**GURU NANAK INSTITUTIONS TECHNICAL CAMPUS (AUTONOMOUS)**

**School of Engineering & Technology**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**UI design-Flutter**

**Lab Manual**

**[Course Code:** **22SDOCS03]**

**For the Academic year 2024-25**

**III B.Tech. I Semester**



**Guru Nanak Institutions Technical Campus (Autonomous)**

Ibrahimpatnam, R R District – 501 506 (T. S.)

**LAB MANUAL FOR THE ACADEMIC YEAR 2024-25**

**Name of the Lab :** UI-design Flutter

**Lab Course Code :** 22SDOCS03

**Year & Semester : III Year I Sem**

**Branch : CSE**

**No. of Hours : 2 Practical Hours per Week**

**No. of Credits : 1.5**

**Document No. :**

**Date of Revision :**

**Date of Issue :**

**Prepared By :**  **Sheikh Riyaz ul Haq**

**Programmer : N. Lavanya**

**Verified by : Dr. Kishore Verma S**

**Authorized by : Dr. Geeta Tripathi (HOD)**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **CONTENTS** | **PAGE No.** |
|  | Lab Course Syllabus |  |
|  | Vision & Mission of the Institute |  |
|  | Vision & Mission of the Department |  |
|  | Program Educational Objectives (PEOs) |  |
|  | Program Outcomes (POs) |  |
|  | Program Specific Outcomes (PSOs) |  |
|  | Lab Information Sheet |  |

## **Lab Course SYLLABUS (UI Design -Flutter Lab)**

### 

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

1. a) Install Flutter and Dart SDK.

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

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

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

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

b) Implement media queries and breakpoints for responsiveness.

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

b) Implement navigation with named routes.

5. a) Learn about stateful and stateless widgets.

b) Implement state management using set State and Provider.

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

b) Apply styling using themes and custom styles.

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

b) Implement form validation and error handling.

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

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

9. a) Fetch data from a REST API.

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

10. a) Write unit tests for UI components.

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

## **VISION & MISSION OF THE INSTITUTION: GNITC**

## **VISION OF THE INSTITUTION: GNITC**

To be an internationally renowned institution in Engineering, Management, Pharmacy and related fields to produce scientists, engineers, entrepreneurs, leaders, academicians and thinkers of tomorrow with exemplary professional conduct and adherence to ethical values to serve for changing needs of industry and society.

## **MISSION OF THE INSTITUTION: GNITC**

**M1:** Imbibe soft skills, technical skills, creatively and passion in students.

**M2:** Develop the faculty to reach the International standards.

**M3:** Maintain outcome based student centric teaching learning with high academic standards and quality

that promotes the analytical thinking and independent judgment.

**M4:** Promote research, innovation, product development by collaborating with reputed industries &

reputed universities in India and abroad. Offer collaborative industry programs in emerging areas

and instill the spirit of enterprising.

**M5:** To instill the ethical values in the faculty and students to serve the society.

## **VISION& MISSION OF THE DEPARTMENT**

## **VISION OF THE DEPARTMENT: CSE**

To become a premier Computer Science & Engineering department by imparting high quality education, ethical values, provide creative environment for innovation and global career opportunities.

## **MISSION OF THE DEPARTMENT: CSE**

**M1:** Nurture young individuals into knowledgeable, skillful and ethical professionals in their pursuit of Computer Science & Engineering.

**M2:** Foster the students through excellent teaching learning process and sustain high performance

through innovation.

**M3:** Provide high quality soft skills and advanced industry specific technical trainings to meet global career opportunities.

## **PROGRAMME EDUCATIONAL OBJECTIVES (PEOs):**

**PEO 1:** Graduates shall have the ability to apply knowledge across the disciplines and in emerging areas of Computer Science and Engineering for higher studies, research, employability, product development and handle the realistic problems.

**PEO 2:** Graduates shall have good communication skills, possess ethical values, sense of responsibility to serve the society, and protect the environment.

**PEO 3:** Graduates shall possess academic excellence with innovative insight, managerial skills, leadership qualities, knowledge of contemporary issues and understand the need for lifelong learning for a successful professional career.

## **PROGRAMME OUTCOMES (POs): [Department of Computer Science & Engineering]**

The following list of programme outcomes describes what graduates are expected to know and be able to do at the time of graduation. Graduates at graduation will have:

**1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

**2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

**3. Design/Development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

**4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**6. The Engineer and Society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

**7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**9. Individual and team work:** Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.

**10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**11. Project management and finance:** Demonstrate knowledge and understanding of the engineering management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**12. Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

## **PROGRAMME SPECIFIC OUTCOMES (PSOs):**

**PSO 1:** Ability to apply software technical skills to design, develop and debug optimized solutions for all real world problems.

**PSO 2:** Ascertain knowledge in diverse areas of Computer Science and experience an environment conducive in novel skills for successful career, entrepreneurship and higher studies.

1. **COURSE INFORMATION SHEET**

|  |  |
| --- | --- |
| PROGRAMME: B. Tech.  (Computer Science & Engineering) | DEGREE: B.Tech. |
| COURSE: UI-design Flutter | SEMESTER: III CREDITS: 2 |
| COURSE CODE: 22SDOCS03  REGULATION: R21 | COURSE TYPE: Lab |
| COURSE AREA/DOMAIN: Mobile App Development | CONTACT HOURS: 2+1 (Tutorial) hrs/week |

**SYLLABUS**

**List of Experiments**

|  |  |  |
| --- | --- | --- |
| **Exp. No.** | **Experiment Name** | **Weeks** |
| 1 | 1. Install Flutter and Dart SDK 2. Write a simple Dart program to understand the language basics. | **3 Weeks** |
| 2 | a) Explore various Flutter widgets (Text, Image, Container, etc.).  b) Implement different layout structures using Row, Column, and Stack widgets |
| 3 | a) Design a responsive UI that adapts to different screen sizes.  b) Implement media queries and breakpoints for responsiveness. | **3 Weeks** |
| 4 | a) Set up navigation between different screens using Navigator.  b) Implement navigation with named routes. |
| 5 | a) Learn about stateful and stateless widgets.  b) Implement state management using set State and Provider | **3 Weeks** |
| 6 | a) Create custom widgets for specific UI elements.  b) Apply styling using themes and custom style |
| 7 | a) Design a form with various input fields.  b) Implement form validation and error handling. | **3 Weeks** |
| 8 | a) Add animations to UI elements using Flutter's animation framework.  b) Experiment with different types of animations (fade, slide, etc.). |
| 9 | a) Fetch data from a REST API.  b) Display the fetched data in a meaningful way in the UI. | **4 Weeks** |
| 10 | a) Write unit tests for UI components.  b) Use Flutter's debugging tools to identify and fix issues. |

**COURSE PRE-REQUISITES:**

|  |  |  |
| --- | --- | --- |
| **Course Name** | **Description** | **Sem** |
| Object-Oriented Programming | To understand the basics of object-oriented programming using languages like Java or Python | II |
| Web Technologies | To have knowledge of web development and basic understanding of frontend technologies | III |

**COURSE OBJECTIVE:**

|  |
| --- |
| This lab will provide students with a strong foundation in Flutter development and prepare them for real-world app development challenges. This Lab will equip students with the practical skills to build cross-platform mobile applications efficiently and effectively.   * **Master Flutter fundamentals: Gain a solid understanding of Flutter's architecture, widgets, and state management.** * **Develop cross-platform applications:** Build mobile apps that run seamlessly on both iOS and Android platforms. * **Create visually appealing UIs:** Design and implement user interfaces that adhere to platform-specific design guidelines. * **Optimize app performance:** Identify and address performance bottlenecks in Flutter apps. * **Implement state management effectively:** Utilize appropriate state management solutions for different app scenarios. * **Integrate with external services:** Consume APIs and data sources to build data-driven apps. * **Test and debug Flutter apps:** Write effective unit and widget tests to ensure app quality. |

**COURSE OUTCOMES:**

|  |  |
| --- | --- |
| **S. No.** | **DESCRIPTION** |
| 1 | Students will be able to **define** key Flutter UI components such as widgets, layout, and state management. |
| 2 | Students will **explain** the difference between Stateless Widget and State full Widget.  Students will **describe** the process of building a Flutter layout using Row and Column widgets. Students will **identify** common Flutter UI design patterns. |
| 3 | Students will **use** state management solutions to manage complex app states. Students will **apply** responsive design principles to create UI layouts for different screen sizes. |
| 4 | Students will **analyze** the performance implications of different UI approaches. Students will **differentiate** between various state management solutions based on project requirements. |
| 5 | Students will **justify** the choice of UI components for a specific design problem.  Students will **defend** the usability and accessibility of their Flutter app.  Students will **design** and **develop** a complex Flutter UI for a given user story. |

**CO-PO Mapping (Articulation Matrix)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** |
| **CO1** | **2** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| **CO2** | **-** | **2** | **3** | **-** | **2** | **1** | **-** | **-** | **-** | **3** | **-** | **-** |
| **CO3** | **-** | **-** | **-** | **2** | **3** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| **CO4** | **-** | **-** | **3** | **2** | **2** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| **CO5** | **-** | **-** | **-** | **-** | **3** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |

**Programs/Mini Projects BEYOND SYLLABUS/ADVANCED TOPICS/DESIGN:**

|  |  |
| --- | --- |
| 1 | **Dice App**: |
|  | * This is a simple app where users roll a pair of dice by pressing a button, and the app shows random dice values. This project teaches basic Flutter widgets and layout management. |
| 2 | **Xylophone App**: |
|  | * A fun app that plays different sounds when the user taps on colorful bars. This project introduces you to handling audio assets and interacting with user inputs. |
| 3 | **Quiz App**: |
|  | * A quiz application where users answer a series of questions and get feedback on their performance. This project covers state management, navigating between different screens, and more complex widget interactions. |
| 4 | **BMI Calculator**: |
|  | * This app calculates the Body Mass Index (BMI) based on user inputs and displays the results. The project demonstrates how to manage user inputs, perform calculations, and display the results in a visually appealing way. |

**WEB SOURCE REFERENCES:**

|  |  |
| --- | --- |
| 1 | **Flutter Documentation** <https://flutter.dev/docs> |
| 2 | **Dart Programming Language Documentation** <https://dart.dev/guides> |
| 3 | **Academind - Flutter Tutorial Series** <https://academind.com/learn/flutter> |
| 4 | **GitHub - Flutter Samples** <https://github.com/flutter/samples> |
| 5 | **Flutter** <https://www.youtube.com/c/flutterdev> |
| 6 | **The Net Ninja** <https://www.youtube.com/playlist?list=PL4cUxeGkcC9gZdxaIwjgwqqeRF3EO8sYv> |
| 7 | **Traversy Media** <https://www.youtube.com/playlist?list=PLillGF-RfqbZrjw48EXLdM4dsOhURCLZx> |

**DELIVERY/INSTRUCTIONAL METHODOLOGIES:**

|  |  |  |  |
| --- | --- | --- | --- |
| * PPT & TALK | * STUD. ASSIGNMENT- | * WEB RESOURCES | * VIDEO LECTURES |
| * LCD/SMART BOARDS | * STUD. SEMINARS | ☐ ADD-ON COURSES |  |

**ASSESSMENT METHODOLOGIES-DIRECT:**

|  |  |  |  |
| --- | --- | --- | --- |
| ✓ASSIGNMENTS | **√** STUD. SEMINARS | **√** TESTS/MODEL EXAMS | **√** UNIV. EXAMINATION |
| ✓ STUD. LAB PRACTICES | ✓STUD. VIVA | ✓MINI/MAJOR PROJECTS | ✓CERTIFICATIONS |
| ☐ ADD-ON COURSES | ☐ OTHERS |  |  |

**ASSESSMENT METHODOLOGIES-INDIRECT:**

|  |  |
| --- | --- |
| **√**ASSESSMENT OF COURSE OUTCOMES (BY FEEDBACK, ONCE) | ☐ STUDENT FEEDBACK ON FACULTY (TWICE)-YES |
| ☐ ASSESSMENT OF MINI/MAJOR PROJECTS BY EXT. EXPERTS | ☐ OTHERS |

## **SYLLABUS**

### **UI Design -Flutter Lab**

#### **B.TECH III Year I Sem**

1. a) Install Flutter and Dart SDK.

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

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

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

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

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

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

#### b) Implement navigation with named routes.

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

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

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

#### b) Apply styling using themes and custom styles.

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

#### b) Implement form validation and error handling.

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

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

9. a) Fetch data from a REST API.

b) Display the fetched data in a meaningful way in the UI

10. a) Write unit tests for UI components.

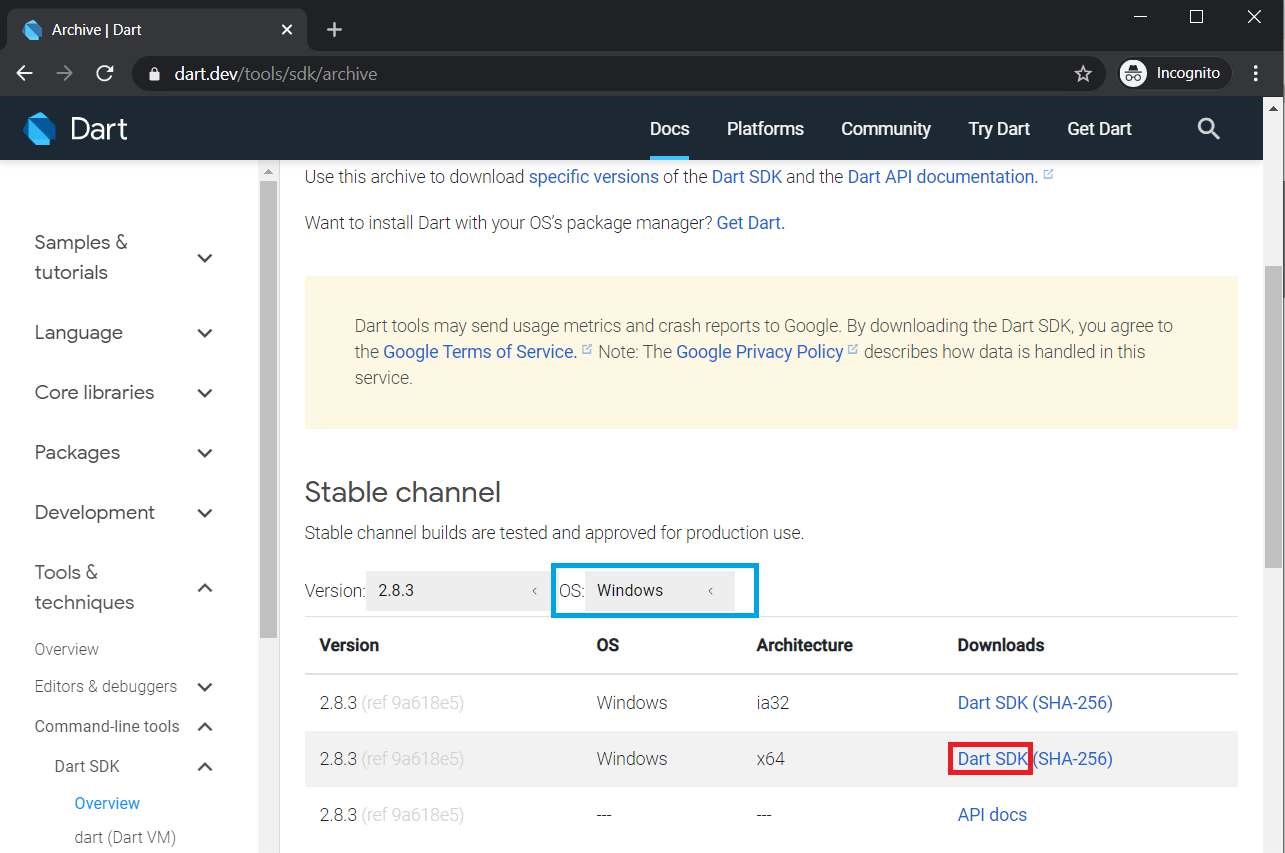
b) Use Flutter's debugging tools to identify and fix issues.

**EXPERIMENT NO: 1.**

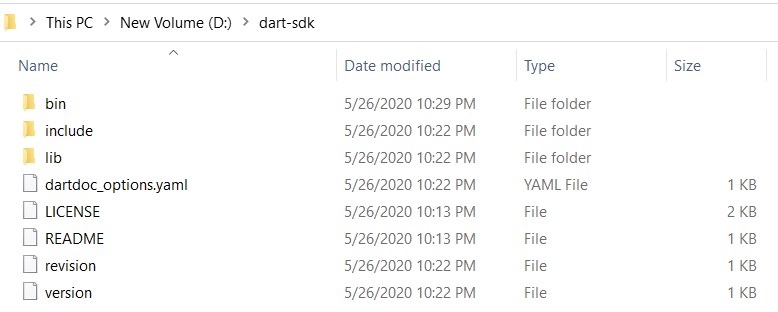
1. a) Install Flutter and Dart SDK.

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

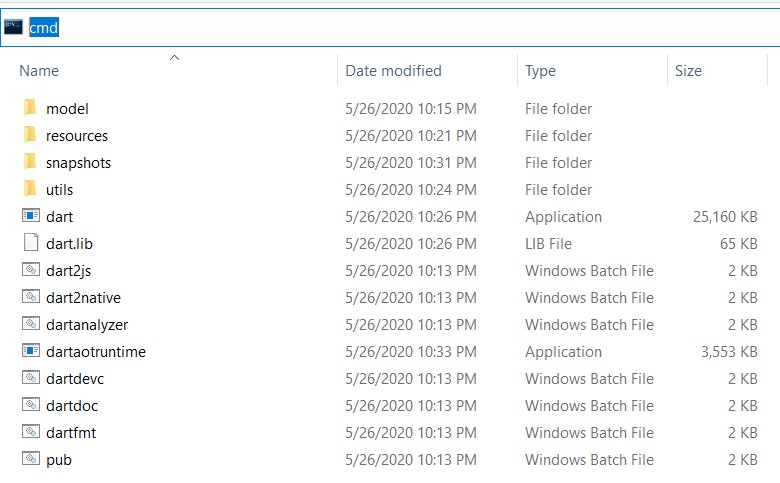
The URL is: https://dart.dev/tools/sdk/archive



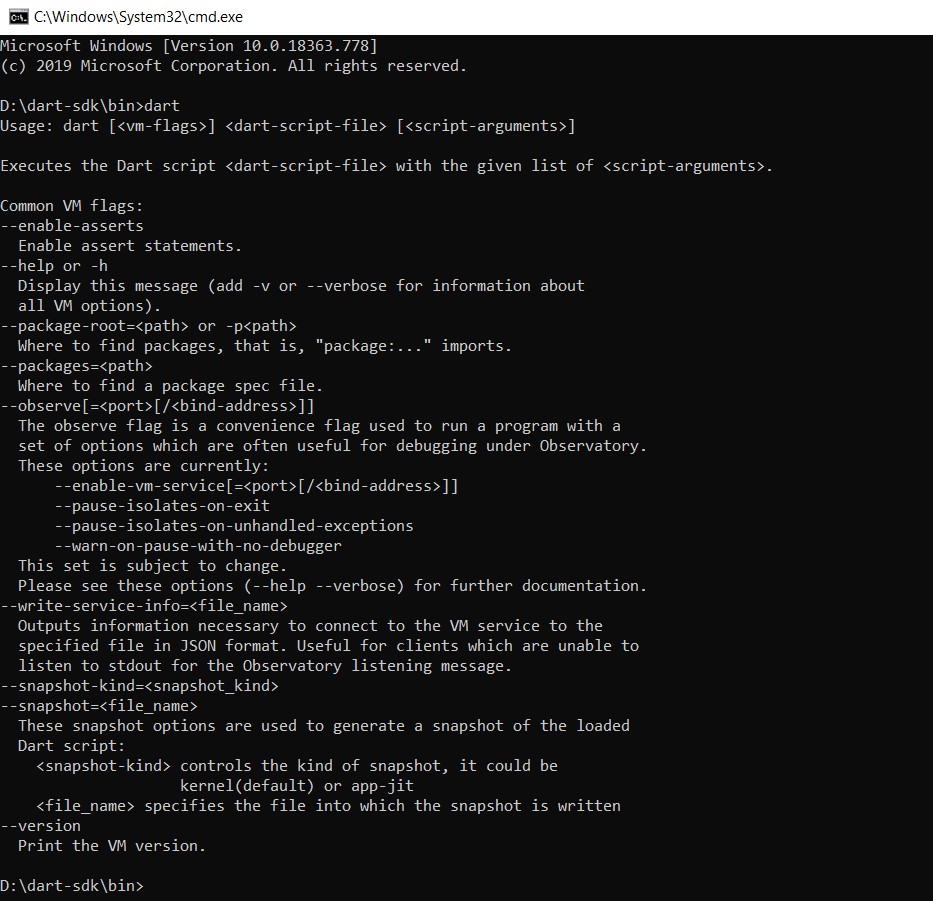
Click on DART SDK to download SDK for Windows 64-Bit Architecture. The download will start and a zip file will be downloaded. **Note:** To download SDK for any other OS select OS of your choice. **Step 2:** Extract the downloaded zip file. Extract the contents of downloaded zip file and after extracting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



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

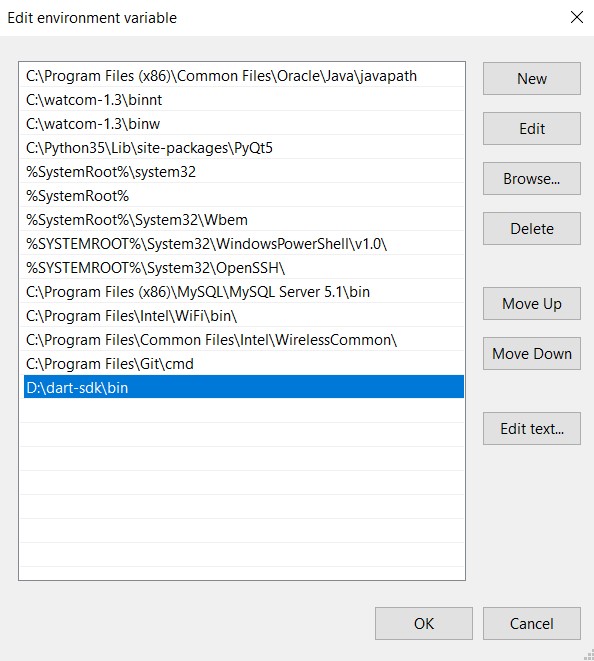


Command Prompt will open with our desired path of bin folder and now type dart”.



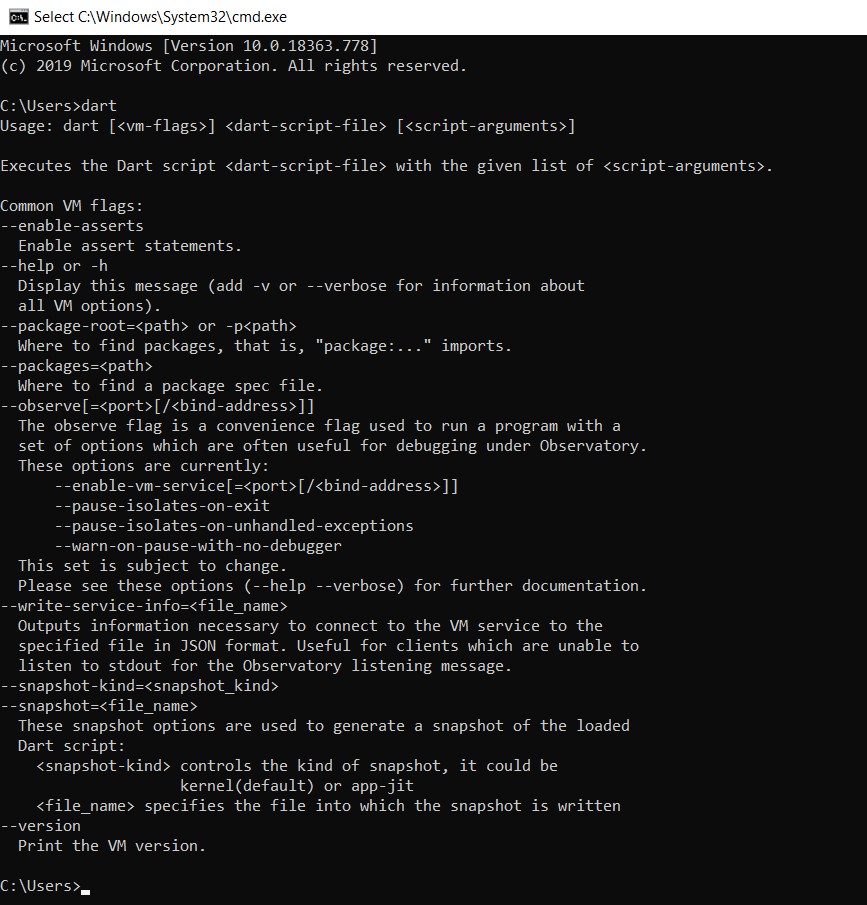
And now we are ready to use dart through bin folder but setting up the path in environment variables will ease our task of Step3 and we can run dart from anywhere in the file system using command prompt.

**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:



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

**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; 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**

**EXPERIMENT NO: 2.**

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

**Text Widget:**

import 'package:flutter/material.dart';

void main() {

  runApp(

    MaterialApp(

      home: Scaffold(

        backgroundColor: Color.fromARGB(255, 233, 118, 30),

        body: Center(

          child: Text('Hello World',style: TextStyle(

            color: Color.fromARGB(255, 11, 7, 255),

            fontSize: 60,

          ),)

          ),

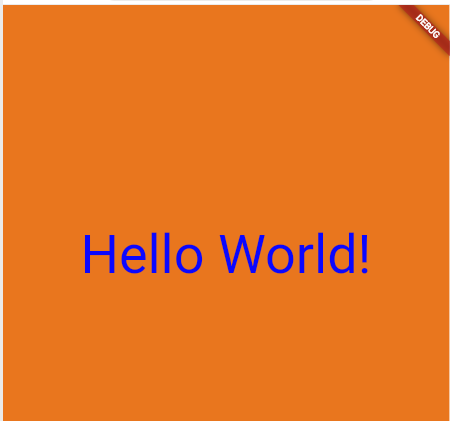
      ),

    ),

  );

}

**Output:**

****

**Image Widget:**

import 'package:flutter/material.dart';

void main() {

  runApp(

    MaterialApp(

      home: Scaffold(

        backgroundColor: Color.fromRGBO(79, 173, 190, 1),

        body: Center(

          child:

 Image.network(

            'https://i.imgur.com/lA89x2h\_d.webp?maxwidth=520&shape=thumb&fidelity=high' // Replace with the actual path to your image on your desktop

          ),

        ),

      ),

    ),

  );

}



**Output**

**Containter Widget:**

import 'package:flutter/material.dart';

void main() {

  runApp(

    MaterialApp(

      home: Scaffold(

        appBar: AppBar(

          title: const Text("Container example"),

        ),

        body: Container(

          height: 200,

          width: double.infinity,

          //color: Colors.purple,

 // Uncomment to set background color

          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),

          ),

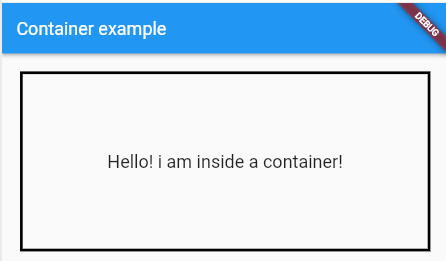
        ),

      ),

    ),

  );

}



**Output:**

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

**ROW Widget:**

import 'package:flutter/material.dart';

void main() {

  runApp(

    MaterialApp(

      home: Scaffold(

        appBar: AppBar(

          title: const Text('Technologies'), // Updated title

        ),

        body: Row(

          mainAxisAlignment: MainAxisAlignment.spaceBetween, // Even spacing

          crossAxisAlignment: CrossAxisAlignment.center, // Center alignment

          children: [

            Container(

              width: 100,

              height: 100,

              color: Colors.red, // Red container

              child: const Center(

                child: Text('React.JS'), // Text inside red container

              ),

            ),

            Container(

              width: 100,

              height: 100,

              color: Colors.green, // Green container

              child: const Center(

                child: Text('Flutter'), // Text inside green container

              ),

            ),

            Container(

              width: 100,

              height: 100,

              color: Colors.orange, // Orange container

              child: const Center(

                child: Text('MySQL'), // Text inside orange container

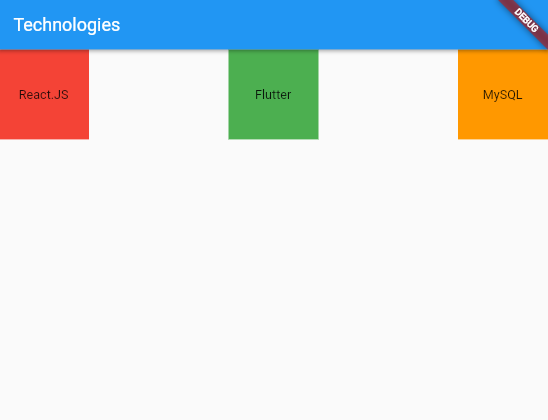
              ),

            ),

          ],

        ),

      ),

    ),

  );

}

**Output:**

**Column Widget:**

import 'package:flutter/material.dart';

void main() {

  runApp(

    MaterialApp(

      home: Scaffold(

        appBar: AppBar(

          title: const Text('Technologies'),

        ),

        body: Column(

          mainAxisAlignment: MainAxisAlignment.spaceBetween,

          crossAxisAlignment: CrossAxisAlignment.center,

          children: [

            Container(

              width: 200,

              height: 100,

              color: Colors.red,

              child: const Center(

                child: Text('React.JS'),

              ),

            ),

            Container(

              width: 200,

              height: 100,

              color: Colors.green,

              child: const Center(

                child: Text('Flutter'),

              ),

            ),

            Container(

              width: 200,

              height: 100,

              color: Colors.orange,

              child: const Center(

                child: Text('MySQL'),

              ),

            ),

          ],

        ),

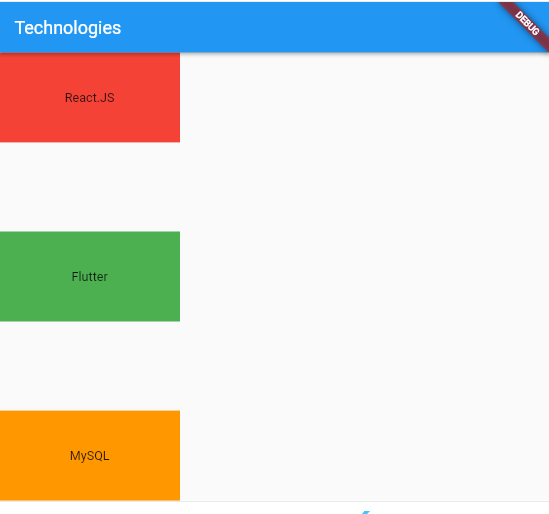
      ),

    ),

  );

}

Output:



**Stack Widget:**

import 'package:flutter/material.dart';

void main() {

runApp(

MaterialApp(

home: Scaffold(

appBar: AppBar(

title: const Text('Technologies'),

),

body: Stack(

children: [

Container(

width: 200,

height: 200,

color: Colors.red,

child: const Center(

child: Text('React.JS', style: TextStyle(color: Colors.white)),

),

),

Positioned(

top: 10,

left: 10,

child: Container(

width: 180,

height: 180,

color: Colors.green,

child: const Center(

child: Text('Flutter', style: TextStyle(color: Colors.white)),

),

),

),

Positioned(

top: 30,

left: 30,

child: Container(

width: 150,

height: 150,

color: Colors.orange,

child: const Center(

child: Text('MySQL', style: TextStyle(color: Colors.white)),

),

),

),

],

),

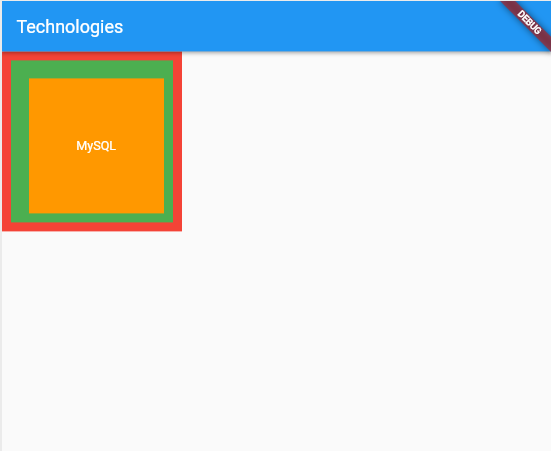
),

),

);

}

**Output:**



**EXPERIMENT NO: 3.**

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

**Ans)**

import 'package:flutter/material.dart';

void main() {

  runApp(

    MaterialApp(

      home: Scaffold(

        appBar: AppBar(

          title: Text('Responsive UI'),

        ),

        body: OrientationBuilder(

          builder: (context, orientation) {

            return LayoutBuilder(

              builder: (context, constraints) {

                if (constraints.maxWidth

 < 300) {

                  // Layout for smaller screens (portrait mode)

                  return Column(

                    crossAxisAlignment: CrossAxisAlignment.start,

                    children: [

                      Container(

                        color: Colors.blue,

                        padding: EdgeInsets.all(16.0),

                        child: Text('Left Content'),

                      ),

                      Container(

                        color: Colors.green,

                        padding: EdgeInsets.all(16.0),

                        child: Text('Right Content'),

                      ),

                    ],

                  );

                } else {

                  // Layout for larger screens (landscape mode)

                  return Row(

                    crossAxisAlignment: CrossAxisAlignment.start,

                    children: [

                      Expanded(

                        flex: 3,

                        child: Container(

                          color: Colors.blue,

                          padding: EdgeInsets.all(16.0),

                          child: Text('Left Content'),

                        ),

                      ),

                      Expanded(

                        flex: 2,

                        child: Container(

                          color: Colors.green,

                          padding: EdgeInsets.all(16.0),

                          child: Text('Right Content'),

                        ),

                      ),

                    ],

                  );

                }

              },

            );

          },

        ),

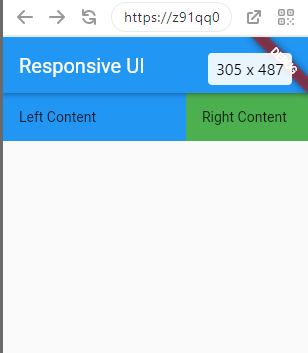
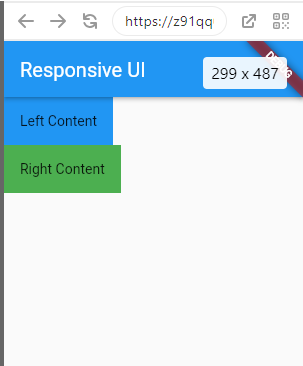
      ),

    ),

  );

}

**Output:**



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

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

void main() {

  runApp(

    MaterialApp(

      home: Scaffold(

        appBar: AppBar(

          title: Text('Responsive UI with MediaQuery'),

        ),

        body: Builder(

          builder: (context) {

            var screenWidth = MediaQuery.of(context).size.width;

            if (screenWidth < 600) {

              // Layout for smaller screens (portrait mode)

              return Column(

                crossAxisAlignment: CrossAxisAlignment.start,

                children: [

                  Container(

                    color: Colors.blue,

                    padding: EdgeInsets.all(16.0),

                    child: Text('Left Content'),

                  ),

                  Container(

                    color: Colors.green,

                    padding: EdgeInsets.all(16.0),

                    child: Text('Right Content'),

                  ),

                ],

              );

            } else {

              // Layout for larger screens (landscape mode)

              return Row(

                crossAxisAlignment: CrossAxisAlignment.start,

                children: [

                  Expanded(

                    flex: 3,

                    child: Container(

                      color: Colors.blue,

                      padding: EdgeInsets.all(16.0),

                      child: Text('Left Content'),

                    ),

                  ),

                  Expanded(

                    flex: 2,

                    child: Container(

                      color: Colors.green,

                      padding: EdgeInsets.all(16.0),

                      child: Text('Right Content'),

                    ),

                  ),

                ],

              );

            }

          },

        ),

      ),

    ),

  );

}

Output:

**EXPERIMENT NO: 4.**

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

import 'package:flutter/material.dart';

void main() {

  runApp(

    MaterialApp(

      home: HomeScreen(),

    ),

  );

}

class HomeScreen extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text('Navigation Example')),

      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            Text('Welcome to the Home Screen'),

            ElevatedButton(

              onPressed: () {

                Navigator.push(

                  context,

                  MaterialPageRoute(builder: (context) => AboutScreen()),

                );

              },

              child: Text('Go to About'),

            ),

          ],

        ),

      ),

    );

  }

}

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: [

            Text('This is the About Screen'),

            ElevatedButton(

              onPressed: () {

                Navigator.pop(context);

              },

              child: Text('Go to Home'),

            ),

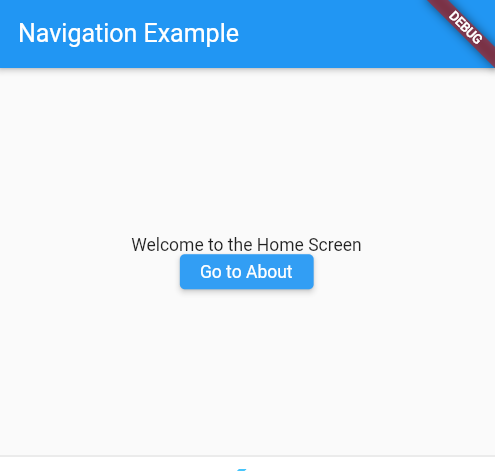
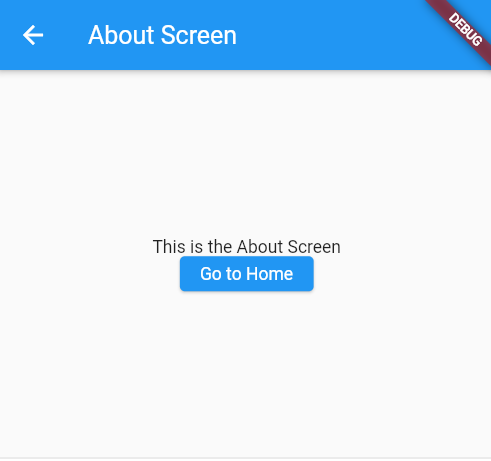
          ],

        ),

      ),

    );

  }

}

**Output:**

import 'package:flutter/material.dart';

void main() {

  runApp(MaterialApp(

    home: Builder(

      builder: (context) => Scaffold(

        appBar: AppBar(

          title: Text('Navigation Example'),

        ),

        body: Center(

          child: Column(

            mainAxisAlignment: MainAxisAlignment.center,

            children: [

              Text('Welcome to the Home Screen'),

              ElevatedButton(

                onPressed: () {

                  Navigator.push(

                    context,

                    MaterialPageRoute(

                      builder: (context) => Scaffold(

                        appBar: AppBar(title: Text('About Screen')),

                        body: Center(

                          child: Column(

                            mainAxisAlignment: MainAxisAlignment.center,

                            children: [

                              Text('This is the About Screen'),

                              ElevatedButton(

                                onPressed: () {

                                  Navigator.pop(context);

                                },

                                child: Text('Go to Home'),

                              ),

                            ],

                          ),

                        ),

                      ),

                    ),

                  );

                },

                child: Text('Go to About'),

              ),

            ],

          ),

        ),

      ),

    ),

  ));

}

**4b) Implement navigation with named routes.**

**Ans)**

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

**void main() {**

**runApp(**

**MaterialApp(**

**initialRoute: '/',**

**routes: {**

**'/': (context) => HomeScreen(), // The home screen is the initial route.**

**'/about': (context) => AboutScreen(), // Named route for the About screen.**

**},**

**),**

**);**

**}**

**class HomeScreen extends StatelessWidget {**

**@override**

**Widget build(BuildContext context) {**

**return Scaffold(**

**appBar: AppBar(title: Text('Navigation Example')),**

**body: Center(**

**child: Column(**

**mainAxisAlignment: MainAxisAlignment.center,**

**children: [**

**Text('Welcome to the Home Screen'),**

**ElevatedButton(**

**onPressed: () {**

**// Navigate to the About Screen using named route**

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

**},**

**child: Text('Go to About'),**

**),**

**],**

**),**

**),**

**);**

**}**

**}**

**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: [**

**Text('This is the About Screen'),**

**ElevatedButton(**

**onPressed: () {**

**// Pop to return to the previous screen (Home Screen)**

**Navigator.pop(context);**

**},**

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

**),**

**],**

**),**

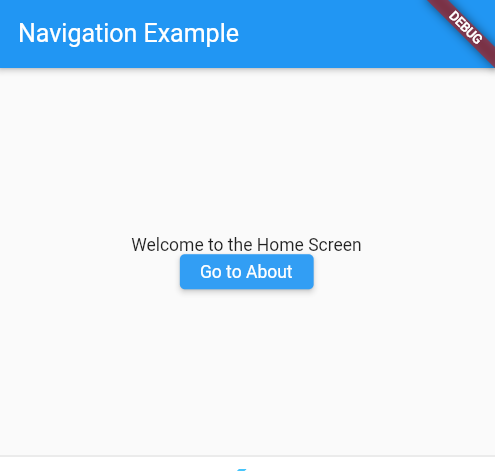
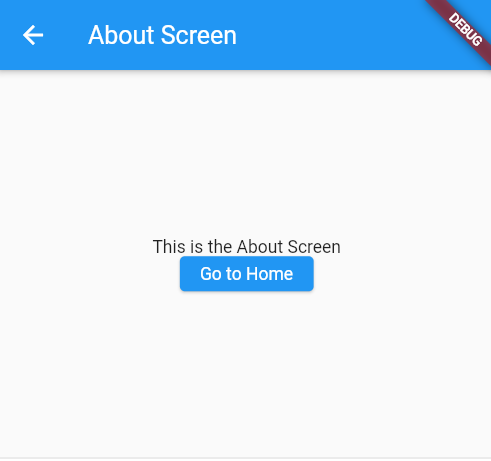
**),**

**);**

**}**

**}**

**Output:**



import 'package:flutter/material.dart';

void main() {

  runApp(

    MaterialApp(

      initialRoute: '/',

      routes: {

        '/': (context) => Scaffold(

              appBar: AppBar(title: Text('Navigation Example')),

              body: Center(

                child: Column(

                  mainAxisAlignment: MainAxisAlignment.center,

                  children: [

                    Text('Welcome to the Home Screen'),

                    ElevatedButton(

                      onPressed: () {

                        // Navigate to the About Screen using named route

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

                      },

                      child: Text('Go to About'),

                    ),

                  ],

                ),

              ),

            ),

        '/about': (context) => Scaffold(

              appBar: AppBar(title: Text('About Screen')),

              body: Center(

                child: Column(

                  mainAxisAlignment: MainAxisAlignment.center,

                  children: [

                    Text('This is the About Screen'),

                    ElevatedButton(

                      onPressed: () {

                        // Navigate back to Home Screen

                        Navigator.pop(context);

                      },

                      child: Text('Go to Home'),

                    ),

                  ],

                ),

              ),

            ),

      },

    ),

  );

}

**EXPERIMENT NO: 5.**

**5. 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(

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) {

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,

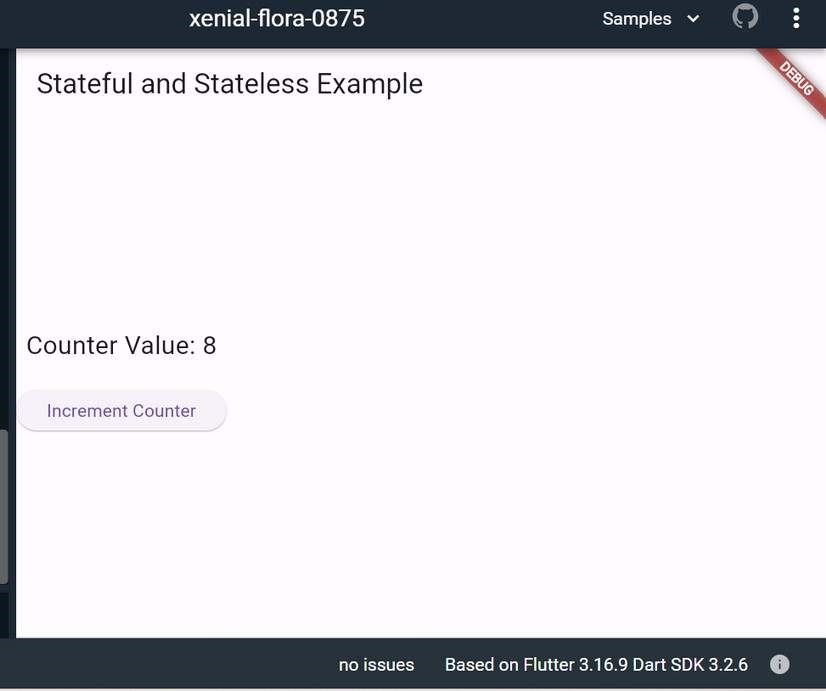
child: Text('Increment Counter'),

);

}

}

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(),

),

);

}

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),

ElevatedButton( onPressed: counterModel.incrementCounter,

child: Text('Increment Counter'),

),

],

),

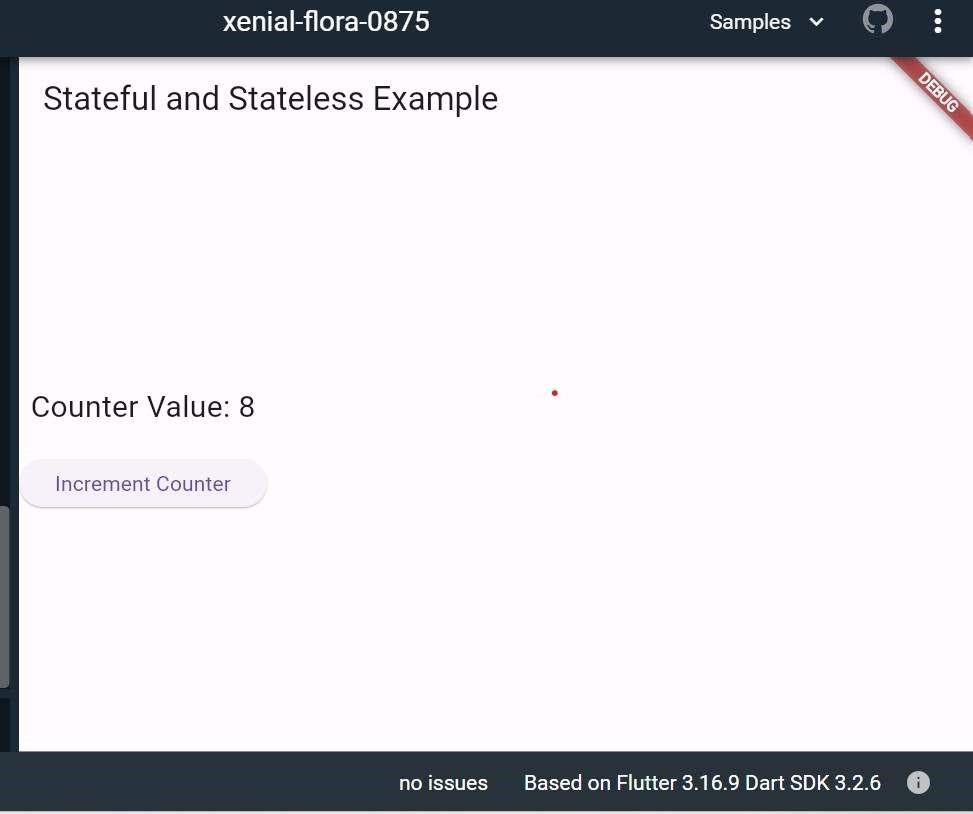
),

);

}

}

**Output:**



**EXPERIMENT NO: 6**

**6. 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, });

@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(),

),

],

);

}

} 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');

},

),

),

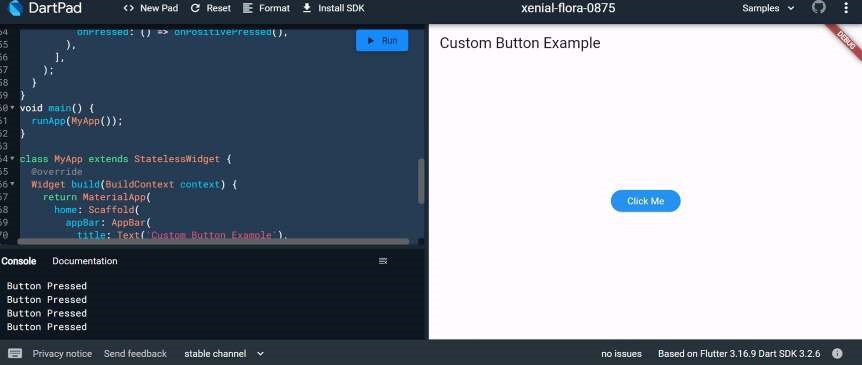
),

);

}

}

**Output:**



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

**Ans)**

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(),

),

),

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".

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

brightness: Brightness.dark,

),

),

child: FloatingActionButton( onPressed: () {},

child: const Icon(Icons.add),

),

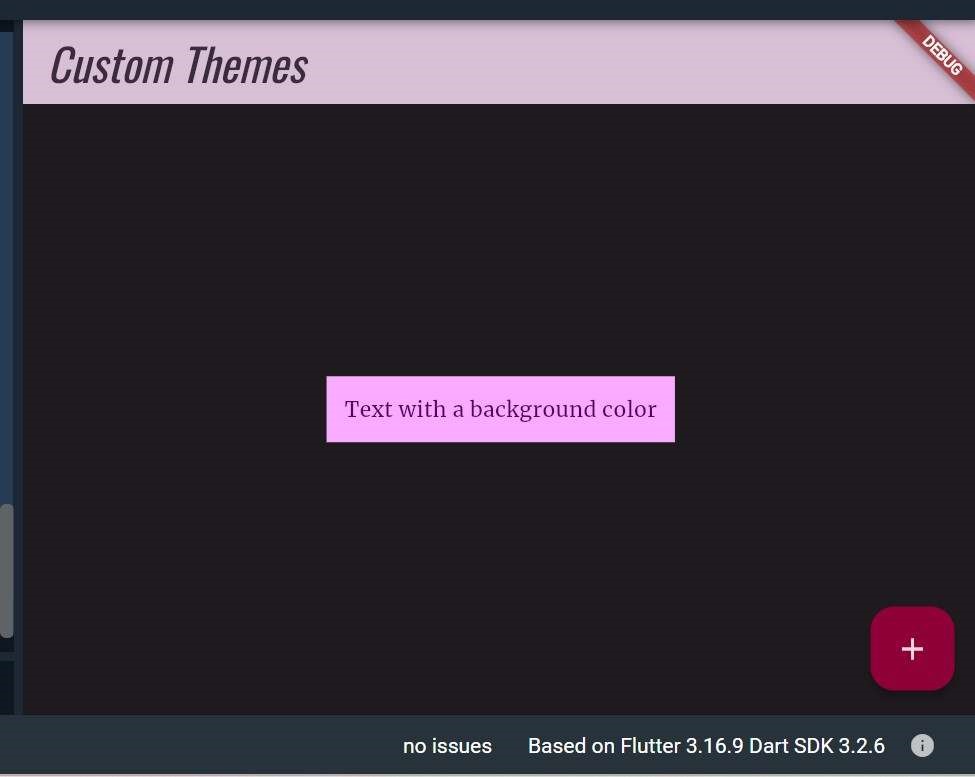
),

);

}

}

Output:



**EXPERIMENT NO: 7.**

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

import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) { return MaterialApp(

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',

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'),

),

],

),

),

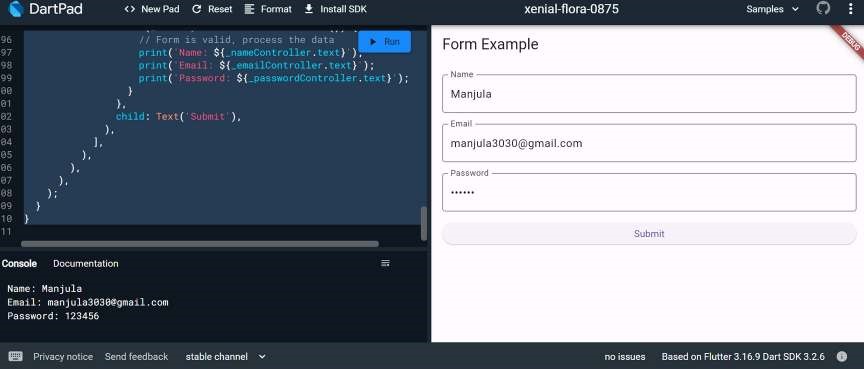
),

);

}

}

Output:



**7 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(); @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),

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'),

),

],

),

),

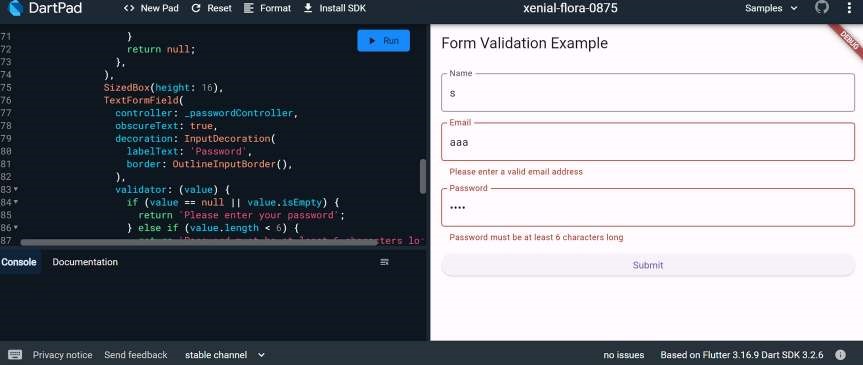
),

);

}

}

**Output:**



**EXPERIMENT NO: 8.**

**8. 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> 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,

),

),

),

),

),

),

);

}

@override void dispose() {

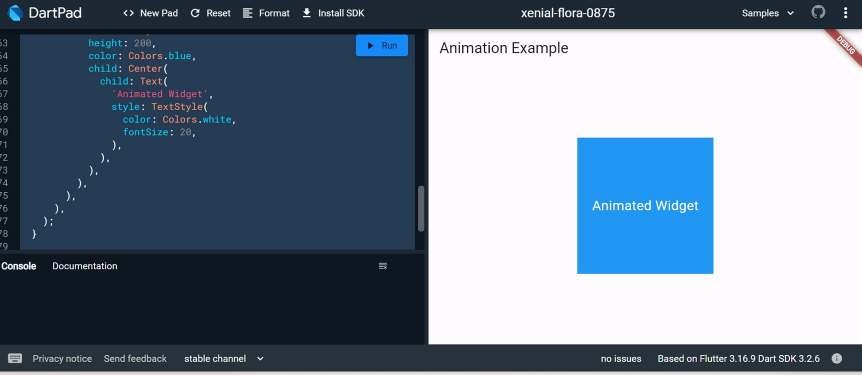
\_animationController.dispose();

super.dispose();

}

}

Output:



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

**Ans)**

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

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) { return MaterialApp( title: 'Fade Animation Example', theme: ThemeData( primarySwatch: Colors.blue,

),

home: FadeAnimationWidget(),

);

}

}

class FadeAnimationWidget extends StatefulWidget {

@override

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

}

class \_FadeAnimationWidgetState extends State<FadeAnimationWidget>

with SingleTickerProviderStateMixin { late AnimationController \_animationController; late Animation<double> \_opacityAnimation;

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

\_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) {

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,

),

),

),

),

),

),

); }

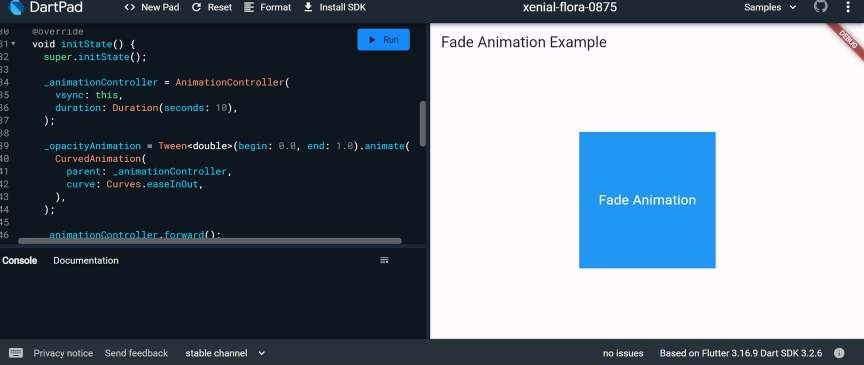
@override void dispose() { \_animationController.dispose(); super.dispose();

}

}

**Output:**

**Fade Animation**



**Slide Animation:**

import 'package:flutter/material.dart';

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

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) { return MaterialApp( title: 'Slide Animation Example', theme: ThemeData( primarySwatch: Colors.blue,

),

home: SlideAnimationWidget(),

);

}

}

class SlideAnimationWidget extends StatefulWidget {

@override

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

}

class \_SlideAnimationWidgetState extends State<SlideAnimationWidget>

with SingleTickerProviderStateMixin { late AnimationController \_animationController; late Animation<Offset> \_slideAnimation;

@override

void initState() { super.initState();

\_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,

),

),

),

),

),

);

}

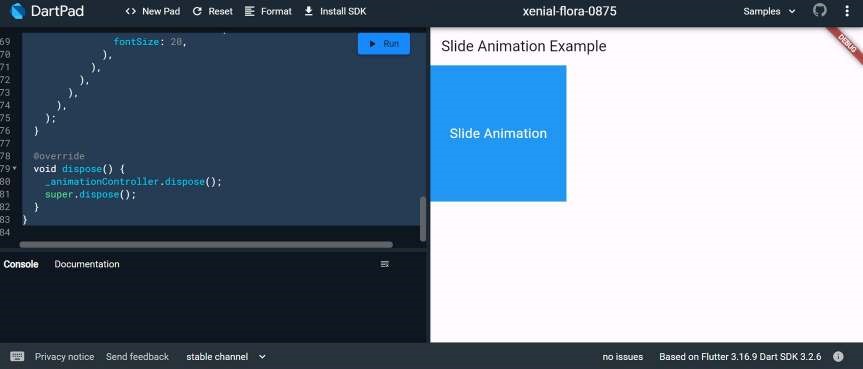
@override void dispose() { \_animationController.dispose(); super.dispose();

}

}

**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(),

);

}

}

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, fontSize: 20,

),

),

),

),

),

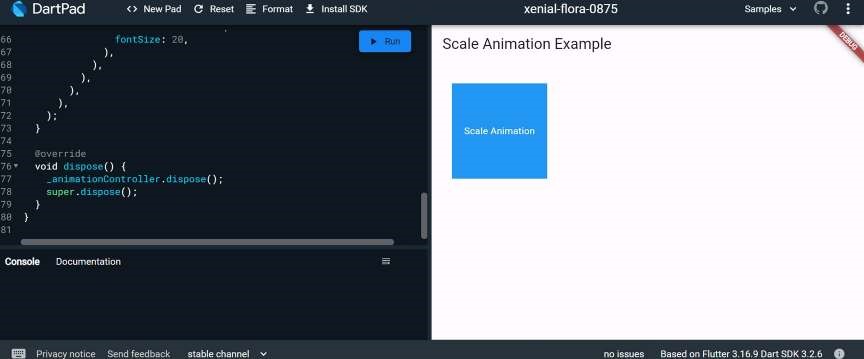
); }

@override void dispose() { \_animationController.dispose(); super.dispose();

}

}

**Output:**



**EXPERIMENT NO: 9.**

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

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

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

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) { return MaterialApp( title: 'API Fetch Example', theme: ThemeData(

primarySwatch: Colors.blue,

),

home: MyApiFetchWidget(),

);

}

}

class MyApiFetchWidget extends StatefulWidget {

@override

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

}

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

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

}

Future<List<Post>> fetchPosts() async {

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

if (response.statusCode == 200) { // If the server returns a 200 OK response, // parse the JSON and return a list of posts.

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

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

} else {

// If the server did not return a 200 OK response, // throw an exception. throw Exception('Failed to load posts');

}

}

@override

Widget build(BuildContext context) {

return Scaffold( appBar: AppBar(

title: Text('API Fetch Example'),

),

body: FutureBuilder<List<Post>>( future: \_posts, builder: (context, snapshot) {

if (snapshot.connectionState == ConnectionState.waiting) {

return CircularProgressIndicator(); } else if (snapshot.hasError) { return Text('Error: ${snapshot.error}');

} else {

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

return ListTile( title: Text(snapshot.data![index].title),

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

);

},

);

}

},

),

);

}

}

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

Post({ required this.userId, required this.id, required this.title, required this.body,

});

factory Post.fromJson(Map<String, dynamic> json) { return Post( userId: json['userId'],

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

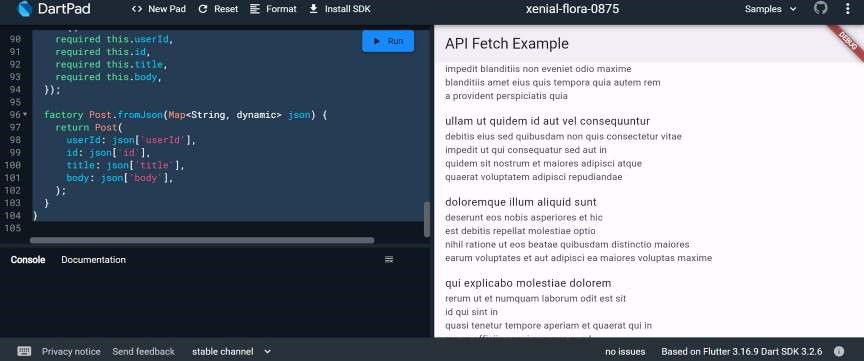
body: json['body'],

);

}

}

**Output:**



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

**Ans)**

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

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

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) { return MaterialApp( title: 'API Fetch Example', theme: ThemeData(

primarySwatch: Colors.blue,

),

home: MyApiFetchWidget(),

);

}

}

class MyApiFetchWidget extends StatefulWidget {

@override

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

}

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

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

}

Future<List<Post>> fetchPosts() async {

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

if (response.statusCode == 200) {

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

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

throw Exception('Failed to load posts');

}

}

@override

Widget build(BuildContext context) {

return Scaffold( appBar: AppBar(

title: Text('API Fetch Example'),

),

body: FutureBuilder<List<Post>>( future: \_posts, builder: (context, snapshot) { if (snapshot.connectionState == ConnectionState.waiting) { return Center(child: CircularProgressIndicator());

} else if (snapshot.hasError) {

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

} else {

return PostList(posts: snapshot.data!);

}

},

),

);

}

}

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

PostList({required this.posts});

@override

Widget build(BuildContext context) { return ListView.builder( itemCount: posts.length, itemBuilder: (context, index) {

return PostItem(post: posts[index]);

},

);

}

}

class PostItem extends StatelessWidget { final Post post;

PostItem({required this.post});

@override

Widget build(BuildContext context) { return Card( margin: EdgeInsets.all(10),

elevation: 3, child: Padding( padding: EdgeInsets.all(15), child: Column(

crossAxisAlignment: CrossAxisAlignment.start, children: [ Text( post.title, style: TextStyle( fontSize: 18,

fontWeight: FontWeight.bold,

),

),

SizedBox(height: 10),

Text( post.body, style: TextStyle(fontSize: 16),

),

],

),

),

);

}

}

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

Post({ required this.userId, required this.id, required this.title, required this.body,

});

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

return Post( userId: json['userId'], id: json['id'], title: json['title'],

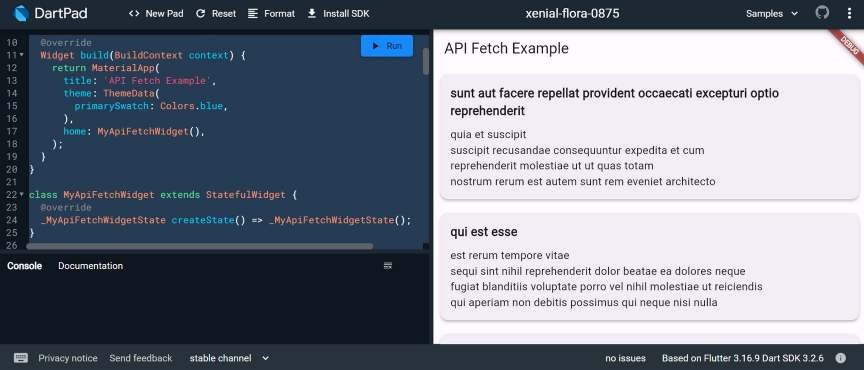
body: json['body'],

);

}

}

**Output:**



**EXPERIMENT NO: 10.**

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

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

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

Add the test or flutter\_test dependency.

Create a test file.

Create a class to test.

Write a test for our class.

Combine multiple tests in a group.

Run the tests.

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

1. Add the test dependency

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

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

content\_copy flutter pub add dev:test

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.

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

content\_copy

counter\_app/

lib/ counter.dart

test/

counter\_test.dart 3. Create a class to test

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

content\_copy class Counter { int value = 0; void increment() => value++;

void decrement() => value--;

}

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

4. Write a test for our class

Inside the counter\_test.dart file, write the first unit test. Tests are defined using the top-level test function, and you can check if the results are correct by using the toplevel expect function. Both of these functions come from the test package.

content\_copy

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

void main() {

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

expect(counter.value, 1);

});

}

5. Combine multiple tests in a group

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

content\_copy

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

void main() {

group('Test start, increment, decrement', () {

test('value should start at 0', () {

expect(Counter().value, 0);

});

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

expect(counter.value, 1);

});

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

expect(counter.value, -1);

});

});

}

6. Run the tests

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

Run tests using IntelliJ or VSCode

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

IntelliJ

Open the counter\_test.dart file

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

VSCode

Open the counter\_test.dart file

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

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

content\_copy

flutter test test/counter\_test.dart

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

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

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

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

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

1. Flutter DevTools:

Run your app with the flutter run command.

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

flutter pub global activate devtools flutter pub global run devtools

Open your app in a Chrome browser and connect it to DevTools by clicking on the "Open DevTools" button in the terminal or by navigating to http://127.0.0.1:9100/. DevTools provides tabs like Inspector, Timeline, Memory, and more.

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.

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.

**TEXT BOOK:**

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

**Reference Books**

1. Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart 2 by Packt Publishing Limited.

2. Beginning App Development with Flutter: Create Cross-Platform Mobile Apps by Rap Payne, 1st edition, Apress.

3. Practical Flutter: Improve your Mobile Development with Google's Latest Open-Source SDK by Frank Zammetti, 1st edition, Apress.