LabW10 – Flutter Tutorial

Objectives:

1. Understand Flutter development with Android Studio
2. Understand Flutter project structure and widgets
3. Learn developing basic flutter application

Tasks:

1. Setup Flutter with Android Studio
2. Create first Flutter Project
3. Create your Flutter To-Do Application

Flutter is an open-source framework to create high quality, high performance mobile applications across mobile operating systems - Android and iOS. It provides a simple, powerful, efficient, and easy to understand SDK to write mobile application in Google's own language, *Dart*.

This tutorial walks through the basics of Flutter framework, installation of Flutter SDK, setting up Android Studio to develop Flutter based application, architecture of Flutter framework and developing all type of mobile applications using Flutter framework.

Task 1: Setup Flutter with Android Studio

1. **Installation in Windows:** For installing flutter on the windows devices, please follow the steps below

   Step 1 − Go to URL, https://flutter.dev/docs/get-started/install/windows and download the latest Flutter SDK. As of October 2021, the version is 3.13.6 and the file is flutter_windows_3.13.6-stable.zip.

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

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

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

   ```
   flutter doctor
   ```

   Step 5 − Running the above command will analyse the system and show its report as shown below –

   ```
   Doctor summary (to see all details, run flutter doctor -v):
   [√] Flutter (Channel stable, 2.5.2 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 1.60.2)
   [!] Connected device
   ! No devices available
   ! Doctor found issues in 1 category.
   ```

   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 startup 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.

2. **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 October 2022, the version is 3.13.6 and the file is flutter_macos_3.13.6- stable.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.

```
flutter doctor
```

```
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 like 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

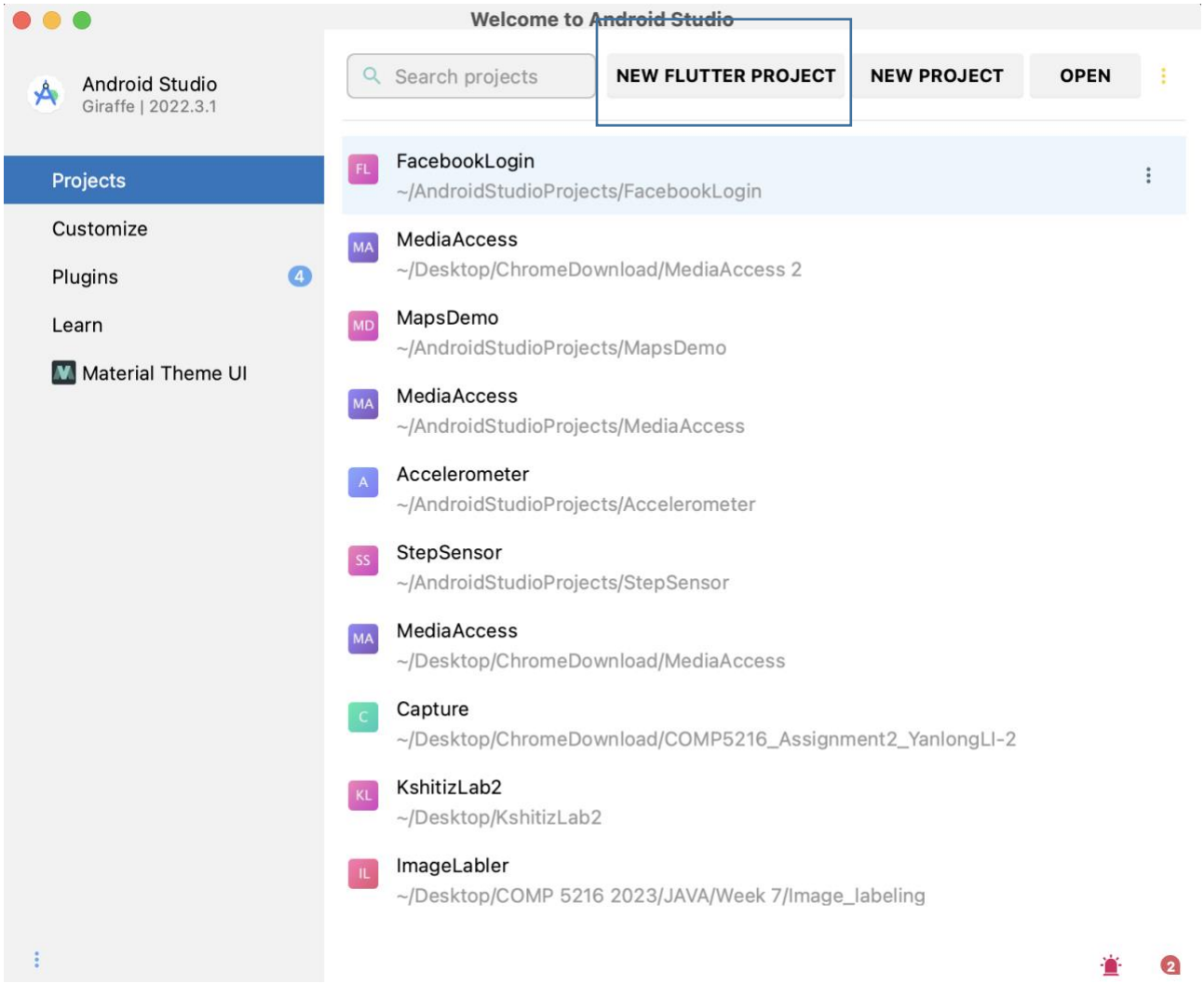
Step 8 − Start an android emulator or connect a real android device to the system to develop android application.

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.

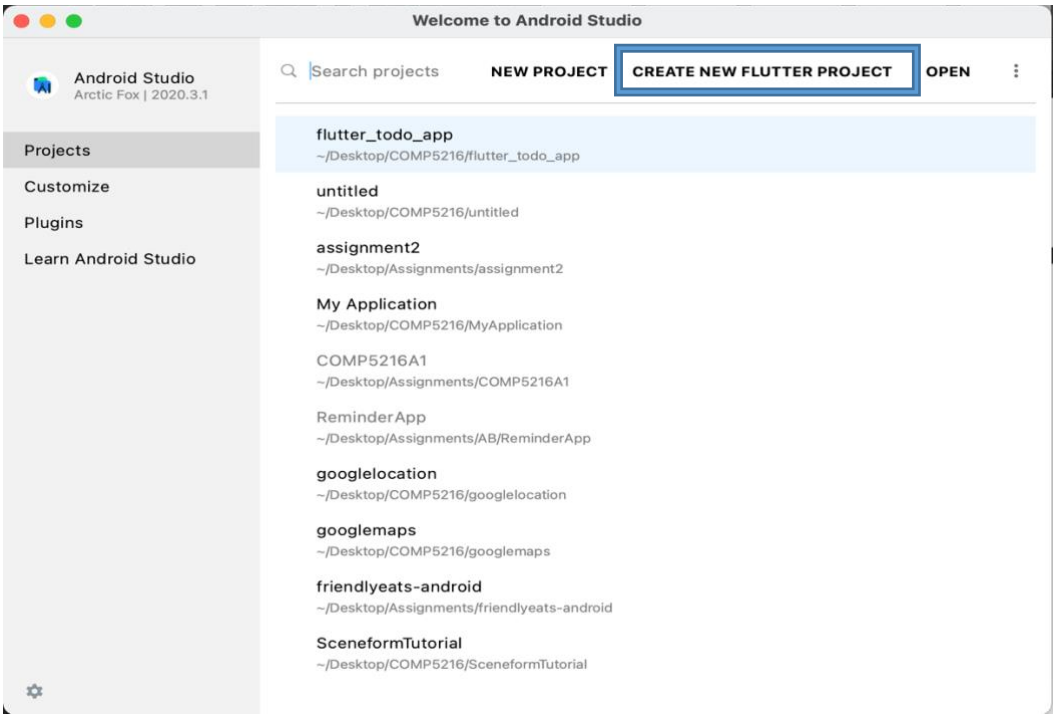
Note: If Flutter is setup correctly, we should be able to see "**New Flutter Project**" option on the Android Studio launch screen.



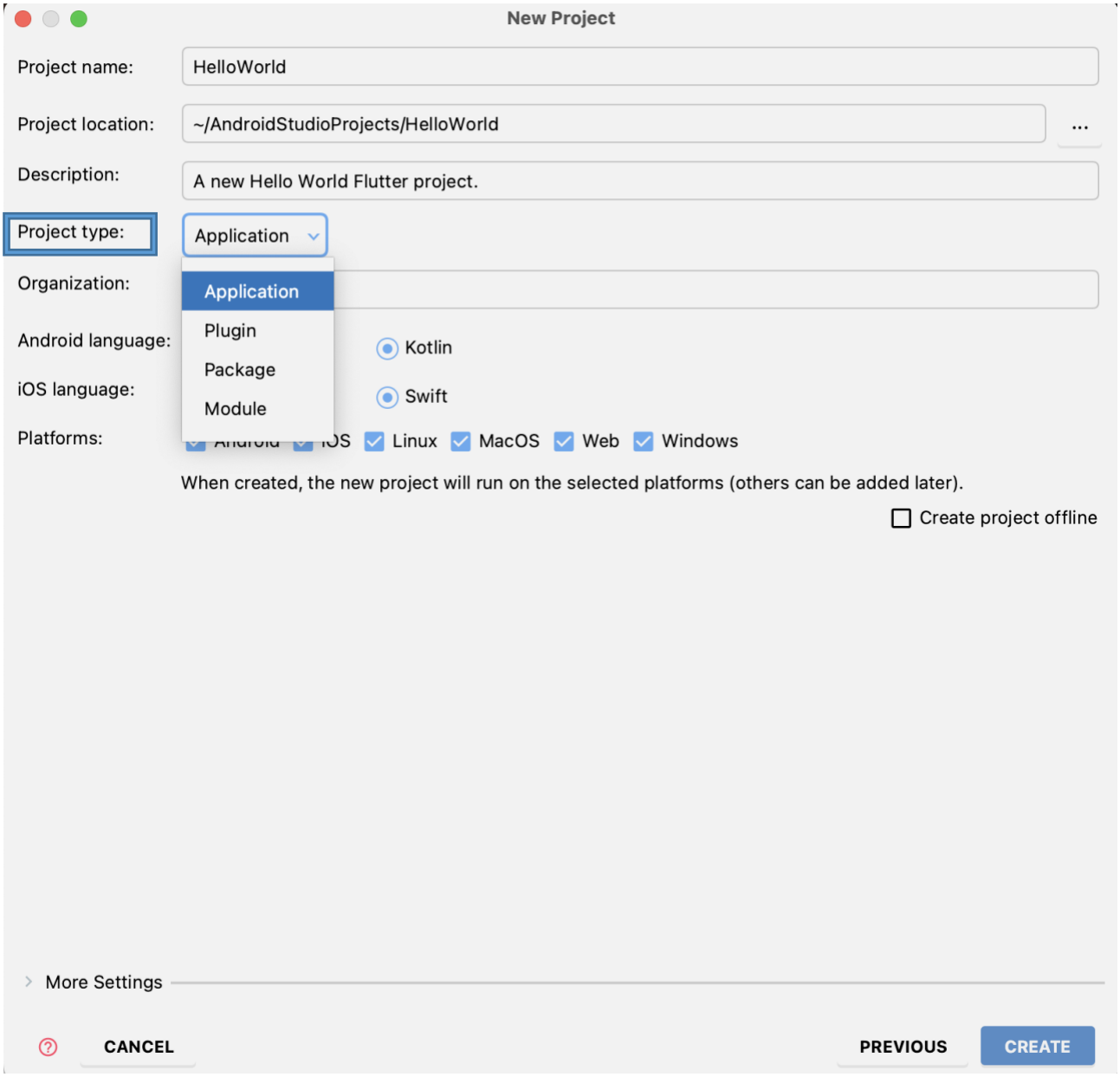
Task 2: Create first Flutter Project

Step 1 − Launch Android Studio

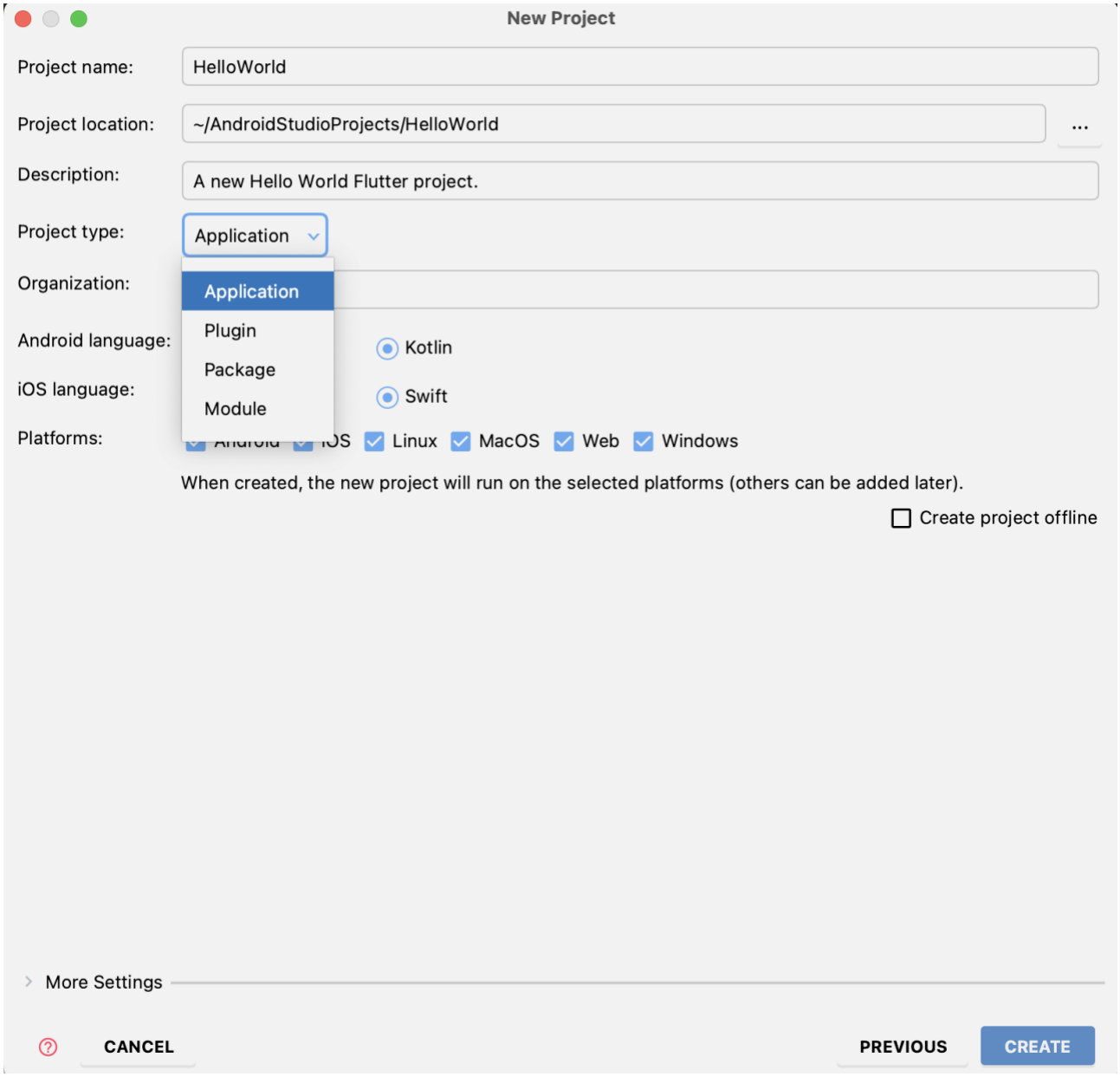
Step 2 – Click on **"CREATE NEW FLUTTER PROJECT"**



Step 3 – Select **"Application"** from the project type and set the flutter SDK path set in **Task 1**



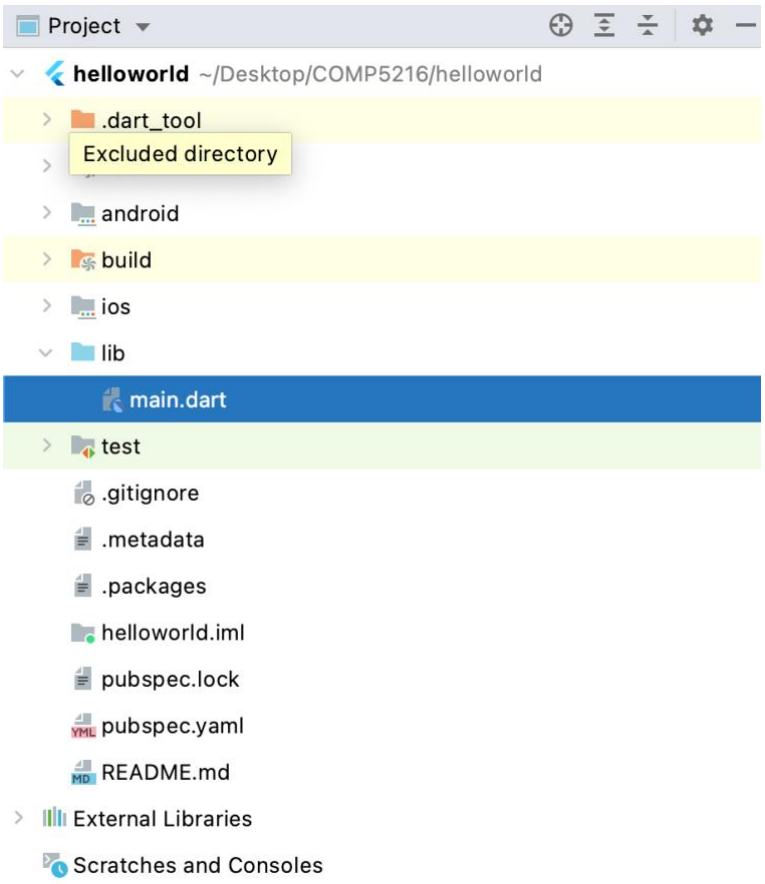
Step 4 – On the same screen add details related to the project like the name of the application "**helloworld**", location **"workspace/directory"**, Android language "**JAVA/Kotlin**", iOS language "**objective-C/Swift"**, description, etc. and click **Create**.



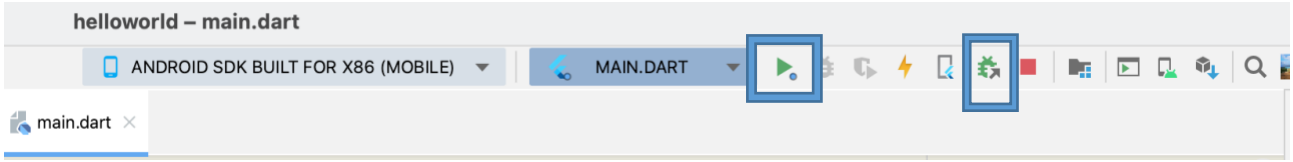
Step 5 – Expand the project structure to analyse various files in the flutter project. Various components of the structure of the application are explained here −

- **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
- **ib/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

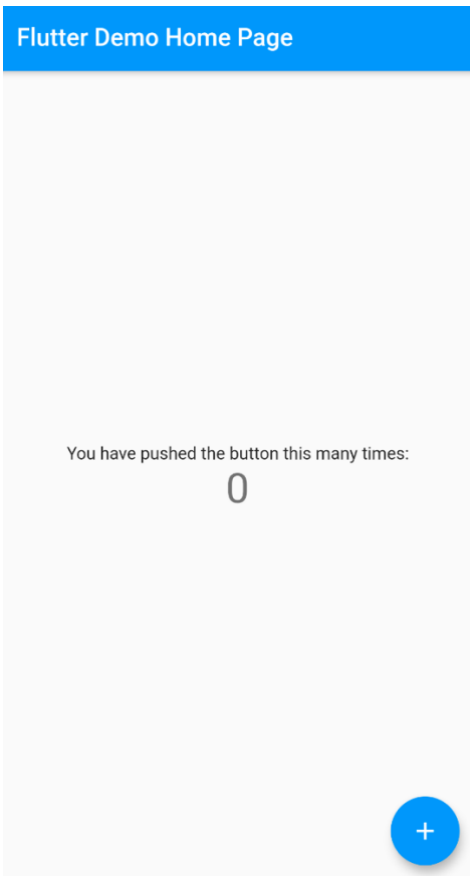
- **.metadata** − auto generated by the flutter tools
- **.packages** − auto generated to track the flutter packages
- **.iml** − project file used by Android studio
- **pubspec.yaml** − Used by **Pub**, Flutter package manager
- **pubspec.lock** − Auto generated by the Flutter package manager, **Pub**
- **README.md** − Project description file written in Markdown format



Step 6 – Select the emulator and click run button to start the application. To debug the application, add breakpoints at the required places and click on debug button.



Step 7 – Emulator should launch an application as in screenshot below. On clicking the "+" floating button count of the counter should increase.



Task 3: Create your Flutter To-Do Application

Step 1 - Open Android Studio and click on the **New Flutter Project** option (Same as Task 2).

Step 2 – Select **App**lication as project type and set the Flutter SDK path, followed by clicking on **next** button (Same as Task 2).

Step 3 – Add Project name: "**flutter_todo_app",** location:" your workspace", description: "**simple todo list app**", Android language: "**JAVA**", iOS language: "**Swift**", etc. and click **create** button

Step 4 - Go to **lib** → **main.dart** File. Download the "**main.dart"** file from canvas and replace the code.

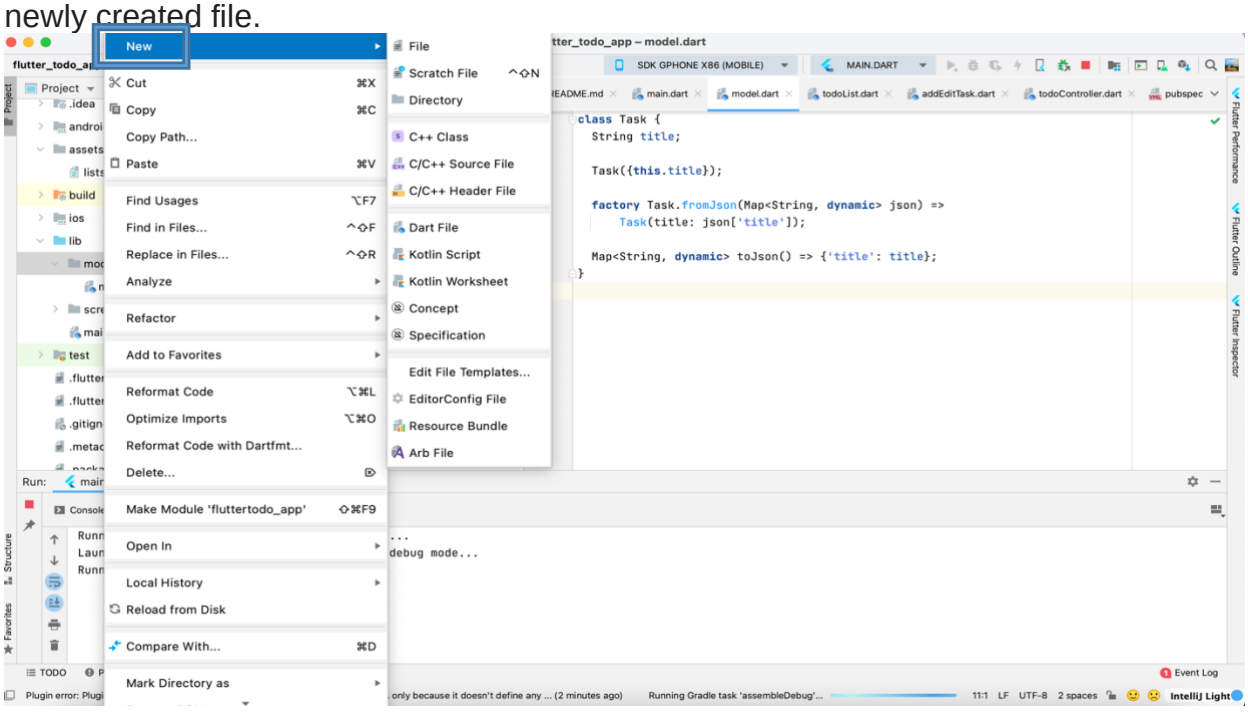
```dart
import 'package:flutter/material.dart';
import 'package:flutter_todo_app/screens/todoList.dart';
import 'package:get/get.dart';
import 'package:get_storage/get_storage.dart';

void main() async {
  await GetStorage.init();
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.red,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),  // ThemeData
      home: TodoList(),
    );  // GetMaterialApp
  }
}
```

Explanation for the code: Since we are going to use the navigation system of GETX and other alerts and snackbar that's why we have converted MaterialApp to GetMaterialApp and we have provided our home screen as TodoList().

Step 5 - Create a new folder Model and add a new dart file by **right clicking** on the folder and select **new option -> dart file**. Copy the code from **model.dart** from canvas to the newly created file.

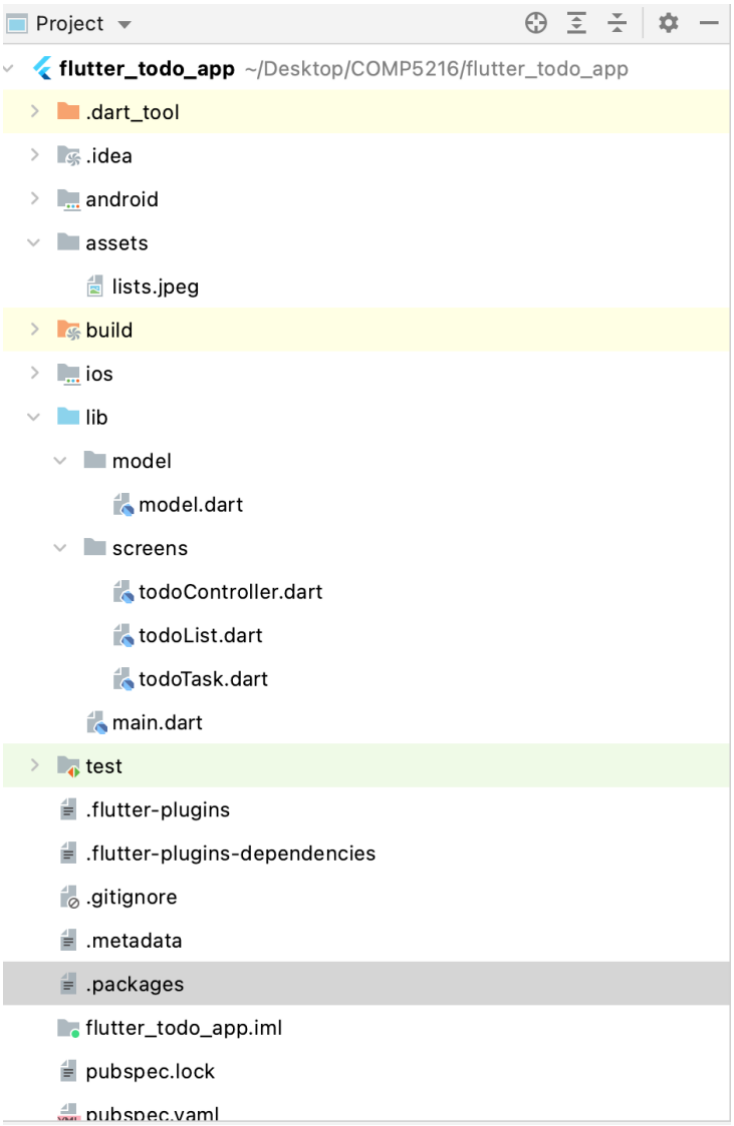


Explanation: The model.dart file creates the models of our TODO application. We will be only storing title as Task. So, we have created a class Task and provided a String title as parameter.

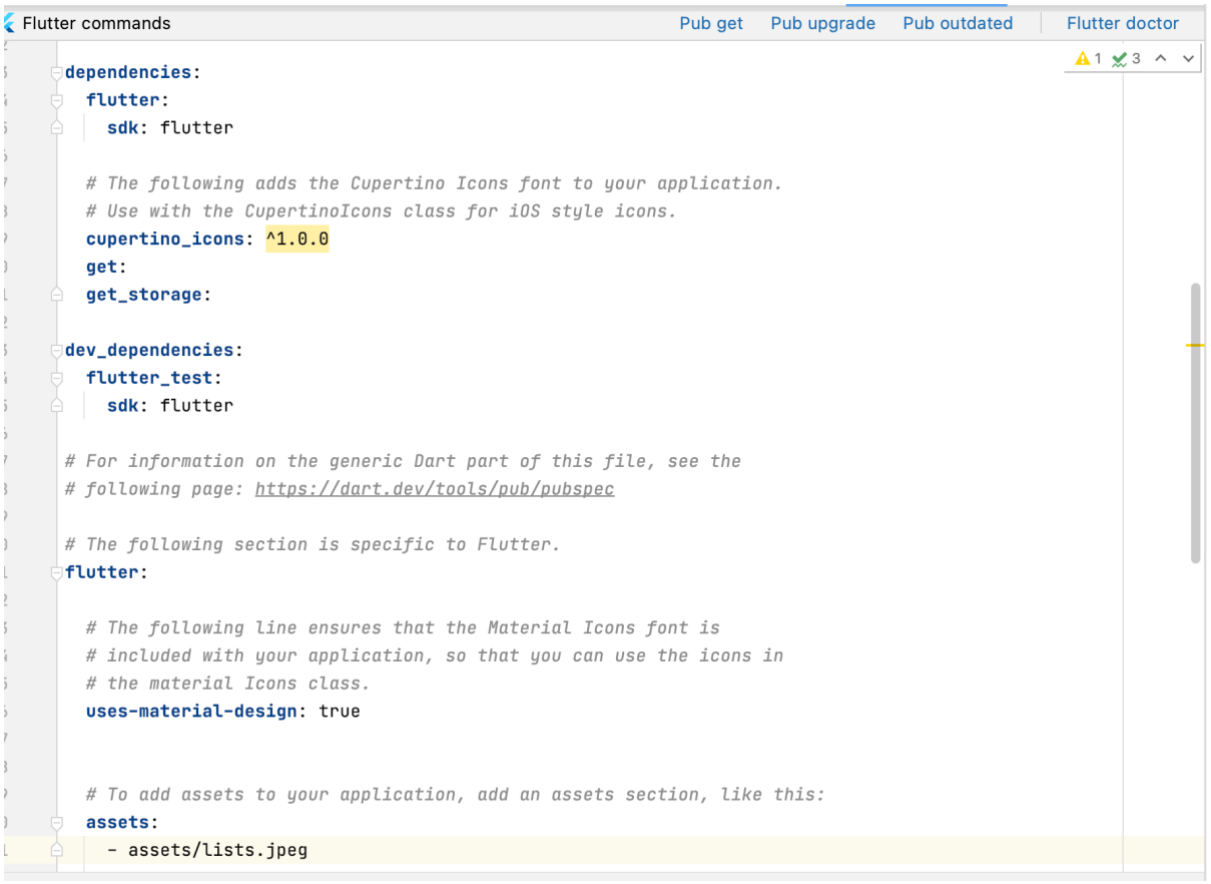
Step 6 – Create a new folder "**screens"** and copy the files "**todoController.dart**", "**todoList.dart**", and "**addEditTask.dart**" in it.

Explanation: **TodoController.dart** is responsible for handling the list retrieval and loading. **todoList.dart** will be holding the logic to create the todo list view with some child widgets. **addEditTask.dart** will create a view with textbox and buttons to update and add new tasks.

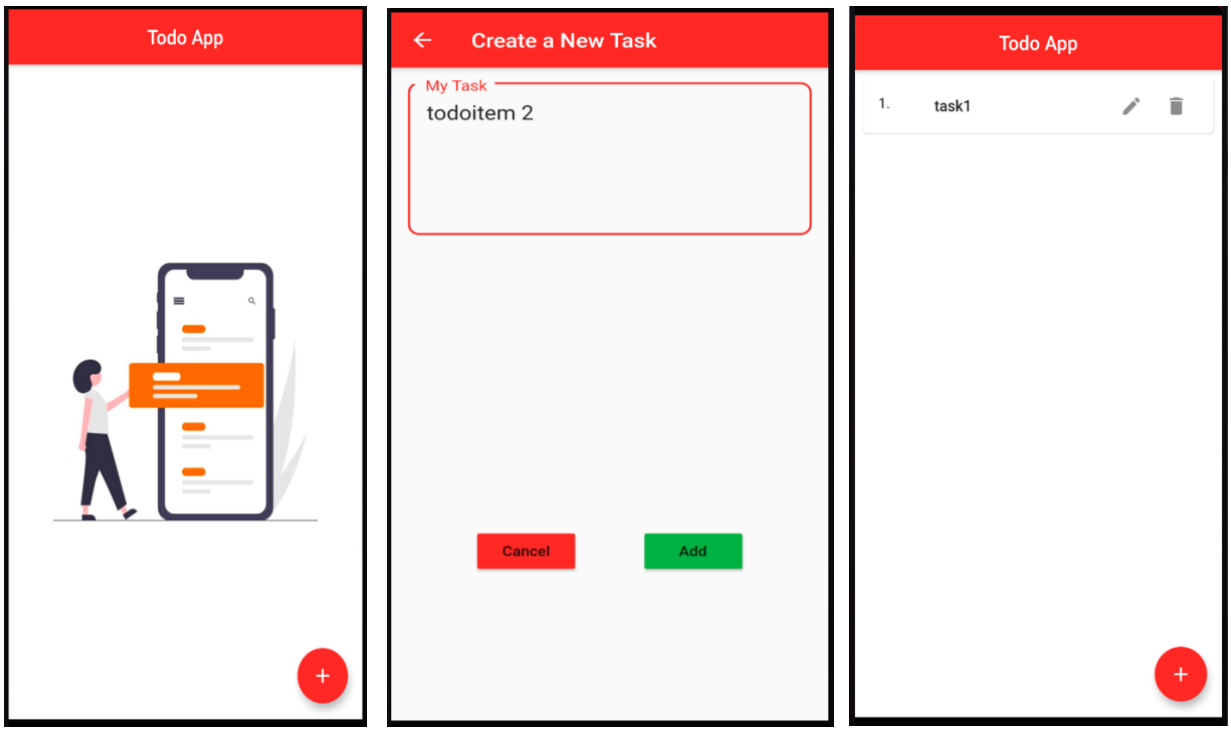
Step 7 – Create a new folder "assets" and copy the "**lists.jpeg**" file in it. Final folder structure should appear like the screenshot below.



Step 8 – Replace the **pubspec.yaml** file from canvas with project and click on **Pub get or Pub Upgrade** at the top of file



Step – 9 Select the emulator and click on run button, Mac users could start an iOS emulator and use them as well to run the application. On building the application final screen should appear like the screen shots below.


Todo App
Create a New Task
My Task
todoitem 2
Cancel
Add
Todo App
1.
task1

References

The tutorial is partially adopted based on the materials from Flutter Developer:
https://flutter.dev/docs/get-started/codelab
https://flutter.dev/docs/reference/tutorials

If you wish to set up Flutter app development environment, please refer to:
https://flutter.dev/docs/get-started/install/windows
https://flutter.dev/docs/get-started/install/macos