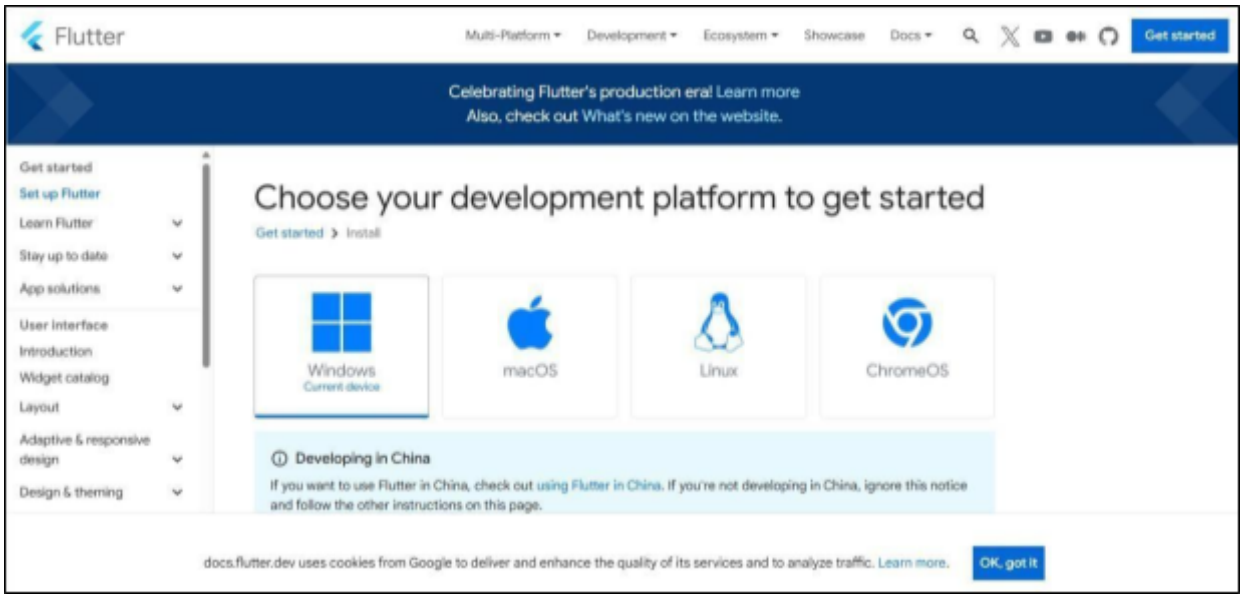
Project Title: Varsity **RollNo. 50**

MAD & PWA Lab Journal

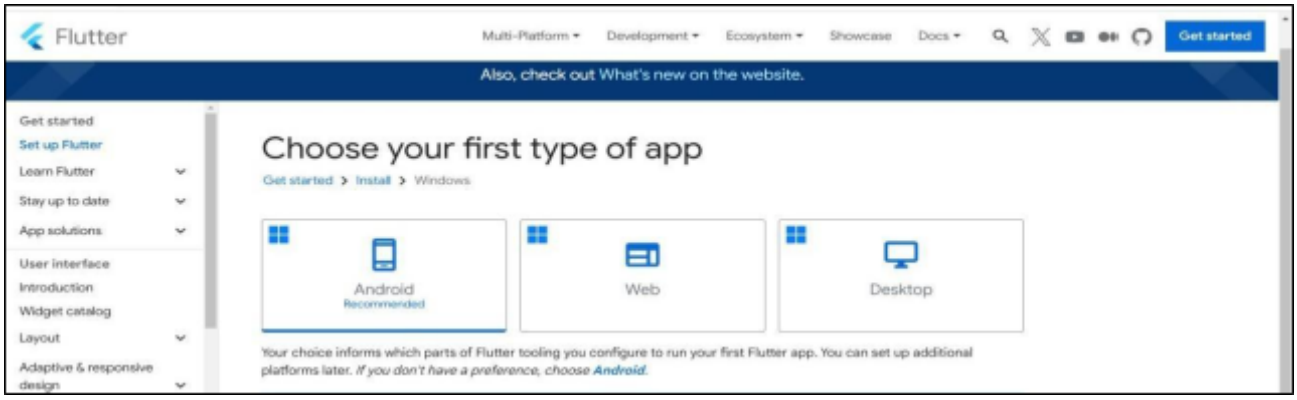
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	50
Name	Siddhant Sathe
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

AIM: - Installation and Configuration of Flutter Environment.

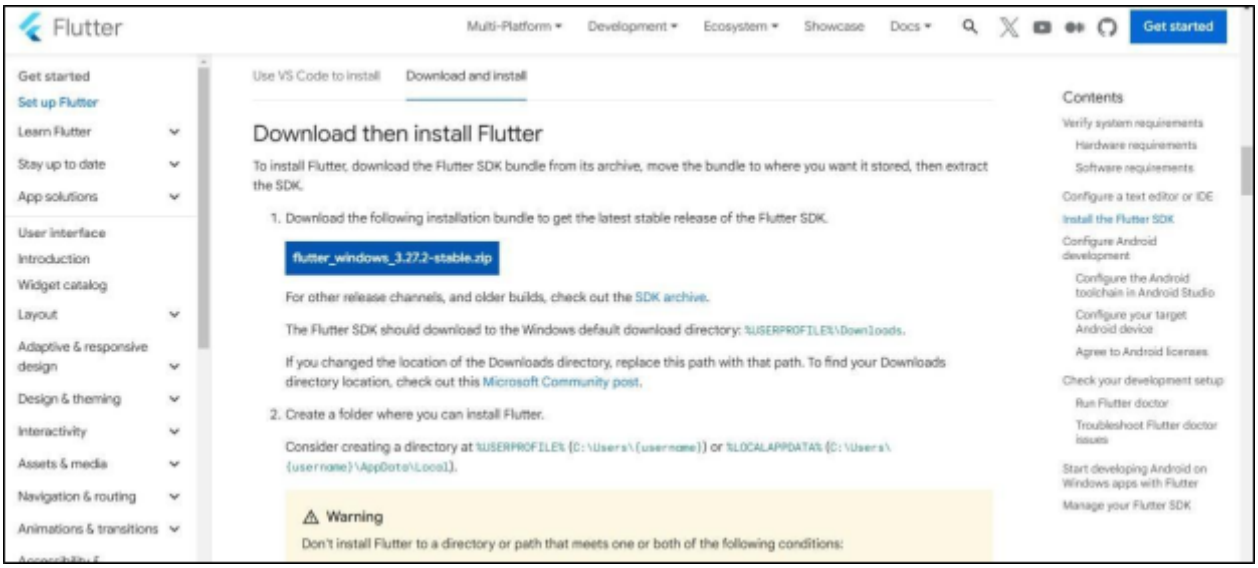
Step 1: Go to the official Flutter website: https://docs.flutter.dev/get-started/install



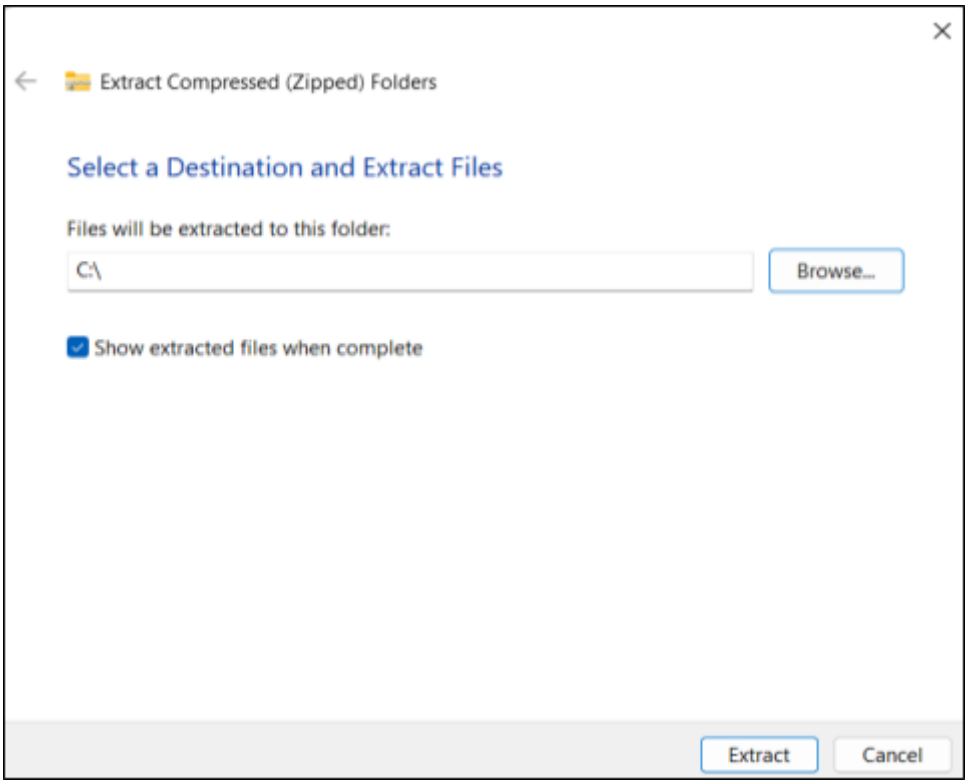
Step 2: To download the latest Flutter SDK, click on the Windows icon > Android



Step 3: For Windows, download the stable release (a .zip file).



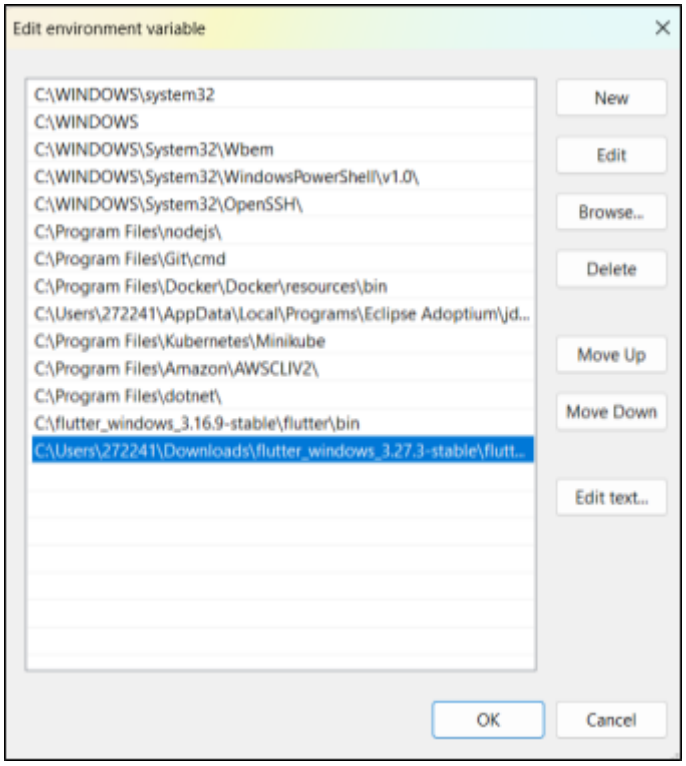
Step 4: Extract the ZIP file to a folder (e.g., C:\flutter).



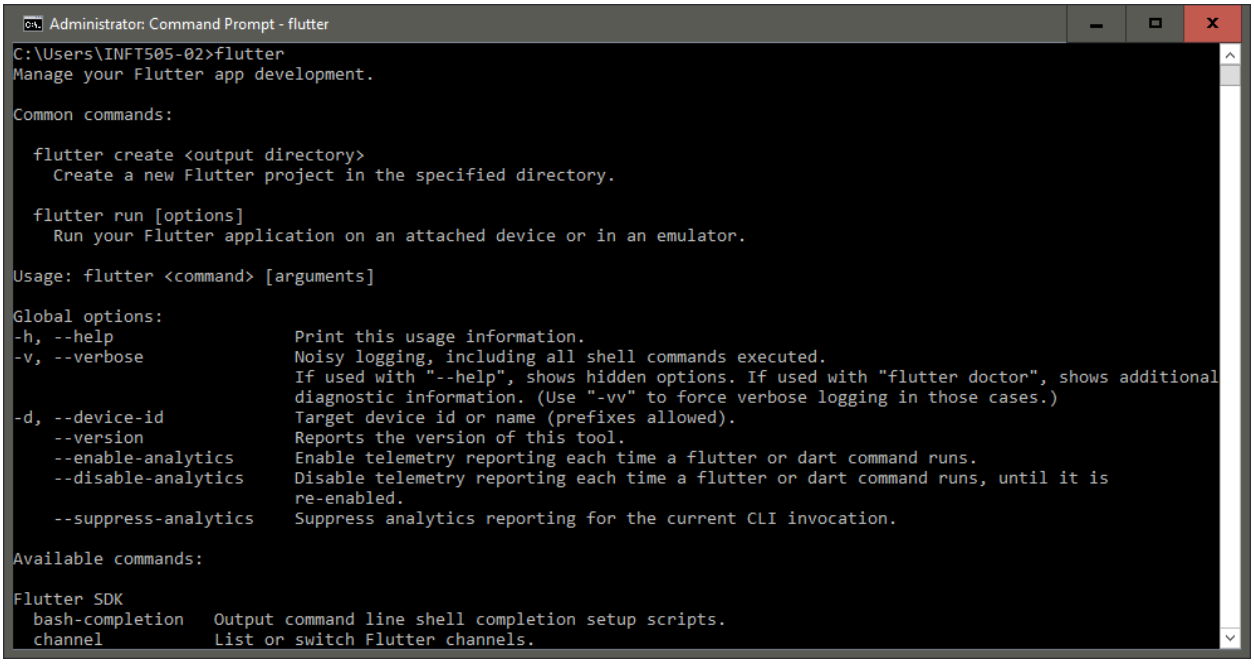
Step 5 :- Add Flutter to System PATH

Right-click on the Start Menu > System > Advanced system settings > Environment Variables. Under System Variables, find Path and click Edit.

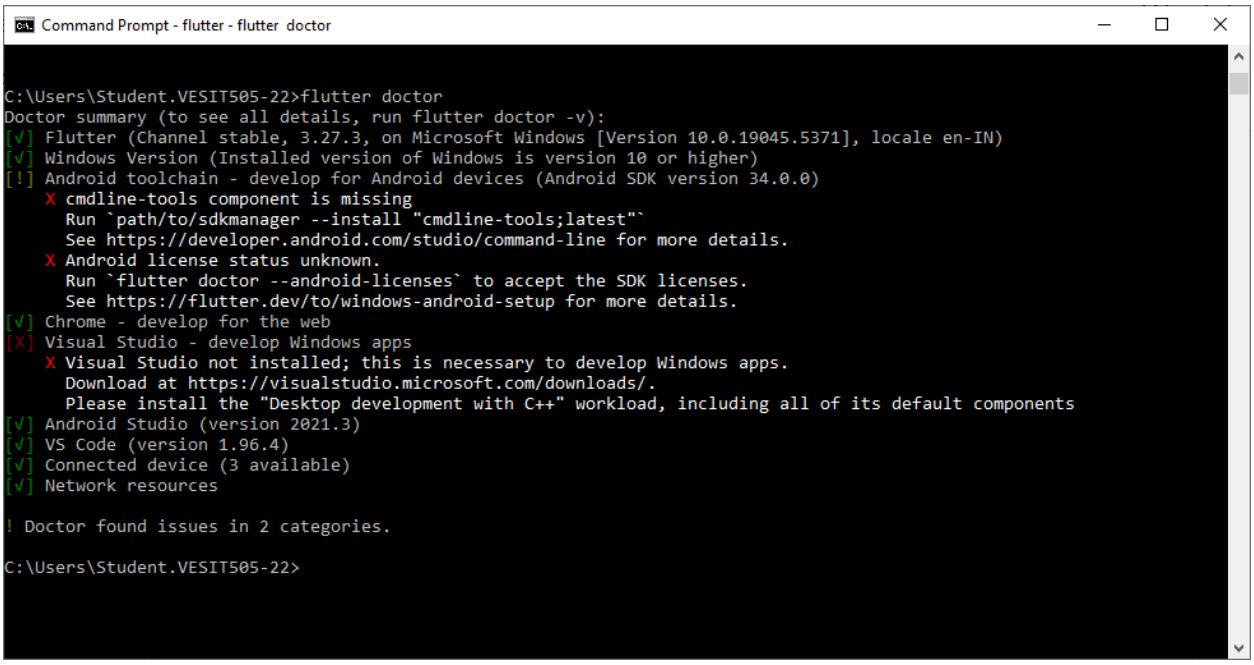
Add the full path to the flutter/bin directory (e.g., C:\flutter\bin).



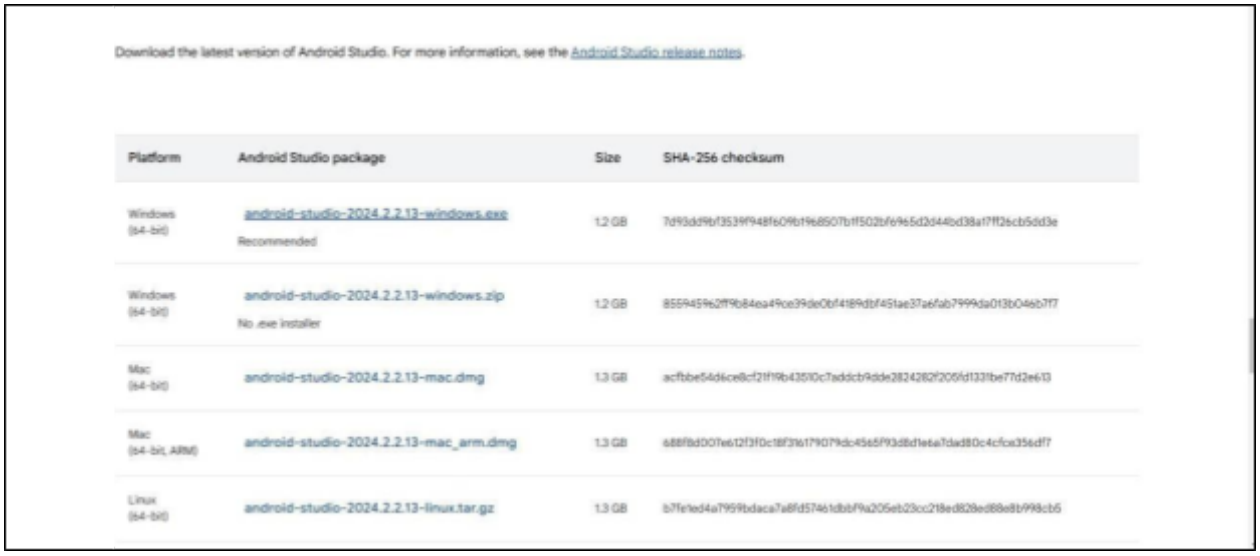
Step 6 : - Now, run the $ flutter command in command prompt.



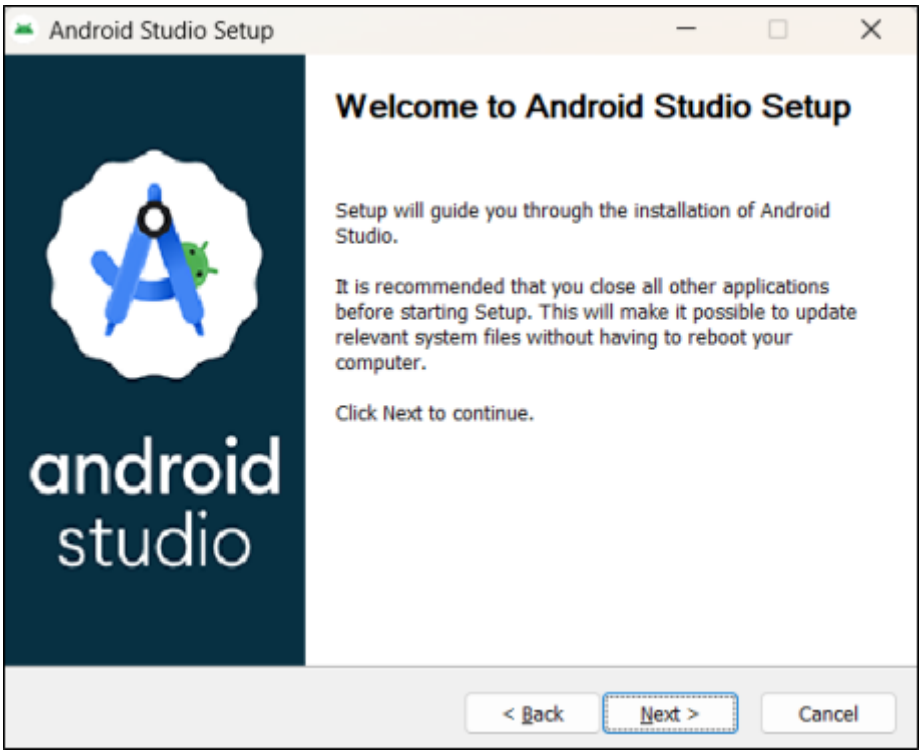
Step 7:- Run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation



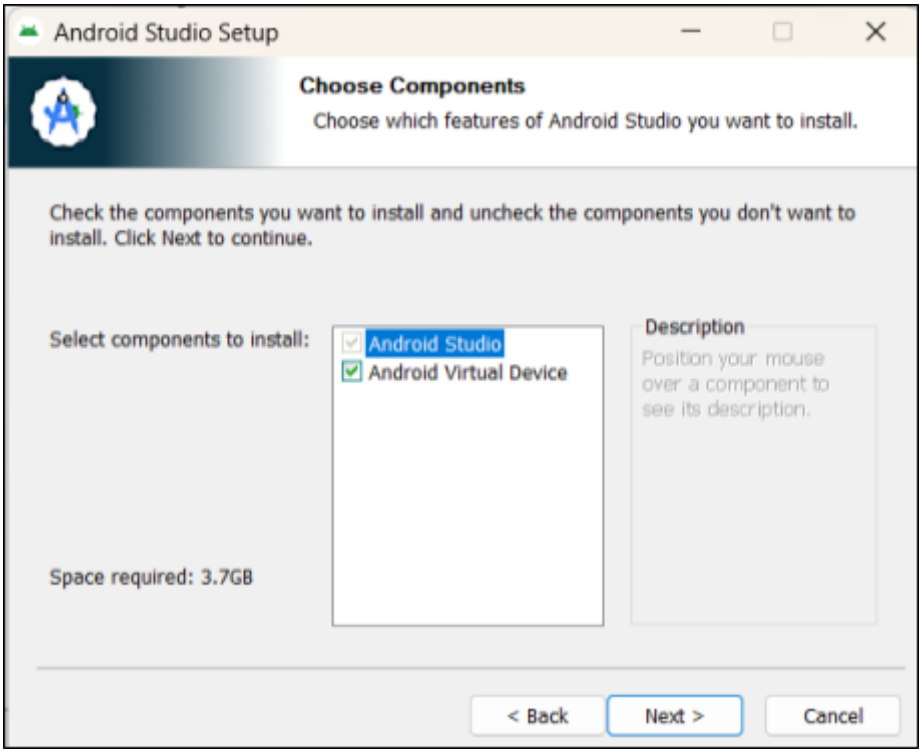
Step 8 : - Go to Android Studio and download the installer.



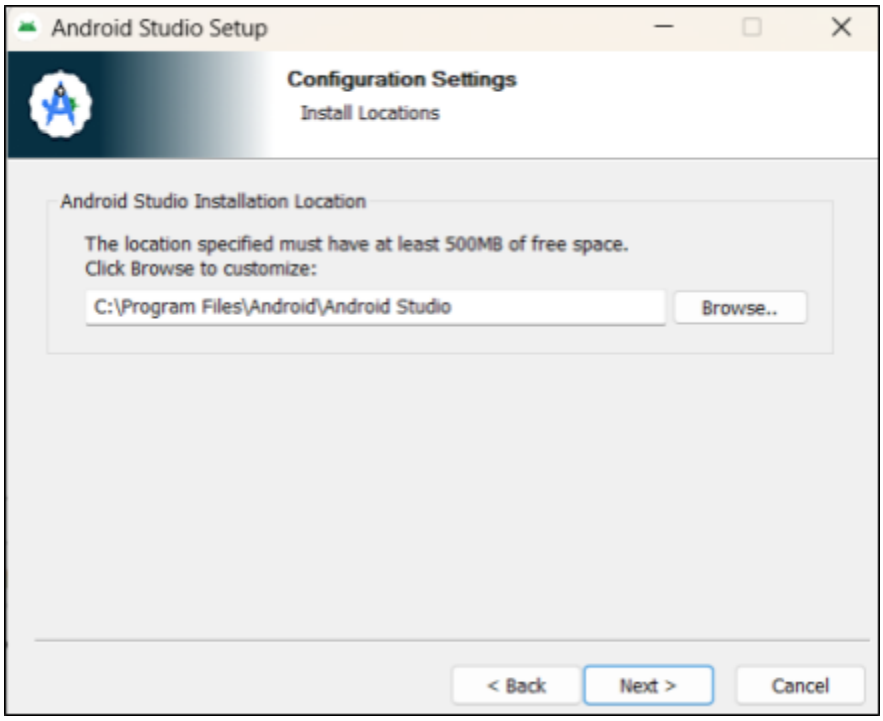
Step 8.1: - When the download is complete, open the .exe file and run it. You will get the following dialog box



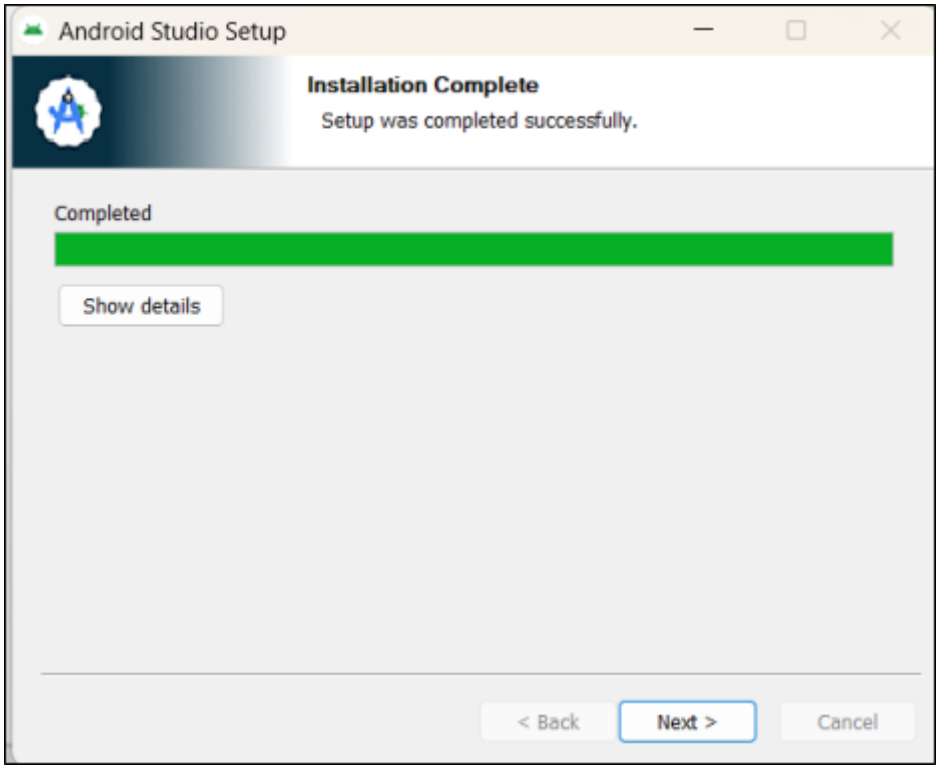
Step 8.2: - Select all the Checkboxes and Click on 'Next' Button.

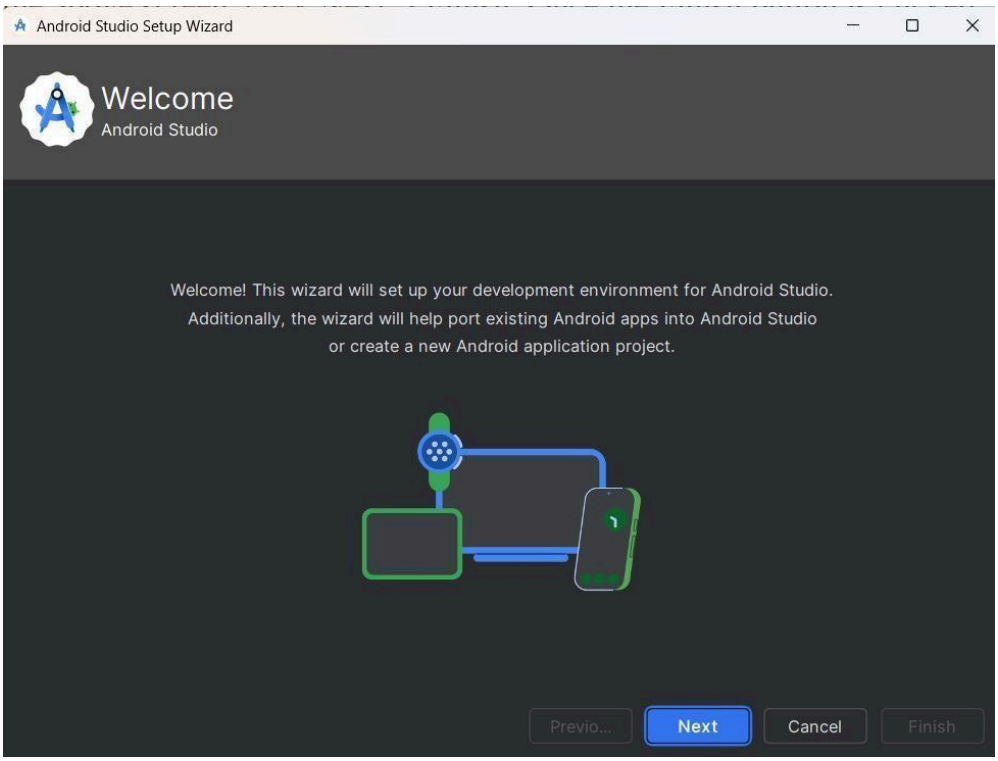


Step 8.3: - Change the destination as per your convenience and click on ‘Next’ Button.

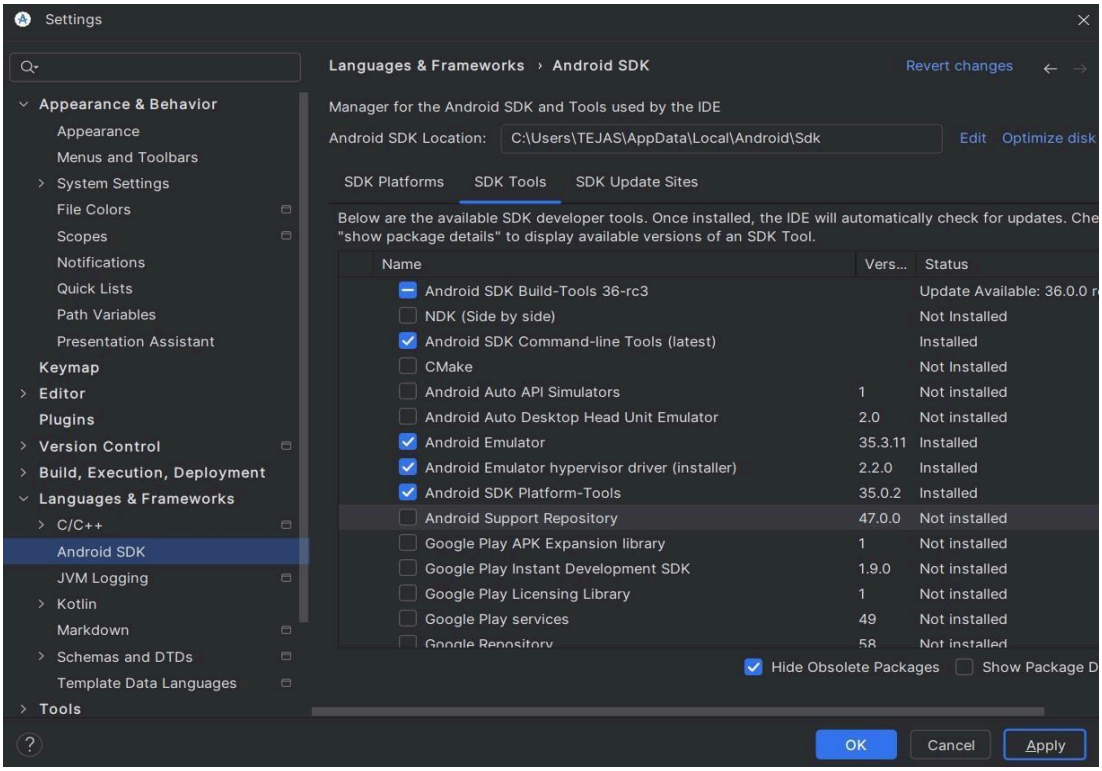


Step 8.4: - Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.



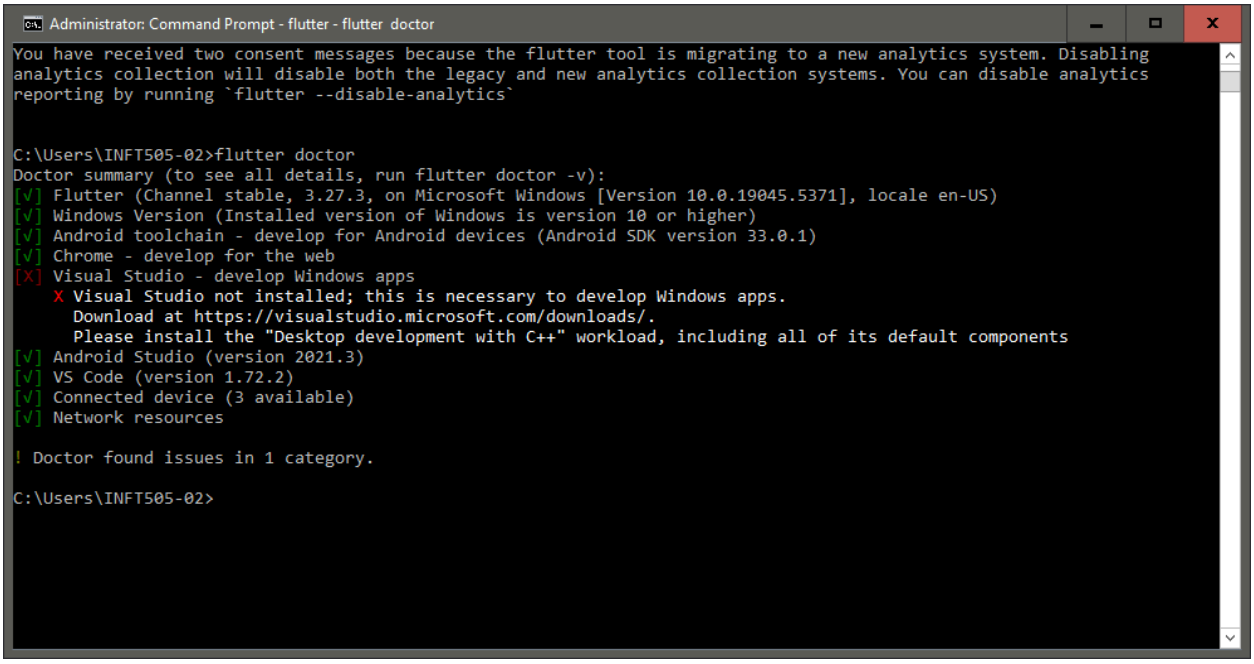


Step 8.5: - Go to Preferences > Appearance & Behavior > System Settings > Android SDK. Select the SDK Tools tab and check Android SDK Command-line Tools and Install it.

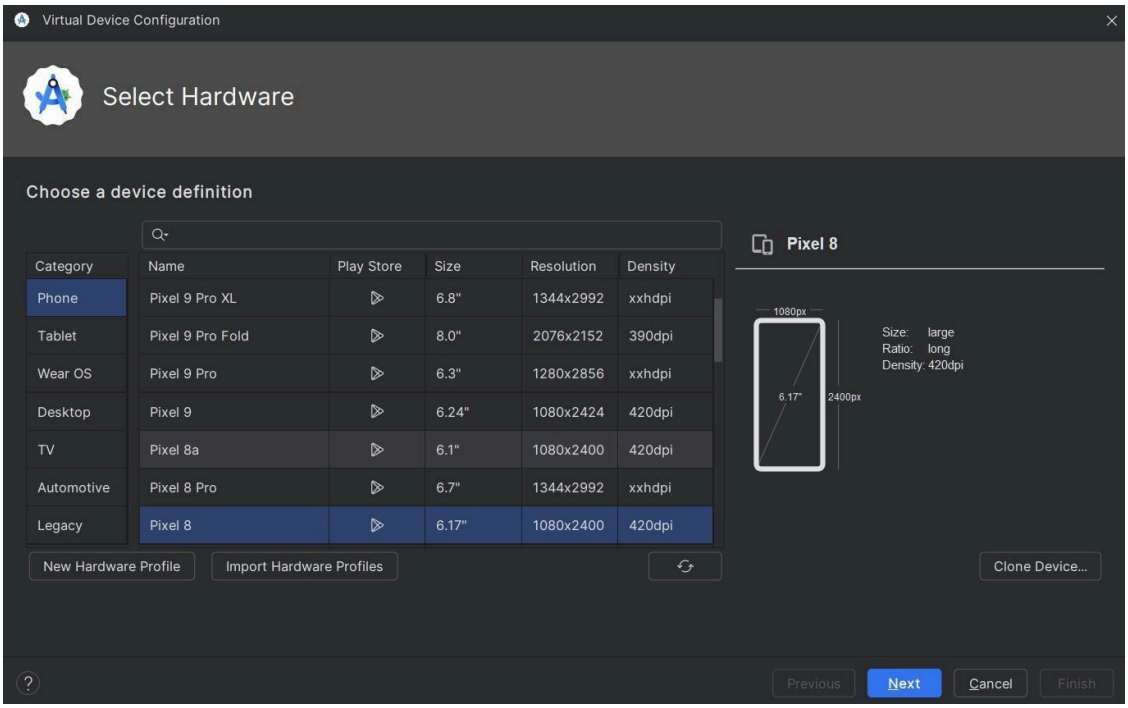


Step 9: - Open a terminal and run the following command

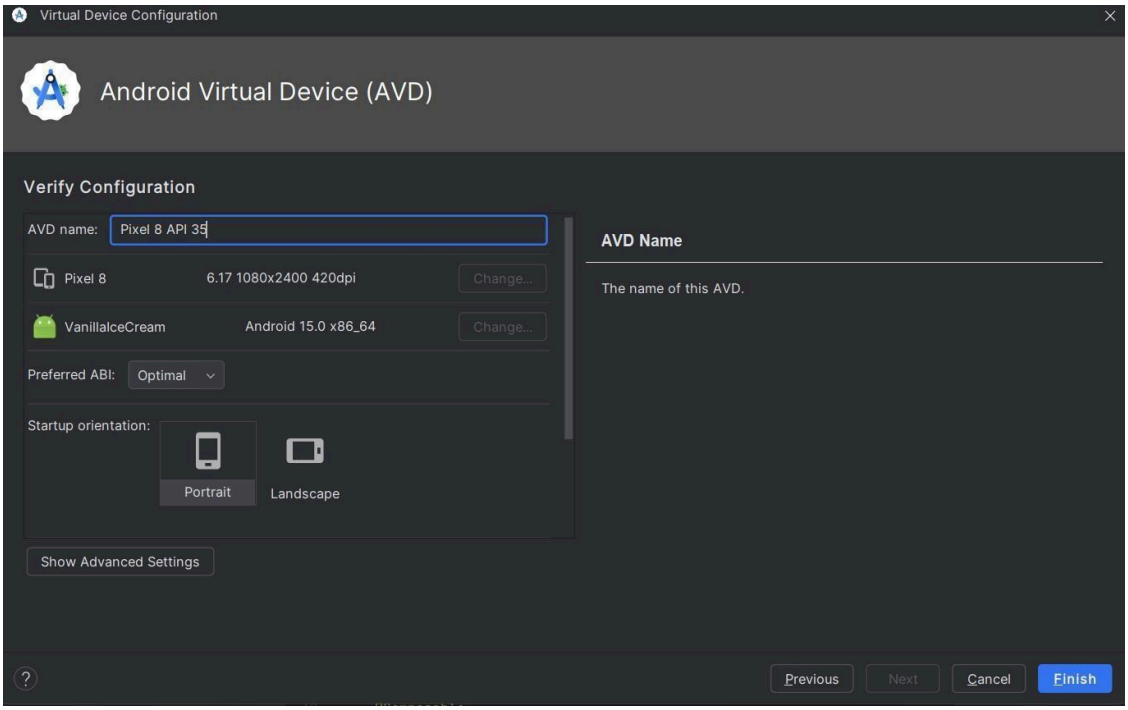
```
C:\Users\INFT505-02>flutter doctor --android-licenses
[===============                          ] 40% Fetch remote repository...        etch remote repository...
Warning: Additionally, the fallback loader failed to parse the XML.
[============================             ] 73% Fetch remote repository...        etch remote repository...
Warning: Additionally, the fallback loader failed to parse the XML.
[======================================] 100% Computing updates...
All SDK package licenses accepted.
```



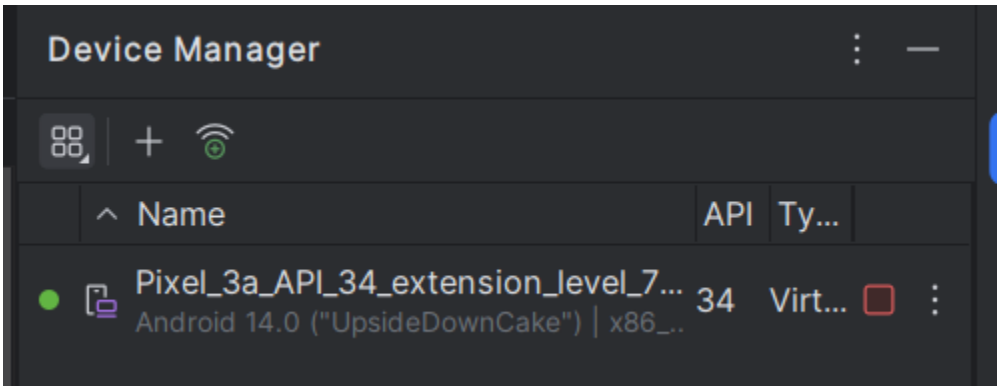
Step 10: - Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application



Step 10.1: - Open Android Studio and go to Tools > AVD Manager. Create a new virtual device.

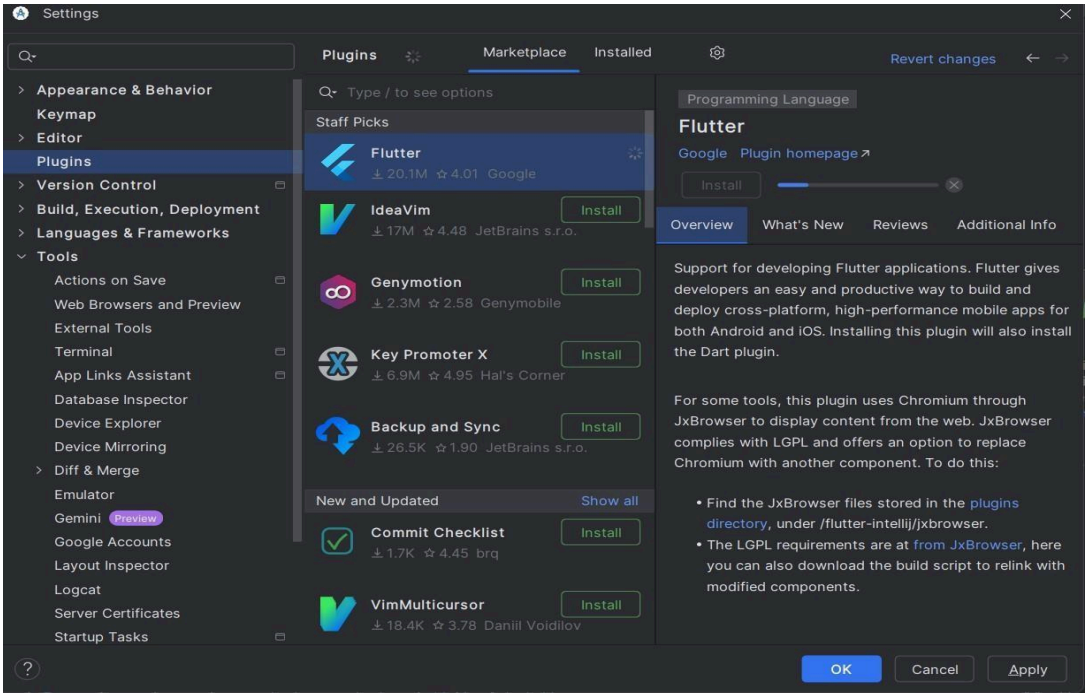


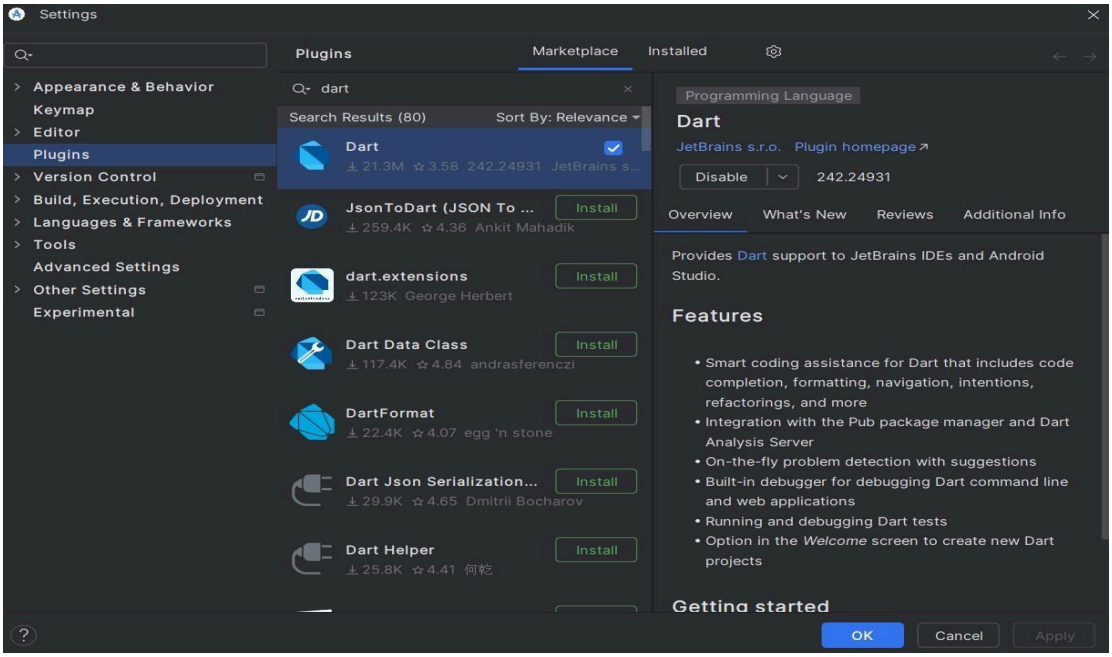
Step 10.2: - Click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen



Step 11: - 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

Step 11.1: - Open the Android Studio and then go to File->Settings->Plugins. Now, search the Flutter plugin. If found, select Flutter plugin and click install





Step 11.2: - Restart the Android Studio

Step 12: - Go to File > New Project > Create Flutter Project, then select the project name and location, and click Next to proceed.

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

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

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

 // This widget is the root of your application.
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Flutter Demo',
     theme: ThemeData(
       // This is the theme of your application.
       //
       // TRY THIS: Try running your application with "flutter run". You'll see
       // the application has a purple toolbar. Then, without quitting the app,
       // try changing the seedColor in the colorScheme below to Colors.green
       // and then invoke "hot reload" (save your changes or press the "hot
       // reload" button in a Flutter-supported IDE, or press "r" if you used
       // the command line to start the app).
       //
       // Notice that the counter didn't reset back to zero; the application
       // state is not lost during the reload. To reset the state, use hot
       // restart instead.
       //
       // This works for code too, not just values: Most code changes can be
       // tested with just a hot reload.
       colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
       useMaterial3: true,
     ),
     home: const MyHomePage(title: 'Flutter Demo Home Page'),
   );
```

```
 }
}

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

 // This widget is the home page of your application. It is stateful, meaning
 // that it has a State object (defined below) that contains fields that affect
 // how it looks.

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

 final String title;

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

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

 void _incrementCounter() {
   setState(() {
     // This call to setState tells the Flutter framework that something has
     // changed in this State, which causes it to rerun the build method below
     // so that the display can reflect the updated values. If we changed
     // _counter without calling setState(), then the build method would not be
     // called again, and so nothing would appear to happen.
     _counter++;
   });
 }
```

```
@override
Widget build(BuildContext context) {
  // This method is rerun every time setState is called, for instance as done
  // by the _incrementCounter method above.
  //
  // The Flutter framework has been optimized to make rerunning build methods
  // fast, so that you can just rebuild anything that needs updating rather
  // than having to individually change instances of widgets.
  return Scaffold(
    appBar: AppBar(
      // TRY THIS: Try changing the color here to a specific color (to
      // Colors.amber, perhaps?) and trigger a hot reload to see the AppBar
      // change color while the other colors stay the same.
      backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      // Here we take the value from the MyHomePage object that was created by
      // the App.build method, and use it to set our appbar title.
      title: Text(widget.title),
    ),
    body: Center(
      // Center is a layout widget. It takes a single child and positions it
      // in the middle of the parent.
      child: Column(
        // Column is also a layout widget. It takes a list of children and
        // arranges them vertically. By default, it sizes itself to fit its
        // children horizontally, and tries to be as tall as its parent.
        //
        // Column has various properties to control how it sizes itself and
        // how it positions its children. Here we use mainAxisAlignment to
        // center the children vertically; the main axis here is the vertical
        // axis because Columns are vertical (the cross axis would be
        // horizontal).
        //
        // TRY THIS: Invoke "debug painting" (choose the "Toggle Debug Paint"
        // action in the IDE, or press "p" in the console), to see the
        // wireframe for each widget.
```

```
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          const Text(
            'Hello Siddhant',
          ),
          Text(
            '$_counter',
            style: Theme.of(context).textTheme.headlineMedium,
          ),
        ],
      ),
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: _incrementCounter,
      tooltip: 'Increment',
      child: const Icon(Icons.add),
    ), // This trailing comma makes auto-formatting nicer for build methods.
  );
 }
}
```

Output:


Flutter Demo Home Page
DEBUG
Hello Siddhant
0
+