ROYAL UNIVERSITY OF BHUTAN
COLLEGE OF SCIENCE AND TECHNOLOGY
PHUENTSHOLING: BHUTAN

PLAGIARISM DECLARATION FORM

Student Name: Divyash Chhetri

Student No: 02200174

Module No and Title of the module: CTE306 - Mobile Application Development
Assignment no and Title of the Assignment: Lab Wok 10

Section H2 of the Royal University of Bhutan's *Wheel of Academic Law* provides the following definition of academic dishonesty:

"Academic dishonesty may be defined as any attempt by a student to gain an unfair advantage in any assessment. It may be demonstrated by one of the following:

Collusion: the representation of a piece of unauthorized group work as the work of a single candidate.

Commissioning: submitting an assignment done by another person as the student's own work.

Duplication: the inclusion in coursework of material identical or substantially similar to material which has already been submitted for any other assessment within the University.

False declaration: making a false declaration in order to receive special consideration by an Examination Board or to obtain extensions to deadlines or exemption from work.

Falsification of data: presentation of data in laboratory reports, projects, etc., based on work purported to have been carried out by the student, which have been invented, altered or copied by the student.

Plagiarism: the unacknowledged use of another's work as if it were one's own.

Examples are:

- verbatim copying of another's work without acknowledgement
- paraphrasing of another's work by simply changing a few words or altering the order of presentation, without acknowledgement
- ideas or intellectual data in any form presented as one's own without acknowledging the source(s)
- making significant use of unattributed digital images such as graphs, tables, photographs, etc. taken from test books, articles, films, plays, handouts, internet, or any other source, whether published or unpublished
- submission of a piece of work which has previously been assessed for a different award or module or at a different institution as if it were new work
- use of any material without prior permission of copyright from appropriate authority or owner of the materials used"

Student Declaration

I confirm that I have read and understood the above definitions of academic dishonesty. I declare that I have not committed any academic dishonesty when completing the attached piece of work.

Signature of Student: Divyash Chhetri

Date: 11 Oct 2022

Lab 10

CTE306 – Mobile Application Development

Date – 11th October 2022

Divyash Chhetri
02200174
BE 3 IT

Module Tutor: Mr. Pema Galey

Department of Information Technology
College of Science and Technology

Aim

To Setup Flutter in Your System and Develop your First app.

Instructions

Perform the task on following topics:

1. Setup Environment for Flutter (Windows or Linux or macOS)
2. Create First App
3. Replace **main.dart** and **pubspec.yaml** with attached files
4. Run Flutter using Android Emulator

Theory

Flutter is a Google-created open and free mobile UI framework that was introduced in May 2017. In a nutshell, it enables you to develop a native mobile app using only one codebase. This implies you may design two separate apps using the same programming language and codebase (for iOS and Android).

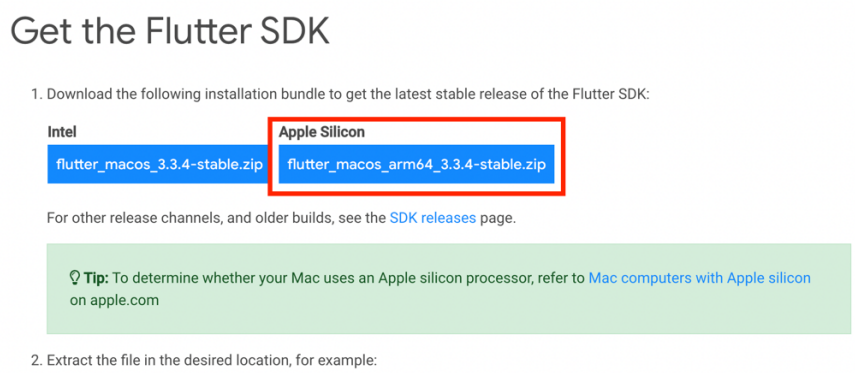
Flutter is made up of two main components:

1. An SDK (Software Development Kit) is a set of tools that will assist you in the development of your apps. Tools for compiling your code into native machine code are included (code for iOS and Android).

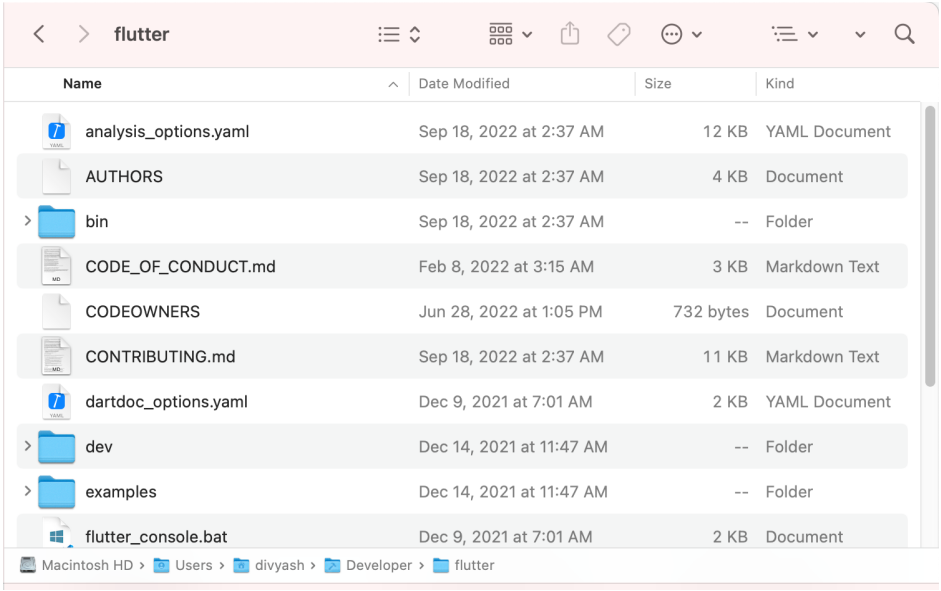
2. A Framework (Widget-based UI Library): A set of reusable user interface components (buttons, text inputs, sliders, and so on) that you may customize to meet your specific needs.You'll utilize the Dart programming language to create Flutter apps. Google established the language in October 2011, but it has come a long way in the last several years. Dart is a front-end programming language that may be used to construct mobile and online apps. Dart is a typed object programming language that you may learn if you have some programming experience. Dart's syntax is comparable to JavaScript.

Procedure to Setup Flutter

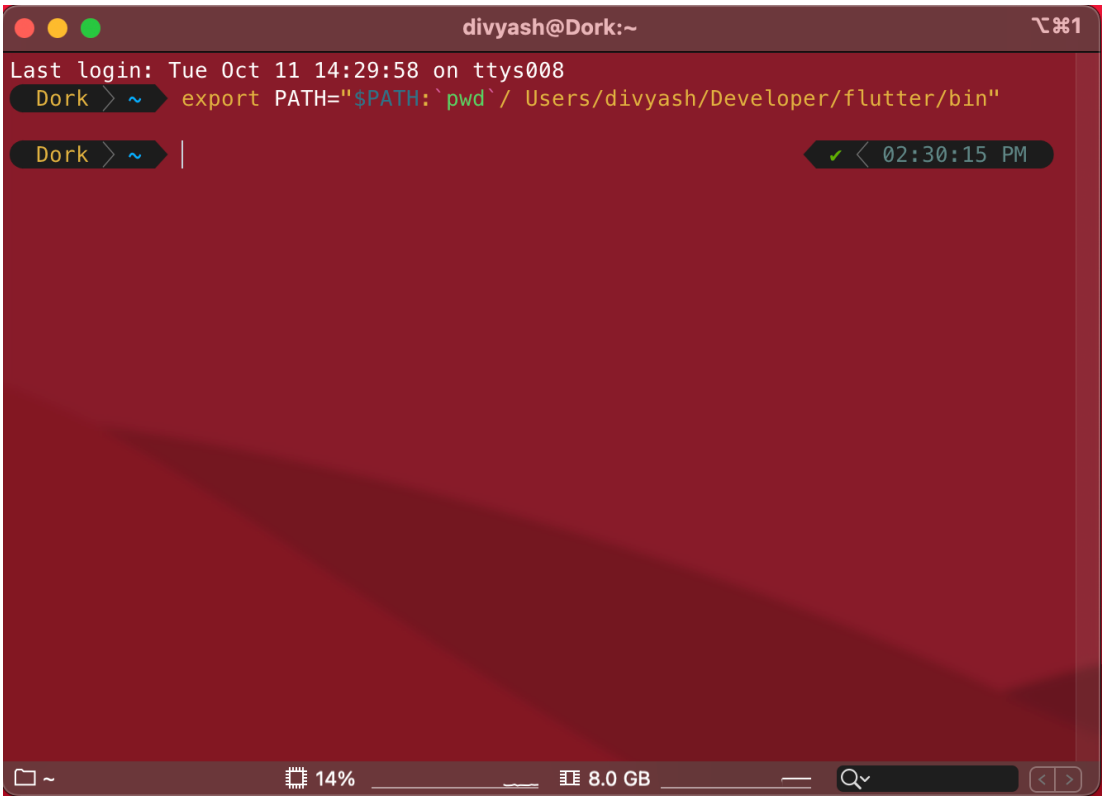
1. Visit https://docs.flutter.dev/get-started/install/macos and download the installation bundle for apple silicon



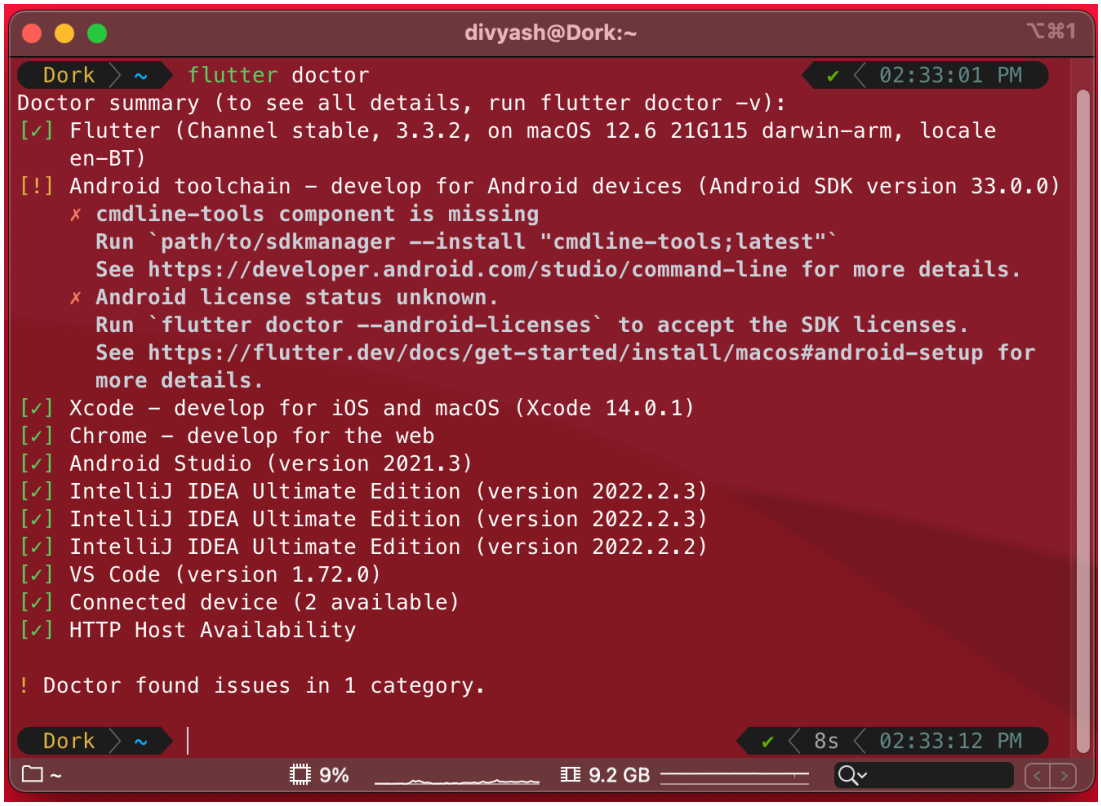
2. Unzip the bundle in a directly and copy its path (/Users/divyash/Developer/flutter) in my case

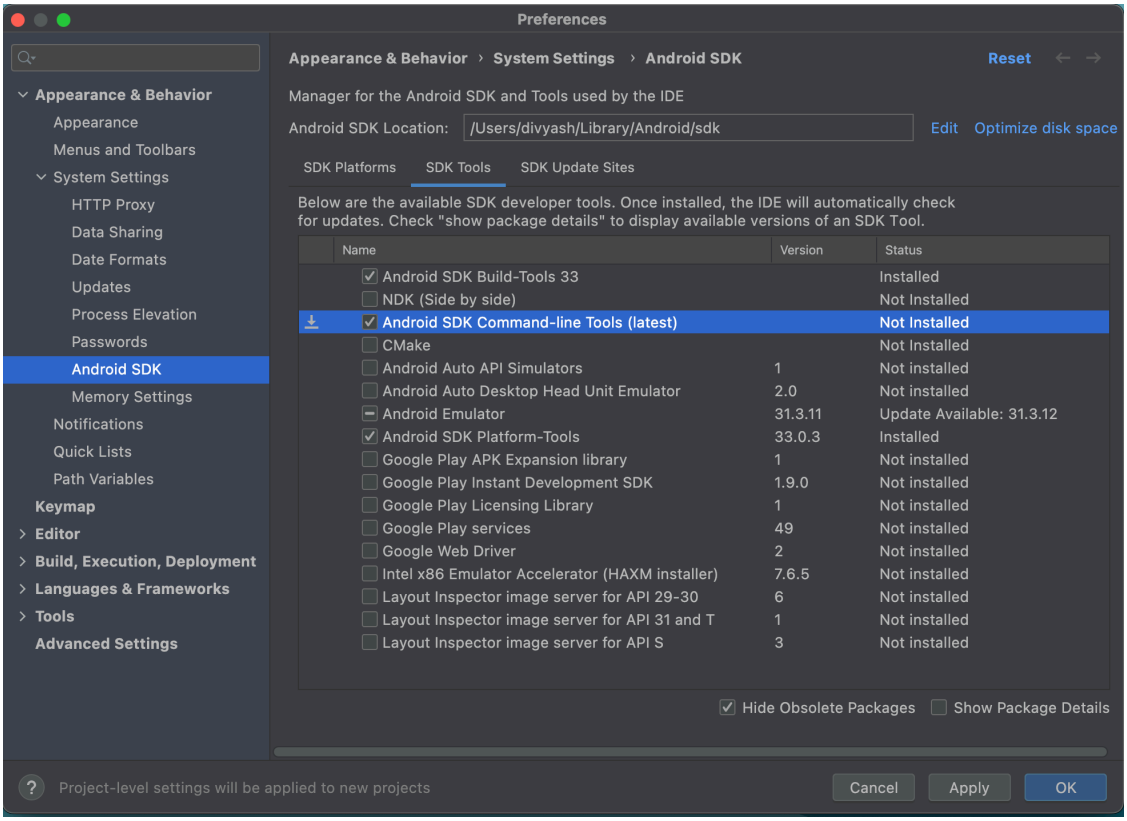


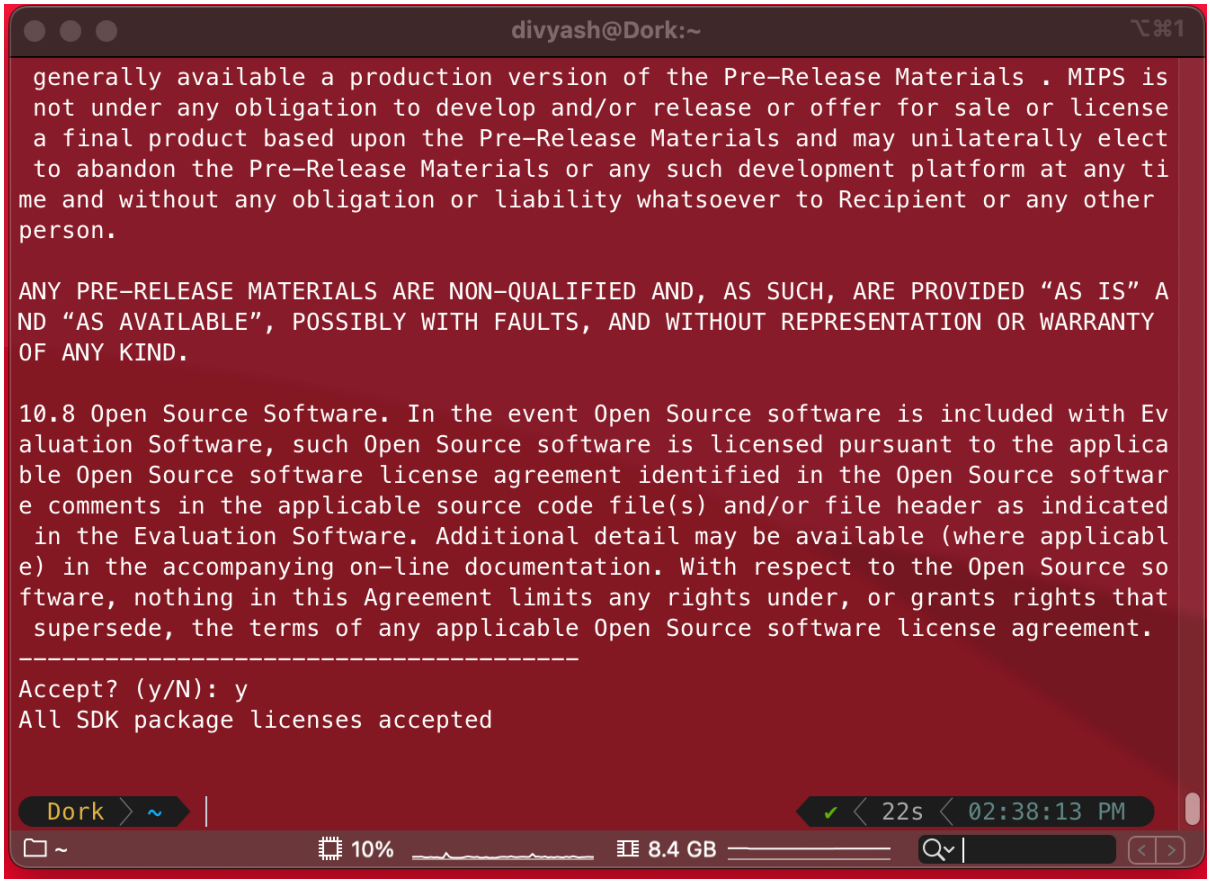
3. Add the bin directory location to the path from terminal (/Users/divyash/Developer/flutter/bin)



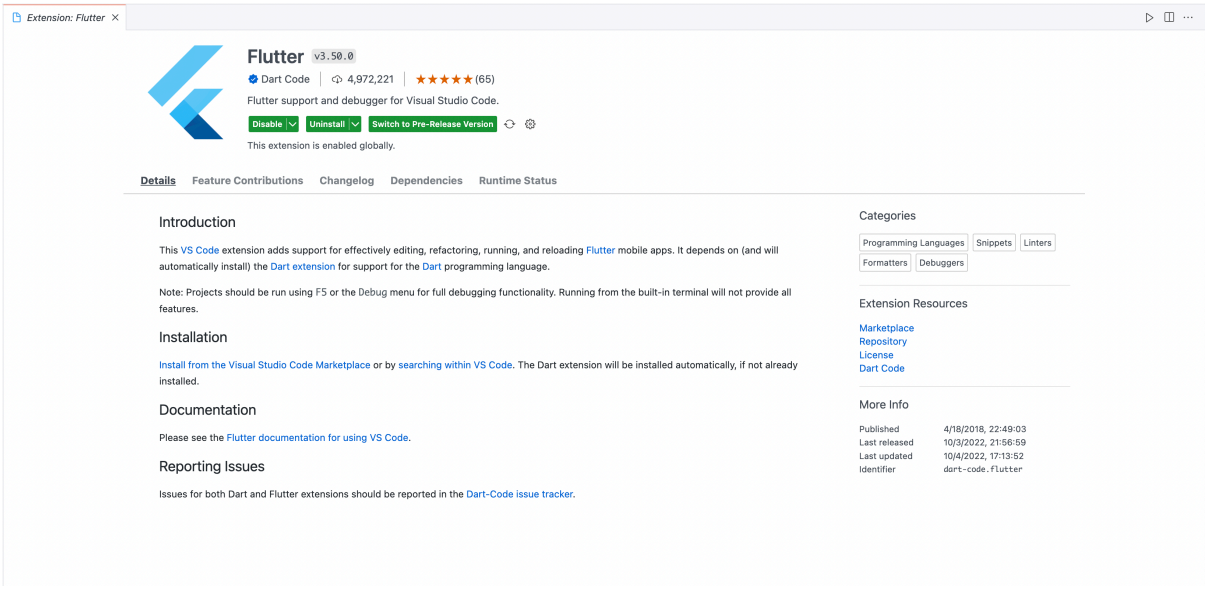
4. Run ‘flutter doctor’ command in terminal and install android command line tools and agree to android licenses as suggested by flutter doctor.


divyash@Dork:~
Dork ~ flutter doctor
02:33:01 PM
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.3.2, on macOS 12.6 21G115 darwin-arm, locale en-BT)
[!] Android toolchain - develop for Android devices (Android SDK version 33.0.0)
✗ cmdline-tools component is missing
Run `path/to/sdkmanager --install "cmdline-tools;latest"`
See https://developer.android.com/studio/command-line for more details.
✗ Android license status unknown.
Run `flutter doctor --android-licenses` to accept the SDK licenses.
See https://flutter.dev/docs/get-started/install/macos#android-setup for more details.
[✓] Xcode - develop for iOS and macOS (Xcode 14.0.1)
[✓] Chrome - develop for the web
[✓] Android Studio (version 2021.3)
[✓] IntelliJ IDEA Ultimate Edition (version 2022.2.3)
[✓] IntelliJ IDEA Ultimate Edition (version 2022.2.3)
[✓] IntelliJ IDEA Ultimate Edition (version 2022.2.2)
[✓] VS Code (version 1.72.0)
[✓] Connected device (2 available)
[✓] HTTP Host Availability
! Doctor found issues in 1 category.
Dork ~
8s 02:33:12 PM
~ 9% 9.2 GB


Preferences
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
HTTP Proxy
Data Sharing
Date Formats
Updates
Process Elevation
Passwords
Android SDK
Memory Settings
Notifications
Quick Lists
Path Variables
Keymap
Editor
Build, Execution, Deployment
Languages & Frameworks
Tools
Advanced Settings
Appearance & Behavior › System Settings › Android SDK
Reset
Manager for the Android SDK and Tools used by the IDE
Android SDK Location: /Users/divyash/Library/Android/sdk
Edit
Optimize disk space
SDK Platforms
SDK Tools
SDK Update Sites
Below are the available SDK developer tools. Once installed, the IDE will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.
Name
Version
Status
Android SDK Build-Tools 33 Installed
NDK (Side by side) Not Installed
Android SDK Command-line Tools (latest) Not Installed
CMake Not Installed
Android Auto API Simulators 1 Not installed
Android Auto Desktop Head Unit Emulator 2.0 Not installed
Android Emulator 31.3.11 Update Available: 31.3.12
Android SDK Platform-Tools 33.0.3 Installed
Google Play APK Expansion library 1 Not installed
Google Play Instant Development SDK 1.9.0 Not installed
Google Play Licensing Library 1 Not installed
Google Play services 49 Not installed
Google Web Driver 2 Not installed
Intel x86 Emulator Accelerator (HAXM installer) 7.6.5 Not installed
Layout Inspector image server for API 29-30 6 Not installed
Layout Inspector image server for API 31 and T 1 Not installed
Layout Inspector image server for API S 3 Not installed
Hide Obsolete Packages
Show Package Details
Project-level settings will be applied to new projects
Cancel
Apply
OK



5. Install Flutter debug tool extension in vs code



6. Run 'flutter create first_app' in terminal
7. cd into first_app dir
8. Enter code . to open in VS Code
9. Start emulator of your choice
10. Debug the app to run in the started emulator

first_app

Program Code

main.dart

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

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app,
try
        // changing the primarySwatch below to Colors.green and then invoke
        // "hot reload" (press "r" in the console where you ran "flutter
run",
        // or simply save your changes to "hot reload" in a Flutter IDE).
        // Notice that the counter didn't reset back to zero; the
application
        // is not restarted.
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
      debugShowCheckedModeBanner: false,
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});

  // This widget is the home page of your application. It is stateful,
meaning
```

```
  // that it has a State object (defined below) that contains fields that
affect
  // how it looks.

  // This class is the configuration for the state. It holds the values (in
this
  // case the title) provided by the parent (in this case the App widget)
and
  // used by the build method of the State. Fields in a Widget subclass are
  // always marked "final".

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  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++;
    });
  }

  @override
  Widget build(BuildContext context) {
    // This method is rerun every time setState is called, for instance as
done
    // by the _incrementCounter method above.
    //
    // The Flutter framework has been optimized to make rerunning build
methods
    // fast, so that you can just rebuild anything that needs updating
rather
    // than having to individually change instances of widgets.
    return Scaffold(
```

```
      appBar: AppBar(
        // Here we take the value from the MyHomePage object that was
created by
        // the App.build method, and use it to set our appbar title.
        title: Text(widget.title),
      ),
      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).
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            const Text(
              'You have pushed the button this many times:',
            ),
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headline4,
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
```

```
        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ), // This trailing comma makes auto-formatting nicer for build
methods.
    );
  }
}
```

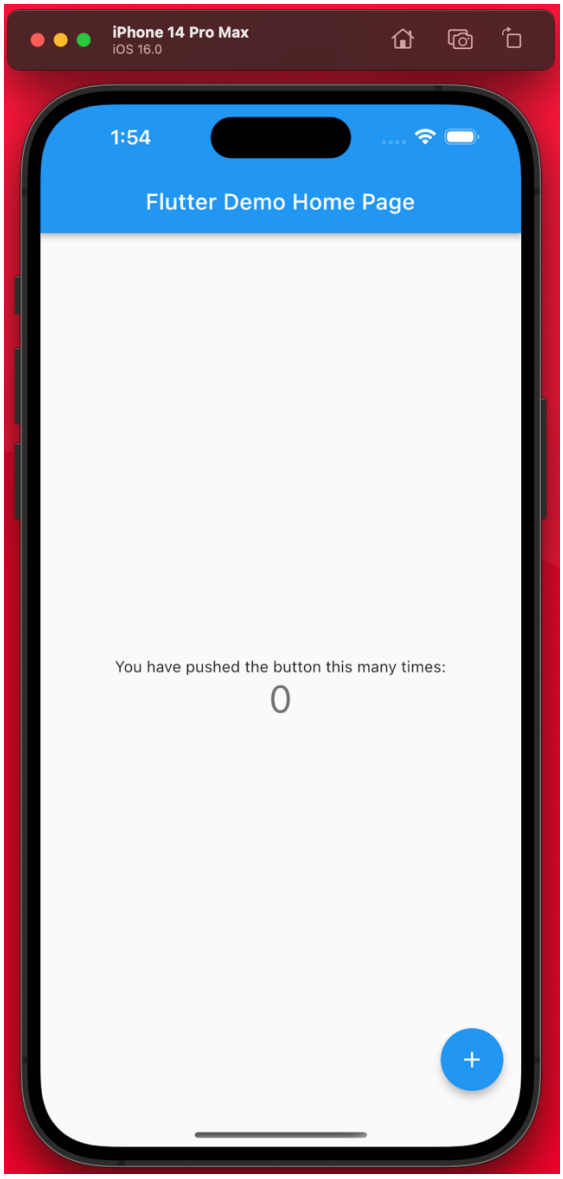
pubspec.yaml

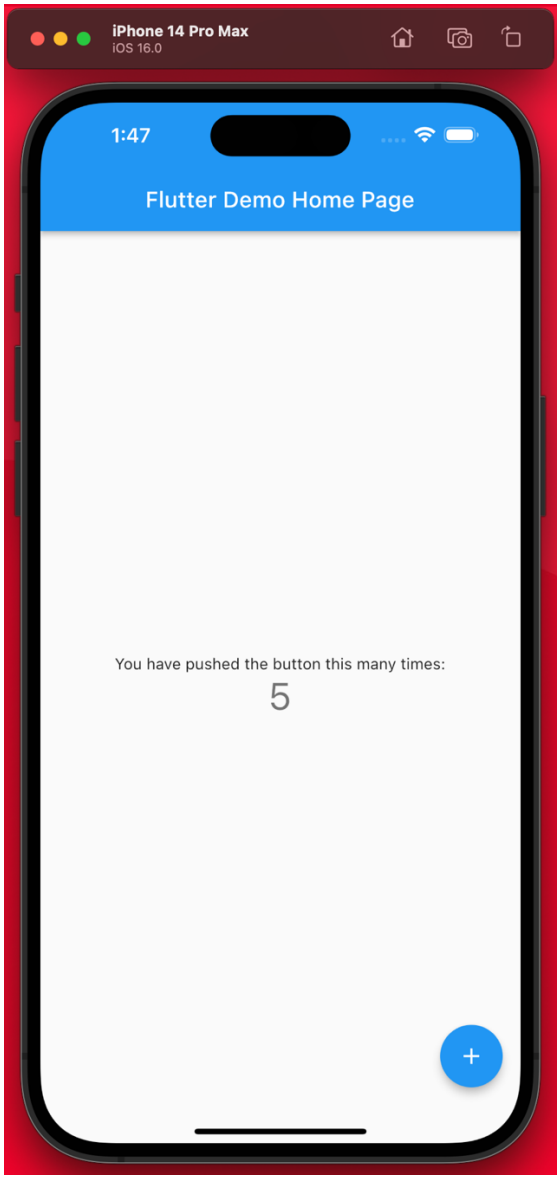
```
name: first_app
description: A new Flutter project.

# The following line prevents the package from being accidentally published
to
# pub.dev using `flutter pub publish`. This is preferred for private
packages.
publish_to: 'none' # Remove this line if you wish to publish to pub.dev

# The following defines the version and build number for your application.
# A version number is three numbers separated by dots, like 1.2.43
# followed by an optional build number separated by a +.
# Both the version and the builder number may be overridden in flutter
# build by specifying --build-name and --build-number, respectively.
# In Android, build-name is used as versionName while build-number used as
versionCode.
# Read more about Android versioning at
https://developer.android.com/studio/publish/versioning
# In iOS, build-name is used as CFBundleShortVersionString while build-
number is used as CFBundleVersion.
# Read more about iOS versioning at
#
https://developer.apple.com/library/archive/documentation/General/Reference
/InfoPlistKeyReference/Articles/CoreFoundationKeys.html
# In Windows, build-name is used as the major, minor, and patch parts
# of the product and file versions while build-number is used as the build
suffix.
version: 1.0.0+1

environment:
  sdk: '>=2.18.1 <3.0.0'

# Dependencies specify other packages that your package needs in order to
work.
# To automatically upgrade your package dependencies to the latest versions
# consider running `flutter pub upgrade --major-versions`. Alternatively,
```

```
# dependencies can be manually updated by changing the version numbers
below to
# the latest version available on pub.dev. To see which dependencies have
newer
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
    sdk: flutter


  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.2

dev_dependencies:
  flutter_test:
    sdk: flutter

  # The "flutter_lints" package below contains a set of recommended lints
to
  # encourage good coding practices. The lint set provided by the package
is
  # activated in the `analysis_options.yaml` file located at the root of
your
  # package. See that file for information about deactivating specific lint
  # rules and activating additional ones.
  flutter_lints: ^2.0.0

# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter packages.
flutter:

  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the icons in
  # the material Icons class.
  uses-material-design: true

  # To add assets to your application, add an assets section, like this:
  # assets:
  #   - images/a_dot_burr.jpeg
  #   - images/a_dot_ham.jpeg

  # An image asset can refer to one or more resolution-specific "variants",
see
```

```
# https://flutter.dev/assets-and-images/#resolution-aware

# For details regarding adding assets from package dependencies, see
# https://flutter.dev/assets-and-images/#from-packages

# To add custom fonts to your application, add a fonts section here,
# in this "flutter" section. Each entry in this list should have a
# "family" key with the font family name, and a "fonts" key with a
# list giving the asset and other descriptors for the font. For
# example:
# fonts:
#   - family: Schyler
#     fonts:
#       - asset: fonts/Schyler-Regular.ttf
#       - asset: fonts/Schyler-Italic.ttf
#         style: italic
#   - family: Trajan Pro
#     fonts:
#       - asset: fonts/TrajanPro.ttf
#       - asset: fonts/TrajanPro_Bold.ttf
#         weight: 700
#
# For details regarding fonts from package dependencies,
# see https://flutter.dev/custom-fonts/#from-packages
```

Output





Conclusion

Flutter is Google's portable UI toolkit for creating attractive, natively built apps for mobile, web, and desktop from a single codebase, as I learnt from the above practical. Flutter is free and open source, and it works with existing code. It is utilized by developers and organizations all around the world. Furthermore, I discovered that flutter has accelerated time-to-market. The Flutter programming framework is faster than its competitors. When compared to an app designed separately for Android and iOS, we should anticipate a Flutter app to take at least twice as short to develop. Through this practical, I learned how to create flutter projects and configure the Flutter file with the SDK that exists in IDEs. Flutter seems much easier than android development in android studio.

References

Android. (n.d.). *Android Studio Docs*. Retrieved 2022, from Developers Android: https://developer.android.com/docs

Thomas, G. (2021, July 09). *What is Flutter and Why You Should Learn it in 2020*. Retrieved Oct 11, 2022, from Freecodecamp: https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/

Flutter. (n.d.). *macOS install Flutter*. Retrieved Oct 11, 2022, from Flutter: https://docs.flutter.dev/get-started/install/macos