Laboratory Manual

For the course of

TINKERING LAB

Branch: CSE /CSM/CAI/CSD/AIML

VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE
(AUTONOMOUS)
Approved by AICTE New Delhi & Affiliated to JNTUK Kakinada
Accredited by NAAC 'A+' and NBA | ISO 9001 : 2015
Vadlamudi - 522 213, Guntur District

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

USER INTERFACE DESIGN USING FLUTTER

For B.Tech
CSE/CSM/CAI/CSD/AIML

III Year I Semester(R-23)

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

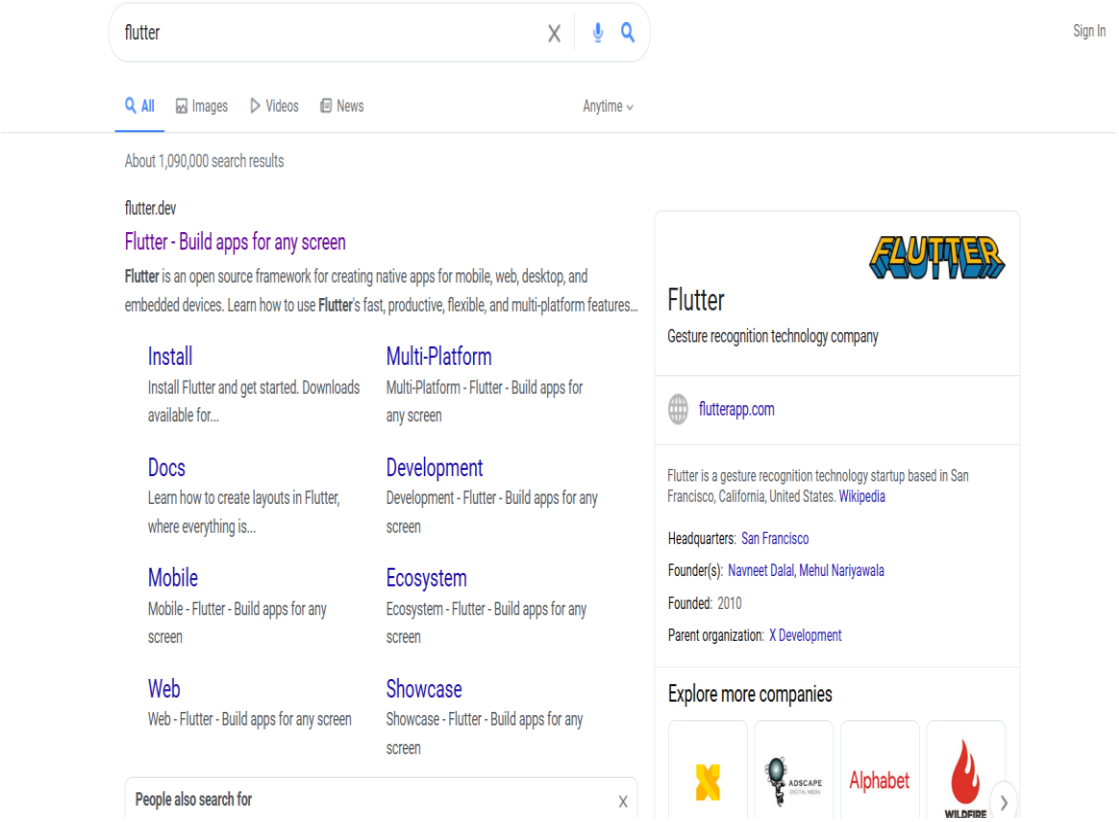
10. a) Write unit tests for UI components.

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

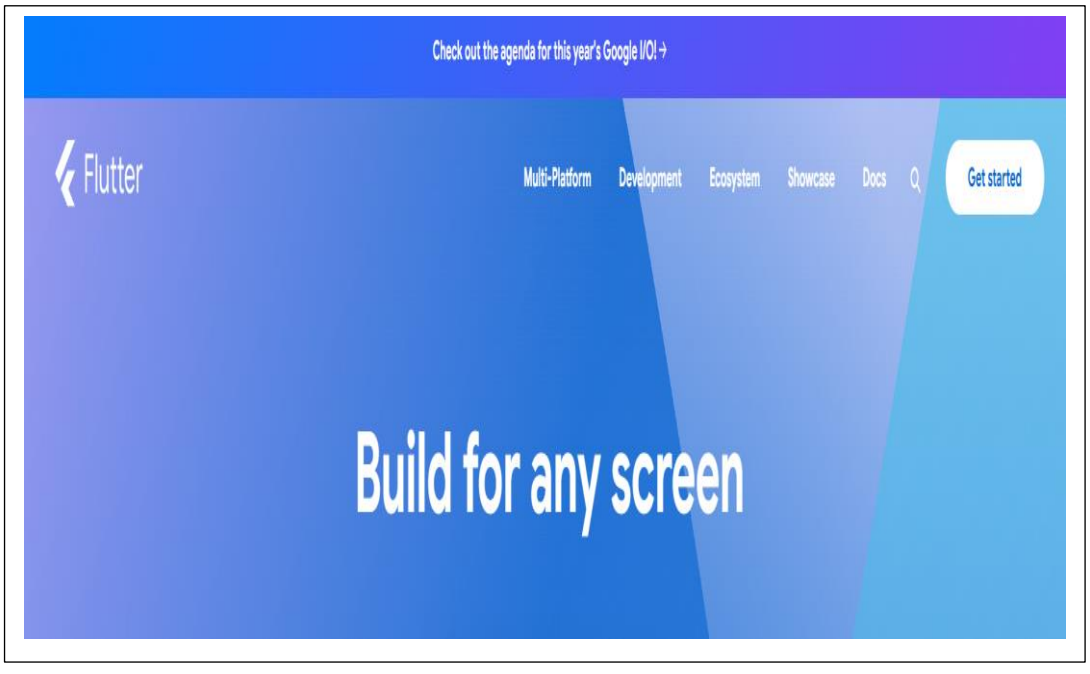
1. Install Flutter and Dart SDK

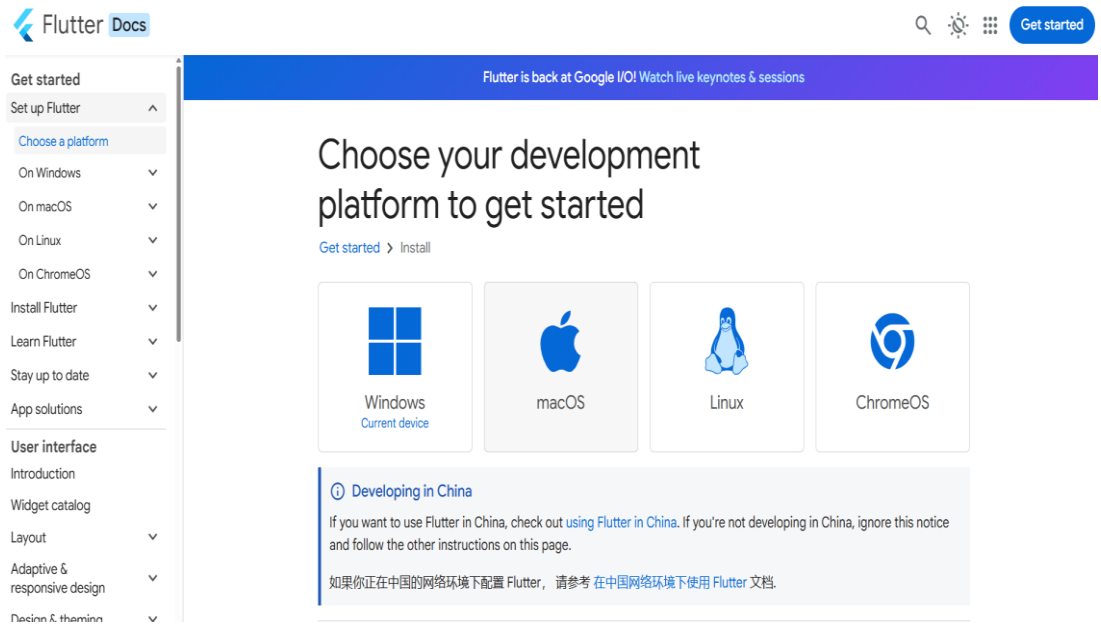
Download Flutter SDK

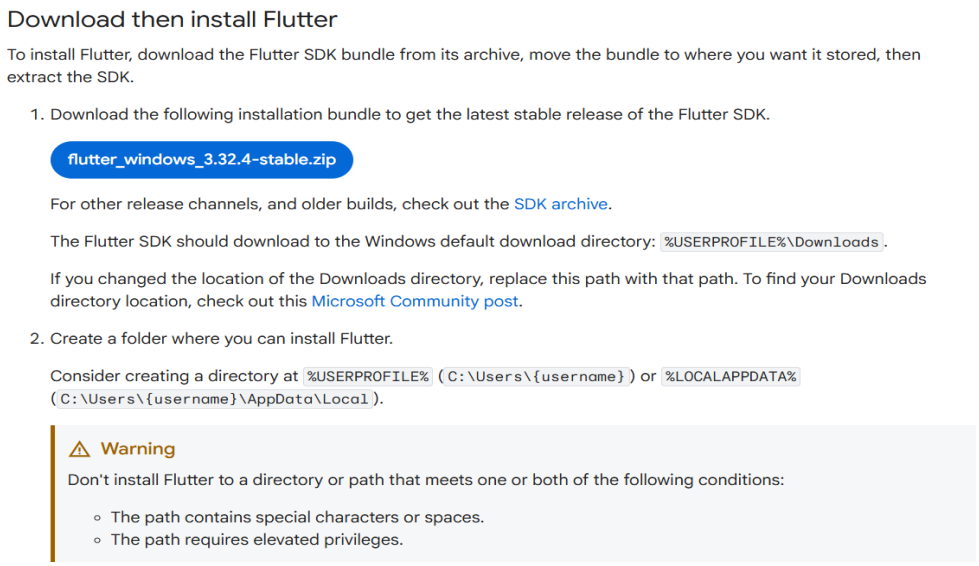
- Go to: https://flutter.dev/docs/get-started/install
- Click on Flutter-Build apps for any screen



Click on Get Started







Extract the ZIP

Extract the zip to a folder like C:\flutter

Add Flutter to PATH

- Open **Start Menu > search "Environment Variables" > Edit system environment variables**
- Click **Environment Variables**
- Under **System variables**, find Path, and click **Edit**
- Click **New**, and add:
  C:\flutter\bin

- Click **OK** to save

Install Flutter & Dart Plugins in VS Code
Open **VS Code**
Go to **Extensions**
Search and install:

- **Flutter**
- **Dart** (this will be auto-installed with Flutter)

2. Write a simple Dart program to understand the language basics

i)

```
void main()

{
var fn="Vignan";
var ln="Lara";
print("Full name is $fn $ln");
}
```

OUTPUT:
D:\program>dart frist.dart //complie code
Full name is Vignan Lara

ii)

```
// Function definition
void Method1(String username)
 {
  print("Hello, $username! Welcome to Dart.");
}

void main()
{
  // Variable declaration
  int number = 10;
  double price = 99.99;
```

```
  String name = "Dart Learner";
  bool isLearning = true;

  // Print variables
  print("Number: $number");
  print("Price: $price");
  print("Name: $name");
  print("Learning Dart: $isLearning");

  // Conditional statement
  if (number > 5)
{
    print("Number is greater than 5");
  } else
 {
    print("Number is 5 or less");
  }

  // Loop example
  print("First 5 natural numbers:");
  for (int i = 1; i <= 5; i++)
{
    print(i);
  }

  // Function call
  Method1 ("Jabeer");
}
```

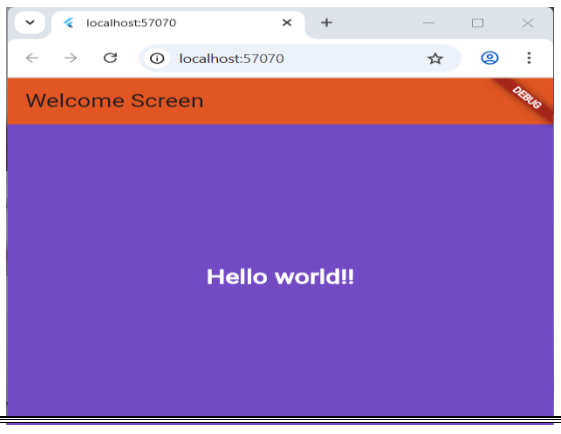
OUTPUT:
Number: 10
Price: 99.99
Name: Dart Learner
Learning Dart: true
Number is greater than 5
First 5 natural numbers:
1
2
3
4
5
Hello, Jabeer! Welcome to Dart.

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

TEXT

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

// Function to run the app
void main() => runApp(const Vlits());

class Vlits extends StatelessWidget {
  const Vlits({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        backgroundColor: Colors.lightGreen,
        appBar: AppBar(
          backgroundColor: Colors.green,
          title: const Text("Welcome Screen"),
        ),
        body: Container(
          child: const Center(
            child: Text(
              "Hello world!!",
              style: TextStyle(
                fontSize: 24,
                fontWeight: FontWeight.bold,
                color: Colors.white,
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

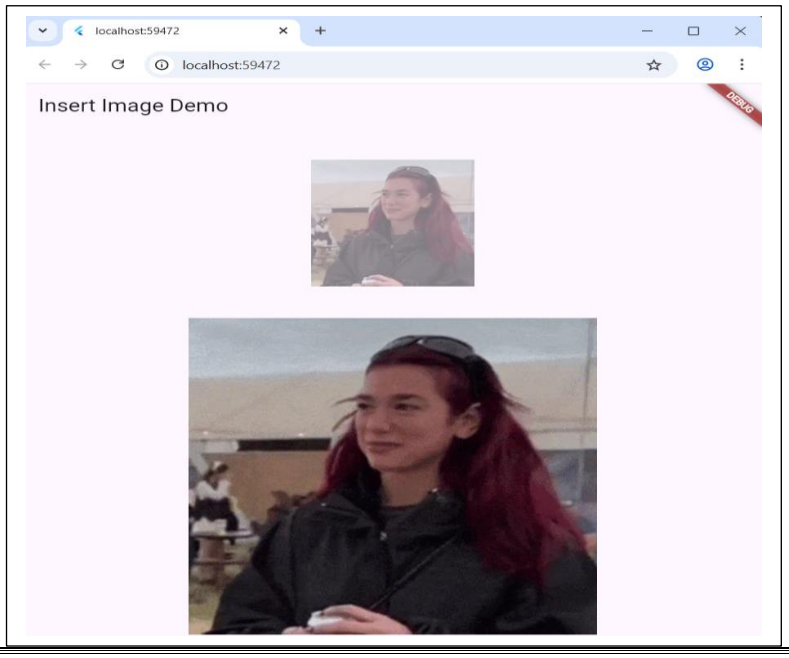
OUTPUT:



IMAGE

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

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Insert Image Demo'),
        ),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              Opacity(
                opacity: 0.5,
                child: Image.asset(
                  'assets/Image/output.webp',
                  height: 200,
                  scale: 2.5,
                ),
              ),
              const SizedBox(height: 20),
              Image.asset(
                'assets/Image/output.webp',
                height: 400,
                width: 400,
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

OUTPUT:



Container

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

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text("Container example"),
        ),
        body: Container(
          height: 200,
          width: double.infinity,
          // color: Colors.purple, // You can use this OR decoration, not both
          alignment: Alignment.center,
          margin: const EdgeInsets.all(20),
          padding: const EdgeInsets.all(30),
          decoration: BoxDecoration(
            border: Border.all(color: Colors.black, width: 3),
          ),
          child: const Text(
            "Hello! I am inside a container!",
            style: TextStyle(fontSize: 20),
          ),
        ),
      ),
    );
  }
}
```

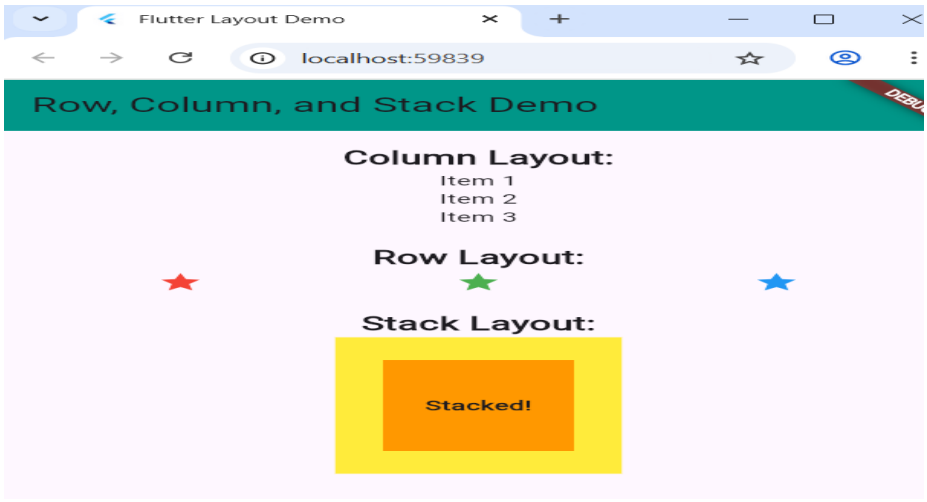
OUTPUT:

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

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

void main() => runApp(const MyLayoutApp());

class MyLayoutApp extends StatelessWidget {
  const MyLayoutApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Layout Demo',
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Row, Column, and Stack Demo'),
          backgroundColor: Colors.teal,
        ),
        body: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            children: [

              // Column Layout
              const Text(
                'Column Layout:',
                style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
              ),
              Column(
                children: const [
                  Text('Item 1'),
                  Text('Item 2'),
                  Text('Item 3'),
                ],
              ),
              const SizedBox(height: 20),

              // Row Layout
              const Text(
                'Row Layout:',
                style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
              ),
              Row(
                mainAxisAlignment: MainAxisAlignment.spaceAround,
                children: const [
```

```
            Icon(Icons.star, color: Colors.red),
            Icon(Icons.star, color: Colors.green),
            Icon(Icons.star, color: Colors.blue),
          ],
        ),
        const SizedBox(height: 20),

        // Stack Layout
        const Text(
          'Stack Layout:',
          style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
        ),
        Stack(
          alignment: Alignment.center,
          children: [
            Container(
              width: 150,
              height: 150,
              color: Colors.yellow,
            ),
            Container(
              width: 100,
              height: 100,
              color: Colors.orange,
            ),
            const Text(
              'Stacked!',
              style: TextStyle(fontWeight: FontWeight.bold),
            ),
          ],
        ),
      ],
    ),
  ),
 );
 }
}
```

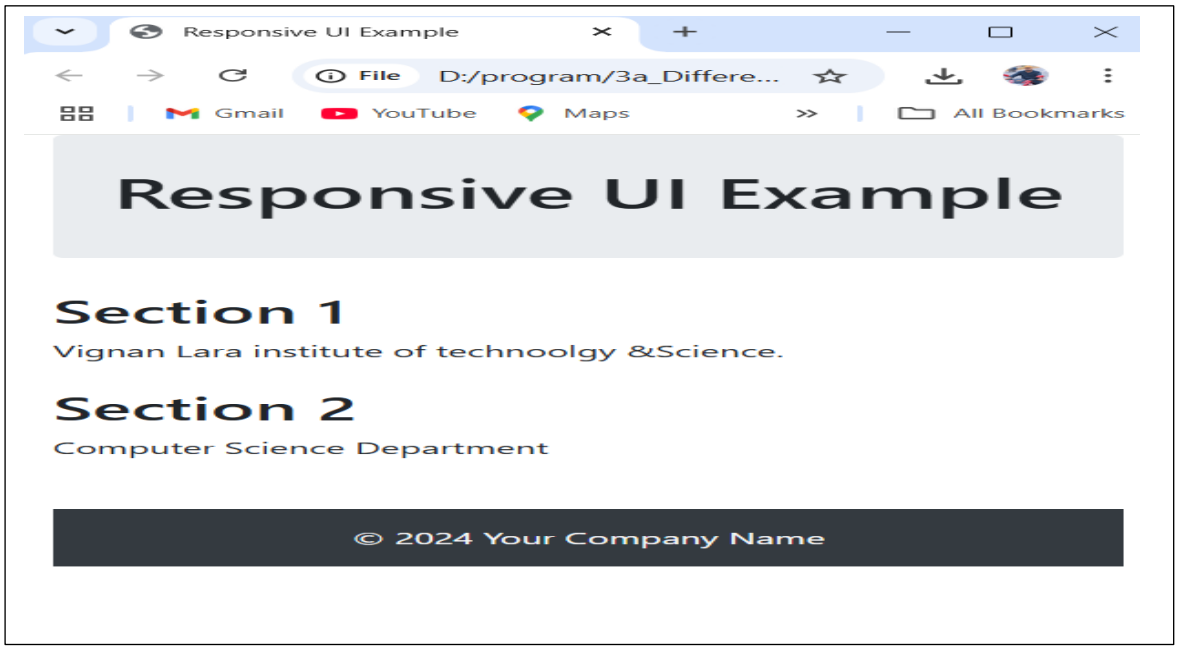
OUTPUT:



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

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Responsive UI Example</title>
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>

  <div class="container">
    <header class="jumbotron text-center">
      <h1>Responsive UI Example</h1>
    </header>

    <main>
      <section class="mb-4">
        <h2>Section 1</h2>
        <p>This is some content for section 1.</p>
      </section>

      <section class="mb-4">
        <h2>Section 2</h2>
        <p>This is some content for section 2.</p>
      </section>
    </main>

    <footer class="bg-dark text-light text-center py-3 mt-5">
      &copy; 2024 Your Company Name
    </footer>
  </div>

  <!-- Bootstrap JS and dependencies -->
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js"></script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

OUTPUT:

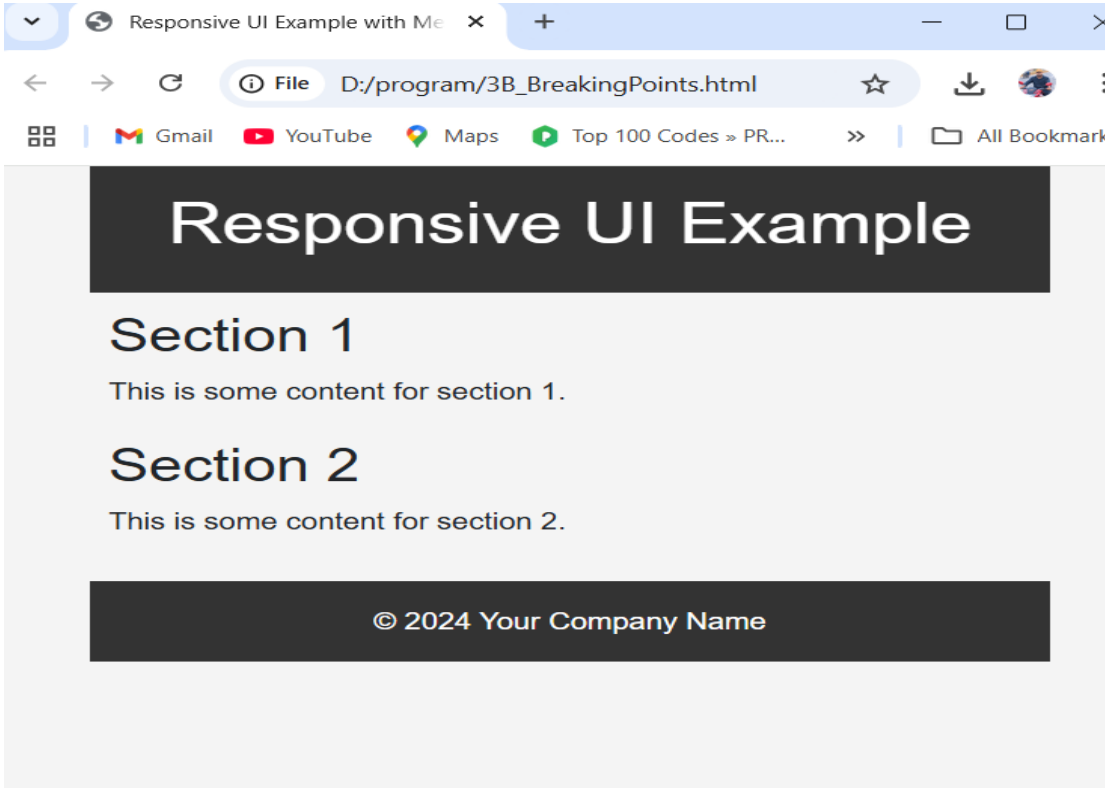


b) Implement media queries and breakpoints for responsiveness.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Responsive UI Example</title>
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>

  <div class="container">
    <header class="jumbotron text-center">
      <h1>Responsive UI Example</h1>
    </header>

    <main>
      <section class="mb-4">
        <h2>Section 1</h2>
        <p>This is some content for section 1.</p>
      </section>

      <section class="mb-4">
        <h2>Section 2</h2>
```

```
      <p>This is some content for section 2.</p>
    </section>
  </main>

  <footer class="bg-dark text-light text-center py-3 mt-5">
    &copy; 2024 Your Company Name
  </footer>
 </div>

 <!-- Bootstrap JS and dependencies -->
 <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
 <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js"></script>
 <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

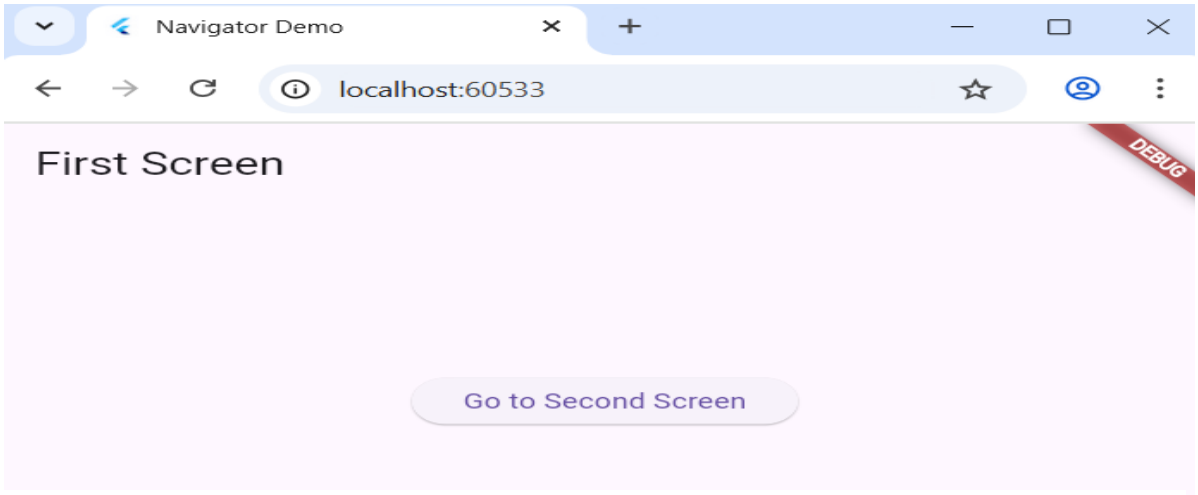
OUTPUT:

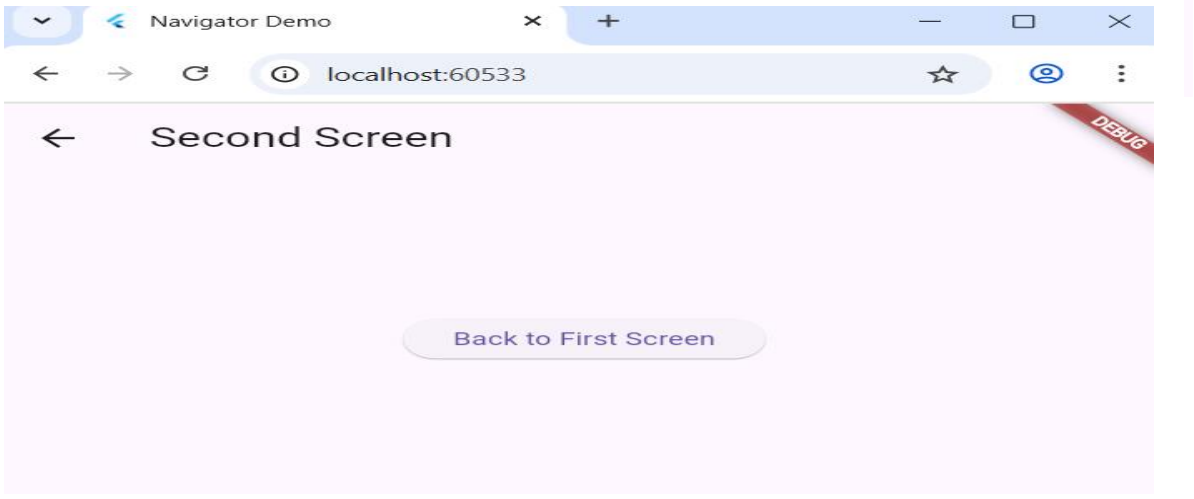


4. a) Set up navigation between different screens using Navigator.

```
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: 'Navigator Demo',
      home: const FirstScreen(),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("First Screen")),
      body: Center(
        child: ElevatedButton(
          child: const Text("Go to Second Screen"),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => const SecondScreen()),
            );
          },
        ),
      ),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
```

```
    return Scaffold(
      appBar: AppBar(title: const Text("Second Screen")),
      body: Center(
        child: ElevatedButton(
          child: const Text("Back to First Screen"),
          onPressed: () {
            Navigator.pop(context);
          },
        ),
      ),
    );
  }
}
```



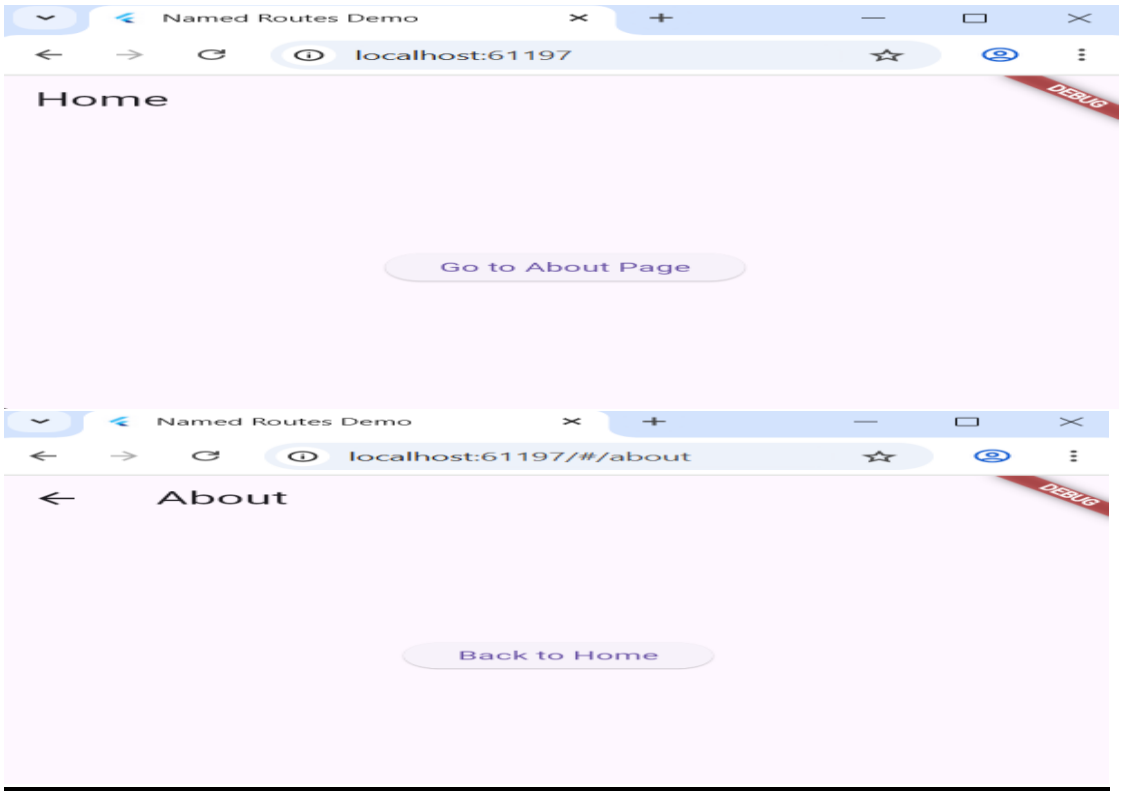


b) Implement navigation with named routes.

```
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: 'Named Routes Demo',
      initialRoute: '/', // First screen to show
      routes: {
        '/': (context) => const HomeScreen(),
        '/about': (context) => const AboutScreen(),
      },
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Home')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Go to About Page'),
          onPressed: () {
            Navigator.pushNamed(context, '/about'); // Navigate using route name
          },
        ),
      ),
    );
  }
}

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

  @override
```

```
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('About')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Back to Home'),
          onPressed: () {
            Navigator.pop(context); // Go back to the previous screen
          },      ),     ),    ); }}
```

OUTPUT:



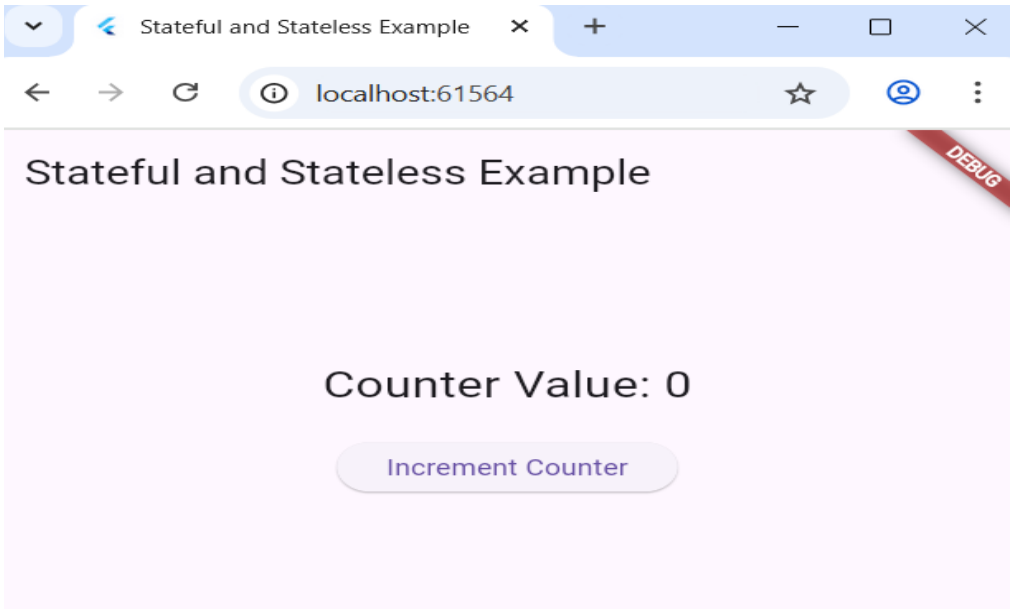
5. a) Learn about stateful and stateless widgets.

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

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

// Stateless Widget
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
```

```
    return MaterialApp(
      title: 'Stateful and Stateless Example',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(),
    );
  }
}

// Stateful Widget
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('Stateful and Stateless Example'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            CounterDisplay(counter),
            SizedBox(height: 20),
            CounterButton(incrementCounter),
          ],
        ),
      ),
    );
  }
}

// Stateless Widget - Display
```

```
class CounterDisplay extends StatelessWidget {
  final int count;

  CounterDisplay(this.count);

  @override
  Widget build(BuildContext context) {
    return Text(
      'Counter Value: $count',
      style: TextStyle(fontSize: 24),
    );
  }
}

// Stateless Widget - Button
class CounterButton extends StatelessWidget {
  final VoidCallback onPressed;

  CounterButton(this.onPressed);

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      onPressed: onPressed,
      child: Text('Increment Counter'),
    );
  }
}
```

OUTPUT:

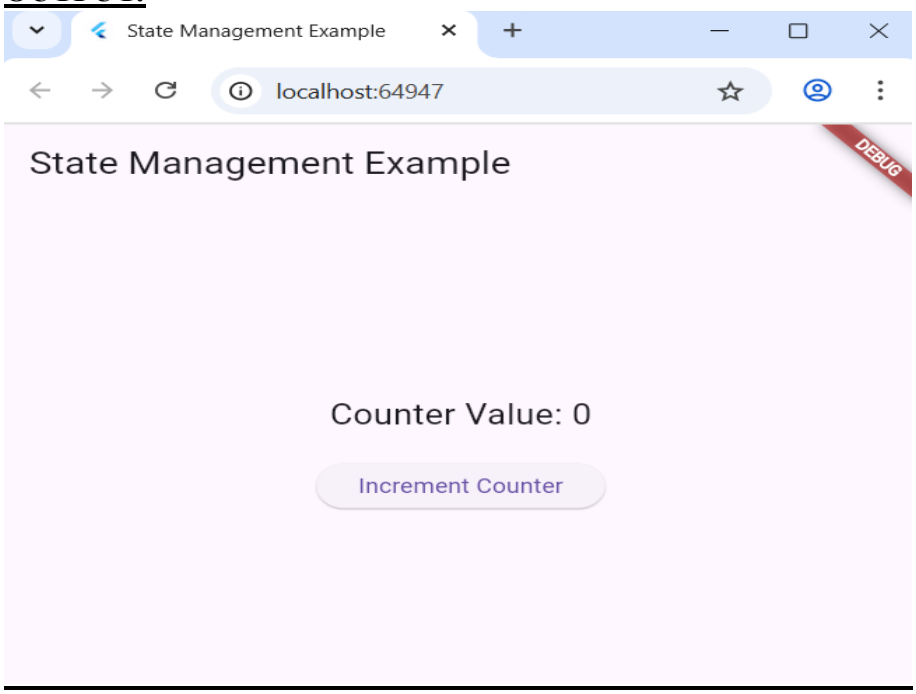


b) Implement state management using set State and Provider.

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

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

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

class CounterModel extends ChangeNotifier {
  int _counter = 0;
  int get counter => _counter;

  void incrementCounter() {
    _counter++;
    notifyListeners();
  }
}

class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final counterModel = Provider.of<CounterModel>(context);

    return Scaffold(
      appBar: AppBar(
```

```
      title: Text('State Management Example'),
    ),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Text(
            'Counter Value: ${counterModel.counter}',
            style: TextStyle(fontSize: 20),
          ),
          SizedBox(height: 20),
          ElevatedButton(
            onPressed: counterModel.incrementCounter,
            child: Text('Increment Counter'),
          ),
        ],
      ),
    ),
  );
 }
}
```

OUTPUT:



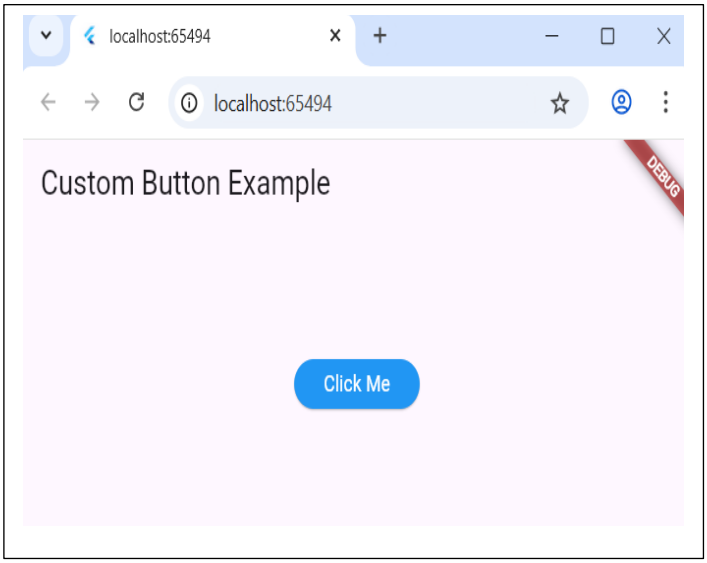
6. a) Create custom widgets for specific UI elements.

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

// CustomButton widget
class CustomButton extends StatelessWidget {
  final String text;
  final VoidCallback onPressed;
  final Color buttonColor;
  final Color textColor;

  const CustomButton({
    Key? key,
    required this.text,
    required this.onPressed,
    this.buttonColor = Colors.blue,
    this.textColor = Colors.white,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      onPressed: onPressed,
      style: ButtonStyle(
        backgroundColor: MaterialStateProperty.all<Color>(buttonColor),
        foregroundColor: MaterialStateProperty.all<Color>(textColor),
      ),
      child: Text(text),
    );
  }
}

// CustomAlertDialog widget
class CustomAlertDialog extends StatelessWidget {
  final String title;
  final String message;
  final String positiveButtonText;
  final String negativeButtonText;
  final VoidCallback onPositivePressed;
  final VoidCallback onNegativePressed;

  const CustomAlertDialog({
    Key? key,
    required this.title,
```

```
    required this.message,
    required this.positiveButtonText,
    required this.negativeButtonText,
    required this.onPositivePressed,
    required this.onNegativePressed,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return AlertDialog(
      title: Text(title),
      content: Text(message),
      actions: [
        CustomButton(
          text: negativeButtonText,
          onPressed: onNegativePressed,
          buttonColor: Colors.grey,
        ),
        CustomButton(
          text: positiveButtonText,
          onPressed: onPositivePressed,
        ),
      ],
    );
  }
}

// Main app
void main() {
  runApp(const MyApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: const Text('Custom Button Example')),
        body: Center(
          child: CustomButton(
            text: 'Click Me',
            onPressed: () {
              print('Button Pressed');
```

```
            // Show custom alert dialog
            showDialog(
              context: context,
              builder: (context) => CustomAlertDialog(
                title: 'Hello!',
                message: 'Do you want to continue?',
                positiveButtonText: 'Yes',
                negativeButtonText: 'No',
                onPositivePressed: () {
                  print('Yes Pressed');
                  Navigator.of(context).pop();
                },
                onNegativePressed: () {
                  print('No Pressed');
                  Navigator.of(context).pop();
                },
              ),
            );
          },
        ),
      ),
    ),
  );
 }
}
```

OUTPUT:

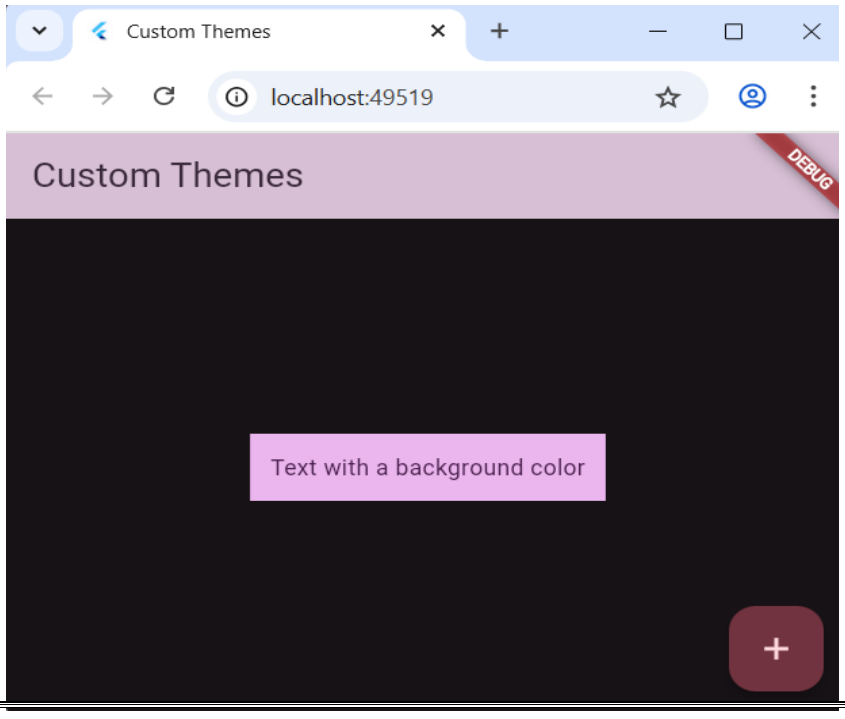


b) Apply styling using themes and custom styles

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

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

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

  @override
  Widget build(BuildContext context) {
    const appName = 'Custom Themes';

    return MaterialApp(
      title: appName,
      theme:  ThemeData(
  colorScheme: ColorScheme.fromSeed(
    seedColor: Colors.purple,
    brightness: Brightness.dark,
  ),
  useMaterial3: true,
),

      home: const MyHomePage(title: appName),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(
          title,
          style: Theme.of(context).textTheme.titleLarge!.copyWith(
              color: Theme.of(context).colorScheme.onSecondary,
            ),
```

```
        ),
        backgroundColor: Theme.of(context).colorScheme.secondary,
      ),
      body: Center(
        child: Container(
          padding: const EdgeInsets.all(12),
          color: Theme.of(context).colorScheme.primary,
          child: Text(
            'Text with a background color',
            style: Theme.of(context).textTheme.bodyMedium!.copyWith(
                  color: Theme.of(context).colorScheme.onPrimary,
                ),
          ),
        ),
      ),
      floatingActionButton: Theme(
        data: Theme.of(context).copyWith(
          colorScheme: ColorScheme.fromSeed(
            seedColor: Colors.pink,
            brightness: Brightness.dark,
          ),
        ),
        child: FloatingActionButton(
          onPressed: () {
            // Add your logic here
          },
          child: const Icon(Icons.add),
        ),
      ),
    );
  }
}
```

OUTPUT:



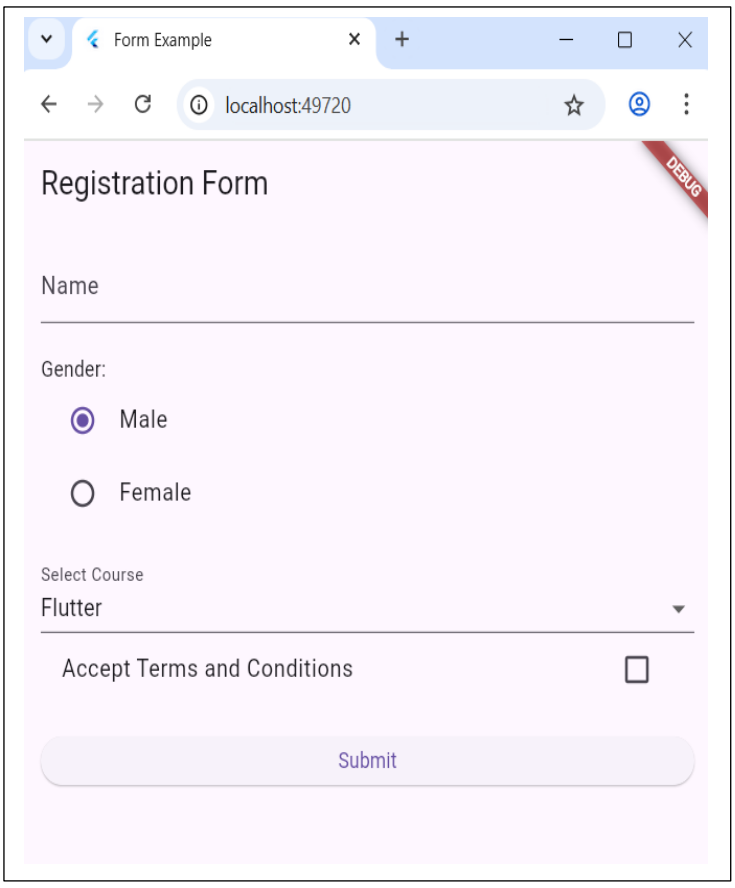
7.a) Design a form with various input fields.

```
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: 'Form Example',
      theme: ThemeData(primarySwatch: Colors.blue),
      home: const MyFormPage(),
    );
  }
}

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

  @override
  State<MyFormPage> createState() => _MyFormPageState();
}

class _MyFormPageState extends State<MyFormPage> {
  final _formKey = GlobalKey<FormState>();

  String name = '';
  String gender = 'Male';
  bool acceptTerms = false;
  String selectedCourse = 'Flutter';

  final List<String> courses = ['Flutter', 'React', 'Angular'];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Registration Form')),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Form(
          key: _formKey,
```

```
child: ListView(
  children: [
    // Name Field
    TextFormField(
      decoration: const InputDecoration(labelText: 'Name'),
      validator: (value) =>
          value == null || value.isEmpty ? 'Enter your name' : null,
      onSaved: (value) => name = value!,
    ),

    // Gender Radio
    const SizedBox(height: 20),
    const Text("Gender:"),
    ListTile(
      title: const Text('Male'),
      leading: Radio<String>(
        value: 'Male',
        groupValue: gender,
        onChanged: (value) => setState(() => gender = value!),
      ),
    ),
    ListTile(
      title: const Text('Female'),
      leading: Radio<String>(
        value: 'Female',
        groupValue: gender,
        onChanged: (value) => setState(() => gender = value!),
      ),
    ),

    // Dropdown for Course
    const SizedBox(height: 20),
    DropdownButtonFormField<String>(
      value: selectedCourse,
      decoration: const InputDecoration(labelText: 'Select Course'),
      items: courses
          .map((course) => DropdownMenuItem(
                value: course,
                child: Text(course),
              ))
          .toList(),
      onChanged: (value) => setState(() => selectedCourse = value!),
    ),

    // Terms Checkbox
    CheckboxListTile(
```

```
            title: const Text('Accept Terms and Conditions'),
            value: acceptTerms,
            onChanged: (value) => setState(() => acceptTerms = value!),
          ),

          // Submit Button
          const SizedBox(height: 20),
          ElevatedButton(
            onPressed: () {
              if (_formKey.currentState!.validate() && acceptTerms) {
                _formKey.currentState!.save();
                showDialog(
                  context: context,
                  builder: (context) => AlertDialog(
                    title: const Text('Form Submitted'),
                    content: Text('Name: $name\n'
                        'Gender: $gender\n'
                        'Course: $selectedCourse'),
                  ),
                );
              } else if (!acceptTerms) {
                ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
                    content: Text('Please accept terms and conditions.')));
              }
            },
            child: const Text('Submit'),
          ),
        ],
      ),
    ),
  ),
);
  }
}
```

OUTPUT:



b) Implement form validation and error handling.

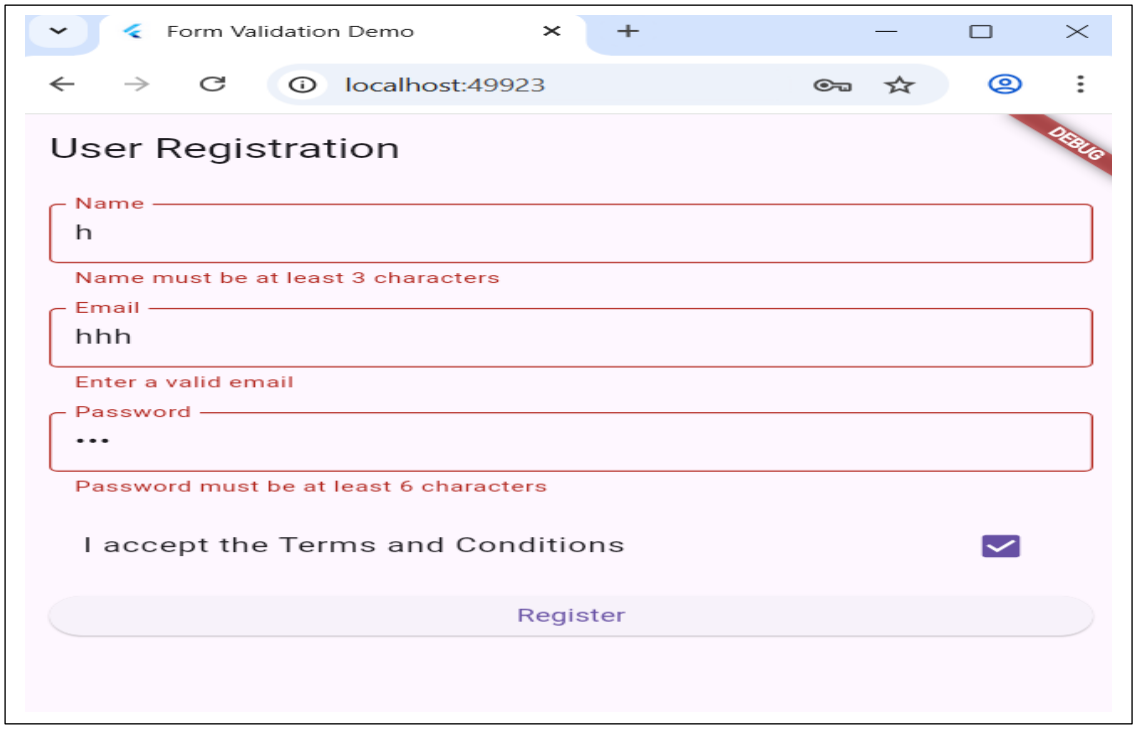
```
import 'package:flutter/material.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Form Validation Demo',
      theme: ThemeData(primarySwatch: Colors.teal),
      home: const FormValidationPage(),
    );
  }
}

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

  @override
  State<FormValidationPage> createState() => _FormValidationPageState();
}

class _FormValidationPageState extends State<FormValidationPage> {
  final _formKey = GlobalKey<FormState>();
  bool _acceptTerms = false;

  String _name = '';
  String _email = '';
  String _password = '';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("User Registration")),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Form(
          key: _formKey,
          child: ListView(
            children: [
```

```
// Name
TextFormField(
  decoration: const InputDecoration(
    labelText: "Name",
    border: OutlineInputBorder(),
  ),
  validator: (value) {
    if (value == null || value.trim().isEmpty) {
      return "Name is required";
    } else if (value.trim().length < 3) {
      return "Name must be at least 3 characters";
    }
    return null;
  },
  onSaved: (value) => _name = value!.trim(),
),
const SizedBox(height: 16),

// Email
TextFormField(
  decoration: const InputDecoration(
    labelText: "Email",
    border: OutlineInputBorder(),
  ),
  keyboardType: TextInputType.emailAddress,
  validator: (value) {
    if (value == null || value.trim().isEmpty) {
      return "Email is required";
    } else if (!RegExp(r"^[\w-\.]+@([\w-]+\.)+[\w]{2,4}")
        .hasMatch(value.trim())) {
      return "Enter a valid email";
    }
    return null;
  },
  onSaved: (value) => _email = value!.trim(),
),
const SizedBox(height: 16),

// Password
TextFormField(
  decoration: const InputDecoration(
    labelText: "Password",
    border: OutlineInputBorder(),
  ),
  obscureText: true,
  validator: (value) {
```

```
    if (value == null || value.isEmpty) {
      return "Password is required";
    } else if (value.length < 6) {
      return "Password must be at least 6 characters";
    }
    return null;
  },
  onSaved: (value) => _password = value!,
),
const SizedBox(height: 16),

// Terms Checkbox
CheckboxListTile(
  title: const Text("I accept the Terms and Conditions"),
  value: _acceptTerms,
  onChanged: (value) {
    setState(() {
      _acceptTerms = value ?? false;
    });
  },
),

const SizedBox(height: 16),

// Submit Button
ElevatedButton(
  onPressed: () {
    final isValid = _formKey.currentState!.validate();

    if (!isValid) return;

    if (!_acceptTerms) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
            content: Text("You must accept terms to proceed.")),
      );
      return;
    }

    _formKey.currentState!.save();

    showDialog(
      context: context,
      builder: (context) => AlertDialog(
        title: const Text("Registration Successful"),
        content: Text(
```

```
                    "Name: $_name\nEmail: $_email\nPassword: $_password"),
                ),
              );
            },
            child: const Text("Register"),                              :
          ),
        ],
      ),
    ),
  ),
 );
 }
}
```

OUTPUT:

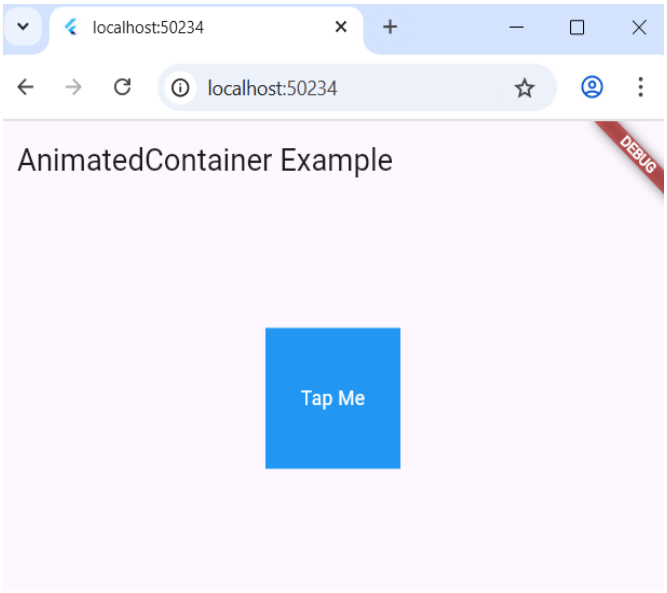


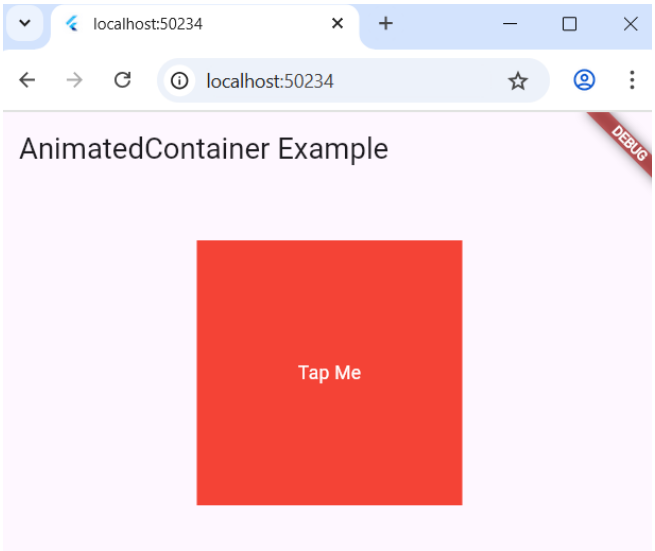
8. a) Add animations to UI elements using Flutter's animation framework.

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

void main() => runApp(const MyApp());

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

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

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

  @override
  State<AnimationDemo> createState() => _AnimationDemoState();
}

class _AnimationDemoState extends State<AnimationDemo> {
  double _width = 100;
  double _height = 100;
  Color _color = Colors.blue;

  void _animateContainer() {
    setState(() {
      _width = _width == 100 ? 200 : 100;
      _height = _height == 100 ? 200 : 100;
      _color = _color == Colors.blue ? Colors.red : Colors.blue;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('AnimatedContainer Example')),
      body: Center(
        child: GestureDetector(
          onTap: _animateContainer,
          child: AnimatedContainer(
            width: _width,
```

```
            height: _height,
            color: _color,
            duration: const Duration(seconds: 1),
            curve: Curves.easeInOut,
            child: const Center(child: Text("Tap Me", style: TextStyle(color: Colors.white))),
          ),
        ),
      ),
    );
  }
}
```

OUTPUT:





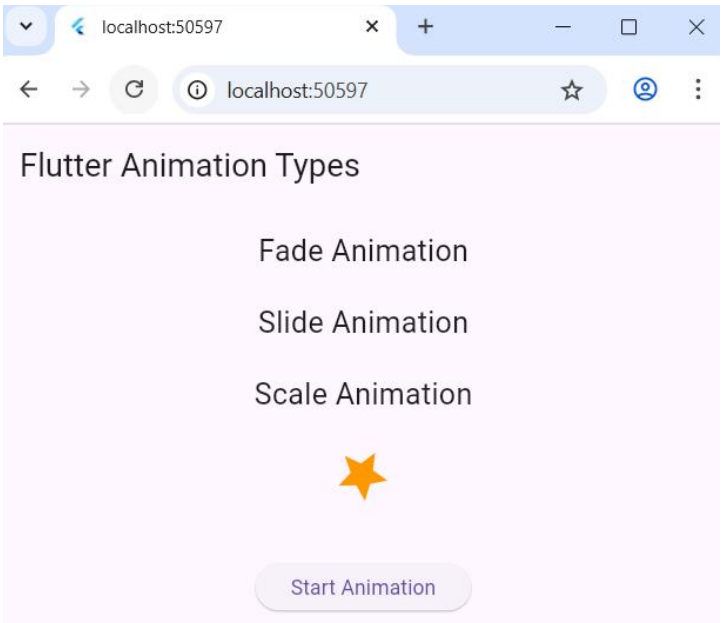
b) Experiment with different types of animations (fade, slide, etc.).

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

void main() => runApp(const MyApp());

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

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

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

  @override
  State<AnimationTypesDemo> createState() => _AnimationTypesDemoState();
}

class _AnimationTypesDemoState extends State<AnimationTypesDemo>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller;
  late Animation<double> _fadeAnimation;
  late Animation<Offset> _slideAnimation;
  late Animation<double> _scaleAnimation;
  late Animation<double> _rotationAnimation;

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      duration: const Duration(seconds: 2),
      vsync: this,
    );

    _fadeAnimation = Tween<double>(begin: 0.0, end: 1.0).animate(_controller);
    _slideAnimation =
        Tween<Offset>(begin: const Offset(-1, 0), end: Offset.zero)
```

```
        .animate(_controller);
  _scaleAnimation =
      Tween<double>(begin: 0.5, end: 1.0).animate(_controller);
  _rotationAnimation =
      Tween<double>(begin: 0.0, end: 2 * 3.1416).animate(_controller);
}

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

void _startAnimation() {
  _controller.forward(from: 0.0);
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: const Text('Flutter Animation Types')),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          FadeTransition(
            opacity: _fadeAnimation,
            child: const Text("Fade Animation",
                style: TextStyle(fontSize: 20)),
          ),
          const SizedBox(height: 20),
          SlideTransition(
            position: _slideAnimation,
            child: const Text("Slide Animation",
                style: TextStyle(fontSize: 20)),
          ),
          const SizedBox(height: 20),
          ScaleTransition(
            scale: _scaleAnimation,
            child: const Text("Scale Animation",
                style: TextStyle(fontSize: 20)),
          ),
          const SizedBox(height: 20),
          RotationTransition(
            turns: _rotationAnimation,
            child: const Icon(Icons.star, size: 40, color: Colors.orange),
```

```
          ),
          const SizedBox(height: 40),
          ElevatedButton(
            onPressed: _startAnimation,
            child: const Text("Start Animation"),
          ),
        ],
      ),
    ),
  );
 }
}
```

OUTPUT:



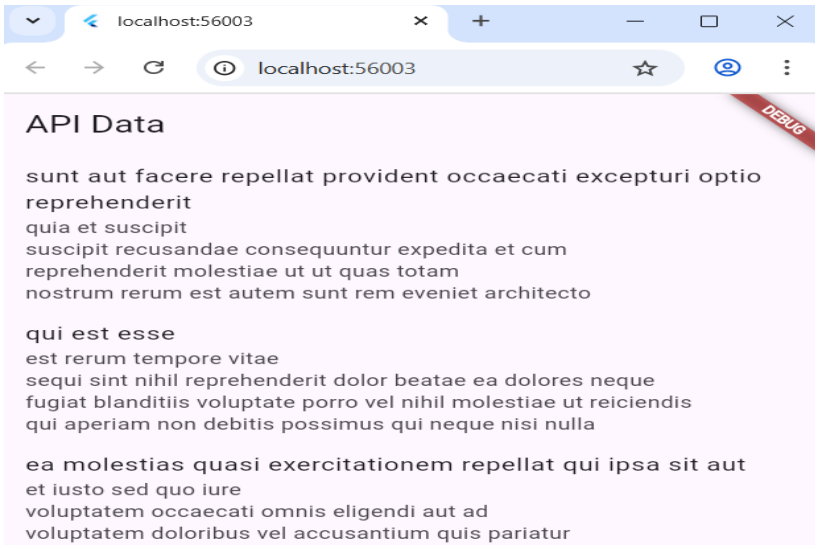
9. a) Fetch data from a REST API.

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

// Model class
class Post {
  final int id;
  final String title;
  final String body;

  Post({required this.id, required this.title, required this.body});
```

```
  factory Post.fromJson(Map<String, dynamic> json) {
    return Post(
      id: json['id'],
      title: json['title'],
      body: json['body'],
    );
  }
}

// Fetch function
Future<List<Post>> fetchPosts() async {
  final response =
      await http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts'));

  if (response.statusCode == 200) {
    List jsonResponse = json.decode(response.body);
    return jsonResponse.map((data) => Post.fromJson(data)).toList();
  } else {
    throw Exception('Failed to load posts');
  }
}

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

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

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

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

  @override
  State<PostListScreen> createState() => _PostListScreenState();
}

class _PostListScreenState extends State<PostListScreen> {
  late Future<List<Post>> futurePosts;
```

```
  @override
  void initState() {
    super.initState();
    futurePosts = fetchPosts();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('API Data')),
      body: FutureBuilder<List<Post>>(
        future: futurePosts,
        builder: (context, snapshot) {
          if (snapshot.hasData) {
            return ListView(
              children: snapshot.data!
                  .map((post) => ListTile(
                        title: Text(post.title),
                        subtitle: Text(post.body),
                      ))
                  .toList(),
            );
          } else if (snapshot.hasError) {
            return Center(child: Text("${snapshot.error}"));
          }

          return const Center(child: CircularProgressIndicator());
        },
      ),
    );
  }
}
```

OUTPUT:

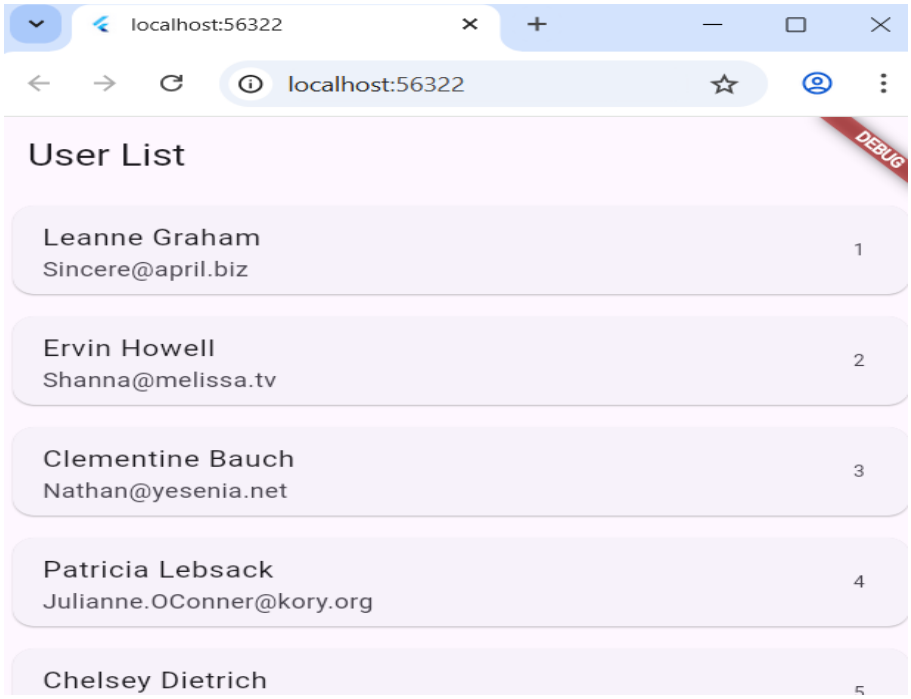


b) Display the fetched data in a meaningful way in the UI

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

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

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

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

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

  @override
  State<UserListScreen> createState() => _UserListScreenState();
}

class _UserListScreenState extends State<UserListScreen> {
  List users = [];
  bool isLoading = true;

  @override
  void initState() {
    super.initState();
    fetchUsers();
  }

  Future<void> fetchUsers() async {
    final response = await http.get(Uri.parse('https://jsonplaceholder.typicode.com/users'));

    if (response.statusCode == 200) {
      setState(() {
        users = json.decode(response.body);
        isLoading = false;
      });
```

```
    } else {
      throw Exception('Failed to load users');
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('User List')),
      body: isLoading
          ? const Center(child: CircularProgressIndicator())
          : ListView.builder(
              itemCount: users.length,
              itemBuilder: (context, index) {
                final user = users[index];
                return Card(
                  margin: const EdgeInsets.all(8),
                  child: ListTile(
                    title: Text(user['name']),
                    subtitle: Text(user['email']),
                    trailing: Text(user['id'].toString()),
                  ),
                );
              },
            ),
    );
  }
}
```

OUTPUT:



10. a) Write unit tests for UI components.

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

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

Add the test or flutter_test dependency.

Create a test file.

Create a class to test.

Write a test for our class.

Combine multiple tests in a group.

Run the tests.

For more information about the test package, see the test package documentation.

1. Add the test dependency

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

To add the test package as a dev dependency, run flutter pub add:

content copy flutter pub add dev test

2. Create a test file

In this example, create two files: counter.dart and counter_test.dart.
The counter.dart file contains a class that you want to test and resides in the lib folder.

The counter_test.dart file contains the tests themselves and lives inside the test folder.

In general, test files should reside inside a test folder located at the root of your Flutter application or package. Test files should always end with test dart, this is the convention used by the test runner when searching for tests. When you're finished, the folder structure should look like this:

content_copy

counter_app/lib/ counter.dart

test/

counter test.dart 3. Create a class

to test

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

content_copy class Counter { int value = 0;

void increment() => value++;

void decrement() => value-;

} Note: For simplicity, this tutorial does not follow the "Test Driven Development" approach. If you're more comfortable with that style of development, you can always go that route.

4. Write a test for our class

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

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

// Import the test package and Counter class import

'package:counter_app/counter.dart'; import

'package:test/test.dart';

void main(){

test('Counter value should be incremented', (){ final counter

Counter(); counter.increment():

expect(counter.value, 1); 1);

5. Combine multiple tests in a group

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

content_copy

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

'package:test/test.dart';

void main() {

group('Test start, increment, decrement', () f test('value should start at 0, () {
expect(Counter().value, 0);

test('value should be incremented', () { Counter(); counter.increment();

final counter-

expect(counter.value, 1); 1):

test('value should be decremented', (){ Counter(); counter.decrement();

final counter =

expect(counter.value, -1); 1):

6. Run the tests

Now that you have a Counter class with tests in place, you can run the tests.

Run tests using IntelliJ or VSCode

The Flutter plugins for IntelliJ and VSCode support running tests. This is often the best option while
Run the tests

Now that you have a Counter class with tests in place, you can run the tests.

Run tests using IntelliJ or VSCode

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

Intellid

Open the counter test.dart file

Go to Run Run 'tests in counter test.dart'. You can also press the appropriate keyboard shortcut for your platform.

VSCode

Open the counter test dart file

Go to Run Start Debugging. You can also press the appropriate keyboard shortcut for your platform. Run tests in a terminal

To run the all tests from the terminal, run the following command from the root of the project:

content copy

flutter test test/counter test.dart

To run all tests you put into one group, run the following command from the root of the project:

content copy flutter test-plain-name "Test start, increment, decrement" This example uses the group created in section 5.

To learn more about unit tests, you can execute this command

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

Ans) Flutter provides a set of debugging tools that can help you identify and fix issues in your app. Here's a step-by-step guide on how to use these tools:

1. Flutter DevTools:

Run your app with the flutter run command.

Open DevTools by running the following command in your terminal: bash

flutter pub global activate devtools flutter pub global

run devtools

Open your app in a Chrome browser and connect it to Dev Tools by clicking on the "Open Dev Tools" button in the terminal or by navigating to http://127.0.0.1:9100/, DevTools provides tabs like Inspector, Timeline, Memory, and more.

2. Flutter Inspector.

Use the Flutter Inspector in your integrated development environment (IDE) like Android Studio or Visual Studio Code.

Toggle the Inspector in Android Studio with the shortcut Alt + Shift + D (Windows/Linux) or Option +

Shift + D (Mac).

Inspect the widget tree, modify widget properties, and observe widget relationships.

3. Hot Reload:

Leverage Hot Reload to see the immediate effect of code changes without restarting the entire app. Press R in the terminal or use the "Hot Reload" button in your IDE.

4. Debugging with Breakpoints: Set breakpoints in your code to pause execution and inspect variables. Use the debugger in your IDE to step through code and identify issues.

5. Logging: Utilize the print function to log messages to the console. 매

print('Debugging message'); View logs in the terminal or the "Logs" tab in Dev Tools.

6. Debug Paint: Enable debug paint to visualize the layout and rendering of widgets. Use the debugPaint SizeEnabled and debugPaint BaselinesEnabled flags.

void main() { debug PaintSizeEnabled = true; // Shows bounding boxes of widgets
runApp(MyApp()); }
7. Memory Profiling:
Use the "Memory" tab in DevTools to analyze memory usage and identify potential memory leak Monitor object allocations and deallocations.

8. Performance Profiling (Timeline):
Analyze app performance using the "Timeline" tab in DevTools. Identify UI jank, slow frames, and performance bottlenecks.

9. Flutter Driver Tests:

Write automated UI tests using Flutter Driver.

Simulate user interactions and validate the correctness of your UI.