SCIENT
INSTITUTE OF TECHNOLOGY
(UGC AUTONOMOUS)
Ibrahimpatnam , R.R Dist, Telangana 501506

FLUTTER LAB – MANUAL

NAME OF THE FACULTY: N.LAXMAN SIR
DEPT.OF.FACULTY :CSE
ACADEMIC YEAR :2024-2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCIENT INSTITUDE OF TECHNOLOGY (AUTONOMOUS INSTITUDE)
IBRAHIMPTNAM,
RANGAREDDY

INDEX

WEEK-1

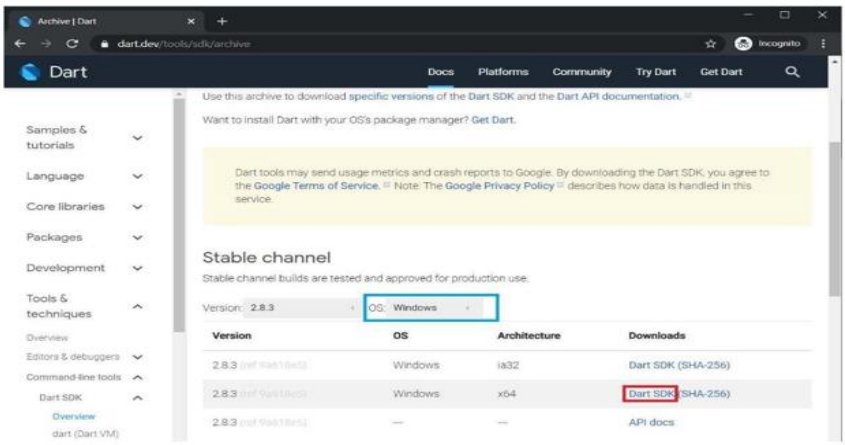
EXPERIMENT NO: 1.

Write code for a simple user registration form for an event.

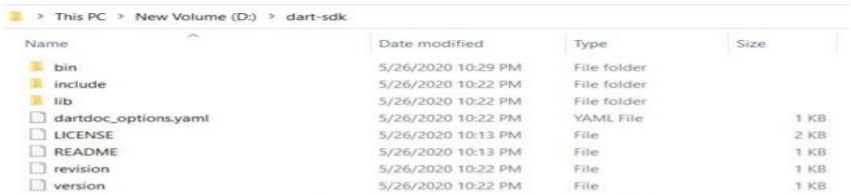
1. a) Install Flutter and Dart SDK.

Dart SDK is a pre-compiled version so we have to download and extract it only. For this follow the below-given instructions: Step 1: Download Dart SDK. Download Dart SDK from the Dart SDK archive page.

The URL is: https://dart.dev/tools/sdk/archive



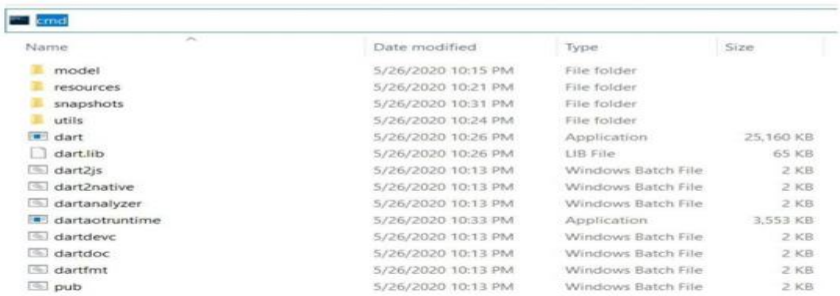
Click on DART SDK to download SDK for Windows 64-Bit Architecture. The download will start and a zip file will be downloaded. **Note:** To download SDK for any other OS select OS of your choice. **Step 2:** Extract the downloaded zip file. Extract the contents of downloaded zip file and after extracting contents of zip file will be as shown:



> This PC > New Volume (D:) > dart-sdk

Name	Date modified	Type	Size
bin	5/26/2020 10:29 PM	File folder	
include	5/26/2020 10:22 PM	File folder	
lib	5/26/2020 10:22 PM	File folder	
dartdoc_options.yaml	5/26/2020 10:22 PM	YAML File	1 KB
LICENSE	5/26/2020 10:13 PM	File	2 KB
README	5/26/2020 10:13 PM	File	1 KB
revision	5/26/2020 10:22 PM	File	1 KB
version	5/26/2020 10:22 PM	File	1 KB

Step 3: Running Dart. Now open bin folder and type "cmd" as given below:



> cmd

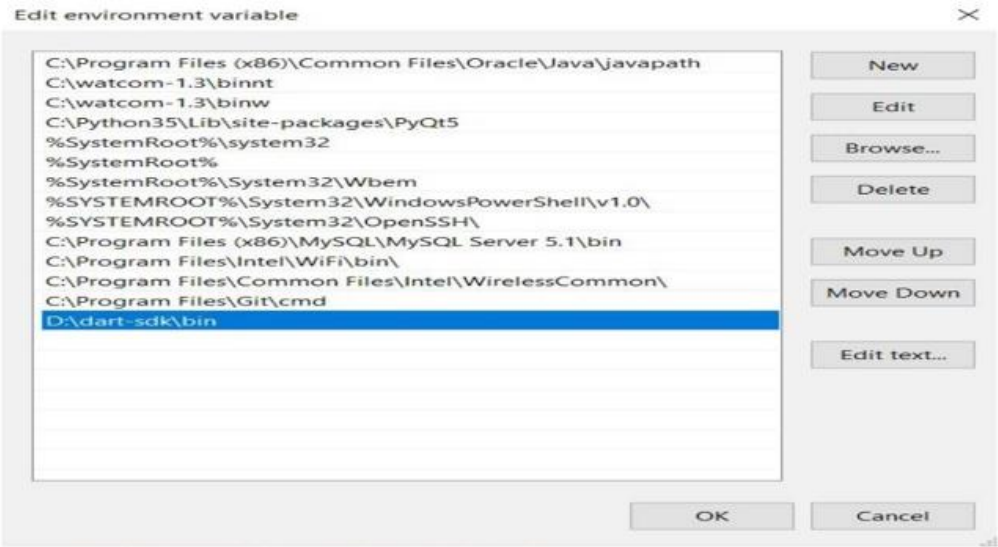
Name	Date modified	Type	Size
model	5/26/2020 10:15 PM	File folder	
resources	5/26/2020 10:21 PM	File folder	
snapshots	5/26/2020 10:31 PM	File folder	
utils	5/26/2020 10:24 PM	File folder	
dart	5/26/2020 10:26 PM	Application	25,160 KB
dart.lib	5/26/2020 10:26 PM	LIB File	65 KB
dart2js	5/26/2020 10:13 PM	Windows Batch File	2 KB
dart2native	5/26/2020 10:13 PM	Windows Batch File	2 KB
dartanalyzer	5/26/2020 10:13 PM	Windows Batch File	2 KB
dartaotruntime	5/26/2020 10:33 PM	Application	3,553 KB
dartdevc	5/26/2020 10:13 PM	Windows Batch File	2 KB
dartdoc	5/26/2020 10:13 PM	Windows Batch File	2 KB
dartfmt	5/26/2020 10:13 PM	Windows Batch File	2 KB
pub	5/26/2020 10:13 PM	Windows Batch File	2 KB

Command Prompt will open with our desired path of bin folder and now type dart".

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18363.778]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\dart-sdk\bin>dart
Usage: dart [<vm-flags>] <dart-script-file> [<script-arguments>]

Executes the Dart script <dart-script-file> with the given list of <script-arguments>.

Common VM flags:
--enable-asserts
  Enable assert statements.
--help or -h
  Display this message (add -v or --verbose for information about
  all VM options).
--package-root=<path> or -p<path>
  Where to find packages, that is, "package:..." imports.
--packages=<path>
  Where to find a package spec file.
--observe[=<port>[/<bind-address>]]
  The observe flag is a convenience flag used to run a program with a
  set of options which are often useful for debugging under Observatory.
  These options are currently:
      --enable-vm-service[=<port>[/<bind-address>]]
      --pause-isolates-on-exit
      --pause-isolates-on-unhandled-exceptions
      --warn-on-pause-with-no-debugger
  This set is subject to change.
  Please see these options (--help --verbose) for further documentation.
--write-service-info=<file_name>
  Outputs information necessary to connect to the VM service to the
  specified file in JSON format. Useful for clients which are unable to
  listen to stdout for the Observatory listening message.
--snapshot-kind=<snapshot_kind>
--snapshot=<file_name>
  These snapshot options are used to generate a snapshot of the loaded
  Dart script:
    <snapshot-kind> controls the kind of snapshot, it could be
                    kernel(default) or app-jit
    <file_name> specifies the file into which the snapshot is written
--version
  Print the VM version.

D:\dart-sdk\bin>
```

And now we are ready to use dart through bin folder but setting up the path in environment variables will ease our task of Step3 and we can run dart from anywhere in the file system using command prompt.

Step 4: Setting up path in environment variables. Open Environment Variables from advanced system settings and add Path in System Variables as depicted in image:



Now we are done to use Dart from anywhere in the file system.

Step 5: Run Dart Using cmd

```
Select C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18363.778]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users>dart
Usage: dart [<vm-flags>] <dart-script-file> [<script-arguments>]

Executes the Dart script <dart-script-file> with the given list of <script-arguments>.

Common VM flags:
--enable-asserts
  Enable assert statements.
--help or -h
  Display this message (add -v or --verbose for information about
  all VM options).
--package-root=<path> or -p<path>
  Where to find packages, that is, "package:..." imports.
--packages=<path>
  Where to find a package spec file.
--observe[=<port>[/<bind-address>]]
  The observe flag is a convenience flag used to run a program with a
  set of options which are often useful for debugging under Observatory.
  These options are currently:
      --enable-vm-service[=<port>[/<bind-address>]]
      --pause-isolates-on-exit
      --pause-isolates-on-unhandled-exceptions
      --warn-on-pause-with-no-debugger
  This set is subject to change.
  Please see these options (--help --verbose) for further documentation.
--write-service-info=<file_name>
  Outputs information necessary to connect to the VM service to the
  specified file in JSON format. Useful for clients which are unable to
  listen to stdout for the Observatory listening message.
--snapshot-kind=<snapshot_kind>
--snapshot=<file_name>
  These snapshot options are used to generate a snapshot of the loaded
  Dart script:
    <snapshot-kind> controls the kind of snapshot, it could be
                    kernel(default) or app-jit
    <file_name> specifies the file into which the snapshot is written
--version
  Print the VM version.

C:\Users>_
```

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

PROGRAM:

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

OUTPUT:

Full name is John Doe

PROGRAM:

```
void main() {
 int num1 = 10; // declaring number1
```

```
  int num2 = 3;  // declaring number2

  // Calculations
  int sum = num1 + num2;
  int diff = num1 - num2;
  int mul = num1 * num2;
  double div = num1 / num2; // It is double because it outputs a number with decimals.

  // Displaying the output
  print("The sum is $sum");
  print("The difference is $diff");
  print("The product is $mul");
  print("The division is $div");
}
```

Output:

The sum is 13

The difference is 7

The product is 30

The division is 3.3333333333333335

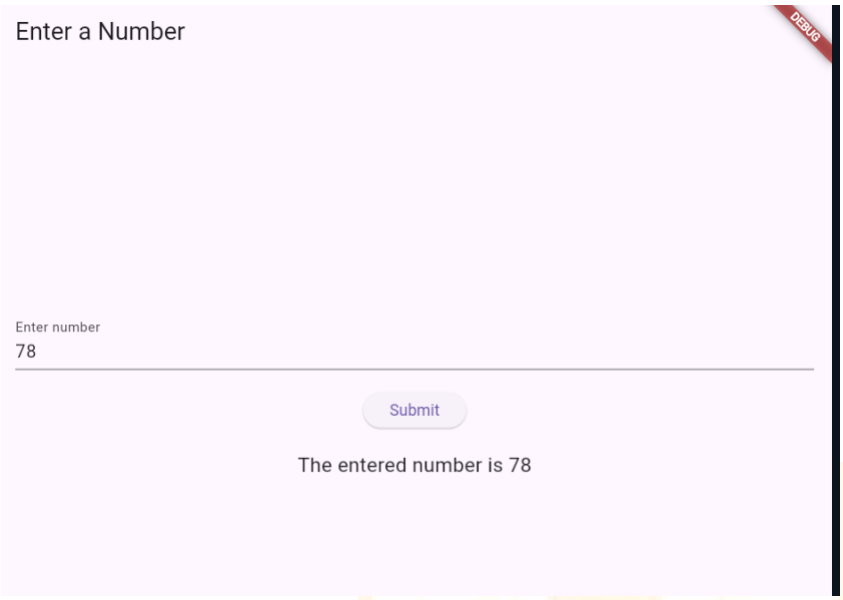
PROGRAM:

```
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('Enter a Number')),
      body: NumberInputWidget(),
    ),
  );
 }
}

class NumberInputWidget extends StatefulWidget {
 @override
 _NumberInputWidgetState createState() => _NumberInputWidgetState();
}

class _NumberInputWidgetState extends State<NumberInputWidget> {
 final TextEditingController _controller = TextEditingController();
 String? _displayText;

 void _showNumber() {
  setState(() {
   int? number = int.tryParse(_controller.text);
   _displayText = number != null
     ? "The entered number is $number"
     : "Please enter a valid number";
  });
 }

 @override
 Widget build(BuildContext context) {
  return Padding(
   padding: const EdgeInsets.all(16.0),
   child: Column(
```

```
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        TextField(
          controller: _controller,
          keyboardType: TextInputType.number,
          decoration: InputDecoration(labelText: 'Enter number'),
        ),
        SizedBox(height: 20),
        ElevatedButton(
          onPressed: _showNumber,
          child: Text('Submit'),
        ),
        SizedBox(height: 20),
        if (_displayText != null)
          Text(
            _displayText!,
            style: TextStyle(fontSize: 18),
          ),
      ],
    ),
  );
}
}
```


<u>OUTPUT:</u>



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

TEXT WIDGET:

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

// Function to trigger the build process
void main() => runApp(const GeeksforGeeks());

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

 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   home: Scaffold(
    backgroundColor: Colors.green,
    appBar: AppBar(
     backgroundColor: Colors.red,
     title: const Text("Welcome  to scient Flutter lab Cse-B"),
    ),
```

```
      // AppBar
      body: Container(
        child: const Center(
          child: Text("Hello Every One this is Varun's  flutter Notes"),
        ),
      ),
      // Container
    ),
    // Scaffold
  );
  // MaterialApp
}
}
```

OUTPUT:

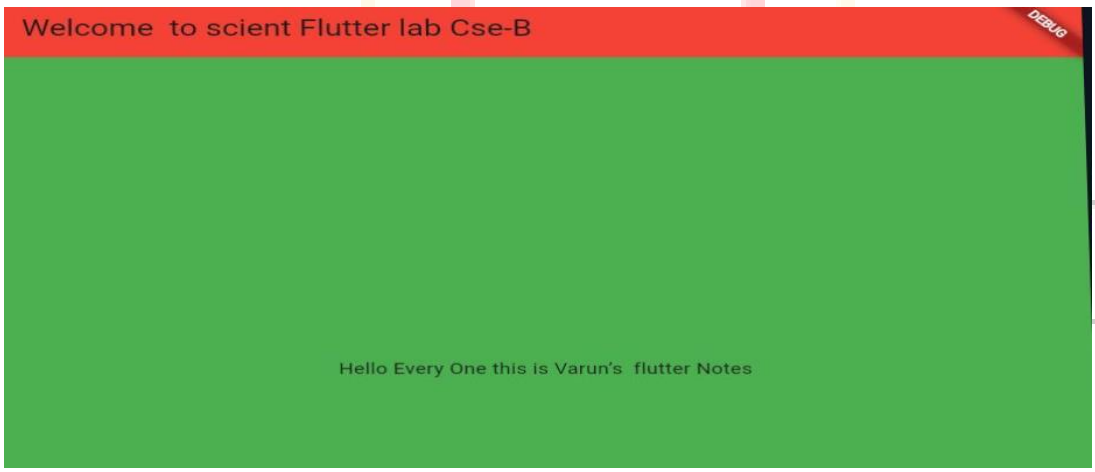


IMAGE WIDGET:

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

// Function to start app building
void main() => runApp(const MyApp());
```

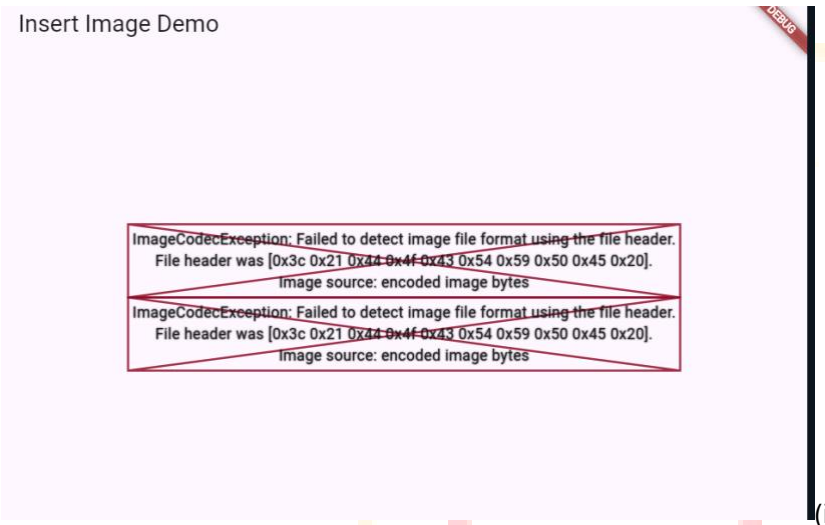
```
class MyApp extends StatelessWidget {
 const MyApp({Key? key}) : super(key: key);

 // This widget is the root of your application
 @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>[
       Image.asset(
        'assets/images/output.gif',
        height: 200,
        scale: 2.5,
        color: const Color.fromARGB(255, 15, 147, 59),
        colorBlendMode: BlendMode.modulate,
        opacity: const AlwaysStoppedAnimation<double>(0.5),
       ), // Image.asset 1
       Image.asset(
        'assets/images/geeksforgeeks.jpg',
        height: 400,
        width: 400,
       ), // Image.asset 2
      ],
     ), // Column
    ), // Center
```

```
    ), // Scaffold
   ); // MaterialApp
  }
}
```

NOTE:

Insert image location path in where needed

Hint: insert image path in line no 22,30 change its height and width according to your image



(its not an accurate outputfile)

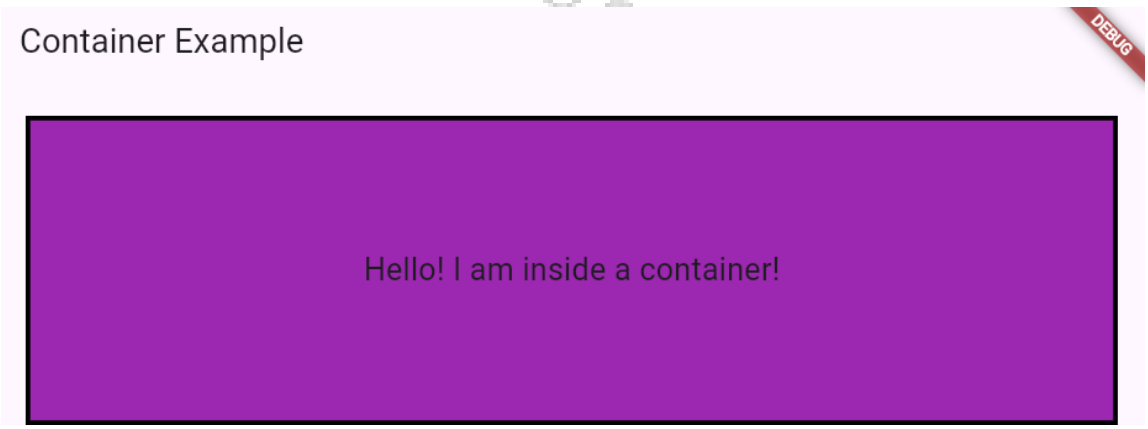
CONTAINER WIDGET:

PROGRAM:

```
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,
        alignment: Alignment.center,
        margin: const EdgeInsets.all(20),
        padding: const EdgeInsets.all(30),
        decoration: BoxDecoration(
          color: Colors.purple,
          border: Border.all(color: Colors.black, width: 3),
        ),
        child: const Text(
          "Hello! I am inside a container!",
          style: TextStyle(fontSize: 20),
        ),
      ), // Container
    ), // Scaffold
  ); // MaterialApp
 }
}
```

OUTPUT:



2b) Implement different layout structures using Row, Column, and stack widgets Row Widget?

Row widget:

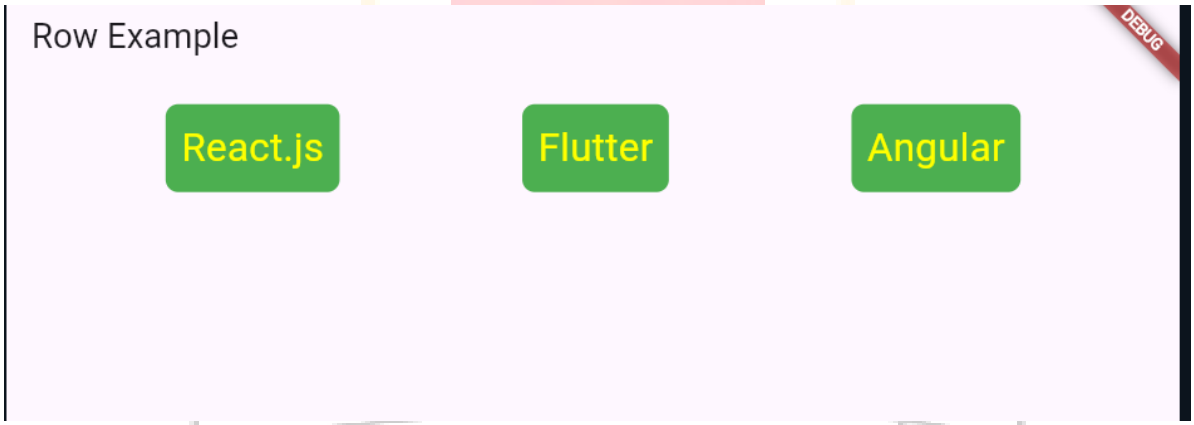
PROGRAM:

```
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: MyHomePage(),
    );
  }
}


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


class MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Row Example"),
      ),
```

```
body: Row(
 mainAxisAlignment: MainAxisAlignment.spaceEvenly,
 children: <Widget>[
  Container(
   padding: const EdgeInsets.all(10),
   decoration: BoxDecoration(
    color: Colors.green,
    borderRadius: BorderRadius.circular(8),
   ),
   child: const Text(
    "React.js",
    style: TextStyle(color: Colors.yellowAccent, fontSize: 25),
   ),
  ),
  Container(
   padding: const EdgeInsets.all(10),
   margin: const EdgeInsets.all(15),
   decoration: BoxDecoration(
    color: Colors.green,
    borderRadius: BorderRadius.circular(8),
   ),
   child: const Text(
    "Flutter",
    style: TextStyle(color: Colors.yellowAccent, fontSize: 25),
   ),
  ),
  Container(
   padding: const EdgeInsets.all(10),
   decoration: BoxDecoration(
    color: Colors.green,
    borderRadius: BorderRadius.circular(8),
```

```
        ),
        child: const Text(
          "Angular",
          style: TextStyle(color: Colors.yellowAccent, fontSize: 25),
        ),
      ),
    ],
  ),
 );
}
}
```

OUTPUT:



Column widget:

PROGRAM:

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

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

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

```
  return MaterialApp(
    home: MyHomePage(),
  );
 }
}

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

class MyHomePageState extends State<MyHomePage> {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: const Text("Flutter Column Example"),
   ),
   body: Column(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: <Widget>[
     Container(
      padding: const EdgeInsets.all(12.0),
      margin: const EdgeInsets.all(20.0),
      decoration: BoxDecoration(
       borderRadius: BorderRadius.circular(8),
       color: Colors.red,
      ),
      child: const Text(
       "React.js",
       style: TextStyle(color: Colors.yellowAccent, fontSize: 20),
```

```
      ),
    ),
    Container(
      padding: const EdgeInsets.all(12.0),
      margin: const EdgeInsets.all(20.0),
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(8),
        color: Colors.red,
      ),
      child: const Text(
        "Flutter",
        style: TextStyle(color: Colors.yellowAccent, fontSize: 20),
      ),
    ),
    Container(
      padding: const EdgeInsets.all(12.0),
      margin: const EdgeInsets.all(20.0),
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(8),
        color: Colors.red,
      ),
      child: const Text(
        "MySQL",
        style: TextStyle(color: Colors.yellowAccent, fontSize: 20),
      ),
    ),
  ],
 ),
);
}
}
```

OUTPUT:

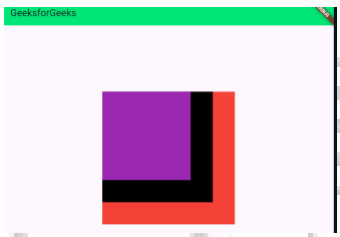
Stack Widget:

PROGRAM:

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


void main() {
 runApp(MaterialApp(
  home: Scaffold(
   appBar: AppBar(
    title: const Text('GeeksforGeeks'),
    backgroundColor: Colors.greenAccent[400],
   ), // AppBar
   body: Center(
    child: SizedBox(
     width: 300,
     height: 300,
      child: Stack(
       children: <Widget>[
        Container(
         width: 300,
         height: 300,
         color: Colors.red,
        ), // Container
        Container(
         width: 250,
```

```
            height: 250,
            color: Colors.black,
          ), // Container
          Container(
            width: 200,
            height: 200,
            color: Colors.purple,
          ), // Container
        ], // <Widget>[]
      ), // Stack
    ), // SizedBox
  ), // Center
 ), // Scaffold
 )); // MaterialApp
}
```

OUTPUT:



WEEK – 2

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

PROGRAM:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
```

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

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

    <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>

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

  <!-- Bootstrap JS and dependencies (jQuery) -->
  <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: (if no output in android studio may be you have to execute in vs code)

3b) Implement medio queries and breakpoints for responsiveness.?

PROGRAM:

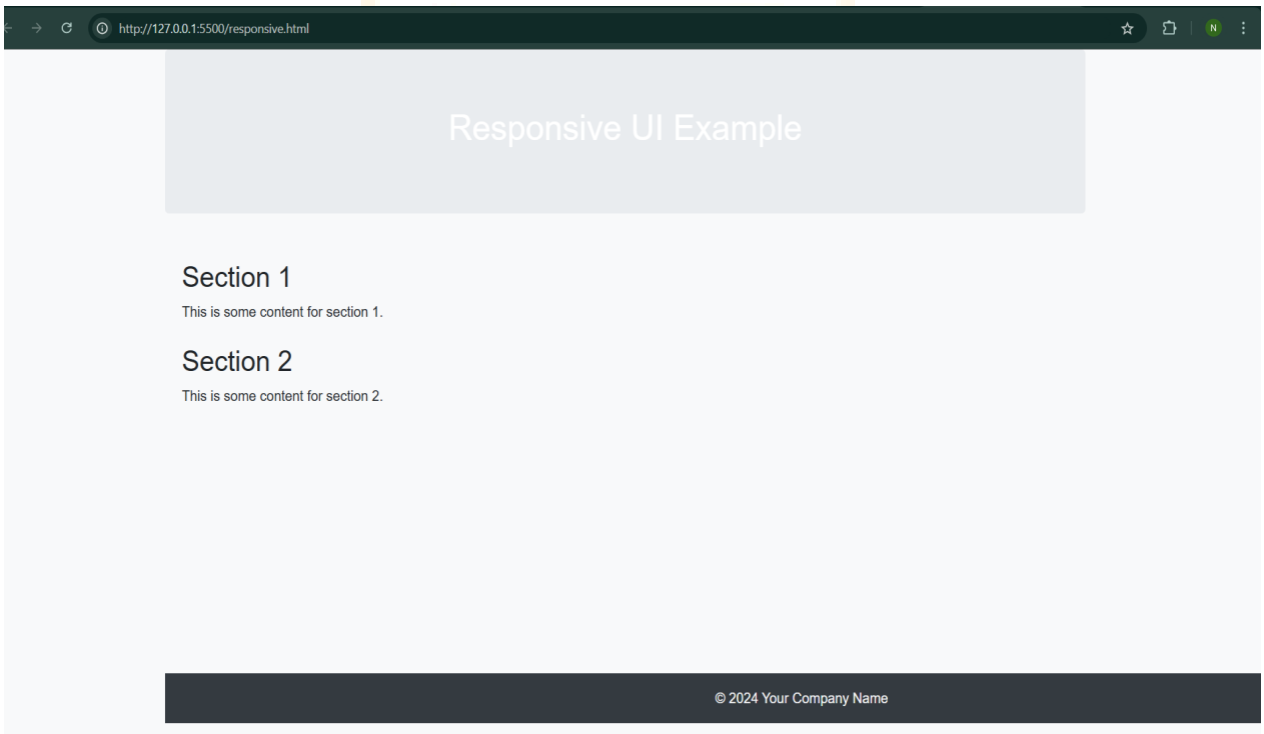
```
<!DOCTYPE html>

<html lang="en">


<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

  <title>Responsive UI Example</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      margin: 0;

      padding: 0;
```

```
  background-color: #f8f9fa; /* light background */
}

header {
  background-color: #333;
  color: white;
  text-align: center;
  padding: 10px;
}

main {
  max-width: 1200px;
  margin: 0 auto;
  padding: 20px;
}

footer {
  margin-bottom: 20px;
  background-color: #333;
  color: white;
  text-align: center;
  padding: 10px;
  position: fixed;
  bottom: 0;
  width: 100%;
}

@media only screen and (max-width: 768px) {
  main {
    padding: 10px;
  }
```

```
        footer {
          position: static;
        }
      }
    </style>
  </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 (jQuery) -->
  <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:



4a) Set up navigation between different screens using navigator.?

PROGRAM:

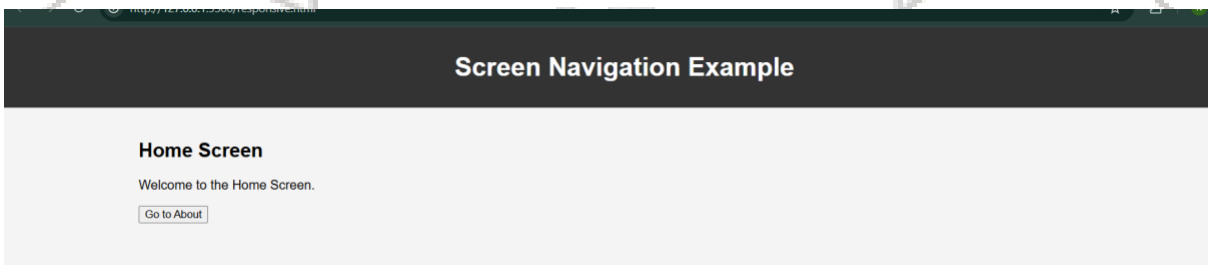
```
<!DOCTYPE html>
<html lang="en">


<head>
```

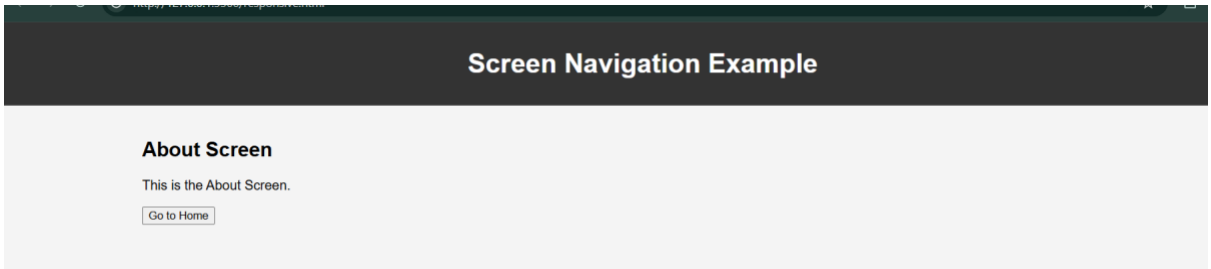
```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Screen Navigation Example</title>
<style>
  body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
  }

  header {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 10px;
  }

  main {
    max-width: 1200px;
    margin: 0 auto;
    padding: 20px;
  }

  section {
    display: none;
  }

  footer {
    background-color: #333;
    color: #fff;
```

```
      text-align: center;
      padding: 10px;
      position: fixed;
      bottom: 0;
      width: 100%;
    }

    .active {
      display: block;
    }
  </style>
</head>

<body>

  <header class="jumbotron text-center">
    <h1>Screen Navigation Example</h1>
  </header>

  <main>
    <section id="home" class="active">
      <h2>Home Screen</h2>
      <p>Welcome to the Home Screen.</p>
      <button onclick="navigateTo('about')">Go to About</button>
    </section>

    <section id="about">
      <h2>About Screen</h2>
      <p>This is the About Screen.</p>
      <button onclick="navigateTo('home')">Go to Home</button>
    </section>
```

```
  </main>

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

  <script>
    function navigateTo(screenId) {
      // Hide all sections
      document.querySelectorAll('section').forEach(section => {
        section.classList.remove('active');
      });

      // Show the selected section
      document.getElementById(screenId).classList.add('active');
    }
  </script>
</body>

</html>
```

OUTPUT:





4b) Implement navigation with named routers.?

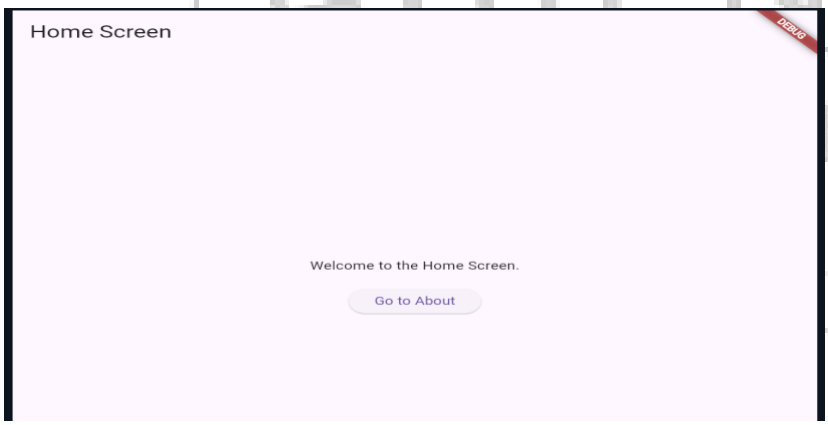
PROGRAM:

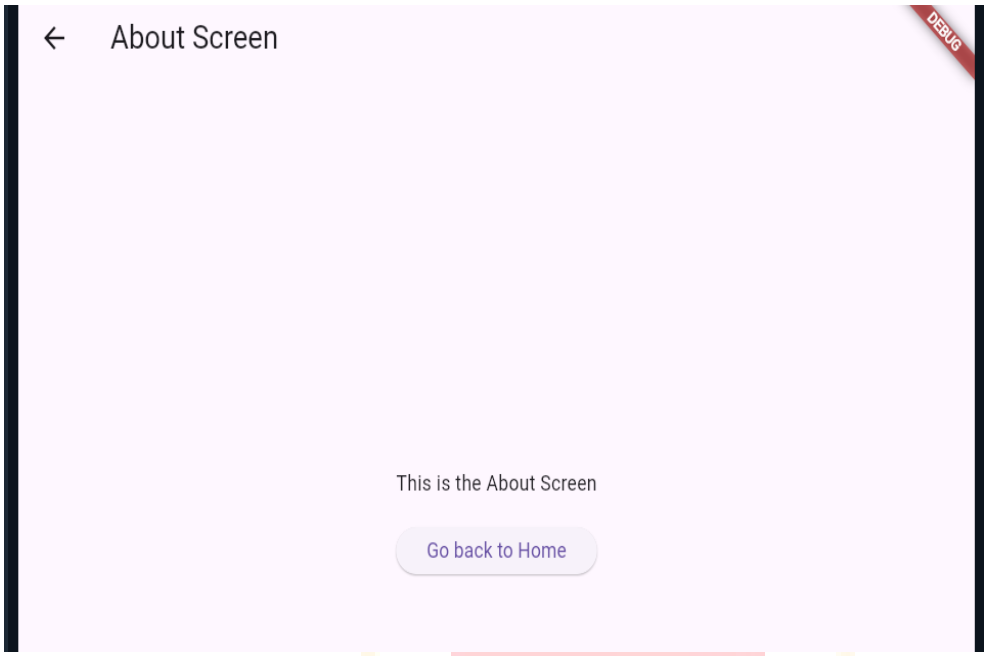
```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Named Routes Navigation Example',
      home: HomeScreen(),
      routes: {
        '/about': (context) => AboutScreen(),
      },
    );
  }
}

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
```

```
  return Scaffold(
    appBar: AppBar(
      title: const Text('Home Screen'),
    ),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          const Text('Welcome to the Home Screen.'),
          const SizedBox(height: 20),
          ElevatedButton(
            onPressed: () {
              Navigator.pushNamed(context, '/about');
            },
            child: const Text('Go to About'),
          ),
        ],
      ),
    ),
  );
 }
}

class AboutScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('About Screen'),
    ),
    body: Center(
```

```
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          const Text('This is the About Screen'),
          const SizedBox(height: 20),
          ElevatedButton(
            onPressed: () {
              Navigator.pop(context);
            },
            child: const Text('Go back to Home'),
          ),
        ],
      ),
    ),
  );
}
}
```

OUTPUT:





WEEK – 3

5a) Learn about Stateful and Stateless widgets.?

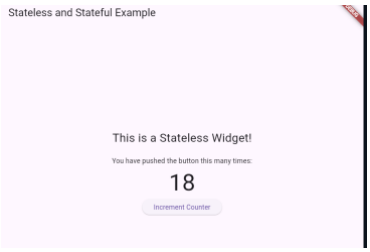
PROGRAM:

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

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

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

```
}

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

class MyHomePageState extends State<MyHomePage> {
 int _counter = 0; // Counter variable

 void _incrementCounter() {
  setState(() {
   _counter++; // Increment counter
  });
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: const Text('Stateless and Stateful Example'),
   ),
   body: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: <Widget>[
     // Stateless Widget
     StatelessExample(),
     const SizedBox(height: 20), // Spacer
     // Stateful Widget
     Text(
      'You have pushed the button this many times:',
```

```
        ),
        Text(
          '$_counter', // Display counter value
          style: TextStyle(fontSize: 48),
        ),
        ElevatedButton(
          onPressed: _incrementCounter, // Call incrementCounter when pressed
          child: const Text('Increment Counter'),
        ),
      ],
    ),
  );
 }
}

// Stateless Widget Example
class StatelessExample extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Center(
    child: Text(
     'This is a Stateless Widget!',
      style: TextStyle(fontSize: 24),
    ),
  );
 }
}
```

OUTPUT:



5b) Implement state management using set state and provider.?

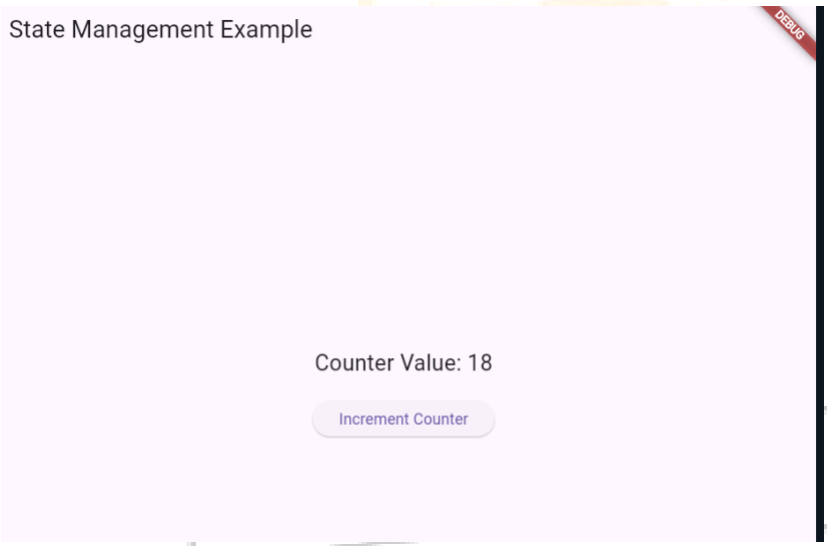
PROGRAM:

```
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; // Private counter variable

 int get counter => _counter; // Getter for counter

 void incrementCounter() {
  _counter++; // Increment the counter
  notifyListeners(); // Notify listeners to rebuild
 }
}

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

  return Scaffold(
   appBar: AppBar(
    title: Text('State Management Example'),
   ),
   body: Center(
    child: Column(
     mainAxisAlignment: MainAxisAlignment.center,
     children: <Widget>[
      Text(
       'Counter Value: ${counterModel.counter}', // Display the counter value
       style: TextStyle(fontSize: 20),
      ),
      SizedBox(height: 20), // Space between text and button
      ElevatedButton(
```

```
          onPressed: counterModel.incrementCounter, // Call incrementCounter when pressed
          child: Text('Increment Counter'),
        ),
      ],
    ),
  ),
);
}
}
```

OUTPUT:



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

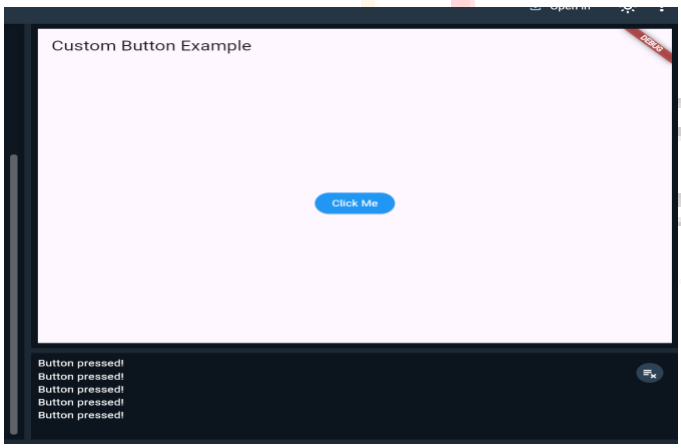
PROGRAM:

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

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

class MyApp extends StatelessWidget {
```

```
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Button Example',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(
          title: Text('Custom Button Example'),
        ),
        body: Center(
          child: CustomButton(
            text: 'Click Me',
            onPressed: () {
              // Handle button click action
              print('Button pressed!');
            },
          ),
        ),
      ),
    );
  }
}

class CustomButton extends StatelessWidget {
  final String text;
  final VoidCallback onPressed;

  CustomButton({required this.text, required this.onPressed});
```

```
 @override
 Widget build(BuildContext context) {
  return ElevatedButton(
   onPressed: onPressed,
   style: ElevatedButton.styleFrom(
    padding: EdgeInsets.symmetric(horizontal: 20, vertical: 15),
    backgroundColor: Colors.blue, // Set the background color here
    foregroundColor: Colors.white, // Set the text color here
   ),
   child: Text(text),
  );
 }
}
```

OUTPUT:



6b) Apply Styling using themes and custom styles.?

PROGRAM:

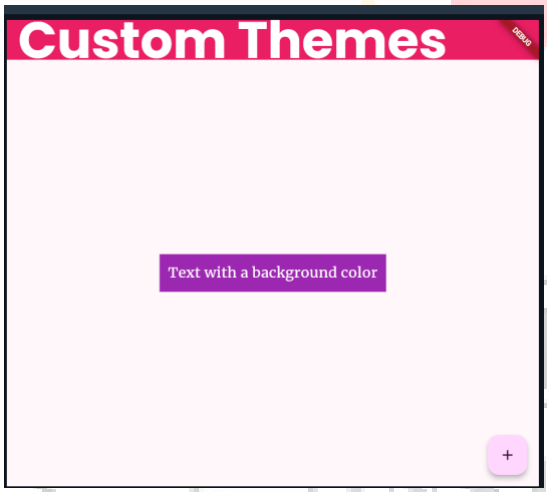
```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';

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

```
}

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

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

  return MaterialApp(
   title: appName,
   theme: ThemeData(
    brightness: Brightness.light,
    colorScheme: ColorScheme.fromSeed(
     seedColor: Colors.purple,
     primary: Colors.purple,
     secondary: Colors.pink,
    ),
    textTheme: TextTheme(
     displayLarge: GoogleFonts.poppins( // Change this to a valid font
      fontSize: 72,
      fontWeight: FontWeight.bold,
     ),
     bodyMedium: GoogleFonts.merriweather(
      fontSize: 20,
     ),
     displaySmall: GoogleFonts.pacifico(),
    ),
   ),
   home: MyHomePage(title: appName),
  );
```

```
  }
}

class MyHomePage extends StatelessWidget {
  final String title;

  const MyHomePage({Key? key, required this.title}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(
          title,
          style: Theme.of(context).textTheme.displayLarge!.copyWith(
              color: Theme.of(context).colorScheme.onSecondary,
            ),
        ),
        backgroundColor: Theme.of(context).colorScheme.secondary,
      ),
      body: Center(
        child: Container(
          padding: const EdgeInsets.symmetric(horizontal: 12, vertical: 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: FloatingActionButton(
      onPressed: () {
        // Action on button pressed
      },
      child: const Icon(Icons.add),
    ),
  );
 }
}
```

OUTPUT:



WEEK – 4

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

PROGRAM:

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

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

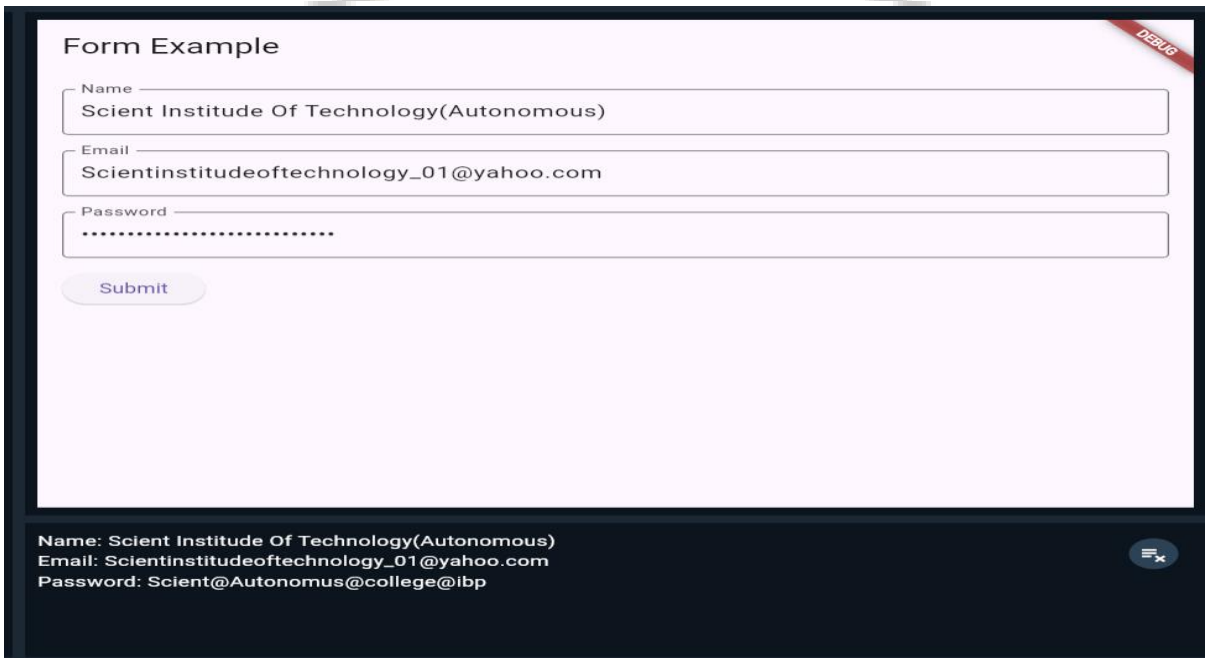
```
class MyApp extends StatelessWidget {
 const MyApp({Key? key}) : super(key: key);

 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   title: 'Form Example',
   theme: ThemeData(
    primarySwatch: Colors.blue,
   ),
   home: const MyForm(),
  );
 }
}

class MyForm extends StatefulWidget {
 const MyForm({Key? key}) : super(key: key);

 @override
 MyFormState createState() => MyFormState();
}

class MyFormState extends State<MyForm> {
 final _formKey = GlobalKey<FormState>();

 final TextEditingController nameController = TextEditingController();
 final TextEditingController emailController = TextEditingController();
 final TextEditingController passwordController = TextEditingController();

 @override
```

```
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: const Text('Form Example')),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Form(
        key: _formKey,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: <Widget>[
            TextFormField(
              controller: nameController,
              decoration: const InputDecoration(
                labelText: 'Name',
                border: OutlineInputBorder(),
              ),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Please enter your name';
                }
                return null;
              },
            ),
            const SizedBox(height: 16),
            TextFormField(
              controller: emailController,
              keyboardType: TextInputType.emailAddress,
              decoration: const InputDecoration(
                labelText: 'Email',
                border: OutlineInputBorder(),
              ),
```

```
          validator: (value) {
            if (value == null || value.isEmpty) {
              return 'Please enter your email';
            } else if (!RegExp(r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$').hasMatch(value))
{
              return 'Please enter a valid email address';
            }
            return null;
          },
        ),
        const SizedBox(height: 16),
        TextFormField(
          controller: passwordController,
          obscureText: true,
          decoration: const InputDecoration(
            labelText: 'Password',
            border: OutlineInputBorder(),
          ),
          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;
          },
        ),
        const SizedBox(height: 16),
        ElevatedButton(
          onPressed: () {
            if (_formKey.currentState!.validate()) {
```

```
            // Form is valid, process the data
            print("Name: ${nameController.text}");
            print("Email: ${emailController.text}");
            print("Password: ${passwordController.text}");
          }
        },
        child: const Text('Submit'),
      ),
    ],
  ),
 ),
),
);
}
}
```

OUTPUT:



Hint: (you need to fill the details in fields)

7b) Implement form Validation and error handling.?

PROGRAM:

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

class MyForm extends StatefulWidget {
  const MyForm({Key? key}) : super(key: key);

  @override
  MyFormState createState() => MyFormState();
}
```

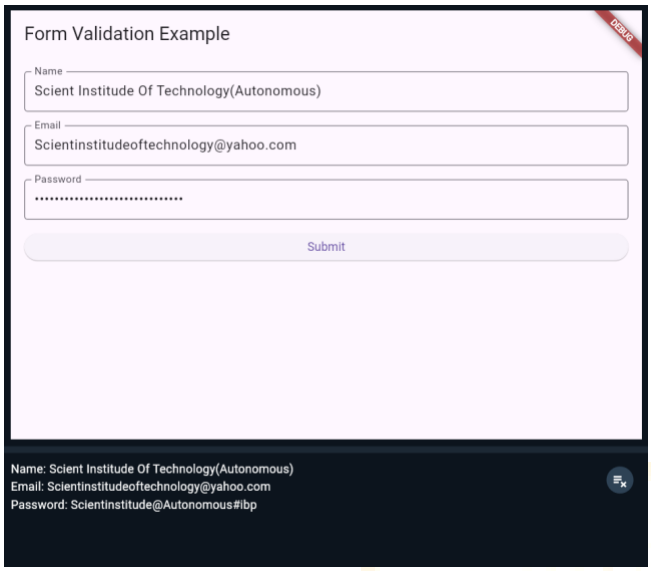
```
class MyFormState extends State<MyForm> {
 final _formKey = GlobalKey<FormState>();

 final TextEditingController nameController = TextEditingController();
 final TextEditingController emailController = TextEditingController();
 final TextEditingController passwordController = TextEditingController();

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(title: const Text('Form Validation Example')),
   body: Padding(
    padding: const EdgeInsets.all(16.0),
    child: Form(
     key: _formKey,
     child: Column(
      crossAxisAlignment: CrossAxisAlignment.stretch,
      children: <Widget>[
       TextFormField(
        controller: nameController,
        decoration: const InputDecoration(
         labelText: 'Name',
         border: OutlineInputBorder(),
        ),
        validator: (value) {
         if (value == null || value.isEmpty) {
          return 'Please enter your name';
         }
         return null;
        },
       ),
```

```
const SizedBox(height: 16),
TextFormField(
 controller: emailController,
 keyboardType: TextInputType.emailAddress,
 decoration: const InputDecoration(
  labelText: 'Email',
  border: OutlineInputBorder(),
 ),
 validator: (value) {
  if (value == null || value.isEmpty) {
   return 'Please enter your email';
  } else if (!RegExp(r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$').hasMatch(value))
{
   return 'Please enter a valid email address';
  }
  return null;
 },
),
const SizedBox(height: 16),
TextFormField(
 controller: passwordController,
 obscureText: true,
 decoration: const InputDecoration(
  labelText: 'Password',
  border: OutlineInputBorder(),
 ),
 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;
        },
      ),
      const SizedBox(height: 16),
      ElevatedButton(
        onPressed: () {
          if (_formKey.currentState!.validate()) {
            // Form is valid, process the data
            print("Name: ${nameController.text}");
            print("Email: ${emailController.text}");
            print("Password: ${passwordController.text}");
          }
        },
        child: const Text('Submit'),
      ),
    ],
  ),
),
),
);
}
}
```

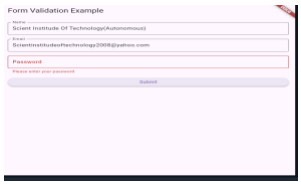
OUTPUT:

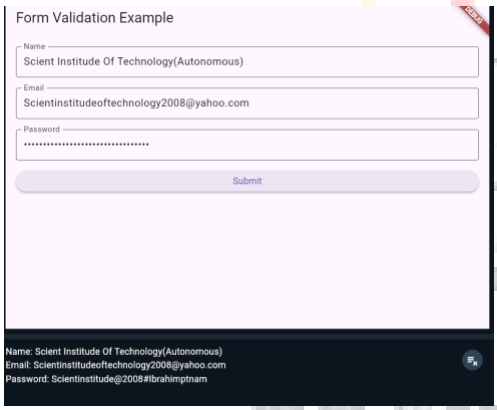
Case1: When all fields are given/filled



Case2:

When we miss some fields, this gives red line as indication/validation of missed fields and no output is generated until we fill those missed fields





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

PROGRAM:

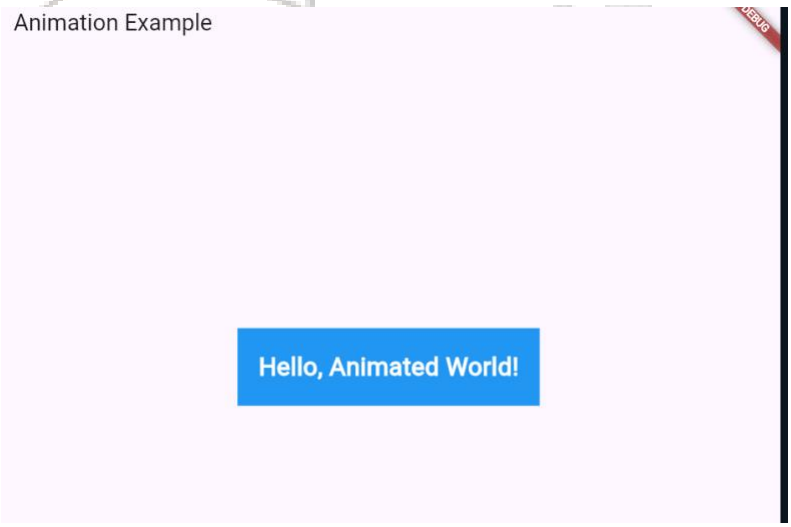
```
import 'package:flutter/material.dart';


void main() {

  runApp(const MyApp());
```

```
}

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

 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   title: 'Animation Example',
   theme: ThemeData(
    primarySwatch: Colors.blue,
   ),
   home: const MyAnimatedWidget(),
  );
 }
}

class MyAnimatedWidget extends StatefulWidget {
 const MyAnimatedWidget({Key? key}) : super(key: key);

 @override
 MyAnimatedWidgetState createState() => MyAnimatedWidgetState();
}

class MyAnimatedWidgetState extends State<MyAnimatedWidget>
  with SingleTickerProviderStateMixin {
 late AnimationController animationController;
 late Animation<double> opacityAnimation;

 @override
 void initState() {
```

```
  super.initState();

  // Initialize the AnimationController
  animationController = AnimationController(
    duration: const Duration(seconds: 2),
    vsync: this,
  );

  // Define the opacity animation from 0.0 to 1.0
  opacityAnimation = Tween<double>(begin: 0.0, end: 1.0).animate(
    CurvedAnimation(parent: animationController, curve: Curves.easeIn),
  );

  // Start the animation
  animationController.forward();
}

@override
void dispose() {
  // Dispose of the animation controller when the widget is disposed
  animationController.dispose();
  super.dispose();
}

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

```
      child: FadeTransition(
        opacity: opacityAnimation,
        child: Container(
          color: Colors.blue,
          padding: const EdgeInsets.all(20),
          child: const Text(
            'Hello, Animated World!',
            style: TextStyle(
              color: Colors.white,
              fontSize: 24,
              fontWeight: FontWeight.bold,
            ),
          ),
        ),
      ),
    ),
  );
}
}
```

OUTPUT:



8b) Experiment with different types of animations (fade, slide, etc.).?

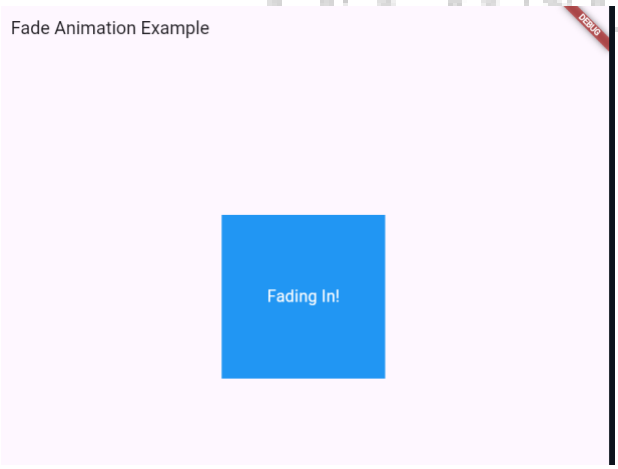
PROGRAM:

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

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

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

class FadeAnimationWidget extends StatefulWidget {
  @override
  FadeAnimationWidgetState createState() => FadeAnimationWidgetState();
}

class FadeAnimationWidgetState extends State<FadeAnimationWidget>
    with SingleTickerProviderStateMixin {
  late AnimationController animationController;
  late Animation<double> opacityAnimation;

  @override
```

```
void initState() {
  super.initState();
  // Initialize the AnimationController
  animationController = AnimationController(
    vsync: this,
    duration: const Duration(seconds: 2),
  );

  // Define the opacity animation from 0.0 to 1.0
  opacityAnimation = Tween<double>(begin: 0.0, end: 1.0).animate(
    CurvedAnimation(parent: animationController, curve: Curves.easeIn),
  );

  // Start the animation
  animationController.forward();
}

@override
void dispose() {
  // Dispose of the animation controller when the widget is disposed
  animationController.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Fade Animation Example'),
    ),
    body: Center(
```

```
      child: FadeTransition(
        opacity: opacityAnimation,
        child: Container(
          width: 200,
          height: 200,
          color: Colors.blue,
          child: const Center(
            child: Text(
              'Fading In!',
              style: TextStyle(
                color: Colors.white,
                fontSize: 20,
              ),
            ),
          ),
        ),
      ),
    ),
  ),
);
}
}
```

OUTPUT:



Slide Animation:

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

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

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

class SlideAnimationWidget extends StatefulWidget {
 @override
 SlideAnimationWidgetState createState() => SlideAnimationWidgetState();
}

class SlideAnimationWidgetState extends State<SlideAnimationWidget>
  with SingleTickerProviderStateMixin {
 late AnimationController animationController;
 late Animation<Offset> slideAnimation;

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

```
  // Initialize the AnimationController
  animationController = AnimationController(
    vsync: this,
    duration: const Duration(seconds: 2),
  );

  // Define the slide animation from left to center
  slideAnimation = Tween<Offset>(
    begin: const Offset(-1.0, 0.0),
    end: Offset.zero,
  ).animate(
    CurvedAnimation(
      parent: animationController,
      curve: Curves.easeInOut,
    ),
  );

  // Start the animation
  animationController.forward();
}

@override
void dispose() {
  // Dispose of the animation controller when the widget is disposed
  animationController.dispose();
  super.dispose();
}

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

```
  return Scaffold(
   appBar: AppBar(
    title: const Text('Slide Animation Example'),
   ),
   body: Center(
    child: SlideTransition(
     position: slideAnimation,
     child: Container(
      width: 200,
      height: 200,
      color: Colors.blue,
      child: const Center(
       child: Text(
        'Slide Animation',
        style: TextStyle(
         color: Colors.white,
         fontSize: 20,
        ),
       ),
      ),
     ),
    ),
   ),
  );
 }
}
```

OUTPUT:

(It must move like slide)

Scale Animation:

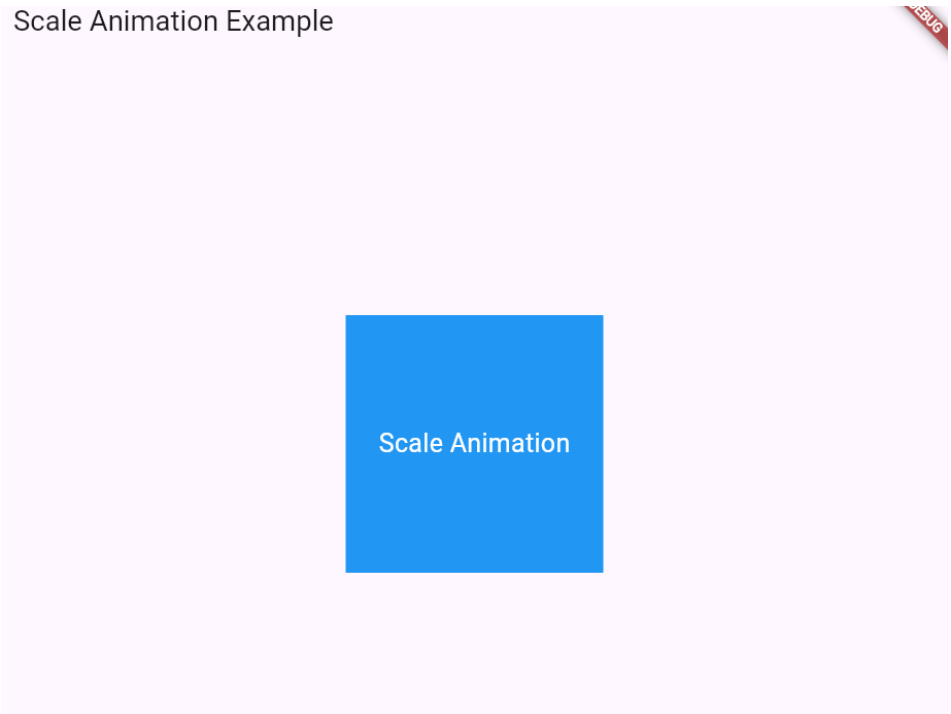
```
import 'package:flutter/material.dart';

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

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

class ScaleAnimationWidget extends StatefulWidget {
 @override
 ScaleAnimationWidgetState createState() => ScaleAnimationWidgetState();
```

```
}

class ScaleAnimationWidgetState extends State<ScaleAnimationWidget>
  with SingleTickerProviderStateMixin {
 late AnimationController animationController;
 late Animation<double> scaleAnimation;

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

  // Initialize the AnimationController
  animationController = AnimationController(
   vsync: this,
   duration: const Duration(seconds: 2),
  );

  // Define the scale animation from 0.5x to 1.0x
  scaleAnimation = Tween<double>(begin: 0.5, end: 1.0).animate(
   CurvedAnimation(
    parent: animationController,
    curve: Curves.easeInOut,
   ),
  );

  // Start the animation
  animationController.forward();
 }

 @override
 void dispose() {
```

```
  // Dispose of the animation controller when the widget is disposed
  animationController.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Scale Animation Example'),
    ),
    body: Center(
      child: ScaleTransition(
        scale: scaleAnimation,
        child: Container(
          width: 200,
          height: 200,
          color: Colors.blue,
          child: const Center(
            child: Text(
              'Scale Animation',
              style: TextStyle(
                color: Colors.white,
                fontSize: 20,
              ),
            ),
          ),
        ),
      ),
    ),
  ),
);
```

```
 }
}
```

OUTPUT:



WEEK - 5

9a) Fetch data from a REST API.?

PROGRAM:

```
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
    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: const Text('API Fetch Example'),
    ),
    body: FutureBuilder<List<Post>>(
      future: posts,
      builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
          return const Center(child: CircularProgressIndicator());
        } else if (snapshot.hasError) {
          return Center(child: Text('Error: ${snapshot.error}'));
        } else {
          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:

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

PROGRAM:

```
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',
```

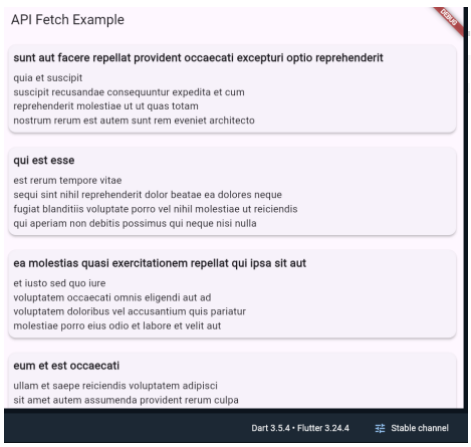
```
    theme: ThemeData(
      primarySwatch: Colors.blue,
    ),
    home: MyApiFetchWidget(),
  );
 }
}

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: const Text('API Fetch Example'),
   ),
   body: FutureBuilder<List<Post>>(
    future: posts,
    builder: (context, snapshot) {
     if (snapshot.connectionState == ConnectionState.waiting) {
      return const Center(child: CircularProgressIndicator());
     } else if (snapshot.hasError) {
      return Center(child: Text('Error: ${snapshot.error}'));
     } else {
      return ListView.builder(
       itemCount: snapshot.data!.length,
       itemBuilder: (context, index) {
        return PostListItem(post: snapshot.data![index]);
       },
      );
     }
    },
   ),
  );
 }
}

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'],
    );
  }
}

class PostListItem extends StatelessWidget {
  final Post post;

  const PostListItem({Key? key, required this.post}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(8.0),
```

```
    child: Card(
      elevation: 3,
      child: Padding(
        padding: const EdgeInsets.all(8.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              post.title,
              style: const TextStyle(fontWeight: FontWeight.bold, fontSize: 18),
            ),
            const SizedBox(height: 8),
            Text(post.body, style: const TextStyle(fontSize: 16)),
          ],
        ),
      ),
    ),
  );
}
}
```

OUTPUT:



10a) Write unit test for UI Components.?

NOTES:

Step-by-Step Guide to Writing Unit Tests

Unit tests are useful for verifying the behavior of a single function, method, or class. The test package in Dart provides basic functionality for writing and running tests, while Flutter adds further capabilities for widget testing.

Steps:

1. **Add the test dependency**:
   First, add the test package to your project as a development dependency by running:

bash

Copy code

```
flutter pub add test --dev
```

2. **Create a test file**:
   Place test files inside a test directory at the root of your project. Test files should ideally be named with a _test.dart suffix for consistency. For example, to test a Counter class, you might create a file called counter_test.dart.
3. **Write tests for your class or function**:
   - Import the test package.
   - Write a test using the test function and provide a description of what you're testing.
   - Use expect to check if the actual output matches the expected outcome.

Here's an example to illustrate:

dart

Copy code

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

import 'package:your_package/counter.dart';


void main() {

  group('Counter', () {

    test('should increment the value', () {

      final counter = Counter();

      counter.increment();
```

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

  test('should decrement the value', () {
    final counter = Counter();
    counter.decrement();
    expect(counter.value, -1);
  });
});
}
```

4. **Run the tests**:
   You can run tests in several ways:

   From terminal we use:

   flutter test

->In an IDE like Visual Studio Code or Android Studio, right-click on the test file and select "Run Test" or use the testing tab to view and run all tests.

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

NOTES:

Here's a corrected and clarified version of your guide for using Flutter's debugging tools:

Flutter Debugging Tools: A Step-by-Step Guide

Flutter provides a comprehensive set of debugging tools to help you identify and resolve issues in your app. Here's how to use them effectively:

1. Flutter DevTools

- **Run** your app using the flutter run command.
- **Activate and open DevTools** by running:

bash

Copy code

flutter pub global activate devtools

flutter pub global run devtools

- Open your app in a **Chrome browser** and connect it to DevTools by clicking 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 for comprehensive app diagnostics.

2. Flutter Inspector

- Use the Flutter Inspector within an **IDE** like **Android Studio** or **Visual Studio Code**.
- **Toggle the Inspector** in Android Studio with Alt + Shift + D (Windows/Linux) or Option + Shift + D (Mac).
- The Inspector lets you explore the widget tree, adjust widget properties, and analyze widget relationships in real-time.

3. Hot Reload

- Leverage **Hot Reload** to see the immediate effects of code changes without restarting the app.
- In the terminal, press r, or click the "Hot Reload" button in your IDE for quick updates.

4. Debugging with Breakpoints

- Set **breakpoints** in your code to pause execution and inspect variable values.
- Use the **debugger** in your IDE to step through code line by line, helping you identify issues by checking variable states and function flows.

5. Logging

- Utilize the print function to log messages to the console, which can be very helpful for tracking execution and debugging.

dart

Copy code

print('Debugging message');

- View logs in the **terminal** or the **Logs tab** in DevTools.

6. Debug Paint

- Enable **debug paint** to visualize widget layout boundaries and rendering. Use the debugPaintSizeEnabled and debugPaintBaselinesEnabled flags:

dart

Copy code

void main() {

```
  debugPaintSizeEnabled = true; // Shows widget bounding boxes
  runApp(MyApp());
}
```

7. Memory Profiling

- Use the **Memory tab** in DevTools to analyze memory usage and detect potential leaks.
- Monitor **object allocations** and **deallocations** to identify inefficient memory use and potential improvements.

8. Performance Profiling (Timeline)

- Analyze app performance with the **Timeline tab** in DevTools.
- The Timeline tab helps identify UI lag, slow frames, and other performance bottlenecks.

9. Flutter Driver Tests

- Write **automated UI tests** using Flutter Driver to simulate user interactions and validate your UI's correctness.
- This can help in ensuring that the app behaves as expected under various user scenarios.

PREPARED BY:

B. Varun Goud
3rd year Cse – B

2022-2026