****

**Vidyavardhini’s**

**College of Engineering & Technology**

Vasai Road (W)

**Department of**

**Information Technology**

**Lab Manual**

| Semester | VI | Class | TE |
| --- | --- | --- | --- |
| Coursecode | ITL604 | Academic year | 2022-23 |
| Course Name | **Mobile App Development & Progressive Web App Lab** | | |
| Name of Faculty | Mr. Sainath Patil | | |
| Supporting staff | Mr. Nitin Shingane | | |

Vidyavardhini’s College of Engineering Technology

Department of Information Technology

****

**Vidyavardhini’s**

**College of Engineering & Technology**

**Vision**

To be a premier institution of technical education, aiming at becoming a valuable resource for industry and society.

**Mission**

* To provide technologically inspiring environment for learning.
* To promote creativity, innovation, and professional activities.
* To inculcate ethical and moral values.
* To cater personal, professional, and societal needs through quality education.

Vidyavardhini’s College of Engineering Technology

Department of Information Technology

**Department Vision:**

To foster and maintain excellence by orienting the captivating minds of the aspiring engineers towards IT-driven technological solutions for the benefit of the society.

**Department Mission:**

* To provide quality education, by employing best and diversified teaching practices and tools, and teaching behind the confines of the university syllabus.
* To keep students abreast with latest technological advancements in the market.
* To prepare students to troubleshoot and solve IT system problems.

**Program Education Objectives (PEOs):**

1. To produce skilled IT Professional to cater social/industrial needs.

2. To inculcate an ability to implement modern practices with ethical and professional responsibilities.

3. To establish graduate as Business Analyst, System Analyst, Data Scientist, Project Leader.

**Program Specific Outcomes (PSOs):**

The graduates will be able to

* Apply and implement IT solutions in allied fields of engineering to solve real word problems.
* Identify social and industrial problems, provide creative solutions, and become a quality asset for society and industry.
* Deploy secured solution using Information Technology practices and strategies.

**Program Outcomes (POs):**

Engineering graduates will be able to:

* **PO1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solutions of complex engineering problems.
* **PO2. Problem Analysis:** Identify, formulate, review research literature, and analyse complex engineering complex reaching substantiated conclusions using first principles of mathematics, and engineering sciences.
* **PO3. 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.
* **PO4. 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.
* **PO5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
* **PO6.** **The engineer and society:** Apply reasoning informed by the contextual knowledge to access societal, health, safety, legal and cultural issues, and the consequent responsibilities relevant to the professional engineering practice.
* **PO7.** **Environment and sustainability:** Understand the impact of the professional engineering solutions in social and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
* **PO8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
* **PO9. Individual and teamwork:** Function effectively as an individual, and as a member or leader in diverse terms, and in multidisciplinary settings.
* **PO10. 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.
* **PO11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects in a multidisciplinary environment.
* **PO12. Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Vidyavardhini’s College of Engineering Technology

Department of Information Technology

**Course Objectives**

| 1 | Learn the basics of the Flutter framework. |
| --- | --- |
| 2 | Develop the App UI by incorporating widgets, layouts, gestures and animation. |
| 3 | Create a production ready Flutter App by including files and firebase backend service. |
| 4 | Learn the Essential technologies, and Concepts of PWAs to get started as quickly and efficiently as possible. |
| 5 | Develop responsive web applications by combining AJAX development techniques with the jQuery JavaScript library. |
| 6 | Understand how service workers operate and also learn to Test and Deploy PWA. |

**Course Outcomes**

| At the end of the course student will be able to: | | Bloom’s Level |
| --- | --- | --- |
| ITL604.1 | Develop cross platform mobile application using Flutter Framework. | Create |
| ITL604.2 | Design and develop interactive Flutter App by using widgets, layouts, and gestures and animation. | Create |
| ITL604.3 | Analyse and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS. | Create |
| ITL604.4 | Identify various PWA frameworks and their requirements. | Apply |
| ITL604.5 | Design and develop a responsive User Interface by applying PWA Design Techniques. | Create |
| ITL604.6 | Develop and Analyse PWA features and deploy it over app hosting solutions. | Create |

Vidyavardhini’s College of Engineering Technology

Department of Information Technology

**Mapping of Experiments with Course Outcomes**

| Exp. No. | Experiment | Course Outcomes | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ITL  604.1 | ITL  604.2 | | ITL  604.3 | ITL  604.4 | | ITL  604.5 | ITL  604.6 | |
| 1**.** | To install and configure Flutter Environment. | 3 |  | |  |  |  | |  | |
| 2. | To develop Mobile App using Flutter. | 3 |  | |  |  |  | |  | |
| 3. | To design Flutter UI by including common widgets. |  | | 2 |  |  |  | | |  |
| 4. | To create an interactive Form using form widget. |  | | 2 |  |  |  | | |  |
| 5. | To design a layout of Flutter App using layout widgets. |  | | 2 |  |  |  | | |  |
| 6. | To include icons, images, charts in Flutter App. |  | | 2 |  |  |  | | |  |
| 7. | To apply navigation, routing and gestures in Flutter App. |  | |  | 2 |  |  | | |  |
| 8. | To Connect Flutter UI with firebase database. |  | |  | 2 |  |  | | |  |
| 9. | To test and deploy production ready Flutter App on Android platform. |  | |  | 2 |  |  | | |  |
| 10. | To create a responsive User Interface using jQuery Mobile/Material UI/Angular UI/React UI for Ecommerce application. |  | |  |  | 3 |  | | |  |
| 11. | To code and register a service worker and complete the install and activation process for a new service worker for the E-commerce |  | |  |  |  | 3 | | |  |
| 12. | To deploy an Ecommerce PWA using SSL enabled static hosting solution. |  | |  |  |  |  | | | 3 |
| 13. | Mini Project |  | |  |  |  |  | | |  |

Class: TE IT Sub: MAD - PWA Lab

Shape

Description automatically generated with medium confidence

**List of Experiments**

| **Sr. No.** | **Title of Experiment** | **Date** | **Grade** | **Sign** |
| --- | --- | --- | --- | --- |
| 1 | To install and configure Flutter Environment |  |  |  |
| 2 | To develop Simple Hello World Mobile App using flutter. |  |  |  |
| 3 | To design Flutter UI by including common widgets. |  |  |  |
| 4 | To create an interactive Form using form widget. |  |  |  |
| 5 | To design a layout of Flutter App using layout widgets. |  |  |  |
| 6 | To include icons, images, charts in Flutter app. |  |  |  |
| 7 | To apply navigation, routing and gestures in Flutter App. |  |  |  |
| 8 | To Connect Flutter UI with fireBase database. |  |  |  |
| 9 | To test and deploy production ready Flutter App on Android platform. |  |  |  |
| 10 | To create a responsive User Interface using jQuery Mobile/ Material UI/ Angular UI/ React UI for Ecommerce application. |  |  |  |
| 11 | To code and register a service worker and complete the install and activation process for a new service worker for the E-commerce PWA. |  |  |  |
| 12 | To deploy an Ecommerce PWA using SSL enabled static hosting solution. |  |  |  |
| 13 | Mini project (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) |  |  |  |
|  |  |  |  |  |
|  | Assignment 1 |  |  |  |
|  | Assignment 2 |  |  |  |
|  | Assignment 3 |  |  |  |
|  | Assignment 4 |  |  |  |
|  | Assignment 5 |  |  |  |
|  | Assignment 6 |  |  |  |

**Experiment No. 1**

**Aim:** To install and configure Flutter Environment.

**Theory:** Flutter is an open-source software development kit which enables smooth and easy cross-platform mobile app development. You can build high quality natively compiled apps for iOS and Android quickly, without having to write the code for the two apps separately. All you need is one codebase for both platforms. Flutter is the only framework with a mobile SDK that provides a responsive style without using a JavaScript bridge, thereby reaching a level of performance that rivals its cousin and direct competitor React Native. It easily integrates with the different platforms such as Android, IOS and Linux, MAC, Windows and Google Fuchsia applications.

To install and run Flutter, your development environment must meet these minimum requirements:

● Operating Systems: Windows 7 SP1 or later (64-bit), x86-64 based.

● Disk Space: 1.64 GB (does not include disk space for IDE/tools).

● Tools: Flutter depends on these tools being available in your environment.

**Procedure:**

1. Download the following installation bundle to get the latest stable release of the Flutter SDK: <https://docs.flutter.dev/get-started/install/windows>
2. Extract the zip file and place the contained flutter in the desired installation location for the Flutter SDK (for example, C:\src\flutter).
3. Update your path

If you wish to run Flutter commands in the regular Windows console, take these steps to add Flutter to the PATH environment variable:

* From the Start search bar, enter ‘env’ and select Edit environment variables for your account.
* Under User variables check if there is an entry called Path:
* If the entry exists, append the full path to flutter\bin using ; as a separator from existing values.
* If the entry doesn’t exist, create a new user variable named Path with the full path to flutter\bin as its value.

*You have to close and reopen any existing console windows for these changes to take effect.*

1. Run flutter doctor using the following command:

* flutter doctor

**Experiment No. 2**

**Aim:** To develop Simple Hello World Mobile App using flutter.

**Theory:**

* **Scaffold**: The Scaffold is designed to be a top-level container for a MaterialApp. This means that adding a Scaffold to each route on a Material app will provide the app with Material's basic visual layout structure.
* **Text**: The Text widget displays a string of text with single style. The string might break across multiple lines or might all be displayed on the same line depending on the layout constraints.
* **Stateless**: A stateless widget is a widget that describes part of the user interface by building a constellation of other widgets that describe the user interface more concretely.
* **Build**: The build method of a stateless widget is typically only called in three situations: the first time the widget is inserted in the tree, when the widget's parent changes its configuration, and when an InheritedWidget it depends on changes.
* **runApp()**: The runApp() function takes the given Widget and makes it the root of the widget tree.
* **Child**: A child widget can itself be a Row, Column, or other complex widget. You can specify how a Row or Column aligns its children, both vertically and horizontally. You can stretch or constrain specific child widgets. You can specify how child widgets use the Row 's or Column 's available space.

**Code:**

*filename - main.dart*

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

**void main() {**

**runApp(const MyApp());**

**}**

**class MyApp extends StatelessWidget {**

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

**@override**

**Widget build(BuildContext context) {**

**return MaterialApp(**

**title: 'Welcome to Flutter',**

**home: Scaffold(**

**appBar: AppBar(**

**title: const Text('Welcome to Flutter'),**

**centerTitle: true,**

**backgroundColor: Colors.*black*,**

**),**

**backgroundColor: Colors.*black54*,**

**body: const Center(**

**child: Text(**

**'Hello World, this is Anish',**

**style: TextStyle(**

**fontWeight: FontWeight.*bold*,**

**fontSize: 30**

**)**

**),**

**),**

**),**

**);**

**}**

**}**

**Output:**

****

**Experiment No. 3**

**Aim:** To design Flutter UI by including common widgets.

**Theory:**

* **Container**: The container in Flutter is a parent widget that can contain multiple child widgets and manage them efficiently through width, height, padding, background color, etc. It is a widget that combines common painting, positioning, and sizing of the child widgets.
* **Child**: A child widget can itself be a Row, Column, or other complex widget. You can specify how a Row or Column aligns its children, both vertically and horizontally. You can stretch or constrain specific child widgets. You can specify how child widgets use the Row 's or Column 's available space.
* **Row**: A widget that displays its children in a horizontal array. he Row widget does not scroll (and in general it is considered an error to have more children in a Row than will fit in the available room). If you have a line of widgets and want them to be able to scroll if there is insufficient room, consider using a ListView.
* **Column**: A widget that displays its children in a vertical array. The Column widget does not scroll (and in general it is considered an error to have more children in a Column than will fit in the available room). If you have a line of widgets and want them to be able to scroll if there is insufficient room, consider using a ListView.

**Code:**

**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(**

**debugShowCheckedModeBanner: false,**

**home: Scaffold(**

**appBar: AppBar(**

**title: const Text('Experiment No.3'),**

**centerTitle: true,**

**backgroundColor: Colors.*black*,**

**),**

**body: Column(**

**children: [**

**Row(**

**mainAxisAlignment: MainAxisAlignment.spaceAround,**

**crossAxisAlignment: CrossAxisAlignment.start,**

**children: [**

**Container(**

**padding: const EdgeInsets.all(20.0),**

**margin: const EdgeInsets.all(20.0),**

**color: Colors.*blue*,**

**child: const Text('TE'),**

**),**

**Container(**

**padding: const EdgeInsets.all(20.0),**

**margin: const EdgeInsets.all(20.0),**

**color: Colors.*green*,**

**child: const Text('IT'),**

**),**

**Container(**

**padding: const EdgeInsets.all(20.0),**

**margin: const EdgeInsets.all(20.0),**

**color: Colors.*grey*,**

**child: Container(**

**padding: const EdgeInsets.all(5.0),**

**color: Colors.*white*,**

**child: const Text('Anish'),**

**),**

**),**

**Container(**

**padding: const EdgeInsets.all(20.0),**

**margin: const EdgeInsets.all(20.0),**

**color: Colors.*blue*[100],**

**child: const Text('Dalvi'),**

**)**

**],**

**),**

**Row(**

**mainAxisAlignment: MainAxisAlignment.spaceAround,**

**crossAxisAlignment: CrossAxisAlignment.start,**

**children: [**

**Container(**

**padding: const EdgeInsets.all(20.0),**

**margin: const EdgeInsets.all(20.0),**

**color: Colors.*yellow*,**

**child: const Text('MAD'),**

**),**

**Container(**

**padding: const EdgeInsets.all(20.0),**

**margin: const EdgeInsets.all(20.0),**

**color: Colors.*pinkAccent*,**

**child: const Text('Experiments'),**

**),**

**Container(**

**padding: const EdgeInsets.all(20.0),**

**margin: const EdgeInsets.all(20.0),**

**color: Colors.*blue*,**

**child: Container(**

**padding: const EdgeInsets.all(5.0),**

**color: Colors.*white*,**

**child: const Text('Assignments'),**

**),**

**)**

**],**

**),**

**Row(**

**mainAxisAlignment: MainAxisAlignment.spaceAround,**

**children: const [**

**Text('This'),**

**Text('is'),**

**Text('Flutter')**

**],**

**),**

**const SizedBox(**

**height: 20,**

**),**

**Row(**

**mainAxisAlignment: MainAxisAlignment.start,**

**children: [**

**Expanded(**

**flex: 1,**

**child: Container(**

**padding: const EdgeInsets.all(20.0),**

**color: Colors.*amber*,**

**child: const Text('Flutter'),**

**),**

**),**

**Expanded(**

**flex: 2,**

**child: Container(**

**padding: const EdgeInsets.all(20.0),**

**color: Colors.*redAccent*,**

**child: const Text('in'),**

**),**

**),**

**Expanded(**

**flex: 3,**

**child: Container(**

**padding: const EdgeInsets.all(20.0),**

**color: Colors.*purple*,**

**child: const Text('Android Studio'),**

**)**

**)**

**],**

**)**

**],**

**)**

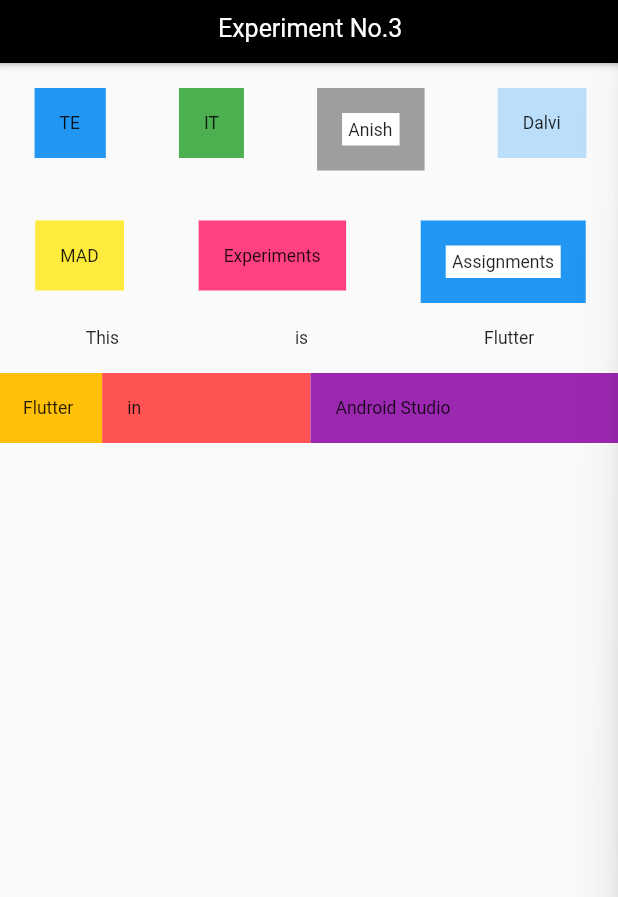
**),**

**);**

**}**

**}**

**Output:**

****

**Experiment No. 4**

**Aim:** To create an interactive Form using form widget.

**Theory:**

Forms are now a critical part of any mobile or web application. Forms are used to gather information from the user. According to your business niche, requirements, and logic, you can create a form for the user, including user authentication, searching, ordering, filtering, booking, etc. The form can have text fields, checkboxes, and radio buttons.

A form widget is provided by the flutter to create forms. This form widget acts as a container, which allows us to group multiple form fields. To create a form, we have to provide a GlobalKey to uniquely identify the form, which will enable us to validate form fields.

To allow users to enter the text fields, the flutter form widget uses a child widget called TextFormField. This child widget is used to render a material design text field. TextFormField also allows us to display validation errors when they occur.

**Code:**

**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) {**

**const appTitle = 'Interactive Flutter Form';**

**return MaterialApp(**

**debugShowCheckedModeBanner: false,**

**title: appTitle,**

**home: Scaffold(**

**appBar: AppBar(**

**title: const Text(appTitle),**

**backgroundColor: Colors.*black*,**

**),**

**body: const Padding(**

**padding: EdgeInsets.all(15.0),**

**child: MyCustomForm(),**

**),**

**),**

**);**

**}**

**}**

**// Create a Form widget.**

**class MyCustomForm extends StatefulWidget {**

**const MyCustomForm({Key? key}) : super(key: key);**

**@override**

**MyCustomFormState createState() {**

**return MyCustomFormState();**

**}**

**}**

**// This class holds data related to the form.**

**class MyCustomFormState extends State<MyCustomForm> {**

**final \_formKey = GlobalKey<FormState>();**

**@override**

**Widget build(BuildContext context) {**

**// Build a Form widget using the \_formKey created above.**

**return Form(**

**key: \_formKey,**

**child: Column(**

**crossAxisAlignment: CrossAxisAlignment.start,**

**children: [**

**TextFormField(**

**decoration: const InputDecoration(**

**border: UnderlineInputBorder(),**

**labelText: 'Name: ',**

**),**

**validator: (value) {**

**if (value == null || value.isEmpty) {**

**return 'Please enter your name';**

**}**

**return null;**

**},**

**),**

**TextFormField(**

**decoration: const InputDecoration(**

**border: UnderlineInputBorder(),**

**labelText: 'Username: ',**

**),**

**validator: (value) {**

**if (value == null || value.isEmpty) {**

**return 'Please enter your username';**

**}**

**return null;**

**},**

**),**

**TextFormField(**

**obscureText: true,**

**decoration: const InputDecoration(**

**border: UnderlineInputBorder(),**

**labelText: 'Password: ',**

**),**

**validator: (value) {**

**if (value == null || value.isEmpty) {**

**return 'Please enter your password';**

**}**

**return null;**

**},**

**),**

**Padding(**

**padding: const EdgeInsets.symmetric(vertical: 16.0),**

**child: ElevatedButton(**

**onPressed: () {**

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

**ScaffoldMessenger.*of*(context).showSnackBar(**

**SnackBar(**

**content: Container(**

**margin: const EdgeInsets.fromLTRB(0, 0, 0, 5),**

**child: const Padding(**

**padding: EdgeInsets.all(20.0),**

**child: Text('Congratulations! You have registered successfully.!'),**

**),**

**),**

**),**

**);**

**}**

**},**

**child: const Text('Submit'),**

**),**

**),**

**],**

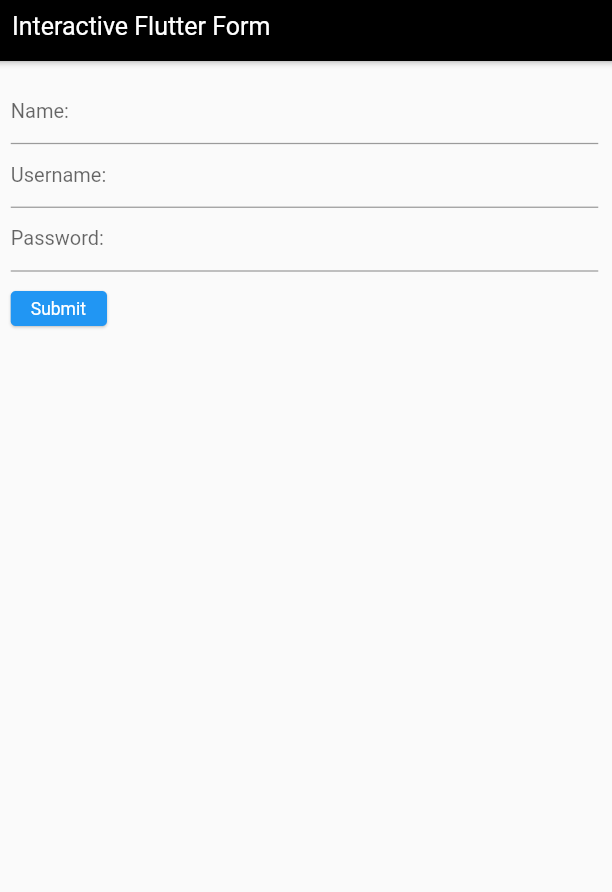
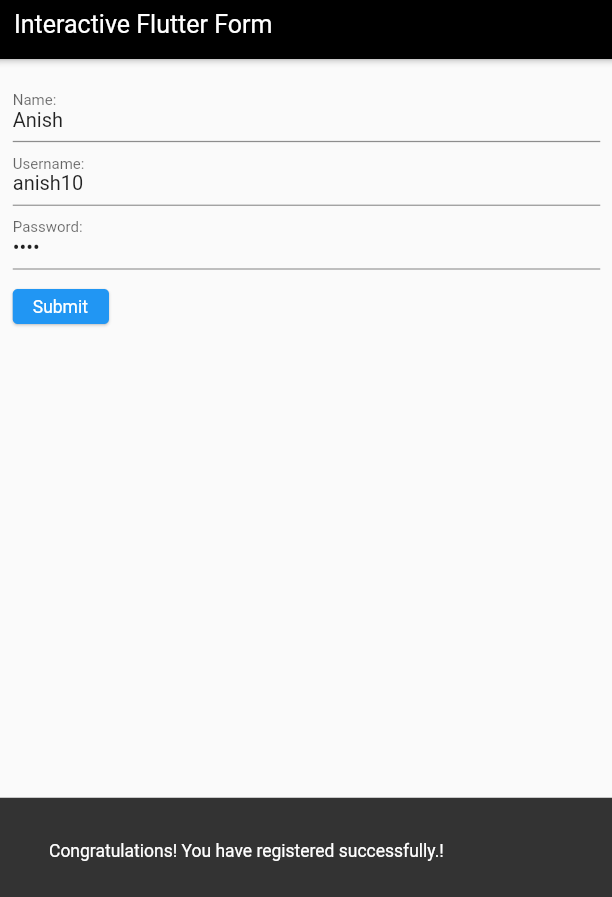
**),**

**);**

**}**

**}**

**Output:**

** **

**Experiment No. 5**

**Aim:** To design a layout of Flutter App using layout widgets.

**Theory:**

In Flutter, almost everything is a widget—even layout models are widgets. The images, icons, and text that you see in a Flutter app are all widgets. But things you don't see are also widgets, such as the rows, columns, and grids that arrange, constrain, and align the visible widgets.

Layout in Flutter defines how the content expands in a given area as the application is used on different areas like the web, mobile devices having different screen sizes, so the content which changes dynamically should be displayed properly to the user.

Code:

**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) {**

**Widget titleSection = Container(**

**padding: const EdgeInsets.all(32),**

**child: Row(**

**children: [**

**Expanded(**

**/\*1\*/**

**child: Column(**

**crossAxisAlignment: CrossAxisAlignment.start,**

**children: [**

**/\*2\*/**

**Container(**

**padding: const EdgeInsets.only(bottom: 8),**

**child: const Text(**

**'Borivali',**

**style: TextStyle(**

**fontWeight: FontWeight.*bold*,**

**),**

**),**

**),**

**Text(**

**'Mumbai, India',**

**style: TextStyle(**

**color: Colors.*grey*[500],**

**),**

**),**

**],**

**),**

**),**

**/\*3\*/**

**Icon(**

**Icons.*star*,**

**color: Colors.*red*[500],**

**),**

**const Text('95'),**

**],**

**),**

**);**

**Color color = Theme.*of*(context).primaryColor;**

**Widget buttonSection = Row(**

**mainAxisAlignment: MainAxisAlignment.spaceEvenly,**

**children: [**

**\_buildButtonColumn(color, Icons.*call*, 'CALL'),**

**\_buildButtonColumn(color, Icons.*near\_me*, 'ROUTE'),**

**\_buildButtonColumn(color, Icons.*share*, 'SHARE'),**

**],**

**);**

**Widget textSection = const Padding(**

**padding: EdgeInsets.all(32),**

**child: Text(**

**'Borivali is a well-established residential locality situated in the northern part of Mumbai that majorly caters to the upper-mid segment. The locality is located along the Western line of the Mumbai Suburban Railway and divided into two suburbs Borivali East and Borivali West by the railway line. The locality is majorly dominated by the multi-storey apartments spread across various completed and under-construction projects. Also, there are several co-operative housing societies situated here. Key developers active in the area are The Wadhwa Group, Rustomjee, Bhoomi Reality and Fortune Group. \n \nBorivali shares widespread connectivity through the SV Road, Link Road and Western Express Highway. Also, the area is directly connected through Borivali Railway Station placed on the western line of the Mumbai suburban railway network. Besides, there is sufficient availability of BEST (Brihanmumbai Electricity Supply and Transport) buses. Moreover, the under-construction metro Line 2 (Dahisar-Mandale) is expected to boost connectivity of the Borivali.',**

**softWrap: true,**

**),**

**);**

**return MaterialApp(**

**debugShowCheckedModeBanner: false,**

**title: 'Layout Demo',**

**home: Scaffold(**

**appBar: AppBar(**

**title: const Text('Layout Demo'),**

**centerTitle: true,**

**),**

**body: ListView(**

**children: [**

**titleSection,**

**buttonSection,**

**textSection,**

**],**

**),**

**),**

**);**

**}**

**Column \_buildButtonColumn(Color color, IconData icon, String label) {**

**return Column(**

**mainAxisSize: MainAxisSize.min,**

**mainAxisAlignment: MainAxisAlignment.center,**

**children: [**

**Icon(icon, color: color),**

**Container(**

**margin: const EdgeInsets.only(top: 8),**

**child: Text(**

**label,**

**style: TextStyle(**

**fontSize: 12,**

**fontWeight: FontWeight.*w400*,**

**color: color,**

**),**

**),**

**),**

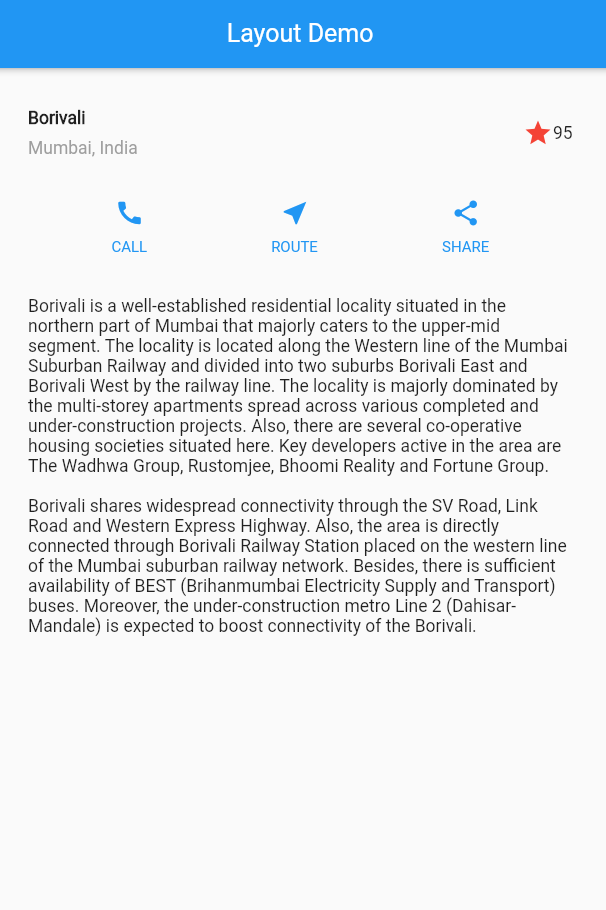
**],**

**);**

**}**

**}**

**Output:**

****

**Experiment No. 6**

**Aim:** To include icons, images, charts in Flutter app.

**Theory:**

1. **Icons**: An icon is a graphic image representing an application or any specific entity containing meaning for the user. It can be selectable and non-selectable. For example, the company's logo is non-selectable. Sometimes it also contains a hyperlink to go to another page. It also acts as a sign in place of a detailed explanation of the actual entity. Flutter provides an Icon Widget to create icons in our applications. We can create icons in Flutter, either using inbuilt icons or with the custom icons. Flutter provides the list of all icons in the Icons class. In this article, we are going to learn how to use Flutter icons in the application
2. **Images**: Displaying images is the fundamental concept of most of the mobile apps. Flutter has an Image widget that allows displaying different types of images in the mobile application.
3. **Charts**: A chart is a graphical representation of data where data is represented by a symbol such as a line, bar, pie, etc. In Flutter, the chart behaves the same as a normal chart. We use a chart in Flutter to represent the data in a graphical way that allows the user to understand them in a simple manner. We can also plot a graph to represents the rise and fall of our values.

**Code & Output**:

***1. Icons***

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

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

**class MyApp extends StatelessWidget {**

**@override**

**Widget build(BuildContext context) {**

**return MaterialApp(**

**debugShowCheckedModeBanner: false,**

**theme: ThemeData(**

**primarySwatch: Colors.*grey*,**

**),**

**home: const MyIconPage(),**

**);**

**}**

**}**

**class MyIconPage extends StatefulWidget {**

**const MyIconPage({super.key});**

**@override**

**\_MyIconPageState createState() => \_MyIconPageState();**

**}**

**class \_MyIconPageState extends State<MyIconPage> {**

**@override**

**Widget build(BuildContext context) {**

**return Scaffold(**

**appBar: AppBar(**

**title: const Text('Flutter Icon Tutorial'),**

**),**

**body: Column(children: <Widget>[**

**//icon with label below it**

**Container(**

**padding: const EdgeInsets.all(30),**

**child: Row(**

**mainAxisAlignment: MainAxisAlignment.spaceAround,**

**children: <Widget>[**

**Column(children: const <Widget>[**

**Icon(**

**Icons.*camera\_front*,**

**size: 70**

**),**

**Text('Front Camera'),**

**]),**

**Column(children: const <Widget>[**

**Icon(**

**Icons.*camera\_enhance*,**

**size: 70**

**),**

**Text('Camera'),**

**]),**

**Column(children: const <Widget>[**

**Icon(**

**Icons.*camera\_rear*,**

**size: 70**

**),**

**Text('Rear Camera'),**

**]),**

**]**

**),**

**)**

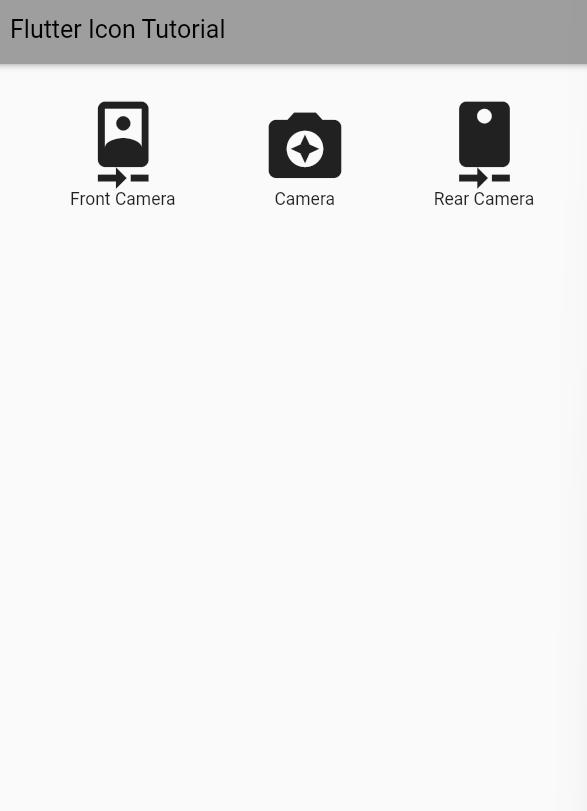
**],**

**)**

**);**

**}**

**}**

** **

***2. Images***

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

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

**class MyApp extends StatelessWidget {**

**@override**

**Widget build(BuildContext context) {**

**return MaterialApp(**

**debugShowCheckedModeBanner: false,**

**title: 'Flutter Image Demo',**

**home: Scaffold(**

**appBar: AppBar(**

**title: const Text('Flutter Image Demo'),**

**),**

**body: Container(**

**color: Colors.*grey*[200],**

**alignment: Alignment.*center*,**

**child: Image.asset('images/test.png')**

**),**

**),**

**);**

**}**

**}**

***3. Charts***

*Add charts\_flutter: ^0.12.0 in pubspec.yaml*

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

**import 'package:charts\_flutter/flutter.dart' as charts;**

**void main() {**

**runApp(MyApp());**

**}**

**class MyApp extends StatelessWidget {**

**@override**

**Widget build(BuildContext context) {**

**return MaterialApp(**

**debugShowCheckedModeBanner: false,**

**title: 'CHARTS',**

**home: ChartsScreen(),**

**);**

**}**

**}**

**class ChartsScreen extends StatefulWidget {**

**@override**

**\_ChartsScreenState createState() => \_ChartsScreenState();**

**}**

**class \_ChartsScreenState extends State<ChartsScreen> {**

**late List<charts.Series<SalesData, String>> \_seriesData;**

**@override**

**void initState() {**

**super.initState();**

**\_generateData();**

**}**

**void \_generateData() {**

**var data = [**

**SalesData('2016', 100),**

**SalesData('2017', 75),**

**SalesData('2018', 200),**

**SalesData('2019', 150),**

**];**

**\_seriesData = [**

**charts.Series(**

**id: 'Sales',**

**data: data,**

**domainFn: (SalesData sales, \_) => sales.year,**

**measureFn: (SalesData sales, \_) => sales.sales,**

**colorFn: (\_, \_\_) => charts.MaterialPalette.*blue*.shadeDefault,**

**),**

**];**

**}**

**@override**

**Widget build(BuildContext context) {**

**return Scaffold(**

**appBar: AppBar(**

**title: Text('Anish Dalvi TE-IT-10'),**

**),**

**body: Center(**

**child: charts.BarChart(**

**\_seriesData,**

**animate: true,**

**vertical: false,**

**barRendererDecorator: charts.BarLabelDecorator<String>(),**

**),**

**),**

**);**

**}**

**}**

**class SalesData {**

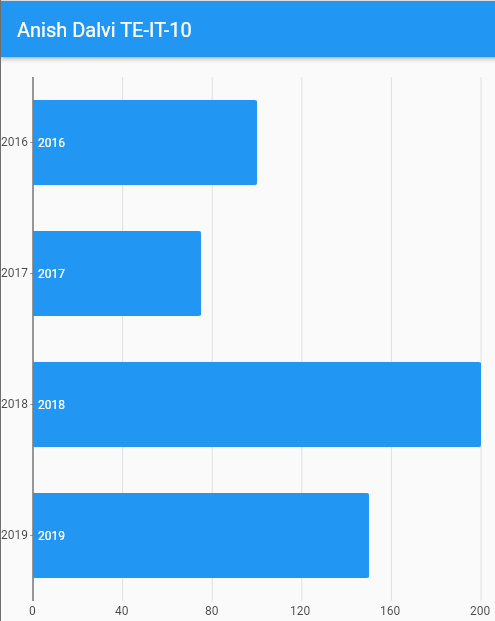
**final String year;**

**final int sales;**

**SalesData(this.year, this.sales);**

**}**

Output:



**Experiment No. 7**

**Aim:** To apply navigation, routing and gestures in Flutter App.

**Theory:**

**Flutter Navigation and Routing**

Navigation and routing are some of the core concepts of all mobile applications, which allows the user to move between different pages. We know that every mobile application contains several screens for displaying different types of information.

In Flutter, screens and pages are known as **routes**, and these routes are just a widget. In Android, a route is similar to an **Activity**, whereas in iOS, it is equivalent to **ViewController.**

In any mobile app, navigating to different page defines the workflow of application and way to handle the navigation is called **routing**. Flutter provides a basic routing class **MaterialPageRoute** and 2 methods **Navigator.push()** and **Navigator.pop()** that shows how to navigate between routes. The Steps to start navigating are:

**Step1**: Create 2 routes in Java.

**Step2**: Navigate to one route from another route by using Navigator.push() method.

**Step3**: Finally, navigate to first route by using Navigator.pop() method.

Create two routes

Here, we are going to create 2 routes for navigation. When we tap the button on first page, it will navigate to second page. Again, when we tap button on the second page, it will return to first page.

Navigate to second route using Navigator.push() method

The Navigator.push() method is used to navigate to a new route. Hence the push() method adds a route on the stack and then manage it by using Navigator, Again we use MaterialPageRoute class that allows transition between routes using a platform-specific animation.

Return to first route using Navigator.pop() method

Now, we need to use Navigator.pop() method to close second route and return the route. The pop() method allows us to remove the current route from stack, manages by the Navigator.

To implement a return to original route, we need to update the **onPressed()** call method in SecondRoute widget.

**Code:**

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

**void main() {**

**runApp(MaterialApp(**

**debugShowCheckedModeBanner: false,**

**title: 'Navigation and Routing',**

**theme: ThemeData(**

**// This is the theme of your application.**

**primarySwatch: Colors.*green*,**

**),**

**home: FirstRoute(),**

**));**

**}**

**class FirstRoute extends StatelessWidget {**

**@override**

**Widget build(BuildContext context) {**

**return Scaffold(**

**appBar: AppBar(**

**title: const Text('Anish Dalvi TE-IT-10 (First Screen)'),**

**),**

**body: Center(**

**child: ElevatedButton(**

**onPressed: () {**

**Navigator.*push*(**

**context,**

**MaterialPageRoute(builder: (context) => SecondRoute()),**

**);**

**},**

**child: Text('Click Here'),**

**),**

**),**

**);**

**}**

**}**

**class SecondRoute extends StatelessWidget {**

**@override**

**Widget build(BuildContext context) {**

**return Scaffold(**

**appBar: AppBar(**

**title: const Text("Anish Dalvi TE-IT-10 (Second Screen)"),**

**),**

**body: Center(**

**child: ElevatedButton(**

**onPressed: () {**

**Navigator.*pop*(context);**

**},**

**child: Text('Go back'),**

**),**

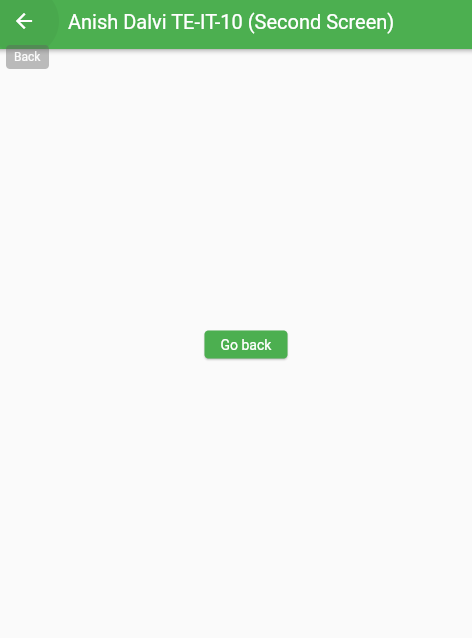
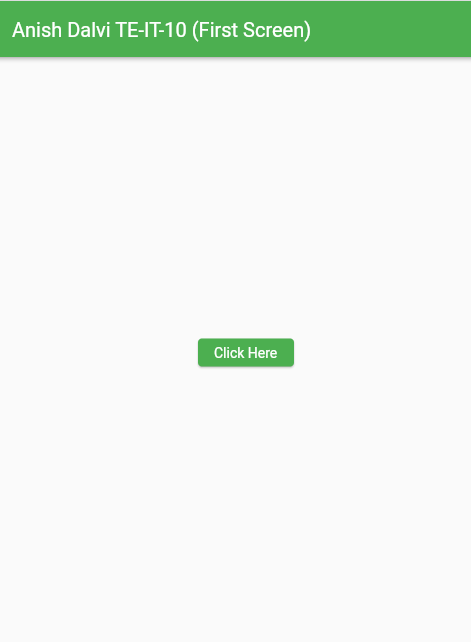
**),**

**);**

**}**

**}**

**Output:**

****

**Experiment No. 8**

**Aim:** To connect flutter UI with firebase database.

**Theory:**

Launch Android Studio and create a new Flutter project.

Open pubspec.yaml file and add the following Flutter dependencies.

dev\_dependencies:

flutter\_test:

sdk: flutter

firebase\_database: ^4.4.0

firebase\_core: ^0.5.3

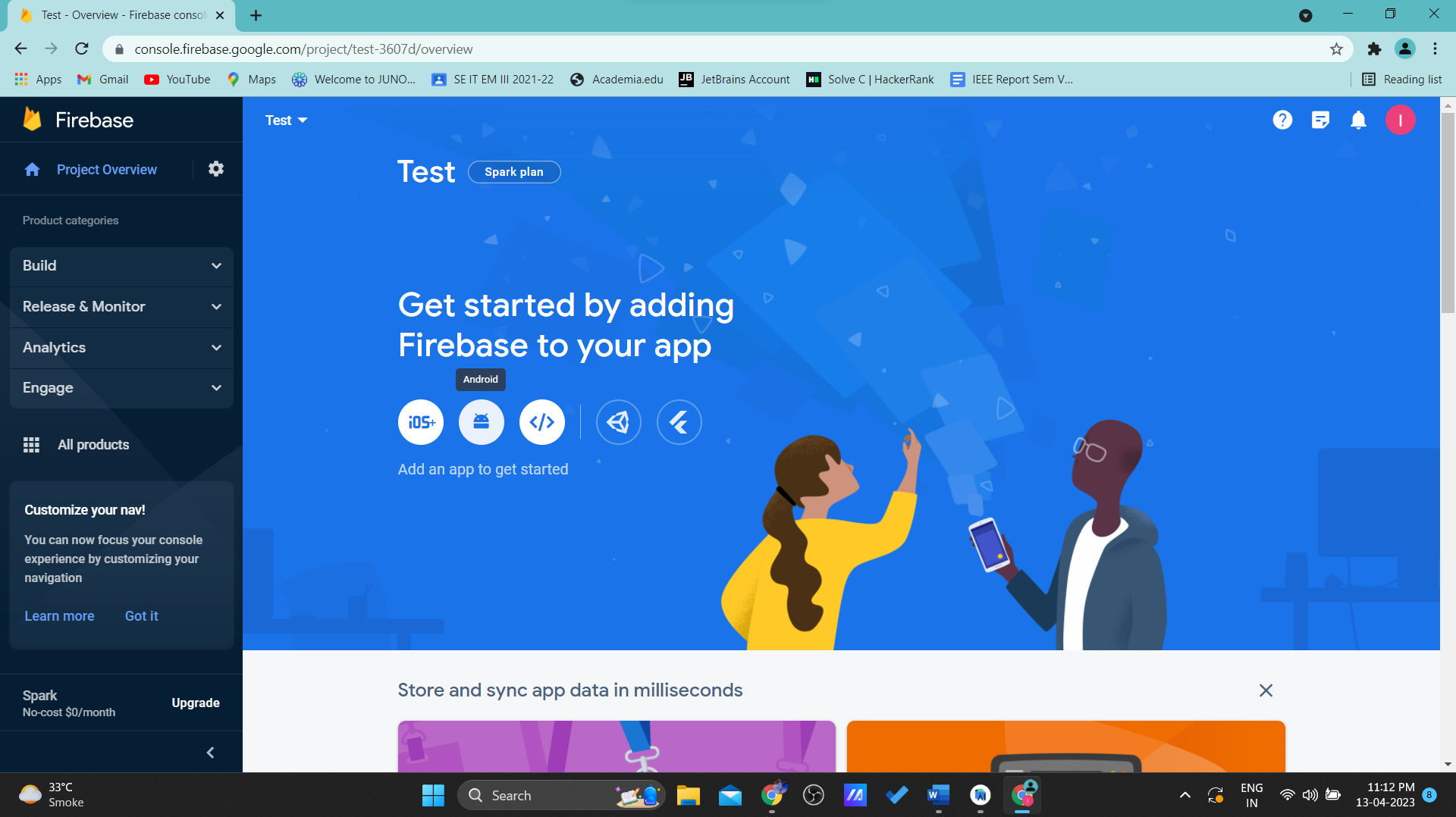
Use the pub get command to retrieve the appropriate dependencies.

Next, open your browser and Naviagate to Firebase’s website. We need to create a firebase project, as shown below.

Graphical user interface

Description automatically generated

Add your Flutter application to Firebase by clicking on Android icon.



Add your application’s package name as shown.

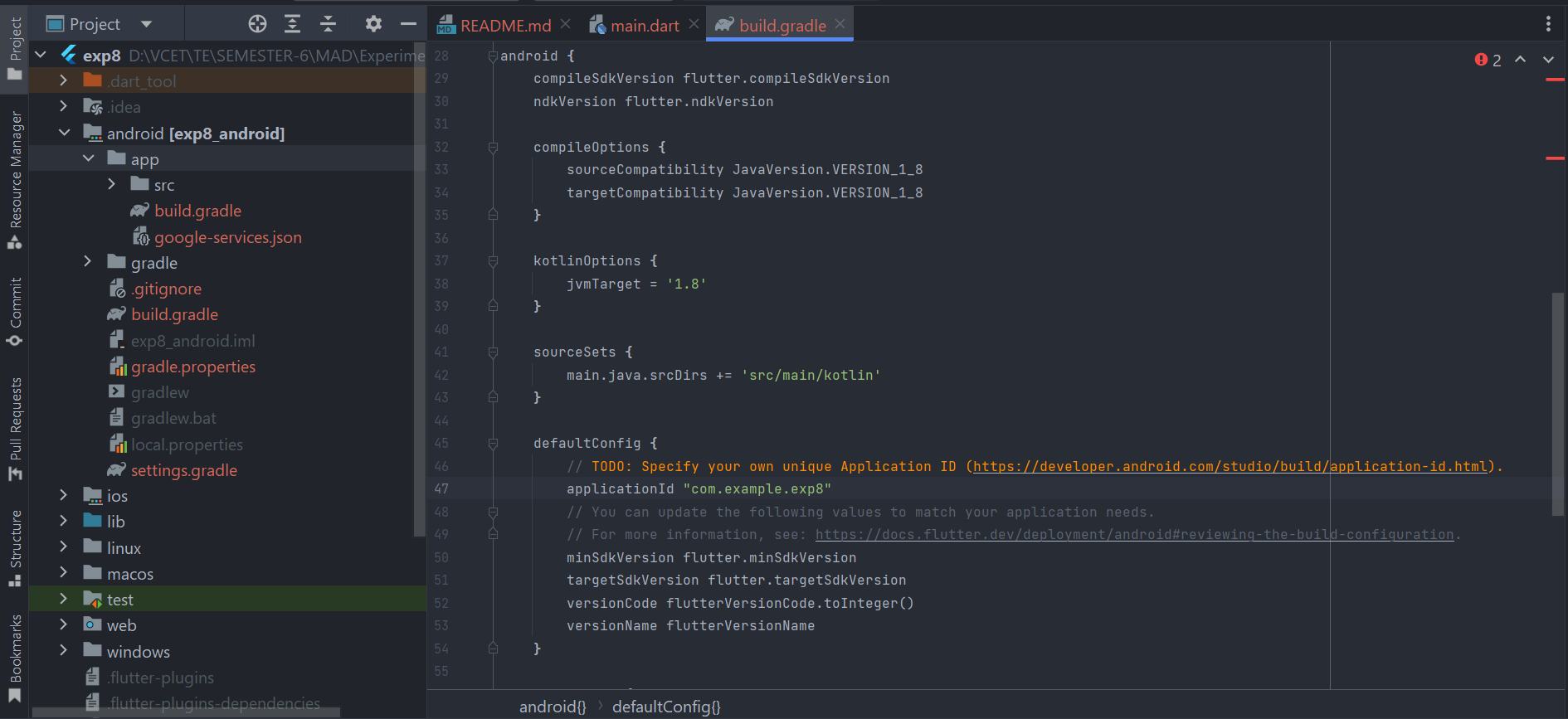
Timeline

Description automatically generated

You can find your package name in app-level build.gradle file. Click register app after completing this step.

Graphical user interface, application

Description automatically generatedDownload the google-services.json file and paste it into android/app folder.



Paste classpath ‘com.google.gms:google-services:4.3.3” in project level build.gradle file.

Dependencies {

classpath ‘com.android.tools.build:gradle:3.5.0’

classpath ‘org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin\_version’

classpath ‘com.google.gms:google-services:4.3.3’

}

Navigate to app-level build.gradle file and add apply plugin: ‘com.google.gms.google-services’ as shown

apply plugin: ‘com.android.application’

apply plugin: ‘kotlin-android’

apply from: ‘$flutterRoot/packages/flutter\_tools/gradle/flutter,gradle’

apply plugin: ‘com.google.gms.google-services’

After project is created successfully, navigate back to Firebase console’s realtime database section and create a new database.

In rules section, ensure that you click on test mode. This allows us to read and write data without authentication. Note that these rules should not be used in a production application since anyone can access the data.

{

“rules”: {

“.read”: “row < 1610053200000”,

“.write”: “row < 1610053200000”,

}

}

We have successfully setup Firebase and Flutter Project.

Building UI

Ensure that you have the following imports:

* package:flutter/material.dart – This import allows you to access Flutter’s built-in widgets and other functionalities.
* package:firebase\_database/firebase\_database.dart – This import helps access the Firebase Realtime database features.
* package firebase\_core/firebase\_core.dart – This dependency enables you connect to different Firebase products.

Storing Data

We will initialize a Firebase databaseReference object and use it to store data.

**final datanaseRef = FirebaseDatabase.\_instance\_.reference();**

**void addData(String Data) {**

**databaseRef.push().set({‘name’: data, ‘comment’: ‘A good season’});**

**}**

**The addData function stores user input along with a string. We’ll call addData function when RaisedButton is clicked.**

**onPressed: () {**

**addData(textcontroller.text);**

**})),**

We access the user input using a textcontroller. We include textcontroller in HomePageState class.

**class \_HomePageState extends State<HomePage> {**

**final textcontroller = TextEditingController();**

**}**

We need to use FutureBuilder for async operations. It allows application to retrieve results once network operation is completed.

**class \_HomePageState extends State<HomePage> {**

**final Future<FirebaseApp> \_future = Firebase.initializeApp();**

**}**

**return Scaffold(**

**appBar: AppBar(**

**title: Text('Anish Dalvi 10'),**

**),**

**body: FutureBuilder(**

**future: \_future,**

**builder: (context, snapshot) {**

**if(snapshot.hasError) {**

**return Text(snapshot.error.toString());**

**} else {**

**return Container()**

**}**

**}**

Retrieving Data

For simplicity, we will retrieve data and printout data in console

**void printFirebase() {**

**databaseRef.once().then((DataSnapshot snapshot) {**

**print('Data : ${snapshot.value}');**

**});}**

**Code:**

**class Homepage extends StatefulWidget {**

**@override**

**\_HomepageState createState() => \_HomepageState();**

**}**

**class \_HomepageState extends State<Homepage> {**

**@override**

**Widget build(BuildContext context) {**

**return Scaffold();**

**}**

**}**

**Scaffold(**

**appBar: AppBar(title: Text("Anish Dalvi 10"),),**

**body: Container(**

**child: Column(**

**children: <Widget>[**

**SizedBox(height: 250.0),**

**Padding(**

**padding: EdgeInsets.all(10.0),**

**child: TextField(),**

**),**

**SizedBox(height: 30.0),**

**Center(**

**child: RaisedButton(**

**color: Colors.pinkAccent,**

**child: Text("Save to Database"),**

**onPressed: () {**

**//Call method flutter upload**

**},**

**);**

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

**import 'home.dart';**

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

**class MyApp extends StatelessWidget{**

**@override**

**Widget build(BuildContext context) {**

**return MaterialApp(**

**title: "Anish Dalvi 10",**

**theme: ThemeData(**

**primarySwatch: Colors.*blue*,**

**visualDensity: VisualDensity.*adaptivePlatformDensity*,**

**),**

**home: Homepage(),**

**);**

**}**

**}**

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

**import 'package:firebase\_core/firebase\_core.dart';**

**import 'package:firebase\_database/firebase\_database.dart';**

**class HomePage extends StatefulWidget {**

**@override**

**\_HomePageState createState() => \_HomePageState();**

**}**

**class \_HomePageState extends State<HomePage> {**

**final textcontroller = TextEditingController();**

**final databaseRef = FirebaseDatabase.instance.reference();**

**final Future<FirebaeApp> \_future = Firebase.*initializeApp*();**

**void addData(String data) {**

**databaseRef.child("data").push().set({**

**'name': data,**

**'comment': 'A good season',**

**});**

**}**

**void printFirebase() {**

**databaseRef.once().then((DataSnapshot snapshot) {**

**print('Data : ${snapshot.value}');**

**});**

**}**

**@override**

**Widget build(BuildContext context) {**

**printFirebase();**

**return Scaffold(**

**appBar: AppBar(**

**title: Text("Anish Dalvi 10"),**

**),**

**body: FutureBuilder(**

**future: \_future,**

**builder: (context, snapshot) {**

**if(snapshot.hasError) {**

**return Text(snapshot.error.toString());**

**} else {**

**return Container(**

**child: Column(**

**children: <Widget>[**

**SizedBox(height: 250.0),**

**Padding(**

**padding: EdgeInsets.all(10.0),**

**child: TextField(**

**controller: textcontroller,**

**),**

**),**

**SizedBox(height: 30.0),**

**Center(**

**child: RaisedButton(**

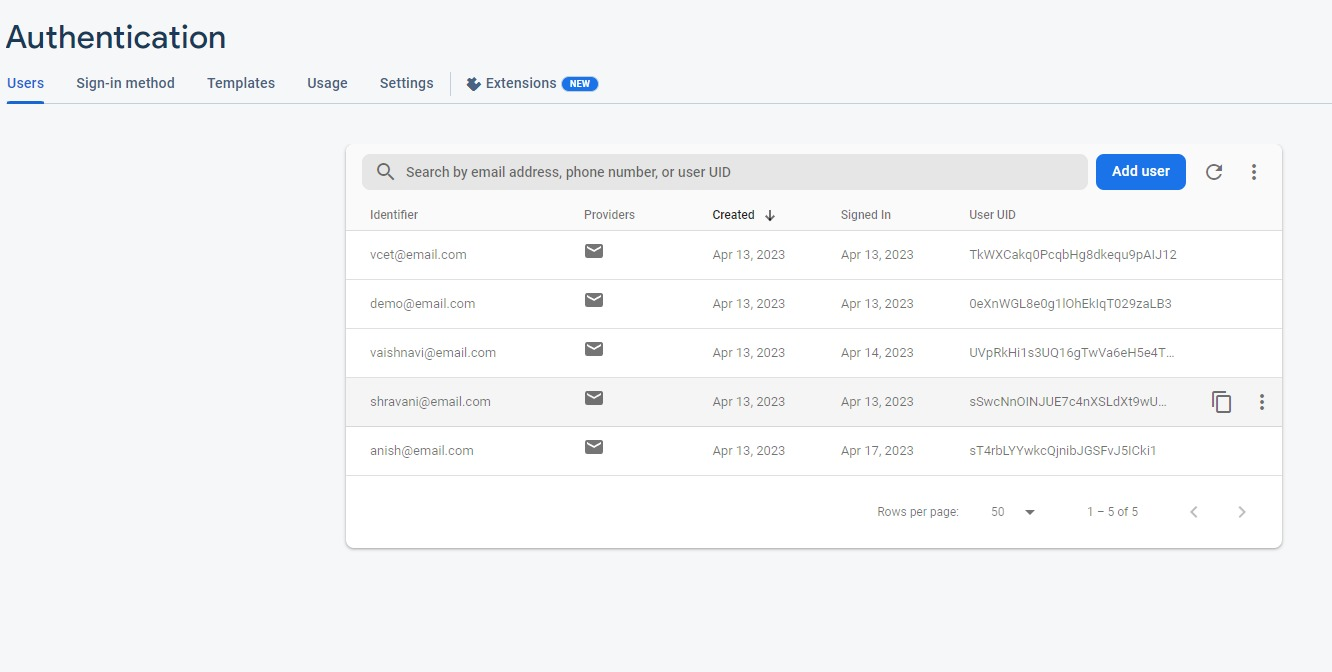
**color: Colors.*pinkAccent*,**

**child: Text("Save to Database"),**

**onPressed: () {**

**addData(textcontroller.text);**

**} ); };**



**Experiment No. 9**

**Aim:** To test & deploy production ready flutter app on android platform.

**Theory:**

Flutter is Google’s mobile app SDK for crafting high-quality native interfaces on iOS and Android in record time. Flutter works with existing code, is used by developers and organizations around the world, and is free and open source.

For users, Flutter makes beautiful app UIs come to life. For developers, Flutter lowers the bar to entry for building mobile apps. It speeds up the development of mobile apps and reduces the cost and complexity of app production across iOS and Android. For designers, Flutter helps deliver the original design vision, without loss of fidelity or compromises. It also acts as a productive prototyping tool.

System requirements:

To install and run Flutter, your development environment must meet these minimum requirements:

* Operating Systems: macOS (64-bit)
* Disk Space: 700 MB (does not include disk space for IDE/tools).
* Tools: Flutter depends on these command-line tools being available in your environment.

• **Setting up Android Studio:**

1. Install Android Studio

2. Download and install Android Studio.

3. Start Android Studio, and go through the ‘Android Studio Setup Wizard’. This installs the latest Android SDK, Android SDK Platform-Tools, which are required by Flutter when developing for Android.

• **Build your app on an Android device:**

To prepare to run and test your Flutter app on an Android device, you’ll need an Android device running Android 4.1 (API level 16) or higher.

Get your Android device ready and follow the steps:

Enable Developer Options and USB debugging on your device.

Graphical user interface, text, application

Description automatically generated

Open the Settings app.

Search for ‘About phone’ and select it.

Find and tap Build number 7 times.

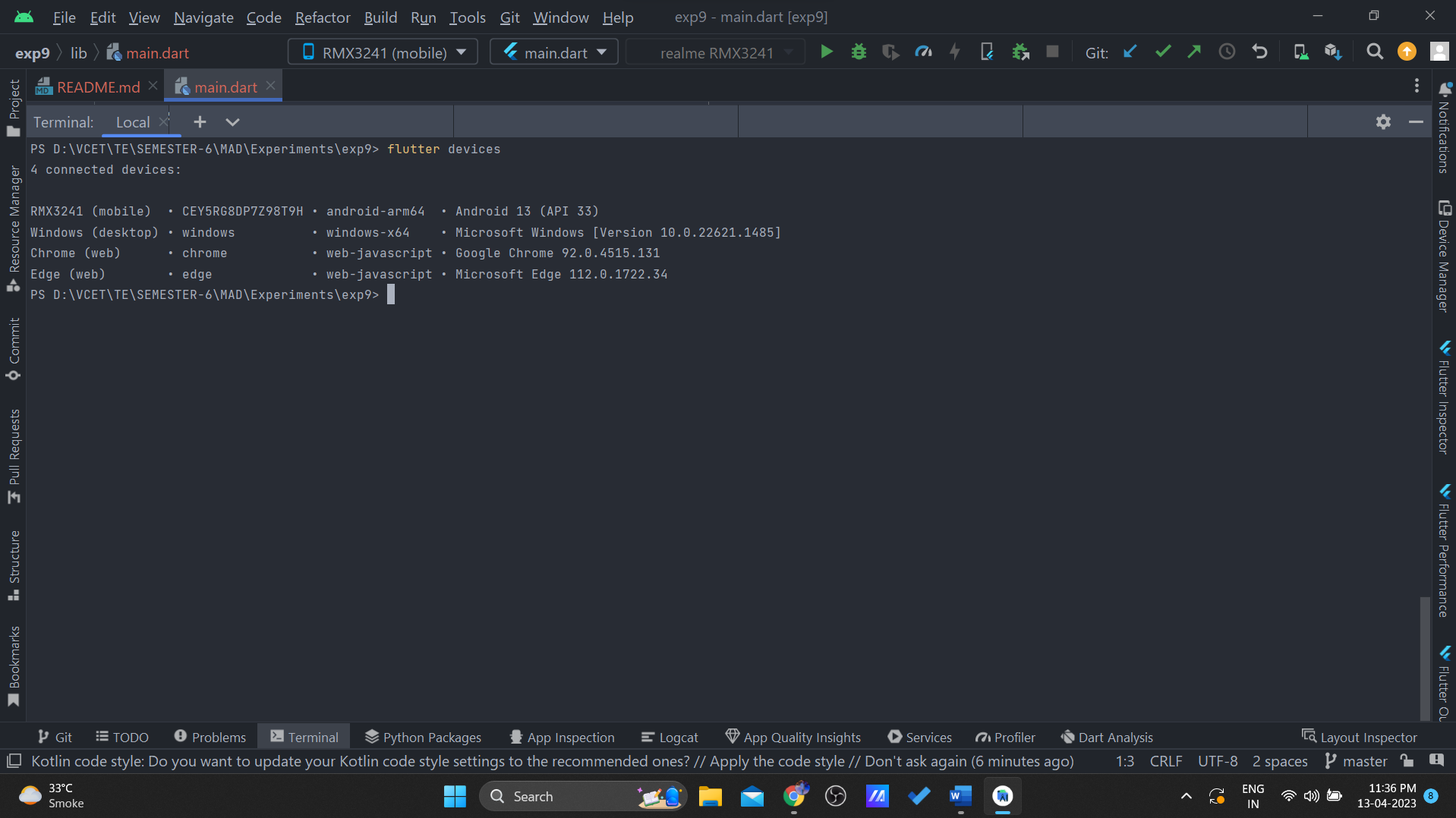
Return to the previous screen to find Developer options near the bottom.

Scroll down and enable USB debugging.

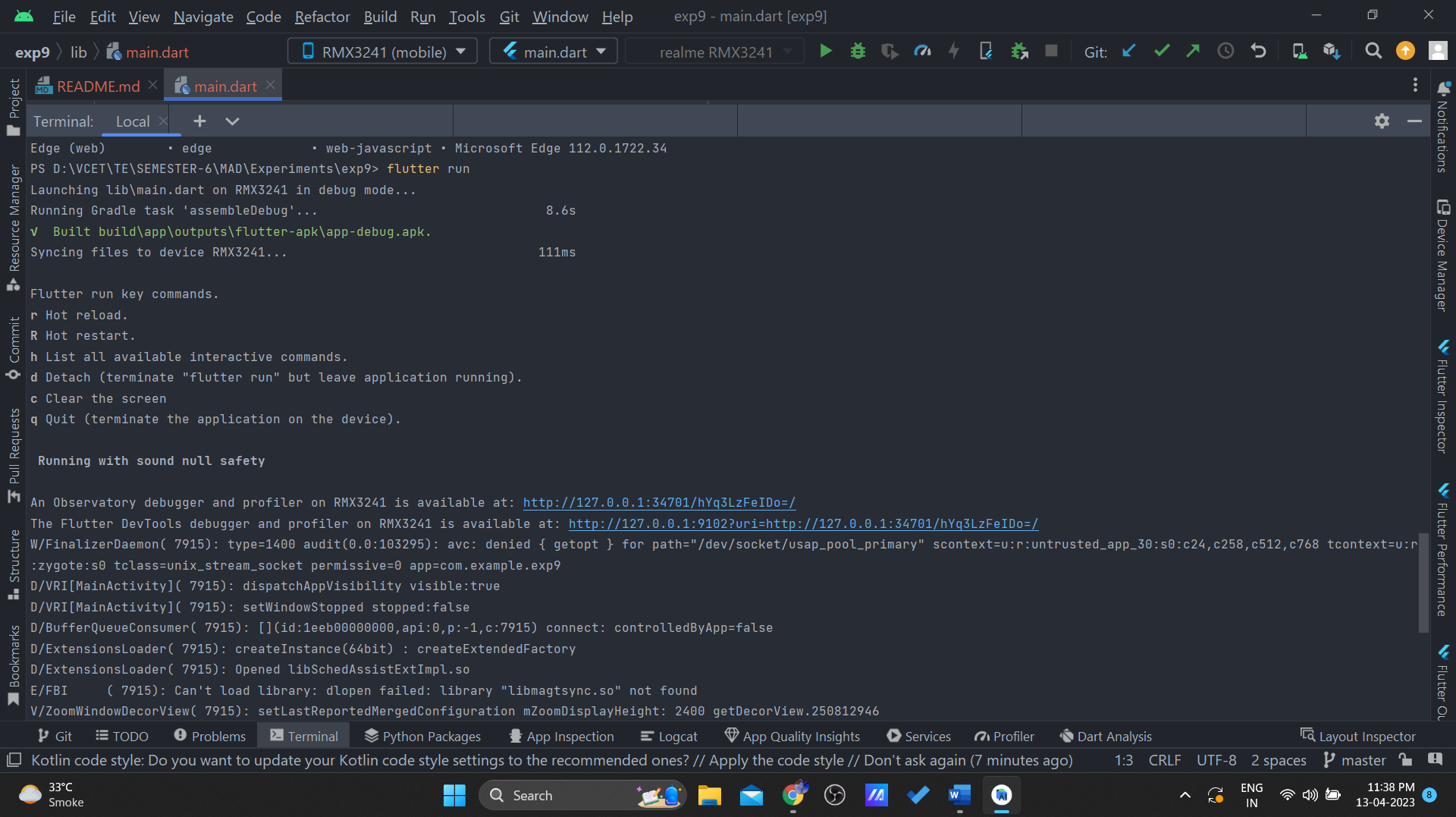
• *Using a USB cable, plug your phone into your computer. If promoted on your device,*

*authorize your computer to access your device.*

*• In the terminal, run the flutter devices command to verify that Flutter recognizes your connected Android device.*

****

• Now all you need to do is to run the flutter run to run it on the android device.

****

**Output:**  Text

Description automatically generated with low confidence Text, letter

Description automatically generated

**Experiment No. 10**

**Aim:** To create a responsive user interface using jQuery mobile for Ecommerce application.

**Theory:**

Over the last few years the devices used to access web applications have grown like anything. We now have mobiles, tablets, desktops, laptops, TV etc. that can be used to access a web site. These all devices vary a lot in sizes. Writing individual applications to cater to all of these different sized and variety of devices is nearly impossible. So we need to design web sites in way that they adjust to the screen size on which they are displayed. The technique developed to solve this problem is Responsive Web Design.

Responsive web design is a series of techniques and technologies that are combined to make delivering a single application across a range of devices as practical as possible. Below is an example of how Google news site looks on different devices.

**• Pillars of Responsive Designs**

*Fluid Grids*

A grid is a way to manage space in a layout in the web world. Fluid grids refer to flexible grid-based layouts that use relative sizing. Traditionally the grids were specified with fixed width columns with widths specified in pixels. But for responsive web design the grid columns widths should be relative to their containers. This helps the responsive web applications to adapt to different screen sizes. In responsive web design the widths are expressed in percentage most of the time.

*Media Queries*

Media queries access the capabilities of device browser and apply styles based on the capabilities that match the query criteria. Media queries enable the web application to adjust itself for optimal viewing based on the browser capabilities.

*Flexible Images and Media*

This feature enables the images and other media to adapt according to the different device sizes and display resolutions. The most basic technique to adapt images and media is by using scaling or CSS overflow property. Though these techniques adapt the image according to the device size but the image download still take up the user’s mobile bandwidth. For better user experience one can use different set of images and media for different devices.

**• jQuery Mobile Framework**

The jQuery Mobile Framework is an open source cross-platform UI framework. It is built using HTML5, CSS3, and the very popular jQuery JavaScript library, and it follows Open Web standards. It provides touch-friendly UI widgets that are specially styled for mobile devices. It has a powerful theming framework to style your applications. It supports AJAX for various tasks, such as page navigation and transitions.

**• How jQuery Mobile supports RWD ?**

*Unified UI*

jQuery Mobile devices unified user experience by designing to HTML5 and CSS3. jQuery codebase will render consistently across all supported platforms without need of customizations.

**• Progressive Enhancement**

Progressive enhancement defines layers of compatibility that allow any user to access the basic content, services, and functionality of a website, while providing an enhanced experience for browsers with better support standards. jQuery Mobile is totally built using this technique.

**• CSS Selector**

jQuery Mobile has predefined set of CSS classes that it applies to HTML elements depending on current orientation or size of device.

**• Orientation Classes**

If the devices is in landscape mode, jQuery Mobile will apply landscape class to HTML elements. Similarly, if device is in portrait mode, portrait class is applied by jQuery Mobile to HTML elements. One can override these orientation. For example, if we want to have an image background in portrait orientation, we can have following CSS rule:

.potrait section{

background-image: url(image/background.png);

}

**• Breakpoint Classes**

In addition to orientation changes, jQuery Mobile provides CSS Selectors for handling different screen sizes. It refers to these as breakpoint classes. It has defined screen size breakpoints at 320 pixels, 480 pixels, 1024 pixels, etc. Corresponding to these sizes, min-width and max-width CSS classes are defined in jQuery Mobile like minwidth-320px, min-width-480px, etc.

As with orientation classes, jQuery Mobile applies these breakpoint classes to HTML elements depending on screen size. These can be defined in application CSS to override default behaviour.

**• Flexible Layout**

In jQuery Mobile most of the components and form elements are designed to be flexible width so that they comfortably fit in width of any device. Additionally, form elements and labels are represented differently based on screen size. On smaller screens, labels are stacked on top of form elements while on wider screen, labels and elements are styled to be on same line in 2 column grid layouts.

**• Responsive Tables**

This is a newly added feature to jQuery Mobile. This allows us to display large amount of tabular data in a way that looks good on both mobiles and desktops. There are 2 modes of responsive tables:

Reflow: It vertically stacks cells in rows by default so that data could be easily readable on mobile phones. Additional style needs to be applied to make table display in traditional row-column format. This is default mode for tables defined using datarole=“table”.

Column Toggle: It hides low priority columns at narrower widths. The column priority is specified using data-priority attribute in <th> element. This can be between 1(Highest) and 6(Lowest). If data-priority attribute is missing for a column then that column is always displayed.

**• Media Queries**

In addition to preset breakpoints, the CSS3 media queries can easily be used with jQuery Mobile. Since it is a JavaScript framework for HTML sites, media queries work seamlessly with jQuery Mobile. One can easily define the apply new media queries to jQuery Mobile applications. For example, if we want grid to be stacked when screen width is 720px or less, we can define the media query like the following:

/\*stack all grids below 720px \*/

@media all and (max-width: 45cm){

-grid-breakpoint .ul-block-a,

-grid-breakpoint .ul-block-b,

-grid-breakpoint .ul-block-c,

-grid-breakpoint .ul-block-d,

-grid-breakpoint .ul-block-e {

width: 100%;

float: none;

}

}

**Conclusion:**

As we can see jQuery Mobile is built keeping responsive design in mind. It provides a lot of out of box features to support responsive design. One of the advantages of jQuery Mobile is that it provides a base framework which is well tested to be working fine on various screen can concentrate on application features without worrying about the cross-browser compatibility and screen size issues.

**Experiment No. 11**

**Aim:** To code and register a service worker & complete the installation and activation process for a new service for the ecommerce PWA.

**Theory:**

Progressive WebApps are a new type of web app that offers native-like capabilities, yet their reach and performance are indistinguishable from a website. This means that, unlike a native app, a PWA’s UI won’t be limited by the user’s device or by system.

E-commerce PWAs can also access the devices’s offline capabilities and can use system-level notification channels to meet needs of users acress platforms. They are optimized for all platforms, and they are fast, reliable, engaging, and usable in every major browser.

**WHAT IS PWA ?**

Progressive WebApps in e-commerce are like native apps with a better build on the web. There is no need to download and install a separate app. It lives in a browser, providing a fast, more reliable, and engaging user experience. They have an installation experience that makes users feel like they are getting an app for the first time.

In addition to providing smooth, native-like experience, they are more secure. They use an application shell, called a service worker, to cache data and resources and serve it to user. This allows the app to be installed on a new device without affecting user’s experience.

**HOW DOES PWA WORK ?**

PWA enables users to access data and content, typically through a browser interface that is hosted on remote servers connected to Internet. In this way, it differs from native apps that often store and access data locally on devices themselves.

Here are 10 common features of PWA for e-commerce:

1. Responsive design
2. Functioning regardless of internet connectivity of the device
3. Fast app-like interactions and animations
4. Instant updates
5. Compatibility with all devices
6. Security
7. Indexing by search engines
8. Interaction with users via push notifications
9. Possibility to install on a home screen
10. Easy distribution via links without need for installation

It allows PWAs to have a more seamless experience for users, as web technologies are already supported and well-understood.

**Code:**

1. **Create a new file named ‘service-worker.js’ in root directory of your project and add the following**

**const CACHE\_NAME = 'my-ecommerce-pwa-cache-v1';**

**const urlsToCache = [**

**'/',**

**'/index.html',**

**'/css/style.css',**

**'/js/app.js',**

**'/images/logo.png',**

**'/images/banner.jpg'**

**];**

**self.addEventListener('install', event => {**

**event.waitUntil(**

**caches.open(CACHE\_NAME)**

**.then(cache => cache.addAll(urlsToCache))**

**);**

**});**

**self.addEventListener('fetch', event => {**

**event.respondWith(**

**caches.match(event.request)**

**.then(response => {**

**if (response) {**

**return response;**

**}**

**return fetch(event.request);**

**})**

**);**

**});**

This code sets up a cache named my-ecommerce-pwa-cache-v1 and caches a few static assets for offline access.

1. **Register the service worker**

In your app.js file, register the service worker by adding the following code:

**if ('serviceWorker' in navigator) {**

**window.addEventListener('load', () => {**

**navigator.serviceWorker.register('/service-worker.js')**

**.then(registration => console.log('ServiceWorker registered'))**

**.catch(error => console.log('ServiceWorker registration failed: ', error));**

**});**

**}**

This code checks if the browser supports service workers, and if it does, registers the service worker at the URL /service-worker.js

1. **Add the following html code:**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8" />**

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

**<meta http-equiv="X-UA-Compatible" content="ie=edge" />**

**<title>E-commerce PWA</title>**

**<link rel="manifest" href="manifest.json" />**

**</head>**

**<body>**

**<main>**

**<section>**

**<h1>Welcome to our E-commerce PWA!</h1>**

**<p>Here you can find the best deals on all kinds of products.</p>**

**</section>**

**<section>**

**<h2>Featured Products</h2>**

**<ul>**

**<li>Product 1</li>**

**<li>Product 2</li>**

**<li>Product 3</li>**

**</ul>**

**</section>**

**</main>**

**<script src="js/app.js"></script>**

**<script>**

**if ("serviceWorker" in navigator) {**

**window.addEventListener("load", () => {**

**navigator.serviceWorker**

**.register("service-worker.js")**

**.then(registration => {**

**console.log("Service worker registered with scope:", registration.scope);**

**})**

**.catch(error => {**

**console.log("Service worker registration failed:", error);**

**});**

**});**

**}**

**</script>**

**</body>**

**</html>**

1. **Run the code**

Open your application in a browser and check the console for a message that says "ServiceWorker registered".

1. **Test it out**

To test the app, close any open instances of the pp in your browser and stop the local server. Run the following in command line to clean out old files in the dist folder, rebuild it and serve the app:

Test your service worker by disconnecting from the internet and refreshing the page. The assets that were cached should still be visible, even though the network is unavailable.

The app should load normally!

**Experiment No. 12**

**Aim:** To deploy an Ecommerce PWA using SSL enabled static hosting solution.

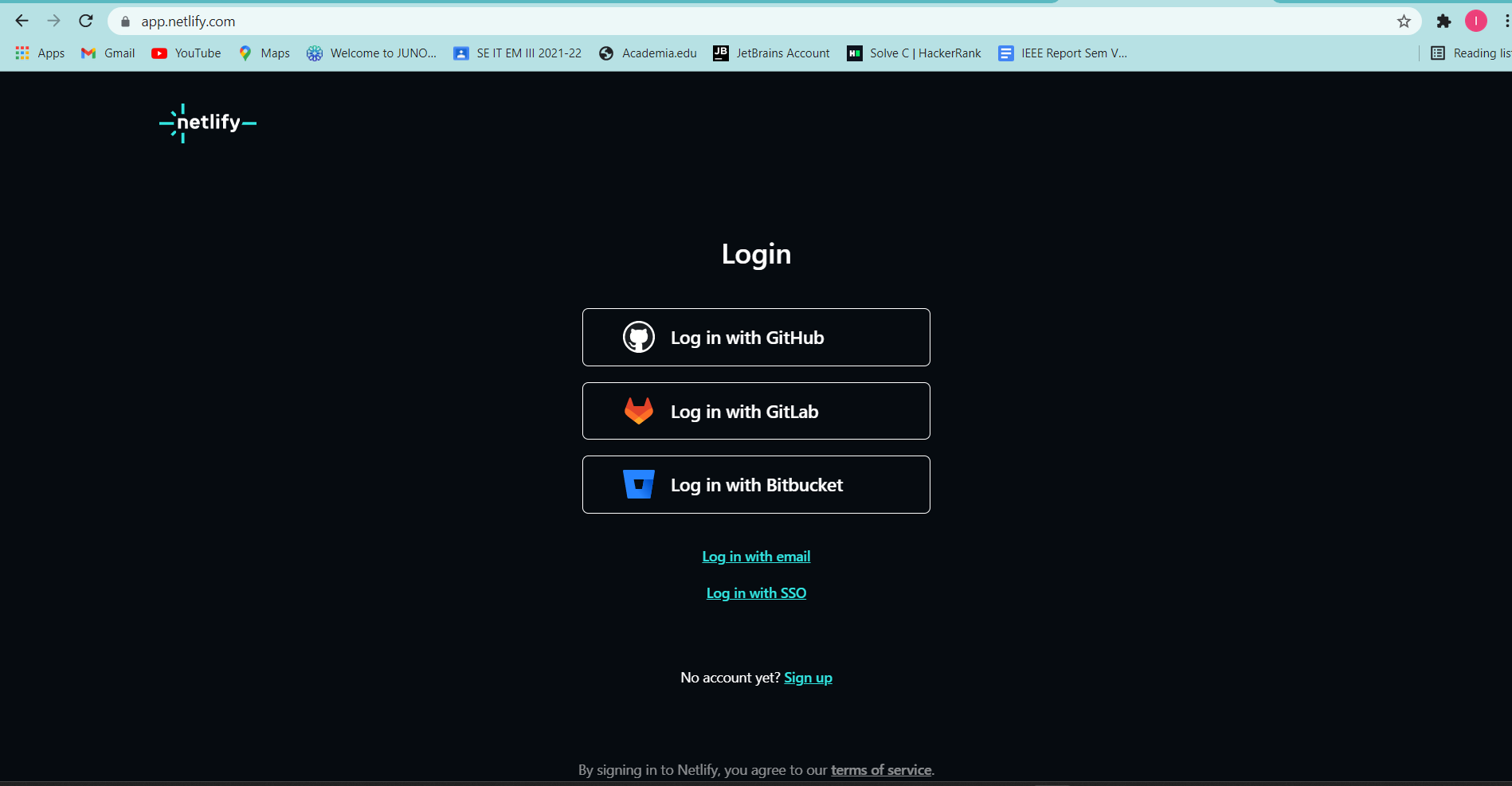
**Theory:**

Following prerequisite in order to host a website/web-app in Netlfiy:

* **A GITHUB ACCOUNT**

**Code:**

**Step 1:**  Create a Netlify account or sign in to your existing account.



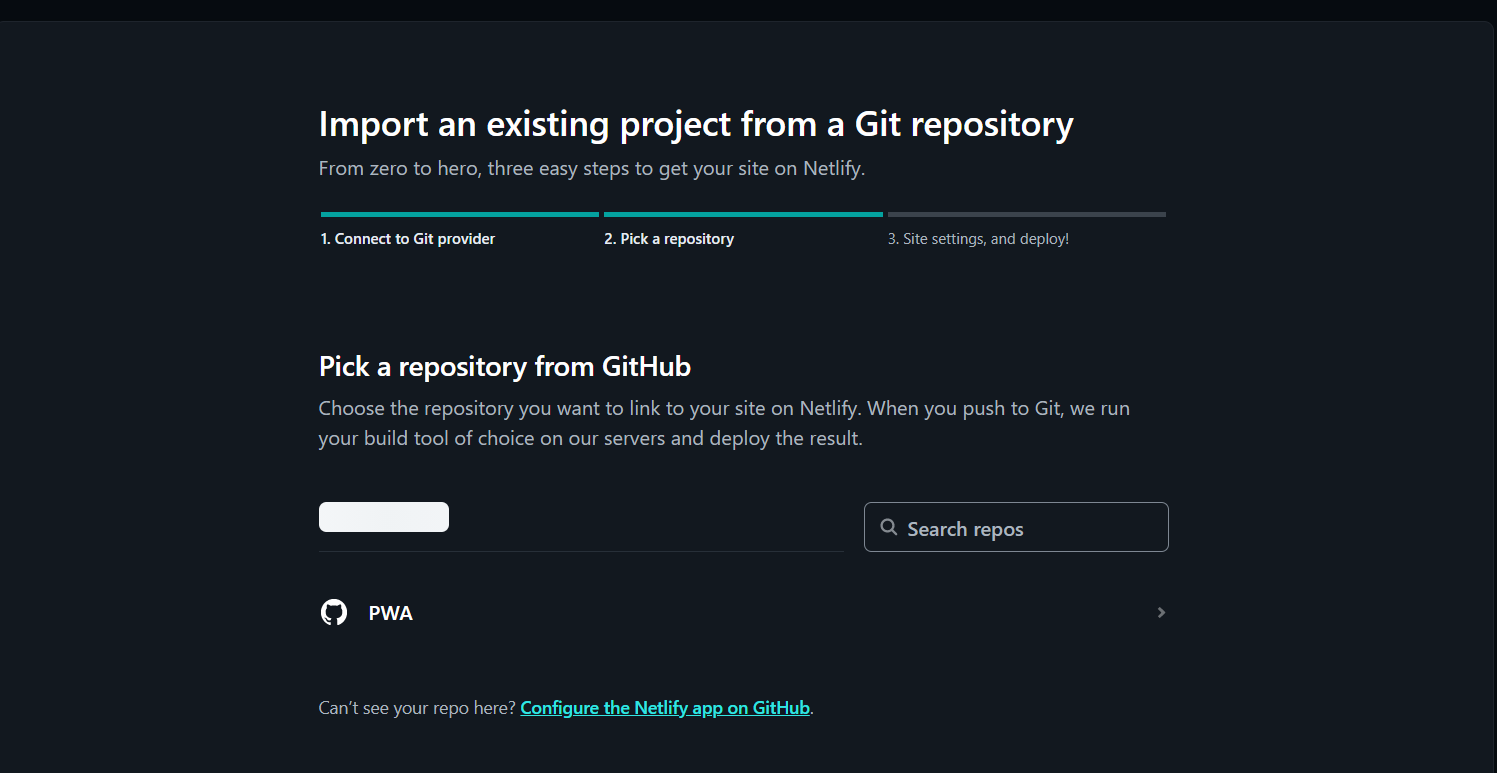
**Step 2**: Click on the "New site from Git" button on the Netlify dashboard.

A screenshot of a computer

Description automatically generated with medium confidence

**Step 3**: Select your Git provider (e.g., GitHub, GitLab, Bitbucket) and authenticate with your account.

**Step 4:** Choose the repository where you have your Ecommerce PWA code.



**Step 5**: Configure your build settings. Set the build command to build your PWA (e.g., npm run build) and set the publish directory to the folder where your built files are located (e.g., dist).

A screenshot of a computer

Description automatically generated with medium confidence

**Step 6**: Enable SSL/TLS by going to "Site settings" > "Domain management" > "HTTPS". Select "Secure" and follow the instructions to set up SSL for your custom domain or the default Netlify subdomain.

A screenshot of a computer

Description automatically generated with medium confidence

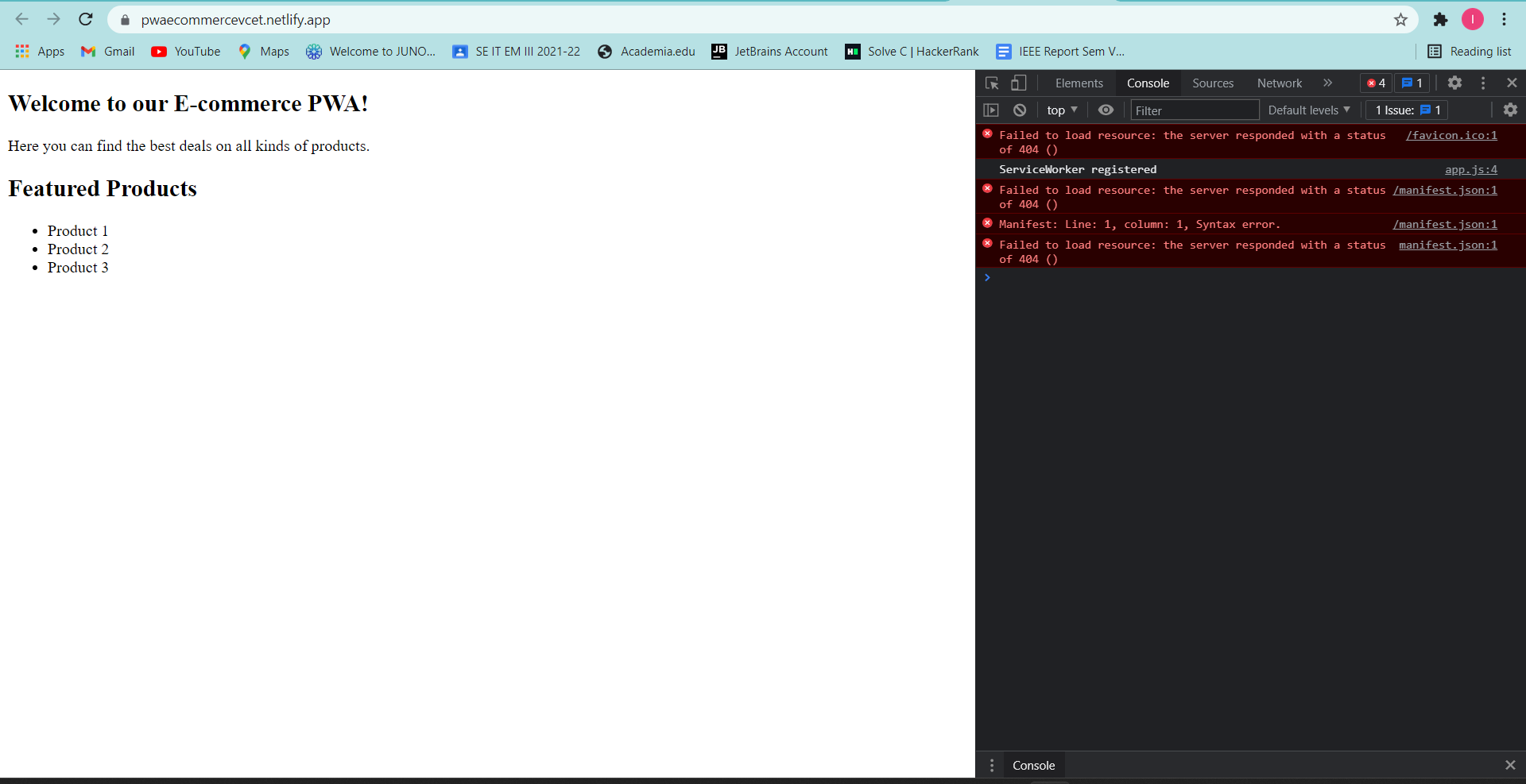
**Step 7:** You can have your custom domain also.

Graphical user interface, text

Description automatically generated

**Step 8**: Netlify will automatically build and deploy your Ecommerce PWA. Once the deploy is complete, you can access your PWA by visiting the Netlify subdomain or your custom domain.

You can view your site by going through the URL of your domain. We access ours by going to <https://pwaecommercevcet.netlify.app/>



**MAD MINI PROJECT REPORT**

**Name: Anish Dalvi**

**Roll No: 10**

**Topic: Instagram Clone**

**Flutter:**

Flutter SDK is an open-source software development kit for building beautiful UI which is natively compiled. The idea is to implement a Pokedex App which will show list of Pokemon along with their images, generation and other information in an attractive manner. Making this app will give us a good revision of flutter and dart basics along with a good grasp on http protocols to fetch data.

**Instagram:**

Instagram is a free photo and video sharing app available on iPhone and Android. People can upload photos or videos to our service and share them with their followers or with a select group of friends. They can also view, comment and like posts shared by their friends on Instagram.

**Code**:

***main.dart***

**import 'package:firebase\_auth/firebase\_auth.dart';**

**import 'package:firebase\_core/firebase\_core.dart';**

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

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

**import 'package:instagram\_clone\_flutter/providers/user\_provider.dart';**

**import 'package:instagram\_clone\_flutter/responsive/mobile\_screen\_layout.dart';**

**import 'package:instagram\_clone\_flutter/responsive/responsive\_layout.dart';**

**import 'package:instagram\_clone\_flutter/responsive/web\_screen\_layout.dart';**

**import 'package:instagram\_clone\_flutter/screens/login\_screen.dart';**

**import 'package:instagram\_clone\_flutter/utils/colors.dart';**

**import 'package:provider/provider.dart';**

**void main() async {**

**WidgetsFlutterBinding.ensureInitialized();**

**// initialise app based on platform- web or mobile**

**if (kIsWeb) {**

**await Firebase.initializeApp(**

**options: const FirebaseOptions(**

**apiKey: "AIzaSyBUOnP1qEduzpY2MJzJleJC481ApdEa5mA",**

**appId: "1:479554440064:web:452fd8520c6e2e68c90f65",**

**messagingSenderId: "479554440064",**

**projectId: "instagram-clone-f0afb",**

**storageBucket: 'instagram-clone-f0afb.appspot.com'**

**),**

**);**

**} else {**

**await Firebase.initializeApp();**

**}**

**runApp(const MyApp());**

**}**

**class MyApp extends StatelessWidget {**

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

**@override**

**Widget build(BuildContext context) {**

**return MultiProvider(**

**providers: [**

**ChangeNotifierProvider(create: (\_) => UserProvider(),),**

**],**

**child: MaterialApp(**

**debugShowCheckedModeBanner: false,**

**title: 'Instagram Clone',**

**theme: ThemeData.dark().copyWith(**

**scaffoldBackgroundColor: mobileBackgroundColor,**

**),**

**home: StreamBuilder(**

**stream: FirebaseAuth.instance.authStateChanges(),**

**builder: (context, snapshot) {**

**if (snapshot.connectionState == ConnectionState.active) {**

**// Checking if the snapshot has any data or not**

**if (snapshot.hasData) {**

**// if snapshot has data which means user is logged in then we check the width of screen and accordingly display the screen layout**

**return const ResponsiveLayout(**

**mobileScreenLayout: MobileScreenLayout(),**

**webScreenLayout: WebScreenLayout(),**

**);**

**} else if (snapshot.hasError) {**

**return Center(**

**child: Text('${snapshot.error}'),**

**);**

**}**

**}**

**// means connection to future hasnt been made yet**

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

**return const Center(**

**child: CircularProgressIndicator(),**

**);**

**}**

**return const LoginScreen();**

**},**

**),**

**),**

**);**

**}**

**}**

***login\_screen.dart***

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

**import 'package:flutter\_svg/flutter\_svg.dart';**

**import 'package:instagram\_clone\_flutter/resources/auth\_methods.dart';**

**import 'package:instagram\_clone\_flutter/responsive/mobile\_screen\_layout.dart';**

**import 'package:instagram\_clone\_flutter/responsive/responsive\_layout.dart';**

**import 'package:instagram\_clone\_flutter/responsive/web\_screen\_layout.dart';**

**import 'package:instagram\_clone\_flutter/screens/signup\_screen.dart';**

**import 'package:instagram\_clone\_flutter/utils/colors.dart';**

**import 'package:instagram\_clone\_flutter/utils/global\_variable.dart';**

**import 'package:instagram\_clone\_flutter/utils/utils.dart';**

**import 'package:instagram\_clone\_flutter/widgets/text\_field\_input.dart';**

**class LoginScreen extends StatefulWidget {**

**const LoginScreen({Key? key}) : super(key: key);**

**@override**

**\_LoginScreenState createState() => \_LoginScreenState();**

**}**

**class \_LoginScreenState extends State<LoginScreen> {**

**final TextEditingController \_emailController = TextEditingController();**

**final TextEditingController \_passwordController = TextEditingController();**

**bool \_isLoading = false;**

**@override**

**void dispose() {**

**super.dispose();**

**\_emailController.dispose();**

**\_passwordController.dispose();**

**}**

**void loginUser() async {**

**setState(() {**

**\_isLoading = true;**

**});**

**String res = await AuthMethods().loginUser(**

**email: \_emailController.text, password: \_passwordController.text);**

**if (res == 'success') {**

**Navigator.of(context).pushAndRemoveUntil(**

**MaterialPageRoute(**

**builder: (context) => const ResponsiveLayout(**

**mobileScreenLayout: MobileScreenLayout(),**

**webScreenLayout: WebScreenLayout(),**

**),**

**),**

**(route) => false);**

**setState(() {**

**\_isLoading = false;**

**});**

**} else {**

**setState(() {**

**\_isLoading = false;**

**});**

**showSnackBar(context, res);**

**}**

**}**

**@override**

**Widget build(BuildContext context) {**

**return Scaffold(**

**resizeToAvoidBottomInset: false,**

**body: SafeArea(**

**child: Container(**

**padding: MediaQuery.of(context).size.width > webScreenSize**

**? EdgeInsets.symmetric(**

**horizontal: MediaQuery.of(context).size.width / 3)**

**: const EdgeInsets.symmetric(horizontal: 32),**

**width: double.infinity,**

**child: Column(**

**crossAxisAlignment: CrossAxisAlignment.center,**

**children: [**

**Flexible(**

**child: Container(),**

**flex: 2,**

**),**

**SvgPicture.asset(**

**'assets/ic\_instagram.svg',**

**color: primaryColor,**

**height: 64,**

**),**

**const SizedBox(**

**height: 64,**

**),**

**TextFieldInput(**

**hintText: 'Enter your email',**

**textInputType: TextInputType.emailAddress,**

**textEditingController: \_emailController,**

**),**

**const SizedBox(**

**height: 24,**

**),**

**TextFieldInput(**

**hintText: 'Enter your password',**

**textInputType: TextInputType.text,**

**textEditingController: \_passwordController,**

**isPass: true,**

**),**

**const SizedBox(**

**height: 24,**

**),**

**InkWell(**

**child: Container(**

**child: !\_isLoading**

**? const Text(**

**'Log in',**

**)**

**: const CircularProgressIndicator(**

**color: primaryColor,**

**),**

**width: double.infinity,**

**alignment: Alignment.center,**

**padding: const EdgeInsets.symmetric(vertical: 12),**

**decoration: const ShapeDecoration(**

**shape: RoundedRectangleBorder(**

**borderRadius: BorderRadius.all(Radius.circular(4)),**

**),**

**color: blueColor,**

**),**

**),**

**onTap: loginUser,**

**),**

**const SizedBox(**

**height: 12,**

**),**

**Flexible(**

**child: Container(),**

**flex: 2,**

**),**

**Row(**

**mainAxisAlignment: MainAxisAlignment.center,**

**children: [**

**Container(**

**child: const Text(**

**'Dont have an account?',**

**),**

**padding: const EdgeInsets.symmetric(vertical: 8),**

**),**

**GestureDetector(**

**onTap: () => Navigator.of(context).push(**

**MaterialPageRoute(**

**builder: (context) => const SignupScreen(),**

**),**

**),**

**child: Container(**

**child: const Text(**

**' Signup.',**

**style: TextStyle(**

**fontWeight: FontWeight.bold,**

**),**

**),**

**padding: const EdgeInsets.symmetric(vertical: 8),**

**),**

**),**

**],**

**),**

**],**

**),**

**),**

**),**

**);**

**}**

**}**

***signup\_screen.dart***

**import 'dart:typed\_data';**

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

**import 'package:flutter\_svg/flutter\_svg.dart';**

**import 'package:image\_picker/image\_picker.dart';**

**import 'package:instagram\_clone\_flutter/resources/auth\_methods.dart';**

**import 'package:instagram\_clone\_flutter/responsive/mobile\_screen\_layout.dart';**

**import 'package:instagram\_clone\_flutter/responsive/responsive\_layout.dart';**

**import 'package:instagram\_clone\_flutter/responsive/web\_screen\_layout.dart';**

**import 'package:instagram\_clone\_flutter/screens/login\_screen.dart';**

**import 'package:instagram\_clone\_flutter/utils/colors.dart';**

**import 'package:instagram\_clone\_flutter/utils/global\_variable.dart';**

**import 'package:instagram\_clone\_flutter/utils/utils.dart';**

**import 'package:instagram\_clone\_flutter/widgets/text\_field\_input.dart';**

**class SignupScreen extends StatefulWidget {**

**const SignupScreen({Key? key}) : super(key: key);**

**@override**

**\_SignupScreenState createState() => \_SignupScreenState();**

**}**

**class \_SignupScreenState extends State<SignupScreen> {**

**final TextEditingController \_usernameController = TextEditingController();**

**final TextEditingController \_emailController = TextEditingController();**

**final TextEditingController \_passwordController = TextEditingController();**

**final TextEditingController \_bioController = TextEditingController();**

**bool \_isLoading = false;**

**Uint8List? \_image;**

**@override**

**void dispose() {**

**super.dispose();**

**\_emailController.dispose();**

**\_passwordController.dispose();**

**\_usernameController.dispose();**

**}**

**void signUpUser() async {**

**// set loading to true**

**setState(() {**

**\_isLoading = true;**

**});**

**// signup user using our authmethodds**

**String res = await AuthMethods().signUpUser(**

**email: \_emailController.text,**

**password: \_passwordController.text,**

**username: \_usernameController.text,**

**bio: \_bioController.text,**

**file: \_image!);**

**// if string returned is sucess, user has been created**

**if (res == "success") {**

**setState(() {**

**\_isLoading = false;**

**});**

**// navigate to the home screen**

**Navigator.of(context).pushReplacement(**

**MaterialPageRoute(**

**builder: (context) => const ResponsiveLayout(**

**mobileScreenLayout: MobileScreenLayout(),**

**webScreenLayout: WebScreenLayout(),**

**),**

**),**

**);**

**} else {**

**setState(() {**

**\_isLoading = false;**

**});**

**// show the error**

**showSnackBar(context, res);**

**}**

**}**

**selectImage() async {**

**Uint8List im = await pickImage(ImageSource.gallery);**

**// set state because we need to display the image we selected on the circle avatar**

**setState(() {**

**\_image = im;**

**});**

**}**

**@override**

**Widget build(BuildContext context) {**

**return Scaffold(**

**resizeToAvoidBottomInset: false,**

**body: SafeArea(**

**child: Container(**

**padding: const EdgeInsets.symmetric(horizontal: 32),**

**width: double.infinity,**

**child: Column(**

**crossAxisAlignment: CrossAxisAlignment.center,**

**children: [**

**Flexible(**

**child: Container(),**

**flex: 2,**

**),**

**SvgPicture.asset(**

**'assets/ic\_instagram.svg',**

**color: primaryColor,**

**height: 64,**

**),**

**const SizedBox(**

**height: 64,**

**),**

**Stack(**

**children: [**

**\_image != null**

**? CircleAvatar(**

**radius: 64,**

**backgroundImage: MemoryImage(\_image!),**

**backgroundColor: Colors.red,**

**)**

**: const CircleAvatar(**

**radius: 64,**

**backgroundImage: NetworkImage(**

**'https://i.stack.imgur.com/l60Hf.png'),**

**backgroundColor: Colors.red,**

**),**

**Positioned(**

**bottom: -10,**

**left: 80,**

**child: IconButton(**

**onPressed: selectImage,**

**icon: const Icon(Icons.add\_a\_photo),**

**),**

**)**

**],**

**),**

**const SizedBox(**

**height: 24,**

**),**

**TextFieldInput(**

**hintText: 'Enter your username',**

**textInputType: TextInputType.text,**

**textEditingController: \_usernameController,**

**),**

**const SizedBox(**

**height: 24,**

**),**

**TextFieldInput(**

**hintText: 'Enter your email',**

**textInputType: TextInputType.emailAddress,**

**textEditingController: \_emailController,**

**),**

**const SizedBox(**

**height: 24,**

**),**

**TextFieldInput(**

**hintText: 'Enter your password',**

**textInputType: TextInputType.text,**

**textEditingController: \_passwordController,**

**isPass: true,**

**),**

**const SizedBox(**

**height: 24,**

**),**

**TextFieldInput(**

**hintText: 'Enter your bio',**

**textInputType: TextInputType.text,**

**textEditingController: \_bioController,**

**),**

**const SizedBox(**

**height: 24,**

**),**

**InkWell(**

**child: Container(**

**child: !\_isLoading**

**? const Text(**

**'Sign up',**

**)**

**: const CircularProgressIndicator(**

**color: primaryColor,**

**),**

**width: double.infinity,**

**alignment: Alignment.center,**

**padding: const EdgeInsets.symmetric(vertical: 12),**

**decoration: const ShapeDecoration(**

**shape: RoundedRectangleBorder(**

**borderRadius: BorderRadius.all(Radius.circular(4)),**

**),**

**color: blueColor,**

**),**

**),**

**onTap: signUpUser,**

**),**

**const SizedBox(**

**height: 12,**

**),**

**Flexible(**

**child: Container(),**

**flex: 2,**

**),**

**Row(**

**mainAxisAlignment: MainAxisAlignment.center,**

**children: [**

**Container(**

**child: const Text(**

**'Already have an account?',**

**),**

**padding: const EdgeInsets.symmetric(vertical: 8),**

**),**

**GestureDetector(**

**onTap: () => Navigator.of(context).push(**

**MaterialPageRoute(**

**builder: (context) => const LoginScreen(),**

**),**

**),**

**child: Container(**

**child: const Text(**

**' Login.',**

**style: TextStyle(**

**fontWeight: FontWeight.bold,**

**),**

**),**

**padding: const EdgeInsets.symmetric(vertical: 8),**

**),**

**),**

**],**

**),**

**],**

**),**

**),**

**),**

**);**

**}**

**}**

***auth\_methods.dart***

**import 'dart:typed\_data';**

**import 'package:cloud\_firestore/cloud\_firestore.dart';**

**import 'package:firebase\_auth/firebase\_auth.dart';**

**import 'package:instagram\_clone\_flutter/models/user.dart' as model;**

**import 'package:instagram\_clone\_flutter/resources/storage\_methods.dart';**

**class AuthMethods {**

**final FirebaseFirestore \_firestore = FirebaseFirestore.*instance*;**

**final FirebaseAuth \_auth = FirebaseAuth.*instance*;**

**// get user details**

**Future<model.User> getUserDetails() async {**

**User currentUser = \_auth.currentUser!;**

**DocumentSnapshot documentSnapshot =**

**await \_firestore.collection('users').doc(currentUser.uid).get();**

**return model.User.*fromSnap*(documentSnapshot);**

**}**

**// Signing Up User**

**Future<String> signUpUser({**

**required String email,**

**required String password,**

**required String username,**

**required String bio,**

**required Uint8List file,**

**}) async {**

**String res = "Some error Occurred";**

**try {**

**if (email.isNotEmpty ||**

**password.isNotEmpty ||**

**username.isNotEmpty ||**

**bio.isNotEmpty ||**

**file != null) {**

**// registering user in auth with email and password**

**UserCredential cred = await \_auth.createUserWithEmailAndPassword(**

**email: email,**

**password: password,**

**);**

**String photoUrl =**

**await StorageMethods().uploadImageToStorage('profilePics', file, false);**

**model.User \_user = model.User(**

**username: username,**

**uid: cred.user!.uid,**

**photoUrl: photoUrl,**

**email: email,**

**bio: bio,**

**followers: [],**

**following: [],**

**);**

**// adding user in our database**

**await \_firestore**

**.collection("users")**

**.doc(cred.user!.uid)**

**.set(\_user.toJson());**

**res = "success";**

**} else {**

**res = "Please enter all the fields";**

**}**

**} catch (err) {**

**return err.toString();**

**}**

**return res;**

**}**

**// logging in user**

**Future<String> loginUser({**

**required String email,**

**required String password,**

**}) async {**

**String res = "Some error Occurred";**

**try {**

**if (email.isNotEmpty || password.isNotEmpty) {**

**// logging in user with email and password**

**await \_auth.signInWithEmailAndPassword(**

**email: email,**

**password: password,**

**);**

**res = "success";**

**} else {**

**res = "Please enter all the fields";**

**}**

**} catch (err) {**

**return err.toString();**

**}**

**return res;**

**}**

**Future<void> signOut() async {**

**await \_auth.signOut();**

**}**

**}**

***profile\_screen.dart***

**import 'package:cloud\_firestore/cloud\_firestore.dart';**

**import 'package:firebase\_auth/firebase\_auth.dart';**

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

**import 'package:instagram\_clone\_flutter/resources/auth\_methods.dart';**

**import 'package:instagram\_clone\_flutter/resources/firestore\_methods.dart';**

**import 'package:instagram\_clone\_flutter/screens/login\_screen.dart';**

**import 'package:instagram\_clone\_flutter/utils/colors.dart';**

**import 'package:instagram\_clone\_flutter/utils/utils.dart';**

**import 'package:instagram\_clone\_flutter/widgets/follow\_button.dart';**

**class ProfileScreen extends StatefulWidget {**

**final String uid;**

**const ProfileScreen({Key? key, required this.uid}) : super(key: key);**

**@override**

**\_ProfileScreenState createState() => \_ProfileScreenState();**

**}**

**class \_ProfileScreenState extends State<ProfileScreen> {**

**var userData = {};**

**int postLen = 0;**

**int followers = 0;**

**int following = 0;**

**bool isFollowing = false;**

**bool isLoading = false;**

**@override**

**void initState() {**

**super.initState();**

**getData();**

**}**

**getData() async {**

**setState(() {**

**isLoading = true;**

**});**

**try {**

**var userSnap = await FirebaseFirestore.*instance***

**.collection('users')**

**.doc(widget.uid)**

**.get();**

**// get post lENGTH**

**var postSnap = await FirebaseFirestore.*instance***

**.collection('posts')**

**.where('uid', isEqualTo: FirebaseAuth.*instance*.currentUser!.uid)**

**.get();**

**postLen = postSnap.docs.length;**

**userData = userSnap.data()!;**

**followers = userSnap.data()!['followers'].length;**

**following = userSnap.data()!['following'].length;**

**isFollowing = userSnap**

**.data()!['followers']**

**.contains(FirebaseAuth.*instance*.currentUser!.uid);**

**setState(() {});**

**} catch (e) {**

**showSnackBar(**

**context,**

**e.toString(),**

**);**

**}**

**setState(() {**

**isLoading = false;**

**});**

**}**

**@override**

**Widget build(BuildContext context) {**

**return isLoading**

**? const Center(**

**child: CircularProgressIndicator(),**

**)**

**: Scaffold(**

**appBar: AppBar(**

**backgroundColor: mobileBackgroundColor,**

**title: Text(**

**userData['username'],**

**),**

**centerTitle: false,**

**),**

**body: ListView(**

**children: [**

**Padding(**

**padding: const EdgeInsets.all(16),**

**child: Column(**

**children: [**

**Row(**

**children: [**

**CircleAvatar(**

**backgroundColor: Colors.*grey*,**

**backgroundImage: NetworkImage(**

**userData['photoUrl'],**

**),**

**radius: 40,**

**),**

**Expanded(**

**flex: 1,**

**child: Column(**

**children: [**

**Row(**

**mainAxisSize: MainAxisSize.max,**

**mainAxisAlignment:**

**MainAxisAlignment.spaceEvenly,**

**children: [**

**buildStatColumn(postLen, "posts"),**

**buildStatColumn(followers, "followers"),**

**buildStatColumn(following, "following"),**

**],**

**),**

**Row(**

**mainAxisAlignment:**

**MainAxisAlignment.spaceEvenly,**

**children: [**

**FirebaseAuth.*instance*.currentUser!.uid ==**

**widget.uid**

**? FollowButton(**

**text: 'Sign Out',**

**backgroundColor:**

**mobileBackgroundColor,**

**textColor: primaryColor,**

**borderColor: Colors.*grey*,**

**function: () async {**

**await AuthMethods().signOut();**

**Navigator.*of*(context)**

**.pushReplacement(**

**MaterialPageRoute(**

**builder: (context) =>**

**const LoginScreen(),**

**),**

**);**

**},**

**)**

**: isFollowing**

**? FollowButton(**

**text: 'Unfollow',**

**backgroundColor: Colors.*white*,**

**textColor: Colors.*black*,**

**borderColor: Colors.*grey*,**

**function: () async {**

**await FireStoreMethods()**

**.followUser(**

**FirebaseAuth.*instance***

**.currentUser!.uid,**

**userData['uid'],**

**);**

**setState(() {**

**isFollowing = false;**

**followers--;**

**});**

**},**

**)**

**: FollowButton(**

**text: 'Follow',**

**backgroundColor: Colors.*blue*,**

**textColor: Colors.*white*,**

**borderColor: Colors.*blue*,**

**function: () async {**

**await FireStoreMethods()**

**.followUser(**

**FirebaseAuth.*instance***

**.currentUser!.uid,**

**userData['uid'],**

**);**

**setState(() {**

**isFollowing = true;**

**followers++;**

**});**

**},**

**)**

**],**

**),**

**],**

**),**

**),**

**],**

**),**

**Container(**

**alignment: Alignment.*centerLeft*,**

**padding: const EdgeInsets.only(**

**top: 15,**

**),**

**child: Text(**

**userData['username'],**

**style: TextStyle(**

**fontWeight: FontWeight.*bold*,**

**),**

**),**

**),**

**Container(**

**alignment: Alignment.*centerLeft*,**

**padding: const EdgeInsets.only(**

**top: 1,**

**),**

**child: Text(**

**userData['bio'],**

**),**

**),**

**],**

**),**

**),**

**const Divider(),**

**FutureBuilder(**

**future: FirebaseFirestore.*instance***

**.collection('posts')**

**.where('uid', isEqualTo: widget.uid)**

**.get(),**

**builder: (context, snapshot) {**

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

**return const Center(**

**child: CircularProgressIndicator(),**

**);**

**}**

**return GridView.builder(**

**shrinkWrap: true,**

**itemCount: (snapshot.data! as dynamic).docs.length,**

**gridDelegate:**

**const SliverGridDelegateWithFixedCrossAxisCount(**

**crossAxisCount: 3,**

**crossAxisSpacing: 5,**

**mainAxisSpacing: 1.5,**

**childAspectRatio: 1,**

**),**

**itemBuilder: (context, index) {**

**DocumentSnapshot snap =**

**(snapshot.data! as dynamic).docs[index];**

**return Container(**

**child: Image(**

**image: NetworkImage(snap['postUrl']),**

**fit: BoxFit.cover,**

**),**

**);**

**},**

**);**

**},**

**)**

**],**

**),**

**);**

**}**

**Column buildStatColumn(int num, String label) {**

**return Column(**

**mainAxisSize: MainAxisSize.min,**

**mainAxisAlignment: MainAxisAlignment.center,**

**children: [**

**Text(**

**num.toString(),**

**style: const TextStyle(**

**fontSize: 18,**

**fontWeight: FontWeight.*bold*,**

**),**

**),**

**Container(**

**margin: const EdgeInsets.only(top: 4),**

**child: Text(**

**label,**

**style: const TextStyle(**

**fontSize: 15,**

**fontWeight: FontWeight.*w400*,**

**color: Colors.*grey*,**

**),**

**),**

**),**

**],**

**);**

**}**

**}**

***pubspec.yaml***

**name: instagram\_clone\_flutter**

**description: A new Flutter project.**

**publish\_to: 'none' *# Remove this line if you wish to publish to pub.dev***

**version: 1.0.0+1**

**environment:**

**sdk: ">=2.15.0 <3.0.0"**

**dependencies:**

**cloud\_firestore: ^3.1.4**

**cupertino\_icons: ^1.0.2**

**firebase\_auth: ^3.3.3**

**firebase\_core: ^1.10.5**

**firebase\_storage: ^10.2.3**

**flutter:**

**sdk: flutter**

**flutter\_staggered\_grid\_view: ^0.4.1**

**flutter\_svg: ^1.0.0**

**image\_picker: ^0.8.4+4**

**intl: ^0.17.0**

**provider: ^6.0.1**

**uuid: ^3.0.5**

**dev\_dependencies:**

**flutter\_lints: ^1.0.0**

**flutter\_test:**

**sdk: flutter**

**flutter:**

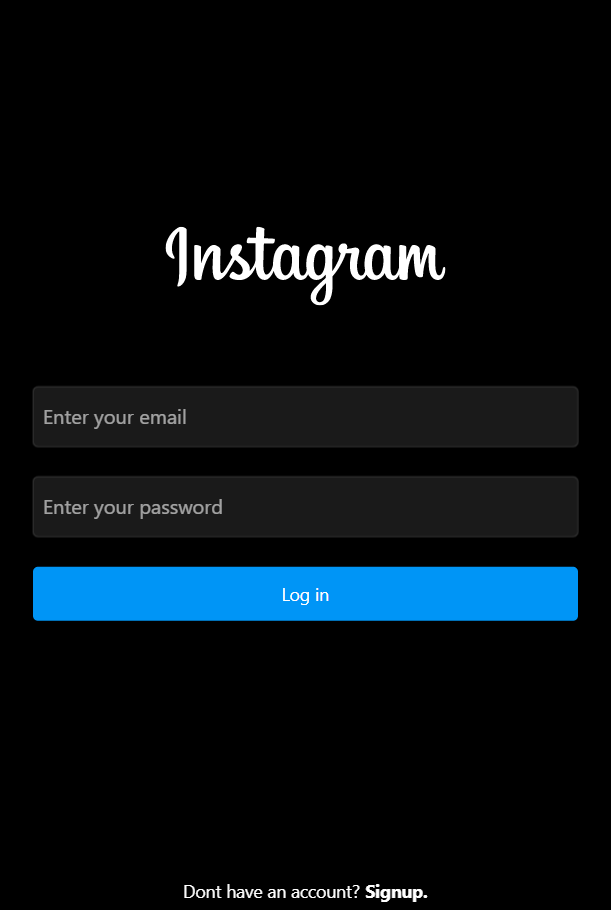
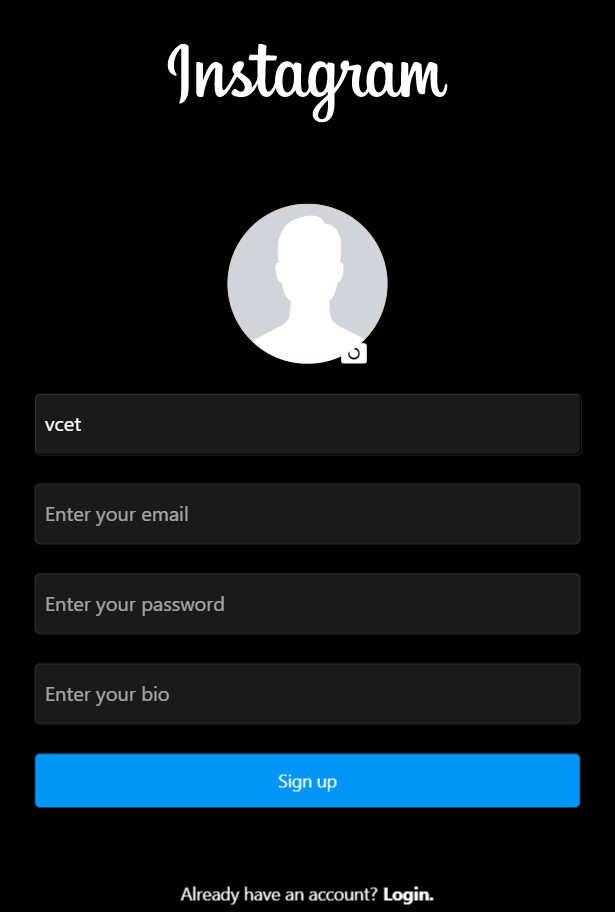
**uses-material-design: true**

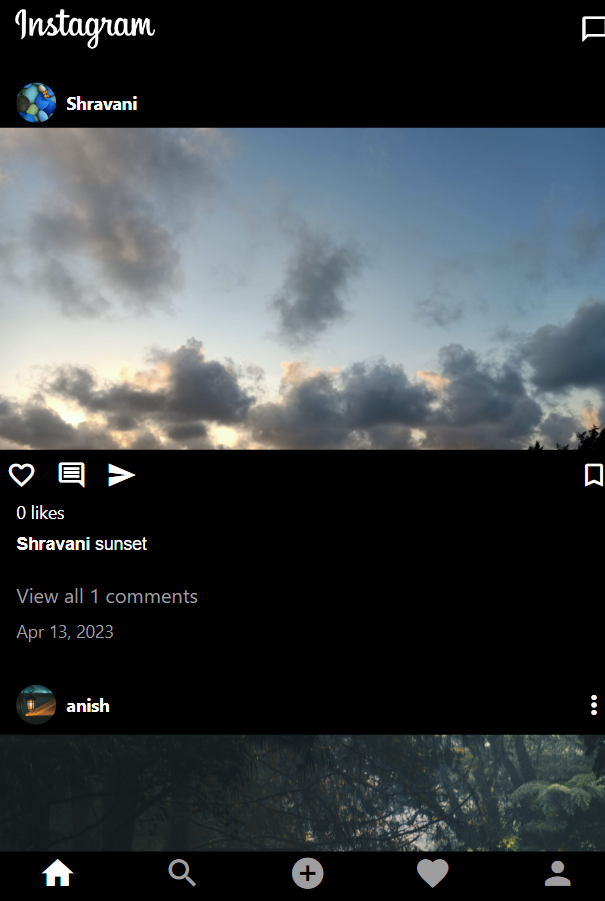
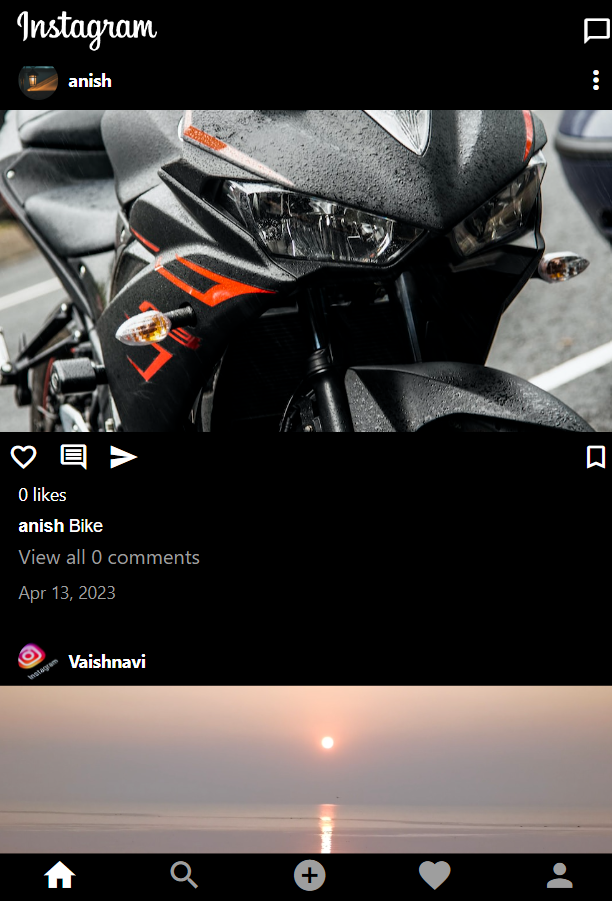
***# To add assets to your application, add an assets section, like this:***

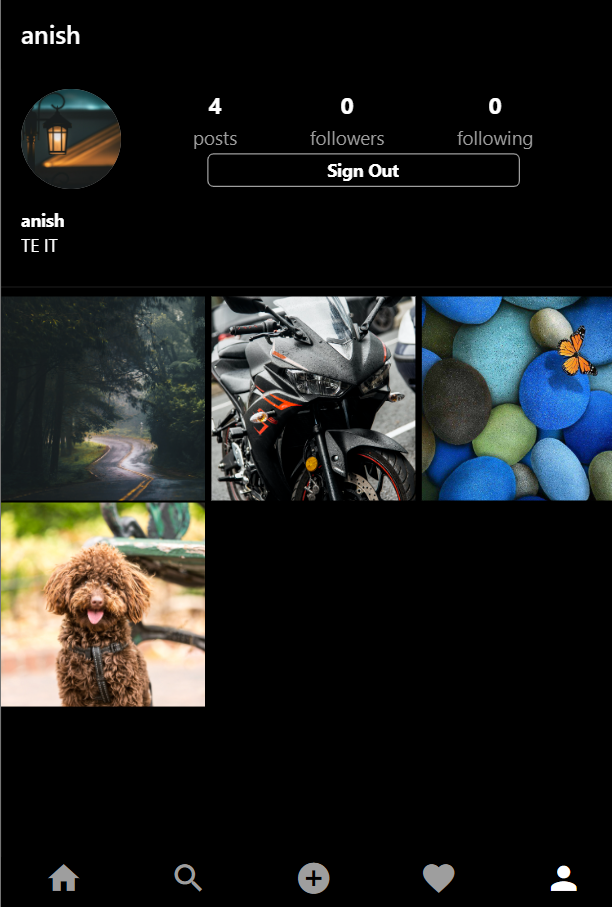
**assets:**

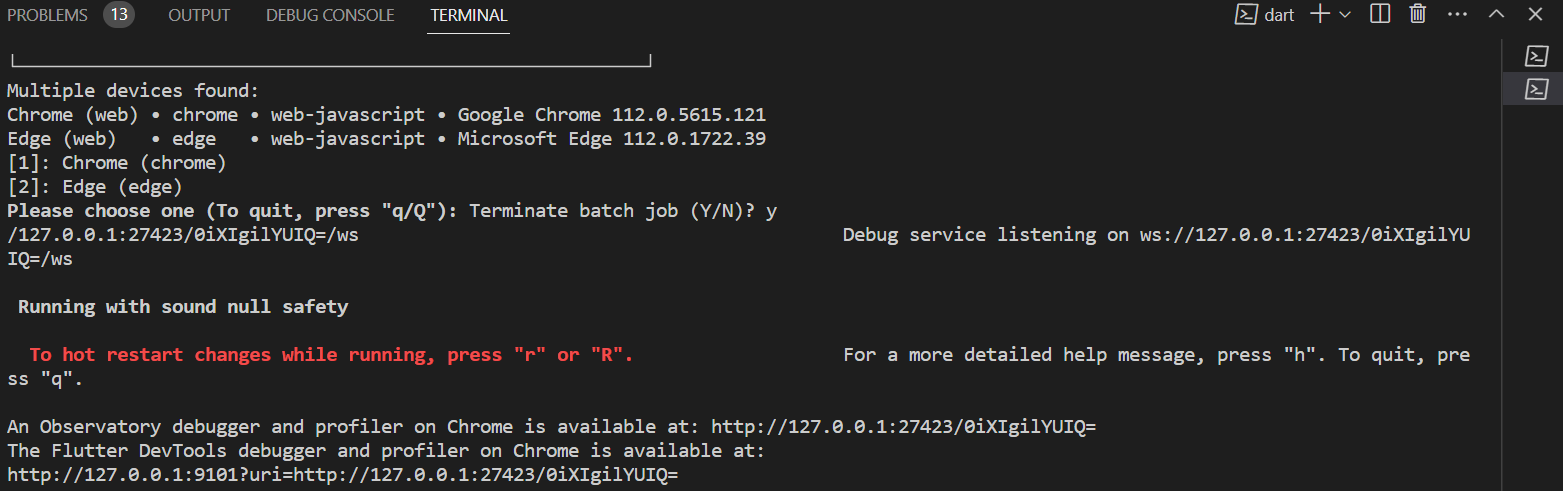
**- assets/ic\_instagram.svg**

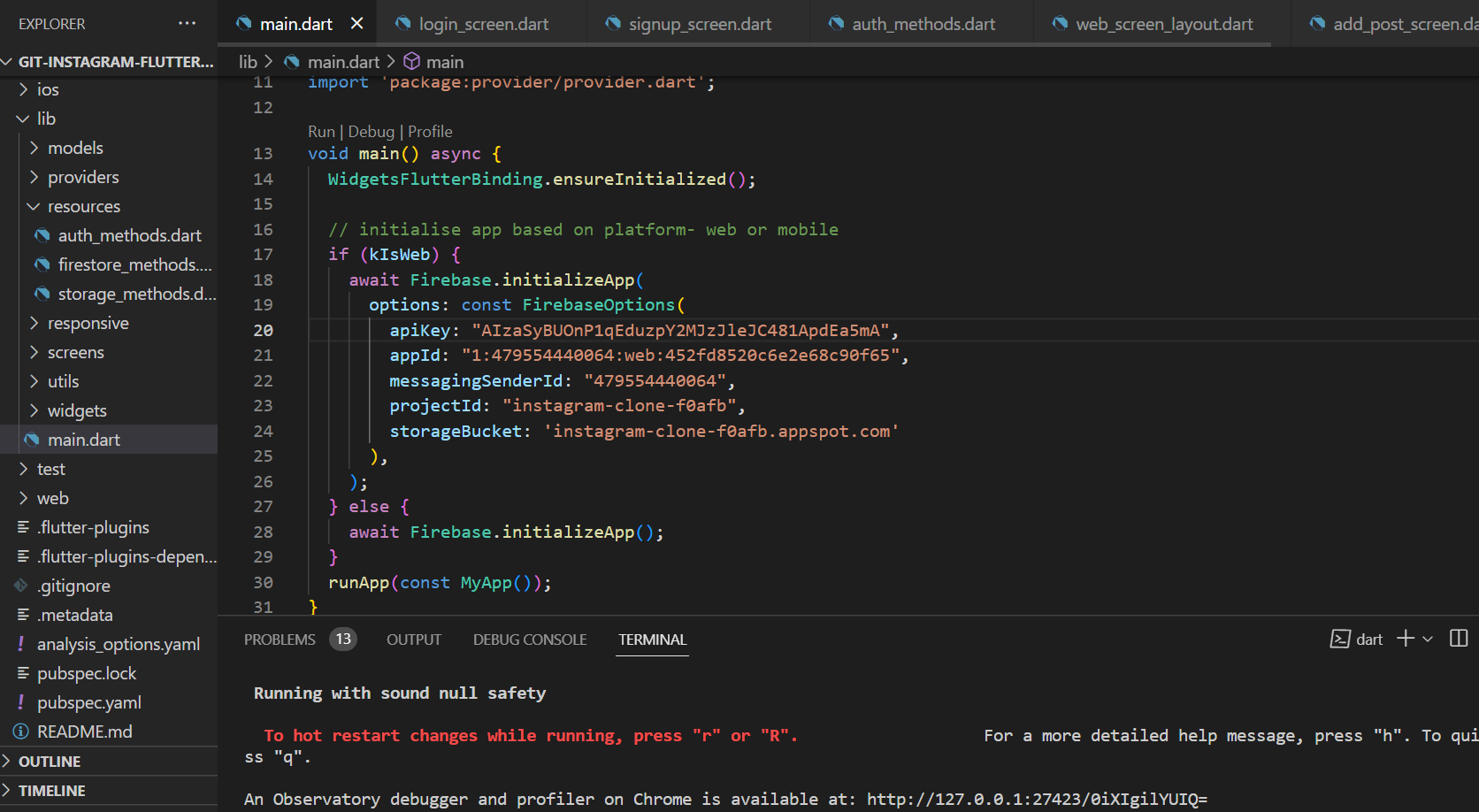
**Output**:

** **

** **

** **

****

****