ANURAG ENGINEERING COLLEGE

(An Autonomous Institution)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

Vision

- To generate Competent Professionals to become part of the Industry and Research Organizations at the National and International levels.

Mission

- To train the students to have in-depth knowledge of the subjects in the field of Computer Science and Engineering.

- To train the students with leadership qualities, team work skills, commitment and ethics thereby making them develop confidence for R & D activities and for placement in multinational and national.

PROGRAM EDUCATIONAL OBJECTIVES

PEO I : Excel in professional career and/or higher education by acquiring knowledge in mathematical, computing and engineering principles

PEO II : Be able to analyze the requirements of the software, understand the technical specifications, design and provide novel engineering solutions and efficient product designs.

PEO III : Adopt to professionalism, ethical attitude, communication skills, team work, lifelong learning in their profession.

PROGRAM SPECIFIC OUTCOMES

PSO 1: **Problem Solving Skills**: Ability to use mathematical abstraction, algorithmic design and appropriate data structures to solve real world problems using different programming paradigms.

PSO 2: **Professional Skills**: Ability to develop computing solutions for problems in multidisciplinary areas by applying software engineering principles.

PSO 3: **Successful Career and Entrepreneurship Skills**: Gain knowledge in diverse areas of computer science, and management skills for successful career, entrepreneurship and higher studies

PROGRAM OUTCOMES

PO 1:	Gain an ability to apply knowledge of mathematics, science and engineering fundamentals appropriate to the discipline.
PO 2:	Develop the competence to identify, analyze, formulate and solve engineering problems.
PO 3:	Acquire an ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.
PO 4:	Are capable to design and conduct experiments, analyze and interpret data in the field of computer science and engineering.
PO 5:	Gain expertise to use the techniques, skills and modern engineering tools with proficiency in basic area of computer science and engineering.
PO 6:	An ability to analyze the local and global impact of computing on individuals, organizations, and society.
PO 7:	Knowledge of contemporary issues.
PO 8:	Sensitive to engage in activities with conscious social responsibility adhering to ethical values.
PO 9:	An ability to function effectively individually and on teams, including diverse and multidisciplinary, to accomplish a common goal.
PO 10:	An ability to articulate professional ideas clearly and precisely in making written and oral presentations.
PO 11:	Recognition of the need for and an ability to engage in continuing professional development.
PO 12:	An understanding of engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects.

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:
    a) Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.
    b) Laboratory Record updated up to the last session experiments
    c) Proper Dress code and Identity card.
4. Sign in the laboratory login register and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
9. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

ANURAG ENGINEERING COLLEGE

(An Autonomous Institution)

III Year B.Tech. CSE - I Sem

L	T	P	C
0	0	2	1

(CS507PC)UIDESIGN-FLUTTER

Course Objectives:

The objectives of this course are to provide:

- To learn installation of SDK of Flutter, X code and Android Emulator.
- How to Create and Organize Folders and Files, Structuring Widgets.
- Understanding Stateless and Stateful Widgets and Widget Tree
- Learning of Dart basics
- Application of Animation to app.

Unit-I: Introducing Flutter and Getting Started

Introducing Flutter, Defining Widgets and Elements, Understanding Widget Lifecycle Events, The Stateless Widget Lifecycle, The Stateful Widget Lifecycle, Understanding the Widget Tree and the Element Tree, Stateless Widget and Element Trees, Stateful Widget and Element Trees, Installing the Flutter SDK, Installing on mac OS, System Requirements, Get the Flutter SDK, Check for Dependencies, iOS Setup: Install Xcode, Android Setup: Install Android Studio, Set Up the Android Emulator, Installing on Windows, System Requirements, Get the Flutter SDK, Check for Dependencies, Install Android Studio, Set Up the Android Emulator, Installing on Linux, System Requirements, Get the Flutter SDK, Check for Dependencies, Install Android Studio, Set Up the Android Emulator, Configuring the Android Studio Editor.

Unit-II: Creating a Hello World App

Setting Up the Project, Using Hot Reload, Using Themes to Style Your App, Using a Global App Theme, Using a Theme for Part of an App, Understanding Stateless and Stateful Widgets, Using External Packages, Searching for Packages, Using Packages

Unit-III: Learning Dart Basics

Use of Dart, Commenting Code, Running the main() Entry Point, Referencing Variables, Declaring Variables, Numbers, Strings, Booleans, Lists, Maps, Runes, Using Operators, Using Flow Statements, if and else, ternary operator, for Loops, while and do-while, while and break, continue, switch and case, Using Functions, Import Packages, Using Classes, Class Inheritance, Class Mixins, Implementing Asynchronous Programming.

Unit-IV: Creating a Starter Project Template

Creating and Organizing Folders and Files, Structuring Widgets.

Understanding the WidgetTree

Introduction to Widgets, Building the Full WidgetTree, Building a Shallow Widget Tree, Refactoring with a Constant, Refactoring with a Method, Refactoring with a Widget Class.

Unit-V: Using Common Widgets

Using Basic Widgets, Safe Area, Container, Text, Rich Text, Column, Row, Column and Row Nesting, Buttons, Floating Action Button, Flat Button, Raised Button, Icon Button, Popup Menu Button, Button Bar, Using Images and Icons, Asset Bundle, Image, Icon, Using Decorators, Using the Form Widget to Validate Text Fields, Checking Orientation.

Adding Animation to an App

Using Animated Container, Using Animated Cross Fade, Using Animated Opacity, Using Animation Controlle, Using Staggered Animations,

Text Books:

1. Marco L. Napoli, Beginning Flutter: A Hands-on Guide to App Development, 1st edition, Wrox publisher.

Reference Books:

1. Flutter for Beginners: An introductory guide to building cross-platform mobile applications withFlutterandDart2, Packt Publishing Limited.
2. Rap Payne, Beginning App Development with Flutter: Create Cross-Platform Mobile Apps, 1stedition, Apress.
3. Frank Zammetti, Practical Flutter: Improve your Mobile Development with Google's LatestOpen-SourceSDK,1stedition, Apress.

Course Outcomes:

Upon the successful completion of this course, the student will be able to:

1. Knowledge on installation of various softwares.
2. Understanding of various Widgets
3. Application of Animation to Apps
4. Implements Flutter Widgets and Layouts
5. Responsive UI Design and with Navigation in Flutter

	PO-1	PO-2	PO-3	PO-4	PO-5	PO-6	PO-7	PO-8	PO-9	PO-10	PO-11	PO-12	PSO-1	PSO-2
CO-1	H	L		M		L							M	M
CO-2	M	M	L	H		M							M	M
CO-3	M	M	M	M	M	M							M	M
CO-4	M	M	M	M	H	L							M	H
CO-5	M	M	L	L	M								L	H

INDEX

INTRODUCTION

In general, developing a mobile application is a complex and challenging task. There are many frameworks available to develop a mobile application. Android provides a native framework based on Java language and iOS provides a native framework based on Objective-C / Shift language.

However, to develop an application supporting both the OSs, we need to code in two different languages using two different frameworks. To help overcome this complexity, there exists mobile frameworks supporting both OS. These frameworks range from simple HTML based hybrid mobile application framework (which uses HTML for User Interface and JavaScript for application logic) to complex language specific framework (which do the heavy lifting of converting code to native code). Irrespective of their simplicity or complexity, these frameworks always have many disadvantages, one of the main drawback being their slow performance.

In this scenario, Flutter – a simple and high-performance framework based on Dart language, provides high performance by rendering the UI directly in the operating system's canvas rather than through native framework.

Flutter also offers many ready to use widgets (UI) to create a modern application. These widgets are optimized for mobile environment and designing the application using widgets is as simple as designing HTML.

To be specific, Flutter application is itself a widget. Flutter widgets also supports animations and gestures. The application logic is based on reactive programming. Widget may optionally have a state. By changing the state of the widget, Flutter will automatically (reactive programming) compare the widget's state (old and new) and render the widget with only the necessary changes instead of re-rendering the whole widget.

Features of Flutter

Flutter framework offers the following features to developers:

- Modern and reactive framework.
- Uses Dart programming language and it is very easy to learn.
- Fast development.
- Beautiful and fluid user interfaces.
- Huge widget catalog.
- Runs same UI for multiple platforms.
- High performance application.

Advantages of Flutter

Flutter comes with beautiful and customizable widgets for high performance and outstanding mobile application. It fulfils all the custom needs and requirements. Besides these, Flutter offers many more advantages as mentioned below:

- Dart has a large repository of software packages which lets you to extend the capabilities of your application.
- Developers need to write just a single code base for both applications (both Android and iOS platforms). Flutter may to be extended to other platform as well in the future.
- Flutter needs lesser testing. Because of its single code base, it is sufficient if we write automated tests once for both the platforms.
- Flutter's simplicity makes it a good candidate for fast development. Its customization capability and extendibility make it even more powerful.
- With Flutter, developers has full control over the widgets and its layout.
- Flutter offers great developer tools, with amazing hot reload.

Disadvantages of Flutter

Despite its many advantages, flutter has the following drawbacks in it:

- Since it is coded in Dart language, a developer needs to learn new language (though it is easy to learn).

- Modern framework tries to separate logic and UI as much as possible but, in Flutter,user interface and logic is intermixed. We can overcome this using smart coding and using high level module to separate user interface and logic.

- Flutter is yet another framework to create mobile application. Developers are having a hard time in choosing the right development tools in hugely populated segment.

Week-1

Installation of Android studio & Flutter.

Windows: Download and install Android Studio

Download Android Studio

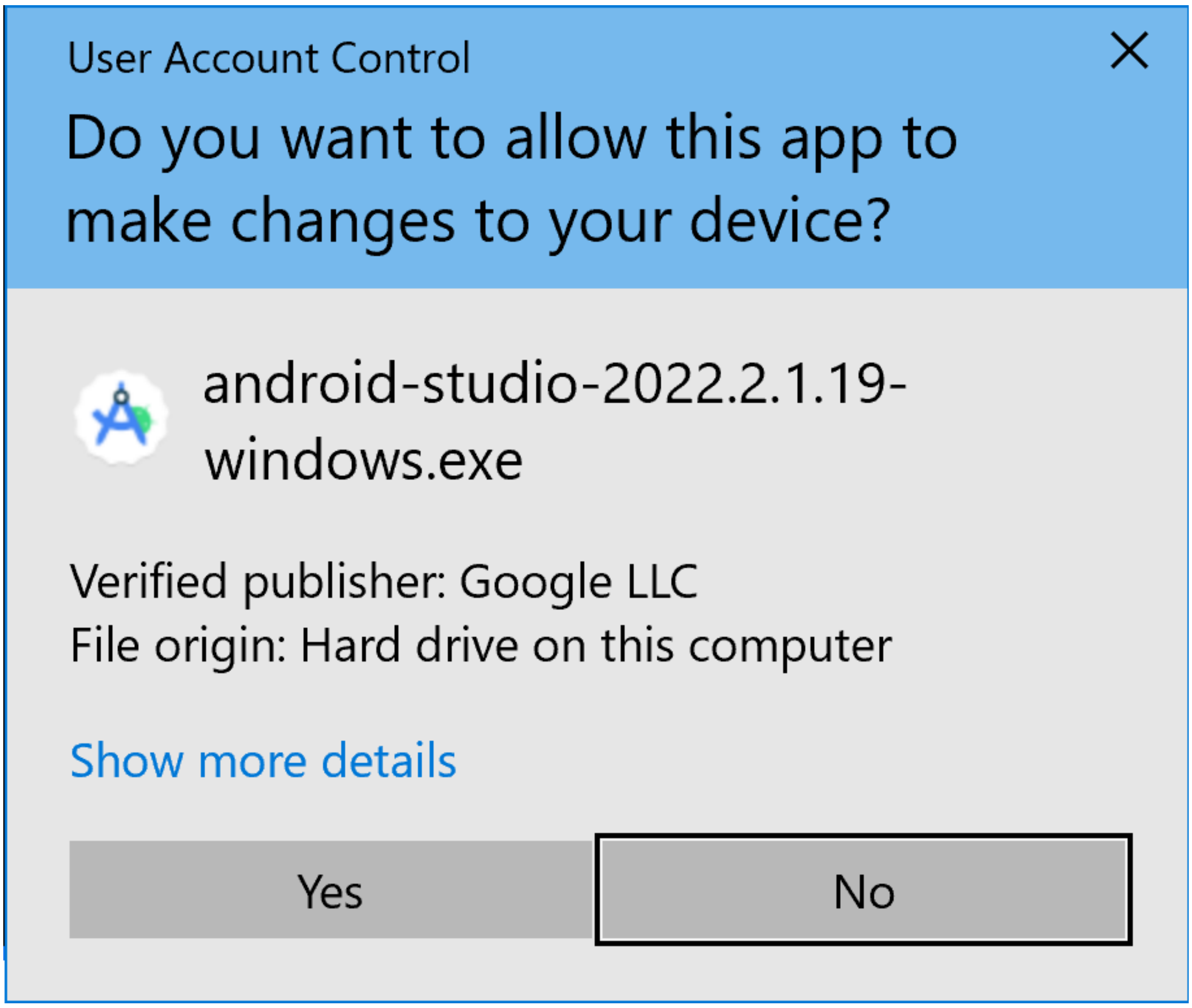
1. Open any web browser and navigate to the Android Studio download page.

This is the Android Developers website, where you can download Android Studio. This page automatically detects your operating system.

2. Click **Download Android Studio**. The **Terms and Conditions** page with the Android Studio **License Agreement** opens.
3. Read the **License Agreement**.
4. At the bottom of the page, if you agree with the terms and conditions, select the **I have read and agree with the above terms and conditions** checkbox.
5. Click **Download Android Studio** to start the download.
6. When prompted, save the file to a location where you can easily locate it, such as the Downloads folder.
7. Wait for the download to complete. This may take a while and may be a good moment to enjoy some tea!

Install Android Studio on Windows

1. Open the folder where you downloaded and saved the Android Studio installation file.
2. Double-click the downloaded file.
3. If you see a **User Account Control** dialog about allowing the installation to make changes to your computer, click **Yes** to confirm the installation.



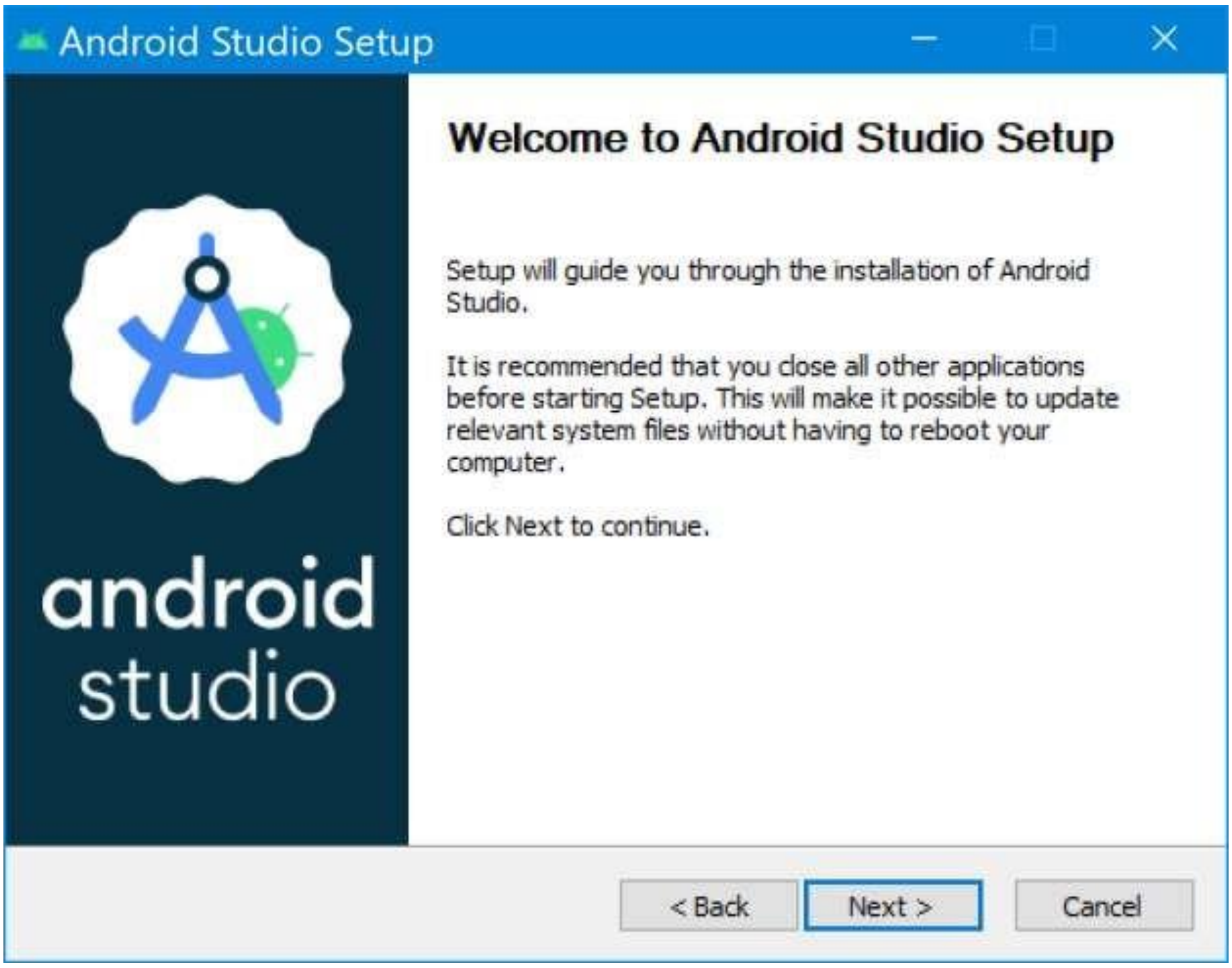
User Account Control

Do you want to allow this app to make changes to your device?

android-studio-2022.2.1.19-windows.exe

Verified publisher: Google LLC
File origin: Hard drive on this computer

The **Welcome to Android Studio Setup** dialog displays.



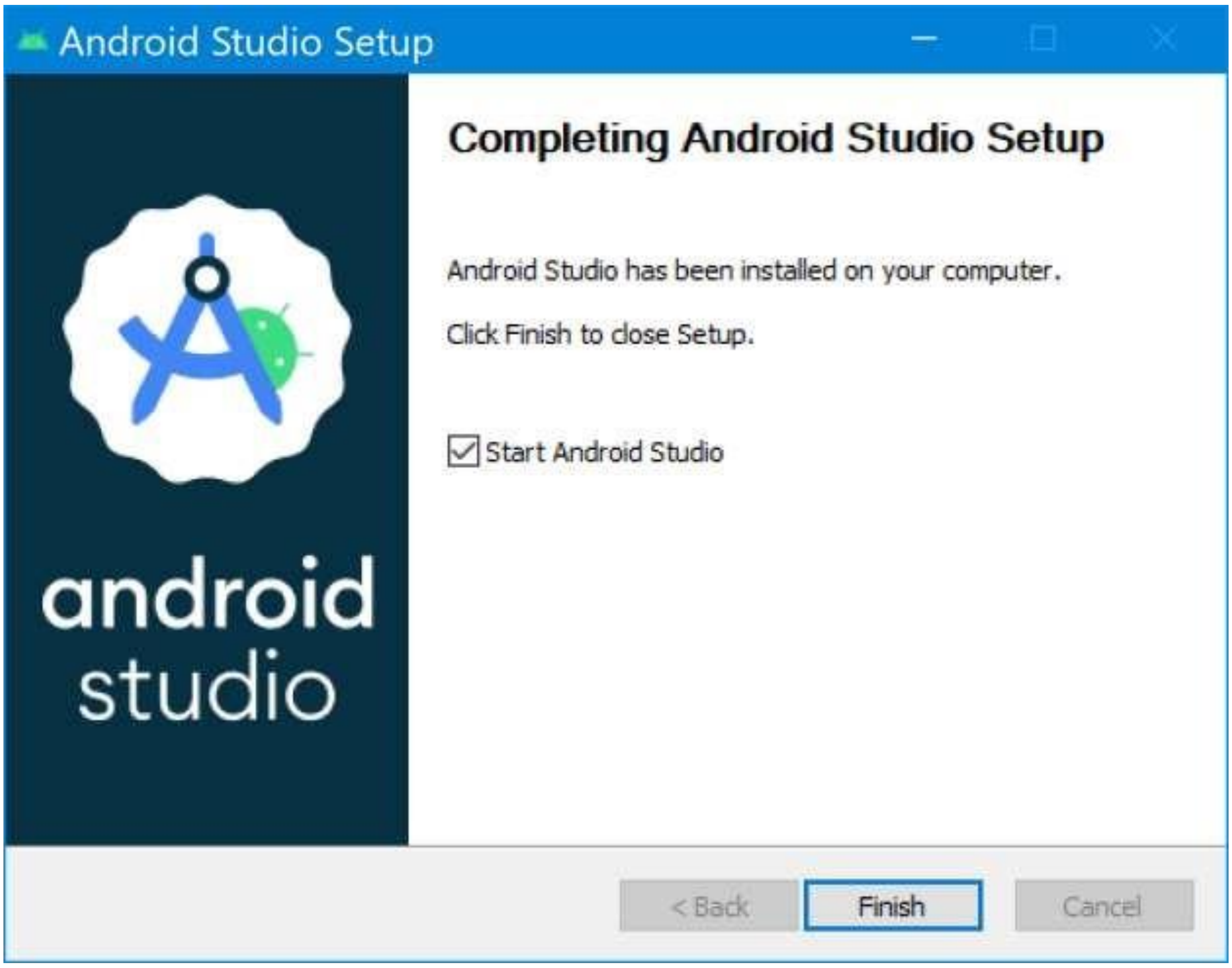
Welcome to Android Studio Setup

Setup will guide you through the installation of Android Studio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

4. Click **Next** to start the installation.
5. Accept the default installation settings for all steps.
6. Click **Finish** when the installation is done to launch Android Studio.



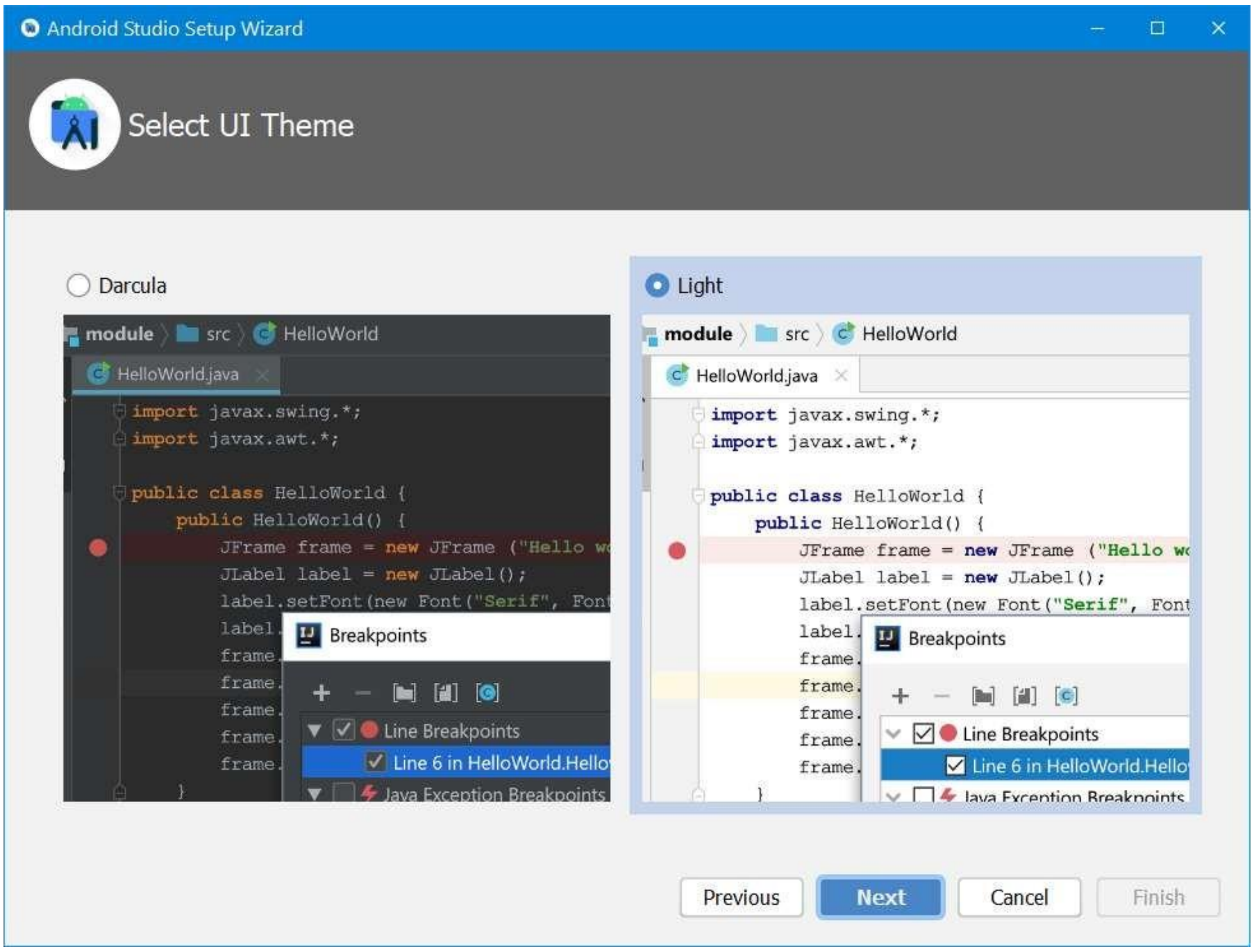
Completing Android Studio Setup

Android Studio has been installed on your computer.

Click Finish to close Setup.

☑ Start Android Studio

7. Choose your preference of light or dark theme when Android Studio first launches. Screenshots in this course use the light theme, but choose whichever one you prefer.

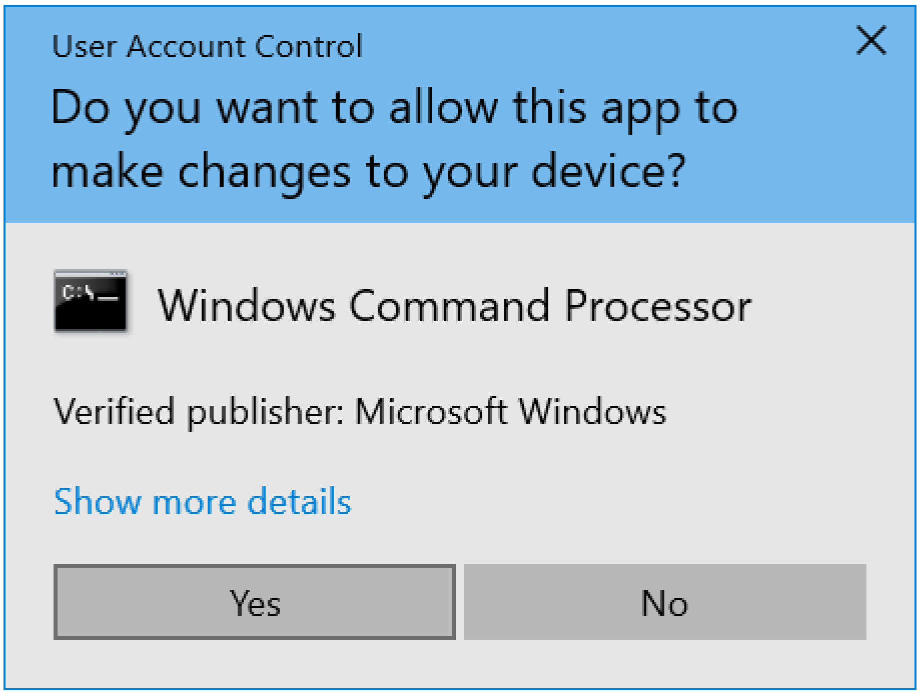

Android Studio Setup Wizard

Select UI Theme

Darcula [✓ Light]

```
import javax.swing.*;
import javax.awt.*;

public class HelloWorld {
    public HelloWorld() {
        JFrame frame = new JFrame ("Hello wo
        JLabel label = new JLabel();
        label.setFont(new Font("Serif", Font
```

8. During the installation, the setup wizard downloads and installs additional components and tools needed for Android app development. This may take some time depending on your internet speed. During this time, you may see a **User Account Control** dialog for **Windows Command Processor**. Click **Yes** to accept the dialog.

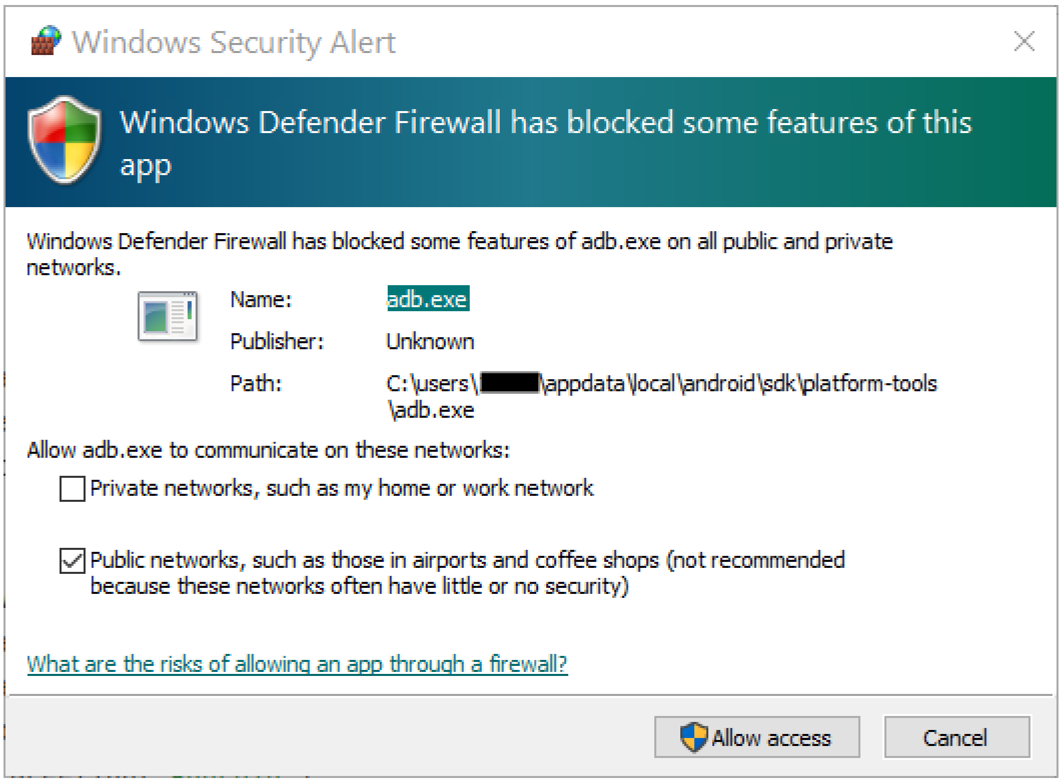

User Account Control

Do you want to allow this app to make changes to your device?

Windows Command Processor

Verified publisher: Microsoft Windows

9. You may also receive a **Windows Security Alert** about adb.exe. Click **Allow Access,** if needed, to continue the installation.



Windows Security Alert

Windows Defender Firewall has blocked some features of this app

Windows Defender Firewall has blocked some features of adb.exe on all public and private networks.

Name: adb.exe

Publisher: Unknown

Path: C:\users\[redacted]\appdata\local\android\sdk\platform-tools\adb.exe

Allow adb.exe to communicate on these networks:

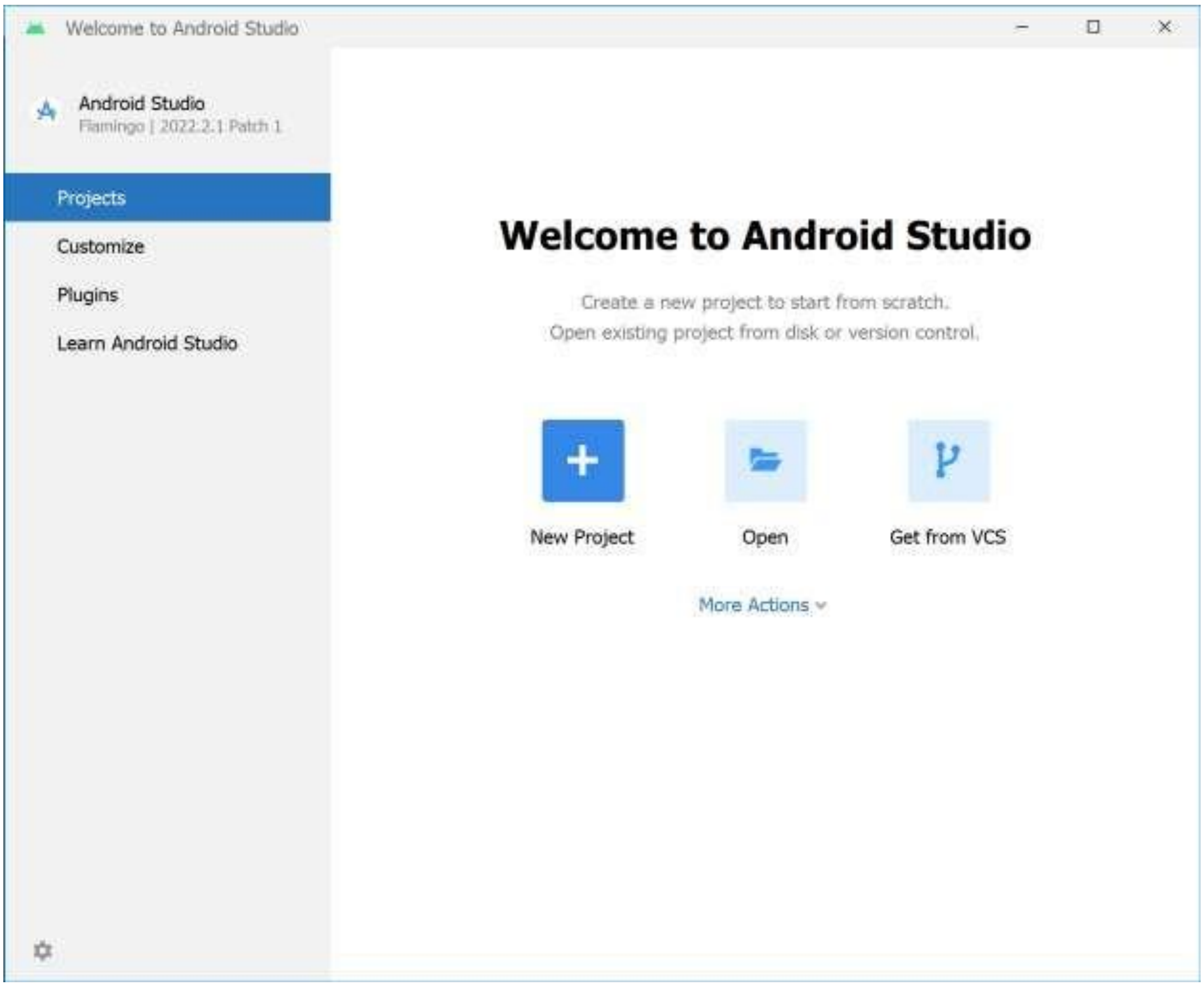
☐ Private networks, such as my home or work network

☑ Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing an app through a firewall?

10. When the download and installation complete, click **Finish**.

The **Welcome to Android Studio** window displays and you're ready to start creating apps!

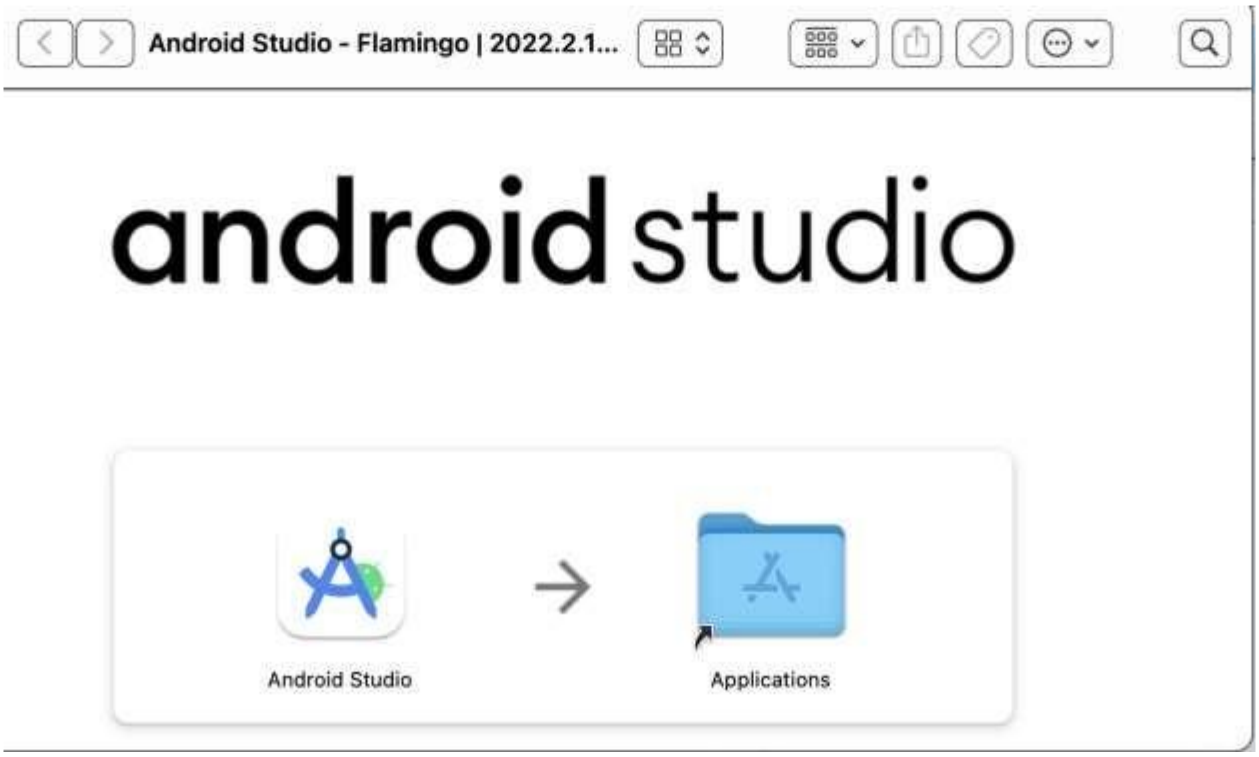

Welcome to Android Studio
Android Studio
Flamingo | 2022.2.1 Patch 1
Projects
Customize
Plugins
Learn Android Studio
Welcome to Android Studio
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
More Actions

Download Android Studio in MacOS

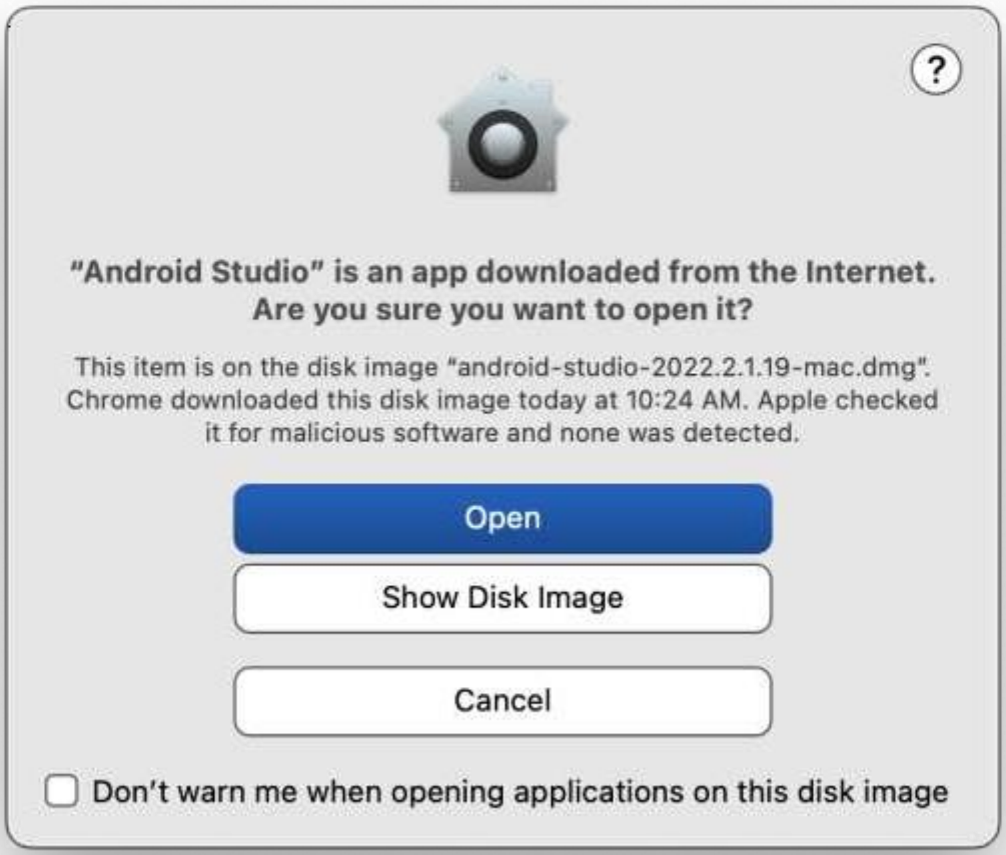
1. Open any web browser and navigate to the Android Studio download page. This is the Android Developers website, where you can download Android Studio. This page automatically detects your operating system.
2. Click **Download Android Studio**. The **Terms and Conditions** page with the Android Studio **License Agreement** opens.
3. Read the **License Agreement**.
4. At the bottom of the page, if you agree with the terms and conditions, select the **I have read and agree with the above terms and conditions** checkbox.
5. Click **Mac with Intel chip** or **Mac with Apple chip** to start the download.
6. When prompted, save the file to a location where you can easily locate it, such as the Downloads folder.
7. Wait for the download to complete. This may take a while and may be a good moment to enjoy some tea!

Install Android Studio on macOS

1. Open the folder where you downloaded and saved the Android Studio installation file.
2. Double-click the downloaded file. The following dialog displays:



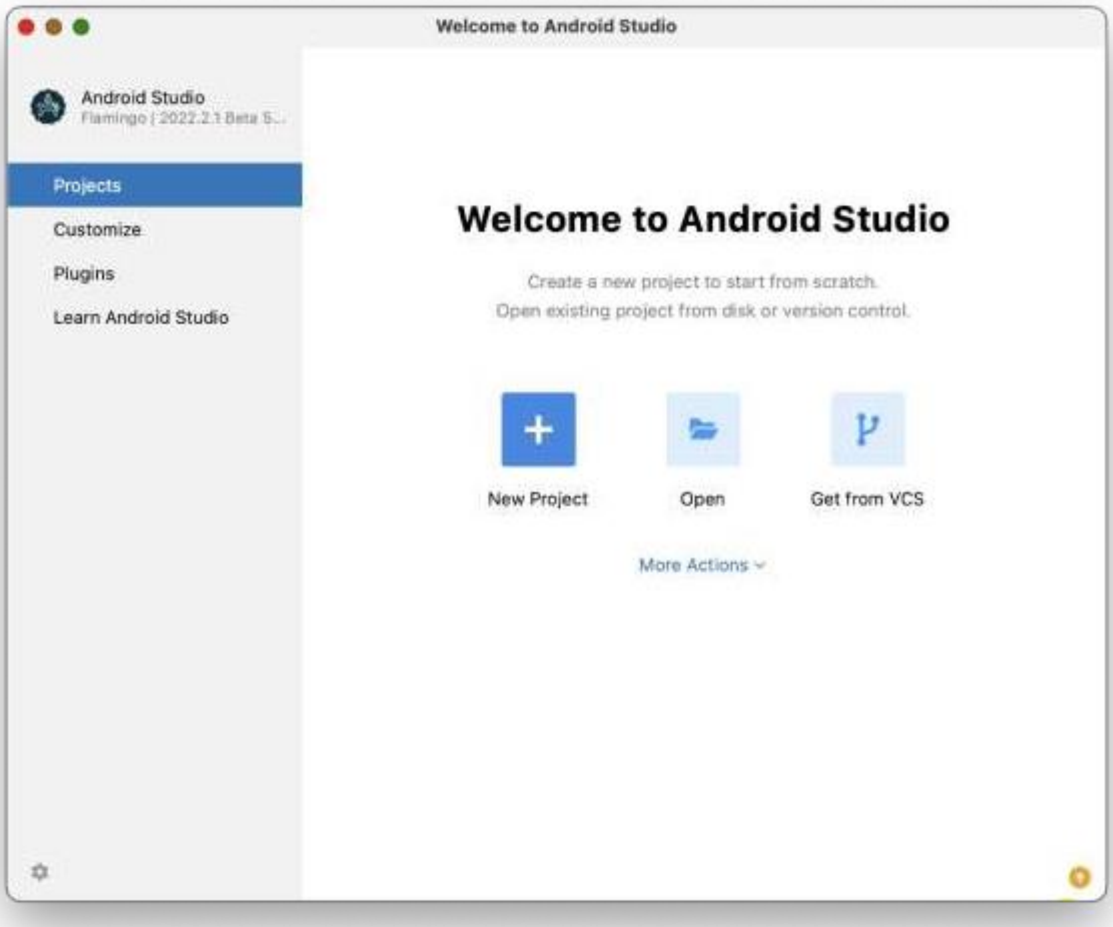
3. Drag the **Android Studio** icon to the Applications folder.
4. In the Applications folder, double-click the **Android Studio** icon to launch the **Android Studio Setup Wizard**.
5. If you see a warning about installing or running a file downloaded from the Internet, accept the installation.


"Android Studio" is an app downloaded from the Internet. Are you sure you want to open it?

This item is on the disk image "android-studio-2022.2.1.19-mac.dmg". Chrome downloaded this disk image today at 10:24 AM. Apple checked it for malicious software and none was detected.

6. Follow the **Android Studio Setup Wizard** and accept the default settings for all steps. During the installation, the setup wizard downloads and installs additional components and tools needed for Android app development. This may take some time, depending on your internet speed. So, you could go refill your tea cup!
7. When the installation completes, Android Studio starts automatically.

The **Welcome to Android Studio** window opens and you're ready to start creating apps!


Welcome to Android Studio
Android Studio
Flamingo | 2022.2.1 Beta 5...
Projects
Customize
Plugins
Learn Android Studio
Welcome to Android Studio
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
More Actions

Flutter Installation

Installation in Windows

In this section, let us see how to install Flutter SDK and its requirement in a windows system.

Step 1: Go to URL, https://flutter.dev/docs/get-started/install/windows and download the latest Flutter SDK. flutter_windows_v3.16.9-stable.zip.

Step 2: Unzip the zip archive in a folder, say C:\flutter\

Step 3: Update the system path to include flutter bin directory.

Step 4: Flutter provides a tool, flutter doctor to check that all the requirement of flutter development is met.

```
C:\flutter doctor
```

Step 5: Running the above command will analyze the system and show its report as shown below:

```
Doctor summary (to see all details, run flutter doctor -v): [√] Flutter (Channel stable,
v1.2.1, on Microsoft Windows [Version 10.0.17134.706], locale en-US) [√] Android
toolchain - develop for Android devices (Android SDK version 28.0.3) [√] Android Studio
(version 3.2) [√] VS Code, 64-bit edition (version 1.29.1) [!] Connected device ! No
devices available ! Doctor found issues in 1 category.
```

The report says that all development tools are available but the device is not connected. We can fix this by connecting an android device through USB or starting an android emulator. **Step 6:** Install the latest Android SDK, if reported by flutter doctor.

Step 7: Install the latest Android Studio, if reported by flutter doctor Step 8: Start an android emulator or connect a real android device to the system.

Step 9: Install Flutter and Dart plugin for Android Studio. It provides start-up template to create new Flutter application, an option to run and debug Flutter application in the Android studio itself, etc.,

- Open Android Studio.
- Click File > Settings > Plugins.
- Select the Flutter plugin and click Install.
- Click Yes when prompted to install the Dart plugin.
- Restart Android studio.

Installation in MacOS

To install Flutter on MacOS, you will have to follow the following steps:

Step 1: Go to URL, https://flutter.dev/docs/get-started/install/macos and download latest Flutter SDK. As of April 2019, the version is 1.2.1 and the file is flutter_macos_v1.2.1stable.zip.

Step 2: Unzip the zip archive in a folder, say /path/to/flutter

Step 3: Update the system path to include flutter bin directory (in ~/.bashrc file).

```
> export PATH="$PATH:/path/to/flutter/bin"
```

Step 4: Enable the updated path in the current session using below command and then verify it as well.

```
source       ~/.bashrc
source
```

```
$HOME/.bash_profile
echo $PATH
```

Flutter provides a tool, flutter doctor to check that all the requirement of flutter development is met. It is similar to the Windows counterpart.

Step 5: Install latest XCode, if reported by flutter doctor

Step 6: Install latest Android SDK, if reported by flutter doctor

Step 7: Install latest Android Studio, if reported by flutter doctor

Step 8: Start an android emulator or connect a real android device to the system to develop android application.

Step 9: Open iOS simulator or connect a real iPhone device to the system to develop iOS application.

Step 10: Install Flutter and Dart plugin for Android Studio. It provides the startup template to create a new Flutter application, option to run and debug Flutter application in the Android studio itself, etc.,

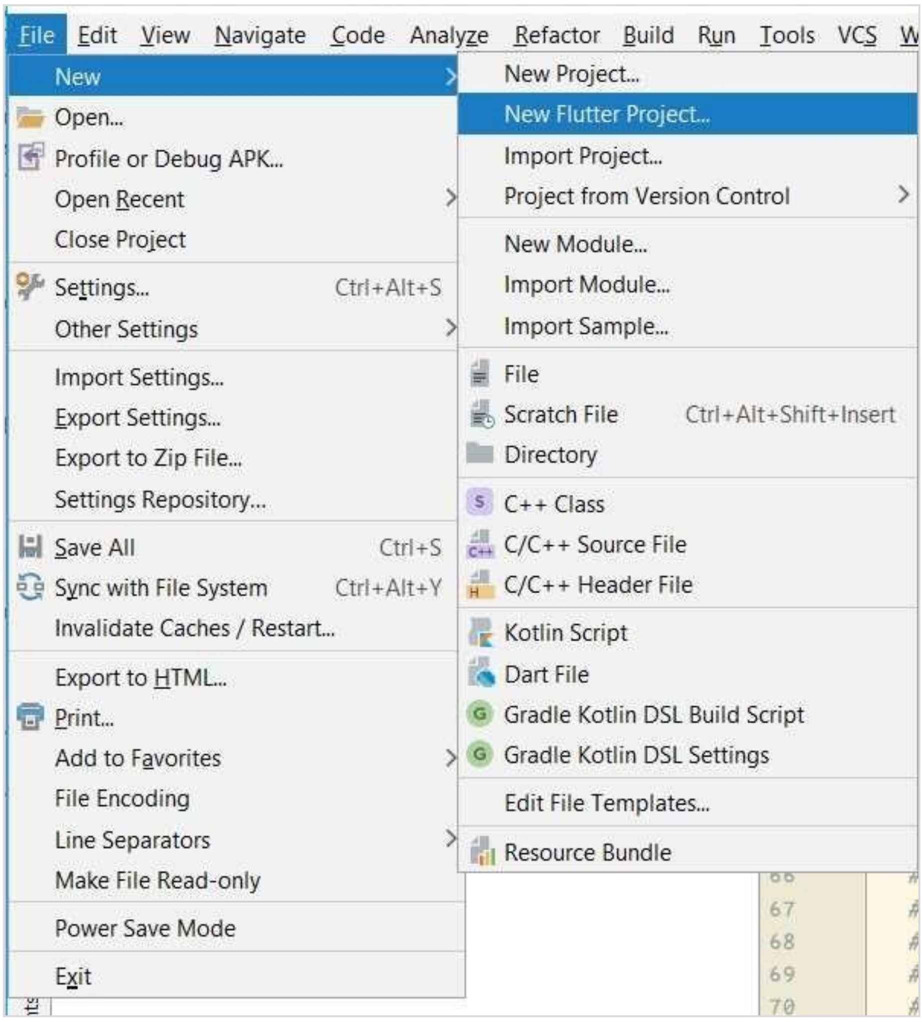
- Open Android Studio.
- Click **Preferences > Plugins**.
- Select the Flutter plugin and click Install.
- Click Yes when prompted to install the Dart plugin.
- Restart Android studio.

Week 2

Create an application using Flutter to print hello world.

Step 1: Open Android Studio

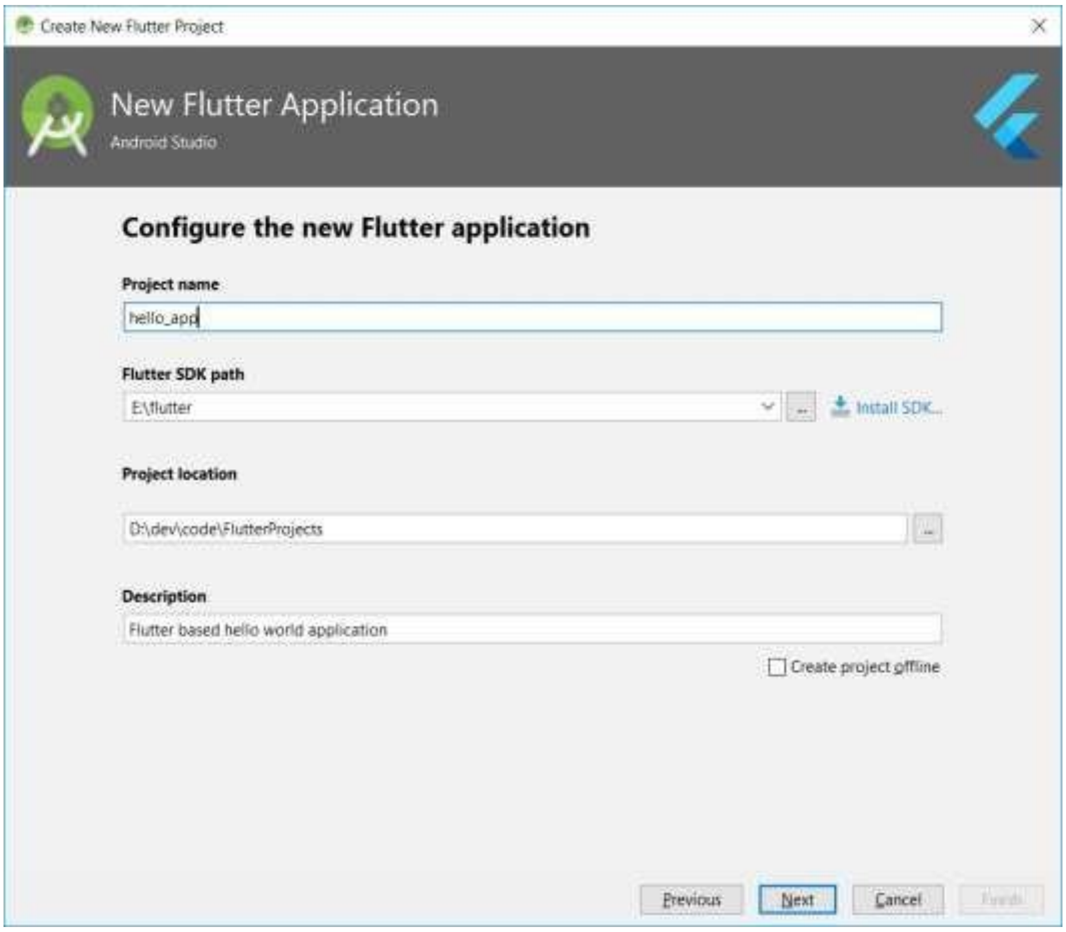
Step 2: Create Flutter Project. For this, click **File -> New -> New Flutter Project**



Step 3: Select Flutter Application. For this, select **Flutter Application** and click **Next**.

Step 4: Configure the application as below and click **Next**.

- Project name: **hello_app**
- Flutter SDK Path: **<path_to_flutter_sdk>**
- Project Location: **<path_to_project_folder>**
- Description: **Flutter based hello world application**


Create New Flutter Project

New Flutter Application

Android Studio

Configure the new Flutter application

Project name: hello_app

Flutter SDK path: E:\flutter

Project location: D:\dev\code\FlutterProjects

Description: Flutter based hello world application

☐ Create project offline

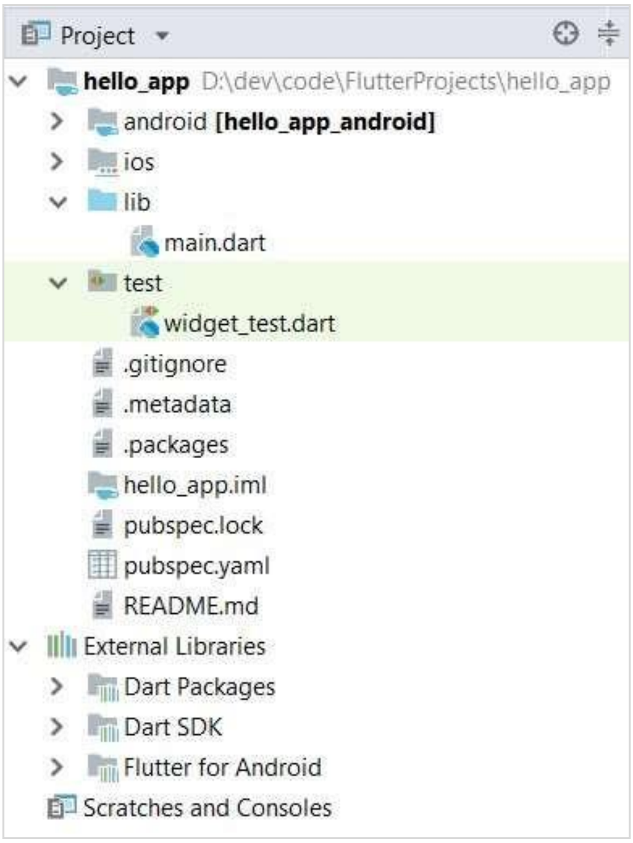
Step 5: Configure Project.

Set the company domain as **flutterapp.tutorialspoint.com** and click **Finish**

Step 6: Enter Company domain.

Android Studio creates a fully working flutter application with minimal functionality. Let us check the structure of the application and then, change the code to do our task.

The structure of the application and its purpose is as follows:



Various components of the structure of the application are explained here:

- **android** - Auto generated source code to create android application
- **ios** - Auto generated source code to create ios application
- **lib** - Main folder containing Dart code written using flutter framework
- **lib/main.dart** - Entry point of the Flutter application
- **test** - Folder containing Dart code to test the flutter application
- **test/widget_test.dart** - Sample code
- **.gitignore** - Git version control file
- **.metadata** - auto generated by the flutter tools
- **.packages** - auto generated to track the flutter packages
- **.iml** - project file used by Android studio
- **pubspec.yaml** - Used by **Pub**, Flutter package manager
- **pubspec.lock** - Auto generated by the Flutter package manager, **Pub**
- **README.md** - Project description file written in Markdown format

Step 7: Replace the dart code in the *lib/main.dart* file with the below code:

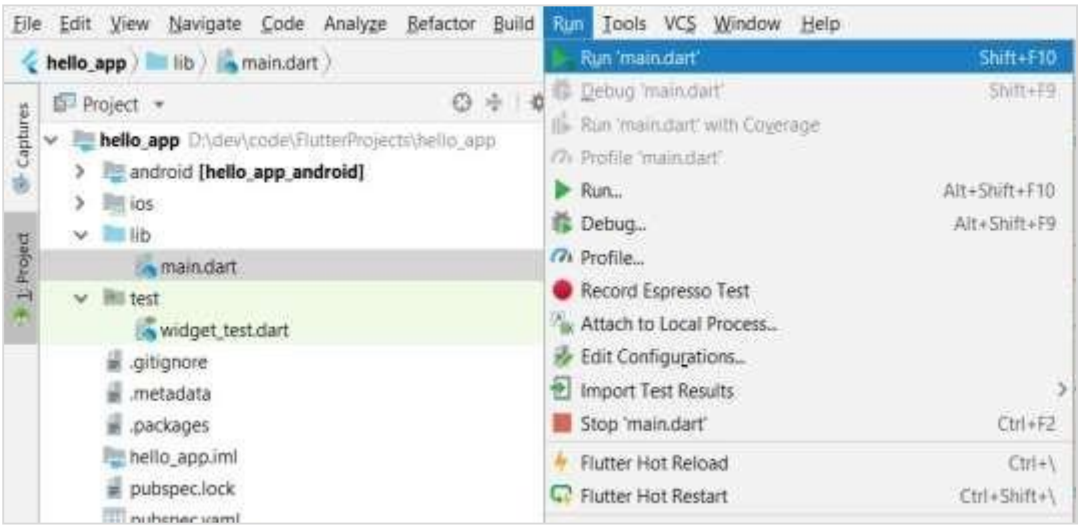
```
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {     return
MaterialApp(
      title: 'Hello World Demo Application',
theme: ThemeData(        primarySwatch:
Colors.blue,
      ),
      home: MyHomePage(title: 'Home page'),
    );
  }
}
class MyHomePage extends StatelessWidget {
  MyHomePage({Key key, this.title}) : super(key: key);

  final String title;

  @override
  Widget build(BuildContext context) {
return Scaffold( appBar: AppBar(
title:Text(this.title), ),
      body: Center( child:
Text(  'Hello World', )
      ),
    );
  }
}
```

Let us understand the dart code line by line.

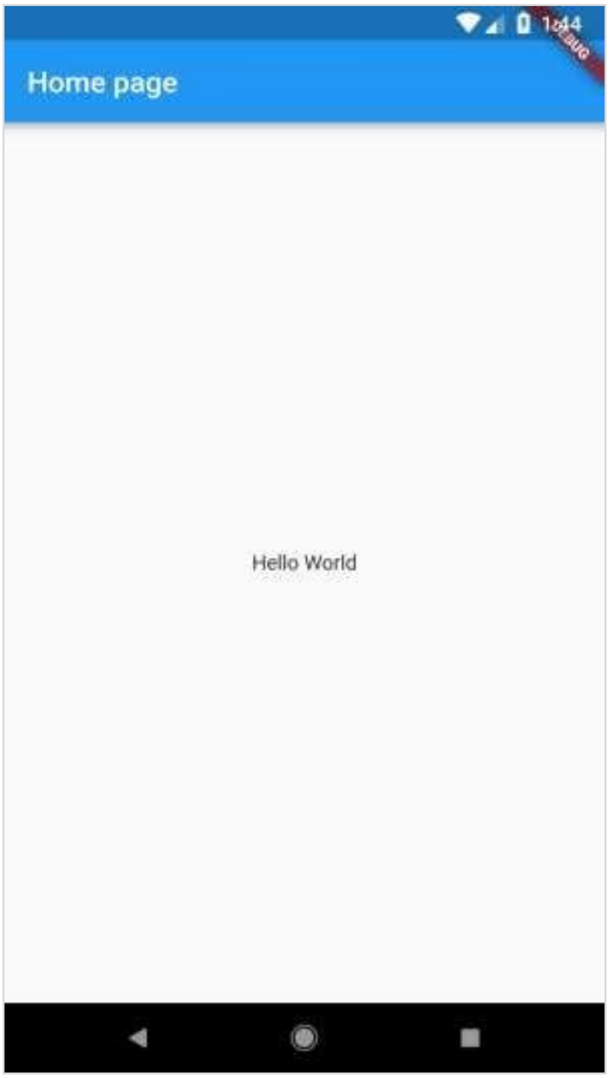
- **Line 1:** imports the flutter package, *material*. The *material* is a flutter package to create user interface according to the Material design guidelines specified by Android.
- **Line 3:** This is the entry point of the Flutter application. Calls *runApp* function and pass it an object of *MyApp* class. The purpose of the *runApp* function is to attach the given widget to the screen.

- **Line 5 - 17:** *Widget* is used to create UI in flutter framework. *StatelessWidget* is a widget, which does not maintain any state of the widget. *MyApp* extends *StatelessWidget* and overrides its *build* method. The purpose of the *build* method is to create a part of the UI of the application. Here, *build* method uses *MaterialApp*, a widget to create the root level UI of the application. It has three properties - *title, theme* and *home*.
    - *title* is the title of the application.
    - *theme* is the theme of the widget. Here, we set *blue* as the overall color of the application using *ThemeData* class and its property, *primarySwatch*.
    - *home* is the inner UI of the application, which we set another widget, *MyHomePage*
- **Line 19 - 38**: *MyHomePage* is same as *MyApp* except it returns *Scaffold* Widget. *Scaffold* is a top level widget next to *MaterialApp* widget used to create UI conforming material design. It has two important properties, *appBar* to show the header of the application and *body* to show the actual content of the application. *AppBar* is another widget to render the header of the application and we have used it in *appBar* property. In *body* property, we have used *Center* widget, which centers it child widget. *Text* is the final and inner most widget to show the text and it is displayed in the center of the screen.

Step 8: Now, run the application using, **Run -> Run main.dart**



Step 9: Finally, the output of the application is as follows:


Home page

Hello World

Week 3

Create an application to implement Decision making and loops using Dart.

The below code snippet checks whether the value of the num variable is positive, zero, or negative, and prints out a message accordingly using an if-else statement. The output will be “5 is a positive number.”

Code:

```
void main() {
  int num = 5;

  if (num > 0) {
    print("$num is a positive number.");
  } else if (num == 0) {
    print("$num is zero.");
  } else {
    print("$num is a negative number.");
  }
}
```

Output: 5 is a positive number.

In this example, we have a variable day set to 5, and we use a switch statement to check its value. Depending on the value of day, we assign the corresponding day name to the variable day Name. If day is not in the range of 1 to 7, we assign the string “Invalid day” to day Name. Finally, we print the value of day Name to the console. The output will be “Today is Friday”.

Code:

```
void main() {
  var day = 5;
  String dayName;

  switch (day) {
    case 1:
      dayName = "Monday";
      break;
    case 2:
      dayName = "Tuesday";
      break;
    case 3:
      dayName = "Wednesday";
      break;
    case 4:
      dayName = "Thursday";
      break;
    case 5:
      dayName = "Friday";
      break;
    case 6:
      dayName = "Saturday";
      break;
    case 7:
      dayName = "Sunday";
      break;
    default:
      dayName = "Invalid day";
  }

  print("Today is ${dayName}");
}
```

Output: Today is Friday

Week 4

Create an application to demonstrate user defined functions using Dart.

In the below example, we declared a function named sum() and passed two integer variables as actual parameters. In the function body, we declared a result variable to store the sum of two numbers and returned the result.

In order to add two values, we called a function with the same name, passed formal parameters 30 and 20. The sum() returned a value which we stored in the variable c and printed the sum on the console.

```
void main() {
  print("Example of add two number using the function");
  // Creating a Function

  int sum(int a, int b){
        // function Body
        int result;
        result = a+b;
        return result;
}
// We are calling a function and storing a result in variable c
var c = sum(30,20);
print("The sum of two numbers is: ${c}");
}
```

Output

```
Example of add two number using the function
The sum of two numbers is: 50
```

Week 5

Create an application to implement object-oriented programming using Dart.

Let’s create a simple Dart application that demonstrates object-oriented programming (OOP) concepts. We’ll cover classes, objects, inheritance, polymorphism, interfaces, and abstract classes.

Class and Object:

- A class in Dart serves as a blueprint for creating objects. It defines the properties (fields) and behaviour (methods) that the objects will have.
- An object is an instance of a class, representing a real-world entity. Objects have both state (data) and behaviour (methods).

Dart Code:

```
// Define a simple class called 'Person'
class Person {
  String name;
  int age;

  // Constructor
  Person(this.name, this.age);

  // Method to greet
  void greet() {
    print('Hello, my name is $name and I am $age years old.');
  }
}

void main() {
  // Create an object of the 'Person' class
  var person1 = Person('Alice', 30);

  // Access object properties and call methods
  person1.greet(); // Output: Hello, my name is Alice and I am 30 years old.
}
```

Output: Hello, my name is Alice and I am 30 years old.

Inheritance:

- Inheritance allows you to create a new class (child class) based on an existing class (parent class).
- The child class inherits properties and methods from the parent class.

Dart Code:

```
// Define a child class 'Student' that inherits from 'Person'
class Student extends Person {
  String major;

  // Constructor
  Student(String name, int age, this.major) : super(name, age);

  // Override the 'greet' method @override
  void greet() {
    print('Hi, I am a student named $name studying $major.');
  }
}

void main() {
  var student1 = Student('Bob', 20, 'Computer Science');
  student1.greet(); // Output: Hi, I am a student named Bob studying Computer Science.
}
```

Output: Hi, I am a student named Bob studying Computer Science.

Polymorphism:

- Polymorphism allows objects of different classes to be treated uniformly.
- We achieve polymorphism through method overriding.

Interfaces:

- An interface defines a contract that other classes must adhere to.
- Dart doesn't have explicit interfaces, but you can achieve similar behaviour using abstract classes.

Abstract Class:

- An abstract class cannot be instantiated directly; it serves as a base for other classes.
- Abstract classes can have abstract methods (only method signature, no implementation).

Dart Code:

```
// Define an abstract class 'Shape'
abstract class Shape {
  double area(); // Abstract method
}

// Implement 'Shape' with a concrete class 'Circle'
class Circle implements Shape {
  double radius;
  Circle(this.radius);

  @override
  double area() {
    return 3.14 * radius * radius;
  }
}

void main() {
  var circle = Circle(5);
  print('Area of the circle: ${circle.area()}'); // Output: Area of the circle: 78.5
}
```

Output: Area of the circle: 78.5

Week-6

Create an application for platform basic widgets (Text, Image, and Icon).
we will create a Flutter application, and use Text Widget to display title in application bar and a message in the body of an application.

Step 1: Create a Flutter application from any of your favorite IDE. This is a good old counter example.

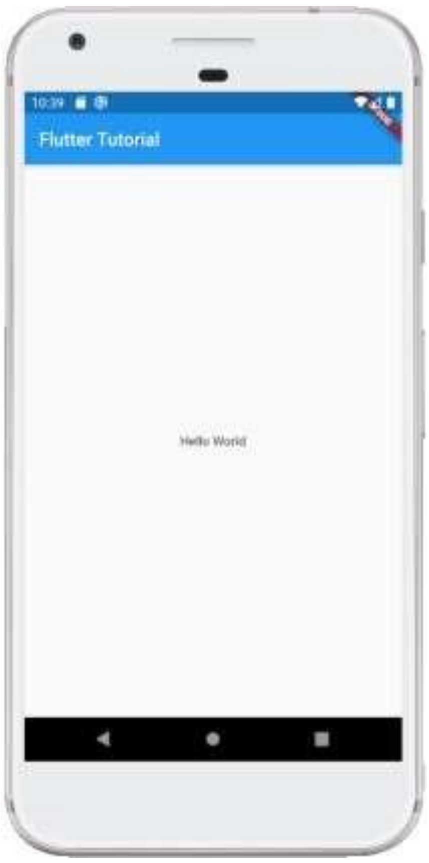
Step 2: Replace the contents of lib/main.dart with the following code.

Code:
main.dart

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Tutorial',
      home: Scaffold(
      appBar: AppBar(
          title: Text('Flutter Text Widget Tutorial'),
        ),
        body: Center(
          child: Text('Hello World'),
        ),
      ),
    );
  }
}
```

When you run this application, the text widgets are displayed in the application as shown in the following figure.

Output:



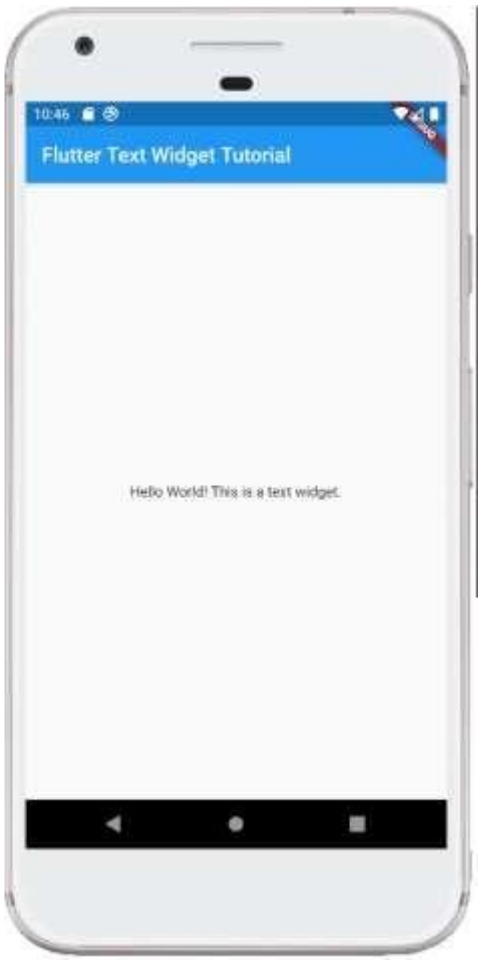
Hello World

Step 3
Let us change the text provided to text widgets and hot reload the application.
Code:
main.dart

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Tutorial',
      home: Scaffold(
      appBar: AppBar(
          title: Text('Flutter Text Widget Tutorial'),
        ),
        body: Center(
          child: Text('Hello World! This is a text widget.'),
        ),
      ),
    );
  }
```

}

Output:



Hello World! This is a text widget.

Flutter Image

- Flutter Image widget displays an image in our Flutter Application.

A quick code snippet to display an image using Image class is

```
const Image(image: NetworkImage('https://www.tutorialkart.com/img/lion.jpg'),)
```

In the above code, we are fetching the image from network, therefore, we have provided NetworkImage (ImageProvider object) to image property.

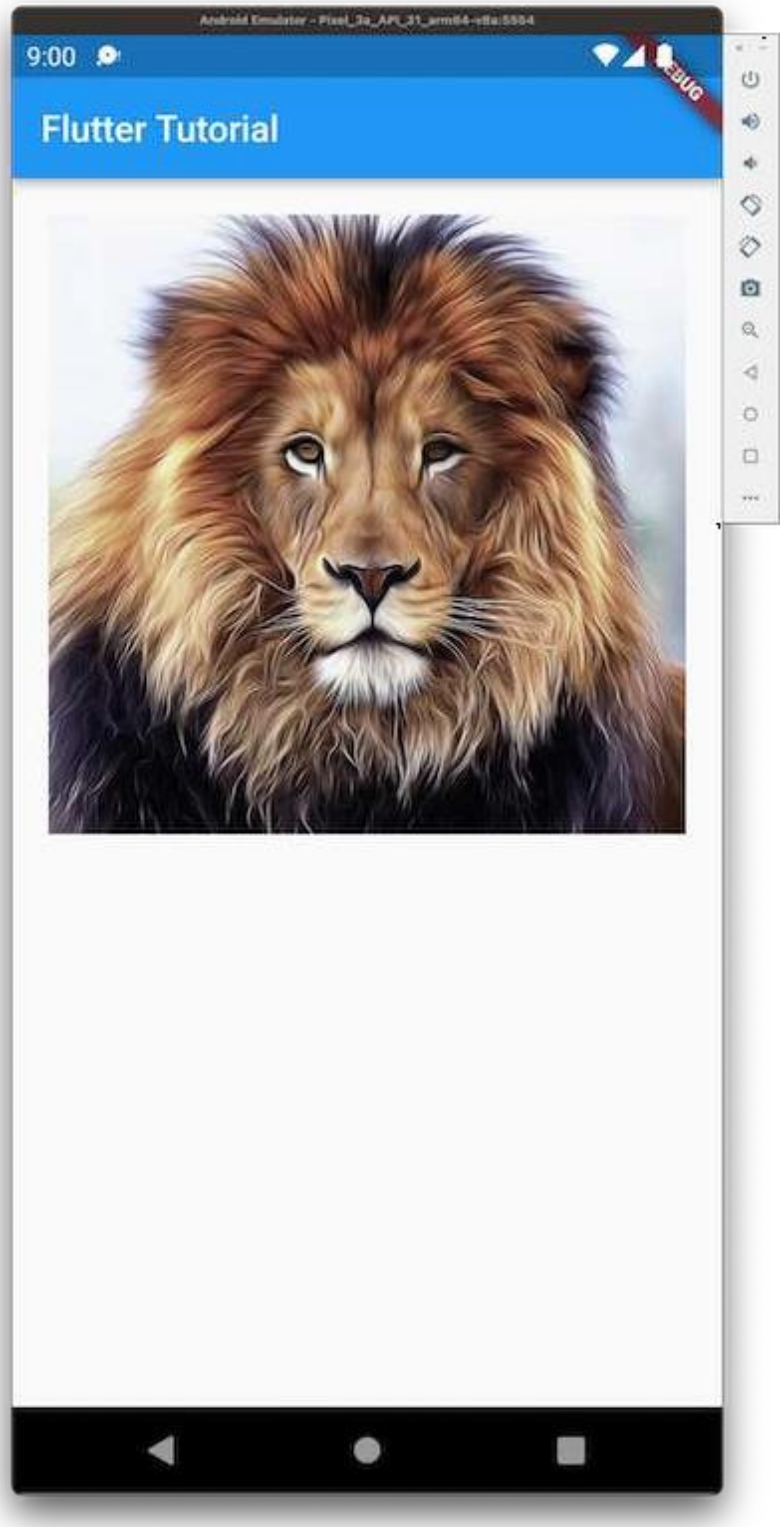
In the following example, we display an image using Image class.

Code:
main.dart

```
import 'dart:ui';

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

```
void main() => runApp(const MyApp());

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

  static const String _title = 'Flutter Tutorial';

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: _title,
      home: Scaffold(
        appBar: AppBar(title: const Text(_title)),
        body: const MyStatefulWidget(),
      ),
    );
  }
}

class MyStatefulWidget extends StatefulWidget {
  const MyStatefulWidget({Key? key}) : super(key: key);

  @override
  State<MyStatefulWidget> createState() => _MyStatefulWidgetState();
}

class _MyStatefulWidgetState extends State<MyStatefulWidget> {
  @override
  Widget build(BuildContext context) {
    return Center(
        child: Column(
          children: <Widget>[
          Container(
            margin: const EdgeInsets.all(20),
            child: const Image(
              image: NetworkImage(
                    'https://www.tutorialkart.com/img/lion.jpg'),
            ),
          ),
        ],
      ),
    );
  }
}
```

Output:


Android Emulator - Pixel_3a_API_31_arm64-v8a:5554
9:00
DEBUG
Flutter Tutorial

Icons can be used as a representative symbol for a quick understanding of the functionality, or path of the navigation, etc.
Following are the list of examples.

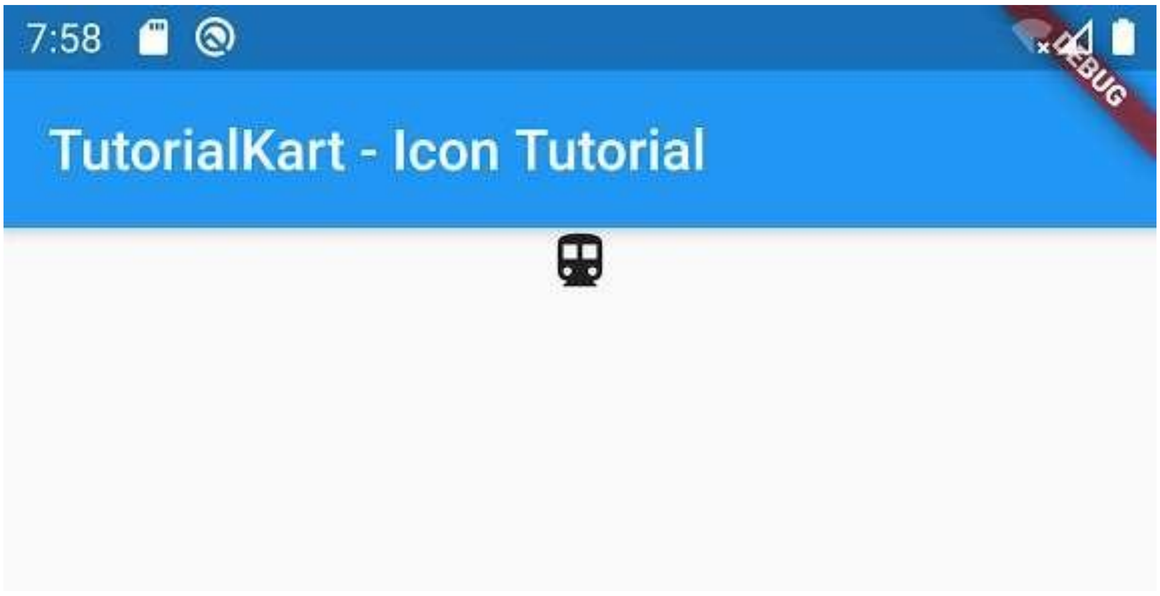
- Basic Icon widget example, with default values.
- Change Icon Size using size property.
- Change Icon Color using color property.
- Icon with a Text Label below it with the help of Column widget.

The list of all icons that come with flutter are available in Icons class. You can also use icons from assets.

```
Icon(Icons.directions_transit)
```

Code:
main.dart

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Icon Tutorial',
      theme: ThemeData(
      primarySwatch: Colors.blue,
      ),
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('TutorialKart - Icon Tutorial'),
      ),
      body: Column(children: <Widget>[
        Center(child: Icon(Icons.directions_transit)),
      ]),
    );
  }
}
```
Output:



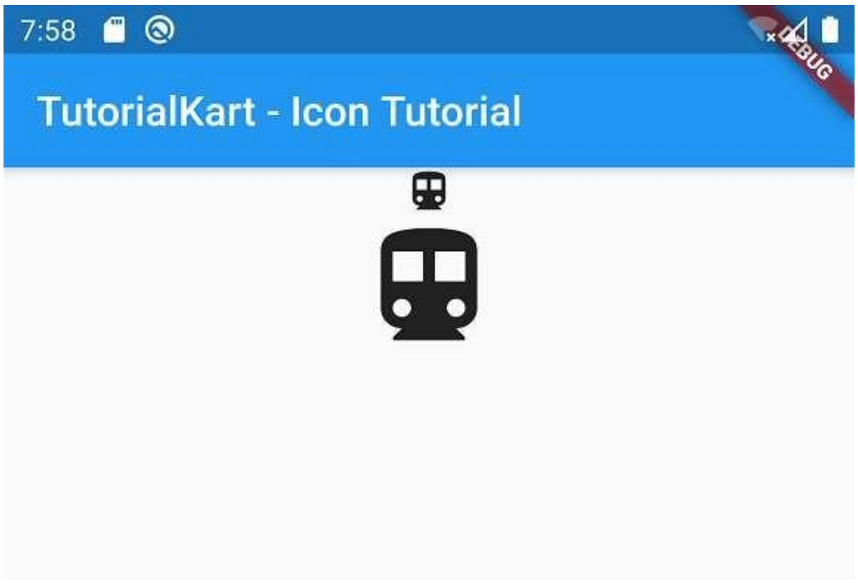
TutorialKart - Icon Tutorial

Increase the Size of Icon
You can increase the size of Icon to a required value by assigning the size property with specific double value.

```
Icon(
  Icons.directions_transit,
  size: 70,
)
```
Following is the complete code to change the size of icon.
Code:
main.dart

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
 // This widget is the root of your application.
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Flutter Icon Tutorial',
     theme: ThemeData(
     primarySwatch: Colors.blue,
     ),
     home: MyHomePage(),
   );
 }
}

class MyHomePage extends StatefulWidget {
 @override
 _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: Text('TutorialKart - Icon Tutorial'),
     ),
     body: Column(children: <Widget>[
       //basic example
       Center(child: Icon(Icons.directions_transit)),
       //increase the size of icon
       Center(child: Icon(Icons.directions_transit, size: 70,)),
     ]),
   );
 }
}
```

Output:



Change Color of Icon

You can change the color of Icon widget using color property. Provide a value of type Color to the color property as shown below. You can specify color using Colors class, Color.fromARGB(), Color.fromRGBO(), etc.

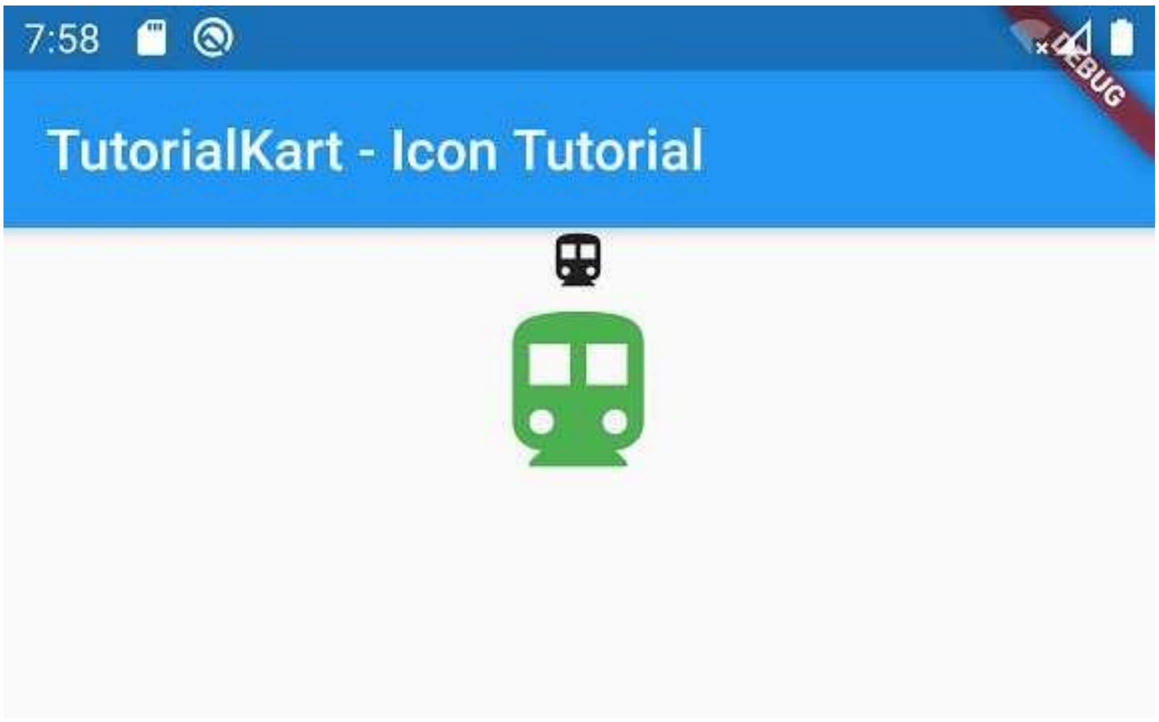
Code:

main.dart

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Icon Tutorial',
      theme: ThemeData(
      primarySwatch: Colors.blue,
      ),
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}
```

```
class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('TutorialKart - Icon Tutorial'),
      ),
      body: Column(children: <Widget>[
        //basic example
        Center(child: Icon(Icons.directions_transit)),
        //change color of icon
        Center(child: Icon(Icons.directions_transit, color:Colors.green, size: 70,)),
      ]),
    );
  }
}
```

Output:



Week 7

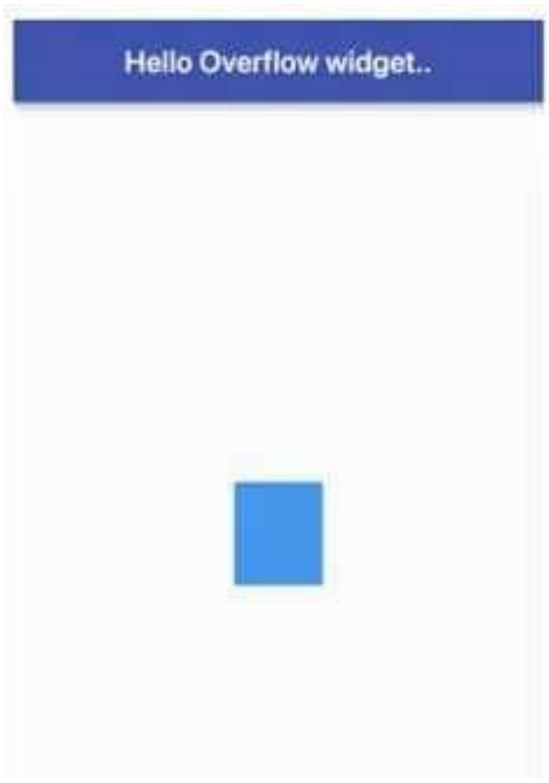
Create an application for Layout widgets (Single child, Multiple Child).

Single Child Widget.

Code:

```
import 'package:flutter/material.dart';
void main() => runApp(MyApp());
class MyApp extends StatelessWidget {
// Root widget of the application
@override
Widget build(BuildContext context) {
return MaterialApp(
title: 'Example of Single Layout',
debugShowCheckedModeBanner: false,
theme: ThemeData(
// Theme of the application
primarySwatch: Colors.indigo,
),
home: SingleWidget(),
);
}

}

class SingleWidget extends StatelessWidget {
@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: Text("Hello Overflow widget.."),
),
body: Center(
child: Container(
height: 70.0,
width: 60.0,
color: Colors.blue,
child: OverflowBox(
minHeight: 90.0,
minWidth: 90.0,
child: Container(
height: 60.0,
width: 60.0,
color: Colors.red,
child: Align(
alignment: Alignment.topRight,
)
),
),
),
),
);
}
}
```

Output:



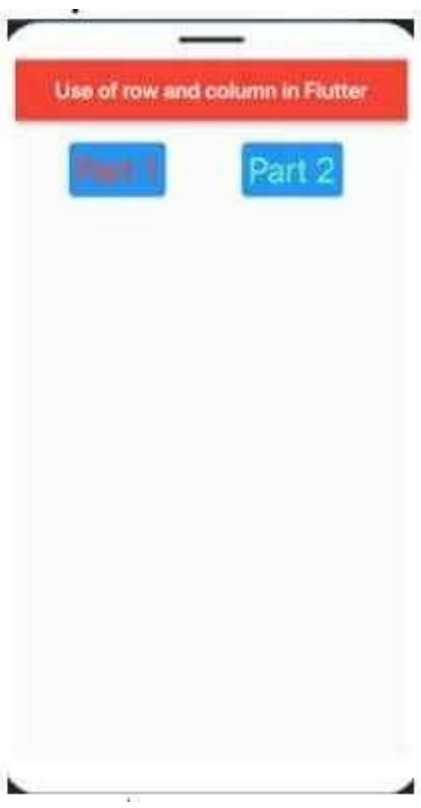
Multiple Child Widget

Code:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {

//Root widget of application

@override

Widget build(BuildContext context) {
```

```
return MaterialApp(

title: 'Example of multiple Child',

debugShowCheckedModeBanner: false,

theme: ThemeData(

// Defining the application theme

primarySwatch: Colors.red,

),

home: MultiChild(),

);

}

}

class MultiChild extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(
title: Text("Use of row and column in Flutter"),

),

body: Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children:<Widget>[

Container(

margin: EdgeInsets.all(10.0),

padding: EdgeInsets.all(6.0),

decoration:BoxDecoration(

borderRadius:BorderRadius.circular(5),

color:Colors.blue

),

child: Text("Part 1",style: TextStyle(color:Colors.red,fontSize:32),),

),

Container(

margin: EdgeInsets.all(20.0),
padding: EdgeInsets.all(6.0),

decoration:BoxDecoration(

borderRadius:BorderRadius.circular(5),

color:Colors.blue

),

child: Text("Part 2",style: TextStyle(color:Colors.greenAccent,fontSize:32),),

)

]

),

);

}

}
```

Output:


Use of row and column in Flutter
Part 1
Part 2

Week 8

Create an application to demonstrate Gesture Detector.

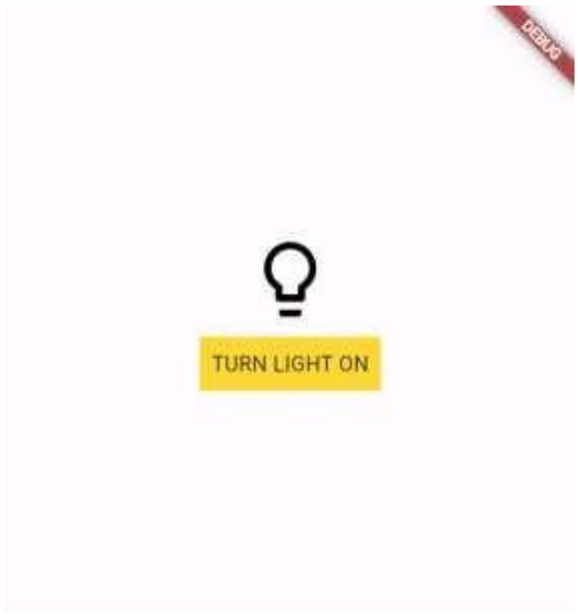
Example-1

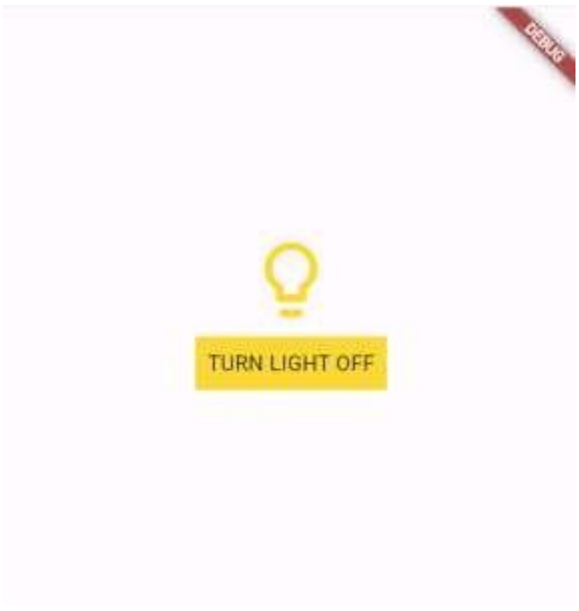
Code:

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

/// Flutter code sample for [GestureDetector].

void main() => runApp(const GestureDetectorExampleApp());

class GestureDetectorExampleApp extends StatelessWidget {

const GestureDetectorExampleApp({super.key});

  @override

  Widget build(BuildContext context) {

    return const MaterialApp(

      home: GestureDetectorExample(),

    );

  }

}

class GestureDetectorExample extends StatefulWidget {

  const GestureDetectorExample({super.key});

  @override

  State<GestureDetectorExample> createState() => _GestureDetectorExampleState();

}

class _GestureDetectorExampleState extends State<GestureDetectorExample> {

  bool _lightIsOn = false;

  @override

  Widget build(BuildContext context) {
```

```
return Scaffold(
  body: Container(
    alignment: FractionalOffset.center,
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: Icon(
          Icons.lightbulb_outline,
            color: _lightIsOn ? Colors.yellow.shade600 : Colors.black,
            size: 60,
          ),
        ),
        GestureDetector(
          onTap: () {
            setState(() {
              // Toggle light when tapped.
              _lightIsOn = !_lightIsOn;
            });
          },
          child: Container(
            color: Colors.yellow.shade600,
            padding: const EdgeInsets.all(8),
                // Change button text when light changes state.
                child: Text(_lightIsOn ? 'TURN LIGHT OFF' : 'TURN LIGHT ON'),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

Output:





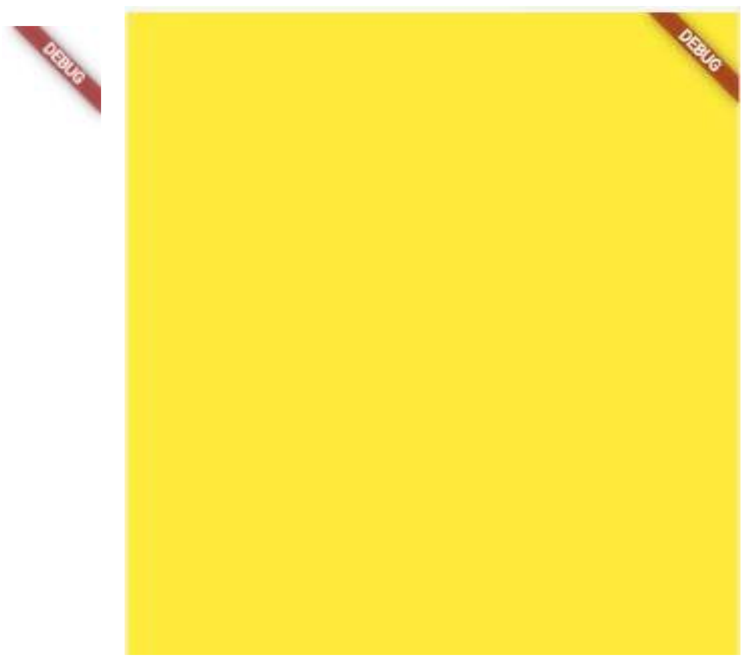
Example-2

This example uses a Container that wraps a GestureDetector widget which detects a tap.

Since the GestureDetector does not have a child, it takes on the size of its parent, making the entire area of the surrounding Container clickable. When tapped, the Container turns yellow by setting the _color field. When tapped again, it goes back to white.

Code:

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

/// Flutter code sample for [GestureDetector].

void main() => runApp(const GestureDetectorExampleApp());

class GestureDetectorExampleApp extends StatelessWidget {

const GestureDetectorExampleApp({super.key});

  @override

  Widget build(BuildContext context) {

    return const MaterialApp(

      home: GestureDetectorExample(),

    );

  }

}

class GestureDetectorExample extends StatefulWidget {

  const GestureDetectorExample({super.key});

  @override

  State<GestureDetectorExample> createState() => _GestureDetectorExampleState();

}

class _GestureDetectorExampleState extends State<GestureDetectorExample> {
  Color _color = Colors.white;
  @override
  Widget build(BuildContext context) {
    return Container(
      color: _color,
      height: 200.0,
      width: 200.0,
      child: GestureDetector(
        onTap: () {
          setState(() {
            _color == Colors.yellow
                ? _color = Colors.white
                : _color = Colors.yellow;
          });
        },
      ),
    );
  }
}
```



Example-3:

This example uses a GestureDetector that wraps a green Container and a second GestureDetector that wraps a yellow Container. The second GestureDetector is a child of the green Container. Both GestureDetectors define an onTap callback. When the callback is called it adds a red border to the corresponding Container.

When the green Container is tapped, it's parent GestureDetector enters the gesture arena. It wins because there is no competing GestureDetector and the green Container shows a red border. When the yellow Container is tapped, it's parent GestureDetector enters the gesture arena. The GestureDetector that wraps the green Container also enters the gesture arena (the pointer events coordinates are inside both GestureDetectors bounds). The GestureDetector that wraps the yellow Container wins because it was the first detector to enter the arena.

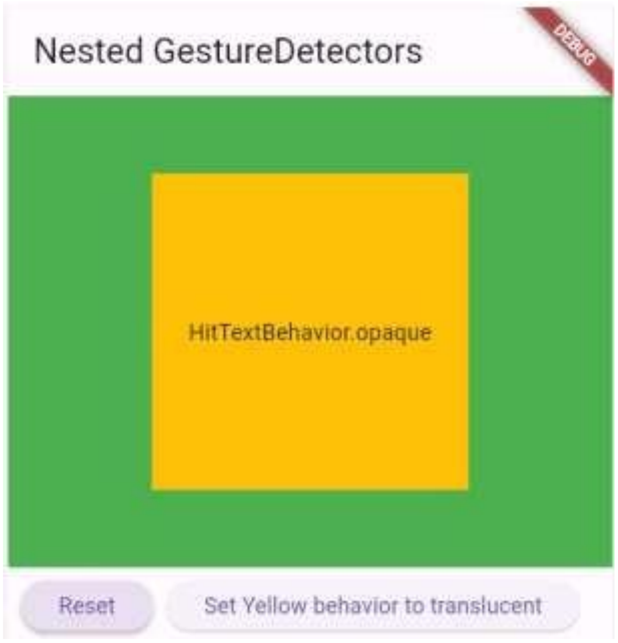
Code:

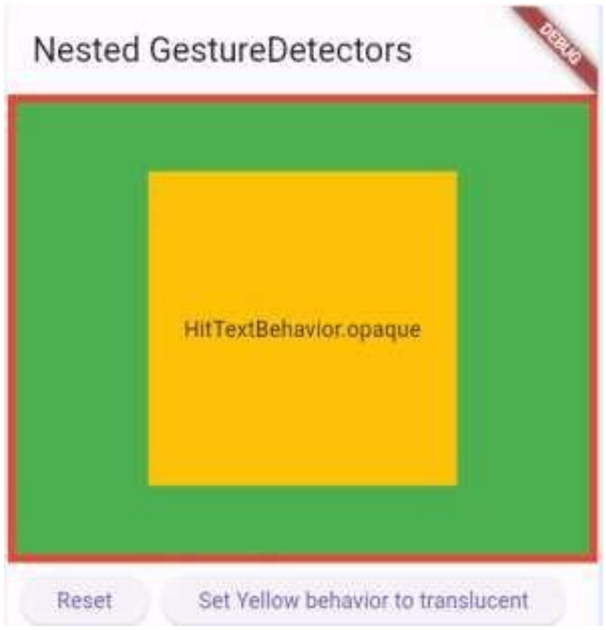
```
import 'package:flutter/gestures.dart';

import 'package:flutter/material.dart';

/// Flutter code sample for [GestureDetector].

void main() {

debugPrintGestureArenaDiagnostics = true;
```

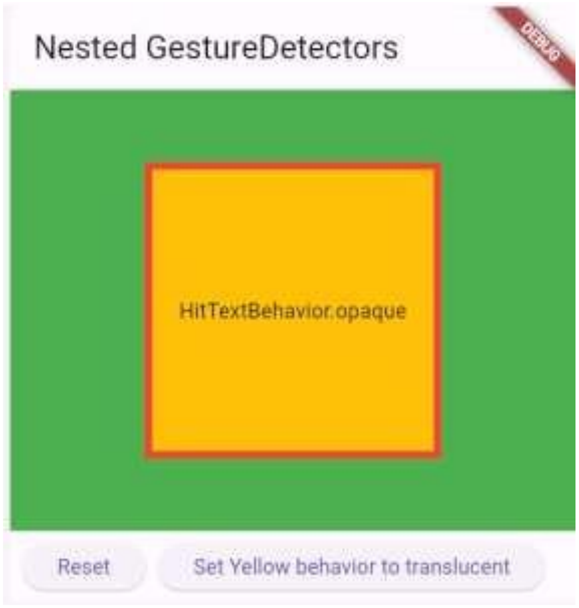
```
  runApp(const NestedGestureDetectorsApp());
}
enum _OnTapWinner { none, yellow, green }
class NestedGestureDetectorsApp extends StatelessWidget {
  const NestedGestureDetectorsApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: const Text('Nested GestureDetectors')),
        body: const NestedGestureDetectorsExample(),
      ),
    );
  }
}
class NestedGestureDetectorsExample extends StatefulWidget {
  const NestedGestureDetectorsExample({super.key});
  @override
  State<NestedGestureDetectorsExample> createState() =>
      _NestedGestureDetectorsExampleState();
}
class _NestedGestureDetectorsExampleState
    extends State<NestedGestureDetectorsExample> {
```

```
bool _isYellowTranslucent = false;
_OnTapWinner _winner = _OnTapWinner.none;
final Border highlightBorder = Border.all(color: Colors.red, width: 5);
@override
Widget build(BuildContext context) {
  return Column(
    children: <Widget>[
      Expanded(
        child: GestureDetector(
          onTap: () {
          debugPrint('Green onTap');
          setState(() {
              _winner = _OnTapWinner.green;
            });
          },
          onTapDown: (_) => debugPrint('Green onTapDown'),
          onTapCancel: () => debugPrint('Green onTapCancel'),
          child: Container(
            alignment: Alignment.center,
            decoration: BoxDecoration(
              border: _winner == _OnTapWinner.green ? highlightBorder : null,
              color: Colors.green,
            ),
child: GestureDetector(
  // Setting behavior to transparent or opaque as no impact on
  // parent-child hit testing. A tap on 'Yellow' is also in
  // 'Green' bounds. Both enter the gesture arena, 'Yellow' wins
  // because it is in front.
  behavior: _isYellowTranslucent
      ? HitTestBehavior.translucent
      : HitTestBehavior.opaque,
  onTap: () {
  debugPrint('Yellow onTap');
  setState(() {
      _winner = _OnTapWinner.yellow;
    });
  },
  child: Container(
    alignment: Alignment.center,
    decoration: BoxDecoration(
    border:
          _winner == _OnTapWinner.yellow ? highlightBorder : null,
      color: Colors.amber,
    ),
    width: 200,
    height: 200,
          child: Text(
            'HitTextBehavior.${_isYellowTranslucent ? 'translucent' : 'opaque'}',
            textAlign: TextAlign.center,
          ),
        ),
      ),
    ),
  ),
),
Padding(
  padding: const EdgeInsets.all(8.0),
  child: Row(
    children: <Widget>[
      ElevatedButton(
        child: const Text('Reset'),
        onPressed: () {
          setState(() {
            _isYellowTranslucent = false;
            _winner = _OnTapWinner.none;
          });
        },
      ),
      const SizedBox(width: 8),
              ElevatedButton(
                child: Text(
                  'Set Yellow behavior to ${_isYellowTranslucent ? 'opaque' : 'translucent'}',
                ),
                onPressed: () {
                  setState(() => _isYellowTranslucent = !_isYellowTranslucent);
                },
              ),
            ],
          ),
        ),
      ],
    );
  }
  @override
  void dispose() {
    debugPrintGestureArenaDiagnostics = false;
    super.dispose();
  }
}
```

Output:


Nested GestureDetectors
DEBUG
HitTextBehavior.opaque
Reset
Set Yellow behavior to translucent


Nested GestureDetectors
DEBUG
HitTextBehavior.opaque
Reset
Set Yellow behavior to translucent


Nested GestureDetectors
DEBUG
HitTextBehavior.opaque
Reset
Set Yellow behavior to translucent

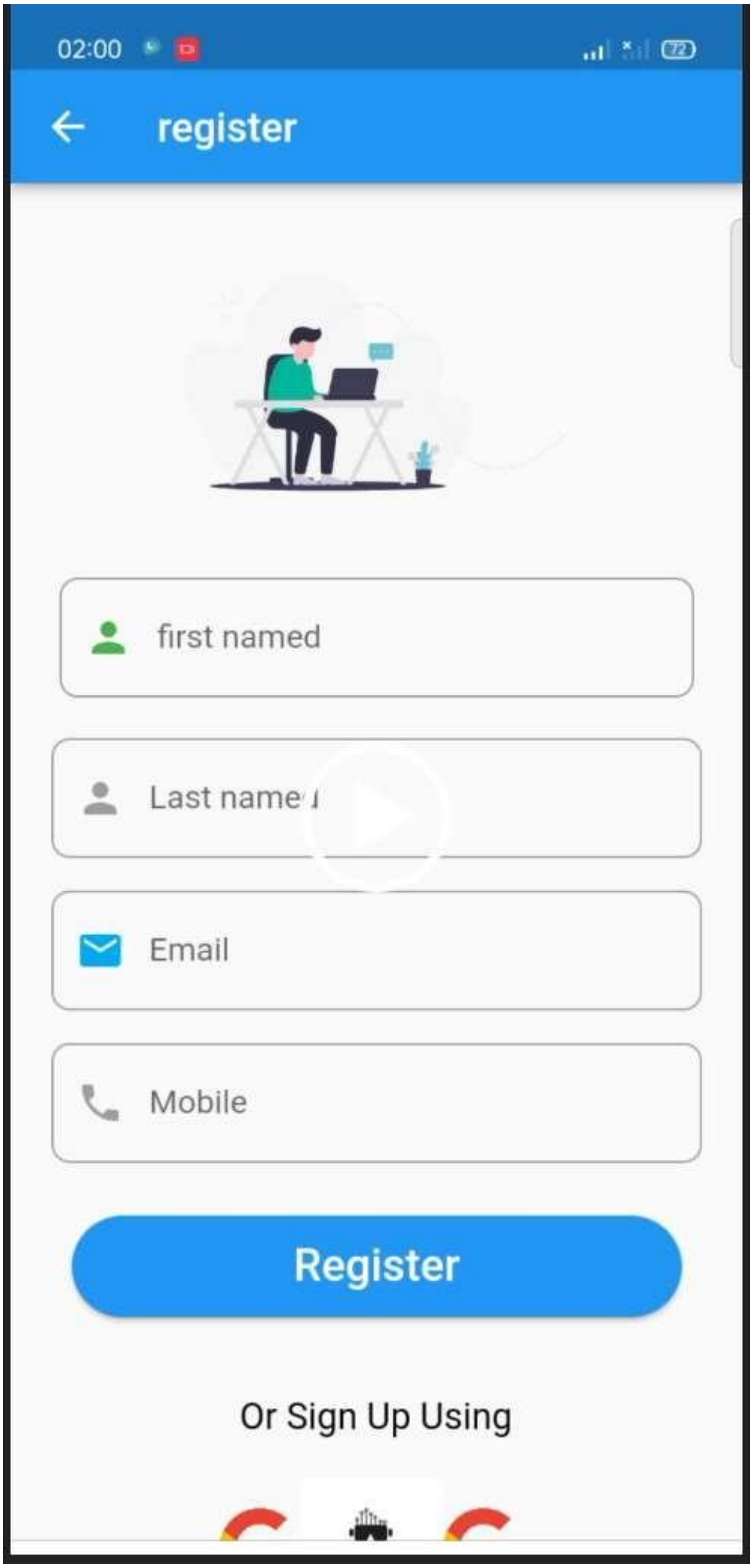
Week 9 & 10

Create an application for Registration form.

Code:

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

import 'package:form_field_validator/form_field_validator.dart';

import 'package:flutter/foundation.dart';


class Register extends StatefulWidget {

const Register({Key? key}) : super(key: key);


@override

State<Register> createState() => _RegisterState();

}


class _RegisterState extends State<Register> {

Map userData = {};

final _formkey = GlobalKey<FormState>();

@override

Widget build(BuildContext context) {

return Scaffold(

        appBar: AppBar(

        title: Text('register'),

        ),
body: SingleChildScrollView(
child: Padding(
        padding: const EdgeInsets.all(12.0),
        child: Form(
                key: _formkey,
                child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: <Widget>[
                        Padding(
                        padding: const EdgeInsets.only(top: 20.0),
                        child: Center(
                                child: Container(
                                width: 200,
                                height: 150,
                                //decoration: BoxDecoration(
                                //borderRadius: BorderRadius.circular(40),
                                //border: Border.all(color: Colors.blueGrey)),
                                child: Image.asset('assets/logo.png'),
                                ),
                        ),
                        ),
                        Padding(
                        padding: const EdgeInsets.all(12.0),
child: TextFormField(
          // validator: ((value) {
          // if (value == null || value.isEmpty) {
          //          return 'please enter some text';
          // } else if (value.length < 5) {
          //          return 'Enter atleast 5 Charecter';
          // }

          // return null;
          // }),
          validator: MultiValidator([
          RequiredValidator(errorText: 'Enter first named'),
          MinLengthValidator(3,
                    errorText: 'Minimum 3 charecter filled name'),
          ]),

          decoration: InputDecoration(
                    hintText: 'Enter first Name',
                    labelText: 'first named',
                    prefixIcon: Icon(
                    Icons.person,
                    color: Colors.green,
                    ),
                                        errorStyle: TextStyle(fontSize: 18.0),
                                        border: OutlineInputBorder(
                                                borderSide: BorderSide(color: Colors.red),
                                                borderRadius:

BorderRadius.all(Radius.circular(9.0)))),
                        ),
                        ),
                        Padding(
                        padding: const EdgeInsets.all(8.0),
                        child: TextFormField(
                                validator: MultiValidator([
                                RequiredValidator(errorText: 'Enter last named'),
                                MinLengthValidator(3,
                                        errorText:
                                                'Last name should be atleast 3 charater'),
                                ]),
                                decoration: InputDecoration(
                                        hintText: 'Enter last Name',
                                        labelText: 'Last named',
                                        prefixIcon: Icon(
                                        Icons.person,
                                        color: Colors.grey,
                                        ),
                                        errorStyle: TextStyle(fontSize: 18.0),
                                        border: OutlineInputBorder(
                                              borderSide: BorderSide(color: Colors.red),
                                              borderRadius:
BorderRadius.all(Radius.circular(9.0)))),
                            ),
                            ),
                            Padding(
                            padding: const EdgeInsets.all(8.0),
                            child: TextFormField(
                                  validator: MultiValidator([
                                  RequiredValidator(errorText: 'Enter email address'),
                                  EmailValidator(
                                        errorText: 'Please correct email filled'),
                                  ]),
                                  decoration: InputDecoration(
                                        hintText: 'Email',
                                        labelText: 'Email',
                                        prefixIcon: Icon(
                                        Icons.email,
                                        color: Colors.lightBlue,
                                        ),
                                        errorStyle: TextStyle(fontSize: 18.0),
                                    border: OutlineInputBorder(
                                                borderSide: BorderSide(color: Colors.red),
                                                borderRadius:

BorderRadius.all(Radius.circular(9.0)))),
                        ),
                        ),
                        Padding(
                        padding: const EdgeInsets.all(8.0),
                        child: TextFormField(
                                    validator: MultiValidator([
                                    RequiredValidator(errorText: 'Enter mobile number'),
                                    PatternValidator(r'(^[0,9]{10}$)',
                                                errorText: 'enter vaid mobile number'),
                                    ]),
                                    decoration: InputDecoration(
                                                hintText: 'Mobile',
                                                labelText: 'Mobile',
                                                prefixIcon: Icon(
                                                Icons.phone,
                                                color: Colors.grey,
                                                ),
                                                border: OutlineInputBorder(
                                                            borderSide: BorderSide(color: Colors.red),
                              borderRadius:
BorderRadius.all(Radius.circular(9)))),
            ),
            ),
            Center(
                child: Padding(
            padding: const EdgeInsets.all(18.0),
            child: Container(
                // margin: EdgeInsets.fromLTRB(200, 20, 50, 0),
                child: RaisedButton(
                child: Text(
                    'Register',
                    style: TextStyle(color: Colors.white, fontSize: 22),
                ),
                onPressed: () {
                    if (_formkey.currentState!.validate()) {
                    print('form submiitted');
                    }
                },
                shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(30)),
                color: Colors.blue,
                ),
        width: MediaQuery.of(context).size.width,

        height: 50,
),
)),
Center(
child: Padding(
        padding: EdgeInsets.only(top: 20),
        child: Center(
        child: Text(
                'Or Sign Up Using',
                style: TextStyle(fontSize: 18, color: Colors.black),
        ),
        ),
),
),
Center(
child: Padding(
        padding: EdgeInsets.only(top: 20, left: 90),
        child: Row(
        children: [
                Container(
                height: 40,

                width: 40,

                child:  Image.asset(

                'assets/google.png',

                fit: BoxFit.cover,

                )),

          Container(

          height: 70,

          width: 70,

          child: Image.asset(

                'assets/vishal.png',

                fit: BoxFit.cover,

          ),

          ),

          Container(

          height: 40,

          width: 40,

          child: Image.asset(

                'assets/google.png',

                fit: BoxFit.cover,

          ),

          ),

    ],
                    ),
                ),
                ),
                Center(
                child: Container(
                    padding: EdgeInsets.only(top: 60),
                    child: Text(
                    'SIGN IN',
                    style: TextStyle(
                        fontSize: 20, fontWeight: FontWeight.bold),
                    ),
                ),
                )
            ],
            )),
    ),
    ));
}
}
```

Output:



02:00

register

first named

Last name

Email

Mobile

Register

Or Sign Up Using

Week 11

Create an application to implement flutter calendar.

Code:

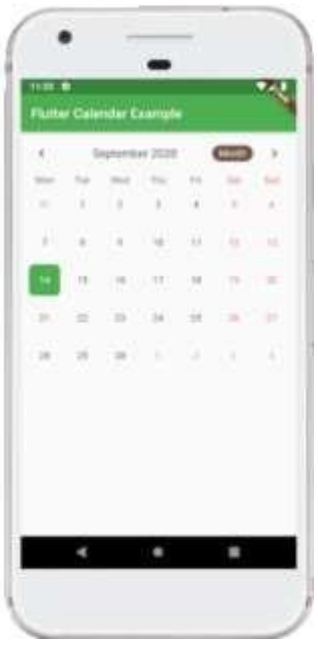
```
import 'package:flutter/material.dart';

import 'package:table_calendar/table_calendar.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      theme: ThemeData(

        primarySwatch: Colors.green,

      ),

      home: HomeCalendarPage(),

    );

  }

}

class HomeCalendarPage extends StatefulWidget {

  @override

  _HomeCalendarPageState createState() => _HomeCalendarPageState();

}

class _HomeCalendarPageState extends State<HomeCalendarPage> {

  CalendarController _controller;

  @override
```

```
void initState() {
  super.initState();
  _controller = CalendarController();
}
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Flutter Calendar Example'),
    ),
    body: SingleChildScrollView(
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
          TableCalendar(
            initialCalendarFormat: CalendarFormat.month,
            calendarStyle: CalendarStyle(
                todayColor: Colors.blue,
                selectedColor: Theme.of(context).primaryColor,
                todayStyle: TextStyle(
                    fontWeight: FontWeight.bold,
                    fontSize: 22.0,
                    color: Colors.white)
```

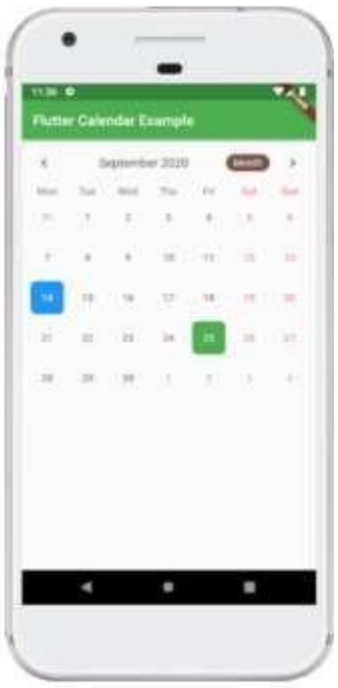
```
),
headerStyle: HeaderStyle(
  centerHeaderTitle: true,
  formatButtonDecoration: BoxDecoration(
  color: Colors.brown,
    borderRadius: BorderRadius.circular(22.0),
  ),
  formatButtonTextStyle: TextStyle(color: Colors.white),
  formatButtonShowsNext: false,
),
startingDayOfWeek: StartingDayOfWeek.monday,
onDaySelected:  (date, events) {
print(date.toUtc());
},
builders: CalendarBuilders(
  selectedDayBuilder: (context, date, events) => Container(
      margin: const EdgeInsets.all(5.0),
      alignment: Alignment.center,
      decoration: BoxDecoration(
          color: Theme.of(context).primaryColor,
          borderRadius: BorderRadius.circular(8.0)),
      child: Text(
        date.day.toString(),
                    style: TextStyle(color: Colors.white),
                  )),
              todayDayBuilder: (context, date, events) => Container(
                  margin: const EdgeInsets.all(5.0),
                  alignment: Alignment.center,
                  decoration: BoxDecoration(
                      color: Colors.blue,
                      borderRadius: BorderRadius.circular(8.0)),
                  child: Text(
                    date.day.toString(),
                    style: TextStyle(color: Colors.white),
                  )),
            ),
            calendarController: _controller,
          )
        ],
      ),
    ),
  );
 }
}
```

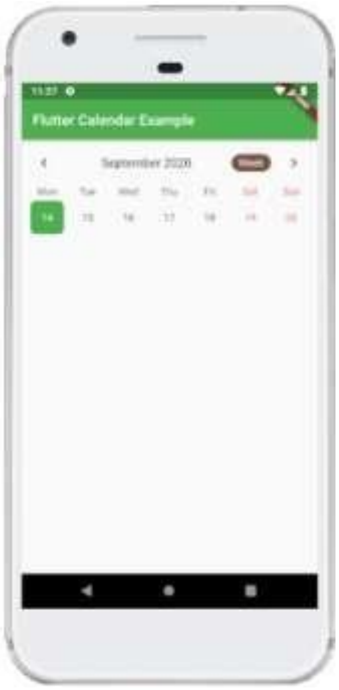
Output:

When we run the app in the device or emulator, we should see the UI similar to the below screenshot. Here, we can see the previous and next arrow icon to display the month. The week starts from Monday, and date 14 is my current date.



If we select another date, we can see that the current date and selected date are in a different color. See the below image.





We can also display the week of the month, as shown in this image.

Week 12

Create an application to implement Animated Text in Flutter

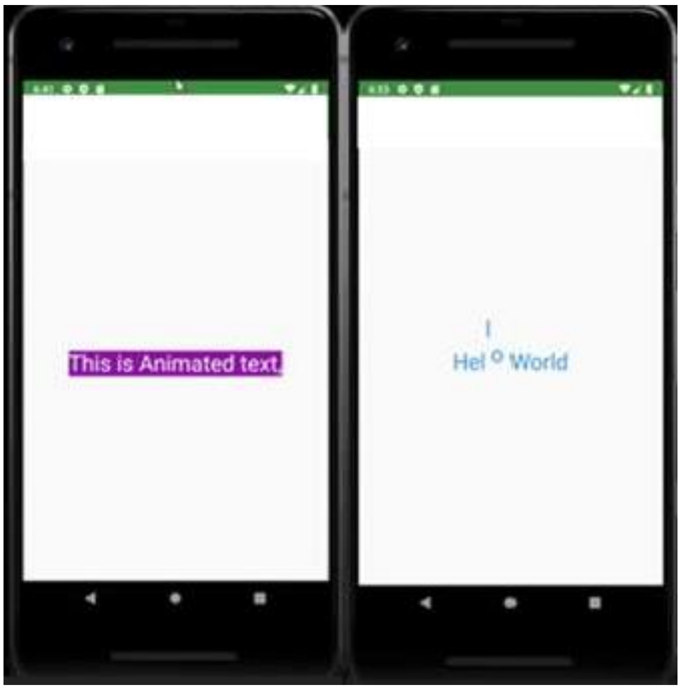
Code:

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

import 'package:flutter/material.dart';

import 'package:animated_text_kit/animated_text_kit.dart';


void main() => runApp(MyApp());


class MyApp extends StatefulWidget {

@override

_MyAppState createState() => _MyAppState();

}


class _MyAppState extends State<MyApp> {

@override

Widget build(BuildContext context) {

        return MaterialApp(

                title: 'Animated Text Kit',

                debugShowCheckedModeBanner: false,

                theme: ThemeData(primarySwatch: Colors.green),

                home: Scaffold(

                appBar: AppBar(
        title: const Text("Welcome to AI"),
        centerTitle: true,
),
body: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
        AnimatedTextKit(
                animatedTexts: [
                RotateAnimatedText('AWESOME',
                        textStyle: TextStyle(
                                fontSize: 30,
                                color: Colors.white,
                                backgroundColor: Colors.blue)),
                RotateAnimatedText('OPTIMISTIC',
                        textStyle: TextStyle(
                                letterSpacing: 3,
                                fontSize: 30,
                                fontWeight: FontWeight.bold,
                                color: Colors.orange)),
                RotateAnimatedText(
                        'Welcome to AI',
                        textStyle: TextStyle(
                        fontSize: 30,
                        decoration: TextDecoration.underline,
                ),
            ),
            ],
            isRepeatingAnimation: true,
            totalRepeatCount: 10,
            pause: Duration(milliseconds: 1000),
        ),
        // SizedBox(height: 10),
        Center(
            child: AnimatedTextKit(
            totalRepeatCount: 40,
            animatedTexts: [
                FadeAnimatedText(
                'First Fade',
                textStyle: const TextStyle(
                    backgroundColor: Colors.green,
                    color: Colors.white,
                    fontSize: 32.0,
                    fontWeight: FontWeight.bold),
                ),
                ScaleAnimatedText(
                'Then Get Bigger',
                duration: Duration(milliseconds: 4000),
                textStyle:
```

```
                                                const TextStyle(color: Colors.indigo, fontSize:
50.0),
                                        ),
                                ],
                                ),
                        ),


                        SizedBox(height: 10),
                        AnimatedTextKit(
                                animatedTexts: [
                                TyperAnimatedText('This is Animated text,',
                                        textStyle: const TextStyle(
                                                color: Colors.white,
                                                fontSize: 30,
                                                backgroundColor: Colors.purple)),
                                TyperAnimatedText('You are viewing it here.',
                                        textStyle: const TextStyle(
                                                fontSize: 20, fontWeight: FontWeight.bold)),
                                ],
                                onTap: () {
                                print("I am executing");
                                },
                        ),
        SizedBox(height: 10),
        Center(
            child: AnimatedTextKit(
            animatedTexts: [
                WavyAnimatedText('Hello World',
                    textStyle: TextStyle(
                    color: Colors.blue,
                    fontSize: 30,
                    )),
                WavyAnimatedText('Look at the waves',
                    textStyle: TextStyle(
                    color: Colors.green[600],
                    fontSize: 30,
                    )),
            ],
            repeatForever: true,
            onTap: () {
                print("Tap Event");
            },
            ),
        ),
        ],
    ),
    ));
```

```
}

}
```

Output:


This is Animated text.
Hel o World