1. Introduction

1.1 Overview of Mobile App Development

In today's fast-paced digital world, mobile applications have become indispensable tools that impact nearly every aspect of human life. From communication, entertainment, education, healthcare, banking, to e-commerce — mobile apps play a vital role in providing instant access to services and enhancing productivity.

With the increasing penetration of smartphones and internet access, there is a growing need for highly responsive, visually appealing, and efficient mobile applications. The software industry seeks professionals capable of delivering cross-platform mobile apps that work seamlessly across Android and iOS platforms without maintaining separate codebases for each. To meet this growing demand, modern development frameworks like **Flutter** have gained immense popularity due to their performance, speed, flexibility, and developer-friendly ecosystem.

Flutter empowers developers to create high-quality native interfaces on both major platforms using a single programming language, Dart. Its powerful widget-based architecture allows for expressive and customizable UIs. With features like hot reload, developers can instantly view the effects of their changes, improve productivity and reduce development time. Moreover, Flutter supports integration with backend services, APIs, and databases, making it suitable for both small-scale and enterprise-grade apps. Its open-source nature and vibrant community contribute to a rich ecosystem of packages and tools.

Startups, enterprises, and freelance developers alike benefit from Flutter's fast development cycle and cost-effective maintenance. The ability to share code across platforms without compromising performance has made Flutter a top choice for modern mobile development.

This lab course introduces students to the essentials of building mobile apps using Flutter, providing them with hands-on experience in designing UI, managing state, handling user input, and integrating external data sources.

1.2 What is Flutter?

Flutter is an open-source UI toolkit developed by **Google** that enables developers to build natively compiled applications for **mobile (Android/iOS), web, and desktop** from a **single codebase**. Flutter uses the **Dart** programming language, which is optimized for UI

development and provides powerful features such as ahead-of-time (AOT) compilation and hot reload.

Flutter combines a high-performance rendering engine, rich set of pre-built customizable widgets, and a reactive programming model that helps developers build beautiful and interactive user interfaces efficiently.

1.3 Why Flutter?

- **Single Codebase**: Write once, deploy everywhere. The same codebase works for Android, iOS, and other platforms.
- **Hot Reload**: Developers can instantly view changes in the code without restarting the app, speeding up development and debugging.
- **Custom UI Design**: Flutter provides a wide variety of widgets for designing flexible and beautiful UIs.
- **Native Performance**: Flutter apps compile to native ARM code, ensuring high performance and smooth animations.
- **Strong Community & Support**: Backed by Google and a large developer community, it has extensive documentation and third-party packages.
- **Integration Capabilities**: Flutter supports integration with Firebase, REST APIs, local databases, payment gateways, and more.

1.4 Real-World Applications of Flutter

Flutter is not just a student-level or startup-friendly framework; it is used by companies around the world to build commercial applications. Some well-known apps built using Flutter include:

- **Google Ads** – Ad management tool built by Google itself.
- **Alibaba** – The largest e-commerce platform in China uses Flutter for parts of its app.
- **Reflectly** – A popular journaling app with advanced animations and AI.
- **BMW** – Used Flutter to unify mobile apps for car management.
- **Dream11** – A fantasy sports platform used widely in India.

These examples show that Flutter is production-ready and highly scalable, suited for both startups and enterprise-grade applications.

1.5 Purpose of the Lab

This lab course is designed to introduce students to the principles and practices of **mobile app development** using **Flutter and Dart**. Through a set of structured experiments, students will learn the fundamentals of building modern, responsive, and functional mobile applications. The lab focuses on both **UI design** and **functional logic**, giving students an end-to-end understanding of app development.

Students will also gain exposure to **state management**, **form handling**, **local storage**, **API integration**, **animations**, and **Firebase authentication** — covering both front-end and back-end integration in mobile apps.

1.6 Applications of This Lab Knowledge

After completing this lab, students will be able to:

- Create cross-platform mobile applications using Flutter.
- Design intuitive and responsive UIs with Material Design.
- Implement user login/signup functionality using Firebase Authentication.
- Work with REST APIs and display data in list views.
- Perform local data storage using shared preferences and SQLite.
- Use animations and advanced widgets to enhance the user experience.
- Build projects and deploy them to real devices or emulators.

1.7 Career and Industry Relevance

Learning Flutter opens doors to a wide range of career opportunities, such as:

- **Mobile App Developer (Flutter Developer)**
- **Cross-Platform App Developer**
- **UI/UX Developer for Mobile Platforms**
- **Freelance App Development**
- **Startup MVP Developer**

Flutter developers are in high demand globally due to the efficiency and speed they bring to the mobile development process. This lab not only helps students gain academic credit but also prepares them for real-world jobs, internships, and entrepreneurial ventures.

1.8 Lab Orientation and Learning Outcomes

The goal of this lab is to equip students with practical knowledge and experience in building complete mobile applications using Flutter. Through step-by-step experiments, students will:

- Understand the architecture and project structure of Flutter apps.
- Design responsive and adaptive user interfaces.
- Implement logic for dynamic UI updates using StatefulWidgets.
- Use routing and navigation to switch between multiple screens.
- Apply state management techniques (Provider).
- Handle user input using forms and validations.
- Store and retrieve data using both local and remote sources.
- Animate UI elements for better user engagement.
- Integrate Firebase for authentication and real-time data storage.

1.9 Tools and Technologies Used

- Flutter SDK
- Dart programming language
- Android Studio / Visual Studio Code
- Firebase Authentication
- http plugin for API calls
- shared_preferences plugin
- sqflite plugin for SQLite
- Provider package for state management
- Google Fonts, Custom Widgets, Animations

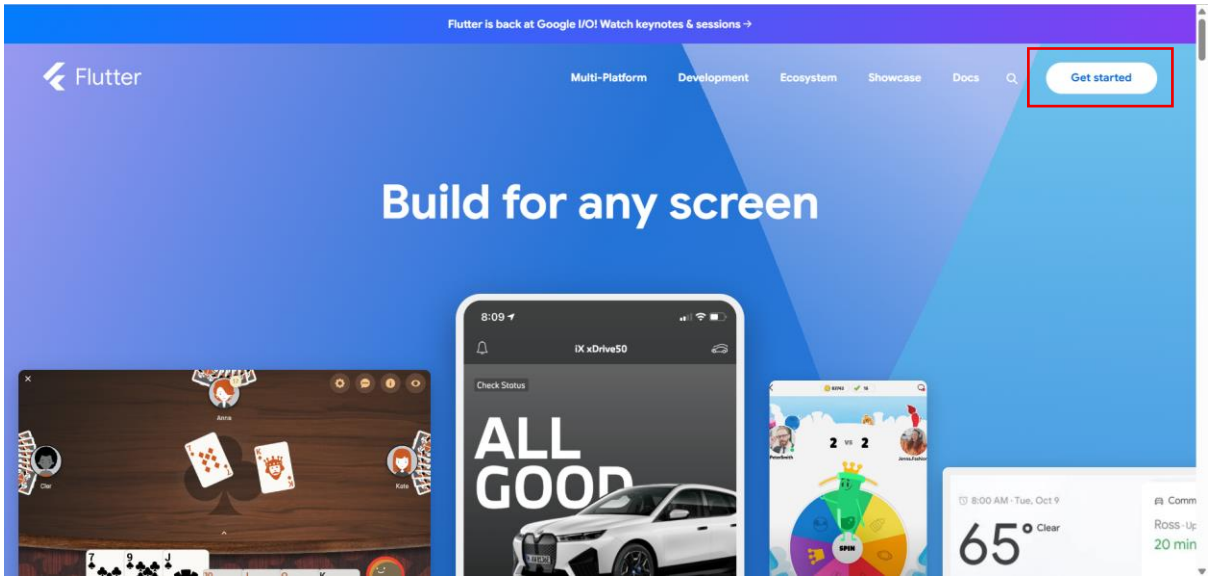
EXPERIMENT NO: 1

1. a) Install Flutter, Android Studio and Dart SDK.

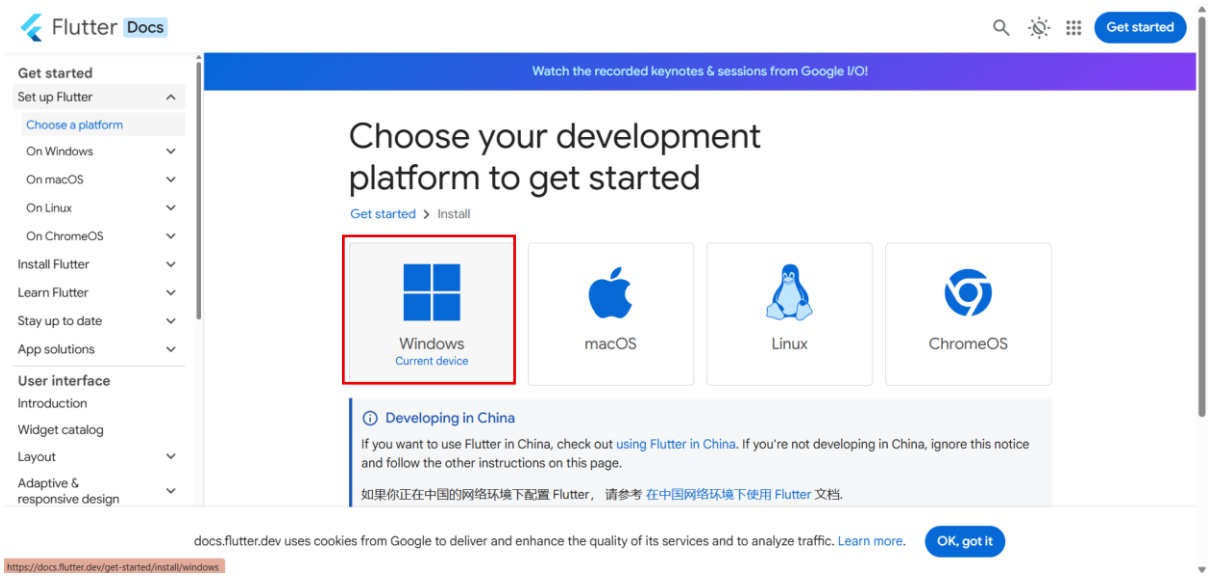
Procedure:

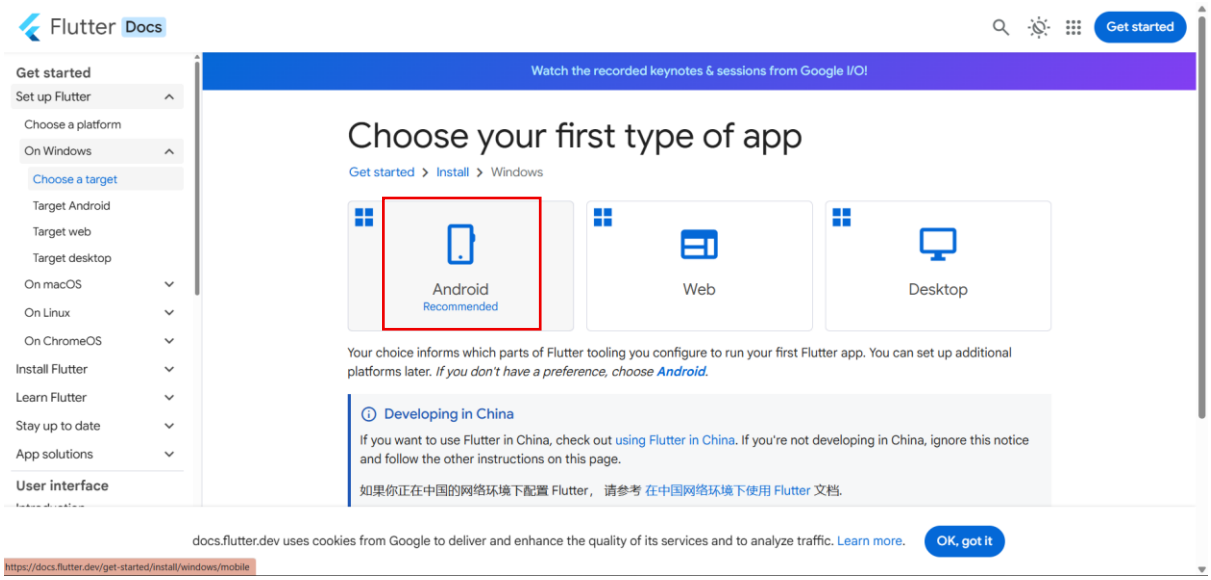
Installation of Flutter SDK:

To install flutter, go to google chrome and type **install flutter**, then open the first link and click Get Started or click on the link: https://docs.flutter.dev/get-started

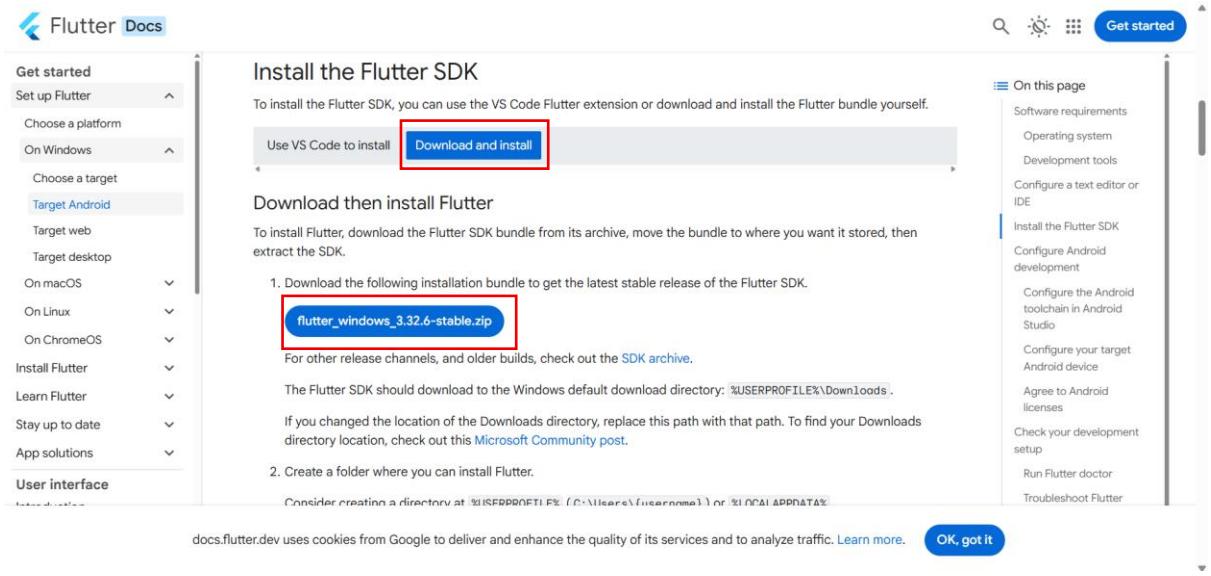


Now choose your development platform to get started, click on your current device **(Windows)**, and then choose your first time app **(Android).**

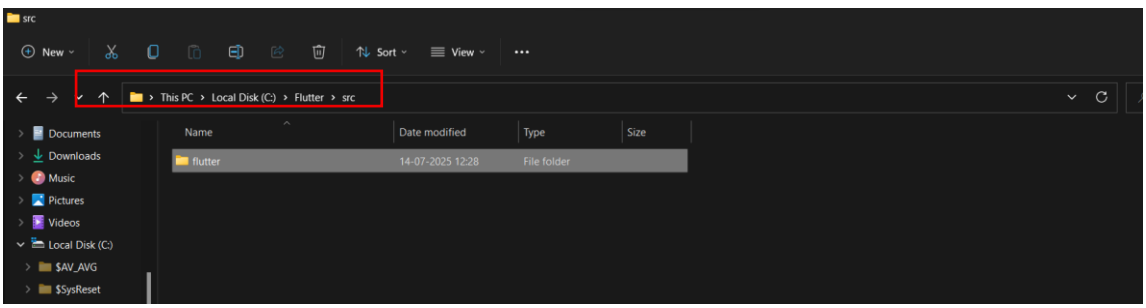




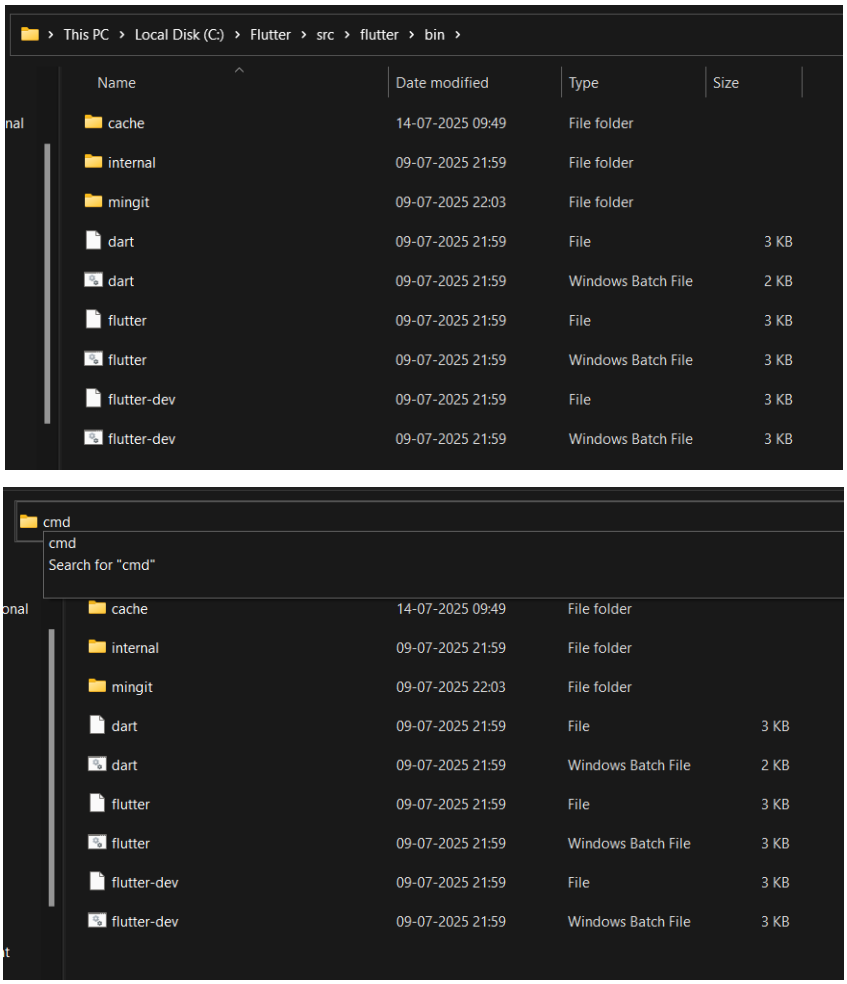
You can see a page of **Start building Flutter Android apps on Windows**, go to **Install the Flutter SDK**, **Download and Install** and download the installation bundle to get the latest stable release of the Flutter SDK.



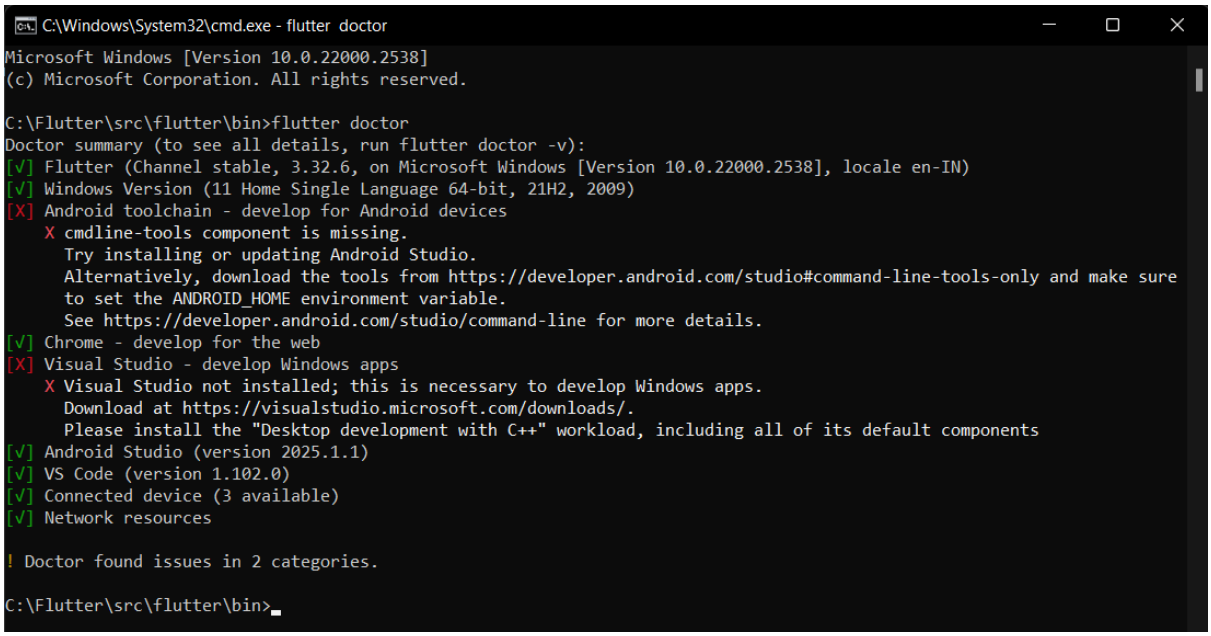
Here is the crucial part after the download. Create a folder "**Flutter**" in **C drive** and in Flutter folder create a folder called "**src**". Now extract the downloaded flutter zip file in this **src** folder.



Now open **C:\Flutter\src\flutter\bin**, click on the address bar and type **cmd**.



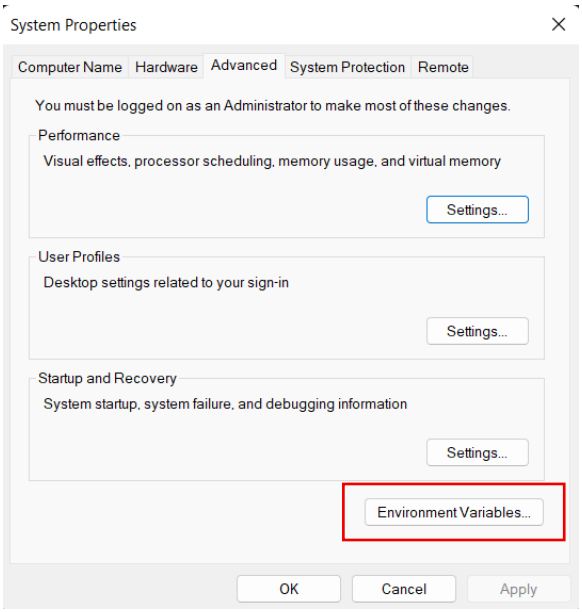
Then command prompt will open with our desired path of bin folder. Now type flutter doctor to check whether the flutter has been installed correctly or not.

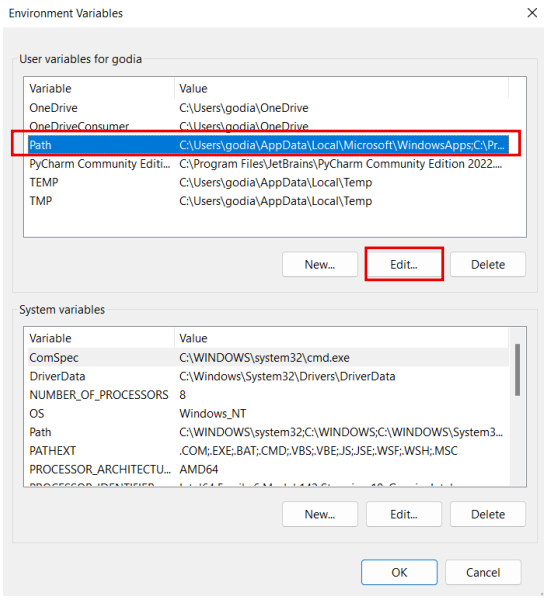


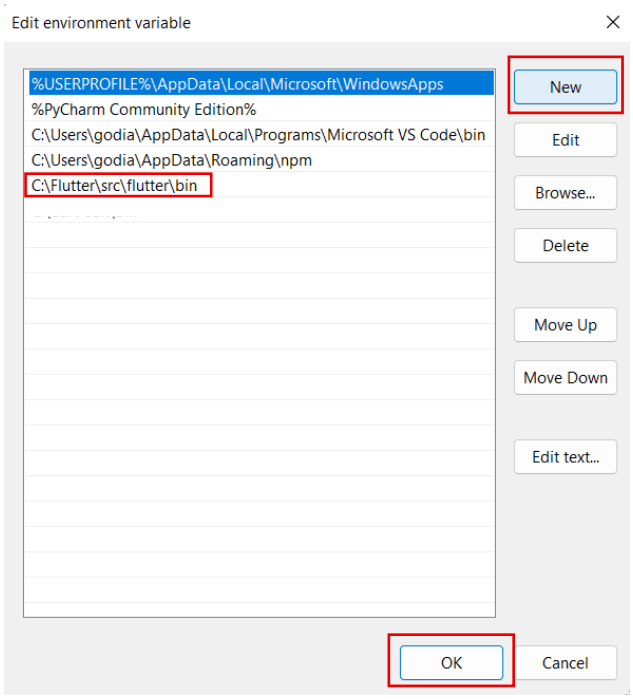
When you directly open command prompt and type flutter doctor. The installed flutter can't be

recognized by the command prompt. This can be tackled by adding the path of the flutter in the system environment variables.

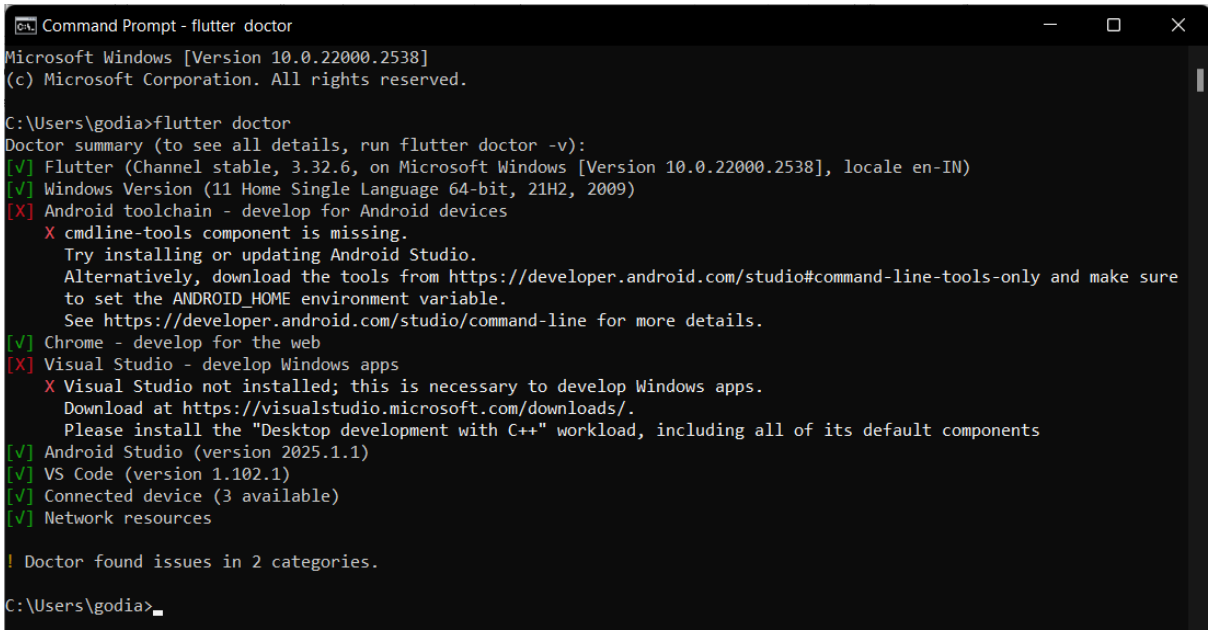
Search for **Edit system environment variables** and open it, a System properties window will open. Click **Environment Variables**, Environment Variables window will be open. Now click **path** and **Edit**. Now click New and copy the flutter path as C:\Flutter\src\flutter\bin. Click OK.







Now open command prompt directly and type flutter doctor, without path in the command prompt the flutter will open. Thus, the installation of flutter has completed and this is what a successful setup looks like.

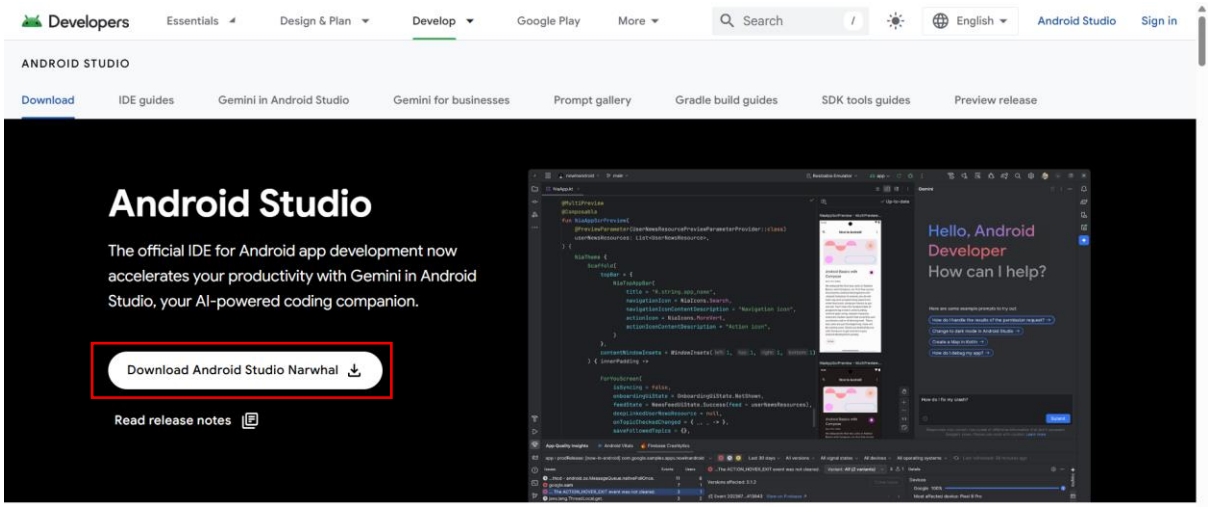


Installation of Dart SDK

After installation of flutter, **dart installation is not needed** as dart is available on the flutter bundles.

Installation of Android Studio

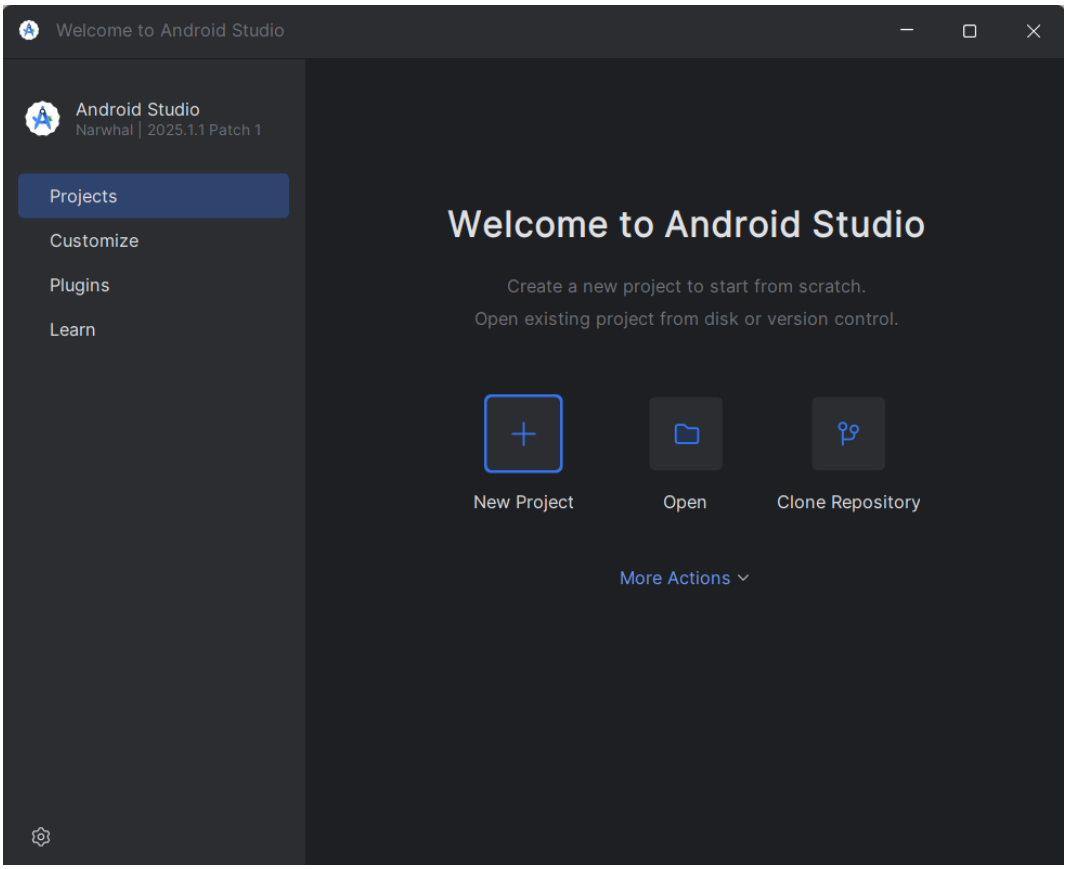
Go to chrome and type **install android studio**, click the download button. Open the file in downloads and finish the installation process.



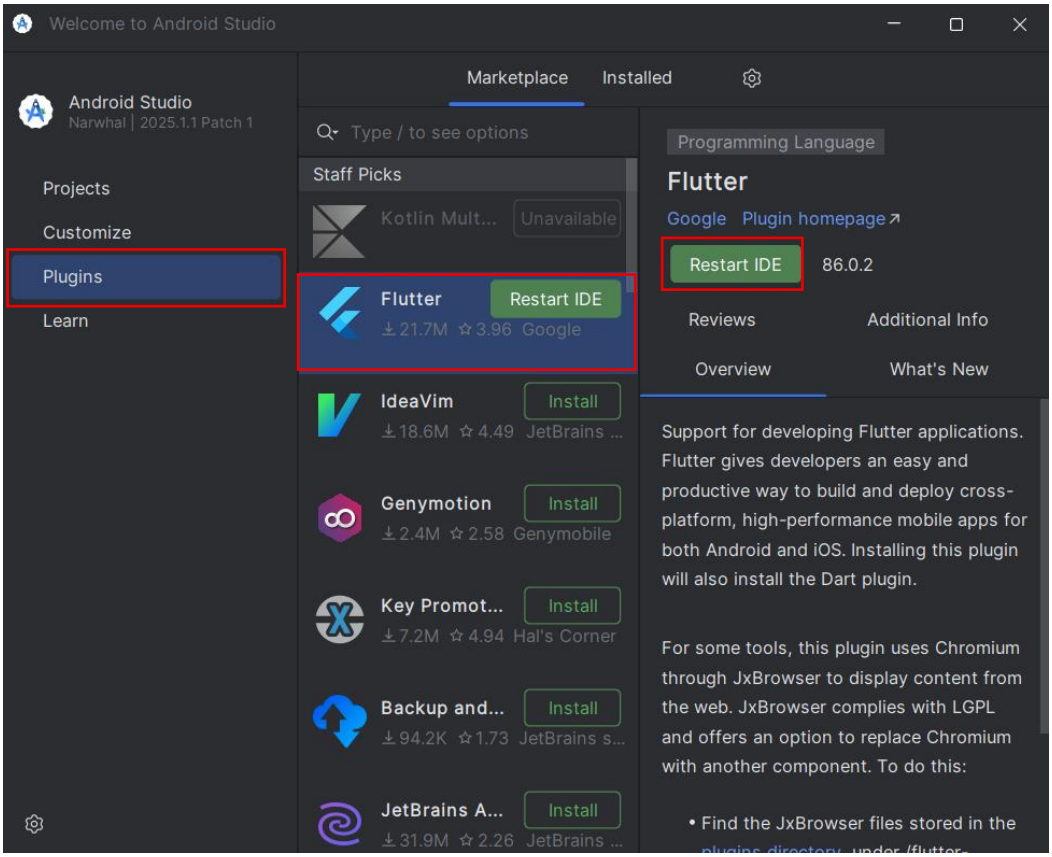
Next → Do not import setting → OK→ Don't send.

Then wizard will open →Next → Standard → Next → Accept → Finish.

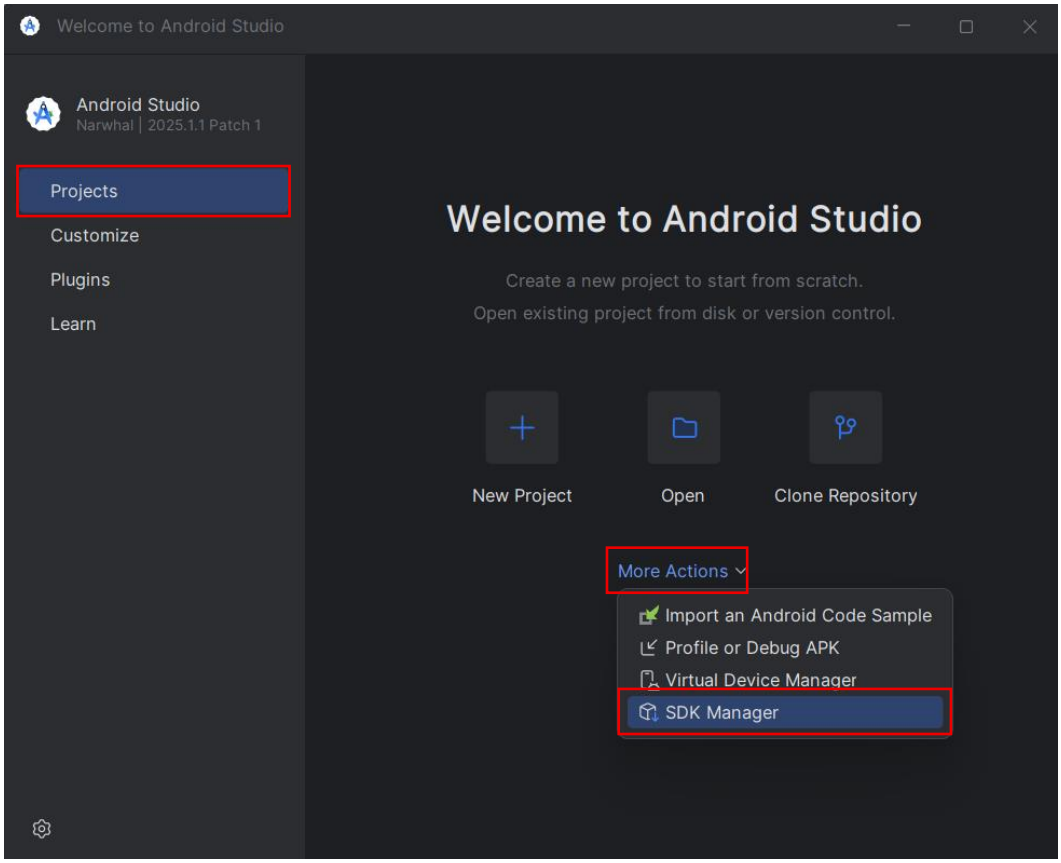
Now, installation of Android Studio has completed.



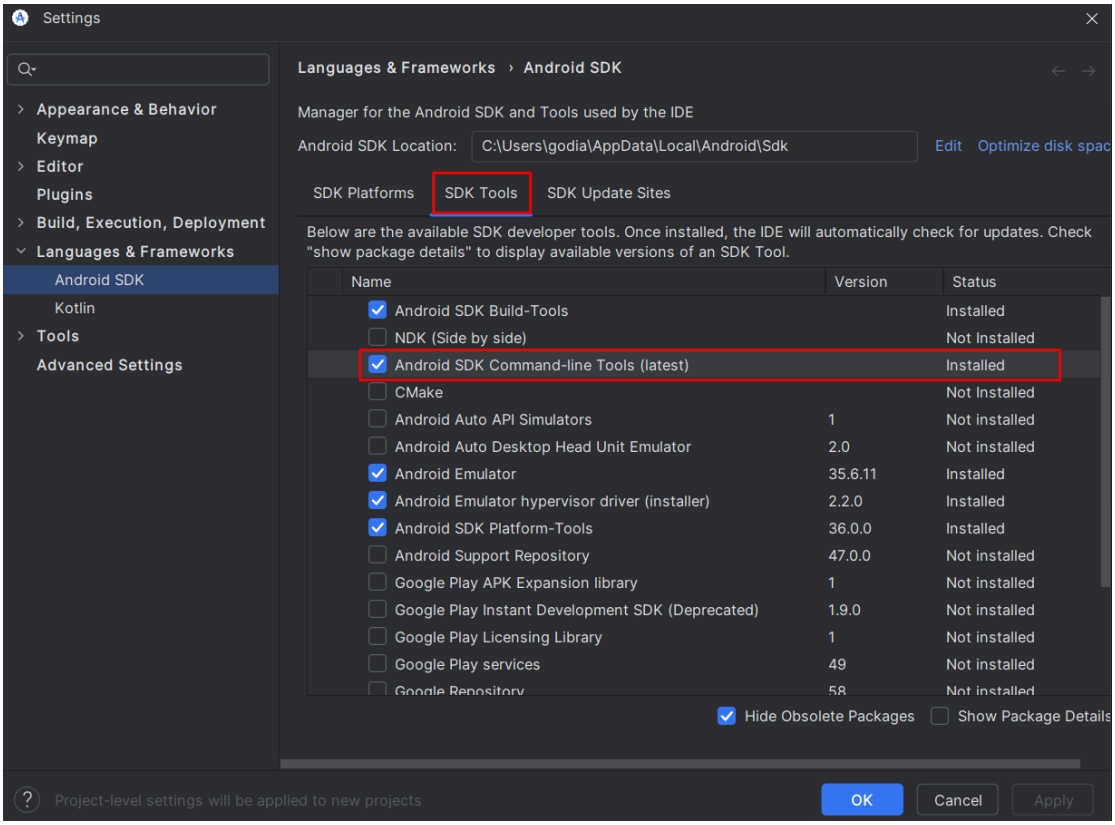
Go to **Plugins** in the menu → **Install flutter** → **Restart IDE**.



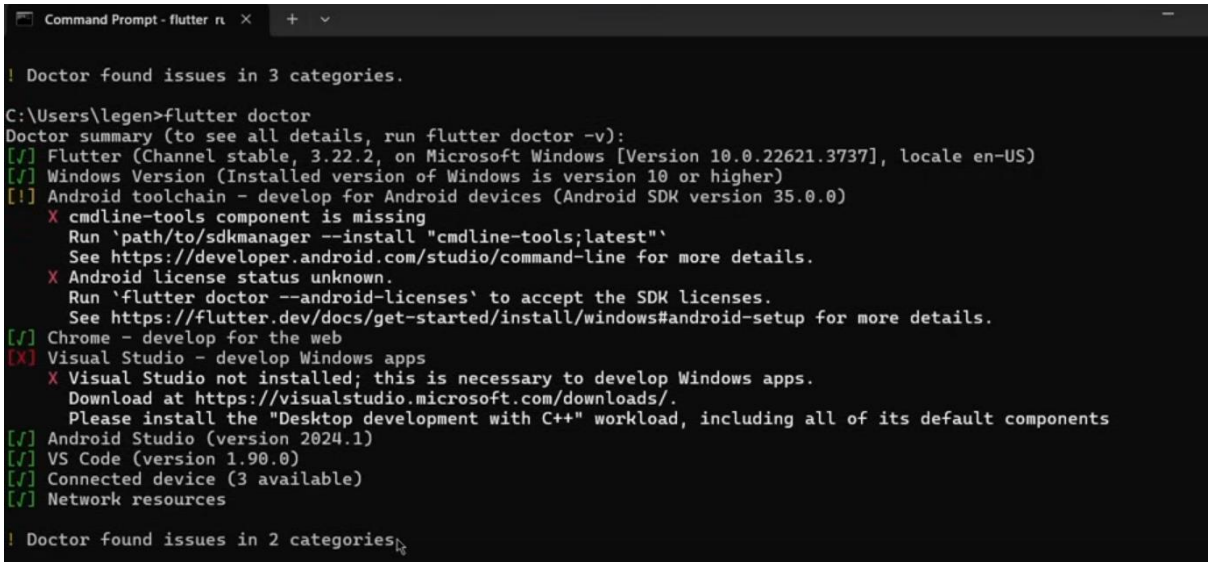
Go to **Projects** in menu, click **more actions** and then go to **SDK Manager**.



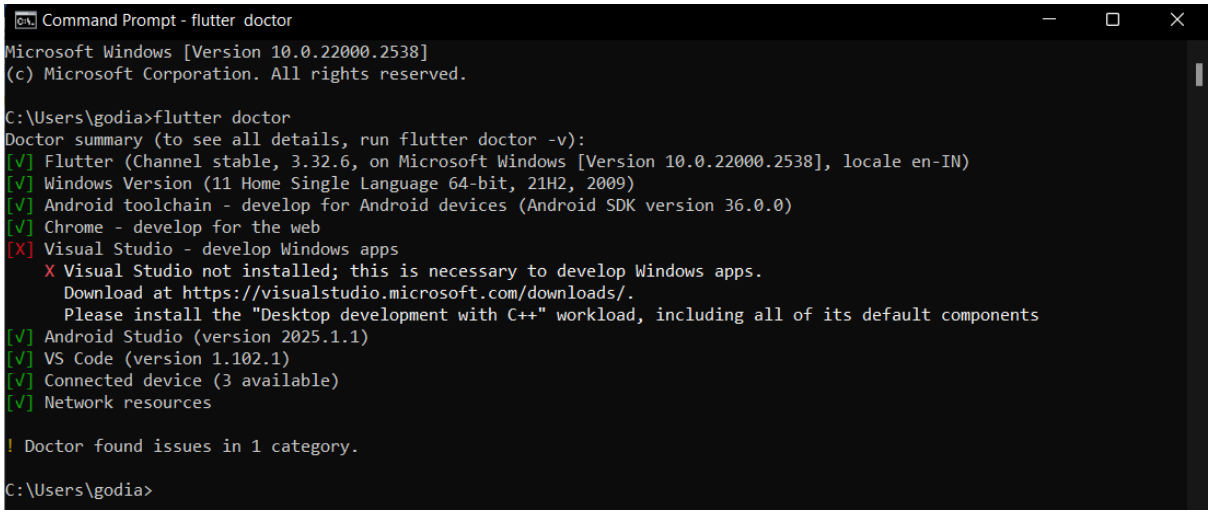
Go to **SDK Tools** → select **Android SDK Component-line Tools (latest)** and download it, and then **finish**.



Now again go to Command Prompt and type flutter doctor, where Android toolchain has missed and Visual Studio has missed. **Visual Studio no need to install**, because we have Android Studio.



For **Android toolchain**, type "**flutter doctor --android licenses**" in command prompt and type y for all licenses. Now check for flutter doctor in command prompt.

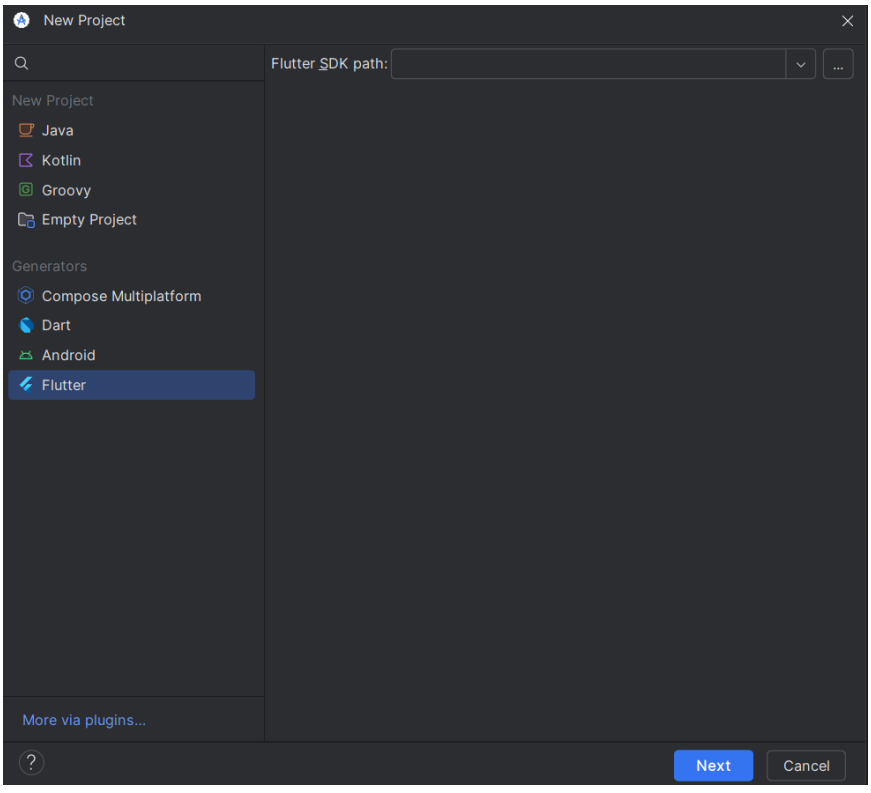


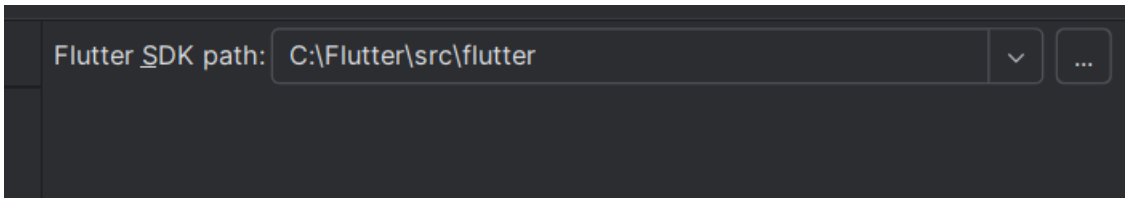
Now everything has been installed for execution of flutter projects.

Go to android studio.

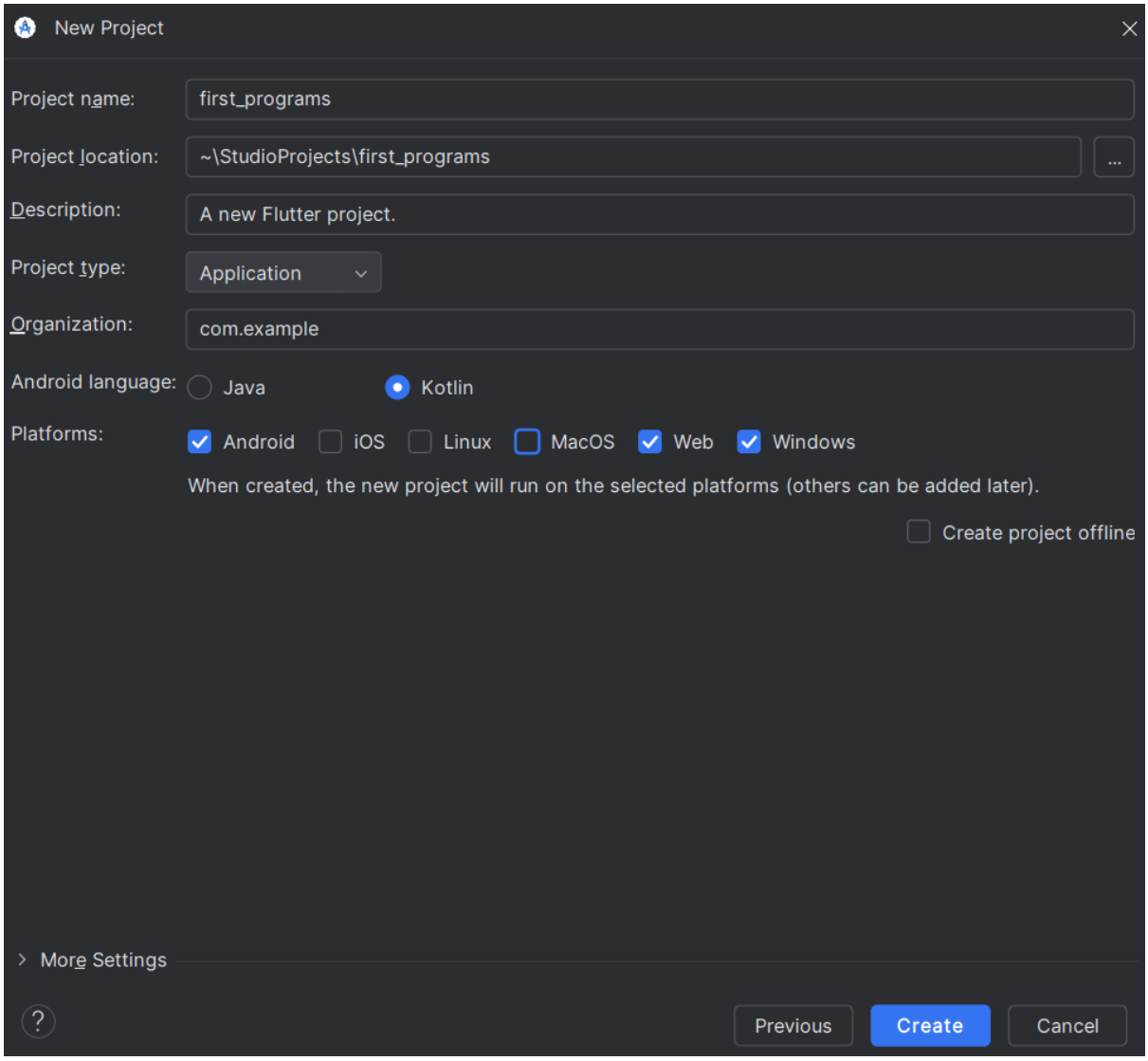
Set Flutter SDK path in Android Studio

Select **New Flutter Project** and in Generators select **flutter**, then in right- top corner select the path of the flutter folder.



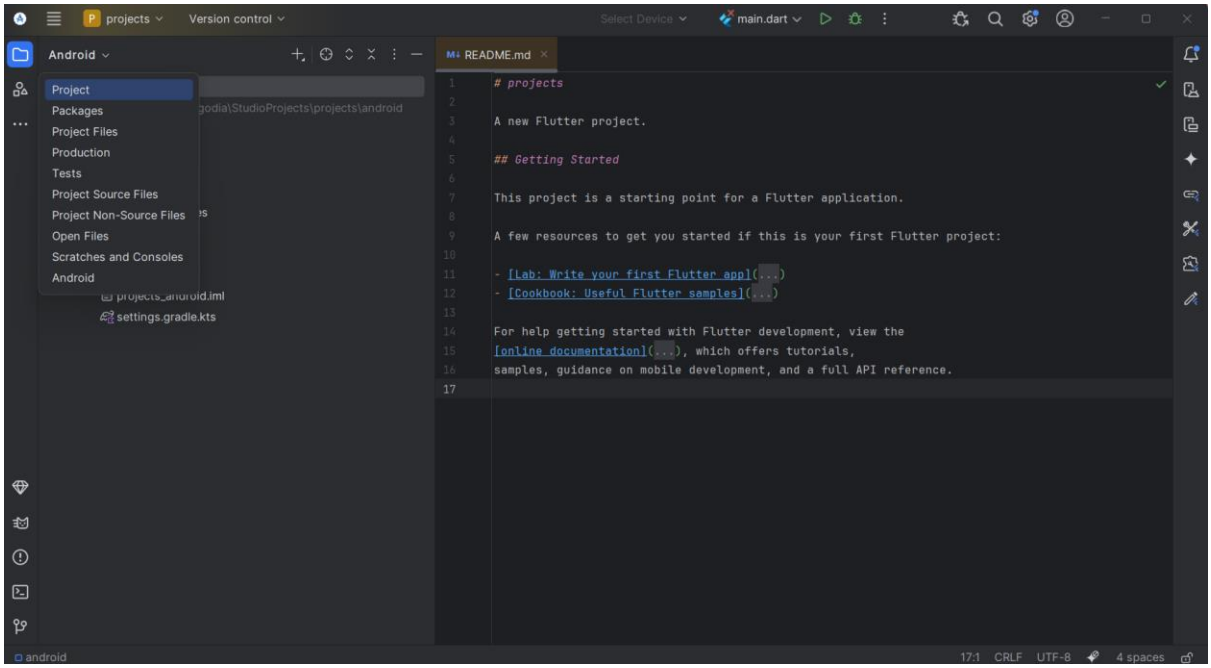


Type the **Project name**, and in platforms select the **required platforms** for coding to the required applications and click **Create**.



Now, the android studio is creating the flutter project, all the files are ready.

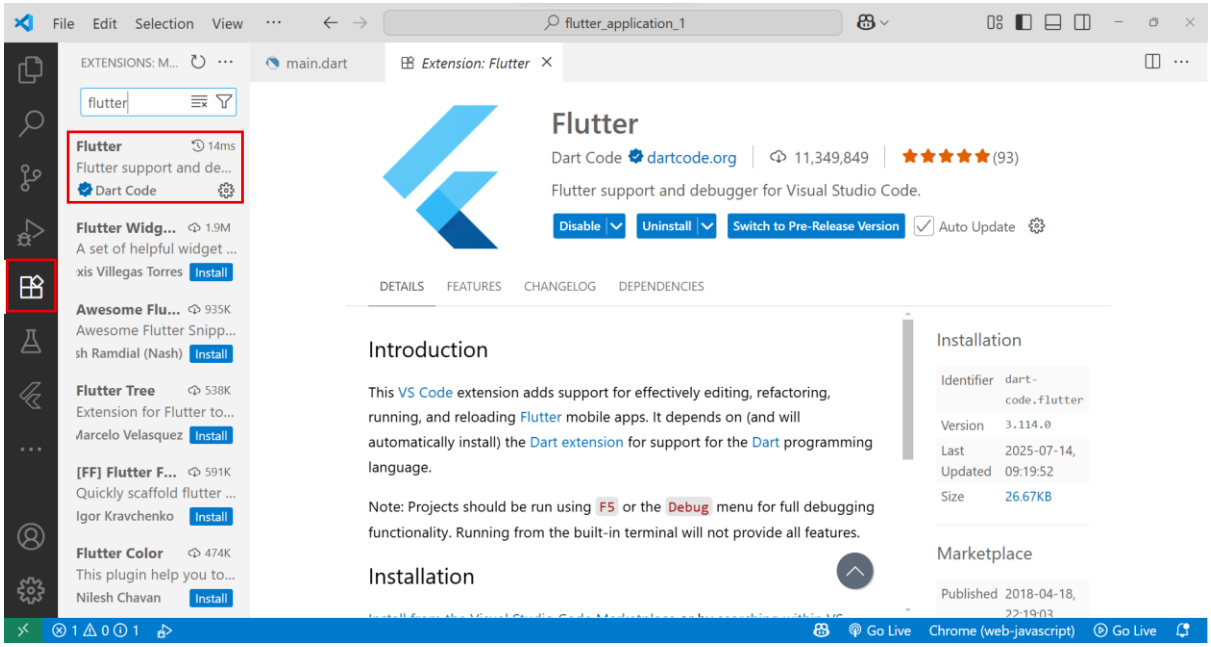
In the left side menu, select **Android** and then **Project**.



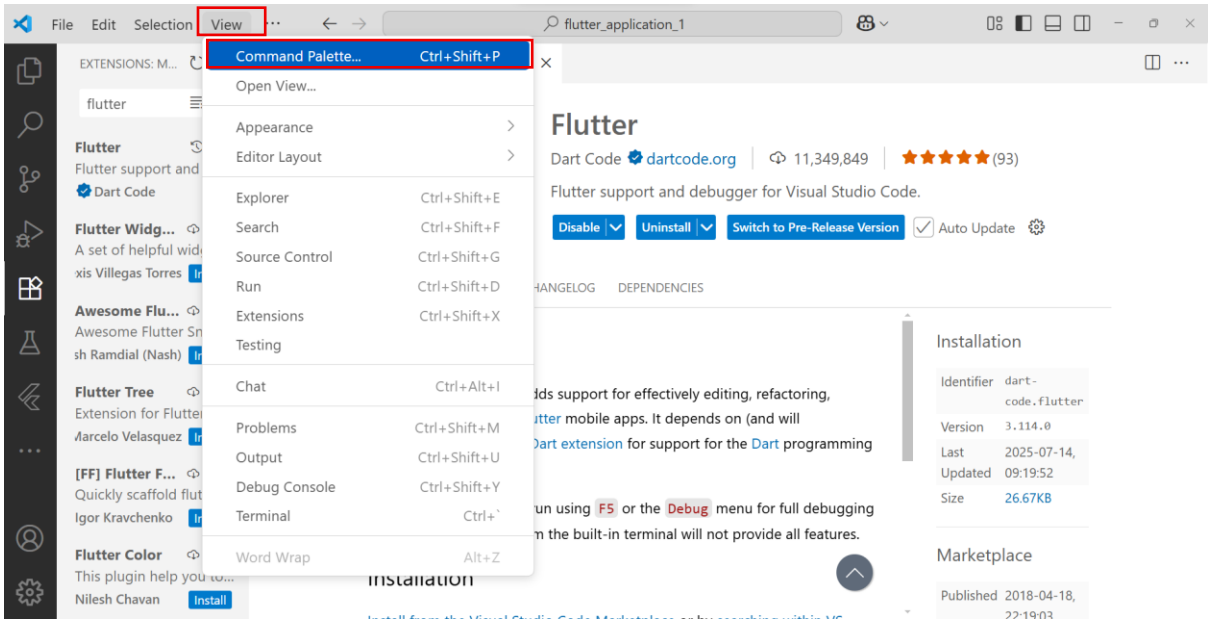
Go to Projects →lib→main.dart. This is the main file.

Installation of VS code

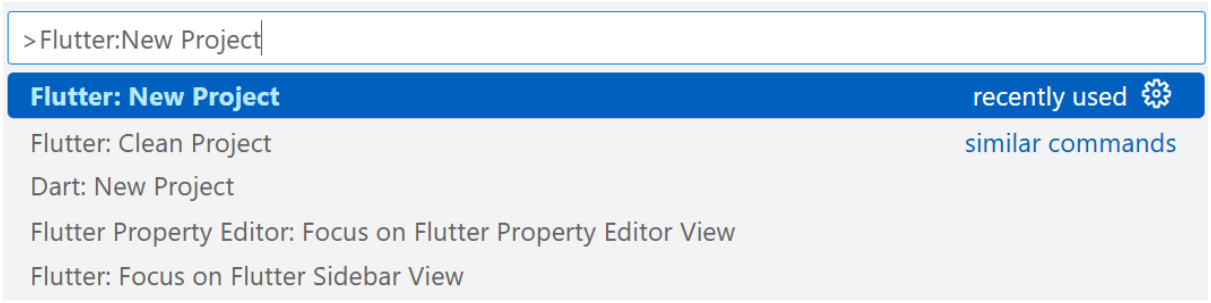
Download Visual Studio Code from the official website and install in your system. Open **VS code**, now install the **extension flutter** in the VS code.

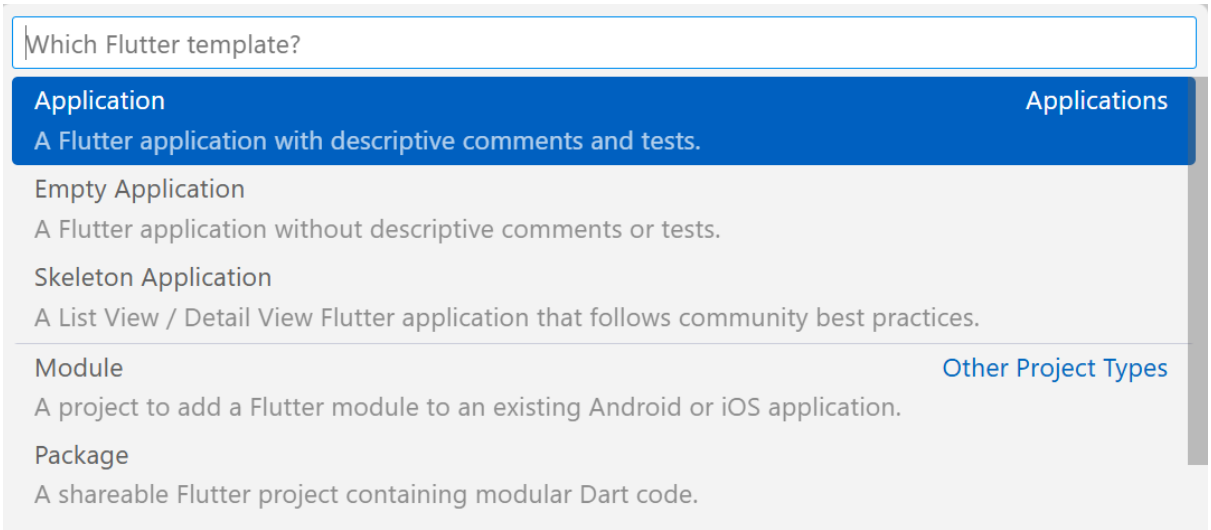


Go to **View → Command palette** or else click **Ctrl+Shift+P**.



The command palette will open, type **Flutter:New Project** and enter. It asks for **Which Flutter template?** Select **Application**

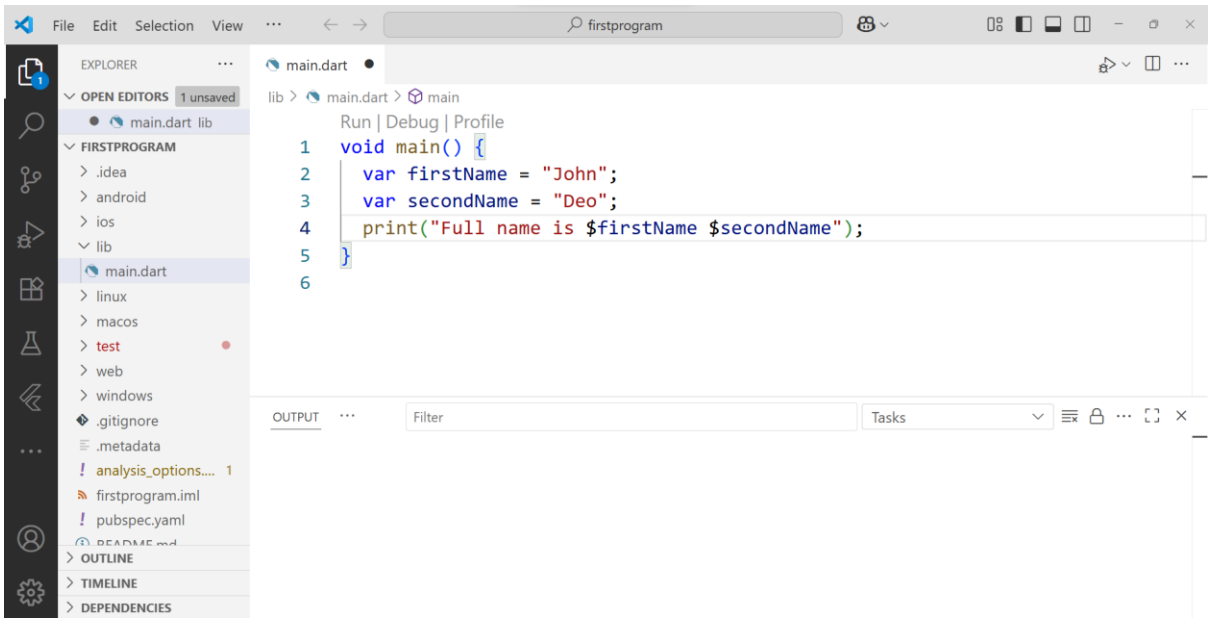




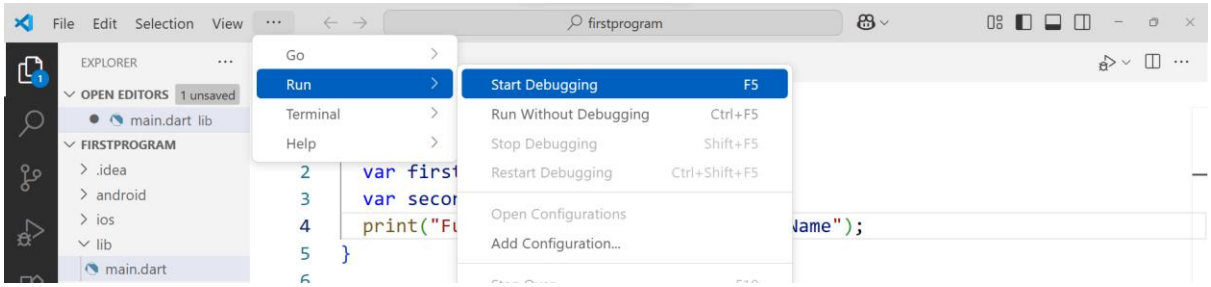
Now, create an **empty folder** in your system and select the created folder where all the projects will be saved. Type the **Project name** and click **enter**. Now everything is ready to start the first program in dart.

1. b) Write a simple Dart program to understand the language basics.

Open the folder from the explorer (left panel), open lib and then main.dart. This is where you need to right the code, erase all the main code and start writing your code in main file → void main().



After writing the code, go to **Run → Start Debugging**.



Now it will launch flutter and output will be displayed in the down panel

Q. Write a simple Dart program to understand string variables and string interpolation.

```
void main()
{
var firstName = "John";
var lastName = "Doe";
print("Full name is $firstName $lastName");
}
```

Output: Full name is John Doe

Q. Write a Dart program to perform basic arithmetic operations.

```
void main()
{
int num1 = 10; //declaring number1
int num2 = 3; //declaring number2
int sum = num1 + num2;        // Calculation
int diff = num1 - num2;
int mul = num1 * num2;
double div = num1 / num2;     // displaying the output
print("The sum is $sum");
print("The diff is $diff");
print("The mul is $mul");
print("The div is $div");
```

```
}
```

Output:

```
The sum is 13
The diff is 7
The mul is 30
The div is 3.3333333333333335
```

Q. Write a Dart program to demonstrate conditional statements.

```
void main() {
  int marks = 75;
  if (marks >= 90) {
    print("Grade: A+");
  }
  else if (marks >= 75) {
    print("Grade: A");
  }
  else if (marks >= 60) {
    print("Grade: B");
  }
  else {
    print("Grade: C");
  } }
```

Output:

```
Grade: A
```

Experiment – 2

2. a. Explore various Flutter widgets (Text, Image, Container, etc.).

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Student Profile Card',
      home: Scaffold(
        backgroundColor: Colors.lightBlue[50],
        appBar: AppBar(
          title: const Text('Profile Card'),
          backgroundColor: Colors.blueAccent,
        ),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              // Profile picture
              const CircleAvatar(
                radius: 60,
                backgroundImage: AssetImage('assets/images/profile.jpg'),
              ),
              const SizedBox(height: 20),

              // Name
              const Text(
                'Amulya',
                style: TextStyle(
                  fontSize: 28,
                  fontWeight: FontWeight.bold,
                  color: Colors.blueAccent,
                ),
              ),

              // Profession
              const Text(
                'Flutter Beginner',
                style: TextStyle(
                  fontSize: 20,
                  color: Colors.black54,
                ),
```

```
),

const SizedBox(height: 20),

// Contact Card
Container(
  margin: const EdgeInsets.symmetric(horizontal: 40),
  padding: const EdgeInsets.all(12),
  decoration: BoxDecoration(
    color: Colors.white,
    borderRadius: BorderRadius.circular(12),
    boxShadow: const [
      BoxShadow(color: Colors.grey, blurRadius: 4),
    ],
  ),
  child: Row(
    children: const [
      Icon(Icons.phone, color: Colors.green),
      SizedBox(width: 10),
      Text(
        '+91 9876543210',
        style: TextStyle(fontSize: 18),
      ),
    ],
  ),
),

const SizedBox(height: 12),

Container(
  margin: const EdgeInsets.symmetric(horizontal: 40),
  padding: const EdgeInsets.all(12),
  decoration: BoxDecoration(
    color: Colors.white,
    borderRadius: BorderRadius.circular(12),
    boxShadow: const [
      BoxShadow(color: Colors.grey, blurRadius: 4),
    ],
  ),
  child: Row(
    children: const [
      Icon(Icons.email, color: Colors.redAccent),
      SizedBox(width: 10),
      Text(
        'amulya@example.com',
        style: TextStyle(fontSize: 18),
      ),
    ],
  ),
),
```

```
          ],
        ),
      ),
    ),
  );
 }
}
```

Steps to setup image in android studio:

1. Right click on the **Project Name → New → Directory →** type **assets.**
2. Right click on the **assets → New → Directory → type images.**
3. Now upload or **copy the image** into **images directory.**
4. Go to pubspec.yaml file in the project folder.
5. Edit the code as

```
flutter:
  uses-material-design: true

  assets:
    - assets/images/profile.jpg
```

6. Run pubget.

Output:

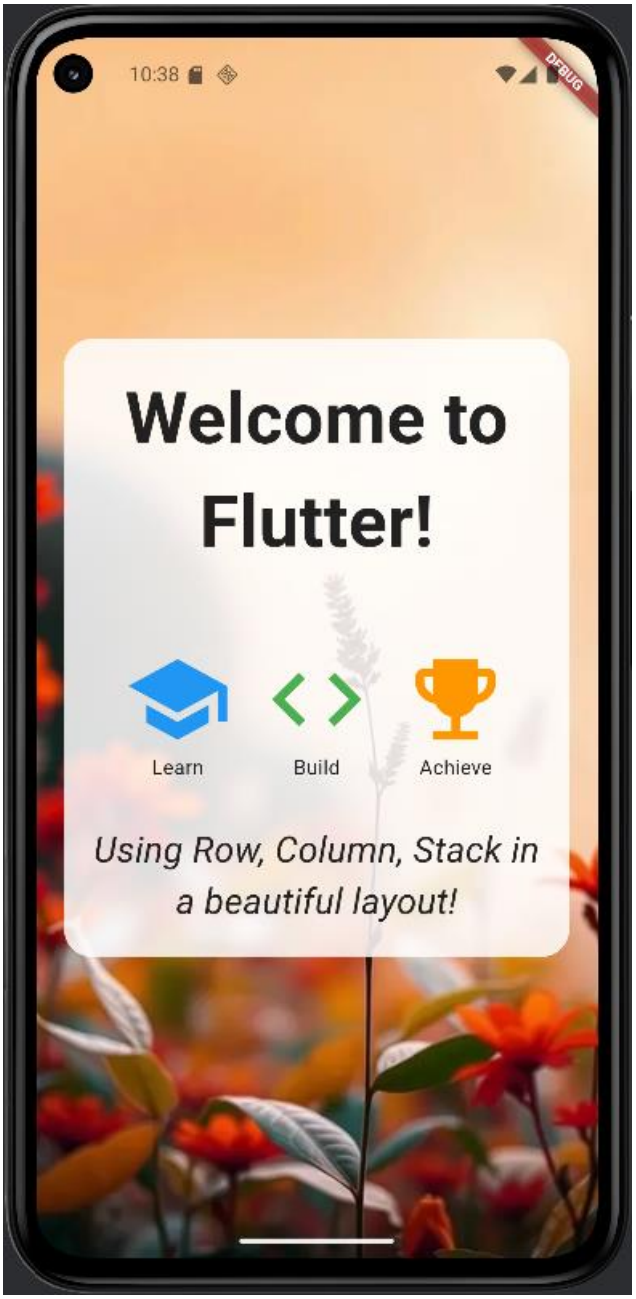

9:31
DEBUG
Profile Card
Amulya
Flutter Beginner
+91 9876543210
amulya@example.com

2. b. Implement different layout structures using Row, Column, and Stack widgets.

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

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

class MyLayoutApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Layout Demo',
      home: Scaffold(
        body: Stack(
          children: [
            // Background image
            Container(
              decoration: BoxDecoration(
                image: DecorationImage(
                  image: AssetImage('assets/images/phone wallpaper.webp'),
                  fit: BoxFit.cover,
                ),
              ),
            ),

            // Foreground content with margin
            Padding(
              padding: const EdgeInsets.symmetric(horizontal: 20.0),
              child: Center(
                child: Container(
                  padding: EdgeInsets.all(20),
                  decoration: BoxDecoration(
                    color: Colors.white.withOpacity(0.85),
                    borderRadius: BorderRadius.circular(20),
                  ),
                  child: Column(
                    mainAxisSize: MainAxisSize.min,
                    children: [
                      Text(
                        'Welcome to Flutter!',
                        textAlign: TextAlign.center,
                        style: TextStyle(
                          fontSize: 50,
                          fontWeight: FontWeight.bold,
                          color: Colors.black87,
                        ),
                      ),
                      SizedBox(height: 50),
```

```
                Row(
                  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                  children: [
                    Column(
                      children: [
                        Icon(Icons.school, size: 75, color: Colors.blue),
                        Text("Learn"),
                      ],
                    ),
                    Column(
                      children: [
                        Icon(Icons.code, size: 75, color: Colors.green),
                        Text("Build"),
                      ],
                    ),
                    Column(
                      children: [
                        Icon(Icons.emoji_events, size: 75, color: Colors.orange),
                        Text("Achieve"),
                      ],
                    ),
                  ],
                ),
                SizedBox(height: 30),
                Text(
                  'Using Row, Column, Stack in a beautiful layout!',
                  textAlign: TextAlign.center,
                  style: TextStyle(
                    fontSize: 25,
                    color: Colors.black87,
                    fontStyle: FontStyle.italic,
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
    ],
  ),
),
);
}
}
```

Output:


10:38
DEBUG
Welcome to Flutter!
Learn
Build
Achieve
Using Row, Column, Stack in a beautiful layout!

Experiment – 3

3. a. Design a responsive UI that adapts to different screen sizes.

b. Implement media queries and breakpoints for responsiveness.

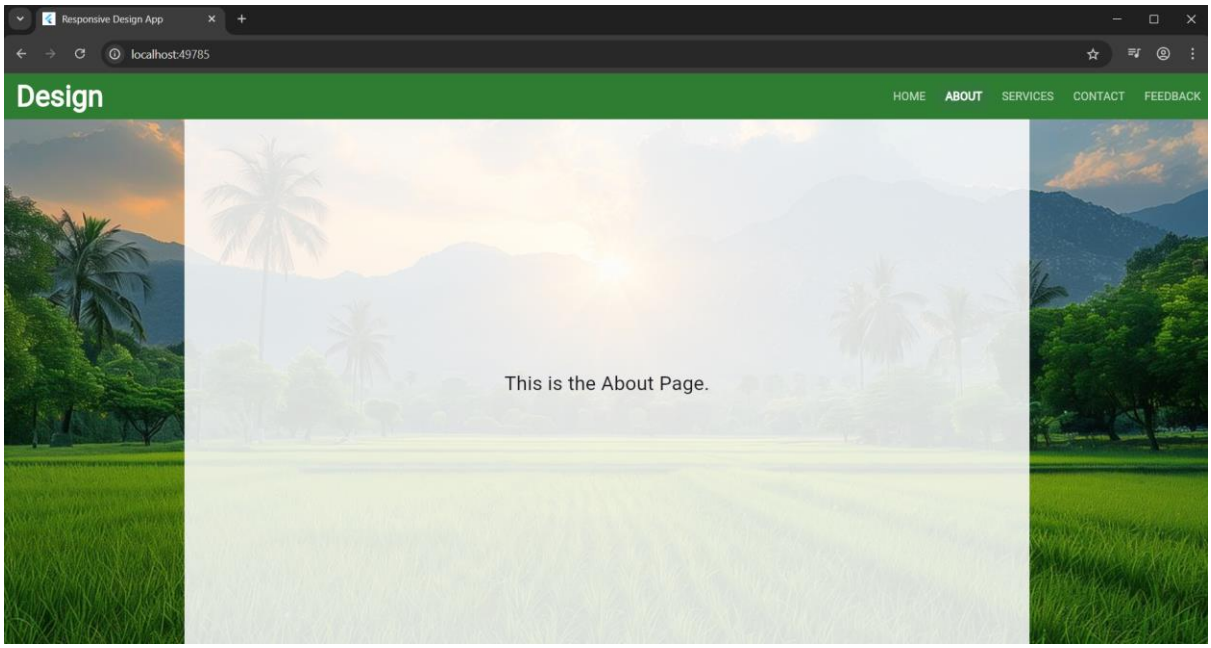
```
import 'package:flutter/material.dart';

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

class MyResponsiveApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Responsive Design App',
      home: HomeScreen(),
      debugShowCheckedModeBanner: false,
    );
  }
}

class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  String selectedPage = 'Home';

  final Map<String, Widget> pageContents = {
    'Home': Center(child: Text('Welcome to the Home Page!', style: TextStyle(fontSize: 24))),
    'About': Center(child: Text('This is the About Page.', style: TextStyle(fontSize: 24))),
    'Services': Center(child: Text('Here are our Services.', style: TextStyle(fontSize: 24))),
    'Contact': Center(child: Text('Contact us at example@flutter.dev', style:
TextStyle(fontSize: 24))),
    'Feedback': Center(child: Text('Give us your Feedback!', style: TextStyle(fontSize: 24))),
  };

  void onMenuSelect(String page) {
    setState(() {
      selectedPage = page;
    });
  }

  Widget buildMenuButton(String title) {
    return TextButton(
```

```
    onPressed: () => onMenuSelect(title),
    child: Text(
     title.toUpperCase(),
     style: TextStyle(
      fontWeight: selectedPage == title ? FontWeight.bold : FontWeight.normal,
      color: selectedPage == title ? Colors.white : Colors.white70,
     ),
    ),
   );
}

@override
Widget build(BuildContext context) {
  final screenWidth = MediaQuery.of(context).size.width;
  final isSmallScreen = screenWidth < 600;

  return Scaffold(
    appBar: AppBar(
     backgroundColor: Colors.green[800],
     title: Text('Design', style: TextStyle(fontWeight: FontWeight.bold,
                                    fontSize: 35,
                                    color: Colors.white)),
     actions: isSmallScreen
        ? [
       PopupMenuButton<String>(
        icon: Icon(Icons.menu, color: Colors.white),
        onSelected: onMenuSelect,
        itemBuilder: (BuildContext context) {
         return ['Home', 'About', 'Services', 'Contact', 'Feedback']
            .map((String choice) => PopupMenuItem<String>(
          value: choice,
          child: Center(
           child: Text(
            choice.toUpperCase(),
            style: TextStyle(
             fontWeight: selectedPage == choice ? FontWeight.bold : FontWeight.normal,
             color: selectedPage == choice ? Colors.green[900] : Colors.red[800],
             fontSize: 15
            ),
           ),
          ),
         ))
            .toList();
        },
       ),
     ]
        : ['Home', 'About', 'Services', 'Contact', 'Feedback']
        .map((title) => buildMenuButton(title))
        .toList(),
    ),
```

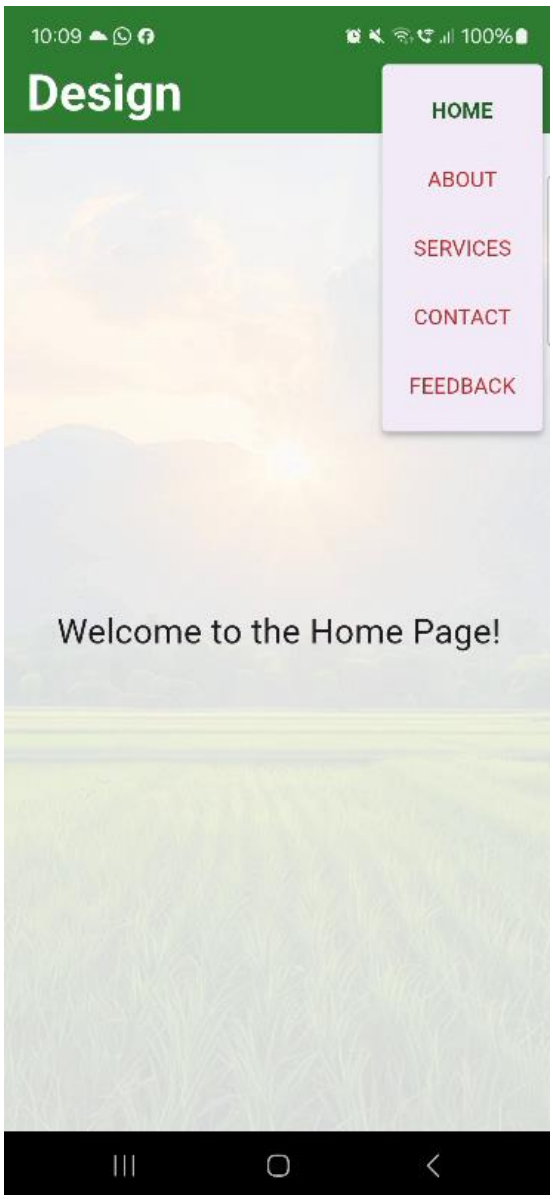
```
      body: Container(
        decoration: BoxDecoration(
          image: DecorationImage(
            image: AssetImage("assets/images/bg.jpg"), // Make sure to include this in
pubspec.yaml
            fit: BoxFit.cover,
          ),
        ),
        child: Center(
          child: Container(
            padding: EdgeInsets.all(20),
            width: isSmallScreen ? double.infinity : screenWidth * 0.7,
            color: Colors.white.withOpacity(0.85),
            child: pageContents[selectedPage],
          ),
        ),
      ),
    );
  }
}
```

Output:

Landscape Mode



Portrait Mode



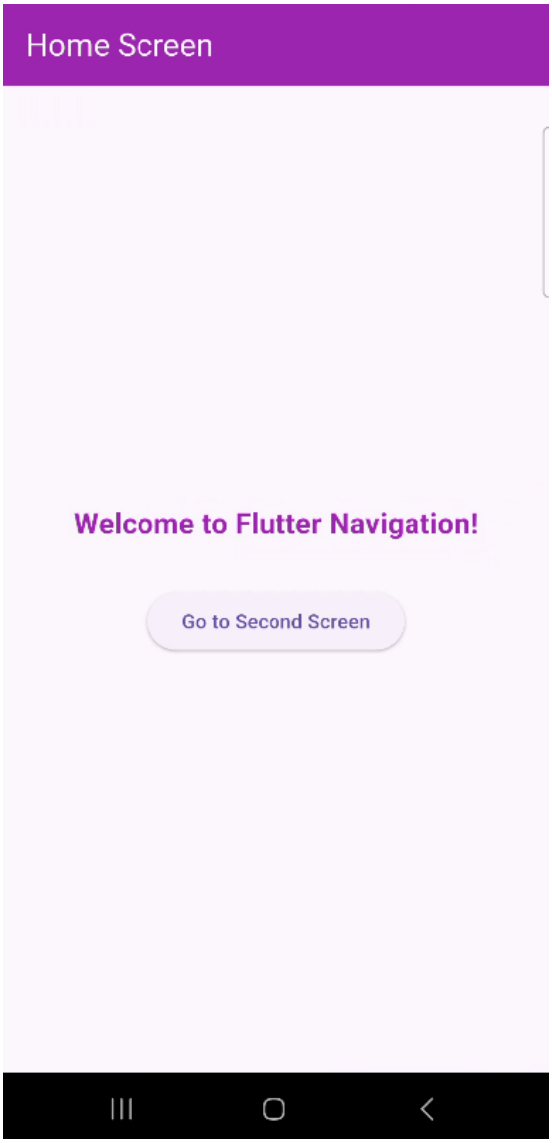
Experiment – 4

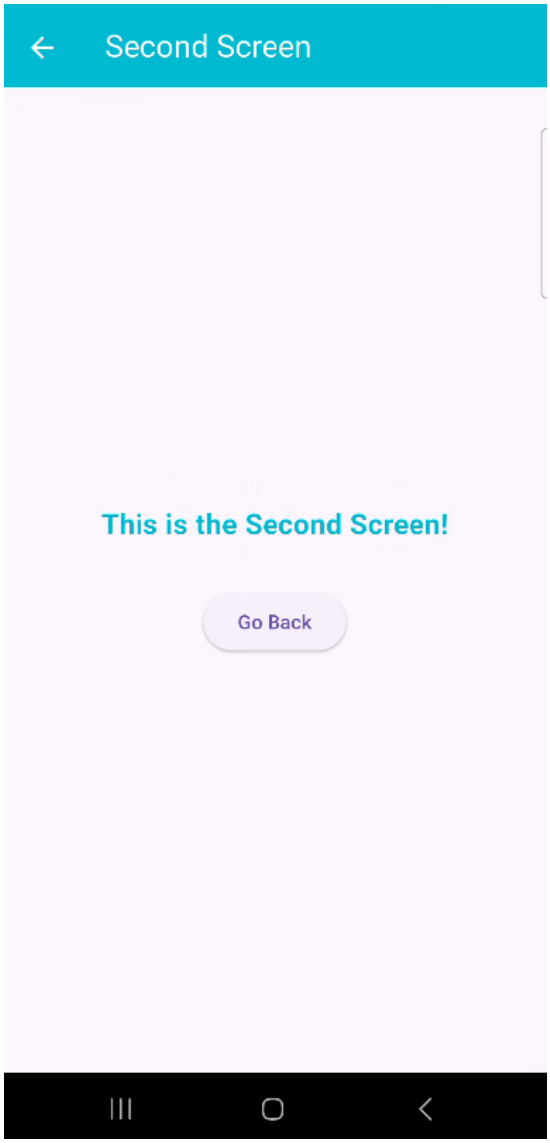
4. a) Set up navigation between different screens using Navigator.

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Navigation Demo',
      debugShowCheckedModeBanner: false,
      home: HomeScreen(),  // Only use 'home' for basic navigation
    );
  }
}

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Home Screen'),
        backgroundColor: Colors.purple,
        foregroundColor: Colors.white,
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              'Welcome to Flutter Navigation!',
              style: TextStyle(
                  fontSize: 20,
                  color: Colors.purple,
                  fontWeight: FontWeight.bold
              ),
            ),
            SizedBox(height: 30),
            ElevatedButton(
              onPressed: () {
                // Basic Navigation - Part A
                Navigator.push(
                  context,
```

```
                MaterialPageRoute(builder: (context) => SecondScreen()),
              );
            },
            child: Text('Go to Second Screen'),
          ),
        ],
      ),
    ),
  );
 }
}

class SecondScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: Text('Second Screen'),
    backgroundColor: Colors.cyan,
    foregroundColor: Colors.white,
   ),
   body: Center(
    child: Column(
     mainAxisAlignment: MainAxisAlignment.center,
     children: [
      Text(
       'This is the Second Screen!',
       style: TextStyle(
          fontSize: 20,
          color: Colors.cyan,
          fontWeight: FontWeight.bold
       ),
      ),
      SizedBox(height: 30),
      ElevatedButton(
       onPressed: () {
        Navigator.pop(context);
       },
       child: Text('Go Back'),
      ),
     ],
    ),
   ),
  );
 }
}
```

Output:


Home Screen
Welcome to Flutter Navigation!
Go to Second Screen


Second Screen
This is the Second Screen!
Go Back

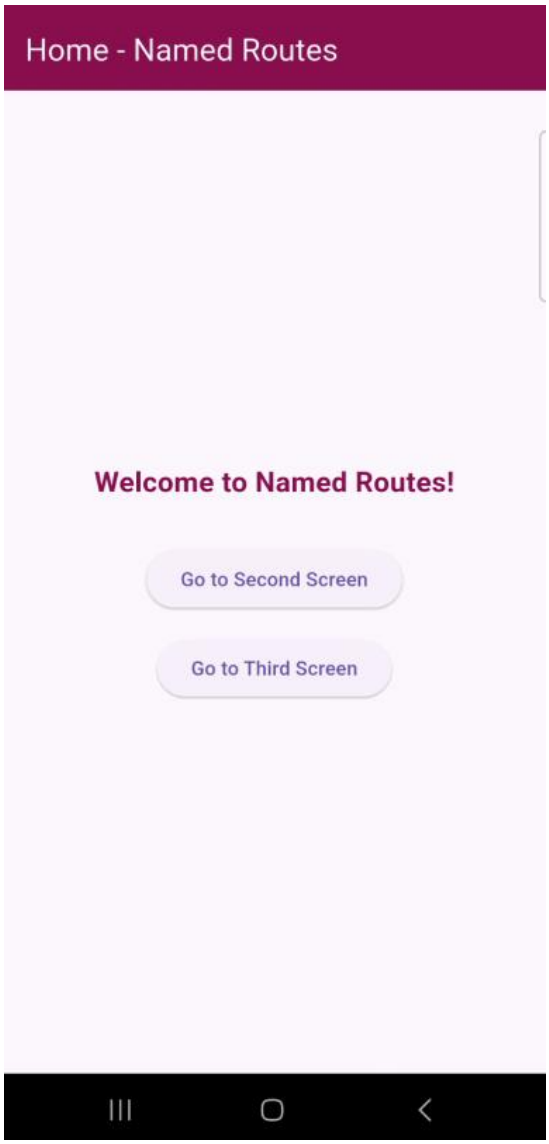
4. b) Implement navigation with named routes.

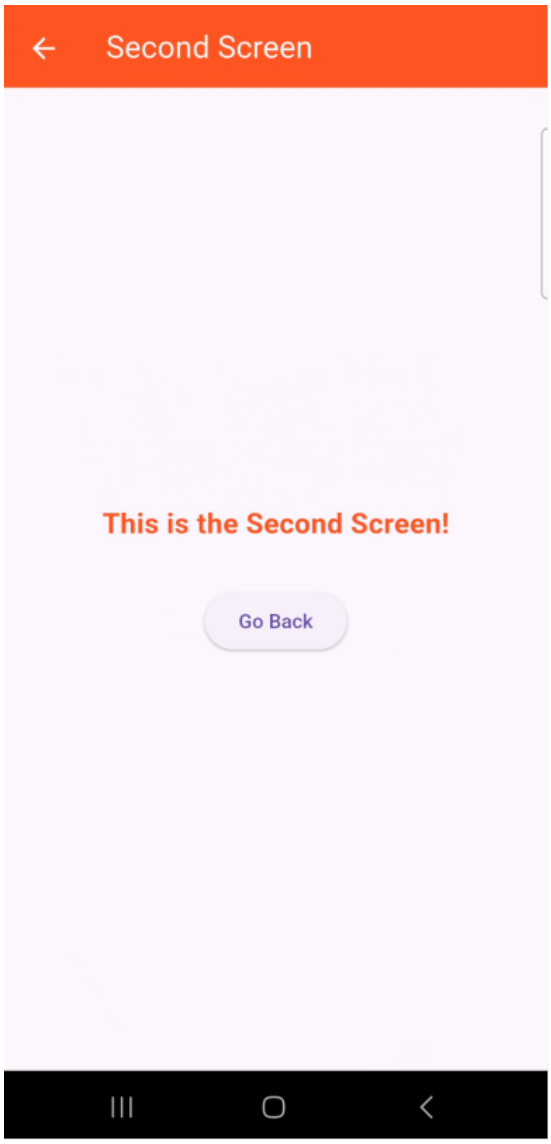
```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Named Routes Demo',
      debugShowCheckedModeBanner: false,
      // Remove 'home' when using named routes
      initialRoute: '/',
      routes: {
        '/': (context) => HomeScreen(),
        '/second': (context) => SecondScreen(),
        '/third': (context) => ThirdScreen(),
      },
    );
  }
}

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Home - Named Routes'),
        backgroundColor: Colors.pink[900],
        foregroundColor: Colors.white,
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              'Welcome to Named Routes!',
              style: TextStyle(
                  fontSize: 20,
                  color: Colors.pink[900],
                  fontWeight: FontWeight.bold
              ),
            ),
            SizedBox(height: 30),
            ElevatedButton(
              onPressed: () {
```

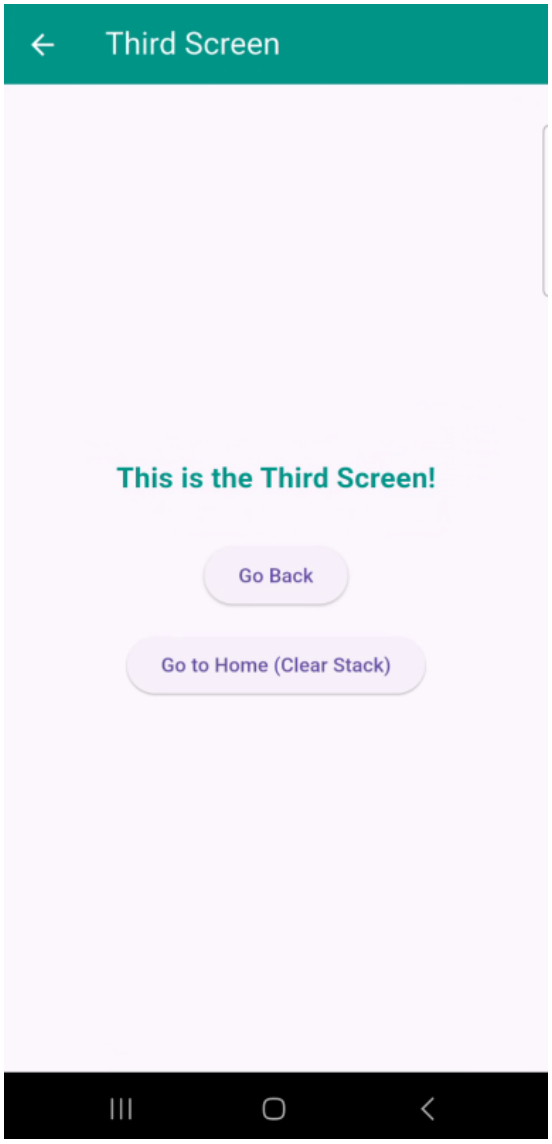
```
              Navigator.pushNamed(context, '/second');
            },
            child: Text('Go to Second Screen'),
          ),
          SizedBox(height: 15),
          ElevatedButton(
            onPressed: () {
              Navigator.pushNamed(context, '/third');
            },
            child: Text('Go to Third Screen'),
          ),
        ],
      ),
    ),
  );
 }
}

class SecondScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Second Screen'),
      backgroundColor: Colors.deepOrange,
      foregroundColor: Colors.white,
    ),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text('This is the Second Screen!',
            style: TextStyle(
              fontSize: 20,
              color: Colors.deepOrange,
              fontWeight: FontWeight.bold
            ),
          ),
          SizedBox(height: 30),
          ElevatedButton(
            onPressed: () => Navigator.pop(context),
            child: Text('Go Back'),
          ),
        ],
      ),
    ),
  );
 }
}
```

```dart
class ThirdScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Third Screen'),
        backgroundColor: Colors.teal,
        foregroundColor: Colors.white,
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text('This is the Third Screen!',
              style: TextStyle(
                  fontSize: 20,
                  color: Colors.teal,
                  fontWeight: FontWeight.bold
              ),
            ),
            SizedBox(height: 30),
            ElevatedButton(
              onPressed: () => Navigator.pop(context),
              child: Text('Go Back'),
            ),
            SizedBox(height: 15),
            ElevatedButton(
              onPressed: () {
                Navigator.pushNamedAndRemoveUntil(context, '/', (route) => false);
              },
              child: Text('Go to Home (Clear Stack)'),
            ),
          ],
        ),
      ),
    );
  }
}
```

Output:


Home - Named Routes
Welcome to Named Routes!
Go to Second Screen
Go to Third Screen


Second Screen
This is the Second Screen!
Go Back


Third Screen
This is the Third Screen!
Go Back
Go to Home (Clear Stack)

Experiment – 5

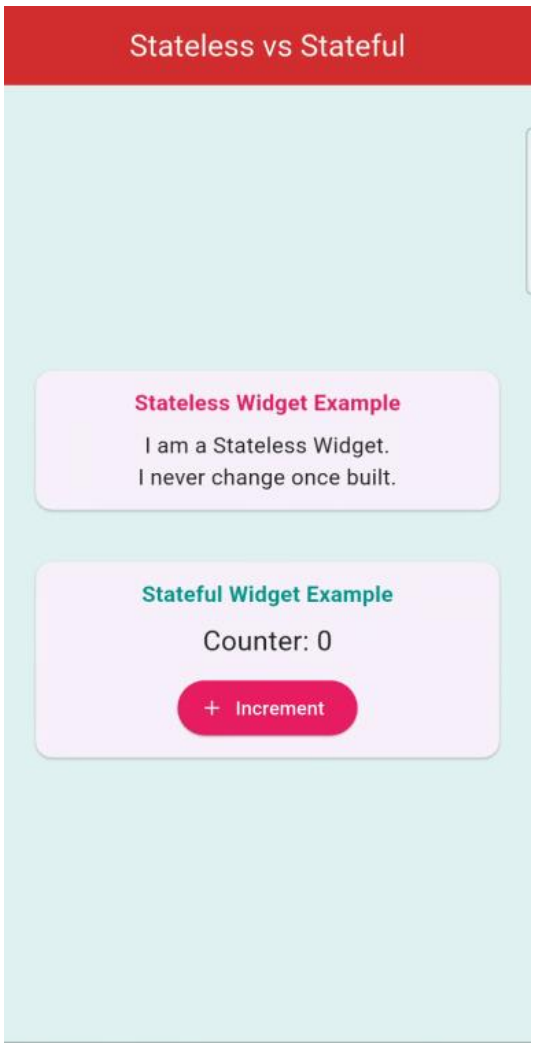
5. a) Learn about stateful and stateless widgets.

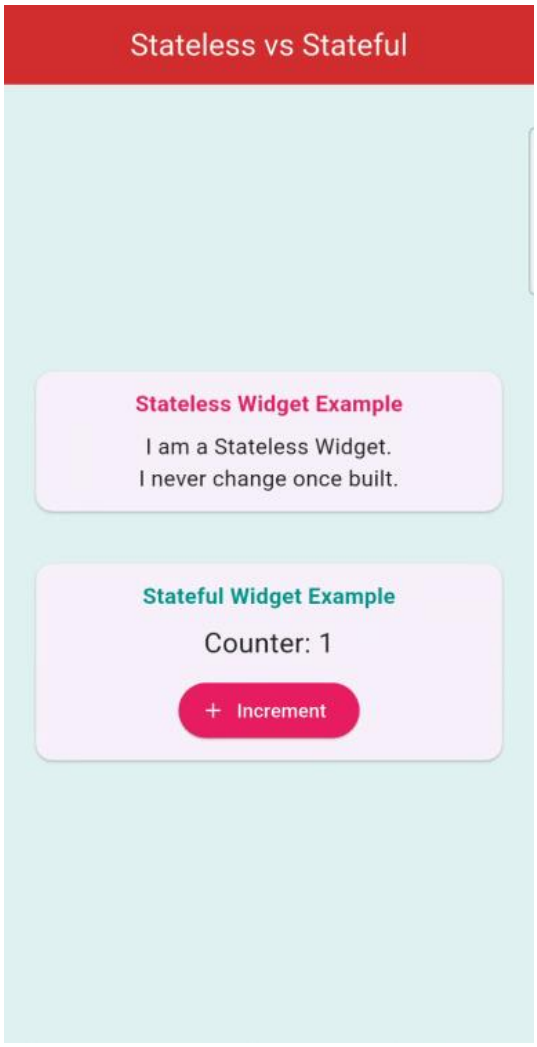
```
import 'package:flutter/material.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Stateless vs Stateful Demo',
      debugShowCheckedModeBanner: false,
      home: const HomePage(),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Stateless vs Stateful'),
        centerTitle: true,
        backgroundColor: const Color(0xFFD32F2F),
        foregroundColor: Colors.white,
      ),
      backgroundColor: Colors.teal[50],
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            StaticTextWidget(),
            const SizedBox(height: 30),
            CounterWidget(),
          ],
        ),
      ),
    );
```

```
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Card(
      child: Center(
        child: Padding(padding: EdgeInsets.all(12),
          child: Column(
            children: [
              Text('Stateless Widget Example',
                  style: TextStyle(
                      fontSize: 16,
                      fontWeight: FontWeight.bold,
                      color: Color(0xFFE91E63),
                  )
              ),
              const SizedBox(height: 8),
              Text('I am a Stateless Widget.\nI never change once built.',
                textAlign: TextAlign.center,
                style: TextStyle(fontSize: 16, color: Colors.black87
                ),
              )
            ],
          ),
        ),
      ),
    );
  }
}


class CounterWidget extends StatefulWidget {
  const CounterWidget({super.key});

  @override
  State<CounterWidget> createState() => _CounterWidgetState();
}

class _CounterWidgetState extends State<CounterWidget> {
  int count = 0;

  void incrementCounter() {
    setState(() {
      count++;
    });
  }
```

```dart
  @override
  Widget build(BuildContext context) {
    return Card(
      child: Center(
        child: Padding(padding: EdgeInsets.all(12),
          child: Column(
            children: [
              Text('StateFull Widget Example',
                  style: TextStyle(
                    fontSize: 16,
                    fontWeight: FontWeight.bold,
                    color: Color(0xFF009688),
                  )
              ),
              const SizedBox(height: 8),
              Text(
                'Counter: $count',
                style: const TextStyle(fontSize: 20, color: Colors.black),
              ),
              const SizedBox(height: 10),
              ElevatedButton.icon(
                onPressed: incrementCounter,
                icon: const Icon(Icons.add),
                label: const Text('Increment'),
                style: ElevatedButton.styleFrom(
                  backgroundColor: const Color(0xFFE91E63),
                  foregroundColor: Colors.white,
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

Output:


Stateless vs Stateful
Stateless Widget Example
I am a Stateless Widget.
I never change once built.
Stateful Widget Example
Counter: 0
+ Increment


Stateless vs Stateful
Stateless Widget Example
I am a Stateless Widget.
I never change once built.
Stateful Widget Example
Counter: 1
+ Increment

5. b) Implement state management using set State and Provider.

Pubspec.yaml

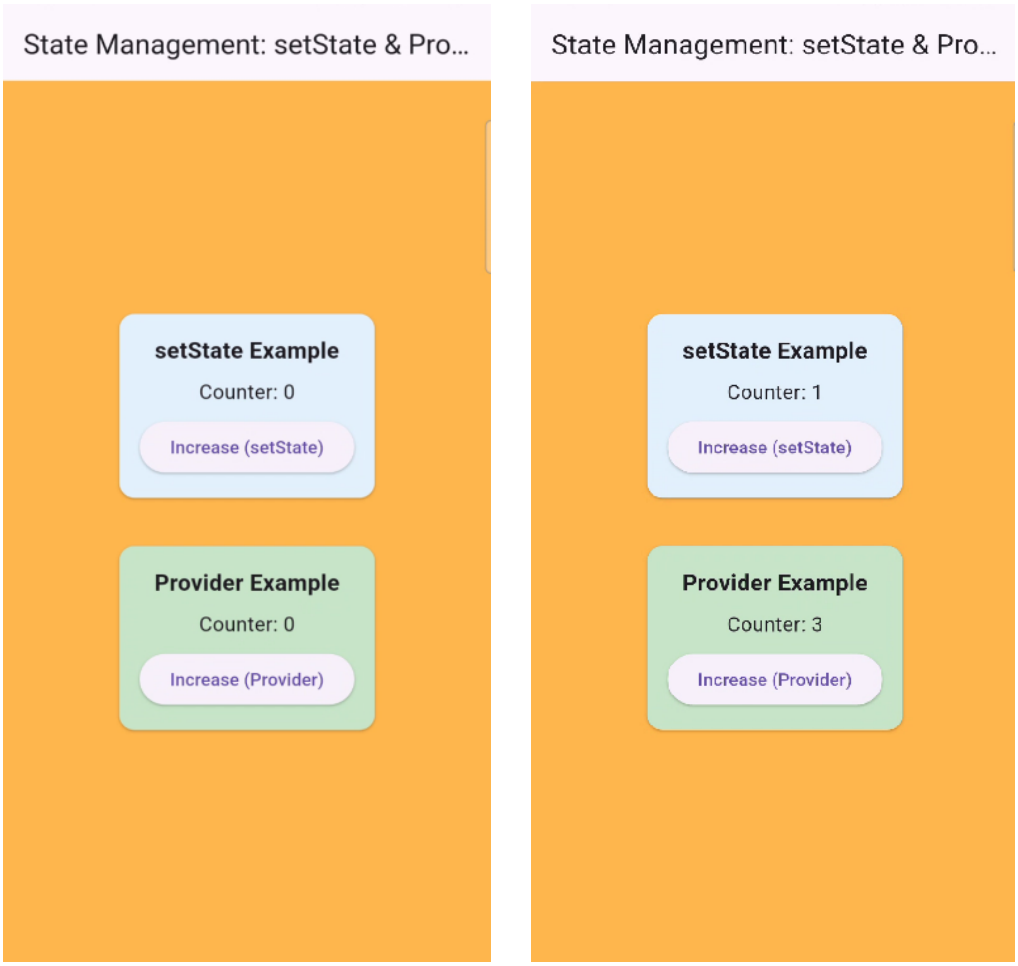
```yaml
dependencies:
  flutter:
    sdk: flutter

  provider: ^6.1.1  # ☑ Add provider here

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.8

dev_dependencies:
  flutter_test:
    sdk: flutter
```

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

// ------ Provider Model ------
class CounterModel extends ChangeNotifier {
  int _count = 0;
  int get count => _count;

  void increment() {
    _count++;
    notifyListeners();
  }
}

void main() {
  runApp(
    ChangeNotifierProvider(
      create: (_) => CounterModel(),
      child: const MyApp(),
    ),
  );
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: const HomePage(),
      debugShowCheckedModeBanner: false,
    );
  }
```

```
}

class HomePage extends StatefulWidget {
  const HomePage({super.key});
  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  int localCounter = 0; // For setState example

  @override
  Widget build(BuildContext context) {
    final providerCounter = Provider.of<CounterModel>(context);

    return Scaffold(
      appBar: AppBar(title: const Text('State Management: setState & Provider')),
      body: Padding(
        padding: const EdgeInsets.all(20),
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              // setState Example
              Card(
                color: Colors.blue.shade50,
                child: Padding(
                  padding: const EdgeInsets.all(16),
                  child: Column(
                    children: [
                      const Text(
                        'setState Example',
                        style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
                      ),
                      const SizedBox(height: 8),
                      Text('Counter: $localCounter', style: const TextStyle(fontSize: 16)),
                      const SizedBox(height: 8),
                      ElevatedButton(
                        onPressed: () => setState(() => localCounter++),
                        child: const Text('Increase (setState)'),
                      ),
                    ],
                  ),
                ),
              ),
              const SizedBox(height: 30),
              // Provider Example
              Card(
                color: Colors.green.shade100,
                child: Padding(
```

```
                padding: const EdgeInsets.all(16),
                child: Column(
                  children: [
                    const Text(
                      'Provider Example',
                      style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
                    ),
                    const SizedBox(height: 8),
                    Text('Counter: ${providerCounter.count}', style: const TextStyle(fontSize: 16)),
                    const SizedBox(height: 8),
                    ElevatedButton(
                      onPressed: () => providerCounter.increment(),
                      child: const Text('Increase (Provider)'),
                    ),
                  ],
                ),
              ),
            ),
          ),
        ],
      ),
    ),
  ),
);
  }
}
```

Output:


State Management: setState & Pro...
setState Example
Counter: 0
Increase (setState)
Provider Example
Counter: 0
Increase (Provider)
State Management: setState & Pro...
setState Example
Counter: 1
Increase (setState)
Provider Example
Counter: 3
Increase (Provider)

Experiment – 6

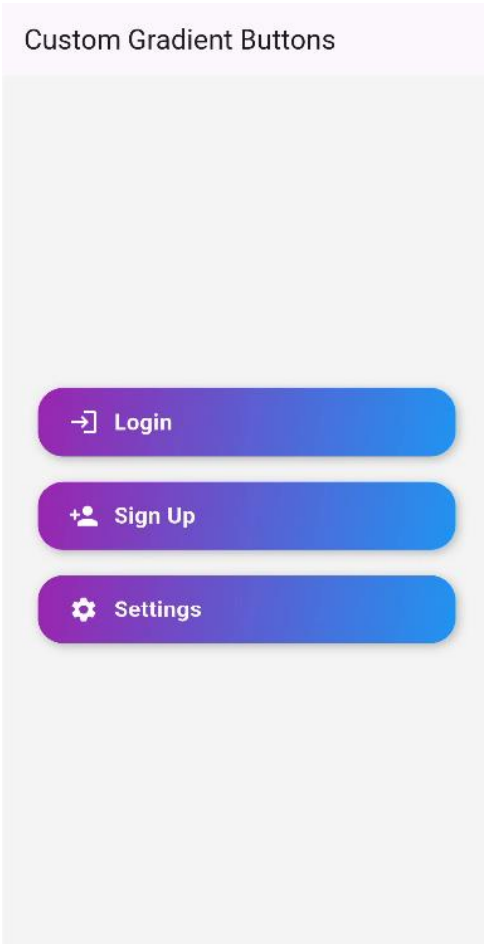
6. a) Create custom widgets for specific UI elements.

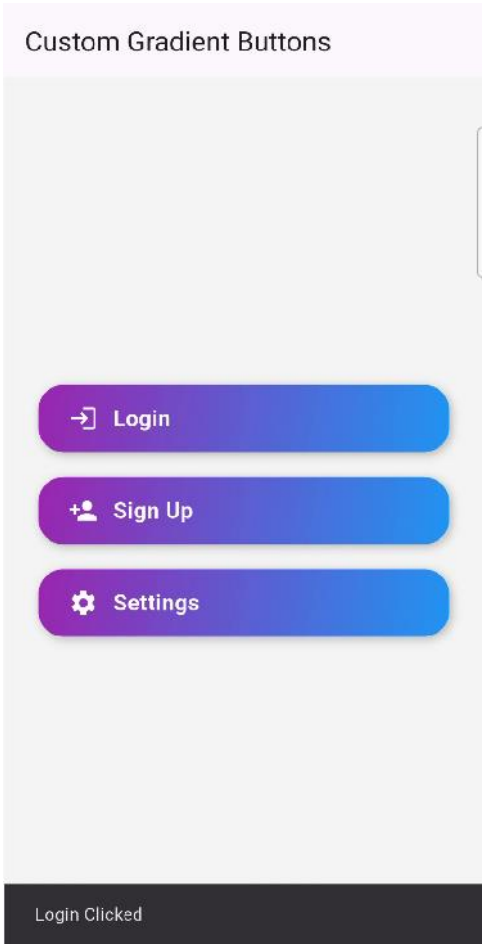
```
import 'package:flutter/material.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      debugShowCheckedModeBanner: false,
      home: HomeScreen(),
    );
  }
}

// Custom Widget: Gradient Button
class GradientButton extends StatelessWidget {
  final String text;
  final IconData icon;
  final VoidCallback onPressed;

  const GradientButton({
    super.key,
    required this.text,
    required this.icon,
    required this.onPressed,
  });

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: onPressed,
      child: Container(
        margin: const EdgeInsets.symmetric(vertical: 10),
        padding: const EdgeInsets.symmetric(horizontal: 24, vertical: 14),
        width: 330,
        decoration: BoxDecoration(
          gradient: const LinearGradient(
            colors: [Colors.purple, Colors.blue],
            begin: Alignment.topLeft,
            end: Alignment.bottomRight,
          ),
```

```
          borderRadius: BorderRadius.circular(20),
          boxShadow: const [
            BoxShadow(color: Colors.black26, blurRadius: 6, offset: Offset(2, 2)),
          ],
        ),
        child: Row(
          mainAxisSize: MainAxisSize.min,
          children: [
            Icon(icon, color: Colors.white, size: 24),
            const SizedBox(width: 12),
            Text(
              text,
              style: const TextStyle(color: Colors.white, fontSize: 18, fontWeight:
FontWeight.bold),
            ),
          ],
        ),
      ),
    );
  }
}

// Home Screen
class HomeScreen extends StatelessWidget {
  const HomeScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[100],
      appBar: AppBar(title: const Text("Custom Gradient Buttons")),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            GradientButton(
              text: "Login",
              icon: Icons.login,
              onPressed: () {
                ScaffoldMessenger.of(context).showSnackBar(
                  const SnackBar(content: Text("Login Clicked")),
                );
              },
            ),
            GradientButton(
              text: "Sign Up",
              icon: Icons.person_add,
              onPressed: () {
                ScaffoldMessenger.of(context).showSnackBar(
                  const SnackBar(content: Text("Sign Up Clicked")),
```

```
            );
          },
        ),
        GradientButton(
          text: "Settings",
          icon: Icons.settings,
          onPressed: () {
            ScaffoldMessenger.of(context).showSnackBar(
              const SnackBar(content: Text("Settings Clicked")),
            );
          },
        ),
      ],
    ),
  ),
 );
 }
}
```

Output:





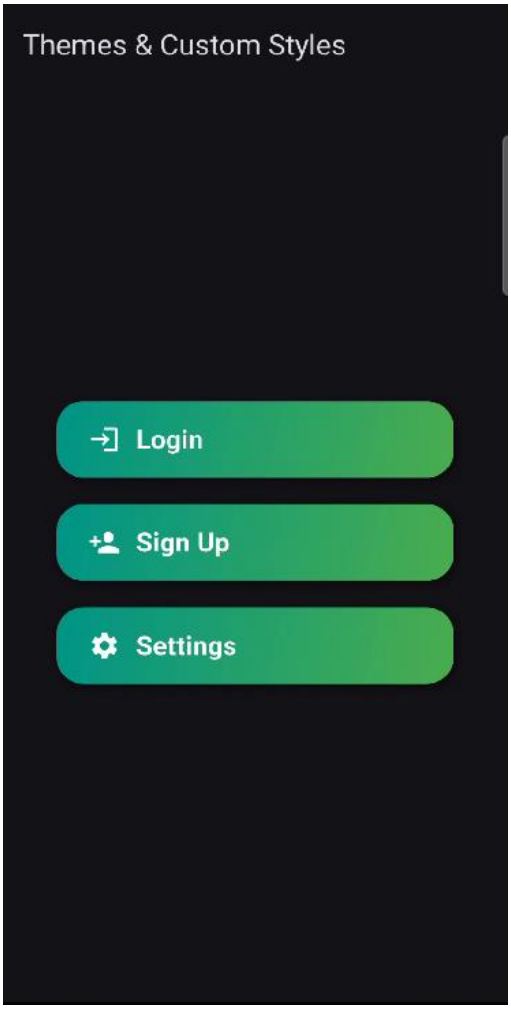
6. b) Apply styling using themes and custom styles.

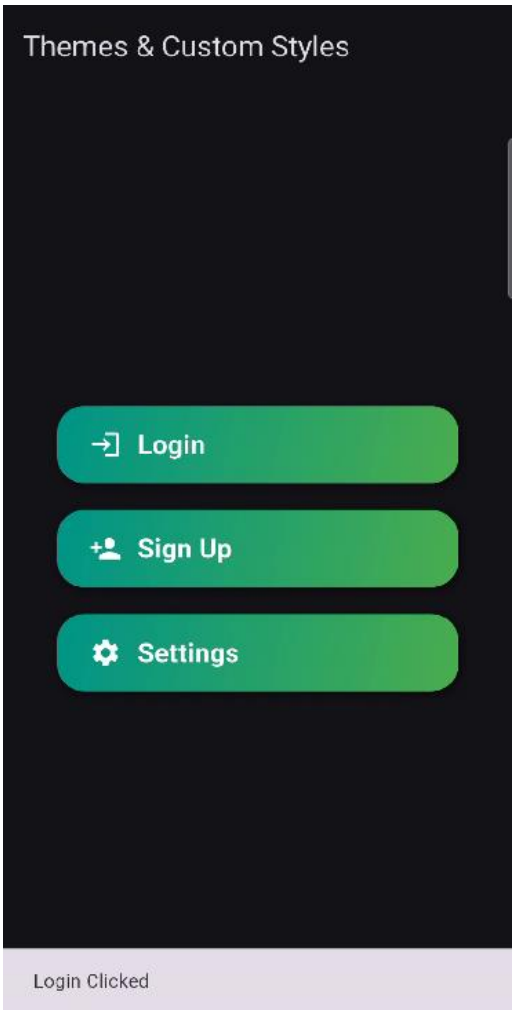
```
import 'package:flutter/material.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        brightness: Brightness.dark,
        textTheme: const TextTheme(
          titleMedium: TextStyle(fontSize: 20, fontWeight: FontWeight.bold, color:
Colors.white),
        ),
      ),
      home: const HomeScreen(),
    );
  }
}

// Custom Widget (uses Theme for styling)
class GradientButton extends StatelessWidget {
  final String text;
  final IconData icon;
  final VoidCallback onPressed;

  const GradientButton({
    super.key,
    required this.text,
    required this.icon,
    required this.onPressed,
  });

  @override
  Widget build(BuildContext context) {
    final textStyle = Theme.of(context).textTheme.titleMedium; // use theme style

    return GestureDetector(
      onTap: onPressed,
      child: Container(
        margin: const EdgeInsets.symmetric(vertical: 10),
        padding: const EdgeInsets.symmetric(horizontal: 24, vertical: 14),
        width: 300,
```

```
        decoration: BoxDecoration(
          gradient: const LinearGradient(colors: [Colors.teal,Colors.green],
            begin: Alignment.topLeft,
            end: Alignment.bottomRight,
          ),
          borderRadius: BorderRadius.circular(20),
          boxShadow: const [
            BoxShadow(color: Colors.black26, blurRadius: 6),
          ],
        ),
        child: Row(
          mainAxisSize: MainAxisSize.min,
          children: [
            Icon(icon, color: Colors.white, size: 24),
            const SizedBox(width: 12),
            Text(text, style: textStyle), // style comes from Theme
          ],
        ),
      ),
    );
  }
}

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

  void _showMessage(BuildContext context, String message) {
    ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text(message)));
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Themes & Custom Styles")),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            GradientButton(
              text: "Login",
              icon: Icons.login,
              onPressed: () => _showMessage(context, "Login Clicked"),
            ),
            GradientButton(
              text: "Sign Up",
              icon: Icons.person_add,
              onPressed: () => _showMessage(context, "Sign Up Clicked"),
            ),
            GradientButton(
              text: "Settings",
```

```
            icon: Icons.settings,
            onPressed: () => _showMessage(context, "Settings Clicked"),
          ),
        ],
      ),
    ),
  );
 }
}
```

Output:





Experiment – 7

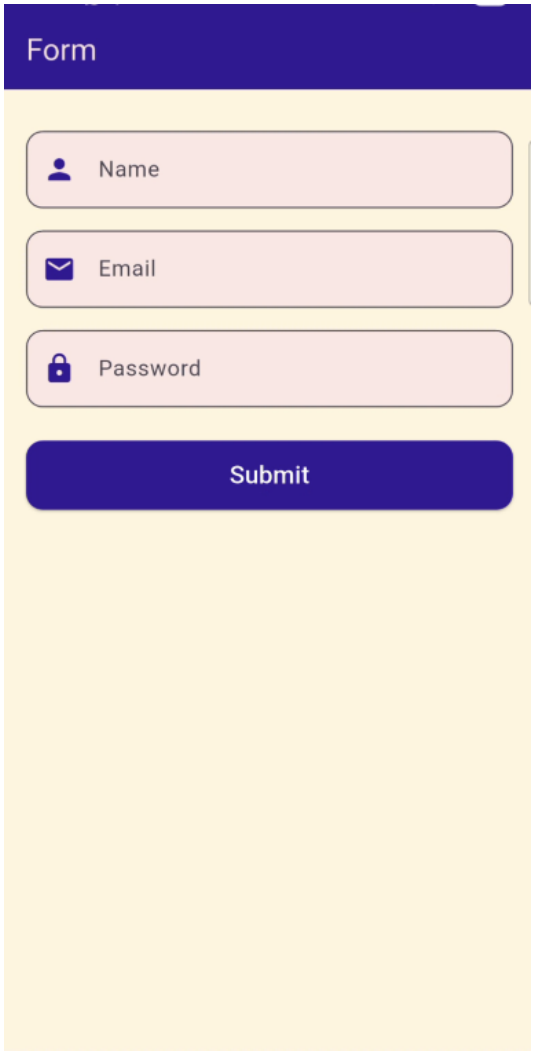
7. a) Design a form with various input fields.
b) Implement form validation and error handling.

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Form Example',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        inputDecorationTheme: InputDecorationTheme(
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(12),
          ),
          filled: true,
          fillColor: Colors.deepOrange[50],
        ),
      ),
      home: MyForm(),
    );
  }
}

class MyForm extends StatefulWidget {
  @override
  _MyFormState createState() => _MyFormState();
}

class _MyFormState extends State<MyForm> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController _nameController = TextEditingController();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.deepPurple[900],
        foregroundColor: Colors.deepOrange[50],
        title: Text('Form'),
```

```
),
backgroundColor: Colors.amber.shade50,
body: Padding(
  padding: EdgeInsets.symmetric(vertical: 30, horizontal: 16),
  child: Form(
    key: _formKey,
    child: Column(
      children: <Widget>[
        TextFormField(
          controller: _nameController,
          decoration: InputDecoration(
            labelText: 'Name',
            prefixIcon: Icon(Icons.person, color: Colors.deepPurple[900]),
          ),
          validator: (value) {
            if (value == null || value.isEmpty) {
              return 'Please enter your name';
            }
            return null;
          },
        ),
        SizedBox(height: 16),

        TextFormField(
          controller: _emailController,
          keyboardType: TextInputType.emailAddress,
          decoration: InputDecoration(
            labelText: 'Email',
            prefixIcon: Icon(Icons.email, color: Colors.deepPurple[900]),
          ),
          validator: (value) {
            if (value == null || value.isEmpty) {
              return 'Please enter your email';
            } else if (!RegExp(r'^[\w-]+(\.[\w-]+)*@[\w-]+(\.[\w-]+)+$')
                .hasMatch(value)) {
              return 'Please enter a valid email address';
            }
            return null;
          },
        ),
        SizedBox(height: 16),

        TextFormField(
          controller: _passwordController,
          obscureText: true,
          decoration: InputDecoration(
            labelText: 'Password',
            prefixIcon: Icon(Icons.lock, color: Colors.deepPurple[900]),
          ),
          validator: (value) {
```

```
                if (value == null || value.isEmpty) {
                  return 'Please enter your password';
                } else if (value.length < 6) {
                  return 'Password must be at least 6 characters long';
                }
                return null;
              },
            ),
            SizedBox(height: 24),

            ElevatedButton.icon(
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.deepPurple[900],
                minimumSize: Size(double.infinity, 50),
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(12),
                ),
              ),
              onPressed: () {
                if (_formKey.currentState!.validate()) {
                  print('Name: ${_nameController.text}');
                  print('Email: ${_emailController.text}');
                  print('Password: ${_passwordController.text}');
                  ScaffoldMessenger.of(context).showSnackBar(
                    SnackBar(
                      content: Text('Form submitted successfully!'),
                      backgroundColor: Colors.green,
                    ),
                  );
                }
              },
              label: Text(
                'Submit',
                style: TextStyle(fontSize: 18, color: Colors.white),
              ),
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

Output:


Form
Name
Email
Password
Submit


Form
Name
David
Email
David@gmail.com
Password
Submit
Form submitted successfully!

Experiment – 8

8. a) Add animations to UI elements using Flutter's animation framework.
b) Experiment with different types of animations (fade, slide, etc.).

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Animation',
      home: AnimationExample(),
    );
  }
}

class AnimationExample extends StatefulWidget {
  @override
  _AnimationExampleState createState() => _AnimationExampleState();
}

class _AnimationExampleState extends State<AnimationExample>
    with SingleTickerProviderStateMixin {
  bool _fadeVisible = false;
  bool _slideVisible = false;

  late AnimationController _controller;
  late Animation<Offset> _slideAnimation;

  @override
  void initState() {
    super.initState();
    _controller =
        AnimationController(vsync: this, duration: Duration(milliseconds: 500));
    _slideAnimation =
        Tween<Offset>(begin: Offset(0, 1), end: Offset(0, 0)).animate(
          CurvedAnimation(parent: _controller, curve: Curves.easeInOut),
        );
  }

  void _toggleFade() {
    setState(() {
      _fadeVisible = !_fadeVisible;
```

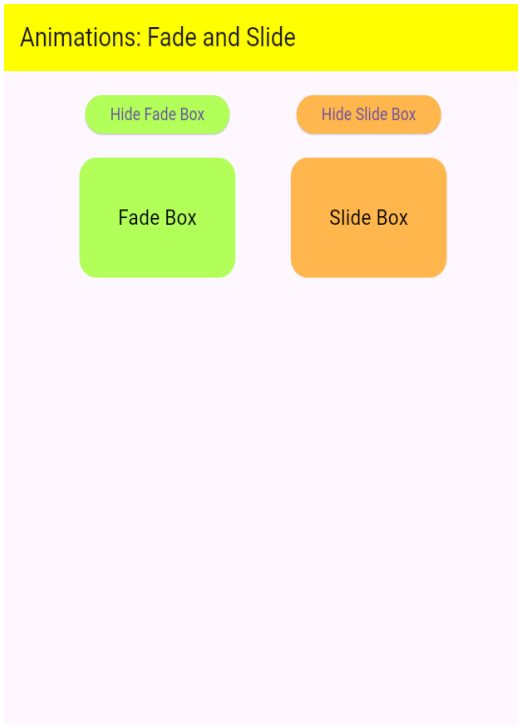
```
  });
}

void _toggleSlide() {
  setState(() {
    _slideVisible = !_slideVisible;
    if (_slideVisible) {
      _controller.forward();
    } else {
      _controller.reverse();
    }
  });
}

@override
void dispose() {
  _controller.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("Animations: Fade and Slide"),
      backgroundColor: Colors.yellowAccent,
    ),
    body: Padding(
      padding: const EdgeInsets.all(20.0),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          // Left Column - Fade
          Column(
            children: [
              ElevatedButton(
                onPressed: _toggleFade,
                child:
                Text(_fadeVisible ? "Hide Fade Box" : "Show Fade Box"),
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.lightGreenAccent,
                ),
              ),
              SizedBox(height: 20),
              AnimatedOpacity(
                opacity: _fadeVisible ? 1.0 : 0.0,
                duration: Duration(seconds: 1),
                child: Container(
                  width: 150,
```

```
            height: 100,
            decoration: BoxDecoration(
              color: Colors.lightGreenAccent,
              borderRadius: BorderRadius.circular(16),
            ),
            child: Center(
              child: Text(
                "Fade Box",
                style: TextStyle(fontSize: 18, color: Colors.black),
              ),
            ),
          ),
        ),
      ],
    ),

    // Right Column - Slide
    Column(
      children: [
        ElevatedButton(
          onPressed: _toggleSlide,
          child:
          Text(_slideVisible ? "Hide Slide Box" : "Show Slide Box"),
          style: ElevatedButton.styleFrom(
            backgroundColor: Colors.orange[300],
          ),
        ),
        SizedBox(height: 20),
        SlideTransition(
          position: _slideAnimation,
          child: Container(
            width: 150,
            height: 100,
            decoration: BoxDecoration(
              color: Colors.orange[300],
              borderRadius: BorderRadius.circular(16),
            ),
            child: Center(
              child: Text(
                "Slide Box",
                style: TextStyle(fontSize: 18, color: Colors.black),
              ),
            ),
          ),
        ),
      ],
    ),
  ],
 ),
),
```

```
    );
  }
}
```

Output:


Animations: Fade and Slide
Show Fade Box
Show Slide Box
Slide Box


Animations: Fade and Slide
Hide Fade Box
Hide Slide Box
Fade Box
Slide Box

Experiment – 9

9. a) Fetch data from a REST API.

b) Display the fetched data in a meaningful way in the UI.


```dart
import 'package:flutter/material.dart';
import 'dart:convert';
import 'package:http/http.dart' as http;

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'REST API Demo',
      home: PostListScreen(),
    );
  }
}

class PostListScreen extends StatefulWidget {
  @override
  _PostListScreenState createState() => _PostListScreenState();
}

class _PostListScreenState extends State<PostListScreen> {
  Future<List<dynamic>> fetchPosts() async {
    final response =
    await http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts'));

    if (response.statusCode == 200) {
      return json.decode(response.body); // convert JSON to List
    } else {
      throw Exception('Failed to load posts');
    }
  }
```

```dart
@override
Widget build(BuildContext context) {
 return Scaffold(
  appBar: AppBar(
   title: Text("REST API Data"),
   centerTitle: true,
  ),
  body: FutureBuilder<List<dynamic>>(
   future: fetchPosts(),
   builder: (context, snapshot) {
    if (snapshot.connectionState == ConnectionState.waiting) {
     return Center(child: CircularProgressIndicator()); // Loading spinner
    } else {
     return ListView.builder(
      itemCount: snapshot.data!.length,
      itemBuilder: (context, index) {
       var post = snapshot.data![index];
       return Card(
        elevation: 4,
        margin: EdgeInsets.symmetric(horizontal: 12, vertical: 8),
        shape: RoundedRectangleBorder(
         borderRadius: BorderRadius.circular(15),
        ),
        child: ListTile(
         leading: CircleAvatar(
          backgroundColor: Colors.blue[900],
          child: Text(
           post['id'].toString(),
           style: TextStyle(color: Colors.white),
          ),
         ),
         title: Text(
          post['title'],
          style: TextStyle(
              fontWeight: FontWeight.bold, color: Colors.black87),
         ),
         subtitle: Text(
          post['body'],
          maxLines: 2,
          overflow: TextOverflow.ellipsis,
          style: TextStyle(color: Colors.black54),
         ),
        ),
       );
      },
     );
    }
   },
  ),
```

```
    );
  }
}
```

Output:


REST API Data
DEBUG
1
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
quia et suscipit
suscipit recusandae consequuntur expedita et cum
2
qui est esse
est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
3
ea molestias quasi exercitationem repellat qui ipsa sit aut
et iusto sed quo iure
voluptatem occaecati omnis eligendi aut ad
4
eum et est occaecati
ullam et saepe reiciendis voluptatem adipisci
sit amet autem assumenda provident rerum culpa
5
nesciunt quas odio
repudiandae veniam quaerat sunt sed
alias aut fugiat sit autem sed est
6
dolorem eum magni eos aperiam quia
ut aspernatur corporis harum nihil quis provident sequi
mollitia nobis aliquid molestiae

Experiment – 10

10. a) Write unit tests for UI components.

How can you ensure that your app continues to work as you add more features or change existing functionality? By writing tests.

Unit tests are handy for verifying the behavior of a single function, method, or class. The test package provides the core framework for writing unit tests, and the flutter_test package provides additional utilities for testing widgets.

This experiment demonstrates the core features provided by the test package using the following steps:

1. Add the test or flutter_test dependency.
2. Create a test file.
3. Create a class to test.
4. Write a test for our class.
5. Combine multiple tests in a group.
6. Run the tests.

1. Add the test dependency

The test package provides the core functionality for writing tests in Dart. This is the best approach when writing packages consumed by web, server, and Flutter apps.

- Open your project in Android Studio.
- At the bottom of the IDE, click the Terminal tab (next to Run, Debug, Logcat).
- To add the test package as a dev dependency, run flutter pub add:
  Type this command and press Enter:

  flutter pub add dev:test

- Android Studio will update your pubspec.yaml automatically. You don't need to edit manually.
- After running, check pubspec.yaml and you'll see under dev_dependencies: something like:

```
38    dev_dependencies:
39      flutter_test:
40        sdk: flutter
41
42      # The "flutter_lints" package below contains a set of recommended lints to
43      # encourage good coding practices. The lint set provided by the package is
44      # activated in the `analysis_options.yaml` file located at the root of your
45      # package. See that file for information about deactivating specific lint
46      # rules and activating additional ones.
47      flutter_lints: ^5.0.0
48      test: ^1.25.15
```

- Run the Pub get.

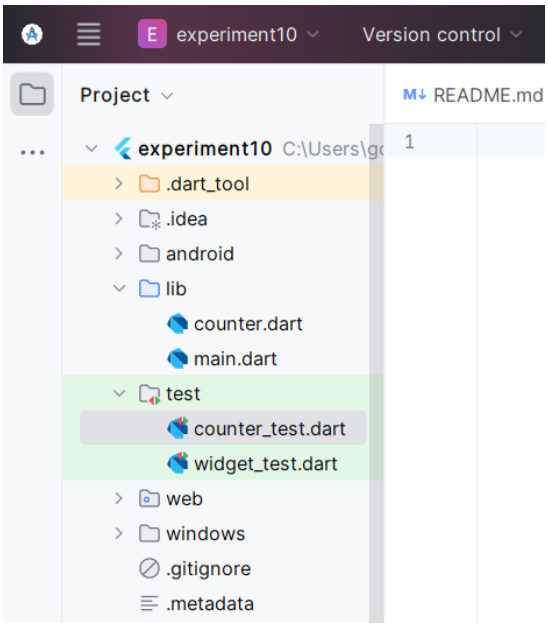
2. Create a test file

In this example, create two files: counter.dart and counter_test.dart.

The counter.dart file contains a class that you want to test and resides in the lib folder. The counter_test.dart file contains the tests themselves and lives inside the test folder.

In general, test files should reside inside a test folder located at the root of your Flutter application or package. Test files should always end with _test.dart, this is the convention used by the test runner when searching for tests.

When you're finished, the folder structure should look like this:



3. Create a class to test

Next, you need a "unit" to test. Remember: "unit" is another name for a function, method, or class. For this example, create a Counter class inside the lib/counter.dart file. It is responsible for incrementing and decrementing a value starting at 0.

```
class Counter {
  int value = 0;                    // A variable that starts with value 0

  void increment() => value++;      // Increases the value by 1
  void decrement() => value--;      // Decreases the value by 1
}
```

4. Write a test for our class

Inside the counter_test.dart file, write the first unit test. Tests are defined using the top-level test function, and you can check if the results are correct by using the top-level expect function. Both of these functions come from the test package.

```
// Import the test package and Counter class
import 'package:experiment10/counter.dart';
```

```dart
import 'package:test/test.dart';

void main() {
  test('Counter value should be incremented', () {
    final counter = Counter();  // Step 1: Create a Counter object
    counter.increment();        // Step 2: Call increment()
    expect(counter.value, 1);   // Step 3: Verify that value == 1
  });
}
```

5. Combine multiple tests in a group

If you want to run a series of related tests, use the flutter_test package group function to categorize the tests. Once put into a group, you can call flutter test on all tests in that group with one command.

```dart
import 'package:counter_app/counter.dart';
import 'package:test/test.dart';

void main() {
  group('Test start, increment, decrement', () {
    test('value should start at 0', () {
      expect(Counter().value, 0);
    });
    test('value should be incremented', () {
      final counter = Counter();
      counter.increment();
      expect(counter.value, 1);
    });
    test('value should be decremented', () {
      final counter = Counter();
      counter.decrement();
      expect(counter.value, -1);
    });
  });
}
```

6. Run the tests

You have written a Counter class (inside lib/counter.dart) and a test file (test/counter_test.dart). Now you need to execute these tests to verify that everything works.

The Flutter plugins for IntelliJ and terminal support running tests. This is often the best option while writing tests because it provides the fastest feedback loop as well as the ability to set breakpoints.

Option 1: Run from Android Studio (IntelliJ)

1. Open the file: test/counter_test.dart.

2. Right-click anywhere inside the editor.
3. Select Run 'tests in counter_test.dart'.
4. The test results will appear at the bottom in the Run tab.
    - Passed tests show with a green check.
    - Failed tests show with a red cross.

Option 2: Run from Terminal

Open the terminal inside Android Studio and type:

flutter test test/counter_test.dart

This will run all the tests inside that file.

If you want to run only the group we created in Step 5, use:

flutter test --plain-name "Test start, increment, decrement"

After running, you should see output like this:

00:00 +3: All tests passed!

This means your 3 tests (start, increment, decrement) worked correctly.

10. b) Use Flutter's debugging tools to identify and fix issues.

Debugging is the process of finding and fixing errors or bugs in code. Flutter provides a set of integrated debugging tools that make it easier to track UI problems, logic errors, and performance issues in real time.

Some of the key debugging tools in Flutter are:

1. **Debug Console (Run Window):**
   Displays logs, exceptions, and print statements from your app. It helps you understand what happens during execution.
2. **Hot Reload / Hot Restart:**
    - Hot Reload quickly updates changes in the code without restarting the app.
    - Hot Restart reloads the whole app, resetting its state. Both features speed up the debugging process.
3. **Flutter Inspector:**
   A powerful tool (found in Android Studio or VS Code) that lets you:
    - Visually inspect widget trees.
    - Highlight UI elements.
    - Check widget properties and constraints.
    - Debug layout issues (for example, overflowing widgets).
4. **Breakpoints and Step Debugging:**
   Breakpoints allow pausing execution at specific lines. Using Step Over, Step Into, and Step Out options, you can trace program logic line by line.
5. **DevTools (Performance Tools):**

- Used for analyzing app performance and memory usage.
- Helps detect frame rendering issues and jank.

Procedure:

1. Open your Flutter project in Android Studio.
2. Go to the Run menu → select Debug 'main.dart'.
3. Observe the Debug Console for any logs or exceptions.
4. Open Flutter Inspector (from the right panel) → click on any widget to inspect its properties.
5. Add a breakpoint in your Dart file (e.g., inside a button's onPressed() function).
6. Run the app in Debug Mode. When execution pauses at the breakpoint, use Step Over or Step Into to trace the flow.
7. Fix the identified issues by modifying the code, then use Hot Reload to view changes instantly.

Example Demonstration:

```
// Example: Debugging a simple counter issue
class Counter {
  int value = 0;
  void increment() {
    value++; // Set a breakpoint here
    print("Counter incremented to $value");
  }
}
```

If the value does not update correctly, you can:

- Check print outputs in the console.
- Inspect the widget displaying the value.
- Verify state management or logic using the debugger.

Output:

- Debug console shows live log messages (Counter incremented to 1).
- Flutter Inspector highlights the widget hierarchy.
- Breakpoint pauses the app at the increment line for inspection.

Result:

The debugging tools in Flutter were successfully used to identify logical and UI issues. Breakpoints, Inspector, and Hot Reload were demonstrated effectively to trace and fix problems in the application.

Additional Experiments

Experiment - 11

11. Integration of Google Maps and Displaying User's Current Location

Objective:

To develop a Flutter application that integrates Google Maps and displays the user's current geographical location dynamically using the google_maps_flutter and geolocator packages.

Prerequisites / Tools Used:

- Flutter SDK – Framework for cross-platform app development.
- google_maps_flutter – Plugin for embedding Google Maps in Flutter apps.
- geolocator – Package for accessing location services.
- Android Studio / VS Code – IDE for Flutter app development.
- Google Cloud Console – To generate API key for Google Maps SDK.

Brief Description:

This experiment introduces how to use Google Maps in a Flutter application and track the user's current location. It helps students understand how to:

- Request location permissions.
- Access GPS coordinates.
- Display live location using map markers.
- Integrate an API key for Google Maps.

Such skills are vital in real-world applications like delivery tracking, ride-sharing, and navigation-based apps.

Step 1: Add Dependencies in pubspec.yaml

```
36      cupertino_icons: ^1.0.8
37
38    dev_dependencies:
39      flutter_test:
40        sdk: flutter
41      google_maps_flutter: ^2.6.0
42      geolocator: ^13.0.1
43      permission_handler: ^11.0.1
44      # The "flutter_lints" package below contains a set of recommended lints to
```

Run: Pub get

Step 2: Enable Location Permission in AndroidManifest.xml

Add this before <application> tag:

<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

<uses-permission android:name="android.permission.INTERNET"/>

Add this inside <application> tag:

<meta-data

android:name="com.google.android.geo.API_KEY"

android:value="YOUR_API_KEY_HERE"/>

(Replace YOUR_API_KEY_HERE with your Google Maps API key from Google Cloud Console.)

Step 3: Main Code (main.dart)

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

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

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

 @override
 Widget build(BuildContext context) {
  return const MaterialApp(
   debugShowCheckedModeBanner: false,
   home: GoogleMapScreen(),
  );
 }
}

class GoogleMapScreen extends StatefulWidget {
 const GoogleMapScreen({super.key});

 @override
 State<GoogleMapScreen> createState() => _GoogleMapScreenState();
}
class _GoogleMapScreenState extends State<GoogleMapScreen> {
 GoogleMapController? _controller;
 LatLng? _currentPosition;
 final Set<Marker> _markers = {};
 Future<void> _getCurrentLocation() async {
  bool serviceEnabled;
  LocationPermission permission;
```

```
  serviceEnabled = await Geolocator.isLocationServiceEnabled();
  if (!serviceEnabled) {
   return Future.error('Location services are disabled.');
  }

  permission = await Geolocator.checkPermission();
  if (permission == LocationPermission.denied) {
   permission = await Geolocator.requestPermission();
   if (permission == LocationPermission.denied) {
    return Future.error('Location permissions are denied.');
   }
  }

  Position position = await Geolocator.getCurrentPosition(
      desiredAccuracy: LocationAccuracy.high);

  setState(() {
   _currentPosition = LatLng(position.latitude, position.longitude);
   _markers.clear();
   _markers.add(Marker(
     markerId: const MarkerId('currentLocation'),
     position: _currentPosition!,
     infoWindow: const InfoWindow(title: "You are here"),
   ));
  });
  _controller?.animateCamera(CameraUpdate.newLatLngZoom(_currentPosition!, 15));
 }

 @override
 void initState() {
  super.initState();
  _getCurrentLocation();
 }
 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
     title: const Text('Google Maps - Current Location'),
     backgroundColor: Colors.blueAccent,
     foregroundColor: Colors.white,
    ),
    body: _currentPosition == null
       ? const Center(child: CircularProgressIndicator())
       : GoogleMap(
```

```
      onMapCreated: (controller) => _controller = controller,
      initialCameraPosition: CameraPosition(
        target: _currentPosition!,
        zoom: 15,
      ),
      markers: _markers,
      myLocationEnabled: true,
      compassEnabled: true,
    ),
    floatingActionButton: FloatingActionButton.extended(
      onPressed: _getCurrentLocation,
      label: const Text('Locate Me'),
      icon: const Icon(Icons.my_location),
    ),
  );
 }
}
```

Output:

Expected Output Screenshots:

1. **App launch screen** – Google Map displayed with zoom controls.
2. **User's current location** shown with a custom marker.
3. **"Locate Me"** button updates the map view dynamically.

Explanation of Tools/Plugins:

Tool	Purpose
google_maps_flutter	Embeds Google Maps widget and supports markers, zoom, gestures.
geolocator	Provides location access and GPS coordinates.
permission_handler	Helps manage Android/iOS runtime permissions.

Alignment with Industry Skills:

- Integrating Google Maps and real-time GPS data is a **core feature in modern apps** like Uber, Zomato, and logistics tracking apps.
- Teaches **API integration**, **location services**, and **real-time interactivity**, making it an *industry-relevant, beyond-curriculum addition*.
- Encourages students to explore **cloud services**, **data privacy**, and **mobile UX optimization**.

Experiment - 12

14. Capture and Preview Image Using Camera in Flutter

Objective:

To develop a Flutter application that accesses the device's camera to capture an image and then displays the captured image within the app interface.

Prerequisites / Tools Used:

- Flutter SDK – For cross-platform app development.
- camera package – Provides camera access to take photos.
- path_provider – Helps to get the directory for storing captured images.
- Android Studio / VS Code – IDE for app creation and debugging.

Brief Description:

This experiment introduces how to integrate a camera in a Flutter app. The user can open the camera, take a photo, and instantly view it within the same app. It helps students understand hardware integration, asynchronous programming, and widget tree updates based on user actions.

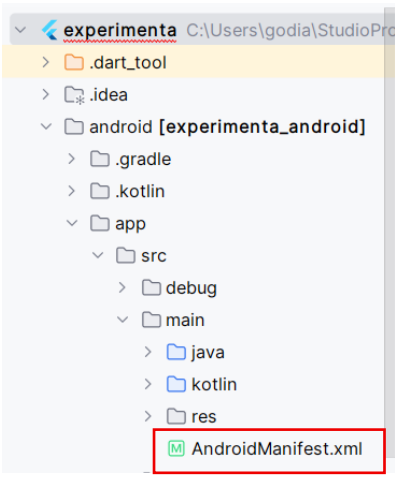
Step 1: Add Dependencies in pubspec.yaml

```
36      cupertino_icons: ^1.0.8
37
38    dev_dependencies:
39      flutter_test:
40        sdk: flutter
41      camera: ^0.11.0+1
42      path_provider: ^2.1.3
43      path: ^1.9.0
44
45      # The "flutter_lints" package below contains a set of recommended lints to
```

Run: flutter pub get

Step 2: Add Required Permissions

In AndroidManifest.xml



Add these before the <application> tag:

<uses-permission android:name="android.permission.CAMERA"/>

<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

Step 3: Main Program – main.dart

```
import 'package:flutter/material.dart';
import 'package:camera/camera.dart';
import 'dart:io';
import 'package:path/path.dart' show join;
import 'package:path_provider/path_provider.dart';

List<CameraDescription>? cameras;
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  cameras = await availableCameras(); // Get available cameras
  runApp(const CameraApp());
}

class CameraApp extends StatelessWidget {
  const CameraApp({super.key});
  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      debugShowCheckedModeBanner: false,
      home: CameraHome(),
    );
  }
}

class CameraHome extends StatefulWidget {
  const CameraHome({super.key});

  @override
  State<CameraHome> createState() => _CameraHomeState();
}

class _CameraHomeState extends State<CameraHome> {
  CameraController? controller;
  XFile? capturedImage;
  @override
  void initState() {
    super.initState();
    controller = CameraController(cameras![0], ResolutionPreset.medium);
```

```
  controller!.initialize().then((_) {
    if (!mounted) return;
    setState(() {});
  });
}

@override
void dispose() {
  controller?.dispose();
  super.dispose();
}

Future<void> capturePhoto() async {
  if (!controller!.value.isInitialized) return;
  final directory = await getApplicationDocumentsDirectory();
  final filePath = join(directory.path, '${DateTime.now()}.png');
  final XFile file = await controller!.takePicture();
  setState(() {
    capturedImage = file;
  });
}

@override
Widget build(BuildContext context) {
  if (controller == null || !controller!.value.isInitialized) {
    return const Center(child: CircularProgressIndicator());
  }
  return Scaffold(
    appBar: AppBar(
      title: const Text('Camera Capture Example'),
      backgroundColor: Colors.teal,
    ),
    body: Column(
      children: [
        Expanded(
          flex: 3,
          child: CameraPreview(controller!),
        ),
        Expanded(
          flex: 2,
          child: Container(
            color: Colors.black12,
            child: Center(
              child: capturedImage == null
```

```
                ? const Text('No image captured yet')
                : Image.file(File(capturedImage!.path)),
            ),
          ),
        ),
        const SizedBox(height: 10),
        ElevatedButton.icon(
          onPressed: capturePhoto,
          icon: const Icon(Icons.camera_alt),
          label: const Text('Capture Photo'),
          style: ElevatedButton.styleFrom(
            backgroundColor: Colors.teal,
            foregroundColor: Colors.white,
            padding: const EdgeInsets.symmetric(horizontal: 25, vertical: 12),
          ),
        ),
        const SizedBox(height: 15),
      ],
    ),
  );
 }
}
```

Output: