Vivekanand Education Society's Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai
Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

Department of Information Technology

CERTIFICATE

This is to certify that **Anuprita Mhapankar** of **D15A** semester **VI,** have successfully completed necessary experiments in the **MAD & PWA Lab** under my supervision in **VES Institute of Technology** during the academic year **2024-2025.**

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Project Title: DMart **Roll No.28**

Name of the Course : MAD & PWA Lab **Course Code :** ITL604

Year/Sem/Class : D15A/D15B **A.Y.:** 24-25

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers : Mrs. Kajal Joseph.

Email : kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

PO1) Basic Engineering knowledge: An ability to apply the fundamental knowledge in mathematics, science and engineering to solve problems in Computer engineering.

PO2) Problem Analysis: Identify, formulate, research literature and analyze computer engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and computer engineering and sciences.

PO3) Design/ Development of Solutions: Design solutions for complex computer engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.

PO4) Conduct investigations of complex engineering problems using research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of information to provide valid conclusions.

PO5) Modern Tool Usage: Create, select and apply appropriate techniques, resources and modern computer engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6) The Engineer and Society: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to computer engineering practice.

PO7) Environment and Sustainability: Understand the impact of professional computer engineering solutions in societal and environmental contexts and demonstrate knowledge of and need for sustainable development.

PO8) Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of computer engineering practice.

PO9) Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams 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 computer engineering and 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.

PO12) 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.

Program specific Outcomes

PSO1) An ability to manage and analyze data / information effectively for making better decisions.

PSO2) Demonstrate the ability to use state of the art technologies and tools including Free and Open Source Software (FOSS) tools in developing software.

Project Title: DMart **Roll No.28**

Lab Objectives:

Sr. No.	Lab Objectives
The Lab experiments aims:	
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.

Lab Outcomes:

Sr. No.	Lab Outcomes	Cognitive levels of attainment as per Bloom's Taxonomy
On Completion of the course the learner/student should be able to:		
1	Understand cross platform mobile application development using Flutter framework	L1, L2
2	Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation	L3
3	Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS	L3, L4
4	Understand various PWA frameworks and their requirements	L1, L2
5	Design and Develop a responsive User Interface by applying PWA Design techniques	L3
6	Develop and Analyse PWA Features and deploy it over app hosting solutions	L3, L4

Index

Sr. No	Experiment Title	LO	DOP	DOS	Grade
1.	To install and configure the Flutter Environment	LO1			15
2.	To design Flutter UI by including common widgets.	LO2			15
3.	To include icons, images, fonts in Flutter app	LO2			14
4.	To create an interactive Form using form widget	LO2			14
5.	To apply navigation, routing and gestures in Flutter App	LO2			14
6.	To Connect Flutter UI with fireBase database	LO3			14
7.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4			
8.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA	LO5			
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5			
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5			
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6			
12.	Assignment-1	LO1,LO2 ,LO3			4
13.	Assignment-2	LO4,LO5 ,LO6			5

MAD & PWA Lab
Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	28
Name	Anuprita Mhapankar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15

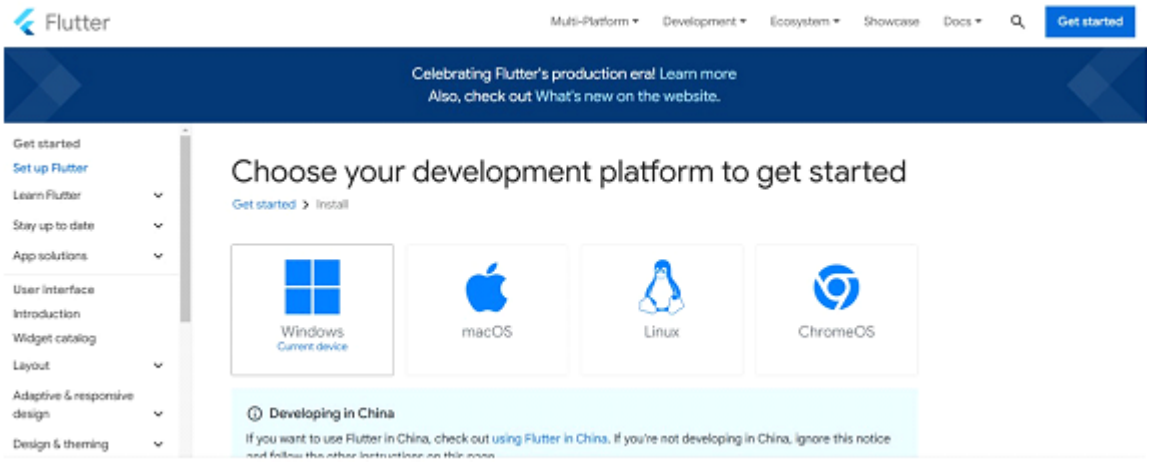
MPL Experiment 1

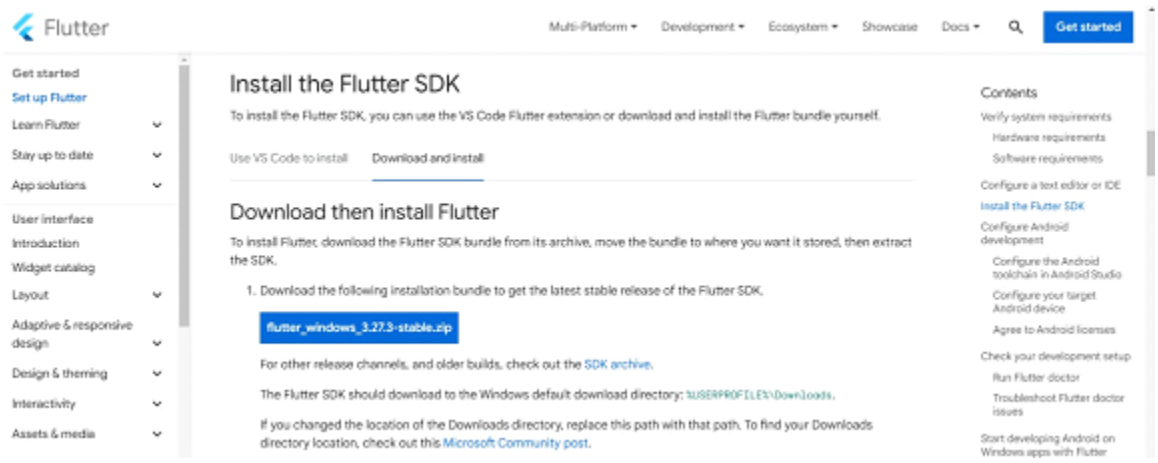
Name: Anuprita Mhapankar **Class:** D15A **Roll no:**28

Aim: Installation and Configuration of Flutter Environment.

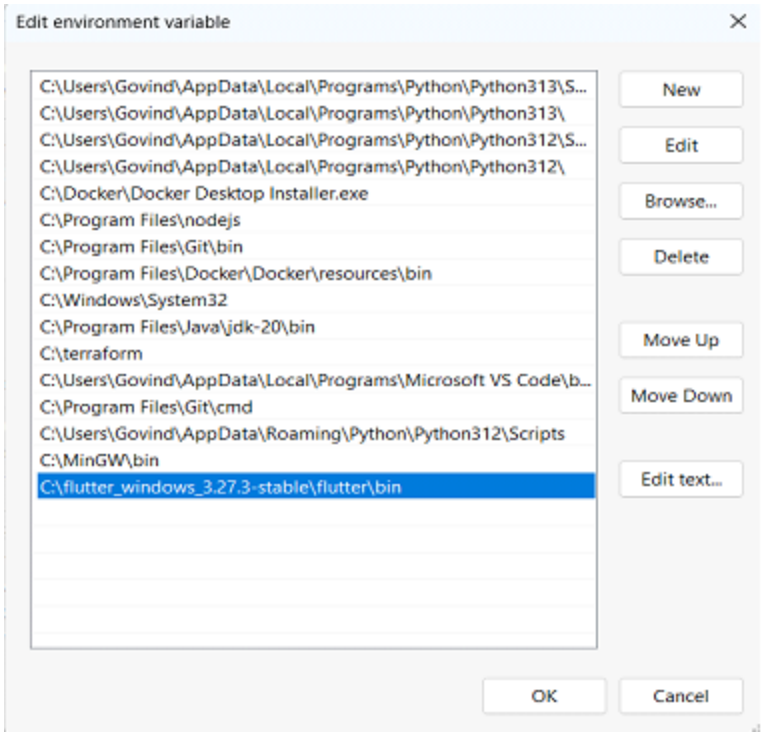
Step 1: Install Flutter

- Download Flutter SDK from the official Flutter website (https://flutter.dev/).
- Download the Flutter SDK for your operating system (Windows, macOS, or Linux).





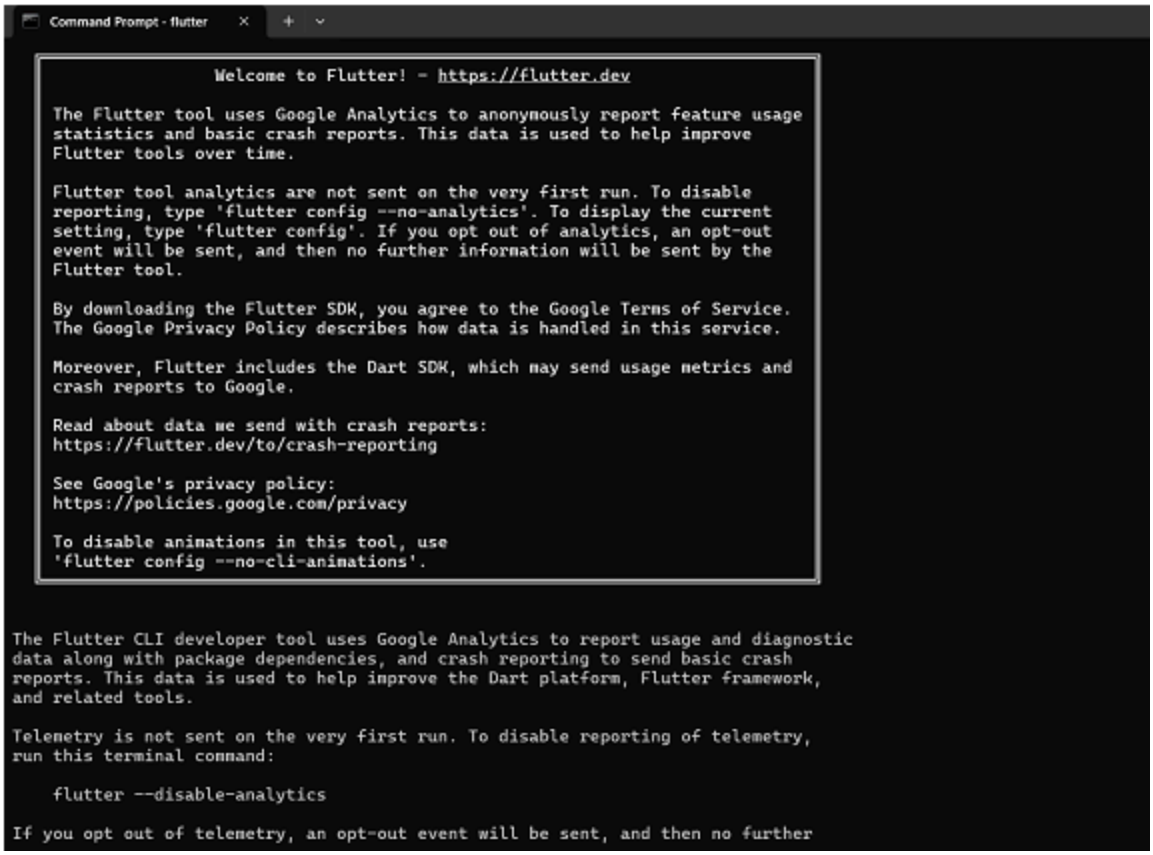
- Extract the downloaded zip file to a preferred location on your computer (e.g., C:\src\flutter for Windows).
- Add Flutter to the PATH
- Locate the flutter\bin directory in the extracted Flutter folder.
- Add this directory to your system's PATH environment variable.



Verify the Installation

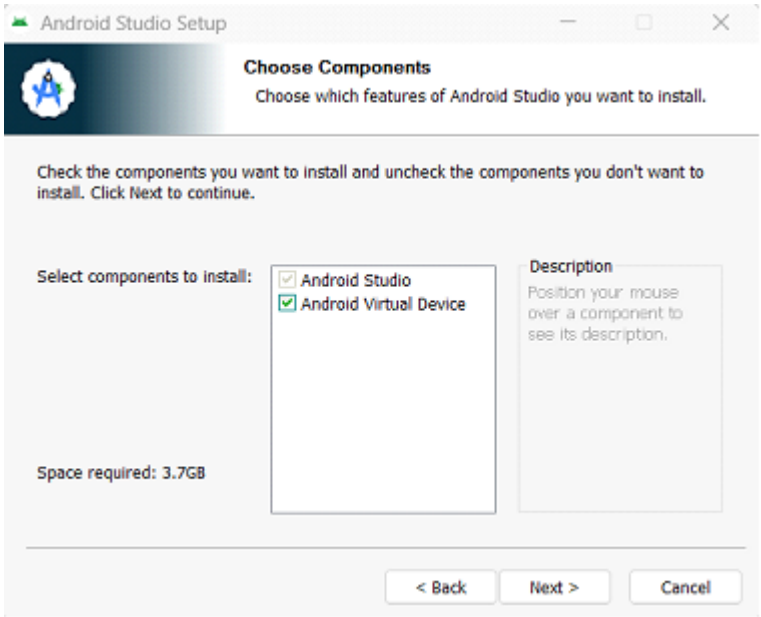
- Open a terminal or command prompt.
- Run the command: **flutter** and **flutter doctor**.

```
Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.27.1, on Microsoft Windows [Version 10.0.22631.4751], locale en-IN)
[X] Windows Version (the doctor check crashed)
    X Due to an error, the doctor check did not complete. If the error message below is not helpful, please let us know
      about this issue at https://github.com/flutter/flutter/issues.
    X ProcessException: Failed to find "powershell" in the search path.
        Command: powershell
[√] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[√] Chrome - develop for the web
[X] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows apps.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[√] Android Studio (version 2023.3)
[√] VS Code (version 1.96.4)
[√] Connected device (4 available)
[√] Network resources

! Doctor found issues in 2 categories.

C:\Users\User>
```

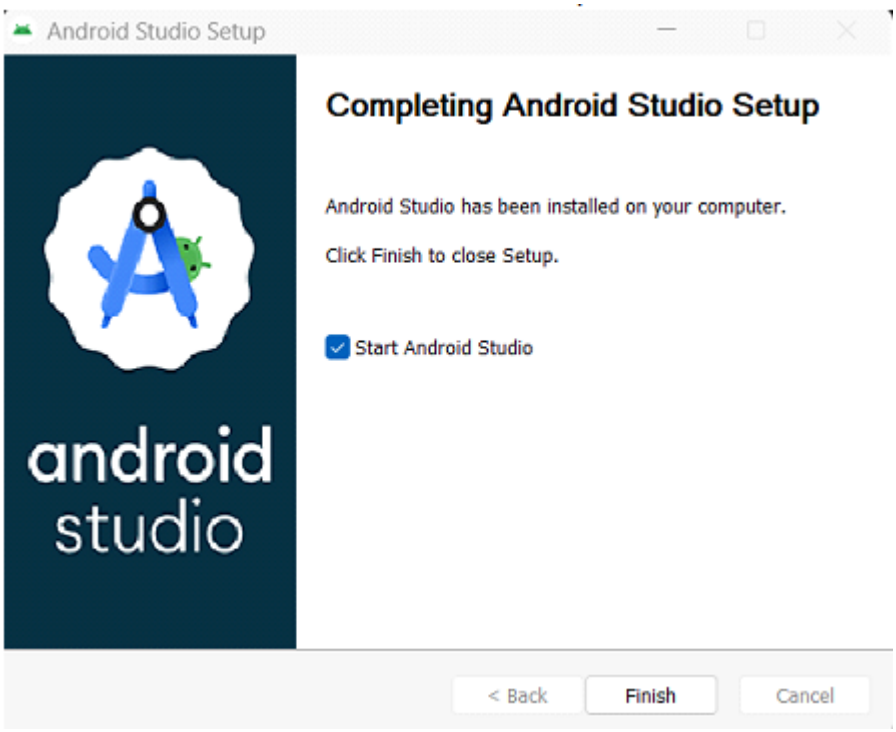


Step 2: Install Android Studio

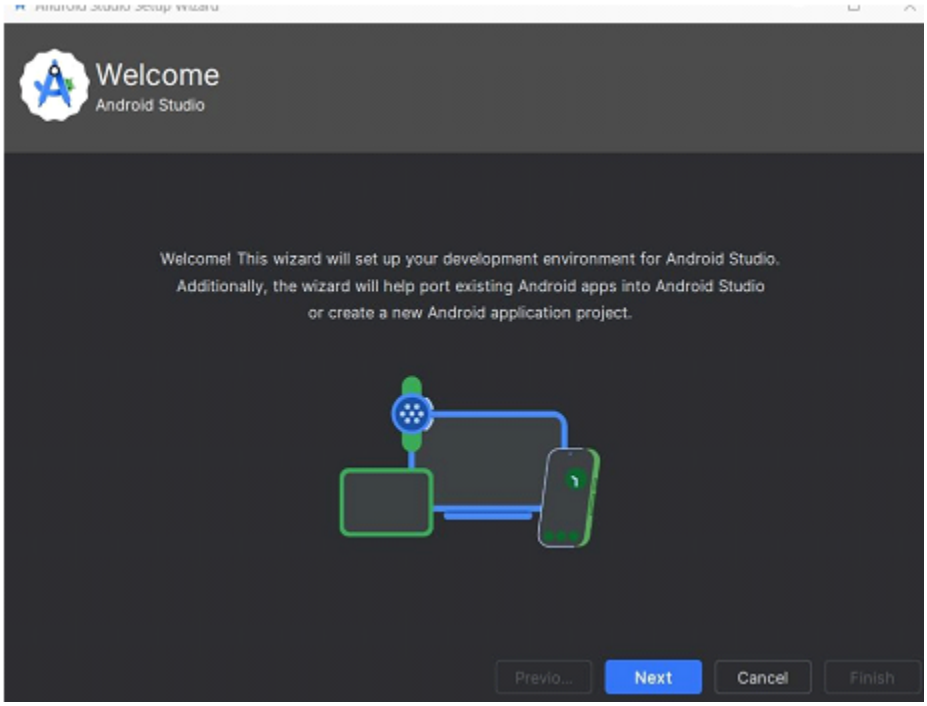
- Download Android Studio
- Go to the Android Studio website. (https://developer.android.com/studio)
- Download the installer for your operating system ((Windows, macOS, or Linux)).



- Run the installer and follow the setup wizard.
- Choose the standard installation option.

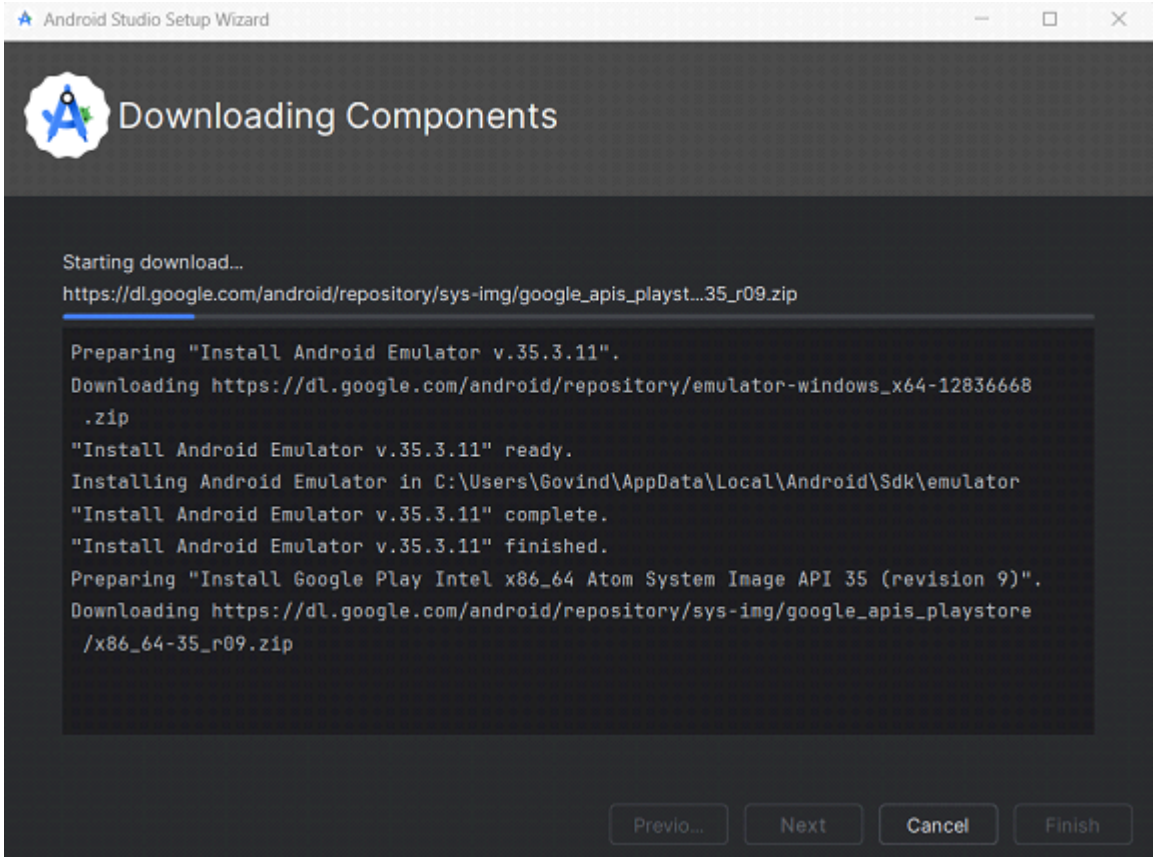


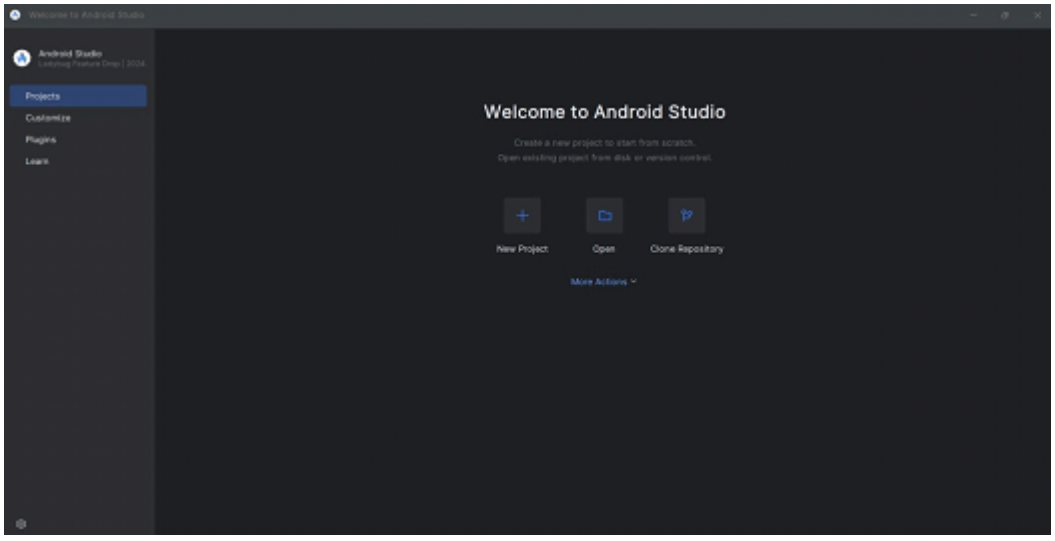
- Install Android SDK Tools
- Open Android Studio.



- Go to Settings/Preferences > Appearance & Behavior > System Settings > Android SDK.

- Select the latest Android API level.
- Ensure "Android SDK Platform" and "Android Virtual Device (AVD)" are selected.
- Click "Apply" and wait for the components to install.

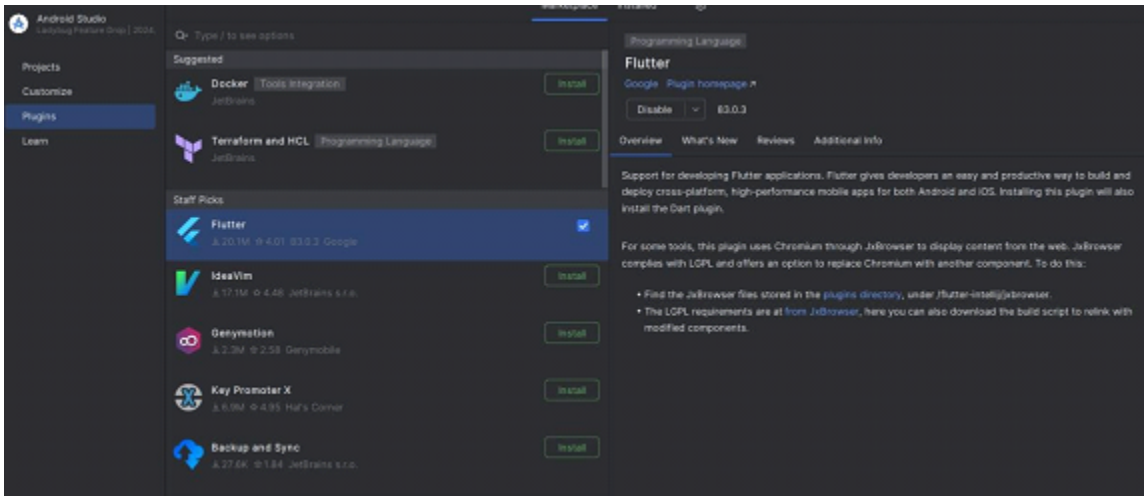




Step 3: Connect Flutter with Android Studio

- Install Flutter and Dart Plugins
- Open Android Studio. Go to File > Settings (Windows/Linux) > Plugins.

- Search for "Flutter" and click "Install." Dart will be installed automatically.
- Restart Android Studio.



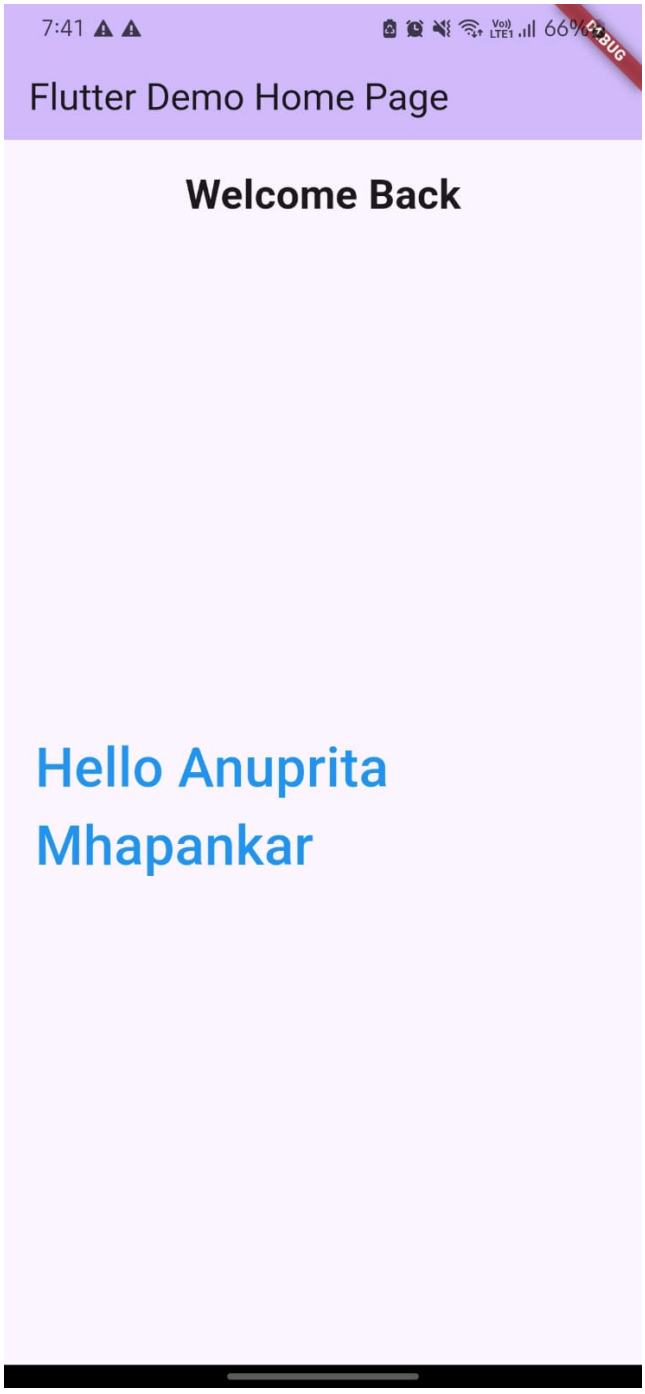
Step 4: Create a New Flutter Project

- Click on New Flutter Project.
- Enter project details and select the Flutter SDK path.
- Click "Finish" to create the project.
- Connect the USB to the device and run the flutter application.
  NOTE: In your mobile device, make sure the USB debugging is turned on.

Code:

```
class _MyHomePageState extends State<MyHomePage> {
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       backgroundColor: Theme.of(context).colorScheme.inversePrimary,
       title: Text(widget.title),
     ),
     body: Column(
       children: [
         const Padding(
           padding: EdgeInsets.all(16.0),
           child: Text(
             'Welcome Back',
             style: TextStyle(
               fontSize: 24,
               fontWeight: FontWeight.bold,
             ),
           ),
         ),
         const Spacer(),
```

```
        const Padding(
          padding: EdgeInsets.all(20.0),
          child: Text(
            'Hello Anuprita Mhapankar',
            style: TextStyle(
              fontSize: 32,
              color: Colors.blue,
              fontWeight: FontWeight.w500,
            ),
          )
        ),
        const Spacer(),
      ],
    ),
  );
 }
}
```



Conclusion: Hello World, is successfully run on the flutter app.

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	28
Name	Anuprita Mhapankar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

MPL Experiment 2

Name: Anuprita Mhapankar **Class:** D15A **Roll no:**28

Aim: To design Flutter UI by including common widgets.

Theory:

Designing a Flutter UI involves using common widgets effectively to create a structured and interactive interface. Layout widgets like `Container`, `Column`, `Row`, and `GridView` help organize elements, while interactive widgets like `GestureDetector`, `InkWell`, and `TextField` enhance user engagement.

For displaying content, widgets like `Text`, `Image`, `Card`, and `ListView` are essential. Navigation widgets such as `BottomNavigationBar`, `Drawer`, and `Navigator` enable seamless movement between screens. Managing state using `setState`, `Provider`, or advanced solutions like `Riverpod` ensures smooth UI updates.

For a **Dmart-like home screen**, a combination of the following widgets can be used:

- `ListView` with a `CarouselSlider` for displaying promotional banners.
- `GridView` for categorizing products.
- `Card` widget to showcase trending items.
- `BottomNavigationBar` for consistent navigation between screens.

Steps:

Step 1: Create a new Flutter project or open an existing one.

Step 2: Design the layout using `Scaffold`, incorporating `AppBar` and `BottomNavigationBar`.

Step 3: Use `CarouselSlider` inside a `ListView` for banners.

Step 4: Implement `GridView` to display product categories.

Step 5: Use `Card` widgets inside a `ListView` to show trending items.

Step 6: Ensure navigation using `BottomNavigationBar` and `Navigator`.

Step 7: Apply `setState` or `Provider` for state management.

Code:

```
//home_page.dart
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:dmart/screens/login/login_page.dart';
import 'package:carousel_slider/carousel_slider.dart';
import 'package:dmart/common_components/header.dart';

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

  Future<void> _logout(BuildContext context) async {
    await FirebaseAuth.instance.signOut();

    // Clear login state from shared preferences
    SharedPreferences prefs = await SharedPreferences.getInstance();
    await prefs.setBool('isLoggedIn', false);

    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (context) => const LoginPage()),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: const Header(),
      body: SingleChildScrollView(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // Parade of Deals Banner
            Container(
              width: double.infinity,
              height: 150,
              padding: const EdgeInsets.all(16.0),
```

```
              alignment: Alignment.center,
              color: Colors.blueAccent,
              child: const Text(
                'Parade of Deals',
                style: TextStyle(fontSize: 24, fontWeight:
FontWeight.bold, color: Colors.white),
                textAlign: TextAlign.center,
              ),
            ),

            // Dmart Exclusive
            const Padding(
              padding: EdgeInsets.all(16.0),
              child: Text(
                'Dmart Exclusive',
                style: TextStyle(fontSize: 20, fontWeight:
FontWeight.bold),
              ),
            ),
            CarouselSlider(
              options: CarouselOptions(enableInfiniteScroll: false),
              items: List.generate(5, (index) {
                return Card(
                  margin: const EdgeInsets.symmetric(horizontal: 8),
                  child: Container(
                    height: 150,
                    width: double.infinity,
                    alignment: Alignment.center,
                    child: Text('Exclusive Item ${index + 1}'),
                  ),
                );
              }),
            ),

            // Best Offer
            const Padding(
              padding: EdgeInsets.all(16.0),
              child: Text(
```

```
                'Best Offer',
                style: TextStyle(fontSize: 20, fontWeight:
FontWeight.bold),
              ),
            ),
            CarouselSlider(
              options: CarouselOptions(enableInfiniteScroll: false),
              items: List.generate(5, (index) {
                return Card(
                  margin: const EdgeInsets.symmetric(horizontal: 8),
                  child: Container(
                    height: 150,
                    width: double.infinity,
                    alignment: Alignment.center,
                    child: Text('Best Offer Item ${index + 1}'),
                  ),
                );
              }),
            ),
          ],
        ),
      ),
    );
  }
}
```

```
//account_page.dart
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:dmart/screens/login/login_page.dart';
```

```
import 'package:dmart/common_components/header.dart';

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

  Future<void> _logout(BuildContext context) async {
    await FirebaseAuth.instance.signOut();

    // Clear login state from shared preferences
    SharedPreferences prefs = await SharedPreferences.getInstance();
    await prefs.setBool('isLoggedIn', false);

    // Navigate back to login screen
    Navigator.pushAndRemoveUntil(
      context,
      MaterialPageRoute(builder: (context) => const LoginPage()),
          (Route<dynamic> route) => false, // Remove all previous
routes
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: const Header(),
      body: Column(
        children: [
          Expanded(
            child: ListView(
              children: [
                _buildAccountOption(Icons.person, "My Profile"),
                _buildAccountOption(Icons.help, "Help @ DMart Ready"),
                _buildAccountOption(Icons.logout, "Sign Out",
isLogout: true, onTap: () => _logout(context)),
                const SizedBox(height: 10),
                _buildPromoBanner(),
              ],
            ),
```

```
          ),
        ],
      ),
    );
  }

  Widget _buildAccountOption(IconData icon, String title, {bool
isLogout = false, VoidCallback? onTap}) {
    return ListTile(
      leading: Icon(icon, color: isLogout ? Colors.red :
Colors.black54),
      title: Text(title, style: TextStyle(fontSize: 16, fontWeight:
FontWeight.w500)),
      trailing: const Icon(Icons.arrow_forward_ios, size: 16, color:
Colors.black54),
      onTap: onTap ?? () {}, // Provide navigation or functionality
    );
  }

  Widget _buildPromoBanner() {
    return Padding(
      padding: const EdgeInsets.all(12.0),
      child: Container(
        decoration: BoxDecoration(
          color: Colors.green.shade100,
          borderRadius: BorderRadius.circular(8),
        ),
        padding: const EdgeInsets.all(12),
        child: Row(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          children: [
            const Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text(
                  "You're SAVING a LOT on every order",
                  style: TextStyle(fontSize: 14, fontWeight:
FontWeight.bold),
```

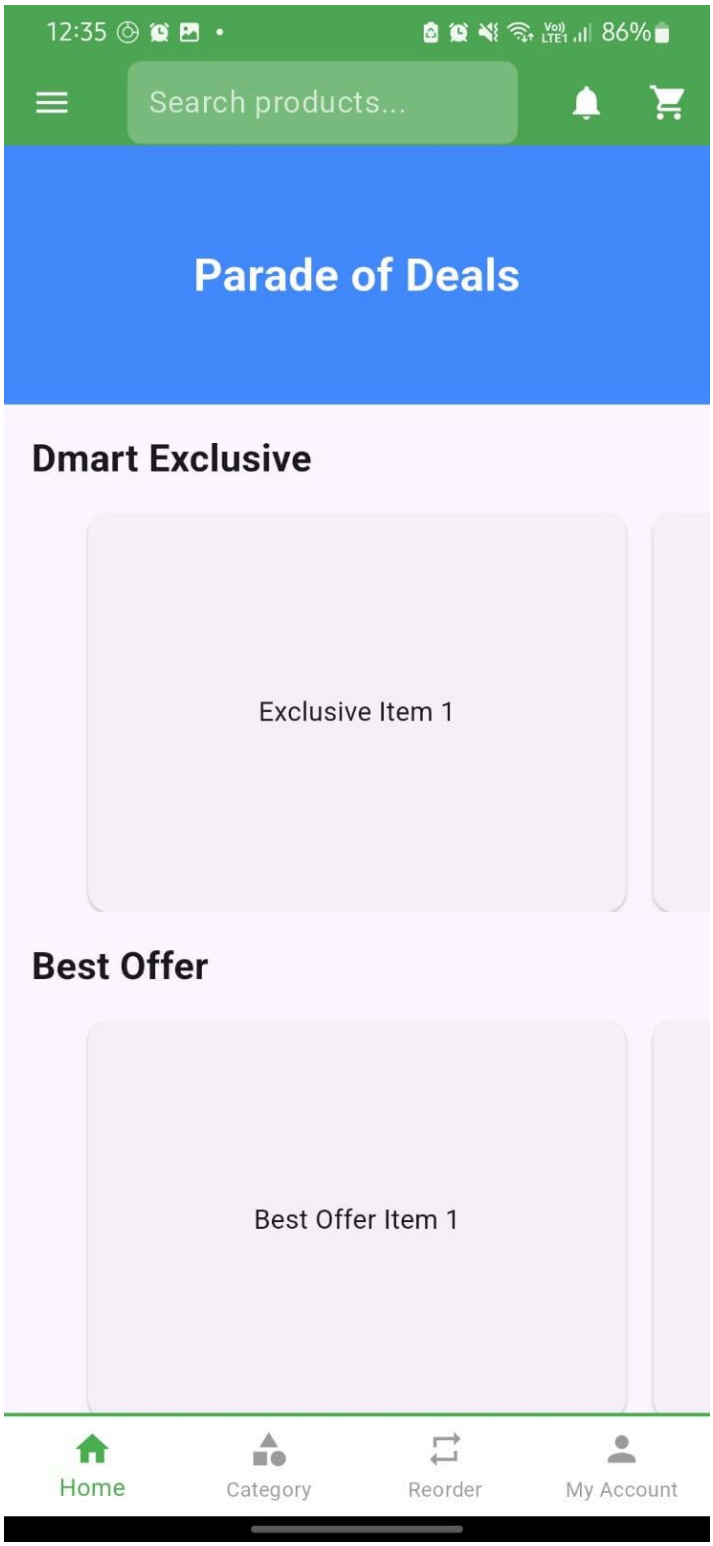
```
                ),
                SizedBox(height: 5),
                Text(
                  "Check My Savings",
                  style: TextStyle(fontSize: 14, fontWeight:
FontWeight.bold, color: Colors.green),
                ),
              ],
            )
          ],
        ),
      ),
    );
  }
}
```

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

class Sidebar extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
```

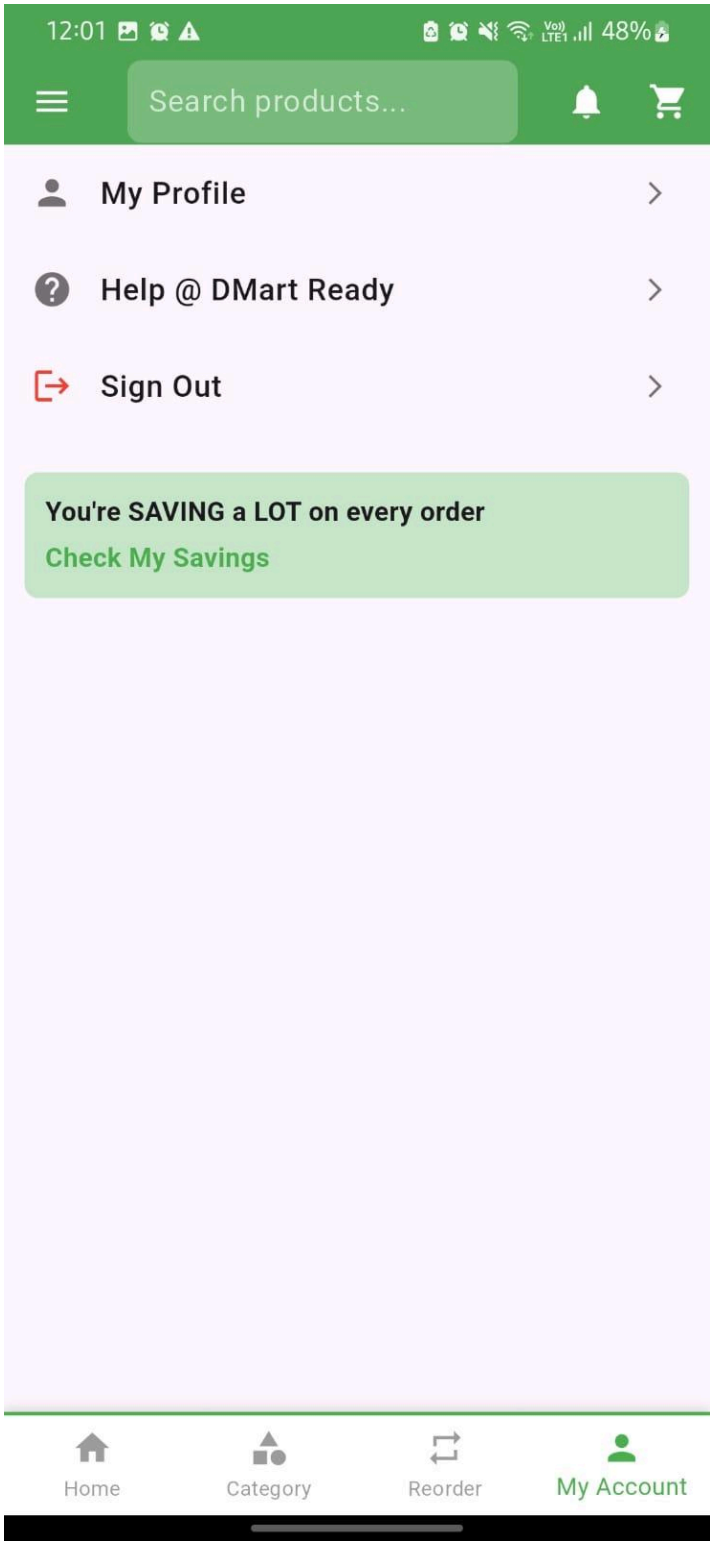
```
    return Drawer(
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          UserAccountsDrawerHeader(
            decoration: BoxDecoration(color: Colors.green),
            accountName: Text(
              "Hi, Anuprita",
              style: TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
            ),
            accountEmail: Text("400615, Thane (West)"),
            currentAccountPicture: CircleAvatar(
              backgroundColor: Colors.white,
              child: Icon(Icons.person, size: 40, color:
Colors.green),
            ),
          ),
          _buildMenuItem(Icons.category, "Shop by Category", context),
          _buildMenuItem(Icons.help, "Help @ Dmart Ready", context),
          _buildMenuItem(Icons.policy, "Refund, Terms and Policies",
context),
          _buildMenuItem(Icons.question_answer, "Frequently Asked
Questions", context),
          _buildMenuItem(Icons.info, "About Us", context),
          Spacer(),
          Padding(
            padding: const EdgeInsets.all(16.0),
            child: Text("Version 5.1.9", style: TextStyle(color:
Colors.grey)),
          ),
        ],
      ),
    );
  }

  Widget _buildMenuItem(IconData icon, String title, BuildContext
context) {
```

```
    return ListTile(
      leading: Icon(icon, color: Colors.green),
      title: Text(title, style: TextStyle(fontSize: 16)),
      onTap: () {
        Navigator.pop(context); // Close drawer when an item is tapped
      },
    );
  }
}
```

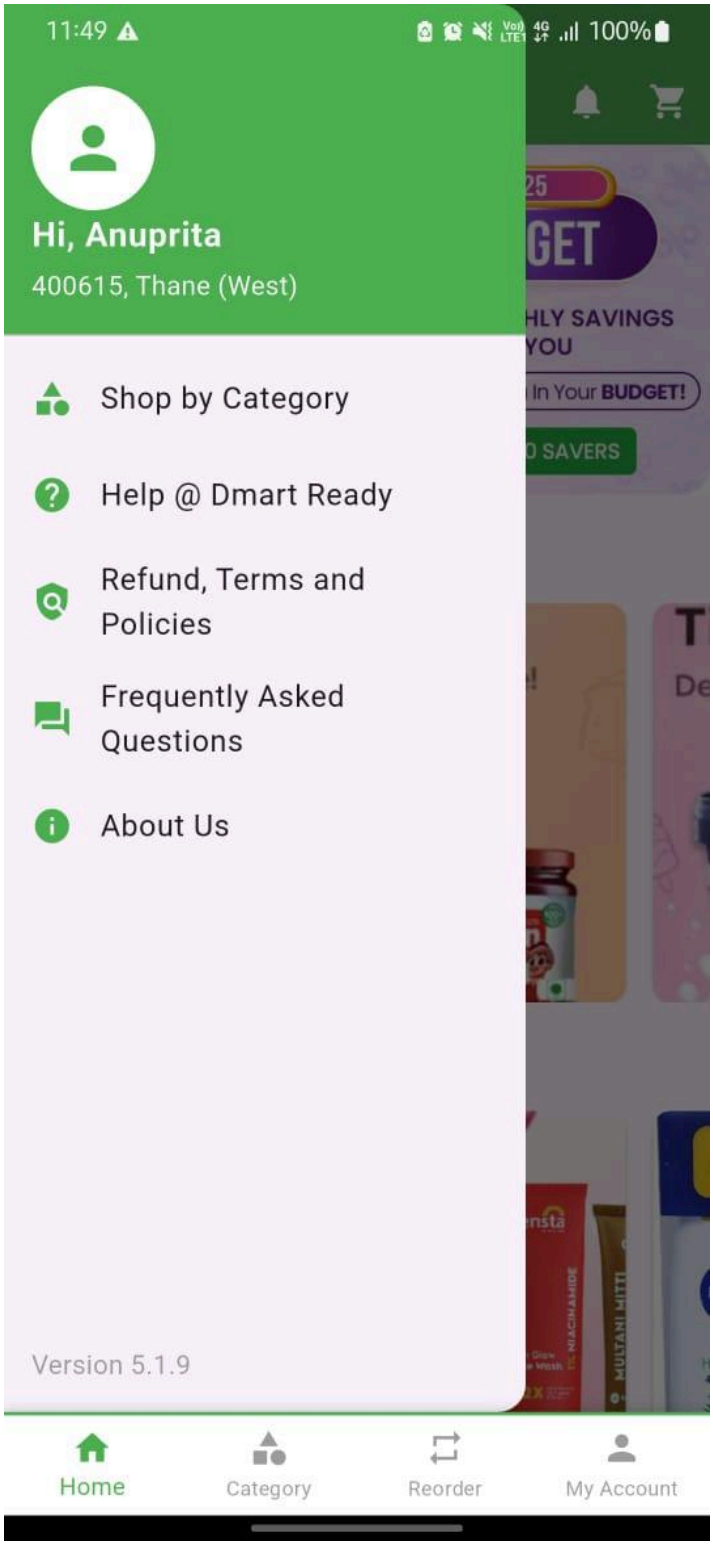
Output:
HomePage:



MyAccount:


12:01
48%
Search products...
My Profile
Help @ DMart Ready
Sign Out
You're SAVING a LOT on every order
Check My Savings
Home
Category
Reorder
My Account

Sidebar:


11:49
100%
Hi, Anuprita
400615, Thane (West)
Shop by Category
Help @ Dmart Ready
Refund, Terms and Policies
Frequently Asked Questions
About Us
Version 5.1.9
Home
Category
Reorder
My Account

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	28
Name	Anuprita Mhapankar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	14

MPL Experiment 3

Name: Anuprita Mhapankar **Class:** D15A **Roll no:**28

Aim: To include icons, images, Fonts in Flutter app.

Theory:

Incorporating icons, images, and custom fonts in a Flutter application enhances the visual appeal and improves the user experience. Flutter provides different ways to add these elements:

1. Icons:

Icons can be added using the built-in `Icons` class with the `Icon` widget. Custom icons can also be used via the `flutter_launcher_icons` package.

2. Images:

Images can be displayed in two ways:

- **From assets**: Images stored locally in the app's assets folder can be loaded using the `Image.asset()` method.
- **From the internet**: Images can be fetched dynamically from a URL using the `Image.network()` method.

Examples:

- **Loading an image from assets:**

```
Image.asset(
  'assets/images/sample.png',
  fit: BoxFit.cover,
  width: double.infinity,
)
```

- **Loading an image from a URL:**

```
Image.network(
  'https://example.com/sample.jpg',
```

```
      width: double.infinity,
      fit: BoxFit.cover,
    )
```

3. Fonts:

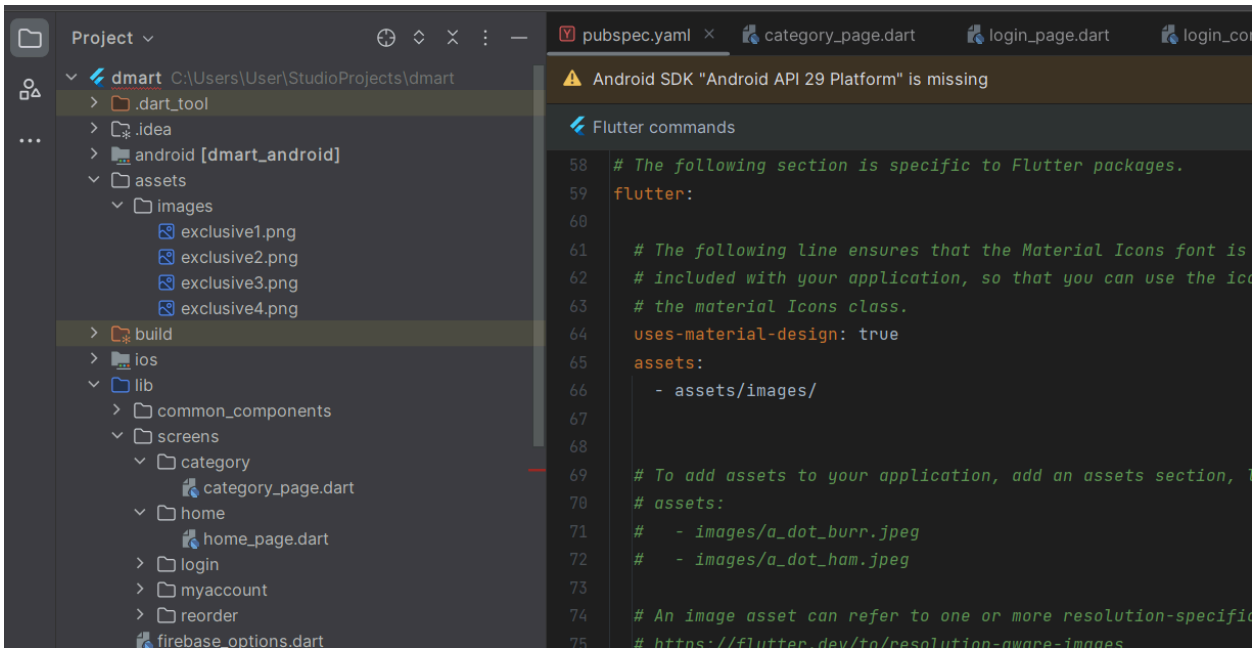
Custom fonts improve typography and branding. To add custom fonts, font files must be placed in the `assets/fonts/` directory and declared in `pubspec.yaml` under the `flutter` section. Using `TextStyle` with the `fontFamily` property applies the custom font to text elements.

Steps:

Step 1: Create an `assets` folder inside the project directory. Inside the `assets` folder, create an `images` folder and add the required images.

Step 2: Open `pubspec.yaml` and add the following under the `flutter` section:

```
flutter:
  assets:
    - assets/images/
```



Step 3: Run the following command in the terminal to apply the changes:

```
flutter pub get
```

Code:

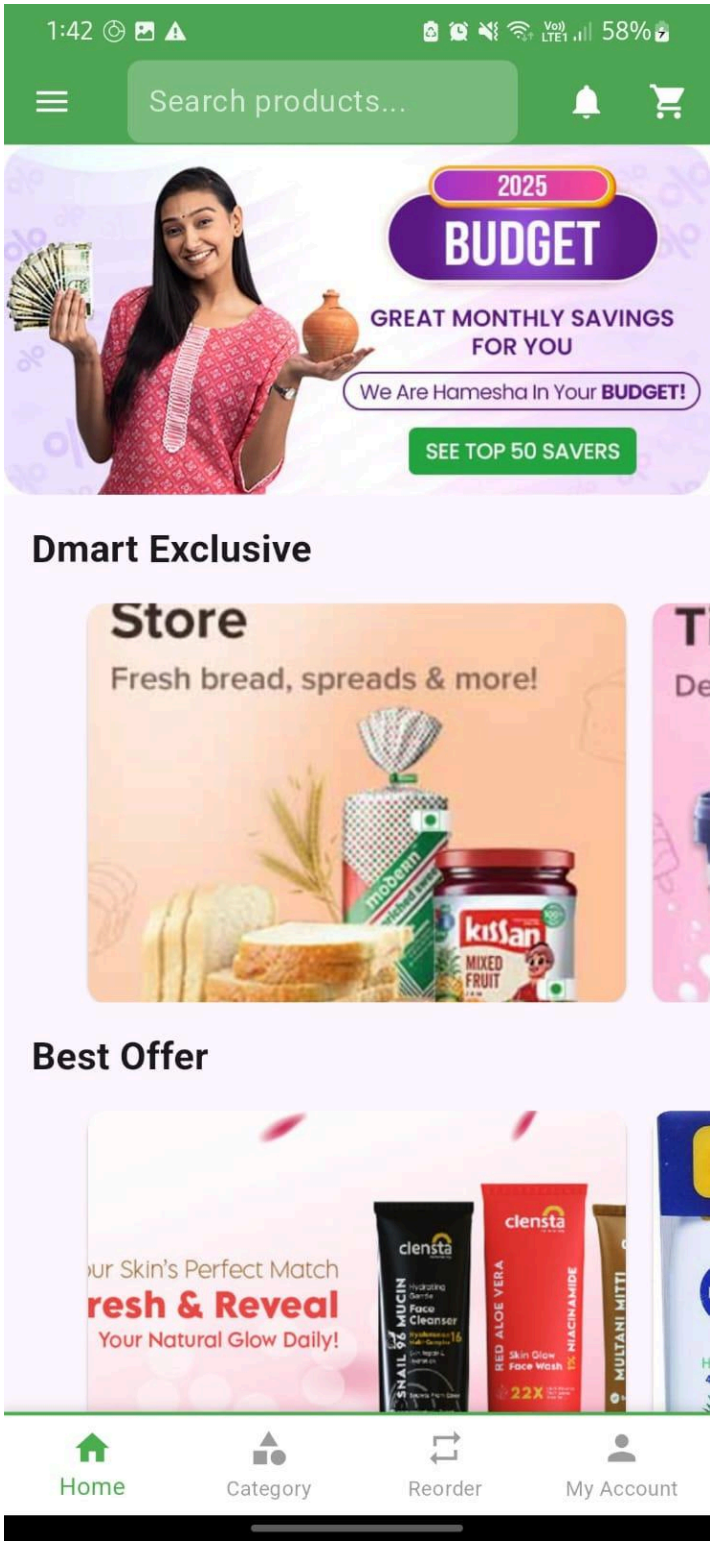
```
//home_page.dart
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:dmart/screens/login/login_page.dart';
import 'package:carousel_slider/carousel_slider.dart';
import 'package:dmart/common_components/header.dart';
import 'package:dmart/common_components/sidebar.dart';


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

  Future<void> _logout(BuildContext context) async {
    await FirebaseAuth.instance.signOut();

    // Clear login state from shared preferences
    SharedPreferences prefs = await SharedPreferences.getInstance();
    await prefs.setBool('isLoggedIn', false);

    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (context) => const LoginPage()),
    );
  }

  @override
  Widget build(BuildContext context) {
    // Image Lists
    final List<String> dmartExclusiveImages = [
      'assets/images/exclusive1.png',
      'assets/images/exclusive2.png',
      'assets/images/exclusive3.png',
      'assets/images/exclusive4.png',
    ];

    final List<String> bestOfferImages = [
```

```
'https://clensta.com/cdn/shop/files/mob-banner4_600x.jpg?v=1737626058'
,

'https://rukminim2.flixcart.com/fk-p-harbour/600/600/images/incom/imag
es/product/Nivea-Body-Lotion-Buy-3-Get-1-Free-Aloe-Hydration-2-x-75-ml
-Extra-Whitening-75-mlBody-Milk-75-ml-1601886438-10076795-1.jpg?q=60',
    ];
    return Scaffold(
      appBar: const Header(),
      drawer: Sidebar(),
      body: SingleChildScrollView(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // Parade of Deals Banner
            SizedBox(
              width: double.infinity,
              child: Image.network(

'https://cdn.dmart.in/images/rwd-mobile/banners/curated/29jan25-curate
d-budget-hp.jpg',
                fit: BoxFit.cover, // Ensures the image covers the
full space
              ),
            ),

            // Dmart Exclusive
            const Padding(
              padding: EdgeInsets.all(16.0),
              child: Text(
                'Dmart Exclusive',
                style: TextStyle(fontSize: 20, fontWeight:
FontWeight.bold),
              ),
            ),
            CarouselSlider(
              options: CarouselOptions(enableInfiniteScroll: false),
```

```
            items: dmartExclusiveImages.map((imagePath) {
              return Card(
                margin: const EdgeInsets.symmetric(horizontal: 8),
                child: ClipRRect(
                  borderRadius: BorderRadius.circular(10),
                  child: Image.asset(imagePath, fit: BoxFit.cover,
width: double.infinity),
                ),
              );
            }).toList(),
          ),

          // Best Offer
          const Padding(
            padding: EdgeInsets.all(16.0),
            child: Text(
              'Best Offer',
              style: TextStyle(fontSize: 20, fontWeight:
FontWeight.bold),
            ),
          ),
          CarouselSlider(
            options: CarouselOptions(enableInfiniteScroll: false),
            items: bestOfferImages.map((imageUrl) {
              return Card(
                margin: const EdgeInsets.symmetric(horizontal: 8),
                child: ClipRRect(
                  borderRadius: BorderRadius.circular(10),
                  child: Image.network(
                    imageUrl,
                    width: double.infinity,
                    fit: BoxFit.cover,
                  ),
                ),
              );
            }).toList(),
          ),
        ],
```

```
          ),
        ),
      );
    }
  }
```

Output:


1:42
58%
Search products...
2025
BUDGET
GREAT MONTHLY SAVINGS FOR YOU
We Are Hamesha In Your BUDGET!
SEE TOP 50 SAVERS
Dmart Exclusive
Store
Fresh bread, spreads & more!
kissan
MIXED FRUIT
Best Offer
clensta
Home
Category
Reorder
My Account

7:24 100%

Shop By Category

- Dairy & Beverages
- Electronics & Appliances
- Grocery
- Home & Kitchen
- Packaged Foods
- Sports & Fitness

Juices

Soft Drinks

Milk & Dairy

Tea & Coffee

Home | Category | Reorder | My Account

7:44 100%

Cooking Oil

Fortune Sunlite Sunflower Oil

Qty: 910 g

DMart: ₹150

~~MRP: ₹190~~

₹40 OFF

ADD

Gemini Refined Sunflower Oil

Qty: 4.550 kg

DMart: ₹782

~~MRP: ₹990~~

₹208 OFF

ADD

Dhara Refined Sunflower Oil

Qty: 13.65 kg

DMart: ₹2219

~~MRP: ₹2970~~

₹751 OFF

ADD

Dalda Refined Sunflower Oil

Qty: 910 g

DMart: ₹142

~~MRP: ₹185~~

₹43 OFF

ADD

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	28
Name	Anuprita Mhapankar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	14

MPL Experiment 4

Name: Anuprita Mhapankar **Class:** D15A **Roll no:**28

Aim: To create an interactive Form using form widget

Theory:

Creating an interactive form in Flutter requires using form-related widgets to collect and validate user input efficiently. The `Form` widget, combined with `TextFormField`, provides a structured way to manage input fields. Various input widgets like `TextField`, `DropdownButton`, `Checkbox`, `Radio`, and `Switch` allow users to enter data in different formats.

Validation and state management can be handled using the `GlobalKey<FormState>` to validate inputs before submission. Wrapping the form in a `SingleChildScrollView` ensures smooth scrolling when multiple fields are present.

A `RaisedButton` (deprecated) or `ElevatedButton` can trigger validation and submission logic. To enhance usability and create a responsive experience, proper padding, spacing, and `InputDecoration` should be applied.

Steps:

Step 1: Create a new Flutter project or open an existing one.

Step 2: Define a `Form` widget inside a StatefulWidget to manage user input.

Step 3: Use `TextFormField` for text input fields with validation logic.

Step 4: Include other input widgets such as `DropdownButton`, `Checkbox`, `Radio`, and `Switch` for additional user selections.

Step 5: Wrap the form inside a `SingleChildScrollView` to ensure smooth scrolling.

Step 6: Implement an `ElevatedButton` to trigger form validation and submission.

Step 7: Use `GlobalKey<FormState>` to manage form validation.

Code:

```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:dmart/screens/home/home_page.dart';
import 'package:shared_preferences/shared_preferences.dart';
```

```
class LoginPage extends StatefulWidget {
  const LoginPage({super.key});

  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final TextEditingController _emailController =
TextEditingController();
  final TextEditingController _passwordController =
TextEditingController();
  final FirebaseAuth _auth = FirebaseAuth.instance;

  final _formKey = GlobalKey<FormState>(); // Form Key for validation

  Future<void> _login() async {
    if (!_formKey.currentState!.validate()) {
      return; // Stop execution if validation fails
    }

    try {
      await _auth.signInWithEmailAndPassword(
        email: _emailController.text.trim(),
        password: _passwordController.text.trim(),
      );

      // Save login state
      SharedPreferences prefs = await SharedPreferences.getInstance();
      await prefs.setBool('isLoggedIn', true);

      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => const HomePage()),
      );
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
```

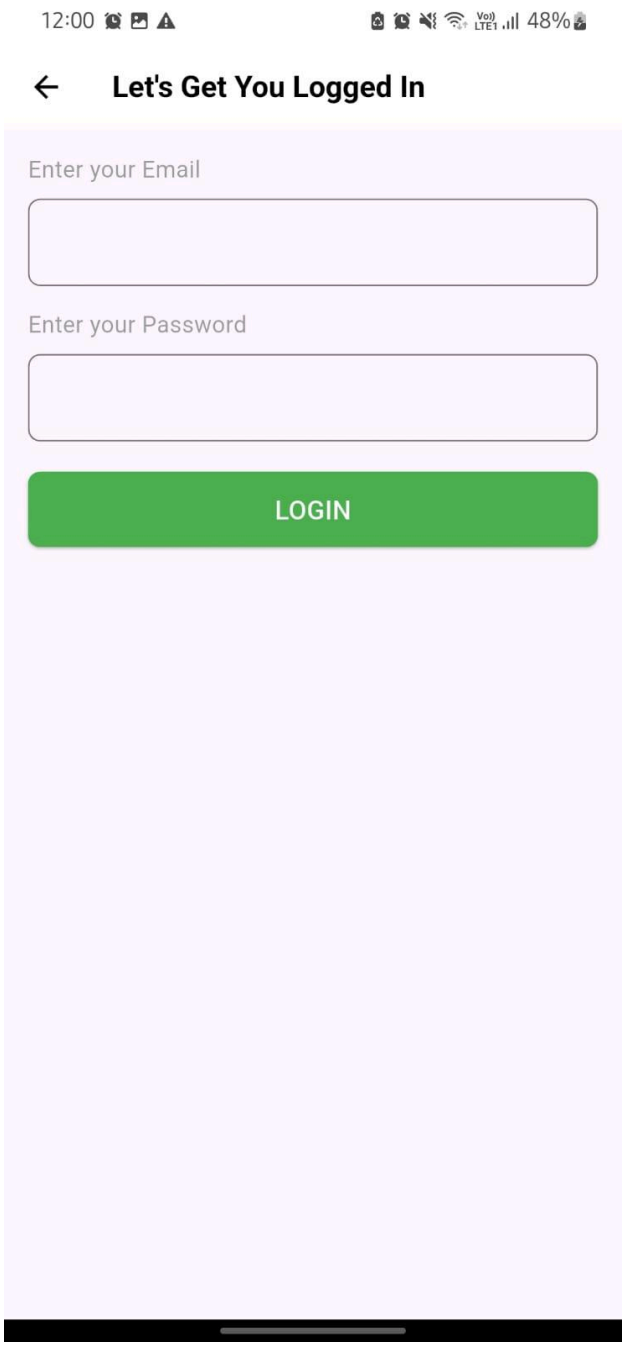
```
        SnackBar(content: Text('Login failed: ${e.toString()}')),
      );
    }
  }

  String? _validateEmail(String? value) {
    if (value == null || value.isEmpty) {
      return "Email cannot be empty";
    }
    String emailPattern =

r'^[a-zA-Z0-9.a-zA-Z0-9.!#$%&’*+/=?^_`{|}~-]+@[a-zA-Z0-9]+\.[a-zA-Z]+'
;
    RegExp regex = RegExp(emailPattern);
    if (!regex.hasMatch(value)) {
      return "Enter a valid email";
    }
    return null;
  }

  String? _validatePassword(String? value) {
    if (value == null || value.isEmpty) {
      return "Password cannot be empty";
    }
    if (value.length < 6) {
      return "Password must be at least 6 characters";
    }
    return null;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: IconButton(
          icon: const Icon(Icons.arrow_back, color: Colors.black),
          onPressed: () {
            Navigator.pop(context);
```

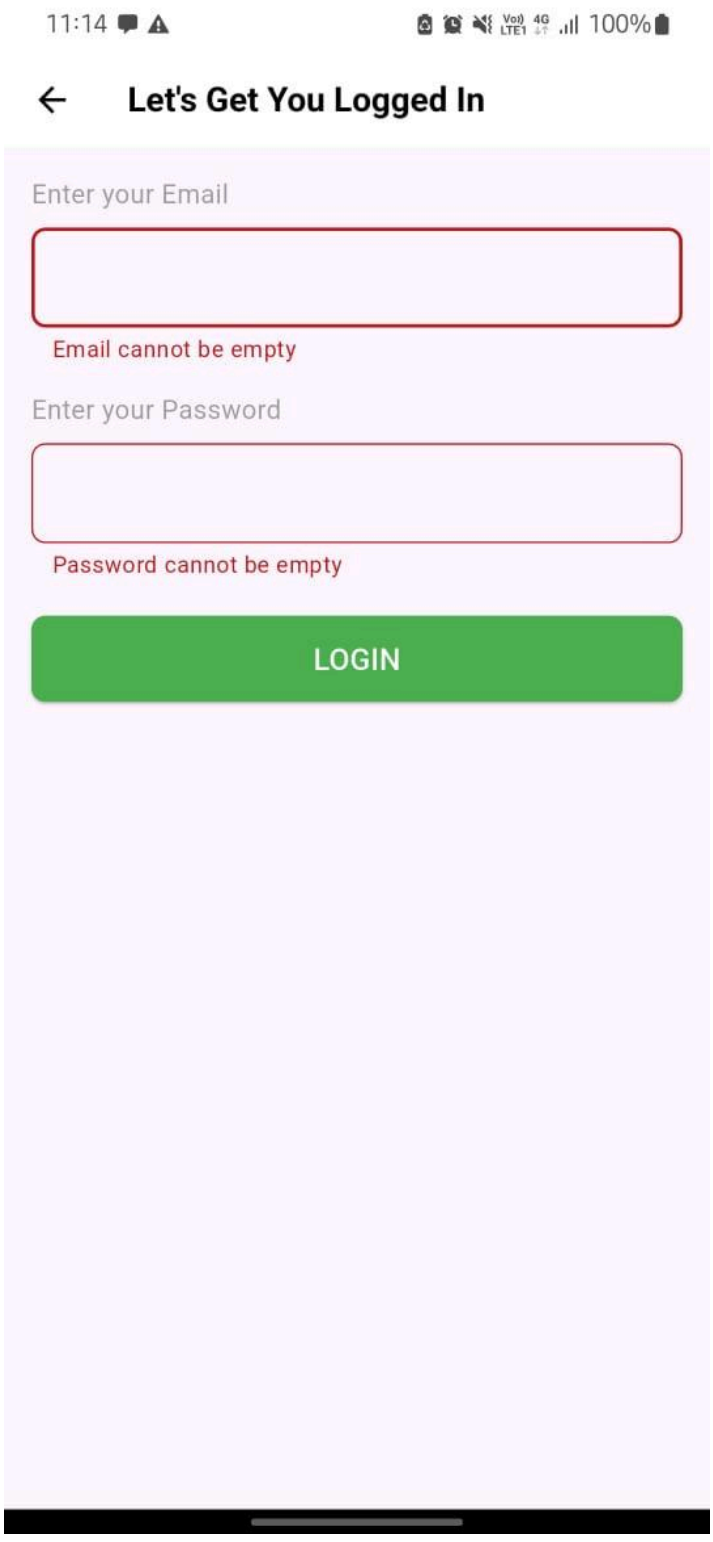
```
          },
        ),
        backgroundColor: Colors.white,
        elevation: 0,
        title: const Text(
          "Let's Get You Logged In",
          style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold,
color: Colors.black),
        ),
        centerTitle: false,
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Form(
          key: _formKey, // Wrap the form for validation
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              const Text(
                "Enter your Email",
                style: TextStyle(fontSize: 14, color: Colors.grey),
              ),
              const SizedBox(height: 8),
              TextFormField(
                controller: _emailController,
                decoration: InputDecoration(
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(8),
                    borderSide: const BorderSide(color: Colors.grey),
                  ),
                  focusedBorder: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(8),
                    borderSide: const BorderSide(color: Colors.blue),
                  ),
                ),
                keyboardType: TextInputType.emailAddress,
                validator: _validateEmail,
              ),
```

```
              const SizedBox(height: 15),
              const Text(
                "Enter your Password",
                style: TextStyle(fontSize: 14, color: Colors.grey),
              ),
              const SizedBox(height: 8),
              TextFormField(
                controller: _passwordController,
                obscureText: true,
                decoration: InputDecoration(
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(8),
                    borderSide: const BorderSide(color: Colors.grey),
                  ),
                  focusedBorder: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(8),
                    borderSide: const BorderSide(color: Colors.blue),
                  ),
                ),
                validator: _validatePassword,
              ),
              const SizedBox(height: 20),
              SizedBox(
                width: double.infinity,
                height: 50, // Match button height in image
                child: ElevatedButton(
                  onPressed: _login,
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.green, // Match button
color
                    foregroundColor: Colors.white,
                    shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(8), //
Rectangular button
                    ),
                  ),
                  child: const Text("LOGIN", style:
TextStyle(fontSize: 16)),
```

```
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

Output:
Login Page




11:14
100%
Let's Get You Logged In
Enter your Email
Email cannot be empty
Enter your Password
Password cannot be empty
LOGIN

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	28
Name	Anuprita Mhapankar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	14

MPL Experiment 5

Name: Anuprita Mhapankar **Class:** D15A **Roll no:**28

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

Theory:

Navigation & Routing:
Flutter provides multiple ways to navigate between screens (pages):

1. **Using Navigator.push() and Navigator.pop()**
   Push a new screen onto the stack and pop it to return to the previous one.

2. **Named Routes**
   Define routes in MaterialApp and navigate using Navigator.pushNamed().

3. **GoRouter Package**
   A declarative approach to handle navigation more efficiently.

Gestures:
Flutter's GestureDetector and InkWell widgets help in detecting user interactions like taps, swipes, and long presses. Some common gestures include:

- **onTap:** Detects a tap event.
- **onDoubleTap:** Recognizes double taps.
- **onLongPress:** Detects a long press on a widget.
- **onHorizontalDrag & onVerticalDrag:** Detects drag/swipe motions.

Steps:
Step 1: Create a Flutter project and define multiple screens.
Step 2: Set up named routes in `MaterialApp`.
Step 3: Implement navigation using `Navigator.push()` and `Navigator.pushNamed()`.
Step 4: Use `GestureDetector` to detect taps, swipes, and long presses.
Step 5: Add buttons or swipes to navigate between screens.

Code:

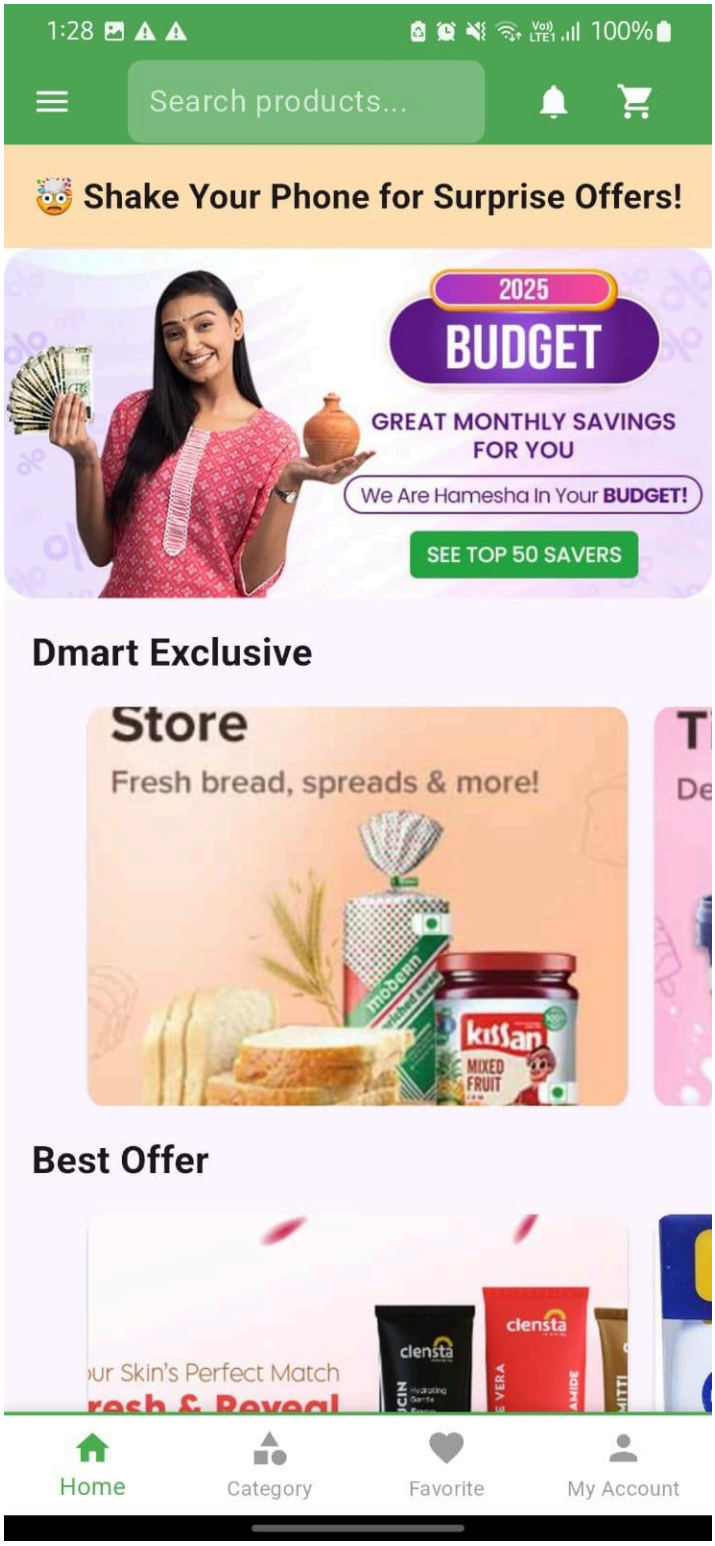
```
//bottom_tab.dart
import 'package:flutter/material.dart';
import 'package:dmart/screens/home/home_page.dart';
import 'package:dmart/screens/category/category_page.dart';
import 'package:dmart/screens/favorite/favorite_page.dart';
import 'package:dmart/screens/myaccount/account_page.dart';
```

```
class BottomTab extends StatefulWidget {
  const BottomTab({super.key});

  @override
  State<BottomTab> createState() => _BottomTabState();
}

class _BottomTabState extends State<BottomTab> {
  int _currentIndex = 0;
  final PageController _pageController = PageController();

  final List<Widget> _pages = [
    const HomePage(),
    const CategoryPage(),
    const FavoritePage(),
    const MyAccountPage(),
  ];

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SizedBox(
        height: double.infinity,
        child: PageView(
          controller: _pageController,
          physics:
              const NeverScrollableScrollPhysics(), // Prevent swipe
navigation
          children: _pages,
          onPageChanged: (index) {
            setState(() {
```

```
              _currentIndex = index;
            });
          },
        ),
      ),
      bottomNavigationBar: Container(
        decoration: const BoxDecoration(
          border: Border(
            top: BorderSide(
                width: 2, color: Colors.green), // Green border at the
top
          ),
          boxShadow: [
            BoxShadow(
              color: Colors.black12,
              blurRadius: 5,
              offset: Offset(0, -2), // Slight elevation effect
            ),
          ],
        ),
        child: BottomNavigationBar(
          type: BottomNavigationBarType.fixed,
          backgroundColor: Colors.white,
          elevation: 10,
          selectedItemColor: Colors.green, // Active icon color
          unselectedItemColor: Colors.grey, // Inactive icon color
          showUnselectedLabels: true,
          showSelectedLabels: true,
          currentIndex: _currentIndex,
          onTap: (int index) {
            _pageController.jumpToPage(index);
          },
          items: const [
            BottomNavigationBarItem(
              icon: Icon(Icons.home),
              label: 'Home',
            ),
            BottomNavigationBarItem(
```

```
            icon: Icon(Icons.category),
            label: 'Category',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.favorite),
            label: 'Favorite',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.person),
            label: 'My Account',
          ),
        ],
      ),
    ),
  );
 }
}
```

Output:


1:28
100%
Search products...
Shake Your Phone for Surprise Offers!
2025
BUDGET
GREAT MONTHLY SAVINGS FOR YOU
We Are Hamesha In Your BUDGET!
SEE TOP 50 SAVERS
Dmart Exclusive
Store
Fresh bread, spreads & more!
kissan
MIXED FRUIT
Best Offer
clensta
Home
Category
Favorite
My Account

1:28 100%

Shop By Category

- Dairy & Beverages
- Electronics & Appliances
- Grocery
- Home & Kitchen
- Packaged Foods
- Sports & Fitness

Juices

Soft Drinks

Milk & Dairy

Tea & Coffee

Home | Category | Favorite | My Account

1:29 100%

Soft Drinks

1

Thumbs Up

Qty: 2.25 L

DMart: ₹75

~~MRP: ₹100~~

₹25 OFF

− 1 +

Sprite

Qty: 2.25 L

DMart: ₹62

~~MRP: ₹100~~

₹38 OFF

ADD

1:29 100%

My Cart

Thumbs Up

Qty: 2.25 L

DMart: ₹75.0

~~MRP: ₹100.0~~

₹25 OFF

1

Total Amount: ₹75.00

Clear Cart

Checkout

1:28 ⚠ ⚠ 100%

Favorites

Real Fruit Power Mixed Fruit

Qty: 1 L

DMart: ₹96.00

~~MRP: ₹115.20~~

₹19.20 OFF

B Natural Mixed Fruit Beverage

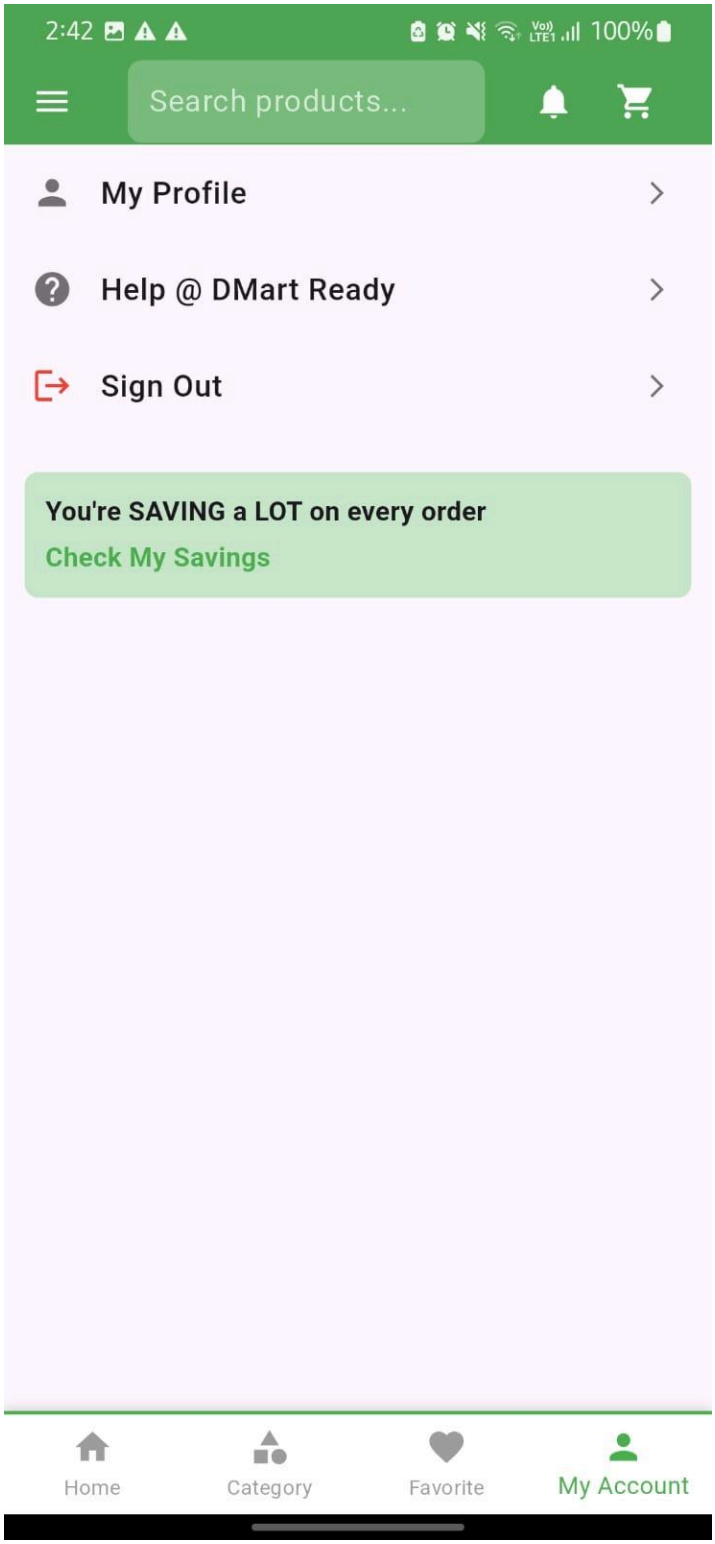
Qty: 1 L

DMart: ₹70.00

~~MRP: ₹84.00~~

₹14.00 OFF

Home Category Favorite My Account


2:42
100%
Search products...
My Profile
Help @ DMart Ready
Sign Out
You're SAVING a LOT on every order
Check My Savings
Home
Category
Favorite
My Account

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	28
Name	Anuprita Mhapankar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	14

MPL Experiment 6

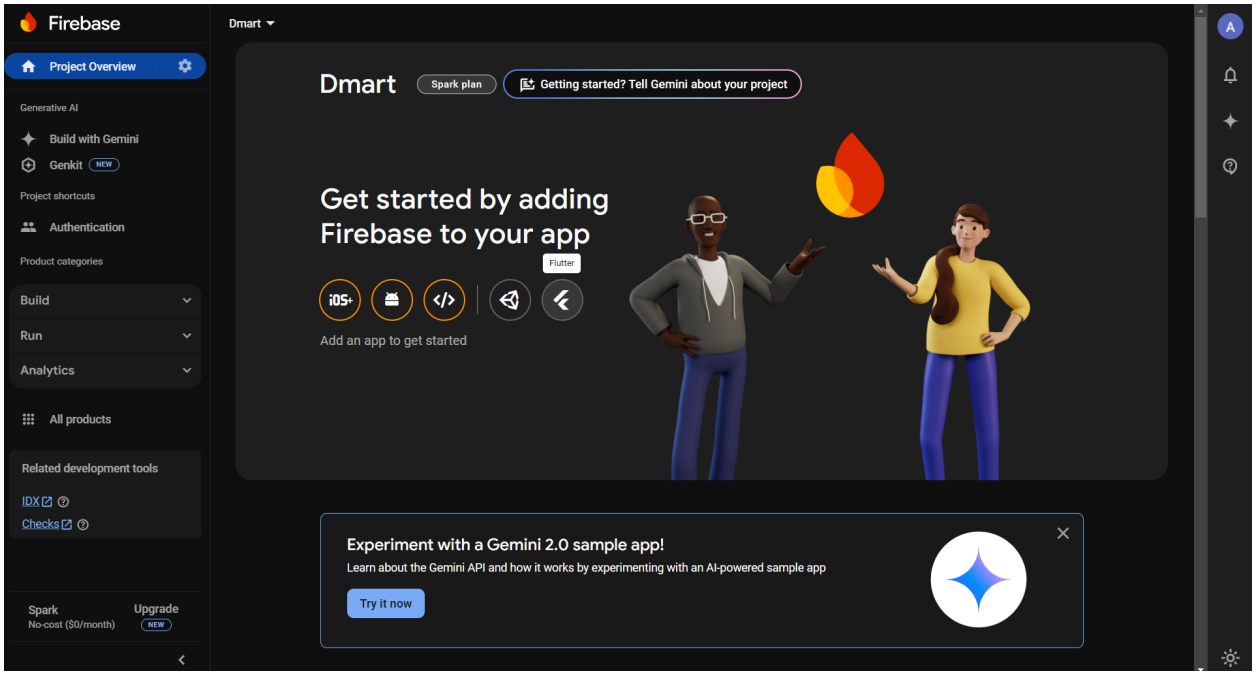
Name: Anuprita Mhapankar **Class:** D15A **Roll no:**28

Aim: How To Set Up Firebase with Flutter for iOS and Android Apps

Steps to Set Up Firebase with Flutter:

Step 1:

Go to the Firebase Console (https://console.firebase.google.com/). Click on "Add Project" and follow the steps to create your Firebase project. Once the project is created, select the Flutter option for connecting your app with Firebase.



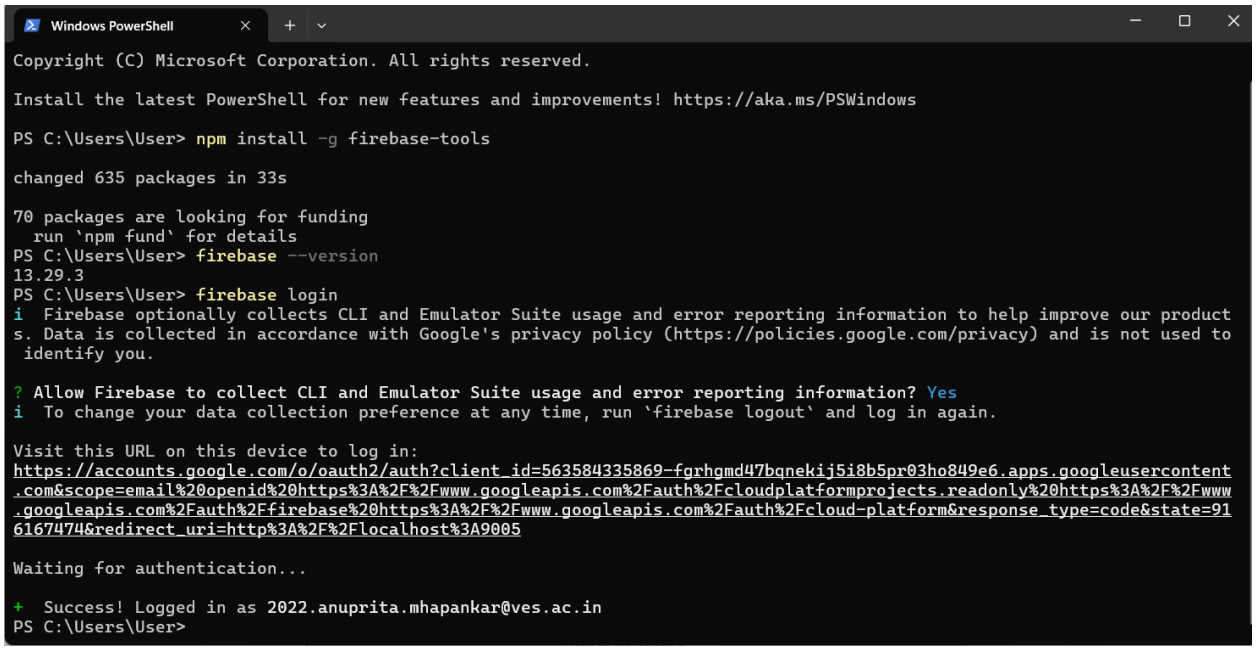
Step 2:

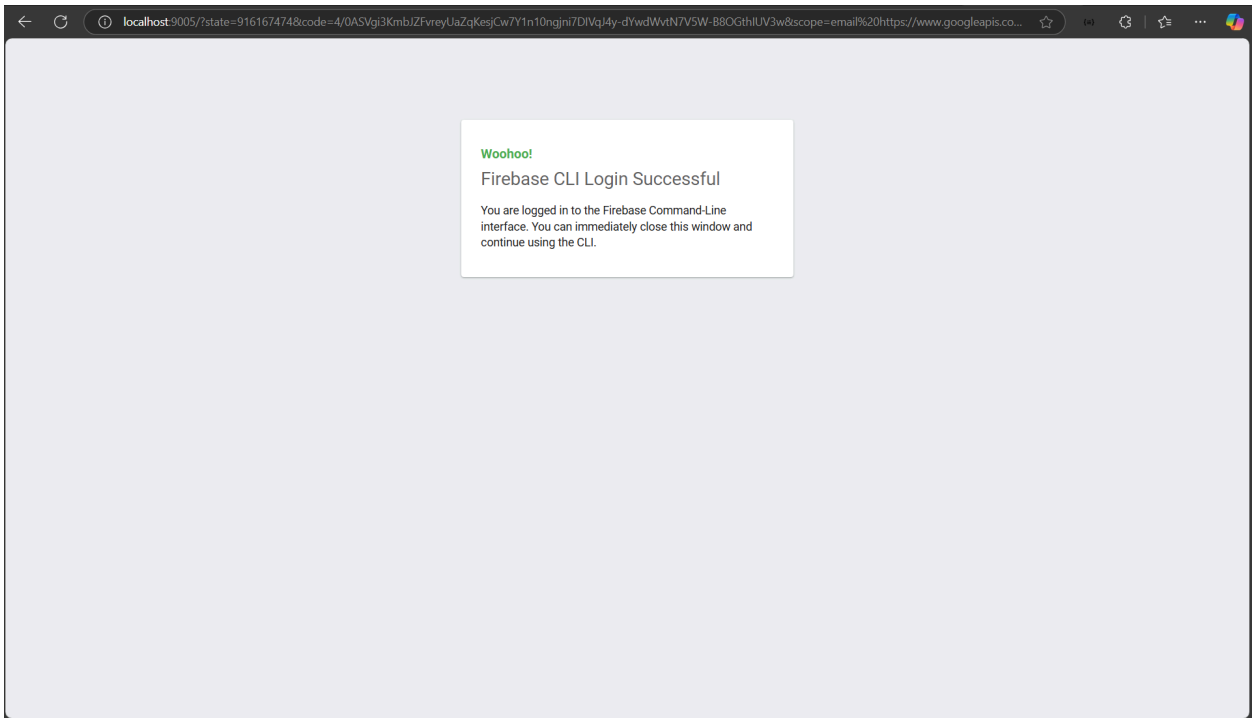
Open **Windows PowerShell** (or any terminal you prefer).

Run the following commands to install Firebase CLI and verify the installation:

```
npm install -g firebase-tools
firebase --version
firebase login
```

This will install the **Firebase CLI**, check the version, and log you into your Firebase account.





Step 3:

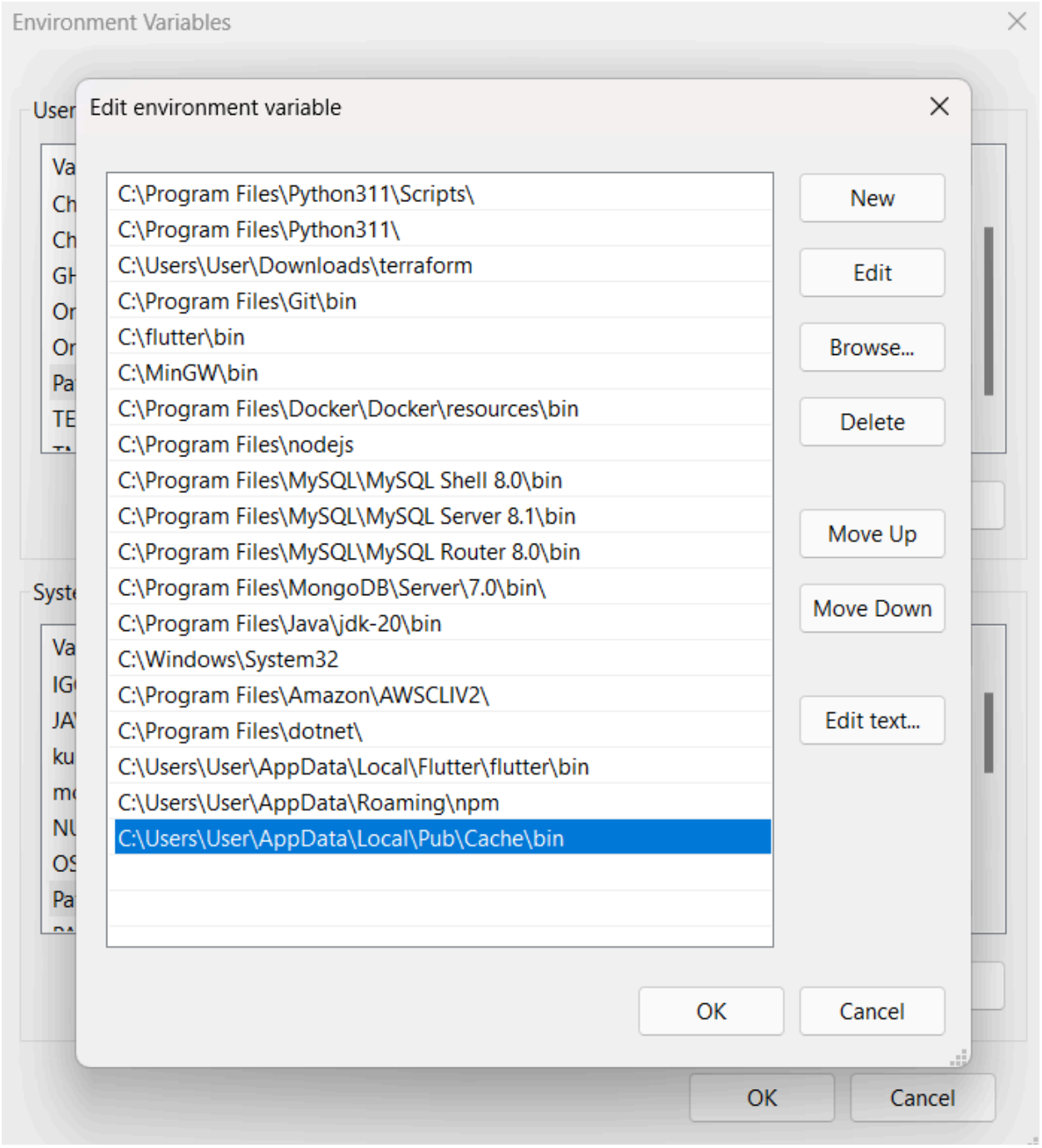
Open your Flutter app in **Android Studio**.

In the terminal of Android Studio, run the following command to activate flutterfire_cli:

```
dart pub global activate flutterfire_cli
```

Add flutterfire to your Environment Variables. You may need to restart Android Studio after

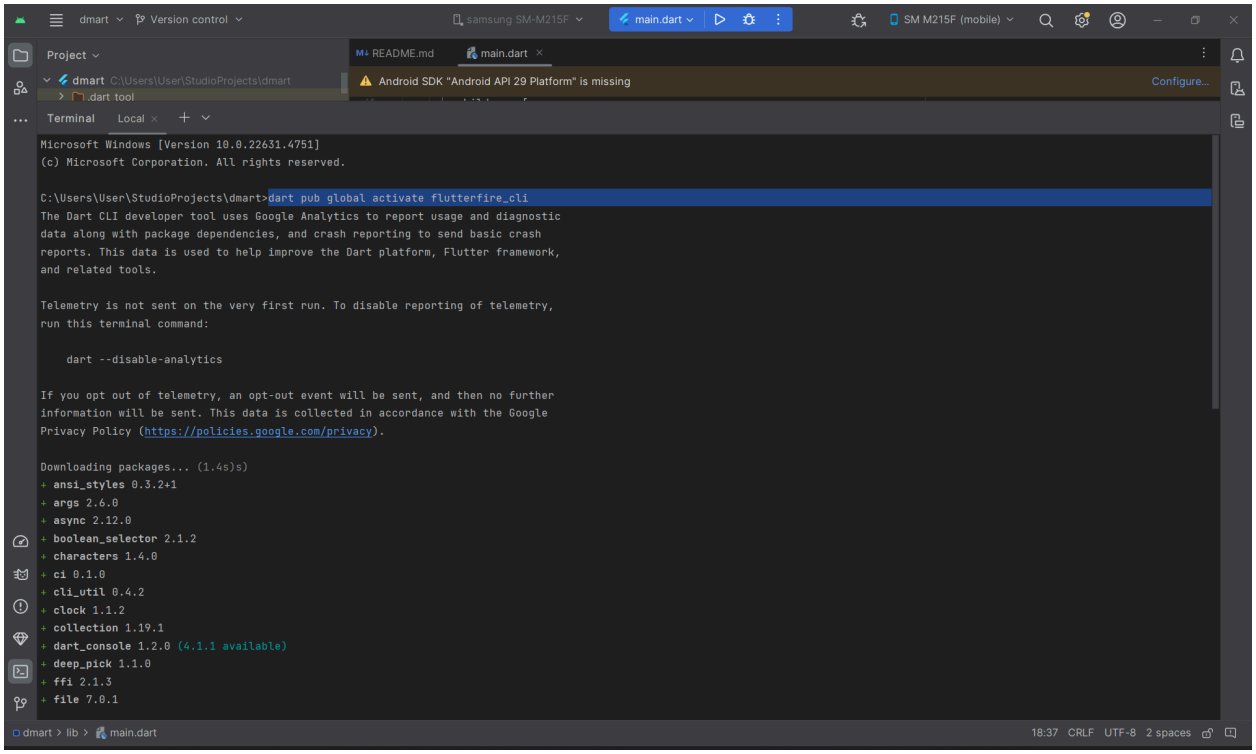
adding it.

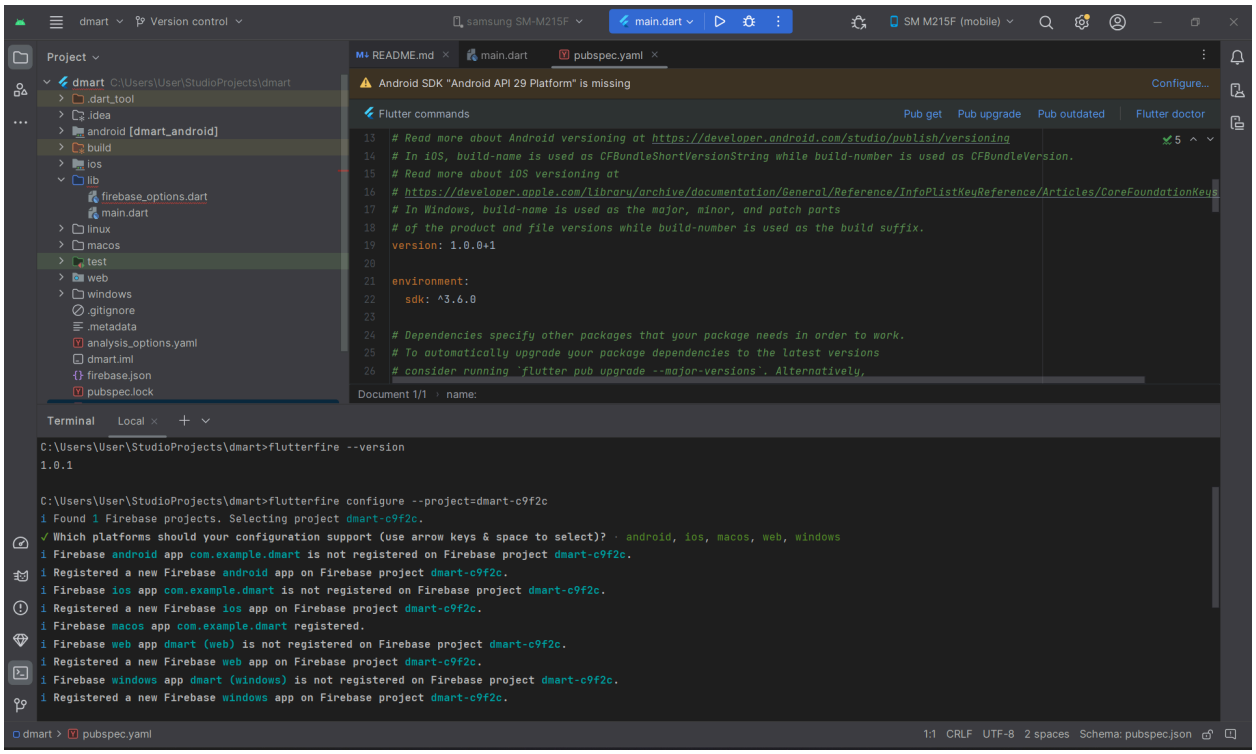


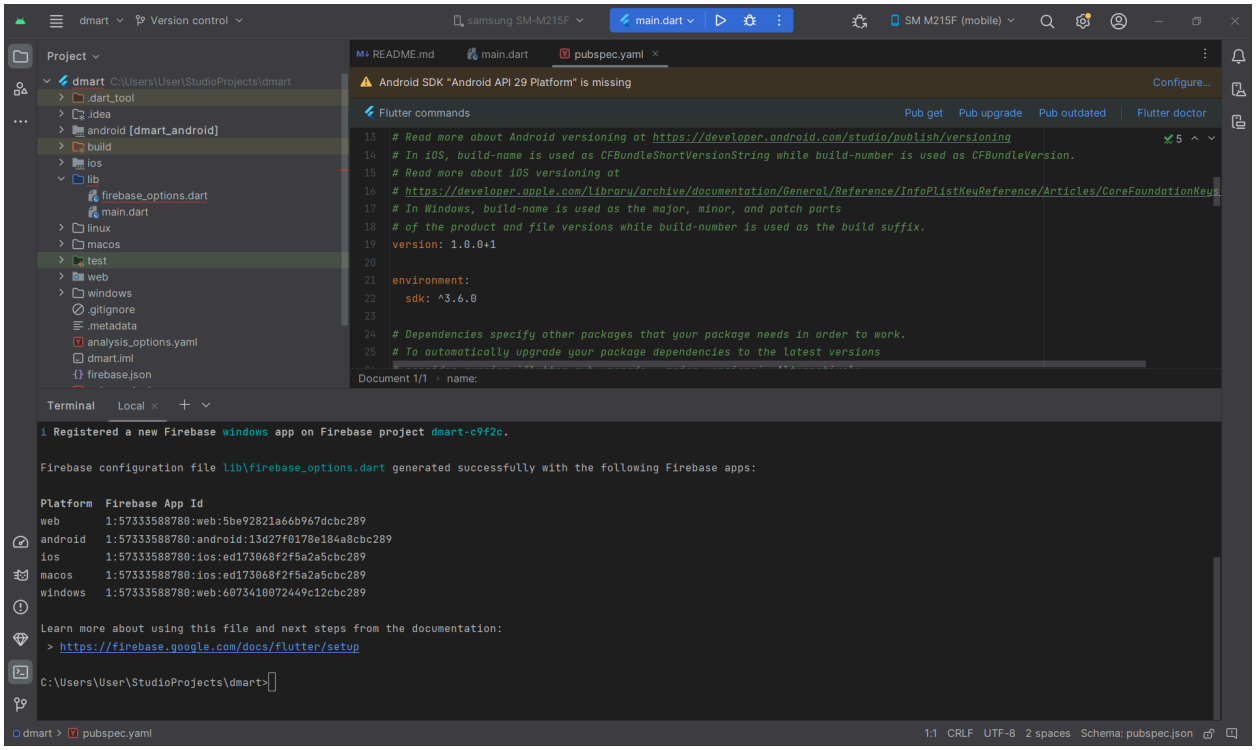
Step 4:
Run the following command to configure Firebase with your project:

```
flutterfire configure --project=dmart-c9f2c
```

Replace `dmart-c9f2c` with your Firebase project ID.


dmart
Version control
samsung SM-M215F
main.dart
SM M215F (mobile)
Project
README.md
main.dart
Android SDK "Android API 29 Platform" is missing
Configure...
dmart C:\Users\User\StudioProjects\dmart
.dart_tool
Terminal
Local
Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.
C:\Users\User\StudioProjects\dmart>dart pub global activate flutterfire_cli
The Dart CLI developer tool uses Google Analytics to report usage and diagnostic
data along with package dependencies, and crash reporting to send basic crash
reports. This data is used to help improve the Dart platform, Flutter framework,
and related tools.
Telemetry is not sent on the very first run. To disable reporting of telemetry,
run this terminal command:
dart --disable-analytics
If you opt out of telemetry, an opt-out event will be sent, and then no further
information will be sent. This data is collected in accordance with the Google
Privacy Policy (https://policies.google.com/privacy).
Downloading packages... (1.4s)s)
+ ansi_styles 0.3.2+1
+ args 2.6.0
+ async 2.12.0
+ boolean_selector 2.1.2
+ characters 1.4.0
+ ci 0.1.0
+ cli_util 0.4.2
+ clock 1.1.2
+ collection 1.19.1
+ dart_console 1.2.0 (4.1.1 available)
+ deep_pick 1.1.0
+ ffi 2.1.3
+ file 7.0.1
dmart > lib > main.dart
18:37 CRLF UTF-8 2 spaces

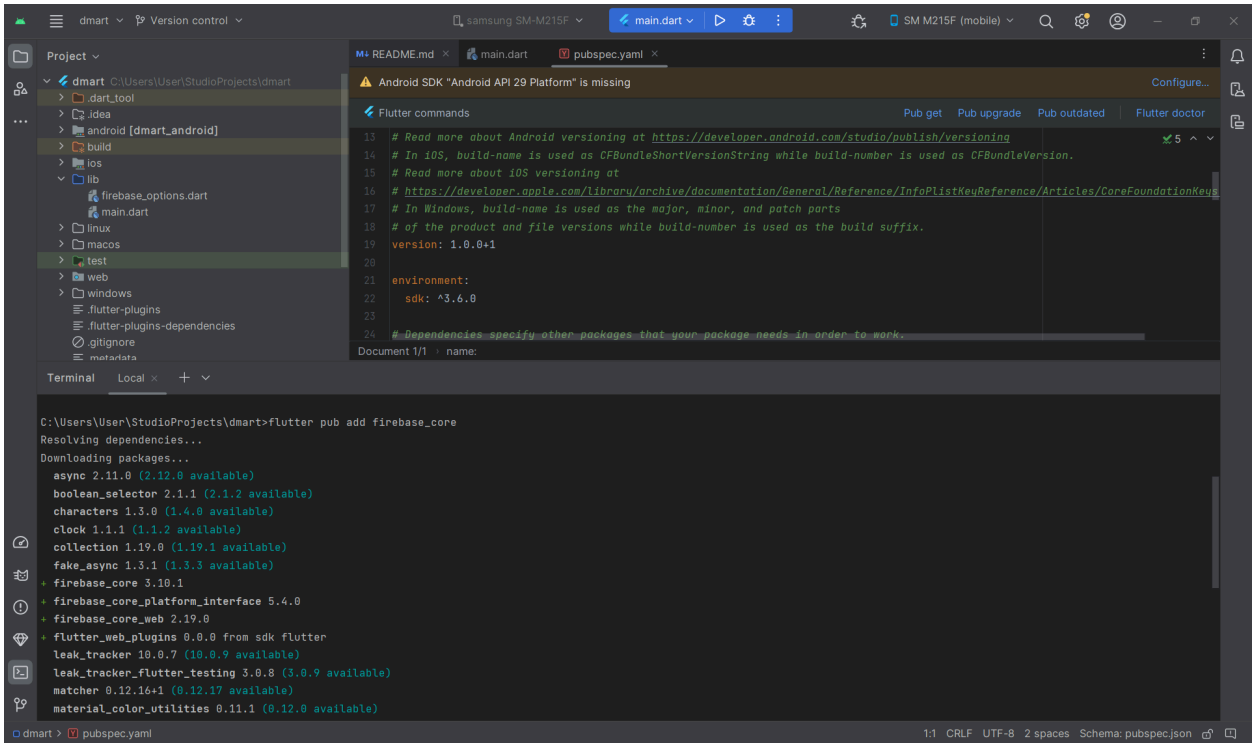

dmart
Version control
samsung SM-M215F
main.dart
SM M215F (mobile)
Project
README.md
main.dart
pubspec.yaml
Android SDK "Android API 29 Platform" is missing
Configure...
Flutter commands
Pub get
Pub upgrade
Pub outdated
Flutter doctor
dmart C:\Users\User\StudioProjects\dmart
.dart_tool
.idea
android [dmart_android]
build
ios
lib
firebase_options.dart
main.dart
linux
macos
test
web
windows
.gitignore
.metadata
analysis_options.yaml
dmart.iml
firebase.json
pubspec.lock
Read more about Android versioning at https://developer.android.com/studio/publish/versioning
In iOS, build-name is used as CFBundleShortVersionString while build-number is used as CFBundleVersion.
Read more about iOS versioning at
https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys
In Windows, build-name is used as the major, minor, and patch parts
of the product and file versions while build-number is used as the build suffix.
version: 1.0.0+1
environment:
sdk: ^3.6.0
Dependencies specify other packages that your package needs in order to work.
To automatically upgrade your package dependencies to the latest versions
consider running `flutter pub upgrade --major-versions`. Alternatively,
Document 1/1
name:
Terminal
Local
C:\Users\User\StudioProjects\dmart>flutterfire --version
1.0.1
C:\Users\User\StudioProjects\dmart>flutterfire configure --project=dmart-c9f2c
i Found 1 Firebase projects. Selecting project dmart-c9f2c.
✓ Which platforms should your configuration support (use arrow keys & space to select)? · android, ios, macos, web, windows
i Firebase android app com.example.dmart is not registered on Firebase project dmart-c9f2c.
i Registered a new Firebase android app on Firebase project dmart-c9f2c.
i Firebase ios app com.example.dmart is not registered on Firebase project dmart-c9f2c.
i Registered a new Firebase ios app on Firebase project dmart-c9f2c.
i Firebase macos app com.example.dmart registered.
i Firebase web app dmart (web) is not registered on Firebase project dmart-c9f2c.
i Registered a new Firebase web app on Firebase project dmart-c9f2c.
i Firebase windows app dmart (windows) is not registered on Firebase project dmart-c9f2c.
i Registered a new Firebase windows app on Firebase project dmart-c9f2c.
dmart > pubspec.yaml
1:1 CRLF UTF-8 2 spaces Schema: pubspec.json



Step 5:

Run this command to add Firebase Core dependency to your app:

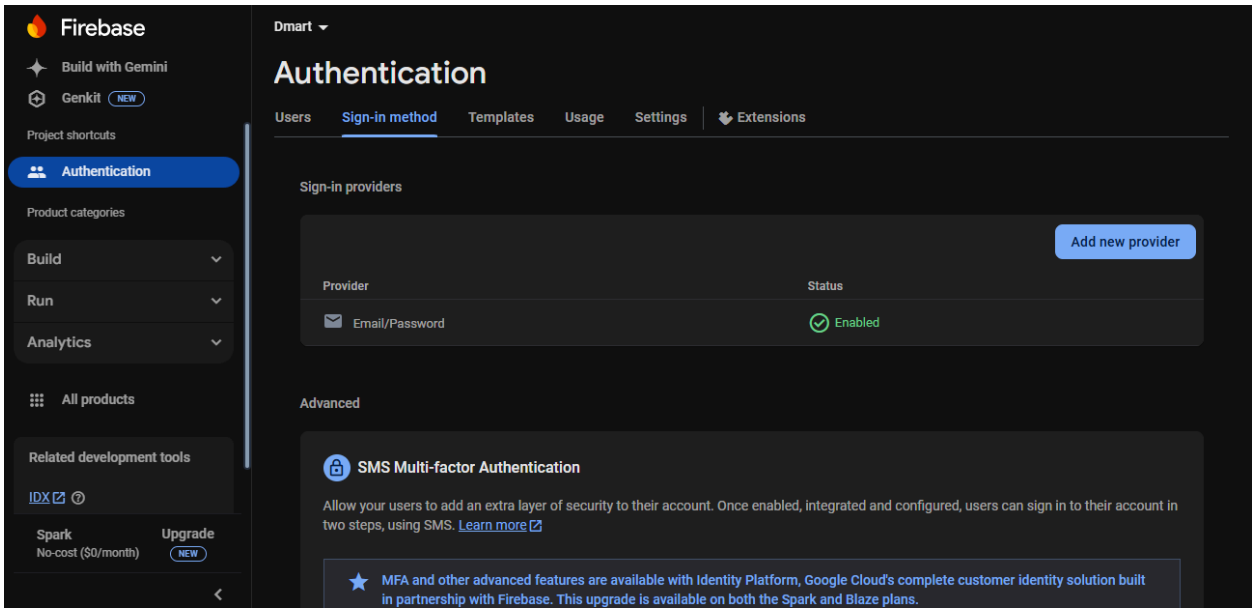
`flutter pub add firebase_core`

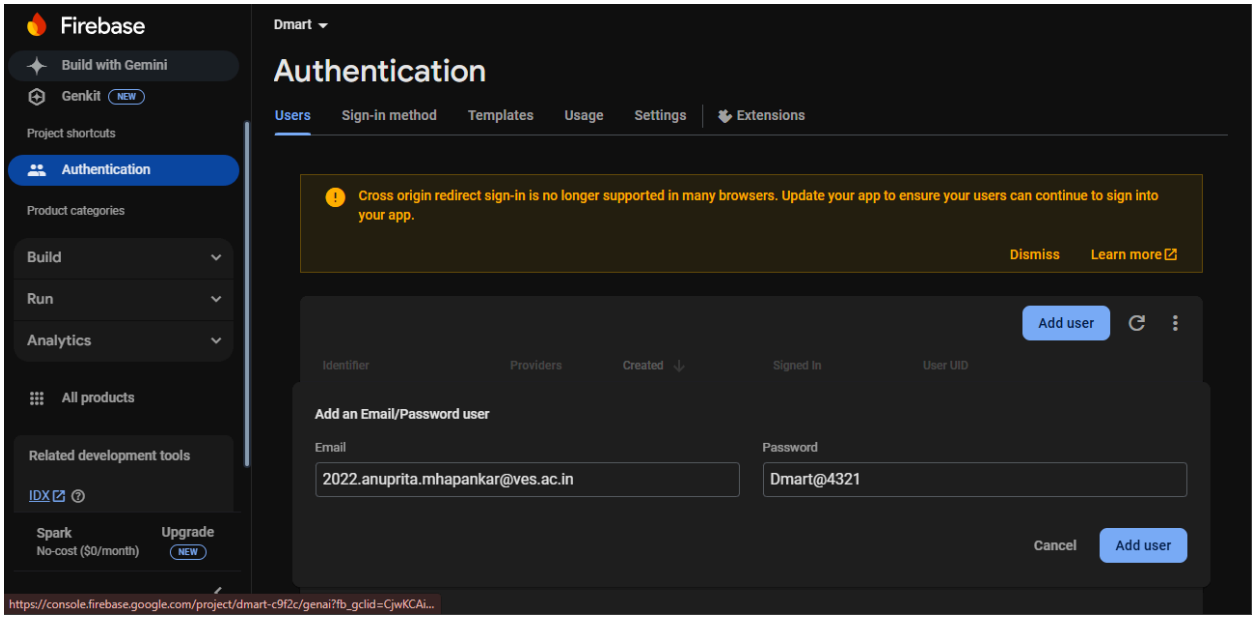


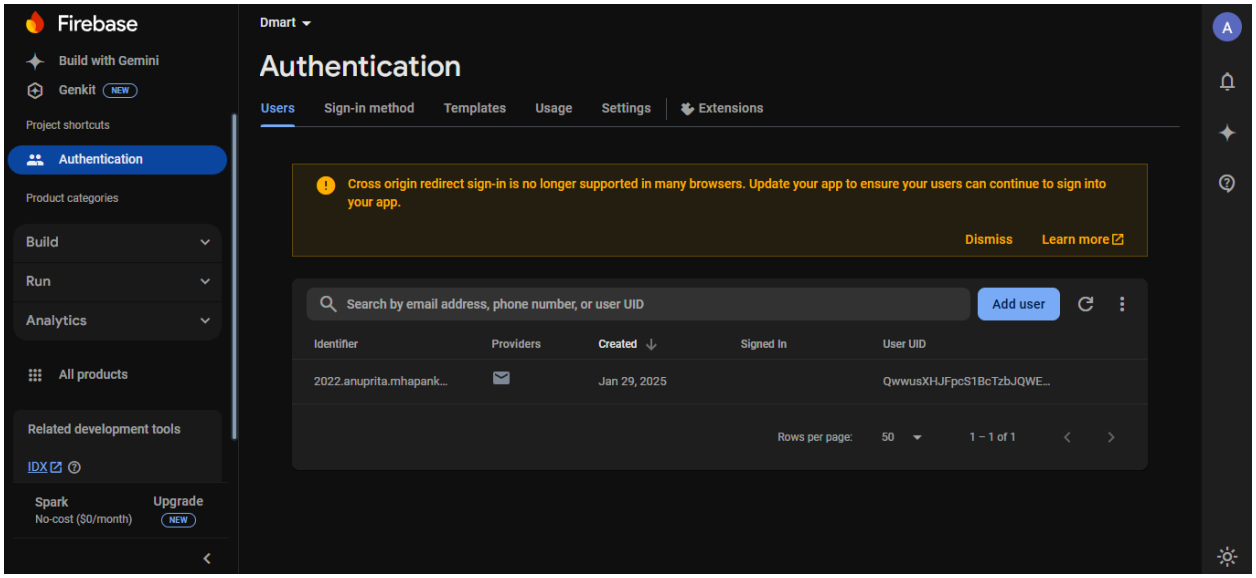
Firebase Authentication SetUp

Step 1:

Go to the Firebase Console (https://console.firebase.google.com/). In your Firebase project, navigate to **Authentication**. Under the **Sign-in method** tab, enable **Email/Password** sign-in. Once enabled, go to the **Users** section and click on **Add User**. Enter a **username** (email) and a **password** for the new user.





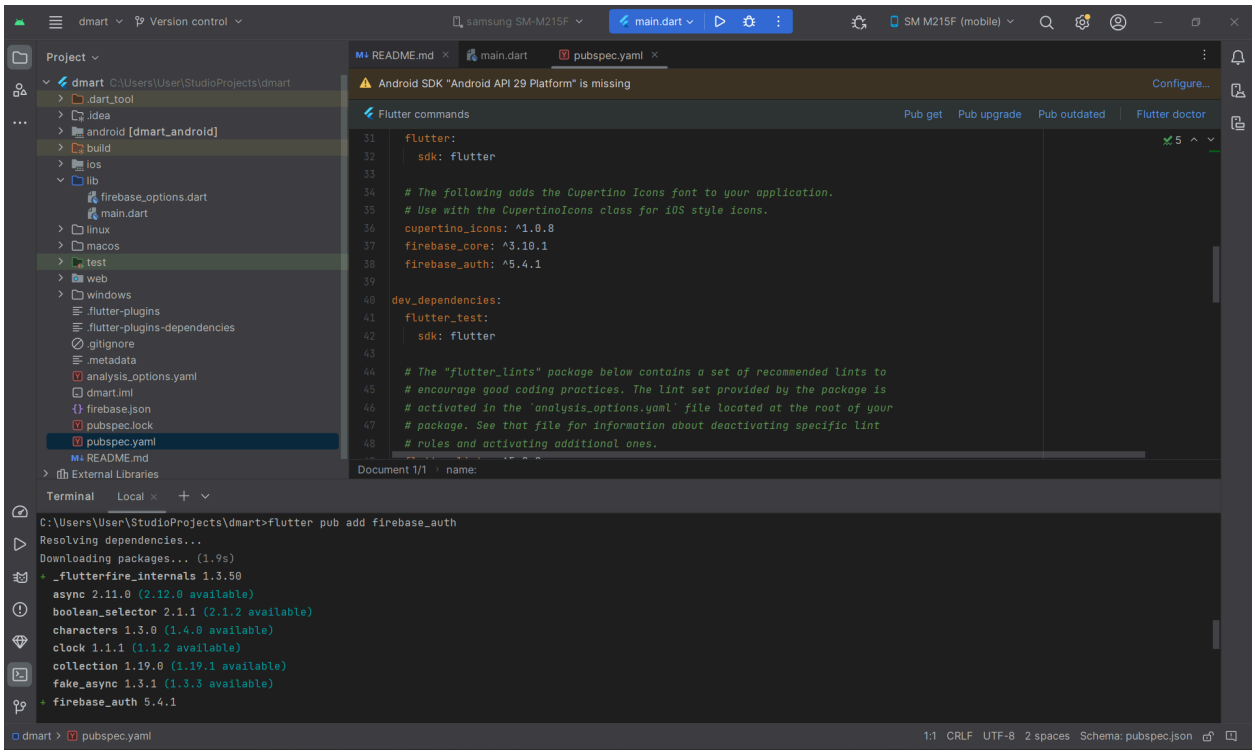


Step 2:

Open your app in **Android Studio**.

In the terminal, run the following command to add the Firebase Authentication dependency:

```
flutter pub add firebase_auth
```

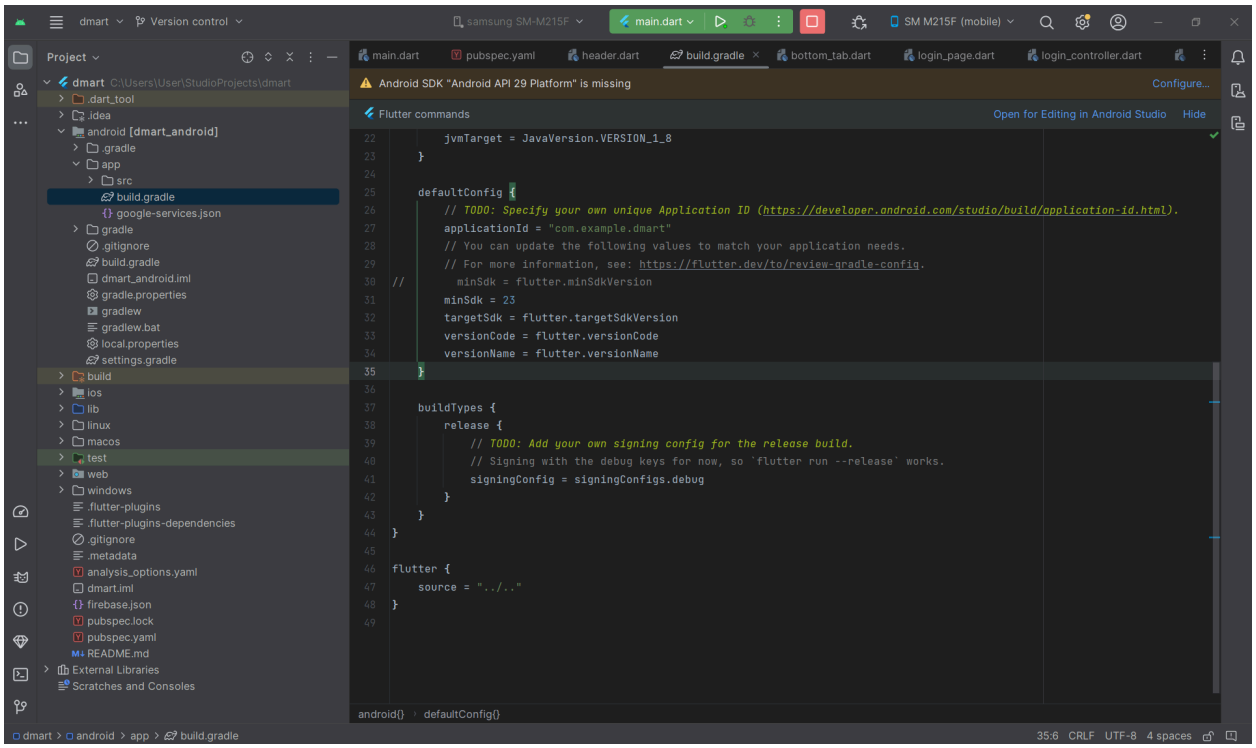


Step 3:

If you encounter any issues, add the following configuration to your `android/app/build.gradle` file:

```
android {
    defaultConfig {
        minSdk = 23
        targetSdk = flutter.targetSdkVersion
        versionCode = flutter.versionCode
        versionName = flutter.versionName
    }
}
```

This ensures that the Firebase dependencies are compatible with your Android app.

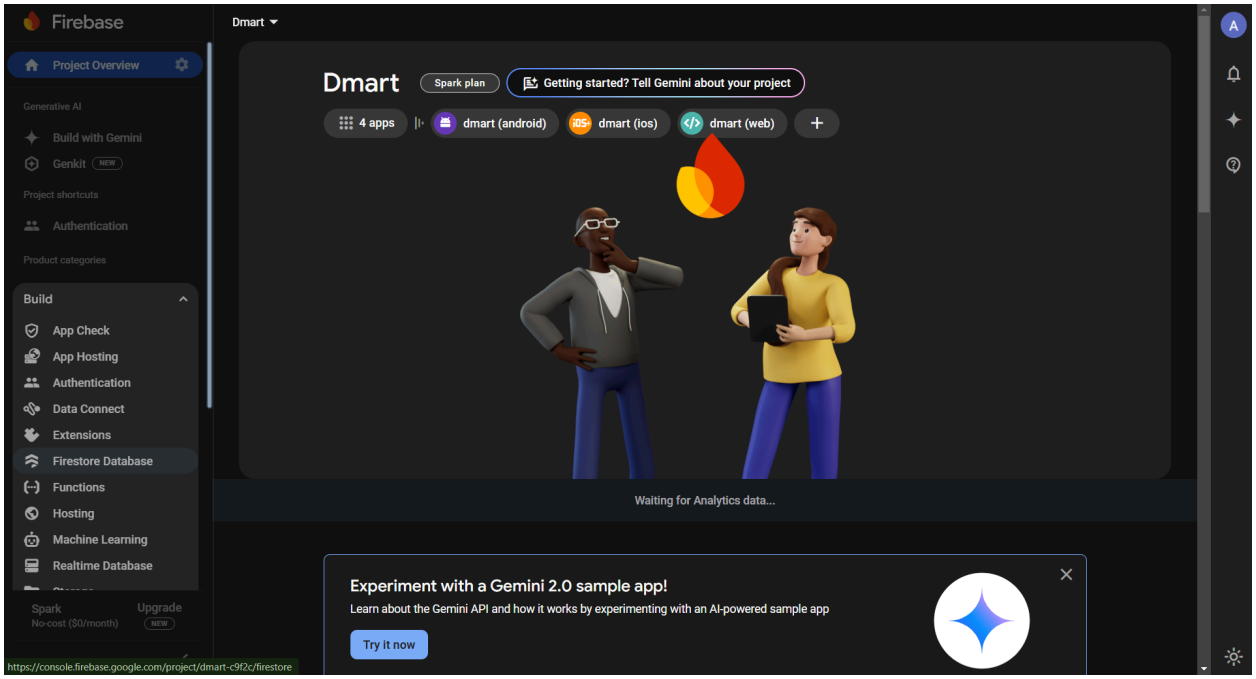


Now, firebase is successfully connected to our app.

Firestore Database Setup

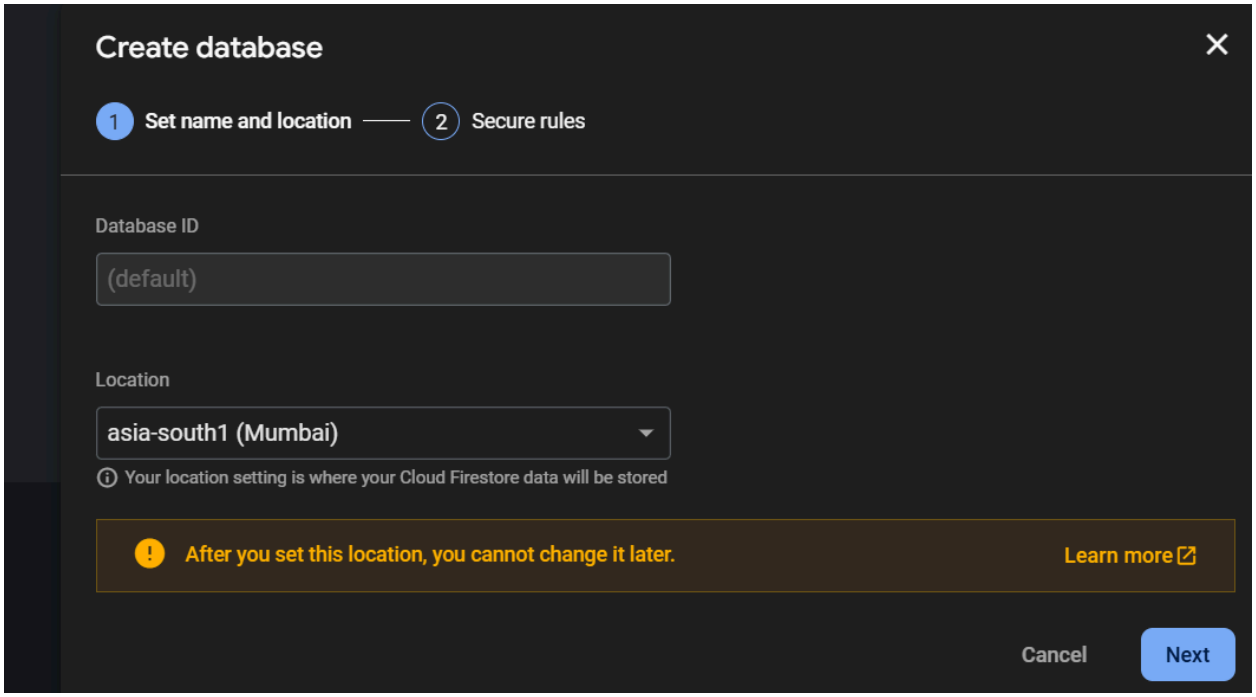
Step 1:

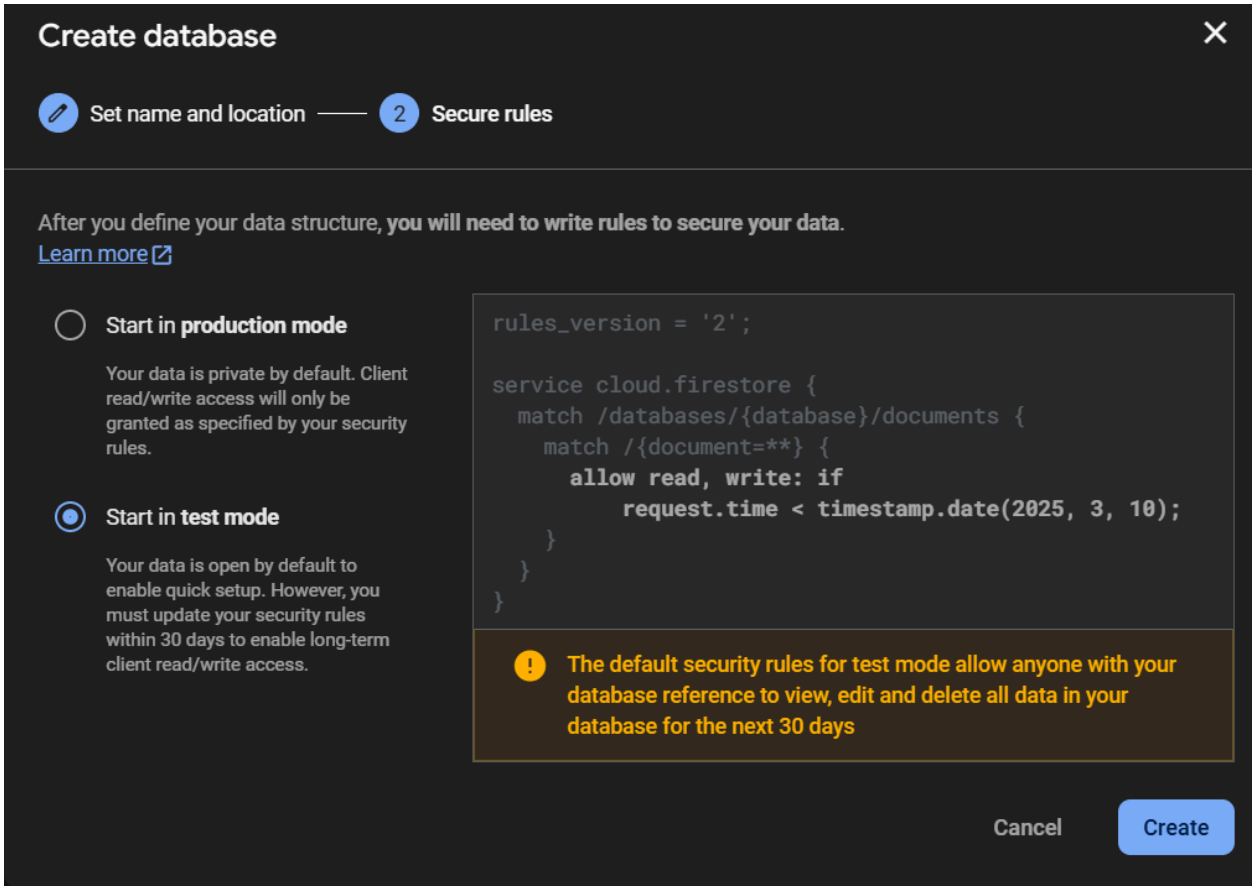
Select **Firestore Database** from **Build** in the sidebar.



Then click on Create database. For location select asia-south1(Mumbai). Then choose **Start in test mode** (for development).

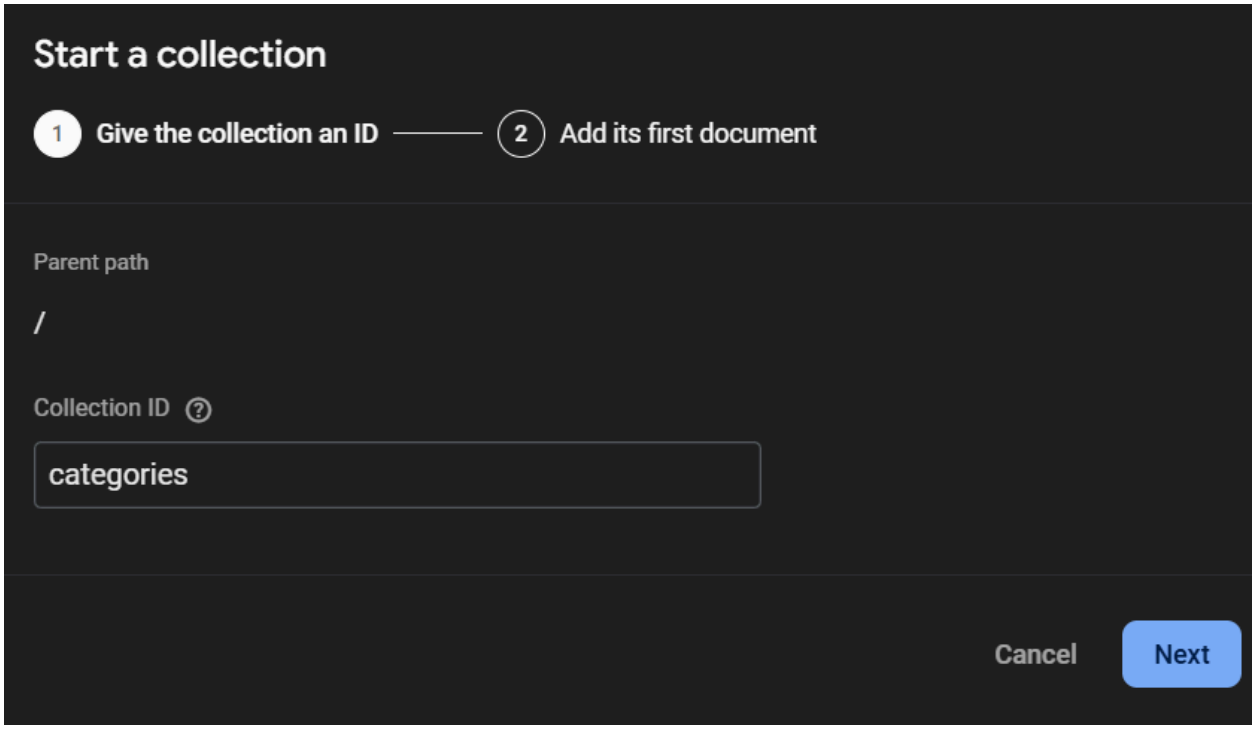
Click **Next** and choose a location, then **Enable**.

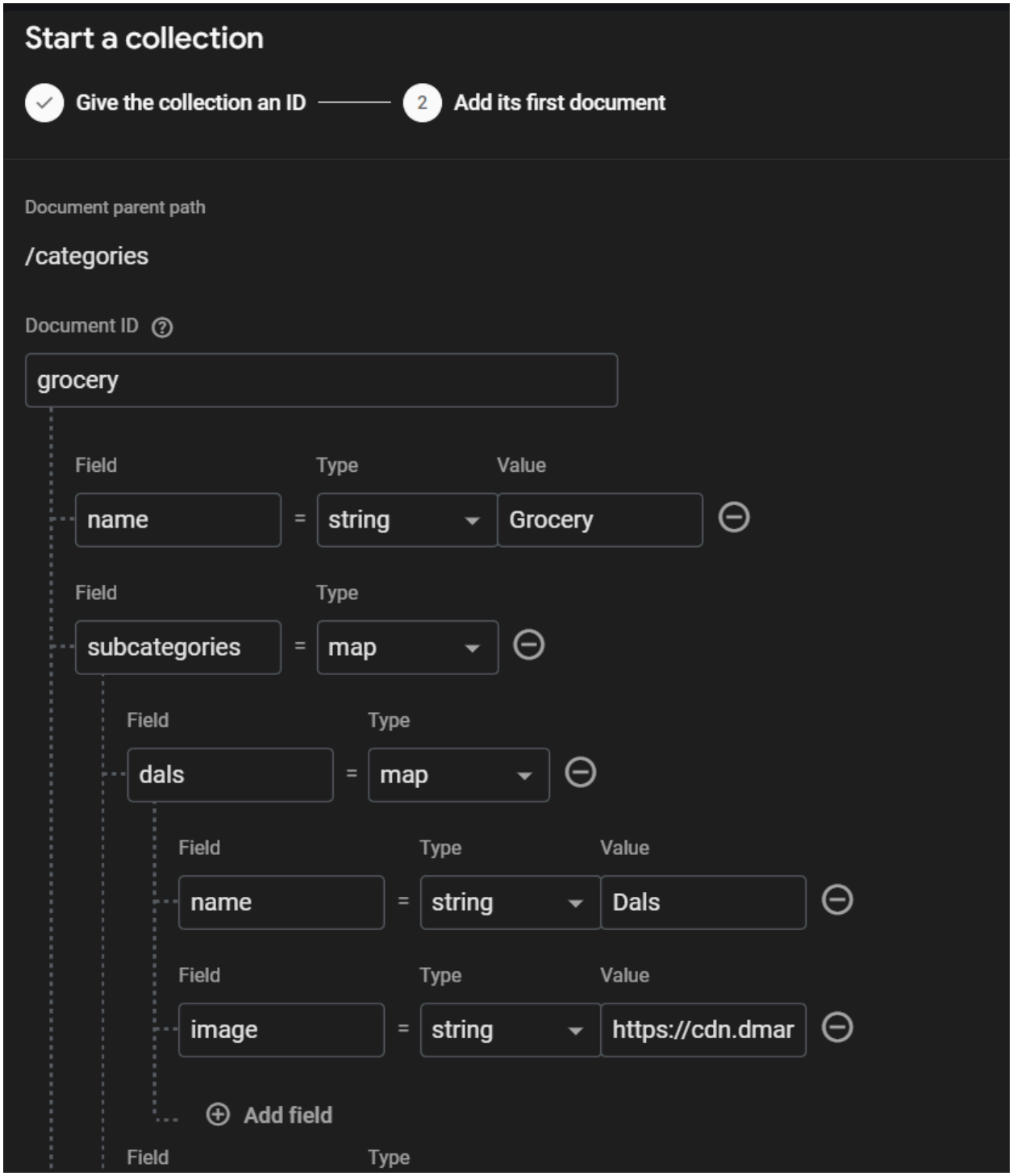


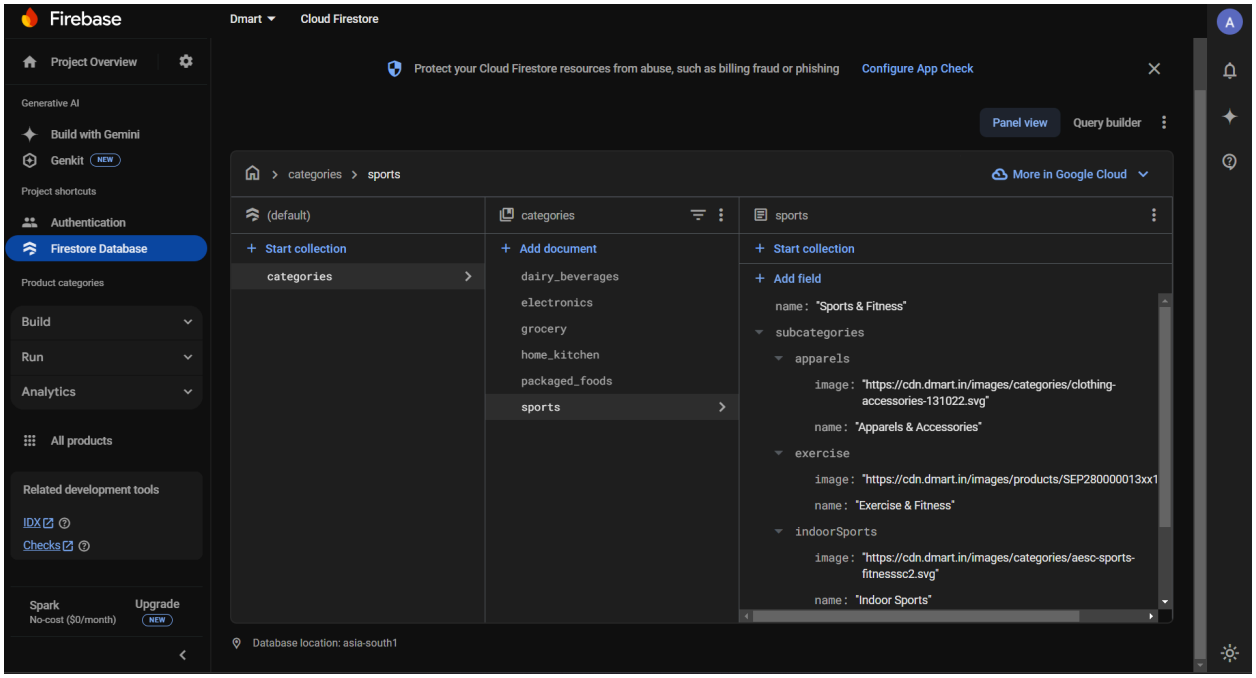


Step 2:
Start by creating a **collection**. Add a **document** within the collection. Define the **fields** as per

your project requirements.




Start a collection
Give the collection an ID
2
Add its first document
Document parent path
/categories
Document ID
grocery
Field
Type
Value
name
=
string
Grocery
Field
Type
subcategories
=
map
Field
Type
dals
=
map
Field
Type
Value
name
=
string
Dals
Field
Type
Value
image
=
string
https://cdn.dmar
Add field
Field
Type



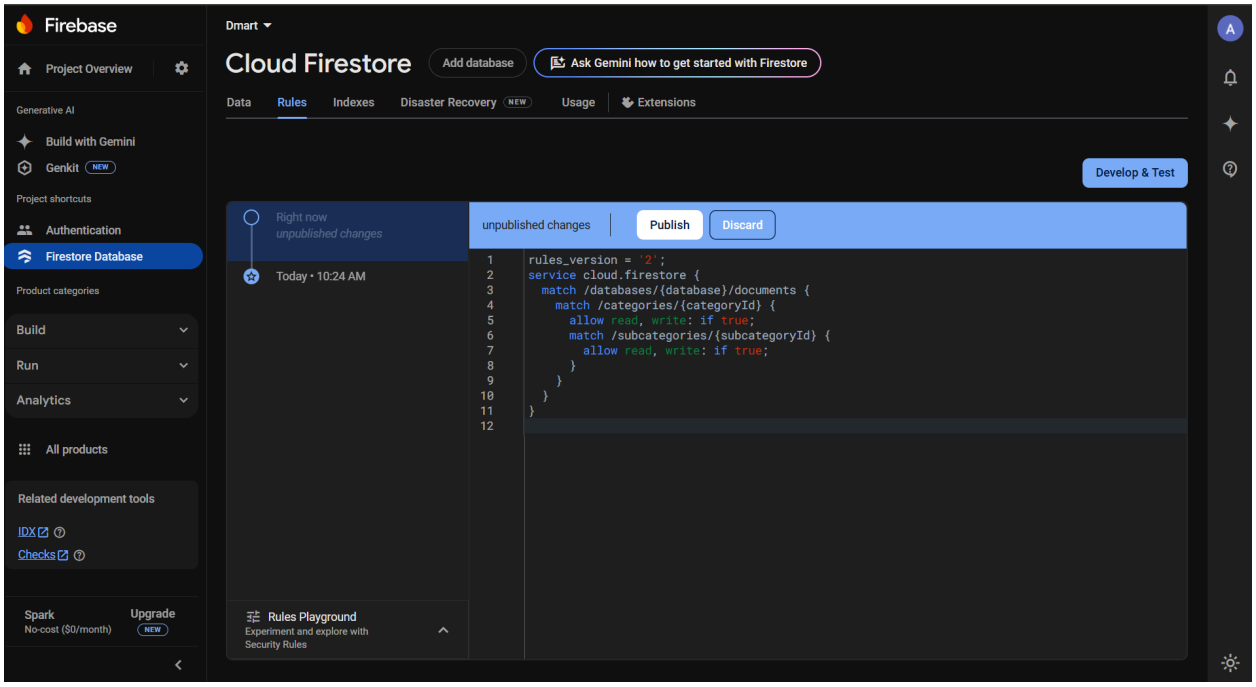
Step 3:
Run the following command to add Firestore to your Flutter app:

```
flutter pub add cloud_firestore
```

Step 4:
Go to Firebase Console → Firestore Database → Rules
Set the following rules:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /categories/{categoryId} {
      allow read, write: if true;
      match /subcategories/{subcategoryId} {
        allow read, write: if true;
      }
    }
  }
}
```

and then click **Publish** to apply changes.



Code:

```
//category_page.dart
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:dmart/screens/category/productList_page.dart';
import 'package:provider/provider.dart';
import 'package:dmart/providers/cart_provider.dart';
import 'package:dmart/screens/cart/cart_page.dart';

class CategoryPage extends StatefulWidget {
  const CategoryPage({super.key});

  @override
  State<CategoryPage> createState() => _CategoryPageState();
}

class _CategoryPageState extends State<CategoryPage> {
  List<Map<String, dynamic>> categories = [];
  String selectedCategoryId = '';

  @override
  void initState() {
    super.initState();
    testFirestoreConnection();
    fetchCategories();
  }
```

```
void testFirestoreConnection() async {
  try {
    // Fetch the Firestore collection
    var snapshot =
        await FirebaseFirestore.instance.collection('categories').get();

    // Print each document's data to the console
    for (var doc in snapshot.docs) {
      print("Category ID: ${doc.id}, Data: ${doc.data()}");
    }

    // Optionally, log the total count of documents
    print("Total Categories: ${snapshot.docs.length}");
  } catch (e) {
    print("Error fetching categories: $e"); // Error handling
  }
}

Future<void> fetchCategories() async {
  try {
    var snapshot =
        await FirebaseFirestore.instance.collection('categories').get();
    setState(() {
      categories = snapshot.docs.map((doc) {
        Map<String, dynamic> data = doc.data();
        return {
          'id': doc.id,
          'name': data['name'] ?? 'Unknown Category',
          'subcategories':
              Map<String, dynamic>.from(data['subcategories'] ?? {})
        };
      }).toList();

      if (categories.isNotEmpty) {
        selectedCategoryId = categories[0]['id'];
      }
    });
  } catch (e) {
    print("Error fetching categories: $e");
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Colors.green,
```

```
  title: const Text(
    'Shop By Category',
    style: TextStyle(
      color: Colors.white, // White text color
      fontWeight: FontWeight.w600, // Semi-bold font weight
    ),
  ),
  actions: [
    IconButton(
      icon: const Icon(Icons.search, color: Colors.white),
      onPressed: () {
        // Implement search functionality
      },
    ),
    Padding(
      padding: const EdgeInsets.only(right: 20), // Add right padding here
      child: Consumer<CartProvider>(
        builder: (_, cart, ch) => cart.itemCount > 0
            ? Badge(
                label: Text(cart.itemCount.toString()),
                child: IconButton(
                  icon: const Icon(Icons.shopping_cart,
                      color: Colors.white),
                  onPressed: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                          builder: (context) => const CartPage()),
                    );
                  },
                ),
              )
            : IconButton(
                icon:
                    const Icon(Icons.shopping_cart, color: Colors.white),
                onPressed: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder: (context) => const CartPage()),
                  );
                },
              ),
      ),
    ),
  ],
),
```

```
    body: Row(
      children: [
        // Left side - Category List
        Container(
          width: 150,
          color: Colors.white,
          child: ListView.builder(
            itemCount: categories.length,
            itemBuilder: (context, index) {
              final category = categories[index];
              final isSelected = category['id'] == selectedCategoryId;
              return Container(
                color: isSelected ? Colors.lightGreen.withOpacity(0.2) : null,
                child: ListTile(
                  title: Text(
                    category['name'],
                    style: TextStyle(
                      fontSize: 14,
                      color: isSelected ? Colors.green : Colors.black,
                    ),
                  ),
                  onTap: () {
                    setState(() {
                      selectedCategoryId = category['id'];
                    });
                  },
                ),
              );
            },
          ),
        ),
        // Vertical Divider
        Container(
          width: 1,
          color: Colors.grey[300],
        ),
        // Right side - Subcategories Grid
        Expanded(
          child: categories.isEmpty
              ? const Center(child: CircularProgressIndicator())
              : buildSubcategoriesGrid(),
        ),
      ],
    ),
  );
}
```

```
Widget buildSubcategoriesGrid() {
  final selectedCategory = categories.firstWhere(
    (category) => category['id'] == selectedCategoryId,
    orElse: () => categories[0],
  );

  final subcategories =
      selectedCategory['subcategories'] as Map<String, dynamic>;

  return GridView.builder(
    padding: const EdgeInsets.all(16),
    gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
      crossAxisCount: 2, // Keeps two tiles per row
      crossAxisSpacing: 16,
      mainAxisSpacing: 16,
      childAspectRatio: 0.75, // Decreased to make tiles bigger
    ),
    itemCount: subcategories.length,
    itemBuilder: (context, index) {
      String subcategoryId = subcategories.keys.elementAt(index);
      Map<String, dynamic> subcategoryData = subcategories[subcategoryId];

      return GestureDetector(
        onTap: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => ProductListPage(
                categoryId: selectedCategoryId,
                subcategoryId: subcategoryId,
                subcategoryName: subcategoryData['name'] ?? 'Unknown',
              ),
            ),
          );
        },
        child: buildSubcategoryTile(
          subcategoryData['name'] ?? 'Unknown',
          subcategoryData['image'] ?? '',
        ),
      );
    },
  );
}

Widget buildSubcategoryTile(String title, String imageUrl) {
  return SizedBox(
    width: 160,
```

```
height: 180, // Adjusted height to prevent overflow
child: Card(
  color: Colors.white,
  elevation: 5,
  shape: RoundedRectangleBorder(
    borderRadius: BorderRadius.circular(12),
  ),
  shadowColor: Colors.black38,
  child: Padding(
    padding: const EdgeInsets.all(8.0),
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      mainAxisSize: MainAxisSize
          .min, // Ensures content doesn't expand unnecessarily
      children: [
        SizedBox(
          width: 80,
          height: 80,
          child: imageUrl.isNotEmpty
              ? Image.network(
                  imageUrl,
                  width: 80,
                  height: 80,
                  fit: BoxFit.cover,
                  errorBuilder: (context, error, stackTrace) {
                    return const Icon(Icons.image_not_supported,
                        size: 80);
                  },
                )
              : const Icon(Icons.image_not_supported, size: 80),
        ),
        Expanded(
          // This ensures text doesn't push content beyond limits
          child: Text(
            title,
            style: const TextStyle(
              fontWeight: FontWeight.bold,
              fontSize: 12,
            ),
            textAlign: TextAlign.center,
            maxLines: 2,
            overflow: TextOverflow.ellipsis,
          ),
        ),
      ],
    ),
  ),
```

```
        ),
      );
    }
  }
```

```dart
//productList_page.dart
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:dmart/providers/cart_provider.dart';
import 'package:dmart/providers/favorite_provider.dart';
import 'package:dmart/screens/cart/cart_page.dart';

class ProductListPage extends StatefulWidget {
  final String categoryId;
  final String subcategoryId;
  final String subcategoryName;

  const ProductListPage({
    super.key,
    required this.categoryId,
    required this.subcategoryId,
    required this.subcategoryName,
  });

  @override
  State<ProductListPage> createState() => _ProductListPageState();
}

class _ProductListPageState extends State<ProductListPage> {
  List<Map<String, dynamic>> products = [];
  // Map<String, int> productQuantities = {};
  // Map<String, bool> wishlist = {};
  bool isLoading = true;

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

  Future<void> fetchProducts() async {
    try {
      var doc = await FirebaseFirestore.instance
          .collection('categories')
          .doc(widget.categoryId)
          .get();
      if (!doc.exists) {
        setState(() => isLoading = false);
        return;
      }
      var data = doc.data();
```

```
    if (data == null) {
      setState(() => isLoading = false);
      return;
    }
    var subcategories = data['subcategories'] as Map<String, dynamic>?;
    var subcategory =
        subcategories?[widget.subcategoryId] as Map<String, dynamic>?;
    var productsList = subcategory?['products'] as List<dynamic>?;
    if (productsList != null) {
      setState(() {
        products = productsList.map((product) {
          return {
            'id': product['id'] ?? '',
            'name': product['name'] ?? 'Unknown Product',
            'imageUrl': product['imageUrl'] ?? '',
            'dmartPrice': product['dmartPrice']?.toString() ?? '0',
            'mrpPrice': product['mrpPrice']?.toString() ?? '0',
            'quantity': product['quantity'] ?? '',
            'isOutOfStock': product['isOutOfStock'] ?? false,
          };
        }).toList();
        isLoading = false;
      });
    } else {
      setState(() => isLoading = false);
    }
  } catch (e) {
    print("Error fetching products: $e");
    setState(() => isLoading = false);
  }
}
void incrementQuantity(String productId) {
  final cartProvider = Provider.of<CartProvider>(context, listen: false);
  final product = products.firstWhere((p) => p['id'] == productId);

  cartProvider.addItem(
    productId: productId,
    name: product['name'],
    dmartPrice: double.parse(product['dmartPrice']),
    mrpPrice: double.parse(product['mrpPrice']),
    imageUrl: product['imageUrl'],
    quantity: product['quantity'],
  );
}
void decrementQuantity(String productId) {
  final cartProvider = Provider.of<CartProvider>(context, listen: false);
  cartProvider.removeItem(productId);
```

```
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Colors.green,
      title: Text(
        widget.subcategoryName,
        style: const TextStyle(
          color: Colors.white,
          fontWeight: FontWeight.w600,
        ),
      ),
      iconTheme: const IconThemeData(color: Colors.white),
      actions: [
        IconButton(
          icon: const Icon(Icons.search, color: Colors.white),
          onPressed: () {
            // Implement search functionality
          },
        ),
        Padding(
          padding: const EdgeInsets.only(right: 20), // Add right padding here
          child: Consumer<CartProvider>(
            builder: (_, cart, ch) => cart.itemCount > 0
                ? Badge(
                    label: Text(cart.itemCount.toString()),
                    child: IconButton(
                      icon: const Icon(Icons.shopping_cart,
                          color: Colors.white),
                      onPressed: () {
                        Navigator.push(
                          context,
                          MaterialPageRoute(
                              builder: (context) => const CartPage()),
                        );
                      },
                    ),
                  )
                : IconButton(
                    icon:
                        const Icon(Icons.shopping_cart, color: Colors.white),
                    onPressed: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(
```

```
                            builder: (context) => const CartPage()),
                      );
                    },
                  ),
          ),
        ),
      ],
    ),
    body: isLoading
        ? const Center(child: CircularProgressIndicator())
        : products.isEmpty
            ? const Center(
                child: Text(
                  'No products available',
                  style: TextStyle(fontSize: 16),
                ),
              )
            : ListView.builder(
                padding: const EdgeInsets.all(8),
                itemCount: products.length,
                itemBuilder: (context, index) {
                  final product = products[index];
                  return buildProductTile(product);
                },
              ),
  );
}

Widget buildProductTile(Map<String, dynamic> product) {
  String productId = product['id'];

  print("productId ID: ${productId}");
  print("productcheck: ${product}");

  return Consumer2<CartProvider, FavoriteProvider>(
    builder: (context, cart, favorite, child) {
      int quantity = cart.getItemQuantity(productId);
      bool isWishlisted = favorite.isInFavorites(productId);

      return Card(
        margin: const EdgeInsets.symmetric(horizontal: 10, vertical: 5),
        elevation: 4,
        shadowColor: Colors.black26,
        color: Colors.white,
        shape:
            RoundedRectangleBorder(borderRadius: BorderRadius.circular(10)),
        child: Padding(
```

```
padding: const EdgeInsets.all(10.0),
child: Row(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
    ClipRRect(
      borderRadius: BorderRadius.circular(8),
      child: Image.network(
        product['imageUrl'] ?? '',
        width: 100,
        height: 120,
        fit: BoxFit.cover,
        errorBuilder: (context, error, stackTrace) =>
            const Icon(Icons.image_not_supported, size: 100),
      ),
    ),
    const SizedBox(width: 10),
    Expanded(
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(
            product['name'] ?? 'Unknown Product',
            style: const TextStyle(
              fontWeight: FontWeight.bold,
              fontSize: 16,
            ),
            maxLines: 2,
            overflow: TextOverflow.ellipsis,
          ),
          const SizedBox(height: 5),
          Align(
            alignment: Alignment.centerLeft,
            child: Container(
              width: double.infinity,
              padding: const EdgeInsets.symmetric(
                  horizontal: 6, vertical: 4),
              decoration: BoxDecoration(
                border: Border.all(color: Colors.grey, width: 1),
                borderRadius: BorderRadius.circular(5),
              ),
              child: Text(
                "Qty: ${product['quantity']}",
                style: const TextStyle(
                    fontSize: 12, fontWeight: FontWeight.w500),
              ),
            ),
          ),
```

```
                    Row(
                      mainAxisAlignment: MainAxisAlignment.spaceBetween,
                      crossAxisAlignment: CrossAxisAlignment.center,
                      children: [
                        Column(
                          crossAxisAlignment: CrossAxisAlignment.start,
                          children: [
                            Text(
                              "DMart: ₹${product['dmartPrice']}",
                              style: const TextStyle(
                                fontSize: 14,
                                color: Colors.green,
                                fontWeight: FontWeight.bold,
                              ),
                            ),
                            const SizedBox(height: 2),
                            Text(
                              "MRP: ₹${product['mrpPrice']}",
                              style: const TextStyle(
                                fontSize: 14,
                                decoration: TextDecoration.lineThrough,
                                color: Colors.grey,
                              ),
                            ),
                          ],
                        ),
                        const SizedBox(width: 10),
                        Expanded(
                          child: Align(
                            alignment: Alignment.centerRight,
                            child: Container(
                              padding: const EdgeInsets.symmetric(
                                  horizontal: 8, vertical: 4),
                              decoration: BoxDecoration(
                                color: Colors.amber.shade100,
                                borderRadius: BorderRadius.circular(10),
                              ),
                              child: Text(
                                "₹${(int.parse(product['mrpPrice']) -
int.parse(product['dmartPrice']))} OFF",
                                style: const TextStyle(
                                  fontSize: 12,
                                  fontWeight: FontWeight.bold,
                                  color: Colors.brown,
                                ),
                              ),
                            ),
```

```
      ),
    ),
  ],
),
const SizedBox(height: 10),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceBetween,
  children: [
    Container(
      decoration: BoxDecoration(
        color: Colors.white,
        border:
            Border.all(color: Colors.white70, width: 2),
        borderRadius: BorderRadius.circular(5),
        boxShadow: [
          BoxShadow(
            color: Colors.grey.withOpacity(0.5),
            blurRadius: 4,
            offset: const Offset(2, 2),
          ),
        ],
      ),
      child: IconButton(
        onPressed: () {
          // Toggle favorite status
          favorite.toggleFavorite(
            productId: productId,
            name: product['name'],
            dmartPrice:
                double.parse(product['dmartPrice']),
            imageUrl: product['imageUrl'],
            quantity: product['quantity'],
          );
        },
        icon: Icon(
          favorite.isInFavorites(productId)
              ? Icons.favorite
              : Icons.favorite_border,
          color: Colors.green,
        ),
      ),
    ),
    product['isOutOfStock']
        ? const Text(
            "Out Of Stock",
            style: TextStyle(
              color: Colors.red,
```

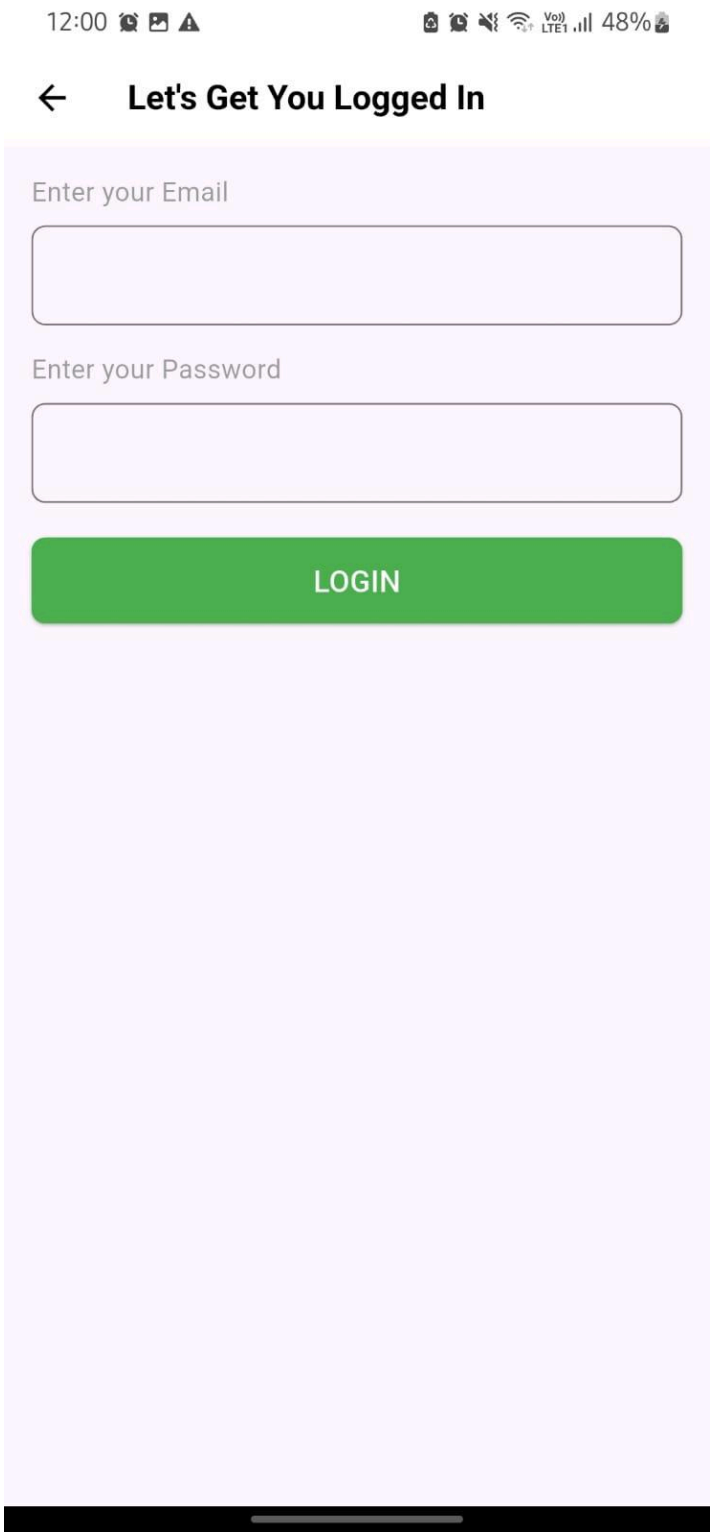
```
                fontWeight: FontWeight.bold,
              ),
            )
          : quantity == 0
              ? SizedBox(
                  width: 120,
                  child: ElevatedButton(
                    onPressed: () {
                      cart.addItem(
                        productId: productId,
                        name: product['name'],
                        dmartPrice: double.parse(
                            product['dmartPrice']),
                        mrpPrice: double.parse(
                            product['mrpPrice']),
                        imageUrl: product['imageUrl'],
                        quantity: product['quantity'],
                      );
                    },
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.green,
                      shape: RoundedRectangleBorder(
                        borderRadius:
                            BorderRadius.circular(5),
                      ),
                      padding: const EdgeInsets.symmetric(
                          vertical: 12),
                    ),
                    child: const Row(
                      mainAxisAlignment:
                          MainAxisAlignment.center,
                      children: [
                        Icon(Icons.shopping_cart,
                            size: 18, color: Colors.white),
                        SizedBox(width: 5),
                        Text(
                          'ADD',
                          style: TextStyle(
                            color: Colors.white,
                            fontWeight: FontWeight.w500,
                          ),
                        ),
                      ],
                    ),
                  ),
                )
              : Container(
```

```
width: 120,
decoration: BoxDecoration(
  border: Border.all(
      color: Colors.green, width: 2),
  borderRadius: BorderRadius.circular(5),
),
child: Row(
  children: [
    Expanded(
      child: SizedBox(
        height: 40,
        child: ElevatedButton(
          onPressed: () =>
              cart.removeItem(productId),
          style: ElevatedButton.styleFrom(
            backgroundColor: Colors.green,
            alignment: Alignment.center,
            minimumSize:
                const Size(0, 40),
            padding: EdgeInsets.zero,
            shape:
                const RoundedRectangleBorder(
            borderRadius:
                BorderRadius.only(
            topLeft:
                Radius.circular(3),
            bottomLeft:
                Radius.circular(3),
            ),
          ),
        ),
        child: const Icon(Icons.remove,
            color: Colors.white,
            size: 24),
      ),
    ),
    Container(
      alignment: Alignment.center,
      width: 40,
      child: Text(
        quantity.toString(),
        style: const TextStyle(
          fontSize: 16,
          fontWeight: FontWeight.bold,
        ),
      ),
```

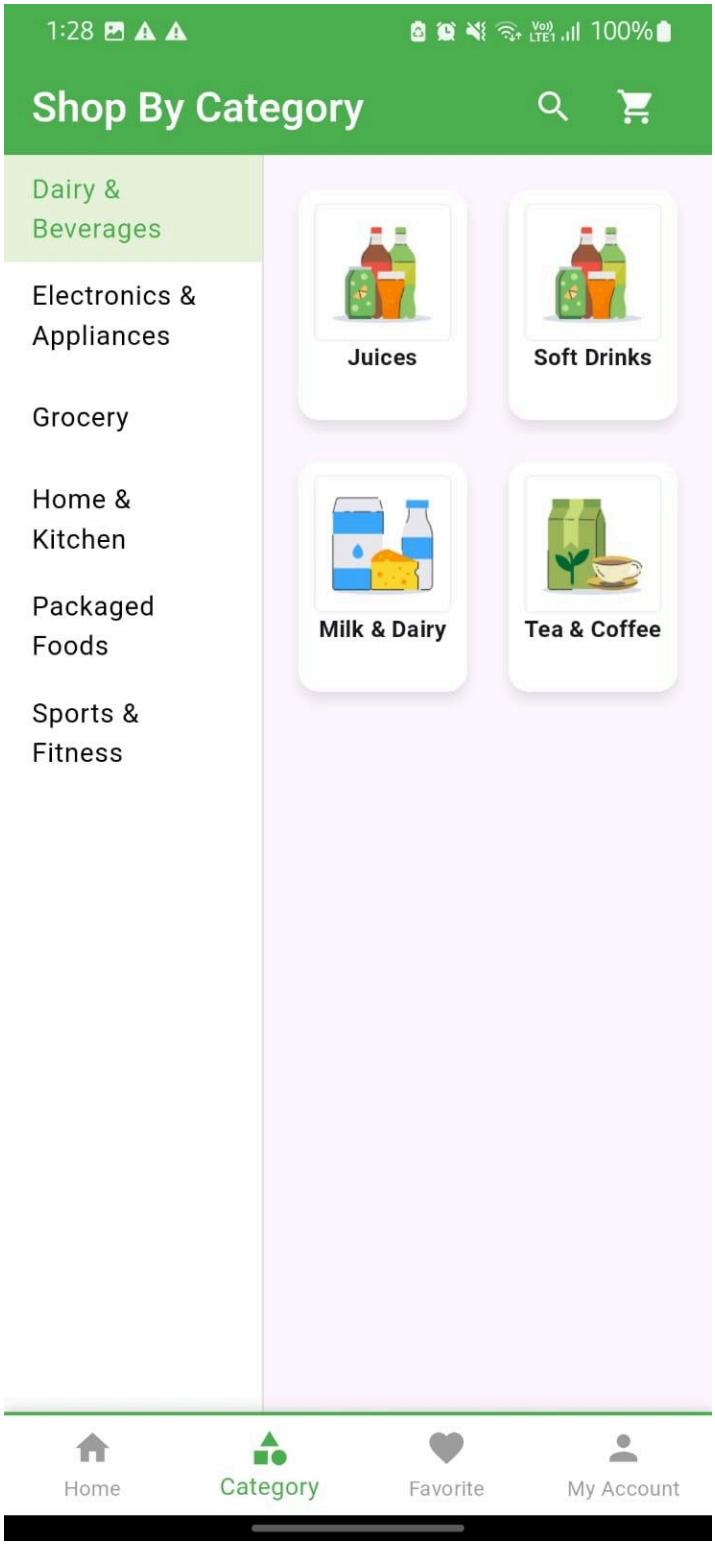
```
                                ),
                                Expanded(
                                  child: SizedBox(
                                    height: 40,
                                    child: ElevatedButton(
                                      onPressed: () => cart.addItem(
                                        productId: productId,
                                        name: product['name'],
                                        dmartPrice: double.parse(
                                            product['dmartPrice']),
                                        mrpPrice: double.parse(
                                            product['mrpPrice']),
                                        imageUrl: product['imageUrl'],
                                        quantity: product['quantity'],
                                      ),
                                      style: ElevatedButton.styleFrom(
                                        backgroundColor: Colors.green,
                                        alignment: Alignment.center,
                                        minimumSize:
                                            const Size(0, 40),
                                        padding: EdgeInsets.zero,
                                        shape:
                                            const RoundedRectangleBorder(
                                          borderRadius:
                                              BorderRadius.only(
                                          topRight:
                                              Radius.circular(3),
                                          bottomRight:
                                              Radius.circular(3),
                                          ),
                                        ),
                                      ),
                                      child: const Icon(Icons.add,
                                          color: Colors.white,
                                          size: 24),
                                    ),
                                  ),
                                ),
                              ],
                            ),
                          ),
                        ],
                      ),
                    ),
                  ),
                ],
```

```
            ),
          ),
        );
      },
    );
  }
}
```

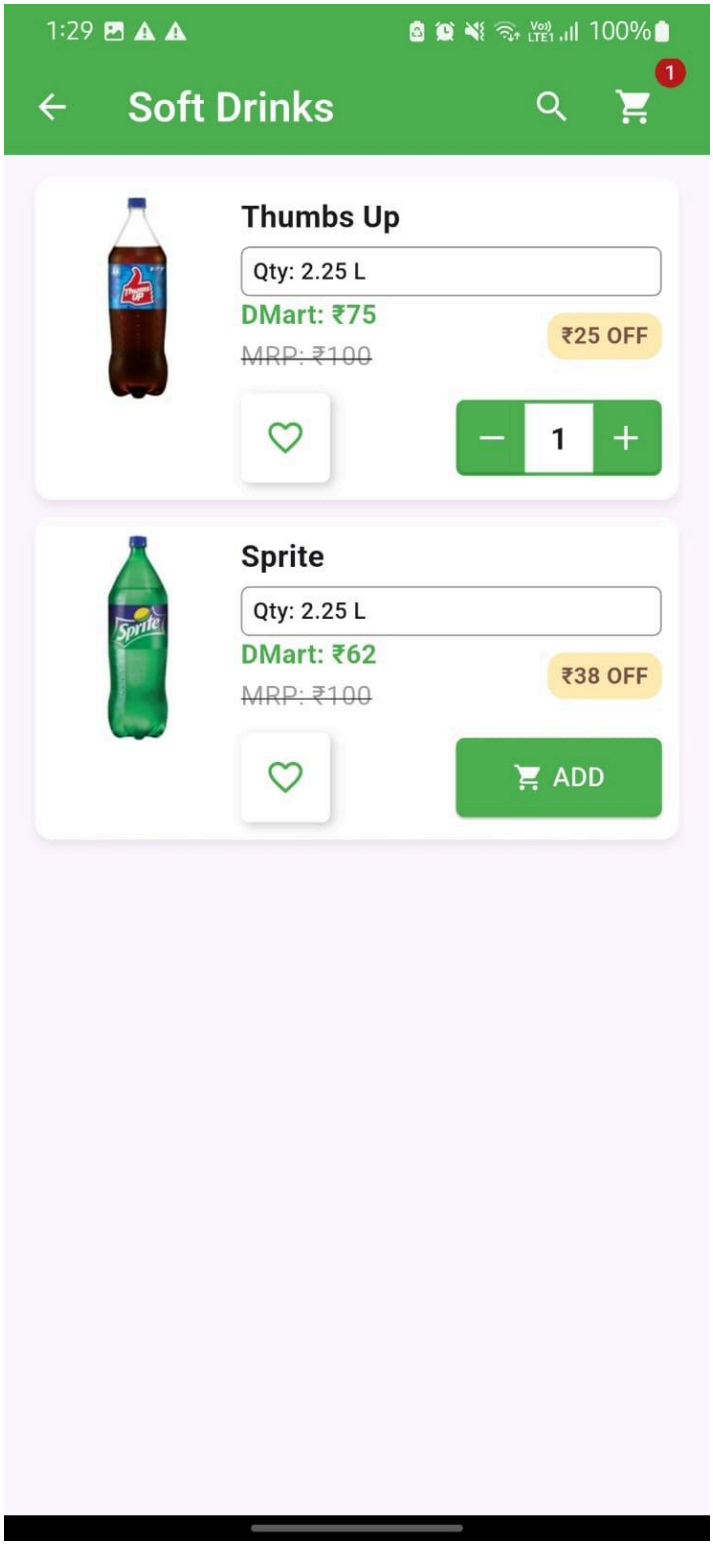
Output:
Login Page:



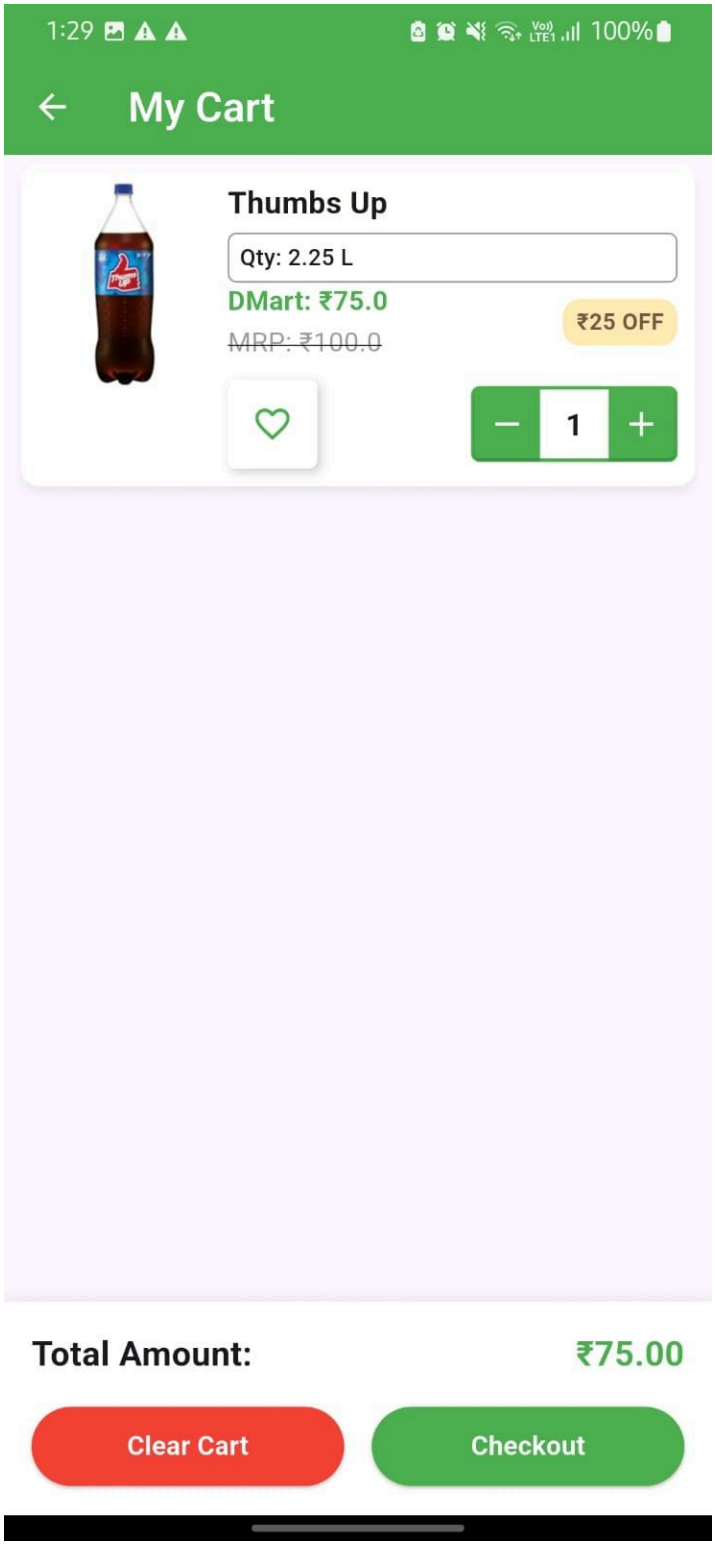
Category Page:



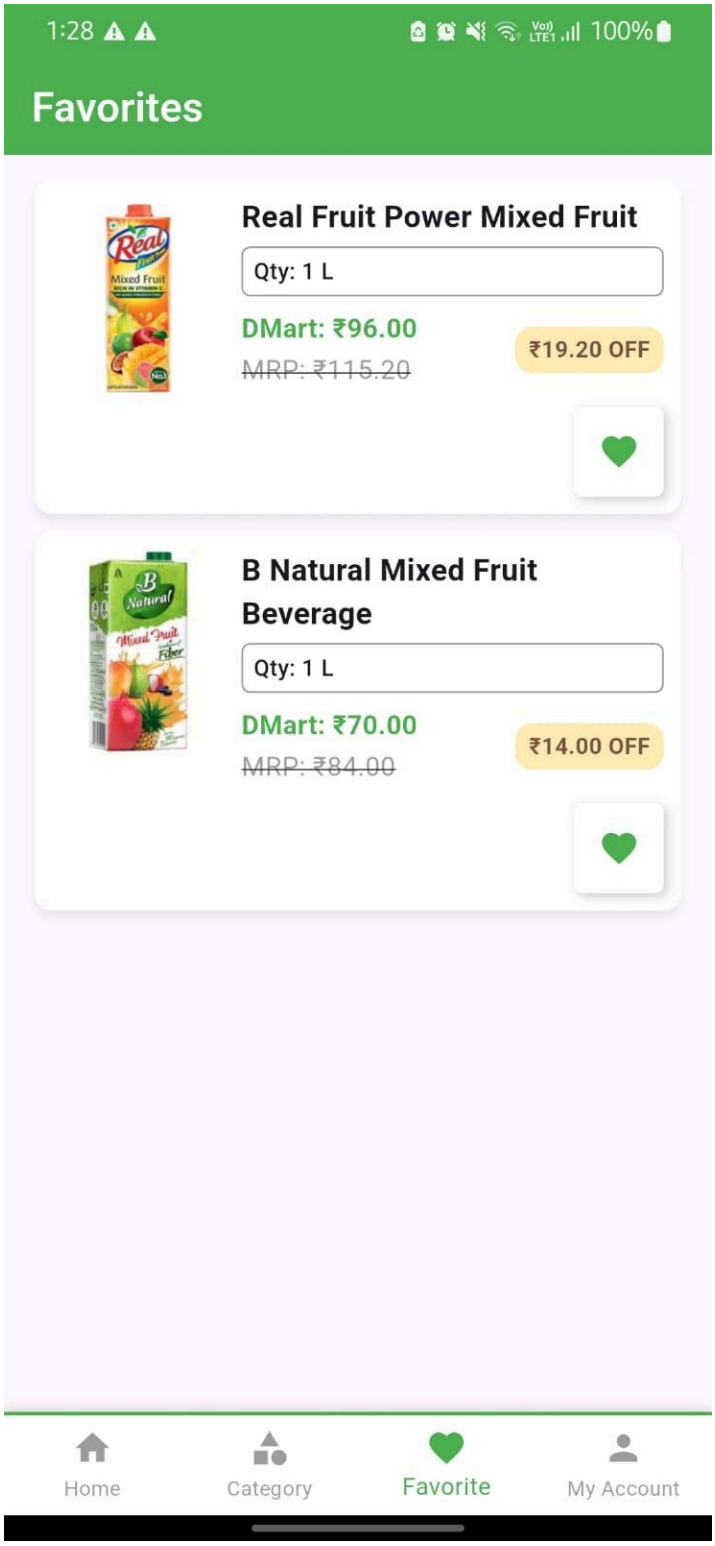
Product List Page:



Cart Page:


1:29
100%
My Cart
Thumbs Up
Qty: 2.25 L
DMart: ₹75.0
MRP: ₹100.0
₹25 OFF
1
Total Amount:
₹75.00
Clear Cart
Checkout

Favorite Page:



MAD & PWA Lab
Journal

Experiment No.	07
Experiment Title.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.
Roll No.	28
Name	Anuprita Mhapankar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

MPL Experiment 7 (PWA)

Name: Anuprita Mhapankar **Class:** D15A **Roll no:**28

Aim: To write meta data of your PWA in a Web app manifest file to enable “add to homescreen feature”.

Theory:

Regular Web App
A regular web app is a website that is designed to be accessible on all mobile devices such that the content gets fit as per the device screen. It is designed using a web technology stack (HTML, CSS, JavaScript, Ruby, etc.) and operates via a browser. They offer various native-device features and functionalities. However, it entirely depends on the browser the user is using. In other words, it might be possible that you can access a native-device feature on Chrome but not on Safari or Mozilla Firefox because the browsers are incompatible with that feature.

Progressive Web App
Progressive Web App (PWA) is a regular web app, but some extras enable it to deliver an excellent user experience. It is a perfect blend of desktop and mobile application experience to give both platforms to the end-users.

Difference between PWAs vs. Regular Web Apps:
Progressive Web is different and better than a Regular Web app with features like:

1. Native Experience
   Though a PWA runs on web technologies (HTML, CSS, JavaScript) like a Regular web app, it gives user experience like a native mobile application. It can use most native device features, including push notifications, without relying on the browser or any other entity. It offers a seamless and integrated user experience that it is quite tough for one to differentiate between a PWA and a Native application by considering its look and feel.
2. Ease of Access
   Unlike other mobile apps, PWAs do not demand longer download time and make memory space available for installing the applications. The PWAs can be shared and installed by a link, which cuts down the number of steps to install and use. These applications can easily keep an app icon on the user's home screen, making the app easily accessible to the users and helps the brands remain in the users' minds, and improving the chances of interaction.
3. Faster Services
   PWAs can cache the data and serve the user with text stylesheets, images, and other web content even before the page loads completely. This lowers the waiting time for the end-users and helps the brands improve the user engagement and retention rate, which eventually adds value to their business.

Features of the Progressive Web App

The main features are:

- **Progressive**
  They work for every user, regardless of the browser chosen because they are built at the base with progressive improvement principles.
- **Responsive**
  They adapt to the various screen sizes: desktop, mobile, tablet, or dimensions that can later become available.
- **App-like**
  They behave with the user as if they were native apps, in terms of interaction and navigation.
- **Updated**
  Information is always up-to-date thanks to the data update process offered by service workers.
- **Secure**
  Exposed over HTTPS protocol to prevent the connection from displaying information or altering the contents.
- **Searchable**
  They are identified as “applications” and are indexed by search engines.
- **Reactivable**
  Make it easy to reactivate the application thanks to capabilities such as web notifications.
- **Installable**
  They allow the user to “save” the apps that he considers most useful with the corresponding icon on the screen of his mobile terminal (home screen) without having to face all the steps and problems related to the use of the app store.
- **Linkable**
  Easily shared via URL without complex installations.

Code:

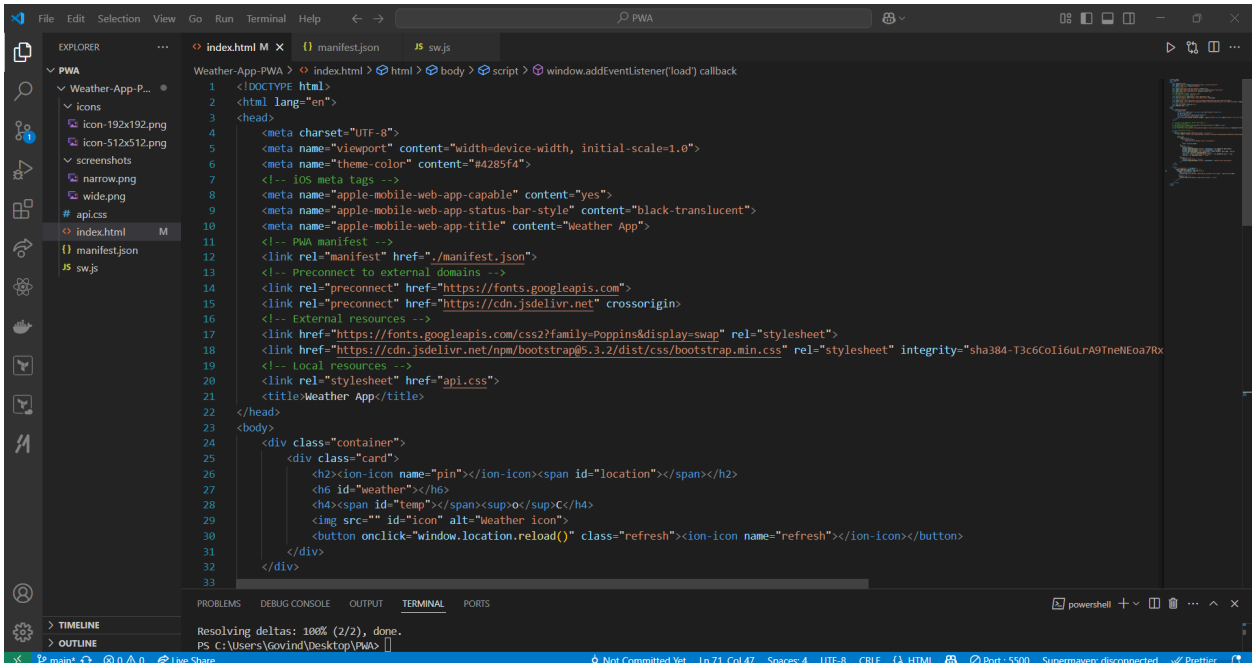
```
//Manifest.json
{
    "name": "Weather App",
    "short_name": "Weather",
    "description": "A progressive weather application",
    "start_url": "/index.html",
    "scope": "/",
    "display": "standalone",
    "orientation": "portrait-primary",
    "background_color": "#ffffff",
    "theme_color": "#4285f4",
    "categories": ["weather", "utilities"],
    "lang": "en-US",
```

```
"dir": "ltr",
"icons": [
  {
    "src": "icons/icon-72x72.png",
    "sizes": "72x72",
    "type": "image/png",
    "purpose": "any"
  },
  {
    "src": "icons/icon-96x96.png",
    "sizes": "96x96",
    "type": "image/png",
    "purpose": "any"
  },
  {
    "src": "icons/icon-128x128.png",
    "sizes": "128x128",
    "type": "image/png",
    "purpose": "any"
  },
  {
    "src": "icons/icon-144x144.png",
    "sizes": "144x144",
    "type": "image/png",
    "purpose": "any"
  },
  {
    "src": "icons/icon-152x152.png",
    "sizes": "152x152",
    "type": "image/png",
    "purpose": "any"
  },
  {
    "src": "icons/icon-192x192.png",
    "sizes": "192x192",
    "type": "image/png",
    "purpose": "any maskable"
  },
```

```
    {
      "src": "icons/icon-384x384.png",
      "sizes": "384x384",
      "type": "image/png",
      "purpose": "any"
    },
    {
      "src": "icons/icon-512x512.png",
      "sizes": "512x512",
      "type": "image/png",
      "purpose": "any maskable"
    }
  ],
  "screenshots": [
    {
      "src": "screenshots/wide.png",
      "sizes": "1280x720",
      "type": "image/png",
      "form_factor": "wide",
      "label": "Desktop view"
    },
    {
      "src": "screenshots/narrow.png",
      "sizes": "412x915",
      "type": "image/png",
      "form_factor": "narrow",
      "label": "Mobile view"
    }
  ],
  "shortcuts": [
    {
      "name": "Current Weather",
      "short_name": "Now",
      "description": "View current weather conditions",
      "url": "/?view=current",
      "icons": [
        {
          "src": "icons/icon-96x96.png",
```

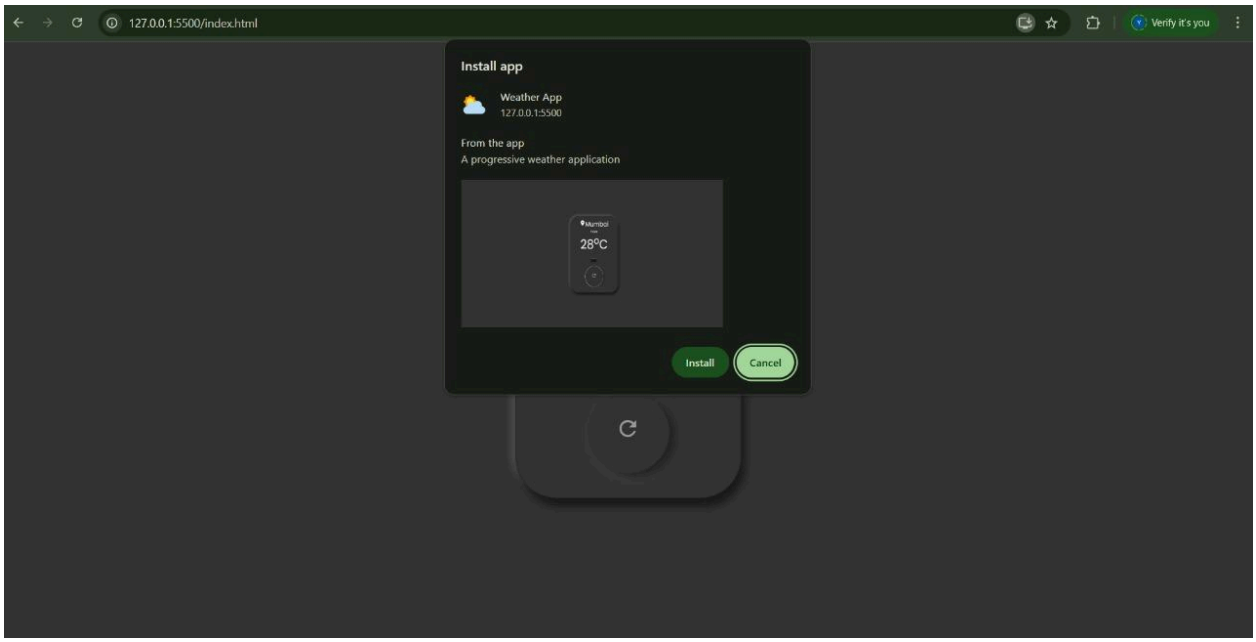
```
          "sizes": "96x96"
        }
      ]
    },
    {
      "name": "Forecast",
      "short_name": "Forecast",
      "description": "View weather forecast",
      "url": "/?view=forecast",
      "icons": [
        {
          "src": "icons/icon-96x96.png",
          "sizes": "96x96"
        }
      ]
    }
  ]
}
```

Output:

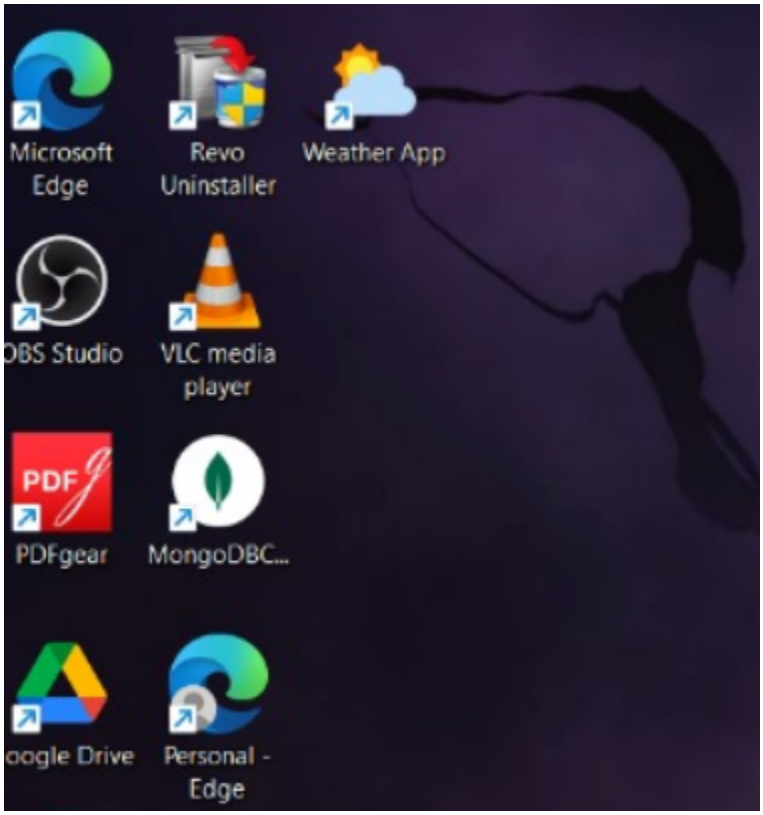

File Edit Selection View Go Run Terminal Help
PWA
EXPLORER
index.html M
manifest.json
sw.js
Weather-App-PWA > index.html > html > body > script > window.addEventListener('load') callback
PWA
Weather-App-P...
icons
icon-192x192.png
icon-512x512.png
screenshots
narrow.png
wide.png
api.css
index.html M
manifest.json
sw.js
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="theme-color" content="#4285f4">
<!-- iOS meta tags -->
<meta name="apple-mobile-web-app-capable" content="yes">
<meta name="apple-mobile-web-app-status-bar-style" content="black-translucent">
<meta name="apple-mobile-web-app-title" content="Weather App">
<!-- PWA manifest -->
<link rel="manifest" href="./manifest.json">
<!-- Preconnect to external domains -->
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://cdn.jsdelivr.net" crossorigin>
<!-- External resources -->
<link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7Rx
<!-- Local resources -->
<link rel="stylesheet" href="api.css">
<title>Weather App</title>
</head>
<body>

<h2><ion-icon name="pin"></ion-icon></h2>
<h6 id="weather"></h6>
<h4>^oC</h4>
<img src="" id="icon" alt="Weather icon">
<button onclick="window.location.reload()" class="refresh"><ion-icon name="refresh"></ion-icon></button>

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS
powershell
TIMELINE
OUTLINE
Resolving deltas: 100% (2/2), done.
PS C:\Users\Govind\Desktop\PWA>


127.0.0.1:5500/index.html
Verify it's you
Install app
Weather App
127.0.0.1:5500
From the app
A progressive weather application
28°C
Install
Cancel

Shortcut added to Home Screen:



MAD & PWA Lab
Journal

Experiment No.	08
Experiment Title.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA
Roll No.	28
Name	Anuprita Mhapankar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

MPL Experiment 8 (PWA)

Name: Anuprita Mhapankar **Class:** D15A **Roll no:** 28

Aim: To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA.

Theory:

Service Worker

Service Worker is a script that works on browser background without user interaction independently. Also, It resembles a proxy that works on the user side. With this script, you can track network traffic of the page, manage push notifications and develop “offline first” web applications with Cache API.

Things to note about Service Worker:

- A service worker is a programmable network proxy that lets you control how network requests from your page are handled.
- Service workers only run over HTTPS. Because service workers can intercept network requests and modify responses, "man-in-the-middle" attacks could be very bad.
- The service worker becomes idle when not in use and restarts when it's next needed. You cannot rely on a global state persisting between events. If there is information that you need to persist and reuse across restarts, you can use IndexedDB databases.

What can we do with Service Workers?

- You can manage all network traffic of the page and do any manipulations. For example, when the page requests a CSS file, you can send plain text as a response or when the page requests an HTML file, you can send a png file as a response. You can also send a true response too.

- You can Cache
  You can cache any request/response pair with Service Worker and Cache API and you can access these offline content anytime.

- You can manage Push Notifications
  You can manage push notifications with Service Worker and show any information message to the user.

- You can Continue
  Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

What can't we do with Service Workers?

- You can't access the Window
  You can't access the window, therefore, You can't manipulate DOM elements. But, you can communicate to the window through post Message and manage processes that you want.

- You can't work it on 80 Port
  Service Worker just can work on HTTPS protocol. But you can work on localhost during development.

Codes:

```
//Serviceworker.js

// sw.js - Complete Service Worker for E-commerce PWA
const CACHE_NAME = 'ecommerce-pwa-v2';
const API_CACHE = 'ecommerce-api-v1';
const ASSETS_TO_CACHE = [
  '/',
  '/index.html',
  '/manifest.json',
  '/offline.html',
  '/css/main.min.css',
  '/js/app.min.js',
  '/icons/icon-192x192.png',
  '/icons/icon-512x512.png',
  '/images/placeholder-product.jpg'
];

// ======================
// Install Event
// ======================
self.addEventListener('install', (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => {
        console.log('[Service Worker] Cache opened');
        return cache.addAll(ASSETS_TO_CACHE);
      })
```

```
      .then(() => self.skipWaiting())
  );
});

// ======================
// Activate Event
// ======================
self.addEventListener('activate', (event) => {
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cacheName) => {
          if (cacheName !== CACHE_NAME && cacheName !== API_CACHE) {
            console.log('[Service Worker] Deleting old cache:',
cacheName);
            return caches.delete(cacheName);
          }
        })
      );
    })
    .then(() => self.clients.claim())
  );
});

// ======================
// Fetch Event
// ======================
self.addEventListener('fetch', (event) => {
  const { request } = event;
  const url = new URL(request.url);

  // 1. Skip non-GET requests and chrome-extension
  if (request.method !== 'GET' || url.protocol ===
'chrome-extension:') {
    return;
  }

  // 2. API Requests (Network First with Cache Fallback)
```

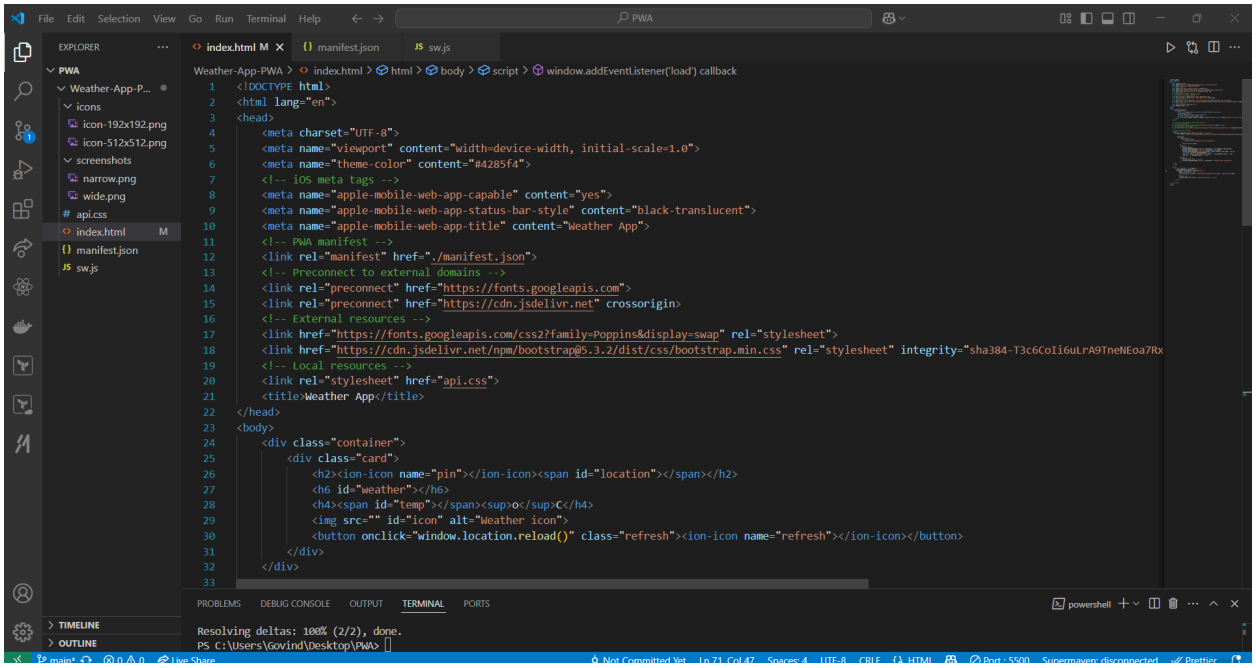
```
if (url.pathname.startsWith('/api/')) {
  event.respondWith(
    fetch(request)
      .then(networkResponse => {
        // Cache successful API responses
        if (networkResponse.ok) {
          const clone = networkResponse.clone();
          caches.open(API_CACHE)
            .then(cache => cache.put(request, clone));
        }
        return networkResponse;
      })
      .catch(() => {
        // Return cached version if available
        return caches.match(request)
          .then(cachedResponse => cachedResponse || Response.json(
            { error: 'Network error' },
            { status: 503 }
          ));
      })
  );
  return;
}

// 3. Static Assets (Cache First with Network Fallback)
event.respondWith(
  caches.match(request)
    .then(cachedResponse => {
      // Return cached version if found
      if (cachedResponse) {
        return cachedResponse;
      }

      // Otherwise fetch from network
      return fetch(request)
        .then(networkResponse => {
          // Cache successful responses
          if (networkResponse.ok) {
```

```
            const clone = networkResponse.clone();
            caches.open(CACHE_NAME)
              .then(cache => cache.put(request, clone));
          }
          return networkResponse;
        })
        .catch(() => {
          // Special handling for HTML pages
          if (request.headers.get('accept').includes('text/html')) {
            return caches.match('/offline.html');
          }
          // Return placeholder for images
          if (request.headers.get('accept').includes('image')) {
            return caches.match('/images/placeholder-product.jpg');
          }
        });
    })
  );
});

// ======================
// Background Sync
// ======================
self.addEventListener('sync', (event) => {
  if (event.tag === 'sync-cart') {
    event.waitUntil(
      // Get cart data from IndexedDB
      getCartData()
        .then(cartItems => {
          return fetch('/api/cart/sync', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(cartItems)
          });
        })
        .then(() => {
          return showNotification('Cart Synced', 'Your cart has been
updated');
```

```
        })
        .catch(err => {
          console.error('Sync failed:', err);
        })
    );
  }
});

// ======================
// Push Notifications
// ======================
self.addEventListener('push', (event) => {
  let data = {};
  try {
    data = event.data.json();
  } catch (e) {
    data = {
      title: 'New Update',
      body: 'Check out our latest products!',
      icon: '/icons/icon-192x192.png',
      url: '/'
    };
  }

  const options = {
    body: data.body,
    icon: data.icon || '/icons/icon-192x192.png',
    badge: '/icons/icon-96x96.png',
    data: {
      url: data.url || '/'
    }
  };

  event.waitUntil(
    self.registration.showNotification(data.title, options)
  );
});
```

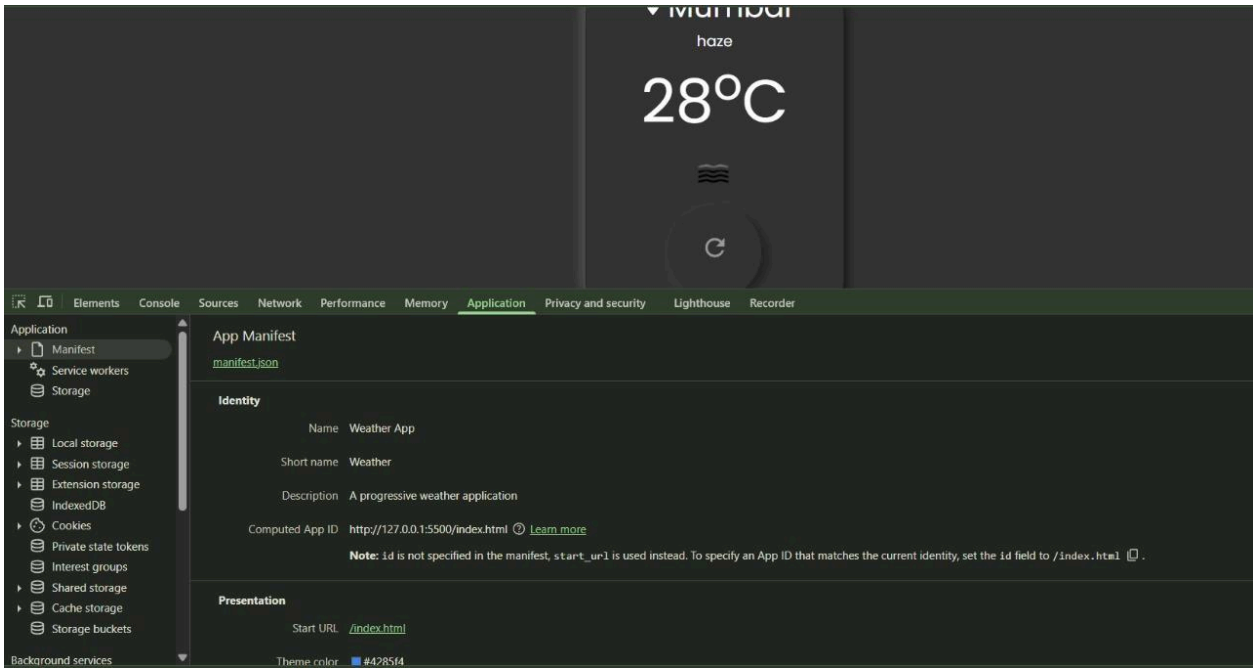
```
self.addEventListener('notificationclick', (event) => {
  event.notification.close();
  event.waitUntil(
    clients.matchAll({ type: 'window' })
      .then(clientList => {
        for (const client of clientList) {
          if (client.url === event.notification.data.url && 'focus' in
client) {
            return client.focus();
          }
        }
        if (clients.openWindow) {
          return clients.openWindow(event.notification.data.url);
        }
      })
  );
});

// ======================
// Helper Functions
// ======================
async function getCartData() {
  // In a real app, you would use IndexedDB
  return new Promise(resolve => {
    resolve([]);
  });
}

async function showNotification(title, body) {
  return self.registration.showNotification(title, { body });
}
```

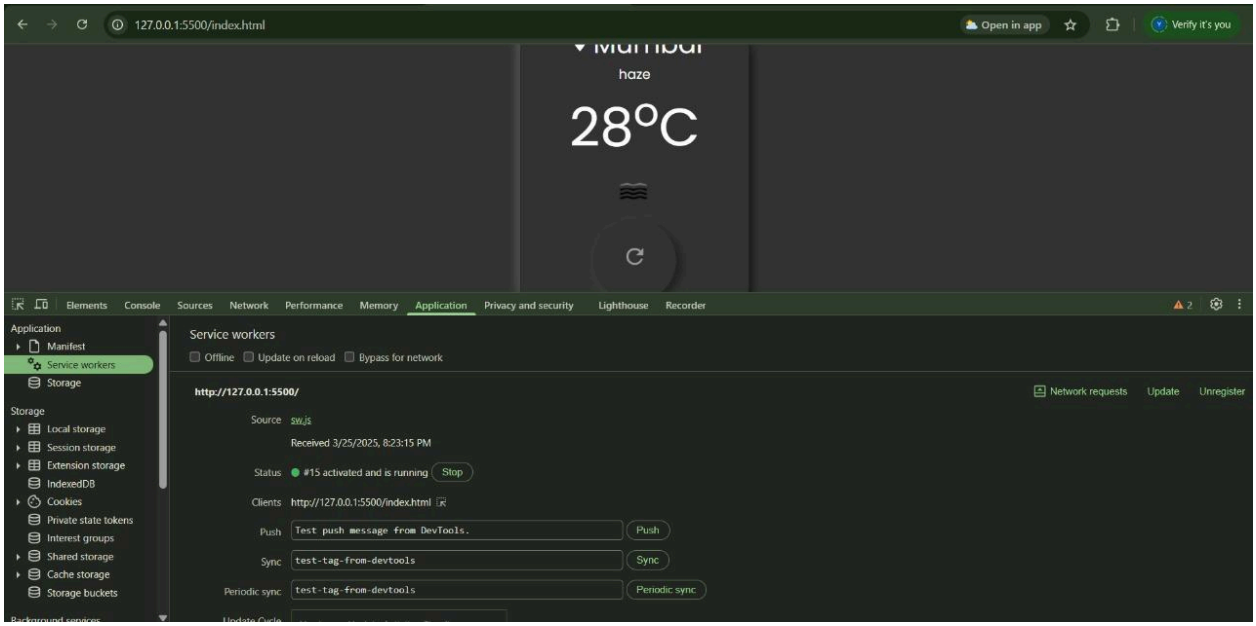
Output:


File Edit Selection View Go Run Terminal Help
PWA
EXPLORER
index.html M
manifest.json
sw.js
Weather-App-PWA > index.html > html > body > script > window.addEventListener('load') callback
PWA
Weather-App-P...
icons
icon-192x192.png
icon-512x512.png
screenshots
narrow.png
wide.png
api.css
index.html M
manifest.json
sw.js
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="theme-color" content="#4285f4">
<!-- iOS meta tags -->
<meta name="apple-mobile-web-app-capable" content="yes">
<meta name="apple-mobile-web-app-status-bar-style" content="black-translucent">
<meta name="apple-mobile-web-app-title" content="Weather App">
<!-- PWA manifest -->
<link rel="manifest" href="./manifest.json">
<!-- Preconnect to external domains -->
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://cdn.jsdelivr.net" crossorigin>
<!-- External resources -->
<link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7Rx
<!-- Local resources -->
<link rel="stylesheet" href="api.css">
<title>Weather App</title>
</head>
<body>

<h2><ion-icon name="pin"></ion-icon></h2>
<h6 id="weather"></h6>
<h4>^oC</h4>
<img src="" id="icon" alt="Weather icon">
<button onclick="window.location.reload()" class="refresh"><ion-icon name="refresh"></ion-icon></button>

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS
powershell
Resolving deltas: 100% (2/2), done.
PS C:\Users\Govind\Desktop\PWA>
TIMELINE
OUTLINE


haze
28°C
Elements Console Sources Network Performance Memory Application Privacy and security Lighthouse Recorder
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
App Manifest
manifest.json
Identity
Name Weather App
Short name Weather
Description A progressive weather application
Computed App ID http://127.0.0.1:5500/index.html Learn more
Note: id is not specified in the manifest, start_url is used instead. To specify an App ID that matches the current identity, set the id field to /index.html .
Presentation
Start URL /index.html
Theme color #4285f4


127.0.0.1:5500/index.html
Open in app
Verify it's you
haze
28°C
Elements
Console
Sources
Network
Performance
Memory
Application
Privacy and security
Lighthouse
Recorder
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Network requests
Update
Unregister
Source
sw.js
Received 3/25/2025, 8:23:15 PM
Status
#15 activated and is running
Stop
Clients
http://127.0.0.1:5500/index.html
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync

MAD & PWA Lab
Journal

Experiment No.	09
Experiment Title.	To implement Service worker events like fetch, sync and push for E-commerce PWA
Roll No.	28
Name	Anuprita Mhapankar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

MPL Experiment 9 (PWA)

Name: Anuprita Mhapankar **Class:** D15A **Roll no:** 28

Aim: To implement Service worker events like fetch, sync and push for E-commerce PWA.

Theory:

Service Worker

Service Worker is a script that works on browser background without user interaction independently. Also, It resembles a proxy that works on the user side. With this script, you can track network traffic of the page, manage push notifications and develop “offline first” web applications with Cache API.

Things to note about Service Worker:

- A service worker is a programmable network proxy that lets you control how network requests from your page are handled.
- Service workers only run over HTTPS. Because service workers can intercept network requests and modify responses, "man-in-the-middle" attacks could be very bad.
- The service worker becomes idle when not in use and restarts when it's next needed. You cannot rely on a global state persisting between events. If there is information that you need to persist and reuse across restarts, you can use IndexedDB databases.
- Service workers make extensive use of promises, so if you're new to promises, then you should stop reading this and check out Promises, an introduction.

Fetch Event

You can track and manage page network traffic with this event. You can check existing cache, manage “cache first” and “network first” requests and return a response that you want.

Of course, you can use many different methods but you can find in the following example a “cache first” and “network first” approach. In this example, if the request’s and current location’s origin are the same (Static content is requested.), this is called “cacheFirst” but if you request a targeted external URL, this is called “networkFirst”.

- **CacheFirst** - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.
- **NetworkFirst** - In this function, firstly we can try getting an updated response from the network, if this process completed successfully, the new response will be cached and returned. But if this process fails, we check whether the request has been cached

before or not. If a cache exists, it is returned to the page, but if not, this is up to you. You can return dummy content or information messages to the page.

Sync Event

Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are writing e-mail content and we didn't realize it. When completing the writing, we click the send button.

Push Event

This is the event that handles push notifications that are received from the server. You can apply any method with received data.

We can check in the following example.
"Notification.requestPermission();" is the necessary line to show notification to the user. If you don't want to show any notification, you don't need this line.

In the following code block is in sw.js file. You can handle push notifications with this event. In this example, I kept it simple. We send an object that has "method" and "message" properties. If the method value is "pushMessage", we open the information notification with the "message" property.

Code:

```
//Serviceworker.js
// sw.js - Complete Service Worker for E-commerce PWA
const CACHE_NAME = 'ecommerce-pwa-v2';
const API_CACHE = 'ecommerce-api-v1';
const ASSETS_TO_CACHE = [
  '/',
  '/index.html',
  '/manifest.json',
  '/offline.html',
  '/css/main.min.css',
  '/js/app.min.js',
  '/icons/icon-192x192.png',
  '/icons/icon-512x512.png',
```

```
  '/images/placeholder-product.jpg'
];

// ======================
// Install Event
// ======================
self.addEventListener('install', (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => {
        console.log('[Service Worker] Cache opened');
        return cache.addAll(ASSETS_TO_CACHE);
      })
      .then(() => self.skipWaiting())
  );
});

// ======================
// Activate Event
// ======================
self.addEventListener('activate', (event) => {
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cacheName) => {
          if (cacheName !== CACHE_NAME && cacheName !== API_CACHE) {
            console.log('[Service Worker] Deleting old cache:',
cacheName);
            return caches.delete(cacheName);
          }
        })
      );
    })
    .then(() => self.clients.claim())
  );
});

// ======================
```

```
// Fetch Event
// ======================
self.addEventListener('fetch', (event) => {
  const { request } = event;
  const url = new URL(request.url);

  // 1. Skip non-GET requests and chrome-extension
  if (request.method !== 'GET' || url.protocol ===
'chrome-extension:') {
    return;
  }

  // 2. API Requests (Network First with Cache Fallback)
  if (url.pathname.startsWith('/api/')) {
    event.respondWith(
      fetch(request)
        .then(networkResponse => {
          // Cache successful API responses
          if (networkResponse.ok) {
            const clone = networkResponse.clone();
            caches.open(API_CACHE)
              .then(cache => cache.put(request, clone));
          }
          return networkResponse;
        })
        .catch(() => {
          // Return cached version if available
          return caches.match(request)
            .then(cachedResponse => cachedResponse || Response.json(
              { error: 'Network error' },
              { status: 503 }
            ));
        })
    );
    return;
  }

  // 3. Static Assets (Cache First with Network Fallback)
```

```
  event.respondWith(
    caches.match(request)
      .then(cachedResponse => {
        // Return cached version if found
        if (cachedResponse) {
          return cachedResponse;
        }

        // Otherwise fetch from network
        return fetch(request)
          .then(networkResponse => {
            // Cache successful responses
            if (networkResponse.ok) {
              const clone = networkResponse.clone();
              caches.open(CACHE_NAME)
                .then(cache => cache.put(request, clone));
            }
            return networkResponse;
          })
          .catch(() => {
            // Special handling for HTML pages
            if (request.headers.get('accept').includes('text/html')) {
              return caches.match('/offline.html');
            }
            // Return placeholder for images
            if (request.headers.get('accept').includes('image')) {
              return caches.match('/images/placeholder-product.jpg');
            }
          });
      })
  );
});

// ======================
// Background Sync
// ======================
self.addEventListener('sync', (event) => {
  if (event.tag === 'sync-cart') {
```

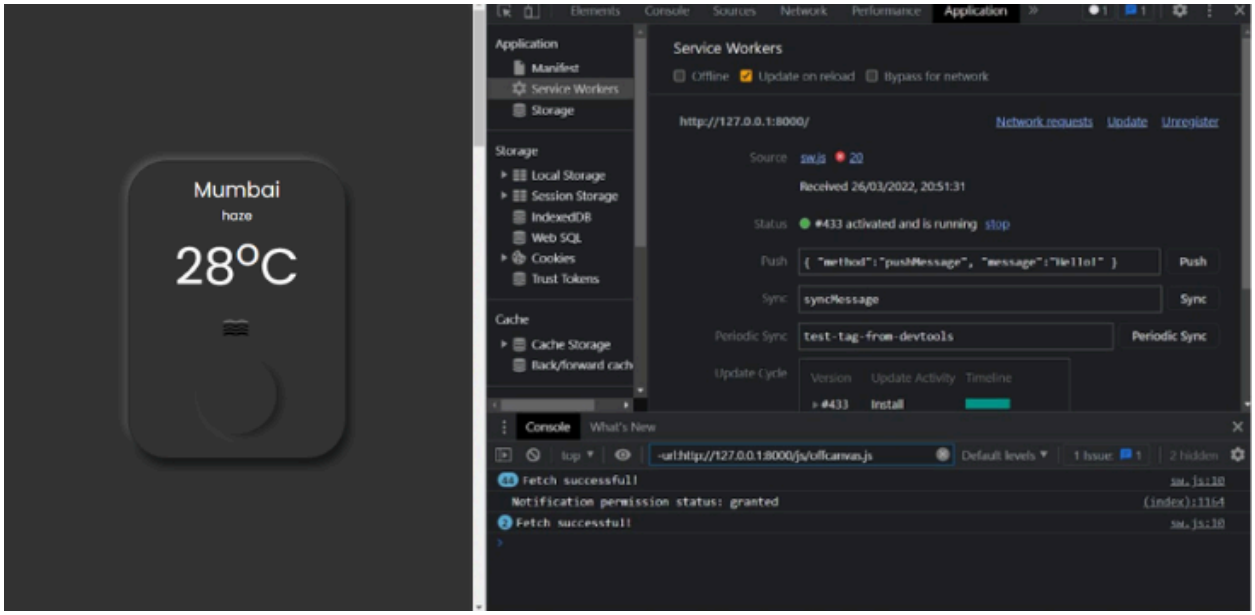
```
    event.waitUntil(
      // Get cart data from IndexedDB
      getCartData()
        .then(cartItems => {
          return fetch('/api/cart/sync', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(cartItems)
          });
        })
        .then(() => {
          return showNotification('Cart Synced', 'Your cart has been
updated');
        })
        .catch(err => {
          console.error('Sync failed:', err);
        })
    );
  }
});

// ======================
// Push Notifications
// ======================
self.addEventListener('push', (event) => {
  let data = {};
  try {
    data = event.data.json();
  } catch (e) {
    data = {
      title: 'New Update',
      body: 'Check out our latest products!',
      icon: '/icons/icon-192x192.png',
      url: '/'
    };
  }

  const options = {
```

```
    body: data.body,
    icon: data.icon || '/icons/icon-192x192.png',
    badge: '/icons/icon-96x96.png',
    data: {
      url: data.url || '/'
    }
  };

  event.waitUntil(
    self.registration.showNotification(data.title, options)
  );
});

self.addEventListener('notificationclick', (event) => {
  event.notification.close();
  event.waitUntil(
    clients.matchAll({ type: 'window' })
      .then(clientList => {
        for (const client of clientList) {
          if (client.url === event.notification.data.url && 'focus' in
client) {
            return client.focus();
          }
        }
        if (clients.openWindow) {
          return clients.openWindow(event.notification.data.url);
        }
      })
  );
});

// ======================
// Helper Functions
// ======================
async function getCartData() {
  // In a real app, you would use IndexedDB
  return new Promise(resolve => {
    resolve([]);
```

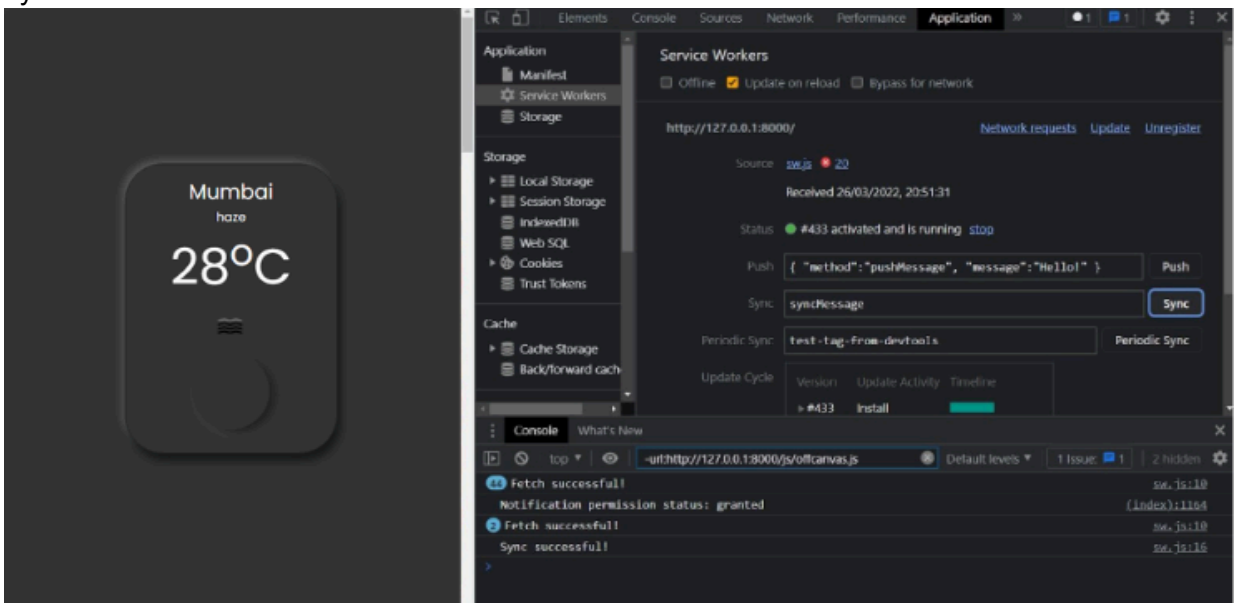
```
  });
}

async function showNotification(title, body) {
  return self.registration.showNotification(title, { body });
}
```

Output:

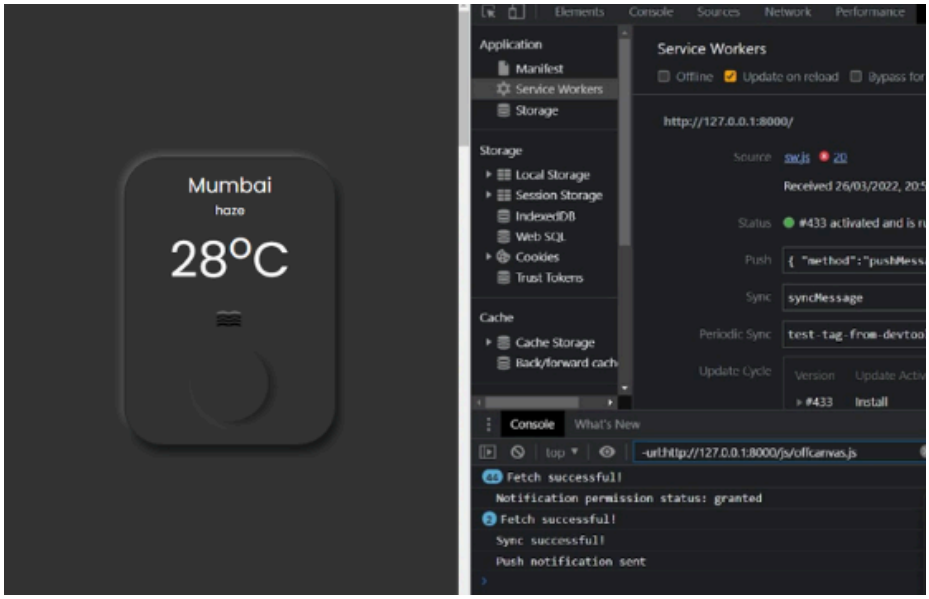
Fetch Event



Sync event



Push event



MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	28
Name	Anuprita Mhapankar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

MPL Experiment 10 (PWA)

Name: Anuprita Mhapankar **Class:** D15A **Roll no:** 28

Aim: To study and implement deployment of Ecommerce PWA to GitHub Page.

Theory:

GitHub Pages

Public web pages are freely hosted and easily published. Public webpages hosted directly from your GitHub repository. Just edit, push, and your changes are live.

GitHub Pages provides the following key features:

- Blogging with Jekyll
- Custom URL
- Automatic Page Generator

Reasons for favoring this over Firebase:

- Free to use
- Right out of github
- Quick to set up

Companies Using GitHub Pages:

GitHub Pages is used by Lyft, CircleCI, and HubSpot.
GitHub Pages is listed in **775 company stacks** and **4401 developer stacks**.

Pros

- Very familiar interface if you are already using GitHub for your projects.
- Easy to set up. Just push your static website to the `gh-pages` branch and your website is ready.
- Supports Jekyll out of the box.
- Supports custom domains. Just add a file called CNAME to the root of your site, add an
- A record in the site's DNS configuration, and you are done.

Cons

- The code of your website will be public, unless you pay for a private repository.
- Currently, there is no support for HTTPS for custom domains. It's probably coming soon though.
- Although Jekyll is supported, plug-in support is rather spotty.

Firebase

The Realtime App Platform. Firebase is a cloud service designed to power real-time, collaborative applications. Simply add the Firebase library to your application to gain access to a shared data structure; any changes you make to that data are automatically synchronized with the Firebase cloud and with other clients within milliseconds.

Some of the features offered by Firebase are:

- Add the Firebase library to your app and get access to a shared data structure. Any changes made to that data are automatically synchronized with the Firebase cloud and with other clients within milliseconds.
- Firebase apps can be written entirely with client-side code, update in real-time out-of-the-box, interoperate well with existing services, scale automatically, and provide strong data security.
- Data Accessibility- Data is stored as JSON in Firebase. Every piece of data has its own URL which can be used in Firebase's client libraries and as a REST endpoint. These URLs can also be entered into a browser to view the data and watch it update in real-time.

Reasons for favoring over GitHub Pages:

- Realtime backend made easy
- Fast and responsive

Companies Using Firebase:

Instacart, 9GAG, and Twitch are some of the popular companies that use Firebase
Firebase has a broader approval, being mentioned in 1215 company stacks & 4651 developers stacks

Pros

- Hosted by Google. Enough said.
- Authentication, Cloud Messaging, and a whole lot of other handy services will be available to you.
- A real-time database will be available to you, which can store 1 GB of data.
- You'll also have access to a blob store, which can store another 1 GB of data.
- Support for HTTPS. A free certificate will be provisioned for your custom domain within 24 hours.

Cons

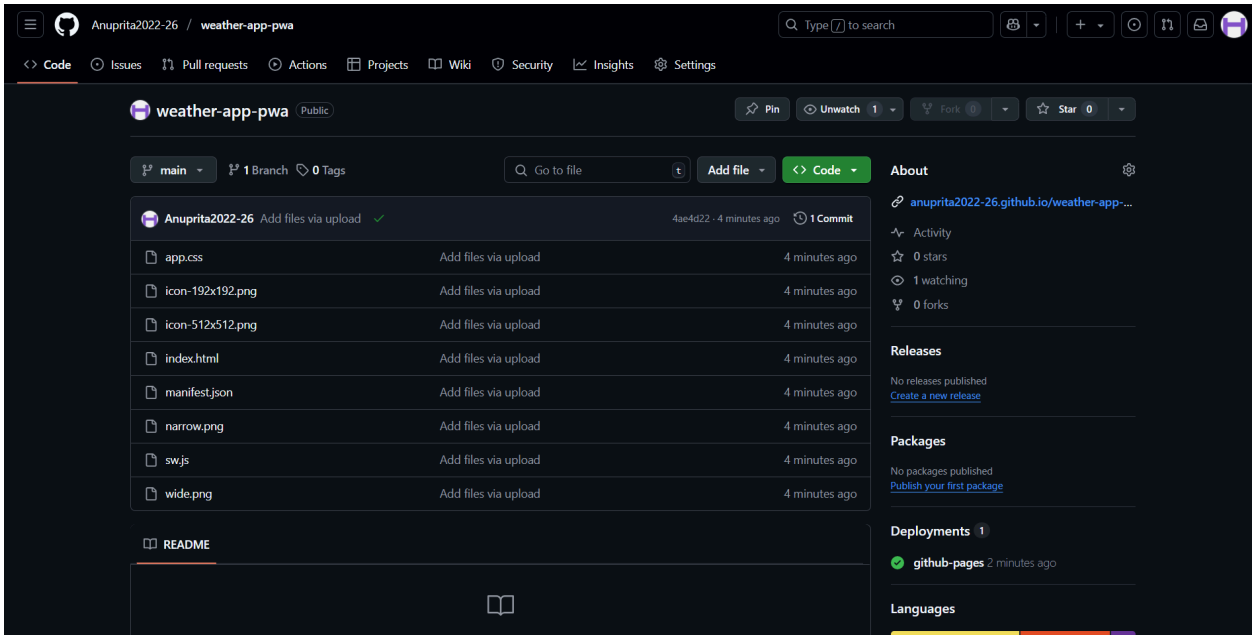
- Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
- Command-line interface only.
- No in-built support for any static site generator.

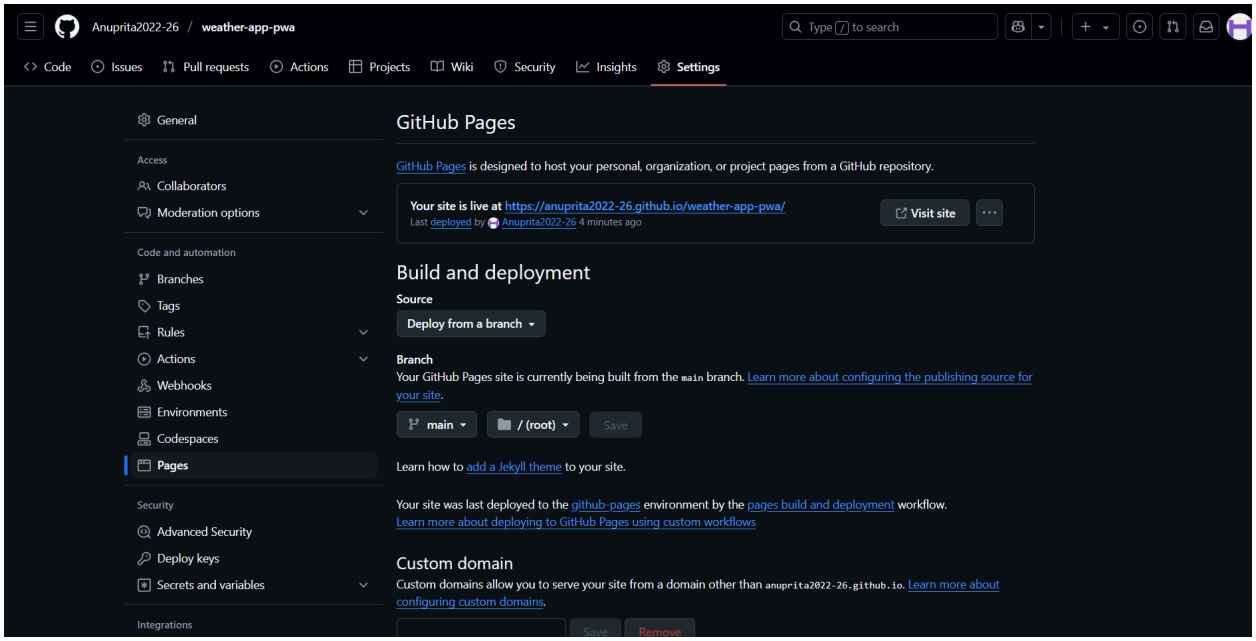
Link to our GitHub repository:

https://github.com/Anuprita2022-26/weather-app-pwa

Link to our Hosted website:
https://anuprita2022-26.github.io/weather-app-pwa/

Github Screenshot:





MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	28
Name	Anuprita Mhapankar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

MPL Experiment 11 (PWA)

Name: Anuprita Mhapankar **Class:** D15A **Roll no:** 28

Aim: To use google Lighthouse PWA Analysis Tool to test the PWA functioning.

Theory:

Google Lighthouse :

Google Lighthouse is a tool that lets you audit your web application based on a number of parameters including (but not limited to) performance, based on a number of metrics, mobile compatibility, Progressive Web App (PWA) implementations, etc. All you have to do is run it on a page or pass it a URL, sit back for a couple of minutes and get a very elaborate report, not much short of one that a professional auditor would have compiled in about a week.

The best part is that you have to set up almost nothing to get started. Let's begin by looking at some of the top features and audit criteria used by Lighthouse.

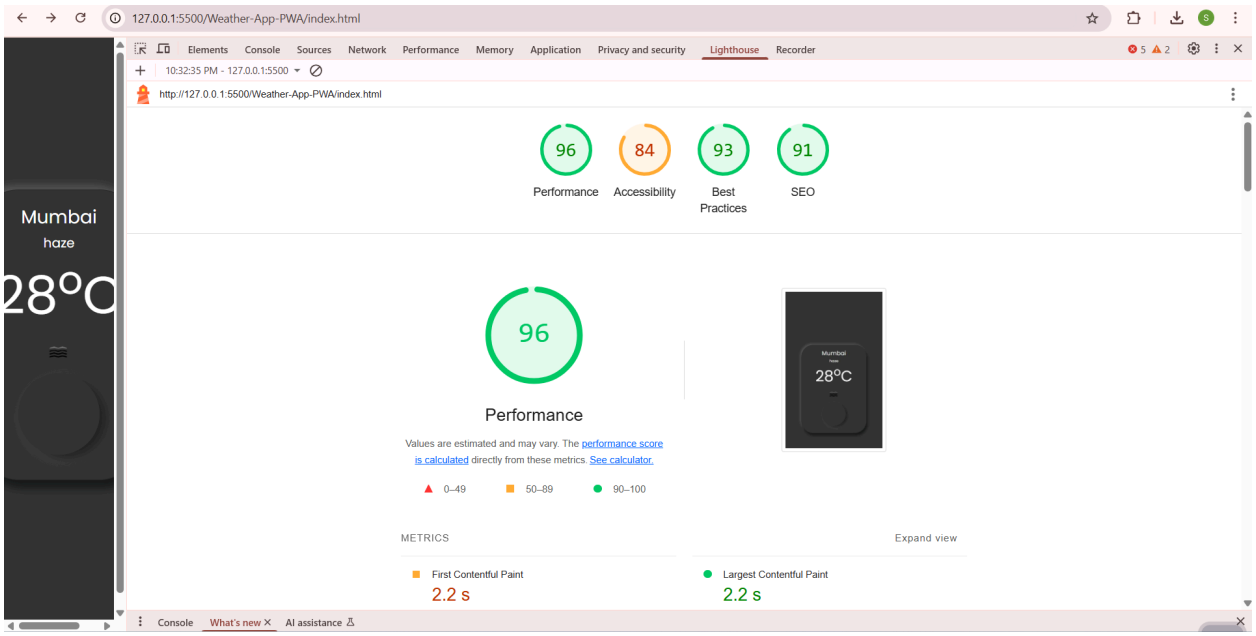
Key Features and Audit Metrics:

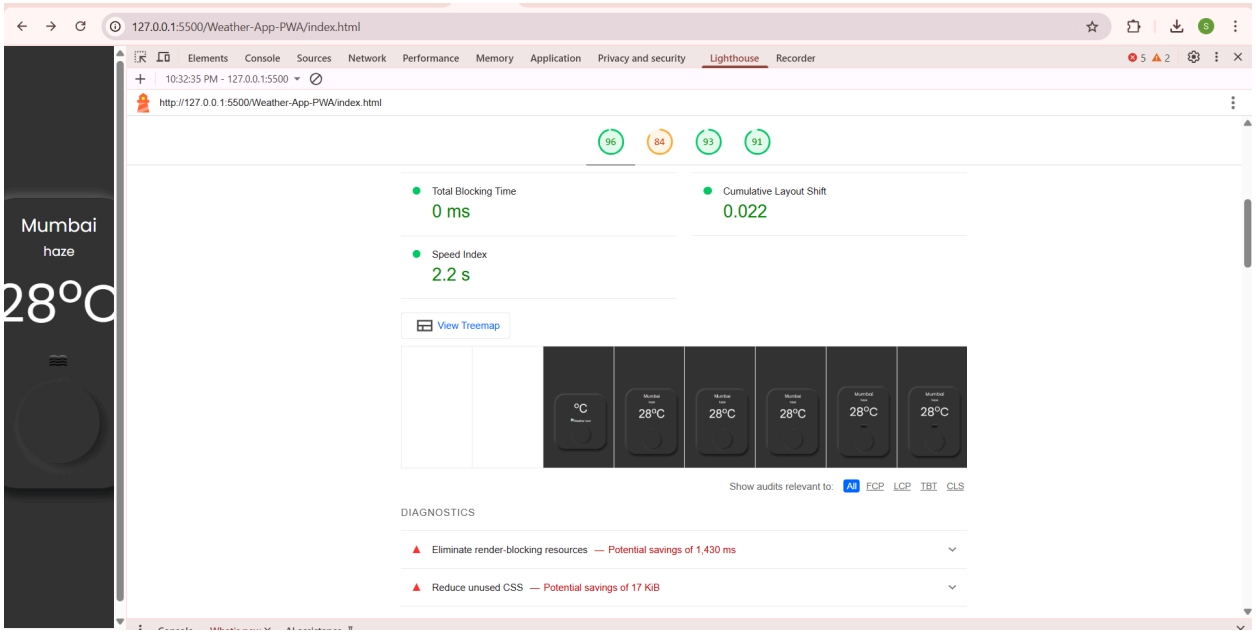
Google Lighthouse has the option of running the Audit for Desktop as well as mobile version of your page(s). The top metrics that will be measured in the Audit are:

- **Performance:**
  Measures loading speed, content display time, and overall site efficiency (score out of 100).
- **PWA Score (Mobile):**
  Evaluates adherence to Google's Baseline PWA checklist, including Service Worker implementation, offline functionality, and script-disabled performance.
- **Accessibility:**
  Assesses website accessibility features like `aria-` attributes, screen-reader compatibility, and semantic HTML tags. Scores are pass/fail.
- **Best Practices:**
  Checks adherence to industry standards, including HTTPS usage and security measures.

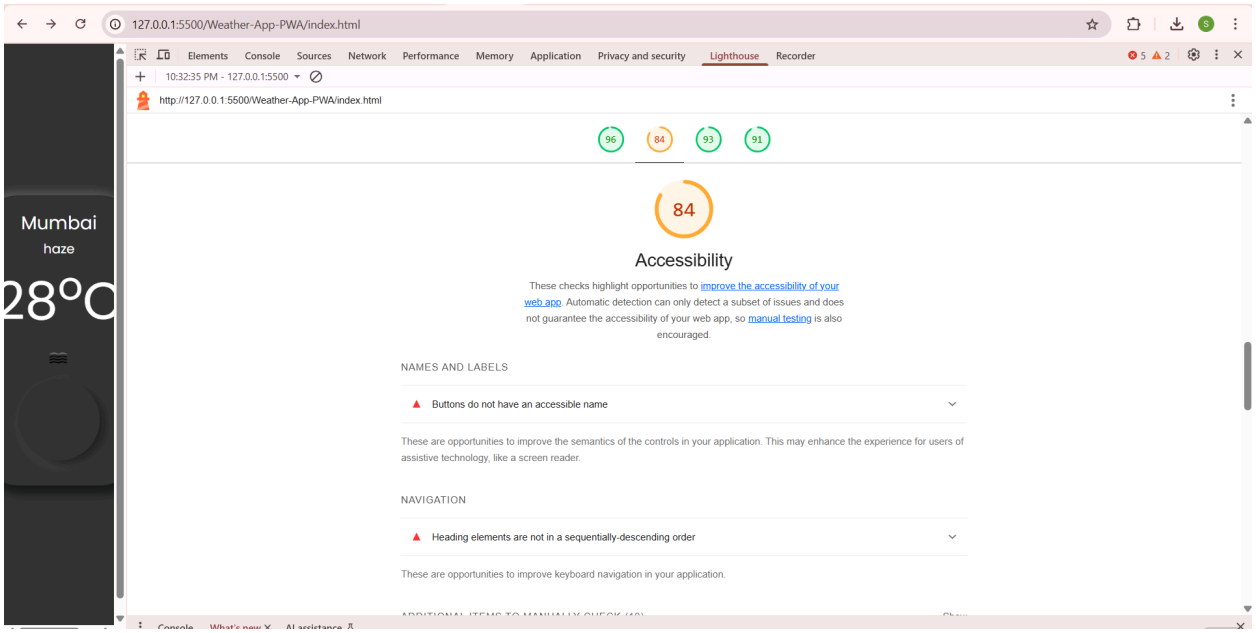
Output:

a) Performance

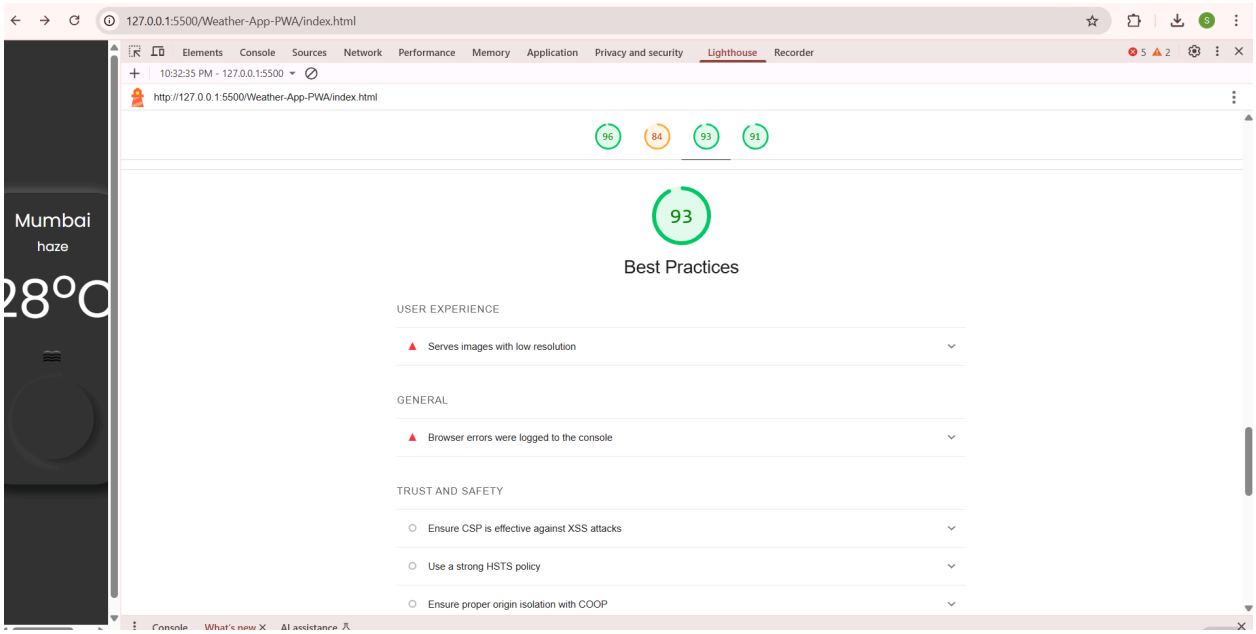




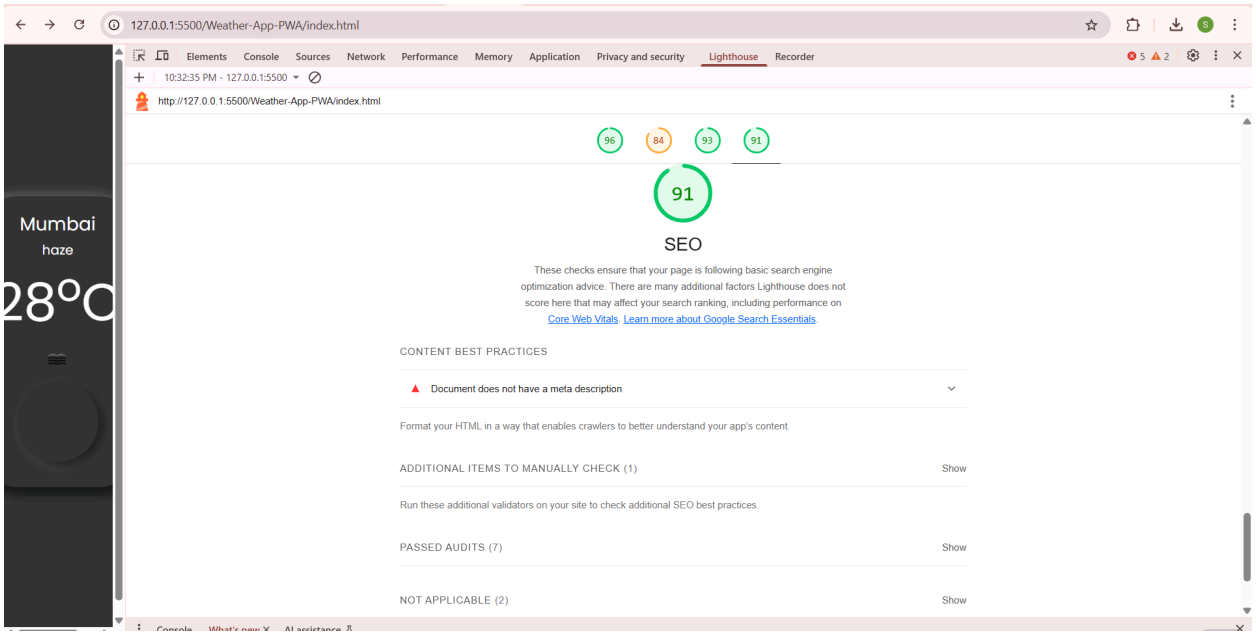
b) Accessibility



c) Best Practices



d) Search Engine Optimization (SEO)



Conclusion:

Thus we successfully used google Lighthouse PWA Analysis Tool for testing the PWA functioning.

MAD & PWA Lab
Journal

Experiment No.	Assignment-1
Assignment 1 Questions	1. Flutter Overview:Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community. 2. Widget Tree and Composition: Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree. 3. State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable. 4. Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.
Roll No.	28
Name	Anuprita Mhapankar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	4

MPL Assignment 1

Q1] Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community.

→ **Solution:**

Key features of Flutter:

a) Single Codebase:
Write once, run on both Android and iOS.

b) Hot Reload
Instantly see changes in code without restarting the app.

c) Widget - Based UI
Everything in Flutter is a widget, making UI development fast and flexible.

d) High Performance
Uses the Dart language and its own rendering engine (Skia) for smooth animation.

e) Cross Platform Support
Support mobile, web and desktop application.

How Flutter differs from Traditional Approaches

a) Traditional Approach:

Native development requires separate codebase for Android (Java / Kotlin) and ios (swift / Objective-c) leading to more effort and maintenance.

b) Flutter

It uses a single part codebase to create apps for multiple platforms, reducing development time and cost.

Why Flutter is popular among developers

a) Saves time and effort with cross-platform development.

b) Hot reload boost productivity by allowing quick iterations.

c) Strong community support and growing adoption in industries.

d) Modern UI capabilities make app development more efficient.

q2] Widget Tree and Composition: Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree.

→ Solution:

Widget Tree in Flutter:

The widget tree represents the structure of UI elements in an application. Every UI element, form simple text to complex layouts, is a widget. These widgets are arranged hierarchically, forming a tree-like structure where each widget is a node.

There are two types of widgets:

a) Stateless Widget
   Does not change once built.
   Example: Text, Icon.

b) Stateful Widget
   Can change dynamically
   Example: Textfield, Checkbox.

Widget Composition

Flutter follows a composition-based approach, meaning complex UIs are built by combining smaller widgets. Instead of modifying a single large widget, developers create multiple reusable widgets and nest them inside each other.

For example: To create a simple UI with a Text inside a Container.

```
class MyWidget extends StatelessWidget {
  @override
  Widget build (BuildContext context) {
    return Container (
      padding: EdgeInsets.all (16),
      color: Colors.blue,
      child: Text (
        'Hello, Flutter!'
        style: TextStyle (fontSize: 20),
      ),
    );
  }.
}
```

Here, Container is a parent widget (provides padding and background color).
Text is a child widget (displays content)

Commonly used Widgets in a Widget Tree.

a) Structural Widgets - Defines layout.

(i) Container
Hold and styles child Widgets.

(ii) Column / Row
Arranges widgets vertically / horizontally.

(iii) Stack
Overlap widgets.

b) Interactive Widgets - Handle use input.
(i) Elevated Button - Clickable button.
(ii) TextField - Input field.
(iii) GestureDetector - Detects touch events
c) Styling and Display Widgets
Modify appearance.
(i) Padding - Adds space around widgets.
(ii) SizedBox - Defines fixed width / height.
(iii) DecoratedBox - Applies backgrounds, borders.

Example: Building a simple UI

```
class MyApp extends StatelessWidget {
  @override
  Widget build (BuildContext context) {
    return Scaffold (
      appBar: AppBar (title: Text ("Flutter
        App")),
      body: Center (
        child: Column (
          mainAxisAlignment: MainAxisAlignment.
                                center,
          children: [
            Text ("Welcome to Flutter!"),
            ElevatedButton (
              onPressed: () {},
              child: Text ("Click Me"),
            ),
          ],
        ),
      ),
    );
  }
}
```

This widget tree consists of:

(i) Scaffold : main structure.

(ii) AppBar : Title bar.

(iii) Column : Arranges text and button.

(iv) Text & ElevatedButton : Child widgets.

Q3] State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different statement management approaches available in Flutter, such as setState, Provider and Riverpod. Provide scenarios where each approach is suitable.

→ Solution:

Importance of State Management in Flutter.

State management is crucial in Flutter applications because it helps manage UI updates efficiently. Without proper state management, applications can become unresponsive, inefficient and difficult to maintain. It ensures that UI components update correctly when data changes, leading to better performance and user experience.

Comparision of State Management Approaches

Approach	set state	Provider	Riverpod
• Simplicity	Very simple	moderate	more structured
• performance	Rebuilds the entire widget	Efficient with change Notifier	Optimized dependency injection

	set state	provider	Riverpod
Scalability	Not scalable for large apps	suitable for medium to large apps	Highly scalable and testable
Use case	small apps, local UI state changes	Apps needing dependency injection and shared state	complex apps needing better modularity and testability

When to use Each Approach

a) set State

Best for small Apps and local UI updates

- When managing UI-related state inside a single widget.
- Example: Updating a counter in a basic app (setState(() {})).

b) Provider

Best for Medium scale Apps with shared state)

- When multiple widgets need access to shared state.
- Example: A shopping app where multiple pages need access to user login details.

c) Riverpod

Best for large scale and complex Applications.

- When you need better performance, modularity, and testability
- Example: A large scale e-commerce app with multiple providers managing different states.

Q4] Firebase Integration in Flutter: Explain the process of integrating Firebase with flutter application. Disuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in flutter development and provide brief overview of how data synchronization is achieved.

→ Solution:

Firebase Integration in Flutter

a) Create a Firebase Project: Go to Fiebase console, create a new project and register your app.

b) Add Firebase SDK: Download google-services.json and place it in the appropriate directories.

c) Install dependencies: Add Firebase packages in pubspec.yaml.

```
dependencies:
    firebase_core: latest_version
    firebase_auth: latest_version
    cloud_firestore: latest_version
```

d) Initialize Firebase: In main.dart, initialize Firebase

```
void main() async {
    WidgetsFlutterBinding.ensureInitialized();
    await Firebase.initializeApp();
    runApp(MyApp());
}
```

e) Use firebase services: Implement authentication, Firestore database or any other required service in your app.

Benefits of using Firebase as a Backend Solution:

a) No server Management: Fully managed backend without the need to maintain servers.

b) Scalability - Handles large user bases and real time data updates efficiently.

c) Authentication - Provides ready to use authentication with Google, email / password, etc.

d) Real time Database and Firestore - Enables seamless realtime data sync across devices.

Common Firebase services used in Flutter

a) Firebase Authentication
Secure login / signup (Google, Email, etc)

b) Cloud Firestore
Real time NOSQL database for structured data storage

c) Firebase storage
Store images, files and media securely

d) Cloud Messaging
Push Notifications

e) Crashlytics
Monitor and fix crashes in real time

Data Synchronization in Firebase

Firebase Firestore and Real time Database use real-time listners, ensuring data is automatically updated across all devices.

Example:

```
StreamBuilder(
  stream: FirebaseFirestore.instance.collection('users').
      snapshots(),
  builder: (context, snapshot) {
    if (!snapshot.hasData) return CircularProgressIndicator();

    var data = snapshot.data!.docs;
    return ListView.builder(
      itemCount: data.length,
      itemBuilder: (context, index) {
        return ListTile(title:
            Text(data[index]['name'])
        );
      };
    )
  }
);
```

This ensures that changes in Firestore reflect instantly in the app without manual refresh.

MAD & PWA Lab
Journal

Experiment No.	Assignment-2
Assignment 2 Questions	1. Define Progressive Web App (PWA) and explain its significance in modern web development.Discuss the key characteristics that differentiate PWAs from traditional mobile apps 2. Define responsive web design and explain its importance in the context of Progressive Web Apps.Compare and contrast responsive, fluid, and adaptive web design approaches. 3. Describe the lifecycle of Service Workers, including registration, installation, and activation phases. 4. Explain the use of IndexedDB in the Service Worker for data storage.
Roll No.	28
Name	Anuprita Mhapankar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4:Understand various PWA frameworks and their requirements LO5: Design and Develop a responsive User Interface by applying PWA Design techniques LO6:Develop and Analyze PWA Features and deploy it over app hosting solutions
Grade:	5

MPL Assignment 2

Q[] Define Progressive Web App (PWA) and explain its significance in modern web development. Discuss the key charachteristics that differentiate PWAs from traditional mobile apps.

→ Solution:

A progressive Web App (PWA) is a web application that leverages modern web technologies to deliver an app-like experience to users. It combines the best features of web and mobile apps, offering fast performance, offline capabilities, and cross platform support without requiring installing from an app store.

SIGNIFICANCE IN MODERN WEB DEVELOPMENT:

a) Improved User Experience:
PWAs provide a smooth, app-like interface with responsive design.

b) Offline Functionality:
Service workers enable caching, allowing PWAs to work without an internet connection.

c) Cross Platform Compatibility:
A single codebase runs on different devices (desktop, mobile, tablet)

d) Cost - Effective Development
Eliminates the need to develop seperate apps for Android and iOS.

e) No App Store Dependency:
Users can install PWAs directly from Browser, reducing installation barriers.

Key characteristics Differentiating PWA from Traditional Mobile Apps.

Feature	PWA	Traditional Mobile App
• Installation	Added to the home screen via browsen	Installed via app store.
• Performance	Fast and lightweight	May consume more storage and memory
• Offline Access	Works offline with cached data	Fully functional offline
• Updates	Auto-updated from the web	Requires manual updates via store.
• Platform	Runs on browsers	Platform specific.
• Development cost	Lower	Higher

Q2] Define responsive web design and explain its importance in the context of Progressive Web Apps. Compare and contrast responsive, fluid and adaptive web design approaches.

→ Solution:

Responsive Web Design (RWD) is a web development approach that ensures websites automatically adjust their layout, content and elements based on the user's device screen size, resolution, and orientation. It uses flexible grids, media queries and scalable images to create a seamless experience across desktops, tablets and smartphones

Importance of RWD in PWA:

a) Enhanced User Experience
PWA aims to work smoothly across devices, and RWD ensures consistent UI/UX.

b) Improved Accessibility
Users can access PWA on any screen size without usability issues.

c) Faster Load Time
Optimized layout and images improve performance, which is critical for PWAs.

d) SEO Benefits
Google prioritizes mobile friendly sites, boosting search rankings.

e) Cost Effective
A single responsive design eliminates the need for seperate mobile and desktop versions.

Comparison of Responsive, Fluid and Adaptive Web Design.

Feature	Responsive	Fluid	Adaptive
Approach	Uses CSS media queries to adjust layout based on screen size.	Uses percentage-based layout for flexible scaling	Uses predefined layout for specific screen size.
Flexibility	Highly flexible	Fully Flexible	Less Flexible.
Performance	Optimized for all devices	can be inconsistent on very large screen	May load unnecessary elements for non-matching screen size
Ease of Implementation	Moderate	Simple	Complex
Best For	Modern website	Simple website	Apps/ website

Q3] Describe the lifecycle of service workers, including registration, installation, and activation phases.

→ Solution:

Lifecycle of Service Workers:

It has three main phases:

(a) Registration

- The service worker is registered in the JavaScript code using navigator.serviceWorker.register().
- This tells the browser where to find the service worker script and initiates its lifecycle.

```
if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('/service-worker.js')
  .then(reg => console.log('Service Worker Registered', reg))
  .catch(err => console.log('Service Worker Registration Failed', err));
}
```

(b) Installation

- After registration, the browser tries to install the service worker.
- The install event is triggered where resources can be preloaded.
- If successful, the service worker moves to the activation phase; otherwise, it retries installation.

```
self.addEventListner('install', event => {
  event.waitUntil(
    caches.open('v1').then(cache => {
      return cache.addAll(['/index.html', '/styles.css',
        '/script.js']);
    })
  );
});
```

(c) Activation

- Once installed, the activate event is triggered.
- The old caches of outdated service workers are cleaned up in this phase.
- The service worker takes control of clients and starts handling fetch events

```
self.addEventListner('activate', event => {
  event.waitUntil(
    caches.keys().then(keys => {
      return Promise.all(keys.map(key => {
        if (key !== 'v1') {
          return caches.delete(key);
        }
      }));
    })
  );
});
```

Q4] Explain the use of IndexedDB in the service worker for data storage.

→ Solution:

IndexedDB is a client-side, NOSQL database that allows web applications to store structured data in the browser. When combined with a service worker, it helps enable offline functionality and efficient caching of data.

Use of IndexedDB in a Service Worker

(a) Persistent Data Storage

Unlike cache API, which mainly stores request / response objects, IndexedDB allows storing structured data like JSON objects, user preferences and application states.

(b) Offline Support
Enables users to access and modify data even when offline. The data can later sync with the server when online.

(c) Efficient caching
Stores large amounts of data that don't need to be re-fetched from the network, reducing load times.

(d) Background sync
Works with Background sync API to sync data once the internet is available.