MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

VISION & MISSION OF INSTITUTION

Vision:

To be as an ideal academic institution by graduating talented engineers to be ethically strong, competent with quality research and technologies.

Mission:

- Utilize rigorous educational experiences to produce talented engineers
- Create an atmosphere that facilitates the success of students
- Programs that integrate global awareness, communication skills and Leadership qualities
- Education and Research partnership with institutions and industries to prepare the students for interdisciplinary research

VISION & MISSION OF DEPARTMENT

Vision:

To empower the students to be technologically adept, innovative, self-motivated and responsible global citizen possessing human values and contribute significantly towards high quality technical education with ever changing world.

Mission:

- To offer high-quality education in the computing fields by providing an environment where the knowledge is gained and applied to participate in research, for both students and faculty.
- To develop the problem solving skills in the students to be ready to deal with cutting edge technologies of the industry.
- To make the students and faculty excel in their professional fields by inculcating the communication skills, leadership skills, team building skills with the organization of various co-curricular and extra-curricular programmes.
- To provide the students with theoretical and applied knowledge, and adopt an education approach that promotes lifelong learning and ethical growth.

Program Educational Objectives

PEO1 : To induce strong foundation in mathematical and core concepts, which enable them to participate in research, in the field of computer science.

PEO2 : To be able to become the part of application development and problem solving by learning the computer programming methods, of the industry and related domains.

PEO3 : To gain the multidisciplinary knowledge by understanding the scope of association of computer science engineering discipline with other engineering disciplines.

PEO4 : To improve the communication skills, soft skills, organizing skills which build the professional qualities, there by understanding the social responsibilities and ethical attitude.

PROGRAM OUTCOMES(POs)

Engineering Graduates will be able to:

1. **Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Applications of Computing: Ability to use knowledge in various domains to provide solution to new ideas and innovations.
PSO2: Programming Skills: Identify required data structures, design suitable algorithms, develop and maintain software for real world problems
PSO 3: Make use of computational and experimental tools for creating innovative career paths, to be an entrepreneur and desire for higher studies.

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

R24 B.TECH CSE SYLLABUS

CS506PC: UI DESIGN - FLUTTER

B.Tech. III Year I Sem. **L T P C**
0 0 2 1

Course Objectives:

- Learns to Implement Flutter Widgets and Layouts
- Understands Responsive UI Design and with Navigation in Flutter
- Knowledge on Widges and customize widgets for specific UI elements, Themes
- Understand to include animation apart from fetching data

Course Outcomes:

- Implements Flutter Widgets and Layouts
- Responsive UI Design and with Navigation in Flutter
- Create custom widgets for specific UI elements and also Apply styling using themes and custom styles.
- Design a form with various input fields, along with validation and error handling
- Fetches data and write code for unit Test for UI components and also animation

List of Experiments: Students need to implement the following experiments

1. a) Install Flutter and Dart SDK.
   b) Write a simple Dart program to understand the language basics.
2. a) Explore various Flutter widgets (Text, Image, Container, etc.).
   b) Implement different layout structures using Row, Column, and Stack widgets.
3. a) Design a responsive UI that adapts to different screen sizes.
   b) Implement media queries and breakpoints for responsiveness.
4. a) Set up navigation between different screens using Navigator.
   b) Implement navigation with named routes.
5. a) Learn about stateful and stateless widgets.
   b) Implement state management using set State and Provider.
6. a) Create custom widgets for specific UI elements.
   b) Apply styling using themes and custom styles.
7. a) Design a form with various input fields.
   b) Implement form validation and error handling.
8. a) Add animations to UI elements using Flutter's animation framework.
   b) Experiment with different types of animations (fade, slide, etc.).
9. a) Fetch data from a REST API.
   b) Display the fetched data in a meaningful way in the UI.

10. a) Write unit tests for UI components.
    b) Use Flutter's debugging tools to identify and fix issues.

TEXT BOOK:

11. 1. Marco L. Napoli, Beginning Flutter: A Hands-on Guide to App Development.

CO-PO-PSO Mapping Matrix:

CO\PO/PSO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1: Implements Flutter Widgets and Layouts	3	2	3	-	3	-			-				3	-	3
CO2: Responsive UI Design and Navigation in Flutter	3	2	3	-	3	2			-				3	-	3
CO3: Create custom widgets and apply styling	3	2	3	-	3	1			-				3	-	3
CO4: Design a form with validation and error handling	2	3	3	2	3	-			2				3	2	3
CO5: Fetches data, writes code for unit testing and animation	3	3	3	3	3	-			-				3	3	3

Justification for Mapping:

CO1: Implements Flutter Widgets and Layouts

- **PO1 (Engineering knowledge)**: Demonstrates understanding of Flutter layouts and widgets (**3**).
- **PO2 (Problem analysis)**: Requires analysis of UI layout problems (**2**).
- **PO3 (Design/development of solutions)**: Focuses on designing user interfaces using Flutter widgets (**3**).
- **PO5 (Modern tool usage)**: Uses modern mobile development tools like Flutter (**3**).
- **PSO1**: Aligns with understanding and designing web/mobile UI elements (**3**).
- **PSO3**: Promotes modern tools for innovative solutions (**3**).

CO2: Responsive UI Design and Navigation in Flutter

- **PO1**: Demonstrates knowledge of responsive UI and navigation (**3**).
- **PO2**: Analyzing different device sizes and user interaction (**2**).
- **PO3**: Emphasizes design of responsive and navigational components (**3**).
- **PO5**: Uses modern tools for building responsive UIs (**3**).
- **PO6 (Engineer and society)**: Focus on real-world usability and societal needs (**2**).
- **PSO1**: Related to web/mobile design and user experience design (**3**).
- **PSO3**: Focuses on modern tools and innovative paths (**3**).

CO3: Create Custom Widgets and Apply Styling

- **PO1**: Understanding principles of creating custom widgets and styling (**3**).
- **PO2**: Problem-solving for custom widget creation and reuse (**2**).

- **PO3**: Emphasizes on designing specific UI solutions (**3**).
- **PO5**: Leveraging Flutter's advanced features for styling (**3**).
- **PO6**: Applying custom styles based on societal or user preferences (**1**).
- **PSO1**: Involves custom solution designs in web/mobile contexts (**3**).
- **PSO3**: Encourages innovative approaches using modern tools (**3**).

CO4: Design a Form with Validation and Error Handling

- **PO1**: Understanding form elements and validation techniques (**2**).
- **PO2**: Focuses on handling errors and data validation problems (**3**).
- **PO3**: Development of solutions for form-based applications (**3**).
- **PO4 (Investigation of complex problems)**: Investigating and solving validation issues (**2**).
- **PO5**: Uses Flutter for form design and validation tools (**3**).
- **PO9 (Individual and teamwork)**: Collaborating for form creation and testing (**2**).
- **PSO1**: Related to programming and interface handling (**3**).
- **PSO2**: Ensuring reliability of forms and data integrity (**2**).
- **PSO3**: Uses modern tools to ensure form reliability and error handling (**3**).

CO5: Fetches Data, Writes Code for Unit Testing and Animation

- **PO1**: Understanding of data-fetching mechanisms and animations (**3**).
- **PO2**: Analyzing complex data-fetching issues and animation performance (**3**).
- **PO3**: Focuses on developing unit tests for ensuring UI correctness (**3**).
- **PO4**: Investigating the correctness of UI components and animations (**3**).
- **PO5**: Utilizes modern testing tools to ensure component accuracy (**3**).
- **PSO1**: Related to data management, programming, and testing for reliability (**3**).
- **PSO2**: Ensures software reliability, particularly with UI testing and error handling (**3**).
- **PSO3**: Modern tools are used for innovative career paths or entrepreneurship (**3**).

Mapping Levels:

- **3**: Highly related
- **2**: Moderately related
- **1**: Slightly related
- **-**: Not related

This matrix demonstrates how each CO aligns with specific POs and PSOs, ensuring comprehensive coverage of learning objectives and skill development.

Experiment 1:
1. a) Install Flutter and Dart SDK.

System Requirements

- **Operating System: Windows 10 or later (64-bit)**
- **Disk Space: 1.64 GB (does not include disk space for IDE/tools)**
- **Tools: Git for Windows 2.x, PowerShell 5.0, and a compatible IDE (VS Code, Android Studio, IntelliJ)**

Step 1: Get the Flutter SDK

1. **Download the Flutter SDK:**
    - **Visit the Flutter SDK releases page.**
    - **Download the latest stable release of the Flutter SDK (the .zip file).**
2. **Extract the Flutter SDK:**
    - **Extract the downloaded .zip file and place the contained flutter directory in a desired installation location (e.g., C:\src\flutter).**

Step 2: Update Your Path

1. **Locate Your System Path:**
    - **Open the Start Search, type in env, and select Edit the system environment variables.**
    - **In the System Properties window, click on the Environment Variables button.**
2. **Update Path:**
    - **In the User variables section, find the Path variable and click Edit.**
    - **Click New and add the full path to the flutter\bin directory (e.g., C:\src\flutter\bin).**
    - **Click OK to close all windows.**
3. **Verify Flutter is Added:**
    - **Open a new Command Prompt or PowerShell window and run flutter doctor.**

Step 3: Run flutter doctor

1. **Open Command Prompt or PowerShell:**
    - **Type cmd or powershell in the Start menu and open it.**
2. **Run flutter doctor:**
    - **In the terminal, type flutter doctor and press Enter.**
    - **This command checks your environment and displays a report of the status of your Flutter installation.**

Step 4: Install Android Studio

1. **Download Android Studio:**
    - **Visit the Android Studio download page and download the installer.**
2. **Install Android Studio:**
    - **Run the installer and follow the setup wizard to complete the installation.**
    - **During installation, make sure the boxes for the following are checked:**
        - **Android SDK**
        - **Android SDK Platform**
        - **Android Virtual Device**
3. **Setup Android Studio:**
    - **Open Android Studio.**
    - **Complete the Android Studio Setup Wizard, which includes downloading the Android SDK components.**

Step 5: Set Up the Android Emulator

1. **Open Android Studio:**
    - **Go to Tools > AVD Manager.**
2. **Create a Virtual Device:**
    - **Click on Create Virtual Device, select a hardware profile, and click Next.**
    - **Select a system image (e.g., x86 Images tab), download if necessary, and click Next.**
    - **Click Finish to create the AVD.**

Step 6: Set Up the IDE (Visual Studio Code)

1. **Download VS Code:**
    - **Visit the Visual Studio Code download page and download the installer.**
2. **Install VS Code:**
    - **Run the installer and follow the setup wizard to complete the installation.**
3. **Install Flutter and Dart Plugins:**
    - **Open VS Code.**
    - **Go to Extensions (Ctrl+Shift+X).**
    - **Search for and install the Flutter and Dart extensions.**

Step 7: Set Up the Flutter Device

1. **Enable Developer Mode on Your Device:**
    - **Go to Settings > About phone and tap the Build number 7 times to unlock developer options.**
    - **Go to Settings > System > Developer options and enable USB debugging.**
2. **Connect Your Device:**

- **Connect your Android device to your computer via USB.**
- **Run flutter devices in the terminal to verify that Flutter recognizes your connected device.**

Step 8: Create and Run a New Flutter Project

1. **Create a New Flutter Project:**
   - **In VS Code, open the command palette (Ctrl+Shift+P).**
   - **Type Flutter: New Project, then press Enter.**
   - **Select a project name and location to save the project.**
2. **Run the Flutter Project:**
   - **Open the main.dart file in your new project.**
   - **Press F5 to start debugging and run your app.**

That's it! You now have Flutter installed and set up on your Windows machine. You can start building Flutter applications. If you encounter any issues, the flutter doctor command can provide helpful diagnostics.

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

```
import 'package:flutter/material.dart';
void main() {
 runApp(Abc());
}
class Abc extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     home: Def(),
   );
 }
}
```

```
class Def extends StatelessWidget {
 const Def({super.key});
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: Text("Welcome"),
       backgroundColor: Colors.purple,
     ),
     body: Column(
       children: [
         //Widgets
       ],
     ),);}}
```

1. **What is Dart?**
2. **What is a Future in Dart?**
3. **What does the async keyword do in Dart?**
4. **How do you declare a list in Dart?**

5. **What is null safety in Dart?**
6. **What are the key features of the Dart programming language?**
7. **Explain the difference between var, final, and const in Dart.**
8. **How does Dart handle asynchronous programming?**
9. **What are the different types of collections in Dart?**
10. **What are the data types available in Dart?**
11. **How do you define a class in Dart?**
12. **What is the difference between final and const in Dart?**
13. **What is a constructor in Dart?**
14. **How do you handle exceptions in Dart?**
15. **What is Flutter and what are its main components?**
16. **What is a widget in Flutter?**
17. **Explain the difference between StatelessWidget and StatefulWidget.**
18. **How does the Flutter rendering process work?**
19. **What is the build method in Flutter?**
20. **How do you navigate between screens in Flutter?**
21. **What is a Container widget in Flutter?**
22. **What is setState in Flutter?**
23. **What is a Stream in Flutter?**
24. **How do you add external packages to a Flutter project?**
25. **What is hot reload in Flutter?**
26. **How do you handle user input in Flutter?**
27. **What is the MaterialApp widget?**
28. **What is a Scaffold widget?**
29. **What is the purpose of the pubspec.yaml file in a Flutter project?**
30. **What is MediaQuery used for in Flutter, and how can you use it to make a widget responsive?**

Experiment 2:

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

Text Widget:

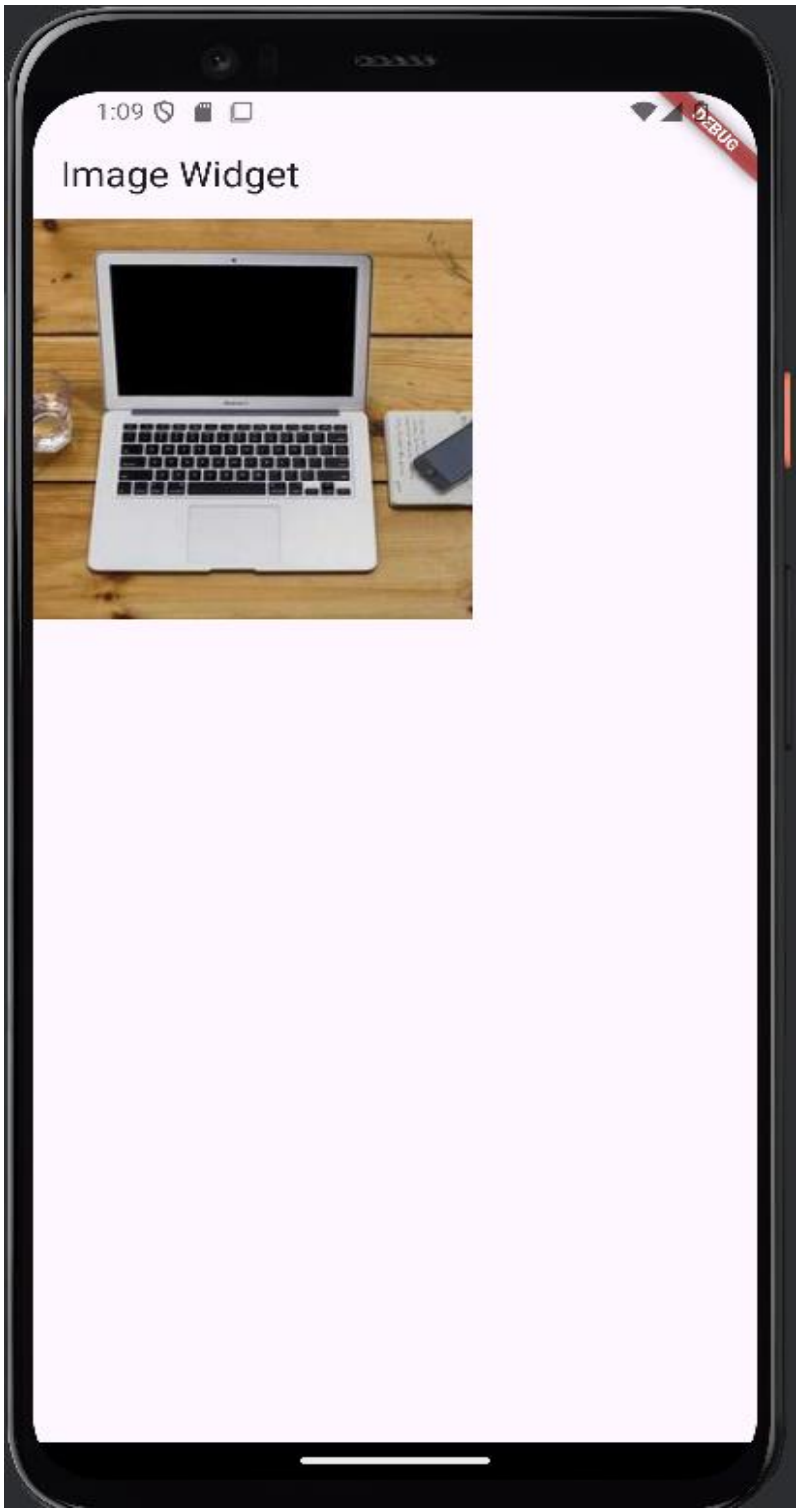
```
import 'package:flutter/material.dart';
void main() {
 runApp(MyApp());
}
class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     home: Abc();
   );}}
class Abc extends StatelessWidget {
 const Abc({super.key});
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(title: Text('Text Widget Example')),
     body: Center(
       child: Text(
         'Hello, Flutter!',
         style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
       ),
     ),
   ),}}
```

Image Widgets

- **Network Image**

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


void main() {
 runApp(MyApp());
}
class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     home: Abc(),
   );
 }
}
class Abc extends StatelessWidget {
 const Abc({super.key});
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: Text("Image Widget"),
    ),
     body: Image.network('https://picsum.photos/250?image=9'),


   );}}
```

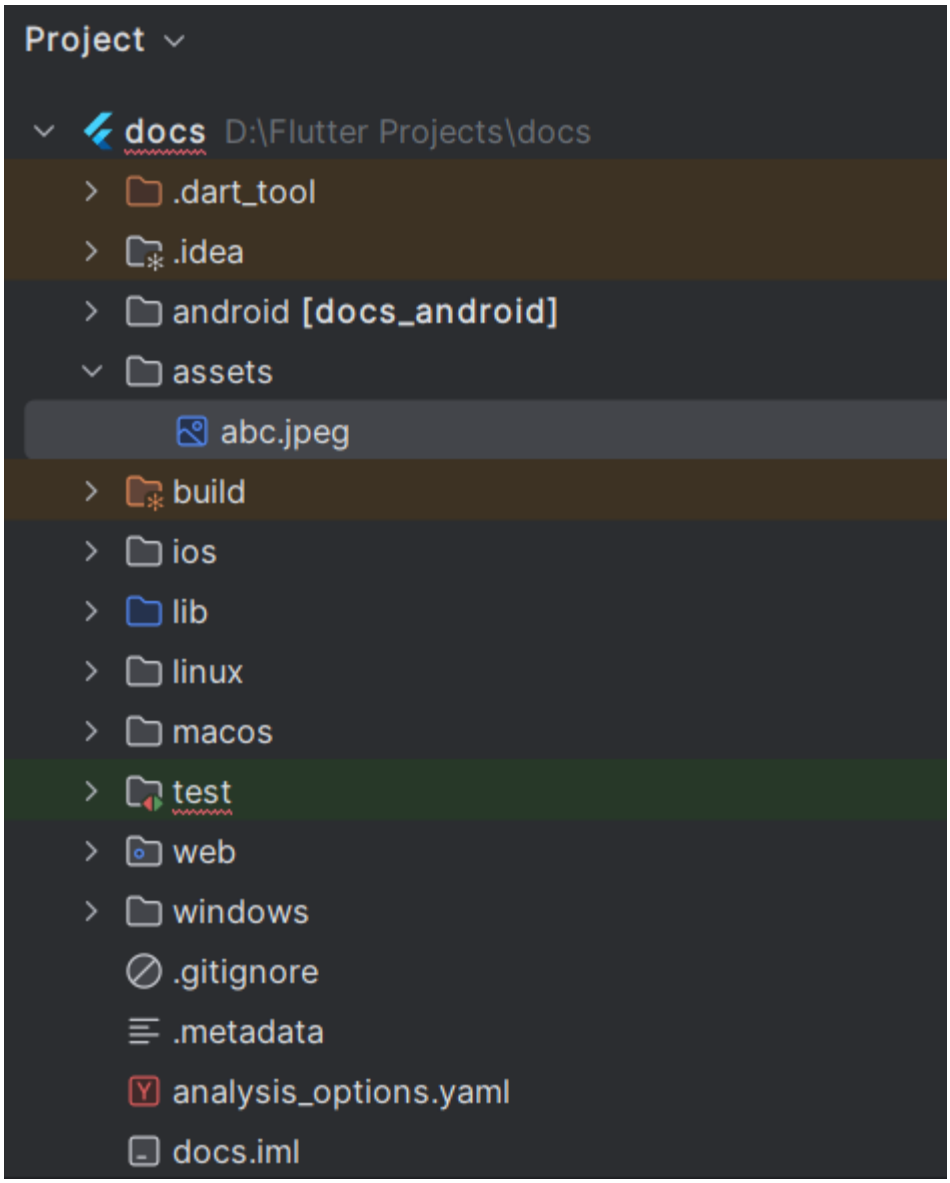
Note: Must Add the Internet Tag in AndoridManifiest.yaml which is inside Android Folder for giving the Internet access to the App.



- **Asset Image or Local image**

- **Add Images to Your Project**

1. **Create an assets directory: In your Flutter project root, create a directory named assets (or any other name you prefer). And copy the image of your requirement inside the assets folder.**



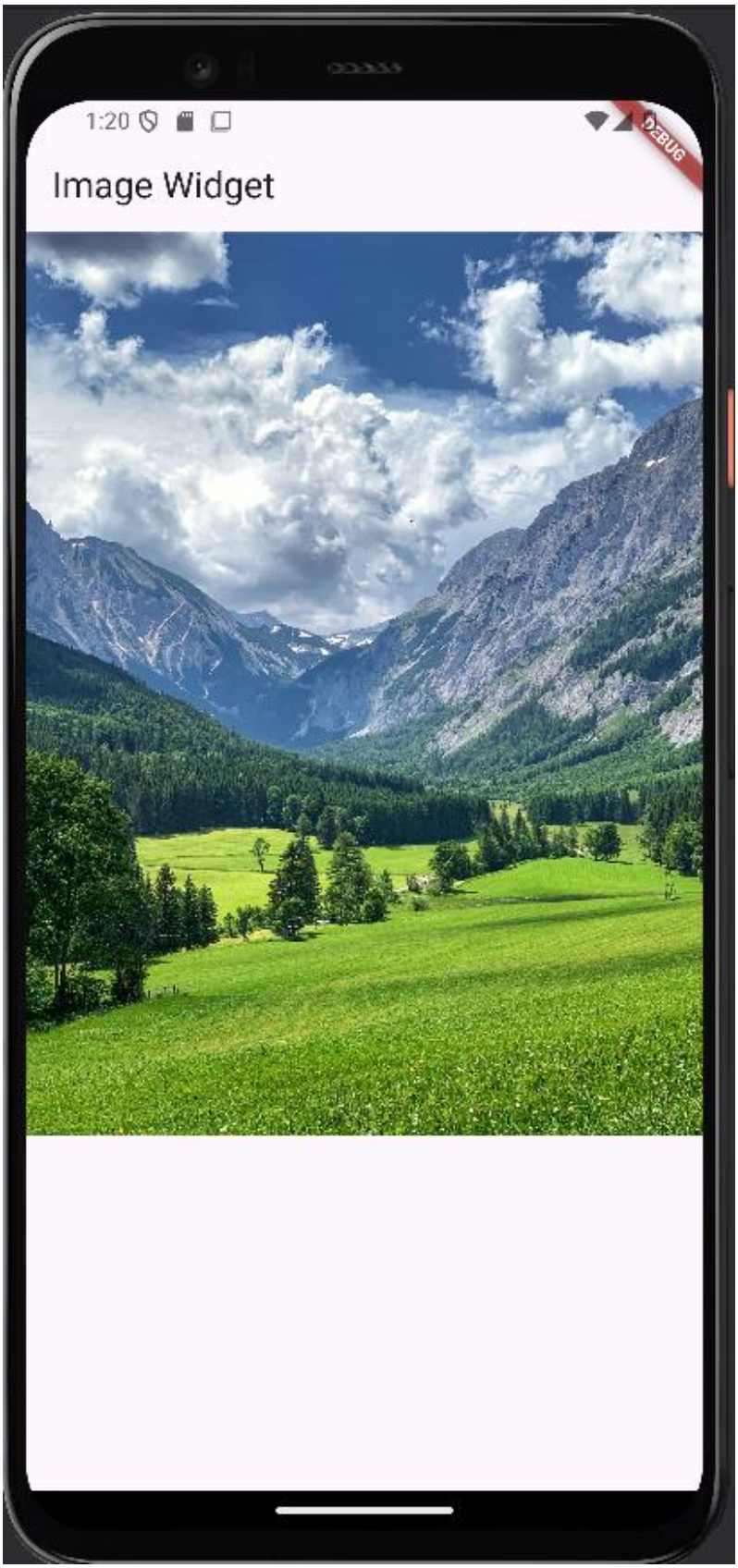
2. **Declare the assets: Open your pubspec.yaml file and add the assets under the flutter section.**

```
assets:
 - assets/abc.jpeg
```

3. **Open Terminal: give the command 'flutter pub get'.**
4. **To Display Images in Your Flutter App, we will be using 'Image.asset' widget.**

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

void main() {

 runApp(MyApp());

}
```

```
class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     home: Abc(),
);}}
class Abc extends StatelessWidget {
 const Abc({super.key});
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: Text("Image Widget"),
     ),
     body: Image.asset('assets/abc.jpeg'),
   );}}
```

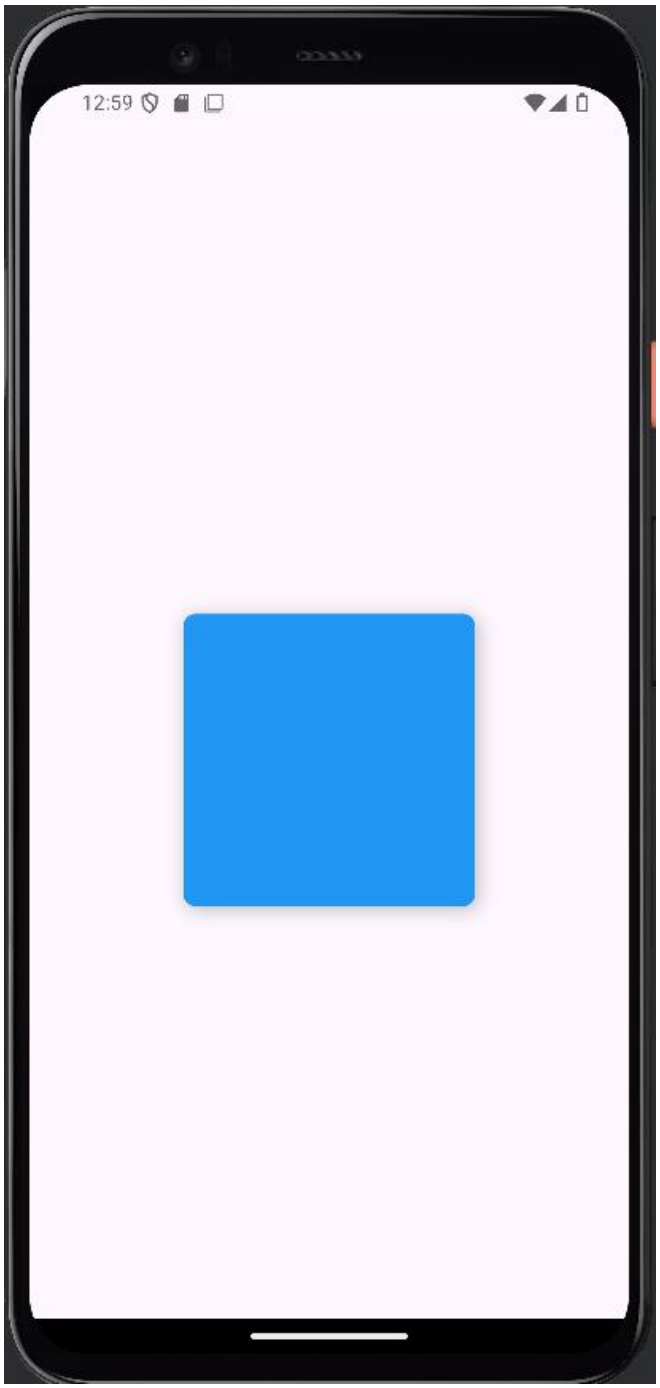


Container Widget

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

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

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     home: Abc(),
   );
 }
}



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

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(title: Text('Container Widget Example')),
     body: Center(
       child: Container(
         width: 200,
         height: 200,
         padding: EdgeInsets.all(16),
         margin: EdgeInsets.all(16),
         decoration: BoxDecoration(
           color: Colors.blue,
           borderRadius: BorderRadius.circular(8),
           boxShadow: [
             BoxShadow(
               color: Colors.black26,
```

```
              blurRadius: 10,
              offset: Offset(2, 2),
            ),
          ],
        ),
        child: Center(
          child: Text(
            'Container',
            style: TextStyle(color: Colors.white, fontSize: 24),
          ),
        ),
      ),
    ),
  );
 }
}
```

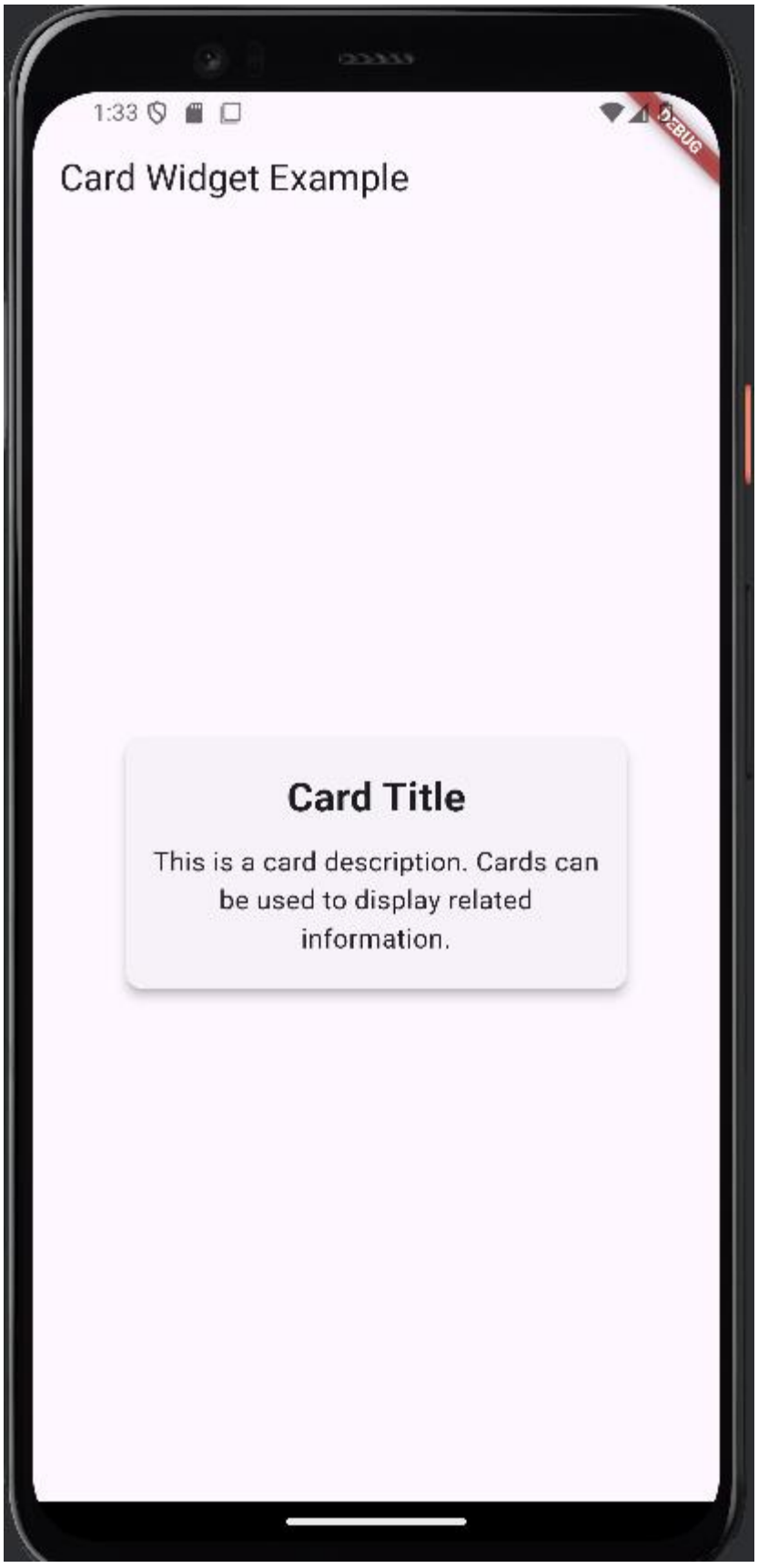


Card Widget

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

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

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     home: Abc();
   );
 }
}

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

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(title: Text('Card Widget Example')),
     body: Center(
       child: Card(
         elevation: 5, // The shadow effect of the card
         shape: RoundedRectangleBorder(
           borderRadius: BorderRadius.circular(10), // Rounded corners
         ),
         child: Container(
           width: 300,
           height: 150,
           padding: EdgeInsets.all(16),
           child: Column(
             mainAxisAlignment: MainAxisAlignment.center,
```

```
              children: [
                Text(
                  'Card Title',
                  style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
                ),
                SizedBox(height: 10),
                Text(
                  'This is a card description. Cards can be used to display related
information.',
                  textAlign: TextAlign.center,
                  style: TextStyle(fontSize: 16),
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
}
```

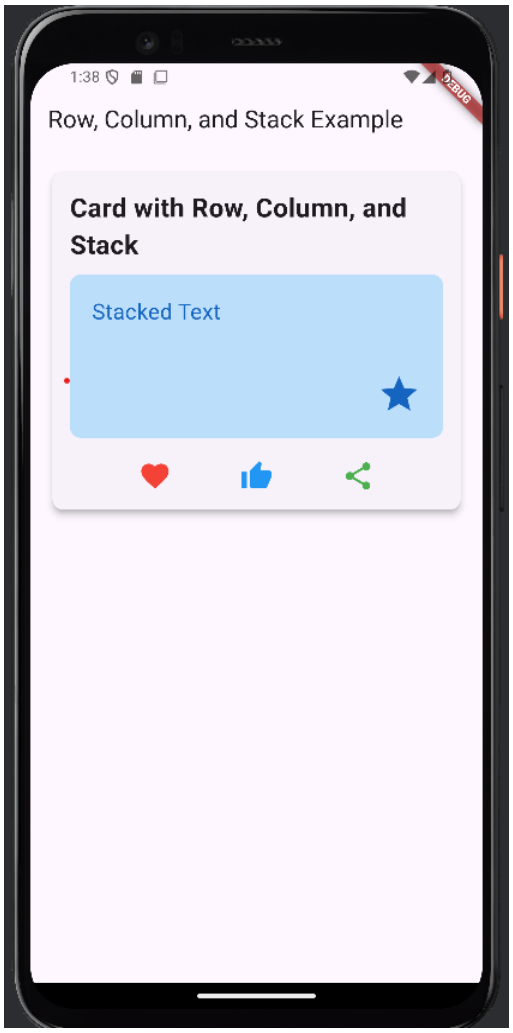


2 b) Implement different layout structures using Row, Column, and Stack widgets.

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

```
void main() {
 runApp(MyApp());
}
class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     home: Scaffold(
       appBar: AppBar(title: Text('Row, Column, and Stack Example')),
       body: Padding(
         padding: const EdgeInsets.all(16.0),
         child: Column(
           children: [
             Card(
               elevation: 5,
               shape: RoundedRectangleBorder(
                 borderRadius: BorderRadius.circular(10),
               ),
               child: Container(
                 width: double.infinity,
                 padding: EdgeInsets.all(16),
                 child: Column(
                   crossAxisAlignment: CrossAxisAlignment.start,
                   children: [
                     Text(
                       'Card with Row, Column, and Stack',
                       style: TextStyle(fontSize: 24, fontWeight:
FontWeight.bold),
                     ),
                     SizedBox(height: 10),
                     Stack(
                       children: [
                         Container(
                           width: double.infinity,
```

```
                height: 150,
                decoration: BoxDecoration(
                  color: Colors.blue[100],
                  borderRadius: BorderRadius.circular(10),
                ),
              ),
              Positioned(
                top: 20,
                left: 20,
                child: Text(
                  'Stacked Text',
                  style: TextStyle(fontSize: 20, color:
Colors.blue[800]),),),
              Positioned(
                bottom: 20,
                right: 20,
                child: Icon(Icons.star, size: 40, color:
Colors.blue[800]),
              ),
            ],
          ),
          SizedBox(height: 20),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: [
              Icon(Icons.favorite, size: 30, color: Colors.red),
              Icon(Icons.thumb_up, size: 30, color: Colors.blue),
              Icon(Icons.share, size: 30, color: Colors.green),
            ],),],),),],),),),);}}
```


1:38
DEBUG
Row, Column, and Stack Example
Card with Row, Column, and Stack
Stacked Text

1. **How can Flexible and Expanded widgets help in creating responsive layouts?**
2. **What does the AspectRatio widget do in Flutter?**
3. **How can MediaQuery be used to adjust font sizes in a responsive design?**
4. **What is the difference between MediaQuery and LayoutBuilder in handling responsive layouts?**
5. **How does the Row widget arrange its children, and when would you use it?**
6. **What is the main difference between Row and Column widgets in Flutter?**
7. **How can the Expanded widget be used within a Column to make child widgets take up available space?**
8. **What does the Stack widget do, and when would you use it in a layout?**
9. **How can you control the positioning of children within a Stack?**
10. **What is the purpose of the Align widget when used inside a Stack?**
11. **How do Row and Column widgets handle overflow when their content exceeds available space?**
12. **How does the Navigator widget manage navigation between screens in Flutter?**
13. **What is the difference between named routes and unnamed (anonymous) routes in Flutter?**
14. **How do you define and use a named route in a Flutter application?**
15. **How can you pass data between routes using unnamed routes?**
16. **What is the role of the RouteSettings class in Flutter's navigation system?**
17. **What is the purpose of the setState method in Flutter?**
18. **How does the setState method impact the widget tree?**
19. **What is the Provider package used for in Flutter?**
20. **How do you define a ChangeNotifier in Flutter using the Provider package?**
21. **How do you provide a ChangeNotifier to the widget tree using Provider?**
22. **How do you access the provided data in a widget using Provider?**
23. **How do you create a basic custom widget in Flutter?**
24. **Why would you create custom widgets in a Flutter application?**
25. **How can you pass data to a custom widget in Flutter?**
26. **How do you handle user interactions in a custom widget?**
27. **Can you create custom widgets with state in Flutter? If so, how?**
28. **What are the benefits of using custom widgets for specific UI elements?**
29. **What are the advantages of using custom widgets in Flutter for building complex UIs?**
30. **How does the null-aware operator (??) work in Dart?**

Experiment 3:

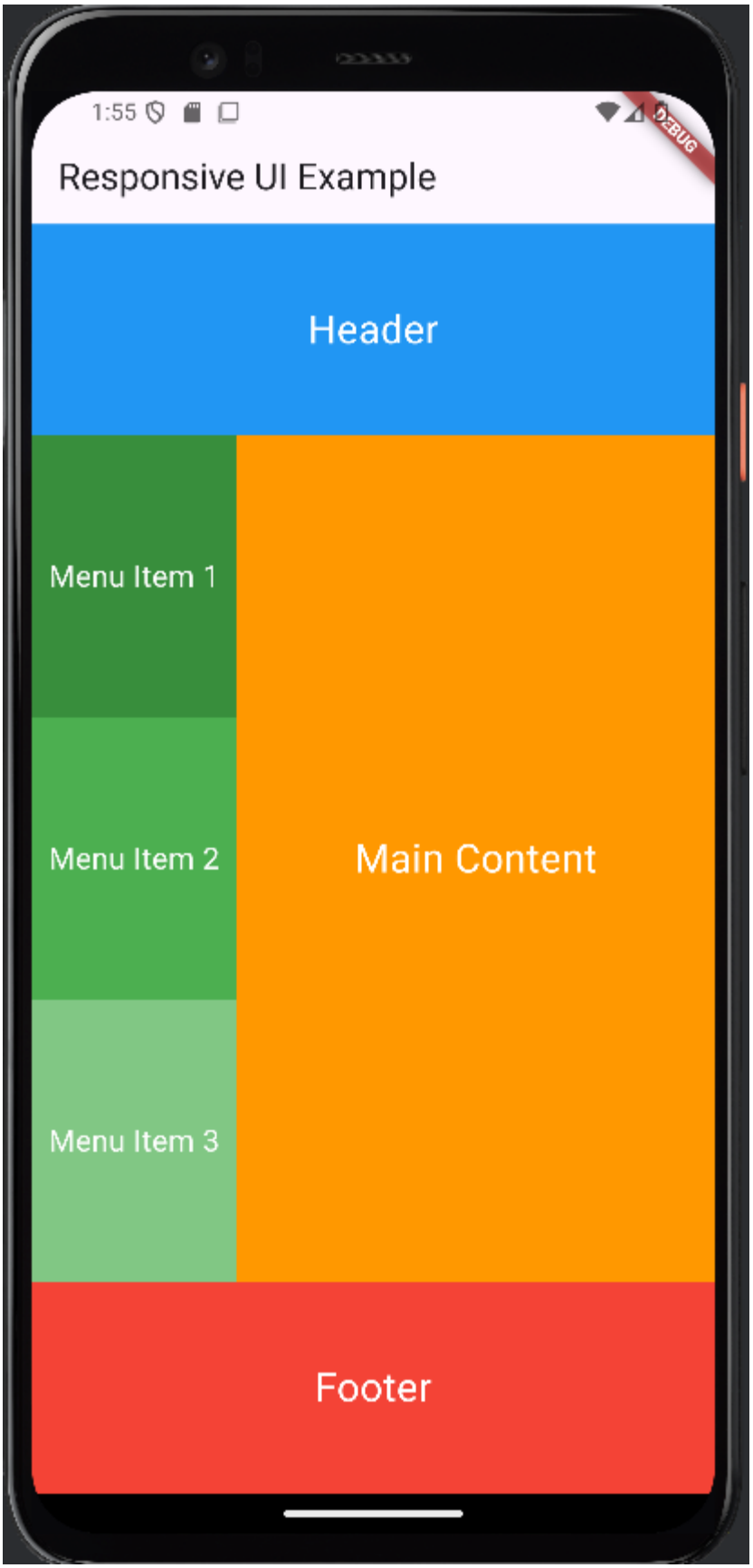
3. a) Design a responsive UI that adapts to different screen sizes.

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


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


class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     home: ResponsiveLayout(),
   );
 }
}


class ResponsiveLayout extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(title: Text('Responsive UI Example')),
     body: Column(
       children: [
         Expanded(
           flex: 2,
           child: Container(
             color: Colors.blue,
             child: Center(
               child: Text(
                 'Header',
                 style: TextStyle(color: Colors.white, fontSize: 24),
               ),
```

```
    ),
  ),
),
Expanded(
  flex: 8,
  child: Row(
    children: [
      Expanded(
        flex: 3,
        child: Container(
          color: Colors.green,
          child: Column(
            children: [
              Expanded(
                flex: 1,
                child: Container(
                  color: Colors.green[700],
                  child: Center(
                    child: Text(
                      'Menu Item 1',
                      style: TextStyle(color: Colors.white, fontSize: 18),
                    ),
                  ),
                ),
              ),
              Expanded(
                flex: 1,
                child: Container(
                  color: Colors.green[500],
                  child: Center(
                    child: Text(
                      'Menu Item 2',
                      style: TextStyle(color:Colors.white,fontSize: 18),
```

```
                ),
              ),
            ),
          ),
          Expanded(
            flex: 1,
            child: Container(
              color: Colors.green[300],
              child: Center(
                child: Text(
                  'Menu Item 3',
                  style: TextStyle(color: Colors.white, fontSize: 18),
                ),
              ),
            ),
          ),
        ],
      ),
    ),
  ),
  Expanded(
    flex: 7,
    child: Container(
      color: Colors.orange,
      child: Center(
        child: Text(
          'Main Content',
          style: TextStyle(color: Colors.white, fontSize: 24),
        ),
      ),
    ),
  ),
],
```

```
  ),
),
Expanded(
  flex: 2,
  child: Container(
    color: Colors.red,
    child: Center(
      child: Text(
        'Footer',
        style: TextStyle(color: Colors.white, fontSize: 24),
      ),
    ),
  ),),],),);}}
```

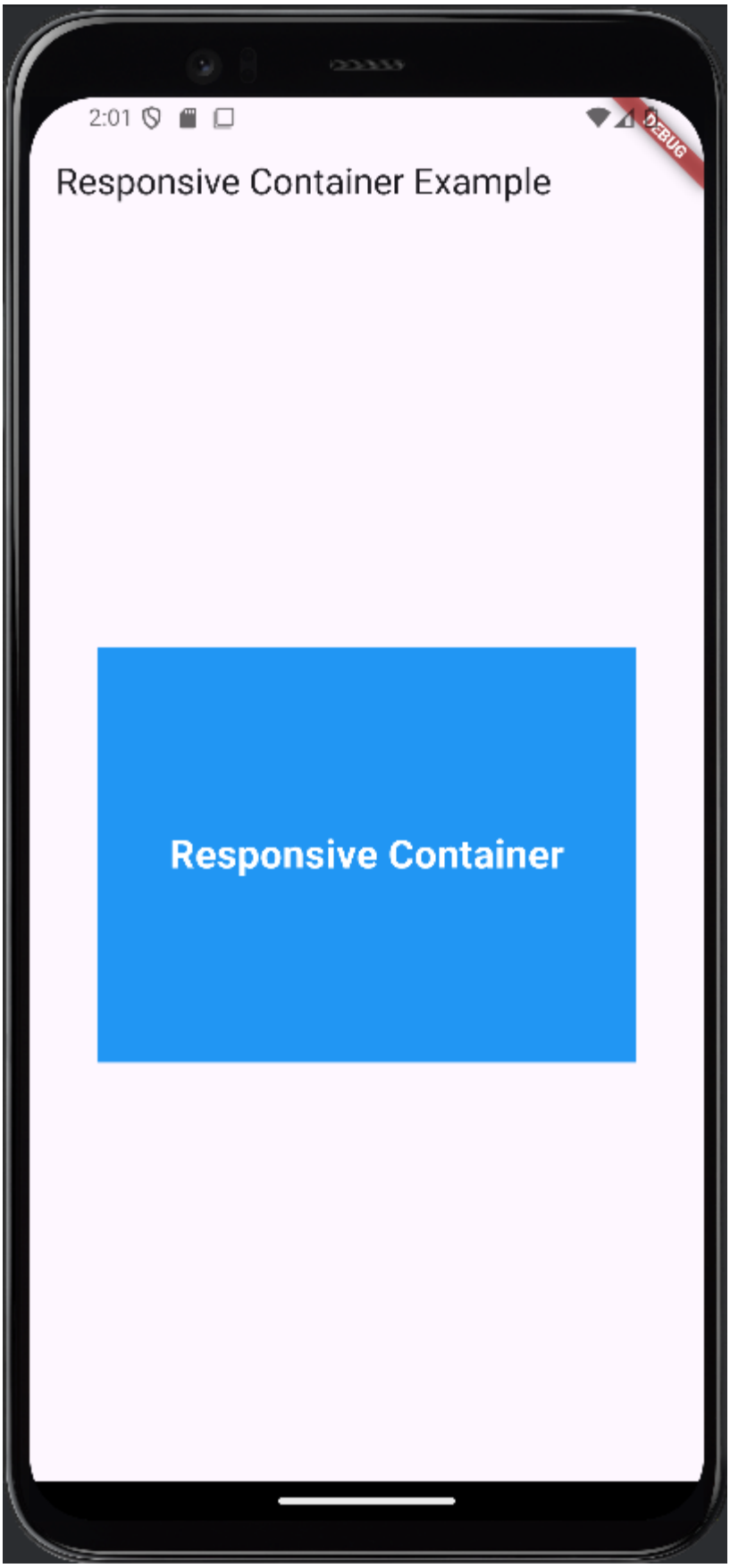


3 b) Implement media queries and breakpoints for responsiveness.

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

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

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     home: ResponsiveContainer(),
   );
 }
}

class ResponsiveContainer extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   // Get the screen size using MediaQuery
   final screenSize = MediaQuery.of(context).size;
   final screenWidth = screenSize.width;
   final screenHeight = screenSize.height;

   // Determine container size based on screen width
   double containerWidth;
   double containerHeight;

   if (screenWidth < 600) {
     // Small screen size (e.g., mobile phones)
     containerWidth = screenWidth * 0.8; // 80% of screen width
     containerHeight = screenHeight * 0.3; // 30% of screen height
   } else {
     // Larger screen size (e.g., tablets or desktops)
     containerWidth = screenWidth * 0.5; // 50% of screen width
```

```
  containerHeight = screenHeight * 0.4; // 40% of screen height
}

return Scaffold(
  appBar: AppBar(title: Text('Responsive Container Example')),
  body: Center(
    child: Container(
      width: containerWidth,
      height: containerHeight,
      color: Colors.blue,
      child: Center(
        child: Text(
          'Responsive Container',
          style: TextStyle(
            color: Colors.white,
            fontSize: 24,
            fontWeight: FontWeight.bold,
          ),),),),),);}}
```



1. **How does the null-aware assignment operator (??=) work in Dart?**
2. **What does the conditional member access operator (?.) do in Dart?**

3. **How is the cascade operator (..) used in Dart?**
4. **What is the purpose of the spread operator (...) in Dart?**
5. **How does the null-aware spread operator (...?) differ from the spread operator in Dart?**
6. **What is the function of the type test operator (is) in Dart?**
7. **What does the type cast operator (as) do in Dart?**
8. **How can the ?. operator be useful in Dart when dealing with potentially null values?**
9. **What does the ! operator do in Dart when used after a variable?**
10. **How do you create an empty String list in Dart?**
11. **How do you use a generic method in Dart?**
12. **List the built-in types in Dart.**
13. **How do you import the Dart math library?**
14. **What is null safety in Dart?**
15. **What is the build method in Flutter?**
16. **How do you handle exceptions in Dart?**
17. **What are the benefits of using the Provider for state management?**
18. **How can you implement form validation in Flutter?**
19. **What are the advantages of responsive design in Flutter applications?**
20. **How can you optimize Flutter applications for performance?**
21. **What is the role of hot restart in Flutter development?**
22. **How does Flutter's widget tree architecture contribute to efficient UI updates?**
23. **What are the best practices for managing state across different screens in Flutter?**
24. **How does the Navigator widget manage navigation between screens in Flutter?**
25. **How do you define and use a named route in a Flutter application?**
26. **How can you pass data between routes using unnamed routes?**
27. **How does the setState method impact the widget tree?**
28. **What is the Provider package used for in Flutter?**

29. What does the Stack widget do, and when would you use it in a layout?

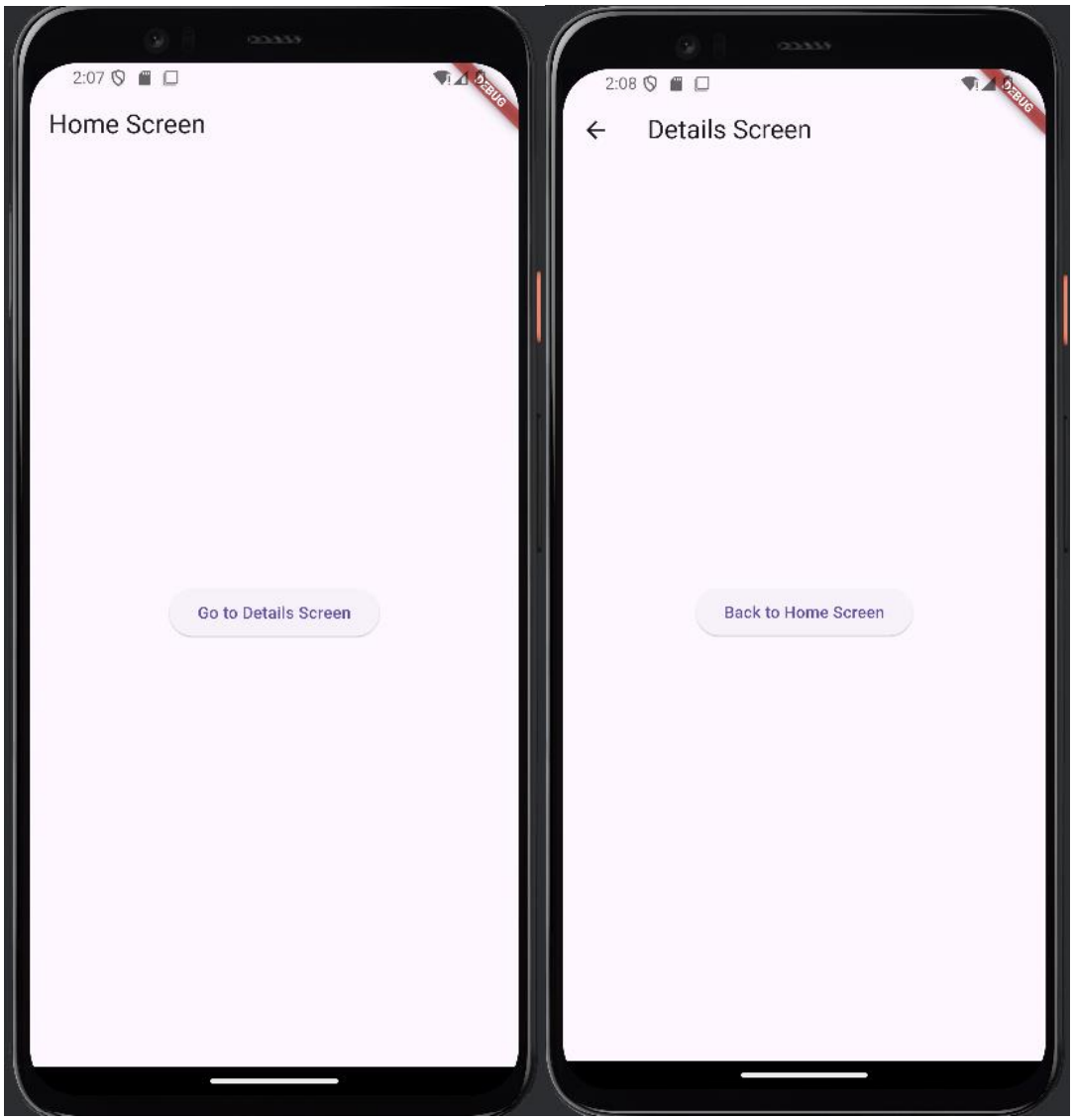
30. What is the purpose of the Align widget when used inside a Stack?

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 Example',
     theme: ThemeData(
       primarySwatch: Colors.blue,
     ),
     home: HomeScreen(),
   );
 }
}


class HomeScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: Text('Home Screen'),
     ),
     body: Center(
       child: ElevatedButton(
         onPressed: () {
           Navigator.push(
             context,
```

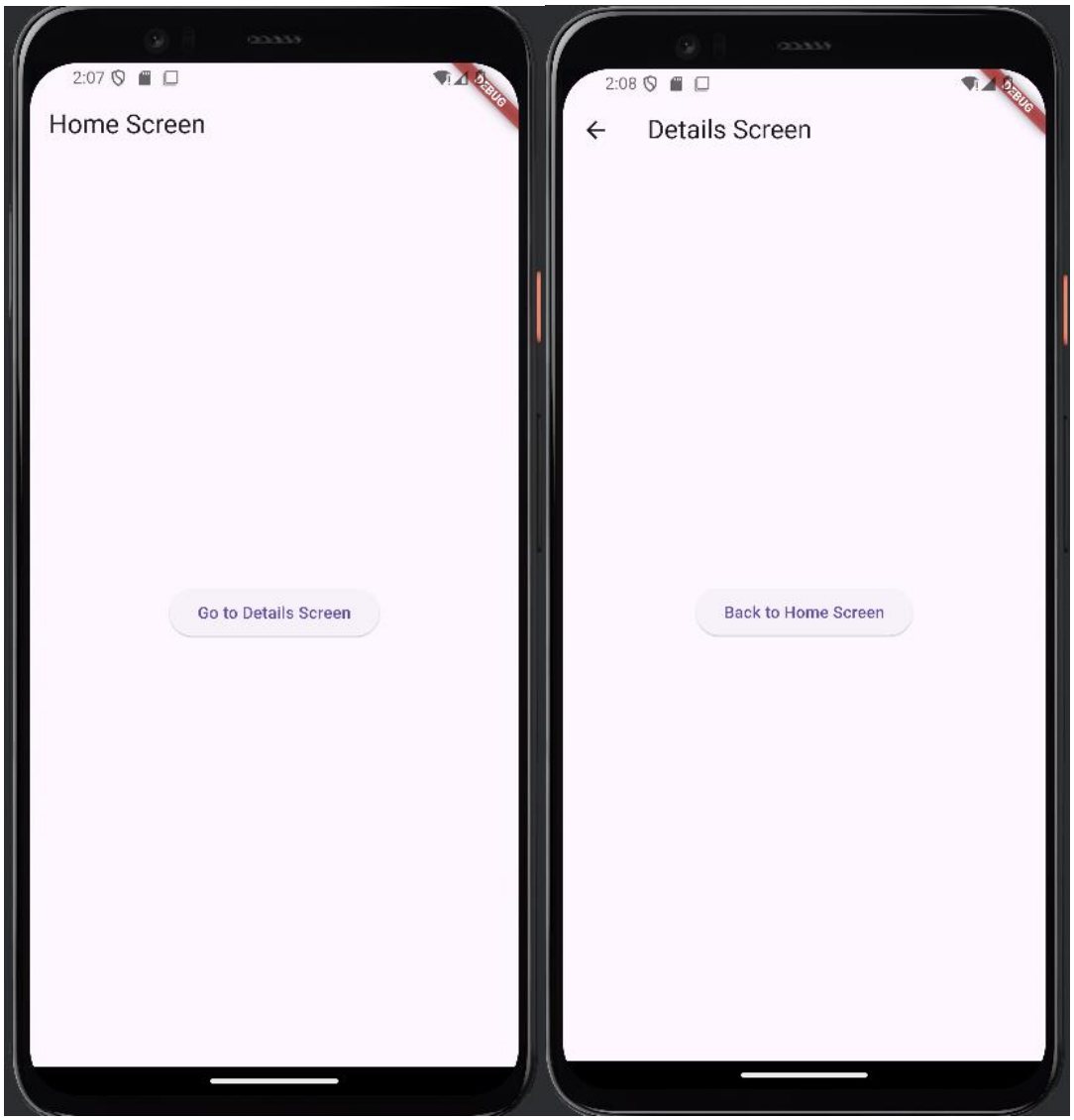
```
            MaterialPageRoute(builder: (context) => DetailsScreen()),
          );
        },
        child: Text('Go to Details Screen'),
      ),
    ),
  );
 }
}

class DetailsScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: Text('Details Screen'),
     ),
     body: Center(
       child: ElevatedButton(
         onPressed: () {
           Navigator.pop(context);
         },
         child: Text('Back to Home Screen'),
       ),),);}}
```


2:07
Home Screen
DEBUG
Go to Details Screen
2:08
Details Screen
DEBUG
Back to Home Screen

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 Example',
     theme: ThemeData(
       primarySwatch: Colors.blue,
     ),
     initialRoute: '/',
     routes: {
       '/': (context) => HomeScreen(),
       '/details': (context) => DetailsScreen(),
     },
   );
 }
}


class HomeScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: Text('Home Screen'),
     ),
     body: Center(
       child: ElevatedButton(
```

```
          onPressed: () {
            Navigator.pushNamed(context, '/details');
          },
          child: Text('Go to Details Screen'),
        ),
      ),
    );
  }
}

class DetailsScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Details Screen'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: Text('Back to Home Screen'),),),);}}
```


2:07
Home Screen
DEBUG
Go to Details Screen
2:08
Details Screen
DEBUG
Back to Home Screen

1. **How do Row and Column widgets handle overflow when their content exceeds available space?**
2. **How can you control the positioning of children within a Stack?**
3. **What are Records in Dart?**
4. **How do you declare a list in Dart?**
5. **How can MediaQuery be used to adjust font sizes in a responsive design?**
6. **How do you provide a ChangeNotifier to the widget tree using Provider?**
7. **What is a Container widget in Flutter?**
8. **How do you navigate between screens in Flutter?**
9. **How does the Row widget arrange its children, and when would you use it?**
10. **How can Flexible and Expanded widgets help in creating responsive layouts?**
11. **What is the main difference between Row and Column widgets in Flutter?**
12. **How does the null-aware operator (??) work in Dart?**
13. **How do you create a basic custom widget in Flutter?**
14. **How can you pass data to a custom widget in Flutter?**
15. **How do you handle user interactions in a custom widget?**
16. **Can you create custom widgets with state in Flutter? If so, how?**
17. **What are the benefits of using custom widgets for specific UI elements?**
18. **How can the Expanded widget be used within a Column to make child widgets take up available space?**

19. **What does the Stack widget do, and when would you use it in a layout?**

20. **How do you create an empty String list in Dart?**

21. **What are the key features of the Dart programming language?**

22. **What is the purpose of the pubspec.yaml file in a Flutter project?**

23. **What is hot reload in Flutter?**

24. **What is the difference between named routes and unnamed routes in Flutter?**

25. **How do you access the provided data in a widget using Provider?**

26. **How do you define a ChangeNotifier in Flutter using the Provider package?**

27. **How do you add external packages to a Flutter project?**

28. **What is the main difference between Row and Column widgets in Flutter?**

29. **What is the Provider package used for in Flutter?**

30. **What is setState in Flutter?**

Experiment 5:

5. a) Learn about stateful and stateless widgets.

Stateless Widgets

StatelessWidget is used when the part of the UI you are building does not change. Stateless widgets are immutable, meaning their properties cannot change once they are created. They are ideal for static content that doesn't depend on any dynamic data.

StatelessWidget is used when the part of the UI you are building does not change. Stateless widgets are immutable, meaning their properties cannot change once they are created. They are ideal for static content that doesn't depend on any dynamic data.

```
class Abc extends StatelessWidget {

 const Abc({super.key});


 @override

 Widget build(BuildContext context) {

   return const Placeholder();

 }

}
```

Stateful Widgets

StatefulWidget is used when the part of the UI you are building can change dynamically. Stateful widgets are mutable and can maintain state that changes over time. This is useful for interactive elements where the appearance or behavior of the widget depends on user input or other factors.

A StatefulWidget consists of two classes:

1. **The StatefulWidget class: This is immutable and creates an instance of the State class.**
2. **The State class: This is where the mutable state is maintained and updated.**

```
class Abc extends StatefulWidget {

 const Abc({super.key});


 @override

 State<Abc> createState() => _AbcState();
```

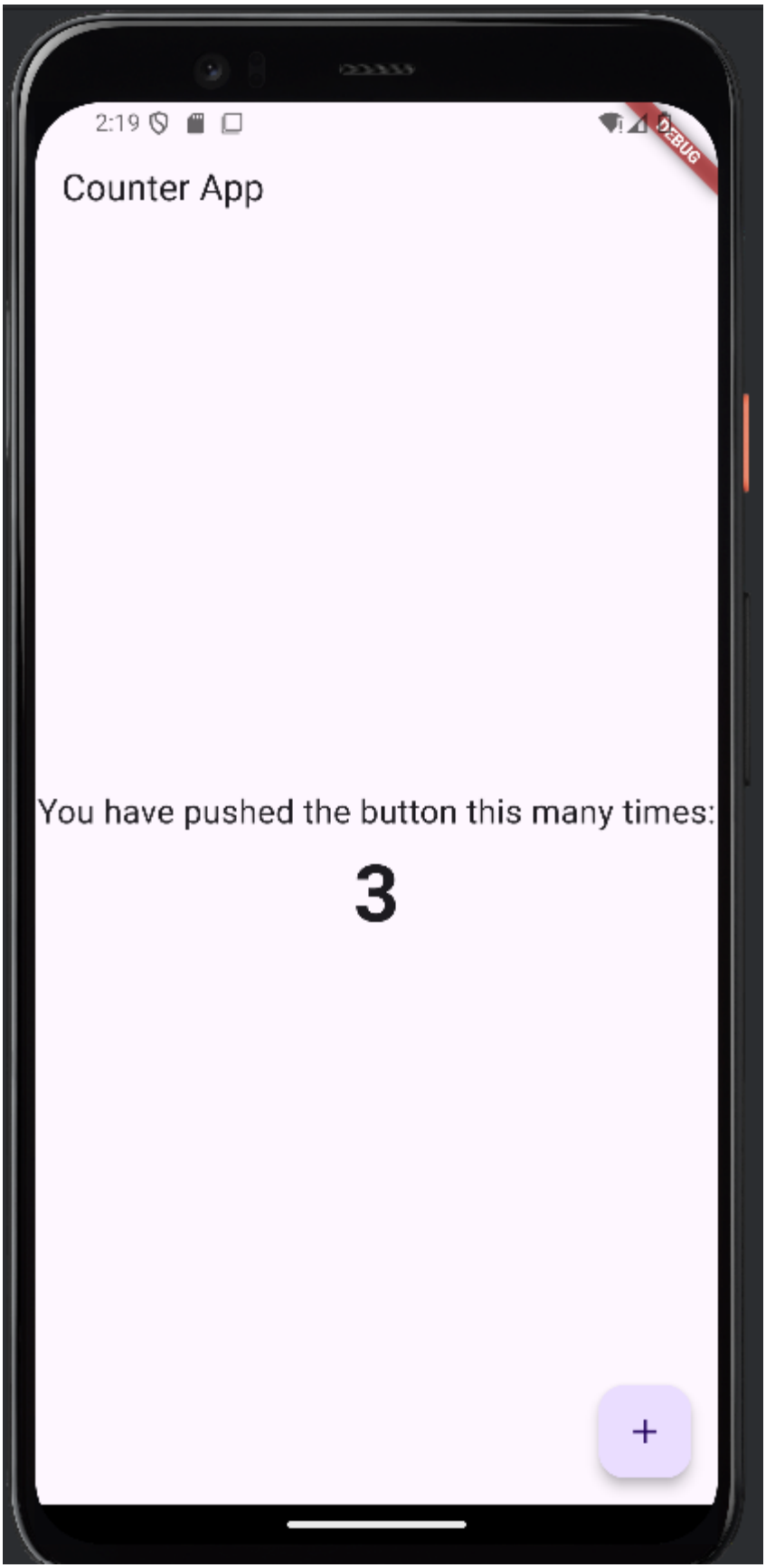
```
}

class _AbcState extends State<Abc> {
 @override
 Widget build(BuildContext context) {
   return const Placeholder();
 }
}
```

For better understanding of StatefulWidget, take a look at this below example

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

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

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'StatefulWidget Example',
     theme: ThemeData(
       primarySwatch: Colors.blue,
     ),
     home: CounterScreen(),
   );
 }
}

class CounterScreen extends StatefulWidget {
 @override
 _CounterScreenState createState() => _CounterScreenState();
```

```
}

class _CounterScreenState extends State<CounterScreen> {
 int _counter = 0; // State variable to keep track of the counter value

 void _incrementCounter() {
   setState(() {
     _counter++; // Update the state
   });
 }

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: Text('Counter App'),
     ),
     body: Center(
       child: Column(
         mainAxisAlignment: MainAxisAlignment.center,
         children: <Widget>[
           Text(
             'You have pushed the button this many times:',
             style: TextStyle(fontSize: 20),
           ),
           Text(
             '$_counter', // Display the current counter value
             style: TextStyle(fontSize: 48, fontWeight: FontWeight.bold),
           ),
         ],
       ),
     ),
     floatingActionButton: FloatingActionButton(
```

```
        onPressed: _incrementCounter, // Increment counter when button is pressed
        tooltip: 'Increment',
        child: Icon(Icons.add),
      ),
    );
  }
}
```



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

State Management Techniques:

- **Local State Management: Managing state within a single widget or a small part of the widget tree.**

```
class CounterScreen extends StatefulWidget {
 @override
 _CounterScreenState createState() => _CounterScreenState();
}
class _CounterScreenState extends State<CounterScreen> {
 int _counter = 0;
 void _incrementCounter() {
   setState(() {
     _counter++;
   });
 }
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(title: Text('Counter')),
     body: Center(
       child: Column(
         mainAxisAlignment: MainAxisAlignment.center,
         children: <Widget>[
           Text('Count: $_counter'),
           ElevatedButton(onPressed: _incrementCounter, child:
Text('Increment')),],),),);}}
```

- **Global State Management: Managing state that needs to be accessed and updated across multiple widgets or the entire application.**

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

// Define the CounterModel with ChangeNotifier
```

```
class CounterModel with ChangeNotifier {
 int _counter = 0;

 int get counter => _counter;

 void increment() {
   _counter++;
   notifyListeners(); // Notify listeners to rebuild
 }
}

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

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Provider Example',
     home: CounterScreen(),
   );
 }
}
```

```
class CounterScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(title: Text('Provider Example')),
     body: Center(
       child: Column(
         mainAxisAlignment: MainAxisAlignment.center,
         children: <Widget>[
           // Use Consumer to listen to changes in CounterModel
           Consumer<CounterModel>(
             builder: (context, counterModel, child) {
               return Text(
                 'Count: ${counterModel.counter}',
                 style: TextStyle(fontSize: 48, fontWeight: FontWeight.bold),
               );
             },
           ),
           SizedBox(height: 20),
           ElevatedButton(
             onPressed: () {
               // Update the state using Provider
               Provider.of<CounterModel>(context, listen: false).increment();
             },
             child: Text('Increment'),
           ),
         ],
       ),
     ),
   );
```

```
 }

}
```

To add the provider package to your Flutter project, follow these steps:

1. Add provider to pubspec.yaml

1. **Open your Flutter project in your preferred IDE or text editor.**
2. **Locate the pubspec.yaml file in the root directory of your project.**

Under the dependencies section, add provider along with the version number. As of the last update, the latest version is 6.0.0, but you should check pub.dev for the latest version.
yaml
Copy code

```
Dependencies:
  flutter:

   sdk: flutter



 # The following adds the Cupertino Icons font to your application.

 # Use with the CupertinoIcons class for iOS style icons.

 cupertino_icons: ^1.0.6

 provider: ^6.0.0  #Add This
```

3. **Save the pubspec.yaml file.**

2. Install the Package

To install the new package, run the following command in the terminal

flutter pub get

This command will fetch the provider package and add it to your project.

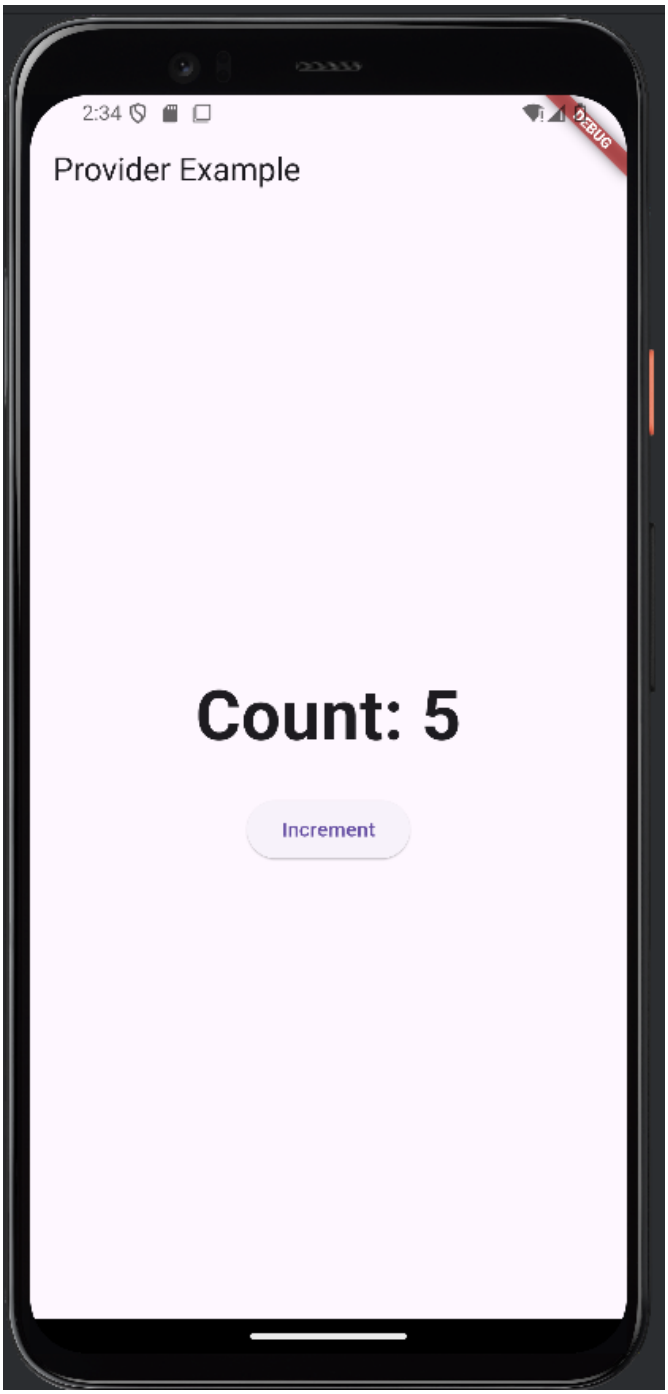
3. Import the Provider Package

In your Dart files where you need to use Provider, import it at the top of the file:

dart

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

Output for State Management using Provider:



1. **How do you access the provided data in a widget using Provider?**
2. **How do you create a basic custom widget in Flutter?**
3. **Why would you create custom widgets in a Flutter application?**
4. **How can you pass data to a custom widget in Flutter?**
5. **How do you handle user interactions in a custom widget?**
6. **Can you create custom widgets with state in Flutter? If so, how?**
7. **What are the benefits of using custom widgets for specific UI elements?**
8. **What are the advantages of using custom widgets in Flutter for building complex UIs?**
9. **How does the null-aware operator (??) work in Dart?**
10. **How does the null-aware assignment operator (??=) work in Dart?**
11. **What does the conditional member access operator (?.) do in Dart?**
12. **How is the cascade operator (..) used in Dart?**
13. **What is the purpose of the spread operator (...) in Dart?**
14. **How does the null-aware spread operator (...?) differ from the spread operator in Dart?**
15. **What is the function of the type test operator (is) in Dart?**
16. **What does the type cast operator (as) do in Dart?**
17. **How can the ?. operator be useful in Dart when dealing with potentially null values?**
18. **What does the ! operator do in Dart when used after a variable?**
19. **How do you create an empty String list in Dart?**
20. **How do you use a generic method in Dart?**

21. List the built-in types in Dart.

22. How do you import the Dart math library?

23. What is null safety in Dart?

24. What is the build method in Flutter?

25. How do you handle exceptions in Dart?

26. What are the benefits of using the Provider for state management?

27. How can you implement form validation in Flutter?

28. What are the advantages of responsive design in Flutter applications?

29. How can you optimize Flutter applications for performance?

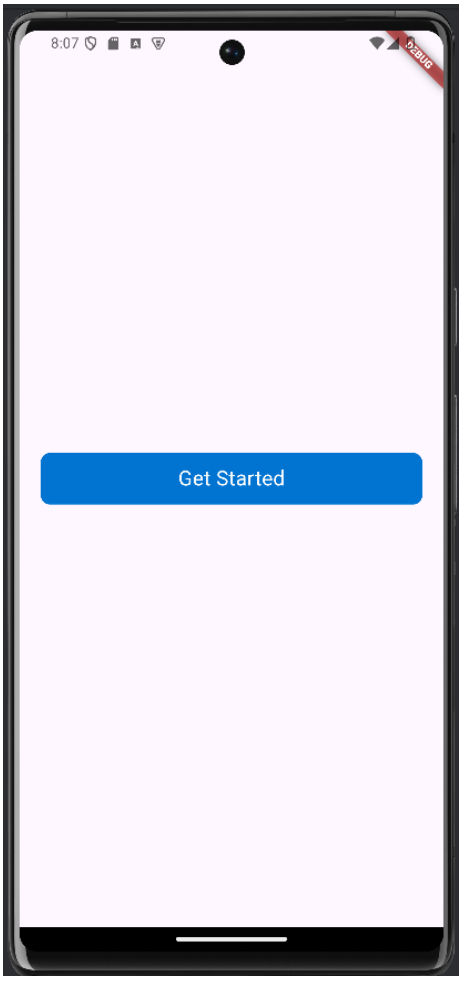
30. What is the role of hot restart in Flutter development?

Experiment 6:

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

```
import 'package:flutter/material.dart';
void main() {
 runApp(Abc());
}
class Abc extends StatelessWidget {
 const Abc({super.key});
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     home: Def(),
   );
 }
}
class Def extends StatelessWidget {
 const Def({super.key});
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     body: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      crossAxisAlignment: CrossAxisAlignment.center,
       children: [
         GestureDetector(
           onTap: () {},
           child: Center(
             child: Container(
               decoration: BoxDecoration(
                 color: Color(0xff0174d2),
                 borderRadius: BorderRadius.circular(9),
               ),
               height: 50,
```

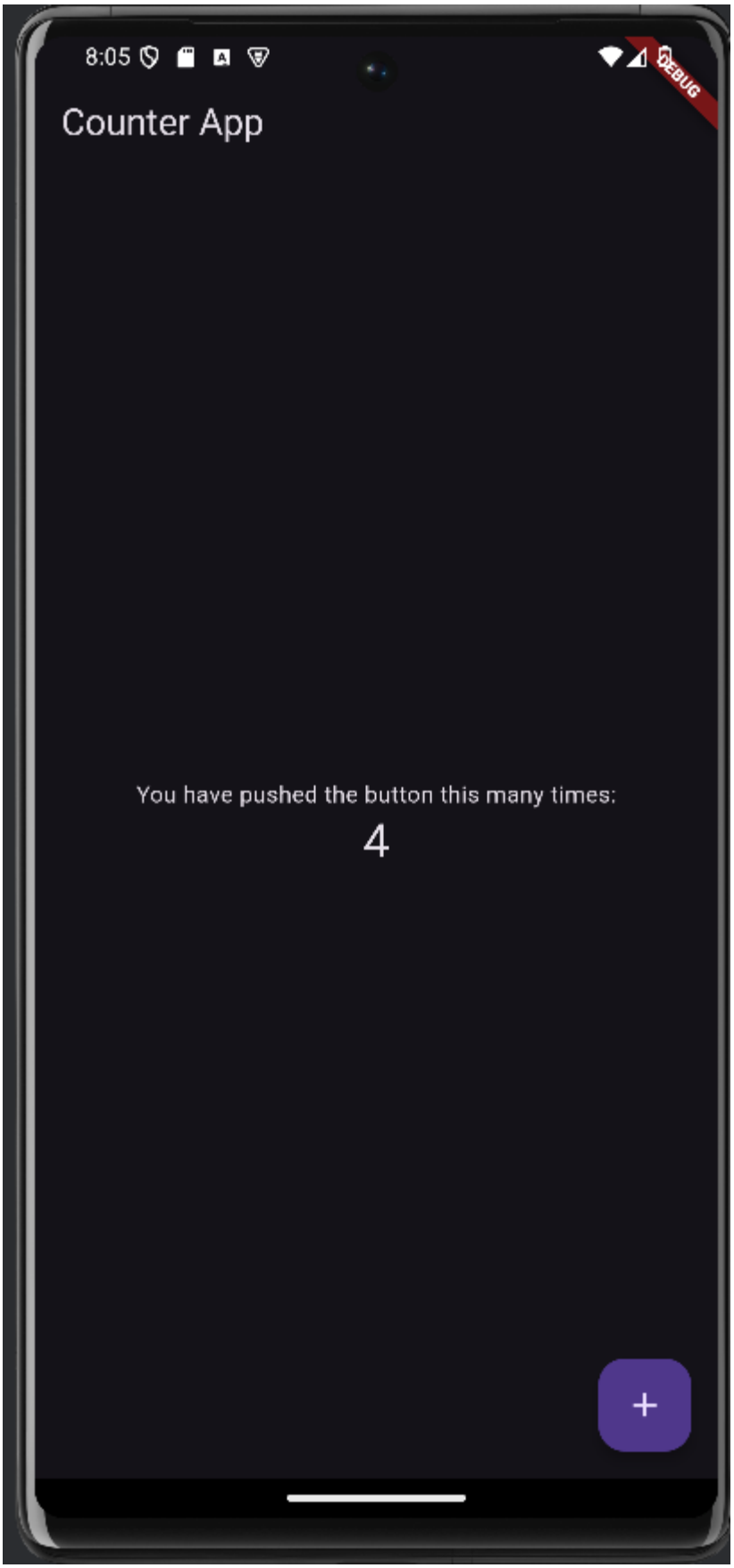
```
width: MediaQuery.of(context).size.width * 0.9,
child: Center(
  child: Text(
    "Get Started",
    style: TextStyle(color: Colors.white, fontSize: 20),
  ),),),),),],),);}}
```



6 b) Apply styling using themes and custom styles.

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


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


class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Flutter Demo',
     theme: ThemeData.dark(),
     home: MyHomePage(),
   );
 }
}


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


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


 void _incrementCounter() {
   setState(() {
     _counter++;
   });
 }


 @override
```

```
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Counter App'),
    ),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Text(
            'You have pushed the button this many times:',
          ),
          Text(
            '$_counter',
            style: Theme.of(context).textTheme.headlineMedium,),],),),
    floatingActionButton: FloatingActionButton(
      onPressed: _incrementCounter,
      tooltip: 'Increment',
      child: Icon(Icons.add),
    ),);}}
```



1. **How do you create an empty String list in Dart?**

2. **How do you use a generic method in Dart?**
3. **List the built-in types in Dart.**
4. **How do you import the Dart math library?**
5. **What is null safety in Dart?**
6. **What is the build method in Flutter?**
7. **How do you handle exceptions in Dart?**
8. **What are the benefits of using the Provider for state management?**
9. **How can you implement form validation in Flutter?**
10. **What are the advantages of responsive design in Flutter applications?**
11. **How can you optimize Flutter applications for performance?**
12. **What is the role of hot restart in Flutter development?**
13. **How does Flutter's widget tree architecture contribute to efficient UI updates?**
14. **What are the best practices for managing state across different screens in Flutter?**
15. **How does the Navigator widget manage navigation between screens in Flutter?**
16. **How do you define and use a named route in a Flutter application?**
17. **How can you pass data between routes using unnamed routes?**
18. **How does the setState method impact the widget tree?**
19. **What is the Provider package used for in Flutter?**
20. **What does the Stack widget do, and when would you use it in a layout?**

21. What is the purpose of the Align widget when used inside a Stack?

22. How do Row and Column widgets handle overflow when their content exceeds available space?

23. How can you control the positioning of children within a Stack?

24. What are Records in Dart?

25. How do you declare a list in Dart?

26. How can MediaQuery be used to adjust font sizes in a responsive design?

27. How do you provide a ChangeNotifier to the widget tree using Provider?

28. What is a Container widget in Flutter?

29. How do you navigate between screens in Flutter?

30. How does the Row widget arrange its children, and when would you use it?

Experiment 7:

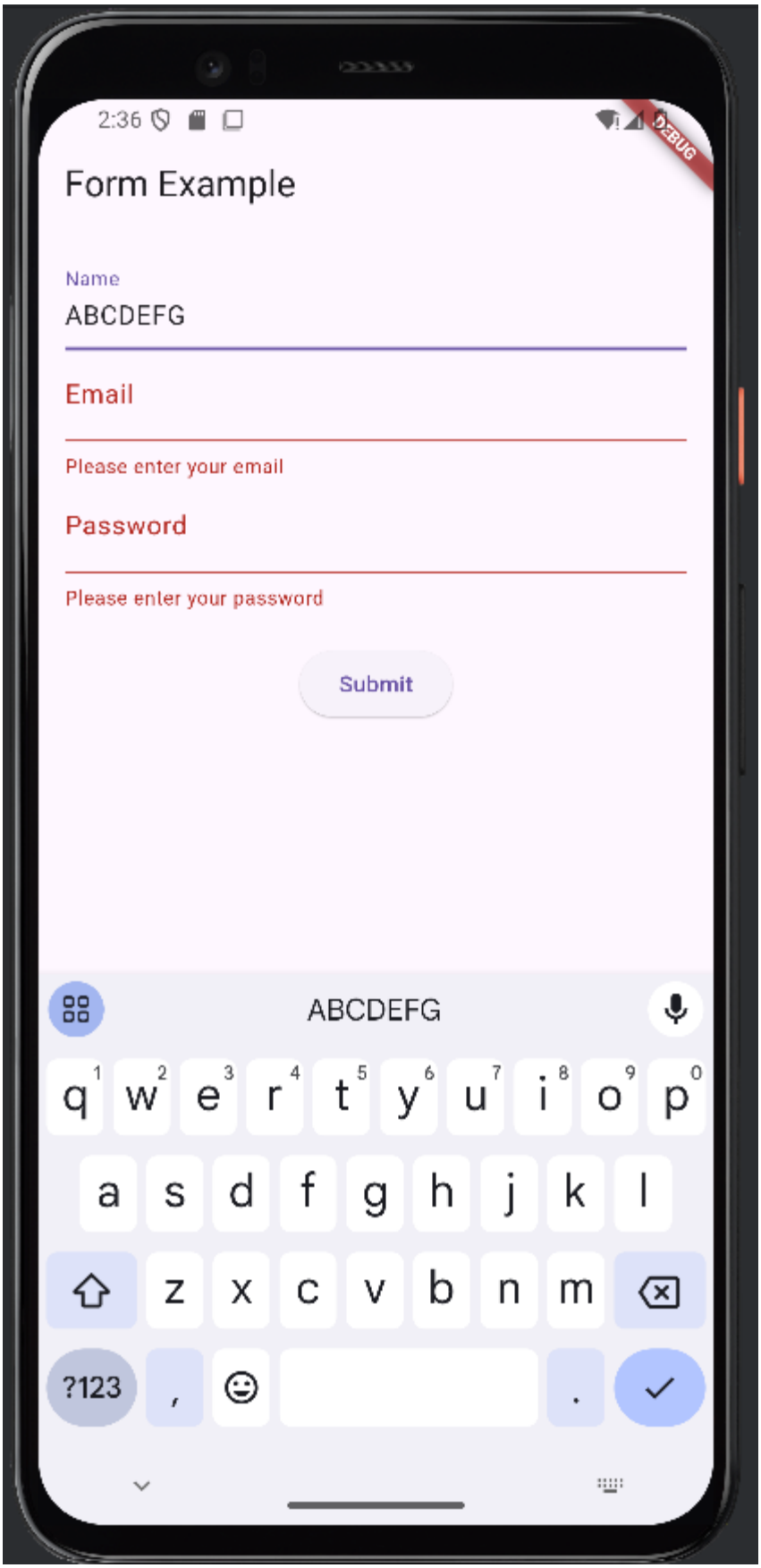
7. a) Design a form with various input fields.

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(
     home: MyForm(),);
}}
class MyForm extends StatefulWidget {
 @override
 _MyFormState createState() => _MyFormState();
}
class _MyFormState extends State<MyForm> {
 final _formKey = GlobalKey<FormState>();
 final _nameController = TextEditingController();
 final _emailController = TextEditingController();
 final _passwordController = TextEditingController();


 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: Text('Form Example'),
     ),
     body: Padding(
```

```
padding: const EdgeInsets.all(16.0),
child: Form(
  key: _formKey,
  child: Column(
    children: <Widget>[
      TextFormField(
        controller: _nameController,
        decoration: InputDecoration(labelText: 'Name'),
        validator: (value) {
          if (value == null || value.isEmpty) {
            return 'Please enter your name';
          }
          return null;
        },
      ),
      TextFormField(
        controller: _emailController,
        decoration: InputDecoration(labelText: 'Email'),
        keyboardType: TextInputType.emailAddress,
        validator: (value) {
          if (value == null || value.isEmpty) {
            return 'Please enter your email';
          } else if (!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(value)) {
            return 'Please enter a valid email address';
          }
          return null;
        },
      ),
      TextFormField(
        controller: _passwordController,
        decoration: InputDecoration(labelText: 'Password'),
        obscureText: true,
        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: 20),
                ElevatedButton(
                  onPressed: () {
                    if (_formKey.currentState?.validate() ?? false) {
                      // If the form is valid, display a snackbar or perform other
actions


                    }
                  },
                  child: Text('Submit'),
                ),],),),),);}}
```


2:36
DEBUG
Form Example
Name
ABCDEFG
Email
Please enter your email
Password
Please enter your password
Submit
ABCDEFG
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123

1. **How do Row and Column widgets handle overflow when their content exceeds available space?**
2. **How can you control the positioning of children within a Stack?**
3. **What are Records in Dart?**
4. **How do you declare a list in Dart?**
5. **How can MediaQuery be used to adjust font sizes in a responsive design?**
6. **How do you provide a ChangeNotifier to the widget tree using Provider?**
7. **What is a Container widget in Flutter?**
8. **How do you navigate between screens in Flutter?**
9. **How does the Row widget arrange its children, and when would you use it?**
10. **How can Flexible and Expanded widgets help in creating responsive layouts?**
11. **What is the main difference between Row and Column widgets in Flutter?**
12. **How does the null-aware operator (??) work in Dart?**
13. **How do you create a basic custom widget in Flutter?**
14. **How can you pass data to a custom widget in Flutter?**
15. **How do you handle user interactions in a custom widget?**
16. **Can you create custom widgets with state in Flutter? If so, how?**
17. **What are the benefits of using custom widgets for specific UI elements?**
18. **How can the Expanded widget be used within a Column to make child widgets take up available space?**

19. **What does the Stack widget do, and when would you use it in a layout?**

20. **How do you create an empty String list in Dart?**

21. **What are the key features of the Dart programming language?**

22. **What is the purpose of the pubspec.yaml file in a Flutter project?**

23. **What is hot reload in Flutter?**

24. **What is the difference between named routes and unnamed routes in Flutter?**

25. **How do you access the provided data in a widget using Provider?**

26. **How do you define a ChangeNotifier in Flutter using the Provider package?**

27. **How do you add external packages to a Flutter project?**

28. **What is the main difference between Row and Column widgets in Flutter?**

29. **What is the Provider package used for in Flutter?**

30. **What is setState in Flutter?**

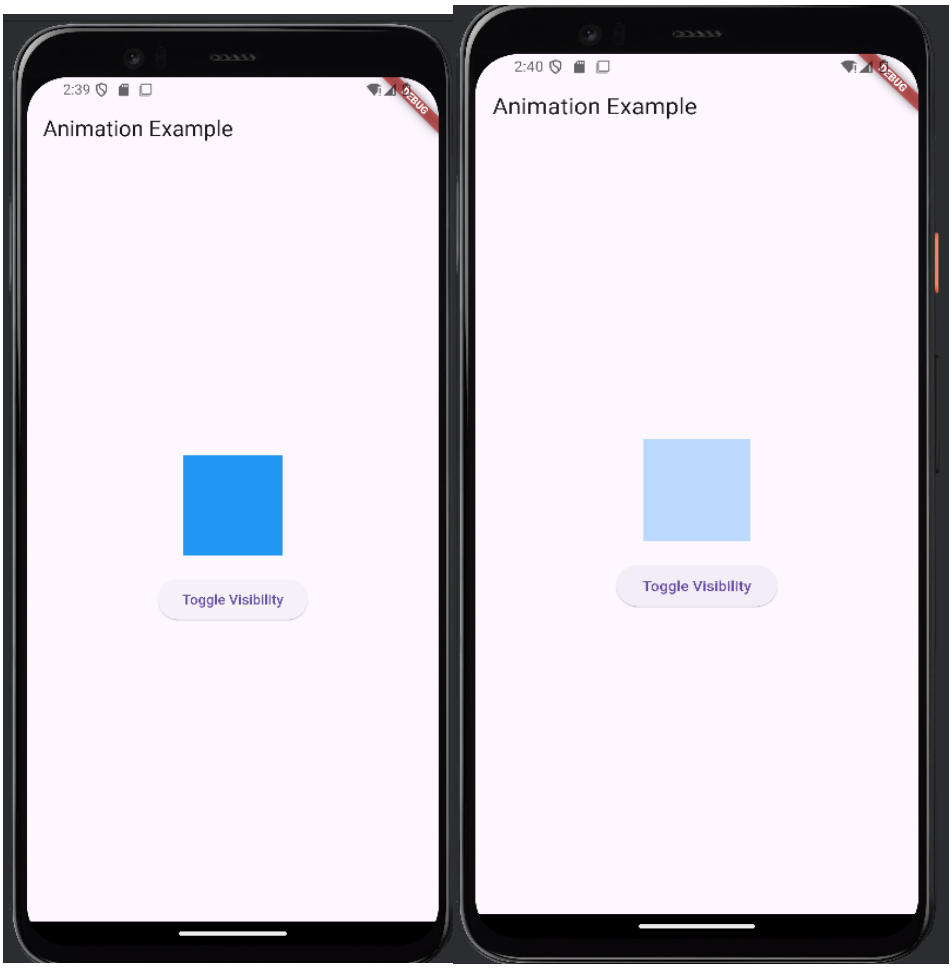
Experiment 8:

8. a) Add animations to UI elements using Flutter's animation framework.

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

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

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Animation Example',
     theme: ThemeData(
       primarySwatch: Colors.blue,
     ),
     home: AnimationScreen(),
   );
 }
}

class AnimationScreen extends StatefulWidget {
 @override
 _AnimationScreenState createState() => _AnimationScreenState();
}

class _AnimationScreenState extends State<AnimationScreen> {
 bool _visible = true;

 void _toggleVisibility() {
   setState(() {
```

```
    _visible = !_visible;
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text('Animation Example')),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          AnimatedOpacity(
            opacity: _visible ? 1.0 : 0.0,
            duration: Duration(seconds: 1),
            child: Container(
              width: 100,
              height: 100,
              color: Colors.blue,
            ),
          ),
          SizedBox(height: 20),
          ElevatedButton(
            onPressed: _toggleVisibility,
            child: Text('Toggle Visibility'),
          ),],),),);}}
```

Output: Fading Animation



8. b) Experiment with Different Types of Animations

Here are examples of different types of animations you can experiment with: fade, slide, and scale.

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


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


class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Animation Examples',
     theme: ThemeData(
       primarySwatch: Colors.blue,
```

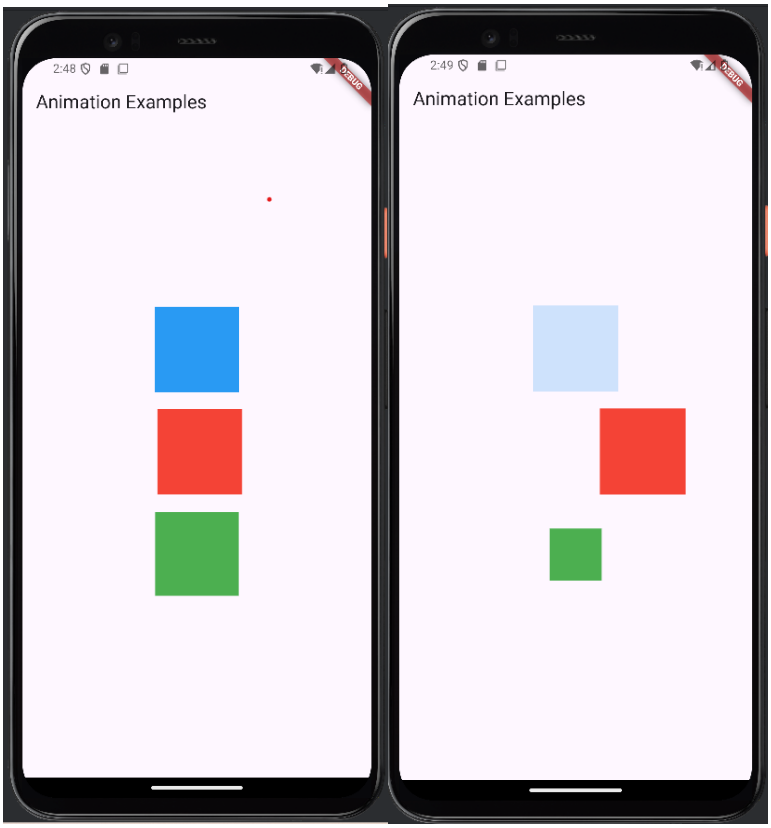
```
      ),
      home: AnimationExamples(),
    );
  }
}
class AnimationExamples extends StatefulWidget {
 @override
 _AnimationExamplesState createState() => _AnimationExamplesState();
}


class _AnimationExamplesState extends State<AnimationExamples> with
SingleTickerProviderStateMixin {
 late AnimationController _controller;
 late Animation<double> _fadeAnimation;
 late Animation<Offset> _slideAnimation;
 late Animation<double> _scaleAnimation;
 @override
 void initState() {
   super.initState();
   _controller = AnimationController(
     duration: Duration(seconds: 2),
     vsync: this,
   )..repeat(reverse: true);
   _fadeAnimation = Tween<double>(begin: 0.0, end: 1.0).animate(_controller);
   _slideAnimation = Tween<Offset>(begin: Offset(1.0, 0.0), end:
Offset.zero).animate(_controller);
   _scaleAnimation = Tween<double>(begin: 0.5, end: 1.0).animate(_controller);
 }
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(title: Text('Animation Examples')),
     body: Center(
       child: Column(
```

```
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            FadeTransition(
              opacity: _fadeAnimation,
              child: Container(
                width: 100,
                height: 100,
                color: Colors.blue,
              ),
            ),
            SizedBox(height: 20),
            SlideTransition(
              position: _slideAnimation,
              child: Container(
                width: 100,
                height: 100,
                color: Colors.red,
              ),
            ),
            SizedBox(height: 20),
            ScaleTransition(
              scale: _scaleAnimation,
              child: Container(
                width: 100,
                height: 100,
                color: Colors.green,),),],),),);}

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

Output :

Fade Animation
Slide Animation

Scale Animation



1. **What are the key features of the Dart programming language?**
2. **What is the purpose of the pubspec.yaml file in a Flutter project?**
3. **What is hot reload in Flutter?**
4. **What is the difference between named routes and unnamed routes in Flutter?**
5. **How do you access the provided data in a widget using Provider?**
6. **How do you define a ChangeNotifier in Flutter using the Provider package?**
7. **How do you add external packages to a Flutter project?**
8. **What is the main difference between Row and Column widgets in Flutter?**
9. **What is the Provider package used for in Flutter?**
10. **What is setState in Flutter?**
11. **What is the role of the RouteSettings class in Flutter's navigation system?**
12. **How can MediaQuery be used to adjust font sizes in a responsive design?**
13. **How does the setState method impact the widget tree?**
14. **What is the Provider package used for in Flutter?**
15. **How does Flutter’s widget tree architecture contribute to efficient UI updates?**
16. **How do you handle user input in Flutter?**
17. **What are the benefits of using the Provider for state management?**
18. **What are the advantages of responsive design in Flutter applications?**

19. **How do you define and use a named route in a Flutter application?**

20. **What is a Stream in Flutter?**

21. **How does Dart handle asynchronous programming?**

22. **What does the build method do in Flutter?**

23. **How do you provide a ChangeNotifier to the widget tree using Provider?**

24. **What is hot reload in Flutter?**

25. **How do you handle exceptions in Dart?**

26. **How do Row and Column widgets handle overflow when their content exceeds available space?**

27. **How do you handle user interactions in a custom widget?**

28. **What is the main difference between Row and Column widgets in Flutter?**

29. **What is the build method in Flutter?**

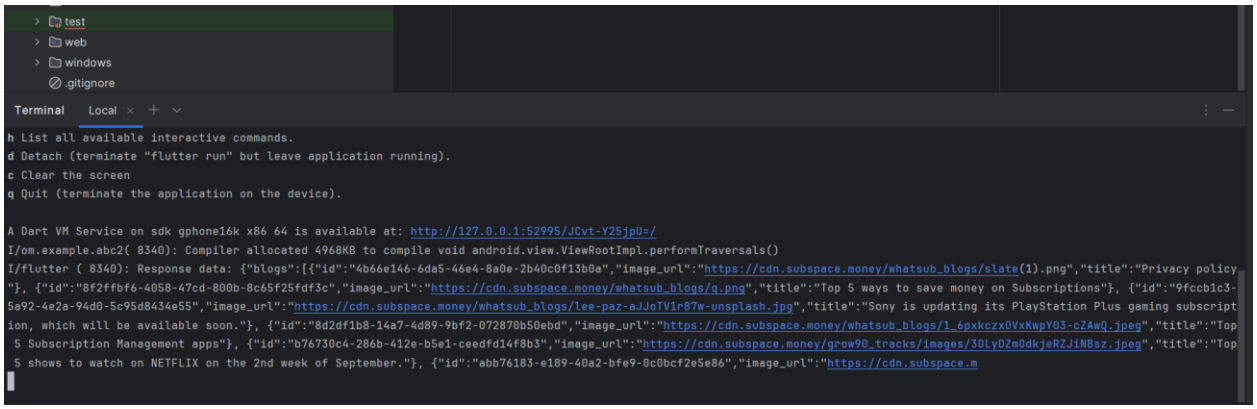
30. **How can you create custom widgets with state in Flutter?**

Experiment 9:

9. a) Fetch data from a REST API.

Note: You have to get the “http” package from “pub.dev” website.

```
import 'package:http/http.dart' as http;

void fetchBlogs() async {

 final String url = 'https://intent-kit-16.hasura.app/api/rest/blogs';

 final String adminSecret =
'32qR4KmXOIpsGPQKMqEJHGJS27G5s7HdSKO3gdtQd2kv5e852SiYwWNfxkZOBuQ6';

 try {

   final response = await http.get(Uri.parse(url), headers: {

     'x-hasura-admin-secret': adminSecret,

   });

   if (response.statusCode == 200) {

     // Request successful, handle the response data here

     print('Response data: ${response.body}');

   } else {

     // Request failed

     print('Request failed with status code: ${response.statusCode}');

     print('Response data: ${response.body}');

   }

 } catch (e) {

      // Handle any errors that occurred during the request

      print('Error: $e');  }

}

void main() {

 fetchBlogs();

}
```


test
web
windows
.gitignore
Terminal
Local
h List all available interactive commands.
d Detach (terminate "flutter run" but leave application running).
c Clear the screen
q Quit (terminate the application on the device).
A Dart VM Service on sdk gphone16k x86 64 is available at: http://127.0.0.1:52995/JCvt-Y25jpU=/
I/om.example.abc2(8340): Compiler allocated 4968KB to compile void android.view.ViewRootImpl.performTraversals()
I/flutter (8340): Response data: {"blogs":[{"id":"4b66e146-6da5-46e4-8a0e-2b40c0f13b0a","image_url":"https://cdn.subspace.money/whatsub_blogs/slate(1).png","title":"Privacy policy
"}, {"id":"8f2ffbf6-4058-47cd-800b-8c65f25fdf3c","image_url":"https://cdn.subspace.money/whatsub_blogs/q.png","title":"Top 5 ways to save money on Subscriptions"}, {"id":"9fccb1c3-
5a92-4e2a-94d0-5c95d8434e55","image_url":"https://cdn.subspace.money/whatsub_blogs/lee-paz-aJJoTV1r87w-unsplash.jpg","title":"Sony is updating its PlayStation Plus gaming subscript
ion, which will be available soon."}, {"id":"8d2df1b8-14a7-4d89-9bf2-072870b50ebd","image_url":"https://cdn.subspace.money/whatsub_blogs/1_6pxkczx0VxKwpY03-cZAwQ.jpeg","title":"Top
5 Subscription Management apps"}, {"id":"b76730c4-286b-412e-b5e1-ceedfd14f8b3","image_url":"https://cdn.subspace.money/grow90_tracks/images/3OLyDZm0dkjeRZJiNBsz.jpeg","title":"Top
5 shows to watch on NETFLIX on the 2nd week of September."}, {"id":"abb76183-e189-40a2-bfe9-0c0bcf2e5e86","image_url":"https://cdn.subspace.m

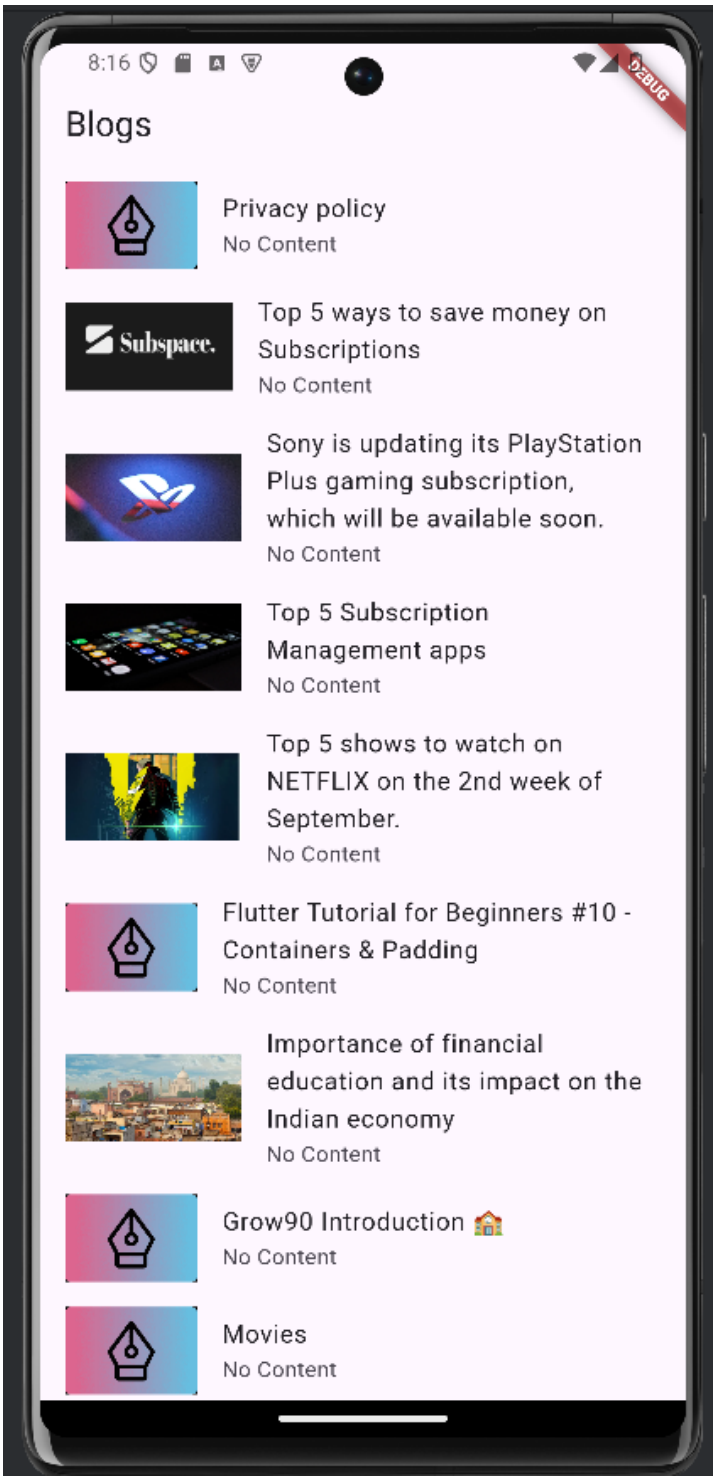
9 b) Display the fetched data in a meaningful way in the UI.

```
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: 'Blog App',
     theme: ThemeData(
       primarySwatch: Colors.blue,
     ),
     home: BlogListScreen(),
   );
 }
}

class BlogListScreen extends StatefulWidget {
 @override
 _BlogListScreenState createState() => _BlogListScreenState();
}

class _BlogListScreenState extends State<BlogListScreen> {
 late Future<List<Blog>> futureBlogs;

 @override
 void initState() {
   super.initState();
```

```
  futureBlogs = fetchBlogs();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Blogs'),
    ),
    body: FutureBuilder<List<Blog>>(
      future: futureBlogs,
      builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
          return Center(child: CircularProgressIndicator());
        } else if (snapshot.hasError) {
          return Center(child: Text('Error: ${snapshot.error}'));
        } else if (!snapshot.hasData || snapshot.data!.isEmpty) {
          return Center(child: Text('No blogs available.'));
        } else {
          final blogs = snapshot.data!;
          return ListView.builder(
            itemCount: blogs.length,
            itemBuilder: (context, index) {
              final blog = blogs[index];
              return ListTile(
                leading: blog.imageUrl != null
                    ? Image.network(blog.imageUrl!)
                    : null,
                title: Text(blog.title),
                subtitle: Text(blog.content),
              );
            },
          );
```

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

Future<List<Blog>> fetchBlogs() async {
  final String url = 'https://intent-kit-16.hasura.app/api/rest/blogs';
  final String adminSecret =
      '32qR4KmXOIpsGPQKMqEJHGJS27G5s7HdSKO3gdtQd2kv5e852SiYwWNfxkZOBuQ6';

  try {
    final response = await http.get(Uri.parse(url), headers: {
      'x-hasura-admin-secret': adminSecret,
    });

    if (response.statusCode == 200) {
      final Map<String, dynamic> data = json.decode(response.body);
      final List<dynamic> blogList =
      data['blogs']; // Adjust this line based on your actual JSON structure
      return blogList.map((json) => Blog.fromJson(json)).toList();
    } else {
      throw Exception('Failed to load blogs');
    }
  } catch (e) {
    throw Exception('Error: $e');
  }
}

class Blog {
  final String title;
  final String content;
```

```
 final String? imageUrl;


 Blog({required this.title, required this.content, this.imageUrl});


 factory Blog.fromJson(Map<String, dynamic> json) {
   return Blog(
     title: json['title'] ?? 'No Title',
     content: json['content'] ?? 'No Content',
     imageUrl:
     json['image_url'], // Adjust the key based on your JSON structure
   );
 }
}
```



Note : https://subspace.money/blog/whatsub-single-docs-internship-onboarding-docs-problem-statement-flutter-blog-explorer

1. **What is the MaterialApp widget?**
2. **How do you declare a list in Dart?**
3. **What are Records in Dart?**
4. **How do you define a class in Dart?**
5. **What does the conditional member access operator (?.) do in Dart?**
6. **How is the cascade operator (..) used in Dart?**
7. **How do you access the provided data in a widget using Provider?**
8. **What are the key features of the Dart programming language?**
9. **What is the difference between final and const in Dart?**
10. **How do you handle exceptions in Dart?**
11. **How do you import the Dart math library?**
12. **How can MediaQuery be used to adjust font sizes in a responsive design?**
13. **What is the difference between MediaQuery and LayoutBuilder in handling responsive layouts?**
14. **How does the Row widget arrange its children, and when would you use it?**
15. **What is the main difference between Row and Column widgets in Flutter?**
16. **How can the Expanded widget be used within a Column to make child widgets take up available space?**
17. **What does the Stack widget do, and when would you use it in a layout?**
18. **How can you control the positioning of children within a Stack?**
19. **What is the purpose of the Align widget when used inside a Stack?**

20. **How do Row and Column widgets handle overflow when their content exceeds available space?**

21. **How does the Navigator widget manage navigation between screens in Flutter?**

22. **What is the difference between named routes and unnamed (anonymous) routes in Flutter?**

23. **How do you define and use a named route in a Flutter application?**

24. **How can you pass data between routes using unnamed routes?**

25. **What is the role of the RouteSettings class in Flutter's navigation system?**

26. **What is the purpose of the setState method in Flutter?**

27. **How does the setState method impact the widget tree?**

28. **What is the Provider package used for in Flutter?**

29. **How do you define a ChangeNotifier in Flutter using the Provider package?**

30. **What are the benefits of using custom widgets for specific UI elements?**

Experiment 10:

10. a) Write unit tests for UI components.

main.dart

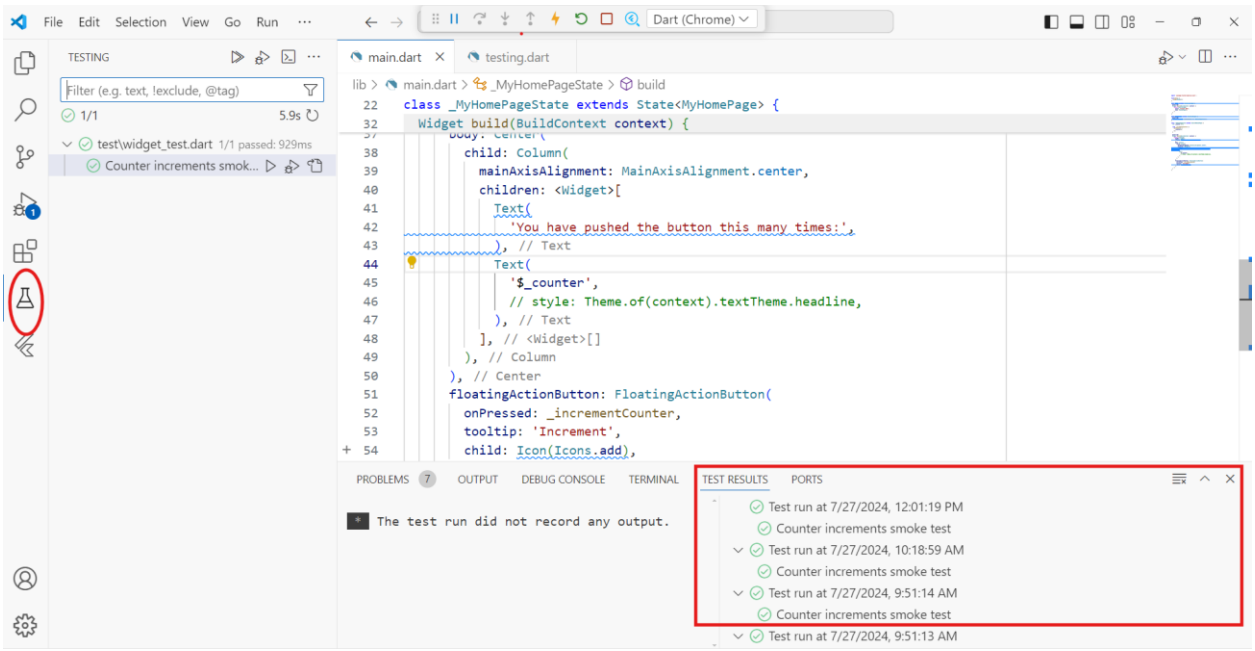
```
import 'package:flutter/material.dart';

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

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

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

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

 void _incrementCounter() {
   setState(() {
     _counter++;
   });
```

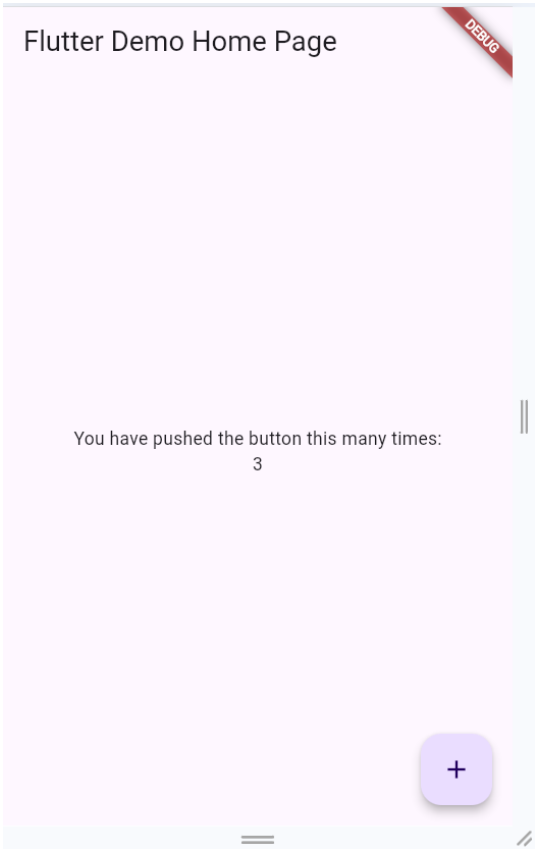
```
 }
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: Text('Flutter Demo Home Page'),
     ),
     body: Center(
       child: Column(
         mainAxisAlignment: MainAxisAlignment.center,
         children: <Widget>[
           Text(
             'You have pushed the button this many times:',
           ),
           Text(
             '$_counter',
             // style: Theme.of(context).textTheme.headline,
           ),
         ],),),
     floatingActionButton: FloatingActionButton(
       onPressed: _incrementCounter,
       tooltip: 'Increment',
       child: Icon(Icons.add),
     ),);}}
```

testing.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:testing/main.dart';

void main() {
 testWidgets('Counter increments smoke test', (WidgetTester tester) async {
   // Build our app and trigger a frame.
   await tester.pumpWidget(MyApp());
```

```
    // Verify that our counter starts at 0.
    expect(find.text('0'), findsOneWidget);
    expect(find.text('1'), findsNothing);

    // Tap the '+' icon and trigger a frame.
    await tester.tap(find.byIcon(Icons.add));
    await tester.pump();

    // Verify that our counter has incremented.
    expect(find.text('0'), findsNothing);
    expect(find.text('1'), findsOneWidget);
  });
}
```

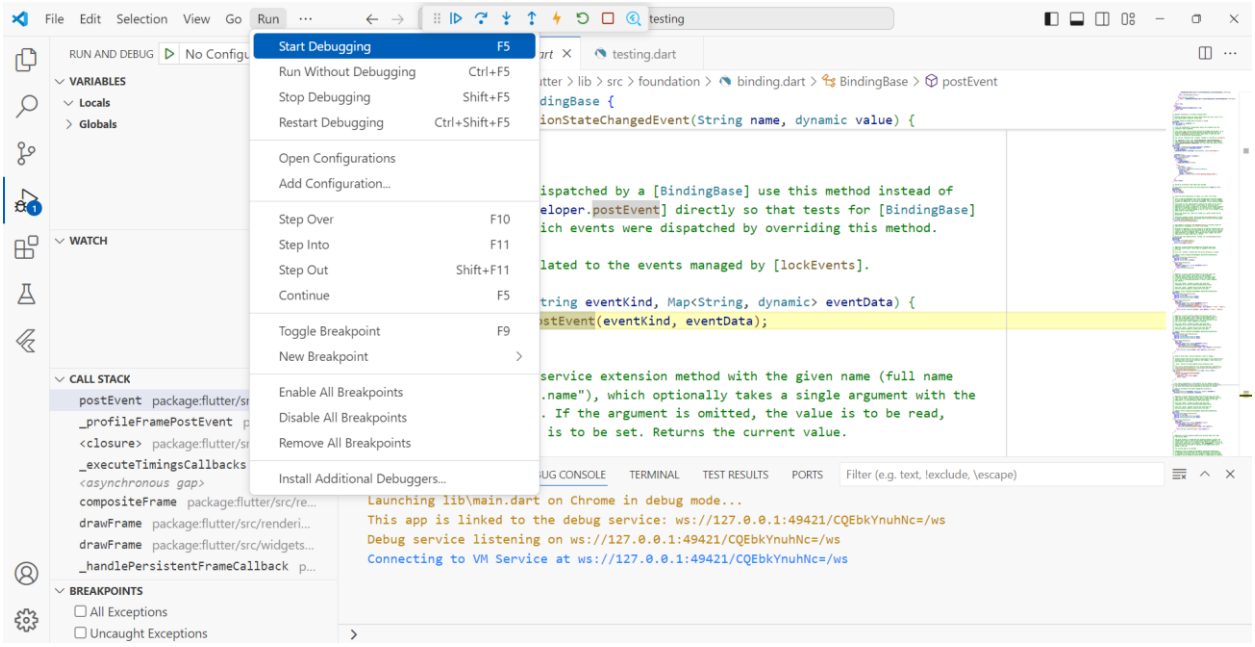




10 b) Use Flutter's debugging tools to identify and fix issues.

Monitor the debug console for any error messages or logs. Flutter provides detailed error messages that often include stack traces and helpful tips.

It is used for checking the flow of the Application by passing through classes, and their functions.



1. **What is unit testing?**
2. **What is integration testing?**
3. **What is the difference between unit testing and integration testing?**
4. **What is test-driven development (TDD)?**
5. **What are mocks and stubs in testing?**
6. **How do you write a unit test in Dart?**
7. **What is the purpose of the test package in Dart?**
8. **How do you run tests in a Flutter project?**
9. **What is widget testing in Flutter?**
10. **How do you write a widget test in Flutter?**
11. **What is the flutter_test package used for?**
12. **How do you mock dependencies in Flutter tests?**
13. **What is the purpose of the mockito package in Dart?**
14. **How do you perform integration testing in Flutter?**
15. **What is the integration_test package in Flutter?**
16. **How do you set up a test environment in Flutter?**
17. **What is code coverage and why is it important?**
18. **How do you measure code coverage in a Flutter project?**
19. **What are the best practices for writing tests in Flutter?**
20. **How do you handle asynchronous code in tests?**
21. **What is continuous integration (CI) and how does it relate to testing?**
22. **How do you set up CI for a Flutter project?**
23. **What is the role of assertions in testing?**
24. **How do you test for exceptions in Dart?**

25. What is a golden test in Flutter?

26. How do you perform golden tests in Flutter?

27. What is the purpose of the testWidgets function in Flutter?

28. How do you test user interactions in Flutter?

29. What is the difference between expect and assert in Dart tests?

30. How do you organize test files in a Flutter project?