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 **MOHIT PATIL** Of **D15A/D15B** 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

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.

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			
2.	To design Flutter UI by including common widgets.	LO2			
3.	To include icons, images, fonts in Flutter app	LO2			
4.	To create an interactive Form using form widget	LO2			
5.	To apply navigation, routing and gestures in Flutter App	LO2			
6.	To Connect Flutter UI with fireBase database	LO3			
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			
13.	Assignment-2	LO4,LO5 ,LO6			

MAD & PWA Lab
Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	36
Name	MOHIT PATIL
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

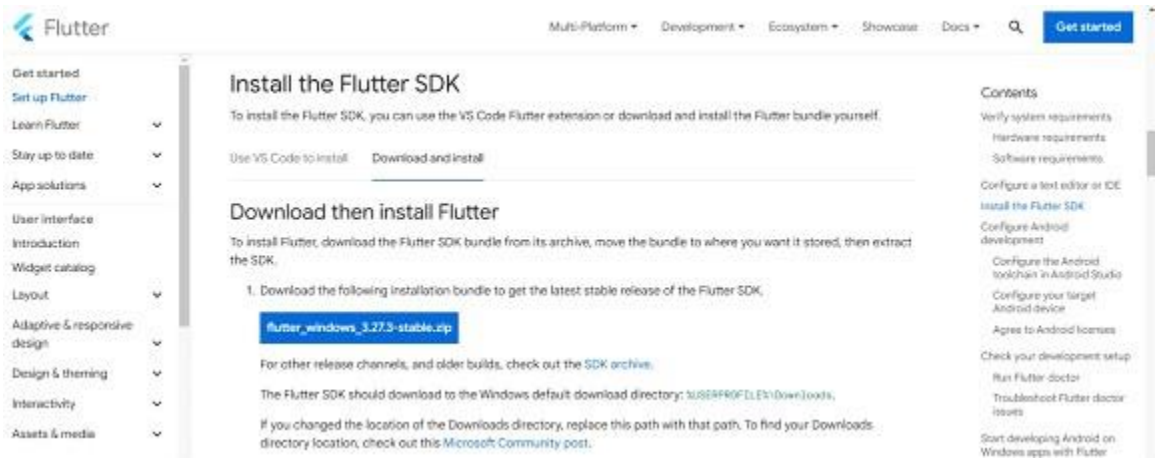
MPL Experiment 1

Name: Mohit Patil **Class:** D15A **Roll no:**36

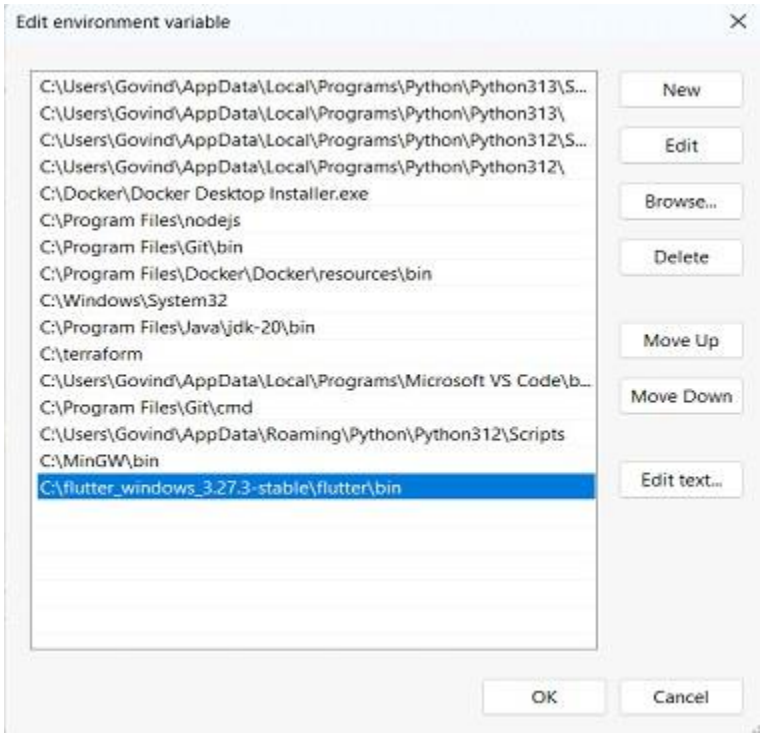
Aim: Installation and Configuration of Flutter Environment.

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).



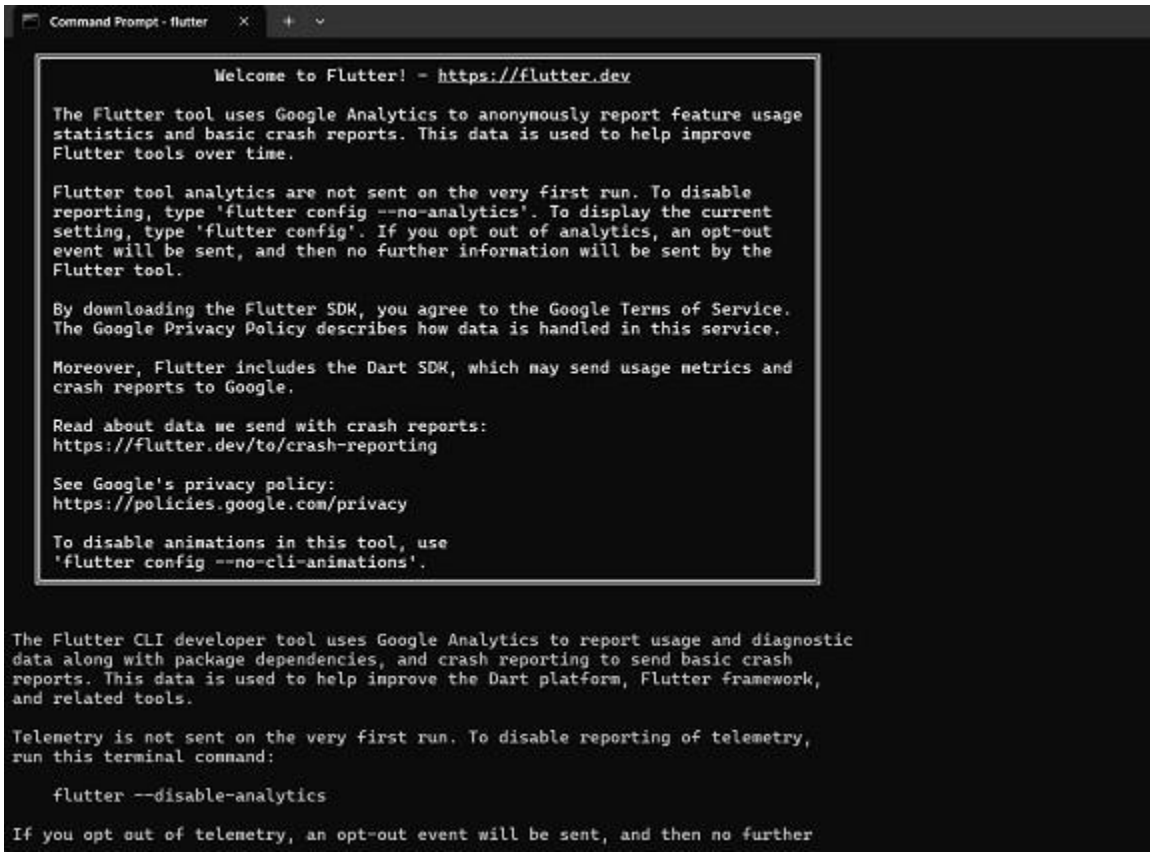
- Extract the downloaded zip file to a preferred location on your computer (e.g., C:\src\flutter for Windows).
- Add Flutter to the PATH
- Locate the flutter\bin directory in the extracted Flutter folder.
- Add this directory to your system's PATH environment variable.



Verify the Installation

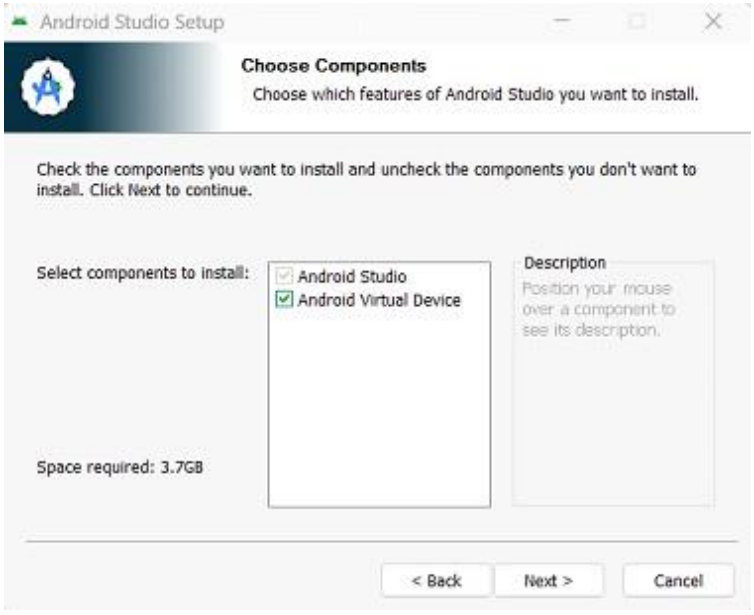
- Open a terminal or command prompt.
- Run the command: **flutter** and **flutter doctor**.

```
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)
[✗] 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>
```

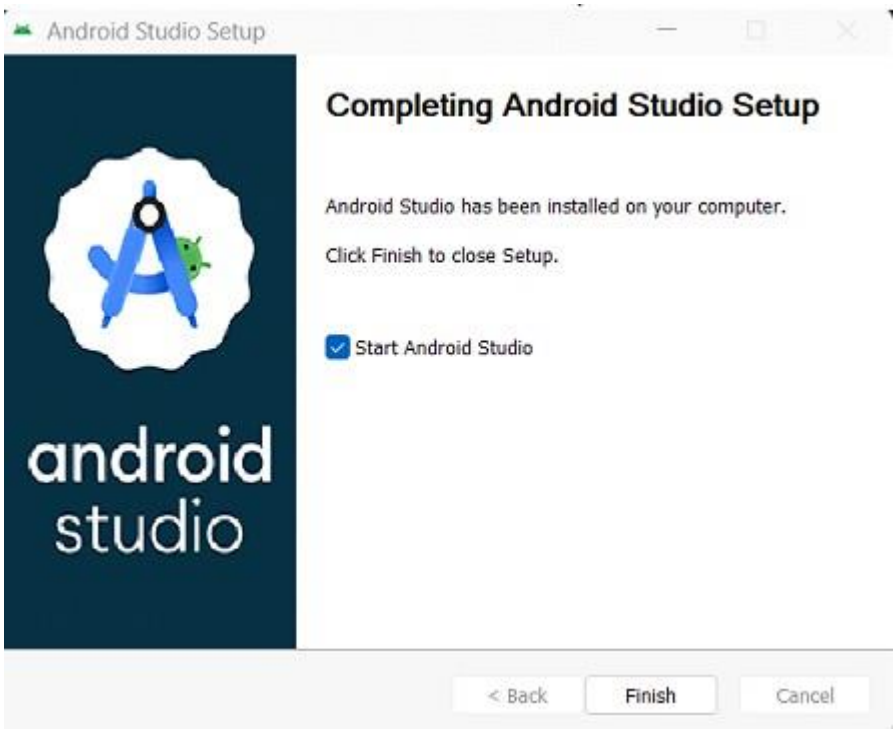


Step 2: Install Android Studio

- 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)).



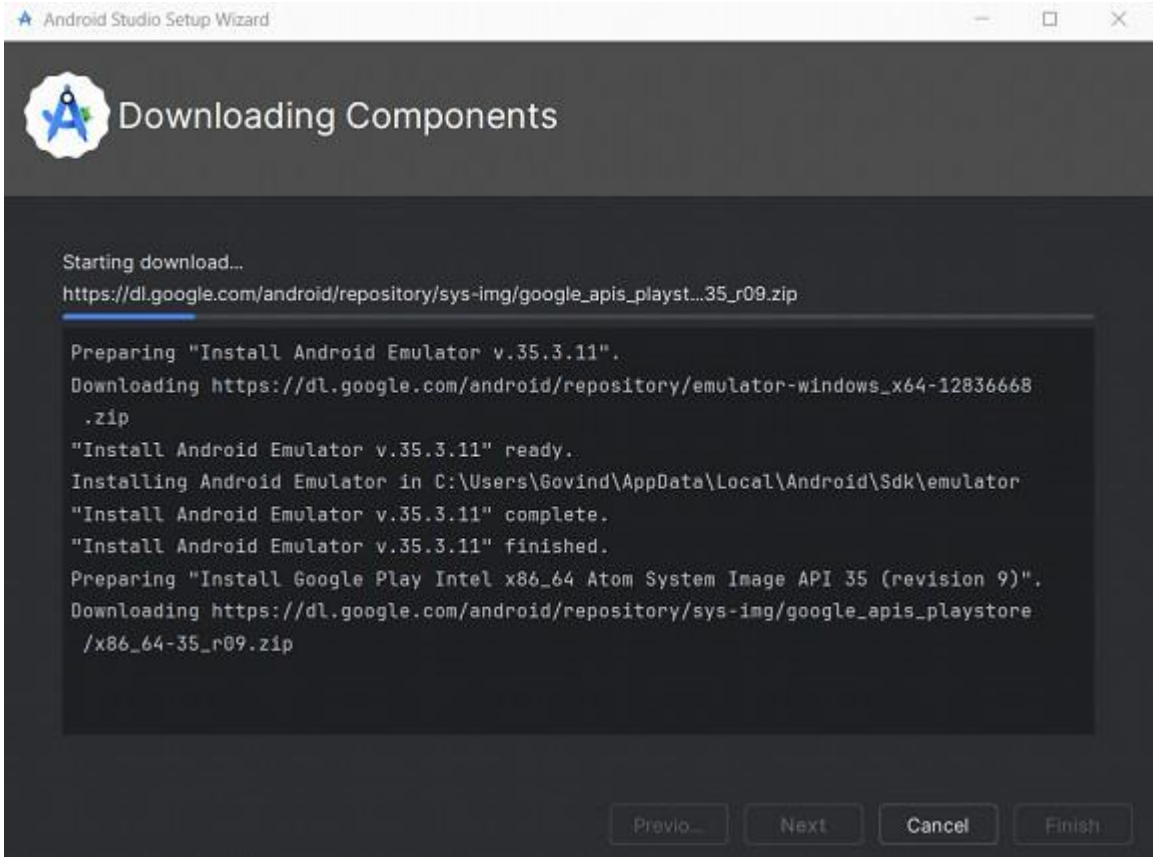
- Run the installer and follow the setup wizard.
- Choose the standard installation option.

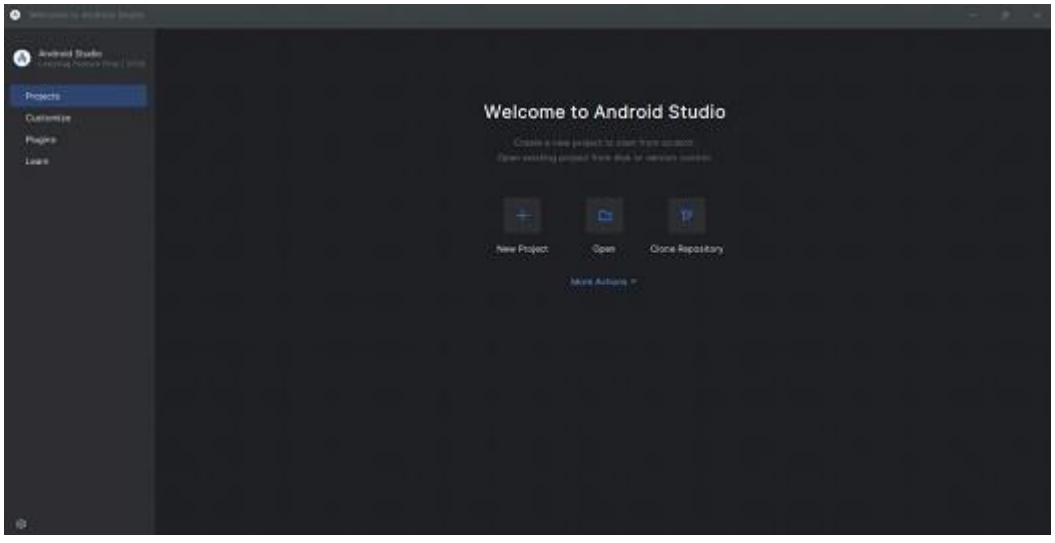


- Install Android SDK Tools
- Open Android Studio.

- 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.

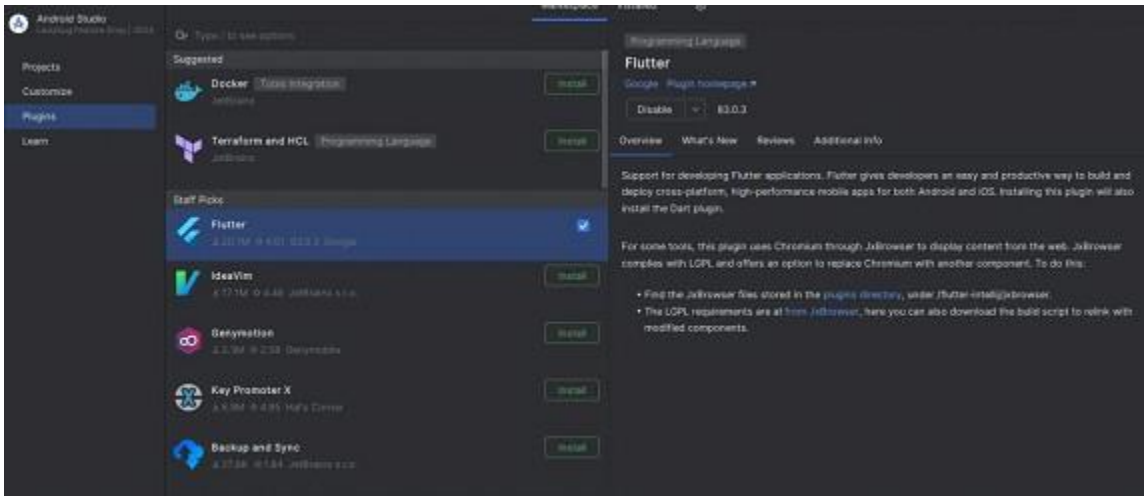




Step 3: Connect Flutter with Android Studio

- 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:

```
proj1 > lib > main.dart > MyApp > build
import 'package:flutter/material.dart';

Run | Debug | Profile | Codeium: Refactor | Explain | Generate Function Comment | ×
void main() {
  runApp(app: const MyApp());
}

Codeium: Refactor | Explain
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Codeium: Refactor | Explain | Generate Function Comment | ×
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text(data: 'welcome'),
        ), // AppBar
        body: const Center(
          child: Text(
            data: 'Hello im Mohit Patil lets create the app!',
            style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
          ), // Text
        ), // Center
      ), // Scaffold
    ); // MaterialApp
  }
}
```

Conclusion: Hello message , 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.	36
Name	MOHIT PATIL
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

MPL Experiment 2

Name: Mohit Patil **Class:** D15A **Roll no:**36

Aim: To design Flutter UI by including common widgets.

Code:

```
//Home screen.dart
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:http/http.dart' as http;
import 'package:url_launcher/url_launcher.dart';
import 'login.dart';
import 'UsedCarScreen.dart';
import 'package:carousel_slider/carousel_slider.dart';
import 'carcompare.dart'; // Import CarCompare Page
import 'BudgetPage.dart';

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

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

class _HomePageState extends State<HomePage> {
  Future<List<Map<String, String>>> fetchCarNews() async {
    final url =
        'https://newsapi.org/v2/everything?q="car industry" OR
"automobile" OR "new car
launch"&language=en&sortBy=publishedAt&apiKey=4c05b7fc955e4736bd656882
83299b74';

    final response = await http.get(Uri.parse(url));

    if (response.statusCode == 200) {
      final data = jsonDecode(response.body);
      List articles = data['articles'];

      return articles.map((article) {
        return {
          'title': (article['title'] ?? '').toString(),
          'image': (article['urlToImage'] ?? '').toString(),
          'url': (article['url'] ?? '').toString(),
        };
      }).toList();
    } else {
      throw Exception('Failed to load news');
    }
  }

  @override
```

```
  Widget build(BuildContext context) {
    final User? user = FirebaseAuth.instance.currentUser;

    if (user == null) {
      Future.delayed(Duration.zero, () {
        Navigator.pushReplacement(
          context,
          MaterialPageRoute(builder: (context) => const LoginPage()),
        );
      });
      return const Scaffold(
        backgroundColor: Colors.black,
        body: Center(child: CircularProgressIndicator(color:
Colors.white)),
      );
    }

    final List<String> featuredCarImages = [
      'assets/harrier.png',
      'assets/white-offroader-jeep-parking.png',
      'assets/harrier.png',
      'assets/mercec.png',
      'assets/creata.png'
    ];

    return Scaffold(
      backgroundColor: Colors.black,
      appBar: AppBar(
        backgroundColor: Colors.black,
        elevation: 0,
        title: Row(
          children: [
            Image.asset('assets/car.png', height: 40),
            const SizedBox(width: 10),
            const Text(
              "CarConnect",
              style: TextStyle(
                color: Colors.white,
                fontSize: 22,
                fontWeight: FontWeight.bold,
                fontFamily: 'Roboto',
                letterSpacing: 1.2,
              ),
            ),
          ],
        ),
        actions: [
          IconButton(
            icon: const Icon(Icons.logout, color: Colors.white),
            onPressed: () async {
              await FirebaseAuth.instance.signOut();
              Navigator.pushReplacement(
                context,
                MaterialPageRoute(builder: (context) => const
LoginPage()),
```

```
              );
            },
          ),
        ],
      ),
      body: SingleChildScrollView(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Padding(
              padding: const EdgeInsets.all(16.0),
              child: TextField(
                style: const TextStyle(color: Colors.white),
                decoration: InputDecoration(
                  hintText: "Search for your car e.g. Creta",
                  hintStyle: TextStyle(color: Colors.white60),
                  filled: true,
                  fillColor: Colors.grey[900],
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(10),
                    borderSide: BorderSide.none,
                  ),
                  prefixIcon: Icon(Icons.search, color:
Colors.white60),
                ),
              ),
            ),
            Padding(
              padding: const EdgeInsets.symmetric(horizontal: 16.0),
              child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  _filterButton("Compare", Icons.scale, onTap: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                          builder: (context) => CarComparisonPage()),
                    );
                  }),
                  _filterButton("Used", Icons.directions_car, onTap:
() {
                    Navigator.push(
                      context,
                      MaterialPageRoute(builder: (context) =>
UsedCarScreen()),
                    );
                  }),
                  _filterButton("Budget", Icons.attach_money, onTap:
() {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) => BudgetPage(
                          carName:
                              "Your Selected Car", // Pass selected
```

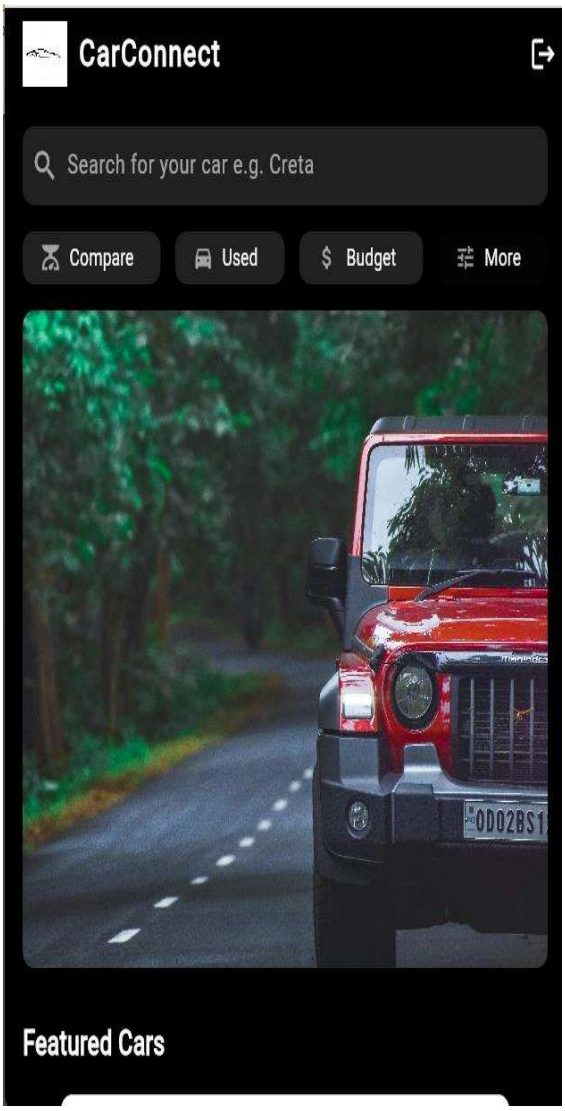
```
car Name
                          exShowroomPrice:
                              1000000, // Pass selected car price
dynamically
                        ),
                      ),
                    );
                  }),
                  _filterButton("More", Icons.tune),
                ],
              ),
            ),
            Padding(
              padding: const EdgeInsets.all(16.0),
              child: ClipRRect(
                borderRadius: BorderRadius.circular(10),
                child: SizedBox(
                  width: double.infinity,
                  height: 400,
                  child: Image.asset(
                    'assets/pexels-samyantak-mohanty-79378681-
8706096.png',
                    fit: BoxFit.cover,
                  ),
                ),
              ),
            ),
            const Padding(
              padding: EdgeInsets.all(16.0),
              child: Text(
                'Featured Cars',
                style: TextStyle(
                    fontSize: 20,
                    fontWeight: FontWeight.bold,
                    color: Colors.white),
              ),
            ),
            CarouselSlider(
              options: CarouselOptions(autoPlay: true,
enlargeCenterPage: true),
              items: featuredCarImages.map((imagePath) {
                return ClipRRect(
                  borderRadius: BorderRadius.circular(10),
                  child: Image.asset(imagePath, fit: BoxFit.cover),
                );
              }).toList(),
            ),
            const Padding(
              padding: EdgeInsets.all(16.0),
              child: Text(
                'Latest Car News',
                style: TextStyle(
                    fontSize: 20,
                    fontWeight: FontWeight.bold,
                    color: Colors.white),
```

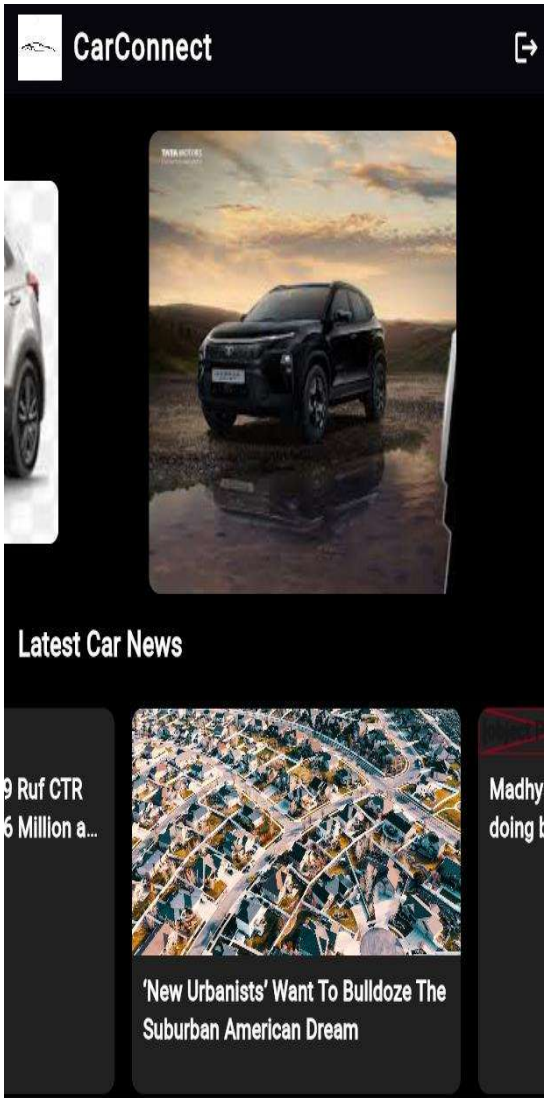
```
            ),
          ),
          FutureBuilder<List<Map<String, String>>>(
            future: fetchCarNews(),
            builder: (context, snapshot) {
              if (snapshot.connectionState ==
ConnectionState.waiting) {
                return const Center(
                    child: CircularProgressIndicator(color:
Colors.white));
              } else if (snapshot.hasError) {
                return Center(
                  child: Text(
                    "Failed to load news",
                    style: TextStyle(color: Colors.white),
                  ),
                );
              } else if (!snapshot.hasData ||
snapshot.data!.isEmpty) {
                return Center(
                  child: Text(
                    "No news available",
                    style: TextStyle(color: Colors.white),
                  ),
                );
              }

              return SizedBox(
                height: 250,
                child: ListView.builder(
                  scrollDirection: Axis.horizontal,
                  itemCount: snapshot.data!.length,
                  itemBuilder: (context, index) {
                    final news = snapshot.data![index];
                    return GestureDetector(
                      onTap: () async {
                        Uri url = Uri.parse(news['url']!);
                        if (await canLaunchUrl(url)) {
                          await launchUrl(url);
                        }
                      },
                      child: Card(
                        color: Colors.grey[900],
                        margin: const EdgeInsets.symmetric(
                            horizontal: 8, vertical: 8),
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(10),
                        ),
                        child: SizedBox(
                          width: 300,
                          child: Column(
                            crossAxisAlignment:
CrossAxisAlignment.start,
                            children: [
                              if (news['image']!.isNotEmpty)
```

```
                                    ClipRRect(
                                      borderRadius: const
BorderRadius.vertical(
                                          top: Radius.circular(10)),
                                      child:
Image.network(news['image']!,
                                          fit: BoxFit.cover,
                                          height: 150,
                                          width: double.infinity),
                                    ),
                                  Padding(
                                    padding: const EdgeInsets.all(10),
                                    child: Text(
                                      news['title']!,
                                      style: const TextStyle(
                                          color: Colors.white,
                                          fontSize: 16,
                                          fontWeight: FontWeight.bold),
                                      maxLines: 2,
                                      overflow: TextOverflow.ellipsis,
                                    ),
                                  ),
                                ],
                              ),
                            );
                          },
                        ),
                      );
                    },
                  ),
                ],
              ),
            ),
          );
        }

  Widget _filterButton(String title, IconData icon, {VoidCallback?
onTap}) {
    return ElevatedButton.icon(
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.grey[900],
        shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(10)),
      ),
      onPressed: onTap,
      icon: Icon(icon, color: Colors.white60),
      label: Text(title, style: const TextStyle(color: Colors.white)),
    );
```

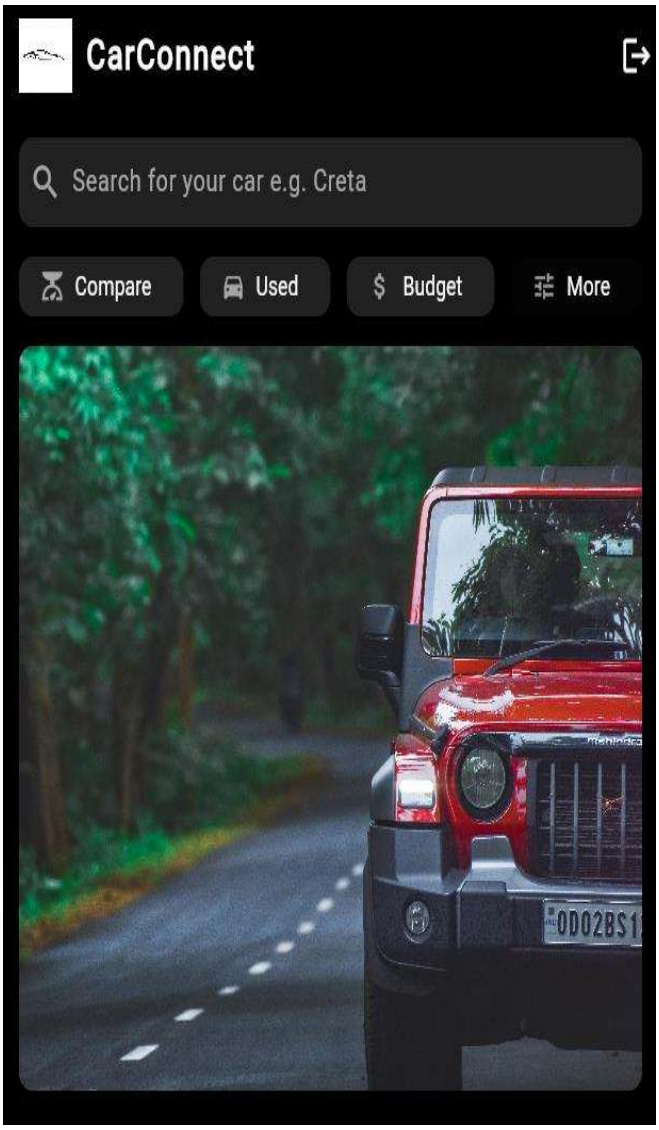
Output:

HomePage:





Widget :



Widget are crated above home screen in the from of buttons that is used to direct to another pages i.e .Compare ,Used ,Budget

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	36
Name	MOHIT PATIL
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

MPL Experiment 3

Name: Mohit Patil **Class:** D15A **Roll no:**36

Aim: To include icons, images, fonts in Flutter app

Theory:

Including icons, images, and custom fonts in a Flutter app enhances visual appeal and user experience. Icons can be added using the Icon widget with built-in Icons class or custom icons via the flutter_launcher_icons package.
Images can be loaded from assets using the Image.asset() method or from the internet using Image.network().
To use local images, they must be placed in the assets folder and declared in pubspec.yaml.
Custom fonts improve typography and branding; they can be added by placing font files in the assets/fonts directory and specifying them in pubspec.yaml under the flutter section.
Using TextStyle with the fontFamily property applies the custom font to text.

Code:
Home :

```dart
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:http/http.dart' as http;
import 'package:url_launcher/url_launcher.dart';
import 'login.dart';
import 'UsedCarScreen.dart';
import 'package:carousel_slider/carousel_slider.dart';
import 'carcompare.dart'; // Import CarCompare Page
import 'BudgetPage.dart';

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

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

class _HomePageState extends State<HomePage> {
  Future<List<Map<String, String>>> fetchCarNews() async {
    final url =
```

```
        'https://newsapi.org/v2/everything?q="car industry" OR
"automobile" OR "new car
launch"&language=en&sortBy=publishedAt&apiKey=4c05b7fc955e4736bd6568828329
9b74';

    final response = await http.get(Uri.parse(url));

    if (response.statusCode == 200) {
      final data = jsonDecode(response.body);
      List articles = data['articles'];

      return articles.map((article) {
        return {
          'title': (article['title'] ?? '').toString(),
          'image': (article['urlToImage'] ?? '').toString(),
          'url': (article['url'] ?? '').toString(),
        };
      }).toList();
    } else {
      throw Exception('Failed to load news');
    }
  }

  @override
  Widget build(BuildContext context) {
    final User? user = FirebaseAuth.instance.currentUser;

    if (user == null) {
      Future.delayed(Duration.zero, () {
        Navigator.pushReplacement(
          context,
          MaterialPageRoute(builder: (context) => const LoginPage()),
        );
      });
      return const Scaffold(
        backgroundColor: Colors.black,
        body: Center(child: CircularProgressIndicator(color:
Colors.white)),
      );
    }
```

```
    final List<String> featuredCarImages = [
      'assets/harrier.png',
      'assets/white-offroader-jeep-parking.png',
      'assets/harrier.png',
      'assets/mercec.png',
      'assets/creata.png'
    ];

    return Scaffold(
      backgroundColor: Colors.black,
      appBar: AppBar(
        backgroundColor: Colors.black,
        elevation: 0,
        title: Row(
          children: [
            Image.asset('assets/car.png', height: 40),
            const SizedBox(width: 10),
            const Text(
              "CarConnect",
              style: TextStyle(
                color: Colors.white,
                fontSize: 22,
                fontWeight: FontWeight.bold,
                fontFamily: 'Roboto',
                letterSpacing: 1.2,
              ),
            ),
          ],
        ),
        actions: [
          IconButton(
            icon: const Icon(Icons.logout, color: Colors.white),
            onPressed: () async {
              await FirebaseAuth.instance.signOut();
              Navigator.pushReplacement(
                context,
                MaterialPageRoute(builder: (context) => const
LoginPage()),
              );
```

```
          },
        ),
      ],
    ),
    body: SingleChildScrollView(
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Padding(
            padding: const EdgeInsets.all(16.0),
            child: TextField(
              style: const TextStyle(color: Colors.white),
              decoration: InputDecoration(
                hintText: "Search for your car e.g. Creta",
                hintStyle: TextStyle(color: Colors.white60),
                filled: true,
                fillColor: Colors.grey[900],
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(10),
                  borderSide: BorderSide.none,
                ),
                prefixIcon: Icon(Icons.search, color: Colors.white60),
              ),
            ),
          ),
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 16.0),
            child: Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                _filterButton("Compare", Icons.scale, onTap: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder: (context) => CarComparisonPage()),
                  );
                }),
                _filterButton("Used", Icons.directions_car, onTap: () {
                  Navigator.push(
                    context,
```

```
                      MaterialPageRoute(builder: (context) =>
UsedCarScreen()),
                    );
                  }),
                  _filterButton("Budget", Icons.attach_money, onTap: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) => BudgetPage(
                          carName:
                              "Your Selected Car", // Pass selected car
Name
                          exShowroomPrice:
                              1000000, // Pass selected car price
dynamically
                        ),
                      ),
                    );
                  }),
                  _filterButton("More", Icons.tune),
                ],
              ),
            ),
            Padding(
              padding: const EdgeInsets.all(16.0),
              child: ClipRRect(
                borderRadius: BorderRadius.circular(10),
                child: SizedBox(
                  width: double.infinity,
                  height: 400,
                  child: Image.asset(

'assets/pexels-samyantak-mohanty-79378681-8706096.png',
                    fit: BoxFit.cover,
                  ),
                ),
              ),
            ),
            const Padding(
              padding: EdgeInsets.all(16.0),
```

```
              child: Text(
                'Featured Cars',
                style: TextStyle(
                    fontSize: 20,
                    fontWeight: FontWeight.bold,
                    color: Colors.white),
              ),
            ),
            CarouselSlider(
              options: CarouselOptions(autoPlay: true, enlargeCenterPage:
true),
              items: featuredCarImages.map((imagePath) {
                return ClipRRect(
                  borderRadius: BorderRadius.circular(10),
                  child: Image.asset(imagePath, fit: BoxFit.cover),
                );
              }).toList(),
            ),
            const Padding(
              padding: EdgeInsets.all(16.0),
              child: Text(
                'Latest Car News',
                style: TextStyle(
                    fontSize: 20,
                    fontWeight: FontWeight.bold,
                    color: Colors.white),
              ),
            ),
            FutureBuilder<List<Map<String, String>>>(
              future: fetchCarNews(),
              builder: (context, snapshot) {
                if (snapshot.connectionState == ConnectionState.waiting) {
                  return const Center(
                      child: CircularProgressIndicator(color:
Colors.white));
                } else if (snapshot.hasError) {
                  return Center(
                    child: Text(
                      "Failed to load news",
                      style: TextStyle(color: Colors.white),
```

```
                    ),
                  );
                } else if (!snapshot.hasData || snapshot.data!.isEmpty) {
                  return Center(
                    child: Text(
                      "No news available",
                      style: TextStyle(color: Colors.white),
                    ),
                  );
                }

                return SizedBox(
                  height: 250,
                  child: ListView.builder(
                    scrollDirection: Axis.horizontal,
                    itemCount: snapshot.data!.length,
                    itemBuilder: (context, index) {
                      final news = snapshot.data![index];
                      return GestureDetector(
                        onTap: () async {
                          Uri url = Uri.parse(news['url']!);
                          if (await canLaunchUrl(url)) {
                            await launchUrl(url);
                          }
                        },
                        child: Card(
                          color: Colors.grey[900],
                          margin: const EdgeInsets.symmetric(
                              horizontal: 8, vertical: 8),
                          shape: RoundedRectangleBorder(
                            borderRadius: BorderRadius.circular(10),
                          ),
                          child: SizedBox(
                            width: 300,
                            child: Column(
                              crossAxisAlignment:
CrossAxisAlignment.start,
                              children: [
                                if (news['image']!.isNotEmpty)
                                  ClipRRect(
```

```
                                    borderRadius: const
BorderRadius.vertical(
                                        top: Radius.circular(10)),
                                    child: Image.network(news['image']!,
                                        fit: BoxFit.cover,
                                        height: 150,
                                        width: double.infinity),
                                  ),
                                Padding(
                                  padding: const EdgeInsets.all(10),
                                  child: Text(
                                    news['title']!,
                                    style: const TextStyle(
                                        color: Colors.white,
                                        fontSize: 16,
                                        fontWeight: FontWeight.bold),
                                    maxLines: 2,
                                    overflow: TextOverflow.ellipsis,
                                  ),
                                ),
                              ],
                            ),
                          ),
                        );
                      },
                    ),
                  );
                },
              ),
            ],
          ),
        ),
      );
  }

  Widget _filterButton(String title, IconData icon, {VoidCallback? onTap})
{
    return ElevatedButton.icon(
      style: ElevatedButton.styleFrom(
```

```
        backgroundColor: Colors.grey[900],
        shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(10)),
      ),
      onPressed: onTap,
      icon: Icon(icon, color: Colors.white60),
      label: Text(title, style: const TextStyle(color: Colors.white)),
    );
  }
}
```

Budget page:

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

class BudgetPage extends StatefulWidget {
  final String? carName;
  final double? exShowroomPrice;
  final String? carType;

  BudgetPage({this.carName, this.exShowroomPrice, this.carType});

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

class _BudgetPageState extends State<BudgetPage> {
  final TextEditingController _carNameController =
TextEditingController();
  final TextEditingController _priceController = TextEditingController();
  final TextEditingController _interestController =
TextEditingController();
  final TextEditingController _durationController =
TextEditingController();
  final TextEditingController _downPaymentController =
TextEditingController();

  String? carName, carType;
```

```
  double? exShowroomPrice, totalCost, loanEMI, totalLoanPayment;
  double gst = 0, cess = 0, roadTax = 0, insurance = 0, registration = 0,
fastag = 500, accessories = 20000;

  @override
  void initState() {
    super.initState();
    if (widget.carName != null) {
      _carNameController.text = widget.carName!;
      carName = widget.carName;
    }
    if (widget.exShowroomPrice != null) {
      _priceController.text = widget.exShowroomPrice!.toString();
      exShowroomPrice = widget.exShowroomPrice;
    }
    if (widget.carType != null) {
      carType = widget.carType;
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Car Budget & Loan Calculator'),
        backgroundColor: Colors.black,
      ),
      backgroundColor: Colors.black,
      body: SingleChildScrollView(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            _buildInputField(_carNameController, 'Car Name'),
            _buildInputField(_priceController, 'Ex-Showroom Price (₹)/Your
estimated price', isNumeric: true),
            _buildDropdownField(),
            _buildInputField(_interestController, 'Interest Rate (%)',
isNumeric: true),
```

```
            _buildInputField(_durationController, 'Loan Duration (Years)',
isNumeric: true),
            _buildInputField(_downPaymentController, 'Down Payment (₹,
Optional)', isNumeric: true),
            const SizedBox(height: 10),
            ElevatedButton(
              onPressed: _calculateTotalCost,
              child: const Text('Get Cost Breakdown'),
            ),
            if (totalCost != null) ...[
              _buildCostBreakdown(),
              ElevatedButton(
                onPressed: _showDetailedBreakdown,
                child: const Text('Show Detailed Breakdown'),
              ),
            ],
          ],
        ),
      ),
    );
  }

  Widget _buildDropdownField() {
    return DropdownButtonFormField<String>(
      value: carType,
      dropdownColor: Colors.black,
      style: const TextStyle(color: Colors.white),
      decoration: _inputDecoration('Select Car Type'),
      items: ['Small Car', 'Mid-Size Car', 'SUV', 'Electric Vehicle (EV)',
'Luxury Car (₹20L+)']
          .map((type) => DropdownMenuItem(value: type, child: Text(type,
style: const TextStyle(color: Colors.white))))
          .toList(),
      onChanged: (value) => setState(() => carType = value),
    );
  }

  Widget _buildInputField(TextEditingController controller, String label,
{bool isNumeric = false}) {
    return Padding(
```

```
      padding: const EdgeInsets.only(bottom: 10),
      child: TextField(
        controller: controller,
        keyboardType: isNumeric ? TextInputType.number :
TextInputType.text,
        style: const TextStyle(color: Colors.white),
        decoration: _inputDecoration(label),
        onChanged: (value) {
          if (controller == _priceController) {
            setState(() {
              exShowroomPrice = double.tryParse(value) ?? 0;
            });
          }
        },
      ),
    );
  }

  InputDecoration _inputDecoration(String label) {
    return InputDecoration(
      labelText: label,
      labelStyle: const TextStyle(color: Colors.white),
      border: const OutlineInputBorder(),
    );
  }

  void _calculateTotalCost() {
    double interestRate = double.tryParse(_interestController.text) ?? 0;
    int loanDuration = int.tryParse(_durationController.text) ?? 0;
    double downPayment = double.tryParse(_downPaymentController.text) ??
0;

    if (exShowroomPrice == null || carType == null) return;

    // Apply tax rules based on car type
    if (carType == 'Small Car') {
      gst = exShowroomPrice! * 0.28;
      cess = exShowroomPrice! * 0.03;
      roadTax = exShowroomPrice! * 0.08;
      insurance = exShowroomPrice! * 0.05;
```

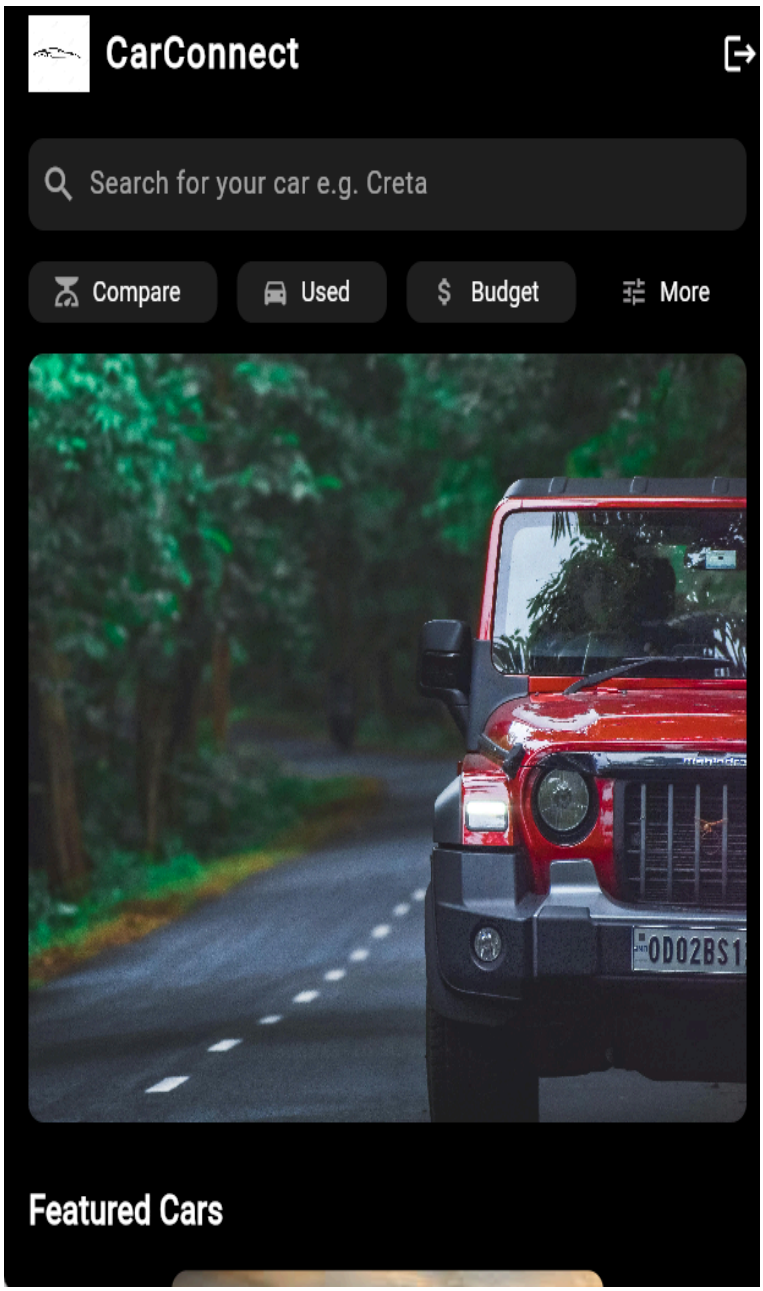
```
      registration = 5000;
    } else if (carType == 'Mid-Size Car') {
      gst = exShowroomPrice! * 0.28;
      cess = exShowroomPrice! * 0.15;
      roadTax = exShowroomPrice! * 0.10;
      insurance = exShowroomPrice! * 0.05;
      registration = 10000;
    } else if (carType == 'SUV') {
      gst = exShowroomPrice! * 0.28;
      cess = exShowroomPrice! * 0.22;
      roadTax = exShowroomPrice! * 0.12;
      insurance = exShowroomPrice! * 0.06;
      registration = 15000;
    } else if (carType == 'Electric Vehicle (EV)') {
      gst = exShowroomPrice! * 0.05;
      cess = 0;
      roadTax = 0;
      insurance = exShowroomPrice! * 0.04;
      registration = 0;
    } else if (carType == 'Luxury Car (₹20L+)') {
      gst = exShowroomPrice! * 0.28;
      cess = exShowroomPrice! * 0.22;
      roadTax = exShowroomPrice! * 0.15;
      insurance = exShowroomPrice! * 0.06;
      registration = 20000;
    }

    // Calculate total on-road cost
    totalCost = exShowroomPrice! + gst + cess + roadTax + insurance +
registration + fastag + accessories;

    // Calculate Loan EMI & Total Loan Cost
    double loanAmount = totalCost! - downPayment;
    double monthlyRate = (interestRate / 100) / 12;
    int totalMonths = loanDuration * 12;

    if (loanAmount > 0 && monthlyRate > 0 && totalMonths > 0) {
      loanEMI = (loanAmount * monthlyRate * pow(1 + monthlyRate,
totalMonths)) /
          (pow(1 + monthlyRate, totalMonths) - 1);
```

```
      totalLoanPayment = loanEMI! * totalMonths;
    }

    setState(() {});
  }

  Widget _buildCostBreakdown() {
    return Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text('Breakdown for $carName ($carType)', style: const
TextStyle(color: Colors.white, fontSize: 18, fontWeight:
FontWeight.bold)),
        _buildCostRow('Total On-Road Price (Without Loan)', totalCost!,
isTotal: true),
        if (loanEMI != null && totalLoanPayment != null) ...[
          _buildCostRow('Monthly EMI', loanEMI!, isTotal: true),
          _buildCostRow('Total Cost with Loan & Interest',
totalLoanPayment!, isTotal: true),
        ],
      ],
    );
  }

  Widget _buildCostRow(String title, double amount, {bool isTotal =
false}) {
    return Padding(
      padding: const EdgeInsets.symmetric(vertical: 6.0),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: [
          Text(title, style: TextStyle(color: Colors.white, fontSize:
isTotal ? 18 : 16, fontWeight: isTotal ? FontWeight.bold :
FontWeight.normal)),
          Text('₹${amount.toStringAsFixed(2)}', style: TextStyle(color:
Colors.white, fontSize: isTotal ? 18 : 16, fontWeight: isTotal ?
FontWeight.bold : FontWeight.normal)),
        ],
      ),
    );
```

```
  }

  void _showDetailedBreakdown() {
    showDialog(
      context: context,
      builder: (context) {
        return AlertDialog(
          backgroundColor: Colors.black,
          title: Text('Detailed Cost Breakdown for $carName ($carType)',
style: const TextStyle(color: Colors.white)),
          content: SingleChildScrollView(
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                _buildCostRow('Ex-Showroom Price', exShowroomPrice!),
                _buildCostRow('GST', gst),
                _buildCostRow('Cess', cess),
                _buildCostRow('Road Tax', roadTax),
                _buildCostRow('Insurance', insurance),
                _buildCostRow('Registration', registration),
                _buildCostRow('Fastag', fastag),
                _buildCostRow('Accessories', accessories),
                _buildCostRow('Total On-Road Price', totalCost!, isTotal:
true),
                if (loanEMI != null && totalLoanPayment != null) ...[
                  _buildCostRow('Loan Amount', totalCost! -
(double.tryParse(_downPaymentController.text) ?? 0)),
                  _buildCostRow('Monthly EMI', loanEMI!),
                  _buildCostRow('Total Loan Payment', totalLoanPayment!),
                ],
              ],
            ),
          ),
          actions: [
            TextButton(
              onPressed: () => Navigator.of(context).pop(),
              child: const Text('Close', style: TextStyle(color:
Colors.white)),
            ),
          ],
```

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

Output:


CarConnect
Search for your car e.g. Creta
Compare
Used
Budget
More
Featured Cars

Find the Right Used Car

Choose your Budget

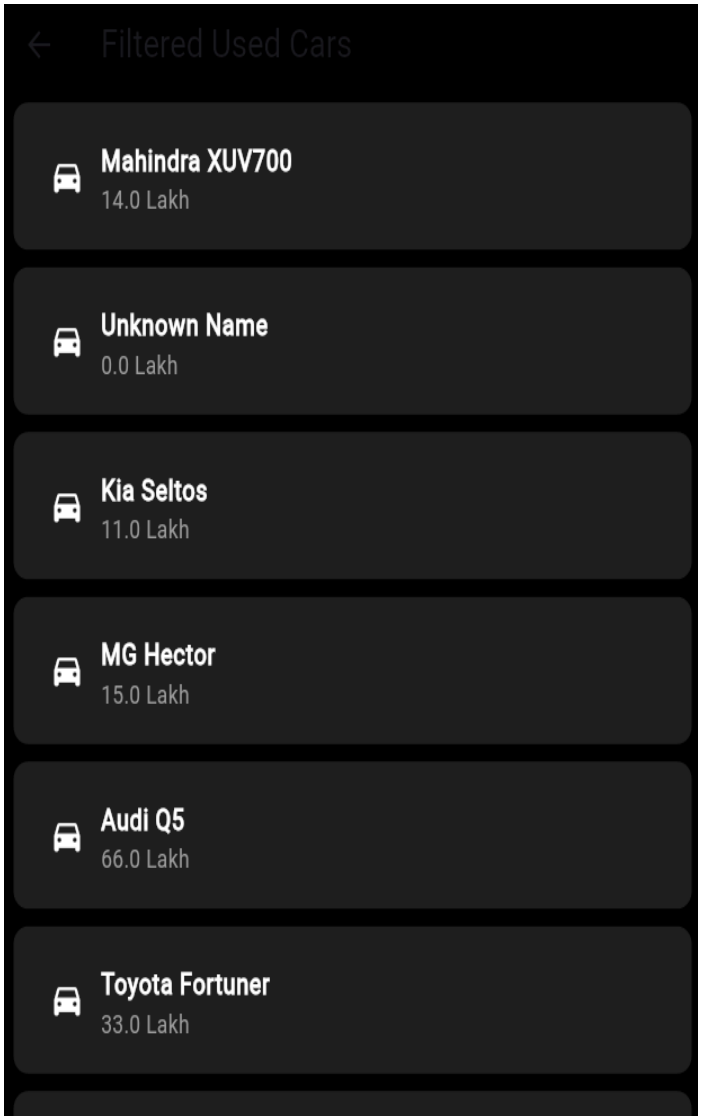
0 Lakh 50 Lakh

Find Used Car

Want to Sell Your Car?

- Get buyers' details via SMS and Email
- Sell your car at best price
- Large number of genuine buyers

Sell Car Online


Filtered Used Cars
Mahindra XUV700
14.0 Lakh
Unknown Name
0.0 Lakh
Kia Seltos
11.0 Lakh
MG Hector
15.0 Lakh
Audi Q5
66.0 Lakh
Toyota Fortuner
33.0 Lakh

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	36
Name	MOHIT PATIL
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Experiment No. 4

Mohit Patil -D15A-36

AIM : To create an interactive Form using form widget.

Widgets :

TextField: For user input (e.g., username, password).
FlatButton or ElevatedButton: For submitting login
credentials. Text: To display static text like "Welcome"
or error messages.
Column: To arrange multiple widgets vertically (e.g., input fields and buttons).
Row: For horizontally aligning widgets (e.g., forgot password link and other actions).
Padding: To add space around widgets.
Icon: To show icons in the input fields or buttons.
Container: For custom styling of sections, buttons, or
inputs. Form: To handle validation of login inputs.
GestureDetector: For detecting taps (e.g., to navigate or submit).
ThemeData: To define the overall visual properties of your app (colors, fonts, etc.).
MaterialApp: To wrap your app with a theme and other configurations.

Code :

- **Register page code:**

```dart
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'login.dart';

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

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

class _RegisterPageState extends State<RegisterPage> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  final FirebaseAuth _auth = FirebaseAuth.instance;

  void _registerUser() async {
    if (_formKey.currentState!.validate()) {
      try {
```

```
      await _auth.createUserWithEmailAndPassword(
        email: _emailController.text.trim(),
        password: _passwordController.text.trim(),
      );
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Registration Successful!')),
      );
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => const LoginPage()),
      );
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Registration Failed: $e')),
      );
    }
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Container(
      color: Colors.black,
      child: Center(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Card(
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(20),
            ),
            elevation: 10,
            color: Colors.white,
            child: Padding(
              padding: const EdgeInsets.all(30.0),
              child: Form(
                key: _formKey,
                child: Column(
                  mainAxisSize: MainAxisSize.min,
                  crossAxisAlignment: CrossAxisAlignment.stretch,
                  children: [
                    Column(
                      children: [
                        Image.asset(
                          'assets/c1.png',
                          height: 150,
                        ),
                        const SizedBox(height: 10),
                        Text(
```

```
            'CarConnect',
            style: GoogleFonts.alegreyaSans(
              fontSize: 40,
              fontWeight: FontWeight.bold,
              color: Colors.black,
            ),
          ),
        ],
      ),
      const SizedBox(height: 20),
      TextFormField(
        controller: _emailController,
        decoration: InputDecoration(
          labelText: 'Email',
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(12),
          ),
          prefixIcon:
              const Icon(Icons.email, color: Colors.black),
        ),
        validator: (value) {
          if (value == null ||
              value.isEmpty ||
              !value.contains('@')) {
            return 'Enter a valid email';
          }
          return null;
        },
      ),
      const SizedBox(height: 15),
      TextFormField(
        controller: _passwordController,
        obscureText: true,
        decoration: InputDecoration(
          labelText: 'Password',
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(12),
          ),
          prefixIcon:
              const Icon(Icons.lock, color: Colors.black),
        ),
        validator: (value) {
          if (value == null || value.length < 6) {
            return 'Password must be at least 6 characters';
          }
          return null;
        },
      ),
      const SizedBox(height: 20),
```

```
                    ElevatedButton(
                      style: ElevatedButton.styleFrom(
                        padding: const EdgeInsets.symmetric(vertical: 14),
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(12),
                        ),
                        backgroundColor: Colors.black,
                      ),
                      onPressed: _registerUser,
                      child: const Text(
                        'Register',
                        style: TextStyle(fontSize: 20, color: Colors.white),
                      ),
                    ),
                    const SizedBox(height: 15),
                    TextButton(
                      onPressed: () {
                        // Simulate successful registration, then navigate to
login page
                        Navigator.pushReplacementNamed(context, '/login');
                      },
                      child: const Text(
                        'Already have an account? Login',
                        style: TextStyle(color: Colors.black),
                      ),
                    ),
                  ],
                ),
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

- **Login page code:**

```dart
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:google_sign_in/google_sign_in.dart';

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

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

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

  // Use the renderButton method for Google Sign-In on web
  void _signInWithGoogle() async {
    final GoogleSignIn googleSignIn = GoogleSignIn();

    try {
      GoogleSignInAccount? googleUser = await googleSignIn.signIn();
      if (googleUser == null) return;

      final GoogleSignInAuthentication googleAuth =
          await googleUser.authentication;

      final AuthCredential credential = GoogleAuthProvider.credential(
        accessToken: googleAuth.accessToken,
        idToken: googleAuth.idToken,
      );

      await _auth.signInWithCredential(credential);
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Google Sign-In Successful!')),
      );
    } catch (e) {
      print("Google Sign-In Error: $e");
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Google Sign-In Failed: $e')),
      );
    }
  }

  void _signInWithEmailAndPassword() async {
```

```
  if (_formKey.currentState!.validate()) {
    try {
      await _auth.signInWithEmailAndPassword(
        email: _emailController.text.trim(),
        password: _passwordController.text.trim(),
      );
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Login Successful!')),
      );

      // Redirect to Home Page after login
      Navigator.pushReplacementNamed(context, '/home');
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Login Failed: $e')),
      );
    }
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Container(
      color: Colors.black,
      child: Center(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Card(
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(20),
            ),
            elevation: 10,
            color: Colors.white,
            child: Padding(
              padding: const EdgeInsets.all(30.0),
              child: Form(
                key: _formKey,
                child: Column(
                  mainAxisSize: MainAxisSize.min,
                  crossAxisAlignment: CrossAxisAlignment.stretch,
                  children: [
                    Column(
                      children: [
                        Image.asset(
                          'assets/c1.png', // Ensure image is in assets folder
                          height: 150,
                        ),
                        const SizedBox(height: 10),
```

```
          Text(
            'CarConnect',
            style: GoogleFonts.alegreyaSans(
              fontSize: 40,
              fontWeight: FontWeight.bold,
              color: Colors.black,
            ),
          ),
        ],
      ),
      const SizedBox(height: 20),
      TextFormField(
        controller: _emailController,
        decoration: InputDecoration(
          labelText: 'Email',
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(12),
          ),
          prefixIcon:
              const Icon(Icons.email, color: Colors.black),
        ),
        validator: (value) {
          if (value == null ||
              value.isEmpty ||
              !value.contains('@')) {
            return 'Enter a valid email';
          }
          return null;
        },
      ),
      const SizedBox(height: 15),
      TextFormField(
        controller: _passwordController,
        obscureText: true,
        decoration: InputDecoration(
          labelText: 'Password',
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(12),
          ),
          prefixIcon:
              const Icon(Icons.lock, color: Colors.black),
        ),
        validator: (value) {
          if (value == null || value.length < 6) {
            return 'Password must be at least 6 characters';
          }
          return null;
        },
      ),
```

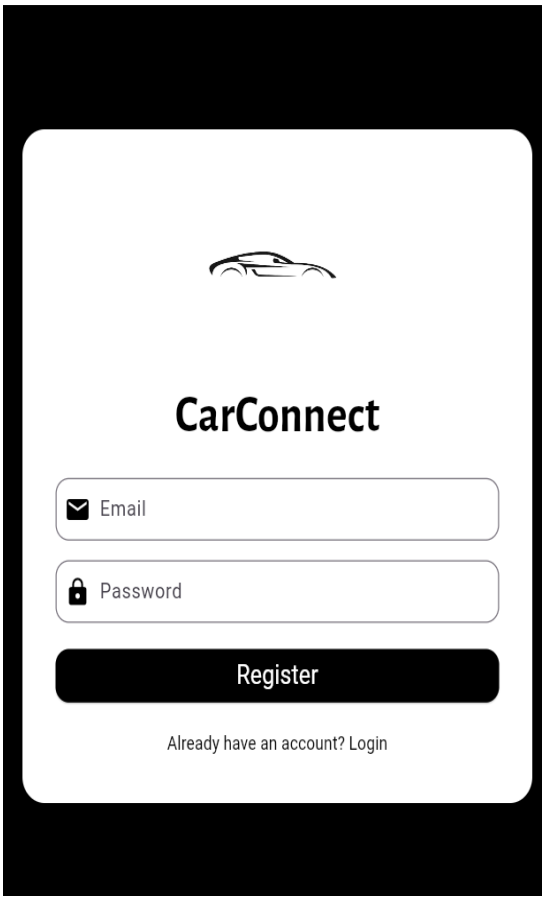
```
                    const SizedBox(height: 20),
                    ElevatedButton(
                      style: ElevatedButton.styleFrom(
                        padding: const EdgeInsets.symmetric(vertical: 14),
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(12),
                        ),
                        backgroundColor: Colors.black,
                      ),
                      onPressed: _signInWithEmailAndPassword,
                      child: const Text(
                        'Login',
                        style: TextStyle(fontSize: 20, color: Colors.white),
                      ),
                    ),
                    const SizedBox(height: 15),
                    ElevatedButton(
                      style: ElevatedButton.styleFrom(
                        backgroundColor: Colors.white,
                        padding: const EdgeInsets.symmetric(vertical: 14),
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(12),
                          side: const BorderSide(color: Colors.black),
                        ),
                      ),
                      onPressed: _signInWithGoogle,
                      child: const Text(
                        'Sign in with Google',
                        style: TextStyle(fontSize: 20, color: Colors.black),
                      ),
                    ),
                  ],
                ),
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

- **Main.dart file code:**

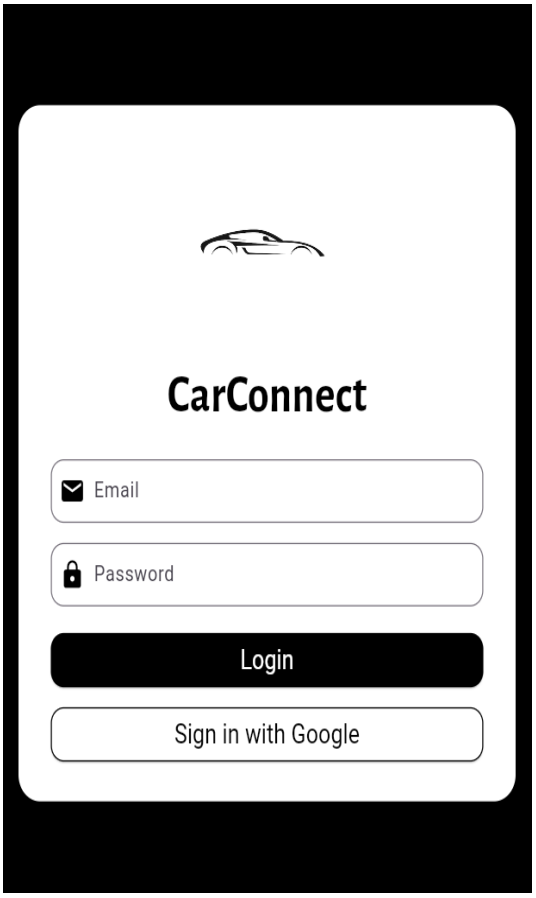
```dart
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:proj1/screens/login.dart';
import 'package:proj1/screens/RegisterPage.dart';
import 'package:proj1/screens/home.dart'; // Import Home Page
import 'firebase_options.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  runApp(const MyApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      initialRoute: '/register',
      routes: {
        '/register': (context) => const RegisterPage(),
        '/login': (context) => const LoginPage(),
        '/home': (context) => const HomePage(), // Add Home Page Route
      },
    );
  }
}
```

Screen Shots :

1.Register page



2.Login page



MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	36
Name	MOHIT PATIL
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

MPL Experiment 5

Name: Mohit Patil **Class:** D15A **Roll no:**36

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.

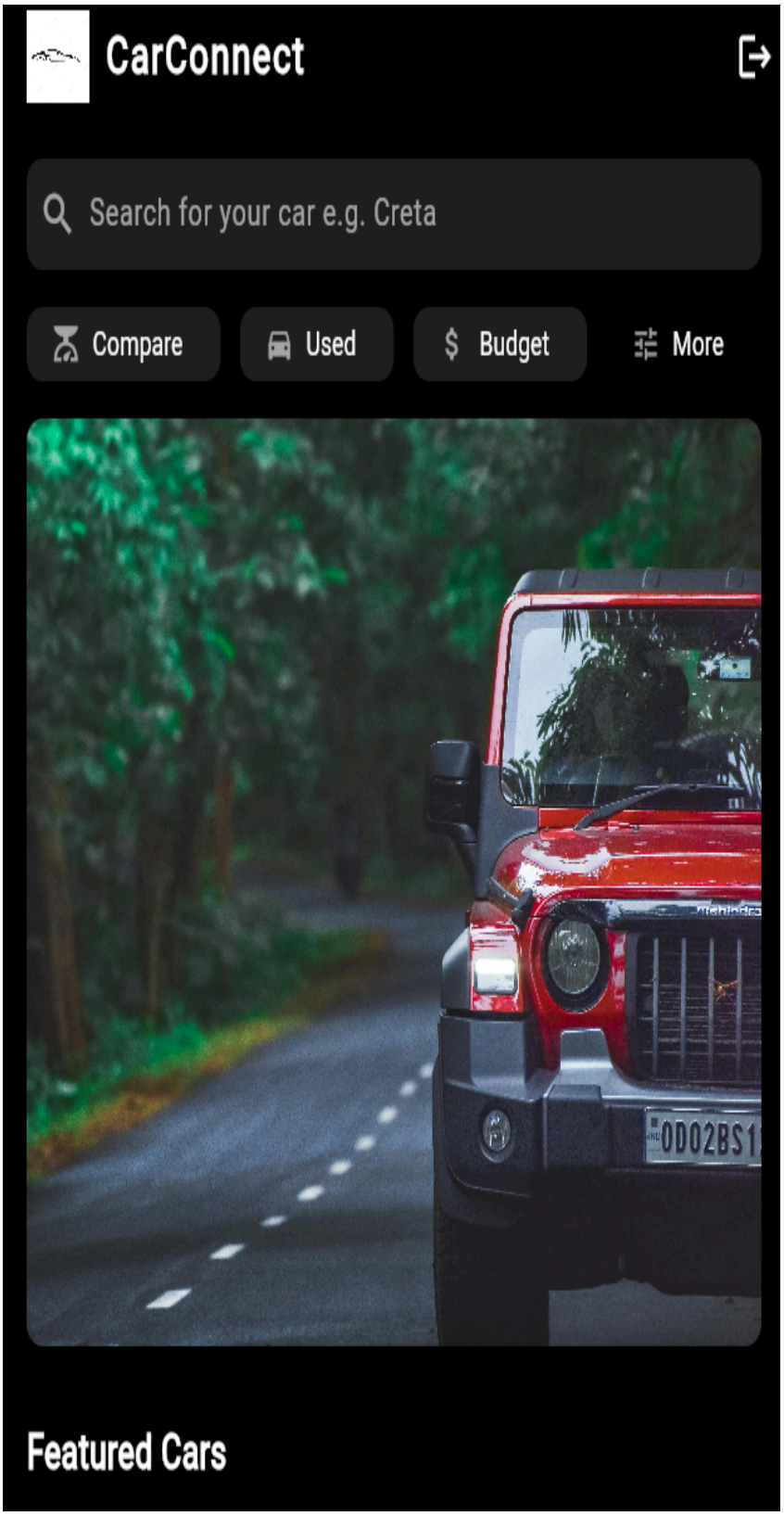
Steps:
Step 1: Create a Flutter project and define multiple screens.
Step 2: Set up named routes in `MaterialApp`.
Step 3: Implement navigation using `Navigator.push()` and `Navigator.pushNamed()`.
Step 4: Use `GestureDetector` to detect taps, swipes, and long presses.
Step 5: Add buttons or swipes to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

Code:

```dart
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:proj1/screens/login.dart';
import 'package:proj1/screens/RegisterPage.dart';
import 'package:proj1/screens/home.dart';
import 'package:proj1/screens/UsedCarScreen.dart';
```

```dart
import 'package:proj1/screens/FilteredUsedCarsScreen.dart';
import 'firebase_options.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  runApp(const MyApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      initialRoute: '/login', // Default route to Login Page
      routes: {
        '/register': (context) => const RegisterPage(),
        '/login': (context) => const LoginPage(),
        '/home': (context) => const HomePage(),
        '/usedCars': (context) => UsedCarScreen(),
      },
      onGenerateRoute: (settings) {
        if (settings.name == '/filteredCars') {
          final args = settings.arguments as Map<String, int>;
          return MaterialPageRoute(
            builder: (context) => FilteredUsedCarsScreen(
              minPrice: args['minBudget']!,
              maxPrice: args['maxBudget']!,
            ),
          );
        }
        return null;
      },
    );
  }
}
```

Output:


CarConnect
Search for your car e.g. Creta
Compare
Used
$ Budget
More
Featured Cars

Compare Cars

Select Car VS Select Car

Compare

POPULAR CAR COMPARISONS

Tata Harrier
onwards

VS

Kia Sonet
onwards

BMW 3 Series
onwards

VS

Audi Q5
onwards

Car Comparison

Feature	MG Hector	Mahindra XUV700
Price	1500000	1400000
Engine	1.5L	2.2L
Mileage	14 kmpl	15 kmpl
Horsepower	143 HP	200 HP
Torque	250 Nm	450 Nm
Fuel Type	Petrol	Diesel
Transmission	CVT	Automatic
Seating Capacity	5	7

Find the Right Used Car

Choose your Budget

0 Lakh 50 Lakh

Find Used Car

Want to Sell Your Car?

- Get buyers' details via SMS and Email
- Sell your car at best price
- Large number of genuine buyers

Sell Car Online

Mahindra XUV700
14.0 Lakh

Unknown Name
0.0 Lakh

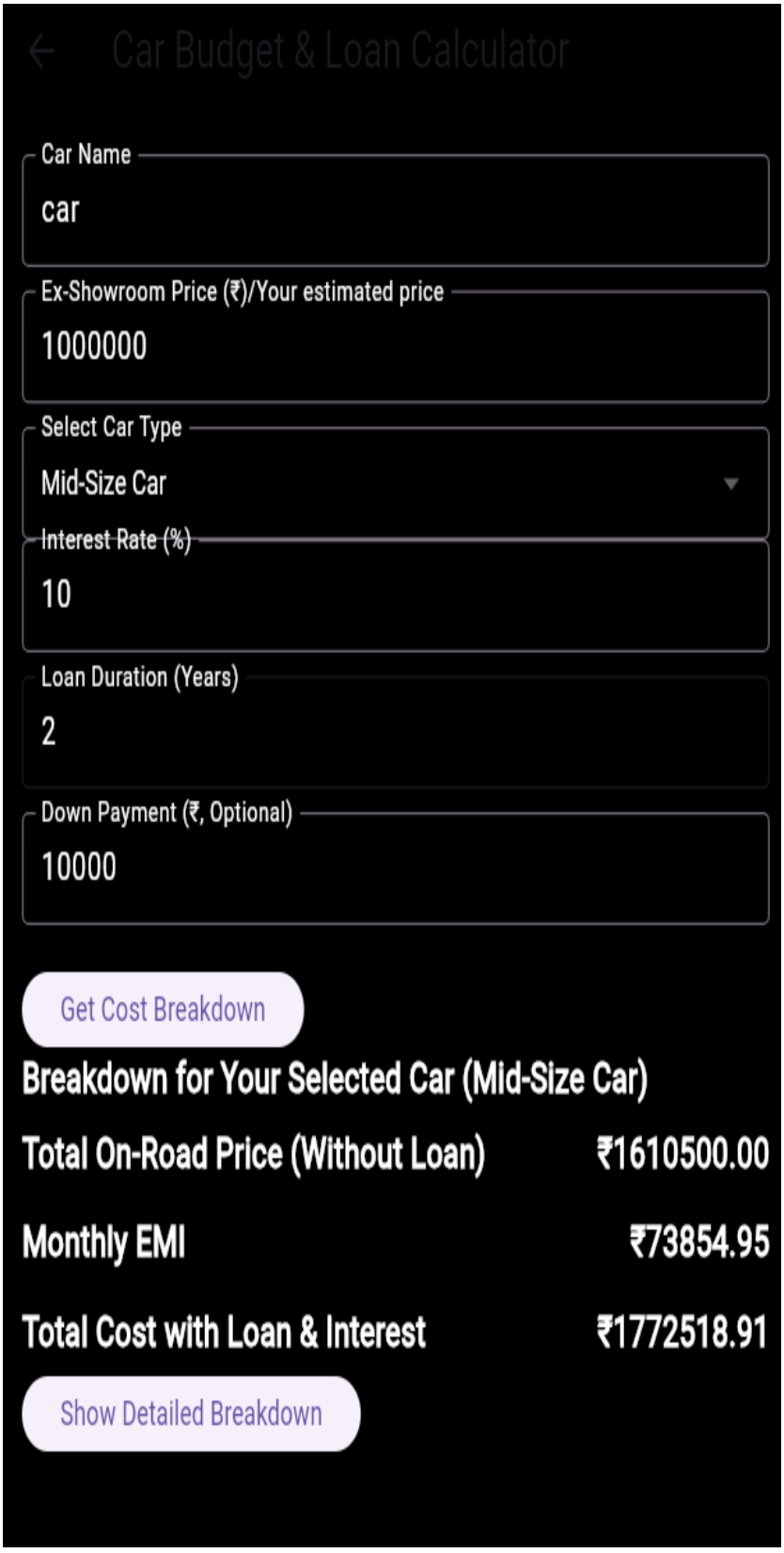
Kia Seltos
11.0 Lakh

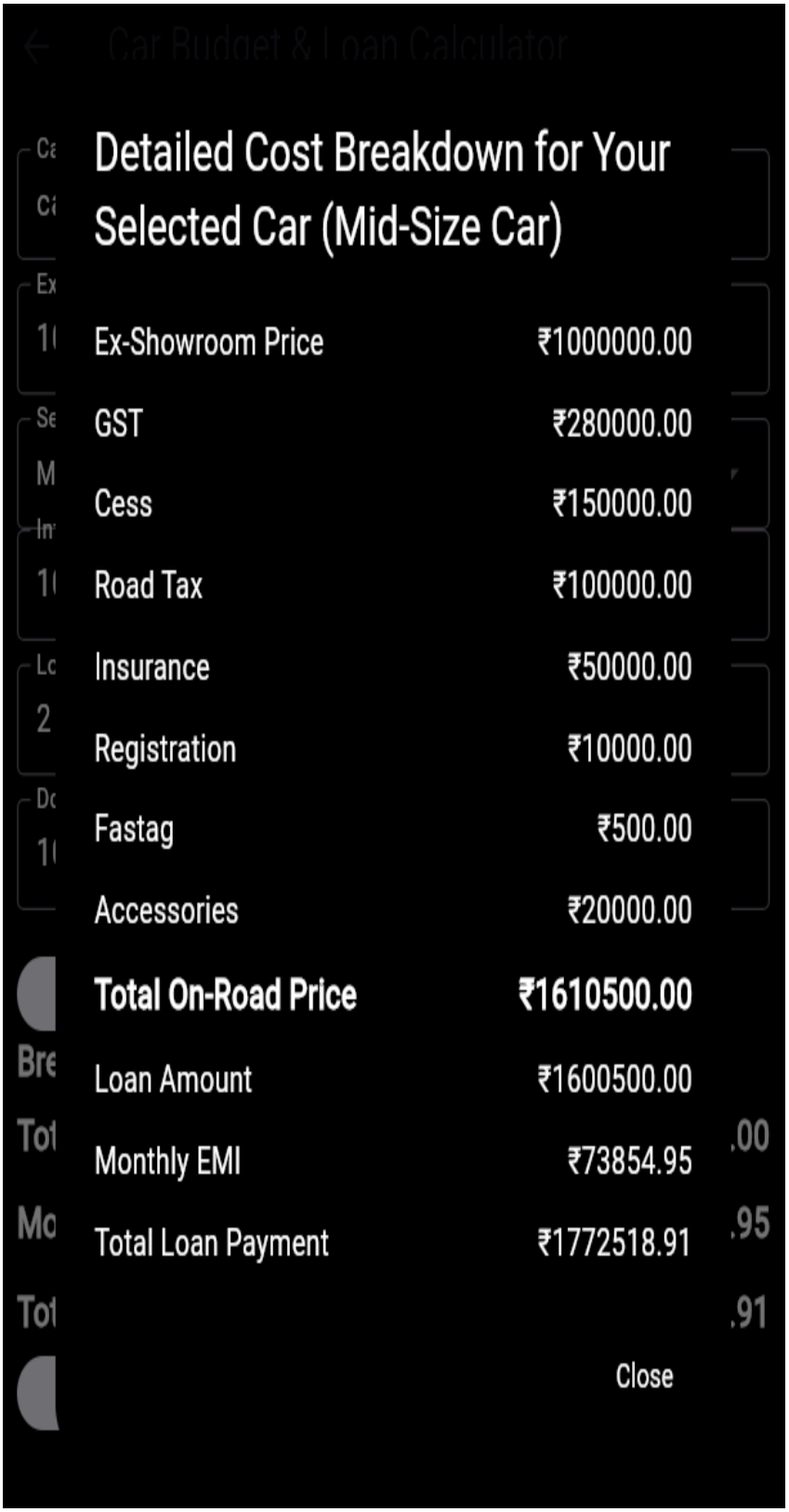
MG Hector
15.0 Lakh

Audi Q5
66.0 Lakh

Toyota Fortuner
33.0 Lakh

Mercedes-Benz E-Class


Car Budget & Loan Calculator
Car Name
car
Ex-Showroom Price (₹)/Your estimated price
1000000
Select Car Type
Mid-Size Car
Interest Rate (%)
10
Loan Duration (Years)
2
Down Payment (₹, Optional)
10000
Get Cost Breakdown
Breakdown for Your Selected Car (Mid-Size Car)
Total On-Road Price (Without Loan) ₹1610500.00
Monthly EMI ₹73854.95
Total Cost with Loan & Interest ₹1772518.91
Show Detailed Breakdown


Car Budget & Loan Calculator
Detailed Cost Breakdown for Your Selected Car (Mid-Size Car)
Ex-Showroom Price ₹1000000.00
GST ₹280000.00
Cess ₹150000.00
Road Tax ₹100000.00
Insurance ₹50000.00
Registration ₹10000.00
Fastag ₹500.00
Accessories ₹20000.00
Total On-Road Price ₹1610500.00
Loan Amount ₹1600500.00
Monthly EMI ₹73854.95
Total Loan Payment ₹1772518.91
Close

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	36
Name	MOHIT PATIL
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:	

MPL Experiment 6

Name: Mohit Patil **Class:** D15A **Roll no:**36

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

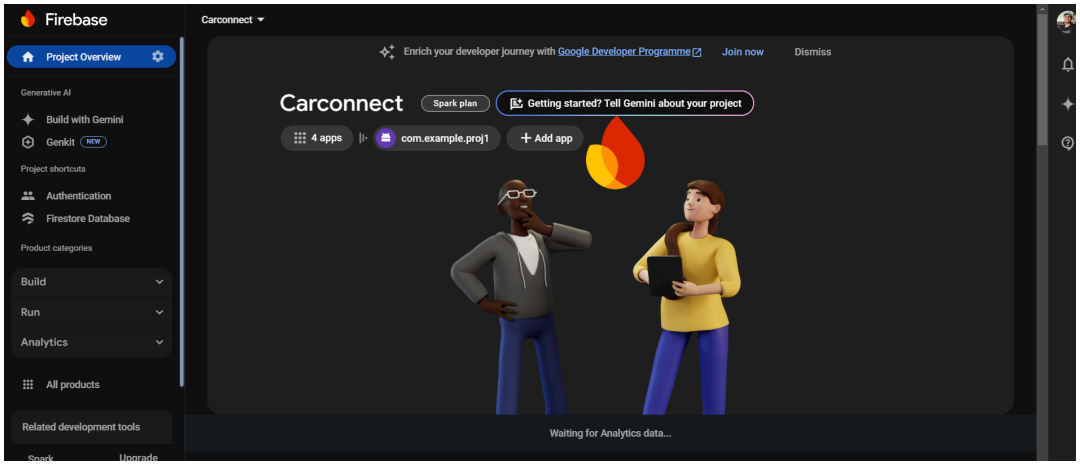
Theory: -

Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. Firebase, a Backend-as-a-Service (BaaS) platform, provides real-time database, authentication, and cloud storage services, making it a powerful backend solution for Flutter applications.
By integrating Firebase with Flutter, developers can store and retrieve data in real time, authenticate users, and manage cloud-based data efficiently. This is particularly useful for applications requiring dynamic content updates and user interactions.

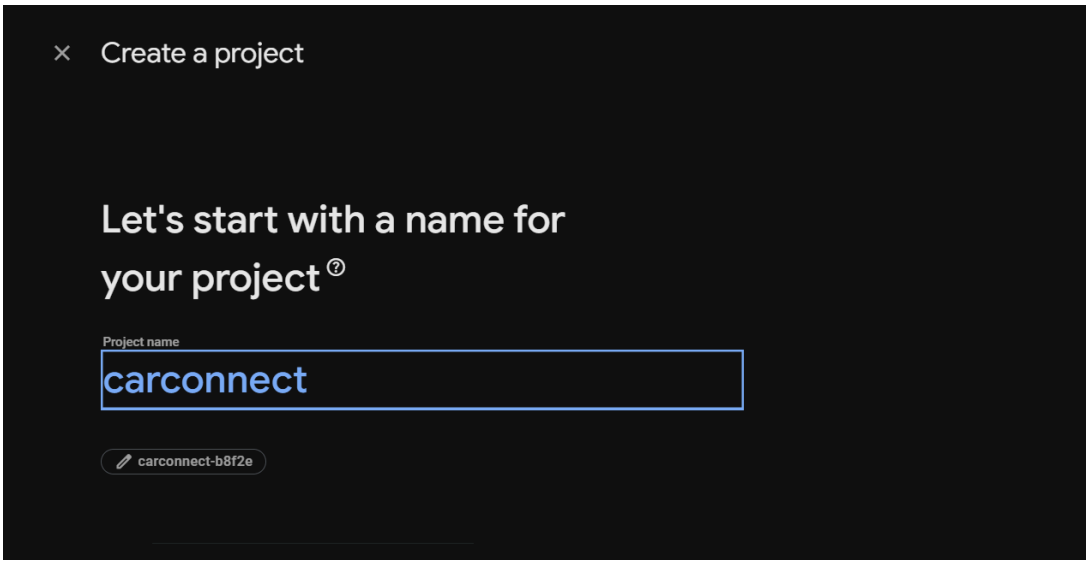
Steps to Connect Flutter UI with Firebase Database:

Step 1:

1.1) Go to Firebase Console and Create a Firebase Project

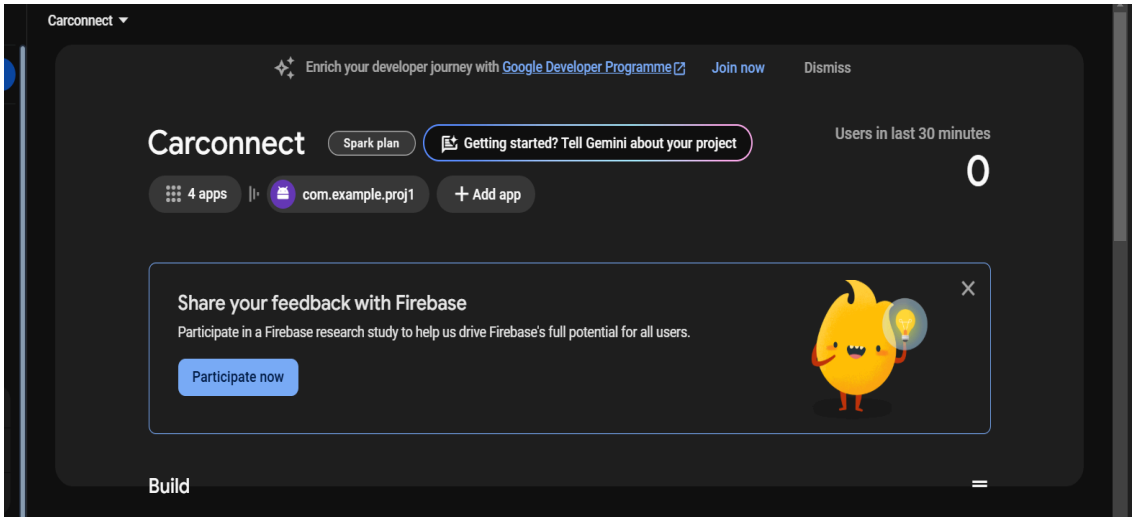


1.2) Click on Create a Project and give it a suitable name.



Step 2:- Add Firebase to Your Flutter App

2.1) Click on Android/iOS/Web based on your Flutter application

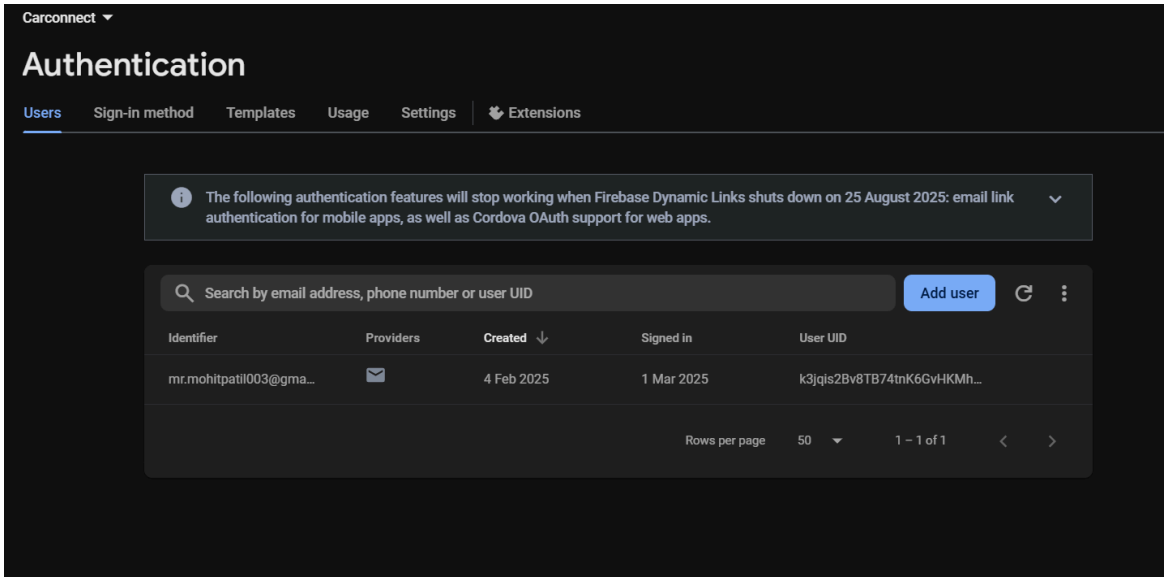


Step 3: - Add Firebase Authentication to Your App

3.1) Add Firebase Authentication Dependencies

```yaml
dependencies:
  flutter:
    sdk: flutter
  firebase_core: ^3.11.0
  firebase_auth: ^5.4.2  # For authentication
  cloud_firestore: ^5.6.3  # For Firestore, if you need it
  firebase_messaging: ^15.2.2
  http: ^0.13.3
  image_picker: ^1.0.4
  tflite_flutter: ^0.11.0
  image: ^3.2.0
  url_launcher: ^6.1.14
```

3.2) Enable Authentication in Firebase Console Go to Firebase Console → Authentication. Click on Sign-in method and enable Email/Password (or any other method like Google). Click Save



3.3)

Implement Authentication in Flutter Modify main.dart

import 'package:firebase_core/firebase_core.dart'; import 'package:firebase_auth/firebase_auth.dart';

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

Step 4: -Configure Firebase Realtime Database

1. Go to Firebase Console → Realtime Database.
2. Click Create Database → Choose location → Set rules (for development, set read/write to true).
3. Click Publish.

CODE:

Login page.dart

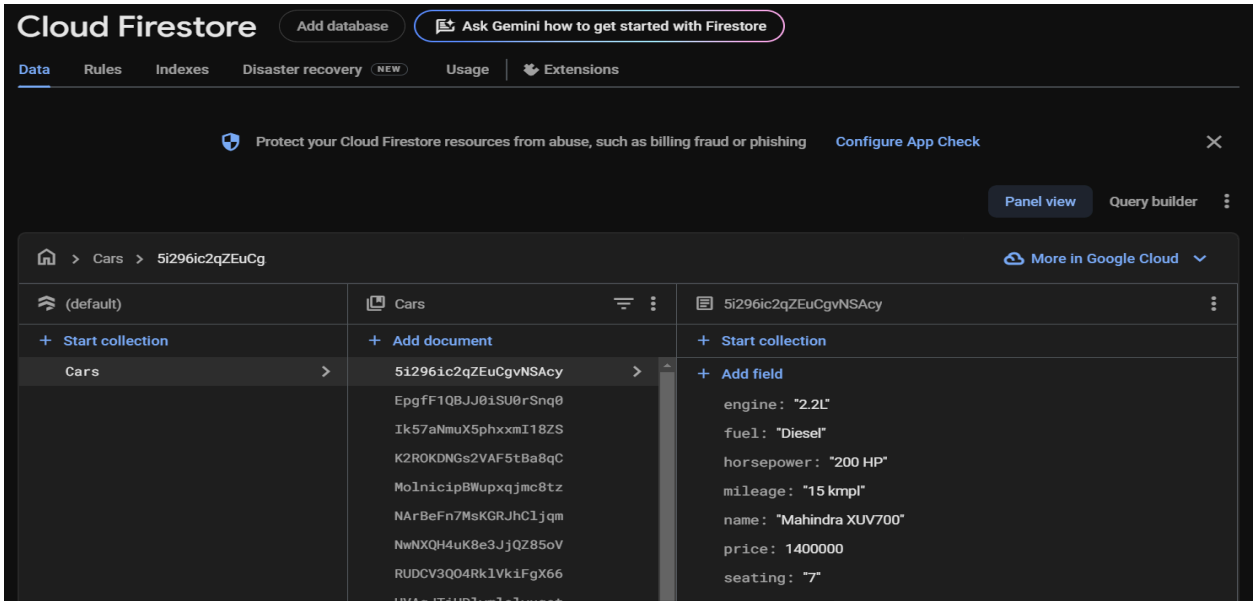
```dart
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:google_sign_in/google_sign_in.dart';

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

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

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

  // Use the renderButton method for Google Sign-In on web
  void _signInWithGoogle() async {
    final GoogleSignIn googleSignIn = GoogleSignIn();

    try {
```

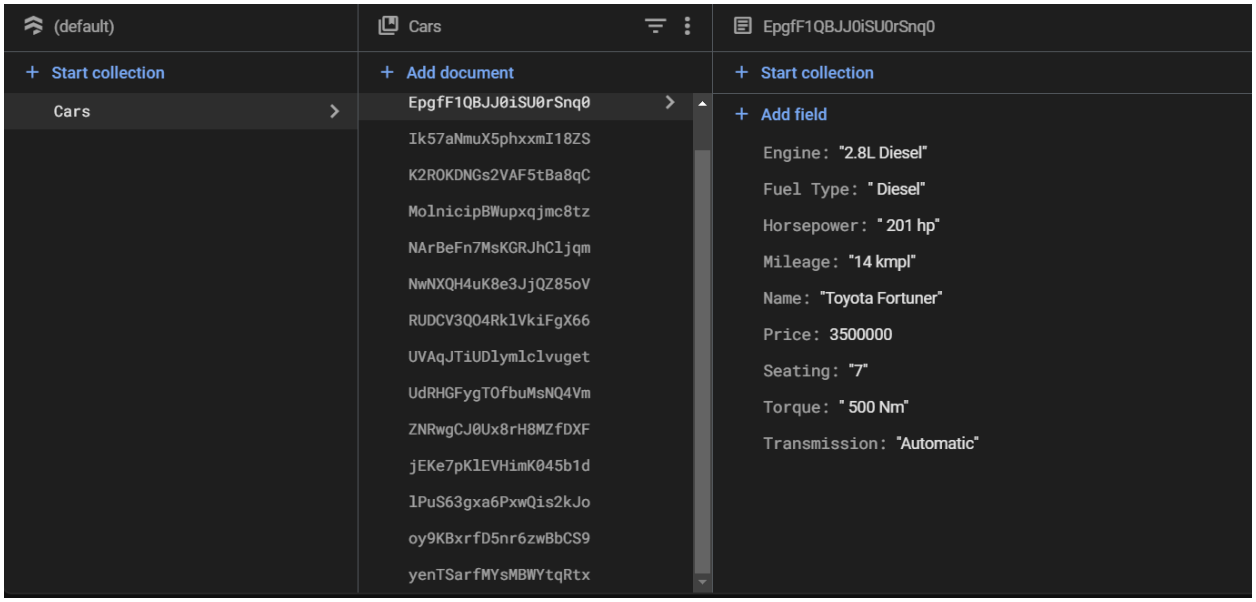
```
    GoogleSignInAccount? googleUser = await googleSignIn.signIn();
    if (googleUser == null) return;

    final GoogleSignInAuthentication googleAuth =
        await googleUser.authentication;

    final AuthCredential credential = GoogleAuthProvider.credential(
      accessToken: googleAuth.accessToken,
      idToken: googleAuth.idToken,
    );

    await _auth.signInWithCredential(credential);
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Google Sign-In Successful!')),
    );
  } catch (e) {
    print("Google Sign-In Error: $e");
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Google Sign-In Failed: $e')),
    );
  }
}

void _signInWithEmailAndPassword() async {
  if (_formKey.currentState!.validate()) {
    try {
      await _auth.signInWithEmailAndPassword(
        email: _emailController.text.trim(),
        password: _passwordController.text.trim(),
      );
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Login Successful!')),
      );

      // Redirect to Home Page after login
      Navigator.pushReplacementNamed(context, '/home');
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Login Failed: $e')),
      );
    }
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
```

```
    body: Container(
      color: Colors.black,
      child: Center(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Card(
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(20),
            ),
            elevation: 10,
            color: Colors.white,
            child: Padding(
              padding: const EdgeInsets.all(30.0),
              child: Form(
                key: _formKey,
                child: Column(
                  mainAxisSize: MainAxisSize.min,
                  crossAxisAlignment: CrossAxisAlignment.stretch,
                  children: [
                    Column(
                      children: [
                        Image.asset(
                          'assets/c1.png', // Ensure image is in assets folder
                          height: 150,
                        ),
                        const SizedBox(height: 10),
                        Text(
                          'CarConnect',
                          style: GoogleFonts.alegreyaSans(
                            fontSize: 40,
                            fontWeight: FontWeight.bold,
                            color: Colors.black,
                          ),
                        ),
                      ],
                    ),
                    const SizedBox(height: 20),
                    TextFormField(
                      controller: _emailController,
                      decoration: InputDecoration(
                        labelText: 'Email',
                        border: OutlineInputBorder(
                          borderRadius: BorderRadius.circular(12),
                        ),
                        prefixIcon:
                            const Icon(Icons.email, color: Colors.black),
                      ),
```

```
      validator: (value) {
        if (value == null ||
            value.isEmpty ||
            !value.contains('@')) {
          return 'Enter a valid email';
        }
        return null;
      },
    ),
    const SizedBox(height: 15),
    TextFormField(
      controller: _passwordController,
      obscureText: true,
      decoration: InputDecoration(
        labelText: 'Password',
        border: OutlineInputBorder(
          borderRadius: BorderRadius.circular(12),
        ),
        prefixIcon:
            const Icon(Icons.lock, color: Colors.black),
      ),
      validator: (value) {
        if (value == null || value.length < 6) {
          return 'Password must be at least 6 characters';
        }
        return null;
      },
    ),
    const SizedBox(height: 20),
    ElevatedButton(
      style: ElevatedButton.styleFrom(
        padding: const EdgeInsets.symmetric(vertical: 14),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(12),
        ),
        backgroundColor: Colors.black,
      ),
      onPressed: _signInWithEmailAndPassword,
      child: const Text(
        'Login',
        style: TextStyle(fontSize: 20, color: Colors.white),
      ),
    ),
    const SizedBox(height: 15),
    ElevatedButton(
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.white,
```

```
                        padding: const EdgeInsets.symmetric(vertical: 14),
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(12),
                          side: const BorderSide(color: Colors.black),
                        ),
                      ),
                      onPressed: _signInWithGoogle,
                      child: const Text(
                        'Sign in with Google',
                        style: TextStyle(fontSize: 20, color: Colors.black),
                      ),
                    ),
                  ],
                ),
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```




(default)
Start collection
Cars
Cars
Add document
EpgfF1QBJJ0iSU0rSnq0
Ik57aNmuX5phxxmI18ZS
K2ROKDNGs2VAF5tBa8qC
MolnicipBWupxqjmc8tz
NArBeFn7MsKGRJhCljqm
NwNXQH4uK8e3JjQZ85oV
RUDCV3QO4RklVkiFgX66
UVAqJTiUDlymlclvuget
UdRHGFygTOfbuMsNQ4Vm
ZNRwgCJ0Ux8rH8MZfDXF
jEKe7pKlEVHimK045b1d
lPuS63gxa6PxwQis2kJo
oy9KBxrfD5nr6zwBbCS9
yenTSarfMYsMBWYtqRtx
EpgfF1QBJJ0iSU0rSnq0
Start collection
Add field
Engine: "2.8L Diesel"
Fuel Type: " Diesel"
Horsepower: " 201 hp"
Mileage: "14 kmpl"
Name: "Toyota Fortuner"
Price: 3500000
Seating: "7"
Torque: " 500 Nm"
Transmission: "Automatic"

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	36
Name	MOHIT PATIL
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:	

MPL Experiment 6

Name: Mohit Patil **Class:** D15A **Roll no:**36

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

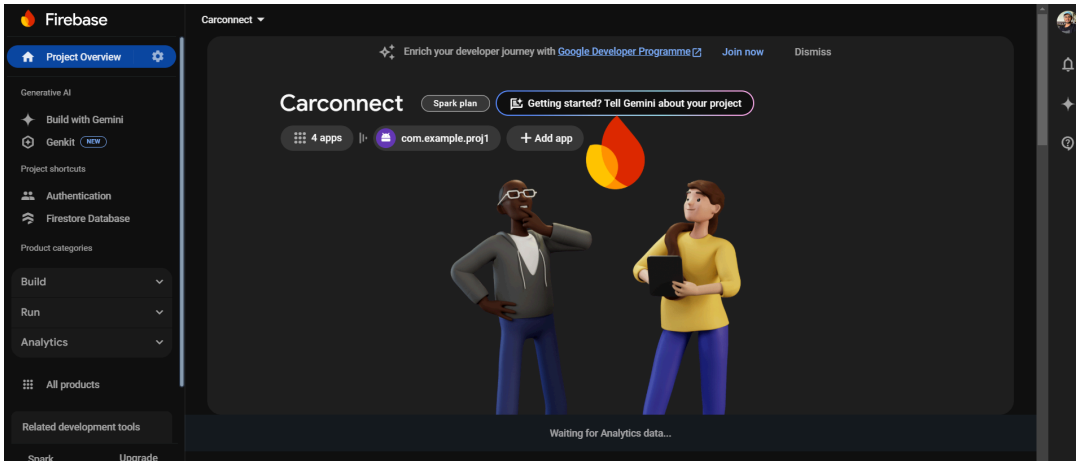
Theory: -

Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. Firebase, a Backend-as-a-Service (BaaS) platform, provides real-time database, authentication, and cloud storage services, making it a powerful backend solution for Flutter applications.
By integrating Firebase with Flutter, developers can store and retrieve data in real time, authenticate users, and manage cloud-based data efficiently. This is particularly useful for applications requiring dynamic content updates and user interactions.

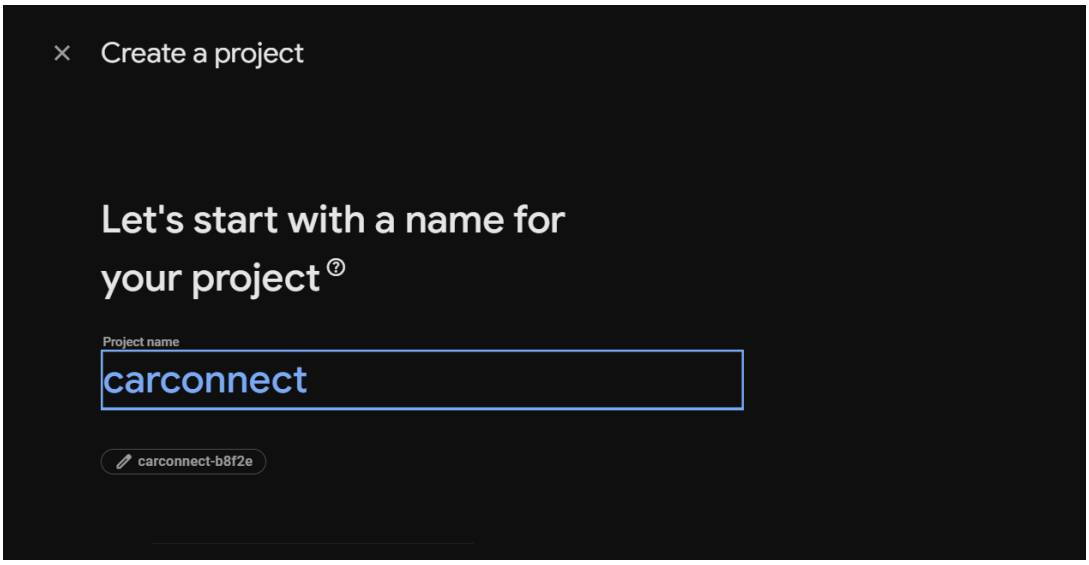
Steps to Connect Flutter UI with Firebase Database:

Step 1:

1.1) Go to Firebase Console and Create a Firebase Project

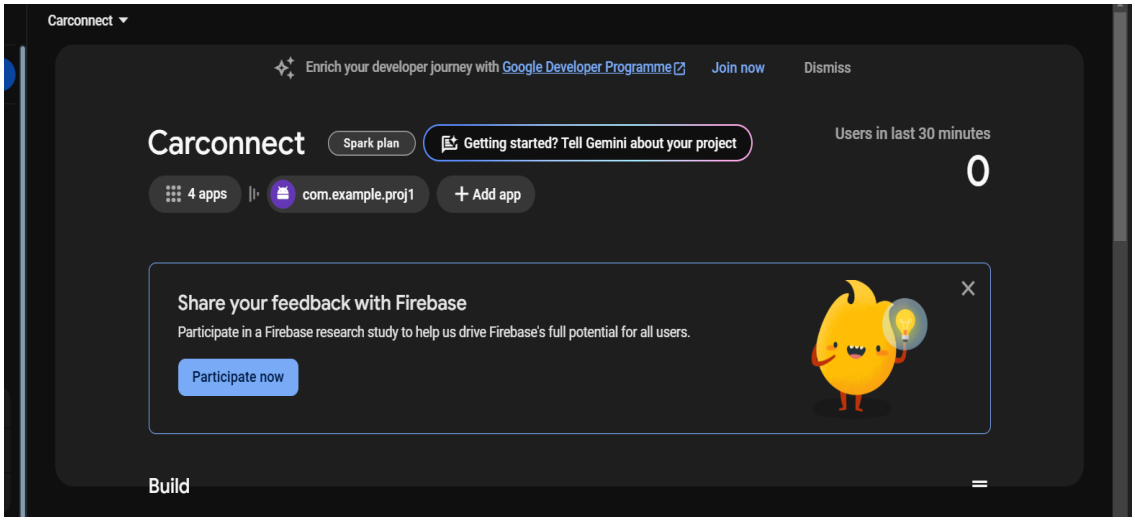


1.2) Click on Create a Project and give it a suitable name.



Step 2:- Add Firebase to Your Flutter App

2.1) Click on Android/iOS/Web based on your Flutter application

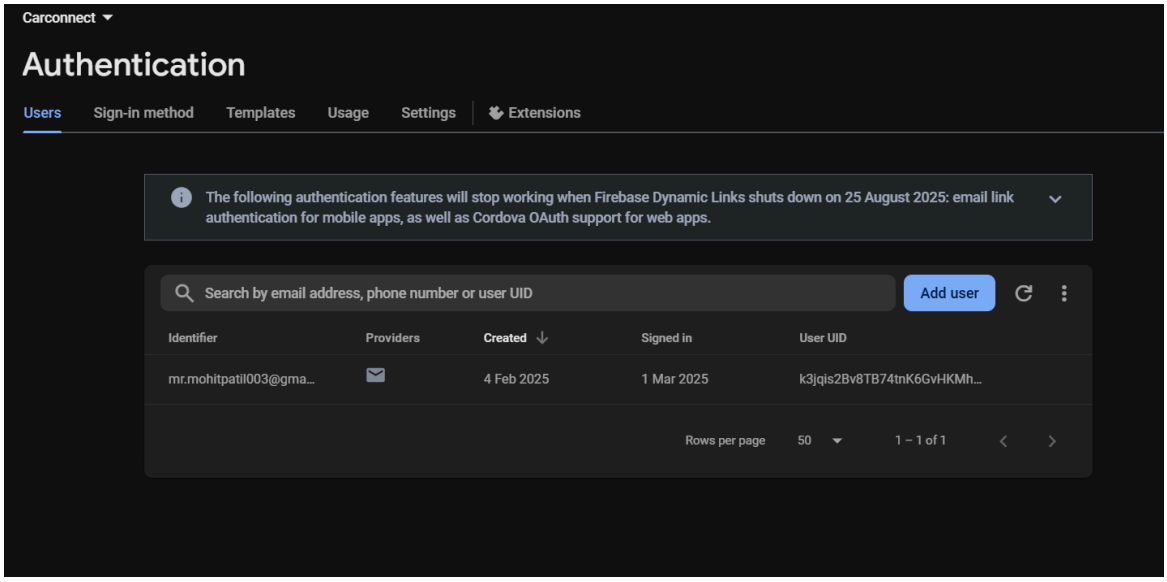


Step 3: - Add Firebase Authentication to Your App

3.1) Add Firebase Authentication Dependencies

```yaml
dependencies:
  flutter:
    sdk: flutter
  firebase_core: ^3.11.0
  firebase_auth: ^5.4.2  # For authentication
  cloud_firestore: ^5.6.3  # For Firestore, if you need it
  firebase_messaging: ^15.2.2
  http: ^0.13.3
  image_picker: ^1.0.4
  tflite_flutter: ^0.11.0
  image: ^3.2.0
  url_launcher: ^6.1.14
```

3.2) Enable Authentication in Firebase Console Go to Firebase Console → Authentication. Click on Sign-in method and enable Email/Password (or any other method like Google). Click Save



3.3)

Implement Authentication in Flutter Modify main.dart

import 'package:firebase_core/firebase_core.dart'; import 'package:firebase_auth/firebase_auth.dart';

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

Step 4: -Configure Firebase Realtime Database

1. Go to Firebase Console → Realtime Database.
2. Click Create Database → Choose location → Set rules (for development, set read/write to true).
3. Click Publish.

CODE:

Login page.dart

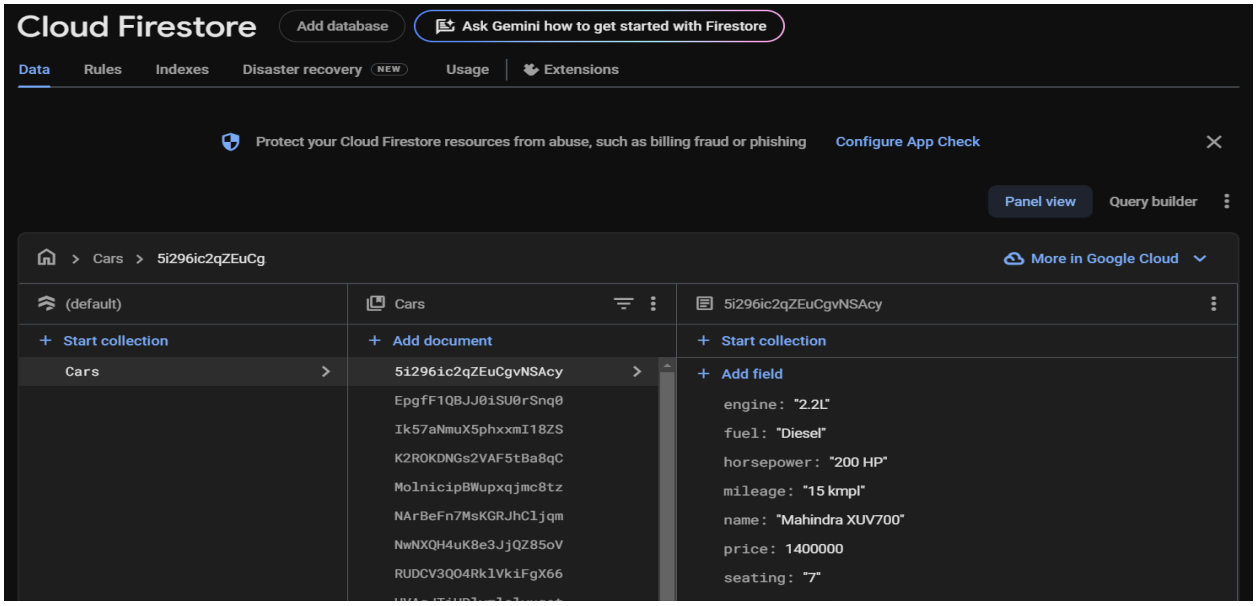
```dart
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:google_sign_in/google_sign_in.dart';

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

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

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

  // Use the renderButton method for Google Sign-In on web
  void _signInWithGoogle() async {
    final GoogleSignIn googleSignIn = GoogleSignIn();

    try {
```

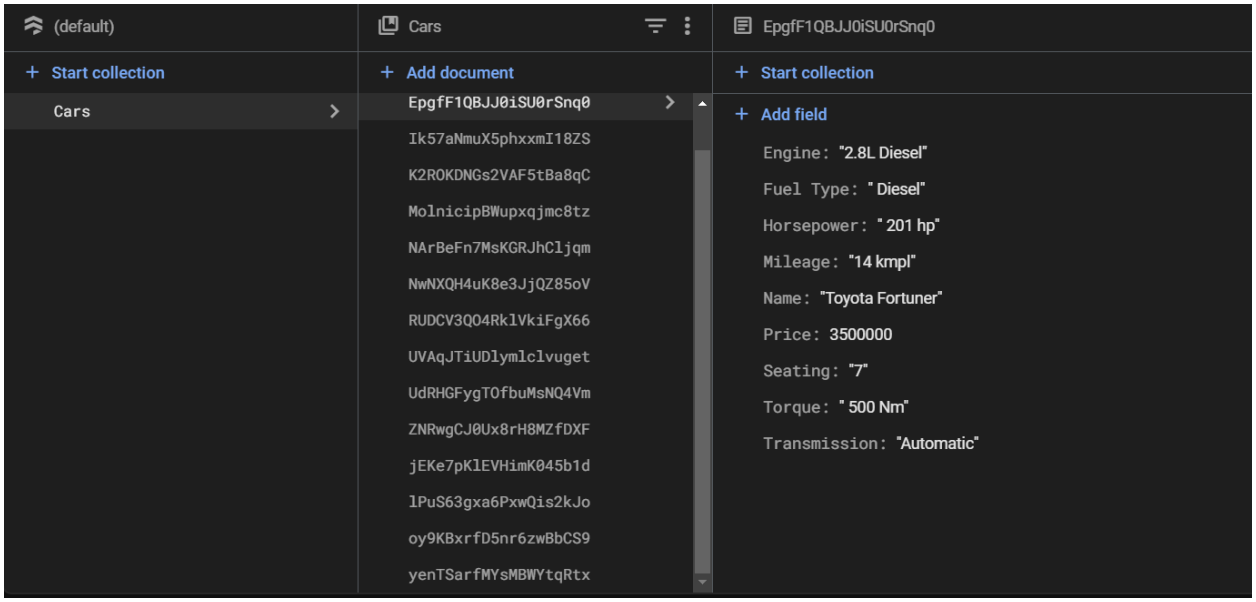
```
    GoogleSignInAccount? googleUser = await googleSignIn.signIn();
    if (googleUser == null) return;

    final GoogleSignInAuthentication googleAuth =
        await googleUser.authentication;

    final AuthCredential credential = GoogleAuthProvider.credential(
      accessToken: googleAuth.accessToken,
      idToken: googleAuth.idToken,
    );

    await _auth.signInWithCredential(credential);
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Google Sign-In Successful!')),
    );
  } catch (e) {
    print("Google Sign-In Error: $e");
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Google Sign-In Failed: $e')),
    );
  }
}

void _signInWithEmailAndPassword() async {
  if (_formKey.currentState!.validate()) {
    try {
      await _auth.signInWithEmailAndPassword(
        email: _emailController.text.trim(),
        password: _passwordController.text.trim(),
      );
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Login Successful!')),
      );

      // Redirect to Home Page after login
      Navigator.pushReplacementNamed(context, '/home');
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Login Failed: $e')),
      );
    }
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
```

```
    body: Container(
      color: Colors.black,
      child: Center(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Card(
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(20),
            ),
            elevation: 10,
            color: Colors.white,
            child: Padding(
              padding: const EdgeInsets.all(30.0),
              child: Form(
                key: _formKey,
                child: Column(
                  mainAxisSize: MainAxisSize.min,
                  crossAxisAlignment: CrossAxisAlignment.stretch,
                  children: [
                    Column(
                      children: [
                        Image.asset(
                          'assets/c1.png', // Ensure image is in assets folder
                          height: 150,
                        ),
                        const SizedBox(height: 10),
                        Text(
                          'CarConnect',
                          style: GoogleFonts.alegreyaSans(
                            fontSize: 40,
                            fontWeight: FontWeight.bold,
                            color: Colors.black,
                          ),
                        ),
                      ],
                    ),
                    const SizedBox(height: 20),
                    TextFormField(
                      controller: _emailController,
                      decoration: InputDecoration(
                        labelText: 'Email',
                        border: OutlineInputBorder(
                          borderRadius: BorderRadius.circular(12),
                        ),
                        prefixIcon:
                            const Icon(Icons.email, color: Colors.black),
                      ),
```

```
      validator: (value) {
        if (value == null ||
            value.isEmpty ||
            !value.contains('@')) {
          return 'Enter a valid email';
        }
        return null;
      },
    ),
    const SizedBox(height: 15),
    TextFormField(
      controller: _passwordController,
      obscureText: true,
      decoration: InputDecoration(
        labelText: 'Password',
        border: OutlineInputBorder(
          borderRadius: BorderRadius.circular(12),
        ),
        prefixIcon:
            const Icon(Icons.lock, color: Colors.black),
      ),
      validator: (value) {
        if (value == null || value.length < 6) {
          return 'Password must be at least 6 characters';
        }
        return null;
      },
    ),
    const SizedBox(height: 20),
    ElevatedButton(
      style: ElevatedButton.styleFrom(
        padding: const EdgeInsets.symmetric(vertical: 14),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(12),
        ),
        backgroundColor: Colors.black,
      ),
      onPressed: _signInWithEmailAndPassword,
      child: const Text(
        'Login',
        style: TextStyle(fontSize: 20, color: Colors.white),
      ),
    ),
    const SizedBox(height: 15),
    ElevatedButton(
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.white,
```

```
                        padding: const EdgeInsets.symmetric(vertical: 14),
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(12),
                          side: const BorderSide(color: Colors.black),
                        ),
                      ),
                      onPressed: _signInWithGoogle,
                      child: const Text(
                        'Sign in with Google',
                        style: TextStyle(fontSize: 20, color: Colors.black),
                      ),
                    ),
                  ],
                ),
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```




(default)
Start collection
Cars
Cars
Add document
EpgfF1QBJJ0iSU0rSnq0
Ik57aNmuX5phxxmI18ZS
K2ROKDNGs2VAF5tBa8qC
MolnicipBWupxqjmc8tz
NArBeFn7MsKGRJhCljqm
NwNXQH4uK8e3JjQZ85oV
RUDCV3QO4RklVkiFgX66
UVAqJTiUDlymlclvuget
UdRHGFygTOfbuMsNQ4Vm
ZNRwgCJ0Ux8rH8MZfDXF
jEKe7pKlEVHimK045b1d
lPuS63gxa6PxwQis2kJo
oy9KBxrfD5nr6zwBbCS9
yenTSarfMYsMBWYtqRtx
EpgfF1QBJJ0iSU0rSnq0
Start collection
Add field
Engine: "2.8L Diesel"
Fuel Type: " Diesel"
Horsepower: " 201 hp"
Mileage: "14 kmpl"
Name: "Toyota Fortuner"
Price: 3500000
Seating: "7"
Torque: " 500 Nm"
Transmission: "Automatic"

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.	36
Name	MOHIT PATIL
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

PWA Experiment -7

Mohit patil 36/D15A

❖ **AIM**

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

❖ **Theory**:-

Regular Web Application

A regular web application is a website designed to be accessible on various devices, ensuring that content adjusts dynamically to different screen sizes. Built using web technologies such as HTML, CSS, JavaScript, and Ruby, these applications function through a browser. While they can leverage some native device features, their functionality is dependent on browser compatibility. For instance, a feature may work on Google Chrome but not on Safari or Mozilla Firefox due to browser limitations.

Progressive Web Application (PWA)

A Progressive Web Application (PWA) is an advanced version of a regular web app, integrating additional features to enhance the user experience. PWAs combine the best elements of desktop and mobile applications, delivering a seamless experience across platforms.

❖ Key Differences Between PWAs and Regular Web Apps

1. Native-Like Experience

While both PWAs and regular web apps utilize standard web technologies like HTML, CSS, and JavaScript, PWAs offer a user experience similar to native mobile applications. PWAs can access device-specific features such as push notifications without relying on a particular browser, creating a smoother, more integrated experience.

2. Ease of Access

Unlike traditional mobile apps that require time-consuming downloads and storage space, PWAs can be installed directly via a URL. Users can add a PWA to their home screen with a simple link, eliminating installation complexities and ensuring easy access while keeping brand presence strong.

3. Enhanced Performance

PWAs utilize caching mechanisms to pre-load content, such as text, stylesheets, and images, allowing for faster loading times. This significantly improves user engagement and retention by reducing waiting periods, ultimately benefiting businesses by increasing interaction rates.

4. Improved User Engagement

PWAs efficiently leverage push notifications and native device features to keep users engaged. Unlike regular web apps, their functionality is not restricted by browser dependencies, enabling businesses to notify users about updates, offers, and promotions without disruptions.

5. Real-Time Updates

A major advantage of PWAs is their ability to update automatically without requiring users to download new versions from an app store. Developers can push updates directly from the server, ensuring that users always access the latest features and improvements instantly.

6. Search Engine Optimization (SEO) Benefits

Since PWAs function within web browsers, they can be indexed by search engines, improving their visibility in search results. This gives them a strategic advantage over native apps, which are limited to app store searches.

7. Cost-Effective Development

Unlike native mobile apps, PWAs do not require approval or submission to app stores, reducing development and maintenance costs.

❖ Advantages and Limitations of PWAs

➢ **Advantages:**

- **Progressive:** Compatible with all browsers and devices, following the principle of progressive enhancement.
- **Responsive:** Adapts to different screen sizes, including desktops, tablets, and smartphones.
- **App-Like Feel:** Mimics the experience of native applications in navigation and interaction.
- **Always Updated:** Service workers ensure real-time updates without requiring user intervention.
- **Secure:** Delivered over HTTPS, ensuring secure data transfer and protection against cyber threats.
- **SEO-Friendly:** Can be indexed by search engines, enhancing discoverability.
- **Re-Engagement:** Enables push notifications to encourage continued user interaction.
- **Installable:** Allows users to add the app to their home screen without app store downloads.
- **Offline Functionality:** Can function in low or no connectivity conditions using cached content.

➢ **Limitations:**

- **Higher Battery Consumption:** PWAs tend to consume more battery due to constant background processes.
- **Limited Hardware Access:** Some device features, such as advanced sensors and Bluetooth, may not be fully accessible.
- **Offline Mode Constraints:** Some offline capabilities remain limited, depending on browser support.
- **No App Store Presence:** PWAs cannot generate traffic from app store searches.
- **Lack of Centralized Control:** Unlike native apps, PWAs do not undergo an official approval process, potentially affecting credibility.

CODE:

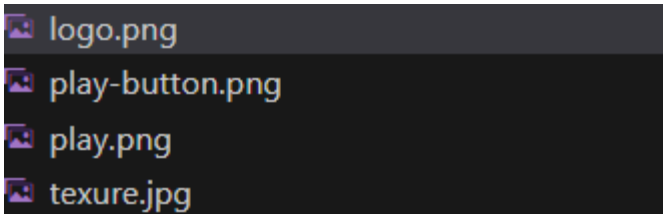
Index.html :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta name="description" content="Web site created using create-react-app" />

    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <link  rel="apple-touch-icon"  href="%PUBLIC_URL%/apple-touch-icon.png">
    <link rel="apple-touch-icon" sizes="192x192" href="%PUBLIC_URL%/logo.png">
    <link rel="apple-touch-icon" sizes="512x512" href="%PUBLIC_URL%/logo.png">

    <link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico" type="image/x-icon"
/>

    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.15.4/css/all.css"
        integrity="sha384-
DyZ88mC6Up2uqS4h/KRgHuoeGwBcD4Ng9SiP4dIRy0EXTlnuz47vAwmeGwVChigm
"
        crossorigin="anonymous" />

    <title>Streamo - Netflix</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

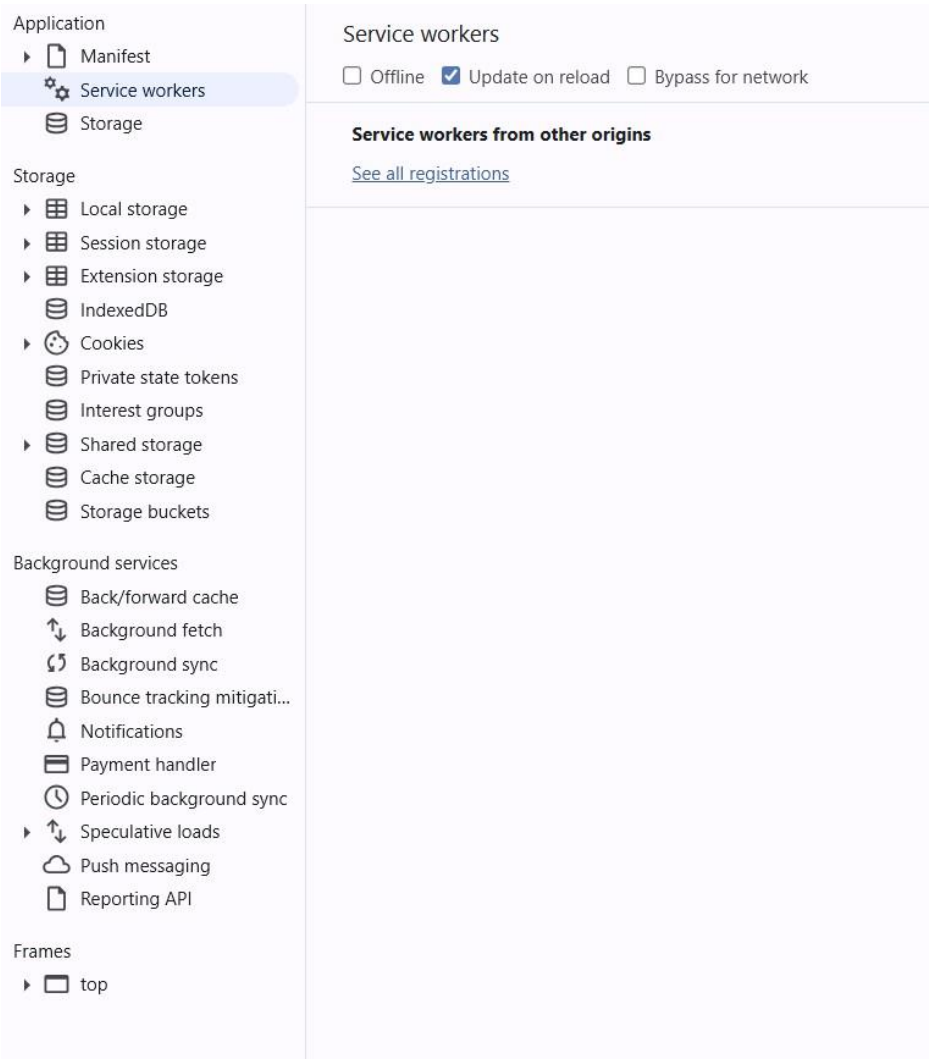
Manifest.json


```
{
  "name": "Streamo - Netflix",
  "short_name": "Streamo",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#000000",
  "theme_color": "#000000",
  "description": "Watch unlimited movies & TV shows.",
  "icons": [
    {
      "src": "/logo.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "/logo.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ]
}
```

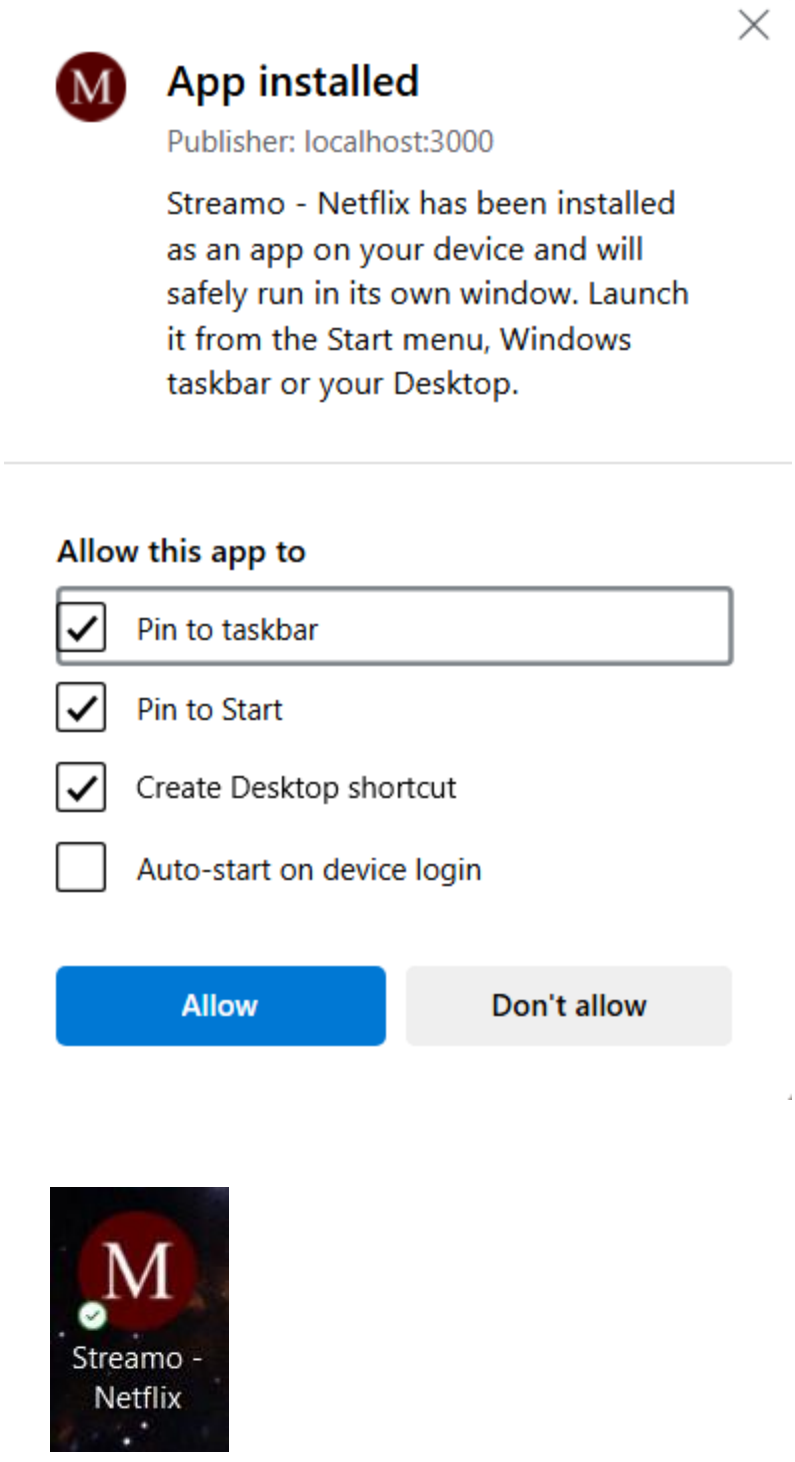
Icons



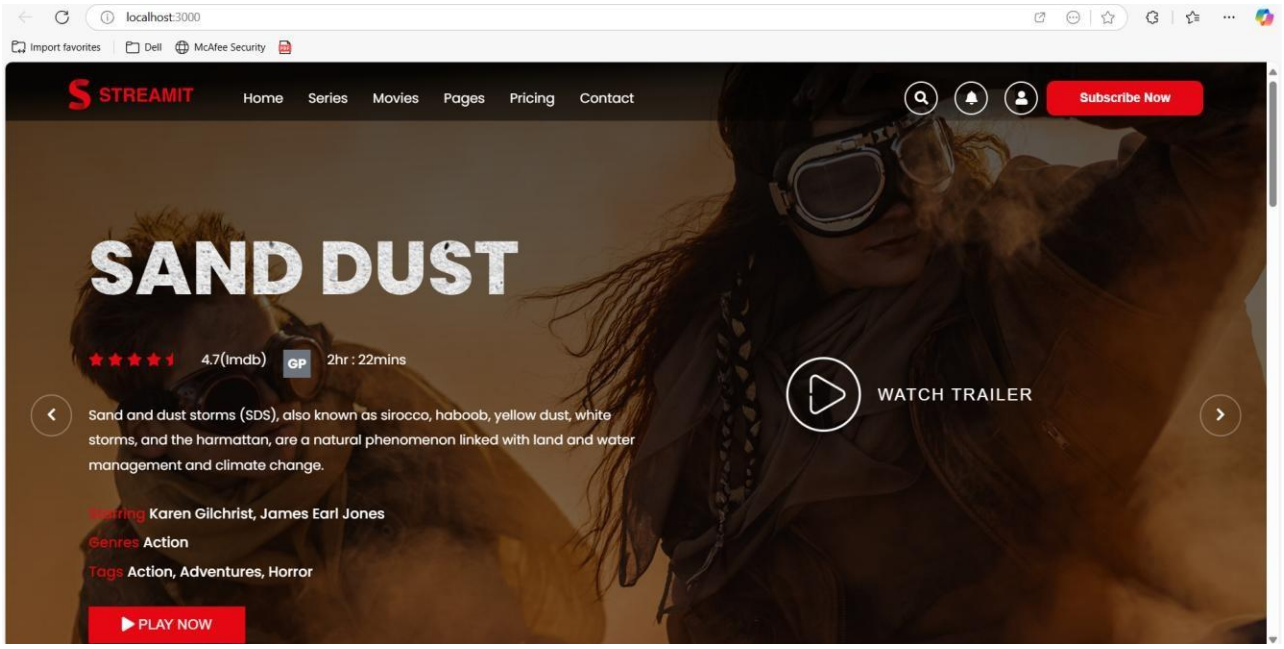
Google Dev

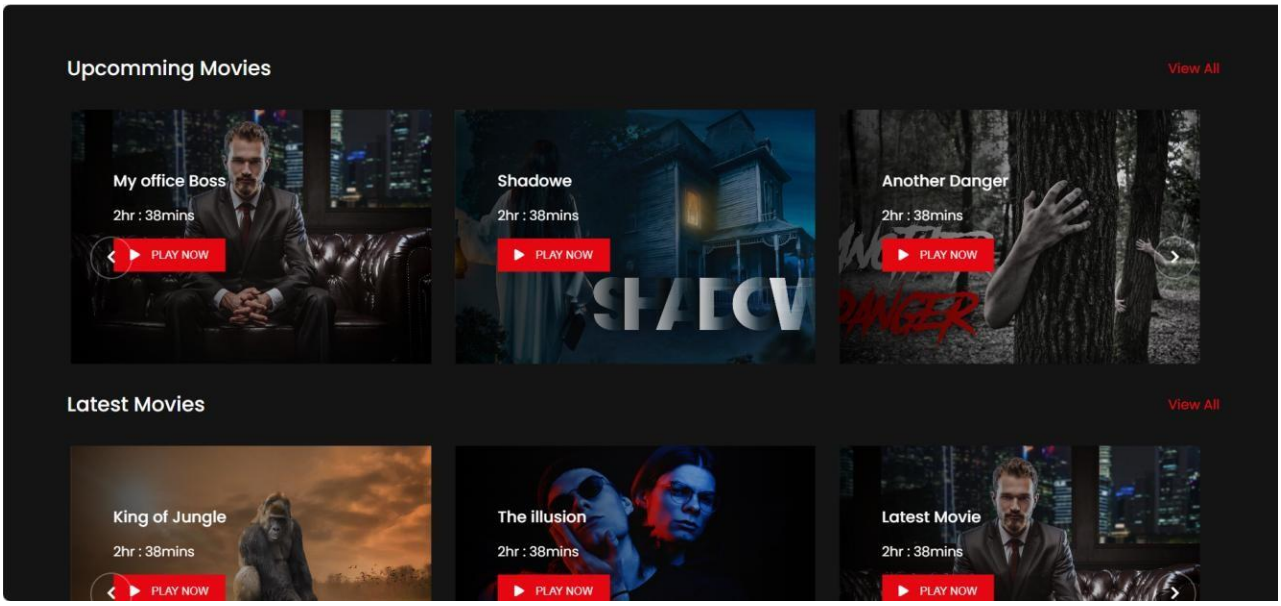


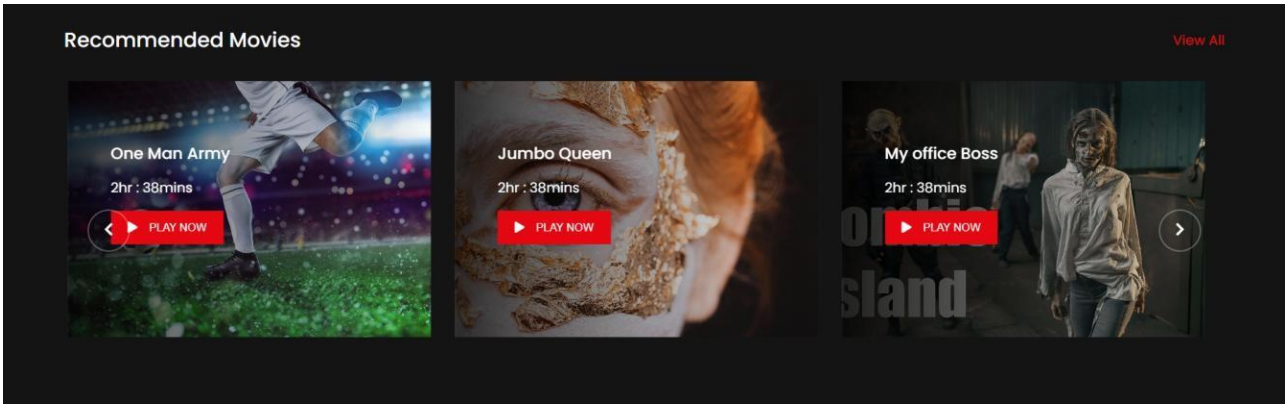
❖ **Output:-**


App installed
Publisher: localhost:3000
Streamo - Netflix has been installed as an app on your device and will safely run in its own window. Launch it from the Start menu, Windows taskbar or your Desktop.
Allow this app to
Pin to taskbar
Pin to Start
Create Desktop shortcut
Auto-start on device login
Allow
Don't allow

Streamo - Netflix


localhost:3000
Import favorites
Dell
McAfee Security
STREAMIT
Home
Series
Movies
Pages
Pricing
Contact
Subscribe Now
SAND DUST
4.7(Imdb)
GP
2hr : 22mins
Sand and dust storms (SDS), also known as sirocco, haboob, yellow dust, white storms, and the harmattan, are a natural phenomenon linked with land and water management and climate change.
Karen Gilchrist, James Earl Jones
Action
Action, Adventures, Horror
PLAY NOW
WATCH TRAILER


Upcomming Movies
View All
My office Boss
2hr : 38mins
PLAY NOW
Shadowe
2hr : 38mins
PLAY NOW
Another Danger
2hr : 38mins
PLAY NOW
Latest Movies
View All
King of Jungle
2hr : 38mins
PLAY NOW
The illusion
2hr : 38mins
PLAY NOW
Latest Movie
2hr : 38mins
PLAY NOW


Recommended Movies
View All
One Man Army
2hr : 38mins
PLAY NOW
Jumbo Queen
2hr : 38mins
PLAY NOW
My office Boss
2hr : 38mins
PLAY NOW

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.	36
Name	MOHIT PATIL
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)
```

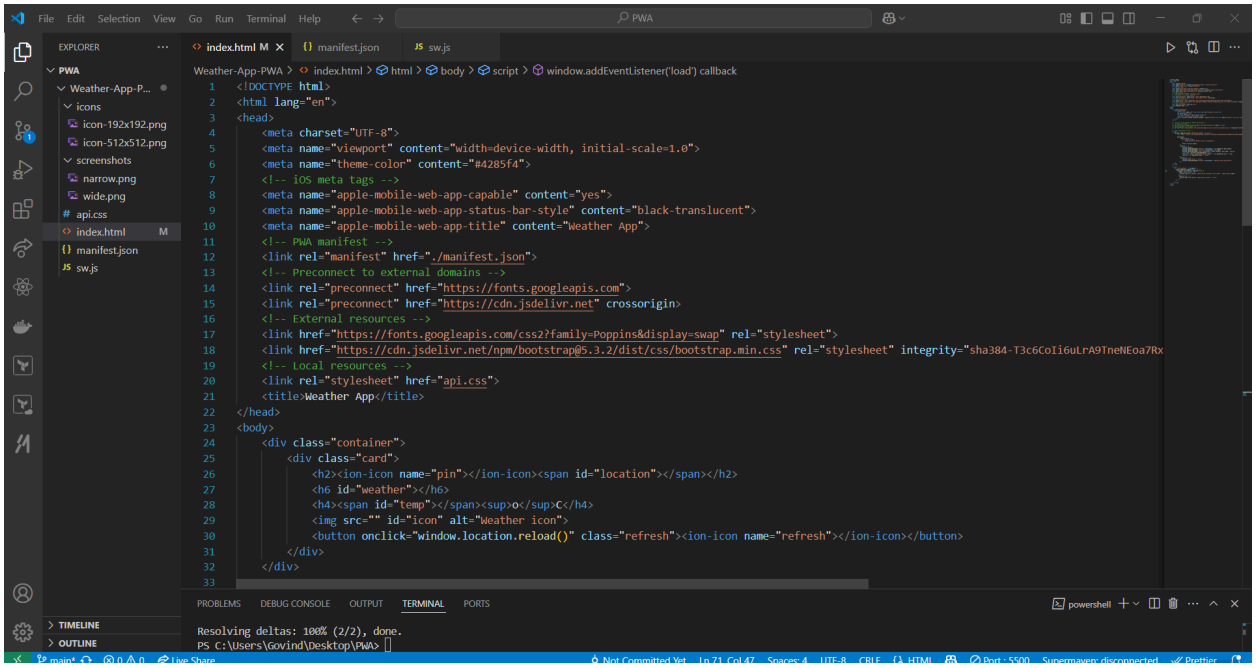
```
if (url.pathname.startsWith('/api/')) {
  event.respondWith(
    fetch(request)
      .then(networkResponse => {
        // Cache successful API responses
        if (networkResponse.ok) {
          const clone = networkResponse.clone();
          caches.open(API_CACHE)
            .then(cache => 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)
  );
});
```

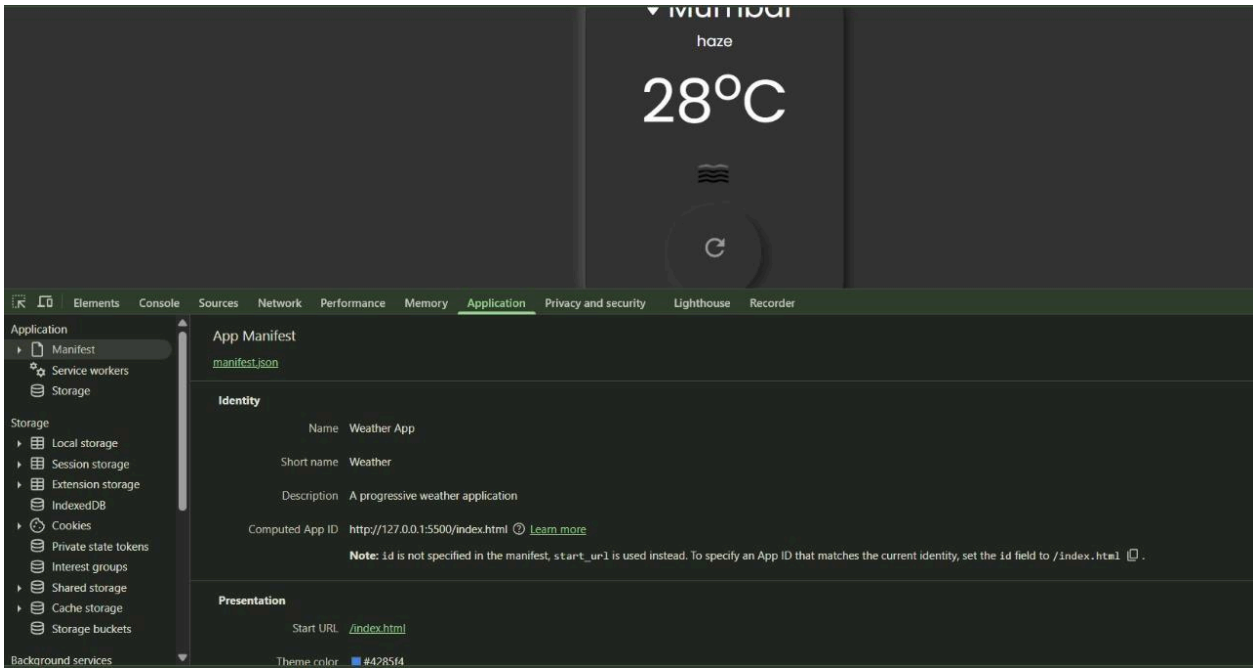
```
self.addEventListener('notificationclick', (event) => {
  event.notification.close();
  event.waitUntil(
    clients.matchAll({ type: 'window' })
      .then(clientList => {
        for (const client of clientList) {
          if (client.url === event.notification.data.url && 'focus' in
client) {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

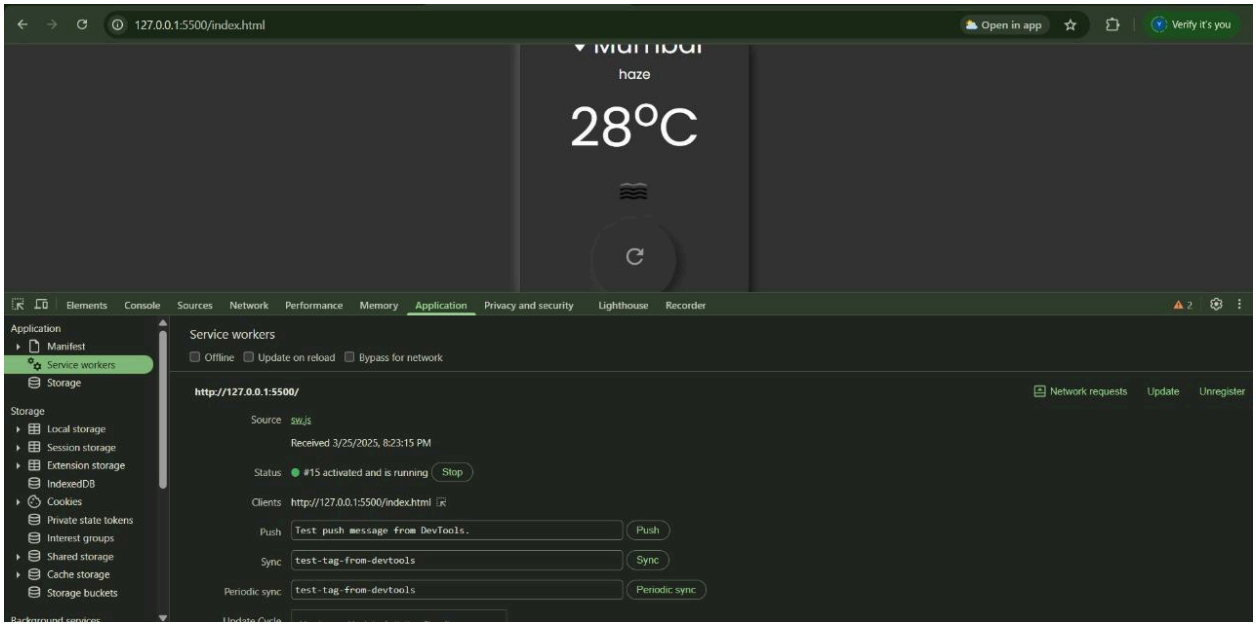
Output:


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.	36
Name	MOHIT PATIL
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: Mohit patil **Class:** D15A **Roll no:** 36

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') {
```

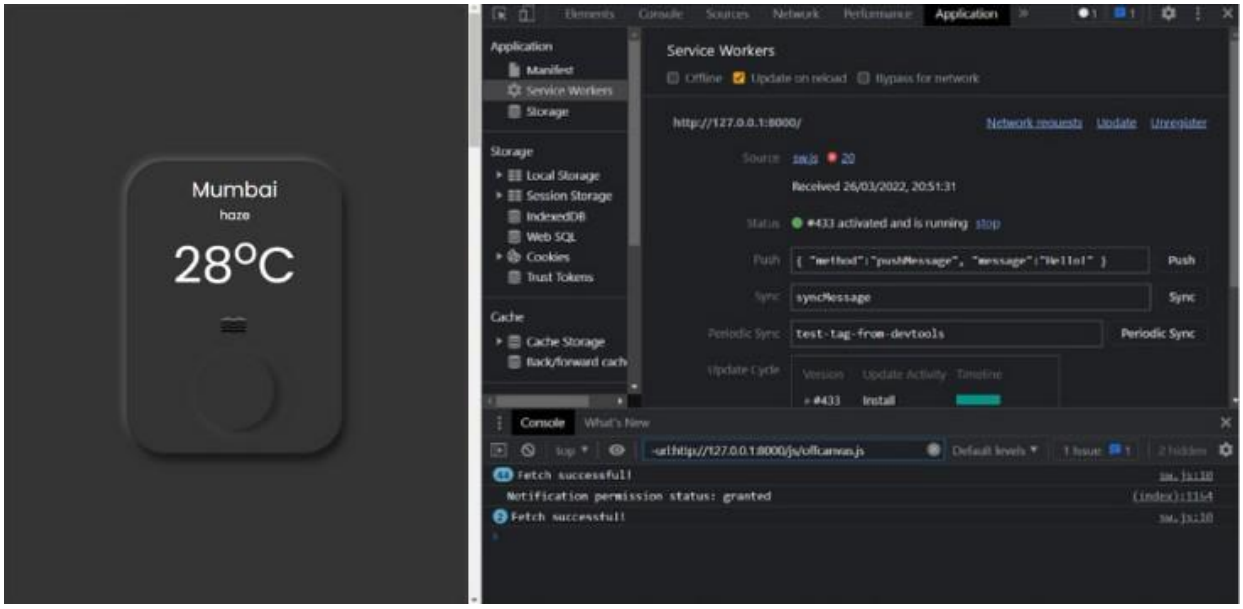
```
    event.waitUntil(
      // Get cart data from IndexedDB
      getCartData()
        .then(cartItems => {
          return fetch('/api/cart/sync', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(cartItems)
          });
        })
        .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([]);
```

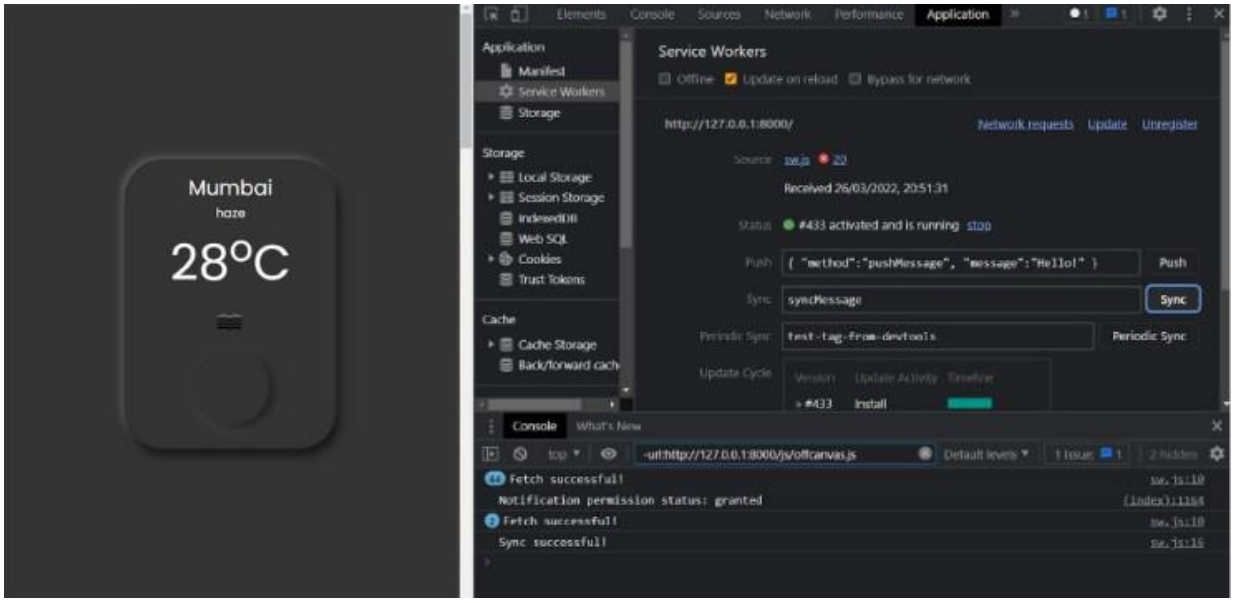
```
  });
}

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.	36
Name	MOHIT PATIL
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: Mohit Patil **Class:** D15A **Roll no:** 36

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

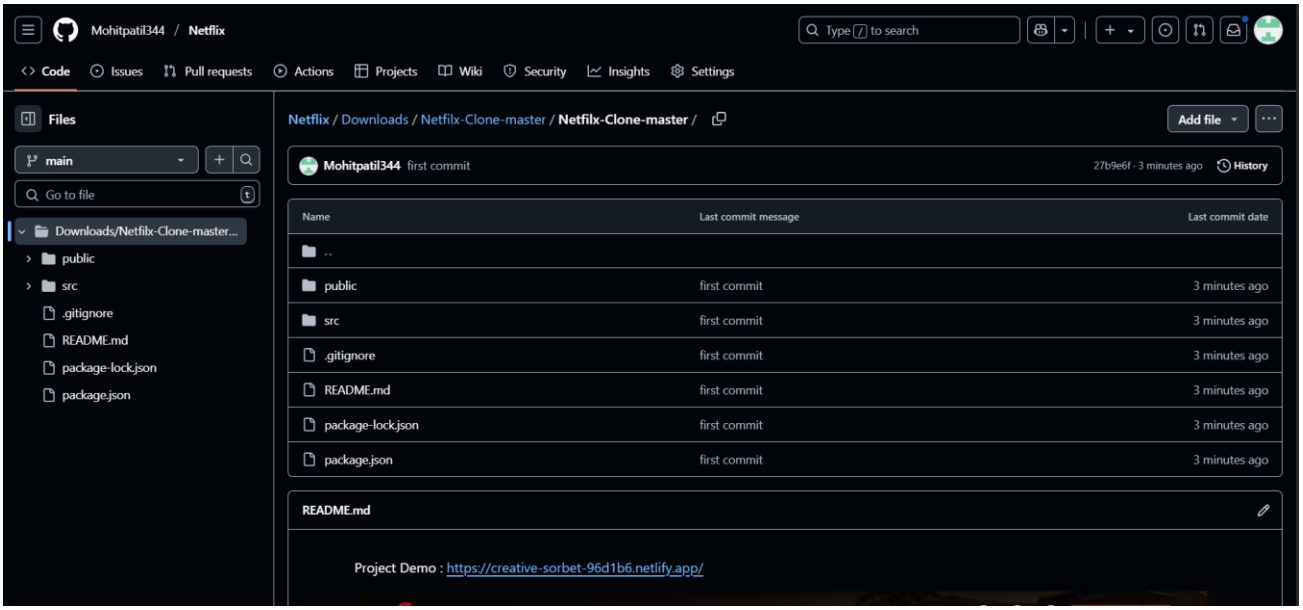
- Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
- Command-line interface only.
- No in-built support for any static site generator.

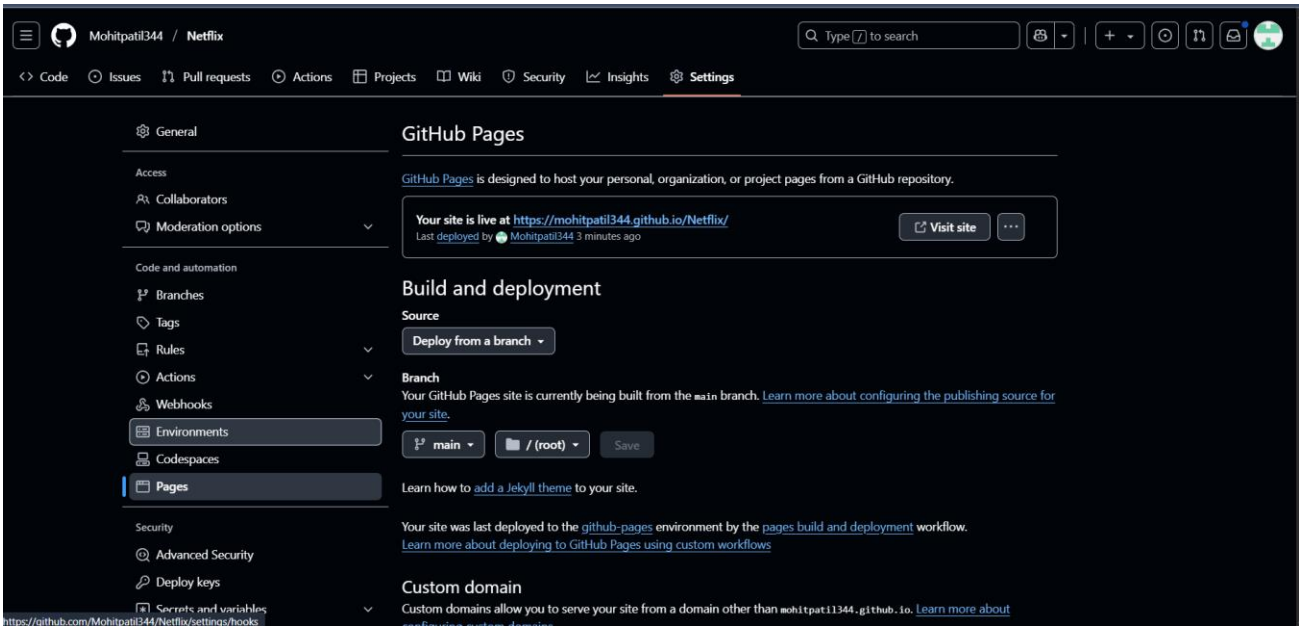
Link to our GitHub repository:

https://github.com/Mohitpatil344/Netflix

Link to our Hosted website:
https://creative-sorbet-96d1b6.netlify.app/

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.	36
Name	MOHIT PATIL
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

PWA Experiment -11

Mohit Patil 36/D15A

❖ Aim:

To use Google Lighthouse PWA Analysis Tool to test the PWA functioning.

❖ Theory:

Google Lighthouse: Overview

Google Lighthouse is an open-source automated tool developed by Google to audit web applications based on multiple parameters, including performance, accessibility, SEO, best practices, and Progressive Web App (PWA) implementations. It provides an in-depth analysis of a webpage by running different tests and generating a detailed report highlighting areas for improvement.

Lighthouse can be executed via **Chrome DevTools**, **Node.js command line**, or as a **browser extension**. It helps developers optimize their applications to enhance user experience, improve mobile responsiveness, and ensure compliance with best web development practices.

Key Features of Google Lighthouse

Lighthouse audits web pages for both **desktop** and **mobile** versions. The key metrics analyzed during an audit are as follows:

- **Performance:**

This metric evaluates how fast a webpage loads and becomes interactive for users. The score is based on various factors, including:

- **First Contentful Paint (FCP):** Measures the time taken to render the first visible content.
- **Largest Contentful Paint (LCP):** Measures the time taken for the largest visible element to load.
- **Time to Interactive (TTI):** Measures how long the page takes to become fully interactive.
- **Speed Index:** Indicates how quickly content is visually displayed.
- **Total Blocking Time (TBT):** Calculates the time a page remains unresponsive due to heavy JavaScript execution.

A **high performance score (closer to 100)** means the website loads quickly and delivers a smooth user experience.

- **Progressive Web App (PWA) Analysis:**

Google Lighthouse checks whether a web application follows the **Baseline PWA Checklist** set by Google. It verifies essential PWA components like:

- **Service Workers:** Ensuring offline functionality and background synchronization.
- **Web App Manifest:** Proper implementation of manifest.json for home screen installation.
- **Viewport Handling:** Ensuring mobile-friendliness with <meta name="viewport">.
- **HTTPS:** Ensuring a secure connection for user safety.
- **Responsive Design:** Optimizing layout and content for different screen sizes.
- **Offline Support:** Verifying if key resources are cached to enable offline access.

A high **PWA score** ensures that the application provides an **app-like experience** on mobile devices.

- **Accessibility:**

Accessibility measures how well a web page supports users with disabilities, including those using screen readers and assistive technologies. Lighthouse evaluates accessibility based on:

- **ARIA Attributes:** Proper use of aria-label, aria-required, etc. for better screen reader support.
- **Text Contrast:** Ensuring readable text against the background color.
- **Keyboard Navigation:** Ensuring all elements are accessible via keyboard (no mouse required).
- **Form Labels:** Ensuring form fields have proper labels and descriptions.
- **Alt Text for Images:** Checking if images have alt attributes for visually impaired users.

Accessibility scores are calculated based on pass/fail criteria. A **low score** means that the website is not user-friendly for people with disabilities.

- **Best Practices:**

Lighthouse evaluates whether a web application follows industry-recommended best practices to ensure security, efficiency, and maintainability. It checks for:

- **Use of HTTPS:** Ensuring a secure connection.
- **Deprecated Code:** Identifying outdated HTML tags, CSS styles, and JavaScript APIs.
- **Password Protection:** Verifying that users can securely input passwords (e.g., disabling "paste" for password fields).
- **Safe JavaScript Execution:** Identifying possible security risks and performance issues in JavaScript code.
- **Geo-Location and Cookie Alerts:** Ensuring compliance with privacy regulations like GDPR by displaying necessary permission prompts.

A **high Best Practices score** ensures the website is built using modern, secure, and efficient coding techniques.

- **SEO (Search Engine Optimization):**

SEO audits help determine how well a webpage is optimized for search engines. Lighthouse checks:

- **Meta Tags:** Ensuring title and meta description are properly set.
- **Mobile-Friendliness:** Verifying that the website is optimized for mobile devices.
- **Canonical URLs:** Preventing duplicate content issues.
- **Crawlability:** Ensuring search engines can index the website properly.

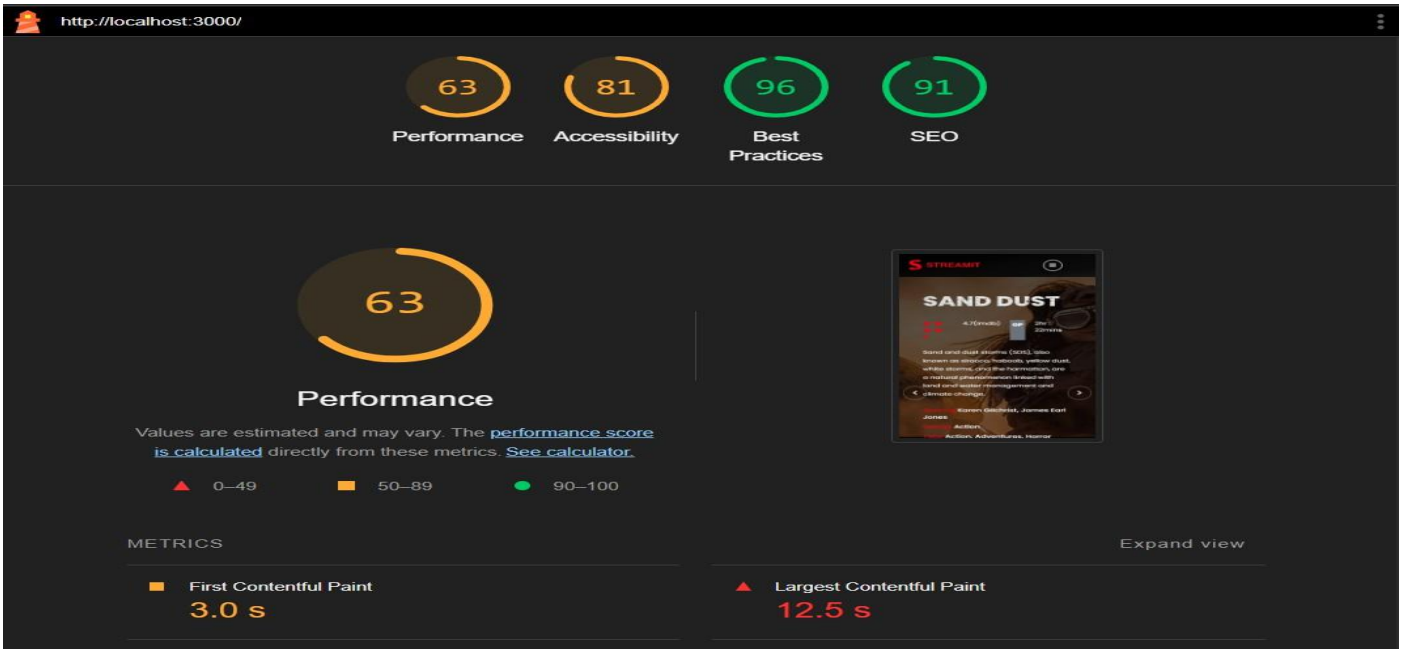
A high **SEO score** improves a website's ranking on search engines like Google.

manifest.json

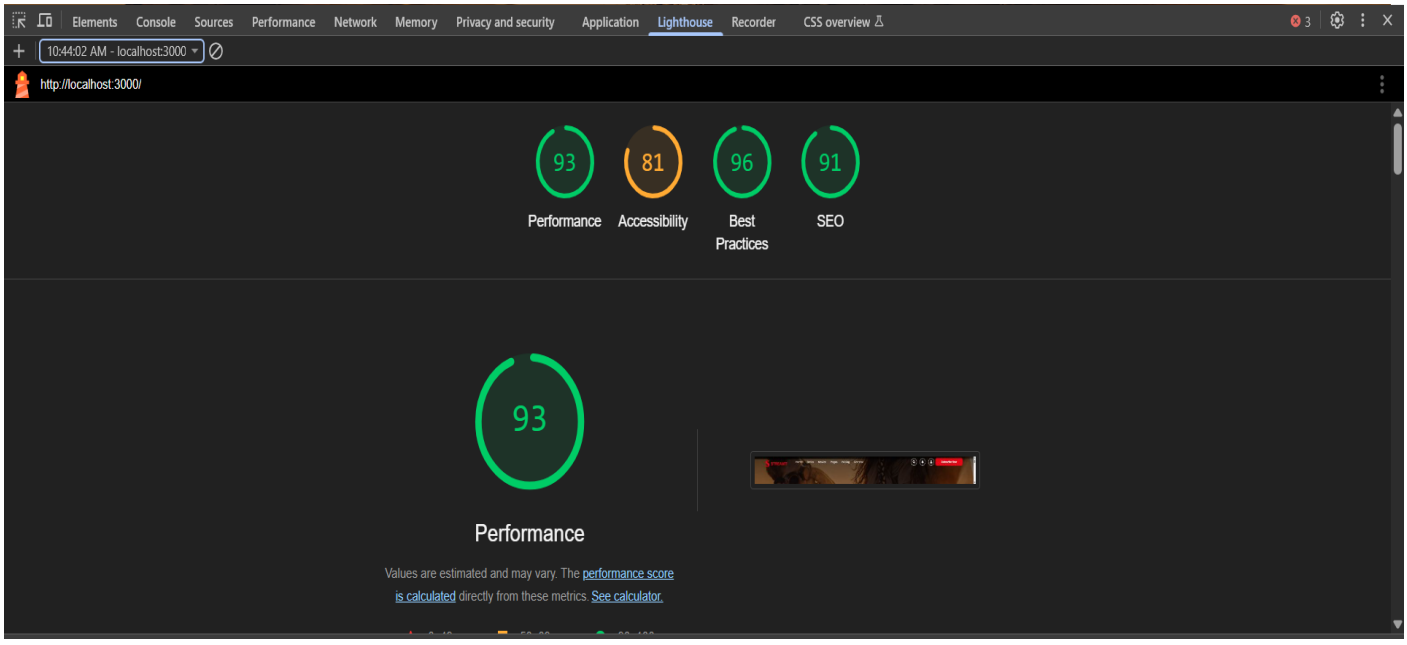
```
{
  "name": "Streamo - Netflix",
  "short_name": "Streamo",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#000000",
  "theme_color": "#000000",
  "description": "Watch unlimited movies & TV shows.",
  "icons": [
    {
      "src": "/logo.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "/logo.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ]
```

❖ Output

- Before Code change



- After code change



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.	36
Name	MOHIT PATIL
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:	

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.	36
Name	MOHIT PATIL
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:	