User Interface Design using Flutter (1012233180)

1. a) Install Flutter and Dart SDK

- ➢ **System Requirements:** Before installation, make sure your system meets these requirements:
- **OS**: Windows 10+, macOS 10.14+, or Linux
- **Disk Space**: Minimum 2.8 GB (excluding IDE/tools)
- **Tools**: Git must be installed

- ➢ **Download Flutter SDK**
- Visit: https://flutter.dev/docs/get-started/install
- Choose your OS and download the **Flutter SDK zip**.

Example for Windows: Extract flutter folder to a location like C:\src\flutter (avoid directories with spaces).

- ➢ **Add Flutter to System Path**

Windows:

1. Go to: **Start > Environment Variables**
2. Add: C:\src\flutter\bin to **Path** in user/system environment variables

macOS/Linux:
Add this to your shell config (.bashrc, .zshrc, or .bash_profile):

```
export PATH="$PATH:`pwd`/flutter/bin"
```

- ➢ **Run Flutter Doctor:** Open **Command Prompt** or **Terminal**, then run: flutter doctor

This checks for

- Flutter SDK
- Dart SDK (comes bundled with Flutter)
- Android Studio / VS Code
- Android toolchain
- Chrome (for web development)

Fix any issues shown by flutter doctor.

- ➢ **Install Android Studio or VS Code**

Android Studio

- Download: https://developer.android.com/studio
- Install Flutter and Dart plugins via:

Android Studio > Preferences > Plugins > Flutter (includes Dart)

Visual Studio Code

- Download: https://code.visualstudio.com
- Install **Flutter** and **Dart** extensions from the **Extensions Marketplace**

- ➢ **Set Up Emulator or Device**

Android:

- In Android Studio, open **AVD Manager** to create a virtual device.
- Or connect a physical Android phone with **USB Debugging** enabled.

- ➢ **Create and Run Your First Flutter App**

```
flutter create sample
cd sample
flutter run
```

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

```
// main() is the entry point of every Dart program
void main() {
  // 1. Variables and data types
  String name = "John";
  int age = 25;
  double height = 5.9;
  bool isStudent = true;
  print("Name: $name");
  print("Age: $age");
  print("Height: $height feet");
  print("Is student? $isStudent");
  // 2. Conditional statement
  if (age >= 18) {
    print("$name is an adult.");
  } else {
    print("$name is a minor.");
  }
  // 3. Looping
  print("Counting from 1 to 5:");
  for (int i = 1; i <= 5; i++) {
    print(i);
  }
  // 4. Function usage
  int result = add(10, 20);
  print("Sum of 10 and 20 is: $result");
  // 5. Class and object
  Person p1 = Person("Alice", 30);
  p1.showDetails();
}
// Function to add two numbers
int add(int a, int b) {
  return a + b;
}
// Class definition
class Person {
  String name;
  int age;
  // Constructor
  Person(this.name, this.age);
  // Method
  void showDetails() {
    print("Person Name: $name, Age: $age");
  }
}
```

Output:

```
Name: John
Age: 25
Height: 5.9 feet
Is student? true
John is an adult.
Counting from 1 to 5:
1
2
3
```

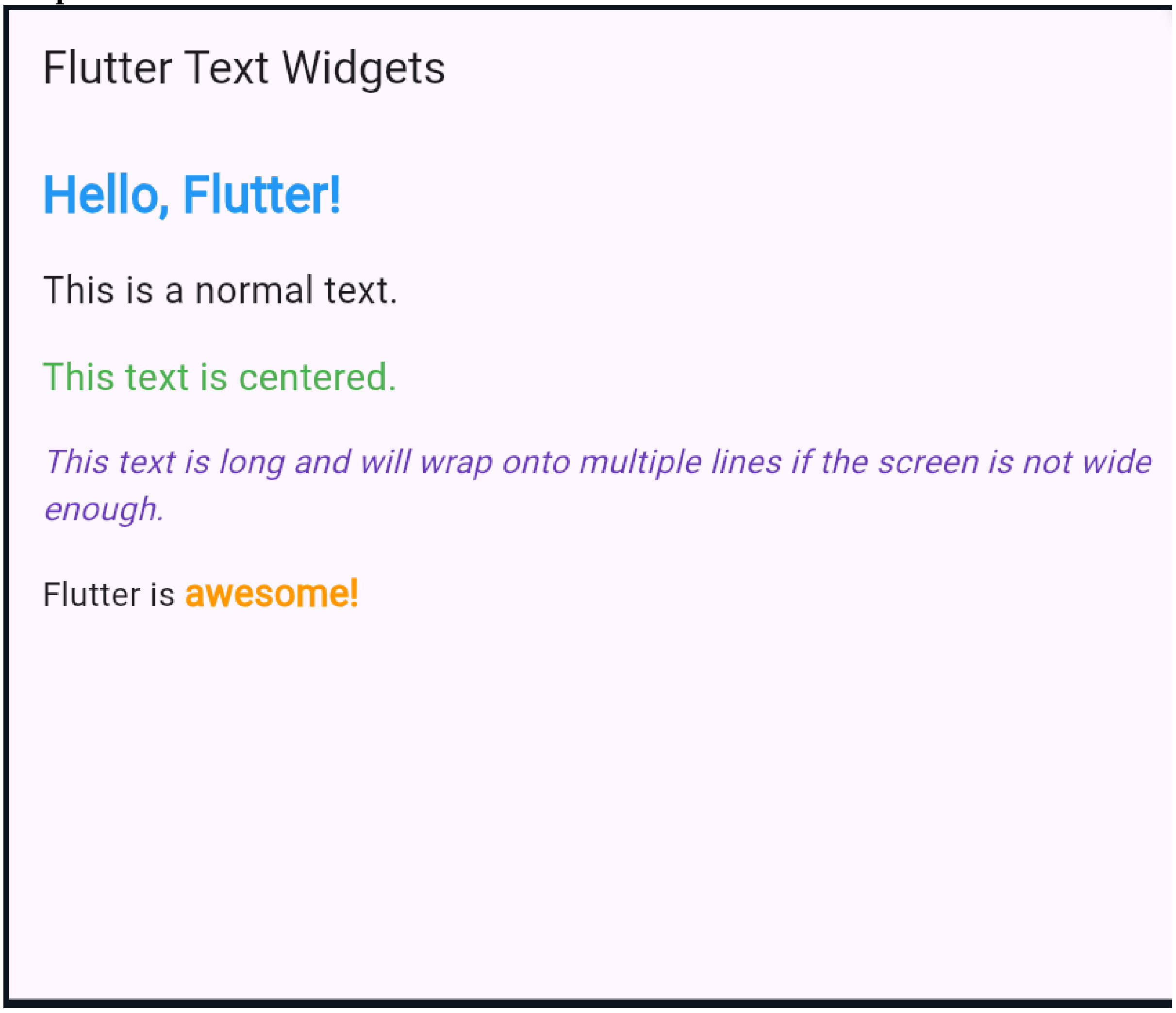
```
4
5
Sum of 10 and 20 is: 30
Person Name: Alice, Age: 30
```

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

Text Flutter:

```
import 'package:flutter/material.dart';
void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Text Widget Example',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Flutter Text Widgets'),
        ),
        body: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Text(
                'Hello, Flutter!',
                style: TextStyle(
                  fontSize: 24,
                  fontWeight: FontWeight.bold,
                  color: Colors.blue,
                ),
              ),
              SizedBox(height: 16),
              Text(
                'This is a normal text.',
                style: TextStyle(
                  fontSize: 18,
                ),
              ),
              SizedBox(height: 16),
              Text(
                'This text is centered.',
                textAlign: TextAlign.center,
                style: TextStyle(
                  fontSize: 18,
                  color: Colors.green,
                ),
              ),
              SizedBox(height: 16),
              Text(
                'This text is long and will wrap onto multiple lines if the screen is not wide 
enough.',
                style: TextStyle(
                  fontSize: 16,
                  fontStyle: FontStyle.italic,
                  color: Colors.deepPurple,
                ),
              ),
              SizedBox(height: 16),
              Text.rich(
                TextSpan(
                  children: [
                    TextSpan(text: 'Flutter is ', style: TextStyle(fontSize: 16)),
                    TextSpan(
                      text: 'awesome!',
                      style: TextStyle(
                        fontWeight: FontWeight.bold,
                        color: Colors.orange,
                        fontSize: 18,
                      ),
                    ),
                  ],
                ),
              ),
            ],
          ),
        ),
      ),
    );
   }
}
```

Output:

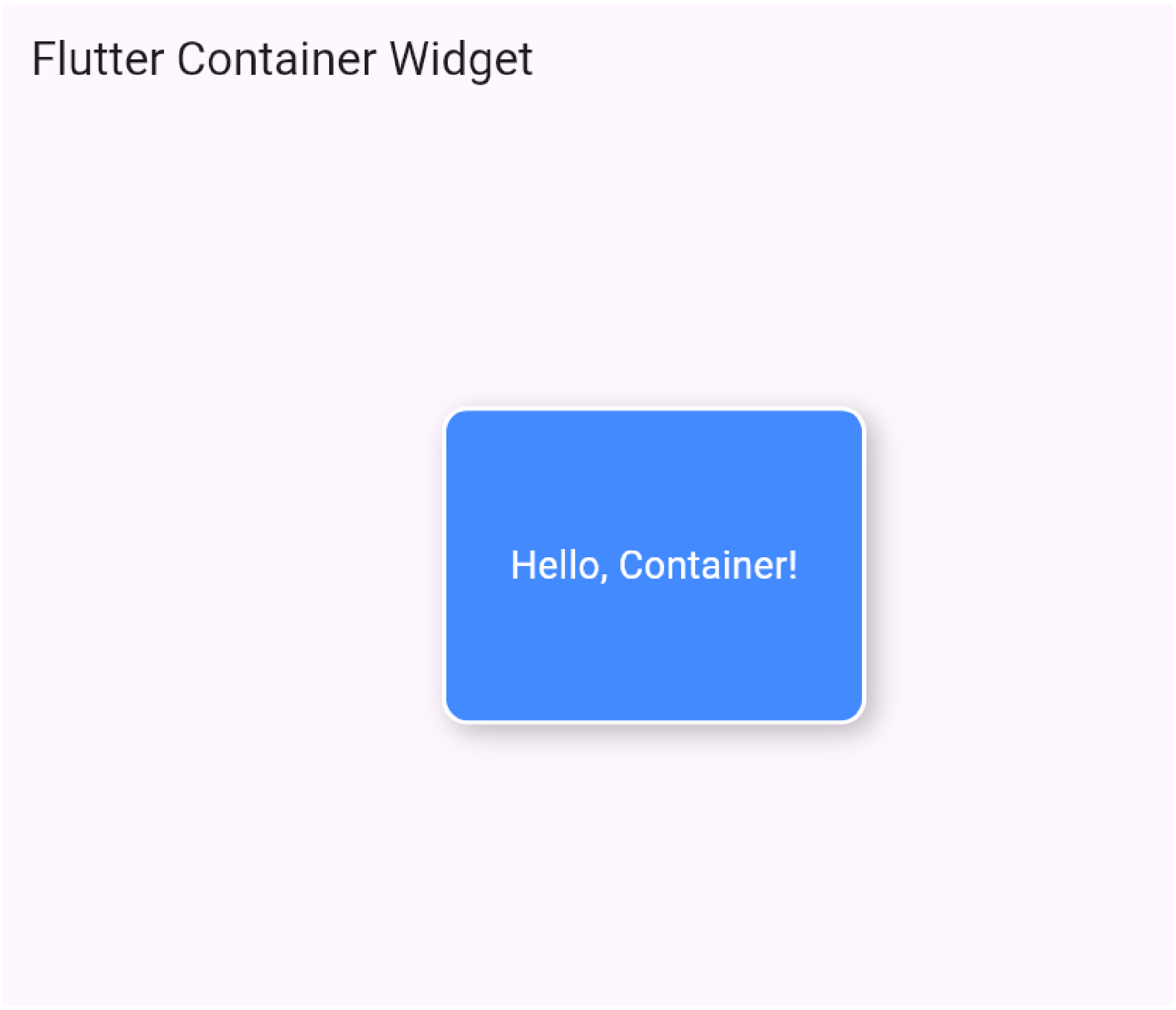


```
Flutter Text Widgets

Hello, Flutter!

This is a normal text.

This text is centered.

This text is long and will wrap onto multiple lines if the screen is not wide
enough.

Flutter is awesome!
```

Container

```
import 'package:flutter/material.dart';
void main() {
  runApp(MyContainerApp());
}
class MyContainerApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Container Example',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Flutter Container Widget'),
        ),
        body: Center(
          child: Container(
            width: 200,
            height: 150,
            padding: EdgeInsets.all(16),
            margin: EdgeInsets.all(20),
            alignment: Alignment.center,
            decoration: BoxDecoration(
              color: Colors.blueAccent,
              border: Border.all(color: Colors.white, width: 2),
              borderRadius: BorderRadius.circular(12),
              boxShadow: [
                BoxShadow(
                  color: Colors.black26,
                  blurRadius: 10,
                  offset: Offset(4, 4),
                ),
              ],
            ),
            child: Text(
              'Hello, Container!',
              style: TextStyle(color: Colors.white, fontSize: 18),
              textAlign: TextAlign.center,
            ),
          ),
        ),
      ),
    );
  }
}
```

Output:

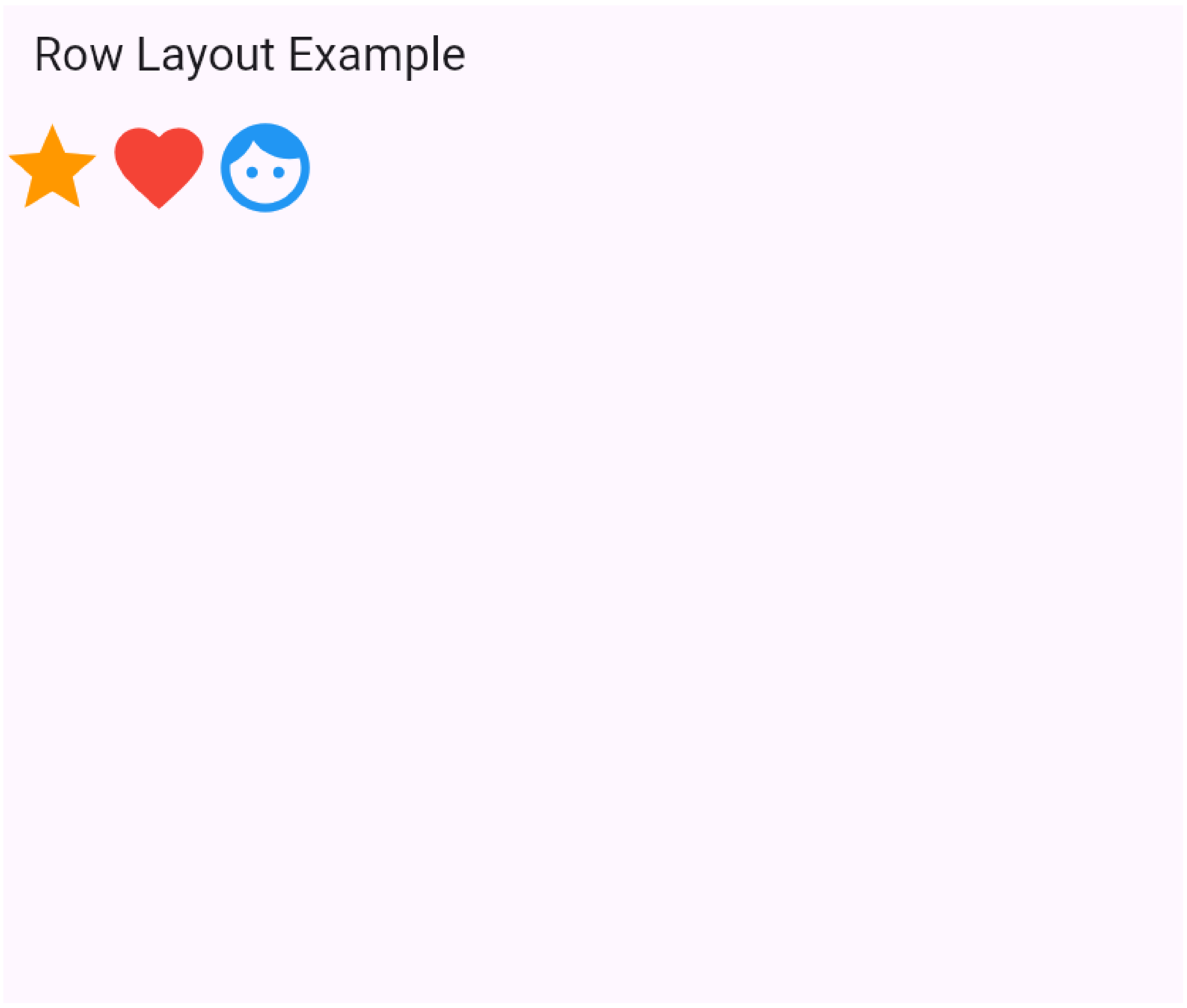


b) Implement different layout structures using Row, Column and Stack widgets

Row Widget layout

```
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 Layout Example")),
        body: Row(
          children: [
            Icon(Icons.star, size: 50, color: Colors.orange),
            Icon(Icons.favorite, size: 50, color: Colors.red),
            Icon(Icons.face, size: 50, color: Colors.blue),
          ],
        ),
      ),
    );
  }
}
```

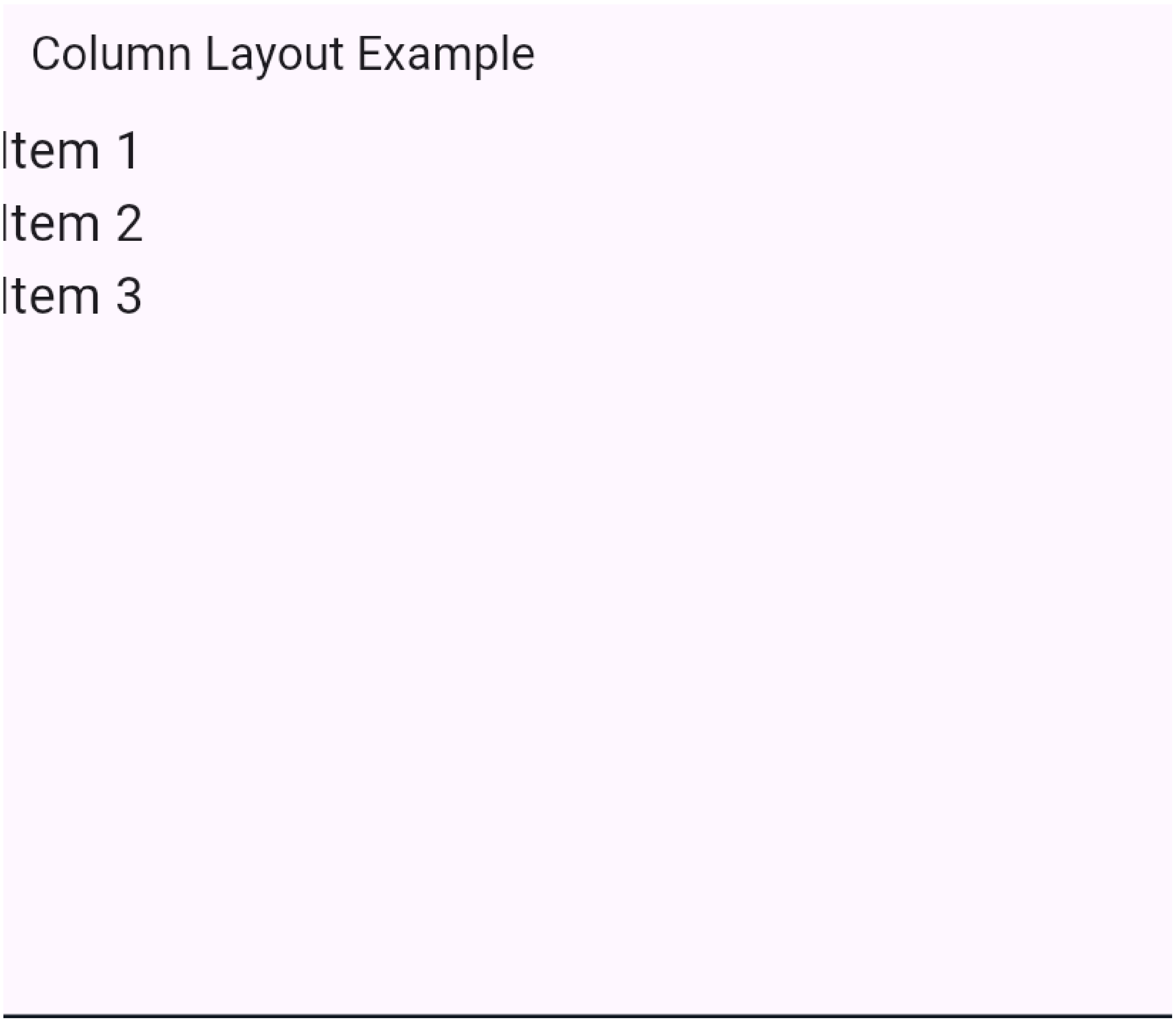
Output:


Row Layout Example

Column Widget layout

```
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("Column Layout Example")),
        body: Column(
          children: [
            Text("Item 1", style: TextStyle(fontSize: 24)),
            Text("Item 2", style: TextStyle(fontSize: 24)),
            Text("Item 3", style: TextStyle(fontSize: 24)),
          ],
        ),
      ),
    );
  }
}
```

Output:


Column Layout Example

Item 1

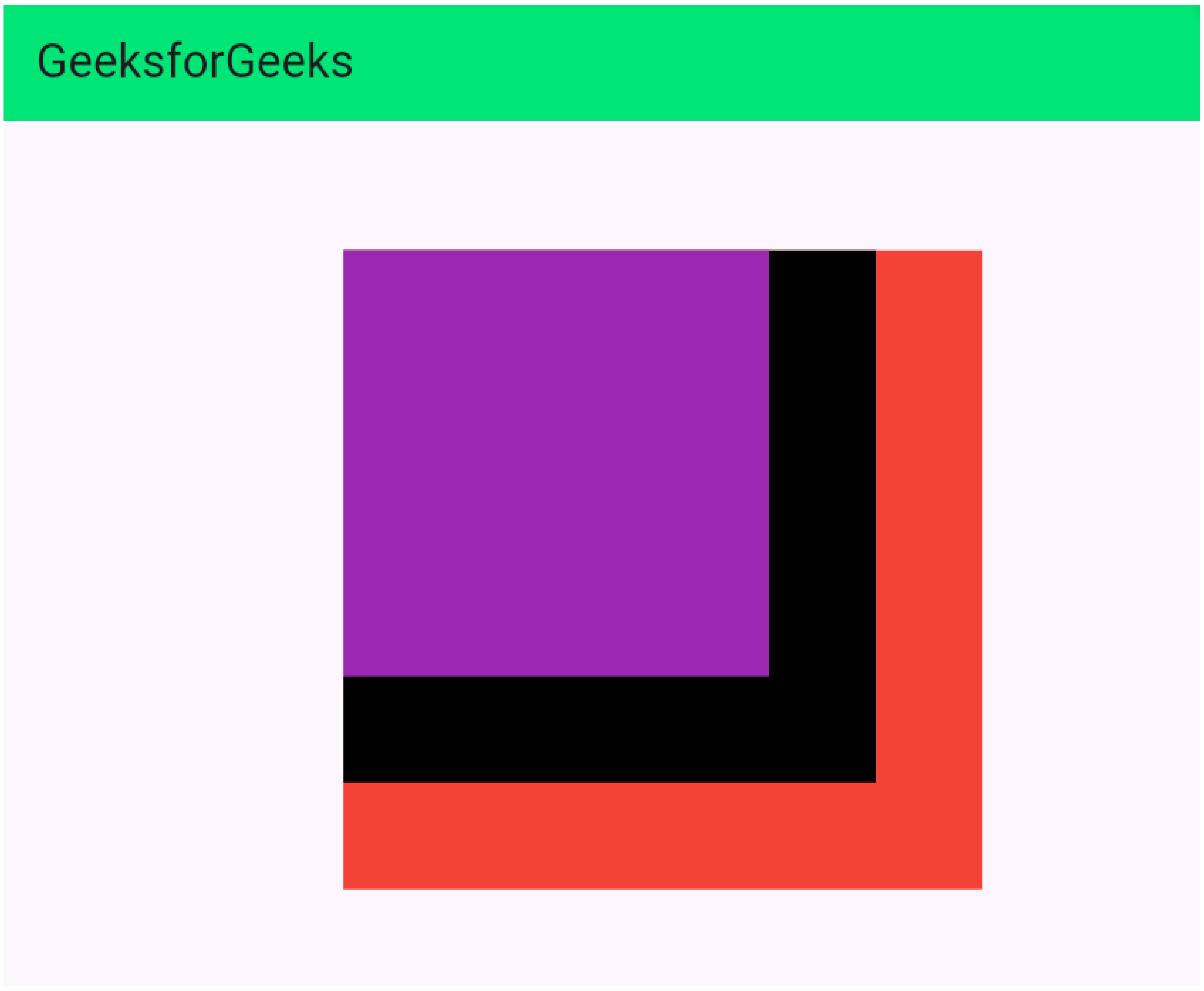
Item 2

Item 3

Stack Widget layout

```
import 'package:flutter/material.dart';
 void main() { runApp(MaterialApp( home: Scaffold( appBar:
 AppBar( title: Text('GeeksforGeeks'),
 backgroundColor: Colors.greenAccent[400],), //AppBar
 body: Center( child: SizedBox( width: 300, height: 300, child: Center( child: Stack( children:
<Widget>[ Container( width: 300, height: 300, color:  Colors.red,), //Container
 Container( width: 250, height: 250, color: Colors.black,), //Container
 Container( height: 200, width: 200, color: Colors.purple,), //Container
 ], //<Widget>[]
 ), //Stack
 ), //Center
 ), //SizedBox
 ) //Center
 ) //Scaffold
 ) //MaterialApp
 );
 }
```

Output:

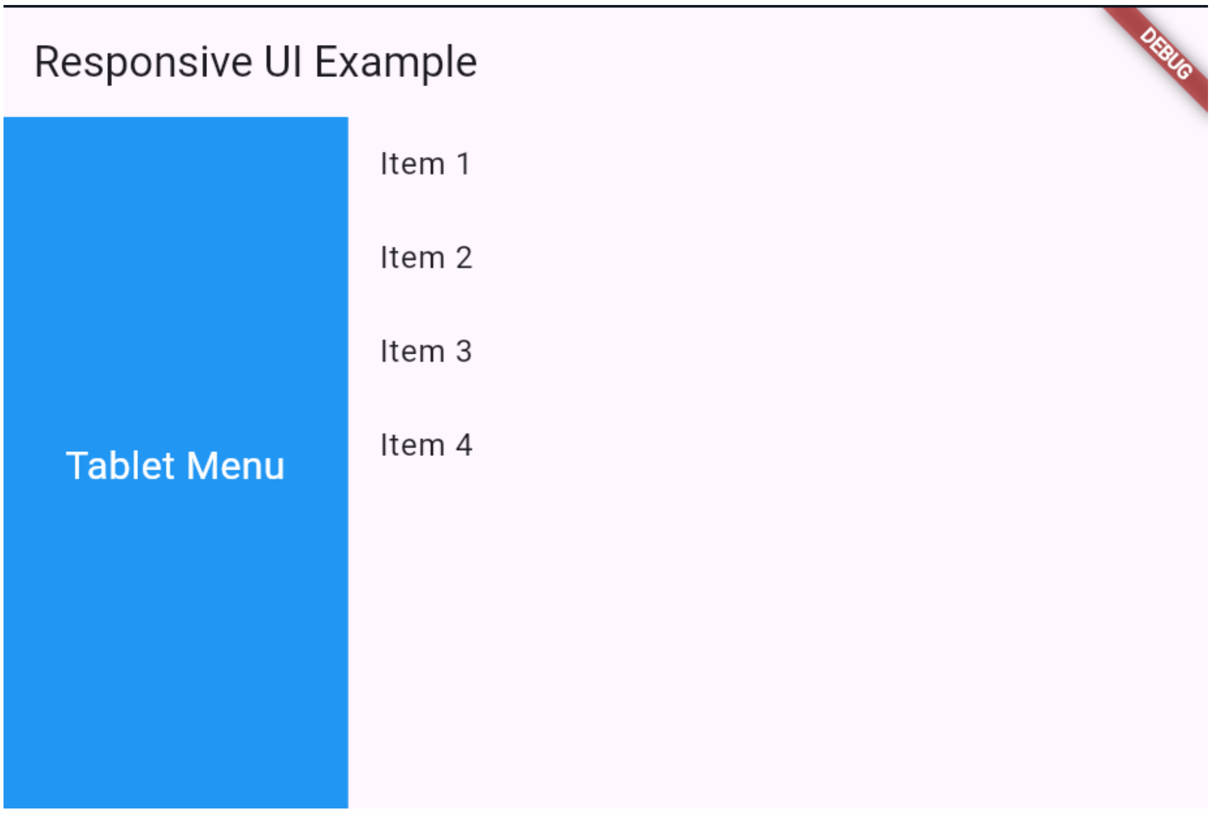


3. a) Design a responsive UI that adapts to different screen sizes

```
import 'package:flutter/material.dart';
void main() {
  runApp(const MyResponsiveApp());
}
class MyResponsiveApp extends StatelessWidget {
  const MyResponsiveApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Responsive UI Example',
      theme: ThemeData(primarySwatch: Colors.blue),
      home: const ResponsiveHomePage(),
    );
  }
}
class ResponsiveHomePage extends StatelessWidget {
  const ResponsiveHomePage({super.key});
 @override
  Widget build(BuildContext context) {
    // Get screen width
    double screenWidth = MediaQuery.of(context).size.width;
 return Scaffold(
      appBar: AppBar(title: const Text("Responsive UI Example")),
      body: LayoutBuilder(
        builder: (context, constraints) {
          if (constraints.maxWidth < 600) {
            // Mobile layout
            return _buildMobileLayout();
          } else if (constraints.maxWidth < 1024) {
            // Tablet layout
            return _buildTabletLayout();
          } else {
            // Desktop layout
            return _buildDesktopLayout();
          }
        },
      ),
    );
  }
  // 📱 Mobile Layout
  Widget _buildMobileLayout() {
    return ListView(
      children: [
        Container(
          color: Colors.blue,
          height: 150,
          child: const Center(
            child: Text("Mobile Header", style: TextStyle(color: Colors.white, fontSize: 20)),
          ),
        ),
        const ListTile(title: Text("Item 1")),
        const ListTile(title: Text("Item 2")),
        const ListTile(title: Text("Item 3")),
      ],
    );
  }

 // 📲 Tablet Layout

  Widget _buildTabletLayout() {
    return Row(
      children: [
        Expanded(
          flex: 2,
          child: Container(
            color: Colors.blue,
            child: const Center(
              child: Text("Tablet Menu", style: TextStyle(color: Colors.white, fontSize: 20)),
            ),
          ),
        ),
        Expanded(
          flex: 5,
          child: ListView(
            children: const [
              ListTile(title: Text("Item 1")),
              ListTile(title: Text("Item 2")),
              ListTile(title: Text("Item 3")),
              ListTile(title: Text("Item 4")),
            ],
          ),
        ),
      ],
    );
  }
 // 💻 Desktop Layout
```

```
  Widget _buildDesktopLayout() {
    return Row(
      children: [
        Expanded(
          flex: 2,
          child: Container(
            color: Colors.blue,
            child: const Center(
              child: Text("Desktop Sidebar", style: TextStyle(color: Colors.white, fontSize: 20)),
            ),
          ),
        ),
        Expanded(
          flex: 5,
          child: GridView.count(
            crossAxisCount: 2,
            children: List.generate(
              6,
              (index) => Card(
                margin: const EdgeInsets.all(10),
                child: Center(child: Text("Card ${index + 1}")),
              ),
            ),
          ),
        ),
      ],
    );
  }
}
```

Output:



Responsive UI Example

Tablet Menu

Item 1

Item 2

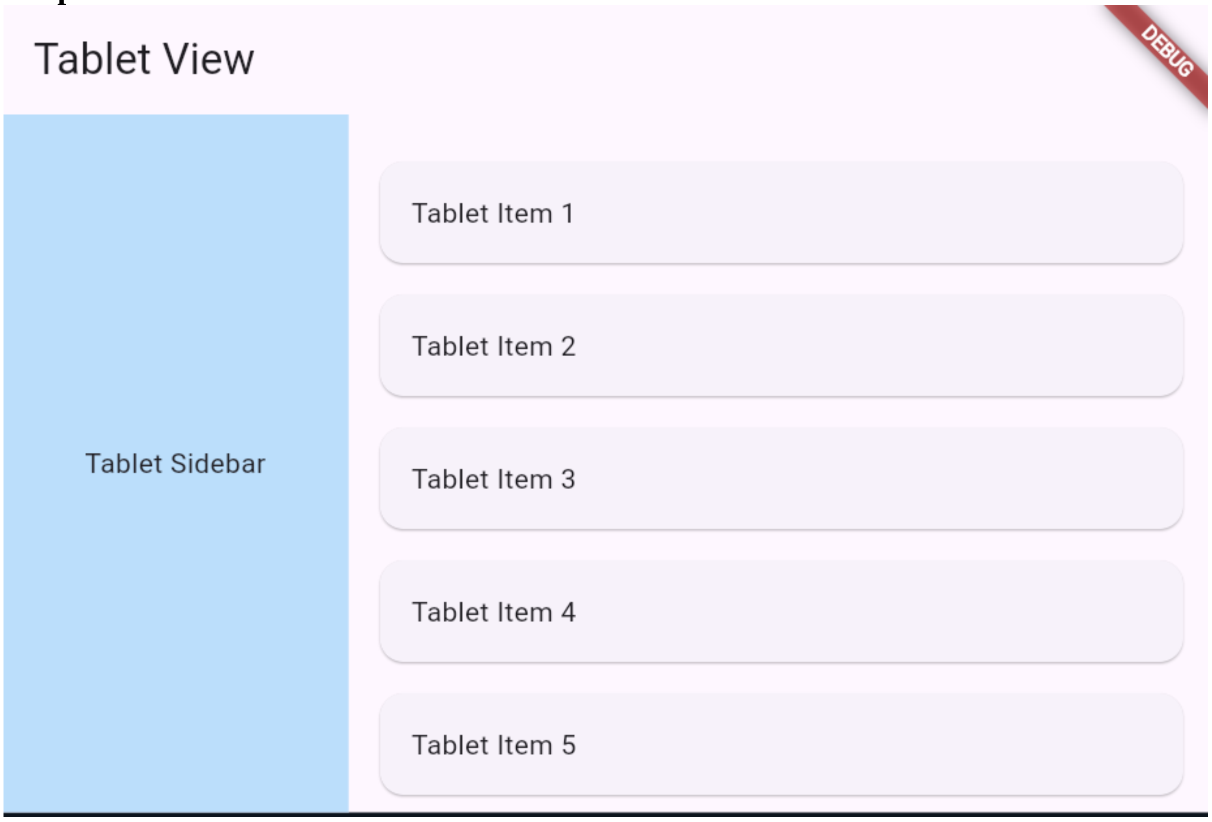
Item 3

Item 4

3 b) Implement media queries and breakpoints for responsiveness

```
import 'package:flutter/material.dart';
void main() {
  runApp(const ResponsiveApp());
}
class ResponsiveApp extends StatelessWidget {
  const ResponsiveApp({super.key});
@override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'MediaQuery Breakpoints Demo',
      theme: ThemeData(primarySwatch: Colors.blue),
      home: const ResponsiveHome(),
    );
  }
}
class ResponsiveHome extends StatelessWidget {
  const ResponsiveHome({super.key});
@override
  Widget build(BuildContext context) {
    // Get screen width using MediaQuery
    double screenWidth = MediaQuery.of(context).size.width;
 // Define breakpoints
    const mobileBreakpoint = 600;
    const tabletBreakpoint = 1024;
// Choose layout based on screen width
    if (screenWidth < mobileBreakpoint) {
      return _mobileLayout();
    } else if (screenWidth < tabletBreakpoint) {
      return _tabletLayout();
    } else {
      return _desktopLayout();
    }
  }
 // 📱 Mobile Layout
  Widget _mobileLayout() {
    return Scaffold(
      appBar: AppBar(title: const Text("Mobile View")),
      body: ListView(
        padding: const EdgeInsets.all(16),
        children: List.generate(
          5,
          (index) => Card(
            margin: const EdgeInsets.symmetric(vertical: 8),
            child: Padding(
              padding: const EdgeInsets.all(16),
              child: Text("Mobile Item ${index + 1}"),
            ),
          ),
        ),
      ),
    );
  }

 // 📲 Tablet Layout

  Widget _tabletLayout() {
    return Scaffold(
      appBar: AppBar(title: const Text("Tablet View")),
      body: Row(
        children: [
          Expanded(
            flex: 2,
            child: Container(
              color: Colors.blue[100],
              child: const Center(child: Text("Tablet Sidebar")),
            ),
          ),
          Expanded(
            flex: 5,
            child: ListView(
              padding: const EdgeInsets.all(16),
              children: List.generate(
                8,
                (index) => Card(
                  margin: const EdgeInsets.symmetric(vertical: 8),
                  child: Padding(
                    padding: const EdgeInsets.all(16),
                    child: Text("Tablet Item ${index + 1}"),
                  ),
                ),
              ),
            ),
          ),
        ],
      ),
    );
  }
 // 💻 Desktop Layout

  Widget _desktopLayout() {
    return Scaffold(
      appBar: AppBar(title: const Text("Desktop View")),
      body: Row(
        children: [
          Expanded(
            flex: 2,
            child: Container(
              color: Colors.blue[200],
              child: const Center(child: Text("Desktop Sidebar")),
            ),
          ),
          Expanded(
            flex: 5,
            child: GridView.count(
              padding: const EdgeInsets.all(16),
              crossAxisCount: 3,
              children: List.generate(
                9,
                (index) => Card(
                  margin: const EdgeInsets.all(8),
                  child: Center(child: Text("Card ${index + 1}")),
                ),
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

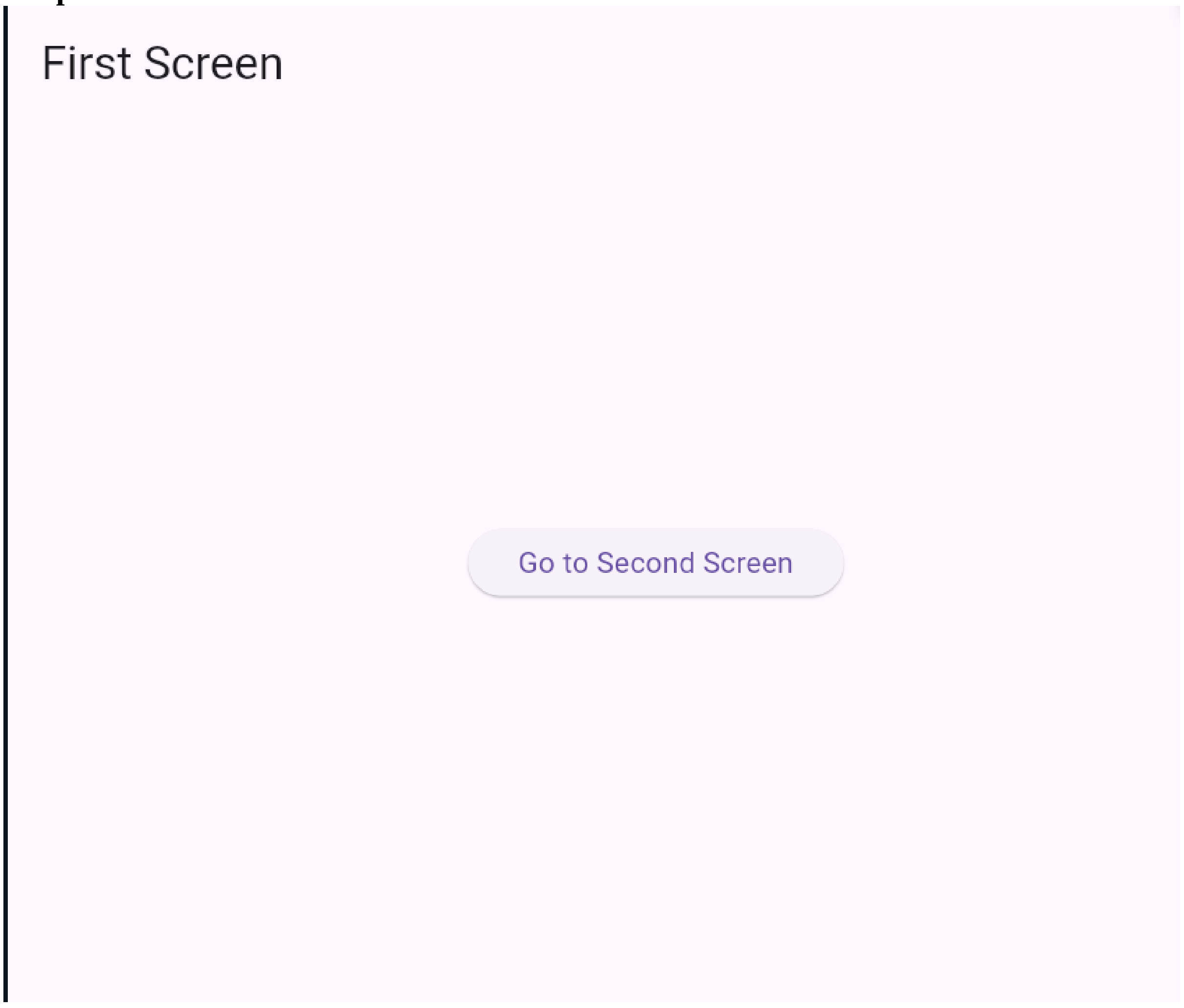
Output:

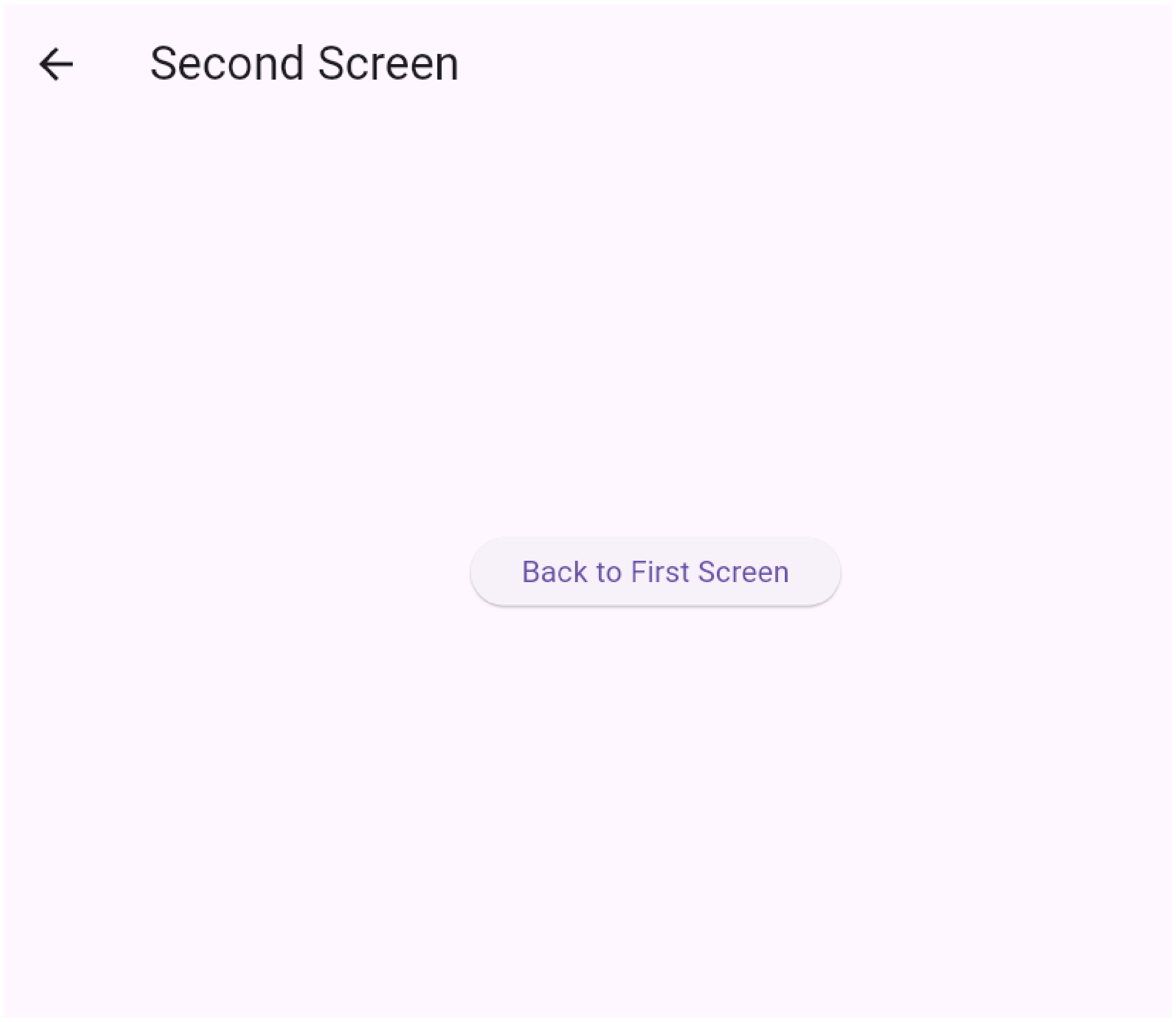


4.A) Set up navigation between different screens using Navigator

```
import 'package:flutter/material.dart';
void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Navigation Demo',
      home: FirstScreen(),
    );
  }
}
class FirstScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('First Screen')),
      body: Center(
        child: ElevatedButton(
          child: Text('Go to Second Screen'),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => SecondScreen()),
            );
          },
        ),
      ),
    );
  }
}
class SecondScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Second Screen')),
      body: Center(
        child: ElevatedButton(
          child: Text('Back to First Screen'),
          onPressed: () {
            Navigator.pop(context);
          },
        ),
      ),
    );
  }
}
```

Output:

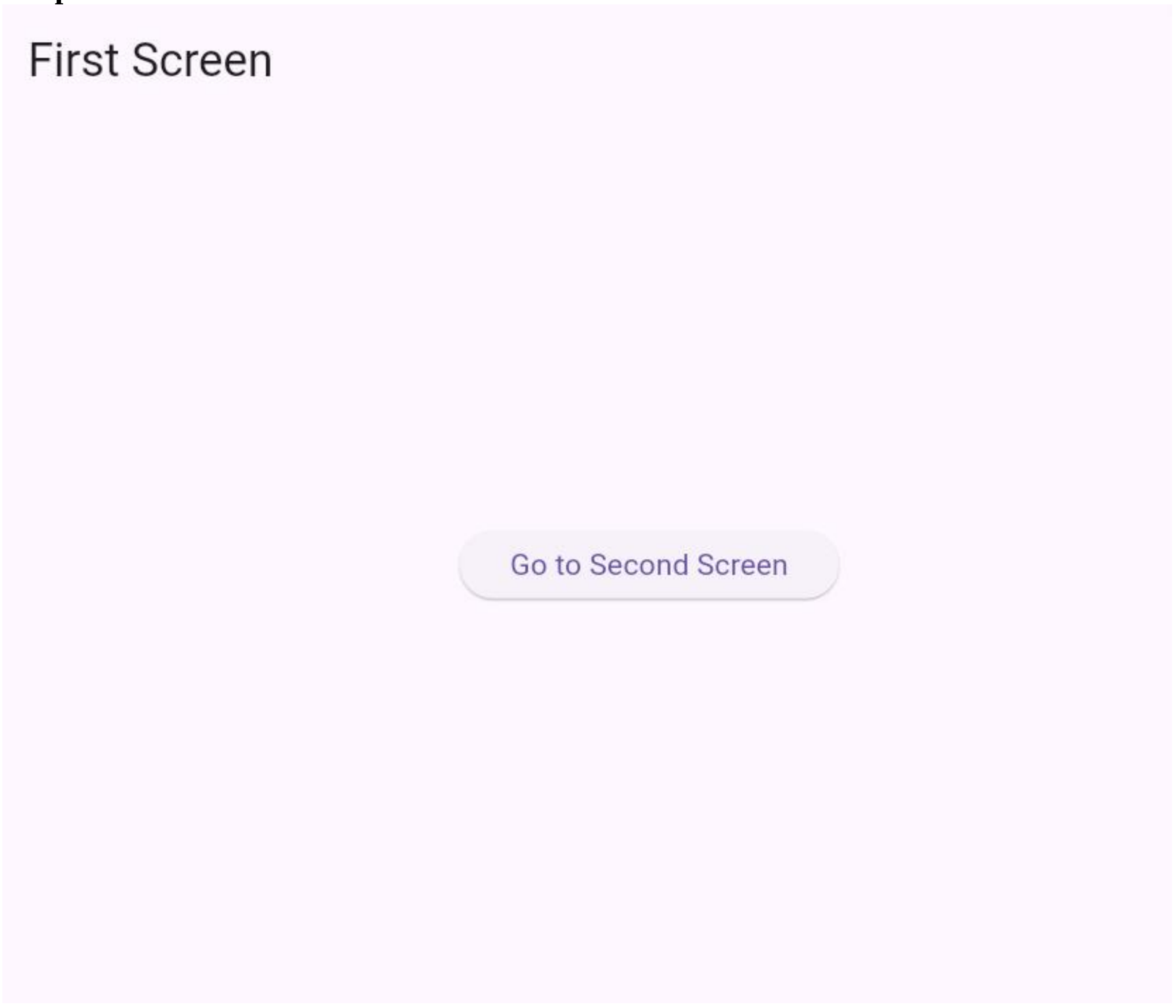


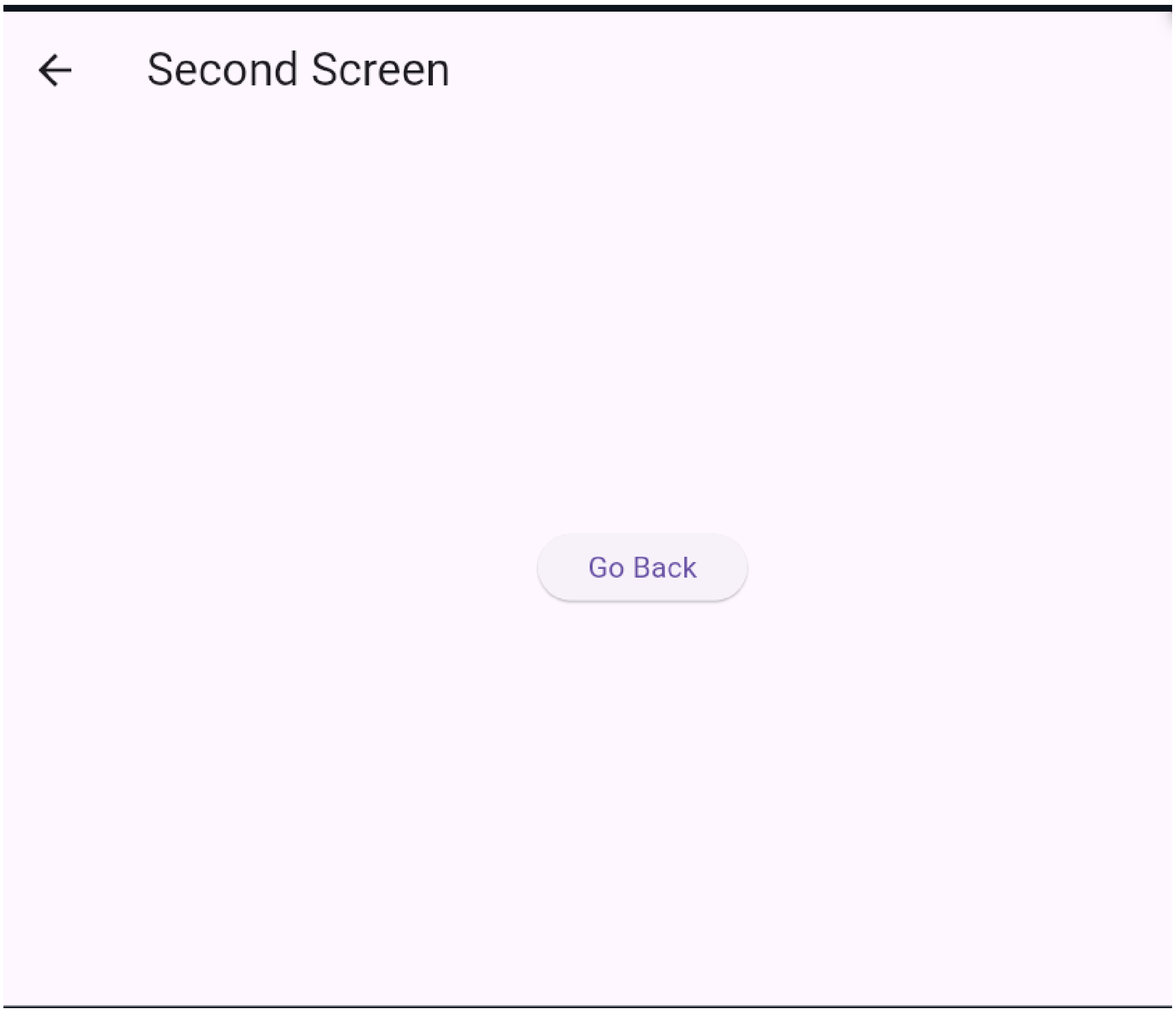


b) Implement navigation with named routes

```
import 'package:flutter/material.dart';
void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Named Routes Demo',
      // Initial route
      initialRoute: '/',
      // Define named routes
      routes: {
        '/': (context) => FirstScreen(),
        '/second': (context) => SecondScreen(),
      },
    );
  }
}
class FirstScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('First Screen')),
      body: Center(
        child: ElevatedButton(
          child: Text('Go to Second Screen'),
          onPressed: () {
            Navigator.pushNamed(context, '/second');
          },
        ),
      ),
    );
  }
}
class SecondScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Second Screen')),
      body: Center(
        child: ElevatedButton(
          child: Text('Go Back'),
          onPressed: () {
            Navigator.pop(context);
          },
        ),
      ),
    );
  }
}
```

Output:



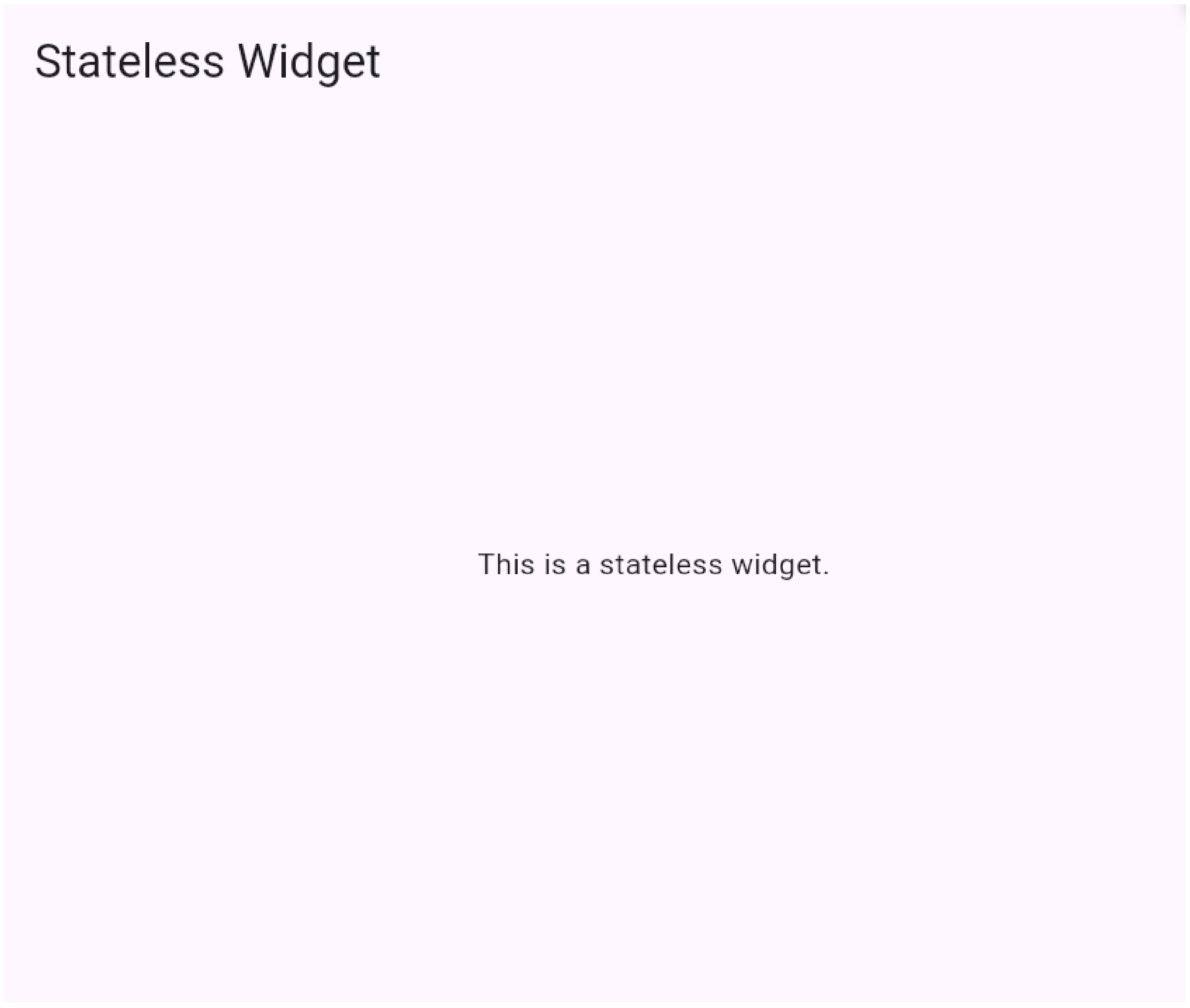


5. Learn about stateful and stateless widgets

Stateful Widget:

```
import 'package:flutter/material.dart';
void main() {
  runApp(MaterialApp(
    home: MyStatelessWidget(),
  ));
}
class MyStatelessWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Stateless Widget')),
      body: Center(
        child: Text('This is a stateless widget.'),
      ),
    );
  }
}
```

Output:

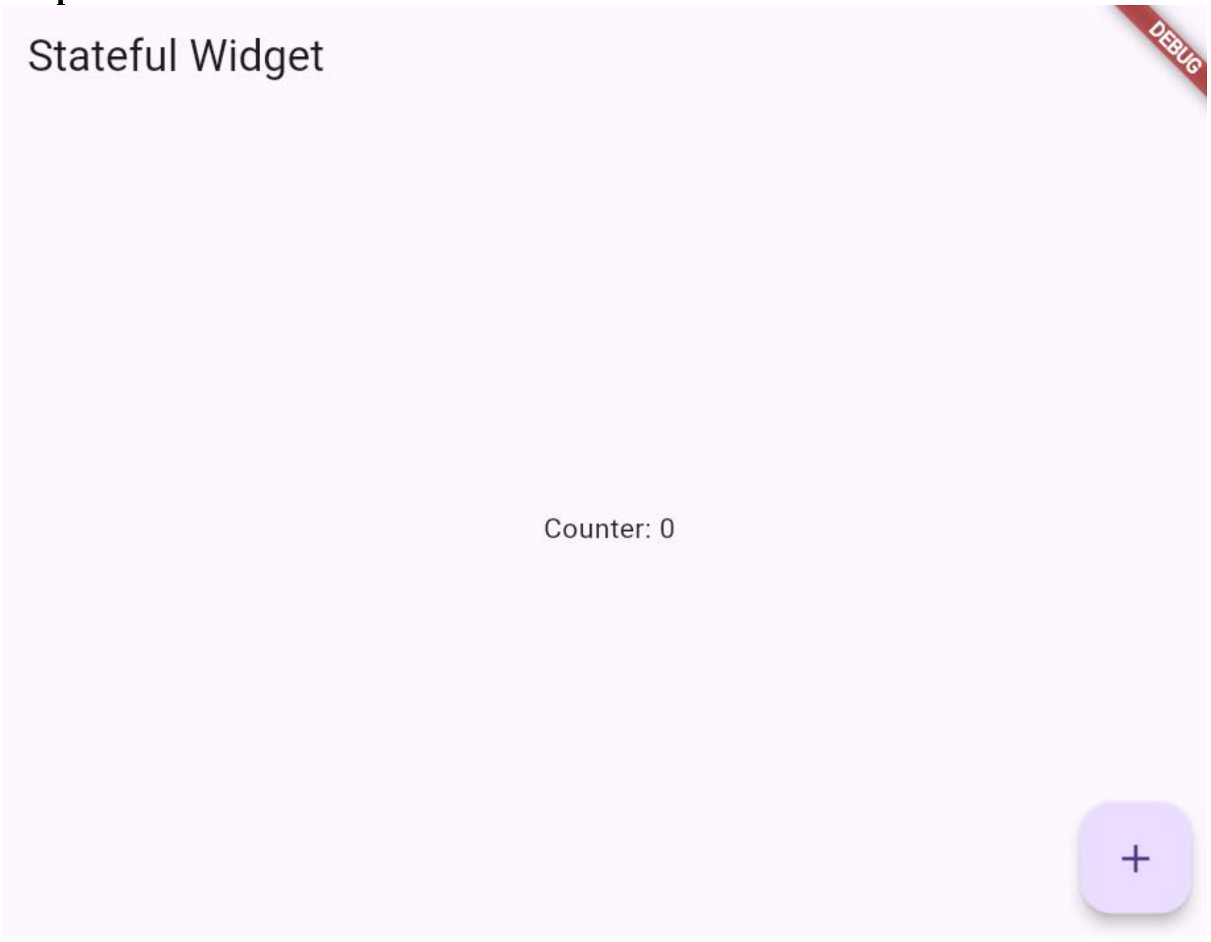

Stateless Widget

This is a stateless widget.

Stateless widget:

```
import 'package:flutter/material.dart';
void main() {
  runApp(MaterialApp(
    home: MyStatefulWidget(), // Try switching with MyStatelessWidget()
  ));
}
class MyStatefulWidget extends StatefulWidget {
  @override
  _MyStatefulWidgetState createState() => _MyStatefulWidgetState();
}
class _MyStatefulWidgetState extends State<MyStatefulWidget> {
  int counter = 0;

  void _incrementCounter() {
    setState(() {
      counter++; // Triggers a UI rebuild
    });
  }
@override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Stateful Widget")),
      body: Center(
        child: Text('Counter: $counter'),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        child: Icon(Icons.add),
      ),
    );
  }
}
```

Output:



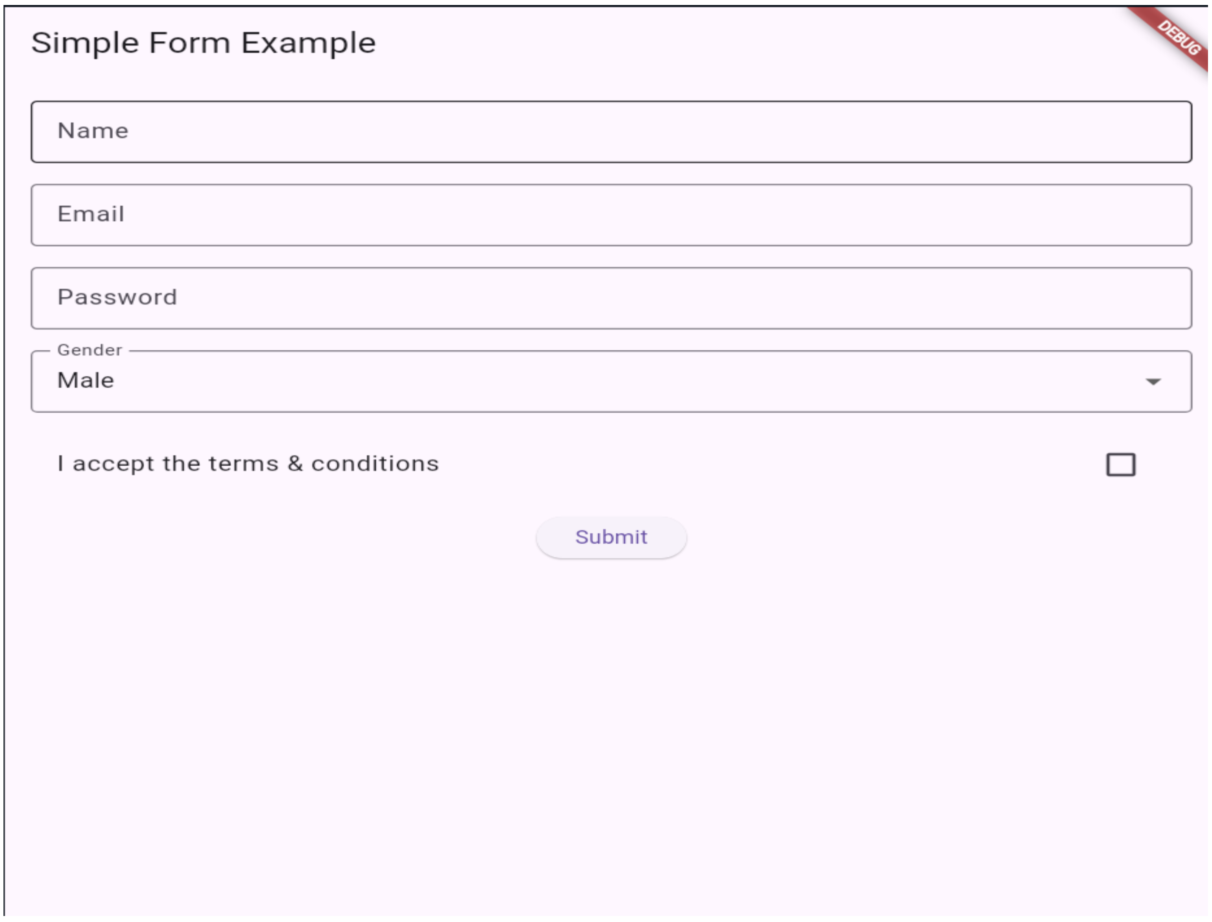
Counter: 0

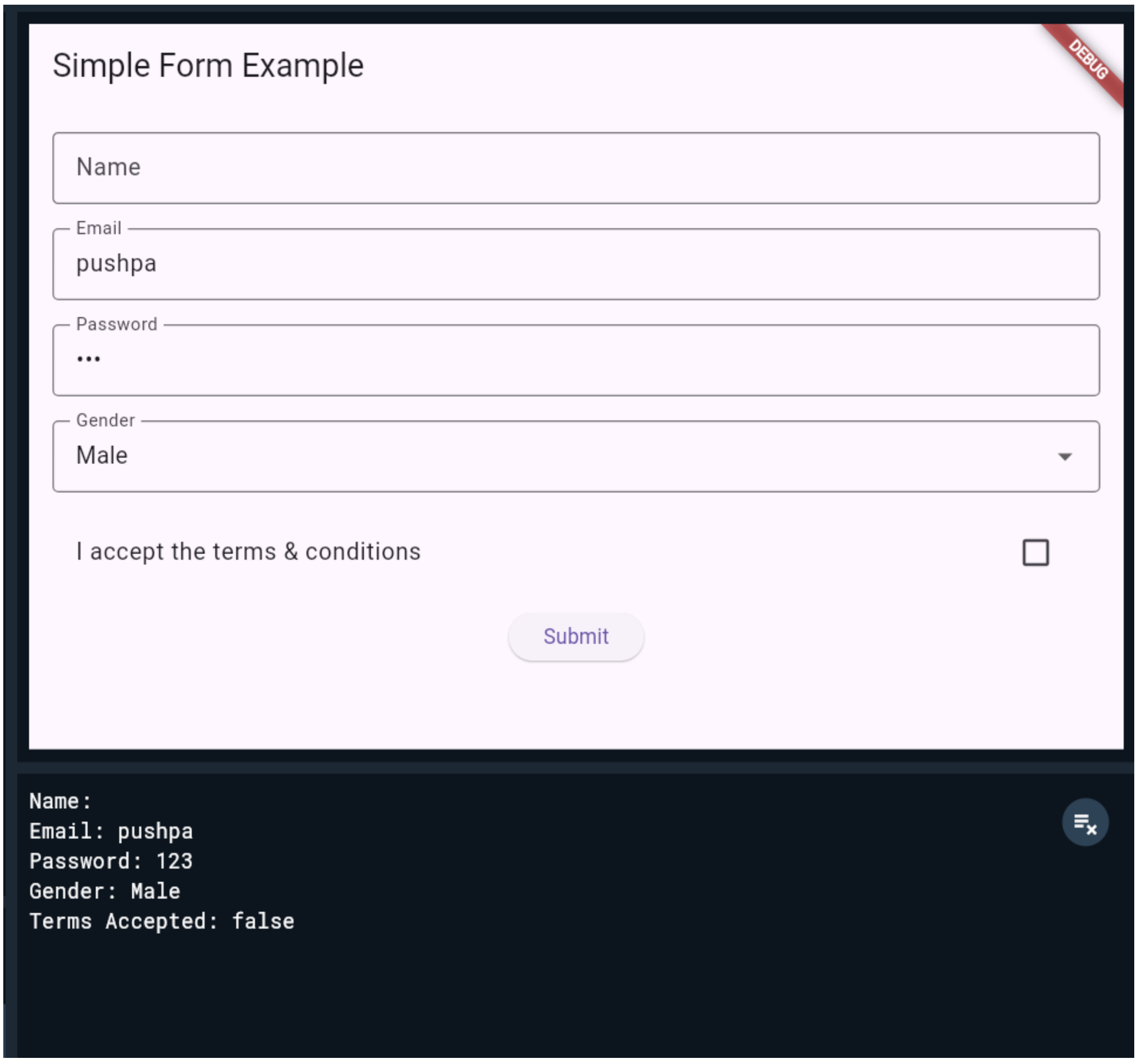
7a) Design a form with various input fields

```
import 'package:flutter/material.dart';
void main() {
  runApp(SimpleFormApp());
}
class SimpleFormApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: "Simple Form",
      home: Scaffold(
        appBar: AppBar(title: Text("Simple Form Example")),
        body: Padding(
          padding: EdgeInsets.all(16),
          child: SimpleForm(),
        ),
      ),
    );
  }
}
class SimpleForm extends StatefulWidget {
  @override
  _SimpleFormState createState() => _SimpleFormState();
}
```

```
class _SimpleFormState extends State<SimpleForm> {
  String name = '';
  String email = '';
  String password = '';
  String selectedGender = 'Male';
  bool termsAccepted = false;
List<String> genders = ['Male', 'Female', 'Other'];
@override
  Widget build(BuildContext context) {
    return SingleChildScrollView(
      child: Column(
        children: [
          // Name Field
          TextField(
            decoration: InputDecoration(
              labelText: "Name",
              border: OutlineInputBorder(),
            ),
            onChanged: (value) {
              name = value;
            },
          ),
          SizedBox(height: 16),

          // Email Field
          TextField(
            decoration: InputDecoration(
              labelText: "Email",
              border: OutlineInputBorder(),
            ),
            keyboardType: TextInputType.emailAddress,
            onChanged: (value) {
              email = value;
            },
          ),
          SizedBox(height: 16),

          // Password Field
          TextField(
            decoration: InputDecoration(
              labelText: "Password",
              border: OutlineInputBorder(),
            ),
            obscureText: true,
            onChanged: (value) {
              password = value;
            },
          ),
          SizedBox(height: 16),

          // Dropdown Field
          DropdownButtonFormField<String>(
            value: selectedGender,
            decoration: InputDecoration(
              labelText: "Gender",
              border: OutlineInputBorder(),
            ),
            items: genders.map((gender) {
              return DropdownMenuItem(
                value: gender,
                child: Text(gender),
              );
            }).toList(),
            onChanged: (value) {
              setState(() {
                selectedGender = value!;
              });
            },
          ),
          SizedBox(height: 16),

          // Checkbox
          CheckboxListTile(
            title: Text("I accept the terms & conditions"),
            value: termsAccepted,
            onChanged: (value) {
              setState(() {
                termsAccepted = value!;
              });
            },
          ),
          SizedBox(height: 16),

          // Submit Button
          ElevatedButton(
            onPressed: () {
              print("Name: $name");
              print("Email: $email");
              print("Password: $password");
              print("Gender: $selectedGender");
              print("Terms Accepted: $termsAccepted");
              ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(content: Text("Form Submitted")),
              );
            },
            child: Text("Submit"),
          ),
        ],
      ),
    );
  }
}
```

Output:





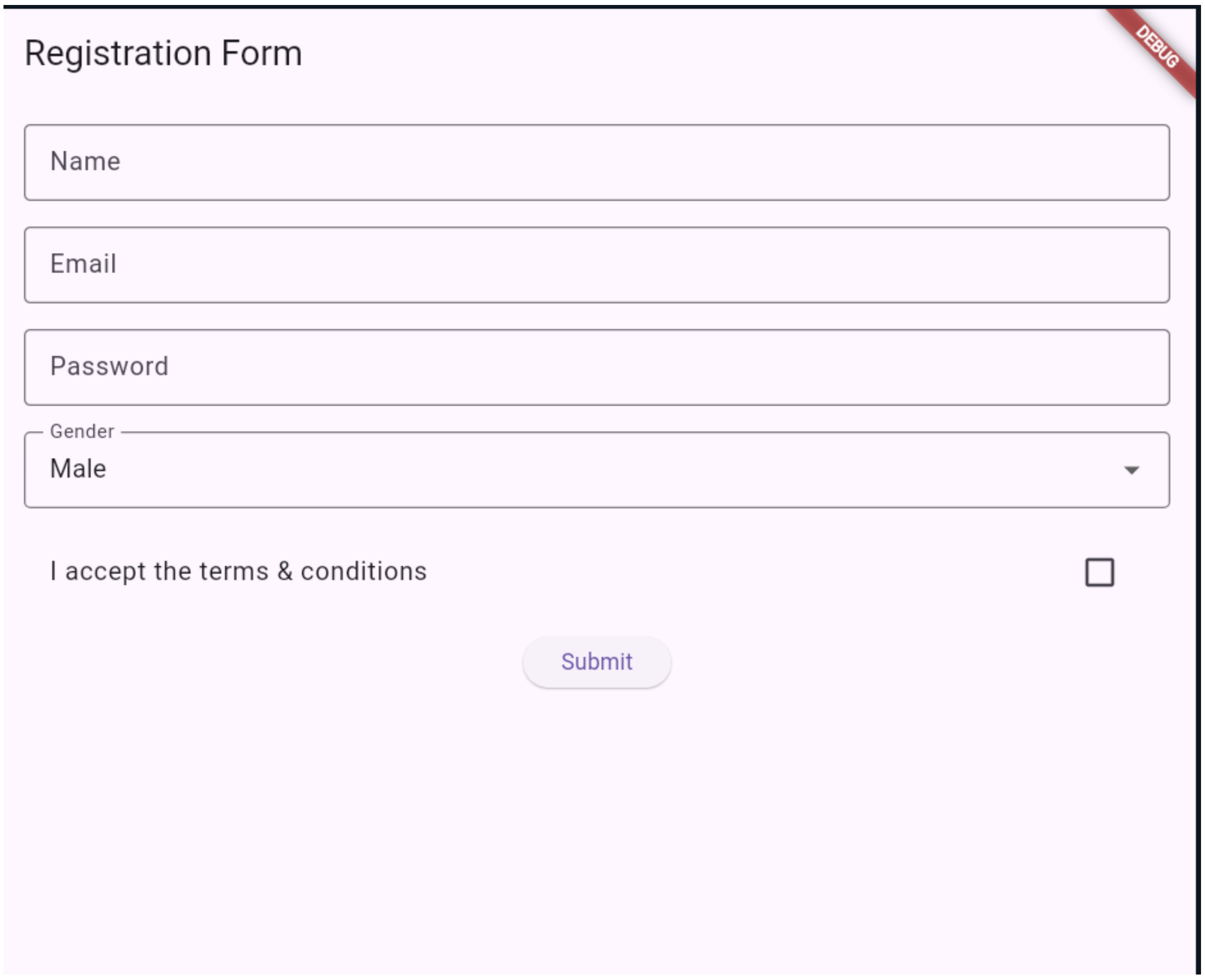
```
Name:
Email: pushpa
Password: 123
Gender: Male
Terms Accepted: false
```

b) Implement form validation and error handling.

```
import 'package:flutter/material.dart';
void main() {
  runApp(MyFormApp());
}
class MyFormApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: "Flutter Form Example",
      home: Scaffold(
        appBar: AppBar(title: Text("Registration Form")),
        body: Padding(
          padding: EdgeInsets.all(16),
          child: RegistrationForm(),
        ),
      ),
    );
  }
}
class RegistrationForm extends StatefulWidget {
  @override
  _RegistrationFormState createState() => _RegistrationFormState();
}
class _RegistrationFormState extends State<RegistrationForm> {
  final _formKey = GlobalKey<FormState>();
  String name = '';
  String email = '';
  String password = '';
  String selectedGender = 'Male';
  bool termsAccepted = false;
List<String> genders = ['Male', 'Female', 'Other'];
@override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey, // Required for validation
      child: SingleChildScrollView(
        child: Column(
          children: [
            // Name Field
            TextFormField(
              decoration: InputDecoration(
                labelText: "Name",
                border: OutlineInputBorder(),
              ),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return "Please enter your name";
                }
                return null;
              },
              onSaved: (value) {
                name = value!;
              },
            ),
            SizedBox(height: 16),
// Email Field
TextFormField(
  decoration: InputDecoration(
    labelText: "Email",
    border: OutlineInputBorder(),
  ),
  keyboardType: TextInputType.emailAddress,
  validator: (value) {
    if (value == null || value.isEmpty) {
      return "Please enter your email";
    }
    if (!RegExp(r'\S+@\S+\.\S+').hasMatch(value)) {
      return "Enter a valid email";
    }
    return null;
  },
  onSaved: (value) {
    email = value!;
  },
),
SizedBox(height: 16),

// Password Field
TextFormField(
  decoration: InputDecoration(
    labelText: "Password",
    border: OutlineInputBorder(),
  ),
  obscureText: true,
  validator: (value) {
    if (value == null || value.length < 6) {
      return "Password must be at least 6 characters";
    }
    return null;
  },
  onSaved: (value) {
    password = value!;
  },
),
SizedBox(height: 16),

// Dropdown Field
DropdownButtonFormField<String>(
  value: selectedGender,
  decoration: InputDecoration(
    labelText: "Gender",
    border: OutlineInputBorder(),
  ),
  items: genders.map((gender) {
    return DropdownMenuItem(
      value: gender,
      child: Text(gender),
    );
  }).toList(),
  onChanged: (value) {
    setState(() {
```

```
                  selectedGender = value!;
                });
              },
            ),
            SizedBox(height: 16),

            // Checkbox
            CheckboxListTile(
              title: Text("I accept the terms & conditions"),
              value: termsAccepted,
              onChanged: (value) {
                setState(() {
                  termsAccepted = value!;
                });
              },
            ),
            SizedBox(height: 16),
            // Submit Button
            ElevatedButton(
              onPressed: () {
                if (_formKey.currentState!.validate()) {
                  if (!termsAccepted) {
                    ScaffoldMessenger.of(context).showSnackBar(
                      SnackBar(content: Text("Please accept the terms")),
                    );
                    return;
                  }
                  _formKey.currentState!.save();
                  ScaffoldMessenger.of(context).showSnackBar(
                    SnackBar(content: Text("Form Submitted!")),
                  );
                  print("Name: $name");
                  print("Email: $email");
                  print("Password: $password");
                  print("Gender: $selectedGender");
                }
              },
              child: Text("Submit"),
            ),
          ],
        ),
      ),
    );
  }
}
```

Output:


Registration Form

Name

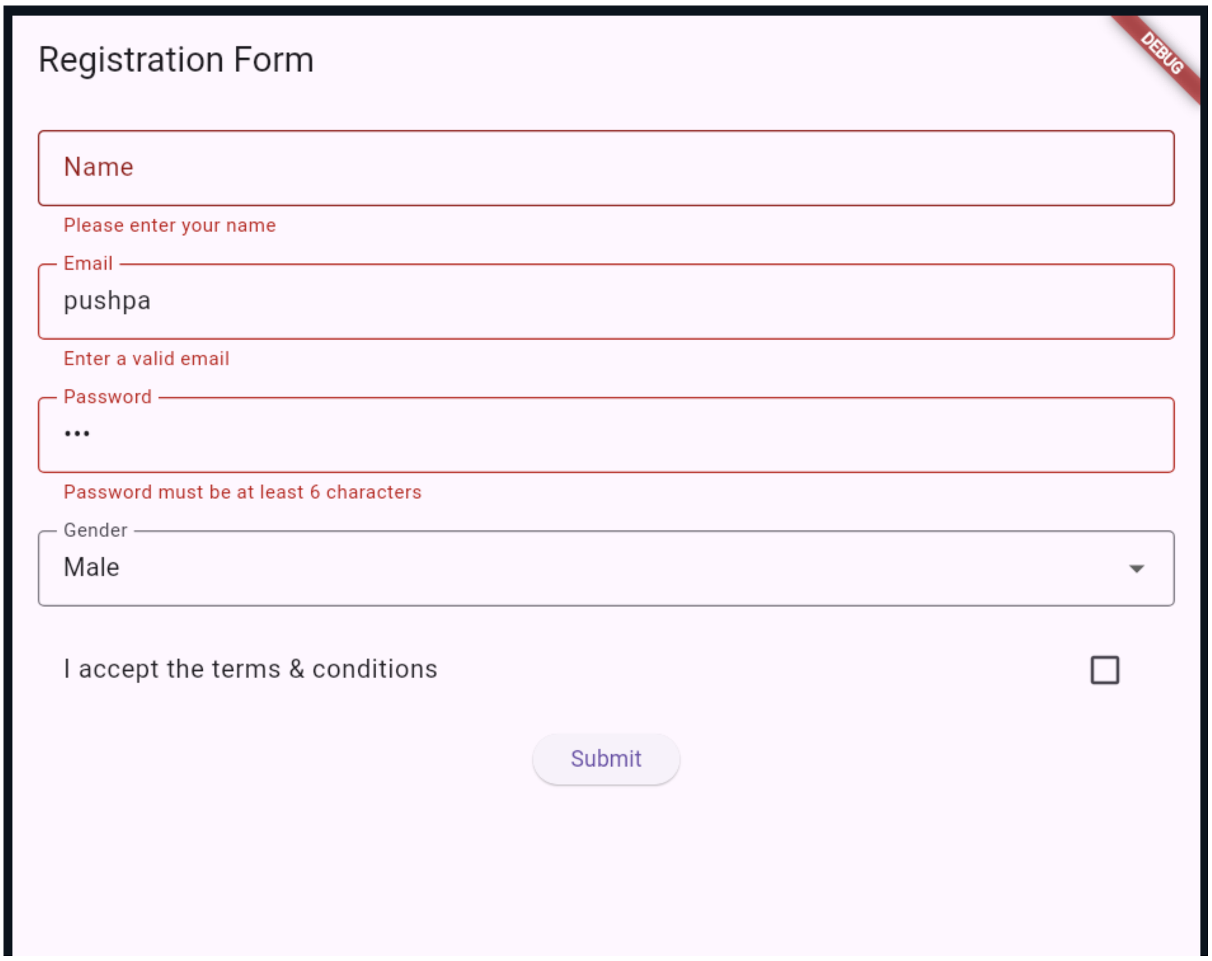
Email

Password

Gender

Male

I accept the terms & conditions


Registration Form

Name

Please enter your name

Email

pushpa

Enter a valid email

Password

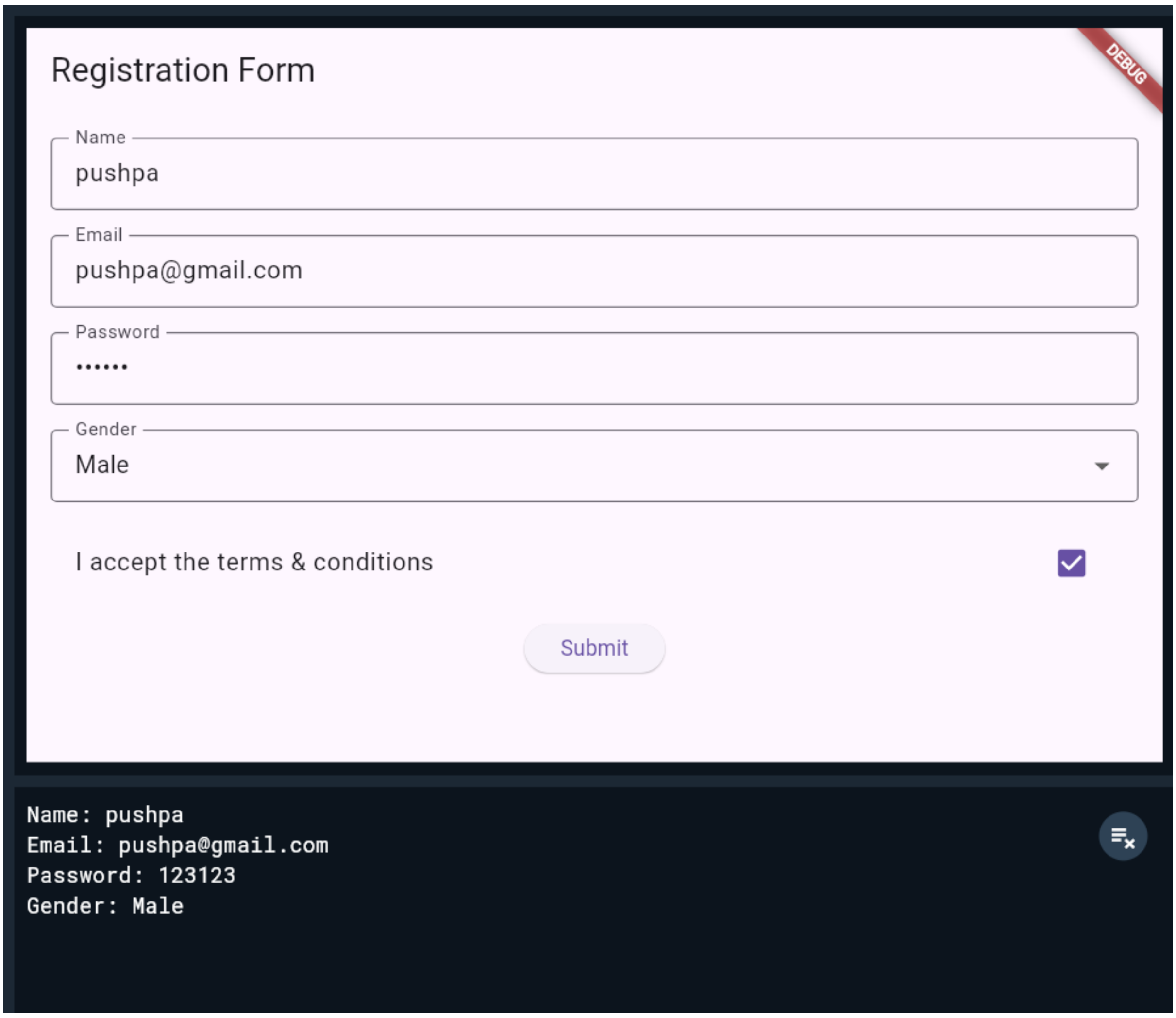
•••

Password must be at least 6 characters

Gender

Male

I accept the terms & conditions


Registration Form

Name: pushpa

Email: pushpa@gmail.com

Password: ••••••

Gender: Male

I accept the terms & conditions ✓

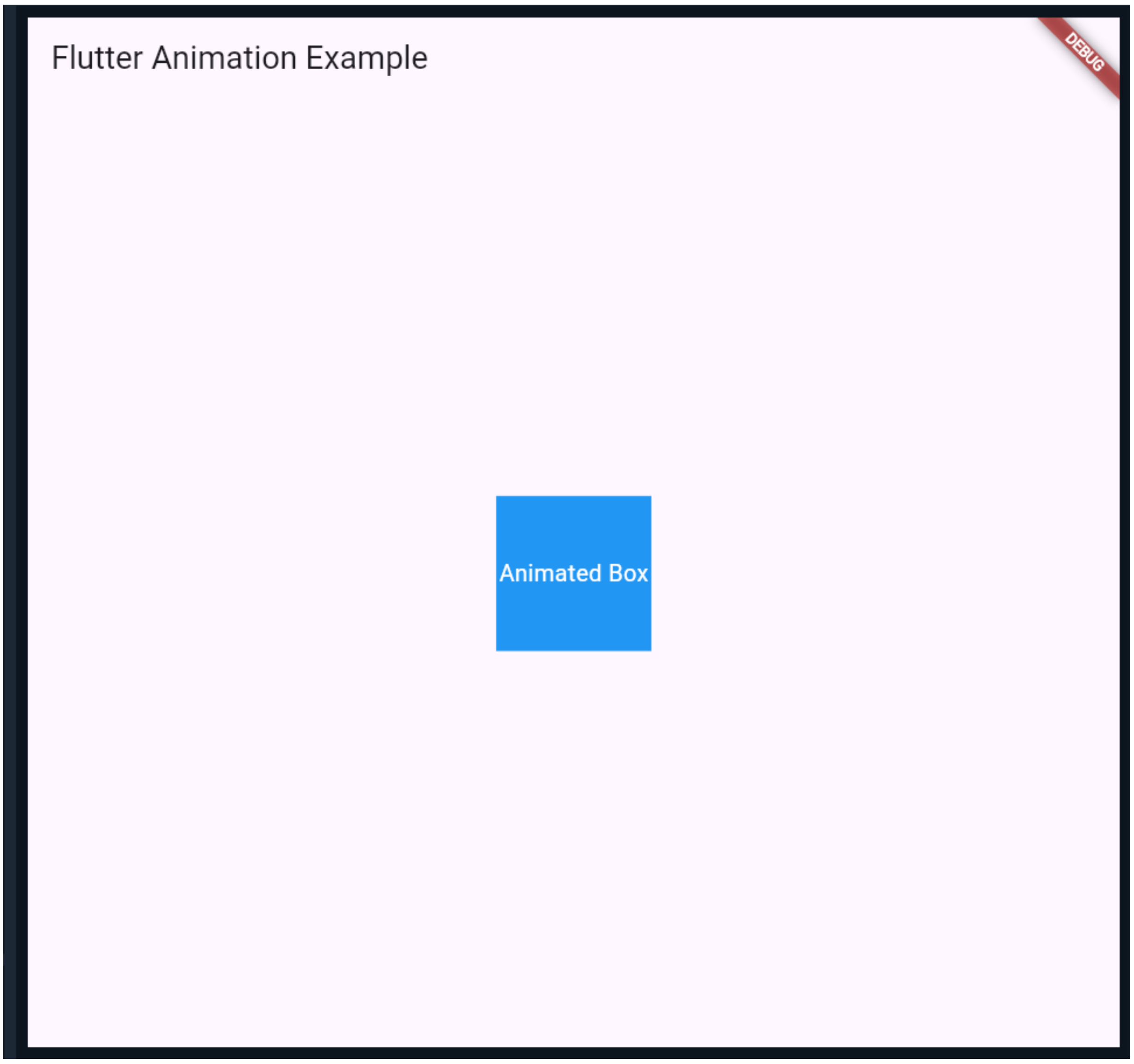
```
Name: pushpa
Email: pushpa@gmail.com
Password: 123123
Gender: Male
```

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

```
import 'package:flutter/material.dart';
void main() {
  runApp(AnimationExampleApp());
}
class AnimationExampleApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: "Animation Example",
      home: Scaffold(
        appBar: AppBar(title: Text("Flutter Animation Example")),
        body: Center(
          child: AnimatedBox(),
        ),
      ),
    );
  }
}
class AnimatedBox extends StatefulWidget {
  @override
  _AnimatedBoxState createState() => _AnimatedBoxState();
}
class _AnimatedBoxState extends State<AnimatedBox>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller;
  late Animation<double> _sizeAnimation;
@override
```

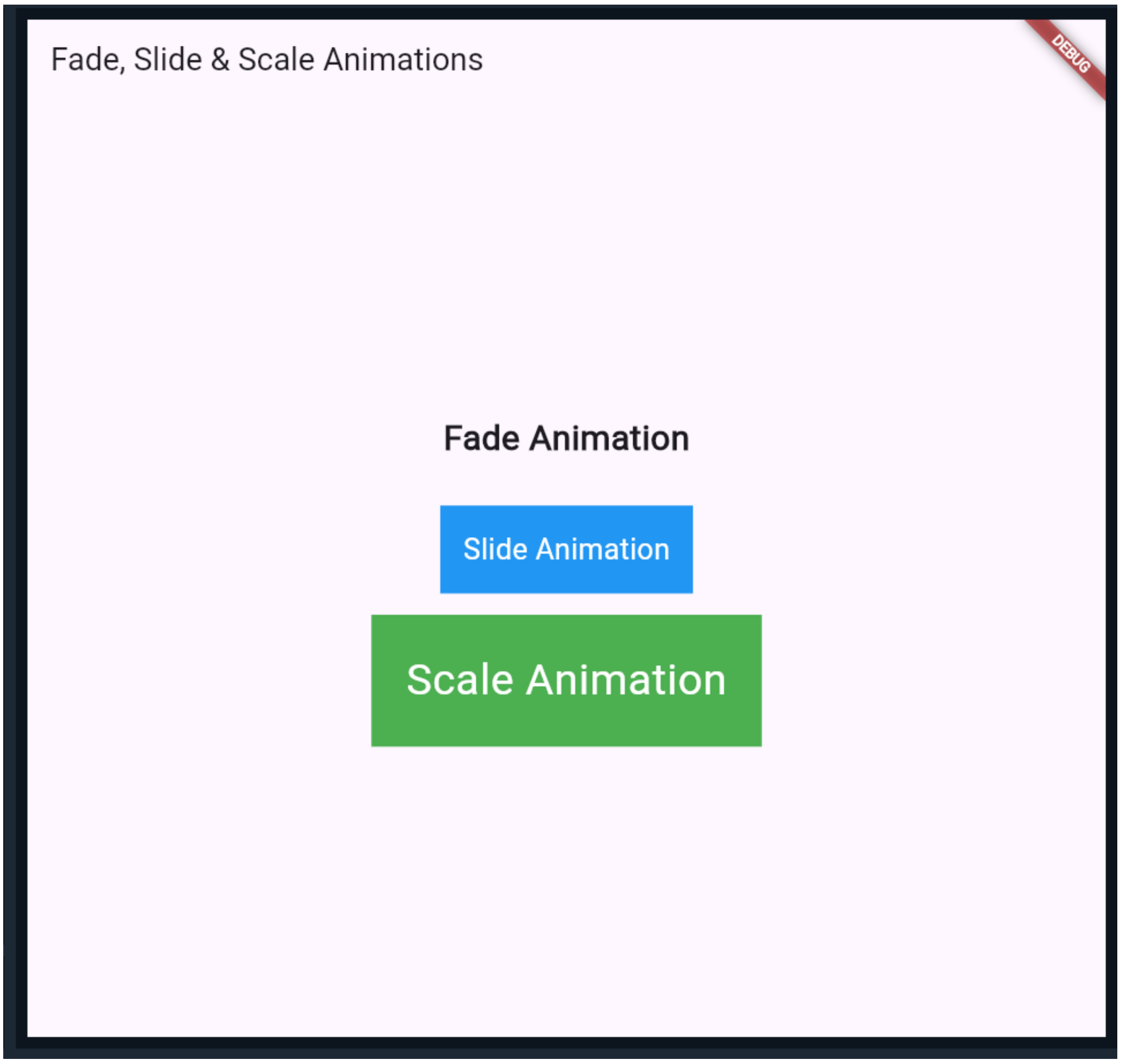
```
  void initState() {
    super.initState();
// Animation controller
    _controller = AnimationController(
      vsync: this,
      duration: Duration(seconds: 2),
    );
// Tween for size change (from 100 to 200)
    _sizeAnimation = Tween<double>(begin: 100, end: 200).animate(
      CurvedAnimation(
        parent: _controller,
        curve: Curves.easeInOut,
      ),
    );
// Loop animation back and forth
    _controller.repeat(reverse: true);
  }
@override
  void dispose() {
    _controller.dispose(); // Clean up
    super.dispose();
  }
@override
  Widget build(BuildContext context) {
    return AnimatedBuilder(
      animation: _sizeAnimation,
      builder: (context, child) {
        return Container(
          width: _sizeAnimation.value,
          height: _sizeAnimation.value,
          color: Colors.blue,
          child: Center(
            child: Text(
              "Animated Box",
              style: TextStyle(color: Colors.white, fontSize: 16),
            ),
          ),
        );
      },
    );
  }
}
```

Output:



b) Experiment with different types of animations (fade, slide, etc.)

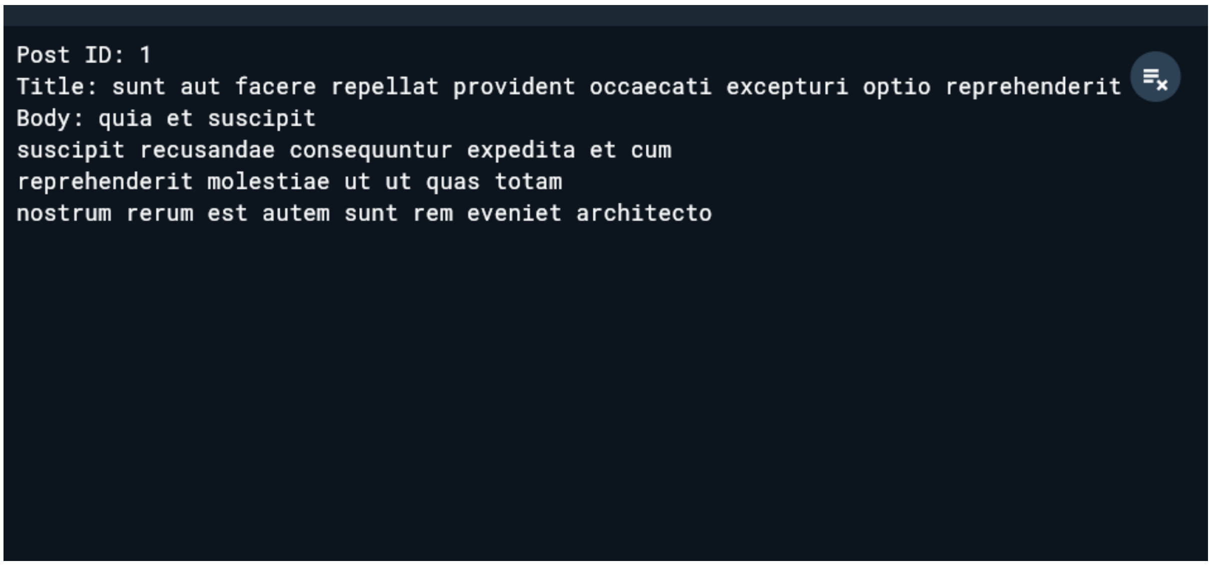
```
import 'package:flutter/material.dart';
void main() {
  runApp(MultiAnimationApp());
}
class MultiAnimationApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: "Multiple Animations",
      home: Scaffold(
        appBar: AppBar(title: Text("Fade, Slide & Scale Animations")),
        body: MultiAnimationDemo(),
      ),
    );
  }
}
class MultiAnimationDemo extends StatefulWidget {
  @override
  _MultiAnimationDemoState createState() => _MultiAnimationDemoState();
}
class _MultiAnimationDemoState extends State<MultiAnimationDemo>
    with TickerProviderStateMixin {
  late AnimationController _fadeController;
  late AnimationController _slideController;
  late AnimationController _scaleController;
  late Animation<double> _fadeAnimation;
  late Animation<Offset> _slideAnimation;
  late Animation<double> _scaleAnimation;
@override
  void initState() {
    super.initState();
 // Fade Animation
    _fadeController =
        AnimationController(vsync: this, duration: Duration(seconds: 2));
    _fadeAnimation = Tween<double>(begin: 0, end: 1).animate(_fadeController);
 // Slide Animation
    _slideController =
        AnimationController(vsync: this, duration: Duration(seconds: 2));
    _slideAnimation =
        Tween<Offset>(begin: Offset(1, 0), end: Offset(0, 0)).animate(
            CurvedAnimation(parent: _slideController, curve: Curves.easeInOut));
 // Scale Animation
    _scaleController =
        AnimationController(vsync: this, duration: Duration(seconds: 2));
    _scaleAnimation =
        Tween<double>(begin: 0.5, end: 1.5).animate(CurvedAnimation(
      parent: _scaleController,
      curve: Curves.easeInOut,
    ));
// Start animations with delays
    _fadeController.forward();
    Future.delayed(Duration(milliseconds: 500), () => _slideController.forward());
    Future.delayed(Duration(milliseconds: 1000), () => _scaleController.forward());
  }
@override
  void dispose() {
    _fadeController.dispose();
    _slideController.dispose();
    _scaleController.dispose();
    super.dispose();
  }
@override
  Widget build(BuildContext context) {
    return Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          // Fade Animation
          FadeTransition(
            opacity: _fadeAnimation,
            child: Text(
              "Fade Animation",
              style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
            ),
          ),
          SizedBox(height: 30),
         // Slide Animation
          SlideTransition(
            position: _slideAnimation,
            child: Container(
              padding: EdgeInsets.all(16),
              color: Colors.blue,
              child: Text(
                "Slide Animation",
                style: TextStyle(color: Colors.white, fontSize: 20),
              ),
            ),
          ),
          SizedBox(height: 30),
          // Scale Animation
          ScaleTransition(
            scale: _scaleAnimation,
            child: Container(
              padding: EdgeInsets.all(16),
              color: Colors.green,
              child: Text(
                "Scale Animation",
                style: TextStyle(color: Colors.white, fontSize: 20),
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```
Output:



9a) Fetching Data From a Rest API

```
import 'dart:convert'; // For JSON decoding
 import 'package:http/http.dart' as http;
void main() async {
 // API endpoint
 final url =
Uri.parse('https://jsonplaceholder.typicode.com/posts/1');
 try {
 // Send GET request
 final response = await http.get(url);
 // Check status
 if (response.statusCode == 200) {
 // Decode JSON
 final data = jsonDecode(response.body);
 // Print result
 print('Post ID: ${data['id']}');
 print('Title: ${data['title']}');
 print('Body: ${data['body']}');
 } else {
 print('Request failed with status:  ${response.statusCode}');
 }
 } catch (e) {
 print('Error occurred: $e');
 }
}
```

Output:



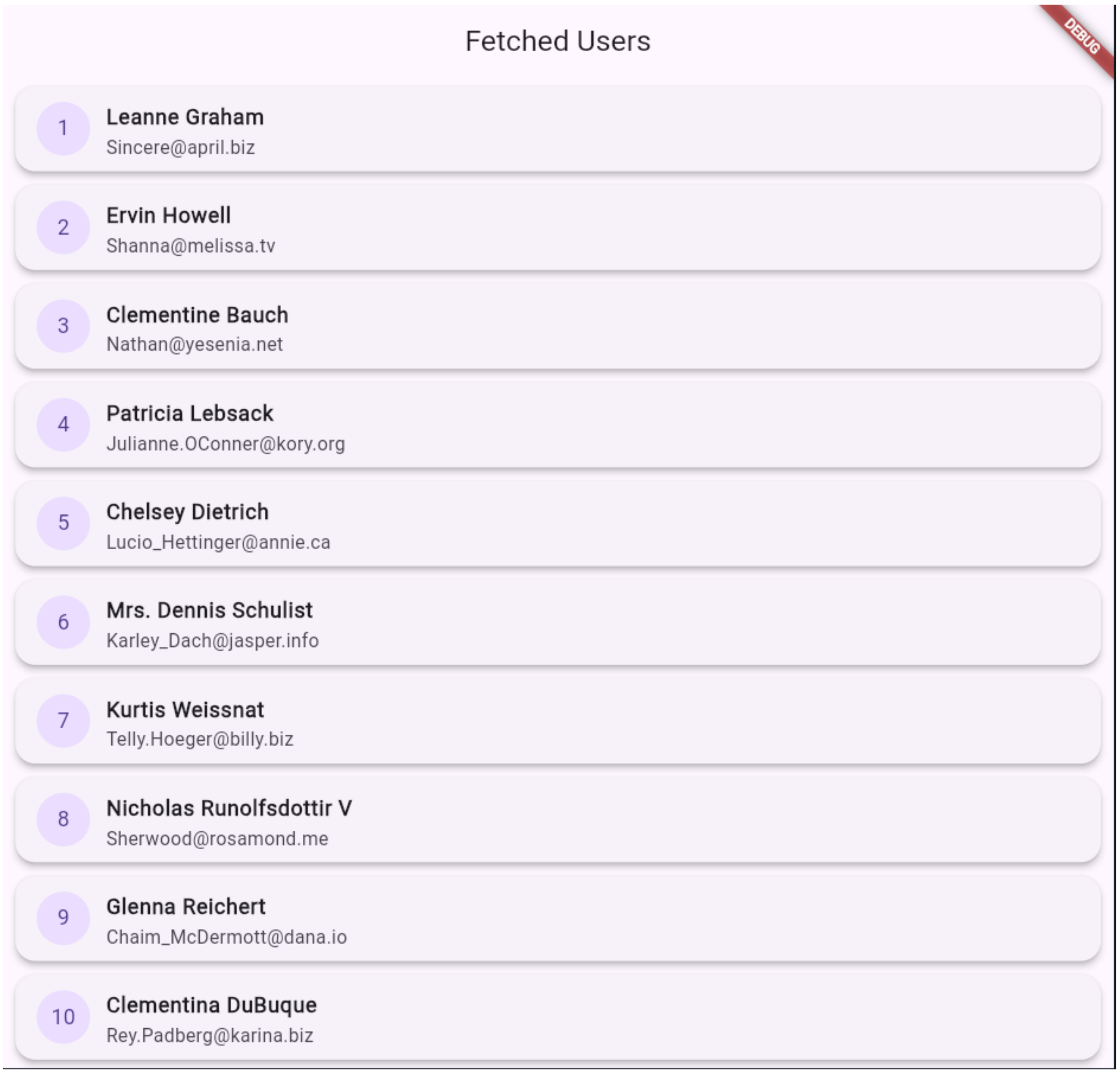
```
Post ID: 1
Title: sunt aut facere repellat provident occaecati excepturi optio reprehenderit
Body: quia et suscipit
suscipit recusandae consequuntur expedita et cum
reprehenderit molestiae ut ut quas totam
nostrum rerum est autem sunt rem eveniet architecto
```

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

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

```
class User {
  final int id;
  final String name;
  final String email;
User({required this.id, required this.name, required this.email});
factory User.fromJson(Map<String, dynamic> json) {
    return User(id: json['id'], name: json['name'], email: json['email']);
  }
}
// Fetch function
Future<List<User>> fetchUsers() async {
  final response = await http.get(
    Uri.parse('https://jsonplaceholder.typicode.com/users'),
  );
if (response.statusCode == 200) {
    List<dynamic> data = json.decode(response.body);
    return data.map((json) => User.fromJson(json)).toList();
  } else {
    throw Exception('Failed to load data');
  }
}
void main() {
  runApp(const MyApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Fetch Data Example',
      theme: ThemeData(primarySwatch: Colors.blue),
      home: const UserScreen(),
    );
  }
}
// UI Screen
class UserScreen extends StatefulWidget {
  const UserScreen({super.key});
@override
  State<UserScreen> createState() => _UserScreenState();
}
class _UserScreenState extends State<UserScreen> {
  late Future<List<User>> futureUsers;
@override
  void initState() {
    super.initState();
    futureUsers = fetchUsers();
  }
@override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Fetched Users'), centerTitle: true),
      body: FutureBuilder<List<User>>(
        future: futureUsers,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const Center(child: CircularProgressIndicator());
          } else if (snapshot.hasError) {
            return Center(child: Text("Error:${snapshot.error}"));
          } else if (!snapshot.hasData || snapshot.data!.isEmpty) {
            return const Center(child: Text('No data found'));
          } else {
            final users = snapshot.data!;
            return ListView.builder(
              itemCount: users.length,
             itemBuilder: (context, index) {
                final user = users[index];
                return Card(
                  margin: const EdgeInsets.symmetric(
                    horizontal: 10,
                    vertical: 5,
                  ),
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(15),
                  ),
                  elevation: 4,
                  child: ListTile(
                    leading: CircleAvatar(child: Text(user.id.toString())),
                    title: Text(
                      user.name,
                      style: const TextStyle(fontWeight: FontWeight.bold),
                    ),
                    subtitle: Text(user.email),
                  ),
                );
              },
            );
          }
        },
      ),
    );
  }
}
```

Output:



```
Fetched Users

1  Leanne Graham
   Sincere@april.biz
2  Ervin Howell
   Shanna@melissa.tv
3  Clementine Bauch
   Nathan@yesenia.net
4  Patricia Lebsack
   Julianne.OConner@kory.org
5  Chelsey Dietrich
   Lucio_Hettinger@annie.ca
6  Mrs. Dennis Schulist
   Karley_Dach@jasper.info
7  Kurtis Weissnat
   Telly.Hoeger@billy.biz
8  Nicholas Runolfsdottir V
   Sherwood@rosamond.me
9  Glenna Reichert
   Chaim_McDermott@dana.io
10 Clementina DuBuque
   Rey.Padberg@karina.biz
```

10 a) Write unit tests for UI components.

Unit testing is a fundamental software testing technique focused on verifying the smallest parts of an application the "units" such as functions, methods, or UI components, work correctly in isolation. When applied to UI components, unit testing ensures that each component behaves as expected before it is integrated into the larger application.

Types of UI Components (Widgets) in Flutter

- Basic Widgets: Simple UI elements like Text, Image, Icon, and Container.
- Layout Widgets: Arrange other widgets on the screen, e.g., Row, Column, Stack.
- Material Components: Widgets that follow Google's Material Design like AppBar, Elevated Button, FloatingActionButton.
- Cupertino Widgets: iOS-style widgets that mimic native iOS design, e.g., Cupertino Button, Cupertino Navigation Bar.
- Interactive Widgets: Widgets that handle user interaction such as buttons, sliders, text fields.

Sample Flutter Program: Counter Button (main.dart)

```
import 'package:flutter/material.dart';
void main() {
runApp(CounterApp());
}
```

```
class CounterApp extends StatelessWidget {
@override
Widget build(BuildContext context) {
return MaterialApp(
home: Scaffold(
appBar: AppBar(title: Text('Counter Button Example')),
body: Center(child: CounterButton()),
),
);
}
}
class CounterButton extends StatefulWidget {
@override
_CounterButtonState createState() => _CounterButtonState();
}
class _CounterButtonState extends State<CounterButton> {
int _count = 0;
void _increment() {
setState(() {
_count++;
});
}
@override
Widget build(BuildContext context) {
return Column(
mainAxisAlignment: MainAxisAlignment.center,
children: [
Text('Button pressed $_count times'),
SizedBox(height: 20),
ElevatedButton(
onPressed: _increment,
child: Text('Increment'),
),
],
);
}
}
```

Unit Test code:

```
import 'package:flutter_test/flutter_test.dart';
import 'package:flutter/material.dart';
import 'package:your_app/main.dart'; // Adjust this import as per your file structure
void main() {
testWidgets('Counter increments when button is tapped', (WidgetTester tester) async {
await tester.pumpWidget(MaterialApp(home: CounterButton()));
// Verify initial count text
expect(find.text('Button pressed 0 times'), findsOneWidget);
expect(find.text('Button pressed 1 times'), findsNothing);
// Tap the increment button once
await tester.tap(find.byType(ElevatedButton));
await tester.pump();
// Verify the count incremented to 1
expect(find.text('Button pressed 1 times'), findsOneWidget);
expect(find.text('Button pressed 0 times'), findsNothing);
});
}
```

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

Debugging: Finding and fixing errors (bugs) in code.
Bug: Bugs cause apps to crash, freeze, or behave unexpectedly.
Debugging tools help developers:

- Detect problems quickly
- Understand why they occur
- Fix them efficiently

Debugging Tools in Flutter:
Flutter provides multiple tools for debugging:

- Flutter DevTools – performance, widget inspector, memory.
- Debug Console (Logs) – shows runtime errors &amp; print statements.
- Hot Reload &amp; Hot Restart – quickly test fixes without full restart.
- Breakpoints &amp; Step Debugging – pause code and check variable values.
- Error Widgets – shows red error screen for UI issues.

Flutter DevTools: Web-based tool for debugging Flutter apps.
Features:
Widget Inspector – see widget tree &amp; UI structure.
Performance – check rendering speed &amp; FPS.
Memory – monitor leaks.
CPU Profiler – track heavy code.
Eg: //Command to open:flutter pub global activate devtoolsflutter run –debug

Debug Console (Logs):
Shows print() outputs, errors, and exceptions. Helps track code execution.
Example Code: print(&quot;App started&quot;);print(&quot;User tapped button&quot;);

Hot Reload &amp; Hot Restart:
Hot Reload: Updates code changes instantly without restarting app.
Hot Restart: Restarts app but keeps code changes.
Example: Changing button color → Hot Reload → see changes immediately.

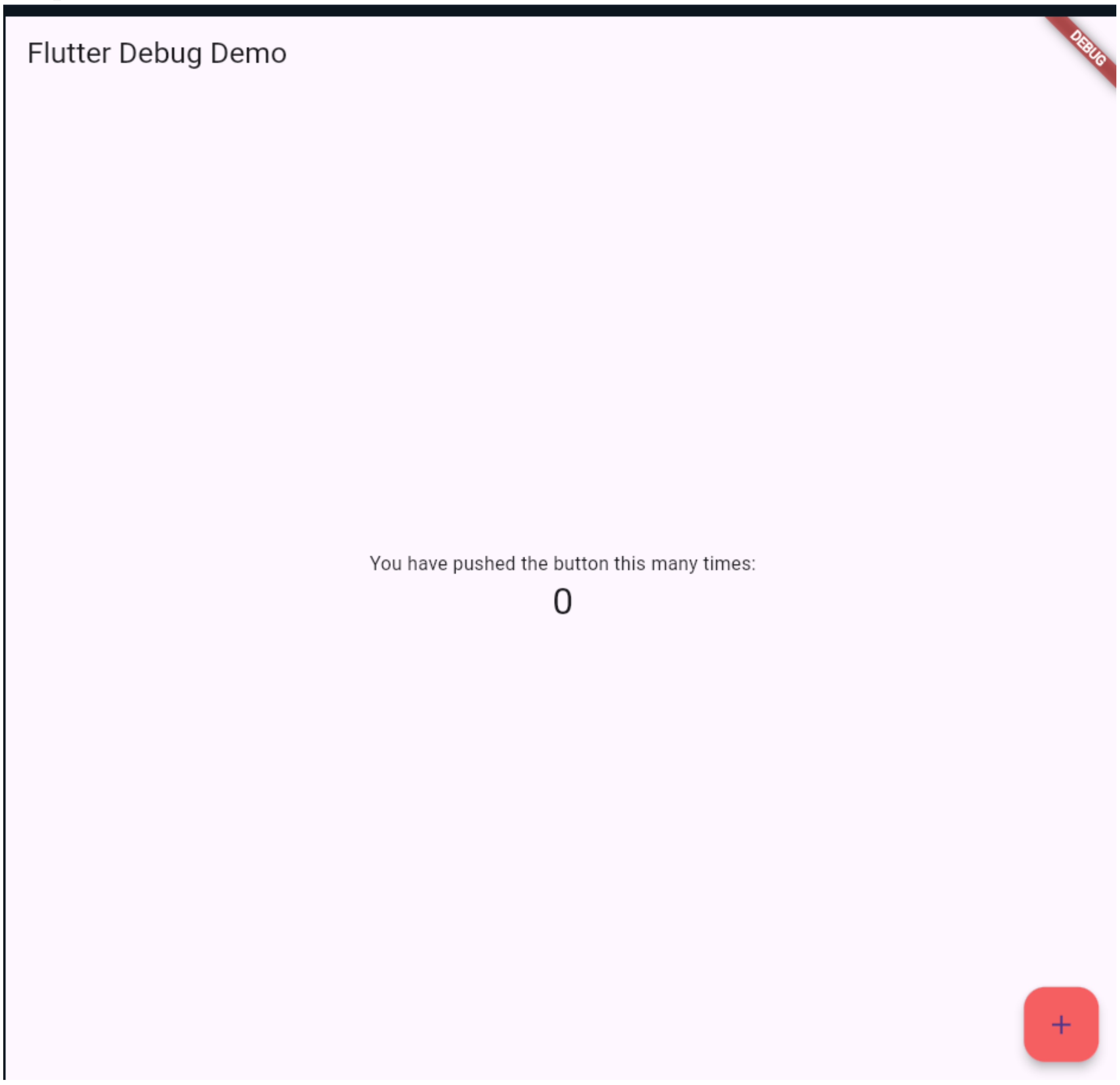
Breakpoints & Step Debugging:
Add a breakpoint in code → app pauses at that line.
Can check: Variable values, Function calls, Program flow
Eg: int sum(int a, int b)

```
{
return a + b; // breakpoint here
}
```

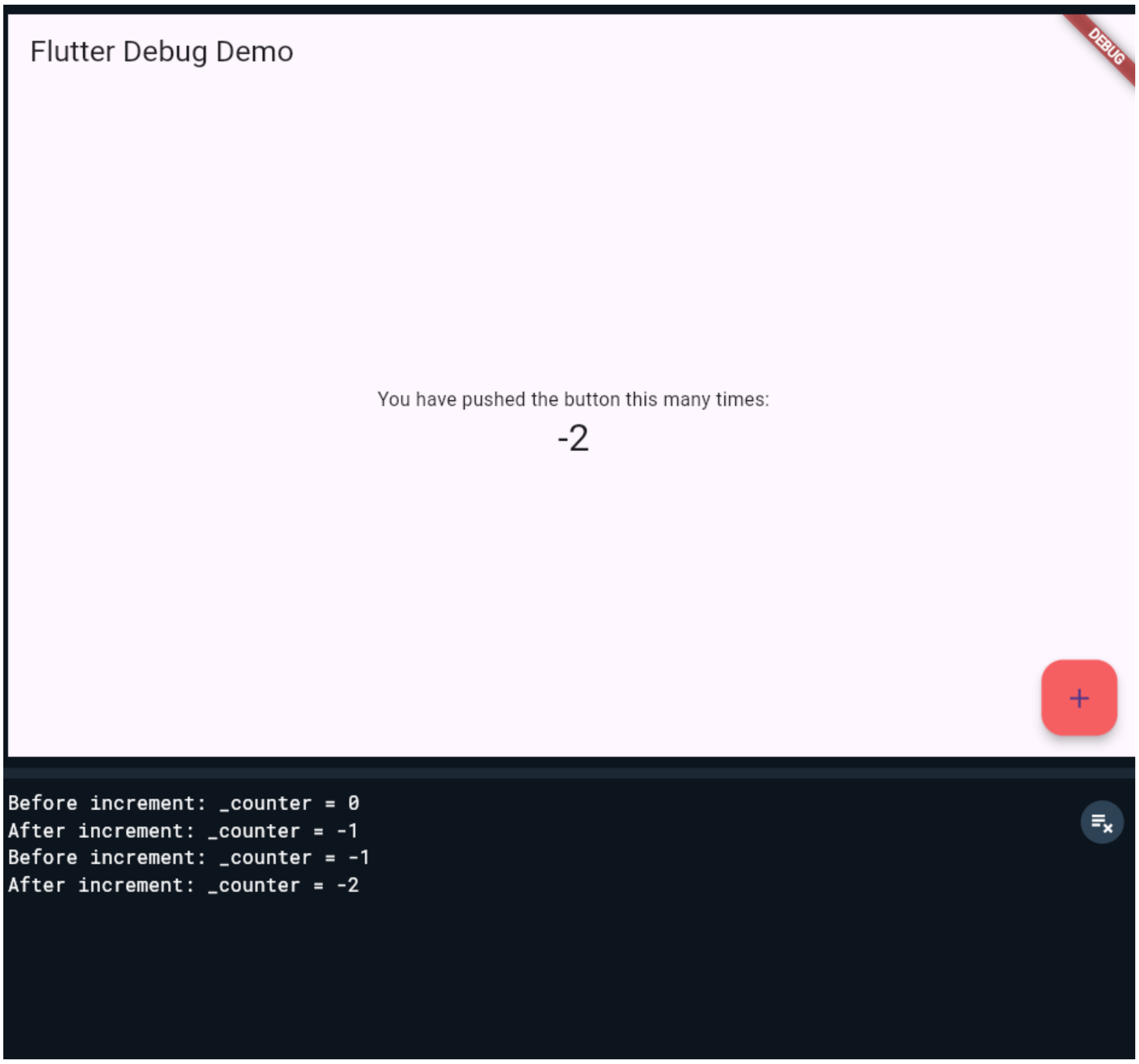
Error Widgets (Red Screen of Death):
When UI fails, Flutter shows a red error screen.
Example: Layout errors, null widget issues.

```
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: 'Flutter Counter Debug',
theme: ThemeData(
primarySwatch: Colors.blue,
),
home: const MyHomePage(title: 'Flutter Debug Demo'),
);
}
}
class MyHomePage extends StatefulWidget {
const MyHomePage({super.key, required this.title});
final String title;
@override
State<MyHomePage> createState() => _MyHomePageState();
}
class _MyHomePageState extends State<MyHomePage> {
int _counter = 0;
void _incrementCounter() {
print('Before increment: _counter = $_counter'); // Logging
setState(() {
// Simulate a bug: accidentally decrementing instead of incrementing
_counter--; // Breakpoint here
});
print('After increment: _counter = $_counter'); // Logging
}
@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: Text(widget.title),
),
body: Center(
child: Column(
mainAxisAlignment: MainAxisAlignment.center,
children: <Widget>[

const Text('You have pushed the button this many times:',
),
Text('$_counter',
style: Theme.of(context).textTheme.headlineMedium,
),
],
),
),
floatingActionButton: FloatingActionButton(
onPressed: _incrementCounter,
tooltip: 'Increment',
backgroundColor: const Color(0xFFF65F62),
child: const Icon(Icons.add),
),
);
}
}
```

Output:



```
You have pushed the button this many times:
0
```



```
Before increment: _counter = 0
After increment: _counter = -1
Before increment: _counter = -1
After increment: _counter = -2
```