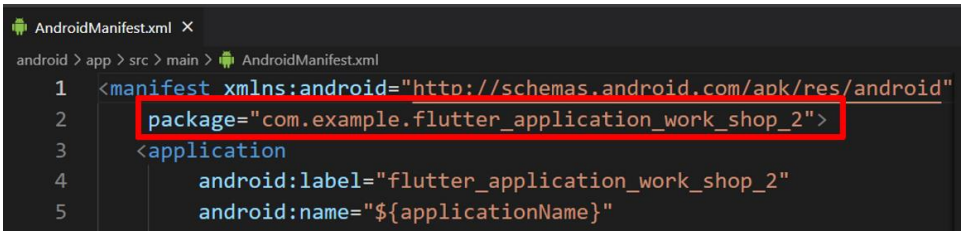
โดยจะแสดงขั้นตอนการเพิ่มแอปพลิเคชันในฝั่ง Android การเพิ่มสามารถเลือกคลิกที่ไอคอนของแต่ระบบปฏิบัติการที่ต้องการใช้งานดังขั้นตอนต่อไปนี้

3.4.1 การหา Package Name จำเป็นต้องสร้างโปรเจกต์ Flutter หรือโปรเจกต์ Flutter เดิมที่เคยสร้างไว้ เพื่อนำไปลงทะเบียนไว้บน Firebase โดยจะหา Package Pame ได้ที่ไฟล์ Android/app/src/main/AndroidManifest.xml ของโปรเจกต์ Flutter ดังภาพประกอบ 8.6



ภาพประกอบ 8.5 Android->app->src->main->AndroidManifest.xml

ที่มา : ณปภัช วรรณตรง (2564 : ...)



ภาพประกอบ 8.6 การหา package name ในโปรเจกต์ Flutter

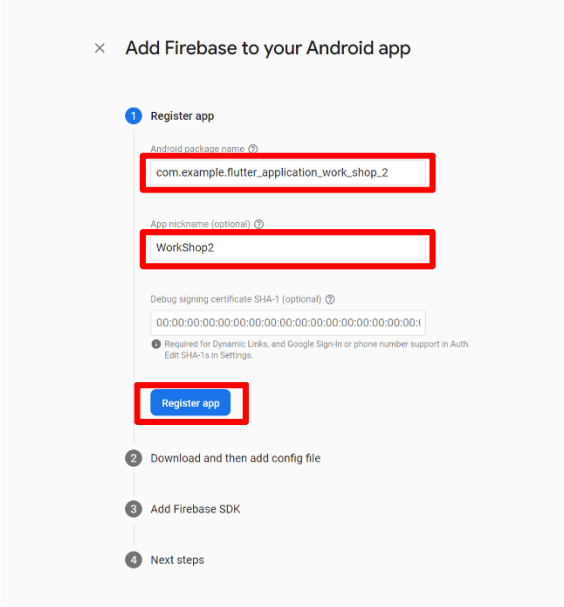
ที่มา : ณปภัช วรรณตรง (2564 : ...)

เมื่อได้ package name ของโปรเจกต์ให้นำไปใส่ที่ช่อง andorid package name ดังภาพประกอบ 8.7

3.4.1.1 ใส่ package name ที่ได้จากโปรเจกต์ Flutter

3.4.1.2 ตั้งชื่อ

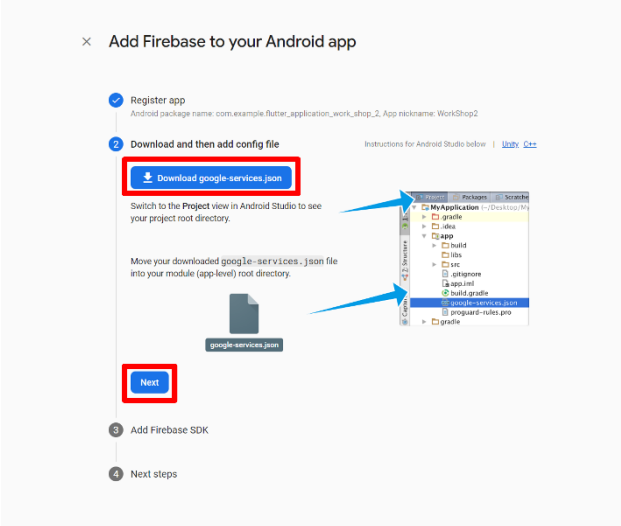
3.4.1.3 กด Register App



ภาพประกอบ 8.7 การเพิ่มแอปพลิเคชันฝั่ง Android

ที่มา : Firebase. (2020 : 1)

3.4.2 เมื่อลงทะเบียนแอปพลิเคชันเสร็จ ให้ดาวน์โหลดไฟล์ google-services.json เอาไว้ และกดถัดไป ดังภาพประกอบ 8.9

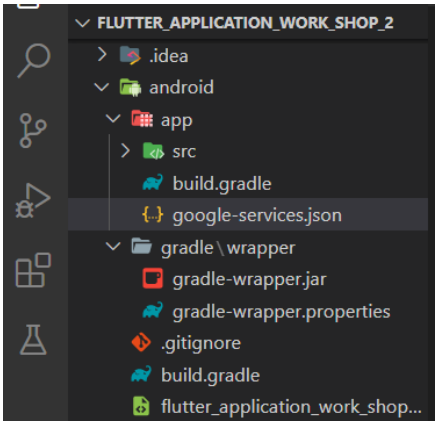


ภาพประกอบ 8.8 การดาวน์โหลดไฟล์ google-services.json

ที่มา : Firebase. (2020 : 1)

3.4.2.1 กดดาวน์โหลด google-service.json

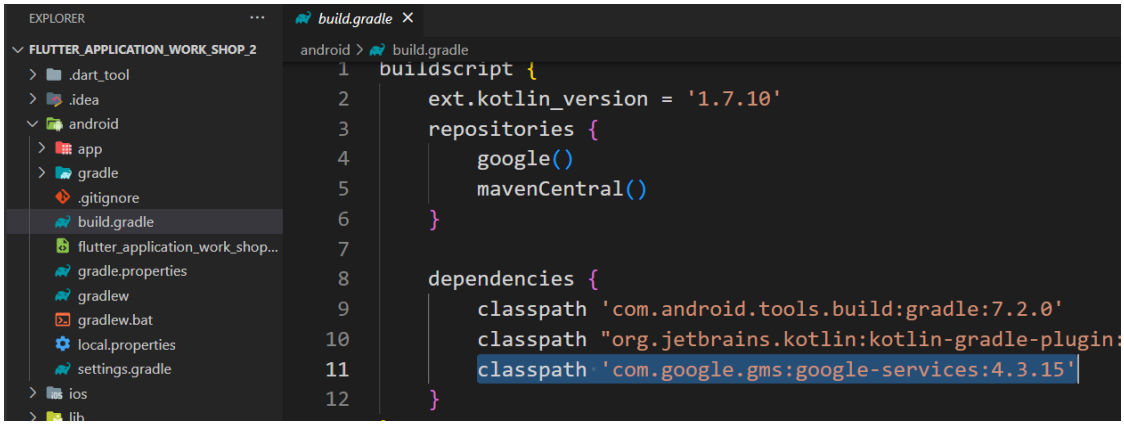
3.4.2.2 กดถัดไป และลากไฟล์ google-services.json ที่ดาวน์โหลด ไปวางไว้ในโปรเจกต์ Flutter ในไฟล์ Android/app



ภาพประกอบ 8.9 การวางไฟล์ google-services.json ใน Android/app

ที่มา : ณปภัช วรรณตรง (2564 : ...)

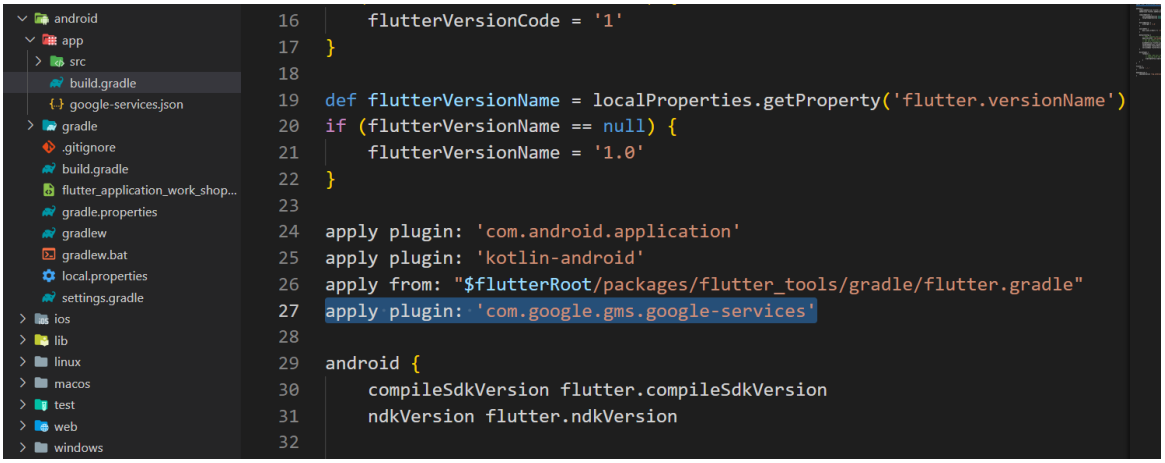
3.4.3 เมื่อวางไฟล์ google-services.json ใน Android/app เสร็จเรียบร้อยให้ไปที่ไฟล์ android\build.gradle หาแท็ก dependencies และวาง classpath 'com.google.gms:google-services:4.3.15' ลงไป ดังภาพประกอบ 8.11



ภาพประกอบ 8.10 การวาง classpath 'com.google.gms:google-services: 4.3.15'

ที่มา : ณปภัช วรรณตรง (2564 : ...)

จากนั้นต้อง apply plugin โดยไปที่ android\app\build.gradle เพิ่ม apply plugin: ‘com.google.gms.google-services’ ดังภาพ



ภาพประกอบ 8.11 การ apply plugin

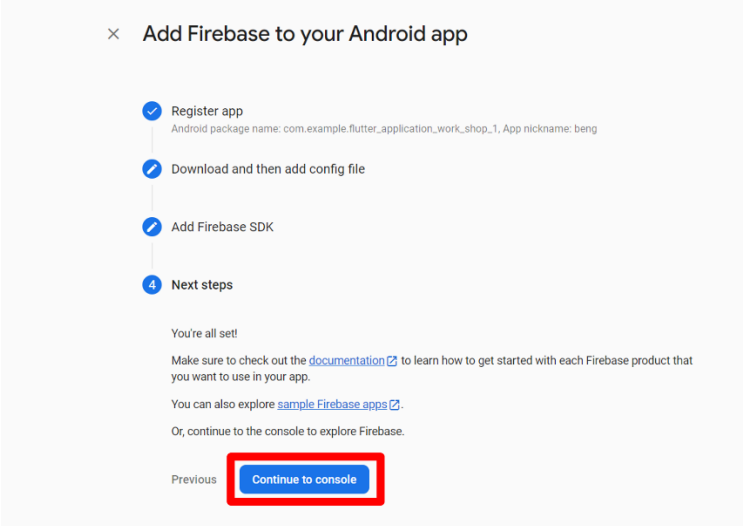
ที่มา : ณปภัช วรรณตรง (2564 : ...)

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  }
```

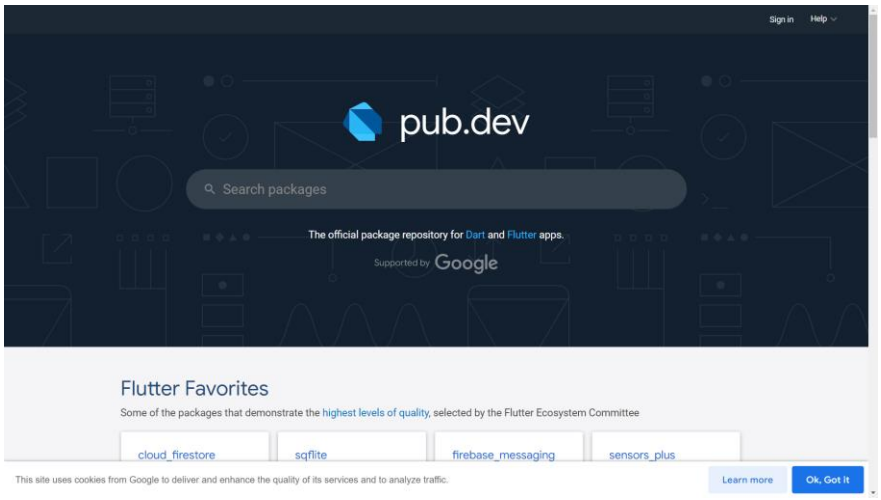
ภาพประกอบ 8.12 การวาง implementation platform
ที่มา : ณปภัช วรรณตรง (2564 : ...)

เมื่อกรอกข้อมูลครบแล้ว ให้กด Continue to console การเพิ่ม Firebase ในแอป Android ก็เสร็จเรียบร้อยแล้ว ดังภาพ



ภาพประกอบ 8.13 การเพิ่ม Firebase ในแอป Android เมื่อเพิ่มข้อมูลครบถ้วน
ที่มา : ณปภัช วรรณตรง (2564 : ...)

3.5 การใช้งาน Firebase มีบริการหลายอย่างมาก ๆ ซึ่งการใช้งานในบริการต่าง ๆ ของ Firebase จำเป็นต้องติดตั้ง Library ในการใช้งาน โดยหา Library ได้จาก https://pub.dev/ ดังภาพ

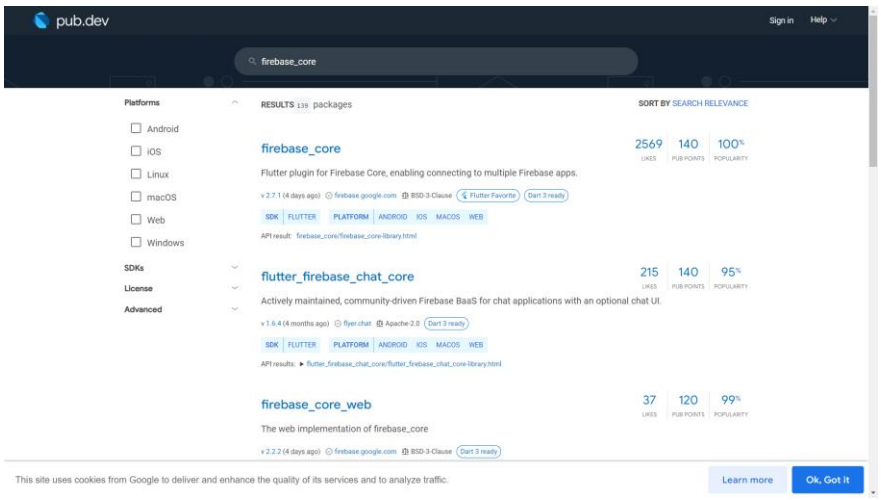


ภาพประกอบ 8.14 เว็ปไซต์ pub.dev
ที่มา : Pub.Dev., (2020 : 1)

การติดตั้ง Library ในโปรเจกต์ Flutter

เนื้อหาส่วนนี้จะเป็นการติดตั้ง Library ที่จะทำให้ Flutter และ Firebase สามารถทำงานร่วมกันได้ ดังมีรายละเอียดขั้นตอนดังนี้

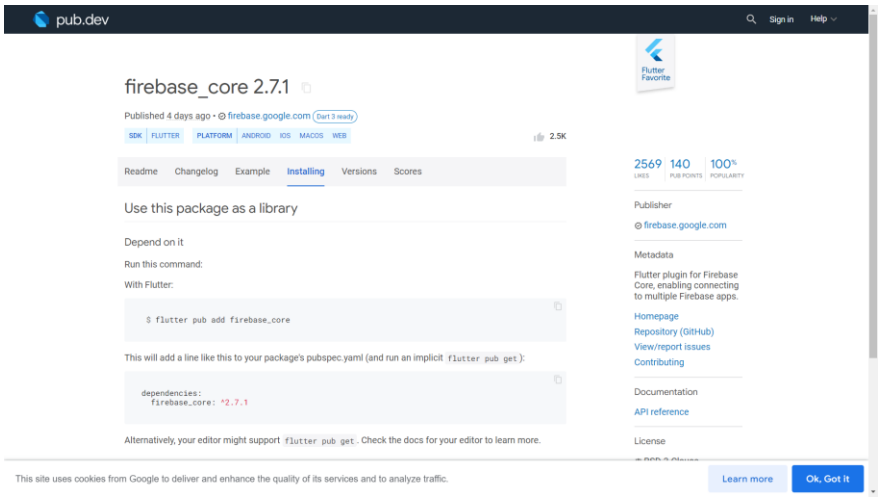
1. กรอกชื่อ Library ที่ต้องการใช้งานแล้วกดค้นหา ผลการค้นหา ซึ่งจะมี Library ให้เลือกใช้งานมากมาย ดังภาพ



ภาพประกอบ 8.15 ผลการค้นหา Library

ที่มา : Pub.Dev., (2020 : 1)

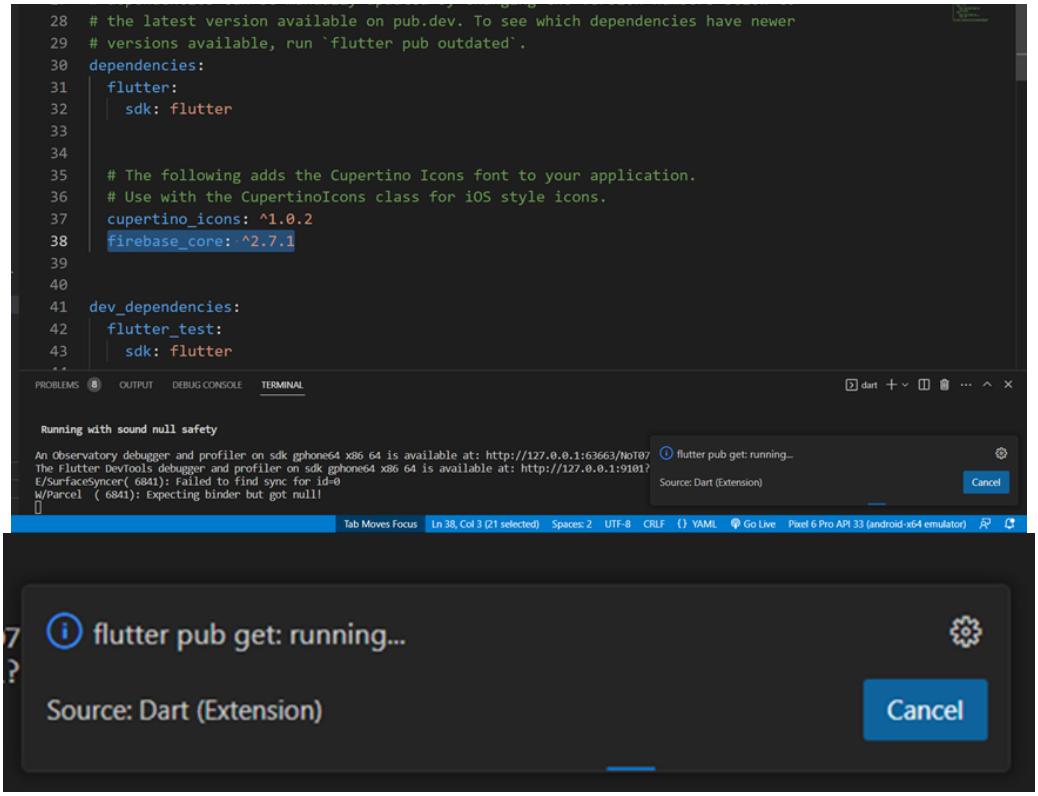
2. จากนั้นให้ทำการคลิกที่ Library ที่ต้องการ ไปที่แท็บเมนู Installing แล้วเลื่อนหา firebase_core: ^2.7.1 แล้วทำการ Copy ไว้ ดังภาพ



ภาพประกอบ 8.16 Library ที่ต้องการใช้งาน

ที่มา : Pub.Dev., (2020 : 1)

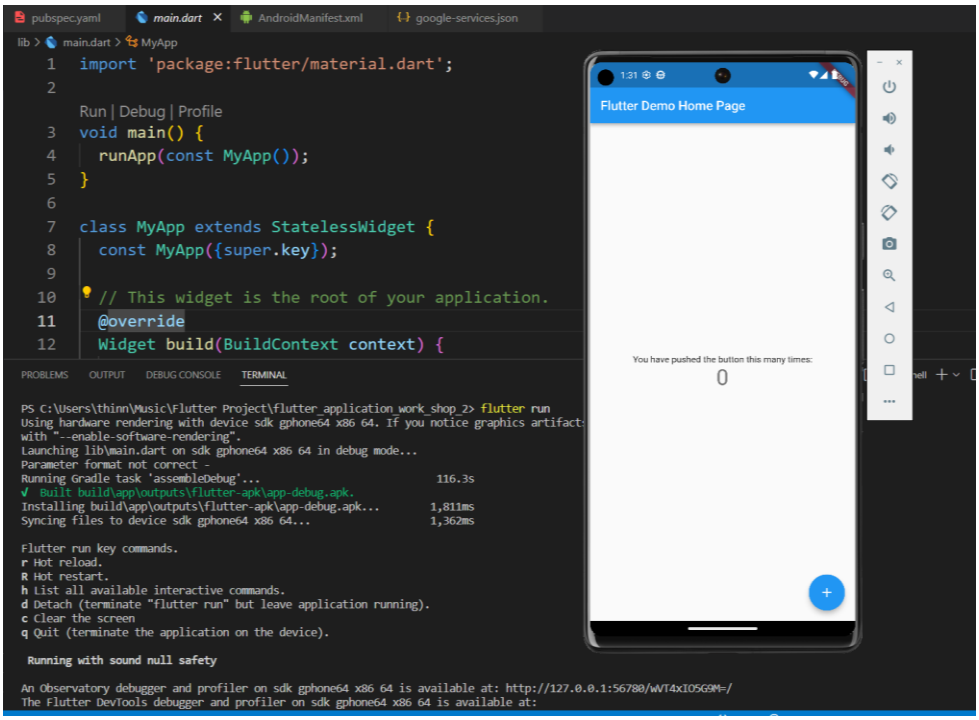
3. ไปที่ไฟล์ pubspec.yaml แล้วนำ firebase_core: ^2.7.1 ไปวางไว้ แล้วทำการกดบันทึก จะขึ้น flutter pub get :running ที่มุมขาวล่างของ VS Code แล้วทำการเปิด Imurater หรือโปรแกรมจำลอง Android ทำการสั่งการทำงานโปรเจกต์เพื่อดูว่า Library ที่ติดตั้งไปสามารถใช้งานกับตัวโปรเจกต์ได้หรือไม่ เมื่อใช้งานได้แล้วการติดตั้ง Library ก็ถือว่าเสร็จสิ้น ดังภาพ



ภาพประกอบ 8.17 การติดตั้ง Library

ที่มา : ณปภัช วรรณตรง (2564 : ...)

หากสั่งการทำงาน และสามารถเปิดหน้าแอปพลิเคชันได้ตามปกติ ไม่ error สามารถเขียน code เรียกใช้งาน Library ได้ ดังภาพ



ภาพประกอบ 8.18 การ run หลังการติดตั้ง Library สำเร็จ

ที่มา : ณปภัช วรรณตรง (2564 : ...)

หากมีการติดตั้ง Library เพิ่มเติมให้สั่งการทำงานแอปพลิเคชันใหม่ทุกครั้ง

การใช้งาน Cloud Firestore

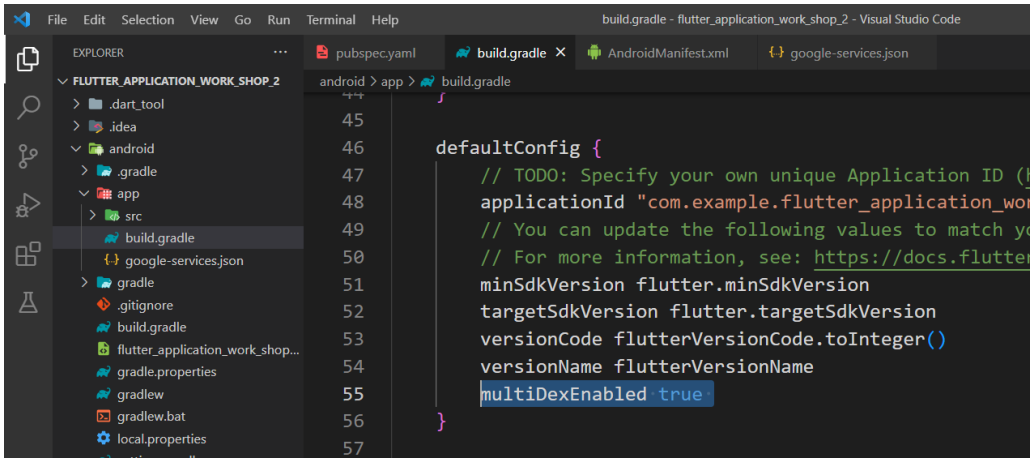
Cloud Firestore หรือเรียกสั้น ๆ ว่า Firestore เป็นบริการฐานข้อมูลแบบใหม่ ใช้สำหรับจัดเก็บข้อมูลและอัพเดตข้อมูลแบบเรียลไทม์ ผนวกกับการค้นหาที่มีประสิทธิภาพ และสามารถปรับขนาดอัตโนมัติตามปริมาณข้อมูลที่ใช้งานที่เพิ่มขึ้น (Horizontal Scaling) (จีราวุธ วารินทร์, 2563 : 1) ในเนื้อหาส่วนนี้จะมาเรียนรู้การใช้งาน Cloud Firestore เพื่อสร้างฐานข้อมูลสำหรับเก็บข้อมูล มีรายละเอียดขั้นตอนดังนี้

1. เริ่มต้นด้วยการติดตั้ง Library ดังต่อไปนี้

    1.1 firebase_core

    1.2 cloud firestore

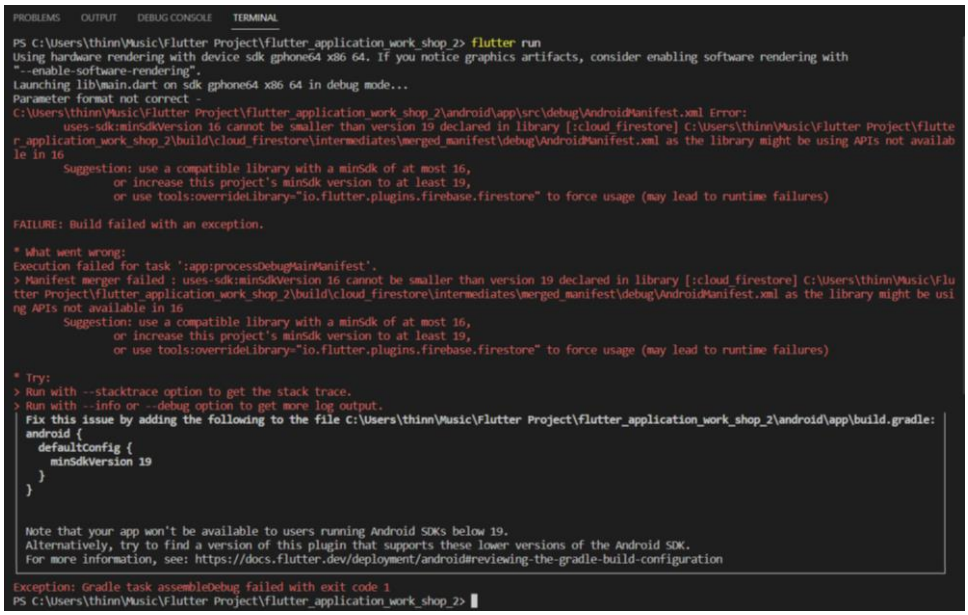
    การใช้งาน cloud firestore ถ้าเจอ ERROR: Cannot fit requested classed in a single dex file แก้ด้วยการเพิ่ม multiDexEnabled true ใน android -> app -> build.gradle ในแท็ก defaultConfig



ภาพประกอบ 8.19 การเพิ่ม multiDexEnabled true

ที่มา : ณปภัช วรรณตรง (2564 : ...)

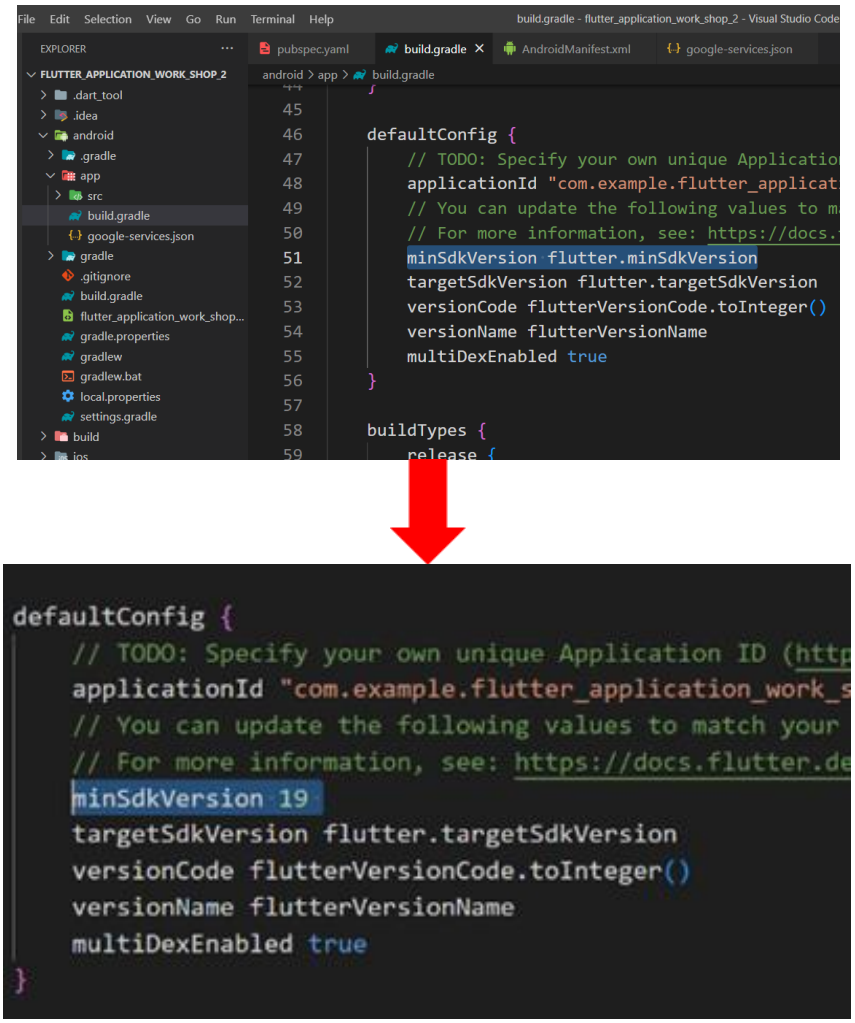
ถ้าเจอ Error:uses-sdk:minSdkVersion 16 cannot be smaller than version 19 declared in Library [:cloud_firestore] ดังภาพ



ภาพประกอบ 8.20 Error:uses-sdk:minSdkVersion

ที่มา : ณปภัช วรรณตรง (2564 : ...)

ให้ทำการไปแก้ไข minSdkVersion flutter.minSdkVersion ที่แท็ก defaultConfig ให้เป็น minSdkVersion 19 ดังภาพ



ภาพประกอบ 8.21 การเพิ่ม minSdkVersion

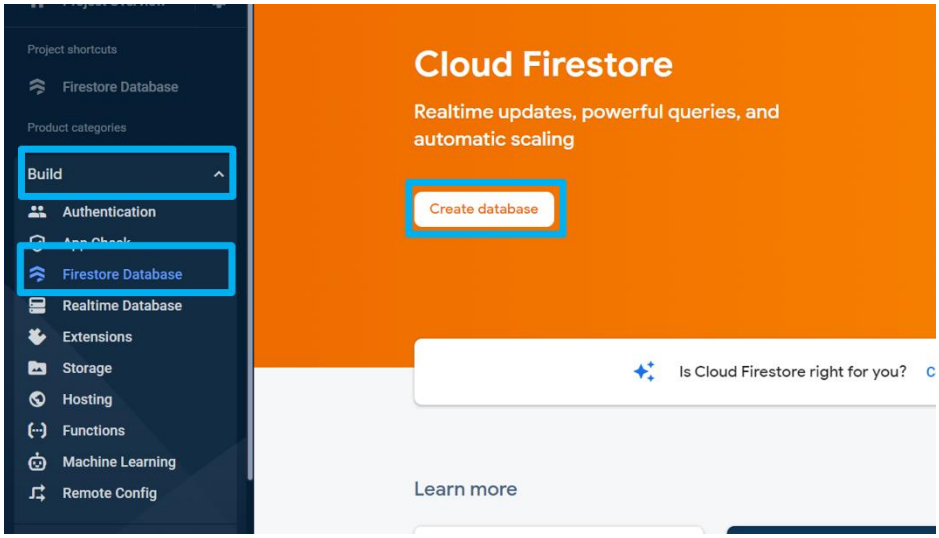
ที่มา : ณปภัช วรรณตรง (2564 : ...)

2. เมื่อติดตั้ง Library สำเร็จแล้ว ให้ไปที่ Firebase โปรเจกต์ เพื่อทำการสร้างฐานข้อมูล Firestore Database ดังภาพ

2.1 ไปที่ build

2.2 เลือก Firestore Database

2.3 กด Create database



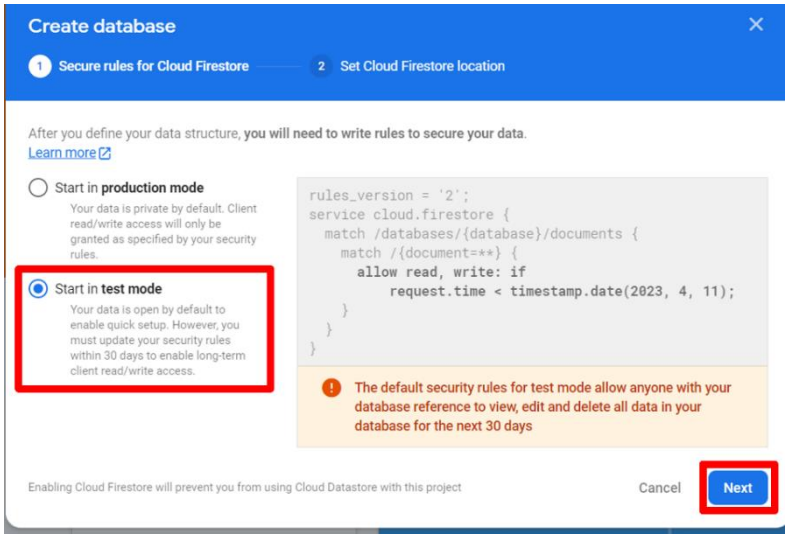
ภาพประกอบ 8.22 การเริ่มต้นสร้าง Firestore Database

ที่มา : Firebase (2020 : 1)

3. หลังจากที่ทำการสร้าง Firestore Database จะแสดงดังภาพประกอบ 8.24 - 8.25 ให้ทำตามขั้นตอนดังต่อไปนี้

3.1 เลือก test โหมด

3.2 กด Next ดังภาพ

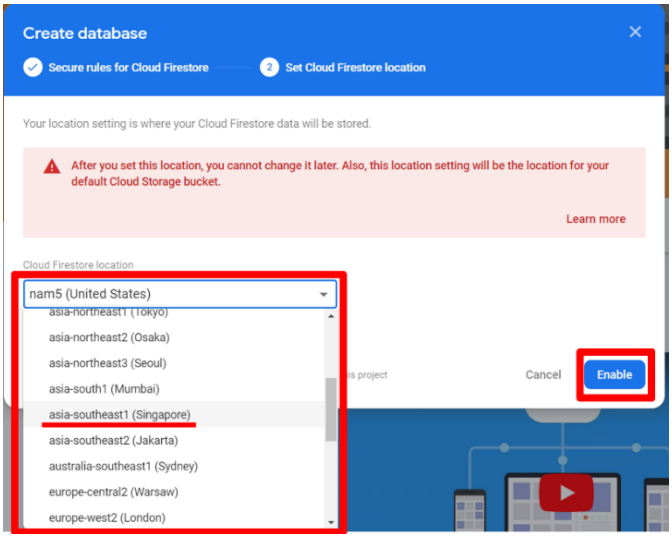


ภาพประกอบ 8.23 การตั้งค่าเริ่มต้นให้กับ Firestore Database

ที่มา : Firebase (2020 : 1)

3.3 เลือกที่ตั้งในการเก็บข้อมูล ของ Firestore Database ของโปรเจกต์

3.4 กด Enable ดังภาพ



ภาพประกอบ 8.24 การเลือกที่ตั้งในการเก็บข้อมูล Firestore Database

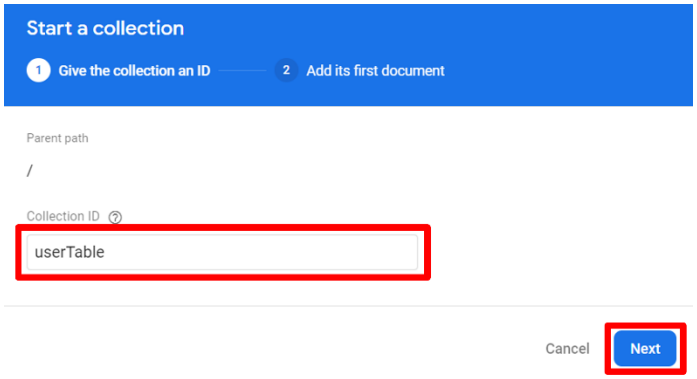
ที่มา : Firebase (2020 : 1)

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)

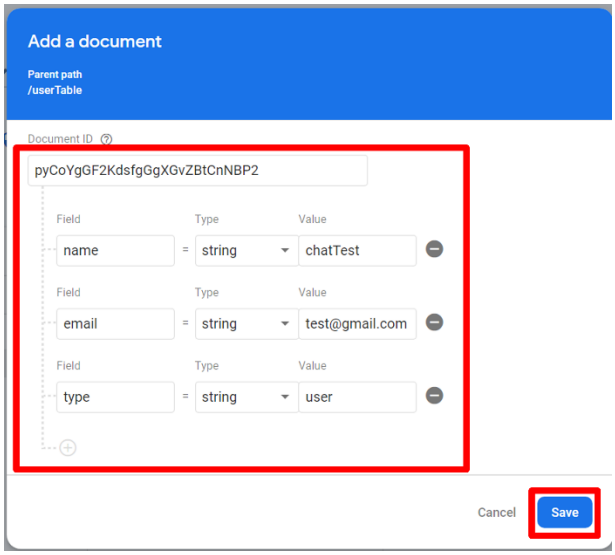
3.6 ทำการเพิ่มข้อมูลลงไปใน document

3.6.1 ใส่ชื่อ Field ลงไป

3.6.2 ระบุประเภทข้อมูลที่ต้องการเก็บ

3.6.3 ระบุข้อมูลที่ต้องการเก็บ

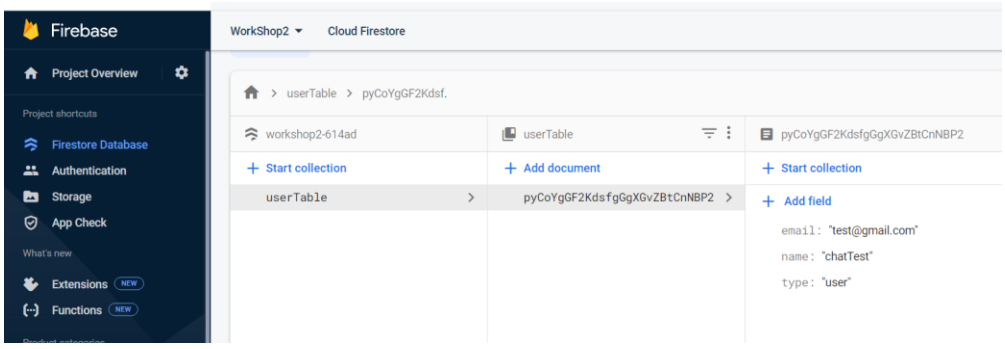
3.6.4 กด Save ดังภาพ



ภาพประกอบ 8.26 การเพิ่มข้อมูลลงไปใน collection

ที่มา : Firebase (2020 : 1)

3.7 เมื่อกรอก Field, Type และ Value เสร็จแล้วจะได้ดังภาพ จากนั้นกดถัดไป ซึ่งจะได้ข้อมูลดังภาพ



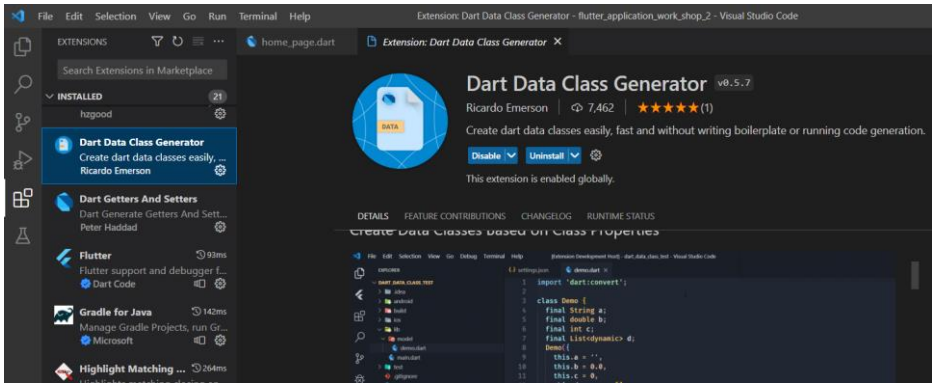
ภาพประกอบ 8.27 ผลที่เพิ่มลงไปใน collection

ที่มา : Firebase (2020 : 1)

การแสดงผลข้อมูลจาก Firestore Database

เนื้อหาส่วนนี้จะเป็นการสร้าง Model ในการอ่านค่าข้อมูลที่ได้จาก Firestore Database มาเก็บไว้ใน Model ที่สร้างไว้ ดังมีรายละเอียดขั้นตอนดังนี้

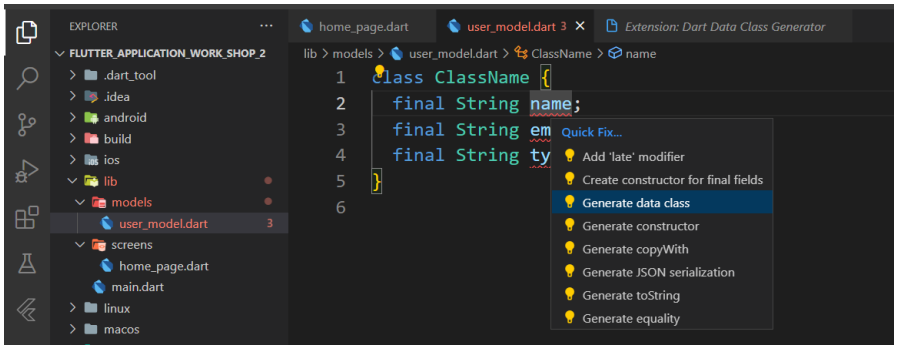
1. สิ่งแรกที่ต้องมีในการสร้าง Model คือ Extension Dart Data Class Generator ด้วยการค้นหาใน Vs Code และติดตั้ง



ภาพประกอบ 8.28 Extension Dart Data Class Generator

ที่มา : ณปภัช วรรณตรง (2564 : ...)

2. เมื่อลง Extension เรียบร้อยแล้ว ทำการปิดและเปิด Vs Code ใหม่ ทำการสร้างโฟลเดอร์ Models ของในโฟลเดอร์ lib เสร็จเรียบร้อยให้ทำการสร้างไฟล์ข้างในโฟลเดอร์ Models ชื่อไฟล์ user_Model.dart ทำการสร้าง Class ประกาศตัวแปรที่การอ่านข้อมูลมาเก็บไว้ แล้วทำการกดที่หลอดไฟ หรือนำเมาส์คลิกตัวแปรที่ประกาศที่ error กด Ctrl+จุด แล้วเลือก Generate data class



ภาพประกอบ 8.29 การสร้าง Model

ที่มา : ณปภัช วรรณตรง (2564 : ...)

เมื่อทำการ Generate data class เรียบร้อยแล้วจะแสดงดังภาพ

```dart
import 'dart:convert';

class ClassName {
  final String name;
  final String email;
  final String type;
  ClassName({
    required this.name,
    required this.email,
    required this.type,
  });

  ClassName copyWith({
    String? name,
    String? email,
    String? type,
  }) {
    return ClassName(
      name: name ?? this.name,
      email: email ?? this.email,
      type: type ?? this.type,
    );
  }

  Map<String, dynamic> toMap() {
    final result = <String, dynamic>{};

    result.addAll({'name': name});
    result.addAll({'email': email});
    result.addAll({'type': type});

    return result;
  }
```

ภาพประกอบ 8.30 Model ที่ Generate แล้ว

ที่มา : ณปภัช วรรณตรง (2564 : ...)

3. หลังจากที่สร้าง Model เสร็จเรียบร้อยแล้ว มาเริ่มต้นเขียน code ในการอ่านข้อมูลจาก Firestore Database มาเก็บไว้ใน Model ที่สร้างไว้ มีขั้นตอนการเขียน code ดังนี้

3.1 เริ่มต้นด้วยการประกาศตัวแปรในรูปแบบ List เพราะ Model ที่สร้างไว้ ในการอ่านข้อมูลออกมาจาก Firestore Database มีข้อมูลมากว่า 1 ตำแหน่ง

3.2 สร้าง initState method เป็น method ที่จะทำงานก่อนทุกครั้งที่เข้ามายังหน้าแอปพลิเคชันที่ทำการสร้าง method นี้ไว้

3.3 สร้าง Future Function คือ function app asynchronous จะดำเนินการสั่งการทำงานโปรแกรมทีละชุดคำสั่ง และจะส่งการทำงานชุดคำสั่งถัดไปทันทีโดยไม่จำเป็นต้องรอชุดคำสั่งก่อนหน้าทำงานเสร็จ ตัวอย่างเช่น ถ้าโปรแกรมเรียกฟังก์ชัน A(); และ B(); ตามลำดับ โปรแกรมจะรัน

ฟังก์ชัน A(); และ B(); ตามลำดับโดยไม่สนใจว่าฟังก์ชัน A(); จะทำงานเสร็จหรือยัง จะไปเรียกฟังก์ชัน B(); ต่อเลยทันที

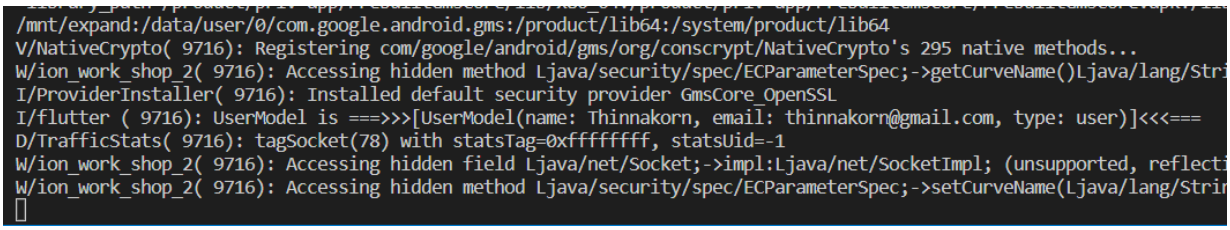
3.4 เขียน Code เรียกใช้งาน Library ของ Firebase Firestore ในการอ่านค่ามาเก็บไว้ใน Model ที่สร้างไว้ ดังภาพ

```dart
class _HomePageState extends State<HomePage> {
  List<UserModel> userModel = [];

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

  Future<Null> readDataUserModel() async {
    Firebase.initializeApp().then((value) async {
      FirebaseFirestore.instance
          .collection('userModel')
          .snapshots()
          .listen((event) {
        for (var element in event.docs) {
          UserModel model = UserModel.fromMap(element.data());
          setState(() {
            userModel.add(model);
          });
          print('UserModel is ===>>>$userModel<<<===');
        }
      });
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: Icon(Icons.home),
        title: Text('Home Page'),
      ),
    );
  }
}
```

ภาพประกอบ 8.31 การอ่านค่าข้อมูลจาก Firestore Database มาเก็บไว้ใน Model
ที่มา : ณปภัช วรรณตรง (2564 : ...)

3.5 นำ Future Function ที่สร้างไปใส่ไว้ใน initState method แล้วทำการสั่งการทำงานโปรเจกต์ Flutter

3.6 เมื่อสั่งการทำงานโปรเจกต์สำเร็จแล้ว ให้ดูที่ console log terminal บน VS Code หาคำว่า UserModel is ===>>>$userModel<<<=== ที่สั่งแสดงผลไว้ใน Future Function จะได้ผลลัพธ์ ดังภาพ



ภาพประกอบ 8.32 การข้อมูลของ userModel ที่ได้บน Terminal ใน VS Code

ที่มา : ณปภัช วรรณตรง (2564 : ...)

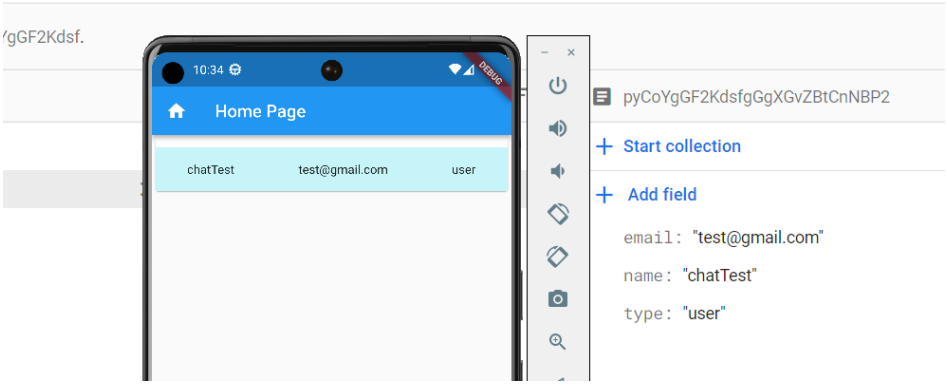
3.7 เมื่อได้ข้อมูลดังภาพประกอบ 8.33 แล้ว ให้เริ่มต้นเขียน Code แสดงข้อมูลที่จาก firestore database ไปแสดงผลบน Imurater หรือตัวจำลอง Android โดยจะเขียนในส่วนของ Body ดังภาพ

```
body: userModel.isEmpty
          ? Text('ไม่พบข้อมูลผู้ใช้งาน')
          : ListView.builder(
              itemCount: userModel.length,
              itemBuilder: (context, index) => Card(
                child: Container(margin: EdgeInsets.only(top: 10),
                  color: Color.fromARGB(255, 198, 244, 248),
                  height: 50,
                  child: Row(
                    mainAxisAlignment: MainAxisAlignment.spaceAround,
                    children: [
                      Text(userModel[index].name),
                      Text(userModel[index].email),
                      Text(userModel[index].type),
                    ],
                  ),
                ),
              ),
            ),
```

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 Authenication

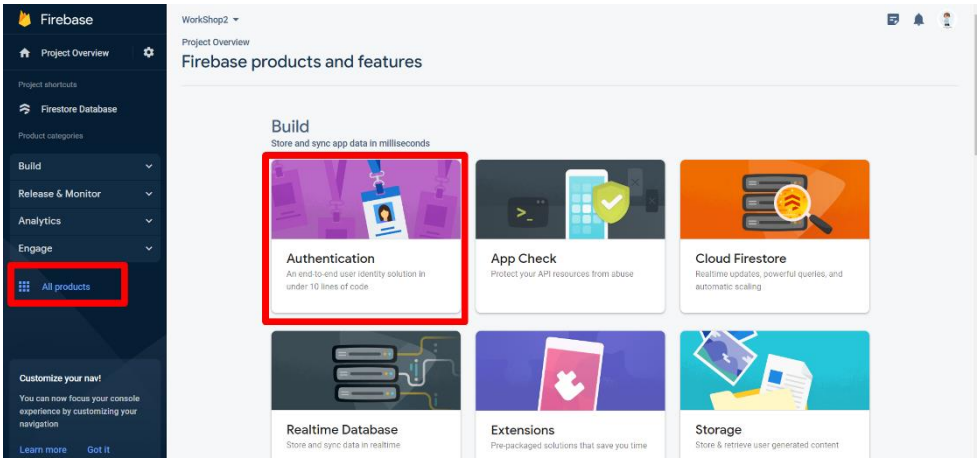
Firebase Authenication คือ การตรวจสอบสิทธิ์ Firebase ให้บริการแบ็กเอนด์ SDK ที่ใช้งานง่าย และไลบรารี UI สำเร็จรูปเพื่อตรวจสอบสิทธิ์ผู้ใช้ในแอปพลิเคชัน รองรับการตรวจสอบสิทธิ์โดยใช้รหัสผ่าน หมายเลขโทรศัพท์ และผู้ให้บริการข้อมูลประจำตัวส่วนกลางยอดนิยม เช่น Google, Facebook และ Twitter และอื่นๆ Firebase (Fiebase, 2566)

เนื้อหาส่วนนี้จะเป็นการติดตั้ง Library สำหรับการตรวจสอบสิทธิ์ผู้ใช้ใน Firebase และการเลือกแพลตฟอร์มที่สามารถล็อกอินผ่าน Firebase โดยในเนื้อหานี้จะใช้รูปแบบการใช้รหัสผ่าน ดังมีรายละเอียดขั้นตอนดังนี้

เริ่มต้นด้วยการติดตั้ง Library ดังต่อไปนี้

1. firebase_auth

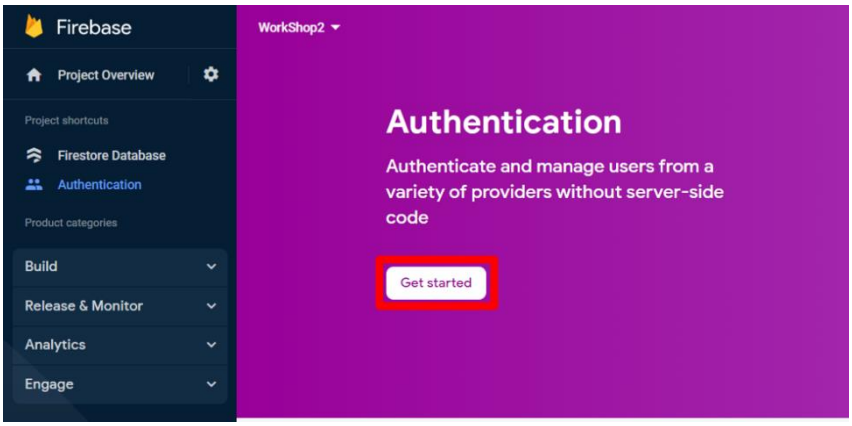
    1.1 ไปที่ All Products เลือกที่ Authentication ดังภาพประกอบ 8.36



ภาพประกอบ 8.35 การเริ่มใช้งาน Firebase Authentication

ที่มา : Firebase (2020 : 1)

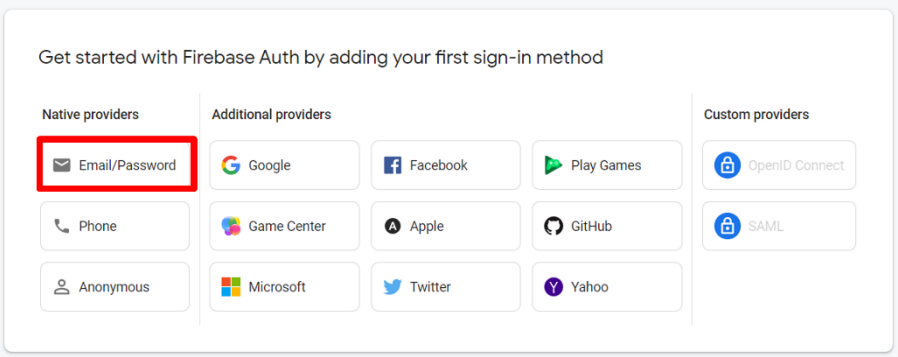
1.2 ทำการกดที่ Get Started ดังภาพ



ภาพประกอบ 8.36 การเปิดการใช้งาน Firebase Authentication

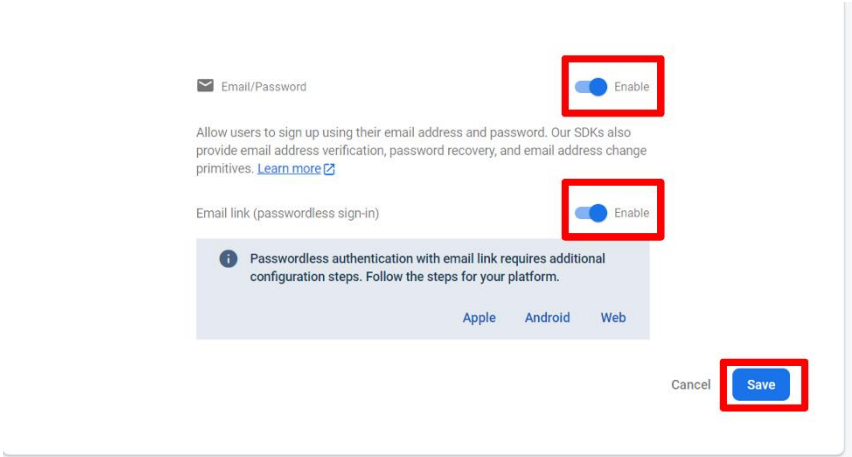
ที่มา : Firebase (2020 : 1)

1.3 เลือกแพลตฟอร์มที่ต้องการใช้งานในการล็อกอิน ดังภาพ



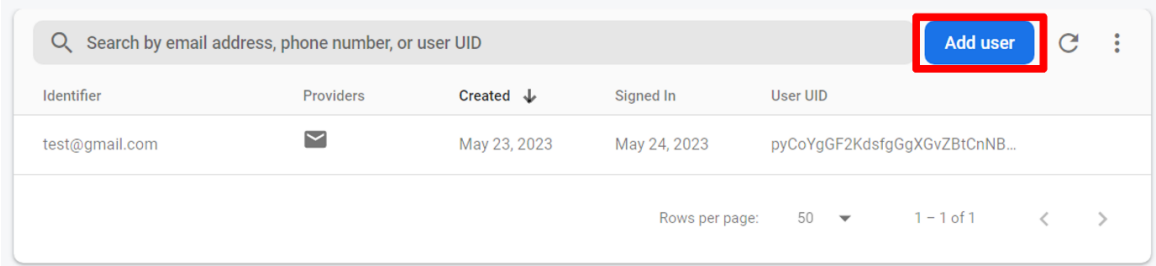
ภาพประกอบ 8.37 แพลตฟอร์มที่สามารถล็อกอินผ่าน Firebase Authentication
ที่มา : Firebase (2020 : 1)

1.4 เลือกการล็อกอินด้วย Email/Password แล้วทำการเปิดการใช้งานและกด Save ให้เรียบร้อย การเปิดการใช้งาน Authentication ก็เป็นอันเสร็จสิ้น ดังภาพ



ภาพประกอบ 8.38 เปิดการใช้งาน Email/Password
ที่มา : Firebase (2020 : 1)

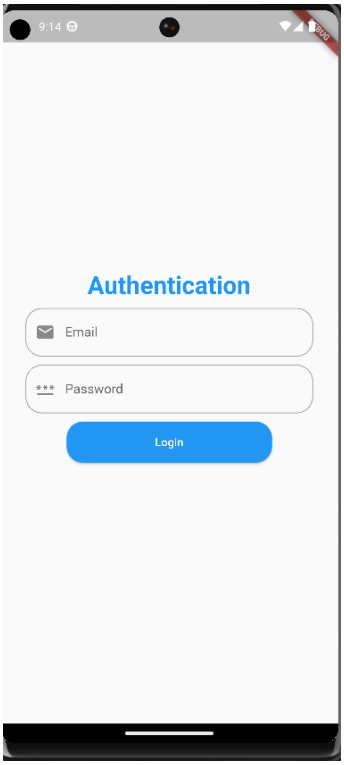
1.5 ทำการเพิ่ม Users โดยกดที่ปุ่ม Add user ใส่ Email/Password ให้ครบถ้วนทำการกดบันทึกให้เรียบร้อย หากเพิ่มข้อมูลสำเร็จจะแสดงดังภาพ



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

```dart
Widget emailTextField() {
    return Container(
      margin: EdgeInsets.only(top: 10),
      width: 350,
      child: TextField(
        decoration: InputDecoration(
          prefixIcon: Icon(Icons.email_rounded),
          label: Text('Email'),
          enabledBorder: OutlineInputBorder(
            borderRadius: BorderRadius.circular(20),
            borderSide: BorderSide(color: Colors.grey),
          ),
          focusedBorder: OutlineInputBorder(
            borderRadius: BorderRadius.circular(20),
            borderSide: BorderSide(color: Colors.blue),
          ),
        ),
      ),
    );
  }
```

ภาพประกอบ 8.41 ตัวอย่างการใช้งาน TextField

ที่มา : ณปภัช วรรณตรง (2564 : ...)

```dart
Widget loginElevatedButton() {
    return Container(
      margin: EdgeInsets.only(top: 10),
      width: 250,
      height: 50,
      child: ElevatedButton(
        style: ElevatedButton.styleFrom(
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(20),
          ),
        ),
        onPressed: () {},
        child: Text('Login'),
      ),
    );
  }
```

ภาพประกอบ 8.42 ตัวอย่างการใช้งาน ElevatedButton

ที่มา : ณปภัช วรรณตรง (2564 : ...)

1.7 เมื่อออกแบบหน้าแอปพลิเคชันเรียบร้อยแล้ว ให้ไปทำการประกาศตัวแปรที่ใช้ในการอ่านค่าจากฐานข้อมูลมาเก็บไว้ในไฟล์ authentication.dart ดังภาพ

```dart
class _AuthenticationState extends State<Authentication> {

  String? email, password, type; // ใช้ในการรับข้อมูลจาก TextField และเช็คประเภทการล็อกอิน
  List<UserModel> userModel = []; // model ในการรับข้อมูลจาก Firestore database

  @override
  Widget build(BuildContext context) {
    return Scaffold();
```

ภาพประกอบ 8.43 ตัวอย่างประกาศตัวแปร
ที่มา : ณปภัช วรรณตรง (2564 : ...)

1.8 นำตัวแปรที่ประกาศในการใช้งานรับค่าจาก TextField ไปเพิ่มในแต่ละ TextField ที่ต้องการรับค่าและนำข้อมูลที่ได้มาใช้งานต่อ ดังภาพ

```dart
Widget emailTextField() {
   return Container(
     margin: EdgeInsets.only(top: 10),
     width: 350,
     child: TextField(
       onChanged: (value) => email = value.trim(),
       decoration: InputDecoration(
         prefixIcon: Icon(Icons.email_rounded),
         label: Text('Email'),
         enabledBorder: OutlineInputBorder(
           borderRadius: BorderRadius.circular(20),
           borderSide: BorderSide(color: Colors.grey),
         ),
         focusedBorder: OutlineInputBorder(
           borderRadius: BorderRadius.circular(20),
           borderSide: BorderSide(color: Colors.blue),
         ),
       ),
     ),
   );
 }
```

ภาพประกอบ 8.44 ตัวอย่างการเพิ่มตัวแปรเพื่อรับค่าข้อมูลจากการกรอกบน TextField
ที่มา : ณปภัช วรรณตรง (2564 : ...)

1.9 เขียนเช็คข้อมูลที่ได้รับจากการกรอกบน TextField กับ Firebase Authentication และ Firestore Database โดยสร้าง Future Function ในการเช็คข้อมูลดังภาพ

```
Future<Null> checkAuthen() async {
   await Firebase.initializeApp().then((value) async {
     await FirebaseAuth.instance
         .signInWithEmailAndPassword(email: email!, password: password!)
         .then((value) async {
       String? uid = value.user!.uid;
       await FirebaseFirestore.instance
           .collection('userTable')
           .doc(uid)
           .snapshots()
           .listen((event) {
         UserModel model = UserModel.fromMap(event.data()!);
         switch (model.type) {
           case 'user':
             Navigator.pushAndRemoveUntil(
                 context,
                 MaterialPageRoute(
                   builder: (context) => HomePage(),
                 ),
                 (route) => false);
             break;
           default:
         }
       });
     });
   });
}
```

ภาพประกอบ 8.45 การเช็คการล็อกอิน Firebase

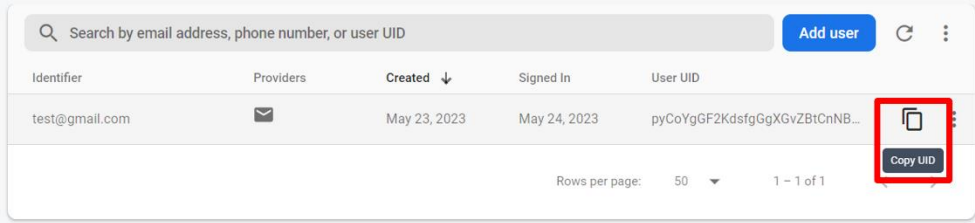
ที่มา : ณปภัช วรรณตรง (2564 : ...)

1.10 นำ Future Function checkAuthen ที่สร้างขึ้นไปใส่ในปุ่ม Login ในส่วนของ onPressed ดังภาพ

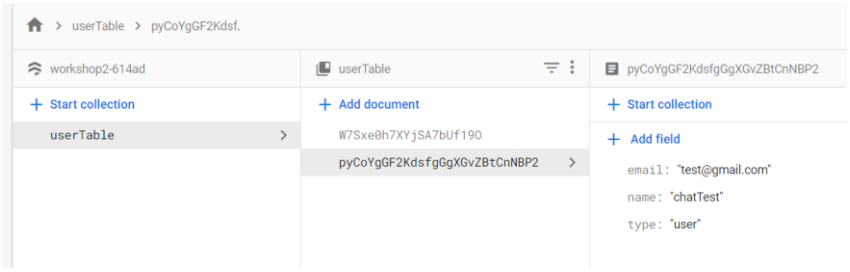
```
1
2  Widget loginElevatedButton() {
3    return Container(
4      margin: EdgeInsets.only(top: 10),
5      width: 250,
6      height: 50,
7      child: ElevatedButton(
8        style: ElevatedButton.styleFrom(
9          shape: RoundedRectangleBorder(
10           borderRadius: BorderRadius.circular(20),
11         ),
12       ),
13       onPressed: () {
14         checkAuthen();
15       },
16       child: Text('Login'),
17     ),
18   );
19 }
```

ภาพประกอบ 8.46 การเพิ่ม Future Function checkAuthen ในปุ่ม Login
ที่มา : ณปภัช วรรณตรง (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

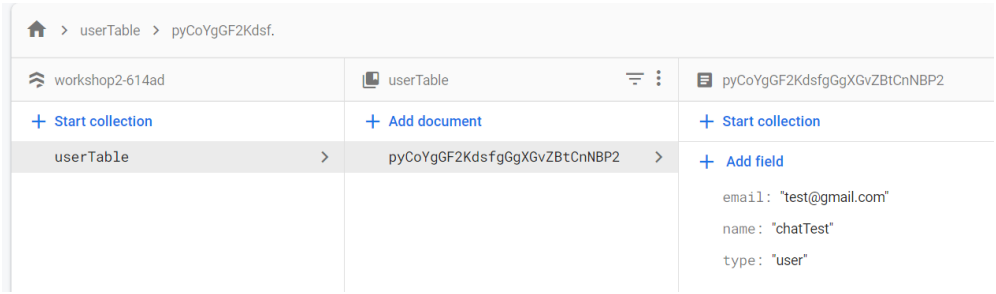


ภาพประกอบ 8.47 การ Copy UID User
ที่มา : Firebase. (2020 : 1)



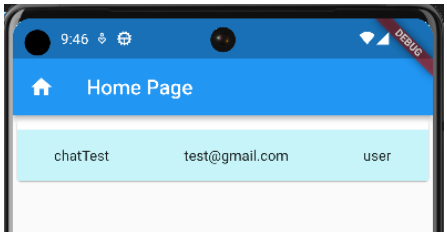
ภาพประกอบ 8.48 ข้อมูลหลังการเพิ่ม Document
ที่มา : Firebase. (2020 : 1)

1.12 เมื่อได้ดังภาพประกอบ 8.49 แล้วให้ทำการลบ Document ตัวเดิมออกเนื่องจาก UID ของ Document ดังกล่าวไม่ตรงกับ UID User ที่ทำการสมัครไว้บน Firebase Authentication ลบข้อมูลเรียบร้อยแล้วจะได้ดังภาพ



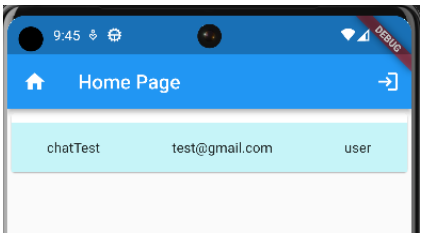
ภาพประกอบ 8.49 ข้อมูลหลังการลบ Document
ที่มา : Firebase. (2020 : 1)

1.13 สั่งการทำงานแอปพลิเคชัน ทำการล็อกอินผ่านหน้าแอปพลิเคชัน โดยใส่ E-mail และ Password ที่ได้ทำการสมัครไว้บน Firebase Authentication หาก E-mail Password และ Type ถูกต้องจะล็อกอินไปยังหน้า Home Page ดังภาพ



ภาพประกอบ 8.50 หน้า Home Page
ที่มา : ณปภัช วรรณตรง (2564 : ...)

1.14 เมื่อสามารถล็อกอินเข้ามาใช้งานแอปพลิเคชันได้แล้ว ให้ทำการสร้างปุ่มสำหรับใช้ในการล็อกเอาท์ไว้บน AppBar และทำการเขียน code ในการล็อกเอาท์ ดังภาพ



ภาพประกอบ 8.51 ปุ่มสำหรับการล็อกเอาท์
ที่มา : ณปภัช วรรณตรง (2564 : ...)

```
appBar: AppBar(
        leading: Icon(Icons.home),
        title: Text('Home Page'),
        actions: [
          IconButton(
              onPressed: () {
                signOut();
              },
              icon: Icon(Icons.login))
        ],
      ),
```

ภาพประกอบ 8.52 การเขียน Code เพิ่ม IconButton ไว้บน AppBar
ที่มา : ณปภัช วรรณตรง (2564 : ...)

```
Future<Null> signOut() async {
   await Firebase.initializeApp().then((value) async {
     await FirebaseAuth.instance.signOut().then((value) {
       Navigator.push(
           context,
           MaterialPageRoute(
             builder: (context) => Authentication(),
           ));
     });
   });
 }
```

ภาพประกอบ 8.53 การเขียน Code ล็อกเอาท์ หรือลงชื่อออกจากระบบ
ที่มา : ณปภัช วรรณตรง (2564 : ...)

การใช้งาน image_picker ร่วมกับ Storage

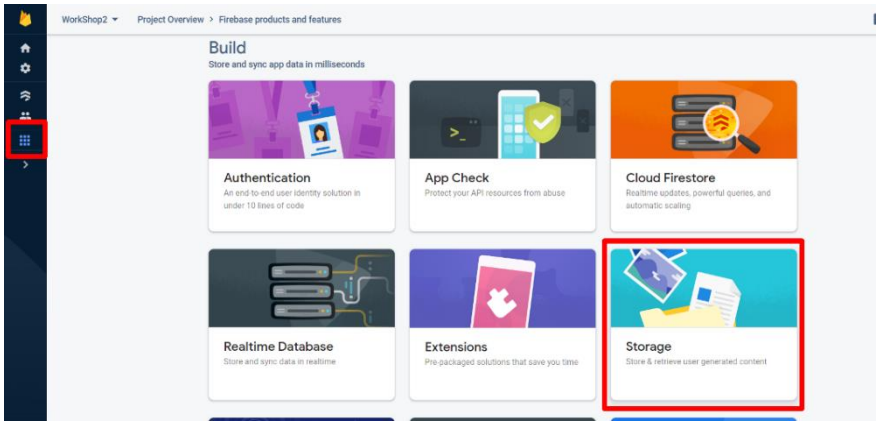
image_picker เป็นปลั๊กอิน (Plugin) ของ Flutter สำหรับ iOS และ Android สำหรับเลือกภาพจากไลบรารีรูปภาพ และถ่ายภาพใหม่ด้วยกล้อง สำหรับการเพิ่มรูปจากกล้อง แกลลอรี่ และวีดีโอในโปรเจกต์ Flutter (pub.dev, 2566)

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

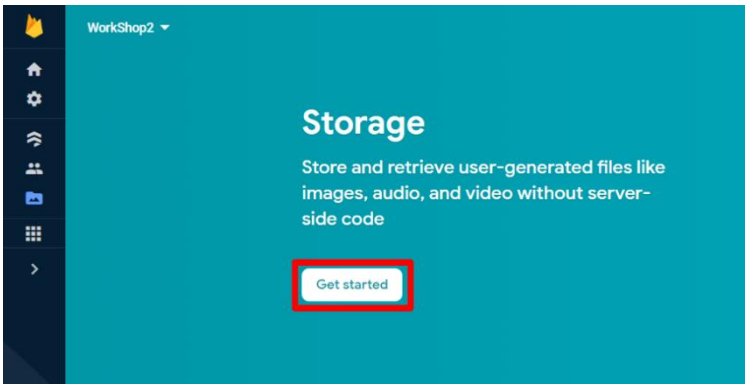
เริ่มต้นด้วยการติดตั้ง Library ดังต่อไปนี้

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



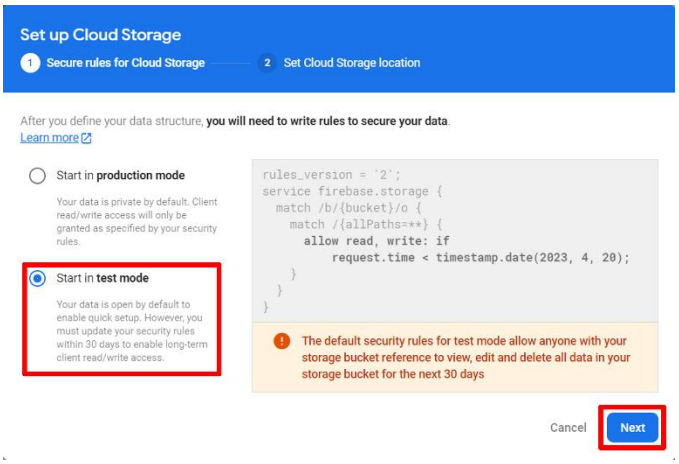
ภาพประกอบ 8.55 การเปิดการใช้งาน Storage

ที่มา : Firebase. (2020 : 1)

2.3 หลังจากที่ทำการกดสร้าง Cloud Storage จะแสดงดังภาพ

2.3.1 เลือก test mode

2.3.1 กด Next

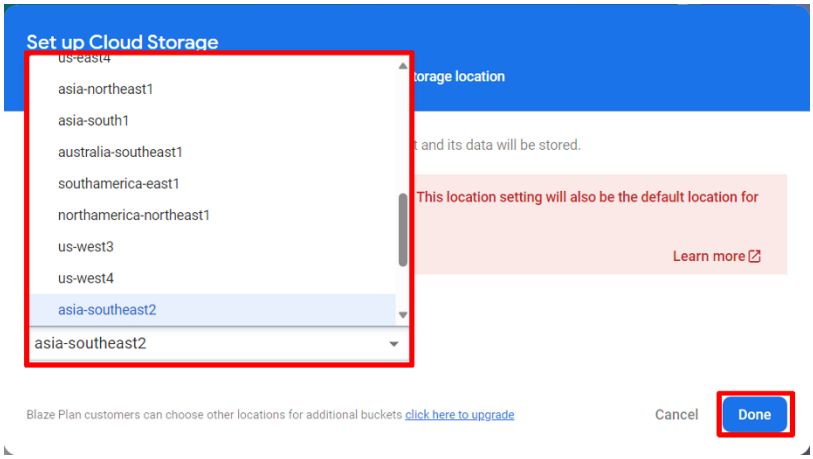


ภาพประกอบ 8.56 การตั้งค่าเริ่มต้นให้กับ Storage

ที่มา : Firebase. (2020 : 1)

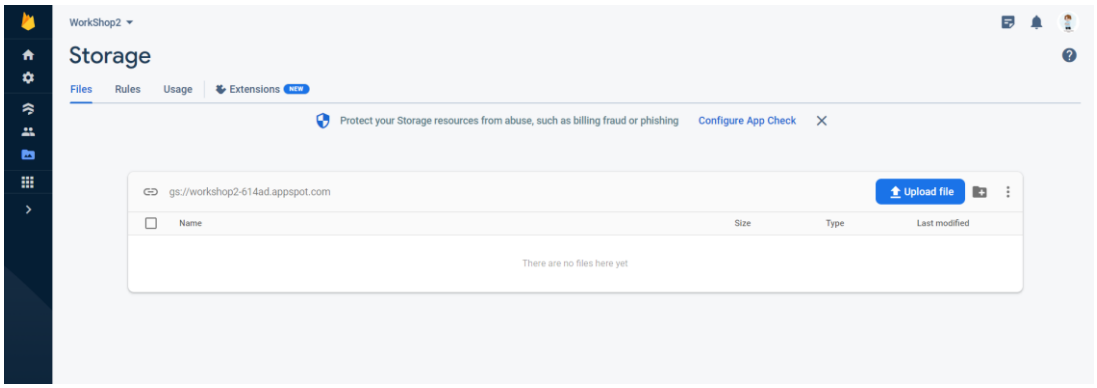
2.3.3 เลือกที่ตั้งในการเก็บข้อมูลของ Cloud Storage ของโปรเจกต์ หากทำการเลือกที่เก็บข้อมูลตั้งแต่ใช้ Firestore Database ไม่ต้องเลือกที่เก็บข้อมูลใหม่สามารถกดสร้าง Cloud Storage ได้ทันที

2.3.4 กด Done ดังภาพ



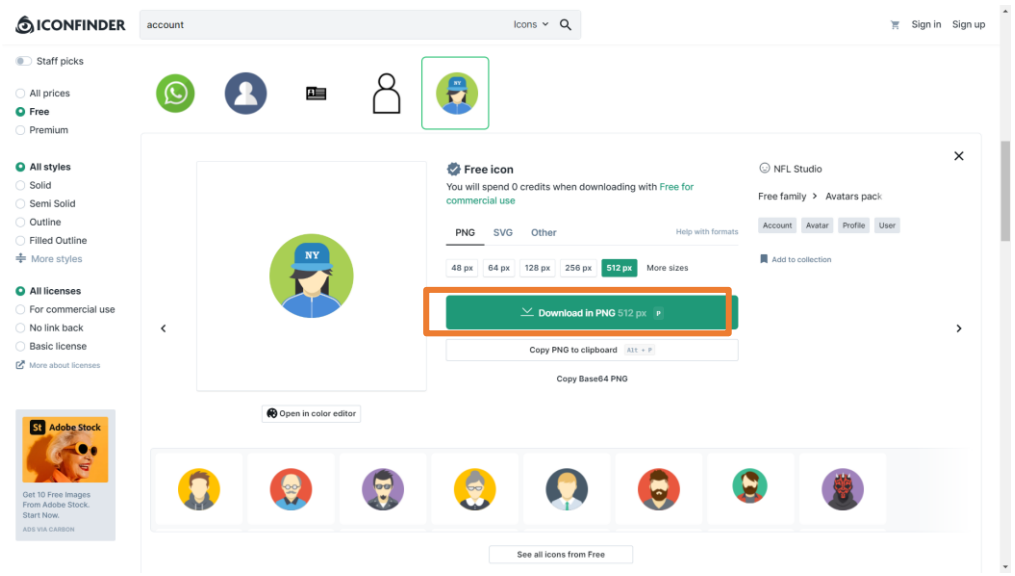
ภาพประกอบ 8.57 การกำหนดตำแหน่งในการเก็บข้อมูลให้กับ Cloud Storage
ที่มา : Firebase. (2020 : 1)

2.4 หลังจากที่ทำการสร้างเรียบร้อยแล้ว จะได้ดังภาพ



ภาพประกอบ 8.58 Cloud Storage สำหรับการใช้เก็บข้อมูลรูปภาพ Firebase
ที่มา : Firebase. (2020 : 1)

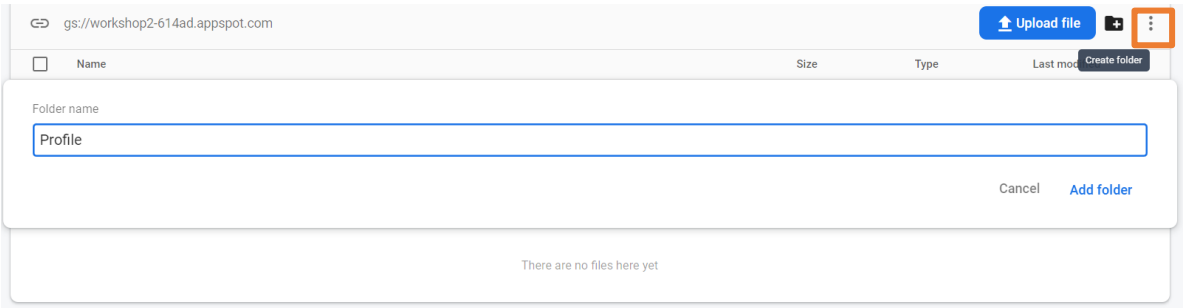
2.5 ไปที่เว็ปไซต์ https://www.iconfinder.com แล้วใส่คำค้นหาว่า Account จากนั้นให้เลือกรูปที่ชอบแล้วดาวน์โหลดเก็บไว้ ดังภาพ



ภาพประกอบ 8.59 การดาวน์โหลดรูปภาพผ่านเว็ปไซต์ Iconfinder

ที่มา : Iconfinder. (2020 : 1)

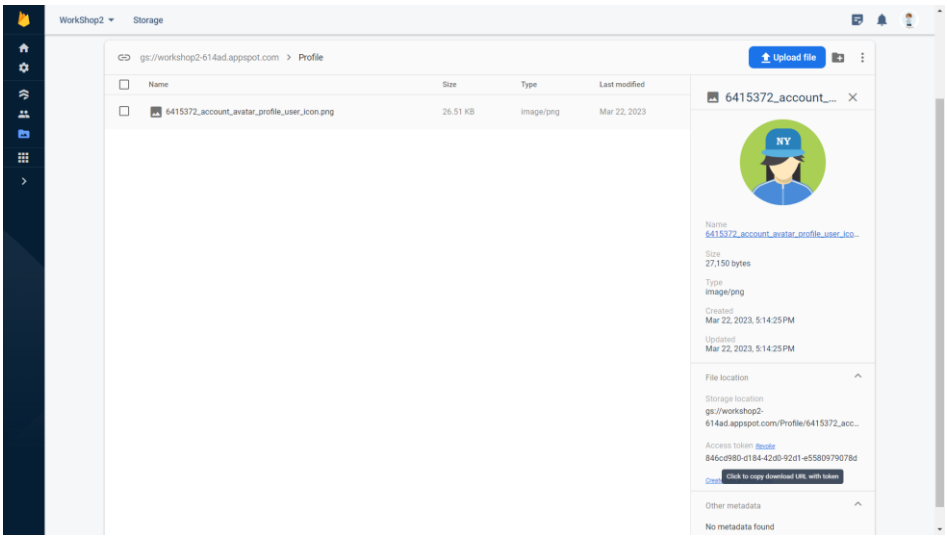
2.6 สร้างโฟลเดอร์ Profile ดังภาพ



ภาพประกอบ 8.60 การสร้างโฟลเดอร์บน Cloud Storage

ที่มา : Firebase. (2020 : 1)

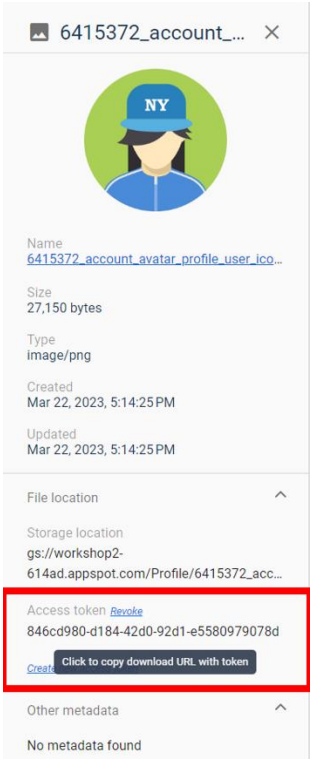
2.7 เมื่อสร้างโฟลเดอร์เสร็จเรียบร้อย ให้กดเข้าไปที่โฟลเดอร์ Profile เพิ่มไฟล์รูปภาพที่ดาวน์โหลดไว้ ดังภาพ



ภาพประกอบ 8.61 ไฟล์รูปภาพที่ทำการอัพโหลดไปยัง Cloud Storage

ที่มา : Firebase. (2020 : 1)

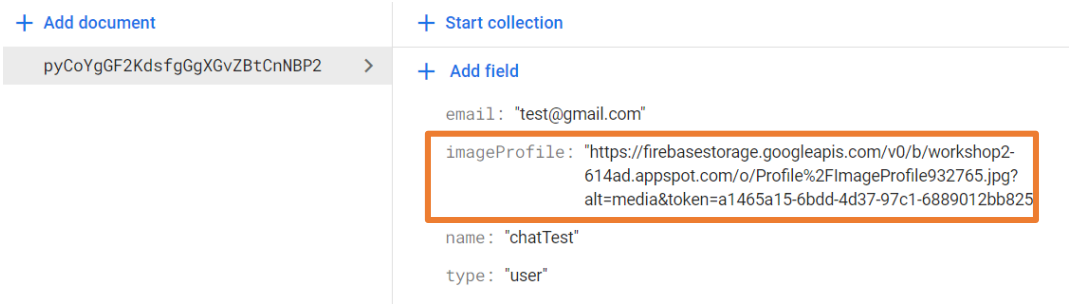
2.8 ไปที่ File location หา Access token แล้ว Copy ไว้ดังภาพ



ภาพประกอบ 8.62 การ Copy Access token

ที่มา : Firebase. (2020 : 1)

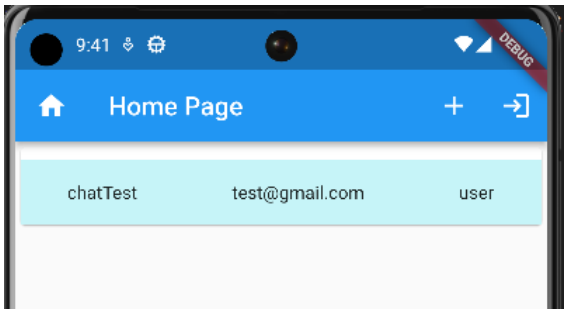
2.9 นำ Storage location ไปทำการสร้าง Field ใน Ducment ของ User โดยมีชื่อ Field ว่า imageProfile จะได้ดังภาพ



ภาพประกอบ 8.63 การเพิ่ม Field imageProfile

ที่มา : Firebase. (2020 : 1)

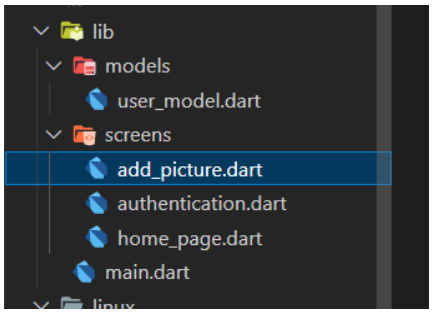
2.10 เพิ่ม Icon Button เพื่อกดไปยังหน้า AddPicturePage ใน AppBar ไว้ข้างหน้าปุ่มออกจากระบบ ดังภาพ



ภาพประกอบ 8.64 การเพิ่ม Icon Button

ที่มา : ณปภัช วรรณตรง (2564 : ...)

2.11 เพิ่มไฟล์ใหม่ ทำการเขียน Code ในไฟล์ให้เรียบร้อยโดยมีชื่อ Class ว่า AddPicturePage ดังภาพ

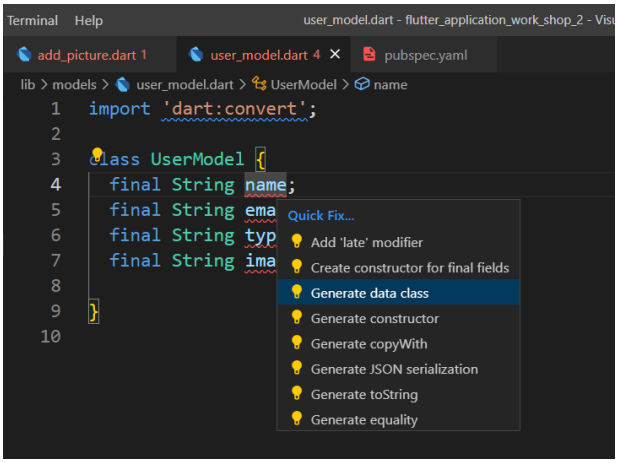


ภาพประกอบ 8.65 การเพิ่มไฟล์ add_picture.dart

ที่มา : ณปภัช วรรณตรง (2564 : ...)

2.12 เขียนโค้ดเมื่อทำการกดไอคอนที่เพิ่มขึ้นมาใหม่สามารถไปยังหน้า AddPicturePage

2.13 แก้ไข UserModel เพิ่ม final String imageProfile; ให้ Generate data Class ใหม่ ดังภาพ



ภาพประกอบ 8.66 การ Generate data Class ใหม่

ที่มา : ณปภัช วรรณตรง (2564 : ...)

2.14 เขียนโค้ดแสดงผลข้อมูลที่เก็บไว้บน Firestore Database และ Cloud Storage เริ่มด้วยการประกาศตัวแปร เขียน Future Fucntion นำ Future ที่สร้างขึ้นไปไว้ใน initState(){} ให้เรียบร้อย ดังภาพ

```dart
String? urlImageProfile, uidUser, newUrlImageProfile;
  UserModel? userModel;

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

  Future<Null> readDataUserLogin() async {
    FirebaseAuth.instance.authStateChanges().listen(
      (event) async {
        uidUser = event!.uid;
        FirebaseFirestore.instance
            .collection('userTable')
            .doc(uidUser)
            .snapshots()
            .listen(
          (event) async {
            setState(
              () {
                userModel = UserModel.fromMap(
                  event.data()!,
                );
              },
            );
            urlImageProfile = userModel!.imageProfile;
          },
        );
      },
    );
  }
```

ภาพประกอบ 8.67 ตัวอย่าง Code หาข้อมูลผู้ใช้งานมาเก็บไว้ใน userModel
ที่มา : ณปภัช วรรณตรง (2564 : ...)

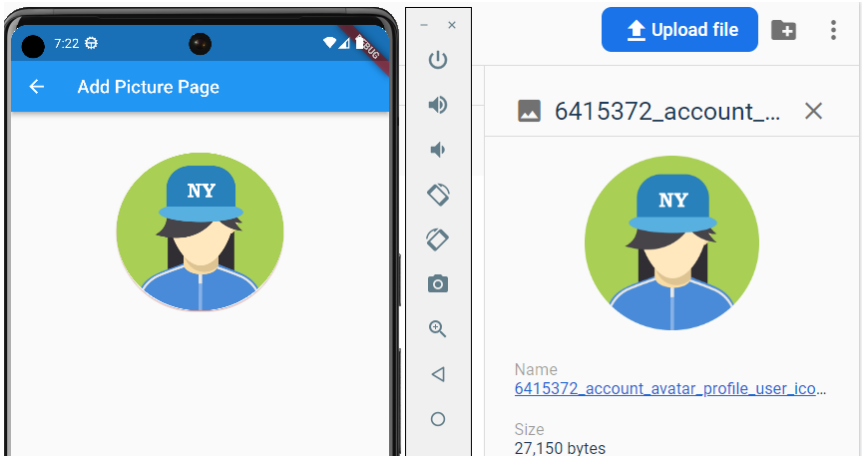
2.15 สร้าง Widget showImage แล้วนำไปใส่ในส่วนของ body ดังภาพ

```
Widget showImage() {
    return Container(
      margin: EdgeInsets.only(top: 40),
      width: 190,
      height: 180,
      child: Card(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.all(
            Radius.circular(100),
          ),
        ),
        shadowColor: Colors.red,
        child: Center(
          child: ClipOval(
            child: file == null
                ? Image.network(
                    urlImageProfile!,
                    width: 200,
                    height: 200,
                    fit: BoxFit.cover,
                  )
                : Image.file(
                    file!,
                    width: 200,
                    height: 200,
                    fit: BoxFit.cover,
                  ),
          ),
        ),
      ),
    );
  }

```

ภาพประกอบ 8.68 ตัวอย่าง Code Widget showImage();

ที่มา : ณปภัช วรรณตรง (2564 : ...)

2.16 เมื่อนำ Widget showImage ไปใส่ในส่วนของ Body แล้ว ทำการสั่งการทำงานโปรเจกต์ แล้วดูผลลัพธ์ดังภาพ



ภาพประกอบ 8.69 ผลการแสดงรูปภาพบน Emulator

ที่มา : ณปภัช วรรณตรง (2564 : ...)

2.16 สร้าง Future ในการเรียกใช้งาน image_picker ดังภาพ

```
Future<Null> choooseImage(ImageSource imageSource) async {
    try {
      var object = await ImagePicker().pickImage(
        source: imageSource,
        maxHeight: 500,
        maxWidth: 500,
      );
      setState(() {
        file = File(object!.path);
      });
    } catch (e) {}
  }
```

ภาพประกอบ 8.70 ตัวอย่าง Code Future ในการเรียกใช้งาน image_picker

ที่มา : ณปภัช วรรณตรง (2564 : ...)

2.17 สร้างปุ่มในการกดเลือก gallery หรือ camera เมื่อทำการกดปุ่มให้เรียกใช้งาน Future chooseImage ดังภาพ

```
Widget choooseImageCamera() {
    return Container(
      padding: EdgeInsets.all(10),
      child: GestureDetector(
        onTap: () => choooseImage(ImageSource.camera),
        child: Container(
          width: 200,
          child: Card(
            color: Colors.blue,
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.all(
                Radius.circular(20),
              ),
            ),
            child: Row(
              children: [
                Container(
                    padding: EdgeInsets.all(10),
                    child: Icon(
                      Icons.camera_alt_rounded,
                      size: 30,
                      color: Colors.white,
                    )),
                Text(
                  'เลือกจากกล้อง',
                  style: TextStyle(
                    fontSize: 18,
                    color: Colors.white,
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
```

ภาพประกอบ 8.71 ตัวอย่าง Code เมื่อทำการกดจะเลือกจาก Camera

ที่มา : ณปภัช วรรณตรง (2564 : ...)

```dart
Widget choooseImageGallery() {
  return Container(
    padding: EdgeInsets.all(10),
    child: GestureDetector(
      onTap: () => choooseImage(ImageSource.gallery),
      child: Container(
        width: 200,
        child: Card(
          color: Colors.blue,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.all(
              Radius.circular(20),
            ),
          ),
          child: Row(
            children: [
              Container(
                  padding: EdgeInsets.all(10),
                  child: Icon(
                    Icons.image,
                    size: 30,
                    color: Colors.white,
                  )),
              Text(
                'เลือกจากแกลลอรี่',
                style: TextStyle(
                  fontSize: 18,
                  color: Colors.white,
                ),
              ),
            ],
          ),
        ),
      ),
    ),
  );
}
```

ภาพประกอบ 8.72 ตัวอย่าง Code เมื่อทำการกดจะเลือกจาก Gallery

ที่มา : ณปภัช วรรณตรง (2564 : ...)

2.18 ทำการเขียน Future ในการบันทึกข้อมูลไปยัง Storage และ Cloud Firestore ดังภาพ

```dart
Future<Null> upLoadPictureToStoage() async {
  Random random = Random();
  int i = random.nextInt(1000000);

  Firebase.initializeApp().then((value) async {
    FirebaseStorage storage = FirebaseStorage.instance;
    Reference reference = storage.ref().child('Profile/ImageProfile$i.jpg');
    UploadTask uploadTask = reference.putFile(file!);

    newUrlImageProfile = await (await uploadTask).ref.getDownloadURL();
    print('UrlImage ===>>> $newUrlImageProfile');

    upDatePictureToCloudFriestore();
  });
}

Future<Null> upDatePictureToCloudFriestore() async {
  Firebase.initializeApp().then((value) async {
    FirebaseFirestore.instance
        .collection('userTable')
        .doc(uidUser)
        .update({"imageProfile": newUrlImageProfile}).then(
      (value) async {},
    );
  });
}
```

ภาพประกอบ 8.73 ตัวอย่าง Code บันทึกข้อมูลไปยัง Storage และ Cloud Firestore
ที่มา : ณปภัช วรรณตรง (2564 : ...)

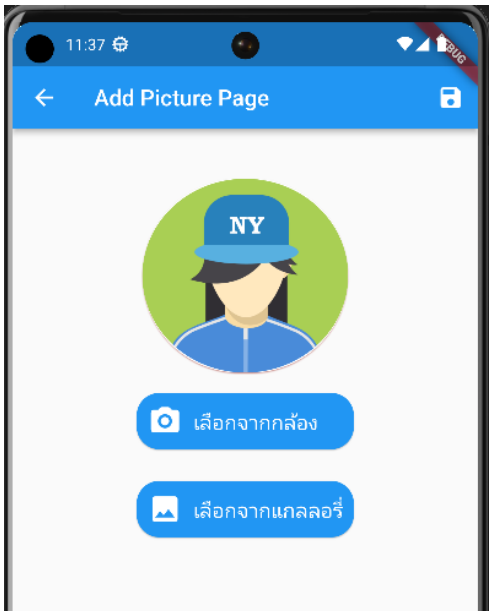
จากภาพประกอบ 8.74 การทำงานของ Future จะเริ่มต้นทำงานที่ Future upLoadPictureTostoage ให้เสร็จเรียบร้อยก่อน แล้วถึงจะไปเรียกใช้งาน Future upDatePictureToColundFirestore

2.19 สร้างปุ่มในการบันทึกข้อมูล เมื่อกดให้เรียกใช้งาน FutureupLoadPictureTostoage ดังภาพ

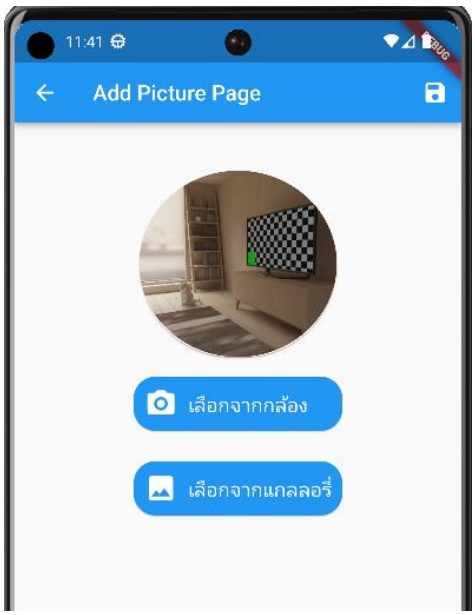
```
appBar: AppBar(
        actions: [
          IconButton(
              onPressed: () {
                upLoadPictureToStoage();
              },
              icon: Icon(Icons.save))
        ],
        title: Text('Add Picture Page'),
      ),
```

ภาพประกอบ 8.74 ตัวอย่าง Code ปุ่มในการกดบันทึกข้อมูล
ที่มา : ณปภัช วรรณตรง (2564 : ...)

2.20 สั่งการทำงานโปรเจกต์ แล้วไปยังหน้าที่จะทำการทดสอบอับโหลดรูปภาพไปยัง Storage และ Cloud Firestore ดังภาพ

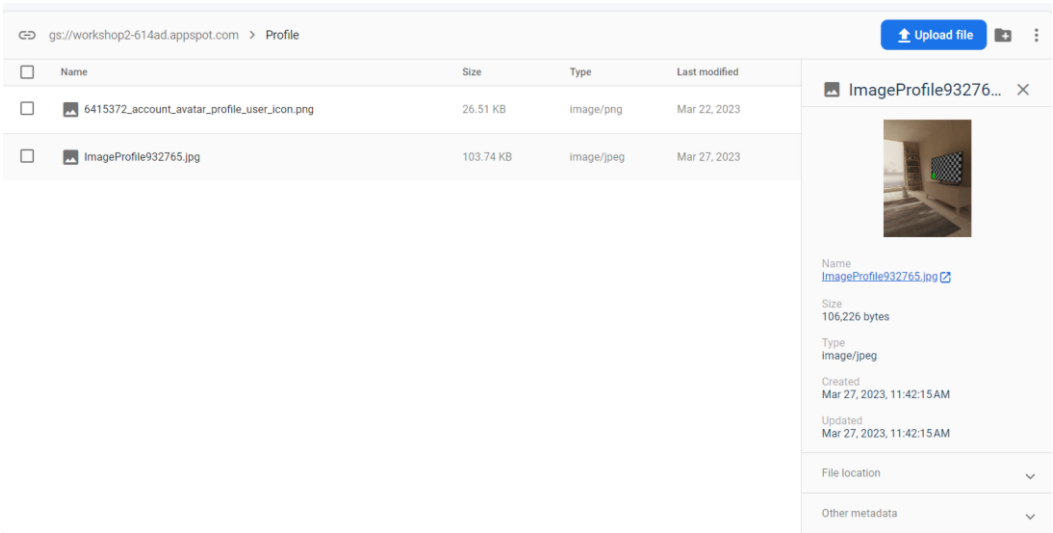


ภาพประกอบ 8.75 ก่อนการเพิ่มรูปภาพ
ที่มา : ณปภัช วรรณตรง (2564 : ...)



ภาพประกอบ 8.76 หลังการเพิ่มรูปภาพ
ที่มา : ณปภัช วรรณตรง (2564 : ...)

หลังจากที่เพิ่มรูปภาพเสร็จแล้ว ทำการบันทึกรูปภาพไปยัง Storage และ Cloud Firestore ไปเช็ครูปภาพที่ Storage และนำ Access token ของรูปภาพที่อัพโหลดขึ้นไปเช็คกับ Cloud Firestore ที่ Field : imageProfile ว่าตรงกันหรือไม่



ภาพประกอบ 8.77 รูปภาพที่อัพขึ้นไปใหม่
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

```dart
Future<Null> checkAuthen() async {
    await Firebase.initializeApp().then((value) async {
      await FirebaseAuth.instance
          .signInWithEmailAndPassword(email: email!, password: password!)
          .then((value) async {
        String? uid = value.user!.uid;
        await FirebaseFirestore.instance
            .collection('userTable')
            .doc(uid)
            .snapshots()
            .listen((event) {
          UserModel model = UserModel.fromMap(event.data()!);
          switch (model.type) {
            case 'user':
              Navigator.pushAndRemoveUntil(
                  context,
                  MaterialPageRoute(
                    builder: (context) => ChatPage(),
                  ),
                  (route) => false);
              break;
            default:
          }
        });
      });
    });
  }
```

ภาพประกอบ 8.78 Code แก้ไขการ Route ไปยัง ChatPage
ที่มา : ณปภัช วรรณตรง (2564 : ...)

3. ประกาศตัวแปรที่ใช้ในหน้า ChatPage ดังภาพประกอบ 8.80

```dart
TextEditingController msg = new TextEditingController();
String? title, body, nameUser;
FirebaseFirestore _firestore = FirebaseFirestore.instance;
UserModel? userModel;
```

ภาพประกอบ 8.79 การประกาศตัวแปรที่ใช้ในหน้า ChatPage
ที่มา : ณปภัช วรรณตรง (2564 : ...)

4. เขียน Future อ่านข้อมูลผู้ใช้งานจาก Firestore Database ไปเก็บไว้ในตัวแปร nameUser ดังภาพ

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

Future<Null> readDataUserLogin() async {
  FirebaseAuth.instance.authStateChanges().listen(
    (event) async {
      String uidUser = event!.uid;
      FirebaseFirestore.instance
          .collection('userTable')
          .doc(uidUser)
          .snapshots()
          .listen(
        (event) async {
          setState(
            () {
              userModel = UserModel.fromMap(
                event.data()!,
              );
            },
          );
          nameUser = userModel!.name;
        },
      );
    },
  );
}
```

ภาพประกอบ 8.80 Code การอ่านข้อมูลจาก Cloud Firestore Database

ที่มา : ณปภัช วรรณตรง (2564 : ...)

5. ออกแบบในส่วนของ AppBar โดยจะแสดงชื่อแล้วปุ่มออกจากระบบ ดังภาพ

```dart
return Scaffold(
  appBar: AppBar(
    actions: [
      IconButton(
          onPressed: () {
            signOut();
          },
          icon: Icon(Icons.login)),
    ],
    title: Text('คุณ $nameUser'),
  ),
```

ภาพประกอบ 8.81 Code การออกแบบปุ่มออกจากระบบ

ที่มา : ณปภัช วรรณตรง (2564 : ...)

จากภาพประกอบ 8.82 จะเห็นได้ว่าเมื่อมีการกดปุ่มทุกครั้งจะเรียกใช้งาน Future signOut เป็น Future ที่เขียนขึ้นมาใช้สำหรับการออกจากระบบการใช้งานของผู้ใช้งาน ดังภาพ

```
Future<Null> signOut() async {
  await Firebase.initializeApp().then((value) async {
    await FirebaseAuth.instance.signOut().then((value) {
      Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => Authentication(),
          ));
    });
  });
}
```

ภาพประกอบ 8.82 Code การออกจาระบบของ Fiebase
ที่มา : ณปภัช วรรณตรง (2564 : ...)

6. ออกแบบในส่วนของ Body จะ Query Snapshot จาก Cloud Firestore Database มาแสดงในรูปแบบ ListView และ Return ค่ากลับในรูปแบบของ Widget ที่มีชื่อว่า Message ดังภาพ

```
body: Container(
  height: size.height / 1.25,
  width: size.width,
  child: StreamBuilder<QuerySnapshot>(
    stream: _firestore
        .collection('chatTable')
        .orderBy('time', descending: false)
        .snapshots(),
    builder: (context, snapshot) {
      if (snapshot.data != null) {
        return ListView.builder(
          itemCount: snapshot.data!.docs.length,
          itemBuilder: (context, index) {
            var map = snapshot.data!.docs[index].data();
            return message(size, map);
          },
        );
      } else {
        return Container();
      }
    },
  ),
),
```

ภาพประกอบ 8.83 ตัวอย่าง Code ในส่วนของ Body
ที่มา : ณปภัช วรรณตรง (2564 : ...)

```dart
Widget message(Size size, var map) {
  return Container(
    width: size.width,
    alignment: map['sendby'] == nameUser
        ? Alignment.centerRight
        : Alignment.centerLeft,
    child: Container(
      padding: EdgeInsets.symmetric(vertical: 10, horizontal: 15),
      margin: EdgeInsets.symmetric(vertical: 10, horizontal: 15),
      decoration: BoxDecoration(
        color: Colors.blue,
        borderRadius: BorderRadius.circular(20),
      ),
      child: Text(
        map['message'],
        style: TextStyle(color: Colors.white, fontSize: 16),
      ),
    ),
  );
}
```

ภาพประกอบ 8.84 ตัวอย่าง Code Widget Message

ที่มา : ณปภัช วรรณตรง (2564 : ...)

จากภาพประกอบ 8.85 เป็นวิดเจ็ตที่เขียนเพื่อแสดงข้อความในหน้าแอปพลิเคชันโดยจะแยกข้อความจากชื่อผู้ใช้งานที่ทำการส่งข้อความ โดยใน Widget นี้จะมีการส่งค่ากลับไปยัง ListView ที่อยู่ในส่วนของ body

7. ส่วนด้านล่างสุดจะออกแบบสำหรับการกรอกข้อความ และปุ่มในการกดส่งข้อความ เมื่อทำการกดส่งข้อความจะเรียกใช้ void function ที่มีชื่อว่า onSentMessage() ดังภาพ

```
floatingActionButton: Container(
      margin: EdgeInsets.only(left: 40, top: 100),
      height: size.height / 8,
      width: size.height,
      alignment: Alignment.center,
      child: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Container(
          height: size.height / 10,
          width: size.height,
          child: TextField(
            controller: msg,
            decoration: InputDecoration(
              suffixIcon: IconButton(
                  onPressed: () {
                    onSentMessage();
                  },
                  icon: Icon(
                    Icons.send,
                    color: Colors.blue,
                  )),
              hintText: 'กรุณากรอกข้อความที่ต้องการส่ง...',
              border: InputBorder.none,
              enabledBorder: OutlineInputBorder(
                borderRadius: BorderRadius.circular(10),
                borderSide: BorderSide(color: Colors.blue),
              ),
              focusedBorder: OutlineInputBorder(
                borderRadius: BorderRadius.circular(10),
                borderSide: BorderSide(color: Colors.blue),
              ),
            ),
          ),
        ),
      ),
    ),
```

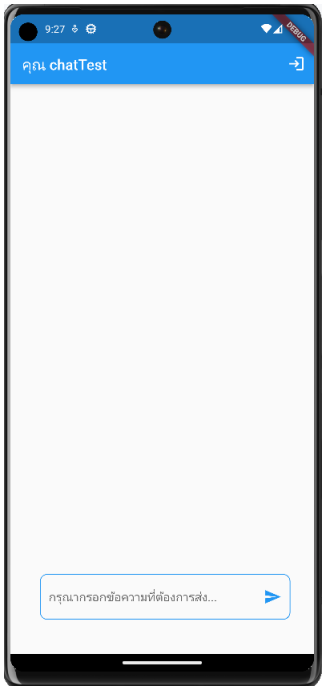
ภาพประกอบ 8.85 ตัวอย่าง Code ในส่วนของการกรอกและส่งข้อความ
ที่มา : ณปภัช วรรณตรง (2564 : ...)

```dart
void onSentMessage() async {
  if (msg.text.isNotEmpty) {
    Map<String, dynamic> message = {
      'sendby': nameUser,
      'message': msg.text,
      'time': FieldValue.serverTimestamp(),
      // 'uidRest': queueModel.uidRest,
    };

    await _firestore.collection('chatTable').add(message);
    msg.clear();
  } else {
    print("Enter Some Text");
  }
}
```

ภาพประกอบ 8.86 ตัวอย่าง Code void function onSentMessage()
ที่มา : ณปภัช วรรณตรง (2564 : ...)

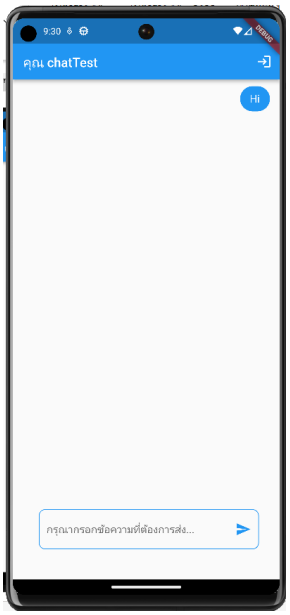
จากภาพประกอบ 8.87 เป็น void function ที่เขียนขึ้นมาเพื่อใช้รับค่าจาก TextField เมื่อมีการกดส่งข้อความและบันทึกไปยัง Cloud Firestore Database

8. สั่งการทำงานโปรเจกต์ ทำการล็อกอินเพื่อไปยังหน้า ChatPage ดังภาพ



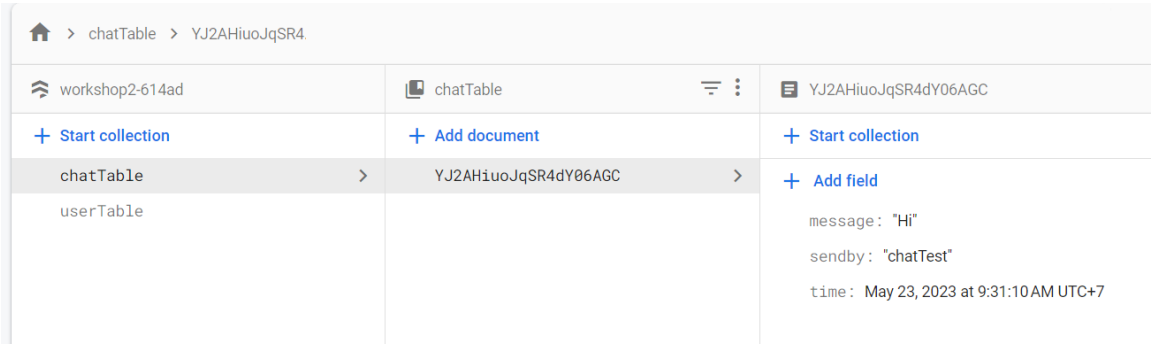
ภาพประกอบ 8.87 หน้า ChatPage
ที่มา : ณปภัช วรรณตรง (2564 : ...)

9. ทดสอบพิมพ์ข้อความแล้วทำการส่งข้อความจะได้ดังภาพ



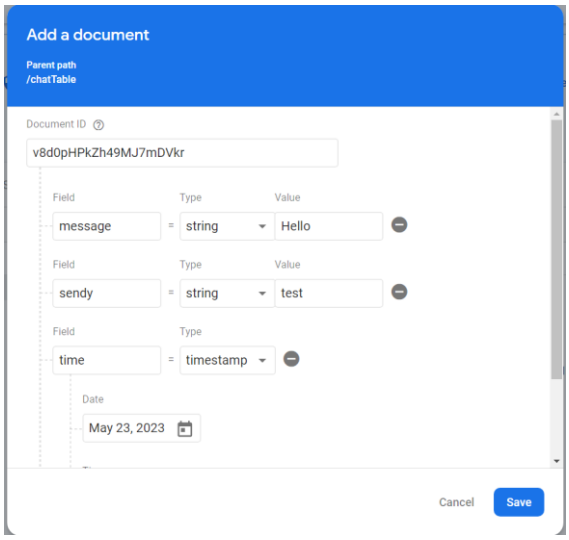
ภาพประกอบ 8.88 หน้า ChatPage ที่มีข้อความ
ที่มา : ณปภัช วรรณตรง (2564 : ...)

จากภาพประกอบ 8.89 ที่ได้ส่งข้อความและแสดงผล เมื่อไปที่ Cloud Firestore Database จะพบข้อมูลที่ได้ทำการส่งโดยจะมี ข้อความที่ส่ง เวลาที่ส่ง ชื่อผู้ส่ง ดังภาพ



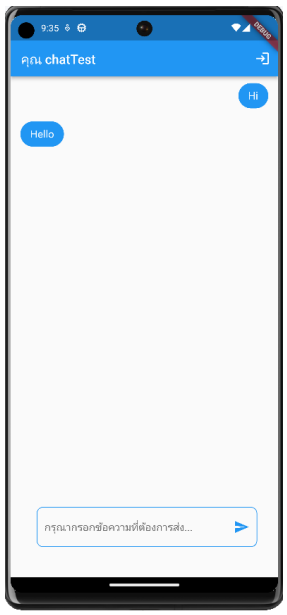
ภาพประกอบ 8.89 ฐานข้อมูล Cloud Firestore Database collection chatTable
ที่มา : Firebase (2020 : 1)

10. เพิ่ม Document ข้างใน Collection chatTable ดังภาพ



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 : ณปภัช วรรณตรง (2564 : ...)

จากภาพประกอบ 8.92 เมื่อลองเพิ่ม Document ข้างใน Collection chatTableประกอบด้วย ข้อความ วันเวลา ชื่อผู้ส่ง หากต้องการทดสอบพิมพ์ข้อความโต้ตอบให้ทำการสร้างบัญชีผู้ใช้เพิ่มขึ้นมา

บทสรุป

สำหรับเนื้อหาที่กลาวมาทั้งหมดในบทนี้ จะพูดถึงนิยามของ Firebase ลักษณะของ Firebase Database การใช้งาน Flutter ร่วมกับ Firebase การติดตั้ง Library ในโปรเจกต์ Flutter การใช้งาน Cloud Firestore การแสดงผลข้อมูลจาก Firestore Database การใช้งาน Firebase Authenication การใช้งาน image_picker ร่วมกับ Storage และประยุกต์ใช้ความรู้ด้วยการสร้างแอปพลิเคชัน Chat ด้วย Firebase

เอกสารอ้างอิง

จีราวุธ 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.

Fiebase (1 มีนาคม 2566). **Cloud Storage for Firebase. สืบค้นจาก,** https://firebase.google.com/docs/storage

Fiebase (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

บรรณานุกรม

จีราวุธ วารินทร์. (2563). **การพัฒนาเว็บแอปพลิเคชันด้วย Firebase ร่วมกับ React.** กรุงเทพฯ: ริโวว่า.

_____. (2563). **การพัฒนาเว็บแอปพลิเคชันด้วย Firebase ร่วมกับ Vue.js.** กรุงเทพฯ: สำนักพิมพ์ซิมพลิฟาย.

_____. (2564). **พัฒนาโมไบล์แอพด้วย Flutter + Dart.** กรุงเทพฯ : สำนักพิมพ์ซิมพลิฟาย.

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.

ลุงวิศวสอนคำนวณ. (11 กันยายน 2564). **สร้างเว็บไซต์ด้วย Python Django (แจงโก้) เฟรมเวิร์กใช้เขียนเว็บยอดฮิตของ Python.** สืบค้นจาก, https://www.facebook.com/UncleEngineer

ศุภชัย สมพานิช. (2563). **การพัฒนาแอปพลิเคชันด้วย Firebase&Kotin.** กรุงเทพฯ: โปรวิชั่น.

สำนักงานสถิติแห่งชาติ. (2564). **สรุปผลที่สำคัญสำรวจการมีการใช้เทคโนโลยีสารสนเทศและการสื่อสารในครัวเรือน 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 : โปรวิชั่น, 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/

Fiebase. (1 มีนาคม 2566). **Cloud Storage for Firebase.** สืบค้นจาก, https://firebase.google.com/docs/storage

_____ (1 มีนาคม 2566). **Firebase 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). **พัฒนาแอพด้วย Flutter สำหรับผู้เริ่มต้น 7 ชั่วโมงเต็ม [FULL COURSE].** สืบค้นจาก, 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

sqlitebrowser. (10 กุมภาพันธ์ 2566). **DB Browser for SQLite.** สืบค้นจาก, https://sqlitebrowser.org/