Department of Information Technology

Academic Year: 2024-25
Semester: VI
Class / Branch / Div: TE- IT A/B/C
Subject: MAD & PWA Lab
Name of Instructor: Mrs. Sujata Oak

Name of Student:
Student ID:
Roll No.
Date of Submission:

Experiment No.:1

Aim: To install and configure Flutter Environment.

Theory:

❖ Flutter Installation

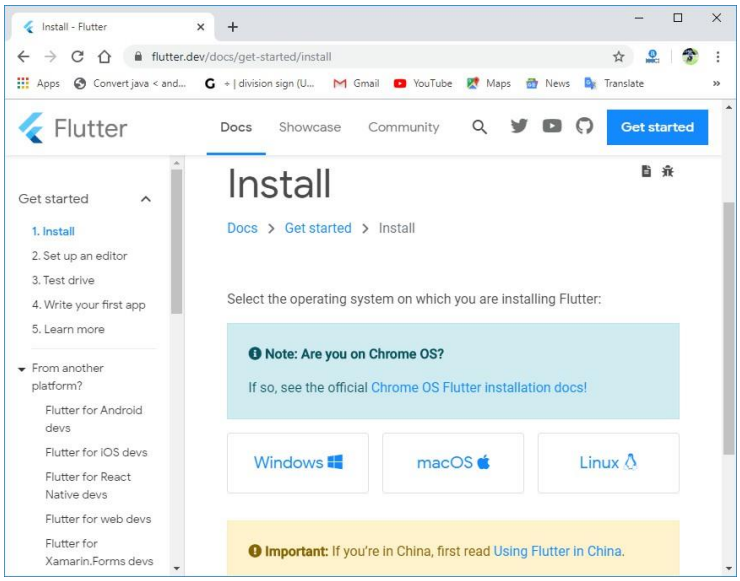
To install and run Flutter, your development environment must meet these minimum requirements:

- Operating Systems: Linux (64-bit)
- Disk Space: 600 MB (does not include disk space for IDE/tools).
- Tools: Flutter depends on these command-line tools being available in your environment.
  - bash
  - curl
  - file
  - git 2.x
  - mkdir
  - rm
  - unzip
  - which
  - xz-utils
  - zip
- Shared libraries: Flutter test command depends on this library being available in your environment.
  - libGLU.so.1 - provided by mesa packages such as libglu1-mesa on Ubuntu/Debian and mesa-libGLU on Fedora.

❖ Install the Flutter SDK

Step 1: Download the installation bundle of the Flutter Software Development Kit for Linux. To download Flutter SDK, Go to its official website

click on **Get started** button, you will get the following screen.



Step 2: Next, to download the latest Flutter SDK, click on the Linux **icon**. Here, you will find the download link for SDK

Install Flutter using snapd

Step 3: The easiest way to install Flutter on Linux is by using snapd. For more information, see Installing snapd.

Once you have snapd, you can install Flutter using the Snap Store, or at the command line:

$ sudo snap install flutter --classic

Note: Once the snap is installed, you can use the following command to display your Flutter SDK path:

$ flutter sdk-path

Step 4: **Run flutter doctor**

Run the following command to see if there are any dependencies you need to install to complete the setup (for verbose output, add the -v flag):

$ flutter doctor

This command checks your environment and displays a report to the terminal window. The Dart SDK is bundled with Flutter; it is not necessary to install Dart separately. Check the output carefully for other software you might need to install or further tasks to perform (shown in **bold** text).

Step 5: Update your path

You can update your PATH variable for the current session at the command line, as shown in Get the Flutter SDK. You'll probably want to update this variable permanently, so you can run flutter commands in any terminal session.

In some cases, your distribution may not permanently acquire the path when using the above directions. When this occurs, you can change the environment variables file directly. These instructions require administrator privileges:

Locate the etc directory at the root of the system, and open the profile file with root privileges.

sudo nano /etc/profile

1. Update the PATH string with the location of your Flutter SDK directory.

   content_copy

   ```
   if [ "`id -u`" -eq 0 ]; then
     PATH="..."
   else
     PATH="/usr/local/bin:...:[PATH_OF_FLUTTER_GIT_DIRECTORY]/bin"
   fi
   export PATH
   ```

2. End the current session or reboot your system.
3. Once you start a new session, verify that the flutter command is available by running:

$ which flutter

Step 6: Android setup

Installing Java OpenJDK

To install Android Studio we require OpenJDK version 8 or above to be installed in our system. First, we will install OpenJDK 8. The installation is very simple, we can start by updating the package index:

$ sudo apt update

Now install the OpenJDK 8 package using the below command

$ sudo apt install openjdk-8-jdk

To verify the installation, we can use the below command to check the version:

$ java -version

Installing Android Studio

To download and install the Android Studio snap package, we have to open our terminal using the Ctrl+Alt+T keyboard shortcut and type the below command:

$ sudo snap install android-studio --classic

When the installation gets completed, we will see the following output:

android-studio 3.3.1.0 from Snapcrafters installed

Now Android Studio has been installed on our Ubuntu desktop.

Starting Android Studio

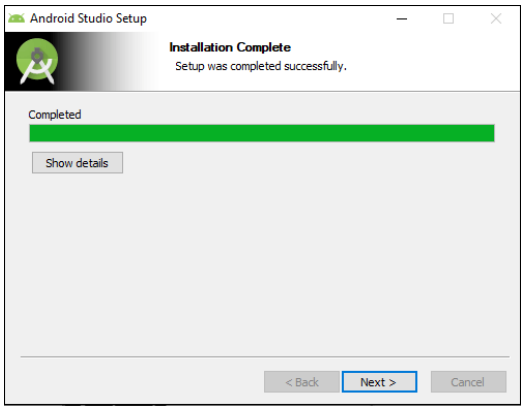
Once the Android Studio is installed in the system using any of the above methods, we can start Android Studio through our terminal by typing android-studio in our terminal or by clicking on the Android Studio icon. When we start Android Studio for the first time, the below window will appear asking us to import Android Studio settings from a previous installation: When we click on the OK button, the Setup Wizard window will appear. We have to click on the Next button to start the initial configuration and post-installation steps. Now option to choose the type of setup we want for Android studio will appear. We can select the "Standard" option or "Custom" option if we want to customize our Android Studio.

Step 7: Install the Android SDK. If the flutter doctor command does not find the Android SDK tool in your system, then you need first to install the Android Studio IDE. To install Android Studio IDE, do the following steps.

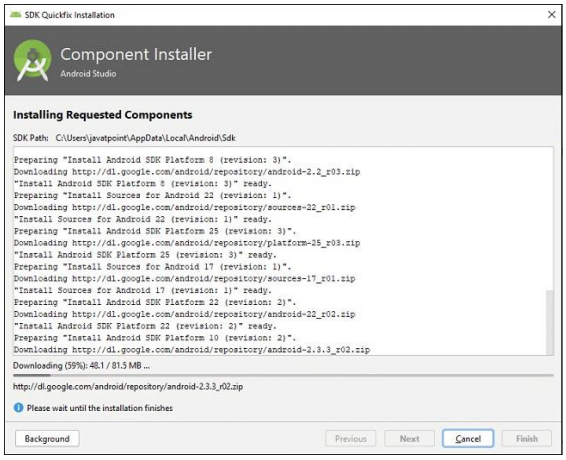
Step 7.1: Download the latest Android Studio executable or zip file from the official site

Step 7.2: When the download is complete, open the **.exe** file and run it. You will get the following dialog box.

Step 7.3: Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.

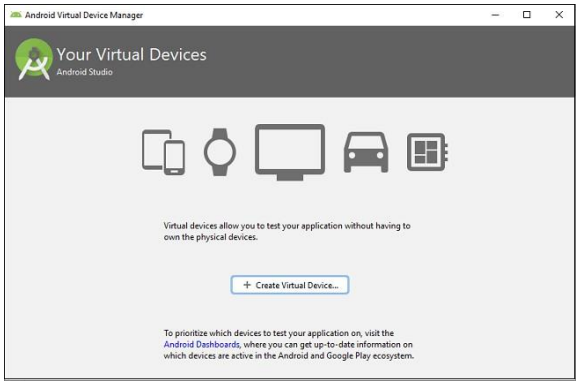


Step 7.4: In the above screen, click Next-> Finish. Once the Finish button is clicked, you need to choose the 'Don't import Settings option' and click OK. It will start the Android Studio.



Step 8: Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application.

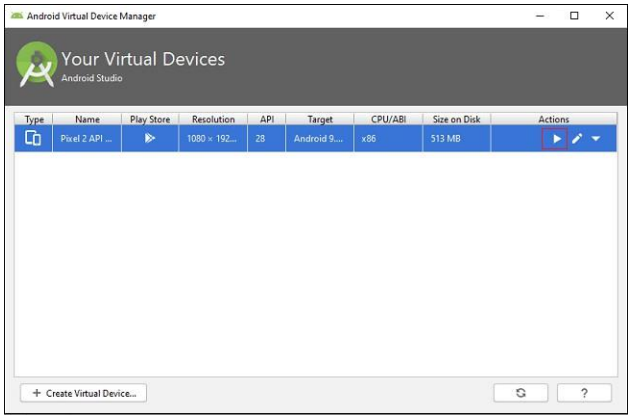
Step 8.1: To set an Android emulator, go to Android Studio > Tools > Android > AVD Manager and select Create Virtual Device. Or, go to Help->Find Action->Type Emulator in the search box. You will get the following screen.



Step 8.2: Choose your device definition and click on Next.

Step 8.3: Select the system image for the latest Android version and click on Next.

Step 8.4: Now, verify the all AVD configuration. If it is correct, click on Finish. The following screen appears.

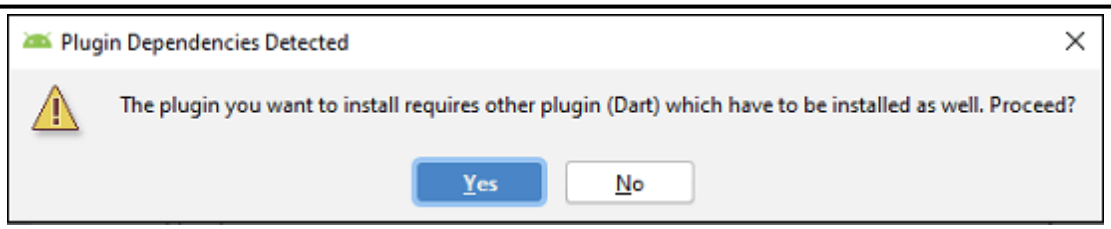


Step 8.5: Last, click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen.

Step 9: Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself. Do the following steps to install these plugins.

Step 9.1: Open the Android Studio and then go to File->Settings->Plugins.

Step 9.2: Now, search the Flutter plugin. If found, select Flutter plugin and click install. When you click on install, it will ask you to install Dart plugin as below screen. Click yes to proceed.



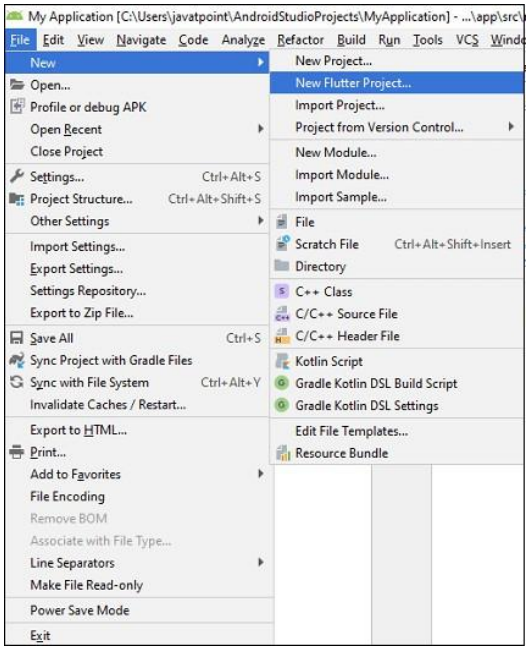
Step 9.3: Restart the Android Studio.

Flutter First Application

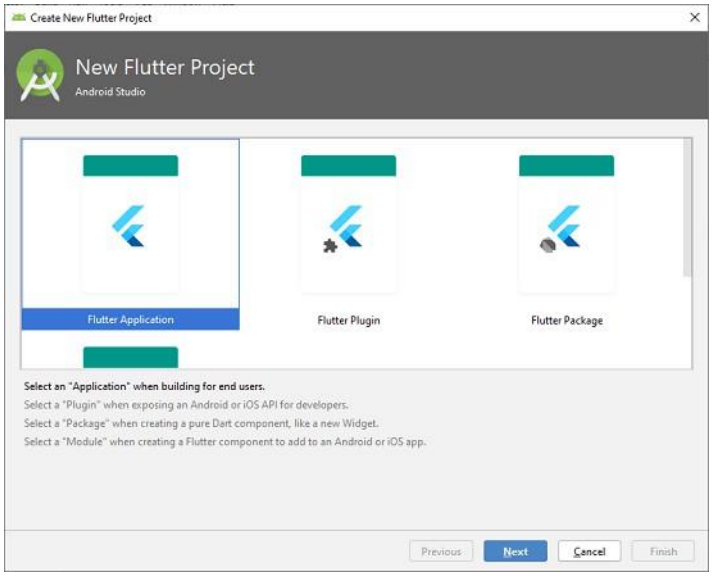
In this section, we are going to learn how to create a simple application in Android Studio to understand the basics of the Flutter application. To create Flutter application, do the following steps:

Step 1: Open the Android Studio.

Step 2: Create the Flutter project. To create a project, go to File-> New->New Flutter Project. The following screen helps to understand it more clearly.



Step 3: In the next wizard, you need to choose the Flutter Application. For this, select Flutter Application-> click Next, as shown in the below screen.



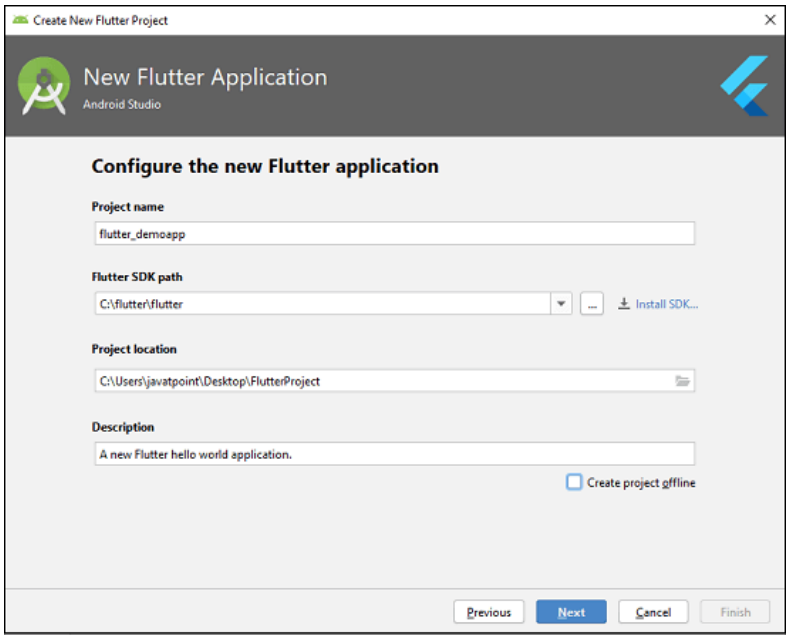
Step 4: Next, configure the application details as shown in the below screen and click on the Next button.

Project Name: Write your Application Name.

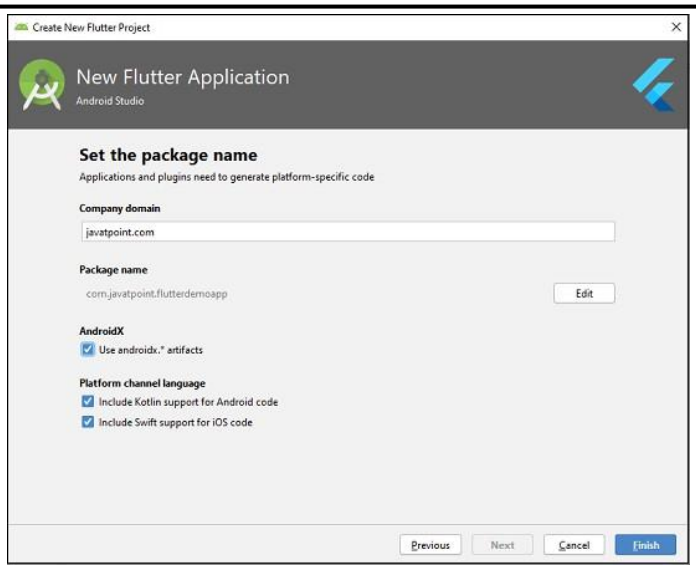
Flutter SDK Path: <path_to_flutter_sdk>

Project Location: <path_to_project_folder>

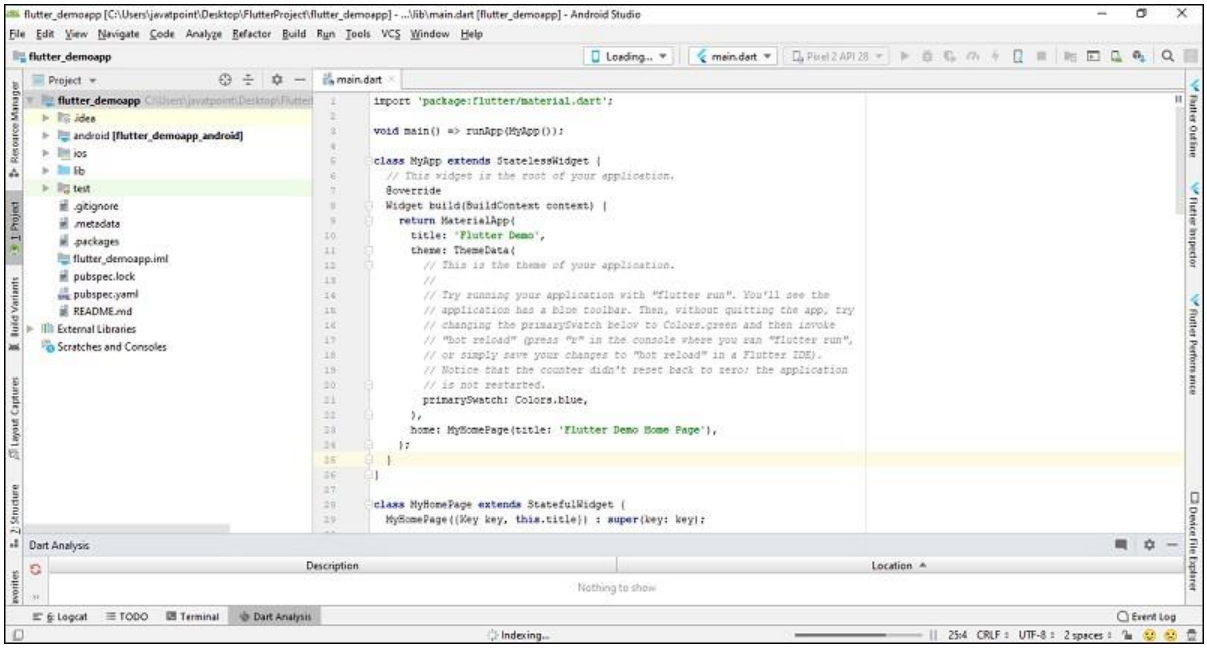
Descriptions: <A new Flutter hello world application>.



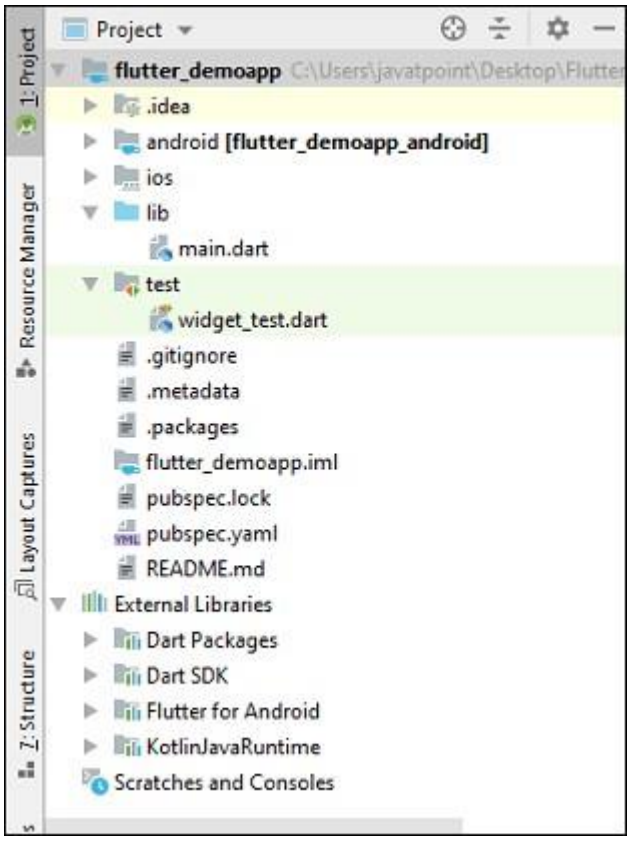
Step 5: In the next wizard, you need to set the company domain name and click the Finish button.



After clicking the Finish button, it will take some time to create a project. When the project is created, you will get a fully working Flutter application with minimal functionality.



Step 6: Now, let us check the structure of the Flutter project application and its purpose. In the below image, you can see the various folders and components of the Flutter application structure, which are going to discuss here.



Step 7: Open the **main.dart** file and replace the code with the following code snippets.

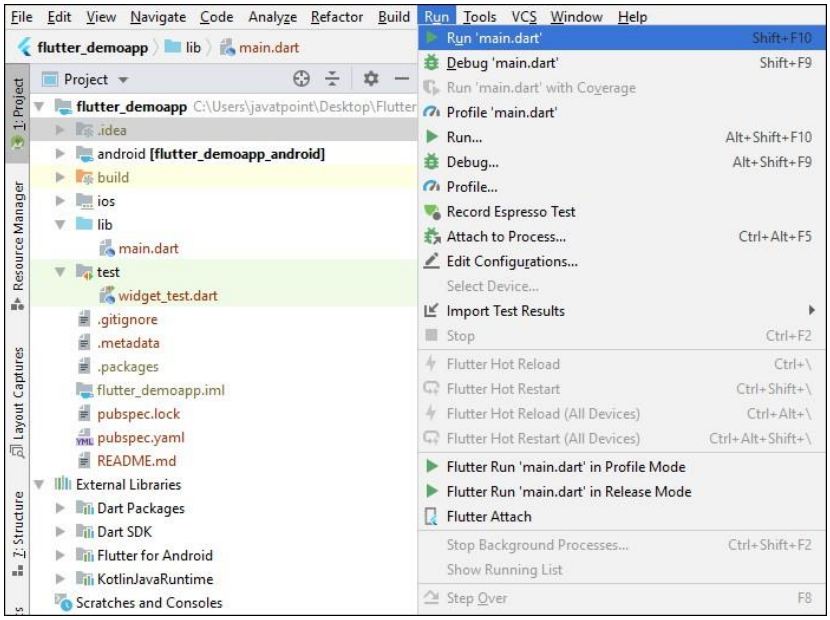
```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
          appBar: AppBar(title: Text("Hello World Flutter Application")),
          body: Center(
              child: Text("Hello, World!", style: TextStyle(fontSize: 24,fontStyle:
FontStyle.italic,fontWeight: FontWeight.bold),
              )
          )
      ),
    );
  }
}
```

Step 8: Let us understand the above code snippet line by line.

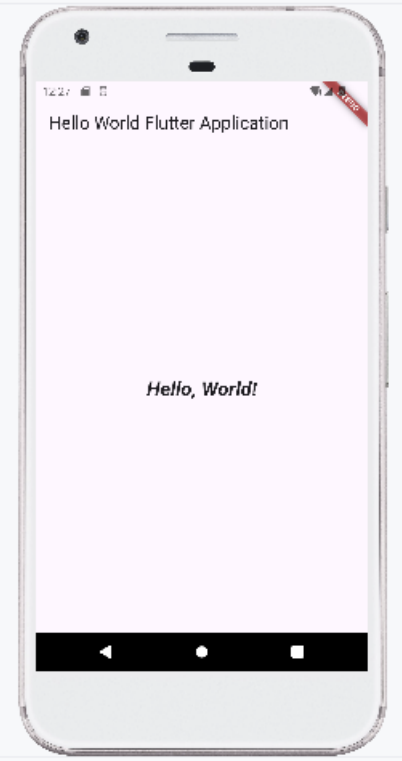
- To start Flutter programming, you need first to import the Flutter package. Here, we have imported a **Material package**. This package allows you to create user interface according to the Material design guidelines specified by Android.
- The second line is an entry point of the Flutter applications similar to the main method in other programming languages. It calls the **runApp** function and pass it an object of **MyApp** The primary purpose of this function is to attach the given widget to the screen.
- Line 5 to 18 is a widget used for creating UI in the Flutter framework. Here, the **StatelessWidget** does not maintain any state of the widget. MyApp extends StatelessWidget that overrides its **build** The build method is used for creating a part of the UI of the application. In this block, the build method uses MaterialApp, a widget to create the root level UI of the application and contains three properties - title, theme, and home.
    1. **Title:** It is the title of the Flutter application.
    2. **Theme:** It is the theme of the widget. By default, it set the blue as the overall color of the application.
    3. **Home:** It is the inner UI of the application, which sets another widget (MyHomePage) for the application.
- Line 19 to 35, the **MyHomePage** is similar to MyApp, except it will return the **Scaffold** Scaffold widget is a top-level widget after the MaterialApp widget for creating the user interface. This widget contains two properties **appBar** and **body**.

The appBar shows the header of the app, and body property shows the actual content of the application. Here, **AppBar** render the header of the application, **Center** widget is used to center the child widget, and **Text** is the final widget used to show the text content and displays in the center of the screen.

Step 9: Now, run the application. To do this, go to Run->Run main.dart, as shown in the below screen.



Step 10: Finally, you will get the output as below screen.



Conclusion: In this experiment we have created complete environment to execute flutter applications and display Hello world.