[III-I UI Design-Flutter Lab Manual-R22](https://www.studocu.com/in/document/jawaharlal-nehru-technological-university-hyderabad/ui-flutter-design/iii-i-ui-design-flutter-lab-manual-r22/84281260?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=iii-i-ui-design-flutter-lab-manual-r22)

[UI Flutter Design (Jawaharlal Nehru Technological University, Hyderabad)](https://www.studocu.com/in/course/jawaharlal-nehru-technological-university-hyderabad/ui-flutter-design/6745237?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=iii-i-ui-design-flutter-lab-manual-r22)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

III B. TECH I SEM CSE (R22)

**UI DESIGN-FLUTTER LAB MANUAL**

**Academic Year 2024-2025**

1

**CS506PC: UI DESIGN-FLUTTER**

# B.Tech. III Year I Sem. L T P C

**0 0 2 1**

# Course Objectives:

* Learns to Implement Flutter Widgets and Layout
* Understands Responsive UI Design and with Navigation in Flutter
* Knowledge on Widges and customize widgets for specific UI elements, Themes
* Understand to include animation apart from fetching data

# Course Outcomes:

* Implements Flutter Widgets and Layouts
* Responsive UI Design and with Navigation in Flutter
* Create custom widgets for specific UI elements and also Apply styling using themes and custom styles.
* Design a form with various input fields, along with validation and error handling
* Fetches data and write code for unit Test for UI components and also animation

# List of Experiments: Students need to implement the following experiments

1. a) Install Flutter and Dart SDK.
   1. Write a simple Dart program to understand the language basics.
2. a) Explore various Flutter widgets (Text, Image, Container, etc.).
   1. Implement different layout structures using Row, Column, and Stack widgets.
3. a) Design a responsive UI that adapts to different screen sizes.
   1. Implement media queries and breakpoints for responsiveness.
4. a) Set up navigation between different screens using Navigator.
   1. Implement navigation with named routes.
5. a) Learn about stateful and stateless widgets.
   1. Implement state management using set State and Provider.
6. a) Create custom widgets for specific UI elements.
   1. Apply styling using themes and custom styles.
7. a) Design a form with various input fields.
   1. Implement form validation and error handling.
8. a) Add animations to UI elements using Flutter's animation framework.
   1. Experiment with different types of animations (fade, slide, etc.).
9. a) Fetch data from a REST API.
   1. Display the fetched data in a meaningful way in the UI.
10. a) Write unit tests for UI components.
    1. Use Flutter's debugging tools to identify and fix issues.

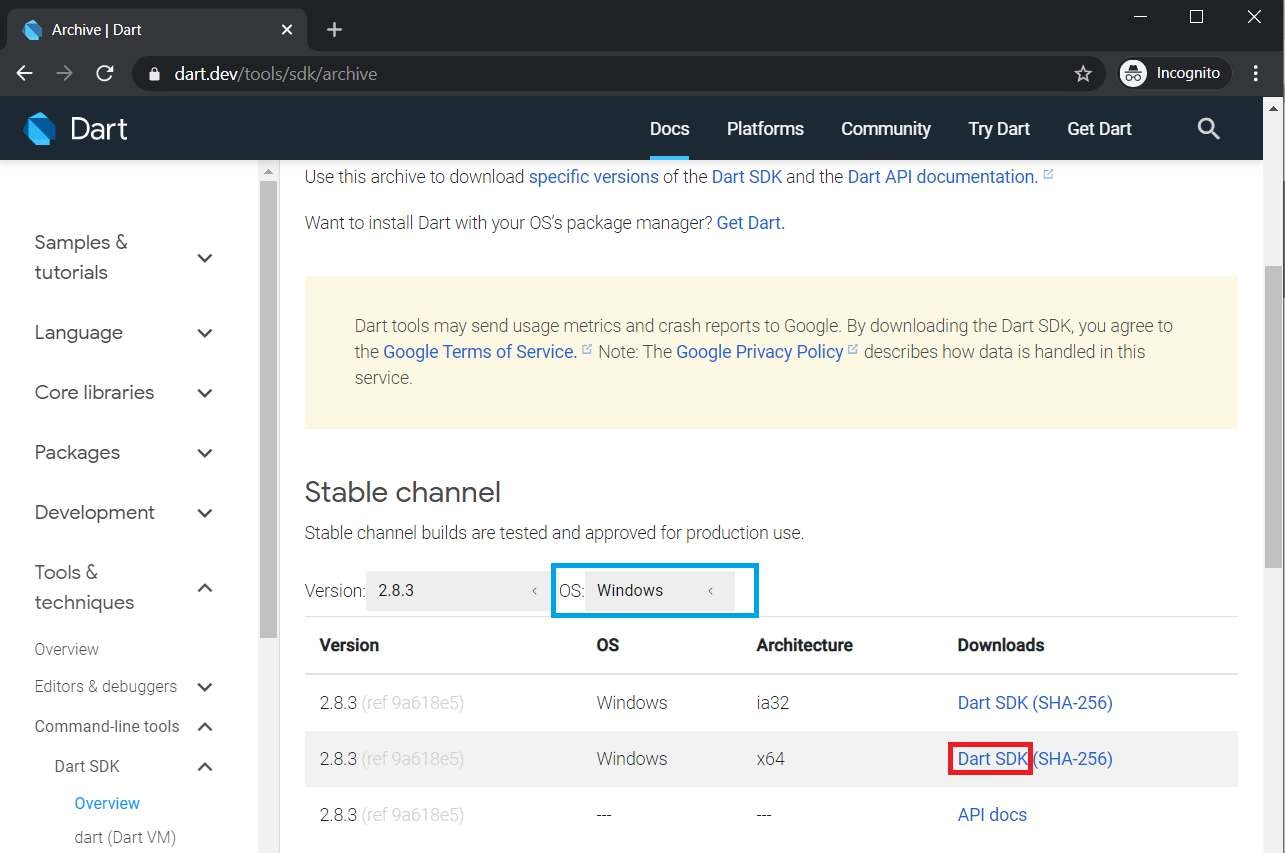
TEXT BOOK:

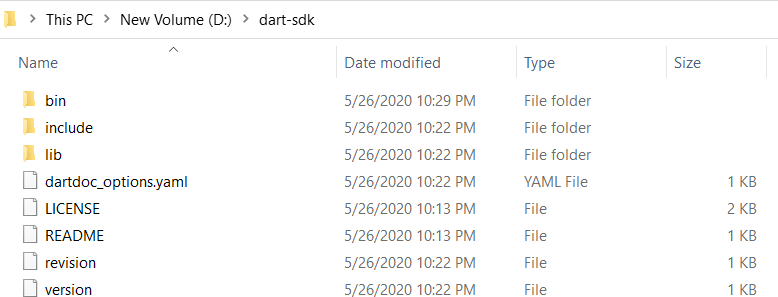
1. Marco L. Napoli, Beginning Flutter: A Hands-on Guide to App Development.

2

# 1. a) Install Flutter and Dart SDK. 2.

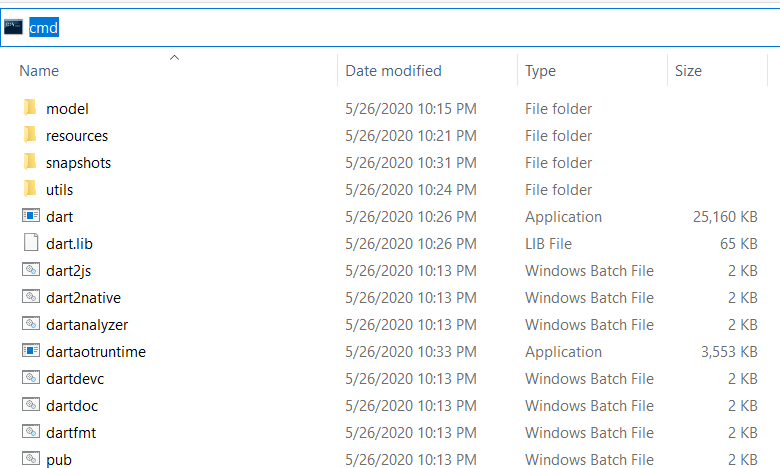
**Ans)** Dart SDK is a pre-compiled version so we have to download and extract it only. For this follow the below-given instructions: **Step 1:** Download Dart SDK. Download Dart SDK from the Dart SDK archive page. The URL is: <https://dart.dev/tools/sdk/archive>



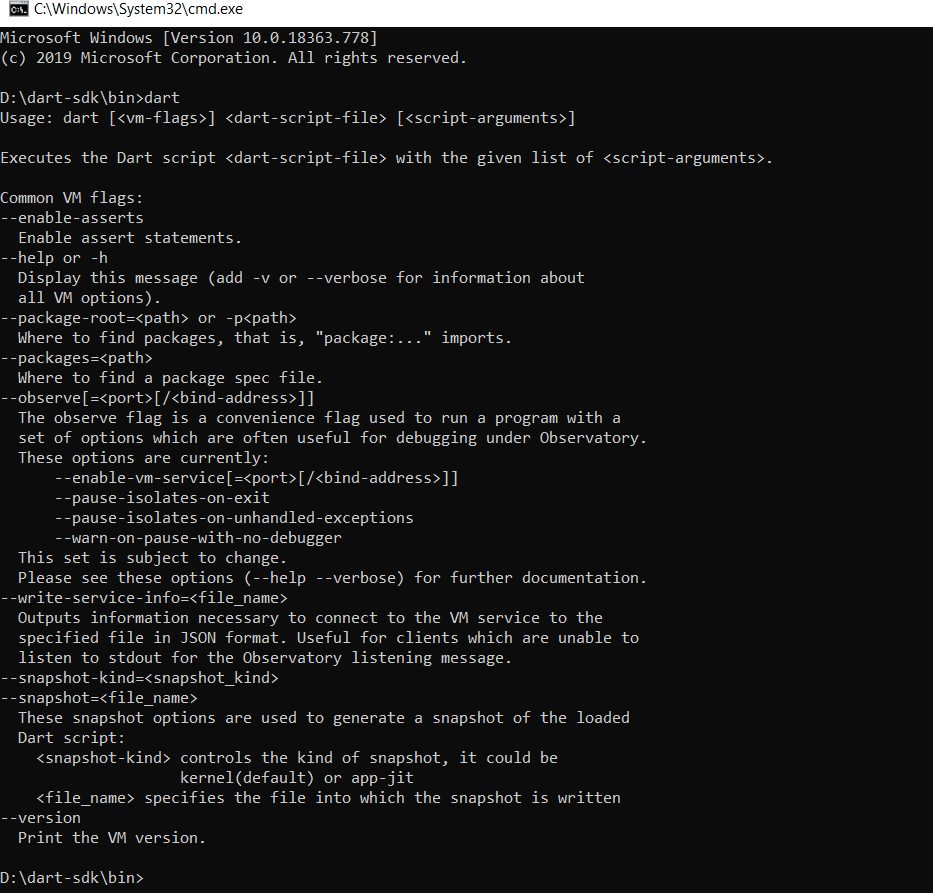
Click on DART SDK to download SDK for Windows 64-Bit Architecture. The download will start and a zip file will be downloaded. **Note:** To download SDK for any other OS select OS of your choice. **Step 2:** Extract the downloaded zip file. Extract the contents of downloaded zip file and after extracting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

3

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



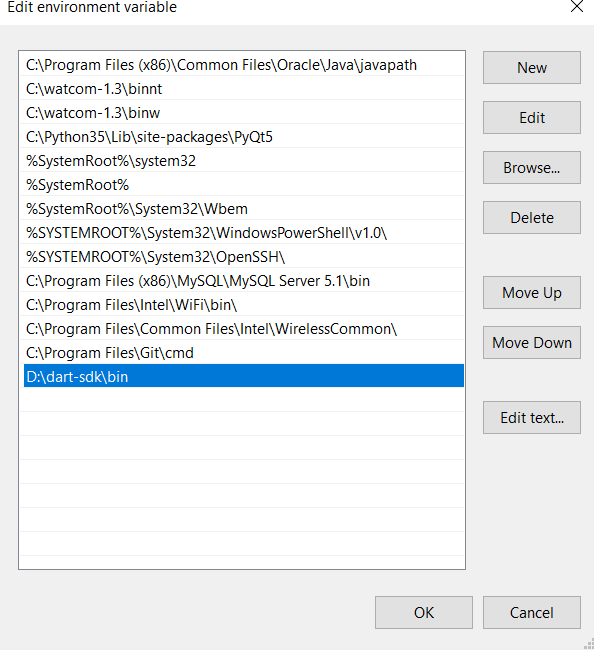
4

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

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

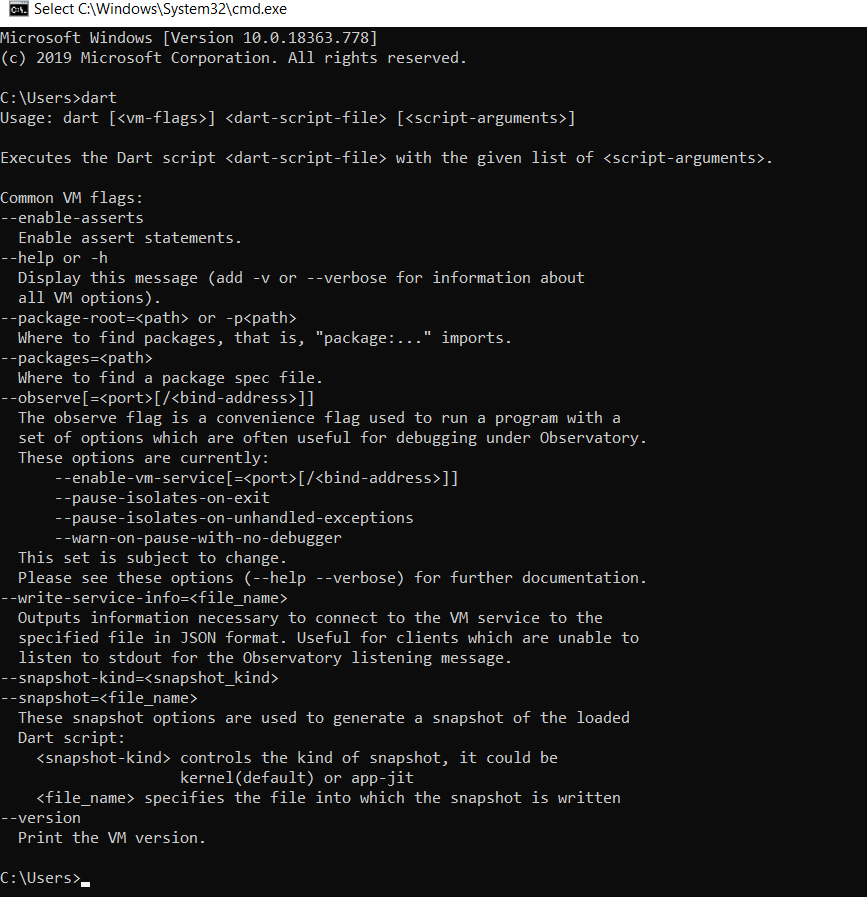
**Step 4:** Setting up path in environment variables. Open Environment Variables from advanced system settings and add Path in System Variables as depicted in image:

5



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

6

**Step 5:** Run Dart Using cmd

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

void main(){

var firstName = "John"; var lastName = "Doe";

print("Full name is $firstName $lastName");

}

# Output: Full name is John Doe

void main() {

int num1 = 10; //declaring number1 int num2 = 3; //declaring number2

// Calculation

int sum = num1 + num2;

7

int diff = num1 - num2; int mul = num1 \* num2;

double div = num1 / num2; // It is double because it outputs number with decimal.

// displaying the output print("The sum is $sum"); print("The diff is $diff"); print("The mul is $mul"); print("The div is $div");

}

# Output:

**The sum is 13 The diff is 7 The mul is 30**

# The div is 3.3333333333333335

import 'dart:io'; void main() {

print("Enter number:");

int? number = int.parse(stdin.readLineSync()!); print("The entered number is ${number}");

}

# Output:

**Enter number:**

# 50

**The entered number is 50**

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

**Text Widget:**

import 'package:flutter/material.dart';

// function to trigger build process

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

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

@override

Widget build(BuildContext context) { return MaterialApp(

8

home: Scaffold(

backgroundColor: Colors.lightGreen, appBar: AppBar(

backgroundColor: Colors.green, title: const Text("welcome Screen"),

), // AppBar body: Container(

child: const Center(

child: Text("Hello world!!"),

), // Center

), // Container

), // Scaffold

); // MaterialApp

}

}

# Output: Hello World!! Image Widget:

import 'package:flutter/material.dart';

// function to start app building

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

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

// This widget is the root

// of your application

@override

Widget build(BuildContext context) { return MaterialApp(

home: Scaffold(

appBar: AppBar( title: const Text(

'Insert Image Demo',

),

),

body: Center( child: Column(

children: <Widget>[ Image.asset('assets/images/output.gif',

height: 200,

scale: 2.5,

9

// color: Color.fromARGB(255, 15, 147, 59), opacity:

const AlwaysStoppedAnimation<double>(0.5)), //Image.asset

Image.asset(

'assets/images/geeksforgeeks.jpg', height: 400,

width: 400,

), // Image.asset

], //<Widget>[]

), //Column

), //Center

),

);

}

}

# Containter Widget:

import 'package:flutter/material.dart'; void main() => runApp(const MyApp()); class MyApp extends StatelessWidget {

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

@override

Widget build(BuildContext context) { return MaterialApp(

home: Scaffold(

appBar: AppBar(

title: const Text("Container example"),

),

body: Container( height: 200,

width: double.infinity,

//color: Colors.purple, alignment: Alignment.center, margin: const EdgeInsets.all(20),

padding: const EdgeInsets.all(30), decoration: BoxDecoration(

border: Border.all(color: Colors.black, width: 3),

),

child: const Text("Hello! i am inside a container!", style: TextStyle(fontSize: 20)),

),

10

),

);

}

}

Output:



# 2b) Implement different layout structures using Row, Column, and Stack widgets Row Widget

import 'package:flutter/material.dart'; void main() { runApp(MyApp()); } class MyApp extends StatelessWidget { @override

Widget build(BuildContext context) { return MaterialApp(

home: MyHomePage()

);

}

}

class MyHomePage extends StatefulWidget { @override

\_MyHomePageState createState() => \_MyHomePageState();

11

}

class \_MyHomePageState extends State<MyHomePage> { @override

Widget build(BuildContext context) { return Scaffold(

appBar: AppBar(

title: Text("Flutter Row Example"),

),

body: Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly, children:<Widget>[

Container(

margin: EdgeInsets.all(12.0), padding: EdgeInsets.all(8.0), decoration:BoxDecoration(

borderRadius:BorderRadius.circular(8), color:Colors.green

),

child:

Text("React.js",style: TextStyle(color:Colors.yellowAccent,fontSize:25),),

),

Container(

margin: EdgeInsets.all(15.0),

12

padding: EdgeInsets.all(8.0), decoration:BoxDecoration(

borderRadius:BorderRadius.circular(8), color:Colors.green

),

TextStyle(color:Colors.yellowAccent,fontSize:25),),

),

Container(

margin: EdgeInsets.all(12.0), padding: EdgeInsets.all(8.0), decoration:BoxDecoration(

borderRadius:BorderRadius.circular(8), color:Colors.green

),

child: Text("Flutter",style:

TextStyle(color:Colors.yellowAccent,fontSize:25),),

)

]

),

);

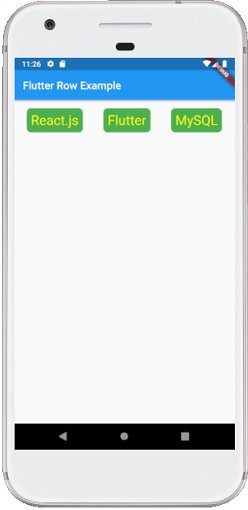
}

}

13

child: Text("MySQL",style:

Output:



**Column Widget:**

import 'package:flutter/material.dart';

void main() { runApp(MyApp()); } class MyApp extends StatelessWidget { @override

Widget build(BuildContext context) { return MaterialApp(

home: MyHomePage()

);

}

}

class MyHomePage extends StatefulWidget { @override

\_MyHomePageState createState() => \_MyHomePageState();

}

class \_MyHomePageState extends State<MyHomePage> { @override

Widget build(BuildContext context) {

14

return Scaffold( appBar: AppBar(

title: Text("Flutter Column Example"),

),

body: Column(

mainAxisAlignment: MainAxisAlignment.spaceBetween, children:<Widget>[

Container(

margin: EdgeInsets.all(20.0), padding: EdgeInsets.all(12.0), decoration:BoxDecoration(

borderRadius:BorderRadius.circular(8), color:Colors.red

),

TextStyle(color:Colors.yellowAccent,fontSize:20),),

),

Container(

margin: EdgeInsets.all(20.0), padding: EdgeInsets.all(12.0), decoration:BoxDecoration(

borderRadius:BorderRadius.circular(8), color:Colors.red

child: Text("React.js",style:

),

TextStyle(color:Colors.yellowAccent,fontSize:20),),

),

Container(

margin: EdgeInsets.all(20.0), padding: EdgeInsets.all(12.0), decoration:BoxDecoration(

borderRadius:BorderRadius.circular(8), color:Colors.red

child: Text("Flutter",style:

),

TextStyle(color:Colors.yellowAccent,fontSize:20),),

)

child: Text("MySQL",style:

]

),

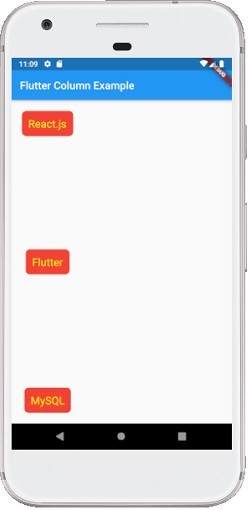
);

}

}

Output:

15



**Stack Widget:**

import 'package:flutter/material.dart'; void main() {

runApp(MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text('GeeksforGeeks'), backgroundColor: Colors.greenAccent[400],

), //AppBar body: Center(

child: SizedBox( width: 300,

height: 300, child: Center(

child: Stack( children: <Widget>[

Container( width: 300,

height: 300, color: Colors.red,

), //Container Container( width: 250,

height: 250,

color: Colors.black,

), //Container

16

Container( height: 200,

width: 200,

color: Colors.purple,

), //Container

], //<Widget>[]

), //Stack

), //Center

), //SizedBox

) //Center

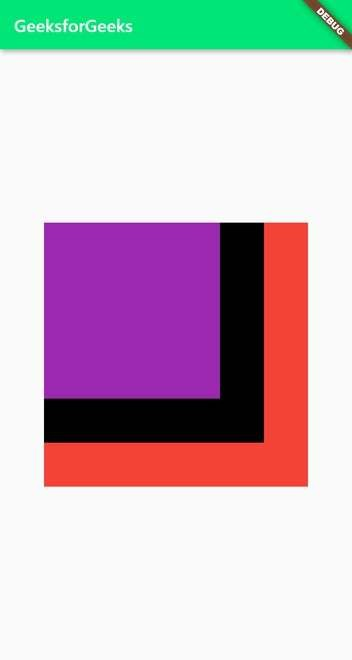
) //Scaffold

) //MaterialApp

);

}

**Output:**



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

**Ans)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<link rel="stylesheet"

href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<title>Responsive UI Example</title>

17

</head>

<body>

<div class="container">

<header class="jumbotron text-center">

<h1>Responsive UI Example</h1>

</header>

<main>

<section class="mb-4">

<h2>Section 1</h2>

<p>This is some content for section 1.</p>

</section>

<section class="mb-4">

<h2>Section 2</h2>

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

</section>

</main>

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

</footer>

</div>

<!-- Bootstrap JS and dependencies (jQuery) -->

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>

<script src=["ht](https://cdn.jsdelivr.net/npm/%40popperjs/core%402.10.2/dist/umd/popper.min.js)t[ps://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js](https://cdn.jsdelivr.net/npm/%40popperjs/core%402.10.2/dist/umd/popper.min.js)"></ script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></ script>

</body>

</html>

**Output:**

18



# 3 b) Implement media queries and breakpoints for responsiveness.

Ans) <!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<link rel="stylesheet"

href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<style>

body {

font-family: Arial, sans-serif; margin: 0;

padding: 0;

background-color: #f4f4f4;

}

header {

background-color: #333; color: #fff;

text-align: center; padding: 1em;

}

main {

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

}

section {

margin-bottom: 20px;

}

footer {

19

background-color: #333; color: #fff;

text-align: center; padding: 1em; position: fixed; bottom: 0;

width: 100%;

}

@media only screen and (max-width: 768px) { main {

padding: 10px;

}

footer {

position: static;

}

}

</style>

<title>Responsive UI Example</title>

</head>

<body>

<div class="container">

<header class="jumbotron text-center">

<h1>Responsive UI Example</h1>

</header>

<main>

<section class="mb-4">

<h2>Section 1</h2>

<p>This is some content for section 1.</p>

</section>

<section class="mb-4">

<h2>Section 2</h2>

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

</section>

</main>

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

</footer>

</div>

<!-- Bootstrap JS and dependencies (jQuery) -->

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>

20

<script src=["ht](https://cdn.jsdelivr.net/npm/%40popperjs/core%402.10.2/dist/umd/popper.min.js)t[ps://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js](https://cdn.jsdelivr.net/npm/%40popperjs/core%402.10.2/dist/umd/popper.min.js)"></ script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></ script>

</body>

</html>

Output:



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

Ans)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Screen Navigation Example</title>

<style>

body {

font-family: Arial, sans-serif; margin: 0;

padding: 0;

background-color: #f4f4f4;

}

header {

background-color: #333; color: #fff;

text-align: center; padding: 1em;

}

21

main {

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

}

section { display: none;

}

footer {

background-color: #333; color: #fff;

text-align: center; padding: 1em; position: fixed; bottom: 0;

width: 100%;

}

.active {

display: block;

}

</style>

</head>

<body>

<header class="jumbotron text-center">

<h1>Screen Navigation Example</h1>

</header>

<main>

<section id="home" class="active">

<h2>Home Screen</h2>

<p>Welcome to the Home Screen.</p>

<button onclick="navigateTo('about')">Go to About</button>

</section>

<section id="about">

<h2>About Screen</h2>

<p>This is the About Screen.</p>

<button onclick="navigateTo('home')">Go to Home</button>

</section>

</main>

<footer class="bg-dark text-light text-center py-3 mt-5">

22

&copy; 2024 Your Company Name

</footer>

<script>

function navigateTo(screenId) {

// Hide all sections document.querySelectorAll('section').forEach(section => {

section.classList.remove('active');

});

// Show the selected section document.getElementById(screenId).classList.add('active');

}

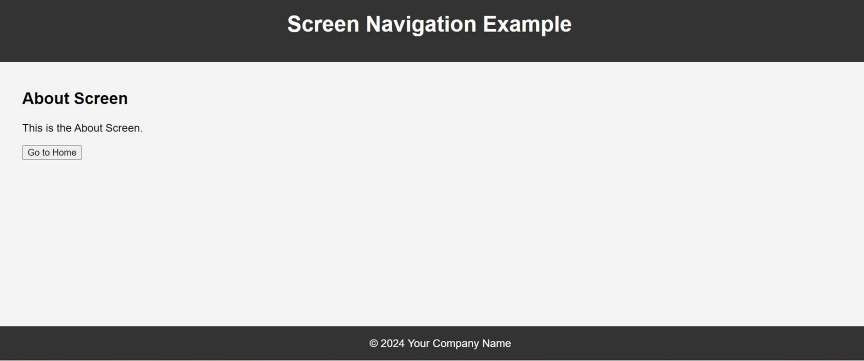
</script>

</body>

</html>

# Output:





23

# 4b) Implement navigation with named routes.

**Ans)**

import 'package:flutter/material.dart'; void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget { @override

Widget build(BuildContext context) { return MaterialApp(

title: 'Named Routes Navigation Example', initialRoute: '/',

routes: {

'/': (context) => HomeScreen(), '/about': (context) => AboutScreen(),

},

);

}

}

class HomeScreen extends StatelessWidget { @override

Widget build(BuildContext context) { return Scaffold(

appBar: AppBar(

title: Text('Home Screen'),

),

body: Center( child: Column(

mainAxisAlignment: MainAxisAlignment.center, children: <Widget>[

Text(

'Welcome to the Home Screen.',

),

SizedBox(height: 20), ElevatedButton( onPressed: () {

Navigator.pushNamed(context, '/about');

},

child: Text('Go to About'),

),

24

],

),

),

);

}

}

class AboutScreen extends StatelessWidget { @override

Widget build(BuildContext context) { return Scaffold(

appBar: AppBar(

title: Text('About Screen'),

),

body: Center( child: Column(

mainAxisAlignment: MainAxisAlignment.center, children: <Widget>[

Text(

'This is the About Screen.',

),

SizedBox(height: 20), ElevatedButton( onPressed: () {

Navigator.pop(context);

},

child: Text('Go back to Home'),

),

],

),

),

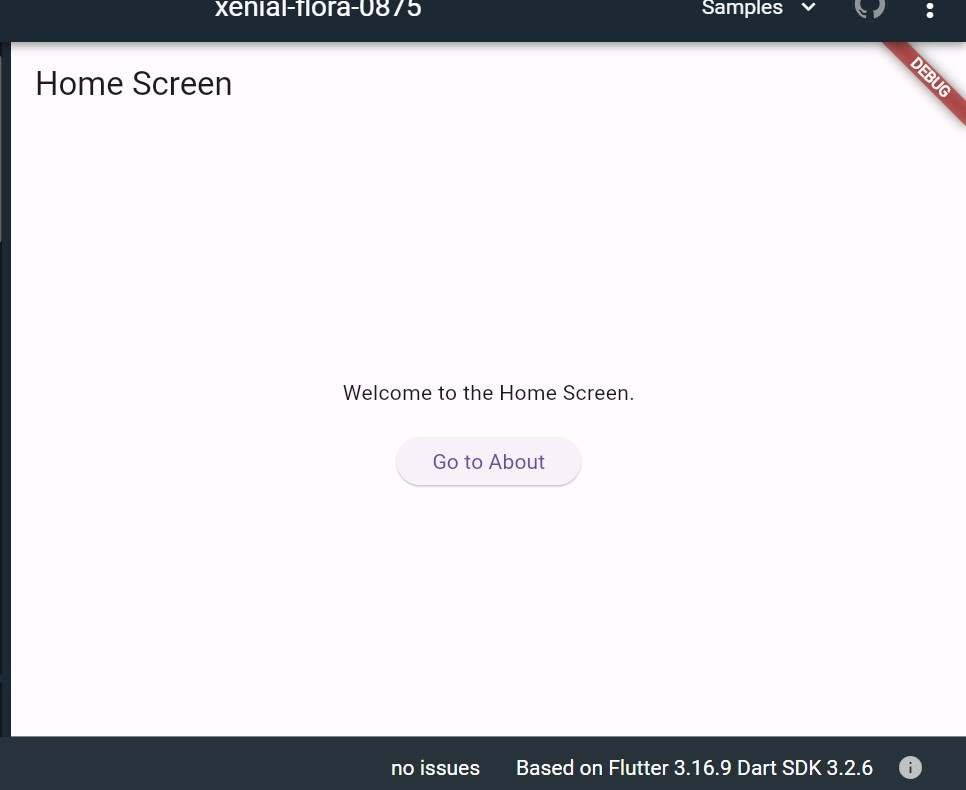
);

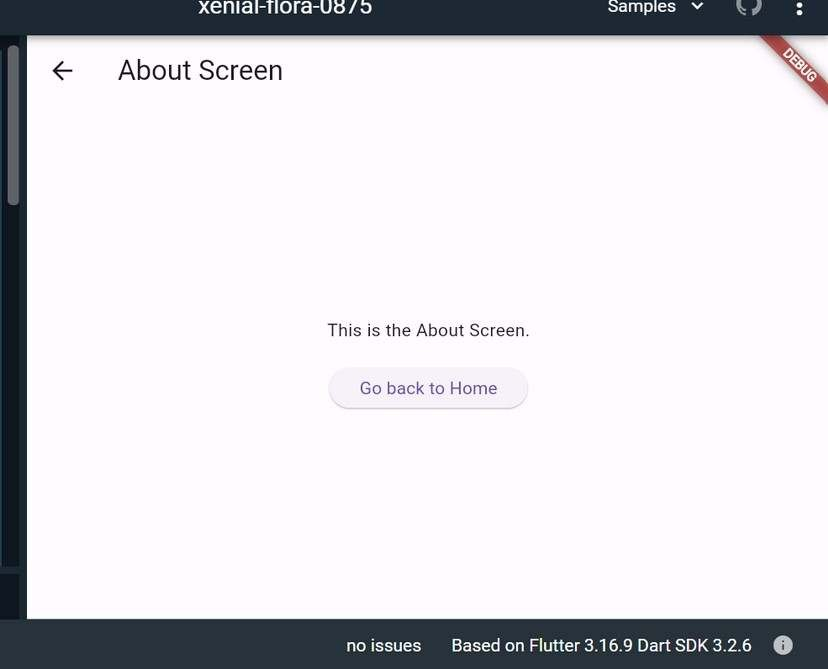
}

}

# Output:

25





# a) Learn about stateful and stateless widgets. Ans)

import 'package:flutter/material.dart'; void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget { @override

Widget build(BuildContext context) { return MaterialApp(

title: 'Stateful and Stateless Example', theme: ThemeData(

26

primarySwatch: Colors.blue,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatefulWidget { @override

\_MyHomePageState createState() => \_MyHomePageState();

}

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

void incrementCounter() { setState(() {

counter++;

});

}

@override

Widget build(BuildContext context) { return Scaffold(

appBar: AppBar(

title: Text('Stateful and Stateless Example'),

),

body: Column(

mainAxisAlignment: MainAxisAlignment.center, children: <Widget>[

CounterDisplay(counter), SizedBox(height: 20), CounterButton(incrementCounter),

],

),

);

}

}

class CounterDisplay extends StatelessWidget { final int count;

CounterDisplay(this.count); @override

Widget build(BuildContext context) {

27

return Text(

'Counter Value: $count', style: TextStyle(fontSize: 20),

);

}

}

class CounterButton extends StatelessWidget { final VoidCallback onPressed;

CounterButton(this.onPressed); @override

Widget build(BuildContext context) { return ElevatedButton(

onPressed: onPressed,

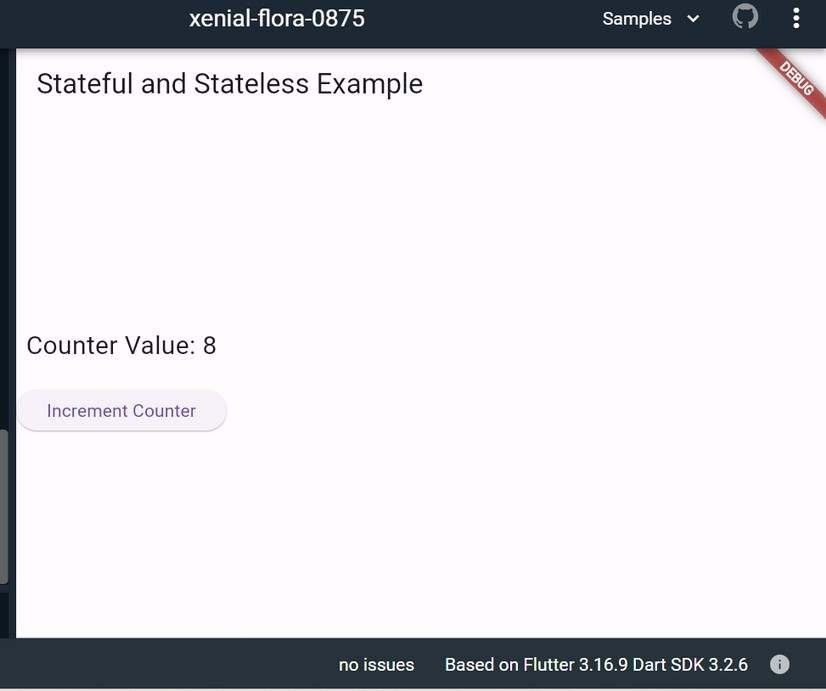
child: Text('Increment Counter'),

);

}

}

Output:



# 5 b) Implement state management using set State and Provider.

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

void main() { runApp(

ChangeNotifierProvider(

create: (context) => CounterModel(), child: MyApp(),

28

),

);

}

class MyApp extends StatelessWidget { @override

Widget build(BuildContext context) { return MaterialApp(

title: 'State Management Example', theme: ThemeData( primarySwatch: Colors.blue,

),

home: CounterPage(),

);

}

}

class CounterModel extends ChangeNotifier { int \_counter = 0;

int get counter => \_counter; void incrementCounter() {

\_counter++; notifyListeners();

}

}

class CounterPage extends StatelessWidget { @override

Widget build(BuildContext context) {

final counterModel = Provider.of<CounterModel>(context);

return Scaffold( appBar: AppBar(

title: Text('State Management Example'),

),

body: Center( child: Column(

mainAxisAlignment: MainAxisAlignment.center, children: <Widget>[

Text(

'Counter Value: ${counterModel.counter}', style: TextStyle(fontSize: 20),

),

SizedBox(height: 20),

29

ElevatedButton(

onPressed: counterModel.incrementCounter, child: Text('Increment Counter'),

),

],

),

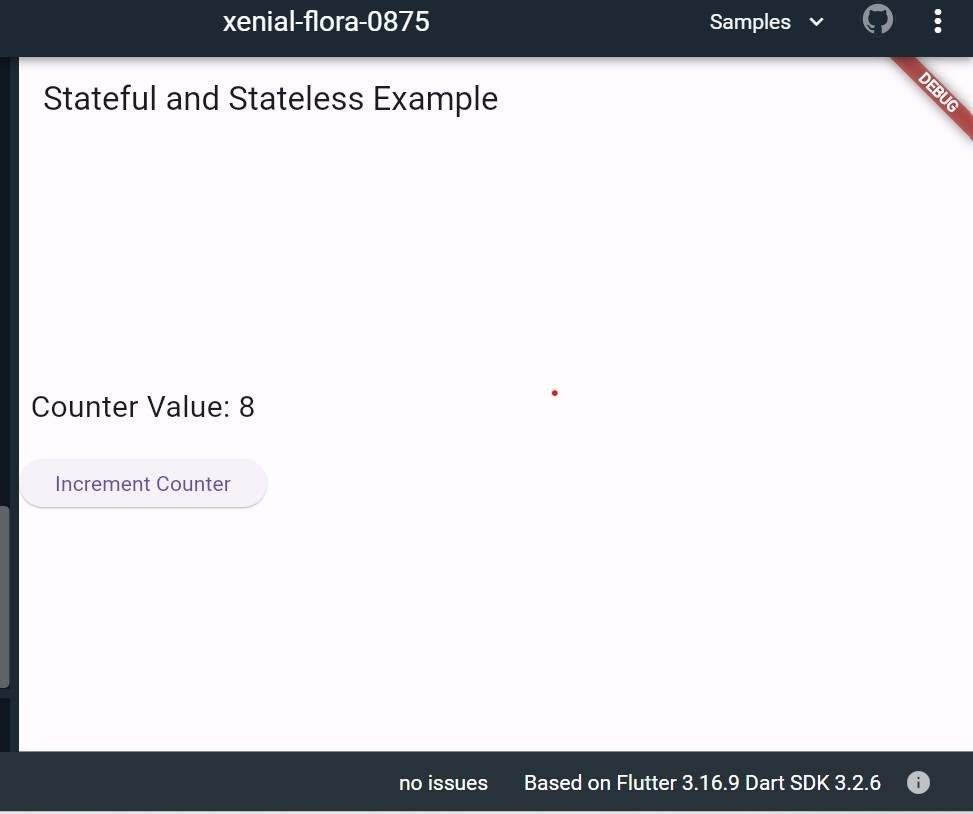
),

);

}

}

# Output:



1. **a) Create custom widgets for specific UI elements.**

# Ans)

import 'package:flutter/material.dart';

class CustomButton extends StatelessWidget { final String text;

final Function onPressed; final Color buttonColor; final Color textColor; CustomButton({

required this.text, required this.onPressed,

this.buttonColor = Colors.blue, this.textColor = Colors.white,

30

});

@override

Widget build(BuildContext context) { return ElevatedButton(

onPressed: () => onPressed(), style: ButtonStyle(

backgroundColor: MaterialStateProperty.all<Color>(buttonColor), foregroundColor: MaterialStateProperty.all<Color>(textColor),

),

child: Text(text),

);

}

}

class CustomAlertDialog extends StatelessWidget { final String title;

final String message;

final String positiveButtonText; final String negativeButtonText; final Function onPositivePressed; final Function onNegativePressed; CustomAlertDialog({

required this.title, required this.message,

required this.positiveButtonText, required this.negativeButtonText, required this.onPositivePressed, required this.onNegativePressed,

});

@override

Widget build(BuildContext context) { return AlertDialog(

title: Text(title), content: Text(message), actions: <Widget>[ CustomButton(

text: negativeButtonText,

onPressed: () => onNegativePressed(),

),

CustomButton(

text: positiveButtonText,

onPressed: () => onPositivePressed(),

),

],

);

31

}

}

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

}

class MyApp extends StatelessWidget { @override

Widget build(BuildContext context) { return MaterialApp(

home: Scaffold( appBar: AppBar(

title: Text('Custom Button Example'),

),

body: Center(

child: CustomButton( text: 'Click Me', onPressed: () {

// Handle button press print('Button Pressed');

},

),

),

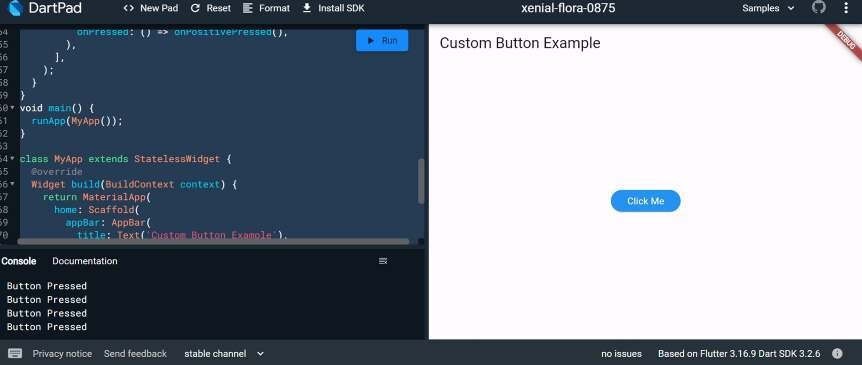
),

);

}

}

# Output:



**6b) Apply styling using themes and custom styles. Ans)**

32

import 'package:flutter/material.dart';

import 'package:google\_fonts/google\_fonts.dart';

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

}

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

@override

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

return MaterialApp( title: appName, theme: ThemeData( useMaterial3: true,

// Define the default brightness and colors. colorScheme: ColorScheme.fromSeed( seedColor: Colors.purple,

// TRY THIS: Change to "Brightness.light"

// and see that all colors change

// to better contrast a light background. brightness: Brightness.dark,

),

// Define the default `TextTheme`. Use this to specify the default

// text styling for headlines, titles, bodies of text, and more. textTheme: TextTheme(

displayLarge: const TextStyle( fontSize: 72,

fontWeight: FontWeight.bold,

),

// TRY THIS: Change one of the GoogleFonts

// to "lato", "poppins", or "lora".

// The title uses "titleLarge"

// and the middle text uses "bodyMedium". titleLarge: GoogleFonts.oswald(

fontSize: 30,

fontStyle: FontStyle.italic,

),

bodyMedium: GoogleFonts.merriweather(), displaySmall: GoogleFonts.pacifico(),

33

),

),

home: const MyHomePage( title: appName,

),

);

}

}

class MyHomePage extends StatelessWidget { final String title;

const MyHomePage({super.key, required this.title}); @override

Widget build(BuildContext context) { return Scaffold(

appBar: AppBar( title: Text(title,

style: Theme.of(context).textTheme.titleLarge!.copyWith( color: Theme.of(context).colorScheme.onSecondary,

)),

backgroundColor: Theme.of(context).colorScheme.secondary,

),

body: Center( child: Container(

padding: const EdgeInsets.symmetric( horizontal: 12,

vertical: 12,

),

color: Theme.of(context).colorScheme.primary, child: Text(

'Text with a background color',

// TRY THIS: Change the Text value

// or change the Theme.of(context).textTheme

// to "displayLarge" or "displaySmall".

style: Theme.of(context).textTheme.bodyMedium!.copyWith( color: Theme.of(context).colorScheme.onPrimary,

),

),

),

),

floatingActionButton: Theme(

data: Theme.of(context).copyWith(

// TRY THIS: Change the seedColor to "Colors.red" or

// "Colors.blue".

34

colorScheme: ColorScheme.fromSeed( seedColor: Colors.pink,

brightness: Brightness.dark,

),

),

child: FloatingActionButton( onPressed: () {},

child: const Icon(Icons.add),

),

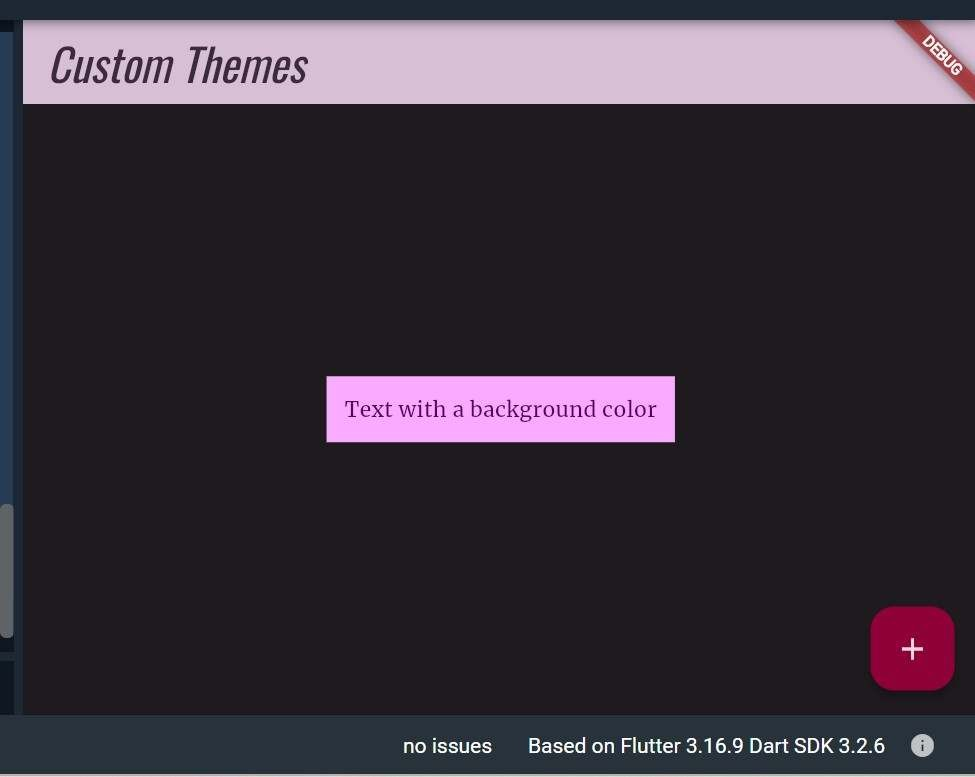
),

);

}

}

Output:



# a) Design a form with various input fields.

import 'package:flutter/material.dart'; void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget { @override

Widget build(BuildContext context) { return MaterialApp(

35

title: 'Form Example', theme: ThemeData(

primarySwatch: Colors.blue,

),

home: MyForm(),

);

}

}

class MyForm extends StatefulWidget { @override

\_MyFormState createState() => \_MyFormState();

}

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

TextEditingController \_nameController = TextEditingController(); TextEditingController \_emailController = TextEditingController(); TextEditingController \_passwordController = TextEditingController();

@override

Widget build(BuildContext context) { return Scaffold(

appBar: AppBar(

title: Text('Form Example'),

),

body: Padding(

padding: EdgeInsets.all(16.0), child: Form(

key: \_formKey, child: Column(

crossAxisAlignment: CrossAxisAlignment.stretch, children: <Widget>[

TextFormField(

controller: \_nameController, decoration: InputDecoration( labelText: 'Name',

border: OutlineInputBorder(),

),

validator: (value) {

if (value == null || value.isEmpty) { return 'Please enter your name';

}

return null;

},

),

SizedBox(height: 16), TextFormField(

controller: \_emailController,

keyboardType: TextInputType.emailAddress, decoration: InputDecoration(

labelText: 'Email',

36

border: OutlineInputBorder(),

),

validator: (value) {

if (value == null || value.isEmpty) { return 'Please enter your email';

} else if (!RegExp(r'^[\w-]+(\.[\w-]+)\*@[\w-]+(\.[\w-]+)+$')

.hasMatch(value)) {

return 'Please enter a valid email address';

}

return null;

},

),

SizedBox(height: 16), TextFormField(

controller: \_passwordController, obscureText: true,

decoration: InputDecoration( labelText: 'Password',

border: OutlineInputBorder(),

),

validator: (value) {

if (value == null || value.isEmpty) { return 'Please enter your password';

} else if (value.length < 6) {

return 'Password must be at least 6 characters long';

}

return null;

},

),

SizedBox(height: 16), ElevatedButton( onPressed: () {

if (\_formKey.currentState!.validate()) {

// Form is valid, process the data print('Name: ${\_nameController.text}'); print('Email: ${\_emailController.text}');

print('Password: ${\_passwordController.text}');

}

},

child: Text('Submit'),

),

],

),

),

),

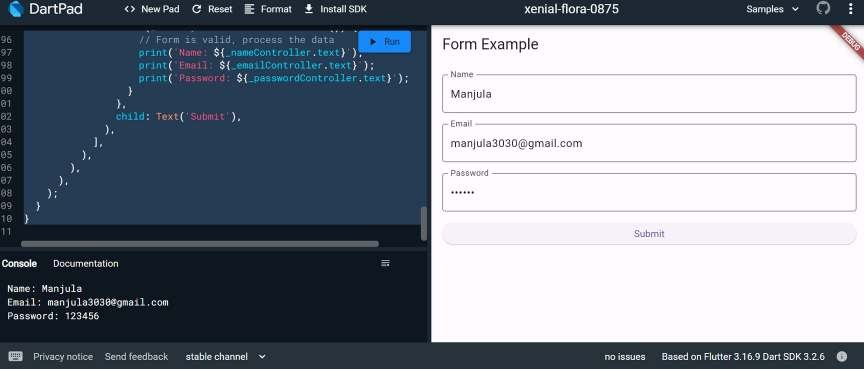
);

}

}

Output:

37



1. **b) Implement form validation and error handling. Ans)**

import 'package:flutter/material.dart'; void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget { @override

Widget build(BuildContext context) { return MaterialApp(

title: 'Form Validation Example', theme: ThemeData( primarySwatch: Colors.blue,

),

home: MyForm(),

);

}

}

class MyForm extends StatefulWidget { @override

\_MyFormState createState() => \_MyFormState();

}

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

TextEditingController \_nameController = TextEditingController(); TextEditingController \_emailController = TextEditingController(); TextEditingController \_passwordController = TextEditingController();

38

@override

Widget build(BuildContext context) { return Scaffold(

appBar: AppBar(

title: Text('Form Validation Example'),

),

body: Padding(

padding: EdgeInsets.all(16.0), child: Form(

key: \_formKey, child: Column(

crossAxisAlignment: CrossAxisAlignment.stretch, children: <Widget>[

TextFormField(

controller: \_nameController, decoration: InputDecoration( labelText: 'Name',

border: OutlineInputBorder(),

),

validator: (value) {

if (value == null || value.isEmpty) { return 'Please enter your name';

}

return null;

},

),

SizedBox(height: 16), TextFormField(

controller: \_emailController,

keyboardType: TextInputType.emailAddress, decoration: InputDecoration(

labelText: 'Email',

border: OutlineInputBorder(),

),

validator: (value) {

if (value == null || value.isEmpty) { return 'Please enter your email';

} else if (!RegExp(r'^[\w-]+(\.[\w-]+)\*@[\w-]+(\.[\w-]+)+$')

.hasMatch(value)) {

return 'Please enter a valid email address';

}

return null;

},

),

SizedBox(height: 16),

39

TextFormField(

controller: \_passwordController, obscureText: true,

decoration: InputDecoration( labelText: 'Password',

border: OutlineInputBorder(),

),

validator: (value) {

if (value == null || value.isEmpty) { return 'Please enter your password';

} else if (value.length < 6) {

return 'Password must be at least 6 characters long';

}

return null;

},

),

SizedBox(height: 16), ElevatedButton( onPressed: () {

if (\_formKey.currentState!.validate()) {

// Form is valid, process the data print('Name: ${\_nameController.text}'); print('Email: ${\_emailController.text}');

print('Password: ${\_passwordController.text}');

}

},

child: Text('Submit'),

),

],

),

),

),

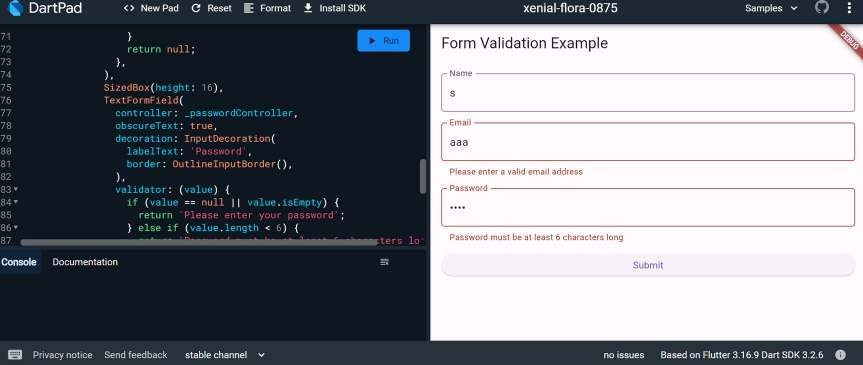
);

}

}

**Output:**

40



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

**Ans)**

import 'package:flutter/material.dart';

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

}

class MyApp extends StatelessWidget { @override

Widget build(BuildContext context) { return MaterialApp(

title: 'Animation Example', theme: ThemeData( primarySwatch: Colors.blue,

),

home: MyAnimatedWidget(),

);

}

}

class MyAnimatedWidget extends StatefulWidget { @override

\_MyAnimatedWidgetState createState() => \_MyAnimatedWidgetState();

}

class \_MyAnimatedWidgetState extends State<MyAnimatedWidget>

41

with SingleTickerProviderStateMixin {

late AnimationController \_animationController; late Animation<double> \_opacityAnimation;

@override

void initState() { super.initState();

// Create an AnimationController with a duration of 1 second

\_animationController = AnimationController( vsync: this,

duration: Duration(seconds: 1),

);

// Create a Tween to animate opacity from 0.0 to 1.0

\_opacityAnimation = Tween<double>(begin: 0.0, end: 1.0).animate( CurvedAnimation(

parent: \_animationController, curve: Curves.easeInOut,

),

);

// Start the animation

\_animationController.forward();

}

@override

Widget build(BuildContext context) { return Scaffold(

appBar: AppBar(

title: Text('Animation Example'),

),

body: Center(

child: FadeTransition( opacity: \_opacityAnimation, child: Container(

width: 200,

height: 200,

color: Colors.blue, child: Center( child: Text(

'Animated Widget', style: TextStyle( color: Colors.white, fontSize: 20,

),

42

),

),

),

),

),

);

}

@override

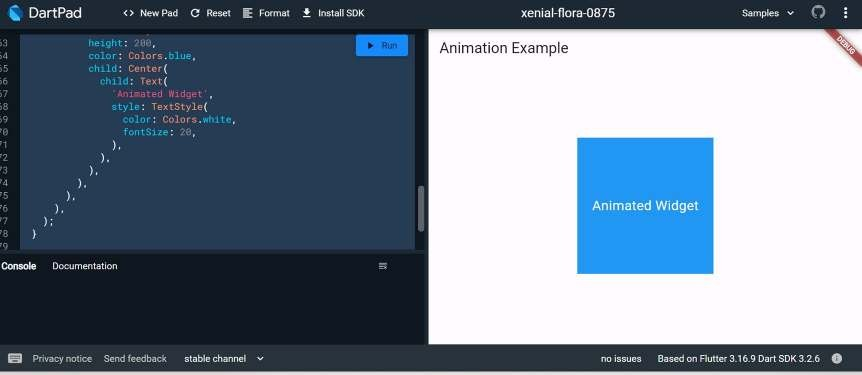
void dispose() {

\_animationController.dispose(); super.dispose();

}

}

Output:



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

import 'package:flutter/material.dart'; void main() {

43

runApp(MyApp());

}

class MyApp extends StatelessWidget { @override

Widget build(BuildContext context) { return MaterialApp(

title: 'Fade Animation Example', theme: ThemeData( primarySwatch: Colors.blue,

),

home: FadeAnimationWidget(),

);

}

}

class FadeAnimationWidget extends StatefulWidget { @override

\_FadeAnimationWidgetState createState() => \_FadeAnimationWidgetState();

}

class \_FadeAnimationWidgetState extends State<FadeAnimationWidget> with SingleTickerProviderStateMixin {

late AnimationController \_animationController; late Animation<double> \_opacityAnimation;

@override

void initState() { super.initState();

\_animationController = AnimationController( vsync: this,

duration: Duration(seconds: 10),

);

\_opacityAnimation = Tween<double>(begin: 0.0, end: 1.0).animate( CurvedAnimation(

parent: \_animationController, curve: Curves.easeInOut,

),

);

\_animationController.forward();

}

@override

Widget build(BuildContext context) {

44

return Scaffold( appBar: AppBar(

title: Text('Fade Animation Example'),

),

body: Center(

child: FadeTransition( opacity: \_opacityAnimation, child: Container(

width: 200,

height: 200,

color: Colors.blue, child: Center( child: Text(

'Fade Animation', style: TextStyle( color: Colors.white, fontSize: 20,

),

),

),

),

),

),

);

}

@override

void dispose() {

\_animationController.dispose(); super.dispose();

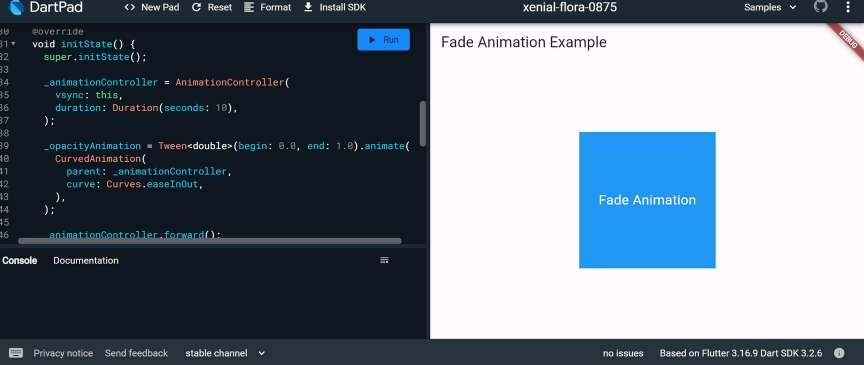
}

}

**Output:**

**Fade Animation**

45



# Slide Animation:

import 'package:flutter/material.dart'; void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget { @override

Widget build(BuildContext context) { return MaterialApp(

title: 'Slide Animation Example', theme: ThemeData( primarySwatch: Colors.blue,

),

home: SlideAnimationWidget(),

);

}

}

class SlideAnimationWidget extends StatefulWidget { @override

\_SlideAnimationWidgetState createState() => \_SlideAnimationWidgetState();

}

class \_SlideAnimationWidgetState extends State<SlideAnimationWidget> with SingleTickerProviderStateMixin {

late AnimationController \_animationController; late Animation<Offset> \_slideAnimation;

@override

46

void initState() { super.initState();

\_animationController = AnimationController( vsync: this,

duration: Duration(seconds: 2),

);

\_slideAnimation = Tween<Offset>( begin: Offset(-1.0, 0.0),

end: Offset(0.0, 0.0),

).animate( CurvedAnimation(

parent: \_animationController, curve: Curves.easeInOut,

),

);

\_animationController.forward();

}

@override

Widget build(BuildContext context) { return Scaffold(

appBar: AppBar(

title: Text('Slide Animation Example'),

),

body: SlideTransition( position: \_slideAnimation, child: Container(

width: 200,

height: 200,

color: Colors.blue, child: Center( child: Text(

'Slide Animation', style: TextStyle( color: Colors.white, fontSize: 20,

),

),

),

),

),

);

}

47

@override

void dispose() {

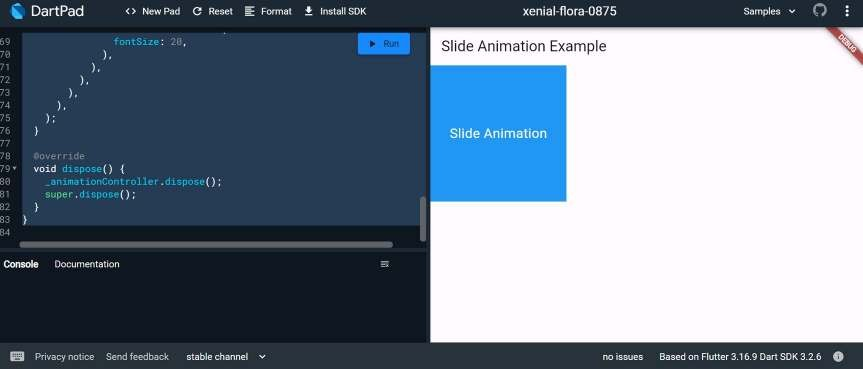
\_animationController.dispose(); super.dispose();

}

}

# Output:

**Slide Animation**



# Scale Animation:

import 'package:flutter/material.dart'; void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget { @override

Widget build(BuildContext context) { return MaterialApp(

title: 'Scale Animation Example', theme: ThemeData( primarySwatch: Colors.blue,

),

home: ScaleAnimationWidget(),

);

}

}

48

class ScaleAnimationWidget extends StatefulWidget { @override

\_ScaleAnimationWidgetState createState() => \_ScaleAnimationWidgetState();

}

class \_ScaleAnimationWidgetState extends State<ScaleAnimationWidget> with SingleTickerProviderStateMixin {

late AnimationController \_animationController; late Animation<double> \_scaleAnimation;

@override

void initState() { super.initState();

\_animationController = AnimationController( vsync: this,

duration: Duration(seconds: 2),

);

\_scaleAnimation = Tween<double>(begin: 0.5, end: 1.0).animate( CurvedAnimation(

parent: \_animationController, curve: Curves.easeInOut,

),

);

\_animationController.forward();

}

@override

Widget build(BuildContext context) { return Scaffold(

appBar: AppBar(

title: Text('Scale Animation Example'),

),

body: ScaleTransition( scale: \_scaleAnimation, child: Container( width: 200,

height: 200,

color: Colors.blue, child: Center( child: Text(

'Scale Animation', style: TextStyle( color: Colors.white,

49

fontSize: 20,

),

),

),

),

),

);

}

@override

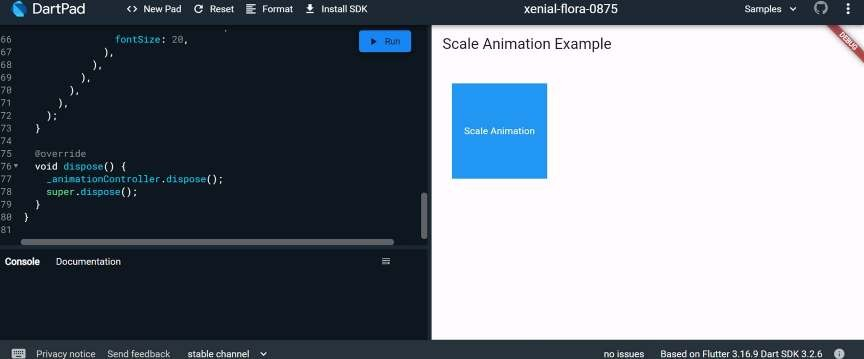
void dispose() {

\_animationController.dispose(); super.dispose();

}

}

# Output:



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

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

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

50

}

class MyApp extends StatelessWidget { @override

Widget build(BuildContext context) { return MaterialApp(

title: 'API Fetch Example', theme: ThemeData( primarySwatch: Colors.blue,

),

home: MyApiFetchWidget(),

);

}

}

class MyApiFetchWidget extends StatefulWidget { @override

\_MyApiFetchWidgetState createState() => \_MyApiFetchWidgetState();

}

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

@override

void initState() { super.initState();

\_posts = fetchPosts();

}

Future<List<Post>> fetchPosts() async { final response =

await http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts'));

if (response.statusCode == 200) {

// If the server returns a 200 OK response,

// parse the JSON and return a list of posts. List<dynamic> data = json.decode(response.body);

List<Post> posts = data.map((post) => Post.fromJson(post)).toList(); return posts;

} else {

// If the server did not return a 200 OK response,

// throw an exception.

throw Exception('Failed to load posts');

}

}

51

@override

Widget build(BuildContext context) { return Scaffold(

appBar: AppBar(

title: Text('API Fetch Example'),

),

body: FutureBuilder<List<Post>>( future: \_posts,

builder: (context, snapshot) {

if (snapshot.connectionState == ConnectionState.waiting) { return CircularProgressIndicator();

} else if (snapshot.hasError) {

return Text('Error: ${snapshot.error}');

} else {

return ListView.builder( itemCount: snapshot.data!.length, itemBuilder: (context, index) { return ListTile(

title: Text(snapshot.data![index].title), subtitle: Text(snapshot.data![index].body),

);

},

);

}

},

),

);

}

}

class Post { final int userId; final int id;

final String title; final String body;

Post({

required this.userId, required this.id, required this.title, required this.body,

});

factory Post.fromJson(Map<String, dynamic> json) { return Post(

userId: json['userId'],

52

id: json['id'], title: json['title'],

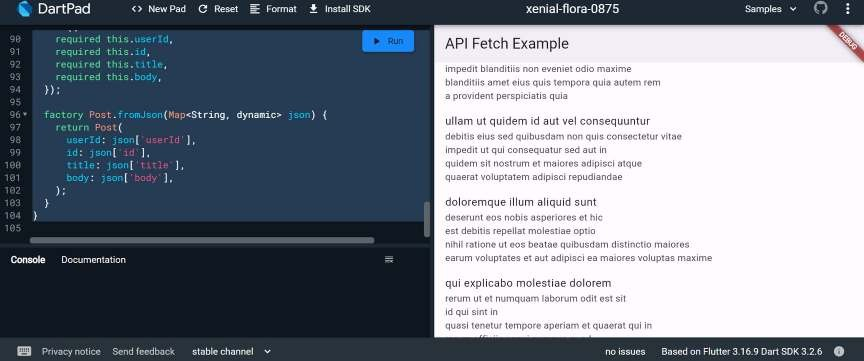
body: json['body'],

);

}

}

# Output:



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

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

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

}

class MyApp extends StatelessWidget { @override

Widget build(BuildContext context) { return MaterialApp(

title: 'API Fetch Example', theme: ThemeData( primarySwatch: Colors.blue,

),

home: MyApiFetchWidget(),

);

}

53

}

class MyApiFetchWidget extends StatefulWidget { @override

\_MyApiFetchWidgetState createState() => \_MyApiFetchWidgetState();

}

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

@override

void initState() { super.initState();

\_posts = fetchPosts();

}

Future<List<Post>> fetchPosts() async { final response =

await http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts'));

if (response.statusCode == 200) {

List<dynamic> data = json.decode(response.body);

List<Post> posts = data.map((post) => Post.fromJson(post)).toList(); return posts;

} else {

throw Exception('Failed to load posts');

}

}

@override

Widget build(BuildContext context) { return Scaffold(

appBar: AppBar(

title: Text('API Fetch Example'),

),

body: FutureBuilder<List<Post>>( future: \_posts,

builder: (context, snapshot) {

if (snapshot.connectionState == ConnectionState.waiting) { return Center(child: CircularProgressIndicator());

} else if (snapshot.hasError) {

return Center(child: Text('Error: ${snapshot.error}'));

} else {

return PostList(posts: snapshot.data!);

}

},

54

),

);

}

}

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

PostList({required this.posts}); @override

Widget build(BuildContext context) { return ListView.builder(

itemCount: posts.length, itemBuilder: (context, index) { return PostItem(post: posts[index]);

},

);

}

}

class PostItem extends StatelessWidget { final Post post;

PostItem({required this.post}); @override

Widget build(BuildContext context) { return Card(

margin: EdgeInsets.all(10), elevation: 3,

child: Padding(

padding: EdgeInsets.all(15), child: Column(

crossAxisAlignment: CrossAxisAlignment.start, children: [

Text( post.title,

style: TextStyle( fontSize: 18,

fontWeight: FontWeight.bold,

),

),

SizedBox(height: 10), Text(

post.body,

55

style: TextStyle(fontSize: 16),

),

],

),

),

);

}

}

class Post { final int userId; final int id;

final String title; final String body;

Post({

required this.userId, required this.id, required this.title, required this.body,

});

factory Post.fromJson(Map<String, dynamic> json) { return Post(

userId: json['userId'], id: json['id'],

title: json['title'], body: json['body'],

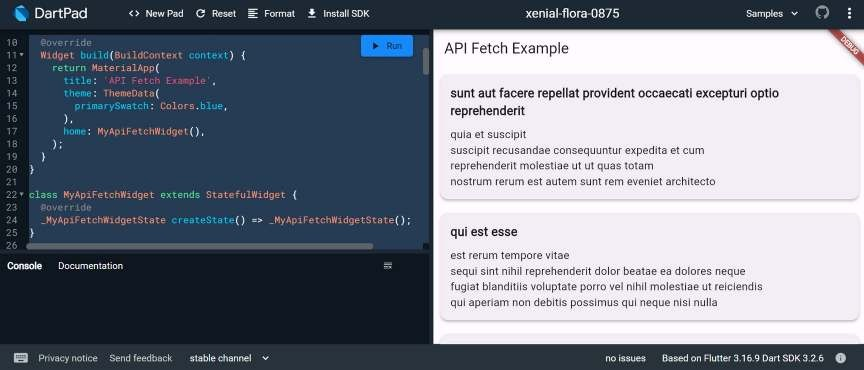
);

}

}

# Output:

56



# 10. a) Write unit tests for UI components.

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

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

Add the test or flutter\_test dependency. Create a test file.

Create a class to test. Write a test for our class.

Combine multiple tests in a group. Run the tests.

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

1. Add the test dependency

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

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

flutter pub add dev:test

1. Create a test file

In this example, create two files: counter.dart and counter\_test.dart.

The counter.dart file contains a class that you want to test and resides in the lib folder. The counter\_test.dart file contains the tests themselves and lives inside the test folder.

57

In general, test files should reside inside a test folder located at the root of your Flutter application or package. Test files should always end with \_test.dart, this is the convention used by the test runner when searching for tests.

When you’re finished, the folder structure should look like this: content\_copy

counter\_app/ lib/

counter.dart test/

counter\_test.dart

1. Create a class to test

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

content\_copy class Counter { int value = 0;

void increment() => value++;

void decrement() => value--;

}

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

1. 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.

content\_copy

// Import the test package and Counter class import 'package:counter\_app/counter.dart'; import 'package:test/test.dart';

void main() {

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

counter.increment();

58

expect(counter.value, 1);

});

}

1. Combine multiple tests in a group

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

content\_copy

import 'package:counter\_app/counter.dart'; import 'package:test/test.dart';

void main() {

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

});

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

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

});

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

counter.decrement();

expect(counter.value, -1);

});

});

}

1. Run the tests

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

Run tests using IntelliJ or VSCode

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

IntelliJ

Open the counter\_test.dart file

59

Go to Run > Run ‘tests in counter\_test.dart’. You can also press the appropriate keyboard shortcut for your platform.

VSCode

Open the counter\_test.dart file

Go to Run > Start Debugging. You can also press the appropriate keyboard shortcut for your platform.

Run tests in a terminal

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

content\_copy

flutter test test/counter\_test.dart

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

content\_copy

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

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

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

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

1. Flutter DevTools:

Run your app with the flutter run command.

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

flutter pub global activate devtools flutter pub global run devtools

Open your app in a Chrome browser and connect it to DevTools by clicking on the "Open DevTools" button in the terminal or by navigating to http://127.0.0.1:9100/.

DevTools provides tabs like Inspector, Timeline, Memory, and more.

1. 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.

60

1. 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.

1. 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.

1. 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.

1. 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());

}

1. Memory Profiling:

Use the "Memory" tab in DevTools to analyze memory usage and identify potential memory leaks.

Monitor object allocations and deallocations.

1. Performance Profiling (Timeline):

Analyze app performance using the "Timeline" tab in DevTools. Identify UI jank, slow frames, and performance bottlenecks.

1. Flutter Driver Tests:

Write automated UI tests using Flutter Driver.

Simulate user interactions and validate the correctness of your UI.

61