J.N.T.U.H. UNIVERSITY COLLEGE OF ENGINEERING HYDERABAD (Autonomous)

KUKATPALLY, HYDERABAD – 500 085


# Certificate

Certified that this is the bonafide record of the practical work done during

the academic year _____ by

Name _____ Roll Number

_____ Class _____ in the

Laboratory of _____ of

the Department of _____

Signature of the Staff Member Signature of the Head of the Department

Date of Examination_____

Signature of the Examiner/s

Internal Examiner

External Examiner

J.N.T.U.H. UNIVERSITY COLLEGE OF ENGINEERING HYDERABAD
(Autonomous)

KUKATPALLY, HYDERABAD – 500 085

| S.No. | Name of the Experiment | Date of Experiment | Page Number | Marks | Remarks |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |

Name _____ Roll Number _____

Class _____ Year_____ Laboratory _____

# List of Experiments

J.N.T.U.H. UNIVERISTY COLLEGE OF ENGINEERING HYDERABAD

1. a) Install Flutter and Dart SDK

Here are the steps to install Flutter with Android Studio:

1. Download and Install Android Studio:
    - Go to the Android Studio download page and download the latest version. o Follow the installation instructions for your operating system.
2. Install Flutter and Dart Plugins:
    - Open Android Studio. o Go to File > Settings (or Preferences on macOS). o Select Plugins from the sidebar. o Search for Flutter and click Install. o Click Yes when prompted to install the Dart plugin as well.
    - Restart Android Studio to apply the changes.
3. Set Up Flutter SDK:
    - Download the Flutter SDK from the Flutter website.
    - Extract the downloaded file to a desired location on your system.
    - Add the Flutter bin directory to your system's PATH.
4. Create a New Flutter Project:
    - Open Android Studio. o Click on Start a new Flutter project from the welcome screen. o Select Flutter Application and click Next.
    - Enter your project name and location, then click Finish.
5. Run Your Flutter App:
    - Connect a physical device or start an emulator.
    - Click the Run button in Android Studio to build and run your Flutter app.

For more detailed instructions, you can refer to the official Flutter documentation

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

```
import 'dart:io'; void
main() {
  print('Enter first number:');
  double num1 = double.parse(stdin.readLineSync()!);
print('Enter second number:');
  double num2 =
double.parse(stdin.readLineSync()!);   print('Choose
an operation (+, -, *, /):');   String operation =
stdin.readLineSync()!;   double result;   switch
(operation) {     case '+':        result = num1 + num2;
break;     case '-':
     result = num1 - num2;
break;     case '*':
     result = num1 * num2;
break;     case '/':
     result = num1 / num2;
break;     default:
print('Invalid operation');
return;   }
print('Result: $result');
}
```

OUTPUT

```
Enter first number:
5
Enter second number:
3
Choose an operation (+, -, *, /):
```

+

Result: 8.0

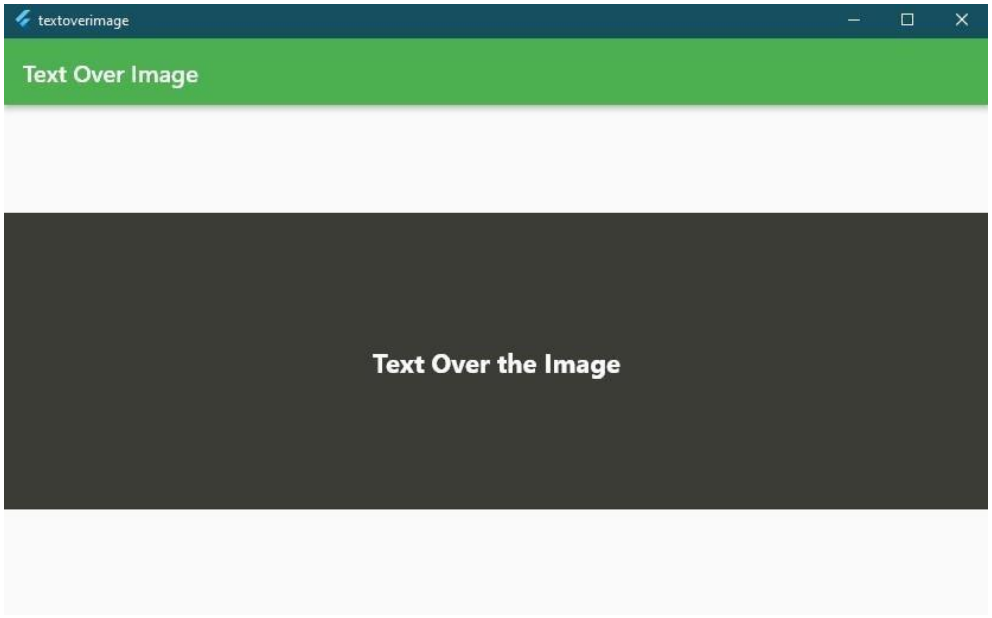
2. a) Explore various Flutter widgets (Text, Image, Container, etc.).

```
import 'package:flutter/material.dart';
void main() {  runApp(RunMyApp());
}
class RunMyApp extends StatelessWidget {  const
RunMyApp({super.key});
@override
Widget  build(BuildContext  context)  {          return
```

```
MaterialApp(   debugShowCheckedModeBanner: false,
theme:          ThemeData(primarySwatch:
Colors.green),   home: Scaffold(  appBar: AppBar(
title: Text("Text Over Image"),
),
body: Center(
child: Stack(
children: [
                Container(
  alignment: Alignment.center,    child: Image.asset(
'assets/s2.png',     height:
200,
                        width: double.infinity,
          fit: BoxFit.cover,
                ),
                ),
   Container(                alignment:  Alignment.center,
child: Text(
                        'Text Over the Image',
style: TextStyle(color: Colors.white,
fontWeight: FontWeight.bold,
fontSize: 24.0),
                        )),
          ],
          ),
),
) ,
```

```
        );
}
}
```

OUTPUT




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

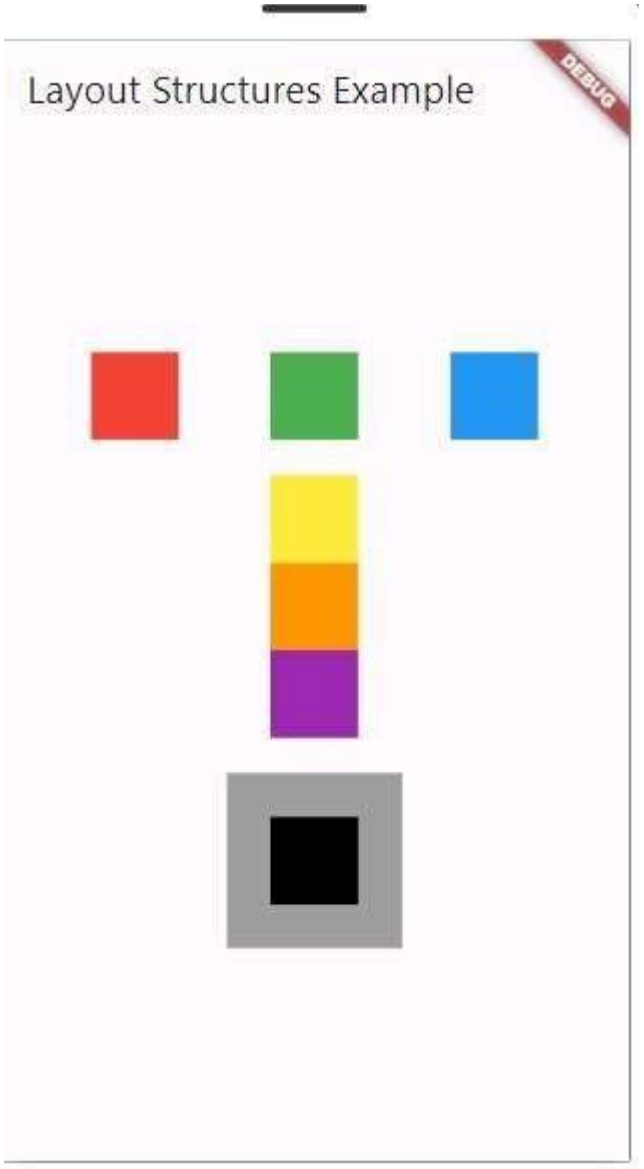
```
import 'package:flutter/material.dart';
void main() {
runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Layout Structures Example'),
```

```
      ),
      body: Center(
child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
children: <Widget>[            Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
children: <Widget>[              Container(              color:
Colors.red,              width: 50,              height: 50,
            ),
            Container(
color: Colors.green,              width:
50,              height:
50,

            ),
            Container(
color: Colors.blue,              width:
50,              height:
50,
            ),
          ],
        ),
        SizedBox(height: 20),
Column(
            mainAxisAlignment: MainAxisAlignment.center,
children: <Widget>[              Container(              color:
Colors.yellow,              width: 50,              height: 50,
            ),
```

```
                Container(
color: Colors.orange,                  width:
50,                height:
50,
                ),
                Container(
color: Colors.purple,                  width:
50,                height:
50,
                ),
              ],
            ),
            SizedBox(height: 20),
            Stack(              alignment:
Alignment.center,              children: <Widget>[
Container(                  color: Colors.grey,
width:
100,                height: 100,
                ),
                Container(
color: Colors.black,                  width:
50,                height:
50,
                ),
              ],
            ),
          ],
        ),
      ),
```

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

OUTPUT

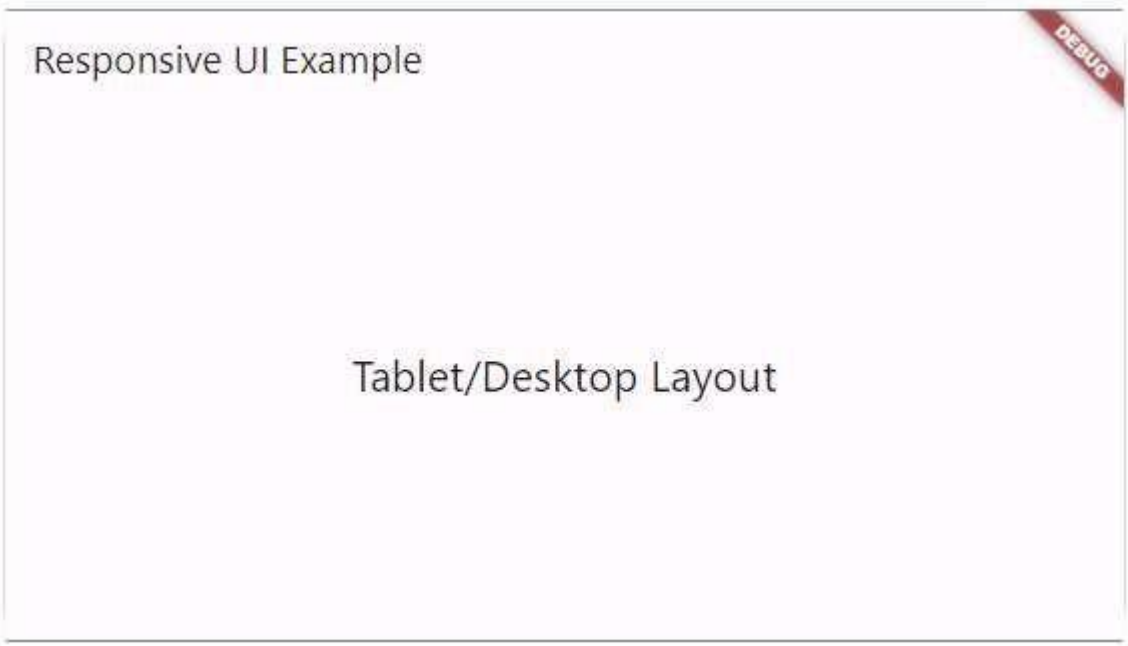


3. a) Design a responsive UI that adapts to different screen sizes.

```
import 'package:flutter/material.dart';
void main() {   runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {     return
MaterialApp(       home:
Scaffold(         appBar: AppBar(
title: Text('Responsive UI Example'),
        ),
        body: ResponsiveLayout(),
      ),
    );
} }
class ResponsiveLayout extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
return LayoutBuilder(       builder: (context,
constraints) {         if
```

```
(constraints.maxWidth < 600) {
return MobileLayout();
        } else {
          return TabletDesktopLayout();
        }
      },
    );
  } }
class MobileLayout extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
return Center(       child: Text(         'Mobile
Layout',         style:
TextStyle(fontSize: 24),
      ),
    );
} }
class TabletDesktopLayout extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
return Center(       child: Text(
'Tablet/Desktop Layout',         style:
TextStyle(fontSize: 24),
      ),
    );
  }
}
```

OUTPUT



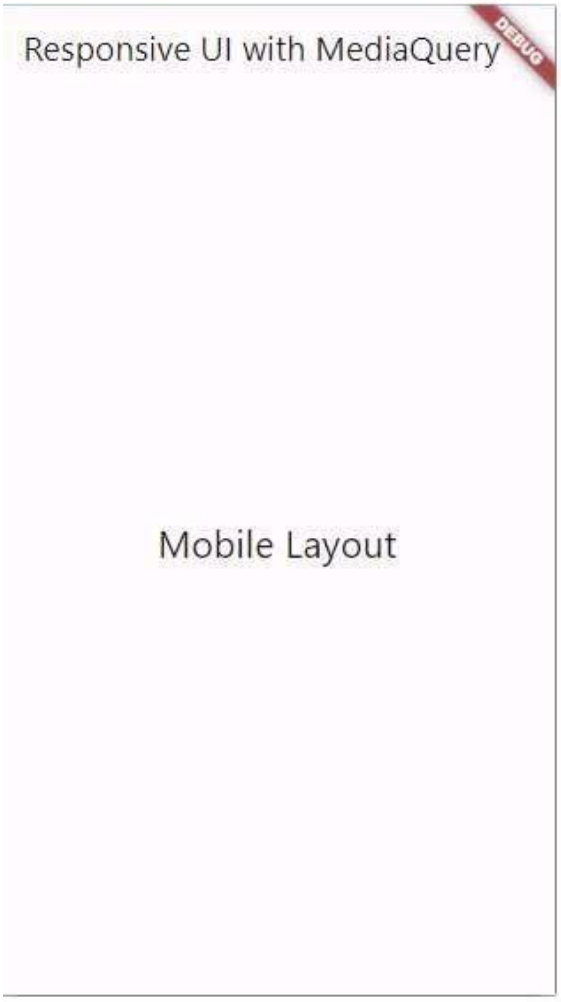

b) Implement media queries and breakpoints for responsiveness.

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

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

}

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {     return

MaterialApp(       home:

Scaffold(         appBar: AppBar(           title:

Text('Responsive UI with MediaQuery'),

        ),

        body: ResponsiveLayout(),

      ),

    );

} }

class ResponsiveLayout extends StatelessWidget {

  @override
```

```
  Widget build(BuildContext context) {      var
screenWidth = MediaQuery.of(context).size.width;     if
(screenWidth < 600) {       return MobileLayout();     }
else if (screenWidth < 1200) {       return TabletLayout();
    } else {        return
DesktopLayout();
    }
  }
}
class MobileLayout extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
return Center(       child: Text(         'Mobile
Layout',         style:
TextStyle(fontSize: 24),
      ),
    );
} }
class TabletLayout extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
return Center(       child: Text(         'Tablet
Layout',         style: TextStyle(fontSize:
24),
      ),
    );
} }
class DesktopLayout extends StatelessWidget {
```

```
  @override
  Widget build(BuildContext context) {
return Center(       child: Text(         'Desktop
Layout',         style: TextStyle(fontSize: 24),

      ),
    );
  }
}
```

OUTPUT

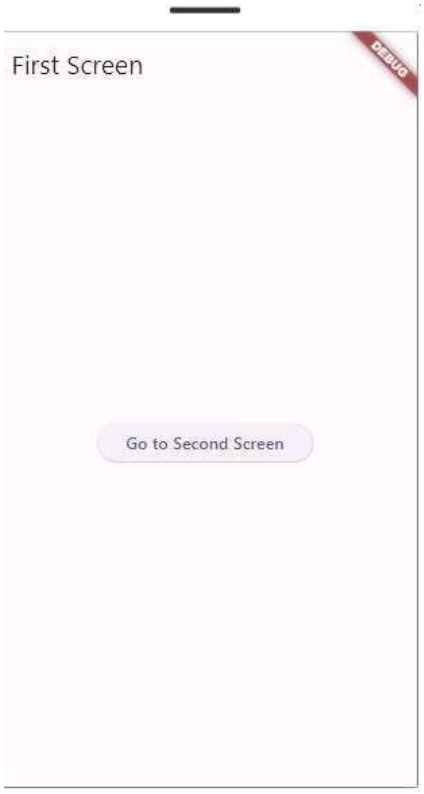


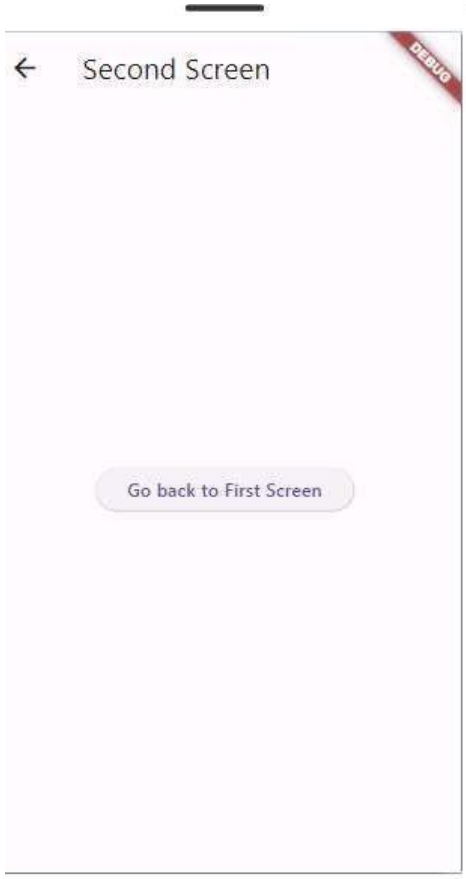
4. a) Set up navigation between different screens using Navigator.

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

```
        ),
      ),
    );
} }
class SecondScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
return Scaffold(         appBar: AppBar(
title: Text('Second Screen'),
      ),
      body: Center(         child:
ElevatedButton(          onPressed:
() {
            Navigator.pop(context);
          },
          child: Text('Go back to First Screen'),
        ),
      ),
    );
  }
}
```

OUTPUT


First Screen
DEBUG
Go to Second Screen


Second Screen
DEBUG
Go back to First Screen

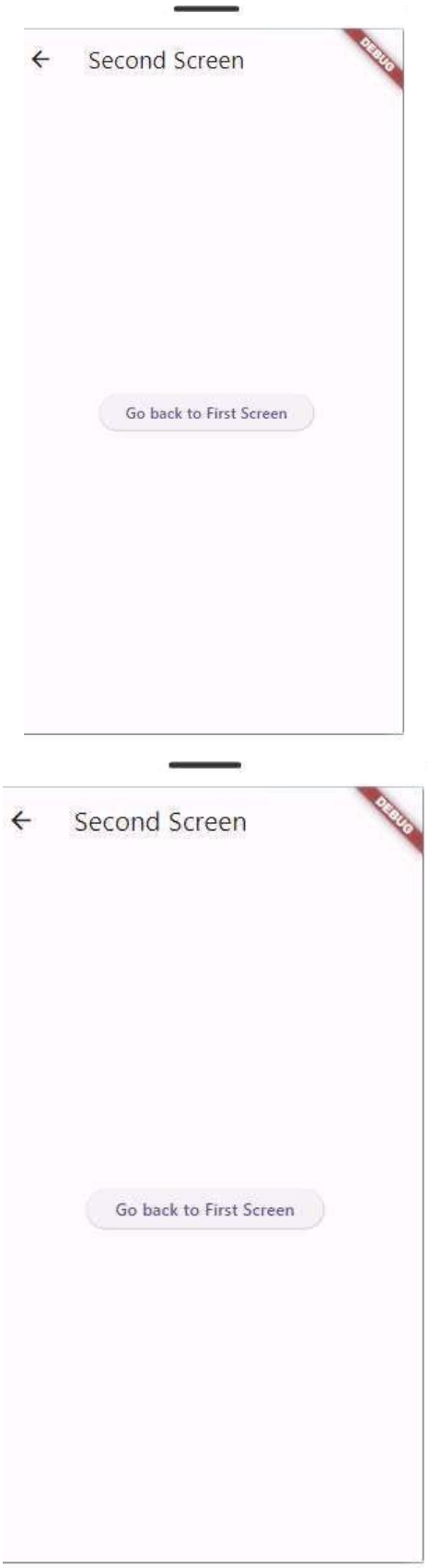
b) Implement navigation with named routes.

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Named Routes Example',
      // Define the routes
      routes: {
        '/': (context) => FirstScreen(),
        '/second': (context) => SecondScreen(),
      },
      initialRoute: '/',
    );
  }
}

class FirstScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('First Screen'),
      ),
```

```
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pushNamed(context, '/second');
          },
          child: Text('Go to Second Screen'),
        ),
      ),
    );
  }
}

class SecondScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Second Screen'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: Text('Go back to First Screen'),
        ),
      ),
    );
  }
```

}

OUTPUT







5. a) Learn about stateful and stateless widgets.

Stateless Widget import

```
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('Stateless Widget Example'),
        ),
        body: Center(
          child: MyStatelessWidget(),
        ),
      ),
    );
  }
}

class MyStatelessWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Text(
```

```
      'Hello, I am a Stateless Widget!',
      style: TextStyle(fontSize: 24),
    );
  }
}
```

OUTPUT


Stateful Widget import

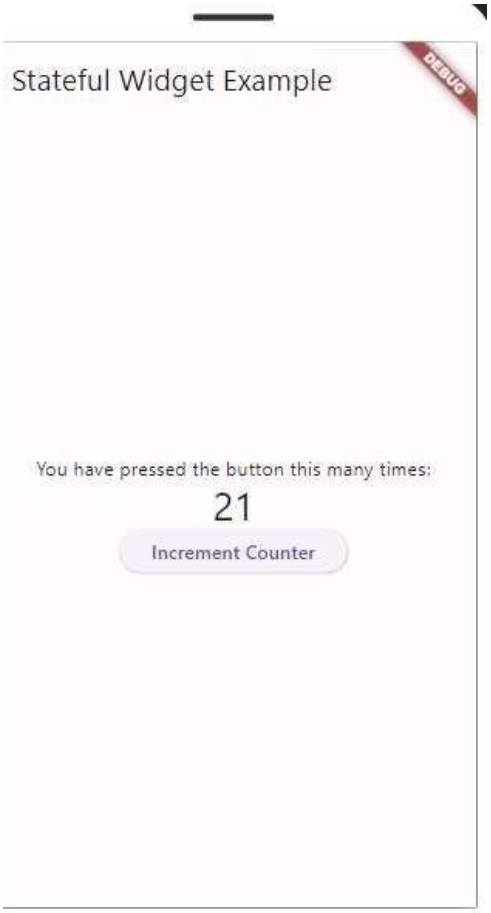
```
import 'package:flutter/material.dart';
```

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

class MyApp extends
StatelessWidget {
  @override
  Widget build(BuildContext context)
{
    return MaterialApp(

debugShowCheckedModeBanner:
false, // Disable debug banner
      home: Scaffold(
        appBar: AppBar(
          title: Text('Stateful Widget
Example'),
        ),
        body: Center(
          child: MyStatefulWidget(),
        ),
      ),
    );
  }
}
```

```
class MyStatefulWidget extends
StatefulWidget {
  @override
  _MyStatefulWidgetState
createState() =>
_MyStatefulWidgetState();
}

class _MyStatefulWidgetState
extends State<MyStatefulWidget> {
  int _counter = 0; // Counter variable

  void _incrementCounter() {
    setState(() {
      _counter++; // Increment the
counter and update the UI
    });
  }

  @override
  Widget build(BuildContext context)
{
    return Column(
      mainAxisAlignment:
MainAxisAlignment.center,
      children: <Widget>[
        Text(
```

```
            'You have pressed the button
this many times:',
            textAlign: TextAlign.center,
          ),
          Text(
            '$_counter', // Display the
counter value
            style:
Theme.of(context).textTheme.headli
neMedium, // Updated style
          ),
          ElevatedButton(
            onPressed:
_incrementCounter, // Call the
increment function
            child: Text('Increment
Counter'),
          ),
        ],
      );
    }
}
```

OUTPUT




b) Implement state management using set State and Provider.

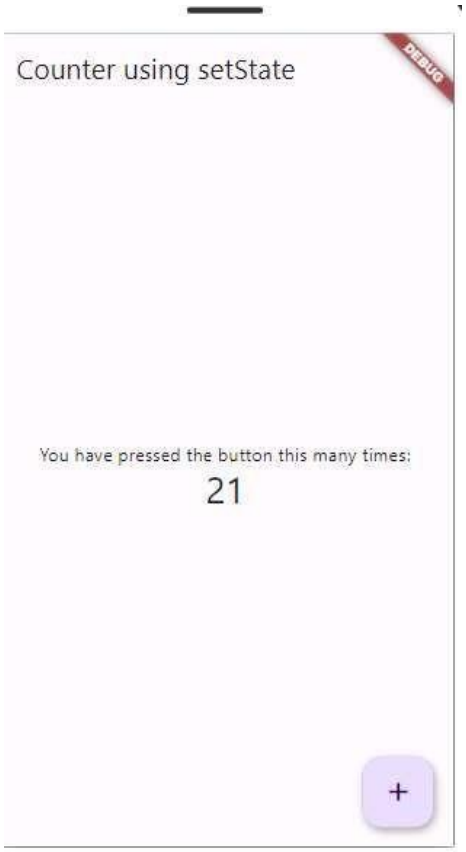
Using set State  import

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
return MaterialApp(
      debugShowCheckedModeBanner: false, // Disable the debug banner
home: CounterScreen(),
    );
  }
}

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

class _CounterScreenState extends State<CounterScreen> {
int _counter = 0; // Counter variable

  void _incrementCounter() {
setState(() {
```

```
    _counter++; // Increment the counter
   });
  }
  @override
  Widget build(BuildContext context) {
return Scaffold(       appBar: AppBar(
       title: Text('Counter using setState'), // App bar title
     ),
     body: Center(
child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
children: <Widget>[
          Text(
            'You have pressed the button this many times:',
style: TextStyle(fontSize: 20), // Add some styling to the text
          ),
          Text(
            '$_counter', // Display the counter value
            style: Theme.of(context).textTheme.headlineMedium, // Use
headlineMedium
          ),
        ],
       ),
     ),
     floatingActionButton: FloatingActionButton(
onPressed: _incrementCounter, // Call increment function
tooltip: 'Increment',
       child: Icon(Icons.add), // Icon for the button
```

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

OUTPUT


Counter using setState
DEBUG
You have pressed the button this many times:
21
+

Using Provider

pubsec.yaml

```
dependencies:   flutter:
sdk: flutter   provider:
^6.0.0
```

main.dart import

```
'package:flutter/material.dart'; import
'package:provider/provider.dart'; void main()
{   runApp(     ChangeNotifierProvider(
create:
(context) => Counter(),       child:
MyApp(),
    ),
  );
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
return MaterialApp(       home:
CounterScreen(),
    );
} }
class Counter with ChangeNotifier {
int _count = 0;   int get count =>
_count;   void increment() {
   _count++;
notifyListeners();
  } }
```

```
class CounterScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {     final counter
=         Provider.of<Counter>(context);
return Scaffold(       appBar: AppBar(
        title: Text('Counter using Provider'),
      ),
      body: Center(
child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
children: <Widget>[
            Text(
              'You have pressed the button this many times:',
            ),
            Text(
              '${counter.count}',
              style: Theme.of(context).textTheme.headline4,
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(         onPressed:
counter.increment,         tooltip:
'Increment',         child: Icon(Icons.add),
      ),
    );
  }
}
```

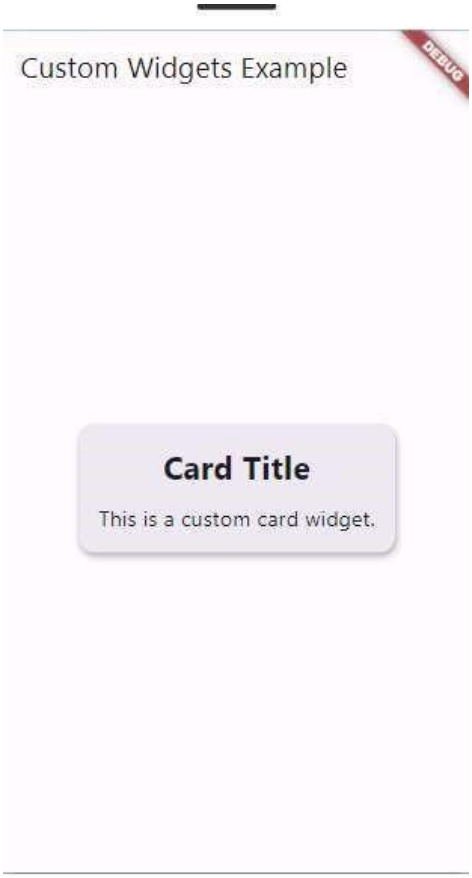
6. a) Create custom widgets for specific UI elements.

Custom Card Widget 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('Custom Widgets Example'),
        ),
        body: Center(
child: CustomCard(
title: 'Card Title',
            description: 'This is a custom card widget.',
          ),
        ),
      ),
    );
  }
}

class CustomCard extends StatelessWidget {
final String title;   final String description;
```

```
  // Constructor with required fields
  CustomCard({required this.title, required this.description});

  @override
  Widget build(BuildContext context) {
return Card(       elevation: 5,
margin: EdgeInsets.all(10),
     child: Padding(
padding: EdgeInsets.all(15),
child: Column(
         mainAxisSize: MainAxisSize.min,
children: <Widget>[             Text(
title,
             style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
           ),
           SizedBox(height: 10),
Text(               description,
             style: TextStyle(fontSize: 16),
           ),
         ],
       ),
     ),
   );
  }
}
```

OUTPUT




b) Apply styling using themes and custom styles.

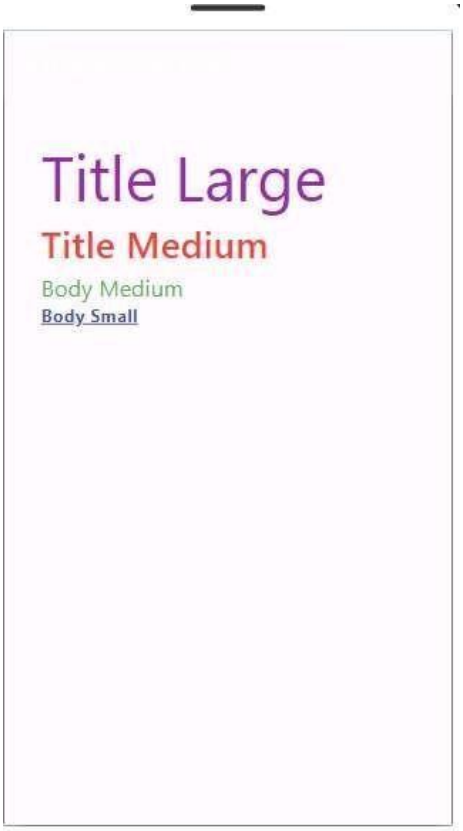
```
import 'package:flutter/material.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      // Hide the debug banner
      debugShowCheckedModeBanner: false,
      title: 'KindaCode.com',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        // Override some of the default text styles
        textTheme: const TextTheme(
          titleLarge: TextStyle(fontSize: 50, color: Colors.purple),
          titleMedium: TextStyle(fontSize: 30, color: Colors.red),
          titleSmall: TextStyle(fontSize: 24, color: Colors.white),
          bodyMedium: TextStyle(fontSize: 18, color: Colors.green),
          bodySmall: TextStyle(
            fontSize: 14,
            color: Colors.indigo,
            fontWeight: FontWeight.bold,
            decoration: TextDecoration.underline,
          ),
        ),
      ),
      home: const MyHomePage(),
    );
  }
```

```
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(
          'KindaCode.com',
          style: Theme.of(context).textTheme.titleSmall,
        ),
      ),
      body: Padding(
        padding: const EdgeInsets.all(30),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              'Title Large',
              style: Theme.of(context).textTheme.titleLarge,
            ),
            Text(
              'Title Medium',
              style: Theme.of(context).textTheme.titleMedium,
            ),
            Text(
```

```
            'Body Medium',
            style: Theme.of(context).textTheme.bodyMedium,
          ),
          Text(
            'Body Small',
            style: Theme.of(context).textTheme.bodySmall,
          ),
        ],
      ),
    ),
  );
 }
}
```

OUTPUT




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

```
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('Form Example'),
        ),
        body: MyCustomForm(),
      ),
```

```
    );
  } }
class MyCustomForm extends StatefulWidget {
  @override
  _MyCustomFormState createState() => _MyCustomFormState();
}
class _MyCustomFormState extends State<MyCustomForm> {
final _formKey = GlobalKey<FormState>();
  @override
  Widget build(BuildContext context) {
return Padding(
     padding: const EdgeInsets.all(16.0),
child: Form(         key: _formKey,
child: ListView(           children:
<Widget>[            TextFormField(
decoration: InputDecoration(
labelText: 'Name',                 border:
OutlineInputBorder(),
            ),
            validator: (value) {               if
(value == null || value.isEmpty) {                  return
'Please enter your name';
              }
return null;
            },
          ),
                SizedBox(height: 16),
  TextFormField(              decoration:
InputDecoration(               labelText:
```

```
'Email',                border:
OutlineInputBorder(),
            ),
            validator: (value) {                if
(value == null || value.isEmpty) {
return 'Please enter your email';
              }
return null;
            },
          ),
              SizedBox(height: 16),
  TextFormField(              decoration:
InputDecoration(              labelText:
'Phone Number',              border: OutlineInputBorder(),
            ),
            validator: (value) {
              if (value == null || value.isEmpty) {
return 'Please enter your phone number';
              }
return null;
            },
          ),
              SizedBox(height: 16),
  TextFormField(              decoration:
InputDecoration(              labelText:
'Password',              border:
OutlineInputBorder(),
            ),
```

```
            obscureText: true,
validator: (value) {
              if (value == null || value.isEmpty) {
return 'Please enter your password';
              }
return null;
            },
          ),
          SizedBox(height: 16),
ElevatedButton(              onPressed: () {
if (_formKey.currentState!.validate()) {
                ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(content: Text('Processing Data')),
                );
              }
            },
            child: Text('Submit'),
          ),
        ],
      ),
    ),
  );
 }
}
```

OUTPUT


b) Implement form validation and error handling. import 'package:flutter/material.dart';

```
void main() {   runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {     return
MaterialApp(       home:
Scaffold(         appBar: AppBar(            title:
Text('Form Validation Example'),
        ),
        body: MyCustomForm(),
      ),
```

```
    );
  } }
class MyCustomForm extends StatefulWidget {
  @override
  _MyCustomFormState createState() => _MyCustomFormState();
}
class _MyCustomFormState extends State<MyCustomForm> {
final _formKey = GlobalKey<FormState>();
  @override
  Widget build(BuildContext context) {
return Padding(
      padding: const EdgeInsets.all(16.0),
child: Form(         key: _formKey,
child:  ListView(           children:
<Widget>[             TextFormField(
              decoration: InputDecoration(
labelText: 'Name',                 border:
OutlineInputBorder(),
              ),
              validator: (value) {
                if (value == null || value.isEmpty) {
return 'Please enter your name';
                }
return null;
              },
            ),
            SizedBox(height: 16),
TextFormField(
```

```
            decoration: InputDecoration(
labelText: 'Email',                 border:
OutlineInputBorder(),
            ),
            validator: (value) {
              if (value == null || value.isEmpty) {
return 'Please enter your email';
              } else if (!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(value)) {
return 'Please enter a valid email address';
              }
return null;
            },
          ),
          SizedBox(height: 16),
          TextFormField(
decoration: InputDecoration(
labelText: 'Phone Number',
border: OutlineInputBorder(),
            ),
            validator: (value) {
              if (value == null || value.isEmpty) {
return 'Please enter your phone number';                } else if
(!RegExp(r'^\d{10}$').hasMatch(value)) {                    return
'Please enter a valid 10-digit phone number';
              }
return null;
            },
          ),
```

```
                SizedBox(height: 16),
  TextFormField(              decoration:
InputDecoration(                labelText:
'Password',                 border: OutlineInputBorder(),
            ),
            obscureText: true,
validator: (value) {
              if (value == null || value.isEmpty) {
return 'Please enter your password';                } else
if (value.length < 6) {
                return 'Password must be at least 6 characters long';
              }
return null;
            },
          ),
          SizedBox(height: 16),
ElevatedButton(              onPressed:
() {
              if (_formKey.currentState!.validate()) {
                ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(content: Text('Processing Data')),
                );
              }
            },
            child: Text('Submit'),
          ),
        ],
      ),
```

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

OUTPUT


Form Validation Example
DEBUG
Name
Anonymous
Email
21
Please enter a valid email address
Phone Number
9876543210
Password
.....
Submit

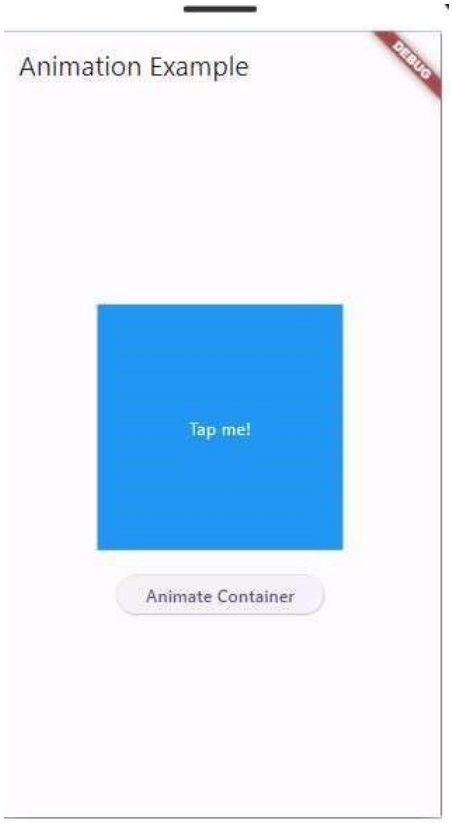
8. a) Add animations to UI elements using Flutter's animation framework. import 'package:flutter/material.dart';

```
void main() {   runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {     return
MaterialApp(       home:
Scaffold(         appBar: AppBar(
title: Text('Animation Example'),
        ),
        body: AnimatedContainerDemo(),
      ),
    );
} }
class AnimatedContainerDemo extends StatefulWidget {
  @override
  _AnimatedContainerDemoState createState() =>
_AnimatedContainerDemoState();
}
class _AnimatedContainerDemoState extends State<AnimatedContainerDemo> {
bool _isExpanded = false;

  void _toggleContainer() {
setState(() {
      _isExpanded = !_isExpanded;
```

```
    });
  }
  @override
  Widget build(BuildContext context) {
return Center(       child: Column(
mainAxisAlignment:
MainAxisAlignment.center,         children:
<Widget>[          AnimatedContainer(
duration: Duration(seconds: 1),             width:
_isExpanded ? 200 : 100,             height: _isExpanded
? 200 : 100,             color: _isExpanded ? Colors.blue :
Colors.red,             child: Center(                child: Text(
'Tap me!',                 style: TextStyle(color:
Colors.white),
              ),
            ),
          ),
               SizedBox(height: 20),
 ElevatedButton(             onPressed:
_toggleContainer,             child: Text('Animate
Container'),
          ),
        ],
      ),
    );
  }
}
```

OUTPUT


Animation Example
DEBUG
Tap me!
Animate Container


Animation Example
DEBUG
Tap me!
Animate Container

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

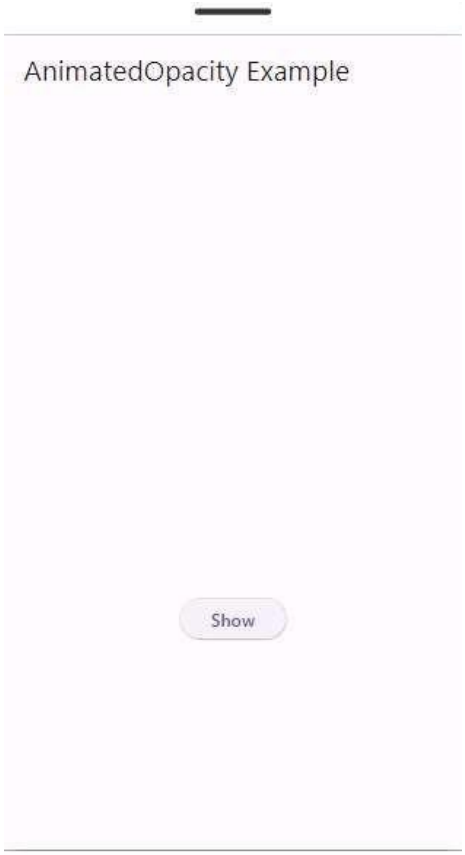
Fade Animation

```
import 'package:flutter/material.dart';
void main() {   runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
return MaterialApp(
     title: 'AnimatedOpacity Example', // App title
theme: ThemeData(
       primarySwatch: Colors.green, // App's primary theme color
     ),
     debugShowCheckedModeBanner: false,       home:
AnimatedOpacityExample(), // Main home page
   );
} }
class AnimatedOpacityExample extends StatefulWidget {
  @override
  _AnimatedOpacityExampleState createState() =>
_AnimatedOpacityExampleState();
}
class _AnimatedOpacityExampleState extends State<AnimatedOpacityExample>
{    bool _isVisible = true; // A variable to control the visibility of the
box
void _toggleVisibility() {     setState(() {
     _isVisible =
```

```
        !_isVisible; // Toggle the visibility when the button is pressed
    });
  }
  @override
  Widget build(BuildContext context) {     return
Scaffold(       appBar: AppBar(
        title: Text('AnimatedOpacity Example'), // AppBar title
      ),
      body: Center(
child: Column(
         mainAxisAlignment: MainAxisAlignment.center,
children: <Widget>[             AnimatedOpacity(
             duration: Duration(seconds: 1), // Animation duration
opacity:
                 _isVisible ? 1.0 : 0.0, // Opacity value based on visibility
child: Container(                 width: 200,                 height: 200,
               color: Colors.blue, // Color of the box
child: Center(                   child: Text(
                  'Fading Box', // Text displayed inside the box
style: TextStyle(
                    color: Colors.white, // Text color
fontSize: 20.0, // Text font size
                  ),
                ),
              ),
            ),
          ),
```

```
            SizedBox(height: 20), // SizedBox for spacing
ElevatedButton(              onPressed:
                  _toggleVisibility, // Call _toggleVisibility when the button is pressed
child: Text(_isVisible ? 'Hide' : 'Show'), // Button text
            ),
          ],
        ),
      ),
    );
  }
}
```

OUTPUT




Slide Animation

```
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('Slide Animation Example'),
        ),
        body: SlideAnimationDemo(),
      ),
    );
  }
```

```
}

class SlideAnimationDemo extends StatefulWidget {
  @override
  _SlideAnimationDemoState createState() => _SlideAnimationDemoState();
}
class _SlideAnimationDemoState extends State<SlideAnimationDemo>
with SingleTickerProviderStateMixin {   late AnimationController
_controller;   late Animation<Offset> _animation;
  @override   void
initState() {
super.initState();
    _controller = AnimationController(
duration: Duration(seconds: 2),       vsync:
this,
    );
    _animation = Tween<Offset>(
begin: Offset(0, 0),       end: Offset(1,
0),
    ).animate(CurvedAnimation(
parent: _controller,       curve:
Curves.easeInOut,
    ));
    _controller.repeat(reverse: true);
  }
  @override   void
dispose() {
_controller.dispose();
super.dispose();
```

```
  }
  @override
  Widget build(BuildContext context) {
return Center(       child:
SlideTransition(         position:
_animation,         child: Container(
width: 100,          height: 100,          color:
Colors.green,          child:
Center(             child: Text(
'Slide Me!',                style: TextStyle(color: Colors.white,
fontSize: 24),
            ),
          ),
        ),
      ),
    );
  }
}
```

OUTPUT

## Week 5

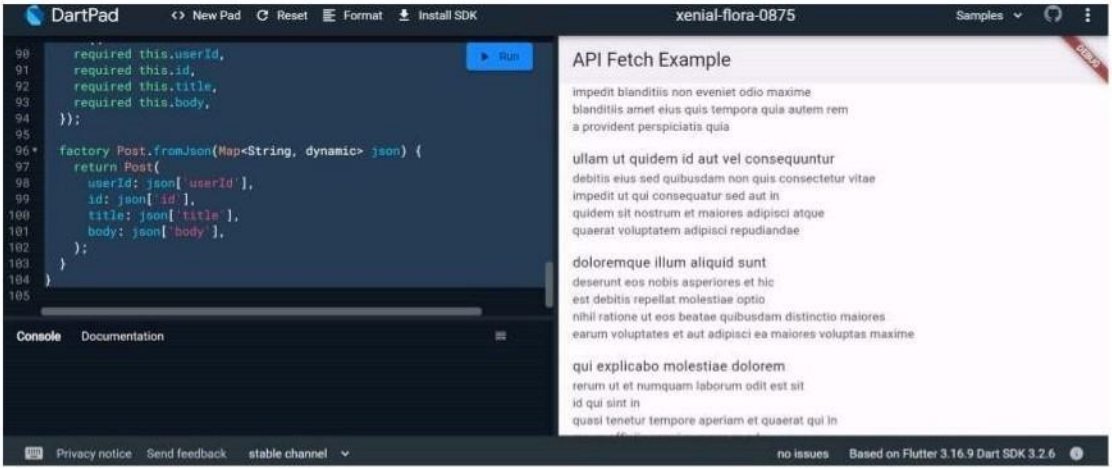
**EXPERIMENT NO: 9.**

**9.a) Fetch data from a REST API. Ans)**

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

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

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

class MyApiFetchWidget extends StatefulWidget {
  @override
  _MyApiFetchWidgetState createState() => _MyApiFetchWidgetState();
}

class _MyApiFetchWidgetState extends State<MyApiFetchWidget> {   late
Future<List<Post>> _posts;

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

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

    if (response.statusCode == 200) {       // If the server returns
a 200 OK response,       // parse the JSON and return a list of
posts.
     List<dynamic> data = json.decode(response.body);
     List<Post> posts = data.map((post) => Post.fromJson(post)).toList();       return posts;
    } else {
     // If the server did not return a 200 OK response,       // throw an
exception.       throw Exception('Failed to load posts');
    }
  }
```

```
  @override
  Widget build(BuildContext context) {
   return Scaffold(       appBar: AppBar(
       title: Text('API Fetch Example'),
     ),
     body: FutureBuilder<List<Post>>(         future:
_posts,         builder: (context, snapshot) {
         if (snapshot.connectionState == ConnectionState.waiting) {
           return CircularProgressIndicator();           } else if
(snapshot.hasError) {            return Text('Error: $
{snapshot.error}');
         } else {
           return ListView.builder(              itemCount:
snapshot.data!.length,              itemBuilder: (context,
index) {
              return ListTile(                  title: Text(snapshot.data!
[index].title),
                subtitle: Text(snapshot.data![index].body),
              );
            },
          );
         }
       },
     ),
   );
  }
}

class Post {   final int userId;
final int id;   final String title;
final String body;

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

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

**Output:**




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

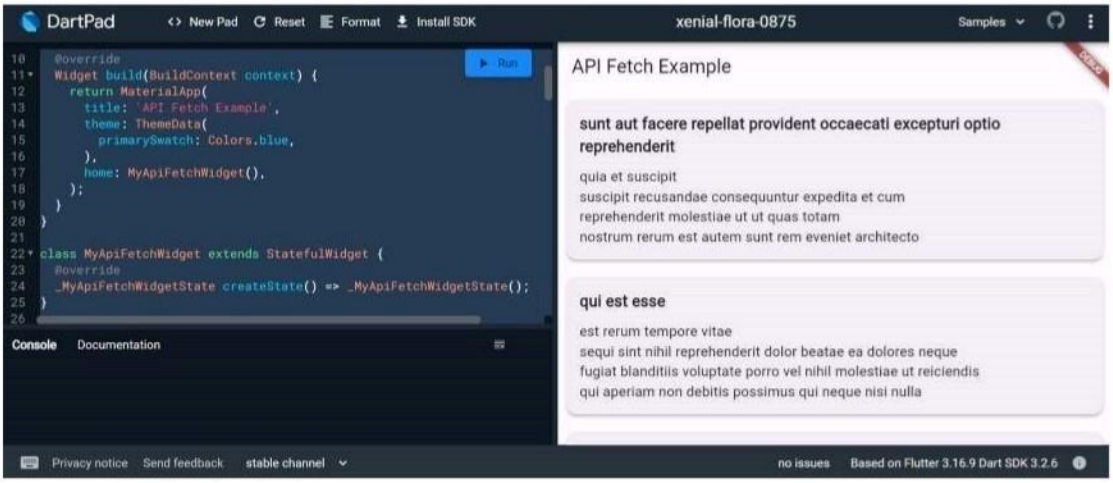
**Ans)**

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

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

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

class MyApiFetchWidget extends StatefulWidget {
  @override
  _MyApiFetchWidgetState createState() => _MyApiFetchWidgetState();
}

class _MyApiFetchWidgetState extends State<MyApiFetchWidget> {   late
Future<List<Post>> _posts;

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

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

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(       appBar: AppBar(
        title: Text('API Fetch Example'),
      ),
      body: FutureBuilder<List<Post>>(         future: _posts,         builder: (context,
snapshot) {           if (snapshot.connectionState == ConnectionState.waiting) {
return Center(child: CircularProgressIndicator());
         } else if (snapshot.hasError) {
           return Center(child: Text('Error: ${snapshot.error}'));
         } else {
           return PostList(posts: snapshot.data!);
         }
        },
      ),
    );
  }
}

class PostList extends StatelessWidget {   final List<Post>
posts;

  PostList({required this.posts});

  @override
  Widget build(BuildContext context) {     return
ListView.builder(       itemCount: posts.length,
itemBuilder: (context, index) {
       return PostItem(post: posts[index]);
      },
    );
  }
}

class PostItem extends StatelessWidget {   final Post post;

  PostItem({required this.post});
```

```
  @override
  Widget build(BuildContext context) {     return Card(
margin: EdgeInsets.all(10),
     elevation: 3,       child:
Padding(         padding: EdgeInsets.all(15),
child: Column(
         crossAxisAlignment: CrossAxisAlignment.start,          children:
[            Text(              post.title,              style:
TextStyle(              fontSize: 18,
               fontWeight: FontWeight.bold,
             ),
           ),
           SizedBox(height: 10),
           Text(              post.body,
style: TextStyle(fontSize: 16),
           ),
         ],
       ),
     ),
   );
  }
}

class Post {   final int userId;
final int id;   final String title;
final String body;

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

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

**Output:**




## EXPERIMENT NO: 10.

### 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 testing 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.

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

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

```
void main() {
  group('Test start, increment, decrement', () {
    test('value should start at 0', () {
      expect(Counter().value, 0);
    });

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

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

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

      expect(counter.value, -1);
    });
  });
}
```

6. Run the tests

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 writing tests because it provides the fastest feedback loop as well as the ability to set breakpoints.

IntelliJ

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 DevTools by clicking on the "Open DevTools" 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.

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

6. Debug Paint:
Enable debug paint to visualize the layout and rendering of widgets. Use the debugPaintSizeEnabled and debugPaintBaselinesEnabled flags.

```
void main() {
  debugPaintSizeEnabled = 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 leaks.
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.