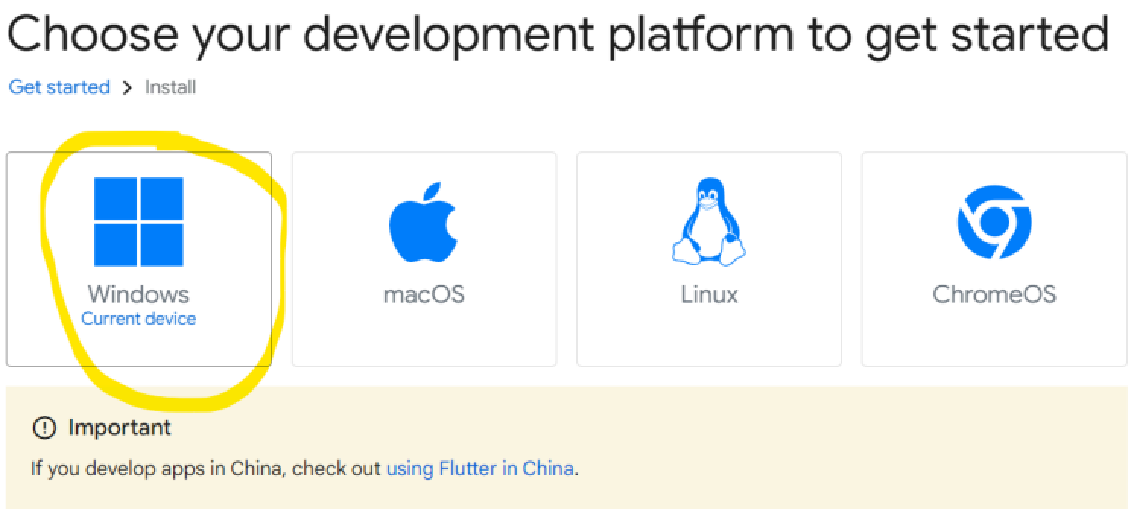
Experiment No: 1.a **Date**:	**Aim**: Install Flutter and Dart SDK.

Install VS Code

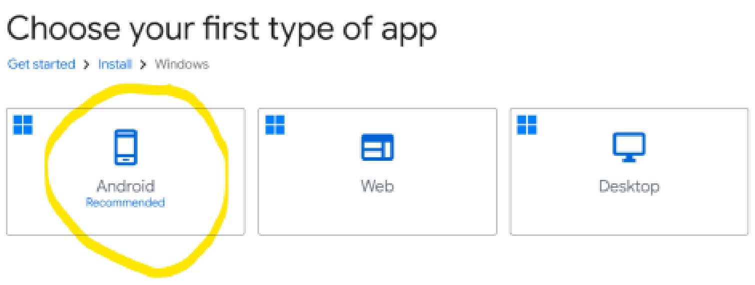
1. Download url: https://docs.flutter.dev/get-started/install/windows


Choose your development platform to get started

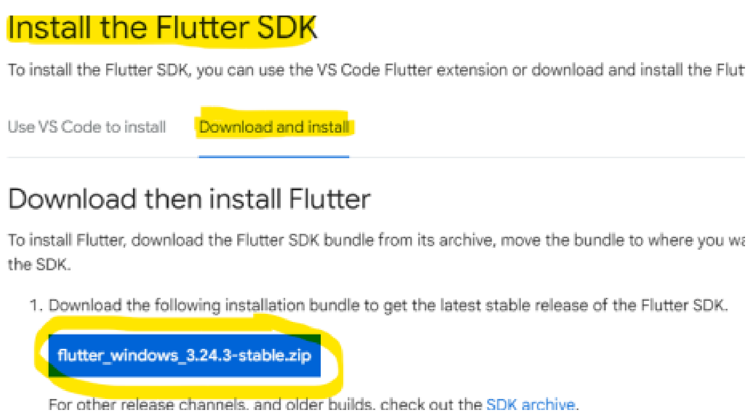
Get started > Install

Important

If you develop apps in China, check out using Flutter in China.


Choose your first type of app

Get started > Install > Windows



Install the Flutter SDK

To install the Flutter SDK, you can use the VS Code Flutter extension or download and install the Flut

Use VS Code to install Download and install

Download then install Flutter

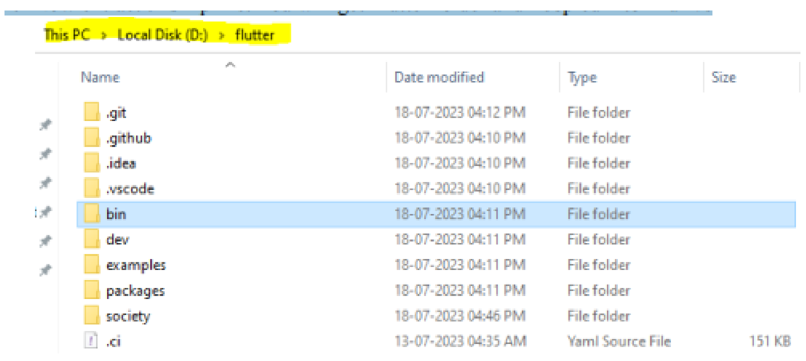
To install Flutter, download the Flutter SDK bundle from its archive, move the bundle to where you wa the SDK.

1. Download the following installation bundle to get the latest stable release of the Flutter SDK.

flutter_windows_3.24.3-stable.zip

For other release channels, and older builds, check out the SDK archive.

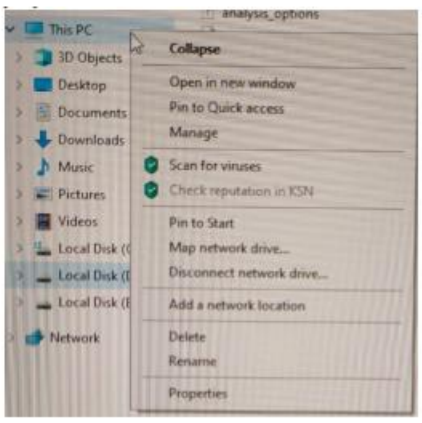
2. Now extract this zip file. You will get Flutter folder and I copied into D drive.



This PC > Local Disk (D:) > flutter

Name	Date modified	Type	Size
.git	18-07-2023 04:12 PM	File folder	
.github	18-07-2023 04:10 PM	File folder	
.idea	18-07-2023 04:10 PM	File folder	
.vscode	18-07-2023 04:10 PM	File folder	
bin	18-07-2023 04:11 PM	File folder	
dev	18-07-2023 04:11 PM	File folder	
examples	18-07-2023 04:11 PM	File folder	
packages	18-07-2023 04:11 PM	File folder	
society	18-07-2023 04:46 PM	File folder	
.ci	13-07-2023 04:35 AM	Yaml Source File	151 KB

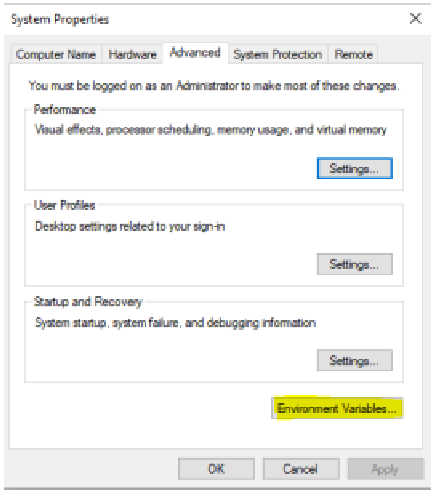
3. Set flutter Path in Environment Variable. Right click on This PC and select properties.



4. Select Advanced System settings

Advanced System Settings

5. Select Environment Variables from below



System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

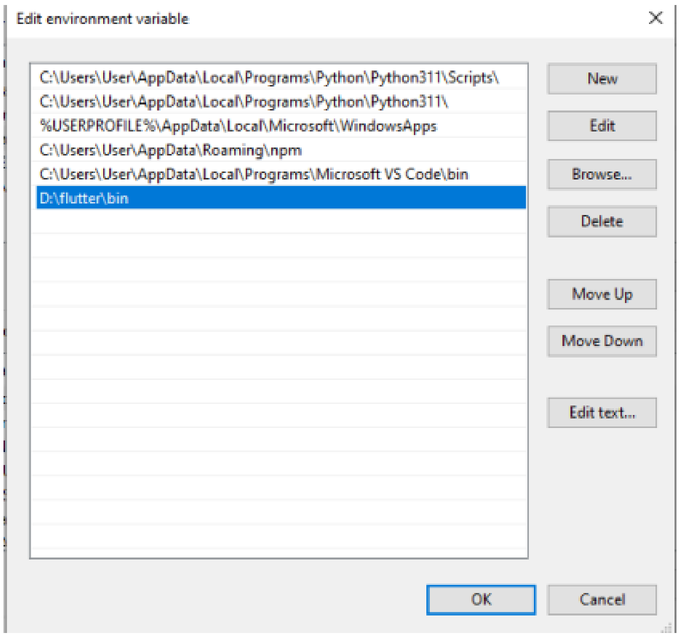
You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles
Desktop settings related to your sign-in

Startup and Recovery
System startup, system failure, and debugging information

6. Select Path and click on edit Add flutter\bin path like below

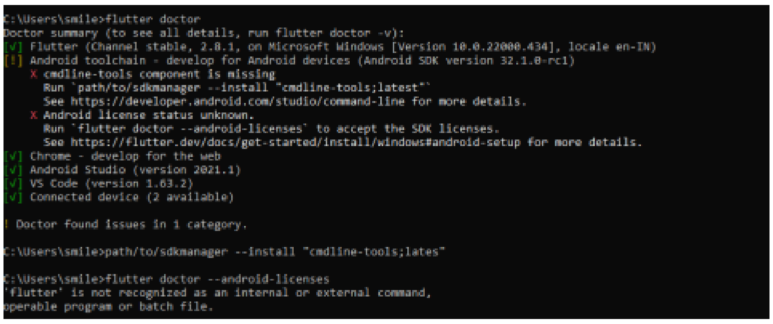


Edit environment variable

```
C:\Users\User\AppData\Local\Programs\Python\Python311\Scripts\
C:\Users\User\AppData\Local\Programs\Python\Python311\
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Users\User\AppData\Roaming\npm
C:\Users\User\AppData\Local\Programs\Microsoft VS Code\bin
D:\flutter\bin
```

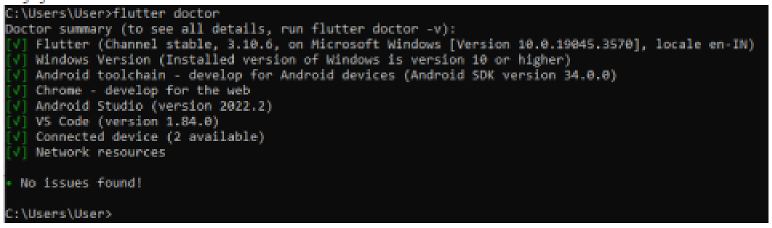
click OK OK OK

7. Go to command prompt and type flutter doctor



```
C:\Users\smile>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 2.8.1, on Microsoft Windows [Version 10.0.22000.434], locale en-IN)
[!] Android toolchain - develop for Android devices (Android SDK version 32.1.0-rc1)
    X cmdline-tools component is missing
      Run `path/to/sdkmanager --install "cmdline-tools;latest"`
      See https://developer.android.com/studio/command-line for more details.
    X Android license status unknown.
      Run `flutter doctor --android-licenses` to accept the SDK licenses.
      See https://flutter.dev/docs/get-started/install/windows#android-setup for more details.
[√] Chrome - develop for the web
[√] Android Studio (version 2021.1)
[√] VS Code (version 1.63.2)
[√] Connected device (2 available)

! Doctor found issues in 1 category.

C:\Users\smile>path/to/sdkmanager --install "cmdline-tools;lates"

C:\Users\smile>flutter doctor --android-licenses
'flutter' is not recognized as an internal or external command,
operable program or batch file.
```

Finally you see



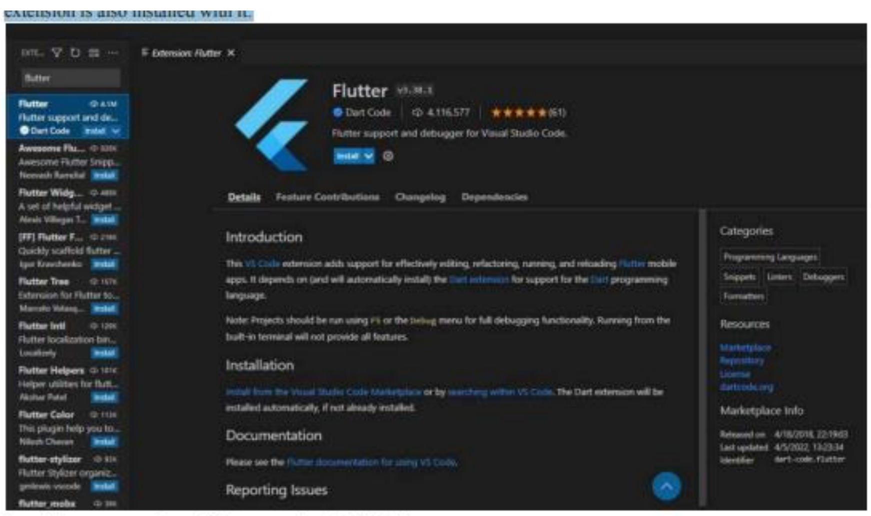
```
C:\Users\User>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.10.6, on Microsoft Windows [Version 10.0.19045.3570], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[√] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[√] Chrome - develop for the web
[√] Android Studio (version 2022.2)
[√] VS Code (version 1.84.0)
[√] Connected device (2 available)
[√] Network resources

• No issues found!

C:\Users\User>
```

Install Extensions in VS Code(https://code.visualstudio.com/download)

Now our setup is all most complete. We just need to install some extensions in VS Code. These extensions are Flutter and Dart.

Step-4.i: Go to VS Code. And go to the extension.

Step-4.ii: Type Flutter and install it. If you install the Flutter extension you'll notice that the Dart extension is also installed with it**.**



extension is also installed with it.

Flutter v3.38.1
Dart Code | 4,116,577 | ★★★★★ (61)
Flutter support and debugger for Visual Studio Code.

Introduction

This VS Code extension adds support for effectively editing, refactoring, running, and reloading Flutter mobile apps. It depends on (and will automatically install) the Dart extension for support for the Dart programming language.

Note: Projects should be run using F5 or the Debug menu for full debugging functionality. Running from the built-in terminal will not provide all features.

Installation

Install from the Visual Studio Code Marketplace or by searching within VS Code. The Dart extension will be installed automatically, if not already installed.

Documentation

Please see the Flutter documentation for using VS Code.

Reporting Issues

Marketplace Info
Released on 4/18/2018, 22:19:03
Last updated 4/5/2022, 13:23:34
Identifier dart-code.flutter

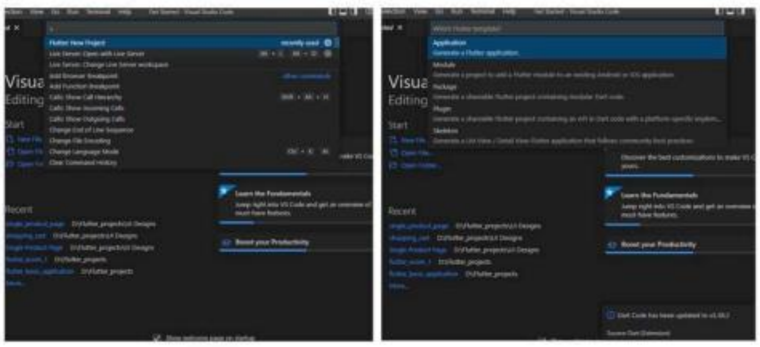
Install flutter extension and Dart extension in VS Code

Create a Flutter Applications in VS Code

Now after installing Flutter on your system, Let's create an app with Flutter.

Step-1: Open VS Code. Then open the command palette. If you can't find the command palette, then can type **Ctrl+Shift+P**. Or you can go to view and then open the command palette.

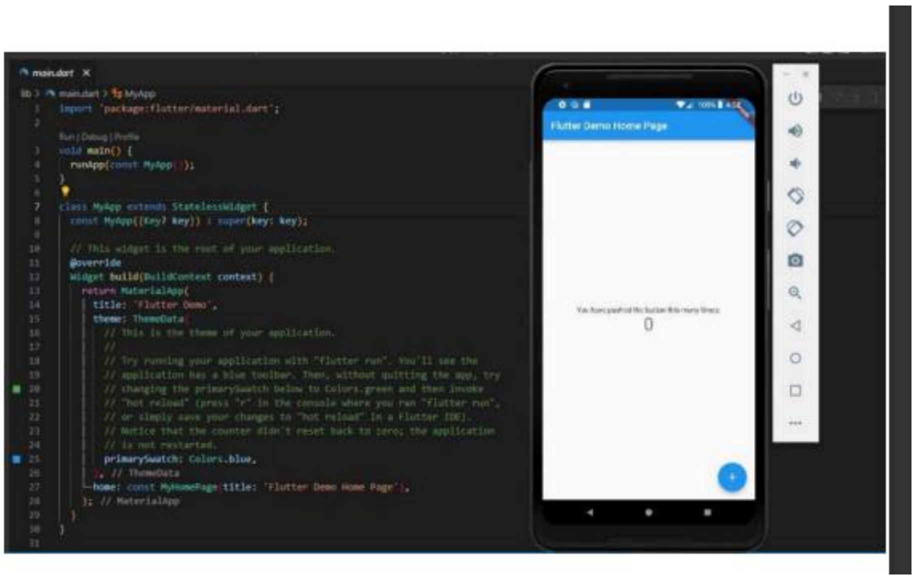
Step-2: Click the **FLutter: New Project**. After that new menu opens. Here click the Application. Then choose your desired folder, where you want to store the project



type Ctrl+Shift+P to open Command Palette

Open command Palette

Step-3: Now give a name to the Application. Here you notice a default application code for Flutter. And Run it.



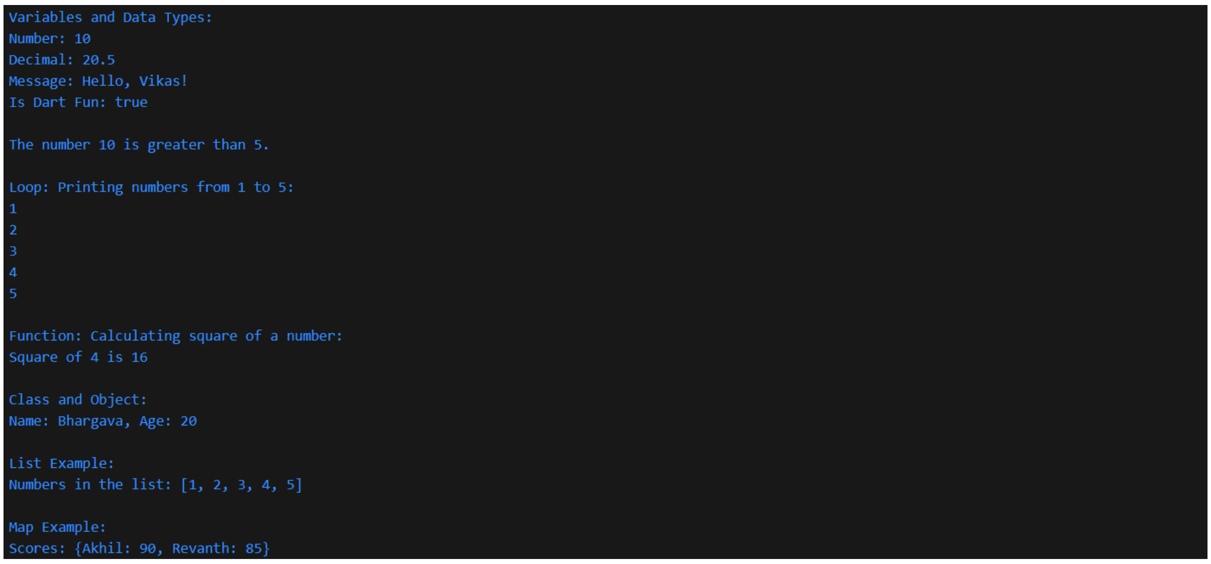
```
import 'package:flutter/material.dart';

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(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app, try
        // changing the primarySwatch below to Colors.green and then invoke
        // "hot reload" (press "r" in the console where you ran "flutter run",
        // or simply save your changes to "hot reload" in a Flutter IDE).
        // Notice that the counter didn't reset back to zero; the application
        // is not restarted.
        primarySwatch: Colors.blue,
      ), // ThemeData
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    ); // MaterialApp
  }
}
```

Experiment No: 1.b **Date**:	**Aim**: Write a simple Dart program to understand the language basics.

Code:

```
void main() {
  // 1. Variables and Data Types
  int number = 10;
  double decimal = 20.5;
  String message = 'Hello, Vikas!';
  bool isDartFun = true;

  print('Variables and Data Types:');
  print('Number: $number');
  print('Decimal: $decimal');
  print('Message: $message');
  print('Is Dart Fun: $isDartFun\n');

  // 2. Conditional Statement
  if (number > 5) {
    print('The number $number is greater than 5.');
  } else {
    print('The number $number is not greater than 5.');
  }

  // 3. Loop
  print('\nLoop: Printing numbers from 1 to 5:');
  for (int i = 1; i <= 5; i++) {
    print(i);
  }
  // 4. Function
  print('\nFunction: Calculating square of a number:');
  int square = calculateSquare(4);
  print('Square of 4 is $square');

  // 5. Class and Object
  print('\nClass and Object:');
  Person person = Person('Bhargava', 20);
  person.displayInfo();

  // 6. List (Array)
  print('\nList Example:');
  List<int> numbers = [1, 2, 3, 4, 5];
  print('Numbers in the list: $numbers');

  // 7. Map (Dictionary)
  print('\nMap Example:');
  Map<String, int> scores = {'Akhil': 90, 'Revanth': 85};
  print('Scores: $scores');
}

// Function to calculate square of a number
int calculateSquare(int number) {
  return number * number;
}

// Class to represent a Person
class Person {
  String name;
  int age;
  Person(this.name, this.age);

  void displayInfo() {
    print('Name: $name, Age: $age');
  }
}
```

Output:



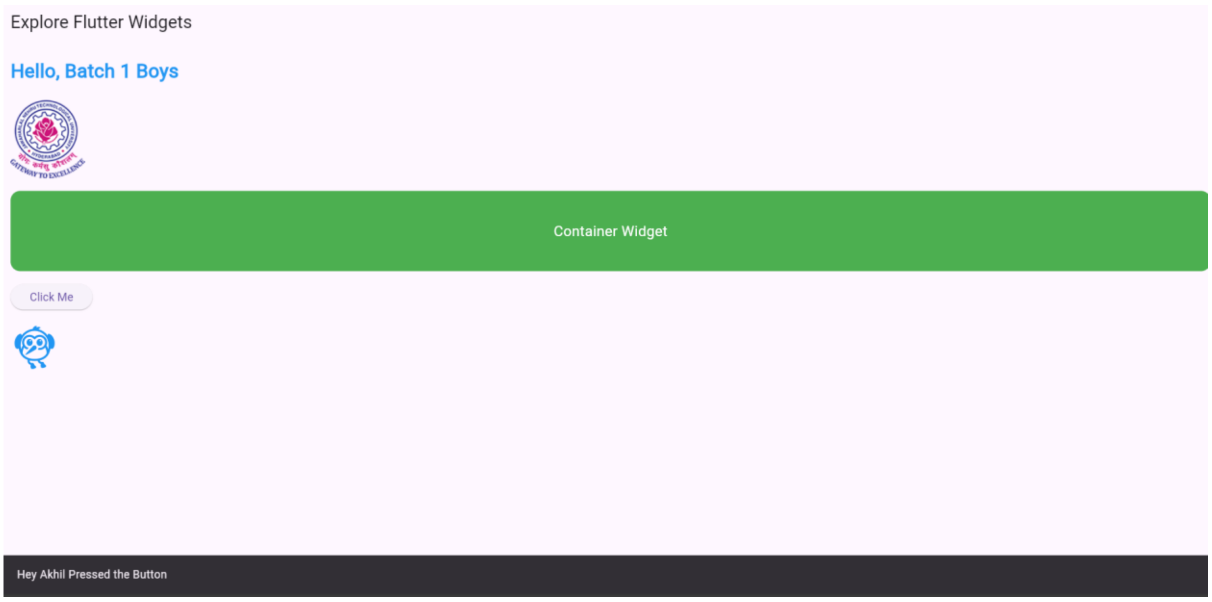
```
Variables and Data Types:
Number: 10
Decimal: 20.5
Message: Hello, Vikas!
Is Dart Fun: true

The number 10 is greater than 5.

Loop: Printing numbers from 1 to 5:
1
2
3
4
5

Function: Calculating square of a number:
Square of 4 is 16

Class and Object:
Name: Bhargava, Age: 20

List Example:
Numbers in the list: [1, 2, 3, 4, 5]

Map Example:
Scores: {Akhil: 90, Revanth: 85}
```

Experiment No: 2.a **Date**:	**Aim**: Explore various Flutter widgets (Text, Image, Container, etc.).

Code:

```
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(
      debugShowCheckedModeBanner: false,
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Explore Flutter Widgets'),
        ),
        body: const WidgetDemo(),
      ),
    );
  }
}

class WidgetDemo extends StatelessWidget {
  const WidgetDemo({super.key});
  @override
  Widget build(BuildContext context) {
    return SingleChildScrollView(
      child: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // 1. Text Widget
            const Text(
              'Hello, Batch 1 Boys',
              style: TextStyle(
                fontSize: 24,
                fontWeight: FontWeight.bold,
                color: Colors.blue,
              ),
            ),
            const SizedBox(height: 16),

            // 2. Image Widget
            const Image(
              image:
NetworkImage('http://jntuhcers.in/resp/images_new/jntuhlogo.png'),
              height: 100,
            ),
            const SizedBox(height: 16),

            // 3. Container Widget
            Container(
              height: 100,
              width: double.infinity,
  decoration: BoxDecoration(
    color: Colors.green,
    borderRadius: BorderRadius.circular(12),
  ),
  child: const Center(
    child: Text(
      'Container Widget',
      style: TextStyle(
        color: Colors.white,
        fontSize: 18,
      ),
    ),
  ),
),
const SizedBox(height: 16),

// 4. ElevatedButton Widget
ElevatedButton(
  onPressed: () {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Hey Akhil Pressed the Button')),
    );
  },
  child: const Text('Click Me'),
),
const SizedBox(height: 16),

// 5. Icon Widget
const Icon(
  Icons.flutter_dash,
  size: 60,
  color: Colors.blue,
              ),
            ],
          ),
        ),
      );
    }
  }
```

Output:


Explore Flutter Widgets

Hello, Batch 1 Boys

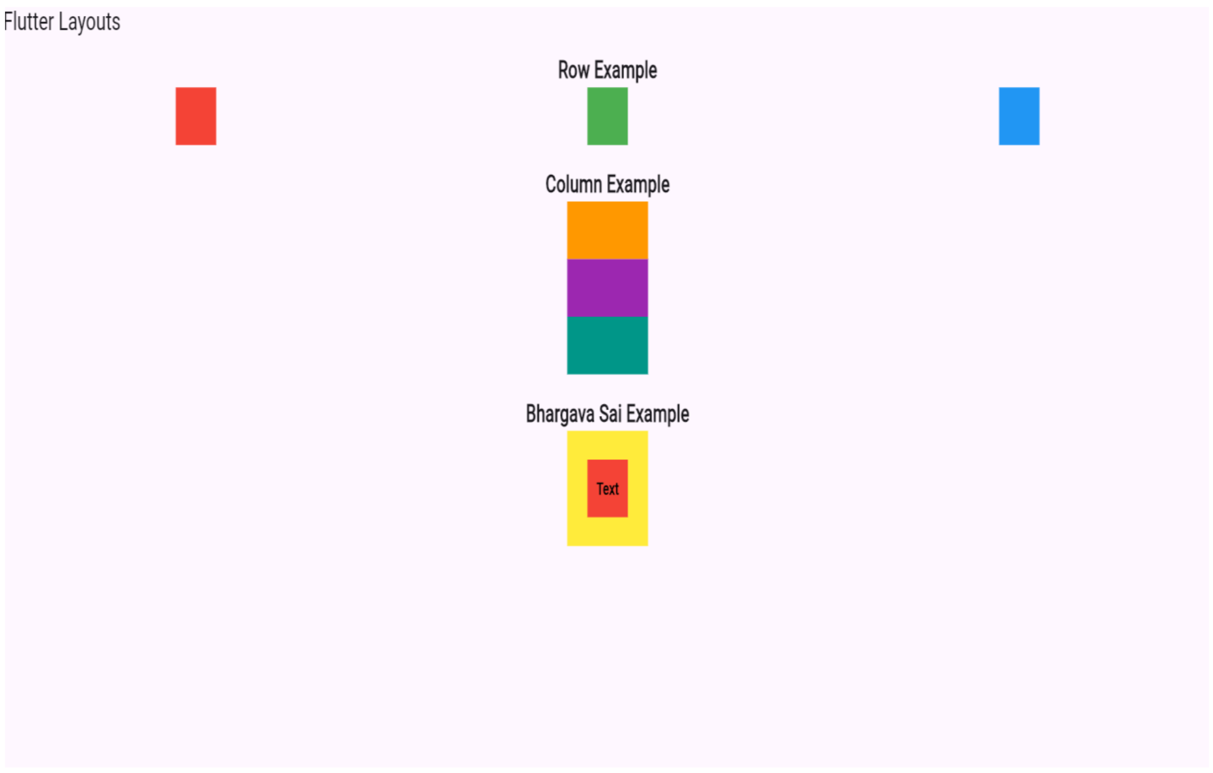
Hey Akhil Pressed the Button

Experiment No: 2.b **Date**:	**Aim**: Implement different layout structures using Row, Column, and Stack widgets.

Code:

```
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(
      debugShowCheckedModeBanner: false,
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Flutter Layouts'),
        ),
        body: Column(
          children: [
            // Row Example
            const Text(
              'Row Example',
              style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
            ),
            Row(
  mainAxisAlignment: MainAxisAlignment.spaceAround,
  children: [
    Container(color: Colors.red, height: 50, width: 50),
    Container(color: Colors.green, height: 50, width: 50),
    Container(color: Colors.blue, height: 50, width: 50),
  ],
),
const SizedBox(height: 20),

// Column Example
const Text(
  'Column Example',
  style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
),
Column(
  children: [
    Container(color: Colors.orange, height: 50, width: 100),
    Container(color: Colors.purple, height: 50, width: 100),
    Container(color: Colors.teal, height: 50, width: 100),
  ],
),
const SizedBox(height: 20),

// Stack Example
const Text(
  'Bhargava Sai Example',
  style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
),
Stack(
  alignment: Alignment.center,
  children: [
    Container(color: Colors.yellow, height: 100, width: 100),
                Container(color: Colors.red, height: 50, width: 50),
                const Text(
                  'Text',
                  style: TextStyle(
                    color: Colors.white,
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ],
            ),
          ],
        ),
      ),
    );
  }
}
```

Output:


Flutter Layouts
Row Example
Column Example
Bhargava Sai Example
Text

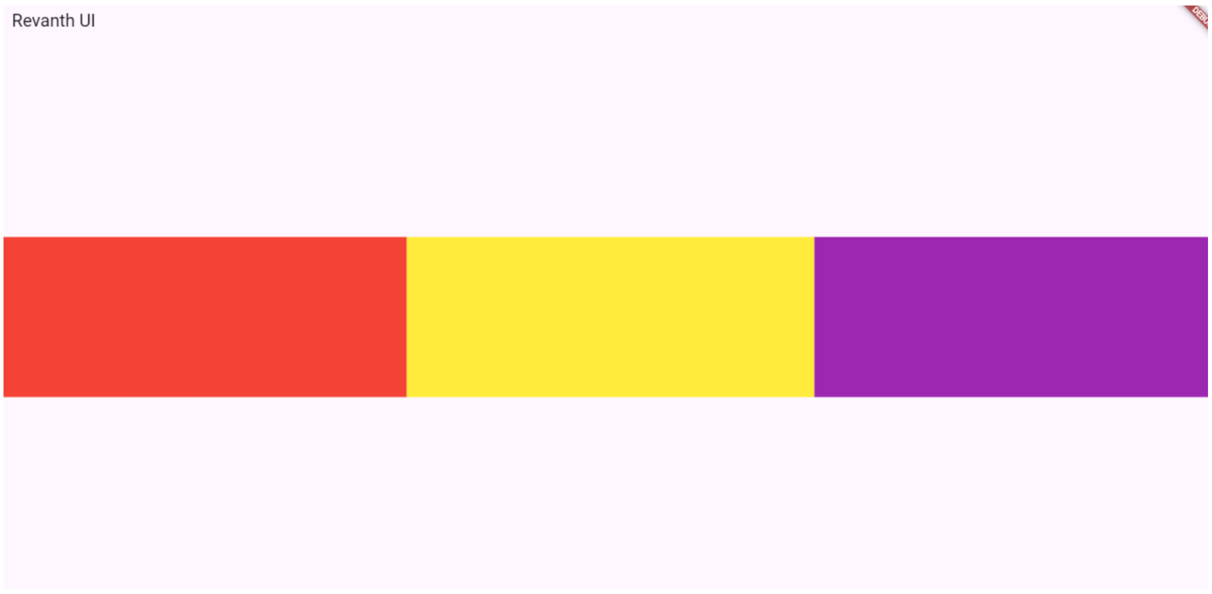
Experiment No: 3.a **Date**:	**Aim**: Design a responsive UI that adapts to different screen sizes.

Code:

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

class ResponsiveUI extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // Get the screen width
    double screenWidth = MediaQuery.of(context).size.width;

    return Scaffold(
      appBar: AppBar(title: Text('Responsive UI')),
      body: LayoutBuilder(
        builder: (context, constraints) {
          // Determine layout based on screen width
          if (screenWidth < 600) {
            return buildSmallScreen();
          } else if (screenWidth < 1200) {
            return buildMediumScreen();
          } else {
            return buildLargeScreen();
          }
        },
      ),
    );
  }

  Widget buildSmallScreen() {
```

```
  return Center(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Text('Small Screen Layout', style: TextStyle(fontSize: 20)),
        SizedBox(height: 20),
        Container(color: Colors.blue, height: 100, width: 100),
      ],
    ),
  );
}

Widget buildMediumScreen() {
  return Center(
    child: Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Expanded(child: Container(color: Colors.green, height: 100)),
        SizedBox(width: 20),
        Expanded(child: Container(color: Colors.orange, height: 100)),
      ],
    ),
  );
}

Widget buildLargeScreen() {
  return Center(
    child: Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: [
        Expanded(child: Container(color: Colors.red, height: 200)),
        Expanded(child: Container(color: Colors.yellow, height: 200)),
          Expanded(child: Container(color: Colors.purple, height: 200)),
        ],
      ),
    );
  }
}

void main() {
  runApp(MaterialApp(home: ResponsiveUI()));
}
```

Output:

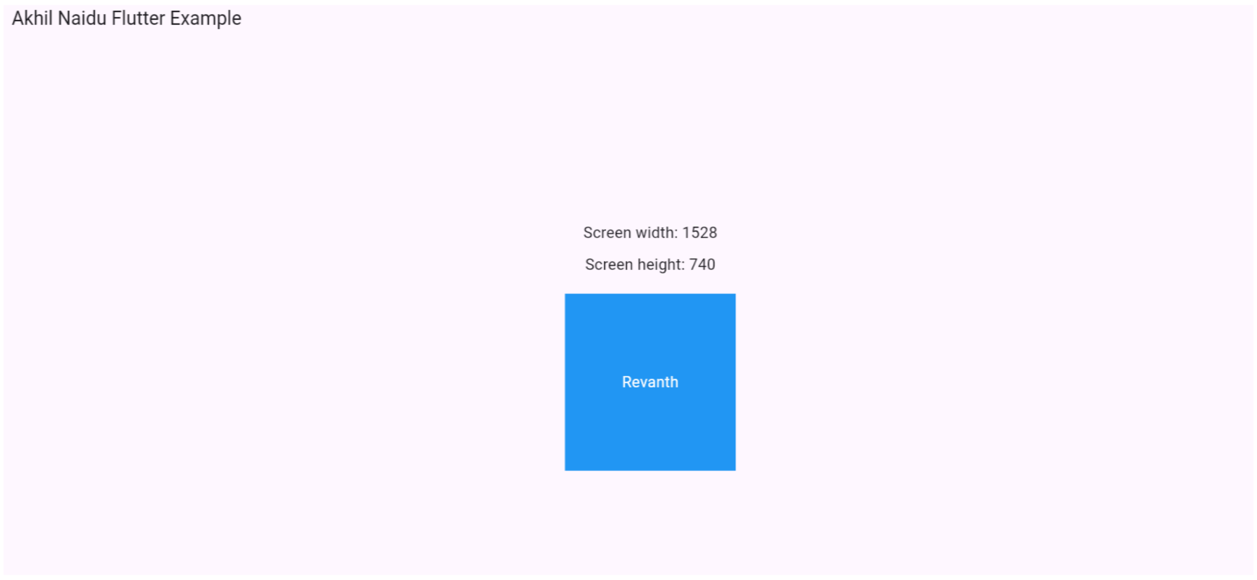


Experiment No: 3.b **Date:**	**Aim:** Implement media queries and breakpoints for responsiveness.

Code:

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

class SimpleMediaQueryExample extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // Get screen width and height using MediaQuery
    double screenWidth = MediaQuery.of(context).size.width;
    double screenHeight = MediaQuery.of(context).size.height;

    return Scaffold(
      appBar: AppBar(
        title: Text('Akhil Naidu Flutter Example'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text(
              'Screen width: $screenWidth',
              style: TextStyle(fontSize: 18),
            ),
            SizedBox(height: 10),
            Text(
              'Screen height: $screenHeight',
              style: TextStyle(fontSize: 18),
            ),
            SizedBox(height: 20),
            // Show different layout based on screen width
            screenWidth > 500
              ? Container(
                  color: Colors.blue,
                  width: 200,
                  height: 200,
                  child: const Center(
                    child: Text(
                      'Revanth',
                      style: TextStyle(color: Colors.white, fontSize: 18),
                    ),
                  ),
                )
              : Container(
                  color: Colors.green,
                  width: 100,
                  height: 100,
                  child: const Center(
                    child: Text(
                      'Small Screen',
                      style: TextStyle(color: Colors.white, fontSize: 16),
                    ),
                  ),
                ),
          ],
        ),
      ),
    );
  }
}
void main() => runApp(MaterialApp(home: SimpleMediaQueryExample()));
```

Output:


Akhil Naidu Flutter Example

```
Screen width: 1528
Screen height: 740
```

Experiment No: 4.a **Date**:	**Aim**: Set up navigation between different screens using Navigator

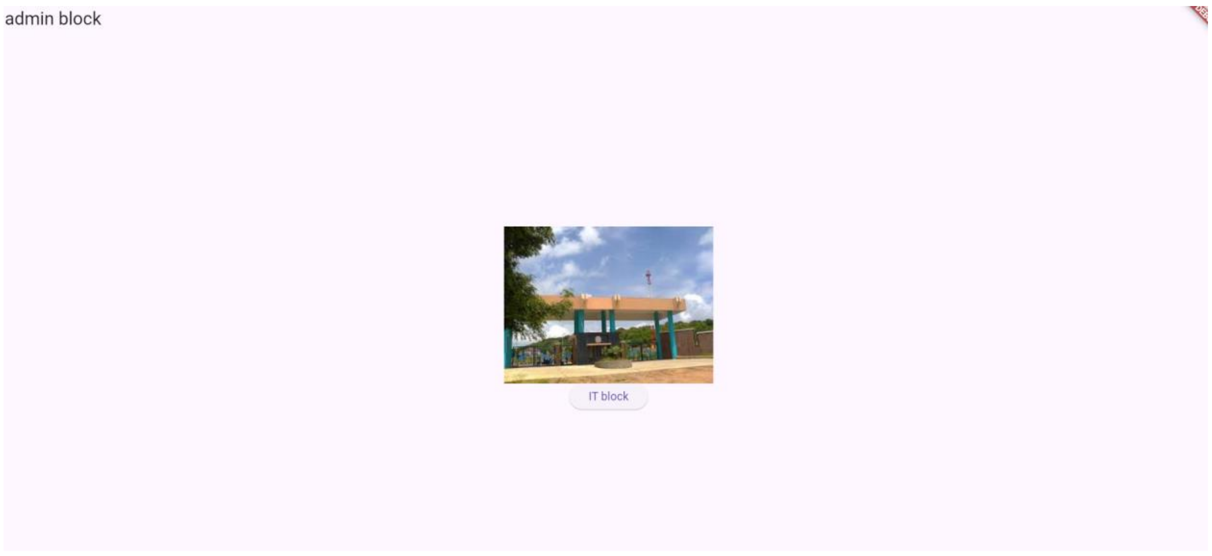
Code: (main.dart)

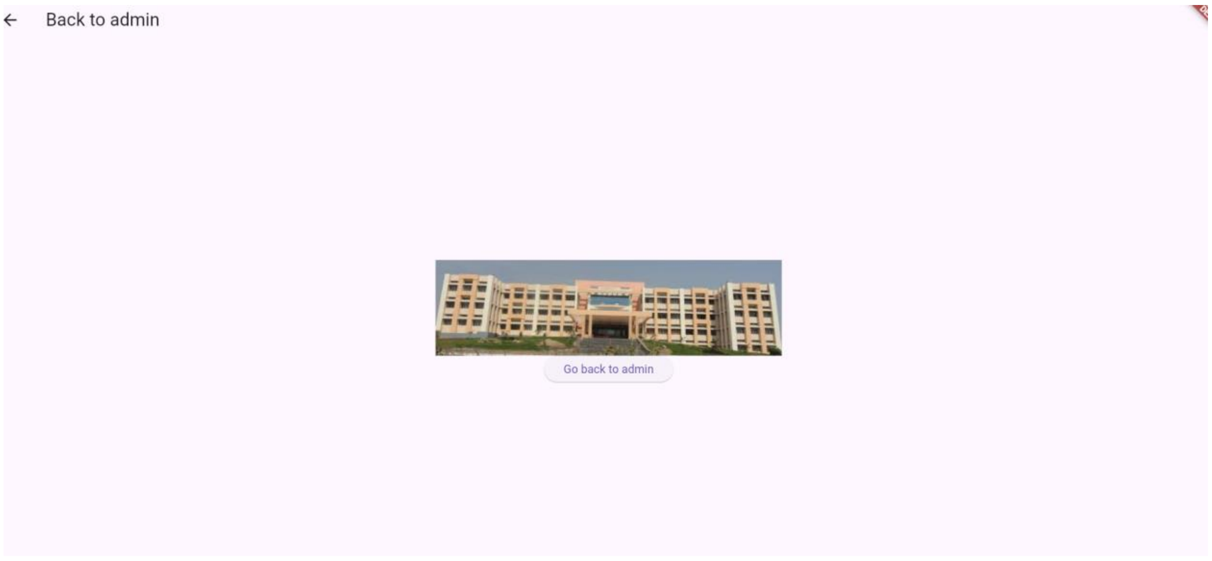
```
import 'package:flutter/material.dart';
import 'ui.dart';

void main() {
  runApp(const MaterialApp(
    title: 'Navigation Basics',
    home: FirstRoute(),
  ));
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('admin block'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            // Image widget to display an image on the FirstRoute
            Image.network(
            'https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcS_sliffgz8EB_LnyTsE3atpsBm9oZUj8i
3xQ&s'), // Change path accordingly
          ElevatedButton(
            child: const Text('IT block'),
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => const SecondRoute()),
              );
            },
          ),
        ],
      ),
    ),
  );
 }
}
```

Code: (ui.dart)

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

void main() {
  runApp(const MaterialApp(
    title: 'Navigation Basics',
    home: SecondRoute(),
  ));
}

class SecondRoute extends StatelessWidget {
  const SecondRoute({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Back to admin'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            // Image widget to display an image on the SecondRoute
            Image.network(
                'https://jntuhcej.ac.in/web//photogallery/Acad.Block-I(1)_.JPG'), //
Change path accordingly
            ElevatedButton(
              onPressed: () {
                Navigator.pop(context);
              },
              child: const Text('Go back to admin'),
            ),
          ],
        ),
      ),
    );
  }
}
```

Output:





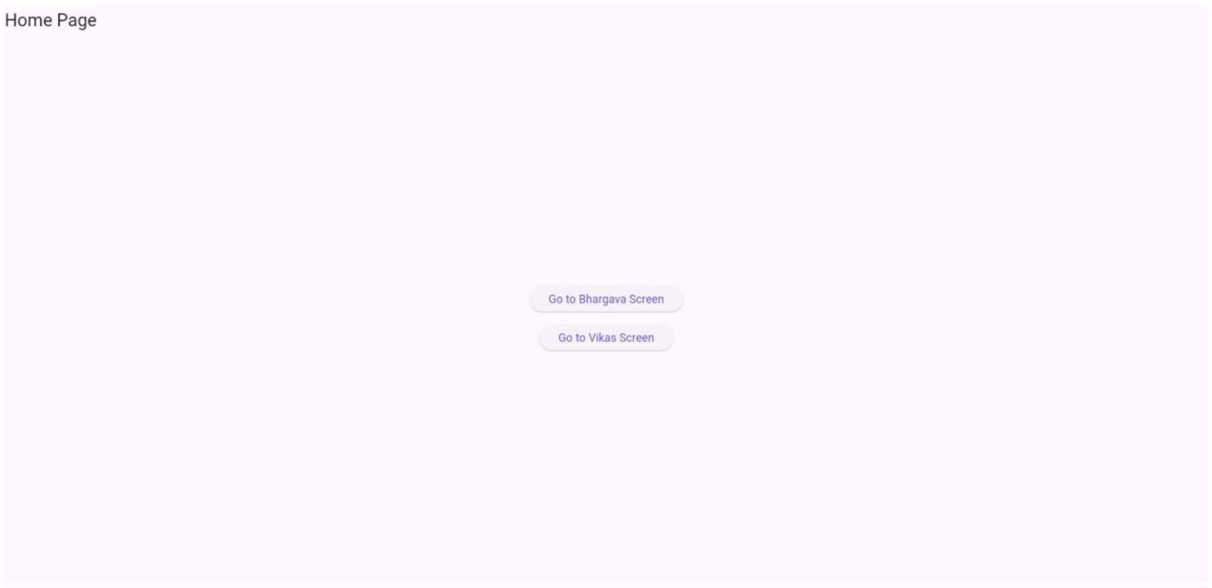
Experiment No: 4.b **Date:**	**Aim**: Implement navigation with named routes.

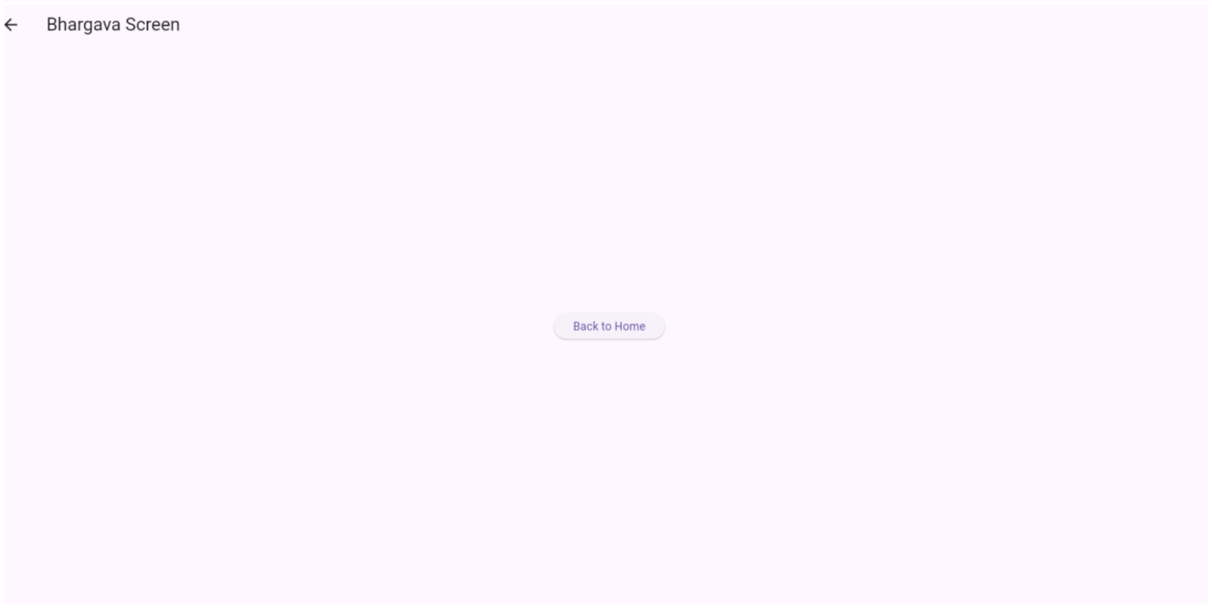
Code:

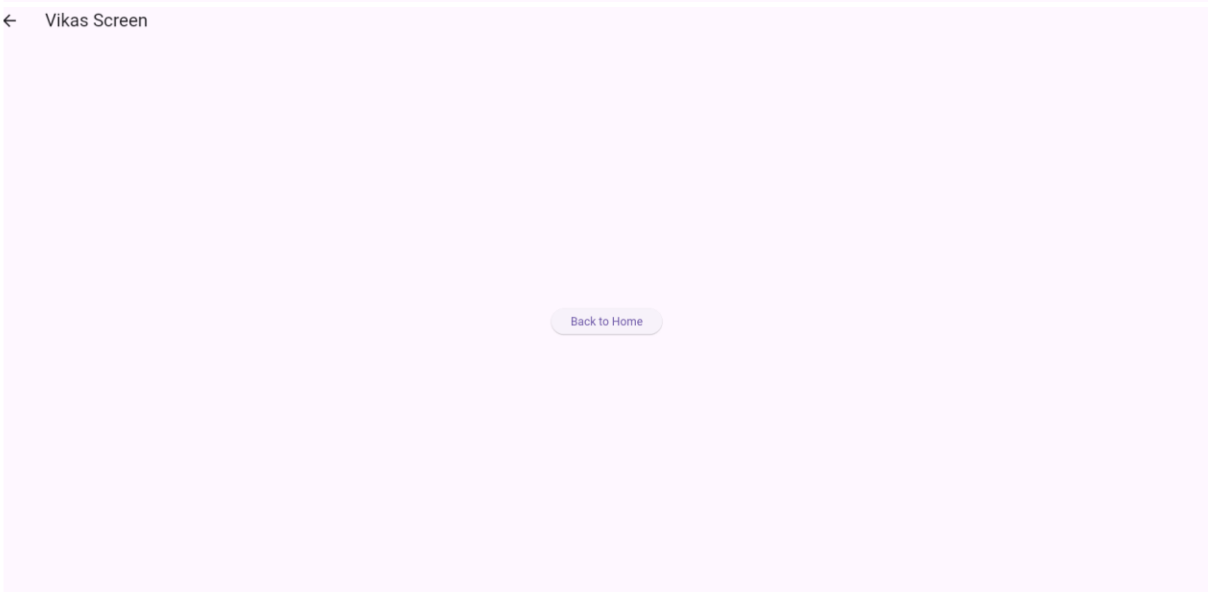
```
import 'package:flutter/material.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      // Define named routes
      initialRoute: '/',
      routes: {
        '/': (context) => const HomePage(),
        '/screen1': (context) => const Screen1(),
        '/screen2': (context) => const Screen2(),
      },
    );
  }
}

// Home Page
```

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Home Page')),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              onPressed: () => Navigator.pushNamed(context, '/screen1'),
              child: const Text('Go to Screen 1'),
            ),
            const SizedBox(height: 16),
            ElevatedButton(
              onPressed: () => Navigator.pushNamed(context, '/screen2'),
              child: const Text('Go to Screen 2'),
            ),
          ],
        ),
      ),
    );
  }
}

// Screen 1
class Screen1 extends StatelessWidget {
  const Screen1({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Screen 1')),
      body: Center(
        child: ElevatedButton(
          onPressed: () => Navigator.pop(context),
          child: const Text('Back to Home'),
        ),
      ),
    );
  }
}

// Screen 2
class Screen2 extends StatelessWidget {
  const Screen2({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Screen 2')),
      body: Center(
        child: ElevatedButton(
          onPressed: () => Navigator.pop(context),
          child: const Text('Back to Home'),
        ),
      ),
    );
  }
}
```

Output:


Home Page
Go to Bhargava Screen
Go to Vikas Screen


Bhargava Screen
Back to Home

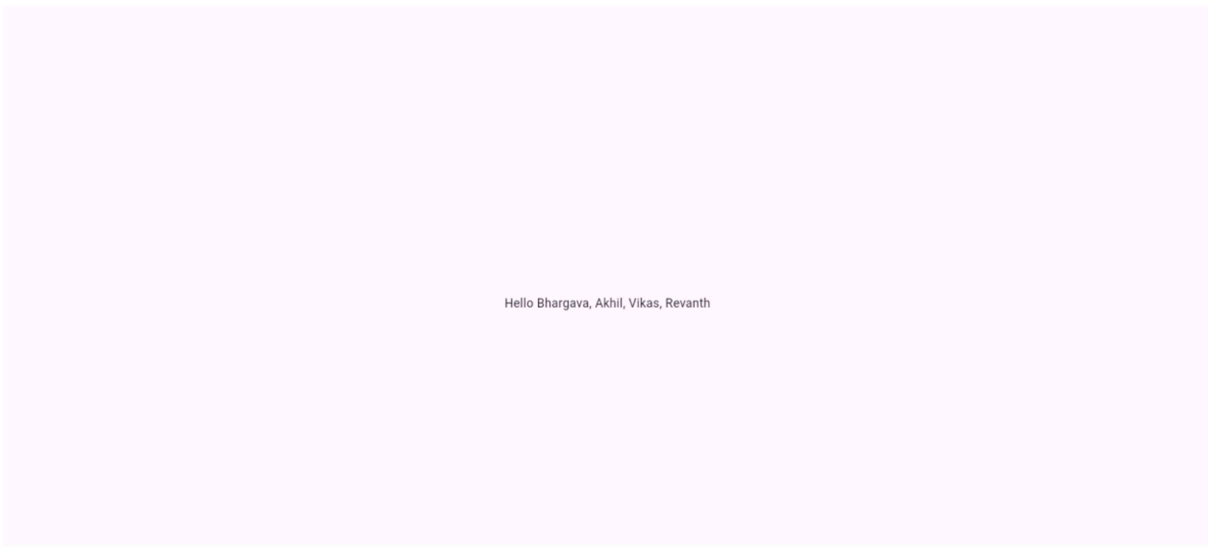

Vikas Screen
Back to Home

Experiment No: 5.a **Date**:	**Aim:** Learn about stateful and stateless widgets.

Code:

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

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

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

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: Scaffold(
        body: Center(
          child: Text('Hello Bhargava, Akhil, Vikas, Revanth'),
        ),
      ),
    );
  }
}
```

Output:



```
Hello Bhargava, Akhil, Vikas, Revanth
```

Experiment No: 5.b

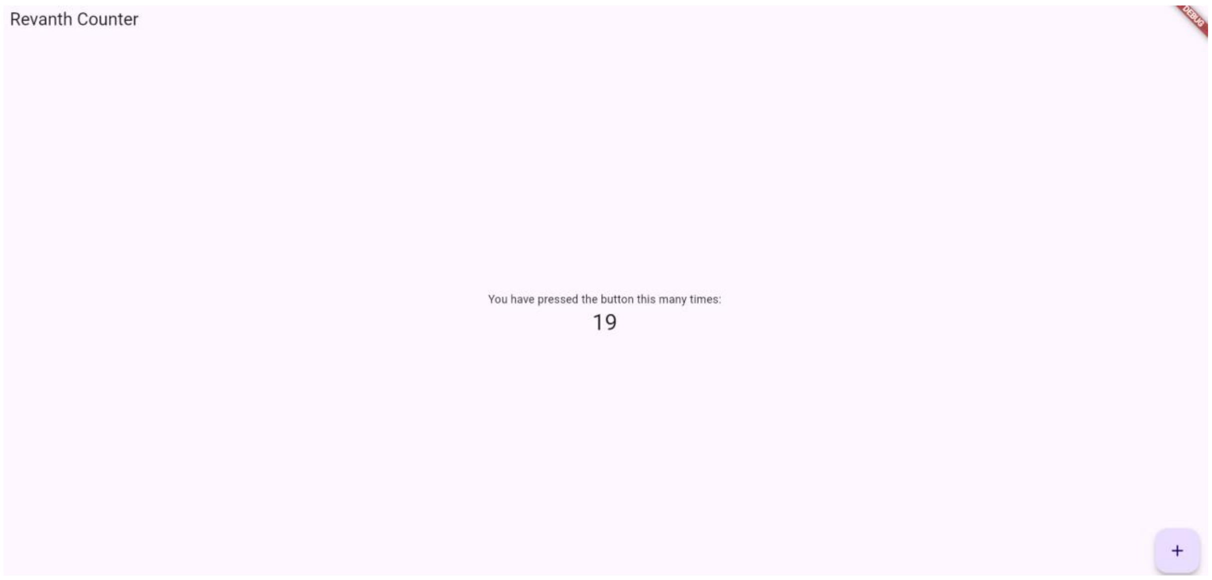
Date:

Aim: Implement state management using set State and Provider.

Code:

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

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

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

class CounterProvider extends ChangeNotifier {
  int _counter = 0;
  int get counter => _counter;
  void increment() {
    _counter++;
    notifyListeners(); // Notify listeners to rebuild widgets
  }
}

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

Output:



You have pressed the button this many times:

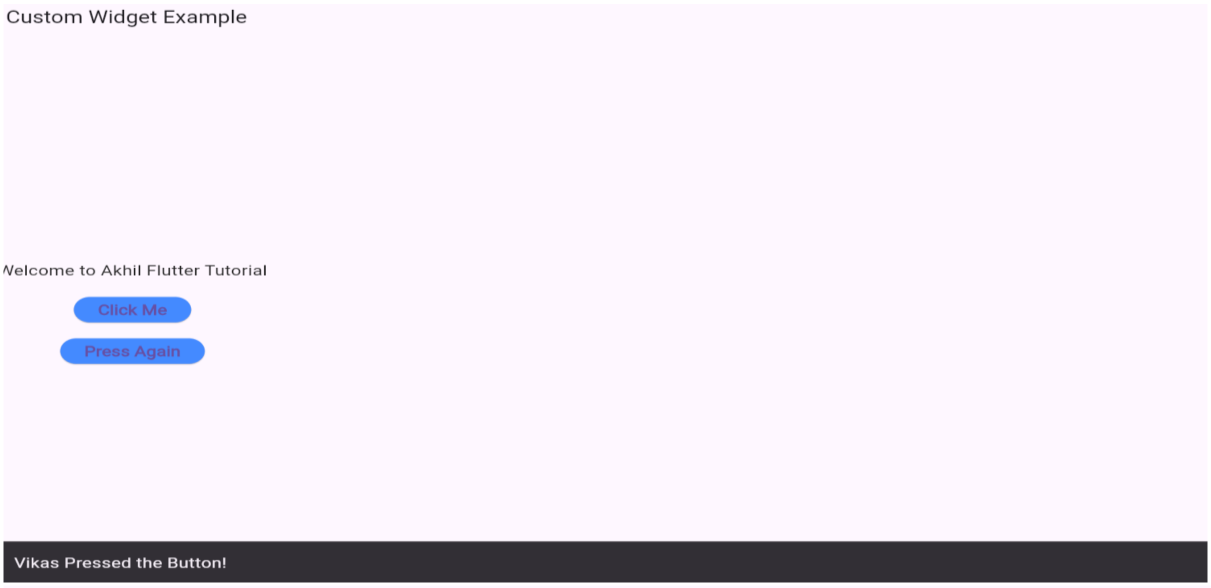
19

Experiment No: 6.a,b **Date**:	**Aim**: Create custom widgets for specific UI elements. & Apply styling using themes and custom styles

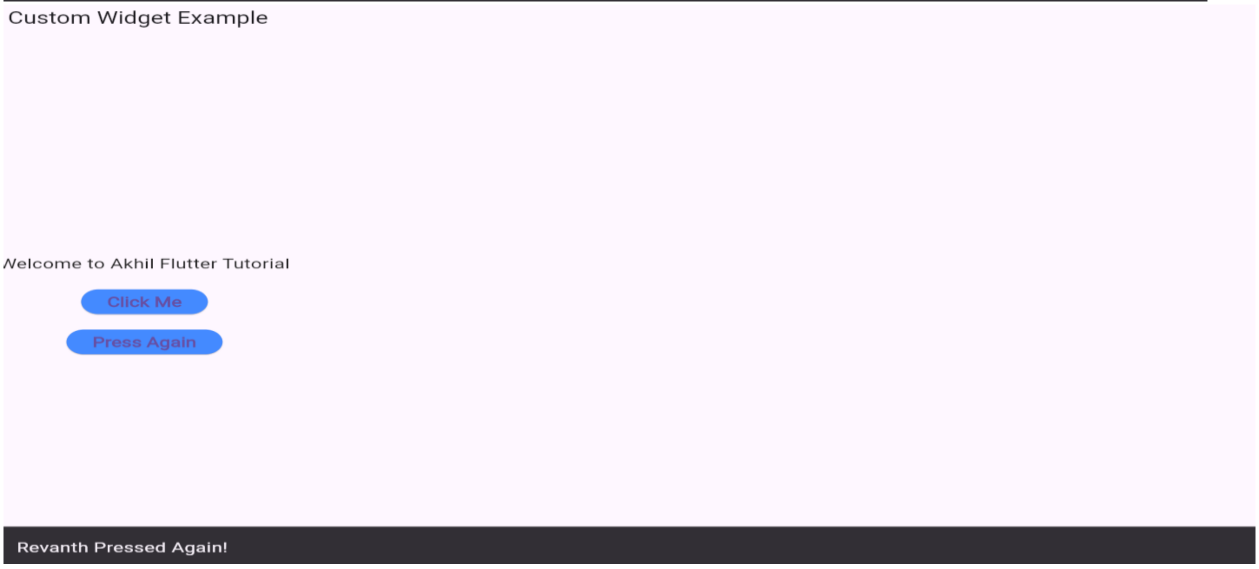
Code:

```
import 'package:flutter/material.dart';
void main() {
 runApp(MyApp());
}
// The main app widget
class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   title: 'Custom Widgets and Styling',
   theme: ThemeData(
    // Define a global theme for the app
    primarySwatch: Colors.blue,
    textTheme: TextTheme(
     bodyMedium: TextStyle(fontSize: 18, color: Colors.black87),
    ),
   ),
   home: HomeScreen(),
  );
 }
}
// HomeScreen widget that uses the custom button and applies styles
class HomeScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
      appBar: AppBar(
        title: Text('Custom Widget Example'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(8.0),
   child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text(
              'Welcome to Akhil Flutter Tutorial',
             style: Theme.of(context).textTheme.bodyMedium, // Use the global
theme
            ),
            SizedBox(height: 20),
            // Use the custom button widget
            CustomButton(
              text: 'Click Me',
              onPressed: () {
                ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(content: Text('Vikas Pressed the Button!')),
                );
              },
            ),
            SizedBox(height: 20),
            // Another instance of the custom button with different text
            CustomButton(
              text: 'Press Again',
              onPressed: () {
                ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(content: Text('Revanth Pressed Again!')),
                );
              },
          ),
        ],
      ),
    ),
  );
 }
}

// Custom button widget
class CustomButton extends StatelessWidget {
 final String text;
 final VoidCallback onPressed;

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

 @override
 Widget build(BuildContext context) {
  return ElevatedButton(
   style: ElevatedButton.styleFrom(
    padding: EdgeInsets.symmetric(horizontal: 24, vertical: 12),
    backgroundColor: Colors.blueAccent, // Custom button color
    textStyle: TextStyle(
     fontSize: 18,
     fontWeight: FontWeight.bold,
     color: Colors.white,
    ),
   ),
   onPressed: onPressed,
   child: Text(text),
  );
 }
}
```

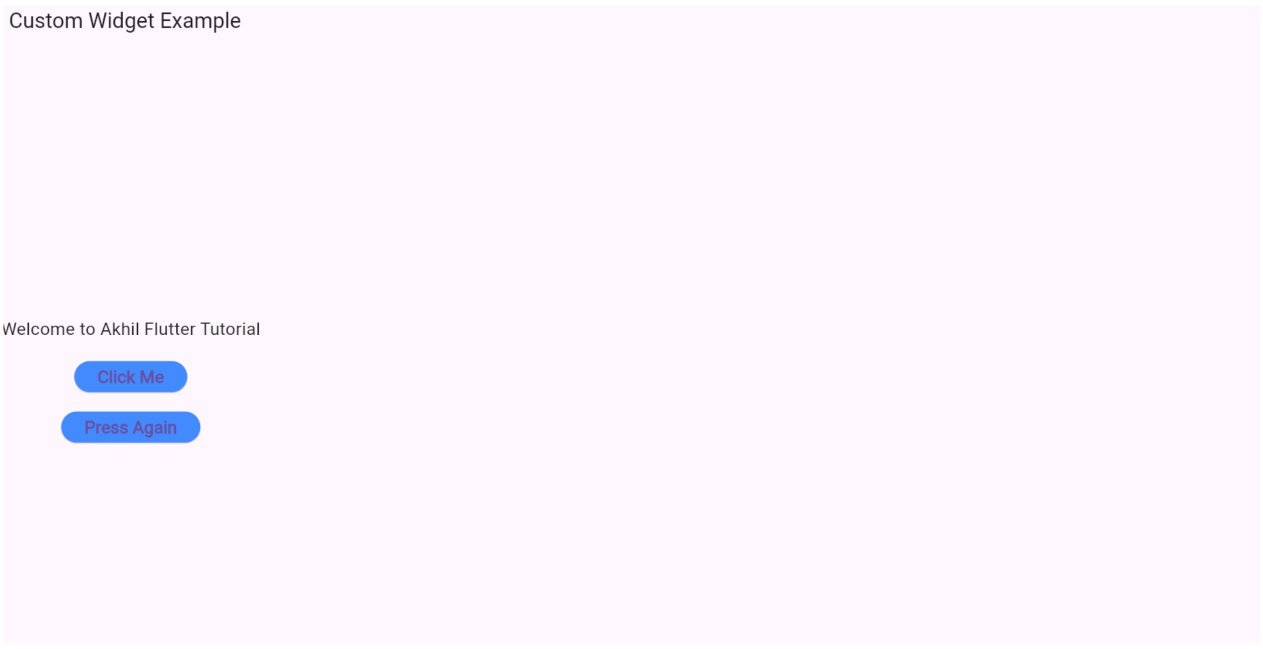
Output:



Vikas Pressed the Button!



Revanth Pressed Again!



Experiment No: 7.a,b **Date**:	**Aim**: Design a form with various input fields. & Implement form validation and error handling.

Code:

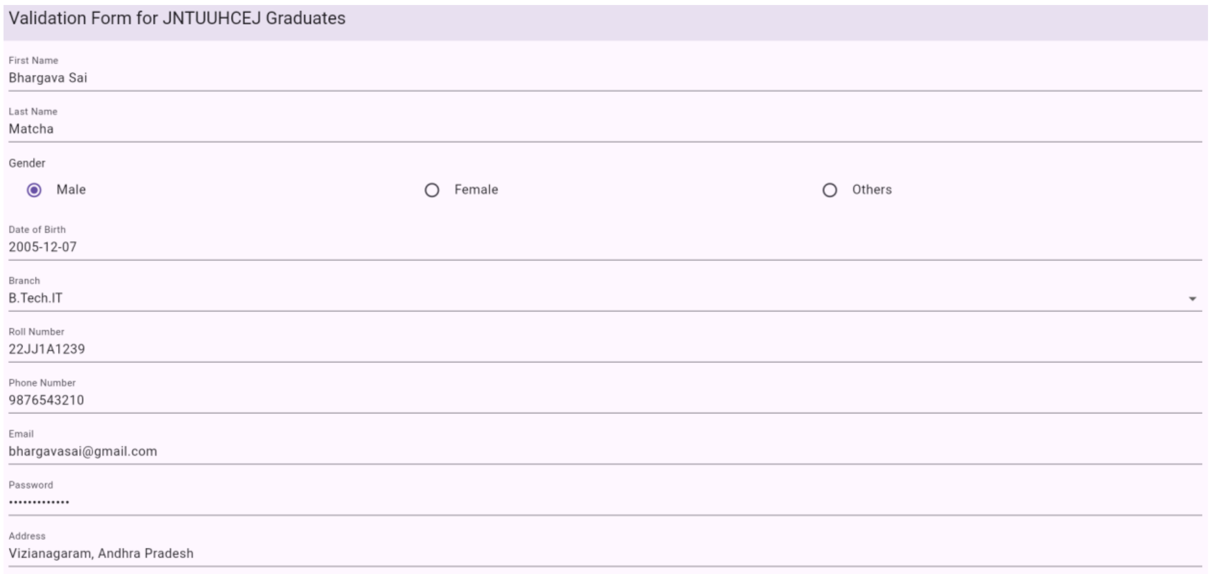
```
import 'package:flutter/material.dart';

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

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

class ValidationForm extends StatefulWidget {
  @override
  _ValidationFormState createState() => _ValidationFormState();
}

class _ValidationFormState extends State<ValidationForm> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController _dobController = TextEditingController();
  final TextEditingController _firstNameController = TextEditingController();
  final TextEditingController _lastNameController = TextEditingController();
  final TextEditingController _rollNumberController = TextEditingController();
  final TextEditingController _phoneNumberController =
TextEditingController();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  final TextEditingController _addressController = TextEditingController();
  final TextEditingController _pinCodeController = TextEditingController();

  String? _gender;
  String? _branch;

  void _resetForm() {
    _formKey.currentState!.reset();
    _dobController.clear();
    _firstNameController.clear();
    _lastNameController.clear();
    _rollNumberController.clear();
    _phoneNumberController.clear();
    _emailController.clear();
    _passwordController.clear();
    _addressController.clear();
    _pinCodeController.clear();
    setState(() {
      _gender = null;
      _branch = null;
    });
  }

  void _submitForm() {
    if (_formKey.currentState!.validate() && _gender != null && _branch !=
null) {
    // Print data to the terminal
    print('First Name: ${_firstNameController.text}');
    print('Last Name: ${_lastNameController.text}');
    print('Gender: $_gender');
    print('Date of Birth: ${_dobController.text}');
    print('Branch: $_branch');
    print('Roll Number: ${_rollNumberController.text}');
    print('Phone Number: ${_phoneNumberController.text}');
    print('Email: ${_emailController.text}');
    print('Address: ${_addressController.text}');
    print('Pin Code: ${_pinCodeController.text}');

    // Show success message
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Form Submitted Successfully!')),
    );

    // Reset the form
    _resetForm();
  } else if (_gender == null) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Please select a gender')),
    );
  } else if (_branch == null) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Please select your branch')),
    );
  }
}

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

```
// Password regex pattern:
// - At least one lowercase letter: (?=.*[a-z])
// - At least one uppercase letter: (?=.*[A-Z])
// - At least one number: (?=.*\d)
// - At least one special character: (?=.*[!@#\$%^&*(),.?":{}|<>])
// - At least 6 characters in total: {6,}
final String passwordPattern =
  r'^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[!@#\$%^&*(),.?":{}|<>]).{6,}$';

return Scaffold(
 appBar: AppBar(
  title: Text("Validation Form for JNTUUHCEJ Graduates"),
 ),
 body: Padding(
  padding: const EdgeInsets.all(16.0),
  child: Form(
   key: _formKey,
   child: ListView(
    children: [
     // First Name
     TextFormField(
      controller: _firstNameController,
      decoration: InputDecoration(labelText: 'First Name'),
      validator: (value) {
       if (value == null || value.isEmpty) {
        return 'Please enter your first name';
       }
       return null;
      },
     ),
     SizedBox(height: 16),
     // Last Name
```

```
TextFormField(
  controller: _lastNameController,
  decoration: InputDecoration(labelText: 'Last Name'),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your last name';
    }
    return null;
  },
),
SizedBox(height: 16),
// Gender
Text("Gender"),
Row(
  children: [
    Expanded(
      child: RadioListTile<String>(
        title: Text("Male"),
        value: "Male",
        groupValue: _gender,
        onChanged: (value) {
          setState(() {
            _gender = value;
          });
        },
      ),
    ),
    Expanded(
      child: RadioListTile<String>(
        title: Text("Female"),
        value: "Female",
        groupValue: _gender,
        onChanged: (value) {
          setState(() {
            _gender = value;
          });
        },
      ),
    ),
    Expanded(
      child: RadioListTile<String>(
        title: Text("Others"),
        value: "Others",
        groupValue: _gender,
        onChanged: (value) {
          setState(() {
            _gender = value;
          });
        },
      ),
    ),
  ],
),
SizedBox(height: 16),
// Date of Birth
TextFormField(
  controller: _dobController,
  decoration: InputDecoration(labelText: 'Date of Birth'),
  readOnly: true,
  onTap: () async {
    DateTime? pickedDate = await showDatePicker(
      context: context,
      initialDate: DateTime.now(),
      firstDate: DateTime(1900),
      lastDate: DateTime.now(),
    );
    if (pickedDate != null) {
      _dobController.text = "${pickedDate.toLocal()}".split(' ')[0];
    }
  },
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please select your date of birth';
    }
    return null;
  },
),
SizedBox(height: 16),
// Branch
DropdownButtonFormField<String>(
  decoration: InputDecoration(labelText: 'Branch'),
  items: [
    "B.Tech.MEC",
    "B.Tech.CSE",
    "B.Tech.EEE",
    "B.Tech.IT",
    "B.Tech.ECE",
    "M.Tech",
    "Others"
  ].map((branch) {
    return DropdownMenuItem<String>(
      value: branch,
      child: Text(branch),
    );
  }).toList(),
  onChanged: (value) {
          setState(() {
            _branch = value;
          });
        },
        validator: (value) {
          if (value == null) {
            return 'Please select your branch';
          }
          return null;
        },
      ),
      SizedBox(height: 16),
      // Roll Number
      TextFormField(
        controller: _rollNumberController,
        decoration: InputDecoration(labelText: 'Roll Number'),
        validator: (value) {
          if (value == null || value.isEmpty) {
            return 'Please enter your roll number';
          } else if (!RegExp(r'^[A-Za-z0-9]{10}$').hasMatch(value)) {
            return 'Roll number must be a mix of alphabets and numbers, 10
characters long';
          }
          return null;
        },
      ),
      SizedBox(height: 16),
      // Phone Number
      TextFormField(
        controller: _phoneNumberController,
        decoration: InputDecoration(labelText: 'Phone Number'),
        keyboardType: TextInputType.phone,
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your phone number';
    } else if (!RegExp(r'^\d{10}$').hasMatch(value)) {
      return 'Phone number must be 10 digits';
    }
    return null;
  },
),
SizedBox(height: 16),
// Email
TextFormField(
  controller: _emailController,
  decoration: InputDecoration(labelText: 'Email'),
  keyboardType: TextInputType.emailAddress,
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your email';
    } else if (!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(value)) {
      return 'Please enter a valid email address';
    }
    return null;
  },
),
SizedBox(height: 16),
// Password
TextFormField(
  controller: _passwordController,
  decoration: InputDecoration(labelText: 'Password'),
  obscureText: true,
  validator: (value) {
    if (value == null || value.isEmpty) {
          return 'Please enter a password';
        } else if (!RegExp(passwordPattern).hasMatch(value)) {
          return 'Password must be at least 6 characters long, include a
capital letter, a lowercase letter, a number, and a special character';
        }
        return null;
      },
    ),
    SizedBox(height: 16),
      // Address
    TextFormField(
      controller: _addressController,
      decoration: InputDecoration(labelText: 'Address'),
      validator: (value) {
        if (value == null || value.isEmpty) {
          return 'Please enter your address';
        }
        return null;
      },
    ),
    SizedBox(height: 16),
    // Pin Code
    TextFormField(
      controller: _pinCodeController,
      decoration: InputDecoration(labelText: 'Pin Code'),
      keyboardType: TextInputType.number,
      validator: (value) {
        if (value == null || value.isEmpty) {
          return 'Please enter your pin code';
        } else if (!RegExp(r'^\d{6}$').hasMatch(value)) {
          return 'Pin code must be 6 digits';
        }
```

```
                return null;
              },
            ),
            SizedBox(height: 16),
            // Submit and Reset buttons
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                ElevatedButton(
                  onPressed: _submitForm,
                  child: Text('Submit'),
                ),
                TextButton(
                  onPressed: _resetForm,
                  child: Text('Reset'),
                ),
              ],
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

Output:



Validation Form for JNTUUHCEJ Graduates

First Name
Bhargava Sai

Last Name
Matcha

Gender
[✓ Male] Female Others

Date of Birth
2005-12-07

Branch
B.Tech.IT

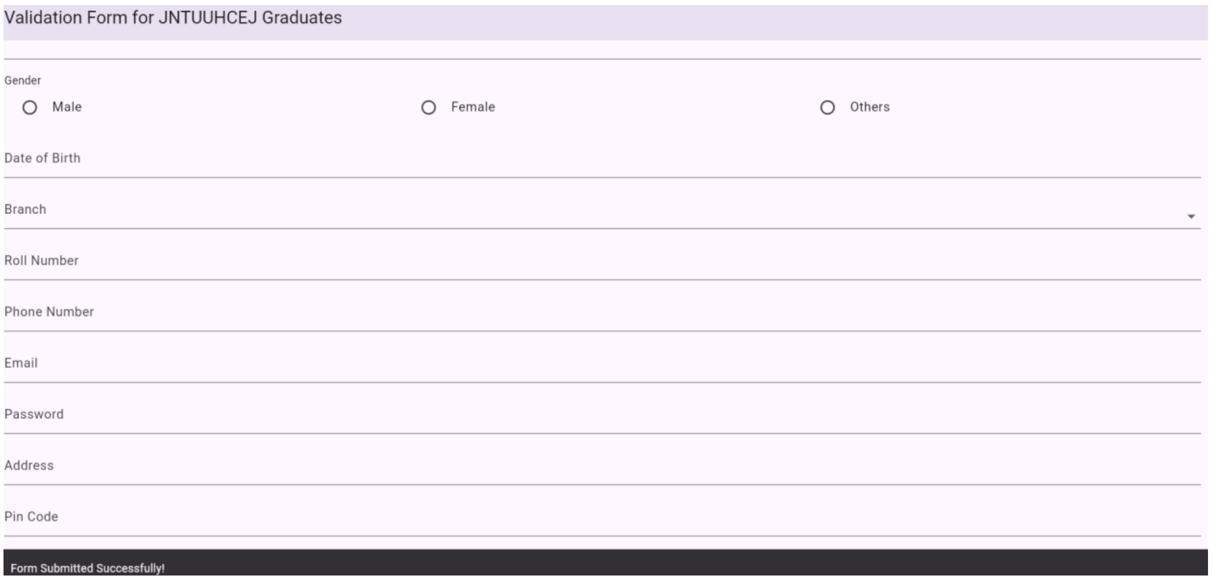
Roll Number
22JJ1A1239

Phone Number
9876543210

Email
bhargavasai@gmail.com

Password
•••••••••••••

Address
Vizianagaram, Andhra Pradesh



Validation Form for JNTUUHCEJ Graduates

Gender
Male Female Others

Date of Birth

Branch

Roll Number

Phone Number

Email

Password

Address

Pin Code

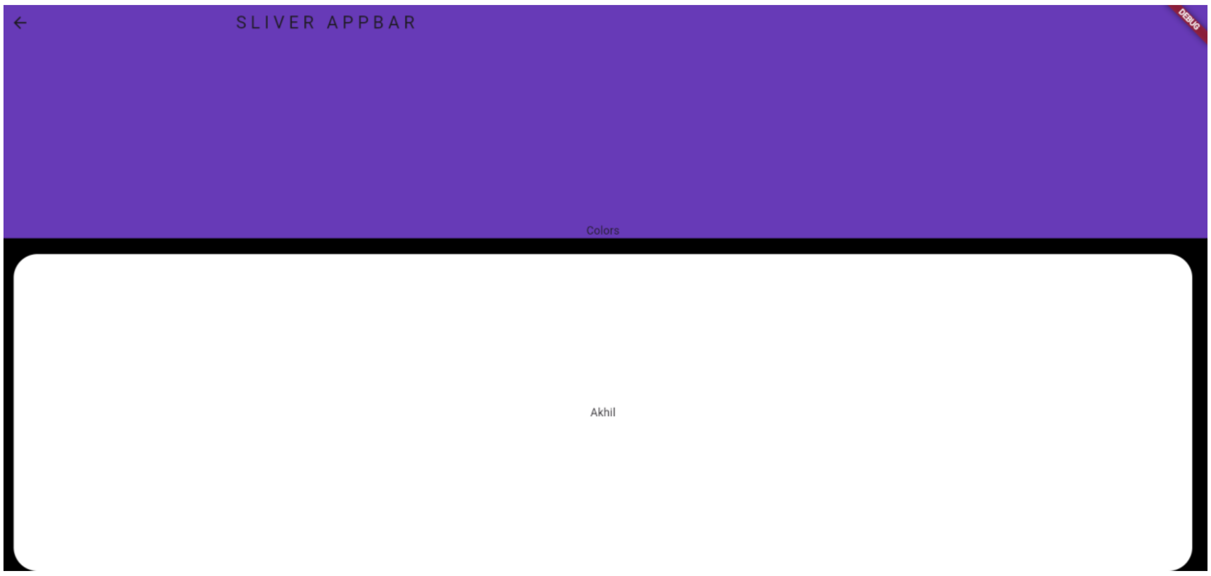
Form Submitted Successfully!

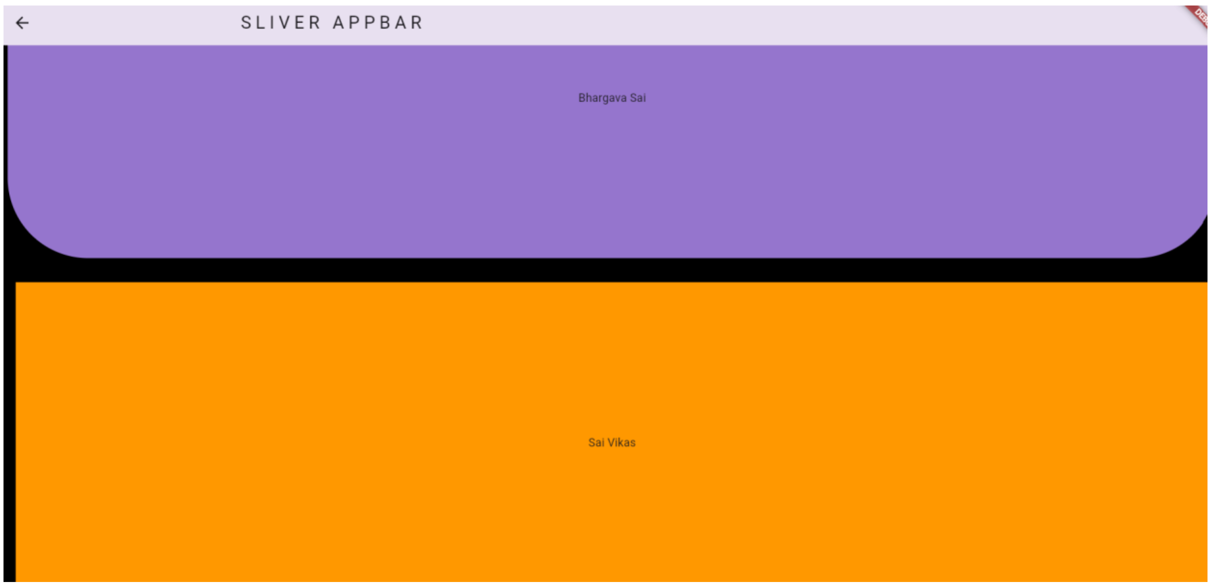
Experiment No: 8.a **Date**:	**Aim**: Add animations to UI elements using Flutter's animation framework.

Code:

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

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

  @override
  Widget build(BuildContext context) {
    var sliverAppBar = SliverAppBar(
      leading: const Icon(Icons.arrow_back),
      title: const Text('                              S L I V E R   A P P B A R     '),
      expandedHeight: 300,
      floating: false,
      pinned: true,
      flexibleSpace: FlexibleSpaceBar(
       background: Container(
        alignment: Alignment.bottomCenter,
        child: Text('Colors'),
          color: Colors.deepPurple[500],
       ),

      ),
    );

    return Scaffold(
      backgroundColor: Colors.black,
      body: CustomScrollView(
slivers: [
  sliverAppBar,

  SliverToBoxAdapter(
    child: Padding(
      padding: const EdgeInsets.all(20.0),
      child: ClipRRect(
        borderRadius: BorderRadius.circular(30),
        child: Container(
          alignment: Alignment.center,
          child: Text("Akhil"),
          height: 400,
          color: Colors.white,
        ),
      ),
    ),
  ),
  SliverToBoxAdapter(
    child: Padding(
      padding: const EdgeInsets.all(10.0),
      child:ClipRRect(
        borderRadius: BorderRadius.circular(100),
        child: Container(
          alignment: Alignment.center,
          child: Text("Bhargava Sai Matcha"),
          height: 400,
          color: Colors.deepPurple[300],
        ),
      ),
    ),
  ),
  SliverToBoxAdapter(
      child: Padding(
        padding: const EdgeInsets.all(20.0),
        child: ClipRRect(
          borderRadius: BorderRadius.circular(0),
          child: Container(
            alignment: Alignment.center,
            child: Text("Sai Vikas"),
            height: 400,
            color: Colors.orange,
          ),
        ),
      ),
     ),
    ],
   ),
  );
 }
}

void main() {
 runApp(const MaterialApp(
  home: CustomAppBar(),
 ));
}
```

Output:


SLIVER APPBAR
DEBUG
Colors
Akhil

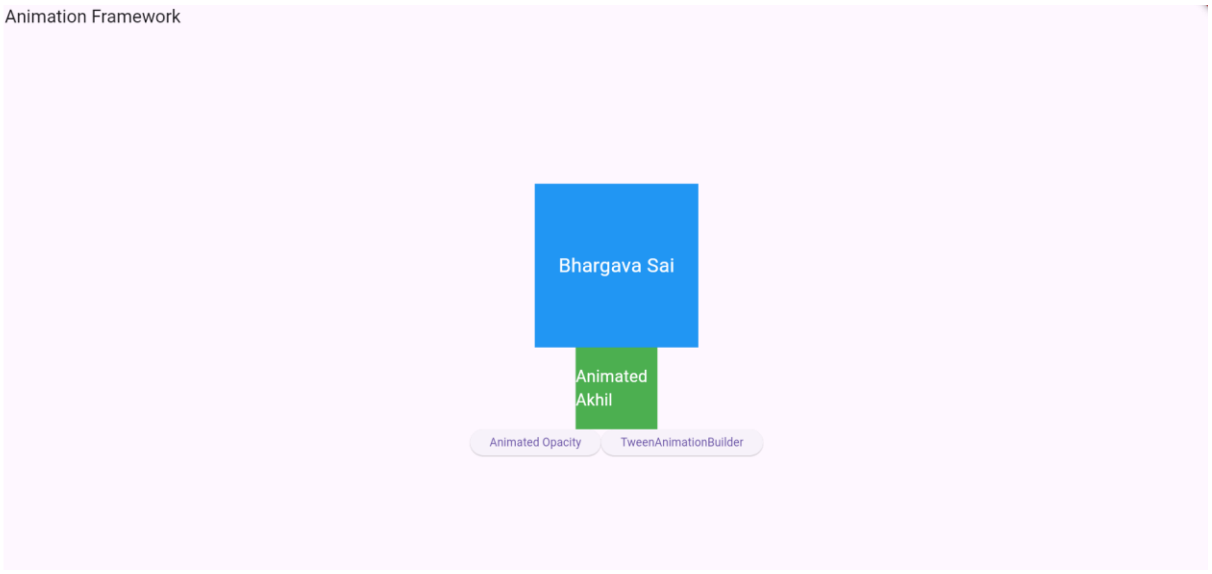

SLIVER APPBAR
Bhargava Sai
Sai Vikas

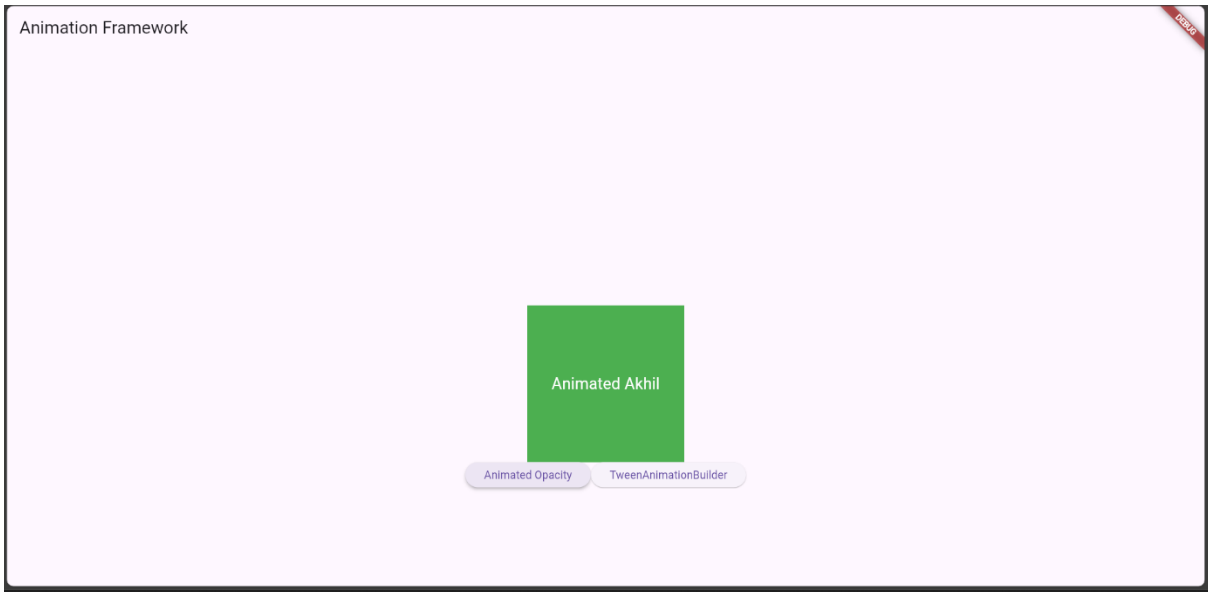
Experiment No: 8.b **Date**:	**Aim**: Experiment with different types of animations (fade, slide, etc.).

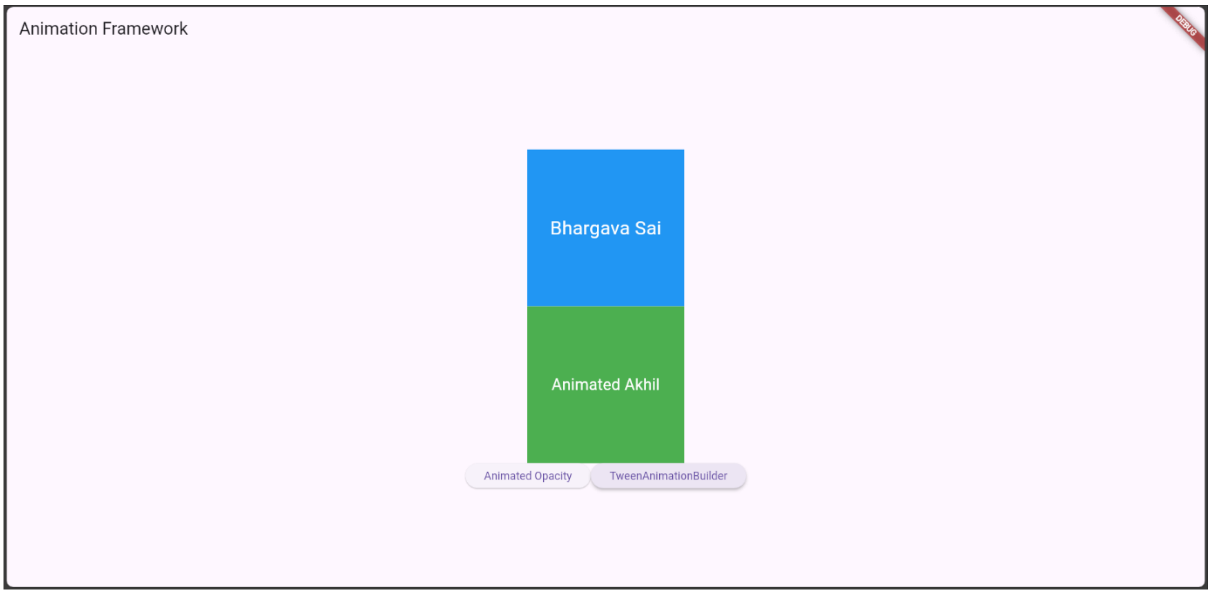
Code:

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

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

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

class AnimatedOpacityExample extends StatefulWidget {
  @override
  _AnimatedOpacityExampleState createState() =>
      _AnimatedOpacityExampleState();
}

class _AnimatedOpacityExampleState extends
State<AnimatedOpacityExample> {
  bool _visible = true;
  bool _isLarge = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
appBar: AppBar(title: Text('Animation Framework')),
body: Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Center(
      child: AnimatedOpacity(
        opacity: _visible ? 1.0 : 0.0,
        duration: Duration(seconds: 1),
        child: Container(
          width: 200,
          height: 200,
          color: Colors.blue,
          child: Center(
            child: Text(
              'Bhargava Sai',
              style: TextStyle(color: Colors.white, fontSize: 24),
            ),
          ),
        ),
      ),
    ),
    TweenAnimationBuilder(
      tween: Tween<double>(begin: 100, end: _isLarge ? 200 : 100),
      duration: Duration(seconds: 1),
      builder: (context, size, child) {
        return Container(
          width: size,
          height: size,
          color: Colors.green,
          child: Center(
            child: Text(
              'Animated Akhil',
                style: TextStyle(color: Colors.white, fontSize: 20),
               ),
              ),
             );
            },
           ),
           Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
             ElevatedButton(
              onPressed: () => setState(() => _visible = !_visible),
              child: Text('Animated Opacity'),
             ),
             ElevatedButton(
              onPressed: () => setState(() => _isLarge = !_isLarge),
              child: Text('TweenAnimationBuilder'),
             ),
            ],
           ),
          ],
         ),
        );
       }
      }
```

Output:


Animation Framework
Bhargava Sai
Animated Akhil
Animated Opacity
TweenAnimationBuilder


Animation Framework
DEBUG
Animated Akhil
Animated Opacity
TweenAnimationBuilder


Animation Framework
DEBUG
Bhargava Sai
Animated Akhil
Animated Opacity
TweenAnimationBuilder

Experiment No: 9.a **Date:**	**Aim**: Write unit tests for UI components.

Unit tests are handy for verifying the behaviour 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:

1. Add the test or flutter_test dependency.
2. Create a test file.
3. Create a class to test.
4. Write a test for our class.
5. Combine multiple tests in a group.
6. Run the tests.

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: flutter pub add dev:test

```
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 l 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 yor Flutter application or package. Test files should always end with _test.dart, this the convention used by the test runner when searching for tests.

When you're finished, the folder structure should look like this:

```
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.

```dart
dart
class Counter {
  int value = 0;

  void increment() => value++;

  void decrement() => value--;
}
```

content_copy

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 top-level expect function. Both of these functions come from the test package.

```
dart
// 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.

```
dart
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
  1. Open the counter_test.dart fi
  2. Go to Run > Run 'tests in counter_test.dart'. You can also press the appropriate keyboard shortcut for your platform.
- VSCode
  1. Open the counter_test.dart fi
  2. 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:

```
flutter test test/counter_test.dart
```

content_copy

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

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

```
flutter test --plain-name "Test start, increment, decrement"
```

Experiment No: 9.b **Date:**	**Aim**: Use Flutter's debugging tools to identify and fix issues

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