LAB 1: Installation of Flutter, Dart and Basics of Dart Language

Aim: To set up the Flutter and Dart development environment and gain an understanding of the Dart programming language.

Objectives:

(a) Install Flutter and Dart SDK to configure the development environment.

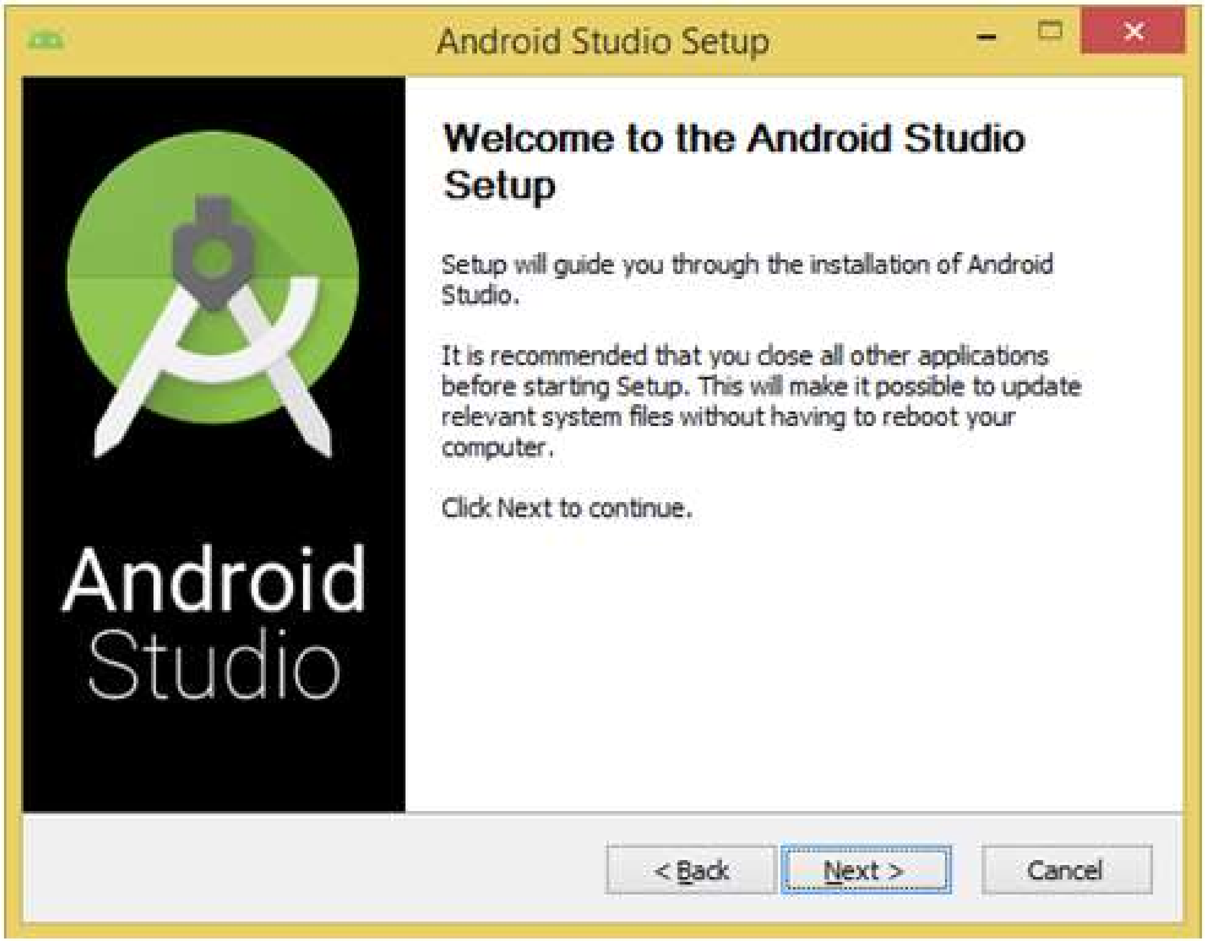
(b) Write and execute a simple Dart program to explore basic language constructs such as variables, data types, and control flow.

1A) INSTALL FLUTTER AND DART SDK.

Important Note: To Install Android Studio we assume that Java is Installed in the system, if not Install Java first and then proceed to the following:

INSTALLING ANDROID STUDIO

Launch the**.exe** file you just downloaded. It will open following screen.



Welcome to the Android Studio Setup

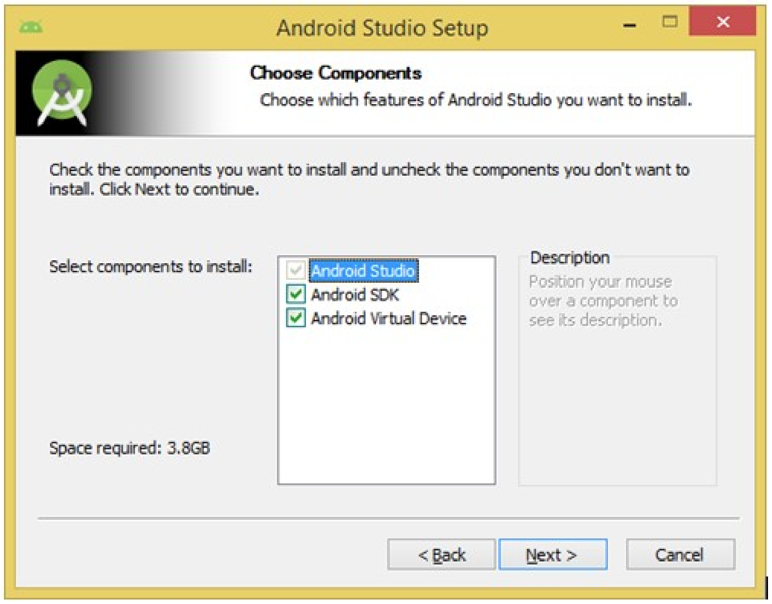
Setup will guide you through the installation of Android Studio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Figure -1

Click on **Next** button.



Choose Components

Choose which features of Android Studio you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

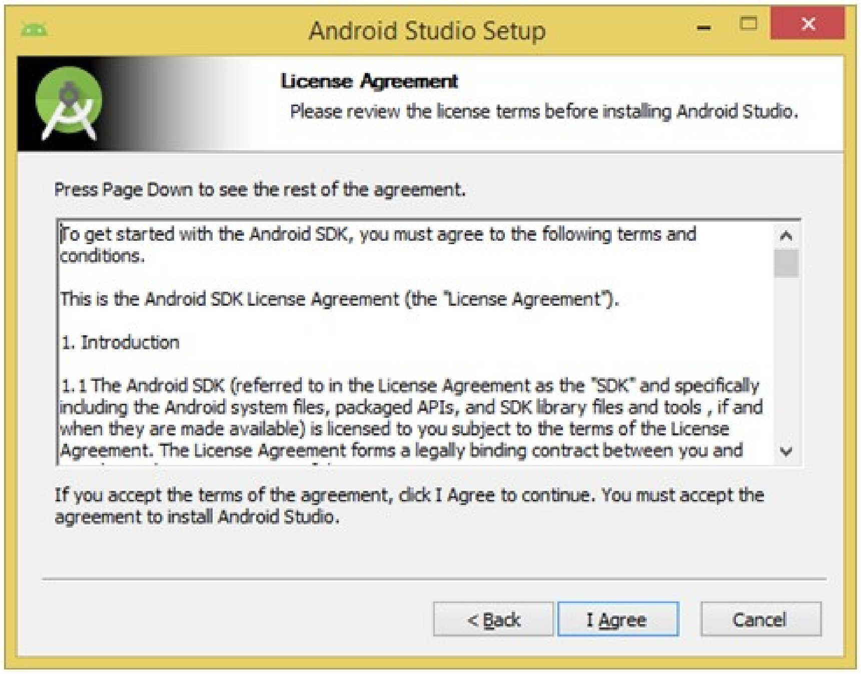
Description

Position your mouse over a component to see its description.

Space required: 3.8GB

Figure -2

Click **Next** button.



License Agreement

Please review the license terms before installing Android Studio.

Press Page Down to see the rest of the agreement.

To get started with the Android SDK, you must agree to the following terms and conditions.

This is the Android SDK License Agreement (the "License Agreement").

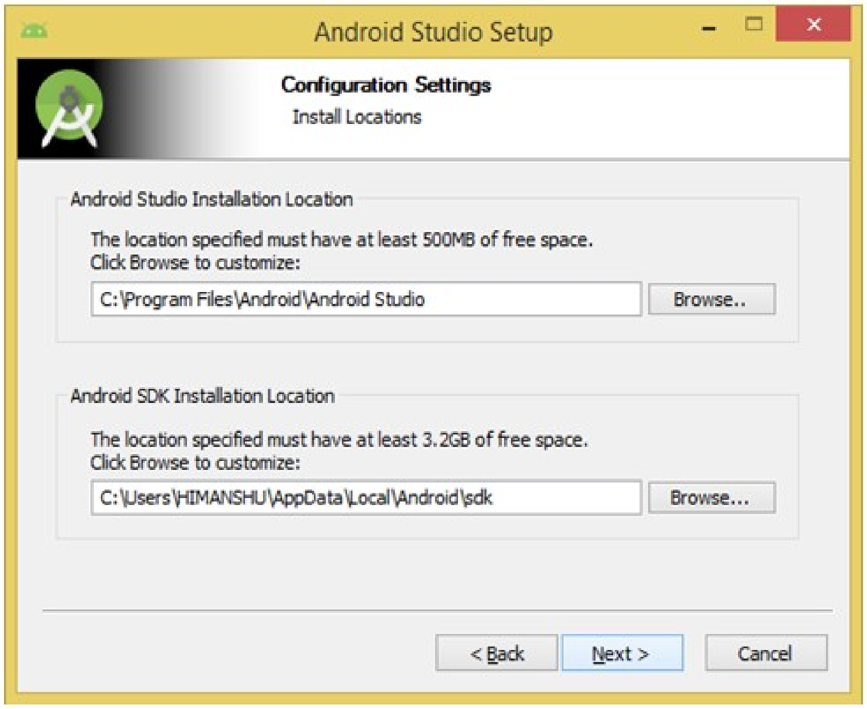
1. Introduction

1.1 The Android SDK (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and SDK library files and tools , if and when they are made available) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Android Studio.

Figure - 3

Click on **I Agree** button.



Android Studio Installation Location

The location specified must have at least 500MB of free space.
Click Browse to customize:

C:\Program Files\Android\Android Studio

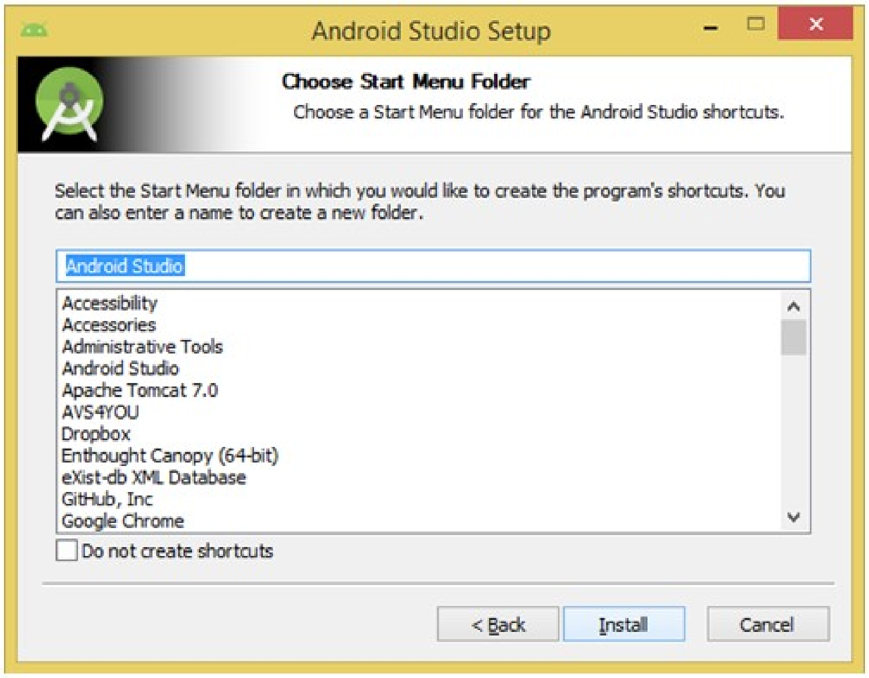
Android SDK Installation Location

The location specified must have at least 3.2GB of free space.
Click Browse to customize:

C:\Users\HIMANSHU\AppData\Local\Android\sdk

Figure - 4

Specify path for Android Studio Installation and Android SDK Installation or use default and press **Next** button.


Android Studio Setup
Choose Start Menu Folder
Choose a Start Menu folder for the Android Studio shortcuts.

Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

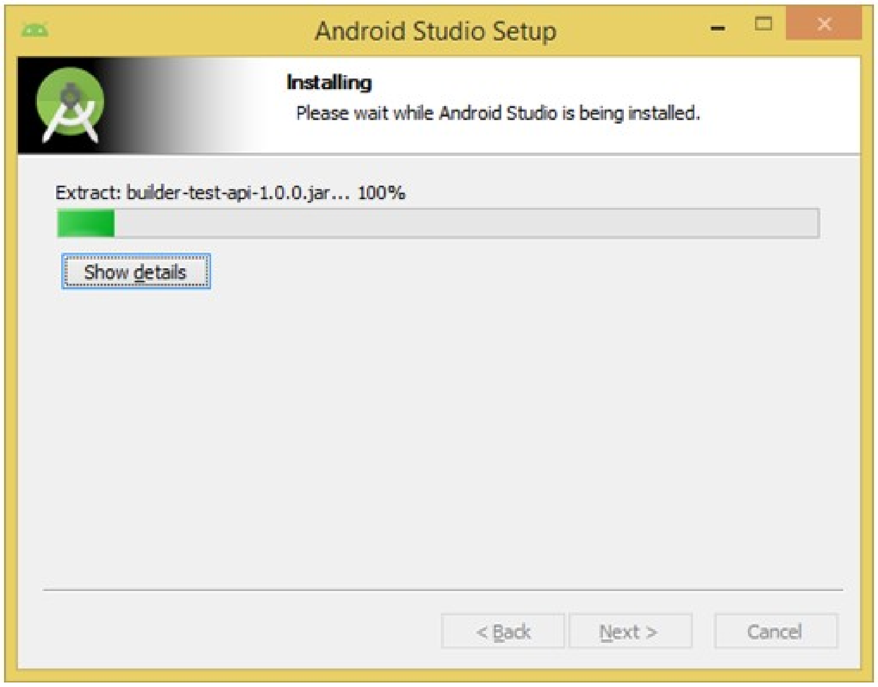
Android Studio

Accessibility
Accessories
Administrative Tools
Android Studio
Apache Tomcat 7.0
AVS4YOU
Dropbox
Enthought Canopy (64-bit)
eXist-db XML Database
GitHub, Inc
Google Chrome

Do not create shortcuts

Figure - 5

Press **Install** Button. This will start installation as shown below.

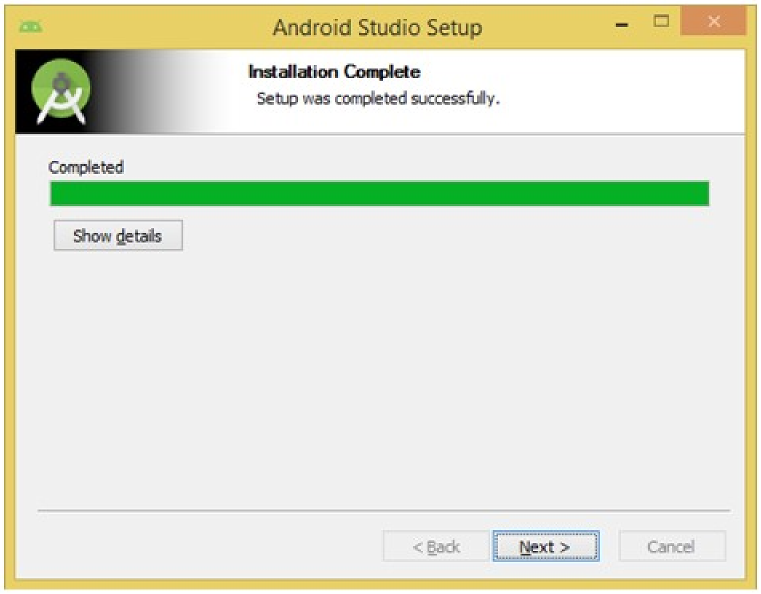

Installing

Please wait while Android Studio is being installed.

Extract: builder-test-api-1.0.0.jar... 100%

Figure - 6

Once installation is finished **completed message** will be displayed as shown below.

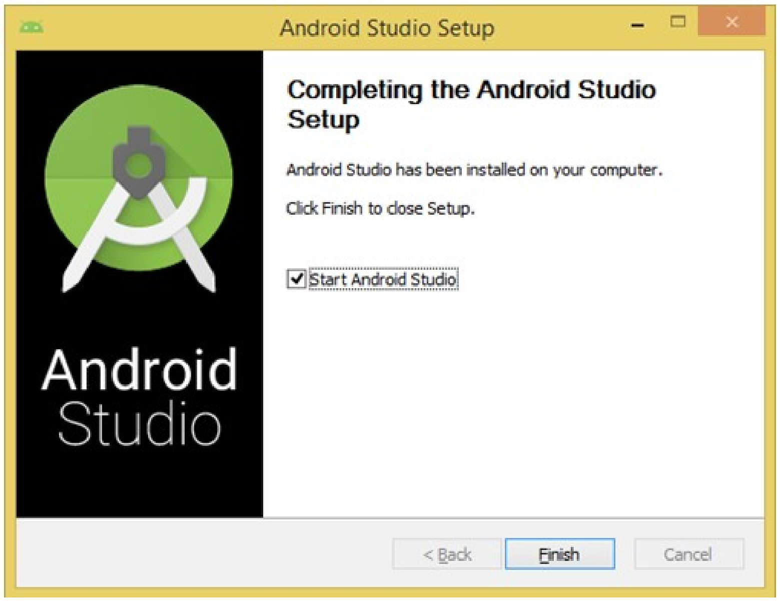

Installation Complete

Setup was completed successfully.

Completed

Figure - 7

Press **Next** button to finish installation and launch Android Studio.



Completing the Android Studio Setup

Android Studio has been installed on your computer.

Click Finish to close Setup.

[✓ Start Android Studio]

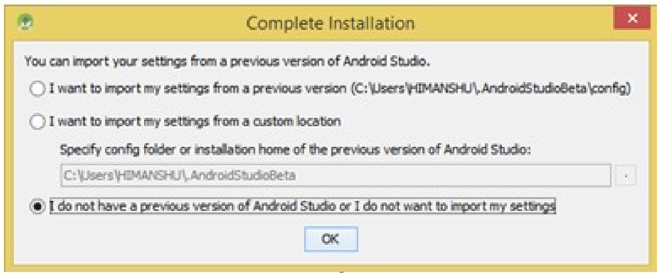
Figure - 8

Important Note: On some Windows systems, the launcher script does not find where Java is installed. If you encounter this problem, you need to set an environment variable indicating the correct location.

Select **Start menu** > **Computer** > **System Properties** > **Advanced System Properties**. Then open **Advanced tab** > **Environment Variables** and add a new system variable **JAVA_HOME** that points to your JDK folder, for example **C:\Program Files\Java\jdk-23\bin.**

LAUNCHING ANDROID STUDIO

Click on **Finish** button to launch Android Studio. It will display following screen.



You can import your settings from a previous version of Android Studio.

I want to import my settings from a previous version (C:\Users\HIMANSHU\.AndroidStudioBeta\config)

I want to import my settings from a custom location

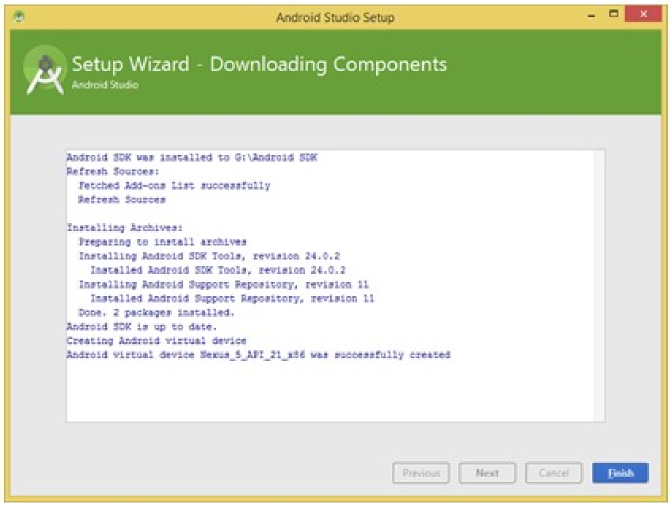
Specify config folder or installation home of the previous version of Android Studio:

C:\Users\HIMANSHU\.AndroidStudioBeta

[✓ I do not have a previous version of Android Studio or I do not want to import my settings]

Figure - 9

Select last option and press **OK**. It will download updates and create virtual device for you.



```
Android SDK was installed to G:\Android SDK
Refresh Sources:
  Fetched Add-ons List successfully
  Refresh Sources

Installing Archives:
  Preparing to install archives
  Installing Android SDK Tools, revision 24.0.2
    Installed Android SDK Tools, revision 24.0.2
  Installing Android Support Repository, revision 11
    Installed Android Support Repository, revision 11
  Done. 2 packages installed.
Android SDK is up to date.
Creating Android virtual device
Android virtual device Nexus_5_API_21_x86 was successfully created
```

Figure - 10

Press **finish** button to start Android Studio with following initial welcome screen.

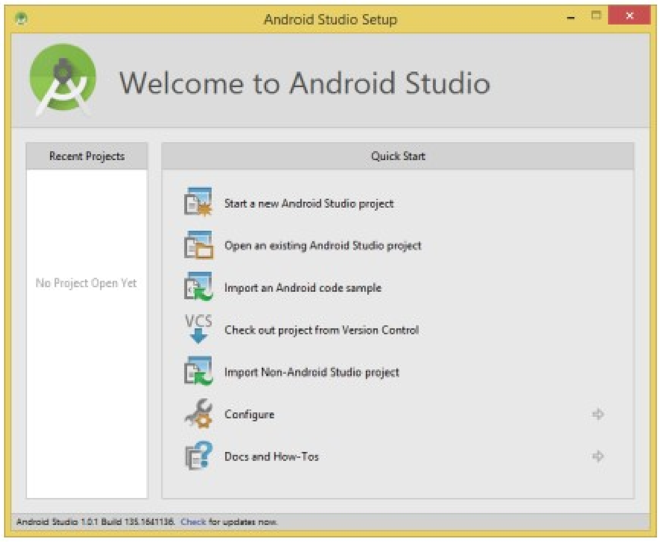
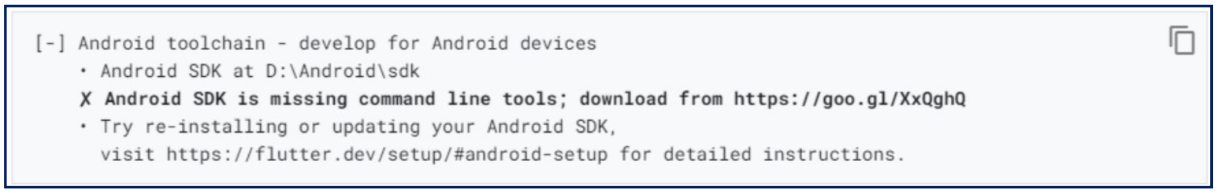

Android Studio Setup
Welcome to Android Studio
Recent Projects
No Project Open Yet
Quick Start
Start a new Android Studio project
Open an existing Android Studio project
Import an Android code sample
VCS
Check out project from Version Control
Import Non-Android Studio project
Configure
Docs and How-Tos
Android Studio 1.0.1 Build 135.1641136. Check for updates now.

Figure - 11

INSTALLING FLUTTER ON WINDOWS

If you are installing Flutter on a Windows PC, follow these instructions below

1. Download the **Flutter SDK** Zip file
2. Extract the zip file and place the flutter folder inside to your desired location.
    a. Don't put it in places that require admin access like **C:\ProgramFiles**!
    b. **Example:** you can create a folder called **dev** in your C: drive root and put it there like this **C:\dev\flutter**.
3. Add Flutter to your Path Variables like so:
    - On your Start Search Bar, type in **env** and select **Edit the Environment Variable for your Account**
    - Click on Environment Variable from **User variables for Account**, There should be a **Path** variable under it
        - If there is no Path variable, you can create a new **User Variable called Path** and give it the value **C:\src\flutter**
        - If there is a Path Variable, just append **C:\src\flutter** by clicking on an empty box and just adding it.
    - Click Okay to finish.
4. Open command prompt with admin privileges and type in flutter doctor. If you see something like this,



```
[-] Android toolchain - develop for Android devices
  • Android SDK at D:\Android\sdk
  ✗ Android SDK is missing command line tools; download from https://goo.gl/XxQghQ
  • Try re-installing or updating your Android SDK,
    visit https://flutter.dev/setup/#android-setup for detailed instructions.
```

CONGRATS! You have flutter installed.

Install the Dart SDK

To install the Dart SDK, use the appropriate package manager for your development platform.
To upgrade the Dart SDK, run the same command to install the Dart SDK from your package manager.

Install using Chocolatey

To install the Dart SDK, use Chocolatey. Chocolatey requires elevated permissions.

Install Chocolatey.

Launch PowerShell with elevated permissions.

```
PS C:\> choco install dart-sdk
```

Change default install path

By default, Chocolatey installs the SDK at **C:\tools\dart-sdk**. To change that location, set the ChocolateyToolsLocation environment variable to your desired installation directory.

Verify your PATH includes Dart

Verify you can run Dart.

```
PS C:\> dart --version
Dart SDK version: 3.2.4 (stable) (Thu Dec 21 19:13:53 2023 +0000) on "win_x64"
```

If your development machine doesn't return a Dart version, add the SDK location to your PATH: In the Windows search box, type **env**. Click **Edit the system environment variables**. Click Environment Variables. In the **user variable section**, select **Path** and click **Edit**. Click **New**, and enter the path of the **dart-sdk** directory. In each window that you just opened, click **Apply** or **OK** to dismiss it and apply the path change.

Upgrade using Chocolatey

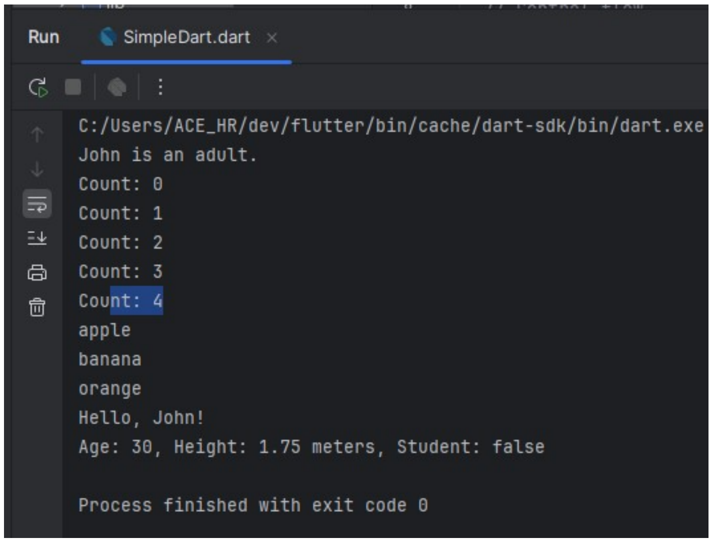
To upgrade the Dart SDK, use the following command.

```
PS C:\> choco upgrade dart-sdk
```

1B) Write a simple Dart program to understand the language basics.

```
void main() {
  // Variables
  String name = "John";
  int age = 30;
  double height = 1.75;
  bool isStudent = false;

  // Control flow
  if (age >= 18) {
    print("$name is an adult.");
  } else {
    print("$name is a minor.");
  }

  // Loops
  for (int i = 0; i < 5; i++) {
    print("Count: $i");
  }

  // Functions
  String greet(String person) {
    return "Hello, $person!";
  }

  // Lists
  List<String> fruits = ["apple", "banana", "orange"];
  fruits.forEach((fruit) => print(fruit));

  // Output
  print(greet(name));
  print("Age: $age, Height: $height meters, Student: $isStudent");
}
```

OUTPUT:



```
C:/Users/ACE_HR/dev/flutter/bin/cache/dart-sdk/bin/dart.exe
John is an adult.
Count: 0
Count: 1
Count: 2
Count: 3
Count: 4
apple
banana
orange
Hello, John!
Age: 30, Height: 1.75 meters, Student: false

Process finished with exit code 0
```

LAB 2: EXPLORING AND IMPLEMENTING FLUTTER WIDGETS AND LAYOUTS

Aim : To understand the use of fundamental Flutter widgets and their role in creating layouts.

Objectives :

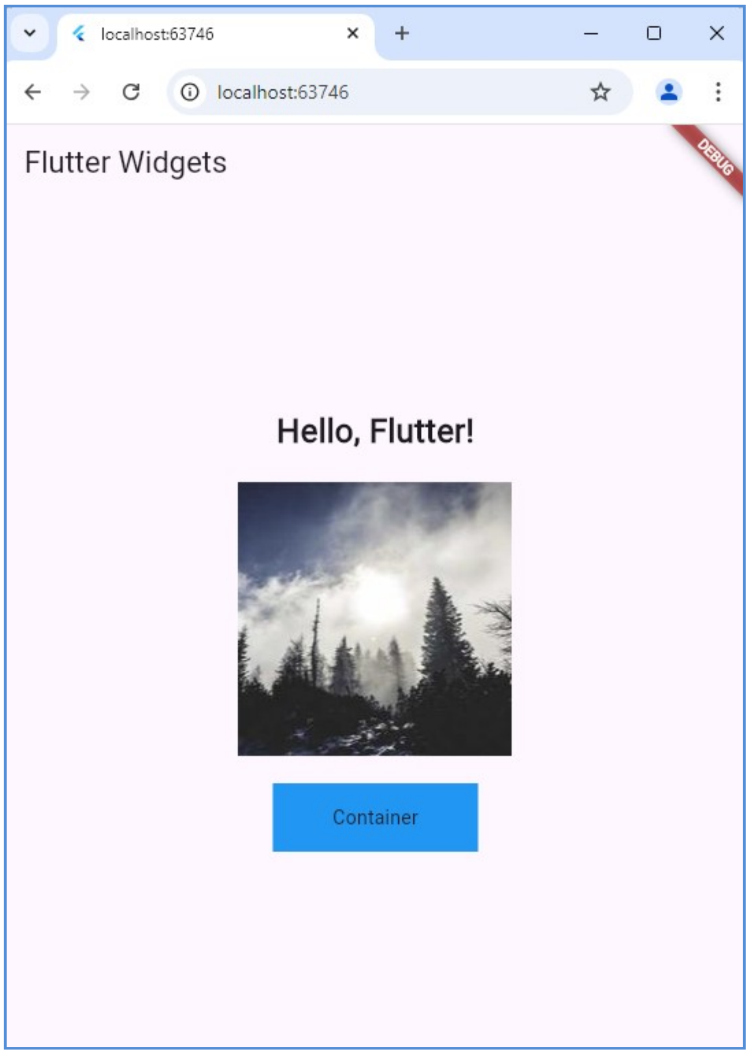
(a) Explore Flutter widgets like `Text`, `Image`, and `Container` to understand their properties and usage.

(b) Design layouts using `Row`, `Column`, and `Stack` widgets to organize UI elements effectively.

2A) EXPLORE VARIOUS FLUTTER WIDGETS (TEXT, IMAGE, CONTAINER, ETC.).

```
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('Flutter Widgets')),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(
                'Hello, Flutter!',
                style: TextStyle(fontSize: 24, fontWeight:
FontWeight.bold),
              ),
              SizedBox(height: 20),
              Image.network(
                'https://picsum.photos/200',
                width: 200,
                height: 200,
              ),
              SizedBox(height: 20),
              Container(
                width: 150,
                height: 50,
                color: Colors.blue,
                child: Center(child: Text('Container')),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

OUTPUT:



Flutter Widgets

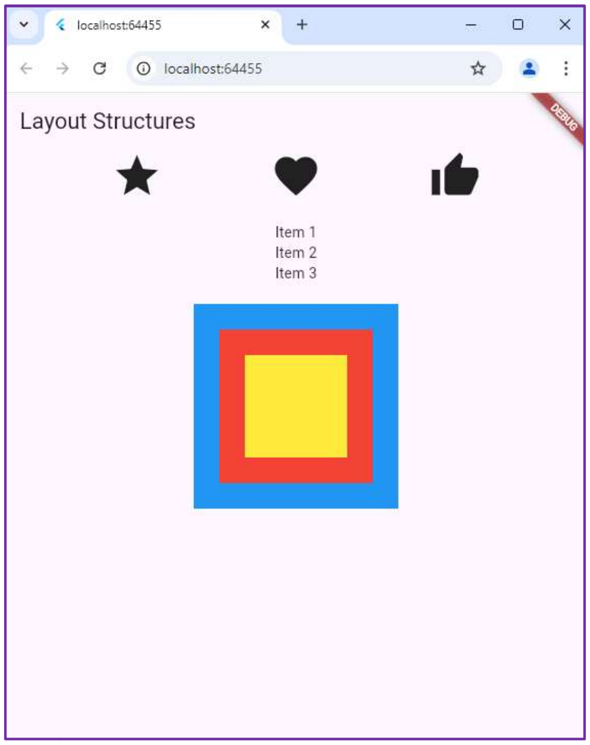
Hello, Flutter!

Container

2B) IMPLEMENT DIFFERENT LAYOUT STRUCTURES USING ROW, COLUMN, AND STACK WIDGETS.

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: Text('Layout Structures')),
        body: Column(
          children: [
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              children: [
                Icon(Icons.star, size: 50),
                Icon(Icons.favorite, size: 50),
                Icon(Icons.thumb_up, size: 50),
              ],
            ),
            SizedBox(height: 20),
            Column(
              children: [
                Text('Item 1'),
                Text('Item 2'),
                Text('Item 3'),
              ],
            ),
            SizedBox(height: 20),
            Stack(
              alignment: Alignment.center,
              children: [
                Container(width: 200, height: 200, color: Colors.blue),
                Container(width: 150, height: 150, color: Colors.red),
                Container(width: 100, height: 100, color: Colors.yellow),
              ],
            ),
          ],
        ),
      ),
    );
  }
}
```

OUTPUT:



Layout Structures

Item 1
Item 2
Item 3

LAB 3: DESIGNING RESPONSIVE USER INTERFACES

Aim: To create a user interface that dynamically adapts to different screen sizes and orientations.

Objectives:

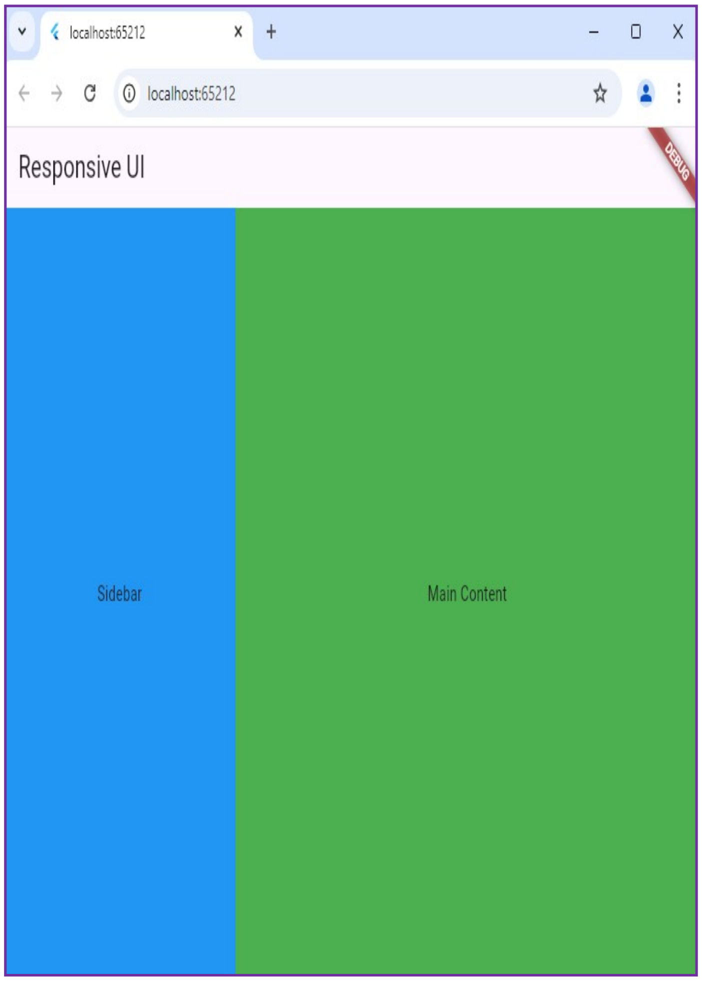
(a) Design a responsive UI that adjusts elements and layouts based on device screen sizes.

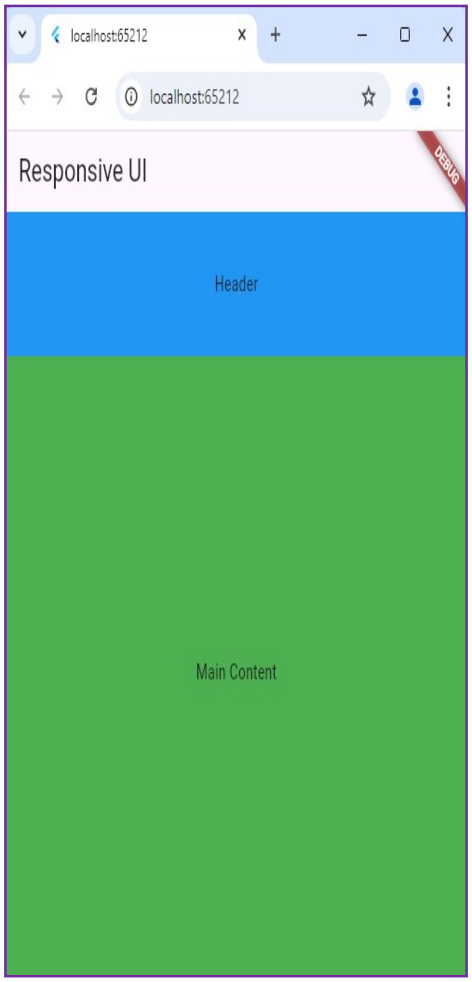
(b) Implement media queries and breakpoints to control the layout and styling for responsiveness.

3A) DESIGN A RESPONSIVE UI THAT ADAPTS TO DIFFERENT SCREEN SIZES.

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: Text('Responsive UI')),
        body: LayoutBuilder(
          builder: (context, constraints) {
            if (constraints.maxWidth > 600) {
              return WideLayout();
            } else {
              return NarrowLayout();
            }
          },
        ),
      ),
    );
  }
}

class WideLayout extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Row(
      children: [
        Expanded(
          flex: 1,
          child: Container(color: Colors.blue, child: Center(child:
Text('Sidebar'))),
        ),
        Expanded(
          flex: 2,
          child: Container(color: Colors.green, child: Center(child:
Text('Main Content'))),
        ),
      ],
    );
  }
}

class NarrowLayout extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Container(
          height: 100,
          color: Colors.blue,
          child: Center(child: Text('Header')),
        ),
        Expanded(
          child: Container(color: Colors.green, child: Center(child:
Text('Main Content'))),
        ),
      ],
    );
  }
}
```

OUTPUT:

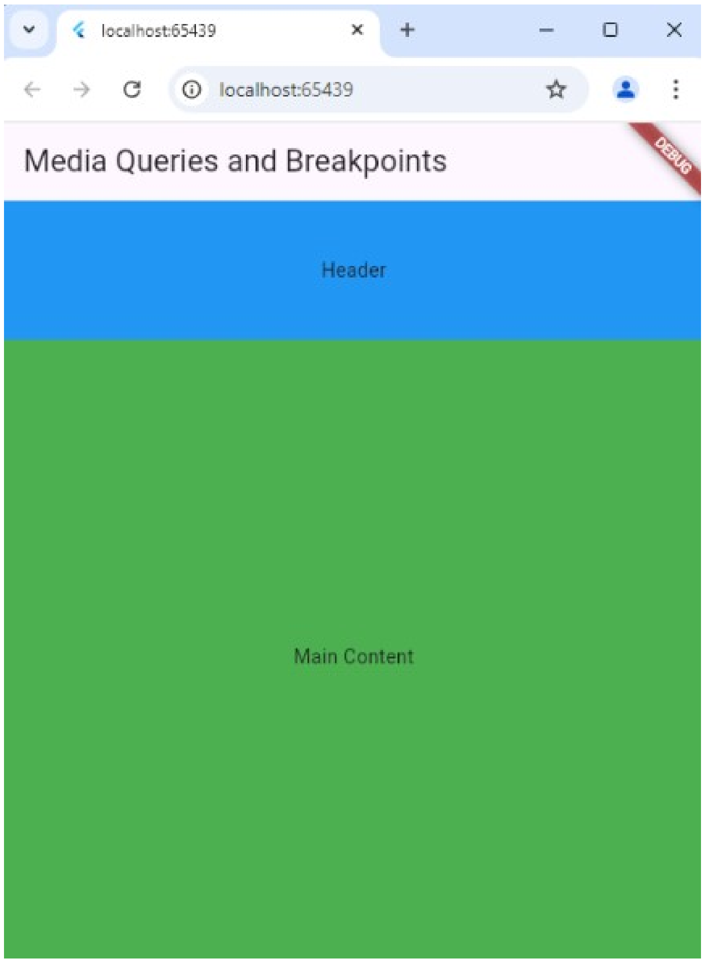




3B) IMPLEMENT MEDIA QUERIES AND BREAKPOINTS FOR RESPONSIVENESS.

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: Text('Media Queries and Breakpoints')),
        body: MediaQuery.of(context).size.width > 600
            ? WideLayout()
            : NarrowLayout(),
      ),
    );
  }
}

class WideLayout extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Row(
      children: [
        Expanded(
          flex: 1,
          child: Container(color: Colors.blue, child: Center(child:
Text('Sidebar'))),
        ),
        Expanded(
          flex: 2,
          child: Container(color: Colors.green, child: Center(child:
Text('Main Content'))),
        ),
      ],
    );
  }
}

class NarrowLayout extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Container(
          height: 100,
          color: Colors.blue,
          child: Center(child: Text('Header')),
        ),
        Expanded(
          child: Container(color: Colors.green, child: Center(child:
Text('Main Content'))),
        ),
      ],
        );
    }
}
```

OUTPUT:



Media Queries and Breakpoints

Header

Main Content

LAB 4: IMPLEMENTING NAVIGATION IN FLUTTER APPLICATIONS

Aim: To enable seamless navigation between different screens in a Flutter application.

Objectives:

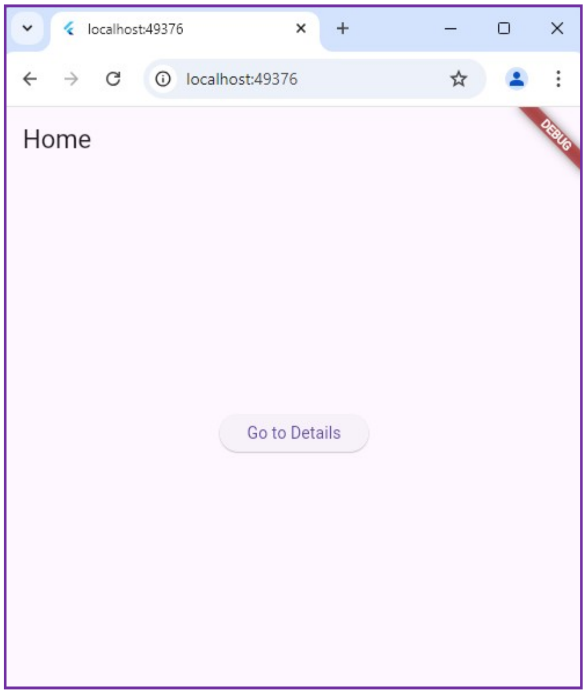
(a) Set up navigation between screens using Flutter's `Navigator` API.

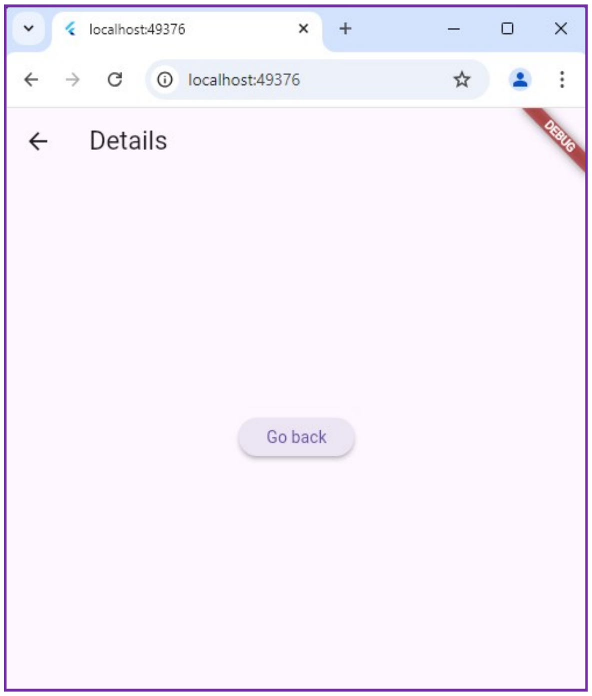
(b) Implement named routes for managing and organizing navigation efficiently.

4A) SET UP NAVIGATION BETWEEN DIFFERENT SCREENS USING NAVIGATOR.

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

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

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

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Home')),
      body: Center(
        child: ElevatedButton(
          child: Text('Go to Details'),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => DetailScreen()),
            );
          },
        ),
      ),
    );
  }
}

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

OUTPUT:



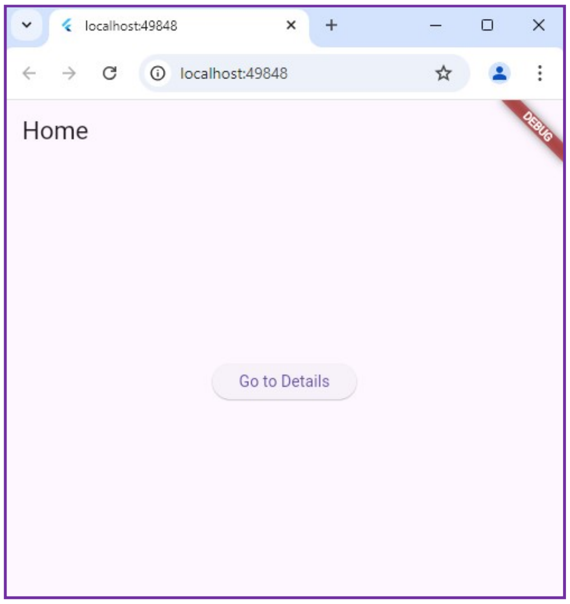


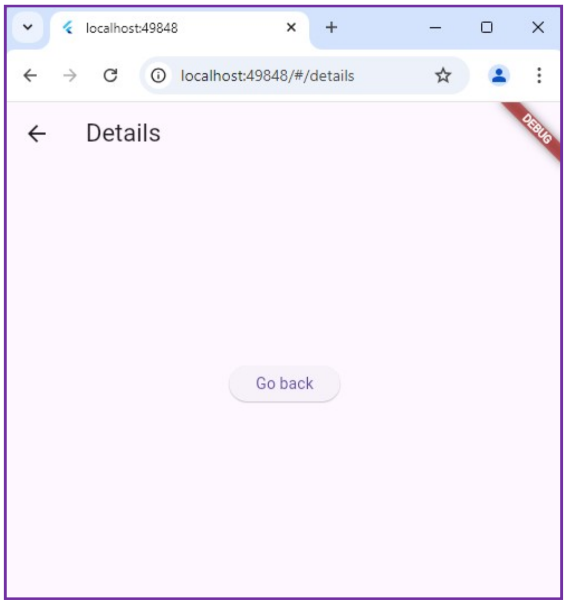
4B) IMPLEMENT NAVIGATION WITH NAMED ROUTES.

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      routes: {
        '/': (context) => HomeScreen(),
        '/details': (context) => DetailScreen(),
      },
    );
  }
}
```

```
class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Home')),
      body: Center(
        child: ElevatedButton(
          child: Text('Go to Details'),
          onPressed: () {
            Navigator.pushNamed(context, '/details');
          },
        ),
      ),
    );
  }
}

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

OUTPUT:





LAB 5: UNDERSTANDING WIDGETS AND MANAGING STATE

Aim: To differentiate between stateful and stateless widgets and manage application state effectively.

Objectives:

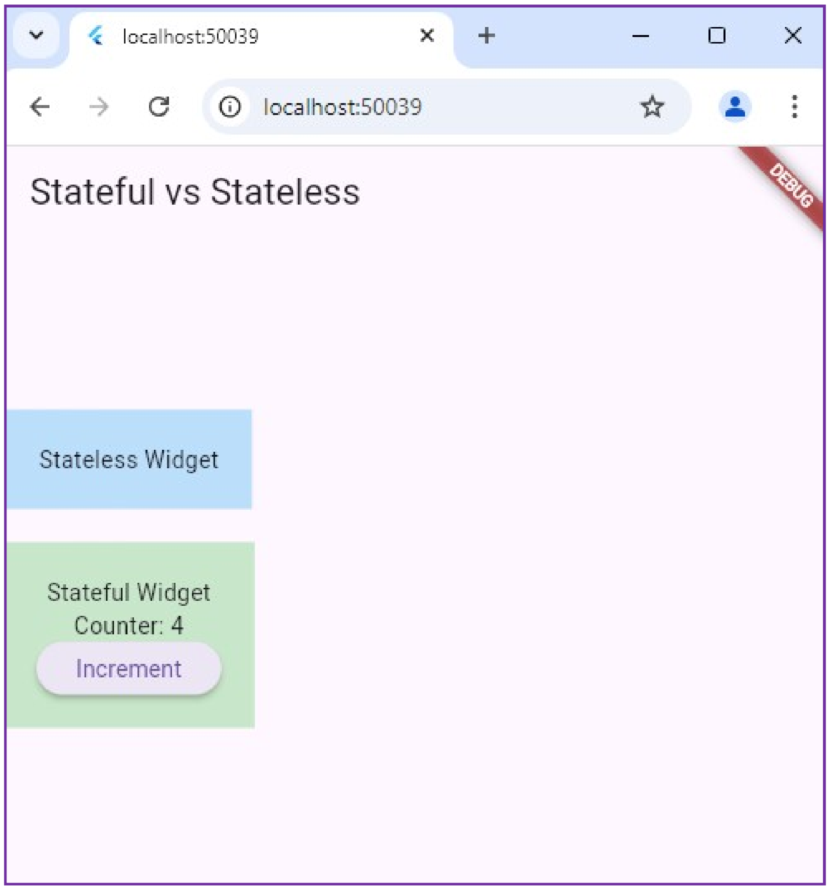
(a) Learn the concepts and use cases of stateful and stateless widgets.

(b) Implement state management using `setState` and the `Provider` package.

5A) LEARN ABOUT STATEFUL AND STATELESS WIDGETS.

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: Text('Stateful vs Stateless')),
        body: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            StatelessExample(),
            SizedBox(height: 20),
            StatefulExample(),
          ],
        ),
      ),
    );
  }
}

class StatelessExample extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.all(20),
      color: Colors.blue[100],
      child: Text('Stateless Widget'),
    );
  }
}

class StatefulExample extends StatefulWidget {
  @override
  _StatefulExampleState createState() => _StatefulExampleState();
}

class _StatefulExampleState extends State<StatefulExample> {
  int _counter = 0;
  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.all(20),
      color: Colors.green[100],
      child: Column(
        children: [
          Text('Stateful Widget'),
          Text('Counter: $_counter'),
          ElevatedButton(
            onPressed: _incrementCounter,
            child: Text('Increment'),
          ),
        ],
      ),
    );
  }
}
```

OUTPUT:



Stateful vs Stateless

Stateless Widget

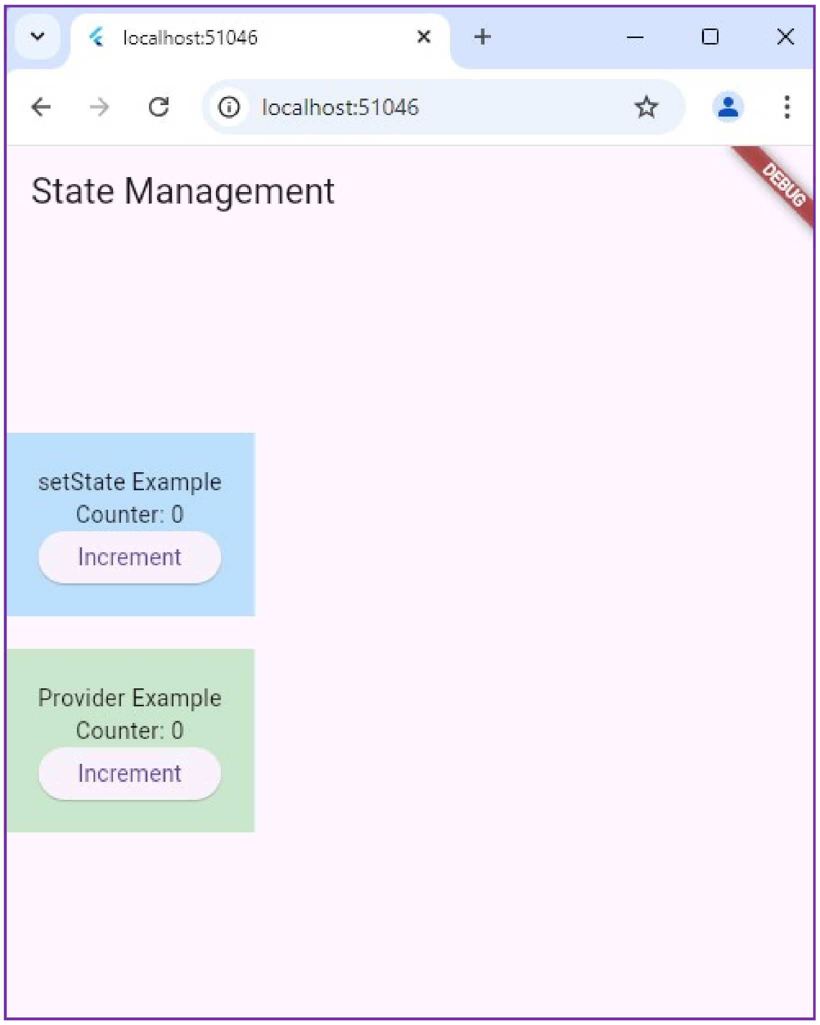
Stateful Widget

Counter: 4

5B) IMPLEMENT STATE MANAGEMENT USING SET STATE AND PROVIDER.

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

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

class CounterModel extends ChangeNotifier {
  int _count = 0;
  int get count => _count;

  void increment() {
    _count++;
    notifyListeners();
  }
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: Text('State Management')),
        body: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            SetStateExample(),
            SizedBox(height: 20),
            ProviderExample(),
          ],
        ),
      ),
    );
  }
}

class SetStateExample extends StatefulWidget {
  @override
  _SetStateExampleState createState() => _SetStateExampleState();
}

class _SetStateExampleState extends State<SetStateExample> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }
  @override
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.all(20),
      color: Colors.blue[100],
      child: Column(
        children: [
          Text('setState Example'),
          Text('Counter: $_counter'),
          ElevatedButton(
            onPressed: _incrementCounter,
            child: Text('Increment'),
          ),
        ],
      ),
    );
  }
}

class ProviderExample extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.all(20),
      color: Colors.green[100],
      child: Column(
        children: [
          Text('Provider Example'),
          Consumer<CounterModel>(
            builder: (context, counter, child) {
              return Text('Counter: ${counter.count}');
            },
          ),
          ElevatedButton(
            onPressed: () {
              Provider.of<CounterModel>(context, listen:
false).increment();
            },
            child: Text('Increment'),
          ),
        ],
      ),
    );
  }
}
```

OUTPUT:



State Management

setState Example
Counter: 0

Provider Example
Counter: 0

LAB 6: CREATING CUSTOM WIDGETS AND STYLING APPLICATIONS

Aim: To design reusable UI components and apply consistent styling across the application.

Objectives:

(a) Create custom widgets for specific UI requirements.

(b) Apply themes and custom styles to enhance the visual appeal of the application.

6A) CREATE CUSTOM WIDGETS FOR SPECIFIC UI ELEMENTS.

CUSTOM BUTTON

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

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

  const CustomButton({
    Key? key,
    required this.text,
    required this.onPressed,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      onPressed: onPressed,
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.blue,
        foregroundColor: Colors.white,
        padding: EdgeInsets.symmetric(horizontal: 20, vertical: 12),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(30),
        ),
      ),
      child: Text(
        text,
        style: TextStyle(fontSize: 18),
      ),
    );
  }
}
```

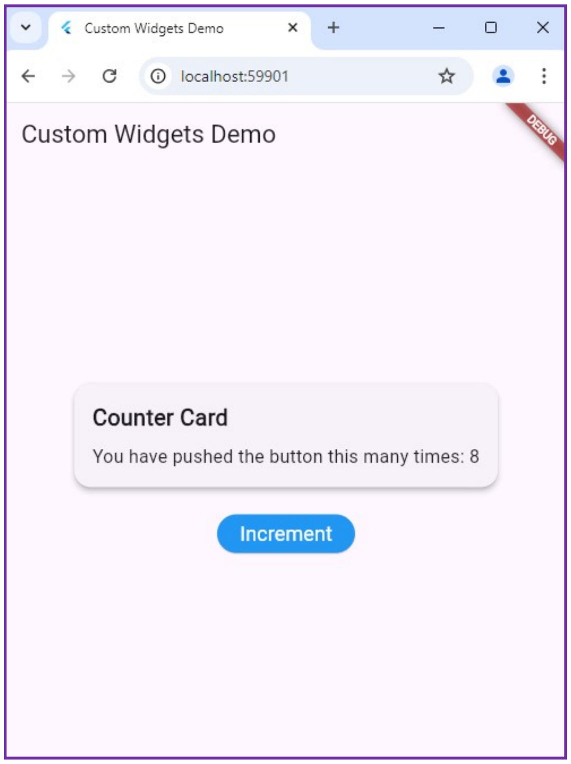
CUSTOM_CARD

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

class CustomCard extends StatelessWidget {
  final String title;
  final String content;

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

  @override
  Widget build(BuildContext context) {
    return Card(
      elevation: 4,
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(15),
      ),
      child: Padding(
        padding: EdgeInsets.all(16),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              title,
              style: TextStyle(
                fontSize: 20,
                fontWeight: FontWeight.bold,
              ),
            ),
            SizedBox(height: 8),
            Text(
              content,
              style: TextStyle(fontSize: 16),
            ),
          ],
        ),
      ),
    );
  }
}
```

Main

```
import 'package:flutter/material.dart';
import 'widgets/custom_button.dart';
import 'widgets/custom_card.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Custom Widgets Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(title: 'Custom Widgets Demo'),
    );
  }
}

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

  final String title;

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

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            CustomCard(
              title: 'Counter Card',
              content: 'You have pushed the button this many times:
$_counter',
            ),
            SizedBox(height: 20),
            CustomButton(
              text: 'Increment',
              onPressed: _incrementCounter,
            ),
          ],
        ),
      ),
    );
  }
}
```

OUTPUT:



Custom Widgets Demo

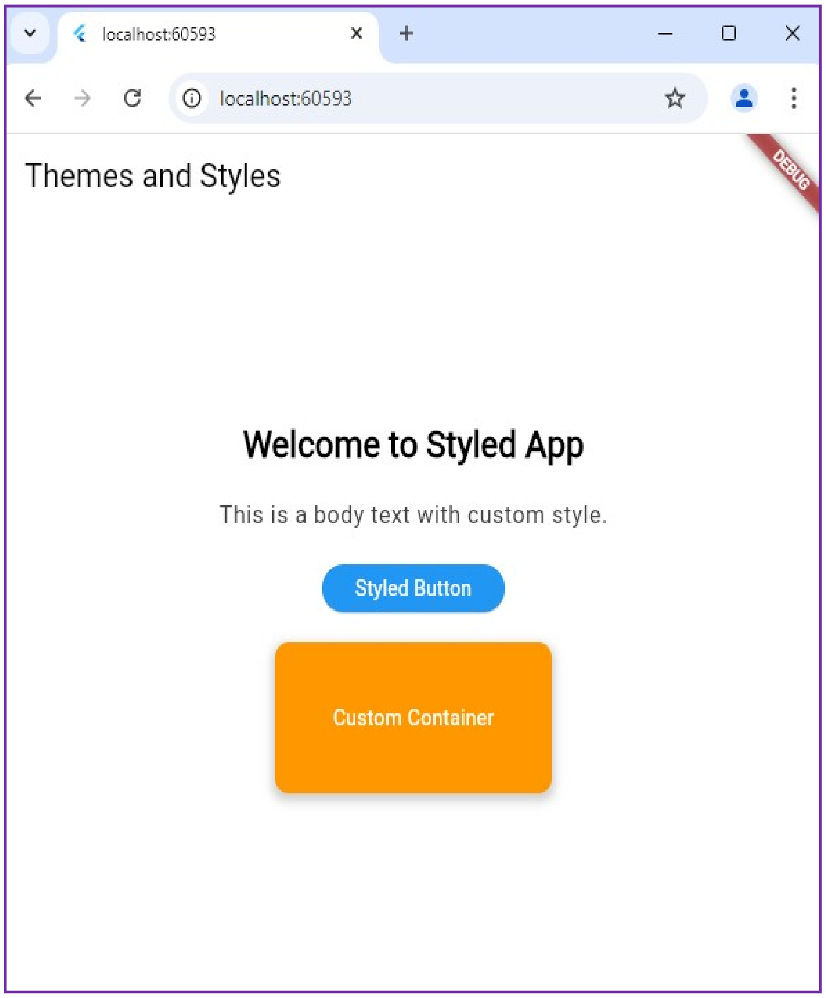
Counter Card

You have pushed the button this many times: 8

6B) APPLY STYLING USING THEMES AND CUSTOM STYLES.

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        primarySwatch: Colors.blue,
        colorScheme: ColorScheme.fromSwatch().copyWith(secondary:
Colors.orange), // Updated accentColor
        textTheme: TextTheme(
          headlineLarge: TextStyle(fontSize: 24, fontWeight:
FontWeight.bold), // Updated text style
          bodyLarge: TextStyle(fontSize: 16, color: Colors.grey[800]), //
Updated text style
        ),
        elevatedButtonTheme: ElevatedButtonThemeData(
          style: ElevatedButton.styleFrom(
            backgroundColor: Colors.blue, // Background color for
ElevatedButton
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(20),
            ),
          ),
        ),
      ),
      home: HomePage(),
    );
  }
}

class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Themes and Styles')),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              'Welcome to Styled App',
              style: Theme.of(context).textTheme.headlineLarge,
            ),
            SizedBox(height: 20),
            Text(
              'This is a body text with custom style.',
              style: Theme.of(context).textTheme.bodyLarge,
            ),
            SizedBox(height: 20),
            ElevatedButton(
              child: Text('Styled Button',style: TextStyle(color:
Colors.white),),
              onPressed: () {},
            ),
            SizedBox(height: 20),
            Container(
              width: 200,
              height: 100,
              decoration: BoxDecoration(
                color: Theme.of(context).colorScheme.secondary, // Updated
accent color usage
                borderRadius: BorderRadius.circular(10),
                boxShadow: [
                  BoxShadow(
                    color: Colors.grey.withOpacity(0.5),
                    spreadRadius: 2,
                    blurRadius: 5,
                    offset: Offset(0, 3),
                  ),
                ],
              ),
              child: Center(
                child: Text(
                  'Custom Container',
                  style: TextStyle(color: Colors.white),
                ),
              ),
            ),
            ],
          ),
        ),
      );
    }
}
```

OUTPUT:



Themes and Styles

Welcome to Styled App

This is a body text with custom style.

LAB 7: DESIGNING AND VALIDATING FORMS

Aim: To build user input forms with validation and error handling mechanisms.

Objectives:

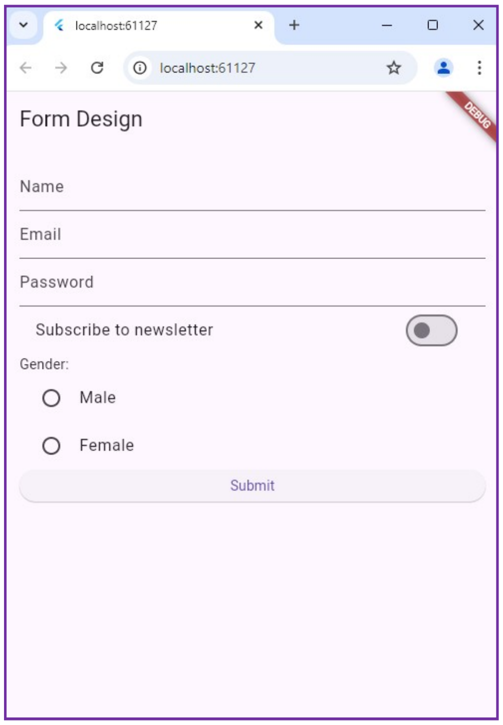
(a) Design a form with various input fields such as text, dropdown, and checkboxes.

(b) Implement form validation and error handling to ensure data integrity and improve user experience.

7A) DESIGN A FORM WITH VARIOUS INPUT FIELDS.

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

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

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

class FormPage extends StatefulWidget {
  @override
  _FormPageState createState() => _FormPageState();
}

class _FormPageState extends State<FormPage> {
  final _formKey = GlobalKey<FormState>();
  String _name = '';
  String _email = '';
  String _password = '';
  bool _subscribe = false;
  String _gender = '';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Form Design')),
      body: Padding(
        padding: EdgeInsets.all(16.0),
        child: Form(
          key: _formKey,
          child: ListView(
            children: [
              TextFormField(
                decoration: InputDecoration(labelText: 'Name'),
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Please enter your name';
                  }
                  return null;
                },
                onSaved: (value) => _name = value!,
              ),
              TextFormField(
                decoration: InputDecoration(labelText: 'Email'),
                keyboardType: TextInputType.emailAddress,
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Please enter your email';
                  }
                  if (!value.contains('@')) {
                    return 'Please enter a valid email';
                  }
                  return null;
                },
                onSaved: (value) => _email = value!,
              ),
              TextFormField(
                decoration: InputDecoration(labelText: 'Password'),
                obscureText: true,
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Please enter a password';
                  }
                  if (value.length < 6) {
                    return 'Password must be at least 6 characters long';
                  }
                  return null;
                },
                onSaved: (value) => _password = value!,
              ),
              SwitchListTile(
                title: Text('Subscribe to newsletter'),
                value: _subscribe,
                onChanged: (bool value) {
                  setState(() {
                    _subscribe = value;
                  });
                },
              ),
              Text('Gender:'),
              RadioListTile(
                title: Text('Male'),
                value: 'male',
                groupValue: _gender,
                onChanged: (value) {
                  setState(() {
                    _gender = value.toString();
                  });
                },
              ),
              RadioListTile(
                title: Text('Female'),
                value: 'female',
                groupValue: _gender,
                onChanged: (value) {
                  setState(() {
                    _gender = value.toString();
                  });
                },
              ),
              ElevatedButton(
                child: Text('Submit'),
                onPressed: () {
                  if (_formKey.currentState!.validate()) {
                    _formKey.currentState!.save();
                    // Process the form data
                    print('Name: $_name');
                    print('Email: $_email');
                    print('Password: $_password');
                    print('Subscribe: $_subscribe');
                    print('Gender: $_gender');
                  }
                },
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

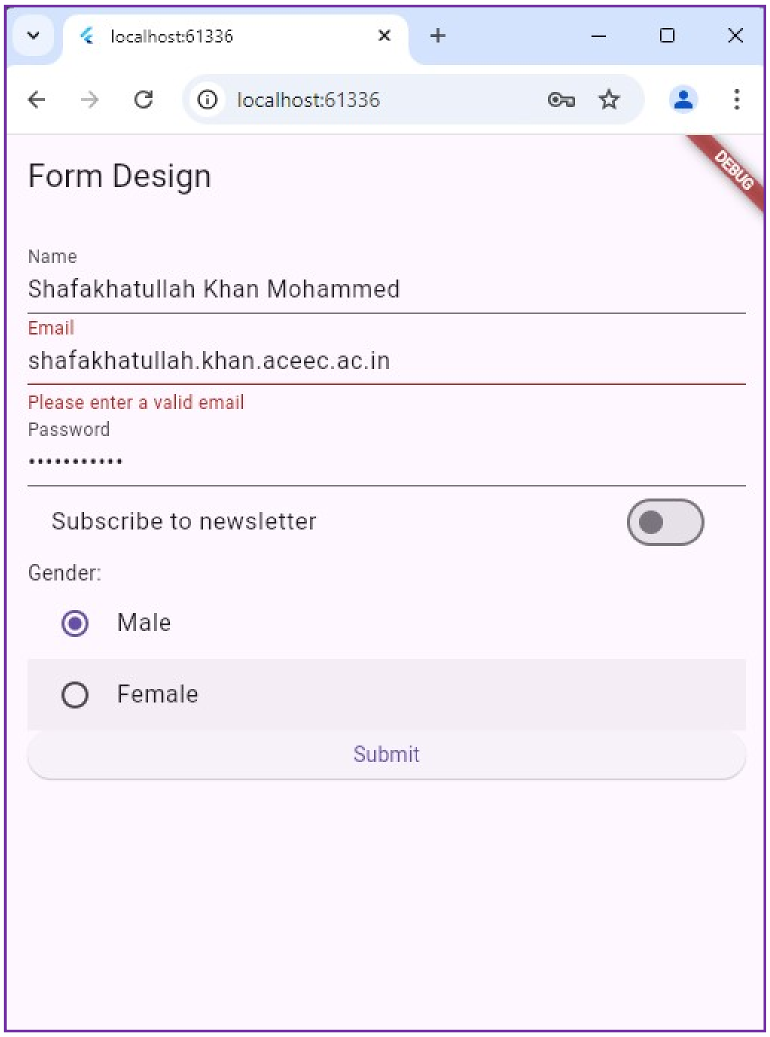
OUTPUT:



b) Implement form validation and error handling. (Code is implemented in Program 7(a).)

```
ElevatedButton(
                child: Text('Submit'),
                onPressed: () {
                  if (_formKey.currentState!.validate()) {
                    _formKey.currentState!.save();
                    // Process the form data
                    print('Name: $_name');
                    print('Email: $_email');
                    print('Password: $_password');
                    print('Subscribe: $_subscribe');
                    print('Gender: $_gender');
                  }
                },
              ),
```

OUTPUT:



Form Design

Name
Shafakhatullah Khan Mohammed

Email
shafakhatullah.khan.aceec.ac.in

Please enter a valid email

Password
•••••••••••

Subscribe to newsletter

Gender:

Male

Female

LAB 8: ADDING ANIMATIONS TO THE USER INTERFACE

Aim: To enhance the user experience by integrating animations into the UI.

Objectives:

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

(b) Experiment with different animation types, such as fade and slide, to create visually engaging interactions.

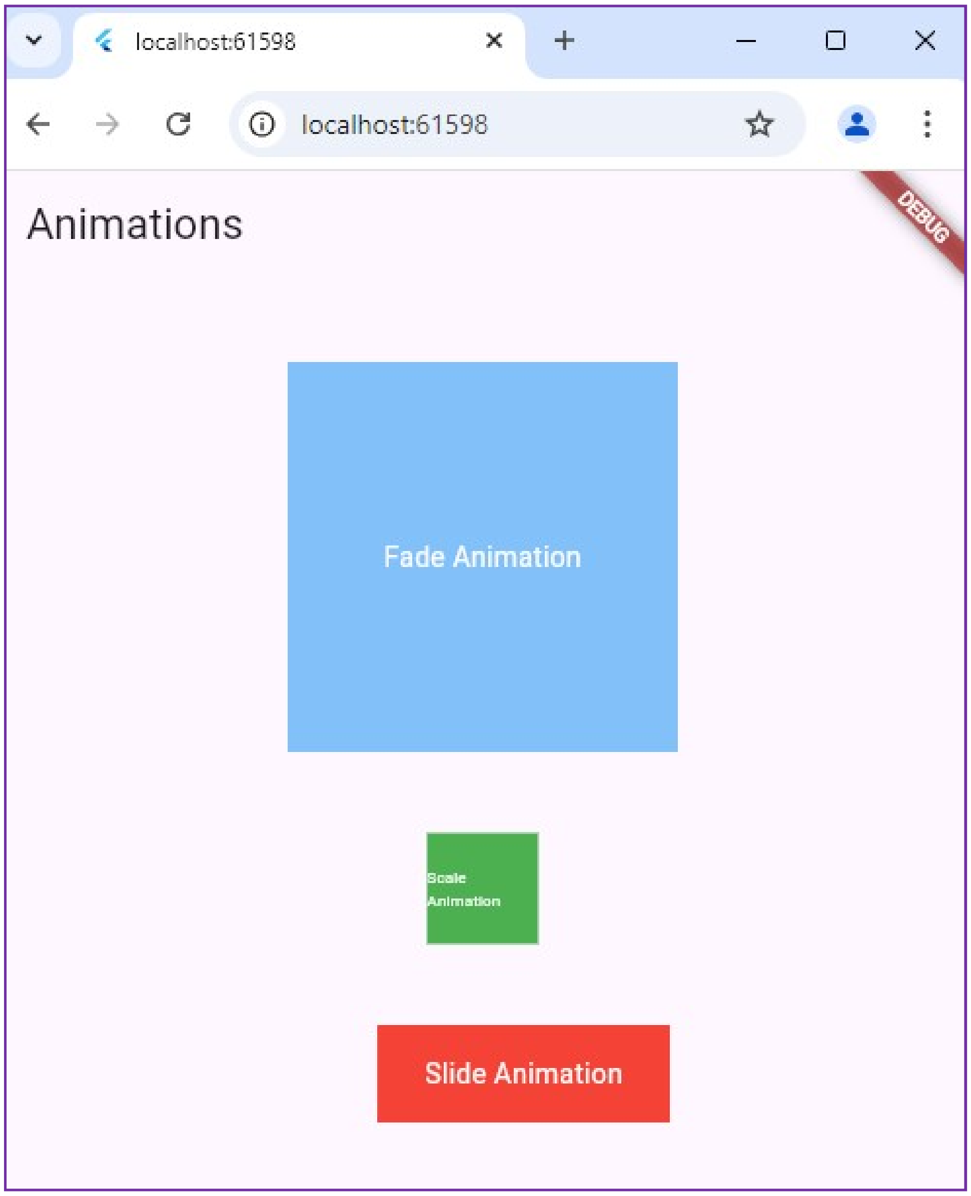
8 A) ADD ANIMATIONS TO UI ELEMENTS USING FLUTTER'S ANIMATION FRAMEWORK.

8 B) EXPERIMENT WITH DIFFERENT TYPES OF ANIMATIONS (FADE, SLIDE, ETC.).

ONE PROGRAM FOR BOTH

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

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

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

class AnimationPage extends StatefulWidget {
  @override
  _AnimationPageState createState() => _AnimationPageState();
}

class _AnimationPageState extends State<AnimationPage> with
SingleTickerProviderStateMixin {

  late AnimationController _controller;
  late Animation<double> _animation;

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      duration: const Duration(seconds: 2),
      vsync: this,
    )..repeat(reverse: true);
    _animation = CurvedAnimation(parent: _controller, curve:
Curves.easeInOut);
  }

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

  @override
  Widget build(BuildContext context) {

    return Scaffold(
      appBar: AppBar(title: Text('Animations')),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            FadeTransition(
              opacity: _animation,
              child: Container(
                width: 200,
                height: 200,
                color: Colors.blue,
                child: Center(
                  child: Text(
                    'Fade Animation',
                    style: TextStyle(color: Colors.white),
                  ),
                ),
              ),
            ),
            SizedBox(height: 20),
            ScaleTransition(
              scale: _animation,
              child: Container(
                width: 100,
                height: 100,
                color: Colors.green,
                child: Center(
                  child: Text(
                    'Scale Animation',
                    style: TextStyle(color: Colors.white),
                  ),
                ),
              ),
            ),
            SizedBox(height: 20),
            SlideTransition(
              position: Tween<Offset>(
                begin: Offset(-1, 0),
                end: Offset(1, 0),
              ).animate(_animation),
              child: Container(
                width: 150,
                height: 50,
                color: Colors.red,
                child: Center(
                  child: Text(
                    'Slide Animation',
                    style: TextStyle(color: Colors.white),
                  ),
                    ),
                  ),
                ),
              ],
            ),
          ),
        );
  }
}
```

OUTPUT:


localhost:61598
localhost:61598
DEBUG
Animations
Fade Animation
Scale Animation
Slide Animation

LAB 9: WORKING WITH REST APIS

Aim: To fetch and display data from a REST API in a Flutter application.

Objectives:

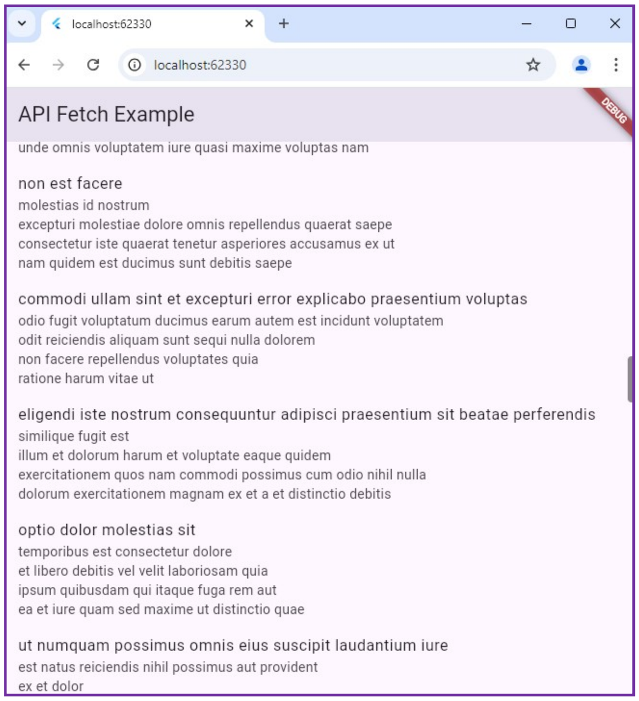
(a) Fetch data from a REST API using Flutter's HTTP package.

(b) Display the fetched data in a meaningful and user friendly way within the UI.

9 A) FETCH DATA FROM A REST API.

```
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(
      home: APIFetchPage(),
    );
  }
}

class APIFetchPage extends StatefulWidget {
  @override
  _APIFetchPageState createState() => _APIFetchPageState();
}

class _APIFetchPageState extends State<APIFetchPage> {
  List<dynamic> _posts = [];
  bool _isLoading = false;

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

  Future<void> _fetchPosts() async {
    setState(() {
      _isLoading = true;
    });

    try {
      final response = await
http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts'));
      if (response.statusCode == 200) {
        setState(() {
          _posts = json.decode(response.body);
          _isLoading = false;
        });
      } else {
        throw Exception('Failed to load posts');
      }
    } catch (e) {
      setState(() {
        _isLoading = false;
      });
      print('Error: $e');
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('API Fetch Example')),
      body: _isLoading
          ? Center(child: CircularProgressIndicator())
          : ListView.builder(
        itemCount: _posts.length,
        itemBuilder: (context, index) {
          final post = _posts[index];
          return ListTile(
            title: Text(post['title']),
            subtitle: Text(post['body']),
          );
        },
      ),
    );
  }
}
```

OUTPUT:



API Fetch Example

unde omnis voluptatem iure quasi maxime voluptas nam

non est facere
molestias id nostrum
excepturi molestiae dolore omnis repellendus quaerat saepe
consectetur iste quaerat tenetur asperiores accusamus ex ut
nam quidem est ducimus sunt debitis saepe

commodi ullam sint et excepturi error explicabo praesentium voluptas
odio fugit voluptatum ducimus earum autem est incidunt voluptatem
odit reiciendis aliquam sunt sequi nulla dolorem
non facere repellendus voluptates quia
ratione harum vitae ut

eligendi iste nostrum consequuntur adipisci praesentium sit beatae perferendis
similique fugit est
illum et dolorum harum et voluptate eaque quidem
exercitationem quos nam commodi possimus cum odio nihil nulla
dolorum exercitationem magnam ex et a et distinctio debitis

optio dolor molestias sit
temporibus est consectetur dolore
et libero debitis vel velit laboriosam quia
ipsum quibusdam qui itaque fuga rem aut
ea et iure quam sed maxime ut distinctio quae

ut numquam possimus omnis eius suscipit laudantium iure
est natus reiciendis nihil possimus aut provident
ex et dolor

9 B) DISPLAY THE FETCHED DATA IN A MEANINGFUL WAY IN THE UI.

POST_ITEM

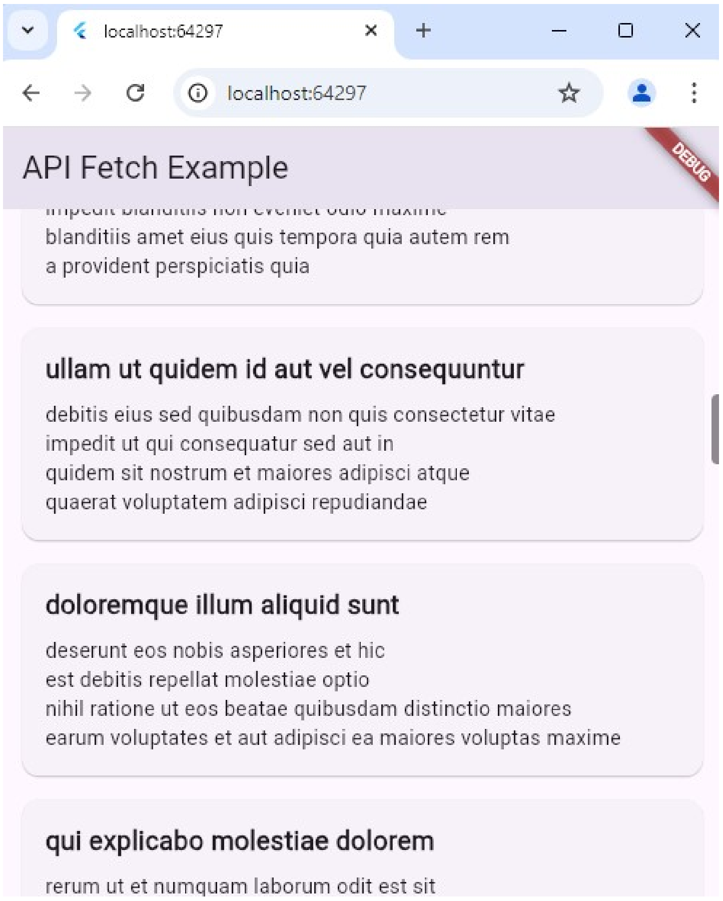
```
import 'package:flutter/material.dart';

class PostItem extends StatelessWidget {
  final String title;
  final String body;

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

  @override
  Widget build(BuildContext context) {
    return Card(
      margin: EdgeInsets.symmetric(horizontal: 16, vertical: 8),
      child: Padding(
        padding: EdgeInsets.all(16),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              title,
              style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
            ),
            SizedBox(height: 8),
            Text(
              body,
              style: TextStyle(fontSize: 14),
            ),
          ],
        ),
      ),
    );
  }
}
```

POST_MAIN

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

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

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

```
}

class APIFetchPage extends StatefulWidget {
  @override
  _APIFetchPageState createState() => _APIFetchPageState();
}

class _APIFetchPageState extends State<APIFetchPage> {
  List<dynamic> _posts = [];
  bool _isLoading = false;

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

  Future<void> _fetchPosts() async {
    setState(() {
      _isLoading = true;
    });

    try {
      final response = await
http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts'));
      if (response.statusCode == 200) {
        setState(() {
          _posts = json.decode(response.body);
          _isLoading = false;
        });
      } else {
        throw Exception('Failed to load posts');
      }
    } catch (e) {
      setState(() {
        _isLoading = false;
      });
      print('Error: $e');
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('API Fetch Example')),
      body: _isLoading
          ? Center(child: CircularProgressIndicator())
          : ListView.builder(
        itemCount: _posts.length,
        itemBuilder: (context, index) {
          final post = _posts[index];
          return PostItem(
            title: post['title'],
            body: post['body'],
          );
        },
      ),
      );
    }
}
```

OUTPUT:



localhost:64297

API Fetch Example

blanditiis amet eius quis tempora quia autem rem
a provident perspiciatis quia

ullam ut quidem id aut vel consequuntur

debitis eius sed quibusdam non quis consectetur vitae
impedit ut qui consequatur sed aut in
quidem sit nostrum et maiores adipisci atque
quaerat voluptatem adipisci repudiandae

doloremque illum aliquid sunt

deserunt eos nobis asperiores et hic
est debitis repellat molestiae optio
nihil ratione ut eos beatae quibusdam distinctio maiores
earum voluptates et aut adipisci ea maiores voluptas maxime

qui explicabo molestiae dolorem

rerum ut et numquam laborum odit est sit

LAB 10: TESTING AND DEBUGGING FLUTTER APPLICATIONS

Aim: To ensure the reliability of UI components and debug applications effectively.

Objectives:

(a) Write unit tests for UI components to verify their functionality and robustness.

(b) Use Flutter's debugging tools to identify and resolve issues in the application.

10 A) WRITE UNIT TESTS FOR UI COMPONENTS.

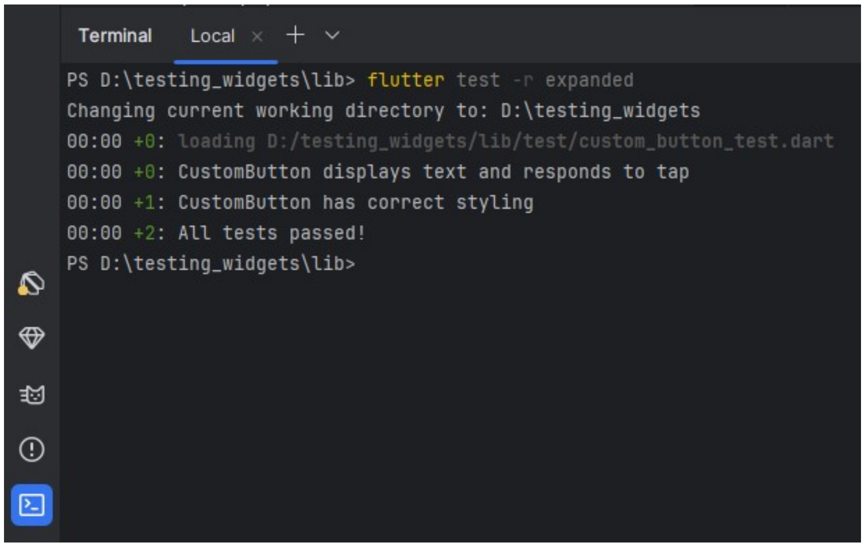
COUSTOM_BUTTON

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

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

  const CustomButton({
    Key? key,
    required this.text,
    required this.onPressed,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      onPressed: onPressed,
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.blue,
        foregroundColor: Colors.white,
        padding: EdgeInsets.symmetric(horizontal: 20, vertical: 12),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(30),
        ),
      ),
      child: Text(
        text,
        style: TextStyle(fontSize: 18),
      ),
    );
  }
}
```

COUNTER_BUTTON_TEST

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import '../widgets/CustomButton.dart';

void main() {
  testWidgets('CustomButton displays text and responds to tap', (WidgetTester
tester) async {
    bool buttonTapped = false;
    await tester.pumpWidget(
      MaterialApp(
        home: Scaffold(
          body: CustomButton(
            text: 'Test Button',
            onPressed: () {
              buttonTapped = true;
            },
          ),
        ),
      ),
    );

    final buttonFinder = find.text('Test Button');
    expect(buttonFinder, findsOneWidget);

    await tester.tap(buttonFinder);
    await tester.pump();

    expect(buttonTapped, isTrue);
  });

  testWidgets('CustomButton has correct styling', (WidgetTester tester) async
{
    await tester.pumpWidget(
      MaterialApp(
        home: Scaffold(
          body: CustomButton(
            text: 'Test Button',
            onPressed: () {},
          ),
        ),
      ),
    );

    final buttonFinder = find.byType(ElevatedButton);
    final button = tester.widget<ElevatedButton>(buttonFinder);

    expect(button.style?.backgroundColor?.resolve({}), equals(Colors.blue));
    expect(button.style?.foregroundColor?.resolve({}), equals(Colors.white));
    expect(button.style?.shape?.resolve({}) as RoundedRectangleBorder,
isNotNull);

    final textFinder = find.text('Test Button');
    final text = tester.widget<Text>(textFinder);
    expect(text.style?.fontSize, equals(18));
  });
}
```

OUTPUT:



```
PS D:\testing_widgets\lib> flutter test -r expanded
Changing current working directory to: D:\testing_widgets
00:00 +0: loading D:/testing_widgets/lib/test/custom_button_test.dart
00:00 +0: CustomButton displays text and responds to tap
00:00 +1: CustomButton has correct styling
00:00 +2: All tests passed!
PS D:\testing_widgets\lib>
```

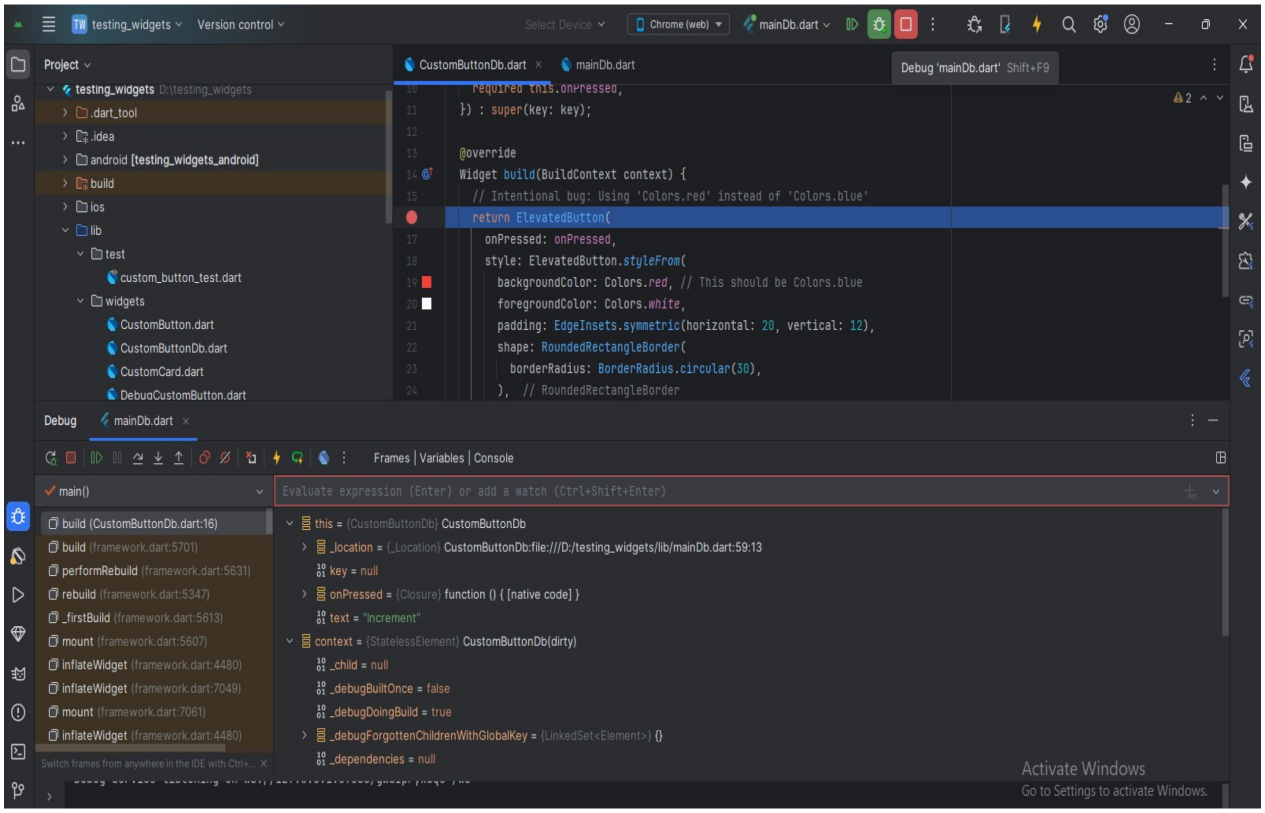
10 B) USE FLUTTER'S DEBUGGING TOOLS TO IDENTIFY AND FIX ISSUES.

```
import 'package:flutter/material.dart';
class CustomButton extends StatelessWidget {
  final String text;
  final VoidCallback onPressed;
  const CustomButton({
    Key? key,
    required this.text,
    required this.onPressed,
  }) : super(key: key);
  @override
  Widget build(BuildContext context) {
    // Intentional bug: Using 'Colors.red' instead of 'Colors.blue'
    return ElevatedButton(
      onPressed: onPressed,
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.red, // This should be Colors.blue
        foregroundColor: Colors.white,
        padding: EdgeInsets.symmetric(horizontal: 20, vertical: 12),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(30),
        ),
      ),
      child: Text(
        text,
        style: TextStyle(fontSize: 18),
      ),
    );
  }
}
```

MAIN_DB

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Custom Button Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(title: 'Custom Button Demo'),
    );
  }
}

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

  final String title;

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

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
      print('Counter incremented: $_counter'); // Debug print
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text(
              'You have pushed the button this many times:',
            ),
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.displayMedium,
            ),
            SizedBox(height: 20),
            CustomButtonDb(
              text: 'Increment',
              onPressed: _incrementCounter,
            ),
          ],
        ),
      ),
    );
  }
}
```

OUTPUT:



```
    required this.onPressed,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // Intentional bug: Using 'Colors.red' instead of 'Colors.blue'
    return ElevatedButton(
      onPressed: onPressed,
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.red, // This should be Colors.blue
        foregroundColor: Colors.white,
        padding: EdgeInsets.symmetric(horizontal: 20, vertical: 12),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(30),
        ), // RoundedRectangleBorder
```

```
main()
build (CustomButtonDb.dart:16)
build (framework.dart:5701)
performRebuild (framework.dart:5631)
rebuild (framework.dart:5347)
_firstBuild (framework.dart:5613)
mount (framework.dart:5607)
inflateWidget (framework.dart:4480)
inflateWidget (framework.dart:7049)
mount (framework.dart:7061)
inflateWidget (framework.dart:4480)
```

```
this = {CustomButtonDb} CustomButtonDb
  _location = {_Location} CustomButtonDb:file:///D:/testing_widgets/lib/mainDb.dart:59:13
  key = null
  onPressed = {Closure} function () { [native code] }
  text = "Increment"
context = {StatelessElement} CustomButtonDb(dirty)
  _child = null
  _debugBuiltOnce = false
  _debugDoingBuild = true
  _debugForgottenChildrenWithGlobalKey = {LinkedSet<Element>} {}
  _dependencies = null
```

testing_widgets
Version control
Select Device
Chrome (web)
mainDb.dart
Debug 'mainDb.dart' Shift+F9
Project
testing_widgets D:\testing_widgets
.dart_tool
.idea
android [testing_widgets_android]
build
ios
lib
test
custom_button_test.dart
widgets
CustomButton.dart
CustomButtonDb.dart
CustomCard.dart
DebugCustomButton.dart
CustomButtonDb.dart
mainDb.dart
Debug
mainDb.dart
Frames | Variables | Console
Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)
Switch frames from anywhere in the IDE with Ctrl+...
Activate Windows
Go to Settings to activate Windows.