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 **DIKSHA ABHINAY UTEKAR** of **D15A** semester **VI,** have successfully completed necessary experiments in the **MAD & PWA Lab** under my supervision in **VES Institute of Technology** during the academic year **2024-2025.**

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Name of the course: MAD & PWA Lab **Course code:** ITL604

Project Title: Snapchat **Roll No. 63**

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

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

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers **:** Mrs. Kajal Joseph.

Email **:**

kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

PO10) Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.

PO11) Project Management and Finance: Demonstrate knowledge and understanding of computer engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12) Life-long Learning: Recognize the need for and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

Program specific Outcomes

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

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

Project Title: Snapchat **Roll No. 63**

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	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

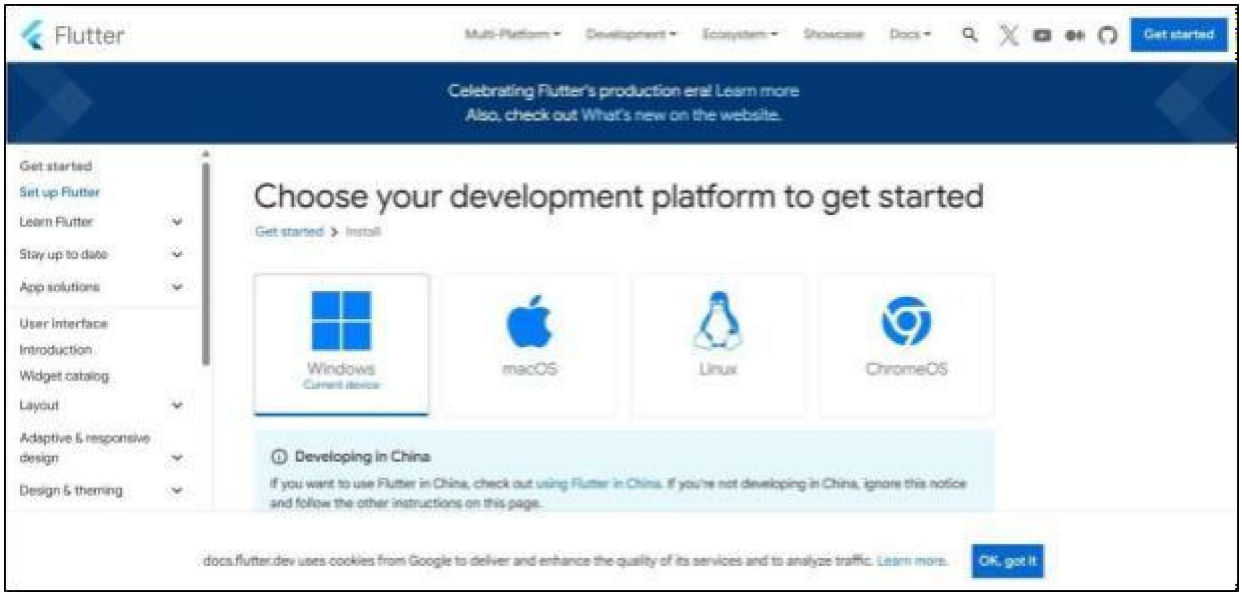
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	**63**
Name	**Diksha Abhinay Utekar**
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

EXPERIMENT NO: - 01

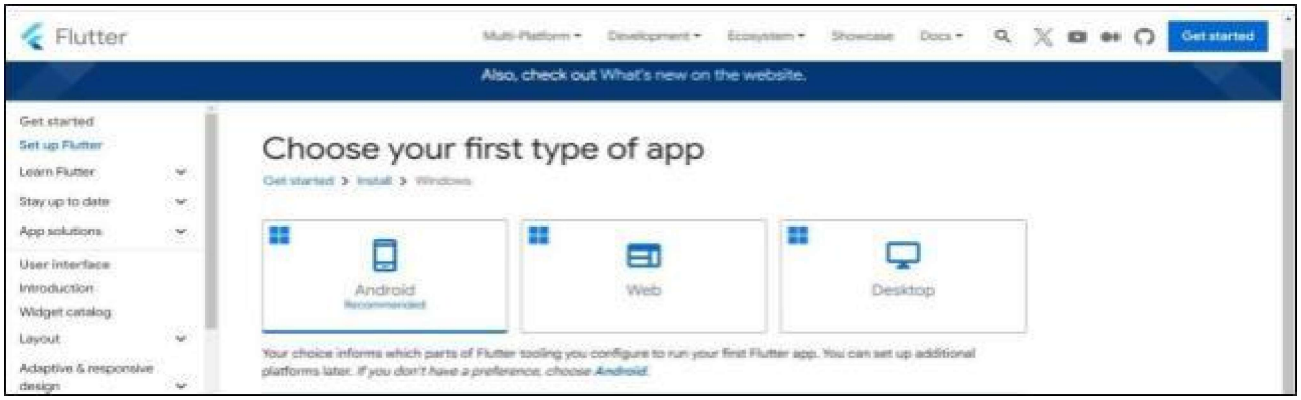
Name:- Diksha Utekar **Class:-** D15A **Roll:No: -** 63

AIM: - Installation and Configuration of Flutter Environment.

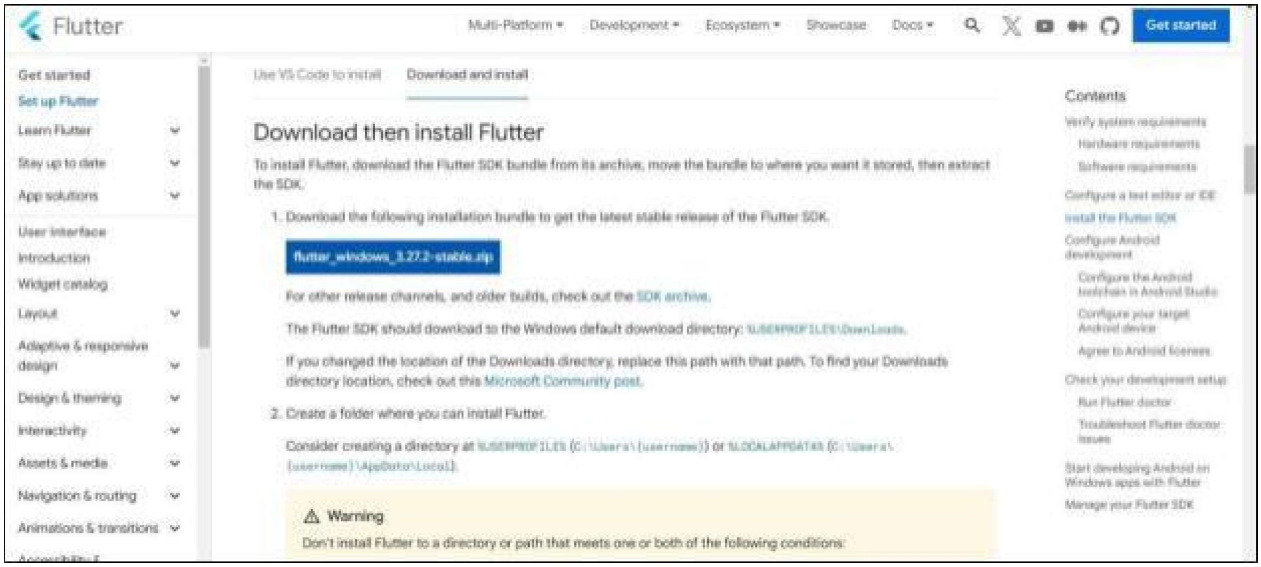
Step 1: Go to the official Flutter website: https://docs.flutter.dev/get-started/install



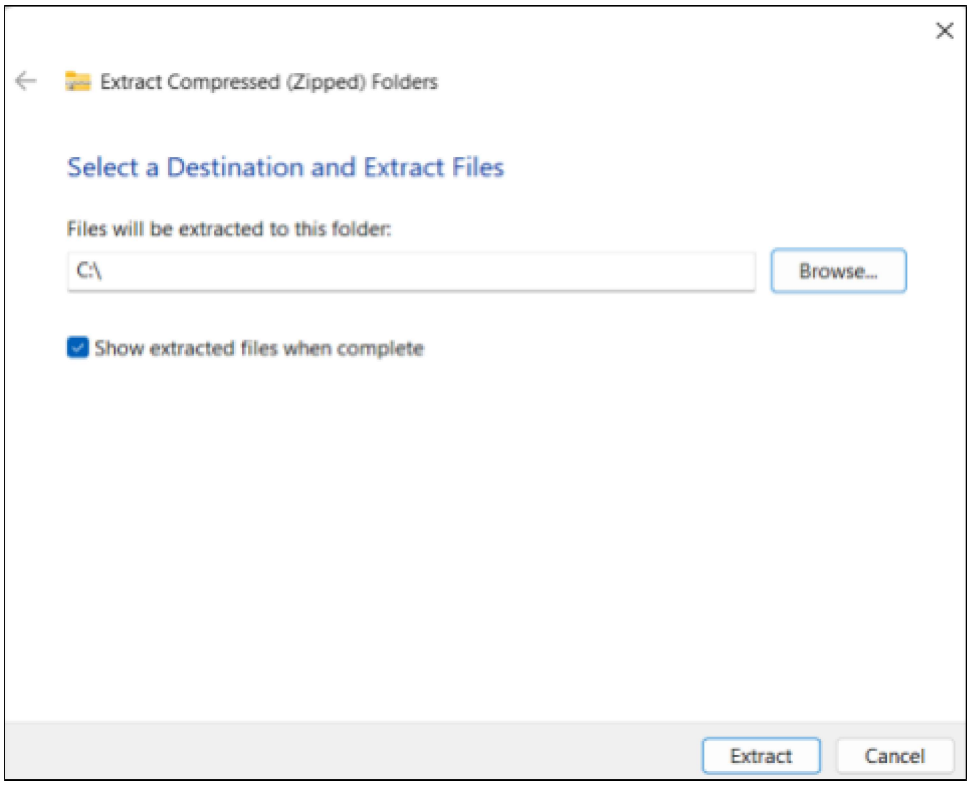
Step 2: To download the latest Flutter SDK, click on the Windows icon > Android



Step 3: For Windows, download the stable release (a .zip file).



Step 4: Extract the ZIP file to a folder (e.g., C:\flutter).

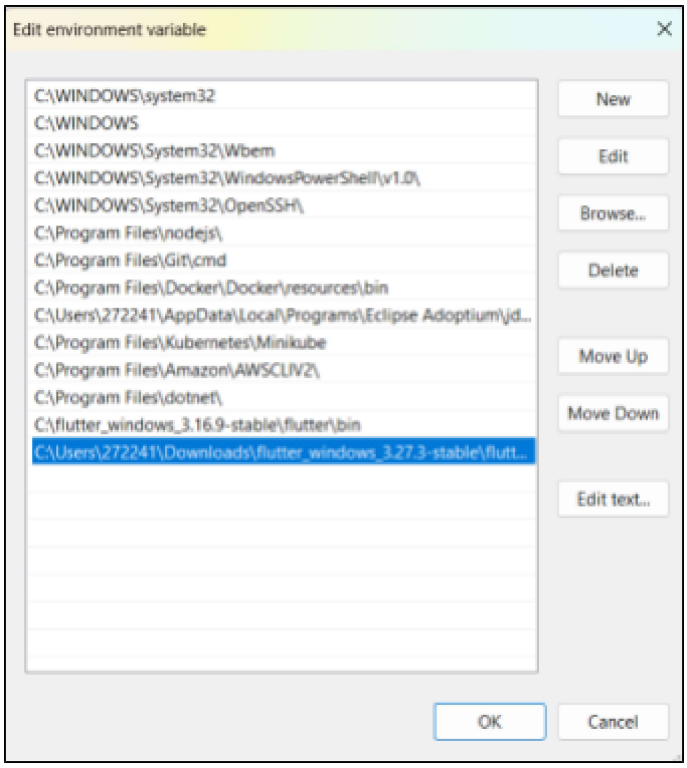


Step 5 :- Add Flutter to System PATH

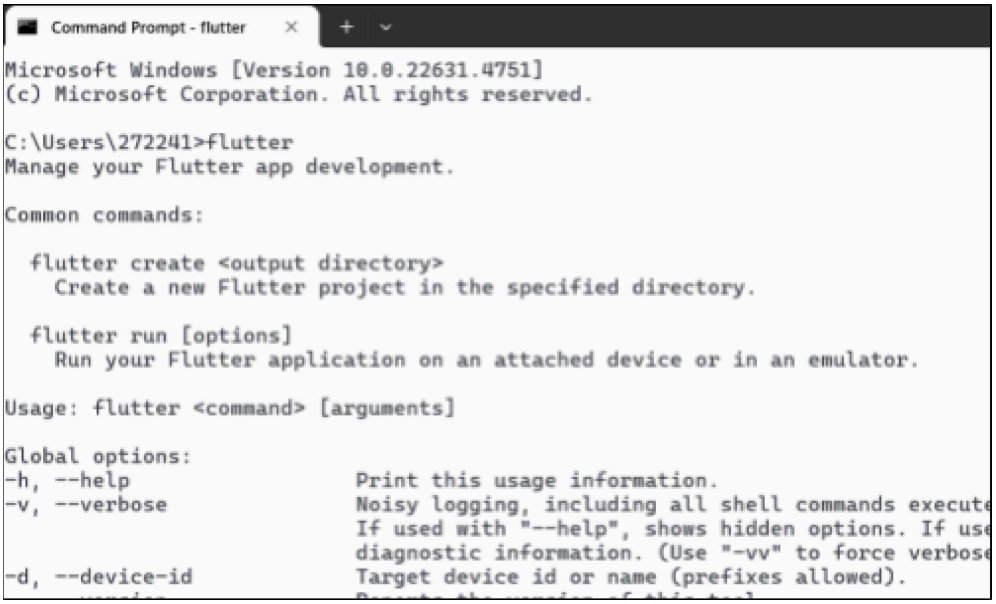
Right-click on the Start Menu > System > Advanced system settings > Environment Variables.

Under System Variables, find Path and click Edit.

Add the full path to the flutter/bin directory (e.g., C:\flutter\bin).



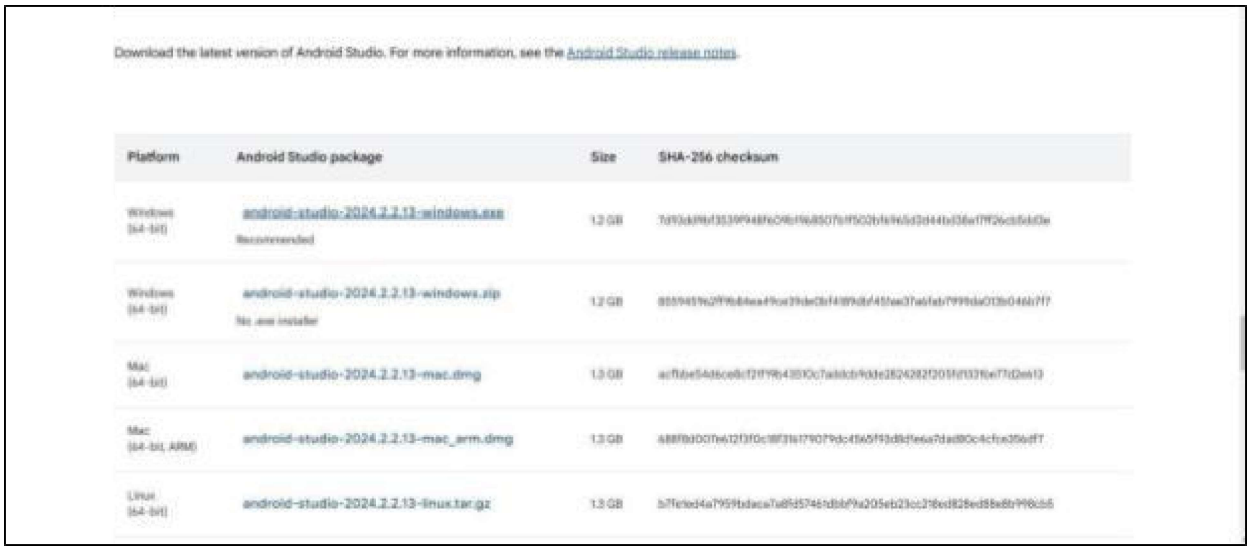
Step 6 : - Now, run the $ flutter command in command prompt.



Step 7:- Run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation

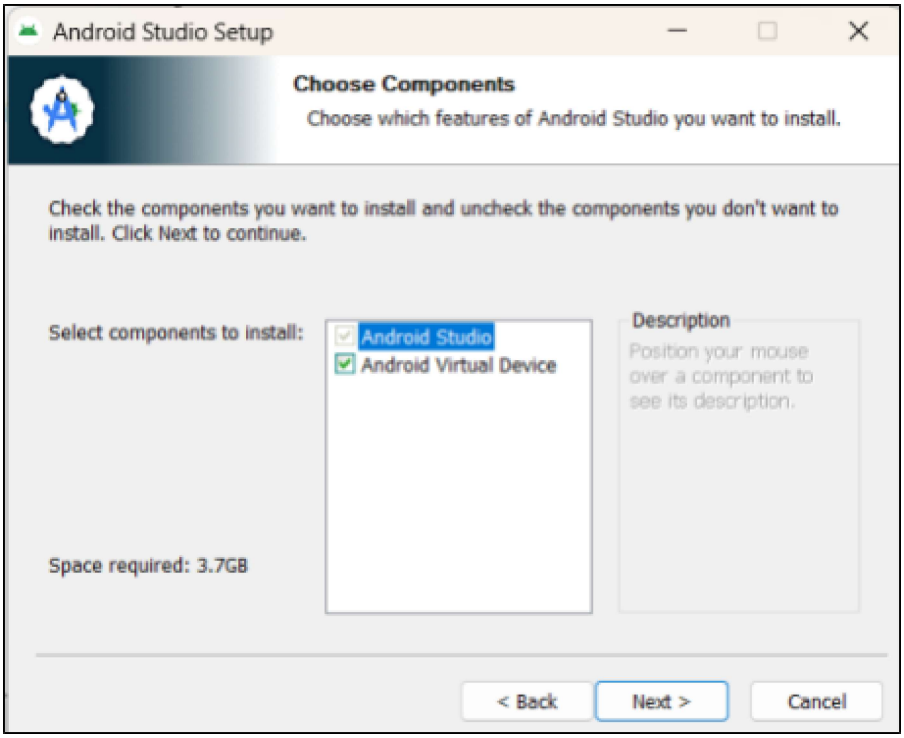
```
C:\Users\272241>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.22631.4751], locale e
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[✓] Chrome - develop for the web
[!] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.12.4)
    X Visual Studio is missing necessary components. Please re-run the Visual Studio installe
      development with C++" workload, and include these components:
        MSVC v142 - VS 2019 C++ x64/x86 build tools
          - If there are multiple build tool versions available, install the latest
        C++ CMake tools for Windows
        Windows 10 SDK
[✓] Android Studio (version 2023.1)
[✓] VS Code (version 1.96.4)
[✓] Connected device (3 available)
[✓] Network resources

! Doctor found issues in 1 category.
```

Step 8 : - Go to Android Studio and download the installer.

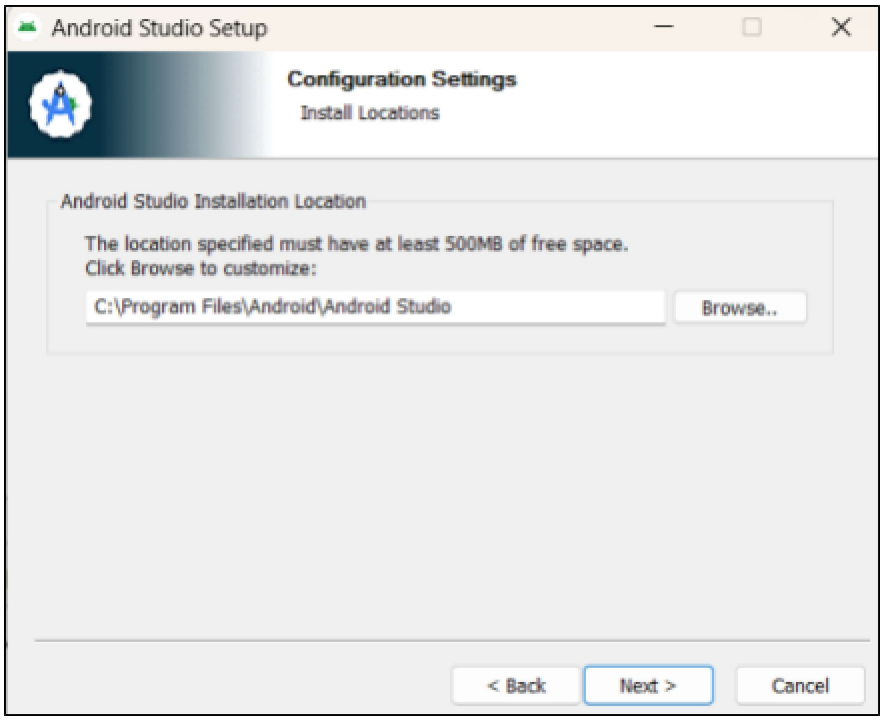


Step 8.1: - When the download is complete, open the .exe file and run it. You will get the following dialog box

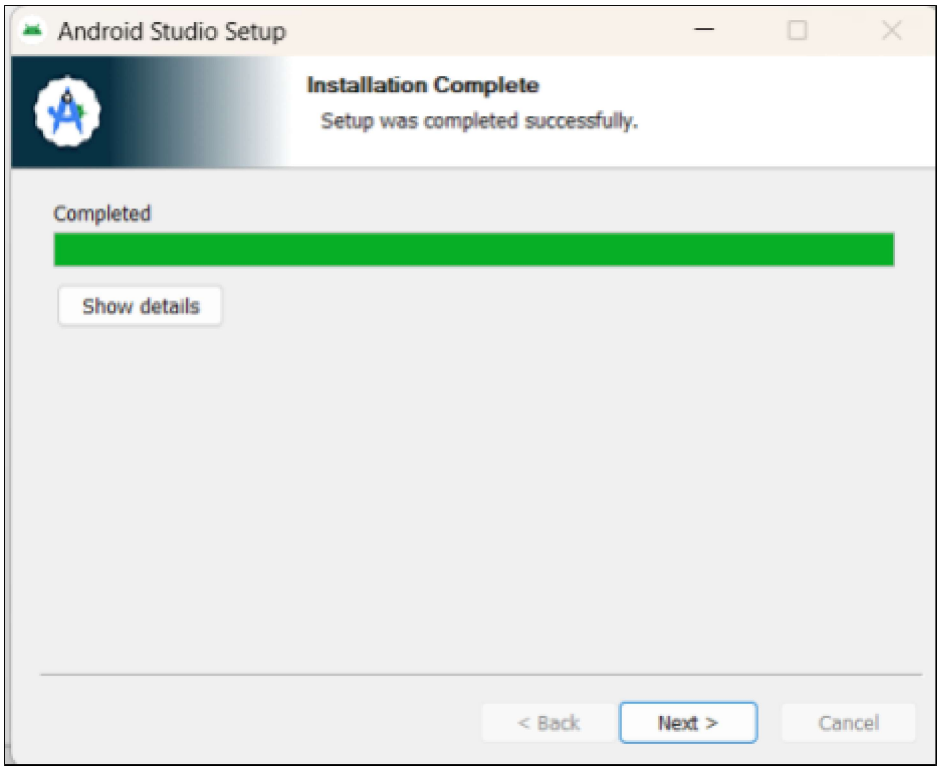
Step 8.2: - Select all the Checkboxes and Click on 'Next' Button.

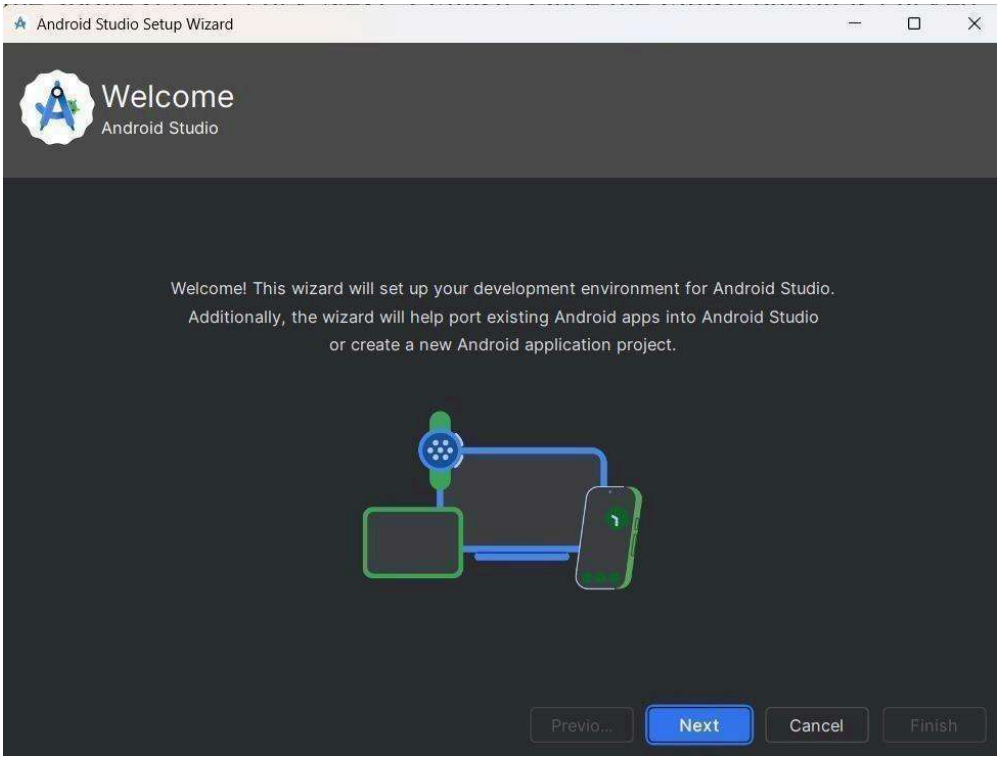


Step 8.3: - Change the destination as per your convenience and click on ‘Next’ Button.

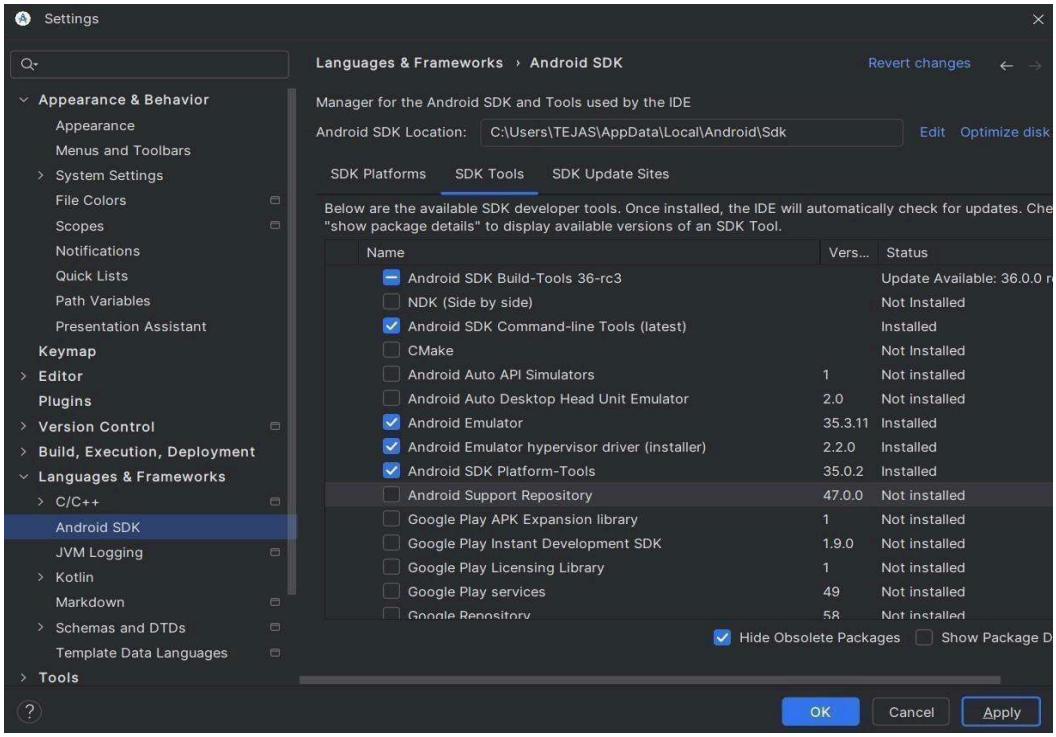


Step 8.4: - Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.





Step 8.5: - Go to Preferences > Appearance & Behavior > System Settings > Android SDK. Select the SDK Tools tab and check Android SDK Command-line Tools and Install it.

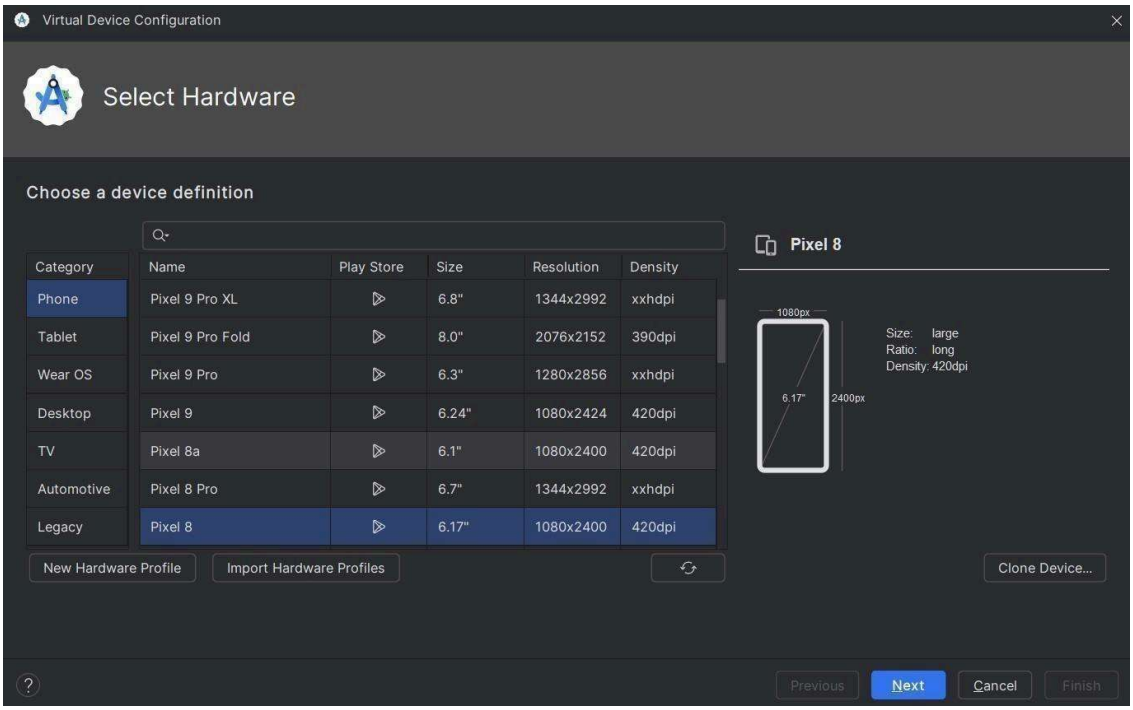


Step 9: - Open a terminal and run the following command

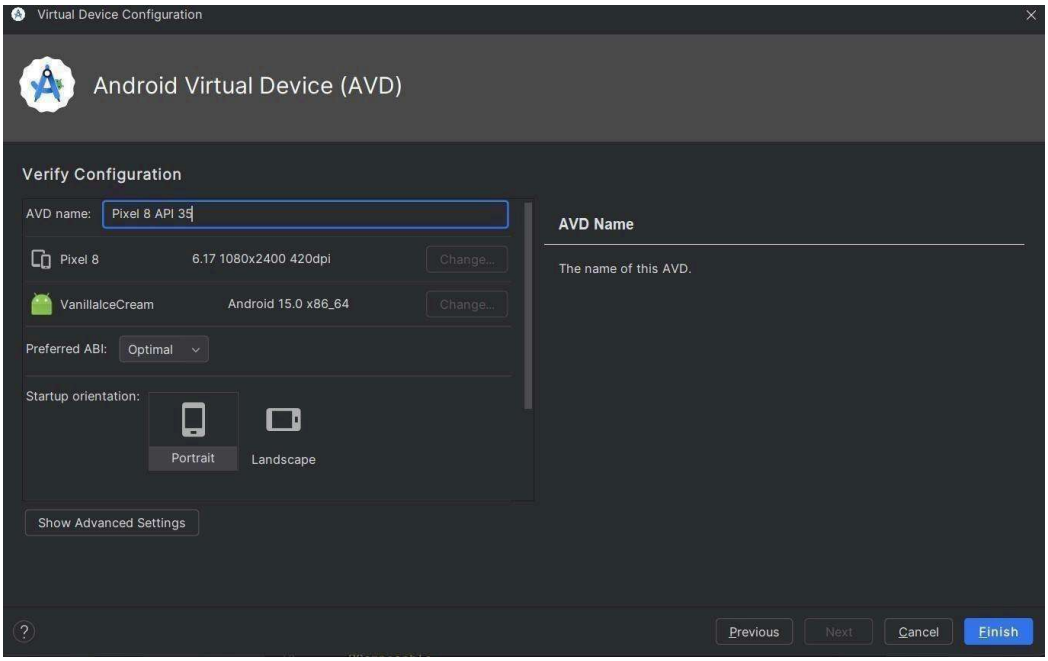
```
C:\Users\272241>flutter doctor --android-licenses
                                                                    Warning: Errors during XML parse:
Warning: Additionally, the fallback loader failed to parse the XML.
Warning: Errors during XML parse:       ] 64% Fetch remote repository...
Warning: Additionally, the fallback loader failed to parse the XML.ry...
[======================================] 100% Computing updates...
All SDK package licenses accepted.


C:\Users\272241>
```

```
C:\Users\272241>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.22631.4751], locale e
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[✓] Chrome - develop for the web
[!] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.12.4)
    X Visual Studio is missing necessary components. Please re-run the Visual Studio installe
      development with C++" workload, and include these components:
        MSVC v142 - VS 2019 C++ x64/x86 build tools
         - If there are multiple build tool versions available, install the latest
        C++ CMake tools for Windows
        Windows 10 SDK
[✓] Android Studio (version 2023.1)
[✓] VS Code (version 1.96.4)
[✓] Connected device (3 available)
[✓] Network resources

! Doctor found issues in 1 category.
```

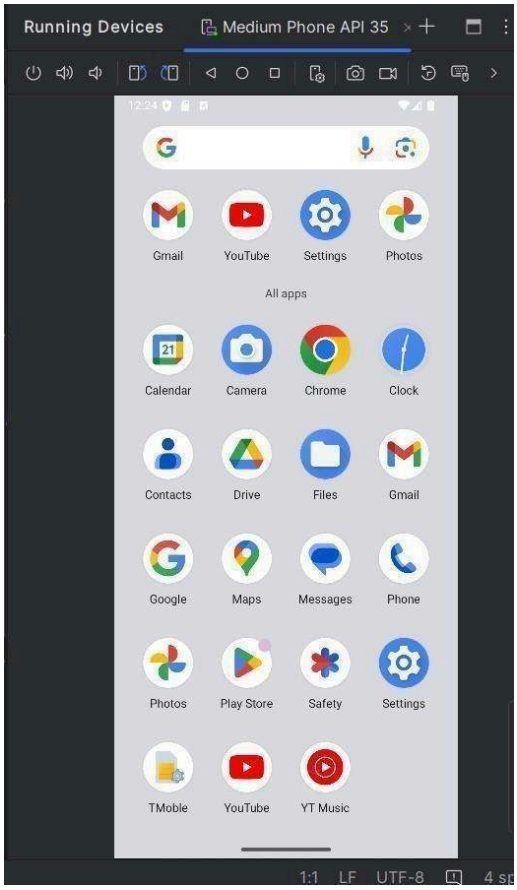
Step 10: - Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application



Step 10.1: - Open Android Studio and go to Tools > AVD Manager. Create a new virtual device.

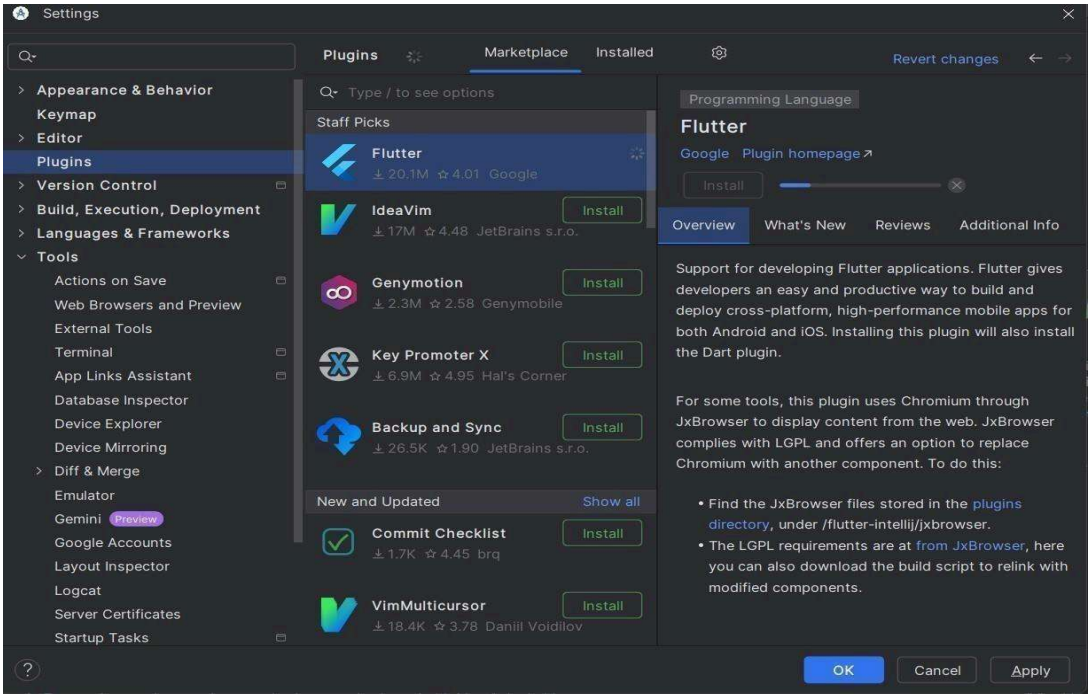


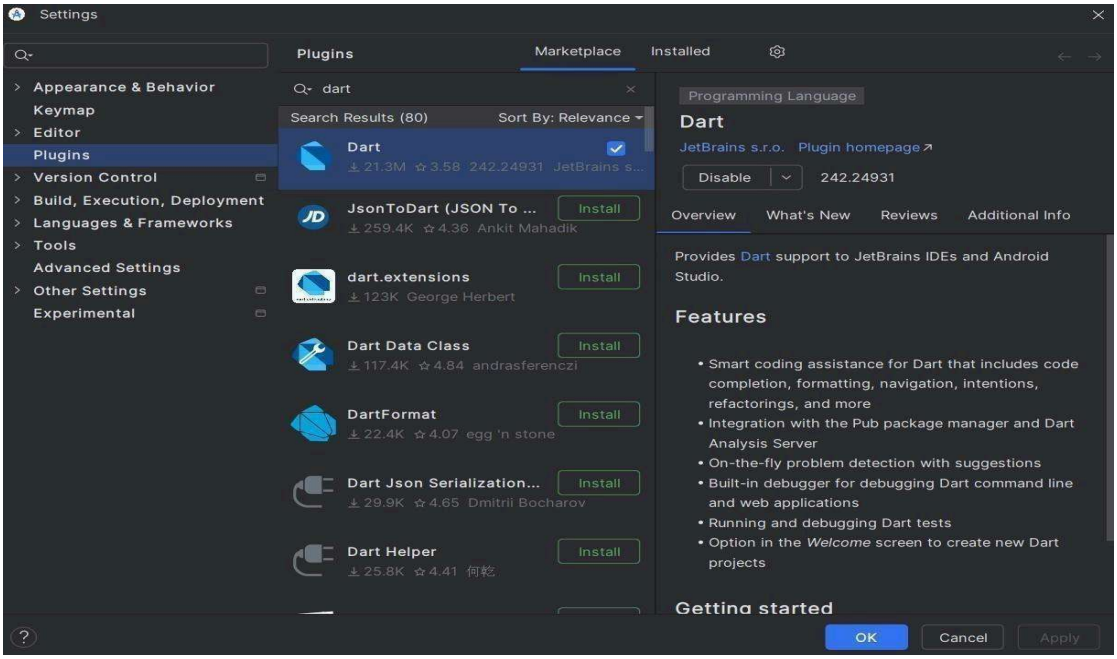
Step 10.2: - Click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen



Step 11: - Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself

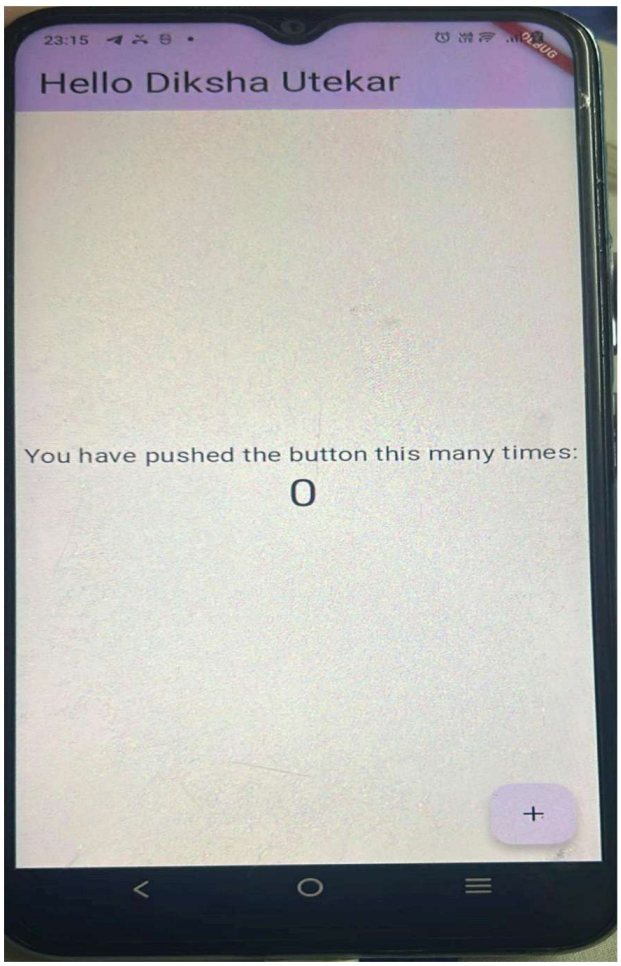
Step 11.1: - Open the Android Studio and then go to File->Settings->Plugins. Now, search the Flutter plugin. If found, select Flutter plugin and click install





Step 11.2: - Restart the Android Studio

Step 12: - Go to File > New Project > Create Flutter Project, then select the project name and location, and click Next to proceed.



MAD & PWA Lab
Journal

Experiment No.	02
Experimen t Title.	To design Flutter UI by including common widgets.
Roll No.	**63**
Name	**Diksha Abhinay Utekar**
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Project title: Snapchat
Roll No : 63

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	63
Name	Diksha Utekar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Aim: To design Flutter UI by including common widgets.

Theory:

Flutter follows a widget-based approach where everything in the UI is a widget. Widgets can be classified into two main types:

- Stateless Widgets: Do not change their state once built (e.g., Text, Container).
- Stateful Widgets: Can update dynamically based on user interaction (e.g., TextField, Checkbox).

Commonly Used Widgets in Flutter-

- Scaffold Widget

The Scaffold widget provides the basic structure for a Flutter app, including an AppBar, Drawer, FloatingActionButton, and BottomNavigationBar. It is a fundamental widget used to create a standard screen layout in Flutter.

- Container Widget

A Container is a box model widget that can hold other widgets. It is commonly used for adding padding, margins, borders, and background decorations.

- Row and Column Widgets
    - Row: Arranges widgets horizontally.
    - Column: Arranges widgets vertically.
      These two widgets are fundamental for designing layouts in Flutter.

- ListView Widget

The ListView widget is used for displaying a scrollable list of items. It is useful for showing large amounts of data dynamically.

- Stack Widget

The Stack widget is used to place widgets on top of each other. This is useful for creating overlapping UI elements such as banners, profile images, or layered designs.

- ElevatedButton Widget

The ElevatedButton widget is used for clickable buttons with a raised effect. It is a commonly used button in Flutter applications.

- TextField Widget

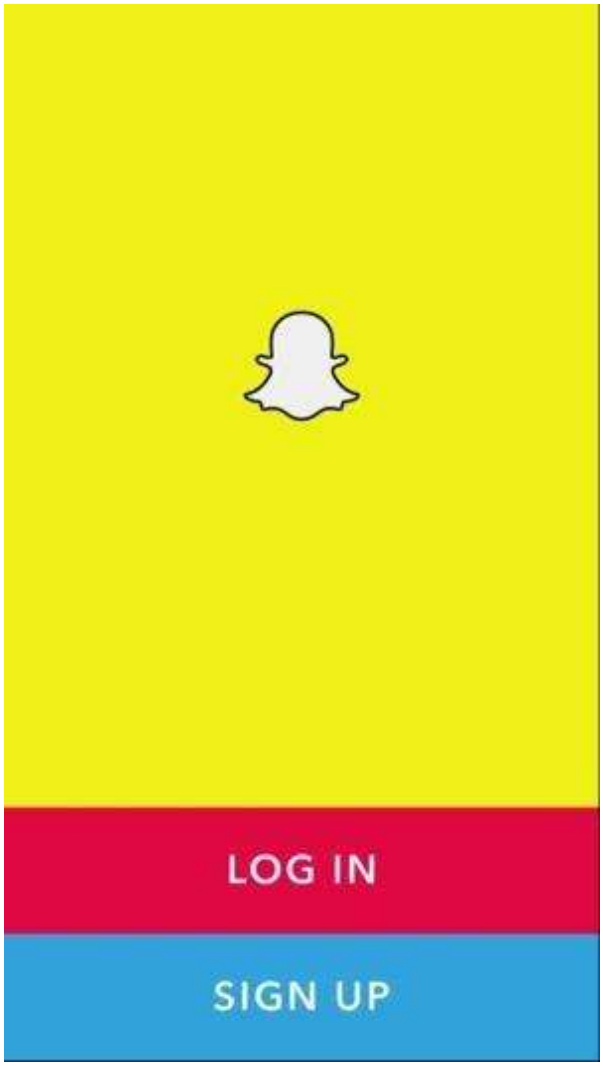
The TextField widget is used to take user input, such as entering a name, email, or password. It is commonly used in forms and authentication screens.

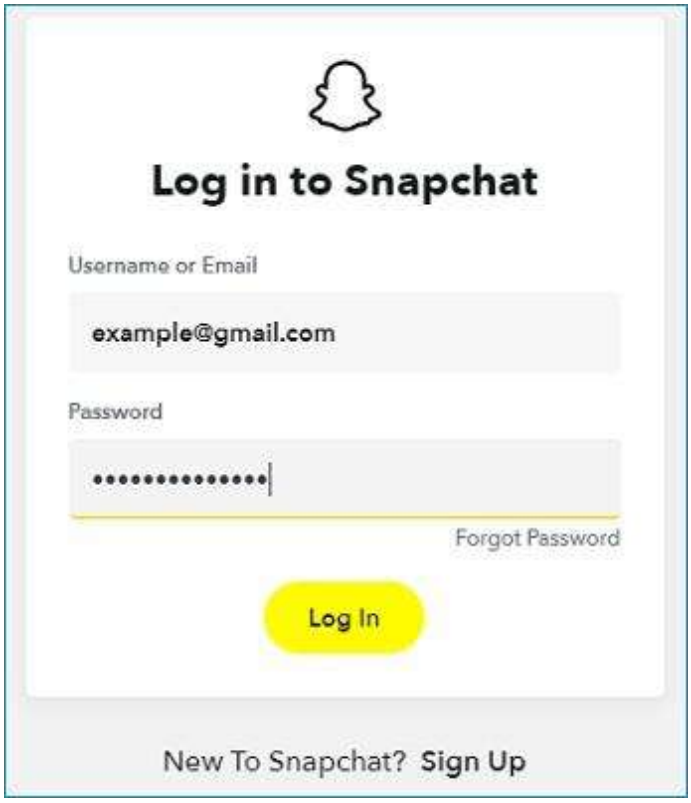
Code:

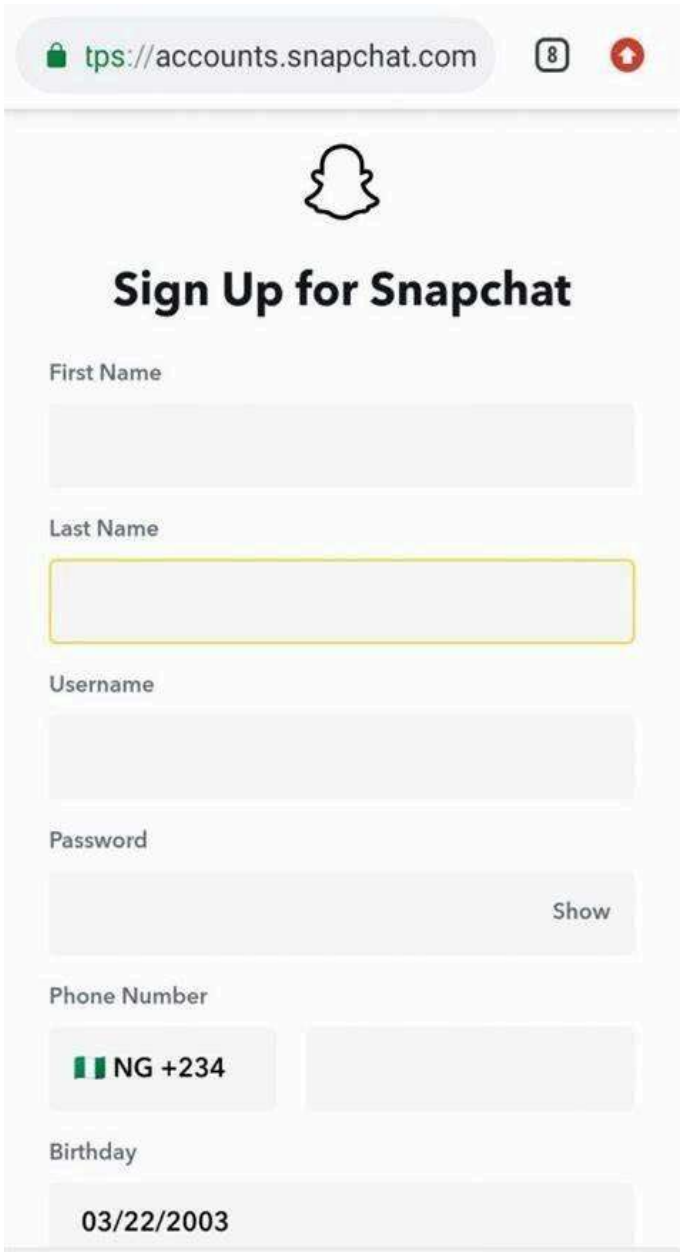
main.dart file

```
import 'package:flutter/material.dart';
import 'package:snapchat_clone_ui/screens/home_screen.dart';
import 'package:snapchat_clone_ui/screens/initial_screen.dart';
import 'package:snapchat_clone_ui/screens/login_screen.dart';
import 'package:snapchat_clone_ui/screens/signup_screen.dart';

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

```
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primaryColor: Color(0xFF838486),
      ),
      initialRoute: '/',
      routes: {
        '/': (context) => InitialScreen(),
        '/login_screen': (context) =>
        LoginScreen(),
        '/signup_screen': (context) => SignupScreen(),
        '/home_screen': (context) => HomeScreen(),

      },
    );
  }
}
```


LOG IN
SIGN UP


Log in to Snapchat
Username or Email
example@gmail.com
Password
Forgot Password
Log In
New To Snapchat? Sign Up


tps://accounts.snapchat.com
8
Sign Up for Snapchat
First Name
Last Name
Username
Password
Show
Phone Number
NG +234
Birthday
03/22/2003

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	**63**
Name	**Diksha Abhinay Utekar**
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: 03

Name: Diksha Utekar Class: D15A
Roll No: 63

Aim:

To implement icons, images, and custom fonts in a Flutter application to enhance its visual aesthetics and user experience.

Theory:

Flutter is a powerful open-source UI framework that enables developers to create natively compiled applications for mobile, web, and desktop platforms from a single codebase. One of Flutter's key advantages is its ability to provide highly customizable user interfaces, making it easier to build visually appealing applications.

In app development, icons, images, and fonts play a crucial role in improving usability and design consistency. These elements help in conveying information effectively and reinforcing branding. Flutter offers seamless integration of these visual components, enhancing both functionality and user engagement.

Importance of Visual Elements in Flutter

- **Enhanced User Experience:** Icons and images make the app more intuitive and visually engaging.
- **Information Conveyance:** Icons provide quick representation, reducing the need for lengthy text descriptions.
- **Branding and Personalization:** Custom fonts and images create a unique brand identity, making the app stand out.

Flutter supports multiple image formats (JPEG, PNG, WebP, GIF, etc.) and allows the integration of icons and fonts effortlessly. Below are the methods to incorporate these elements into a Flutter application:

1. Adding Images in Flutter

Flutter supports **local** and **network images** for UI components.

Local Images

Images stored within the project can be used in the application by defining them in the asset folder and declaring them in the configuration file. Supported image formats include **PNG**, **JPG**, **GIF**, **SVG**, and more.

Network Images

Flutter allows fetching images from external sources using URLs. Cached images can be utilized for improved performance.

2. Adding Custom Fonts in Flutter

Custom fonts allow developers to enhance the UI design and branding of their applications.

Steps to Add Custom Fonts:

1. The font files need to be downloaded and stored in the project directory.
2. These fonts must be declared in the project configuration file.
3. Once declared, the fonts can be applied to text widgets for a distinct appearance.

3. Using Icons in Flutter

Icons play a crucial role in UI clarity and usability. Flutter supports both **built-in Material Icons** and **custom icons**.

Built-in Icons (Material Icons)

Flutter provides a collection of icons that are integrated within the framework, requiring no additional setup.

Custom Icons (SVG, PNG, Font Icons)

Developers can integrate custom icons in various formats, such as **SVG**, **PNG**, or **Font-Based Icons** like FontAwesome. These icons enhance visual consistency and allow for a personalized UI experience.

Code Snippets: home_screen.dart

```
import 'package:flutter/material.dart';
import 'package:snapchat_clone_ui/screens/camera_screen.dart';
import 'package:snapchat_clone_ui/screens/chat_screen.dart';
import
'package:snapchat_clone_ui/screens/location_screen.dart';
import 'package:snapchat_clone_ui/screens/reels_screen.dart';
import 'package:snapchat_clone_ui/screens/stories_screen.dart';
import 'initial_screen.dart';

class HomeScreen extends StatefulWidget {
  @override
  State<HomeScreen> createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  int _selectedIndex = 0;
  static const List<Widget> _widgetOptions = <Widget>[
    LocationScreen(),
    ChatScreen(),
    CameraScreen(),
    StoriesScreen(),
    ReelScreen(),
  ];

  void _onItemTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(child: _widgetOptions[_selectedIndex]),
      bottomNavigationBar: BottomNavigationBar(
        items: <BottomNavigationBarItem>[
          BottomNavigationBarItem(
          backgroundColor: Colors.black,
          icon: Icon(
              Icons.location_on_outlined,
              size: 25.0,
              color: Colors.white,
            ),
            label: "",
          ),
```

```
BottomNavigationBarItem(
  backgroundColor: Colors.black,
```

```
          icon: Icon(
            Icons.chat_bubble_outline_rounded
            , size: 25.0,
            color: Colors.white,
          ),
          label: "",
        ),
        BottomNavigationBarItem(
          backgroundColor: Colors.black,
          icon: Icon(
          Icons.camera_alt_outlined,
          size: 25.0,
            color: Colors.white,
          ),
          label: "",
        ),
        BottomNavigationBarItem(
          backgroundColor: Colors.black,
          icon: Icon(
          Icons.group_outlined,
            size: 25.0,
            color: Color(0XFF10ACFF),
          ),
          label: "",
        ),
        BottomNavigationBarItem(
          backgroundColor: Colors.black,
          icon: Icon(
          Icons.play_arrow_outlined,
          size: 25.0,
            color: Colors.white,
          ),
          label: "",
        ),
      ],
      type: BottomNavigationBarType.fixed,
      currentIndex: _selectedIndex,
      selectedItemColor: Color(0XFF10ACFF),
      backgroundColor: Colors.black,
      onTap: _onItemTapped,
      unselectedItemColor: Colors.white,
    ),
   );
  }
}
```

- **Code for chat_screen.dart**

```
import 'package:flutter/material.dart';
class ChatScreen extends StatefulWidget {
 const ChatScreen({Key? key}) : super(key: key);

 @override
 State<ChatScreen> createState() => _ChatScreenState();
}

class _ChatScreenState extends State<ChatScreen> {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   backgroundColor: Colors.white,
   body: SafeArea(
    child: Column(
     children: [
     Row(
       mainAxisAlignment: MainAxisAlignment.spaceBetween,
       children: [
        Padding(
         padding: const EdgeInsets.all(8.0),
         child: Row(
          children: [
           CircleAvatar(
            backgroundColor: Colors.transparent,
            backgroundImage:
AssetImage("assets/images/hero.png"),
           ),
           SizedBox(width: 10.0),
           CircleAvatar(
            backgroundColor: Colors.grey[100],
            child: Icon(Icons.search, color: Colors.grey[700]),
           ),
          ],
         ),
        ),
       Text(
```

"Chat",
style: TextStyle(fontSize: 20.0, fontWeight:

```
FontWeight.bold),
            ),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: Row(
                children: [
                  CircleAvatar(
                    backgroundColor: Colors.grey[100],
                    child: Icon(Icons.person_add, color: Colors.grey[700]),
                  ),
                  SizedBox(width: 10.0),
                  CircleAvatar(
                    backgroundColor: Colors.grey[100],
                    child: Icon(Icons.more_horiz, color: Colors.grey[700]),
                  ),
                ],
              ),
            ),
          ],
        ),

        //list tile for chats here
        SingleChildScrollView(
        child: Column(
            children:
              [
              ChatTile(
                name: "Team Snapchat",
                image: NetworkImage(
                  "https://us-east1-aws.api.snapchat.com/web-
capture/www.snapchat.com/discover/preview/facebook.png",
                ),
                trailing: Icon(
                  Icons.chat_bubble_outline_sharp,
                  color: Colors.grey[400],
                ),
                child: Row(children: [Text("Blocked")]),
              ),
              ChatTile(
                name: "Richa",
```

```
image: AssetImage("assets/images/hero_4.png"),
trailing: Icon(
  Icons.chat_bubble_outline_sharp,
  color: Colors.grey[400],
),
```

```
  child:
    Row(
    children:
    [
      Icon(Icons.square, color: Colors.red, size: 15.0),
      SizedBox(width: 5.0),
      Text("New Snap", style: TextStyle(color: Colors.red)),
      SizedBox(width: 5.0),
      Text("."),
      SizedBox(width: 5.0),
      Text("3w"),
    ],
  ),
),
ChatTile(
  name: "Anurag",
  image: AssetImage("assets/images/hero_2.png"),
  trailing: Icon(
    Icons.chat_bubble_outline_sharp,
    color: Colors.grey[400],
  ),
  child:
    Row(
    children:
    [
      Icon(Icons.square, color: Colors.purple, size: 15.0),
      SizedBox(width: 5.0),
      Text(
        "New Snap",
        style: TextStyle(color: Colors.purple),
      ),
      SizedBox(width: 5.0),
      Text("."),
      SizedBox(width: 5.0),
      Text("3w"),
    ],
  ),
),
ChatTile(
  name: "Niyati",
```

```
image: AssetImage("assets/images/hero_5.png"),
trailing: Icon(
  Icons.chat_bubble_outline_sharp,
  color: Colors.grey[400],
),
child:
  Row(
  children:
  [
    Icon(Icons.square, color: Colors.red, size: 15.0),
```

```
      SizedBox(width: 5.0),
      Text("New Snap", style: TextStyle(color: Colors.red)),
      SizedBox(width: 5.0),
      Text("."),
      SizedBox(width: 5.0),
      Text("3w"),
    ],
  ),
),
ChatTile(
  name:
  "Ritik",
  image: AssetImage("assets/images/hero_3.png"),
  trailing: Icon(
    Icons.camera_alt_outlined,
    color: Colors.grey[400],
  ),
  child:
    Row(
    children:
    [
      Icon(Icons.chat_bubble_outline_outlined, size: 15.0),
      SizedBox(width: 5.0),
      Text("Tap to chat"),
    ],
  ),
),
ChatTile(
  name: "Richa",
  image: AssetImage("assets/images/hero_4.png"),
  trailing: Icon(
    Icons.chat_bubble_outline_sharp,
    color: Colors.grey[400],
  ),
  child:
    Row(
    children:
    [
      Icon(Icons.square, color: Colors.red, size: 15.0),
      SizedBox(width: 5.0),
      Text("New Snap", style: TextStyle(color: Colors.red)),
```

```
      SizedBox(width: 5.0),
      Text("."),
      SizedBox(width: 5.0),
      Text("3w"),
    ],
  ),
),
ChatTile(
```

```
  name: "Anurag",
  image: AssetImage("assets/images/hero_2.png"),
  trailing: Icon(
    Icons.chat_bubble_outline_sharp,
    color: Colors.grey[400],
  ),
  child:
    Row(
    children:
    [
      Icon(Icons.square, color: Colors.purple, size: 15.0),
      SizedBox(width: 5.0),
      Text(
        "New Snap",
        style: TextStyle(color: Colors.purple),
      ),
      SizedBox(width: 5.0),
      Text("."),
      SizedBox(width: 5.0),
      Text("3w"),
    ],
  ),
),
ChatTile(
  name: "Niyati",
  image: AssetImage("assets/images/hero_5.png"),
  trailing: Icon(
    Icons.camera_alt_outlined,
    color: Colors.grey[400],
  ),
  child:
    Row(
    children:
    [ Icon(
        Icons.send_outlined,
        color:
        Color(0XFF10ACFF), size:
        15.0,
      ),
      SizedBox(width: 5.0),
      Text("Opened"),
```

```
        SizedBox(width: 5.0),
        Text("."),
        SizedBox(width: 5.0),
        Text("3w"),
      ],
    ),
  ),
```

```
            ],
          ),
        ),
      ],
    ),
  ),
  floatingActionButton: FloatingActionButton(
    onPressed: () {},
    backgroundColor: Color(0XFF10ACFF),
    child: Icon(Icons.edit_note_outlined),
  ),
 );
 }
}

//chatTile widget
class ChatTile extends StatelessWidget {
 final name;
 final image;
 final child;
 final trailing;

 const ChatTile({Key? key, this.image, this.name, this.child,
this.trailing})
  : super(key: key);

 @override
 Widget build(BuildContext context) {
  return Column(
   children: [
    Divider(height: 3),
    ListTile(
     leading: CircleAvatar(
      radius: 20.0,
      backgroundColor: Colors.transparent,
      backgroundImage: image,
     ),
     trailing: trailing,
     title: Text(name, style: TextStyle(fontWeight: FontWeight.bold)),
     subtitle: child,
```

```
    ),
    Divider(height: 3),
  ],
);
```

```
  }
}
```

- Code for stories_screen.dart

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

class StoriesScreen extends StatefulWidget {
  const StoriesScreen({Key? key}) : super(key: key);

  @override
  State<StoriesScreen> createState() => _StoriesScreenState();
}

class _StoriesScreenState extends State<StoriesScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: SingleChildScrollView(
          child: Column(
            children:
              [ Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  Padding(
                    padding: const EdgeInsets.all(8.0),
                    child: Row(
                      children: [
                        CircleAvatar(
                          backgroundColor: Colors.transparent,
                          backgroundImage:
AssetImage("assets/images/hero.png"),
                        ),
                        SizedBox(width: 10.0),
                        CircleAvatar(
                          backgroundColor: Colors.grey[100],
                          child: Icon(Icons.search, color: Colors.grey[700]),
                        ),
                      ],
                    ),
```

```
    ),
    Text(
      "Stories",
      style: TextStyle(
        fontSize: 20.0,
        fontWeight: FontWeight.bold,
      ),
    ),
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: Row(
        children: [
          CircleAvatar(
            backgroundColor: Colors.grey[100],
            child: Icon(
              Icons.person_add,
              color: Colors.grey[700],
            ),
          ),
          SizedBox(width: 10.0),
          CircleAvatar(
            backgroundColor: Colors.grey[100],
            child: Icon(
              Icons.more_horiz,
              color: Colors.grey[700],
            ),
          ),
        ],
      ),
    ),
  ],
),
SizedBox(height: 30.0),

//Friends section
Container(
  padding: EdgeInsets.symmetric(horizontal: 20.0),
  child: Column(
    children: [
      Container(
```

```
alignment:
Alignment.topLeft, child:
Text(
  "Friends",
  style: TextStyle(
```

```
      fontSize: 18.0,
      fontWeight: FontWeight.bold,
    ),
  ),
),
SizedBox(height: 20.0),
Container(
  height: 140,
  child: ListView(
    scrollDirection: Axis.horizontal,
    children: [
      Row(
        children: [
          storyBubble(
            name:
            "Anurag",
            image: AssetImage("assets/images/hero_2.png"),
          ),
          SizedBox(width: 15.0),
          storyBubble(
            name: "Richa",
            image: AssetImage("assets/images/hero_4.png"),
          ),
          SizedBox(width: 15.0),
          storyBubble(
            name: "Niyati",
            image: AssetImage("assets/images/hero_5.png"),
          ),
          SizedBox(width: 15.0),
          storyBubble(
            name: "Ritik",
            image: AssetImage("assets/images/hero_3.png"),
          ),
          SizedBox(width: 15.0),
          storyBubble(
            name: "Niyati",
            image: AssetImage("assets/images/hero_5.png"),
          ),
        ],
```

```
            ),
          ],
        ),
      ),
    ],
  ),
```

```
),

//Subscriptions section
Container(
  padding: EdgeInsets.symmetric(horizontal: 20.0),
  child: Column(
    children: [
      Container(
        alignment:
        Alignment.topLeft, child:
        Row(
          children: [
            Text(
              "Subscriptions"
              ,                style:
              TextStyle(
              fontSize: 17.0,
                fontWeight: FontWeight.bold,
              ),
            ),
            SizedBox(width: 5.0),
            Icon(Icons.arrow_forward_ios, size: 15.0),
          ],
        ),
      ),
      SizedBox(height: 20.0),
      Container(
        height: 200.0,
        child: ListView(
          scrollDirection: Axis.horizontal,
          children: [
            Row(
              children: [
                subscriptionTile(
                  name: "Kundu",
                  image: Image.network(

"https://c4.wallpaperflare.com/wallpaper/923/727/796/anime-digital-art-
artwork-2d-portrait-display-hd-wallpaper-preview.jpg",
                    height: 200.0,
                  ),
```

```
),
SizedBox(width: 10.0),
subscriptionTile(
name: "Tumami",
  image: Image.network(
```

```
"https://images.pexels.com/photos/9410606/pexels-photo-
9410606.jpeg?cs=srgb&dl=pexels-zetong-li-9410606.jpg&fm=jpg",
                        height: 200.0,
                      ),
                    ),
                    SizedBox(width: 10.0),
                    subscriptionTile(
                    name: "Jan Goldz",
                      image: Image.network(

"https://c0.wallpaperflare.com/preview/303/473/216/man-standing-on-
mountain-during-sunset.jpg",
                        height: 200.0,
                      ),
                    ),
                    SizedBox(width: 10.0),
                    subscriptionTile(
                    name: "Bastrop",
                      image: Image.network(

"https://c4.wallpaperflare.com/wallpaper/367/257/149/anime-anime-
girls-digital-art-artwork-2d-hd-wallpaper-preview.jpg",
                        height: 200.0,
                      ),
                    ),
                    SizedBox(width: 10.0),
                    subscriptionTile(
                    name: "Mulessa",
                      image: Image.network(
                        "https://wallpapercave.com/wp/wp2722942.jpg",
                        height: 200.0,
                      ),
                    ),
                  ],
                ),
              ],
            ),
          ),
        ),
```

```
    ],
  ),
),

//Discover Section
Container( padding: EdgeInsets.symmetric(horizontal: 20.0, vertical:
20.0),
child:
Column(
children: [
Container(
alignment: Alignment.topLeft,
child: Text( "Discover",
style: TextStyle(
fontSize: 17.0, fontWeight: FontWeight.bold, ), ), ), ListView(
shrinkWrap: true,
children: [ Column(
children: [
Row(
children: [
Expanded( flex: 2, child: DiscoverTile(
 name: "Weird Mud Games", image: Image.network(

"https://c4.wallpaperflare.com/wallpaper/367/257/149/anime-anime-
girls-digital-art-artwork-2d-hd-wallpaper-preview.jpg",
                    height: 380,
                  ),
                ),
              ),
              SizedBox(width: 8.0),
              Expanded(
                flex: 2,
                child: DiscoverTile(
                  name: "Mulessa",
                  image:
                  Image.network(

"https://wallpaperaccess.com/full/1559254.png",
                    height: 380.0,
                  ),
```

```
            ),
          ),
        ],
      ),
    ],
  ),
  SizedBox(height: 0.0),
  Column(
    children: [
      Row(
      children: [
          Expanded(
            flex: 2,
            child: DiscoverTile(
              name: "Weird Mud Games",
              image: Image.network(
"https://c4.wallpaperflare.com/wallpaper/367/257/149/anime-anime-girls-digital-art-artwork-2d-hd-wallpaper-preview.jpg",
                height: 380,
              ),
            ),
          ),
          SizedBox(width: 8.0),
          Expanded(
            flex: 2,
            child: DiscoverTile(
              name: "Mulessa",
              image:
              Image.network(
"https://wallpaperaccess.com/full/1559254.png",
                height: 380.0,
              ),
            ),
          ),
        ],
      ),
    ],
```

```
            ),
          ],
        ),
      ],
    ),
  ),
```

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

class subscriptionTile extends StatelessWidget {
 final name;
 final image;
 const subscriptionTile({Key? key, this.name, this.image}) :
super(key: key);

 @override
 Widget build(BuildContext context) {
  return Column(
   children: [
    Stack(
    children: [
      image,
      Positioned(
      bottom: 2,
       child: Padding(
        padding: const EdgeInsets.symmetric(
         horizontal: 8.0,
         vertical: 2.0,
        ),
        child: Text(
         name,
         style: TextStyle(
          color:
          Colors.white,
          fontWeight: FontWeight.bold,
         ),
        ),
       ),
      ),
     ],
```

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

```
class DiscoverTile extends StatelessWidget {
  final name;
  final image;
  const DiscoverTile({Key? key, this.name, this.image}) : super(key:
key);

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Stack(
        children: [
            image,
            Positioned(
            bottom: 30,
            child: Padding(
                padding: const EdgeInsets.symmetric(
                  horizontal: 8.0,
                  vertical: 2.0,
                ),
                child: Text(
                  name,
                  style: TextStyle(
                    color:
                    Colors.white,
                    fontWeight: FontWeight.bold,
                    fontSize: 20.0,
                  ),
                ),
              ),
            ),
          ],
        ),
      ],
    );
  }
}

class storyBubble extends StatelessWidget {
  final name;
  final image;
```

```
  const storyBubble({Key? key, this.image, this.name}) : super(key:
key);

  @override
```

```
  Widget build(BuildContext context) {
    return Column(
      children: [
        CircleAvatar(
        radius: 45.0,
          backgroundColor: Colors.purple,
          child: CircleAvatar(
            radius: 43.0,
            backgroundColor: Colors.white,
            child: CircleAvatar(backgroundImage: image, radius: 40.0),
          ),
        ),
        SizedBox(height: 10.0),
        Text(name, style: TextStyle(fontWeight: FontWeight.w500)),
      ],
    );
  }
}
```

Screenshots:


localhost:58929
localhost:58929/#/home_screen
Camera Screen


localhost:60137
localhost:60137/#/home_screen
Stories
Friends
Anurag
Richa
Niyati
Ritik
Niyati
Subscriptions

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	**63**
Name	**Diksha Abhinay Utekar**
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: 04

Name: Diksha Utekar
Class: D15A
Roll No: 63

Aim: To create an interactive Form using form widget

Theory:

Flutter provides a Form widget that enables developers to create interactive forms for collecting user input. Forms are an essential part of many applications, such as registration screens, login pages, feedback forms, and data entry modules. They allow structured input gathering, validation, and submission, enhancing user interaction and data integrity.

Flutter's Form widget works in combination with various input fields, validation techniques, and state management to ensure a seamless user experience.

Importance of Forms in App Development

1. Efficient User Input Handling – Forms streamline the process of collecting user data.
2. Validation Mechanism – Ensures only correct and meaningful data is submitted.
3. State Management – Helps in tracking and modifying input fields dynamically.
4. Improved User Experience – Provides structured input fields for better accessibility and usability.

Key Components of Forms in Flutter

1. Form Widget (Form)

The Form widget acts as a container that manages multiple input fields. It provides an easy way to validate and track changes in form fields.

Key Properties:

- key – A GlobalKey<FormState> to track form state and validation.
- child – Contains form fields like TextFormField, DropdownButtonFormField, etc.

- autovalidateMode – Determines when validation messages should appear (disabled, always, onUserInteraction).

2. Input Fields (Form Fields)

a) TextFormField

- A fundamental widget for text input in Flutter.
- Supports validation, formatting, and user interaction.
- Properties:
    - controller – Controls and retrieves the text entered by the user.
    - keyboardType – Defines the type of input (text, number, email, etc.).
    - decoration – Customizes the appearance (label, hint text, icon).
    - validator – Implements validation logic.

b) DropdownButtonFormField

- Provides a dropdown list for selecting predefined options.
- Properties:
    - items – Defines the list of selectable options.
    - value – Holds the currently selected item.
    - onChanged – Executes when the user selects a different item.

c) CheckboxListTile

- Displays a checkbox with a title and subtitle.
- Properties:
    - value – Represents whether the checkbox is selected.
    - onChanged – Detects changes in the selection state.
    - title – Describes the purpose of the checkbox.

d) RadioListTile

- Provides a list of radio buttons where only one can be selected.
- Properties:
    - value – The value assigned to each radio button.
    - groupValue – The currently selected value within the group.
    - onChanged – Updates the selection when a new option is chosen.

e) SwitchListTile

- Allows users to toggle between on/off states.
- Properties:

- value – Represents the current state (true/false).
- onChanged – Detects when the switch is toggled.
- title – Displays the label for the switch.

3. Managing Form State

a) GlobalKey<FormState>

- A unique key used to track the form's state.
- Helps in validation and resetting form fields.

b) FormState Methods

- validate() – Checks if all fields meet the validation rules.
- save() – Saves the form data when validation is successful.
- reset() – Clears all fields in the form.

Form Validation and Submission Process

1. The user enters data into form fields.
2. On submission, FormState.validate() ensures all inputs are correct.
3. If valid, the save() function processes and stores the data.
4. If invalid, appropriate error messages are displayed.

Code Snippets: initial_screen.dart

```
import 'package:flutter/material.dart';
import 'package:snapchat_clone_ui/custom_widgets/custom_widgets.dart';

//Constants

const Color scaffoldColor = Color(0xFFFFFC00);
const Color loginButtonColor = Colors.white;
const Color signupButtonColor = Color(0xFF0EAEFE);

class InitialScreen extends StatefulWidget {
  @override
  _InitialScreenState createState() => _InitialScreenState();
}

class _InitialScreenState extends State<InitialScreen> {
  @override
```

```
Widget build(BuildContext context) {
```

```
return Scaffold(
  backgroundColor: scaffoldColor,
  body: Stack(
    children:
      [
      Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Container(
            height: 180,
            decoration: BoxDecoration(
              image: DecorationImage(
                image: AssetImage("assets/images/icon.png"),
              ),
            ),
          ),
        ],
      ),
      Center(
        child: Padding(
          padding: const EdgeInsets.symmetric(vertical: 30),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.end,
            children: [
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  GestureDetector(
                    onTap: () {
                      Navigator.pushNamed(context, '/login_screen');
                    },
                    child:
                      ReusableButton(
                      btnHeight: 60.0,
                      btnWidth: 130.0,
                      btnColour: loginButtonColor,
                      btnCircularRadius: 80.0,
                      btnChild: Text(
                        "Log in",
                        style: TextStyle(
                          fontSize: 25,
                          fontWeight: FontWeight.bold,
                          color: Colors.black,
                        ),
                      ),
                    ),
                  ),
                  SizedBox(width: 15),
                  GestureDetector(
```

```
onTap: () {
   Navigator.pushNamed(context, '/signup_screen');
  },
  child: ReusableButton(
```

```
                    btnHeight: 60.0,
                    btnWidth: 130.0,
                    btnColour: signupButtonColor,
                    btnCircularRadius: 80.0,
                    btnChild: Text(
                      "Sign Up",
                      style: TextStyle(
                      fontSize: 25,
                        fontWeight: FontWeight.bold,
                        color: Colors.white,
                      ),
                    ),
                  ),
                ),
              ],
            ),
          ],
        ),
      ),
    ),
  ],
 ),
 );
 }
}
```

- Code for login_screen.dart

```
import


'package:flutter/material.dart';


import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:snapchat_clone_ui/custom_widgets/custom_widgets.dart';

const Falcon hiddenEye = Falcon(FontAwesomeIcons.eyeSlash);
const Falcon eye = Falcon(FontAwesomeIcons.eye);

class LoginScreen extends StatefulWidget {
 @override
 _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
```

```
bool _obscureText = true;
Widget eyeStatus = hiddenEye;

void _toggle() {
  setState(() {
    _obscureText = !_obscureText;
    if (_obscureText == false) {
```

```
      eyeStatus = eye;
    } else {
      eyeStatus = hiddenEye;
    }
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      leading: GestureDetector(
        onTap: () {
          Navigator.pop(context);
        },
        child: Icon(Icons.arrow_back_ios, color: Colors.grey),
      ),
      elevation: 0,
      backgroundColor: Colors.white,
    ),
    body: Center(
      child:
        SingleChildScrollView(
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: [
              Text(
                "Log in",
                style: TextStyle(fontSize: 40, color: Colors.black),
              ),
              SizedBox(height: 20),
              CustomSnapTextField(
                label: "USERNAME OR EMAIL",
                isPasswordField: false,
                autoFocus: true,
              ),
              SizedBox(height: 20),
              Column(
                children: [
                  Container(
                    alignment: Alignment.centerLeft,
                    margin: EdgeInsets.symmetric(horizontal: 50),
                    child: Text(
                      "PASSWORD",
                      style: TextStyle(
                        fontSize: 18,
                        fontWeight: FontWeight.bold,
                        color: Color(0xFF51B5E5),
```

```
    ),
  ),
),
```

```
    Padding(
      padding: const EdgeInsets.symmetric(horizontal: 50),
      child: TextField(
        obscureText: _obscureText,
        autofocus: false,
        cursorHeight: 33,
        cursorWidth: 2,
        decoration: InputDecoration(
          suffixIcon: GestureDetector(
          onTap: () {
              _toggle();
            },
            child: eyeStatus,
          ),
          floatingLabelBehavior: FloatingLabelBehavior.never,
          contentPadding: EdgeInsets.all(6),
        ),
        cursorColor: Color(0xFF69B77D),
      ),
    ),
  ],
),
SizedBox(height: 60),
//Forgot your password
GestureDetector(
onTap: () {
    //forgot your password
  },
  child: Text(
    "Forgot your password?",
    style: TextStyle(
      fontSize: 17,
      fontWeight: FontWeight.bold,
      color: Color(0xFF51B5E5),
    ),
  ),
),
SizedBox(height: 90),

//Login button
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 80),
  child: GestureDetector(
    onTap: () {
      Navigator.pushNamed(context, '/home_screen');
    },
    child: Container(
      margin: EdgeInsets.only(top: 20),
```

```
child: Text(
  "Log in",
  style: TextStyle(
```

```
                    fontSize: 25,
                    color: Colors.white,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                alignment: Alignment.center,
                height: 55,
                width: double.infinity,
                decoration: BoxDecoration(
                color: Color(0xFFADB6BD),
                  borderRadius: BorderRadius.circular(80),
                ),
              ),
            ),
          ),
        ],
      ),
    ),
  ),
 );
 }
}
```

- Code for signup_screen.dart import

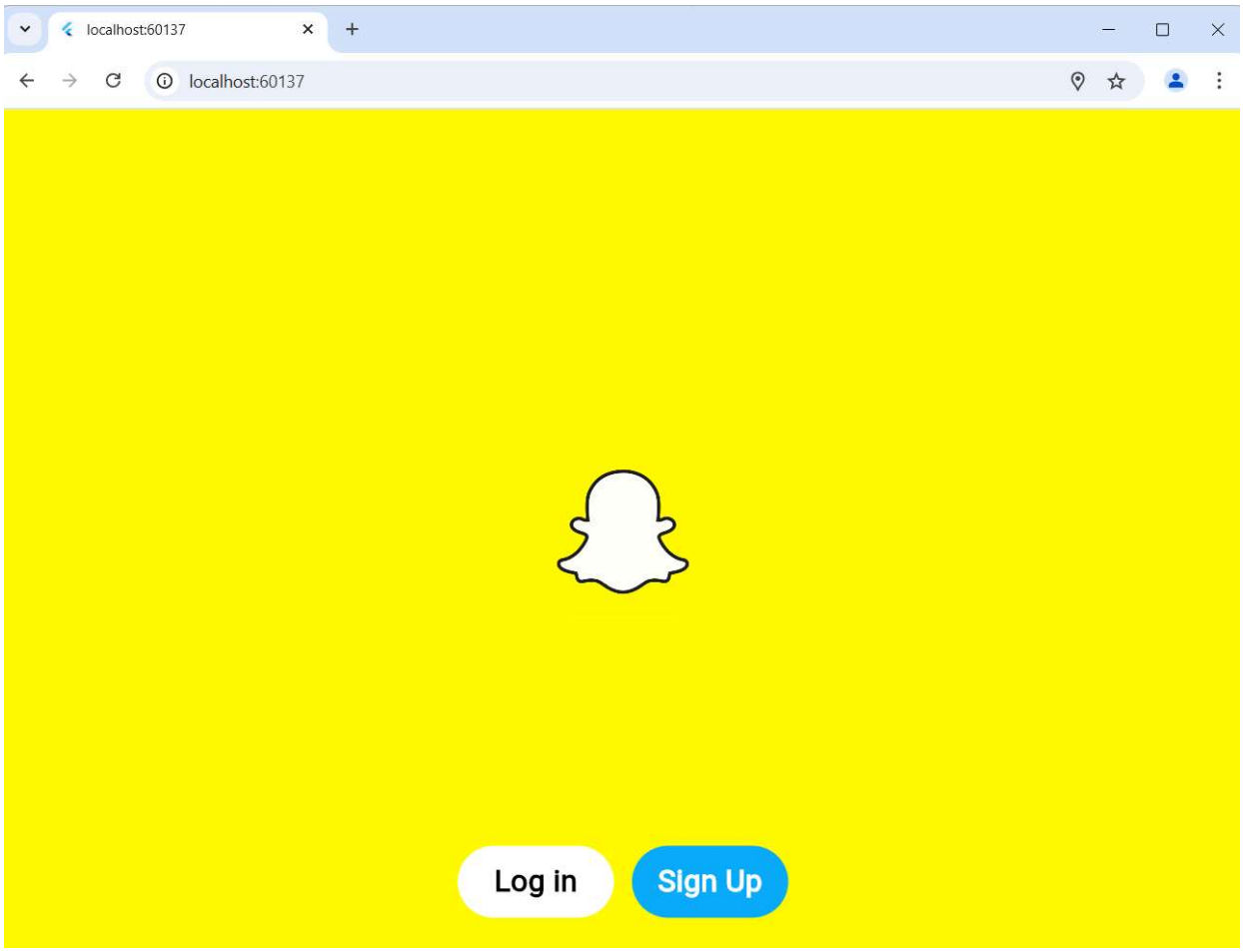
```
'package:flutter/material.dart';
import 'package:snapchat_clone_ui/custom_widgets/custom_widgets.dart';

class SignupScreen extends StatefulWidget {
 @override
 _SignupScreenState createState() => _SignupScreenState();
}

class _SignupScreenState extends State<SignupScreen> {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    leading: GestureDetector(
     onTap: () {
      Navigator.pop(context);
     },
     child: Icon(Icons.arrow_back_ios, color: Colors.grey),
    ),
    elevation: 0,
    backgroundColor: Colors.white,
   ),
```

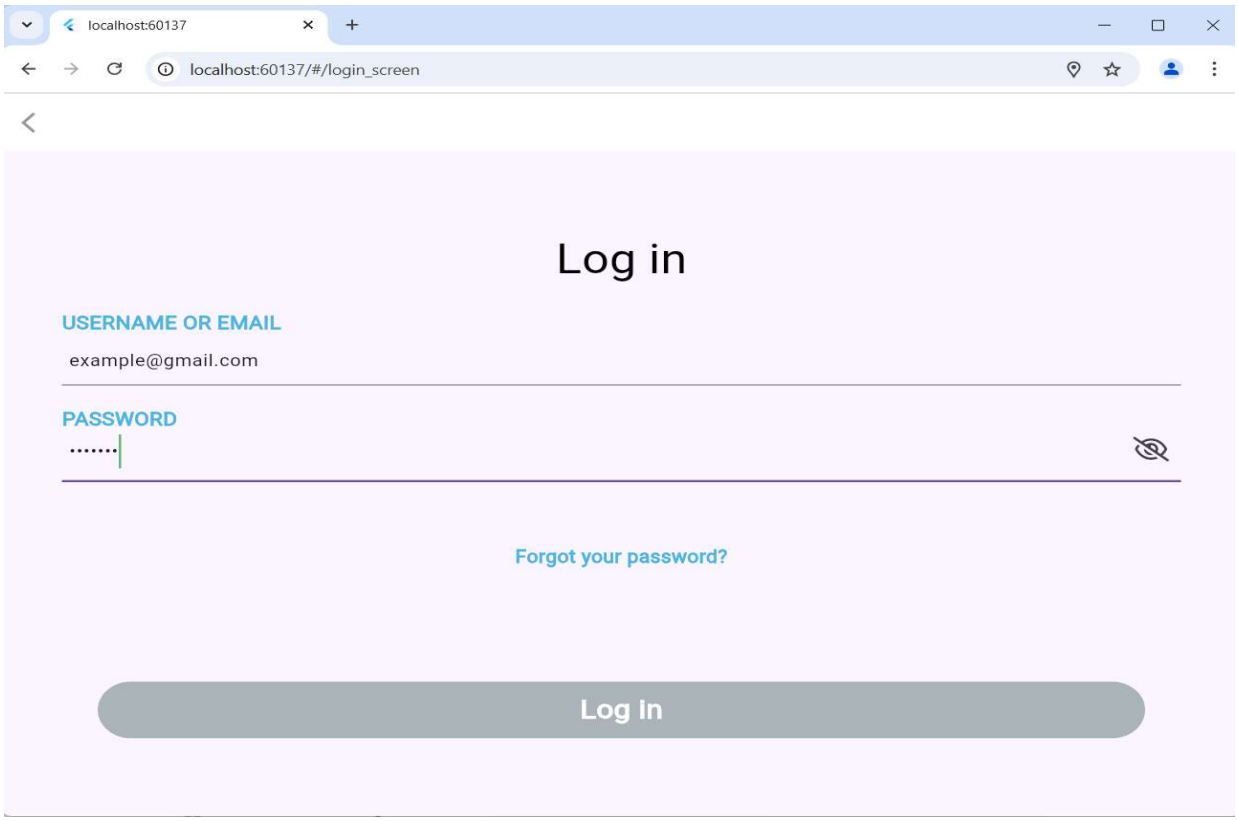
```
    body: Center(
      child:
        SingleChildScrollView(
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: [
              Text(
                "What's your name?",
                style: TextStyle(fontSize: 32, color: Colors.black),
              ),
              SizedBox(height: 20),
              CustomSnapTextField(
              label: "FIRST NAME",
                isPasswordField: false,
                autoFocus: true,
              ),
              SizedBox(height: 20),
              CustomSnapTextField(
              label: "LAST NAME",
                isPasswordField: false,
                autoFocus: true,
              ),
              SizedBox(height: 20),

              Padding(
                padding: const EdgeInsets.symmetric(horizontal: 55),
                child: RichText(
                  text: TextSpan(
                    text:
                        "By tapping Sign up & Accept, you acknowledge that you have
read the ",                                                    style: TextStyle(color:
                                                               Color(0xFFB3B7B8), fontSize:
                                                               17), children: <TextSpan>[
                                                                TextSpan(
                                                                  text: 'Privacy Policy ',
                                                                  style: TextStyle(color:
                                                                  Color(0xFF51B5E5)),
                                                                ),
                                                                TextSpan(
                                                                  text: 'and agree to the ',
                                                                  style: TextStyle(color:
                                                                  Color(0xFFB3B7B8)),
                                                                ),
                                                                TextSpan(
                                                                  text: 'Terms of service.',
                                                                  style: TextStyle(color:
                                                                  Color(0xFF51B5E5)),
```

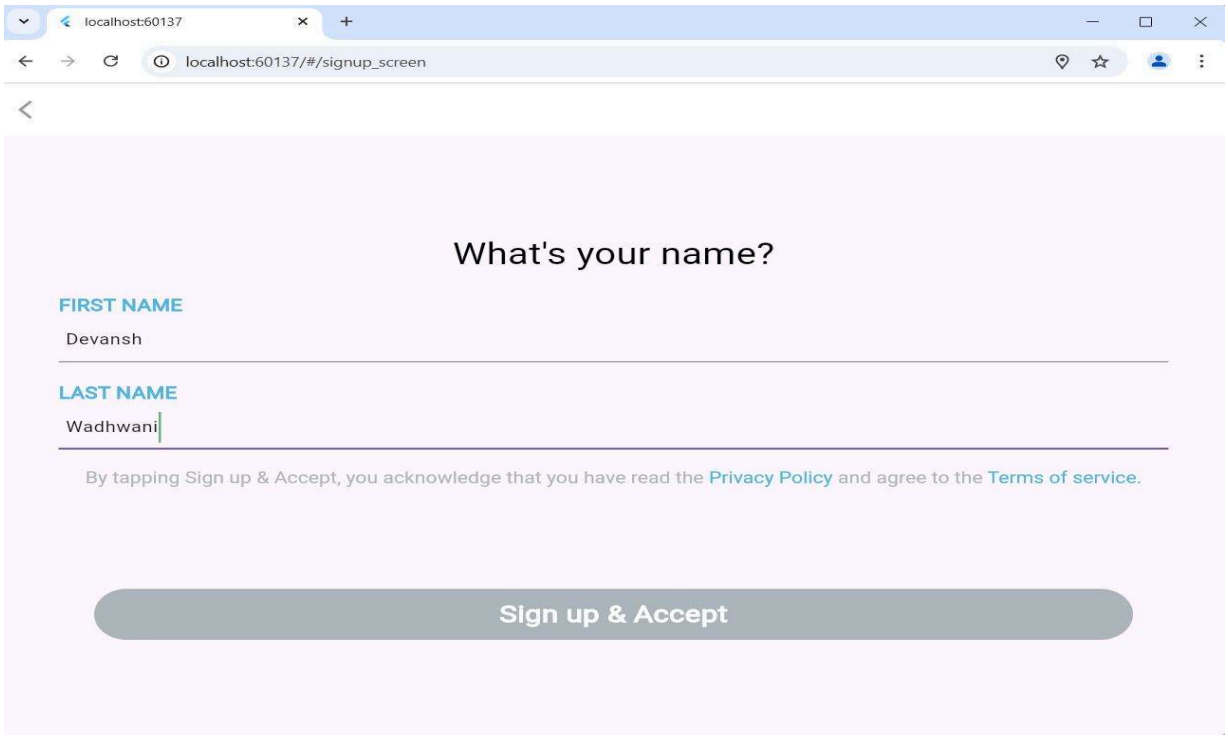
```
  ),                                                    ),
],                                                    ),
            ),

            SizedBox(height: 90),
```

```
            //Signup button
            Padding(
              padding: const EdgeInsets.symmetric(horizontal: 80),
              child: GestureDetector(
                onTap: () {
                  Navigator.pushNamed(context, '/home_screen');
                },
                child: Container(
                  margin: EdgeInsets.only(top: 20),
                  child: Text(
                    "Sign up & Accept",
                    style: TextStyle(
                    fontSize: 25,
                      color: Colors.white,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                  alignment: Alignment.center,
                  height: 55,
                  width: double.infinity,
                  decoration: BoxDecoration(
                  color: Color(0xFFADB6BD),
                    borderRadius: BorderRadius.circular(80),
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

Screenshots:


localhost:60137
localhost:60137
Log in
Sign Up


localhost:60137
localhost:60137/#/login_screen
Log in
USERNAME OR EMAIL
example@gmail.com
PASSWORD
Forgot your password?
Log in


localhost:60137
localhost:60137/#/signup_screen
What's your name?
FIRST NAME
Devansh
LAST NAME
Wadhwani
By tapping Sign up & Accept, you acknowledge that you have read the Privacy Policy and agree to the Terms of service.
Sign up & Accept

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	**63**
Name	**Diksha Abhinay Utekar**
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: 05

Name: Diksha Utekar Class:
D15A
Roll No: 63

Aim: To implement navigation, routing, and gestures in a Flutter application

Theory:

Navigation & Routing in Flutter

Navigation is the process of moving between different screens (or pages) in an app. Since Flutter follows a **widget-based architecture**, each screen is represented as a widget, and navigation is handled using a **stack-based system** (similar to how web browsers manage page history).

Navigation Approaches in Flutter:

1. **Imperative Navigation (Push-Pop Model)**
    - Uses Navigator.push() to open a new screen.
    - Uses Navigator.pop() to return to the previous screen.
    - Works like a stack (Last In, First Out - LIFO).
2. **Declarative Navigation (Named Routes & GoRouter)**
    - Uses predefined route names to navigate.
    - Helps in managing complex app navigation efficiently.
3. **Navigation with State Management**
    - Used in large-scale applications where navigation state is maintained using providers like **Provider, Riverpod, Bloc, or GetX**.
4. **Deep Linking & Dynamic Routing**
    - Allows external URLs to open specific screens within the app.
    - Useful for handling notifications and web links.

Routing in Flutter

Routing determines how users navigate through different sections of an application. It enables:

- **Defining and managing screens efficiently.**
- **Enhancing UX by ensuring smooth transitions.**

- **Passing and receiving data between screens.**

Types of Routing:

- **Basic Routing:** Manually navigating between screens using the Navigator class.
- **Named Routing:** Using a routes map to define screen names and their respective widgets.
- **On-Generate Routing:** Dynamically controlling navigation flow using conditions.
- **Nested Navigation:** Managing multiple navigation flows within tabs or bottom navigation bars.

Gestures in Flutter

Gestures allow users to interact with the app through touch-based actions, providing a more engaging user experience. Flutter has a powerful **GestureDetector** widget that helps recognize and handle different gestures.

Common Gesture Types:

- **Tap Gesture:** Used for buttons, links, and UI interactions.
- **Double Tap:** Often used for liking content or zooming.
- **Long Press:** Triggers additional options or context menus.
- **Swipe & Drag:** Used in carousels, sliders, and scrollable content.
- **Pinch & Zoom:** Common in images and maps.

Gesture Handling Techniques:

- **Using GestureDetector:** To detect gestures manually.
- **Using InkWell or InkResponse:** Provides visual feedback for tappable elements.
- **Custom Gesture Recognition:** Flutter allows combining multiple gestures using GestureRecognizer.

Importance of Navigation, Routing, and Gestures in Flutter Apps

- **User-Friendly Experience** – Navigation helps users seamlessly explore the app.
- **Efficient State Management** – Proper routing keeps the app organized.
- **Enhanced Interactivity** – Gestures make the app feel intuitive and engaging.
- **Optimized Performance** – Declarative navigation and lazy loading improve performance.
- **Platform Consistency** – Flutter's navigation system works smoothly on Android, iOS, and the web.

Code Snippet:

- **main.dart**

```
import 'package:flutter/material.dart';
import 'package:snapchat_clone_ui/screens/home_screen.dart';
import 'package:snapchat_clone_ui/screens/initial_screen.dart';
import 'package:snapchat_clone_ui/screens/login_screen.dart';
import 'package:snapchat_clone_ui/screens/signup_screen.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
    debugShowCheckedModeBanner: false,
      theme: ThemeData(primaryColor: Color(0xFF838486)),
      initialRoute: '/',
      routes: {
        '/': (context) => InitialScreen(),
        '/login_screen': (context) => LoginScreen(),
        '/signup_screen': (context) => SignupScreen(),
        '/home_screen': (context) => HomeScreen(),
      },
    );
  }
}
```

Initial_Screen.dart

```
import 'package:flutter/material.dart';
import 'package:snapchat_clone_ui/custom_widgets/custom_widgets.dart';

//Constants

const Color scaffoldColor = Color(0xFFFFFC00);
const Color loginButtonColor = Colors.white;
const Color signupButtonColor = Color(0xFF0EAEFE);

class InitialScreen extends StatefulWidget {
  @override
  _InitialScreenState createState() => _InitialScreenState();
}

class _InitialScreenState extends State<InitialScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: scaffoldColor,
      body: Stack(
        children:
          [
          Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Container(
                height: 180,
                decoration: BoxDecoration(
                  image: DecorationImage(
                    image: AssetImage("assets/images/icon.png"),
                  ),
                ),
              ),
            ],
          ),
          Center(
            child: Padding(
              padding: const EdgeInsets.symmetric(vertical: 30),
              child: Column(
                mainAxisAlignment: MainAxisAlignment.end,
                children: [
                  Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                      GestureDetector(
                        onTap: () {
```

```
  Navigator.pushNamed(context, '/login_screen');
},
child: ReusableButton(
```

```
                    btnHeight: 60.0,
                    btnWidth: 130.0,
                    btnColour: loginButtonColor,
                    btnCircularRadius: 80.0,
                    btnChild: Text(
                     "Log in",
                     style: TextStyle(
                      fontSize: 25,
                      fontWeight: FontWeight.bold,
                      color: Colors.black,
                     ),
                    ),
                   ),
                  ),
                 ),
                 SizedBox(width: 15),
                 GestureDetector(
                 onTap: () {
                    Navigator.pushNamed(context, '/signup_screen');
                  },
                  child:
                    ReusableButton(
                    btnHeight: 60.0,
                    btnWidth: 130.0,
                    btnColour: signupButtonColor,
                    btnCircularRadius: 80.0,
                    btnChild: Text(
                     "Sign Up",
                     style: TextStyle(
                     fontSize: 25,
                      fontWeight: FontWeight.bold,
                      color: Colors.white,
                     ),
                    ),
                   ),
                  ),
                 ),
                ],
               ),
              ],
             ),
            ),
           ),
          ),
         ],
        ),
       );
      }
    }
```

The app opens with a splash screen, then moves to the welcome screen. Clicking "Sign Up" takes the user to the sign-up page, and after registering, they reach the home screen. Here the app will begin. Finally, Home Screen of Snapchat will

appear.

- Login_Screen.dart

```
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:snapchat_clone_ui/custom_widgets/custom_widgets.dart';

const Falcon hiddenEye = Falcon(FontAwesomeIcons.eyeSlash);
const Falcon eye = Falcon(FontAwesomeIcons.eye);

class LoginScreen extends StatefulWidget {
  @override
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  bool _obscureText = true;
  Widget eyeStatus = hiddenEye;

  void _toggle() {
    setState(() {
      _obscureText = !_obscureText;
      if (_obscureText == false) {
      eyeStatus = eye;
      } else {
        eyeStatus = hiddenEye;
      }
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: GestureDetector(
          onTap: () {
            Navigator.pop(context);
          },
          child: Icon(Icons.arrow_back_ios, color: Colors.grey),
```

```
  ),
  elevation: 0,
  backgroundColor: Colors.white,
),
body: Center(
  child:
    SingleChildScrollView(
    child: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [
          Text(
            "Log in",
            style: TextStyle(fontSize: 40, color: Colors.black),
          ),
          SizedBox(height: 20),
          CustomSnapTextField(
            label: "USERNAME OR EMAIL",
            isPasswordField: false,
            autoFocus: true,
          ),
          SizedBox(height: 20),
          Column(
            children: [
              Container(
                alignment: Alignment.centerLeft,
                margin: EdgeInsets.symmetric(horizontal: 50),
                child: Text(
                  "PASSWORD",
                  style: TextStyle(
                    fontSize: 18,
                    fontWeight: FontWeight.bold,
                    color: Color(0xFF51B5E5),
                  ),
                ),
              ),
              Padding(
                padding: const EdgeInsets.symmetric(horizontal: 50),
                child: TextField(
                  obscureText: _obscureText,
                  autofocus: false,
                  cursorHeight: 33,
                  cursorWidth: 2,
                  decoration: InputDecoration(
```

```
suffixIcon: GestureDetector(
```

```
            onTap: () {
              _toggle();
            },
            child: eyeStatus,
          ),
          floatingLabelBehavior: FloatingLabelBehavior.never,
          contentPadding: EdgeInsets.all(6),
        ),
        cursorColor: Color(0xFF69B77D),
      ),
    ),
  ],
),
SizedBox(height: 60),
//Forgot your password
GestureDetector(
onTap: () {
   //forgot your password
  },
  child: Text(
    "Forgot your password?",
    style: TextStyle(
      fontSize: 17,
      fontWeight: FontWeight.bold,
      color: Color(0xFF51B5E5),
    ),
  ),
),
SizedBox(height: 90),

//Login button
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 80),
  child: GestureDetector(
    onTap: () {
      Navigator.pushNamed(context, '/home_screen');
    },
    child: Container(
      margin: EdgeInsets.only(top: 20),
      child: Text(
        "Log in",
        style: TextStyle(
          fontSize: 25,
          color: Colors.white,
```

```
                    fontWeight: FontWeight.bold,
                  ),
                ),
                alignment: Alignment.center,
                height: 55,
                width: double.infinity,
                decoration: BoxDecoration(
                color: Color(0xFFADB6BD),
                  borderRadius: BorderRadius.circular(80),
                ),
              ),
            ),
          ),
        ],
      ),
    ),
  ),
 ),
 );
 }
}
```

- Reels_Screen.dart

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

class ReelScreen extends StatefulWidget {
 const ReelScreen({Key? key}) : super(key: key);

 @override
 State<ReelScreen> createState() => _ReelScreenState();
}

class _ReelScreenState extends State<ReelScreen> {
 final List<String> videoUrls = [
 'assets/videos/video_1.mp4',
 'assets/videos/video_2.mp4',
 ];

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

```
    return Scaffold(
      body: PageView.builder(
        scrollDirection: Axis.vertical,
        itemCount: videoUrls.length,
        itemBuilder: (context, index) {
          return ReelVideoPlayer(videoUrl: videoUrls[index]);
        },
      ),
    );
  }
}

class ReelVideoPlayer extends StatefulWidget {
  final String videoUrl;
  const ReelVideoPlayer({Key? key, required this.videoUrl}) : super(key: key);

  @override
  State<ReelVideoPlayer> createState() => _ReelVideoPlayerState();
}

class _ReelVideoPlayerState extends State<ReelVideoPlayer> {
  late VideoPlayerController _controller;

  @override
  void initState() {
    super.initState();
    _controller = VideoPlayerController.network(widget.videoUrl)
      ..initialize().then((_) {
        setState(() {});
        _controller.play();
        _controller.setLooping(true);
      });
  }

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

  @override
  Widget build(BuildContext context) {
    return Stack(
      alignment: Alignment.bottomCenter,
```

```
      children: [
        _controller.value.isInitialized
            ? AspectRatio(
              aspectRatio: _controller.value.aspectRatio,
              child: VideoPlayer(_controller),
            )
            : const Center(child: CircularProgressIndicator()),
        Positioned(
          bottom: 20,
          right: 20,
          child: IconButton(
            icon: Icon(
              _controller.value.isPlaying ? Icons.pause : Icons.play_arrow,
              color: Colors.white,
              size: 30,
            ),
            onPressed: () {
              setState(() {
                _controller.value.isPlaying
                    ? _controller.pause()
                    : _controller.play();
              });
            },
          ),
        ),
      ],
    );
  }
}
```

- Chat_Screen.dart

```
import 'package:flutter/material.dart';
class ChatScreen extends StatefulWidget
{
  const ChatScreen({Key? key}) : super(key: key);

  @override
  State<ChatScreen> createState() => _ChatScreenState();
}
```

```
class _ChatScreenState extends State<ChatScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Column(
          children: [
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [
              Padding(
                padding: const EdgeInsets.all(8.0),
                child: Row(
                  children: [
                    CircleAvatar(
                      backgroundColor: Colors.transparent,
                      backgroundImage: AssetImage("assets/images/hero.png"),
                    ),
                    SizedBox(width: 10.0),
                    CircleAvatar(
                      backgroundColor: Colors.grey[100],
                      child: Icon(Icons.search, color: Colors.grey[700]),
                    ),
                  ],
                ),
              ),
              Text(
                "Chat",
                style: TextStyle(fontSize: 20.0, fontWeight: FontWeight.bold),
              ),
              Padding(
                padding: const EdgeInsets.all(8.0),
                child: Row(
                  children: [
                    CircleAvatar(
                      backgroundColor: Colors.grey[100],
                      child: Icon(Icons.person_add, color: Colors.grey[700]),
                    ),
                    SizedBox(width: 10.0),
                    CircleAvatar(
                      backgroundColor: Colors.grey[100],
                      child: Icon(Icons.more_horiz, color: Colors.grey[700]),
                    ),
```

```
            ],
          ),
        ),
      ],
    ),

    //list tile for chats here
    SingleChildScrollView(
    child: Column(
       children: [
        ChatTile(
          name: "Team Snapchat",
          image: NetworkImage(
            "https://us-east1-aws.api.snapchat.com/web-
capture/www.snapchat.com/discover/preview/facebook.png",
          ),
          trailing: Icon(
            Icons.chat_bubble_outline_sharp,
            color: Colors.grey[400],
          ),
          child: Row(children: [Text("Blocked")]),
        ),
        ChatTile(
          name: "Richa",
          image: AssetImage("assets/images/hero_4.png"),
          trailing: Icon(
            Icons.chat_bubble_outline_sharp,
            color: Colors.grey[400],
          ),
          child: Row(
            children: [
              Icon(Icons.square, color: Colors.red, size: 15.0),
              SizedBox(width: 5.0),
              Text("New Snap", style: TextStyle(color: Colors.red)),
              SizedBox(width: 5.0),
              Text("."),
              SizedBox(width: 5.0),
              Text("3w"),
            ],
          ),
        ),
        ChatTile(
          name: "Anurag",
          image: AssetImage("assets/images/hero_2.png"),
```

```
  trailing: Icon(
    Icons.chat_bubble_outline_sharp,
    color: Colors.grey[400],
  ),
  child: Row(
    children: [
      Icon(Icons.square, color: Colors.purple, size: 15.0),
      SizedBox(width: 5.0),
      Text(
        "New Snap",
        style: TextStyle(color: Colors.purple),
      ),
      SizedBox(width: 5.0),
      Text("."),
      SizedBox(width: 5.0),
      Text("3w"),
    ],
  ),
),
ChatTile(
  name: "Niyati",
  image: AssetImage("assets/images/hero_5.png"),
  trailing: Icon(
    Icons.chat_bubble_outline_sharp,
    color: Colors.grey[400],
  ),
  child: Row(
    children: [
      Icon(Icons.square, color: Colors.red, size: 15.0),
      SizedBox(width: 5.0),
      Text("New Snap", style: TextStyle(color: Colors.red)),
      SizedBox(width: 5.0),
      Text("."),
      SizedBox(width: 5.0),
      Text("3w"),
    ],
  ),
),
ChatTile(
  name: "Ritik",
  image: AssetImage("assets/images/hero_3.png"),
  trailing: Icon(
    Icons.camera_alt_outlined,
    color: Colors.grey[400],
```

```
  ),
  child: Row(
    children: [
      Icon(Icons.chat_bubble_outline_outlined, size: 15.0),
      SizedBox(width: 5.0),
      Text("Tap to chat"),
    ],
  ),
),
ChatTile(
  name: "Richa",
  image: AssetImage("assets/images/hero_4.png"),
  trailing: Icon(
    Icons.chat_bubble_outline_sharp,
    color: Colors.grey[400],
  ),
  child: Row(
    children: [
      Icon(Icons.square, color: Colors.red, size: 15.0),
      SizedBox(width: 5.0),
      Text("New Snap", style: TextStyle(color: Colors.red)),
      SizedBox(width: 5.0),
      Text("."),
      SizedBox(width: 5.0),
      Text("3w"),
    ],
  ),
),
ChatTile(
  name: "Anurag",
  image: AssetImage("assets/images/hero_2.png"),
  trailing: Icon(
    Icons.chat_bubble_outline_sharp,
    color: Colors.grey[400],
  ),
  child: Row(
    children: [
      Icon(Icons.square, color: Colors.purple, size: 15.0),
      SizedBox(width: 5.0),
      Text(
        "New Snap",
        style: TextStyle(color: Colors.purple),
      ),
      SizedBox(width: 5.0),
```

```
                  Text("."),
                  SizedBox(width: 5.0),
                  Text("3w"),
                ],
              ),
            ),
            ChatTile(
              name: "Niyati",
              image: AssetImage("assets/images/hero_5.png"),
              trailing: Icon(
                Icons.camera_alt_outlined,
                color: Colors.grey[400],
              ),
              child:
                Row(
                children:
                [ Icon(
                    Icons.send_outlined,
                    color:
                    Color(0XFF10ACFF), size:
                    15.0,
                  ),
                  SizedBox(width: 5.0),
                  Text("Opened"),
                  SizedBox(width: 5.0),
                  Text("."),
                  SizedBox(width: 5.0),
                  Text("3w"),
                ],
              ),
            ),
          ],
        ),
      ),
    ],
  ),
),
floatingActionButton: FloatingActionButton(
  onPressed: () {},
  backgroundColor: Color(0XFF10ACFF),
  child: Icon(Icons.edit_note_outlined),
),
);
}
```

```
}
```

```
//chatTile widget
class ChatTile extends StatelessWidget {
  final name;
  final image;
  final child;
  final trailing;

  const ChatTile({Key? key, this.image, this.name, this.child, this.trailing})
    : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Divider(height:
        3), ListTile(
          leading: CircleAvatar(
            radius: 20.0,
            backgroundColor: Colors.transparent,
            backgroundImage: image,
          ),
          trailing: trailing,
          title: Text(name, style: TextStyle(fontWeight: FontWeight.bold)),
          subtitle: child,
        ),
        Divider(height: 3),
      ],
    );
  }
}
```

- Camera_Screen.dart

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

class CameraScreen extends StatefulWidget {
  const CameraScreen({Key? key}) : super(key: key);

  @override
  State<CameraScreen> createState() => _CameraScreenState();
}
```

```
class _CameraScreenState extends State<CameraScreen> {
  CameraController? _cameraController;
  late List<CameraDescription> cameras;
  bool _isCameraInitialized = false;

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

  Future<void> _initializeCamera() async {
    cameras = await availableCameras();
    _cameraController = CameraController(cameras[0], ResolutionPreset.high);
    await _cameraController!.initialize();
    if (!mounted) return;
    setState(() => _isCameraInitialized = true);
  }

  @override
  void dispose() {
    _cameraController?.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Camera Screen")),
      body:
          _isCameraInitialized
              ? CameraPreview(_cameraController!)
              : const Center(child: CircularProgressIndicator()),
      floatingActionButton: FloatingActionButton(
      onPressed: () async {
          if (_cameraController != null &&
              _cameraController!.value.isInitialized) {
            final image = await _cameraController!.takePicture();
            ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(content: Text("Picture saved at: ${image.path}")),
            );
          }
        },
```

```
        child: const Icon(Icons.camera),
      ),
    );
  }
}
```

- Home_Screen.dart

```
import 'package:flutter/material.dart';
import 'package:snapchat_clone_ui/screens/camera_screen.dart';
import 'package:snapchat_clone_ui/screens/chat_screen.dart';
import
'package:snapchat_clone_ui/screens/location_screen.dart';
import 'package:snapchat_clone_ui/screens/reels_screen.dart';
import 'package:snapchat_clone_ui/screens/stories_screen.dart';
import 'initial_screen.dart';

class HomeScreen extends StatefulWidget {
  @override
  State<HomeScreen> createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  int _selectedIndex = 0;
  static const List<Widget> _widgetOptions = <Widget>[
    LocationScreen(),
    ChatScreen(),
    CameraScreen(),
    StoriesScreen(),
    ReelScreen(),
  ];

  void _onItemTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }

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

```
return Scaffold(
```

```
backgroundColor: Colors.white,
body: SafeArea(child: _widgetOptions[_selectedIndex]),
bottomNavigationBar: BottomNavigationBar(
  items: <BottomNavigationBarItem>[
    BottomNavigationBarItem(
    backgroundColor: Colors.black,
    icon: Icon(
        Icons.location_on_outlined,
        size: 25.0,
        color: Colors.white,
      ),
      label: "",
    ),
    BottomNavigationBarItem(
      backgroundColor: Colors.black,
      icon: Icon(
        Icons.chat_bubble_outline_rounded
        , size: 25.0,
        color: Colors.white,
      ),
      label: "",
    ),
    BottomNavigationBarItem(
      backgroundColor: Colors.black,
      icon: Icon(
      Icons.camera_alt_outlined,
      size: 25.0,
        color: Colors.white,
      ),
      label: "",
    ),
    BottomNavigationBarItem(
      backgroundColor: Colors.black,
      icon: Icon(
      Icons.group_outlined,
        size: 25.0,
        color: Color(0XFF10ACFF),
      ),
      label: "",
    ),
    BottomNavigationBarItem(
      backgroundColor: Colors.black,
      icon: Icon(
      Icons.play_arrow_outlined,
```

```
            size: 25.0,
            color: Colors.white,
          ),
          label: "",
        ),
      ],
      type: BottomNavigationBarType.fixed,
      currentIndex: _selectedIndex,
      selectedItemColor: Color(0XFF10ACFF),
      backgroundColor: Colors.black,
      onTap: _onItemTapped,
      unselectedItemColor: Colors.white,
    ),
   );
  }
}
```

- Sign_Up page.dart

```
import 'package:flutter/material.dart';
import 'package:snapchat_clone_ui/custom_widgets/custom_widgets.dart';

class SignupScreen extends StatefulWidget {
  @override
  _SignupScreenState createState() => _SignupScreenState();
}

class _SignupScreenState extends State<SignupScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: GestureDetector(
          onTap: () {
            Navigator.pop(context);
          },
          child: Icon(Icons.arrow_back_ios, color: Colors.grey),
        ),
        elevation: 0,
        backgroundColor: Colors.white,
      ),
      body: Center(
```

```
      child:
       SingleChildScrollView(
       child: Center(
        child: Column(
         mainAxisAlignment: MainAxisAlignment.spaceEvenly,
         children: [
          Text(
           "What's your name?",
           style: TextStyle(fontSize: 32, color: Colors.black),
          ),
          SizedBox(height: 20),
          CustomSnapTextField(
          label: "FIRST NAME",
           isPasswordField: false,
           autoFocus: true,
          ),
          SizedBox(height: 20),
          CustomSnapTextField(
          label: "LAST NAME",
           isPasswordField: false,
           autoFocus: true,
          ),
          SizedBox(height: 20),

          Padding(
           padding: const EdgeInsets.symmetric(horizontal: 55),
           child: RichText(
            text: TextSpan(
             text:
               "By tapping Sign up & Accept, you acknowledge that you have
read the ",                                                    style: TextStyle(color:
                                                               Color(0xFFB3B7B8), fontSize:
                                                               17), children: <TextSpan>[
                                                                TextSpan(
                                                                 text: 'Privacy Policy ',
                                                                 style: TextStyle(color:
                                                                 Color(0xFF51B5E5)),
                                                                ),
                                                                TextSpan(
                                                                 text: 'and agree to the ',
                                                                 style: TextStyle(color:
                                                                 Color(0xFFB3B7B8)),
                                                                ),
                                                                TextSpan(
```

```
  text: 'Terms of service.',
  style: TextStyle(color:
  Color(0xFF51B5E5)),
),
```

```
                ],
              ),
            ),
          ),

          SizedBox(height: 90),

          //Signup button
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 80),
            child: GestureDetector(
              onTap: () {
                Navigator.pushNamed(context, '/home_screen');
              },
              child: Container(
                margin: EdgeInsets.only(top: 20),
                child: Text(
                  "Sign up & Accept",
                  style: TextStyle(
                  fontSize: 25,
                    color: Colors.white,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                alignment: Alignment.center,
                height: 55,
                width: double.infinity,
                decoration: BoxDecoration(
                color: Color(0xFFADB6BD),
                  borderRadius: BorderRadius.circular(80),
                ),
              ),
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```

- Stories_Screen.dart

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

class StoriesScreen extends StatefulWidget {
  const StoriesScreen({Key? key}) : super(key: key);

  @override
  State<StoriesScreen> createState() => _StoriesScreenState();
}

class _StoriesScreenState extends State<StoriesScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child:
          SingleChildScrollView(
          child: Column(
            children: [
              Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  Padding(
                    padding: const EdgeInsets.all(8.0),
                    child: Row(
                      children: [
                        CircleAvatar
                        (
                          backgroundColor: Colors.transparent,
                          backgroundImage: AssetImage("assets/images/hero.png"),
                        ),
                        SizedBox(width: 10.0),
                        CircleAvatar(
                          backgroundColor: Colors.grey[100],
                          child: Icon(Icons.search, color: Colors.grey[700]),
                        ),
                      ],
                    ),
                  ),
                  Text(
                    "Stories",
```

```
style: TextStyle(
  fontSize: 20.0,
```

```
        fontWeight: FontWeight.bold,
      ),
    ),
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: Row(
        children: [
          CircleAvatar
          (
            backgroundColor: Colors.grey[100],
            child: Icon(
              Icons.person_add,
              color: Colors.grey[700],
            ),
          ),
          SizedBox(width: 10.0),
          CircleAvatar(
            backgroundColor: Colors.grey[100],
            child: Icon(
              Icons.more_horiz,
              color: Colors.grey[700],
            ),
          ),
        ],
      ),
    ),
  ],
),
SizedBox(height: 30.0),

//Friends section
Container(
  padding: EdgeInsets.symmetric(horizontal: 20.0),
  child: Column(
    children: [
      Container(
        alignment: Alignment.topLeft,
        child: Text(
          "Friends",
          style: TextStyle(
          fontSize: 18.0,
            fontWeight: FontWeight.bold,
          ),
        ),
```

),

```
      SizedBox(height: 20.0),
      Container(
        height: 140,
        child: ListView(
          scrollDirection: Axis.horizontal,
          children: [
            Row(
              children:
              [
                storyBubble(
                  name: "Anurag",
                  image: AssetImage("assets/images/hero_2.png"),
                ),
                SizedBox(width: 15.0),
                storyBubble(
                  name: "Richa",
                  image: AssetImage("assets/images/hero_4.png"),
                ),
                SizedBox(width: 15.0),
                storyBubble(
                  name: "Niyati",
                  image: AssetImage("assets/images/hero_5.png"),
                ),
                SizedBox(width: 15.0),
                storyBubble(
                  name: "Ritik",
                  image: AssetImage("assets/images/hero_3.png"),
                ),
                SizedBox(width: 15.0),
                storyBubble(
                  name: "Niyati",
                  image: AssetImage("assets/images/hero_5.png"),
                ),
              ],
            ),
          ],
        ),
      ),
    ],
  ),
),

//Subscriptions section
Container(
```

```
padding: EdgeInsets.symmetric(horizontal: 20.0),
```

```
            child: Column(
              children: [
              Container(
                alignment: Alignment.topLeft,
                child: Row(
                  children:
                    [ Text(
                      "Subscriptions"
                      ,                style:
                      TextStyle(
                      fontSize: 17.0,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                    SizedBox(width: 5.0),
                    Icon(Icons.arrow_forward_ios, size: 15.0),
                  ],
                ),
              ),
              SizedBox(height: 20.0),
              Container(
                height: 200.0,
                child: ListView(
                  scrollDirection: Axis.horizontal,
                  children: [
                    Row(
                      children:
                      [
                        subscriptionTile(
                          name: "Kundu",
                          image: Image.network(
                            "https://c4.wallpaperflare.com/wallpaper/923/727/796/anime-
digital-art-artwork-2d-portrait-display-hd-wallpaper-preview.jpg",
                            height: 200.0,
                          ),
                        ),
                        SizedBox(width: 10.0),
                        subscriptionTile(
                        name: "Tumami",
                          image: Image.network(
                            "https://images.pexels.com/photos/9410606/pexels-photo-
9410606.jpeg?cs=srgb&dl=pexels-zetong-li-9410606.jpg&fm=jpg",
                            height: 200.0,
                          ),
```

```
),
SizedBox(width: 10.0),
```

```
                    subscriptionTile(
                      name: "Jan Goldz",
                      image: Image.network(
                        "https://c0.wallpaperflare.com/preview/303/473/216/man-
standing-on-mountain-during-sunset.jpg",
                        height: 200.0,
                      ),
                    ),
                    SizedBox(width: 10.0),
                    subscriptionTile(
                    name: "Bastrop",
                      image: Image.network(
                        "https://c4.wallpaperflare.com/wallpaper/367/257/149/anime-
anime-girls-digital-art-artwork-2d-hd-wallpaper-preview.jpg",
                        height: 200.0,
                      ),
                    ),
                    SizedBox(width: 10.0),
                    subscriptionTile(
                    name: "Mulessa",
                      image: Image.network(
                        "https://wallpapercave.com/wp/wp2722942.jpg",
                        height: 200.0,
                      ),
                    ),
                  ],
                ),
              ],
            ),
          ),
        ),

        //Discover Section
        Container(
          padding: EdgeInsets.symmetric(horizontal: 20.0, vertical: 20.0),
          child: Column(
            children: [
              Container(
                alignment: Alignment.topLeft,
                child: Text(
                  "Discover",
                  style: TextStyle(
```

```
                    fontSize: 17.0,
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ),
              ListView(
                shrinkWrap: true,
                children: [
                Column(
                  children: [
                    Row(
                    children: [
                      Expanded(
                        flex: 2,
                        child: DiscoverTile(
                          name: "Weird Mud Games",
                          image: Image.network(

"https://c4.wallpaperflare.com/wallpaper/367/257/149/anime-anime-girls-digital-art-artwork-2d-hd-wallpaper-preview.jpg",
                            height: 380,
                          ),
                        ),
                      ),
                      SizedBox(width: 8.0),
                      Expanded(
                        flex: 2,
                        child: DiscoverTile(
                          name: "Mulessa",
                          image: Image.network(
                            "https://wallpaperaccess.com/full/1559254.png",
                            height: 380.0,
                          ),
                        ),
                      ),
                    ],
                  ),
                ],
              ),
              SizedBox(height: 0.0),
              Column(
                children: [
                  Row(
                  children: [
```

```
                        Expanded(
                          flex: 2,
                          child: DiscoverTile(
                            name: "Weird Mud Games",
                            image: Image.network(

"https://c4.wallpaperflare.com/wallpaper/367/257/149/anime-anime-girls-digital-art-artwork-2d-hd-wallpaper-preview.jpg",
                              height: 380,
                            ),
                          ),
                        ),
                        SizedBox(width: 8.0),
                        Expanded(
                          flex: 2,
                          child: DiscoverTile(
                            name: "Mulessa",
                            image: Image.network(
                              "https://wallpaperaccess.com/full/1559254.png",
                              height: 380.0,
                            ),
                          ),
                        ),
                      ],
                    ),
                  ],
                ),
              ],
            ),
          ),
        ],
      ),
    ),
  );
 }
}

class subscriptionTile extends StatelessWidget {
 final name;
 final image;
 const subscriptionTile({Key? key, this.name, this.image}) : super(key: key);
```

```
  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Stack(
        children: [
            image,
            Positioned(
            bottom: 2,
            child: Padding(
              padding: const EdgeInsets.symmetric(
                horizontal: 8.0,
                vertical: 2.0,
              ),
              child: Text(
                name,
                style: TextStyle(
                  color: Colors.white,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ),
          ),
        ),
      ],
    ),
  ],
    );
  }
}

class DiscoverTile extends StatelessWidget {
  final name;
  final image;
  const DiscoverTile({Key? key, this.name, this.image}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Stack(
        children: [
            image,
            Positioned(
```

```
            bottom: 30,
            child: Padding(
              padding: const EdgeInsets.symmetric(
                horizontal: 8.0,
                vertical: 2.0,
              ),
              child: Text(
                name,
                style: TextStyle(
                  color: Colors.white,
                  fontWeight: FontWeight.bold,
                  fontSize: 20.0,
                ),
              ),
            ),
          ),
        ],
      ),
    ],
   );
  }
}

class storyBubble extends StatelessWidget {
  final name;
  final image;
  const storyBubble({Key? key, this.image, this.name}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        CircleAvatar(
        radius: 45.0,
          backgroundColor: Colors.purple,
          child: CircleAvatar(
            radius: 43.0,
            backgroundColor: Colors.white,
            child: CircleAvatar(backgroundImage: image, radius: 40.0),
          ),
        ),
        SizedBox(height: 10.0),
        Text(name, style: TextStyle(fontWeight: FontWeight.w500)),
      ],
```

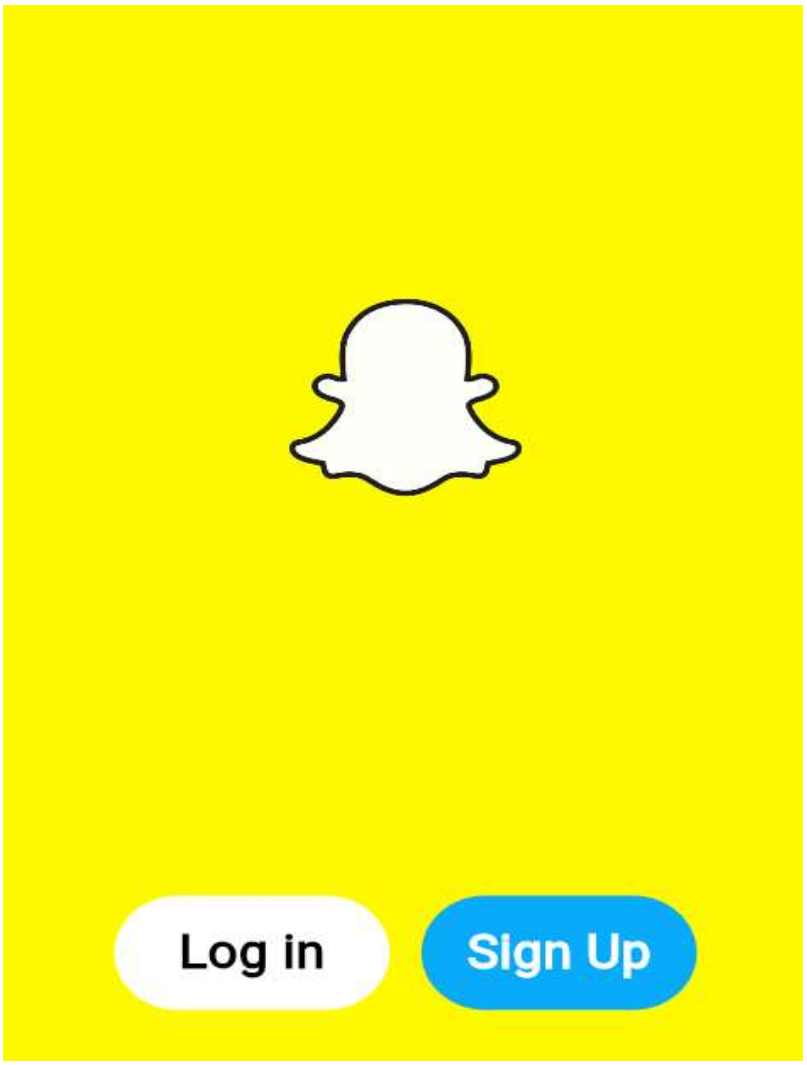
```
    );
  }
}
```

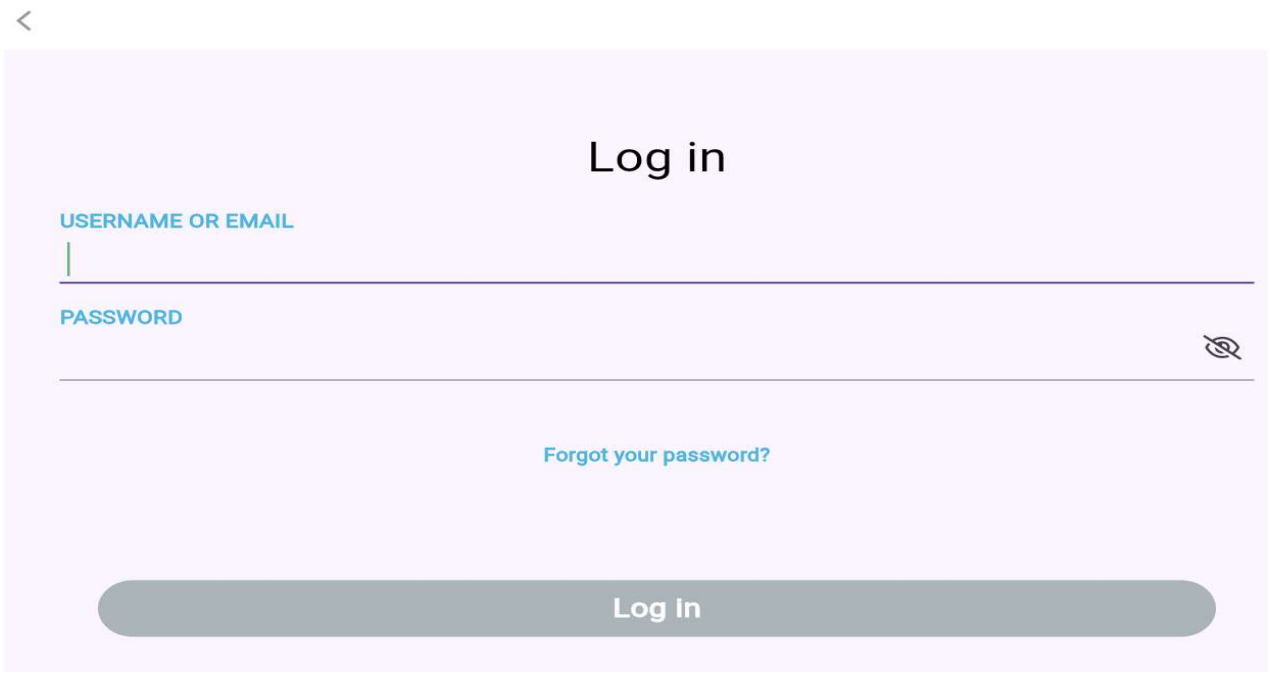
- Location_Screen.dart

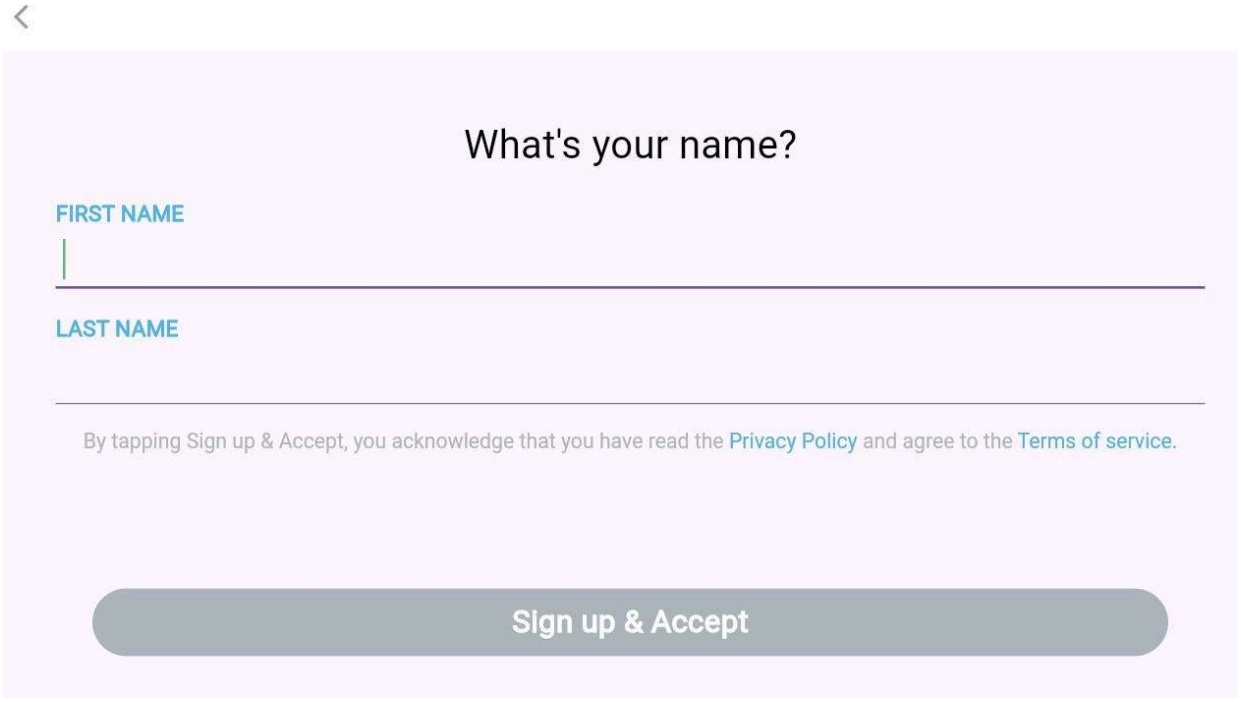
```
import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:geolocator/geolocator.dart';
import 'package:permission_handler/permission_handler.dart';

class LocationScreen extends StatefulWidget {
  const LocationScreen({Key? key}) : super(key: key);

  @override
  State<LocationScreen> createState() => _LocationScreenState();
}

class _LocationScreenState extends State<LocationScreen> {
  GoogleMapController? _mapController;
  LatLng _currentPosition = const LatLng(37.7749, -122.4194); // Default to SF
  Set<Marker> _markers = {};

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

  Future<void> _getCurrentLocation() async {
    var status = await Permission.location.request();
    if (status.isGranted) {
      Position position = await Geolocator.getCurrentPosition(
        desiredAccuracy: LocationAccuracy.high,
      );
      setState(() {
        _currentPosition = LatLng(position.latitude, position.longitude);
        _markers.add(
          Marker(
            markerId: const MarkerId("currentLocation"),
            position: _currentPosition,
```

```
          infoWindow: const InfoWindow(title: "You"),
        ),
      );
    });

_mapController?.animateCamera(CameraUpdate.newLatLng(_currentPosition));
   }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: GoogleMap(
        initialCameraPosition:
        CameraPosition( target:
        _currentPosition,
          zoom: 14,
        ),
        myLocationEnabled: true,
        myLocationButtonEnabled: true,
        markers: _markers,
        onMapCreated: (controller) {
          _mapController = controller;
        },
      ),
    );
  }
}
```

Screenshot:




Log in
USERNAME OR EMAIL
PASSWORD
Forgot your password?
Log in

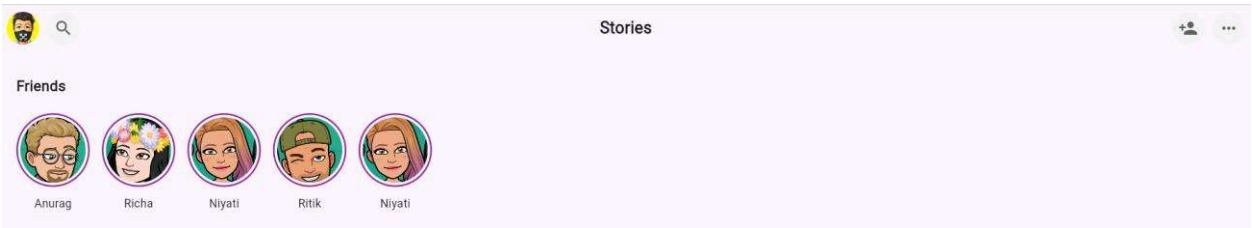

What's your name?
FIRST NAME
LAST NAME
By tapping Sign up & Accept, you acknowledge that you have read the Privacy Policy and agree to the Terms of service.
Sign up & Accept


Chat
Team Snapchat
Blocked
Richa
New Snap . 3w
Anurag
New Snap . 3w
Niyati
New Snap . 3w
Ritik
Tap to chat
Richa
New Snap . 3w
Anurag
New Snap . 3w
Niyati
Opened . 3w


localhost:58929
localhost:58929/#/home_screen
Camera Screen

Reels_Screen :

Stories_Page :



MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	**63**
Name	**Diksha Abhinay Utekar**
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:	

D

EXPERIMENT NO: - 06

Name:- Diksha Utekar **Class:-** D15A **Roll:No: -** 63

AIM: - To connect Flutter UI with Firebase database.

Theory: -

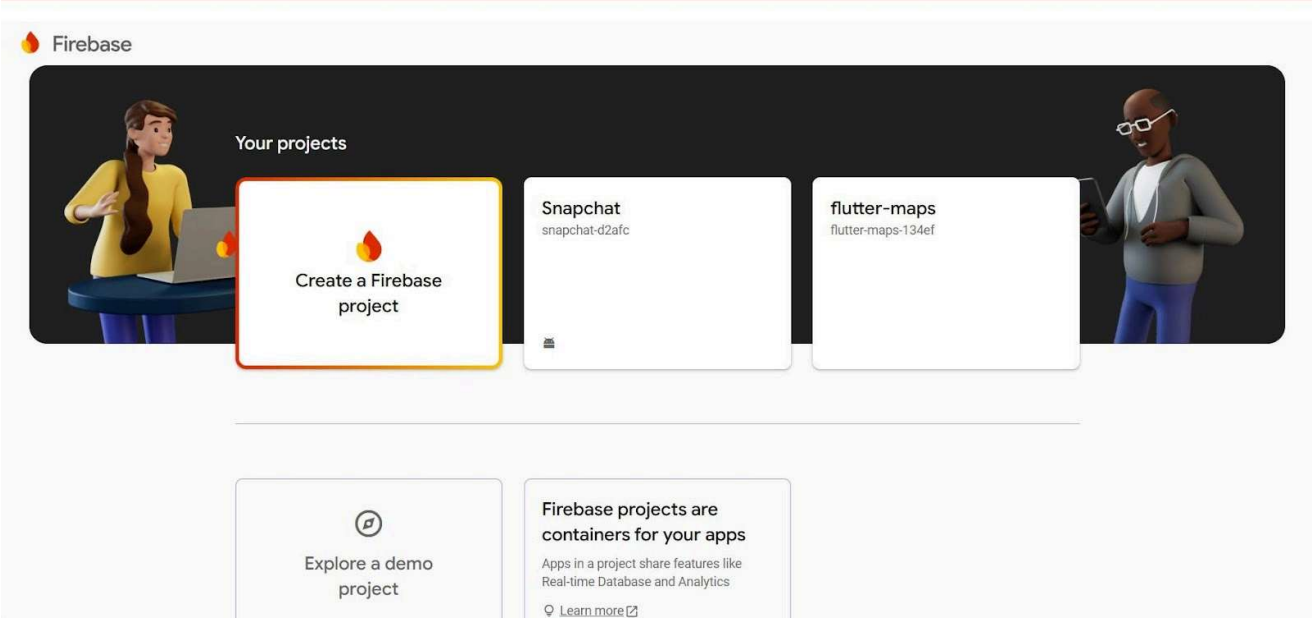
Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. Firebase, a Backend-as-a-Service (BaaS) platform, provides real-time database, authentication, and cloud storage services, making it a powerful backend 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**

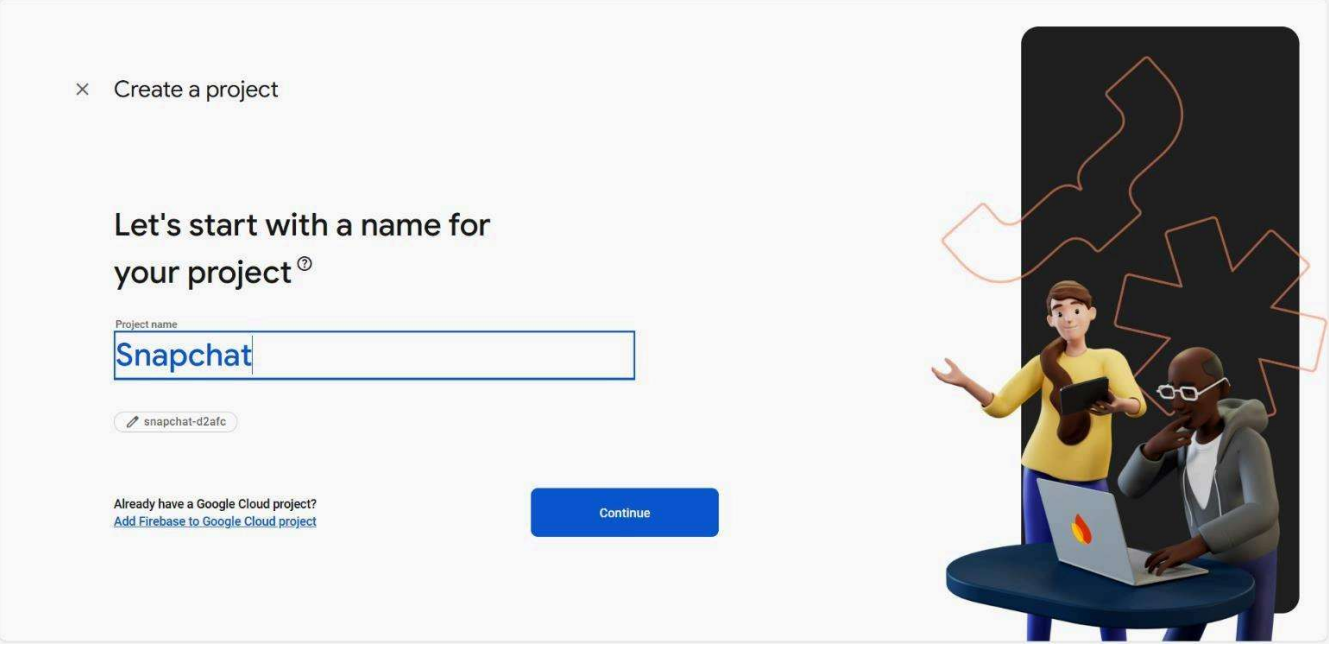
Step 1:

1.1) Go to Firebase Console and Create a Firebase Project



D

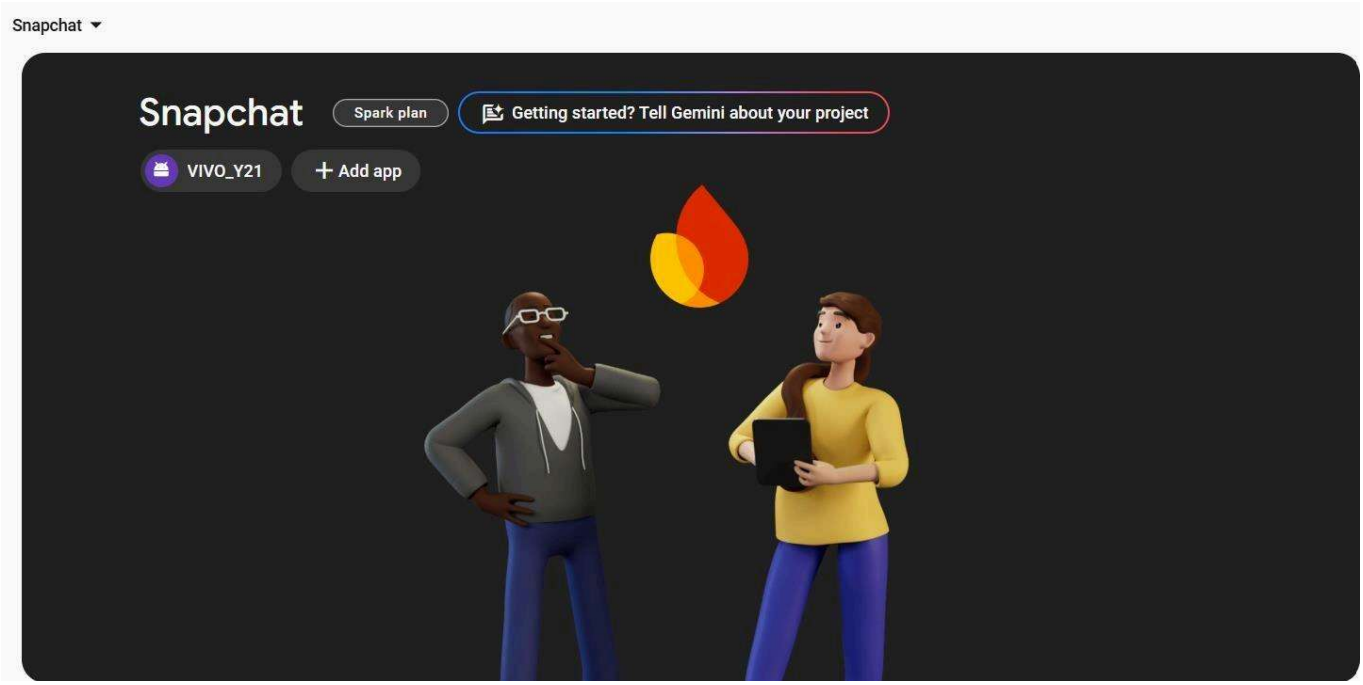
1.2) Click on Create a Project and give it a suitable name.



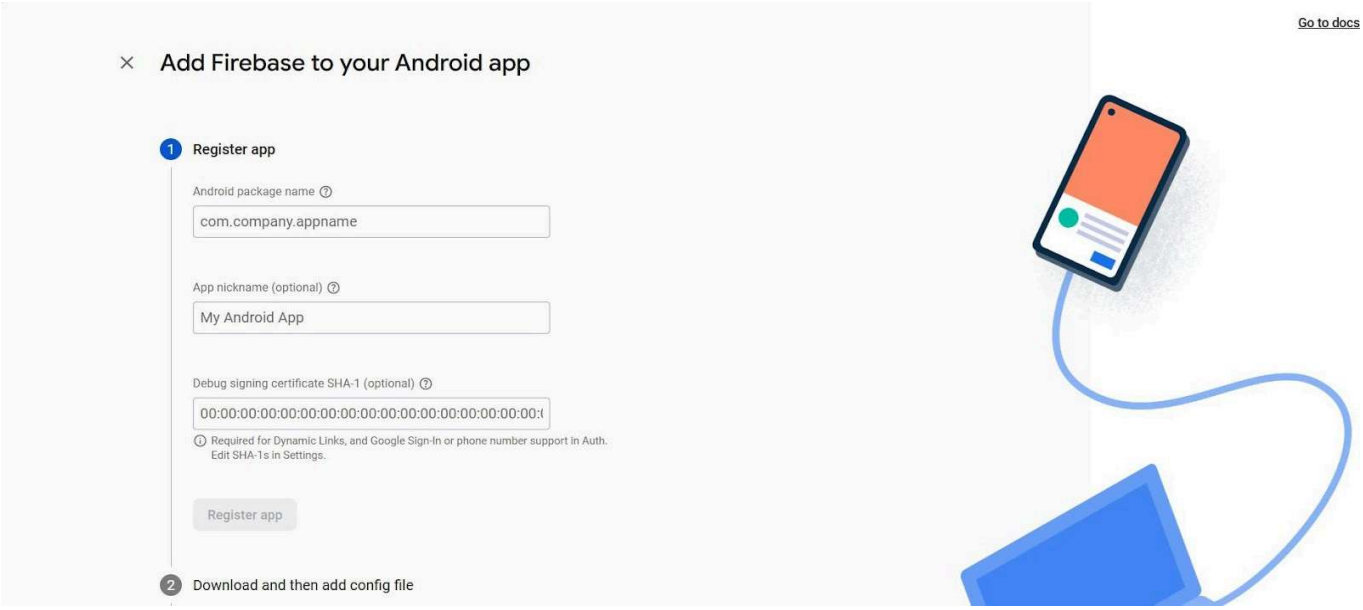
D

Step 2:- Add Firebase to Your Flutter App

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

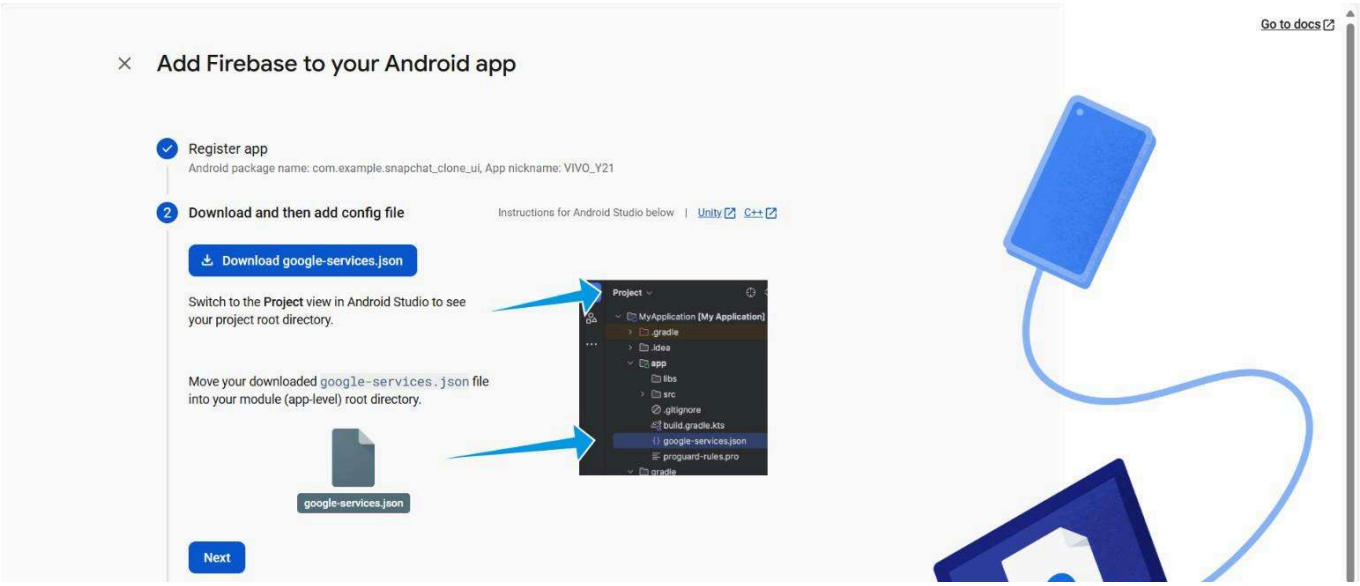


2.2) Register your app with a unique package name (found in android/app/build.gradle for Android).



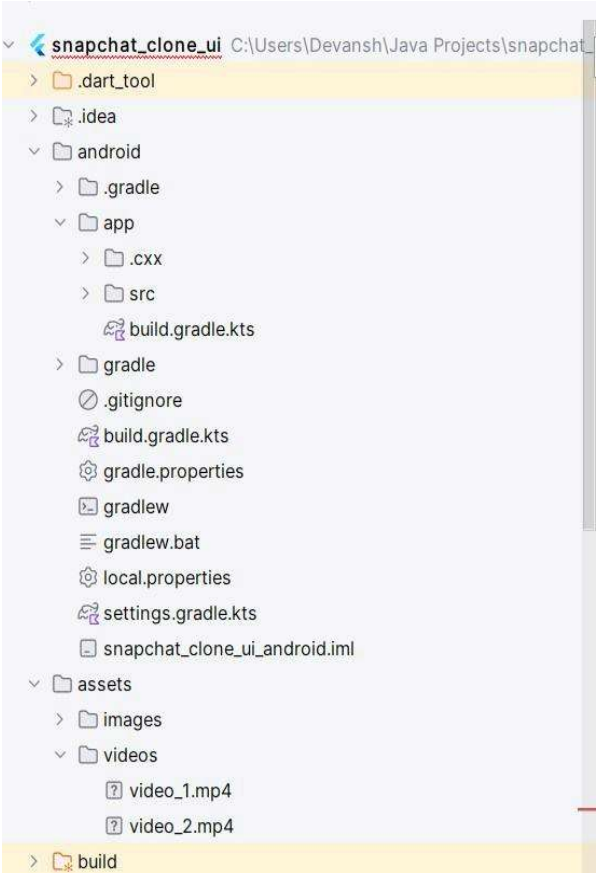
D

2.3) Download the google-services.json (for Android) & place the JSON file inside android/app/ directory.



2.4) Add Firebase SDK dependencies to android/build.gradle

D



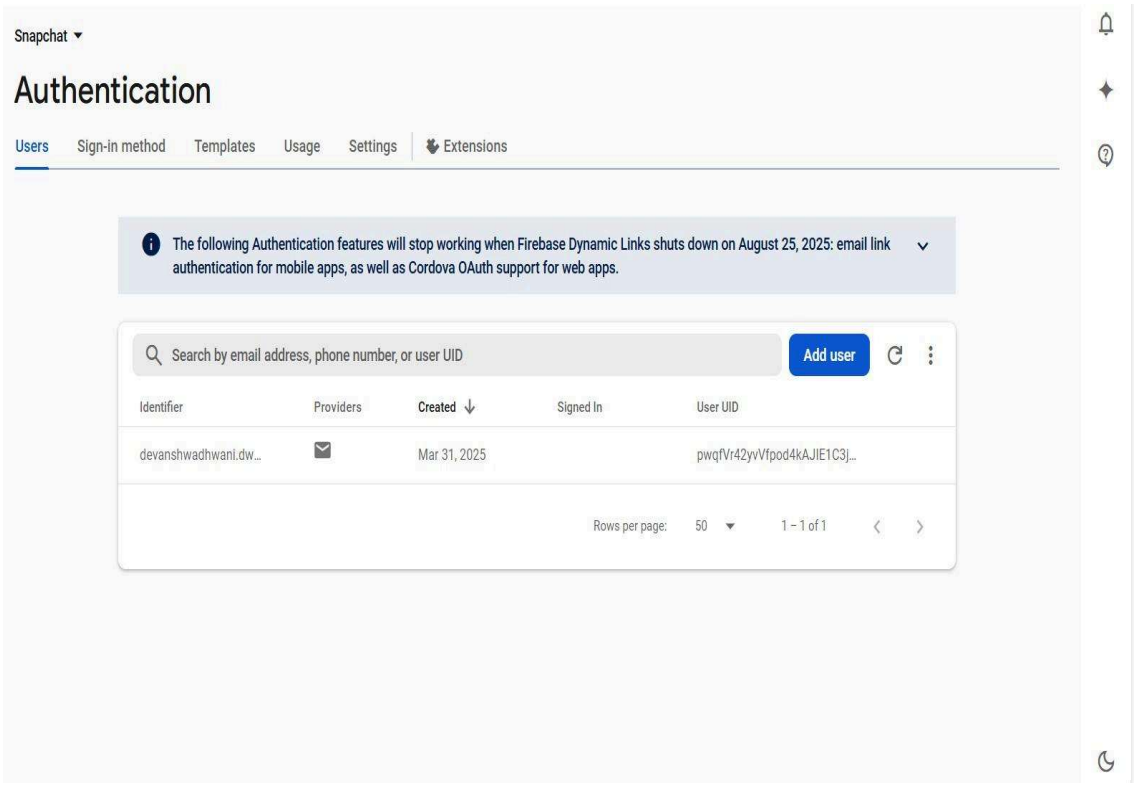
Step 3: - Add Firebase Authentication to Your App

3.1) Add Firebase Authentication Dependencies

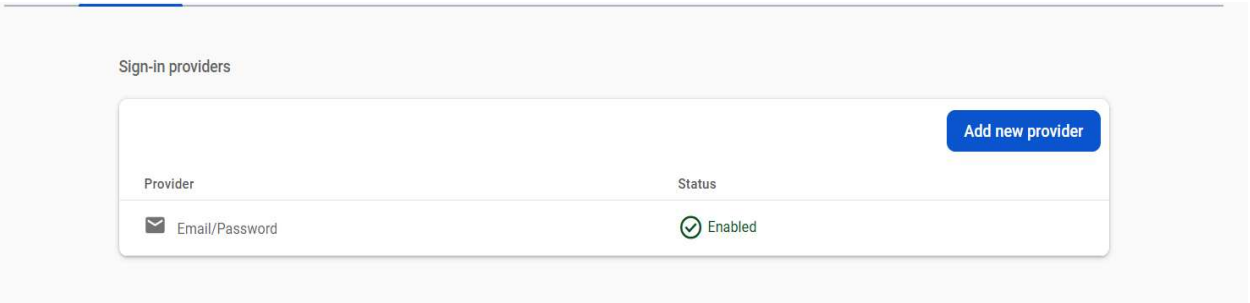
```
firebase_core: ^3.12.1
flutter_web_auth: ^0.6.0
firebase_auth: ^5.5.1
http: ^1.3.0
```

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

D


Snapchat
Authentication
Users
Sign-in method
Templates
Usage
Settings
Extensions
The following Authentication features will stop working when Firebase Dynamic Links shuts down on August 25, 2025: email link authentication for mobile apps, as well as Cordova OAuth support for web apps.
Search by email address, phone number, or user UID
Add user
Identifier
Providers
Created
Signed In
User UID
devanshwadhwani.dw...
Mar 31, 2025
pwqfVr42yvVfpod4kAJIE1C3j...
Rows per page:
50
1 – 1 of 1

D



3.3) Implement Authentication in Flutter Modify main.dart

```
import 'package:flutter/material.dart';
import 'package:snapchat_clone_ui/screens/home_screen.dart';
import 'package:snapchat_clone_ui/screens/initial_screen.dart';
import 'package:snapchat_clone_ui/screens/login_screen.dart';
import 'package:snapchat_clone_ui/screens/signup_screen.dart';
import ‘package:firebase_core’

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
    debugShowCheckedModeBanner: false,
      theme: ThemeData(primaryColor: Color(0xFF838486)),
      initialRoute: '/',
      routes: {
        '/': (context) => InitialScreen(),
        '/login_screen': (context) => LoginScreen(),
        '/signup_screen': (context) =>
        SignupScreen(), '/home_screen': (context) =>
        HomeScreen(),
      },
    );
  }
}
```

Step 4: -Configure Firebase Realtime Database

4.1) Go to Firebase Console → Realtime Database.

4.2) Click **Create Database** → Choose location → Set rules (for development, set read/write to true).

4.3) Click **Publish**.

D

- Code:

- Login_Screen.dart

```
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:snapchat_clone_ui/custom_widgets/custom_widgets.dart';

const FaIcon hiddenEye = FaIcon(FontAwesomeIcons.eyeSlash);
const FaIcon eye = FaIcon(FontAwesomeIcons.eye);

class LoginScreen extends StatefulWidget {
  @override
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  bool _obscureText = true;
  Widget eyeStatus = hiddenEye;

  void _toggle() {
    setState(() {
      _obscureText = !_obscureText;
      if (_obscureText == false) {
      eyeStatus = eye;
      } else {
        eyeStatus = hiddenEye;
      }
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: GestureDetector(
          onTap: () {
          Navigator.pop(context);
          },
          child: Icon(Icons.arrow_back_ios, color: Colors.grey),
        ),
        elevation: 0,
        backgroundColor: Colors.white,
      ),
      body: Center(
```

D

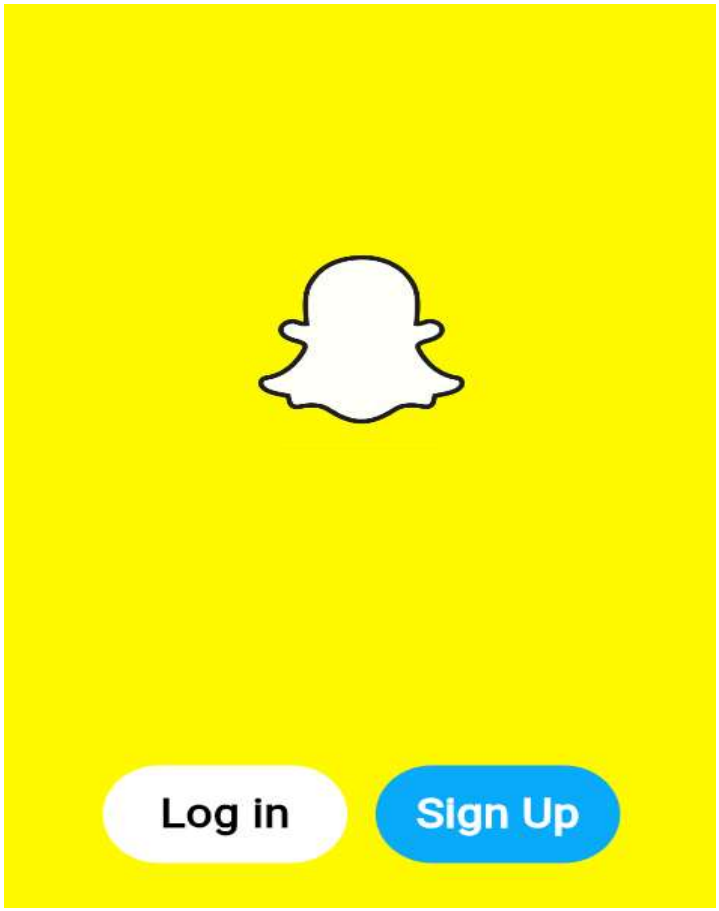
```
child: SingleChildScrollView(
  child: Center(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: [
        Text(
          "Log
          in",
          style: TextStyle(fontSize: 40, color: Colors.black),
        ),
        SizedBox(height: 20),
        CustomSnapTextField(
          label: "USERNAME OR EMAIL",
          isPasswordField: false,
          autoFocus: true,
        ),
        SizedBox(height: 20),
        Column(
          children: [
            Container(
              alignment: Alignment.centerLeft,
              margin: EdgeInsets.symmetric(horizontal: 50),
              child: Text(
                "PASSWORD",
                style: TextStyle(
                  fontSize: 18,
                  fontWeight: FontWeight.bold,
                  color: Color(0xFF51B5E5),
                ),
              ),
            ),
            Padding(
              padding: const EdgeInsets.symmetric(horizontal: 50),
              child: TextField(
                obscureText: _obscureText,
                autofocus: false,
                cursorHeight: 33,
                cursorWidth: 2,
                decoration: InputDecoration(
                  suffixIcon: GestureDetector(
                  onTap: () {
                      _toggle();
                    },
                    child: eyeStatus,
                  ),
                  floatingLabelBehavior: FloatingLabelBehavior.never,
                  contentPadding: EdgeInsets.all(6),
                ),
                cursorColor: Color(0xFF69B77D),
              ),
            ),
```

],

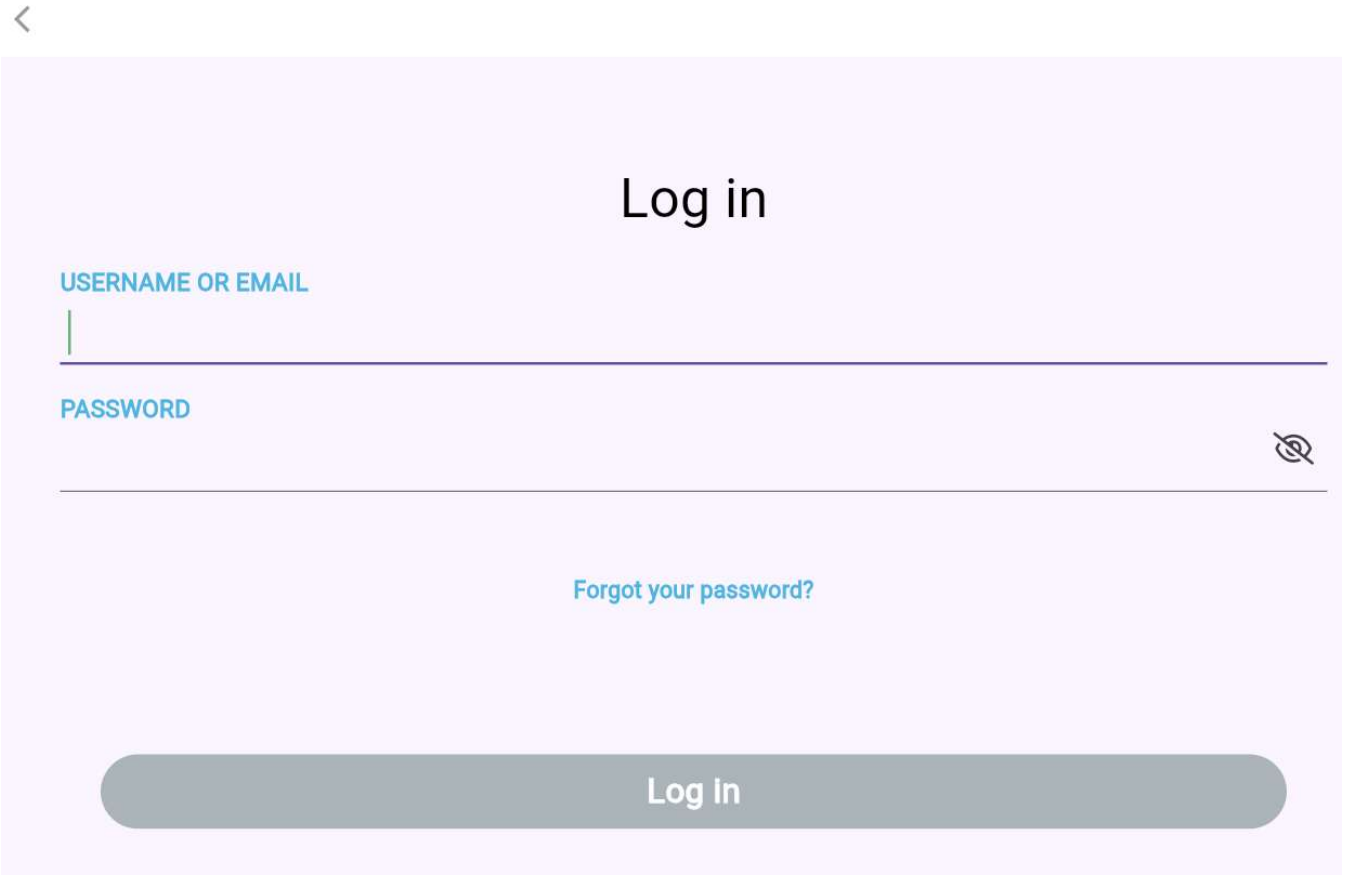
D

```
            ),
            SizedBox(height: 60),
            //Forgot your password
            GestureDetector(
            onTap: () {
               //forgot your password
              },
              child: Text(
                "Forgot your password?",
                style: TextStyle(
                fontSize: 17,
                  fontWeight: FontWeight.bold,
                  color: Color(0xFF51B5E5),
                ),
              ),
            ),
            SizedBox(height: 90),

            //Login button
            Padding(
              padding: const EdgeInsets.symmetric(horizontal: 80),
              child: GestureDetector(
                onTap: () {
                  Navigator.pushNamed(context, '/home_screen');
                },
                child: Container(
                  margin: EdgeInsets.only(top: 20),
                  child: Text(
                    "Log in",
                    style: TextStyle(
                      fontSize: 25,
                      color: Colors.white,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                  alignment: Alignment.center,
                  height: 55,
                  width: double.infinity,
                  decoration: BoxDecoration(
                  color: Color(0xFFADB6BD),
                    borderRadius: BorderRadius.circular(80),
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
```

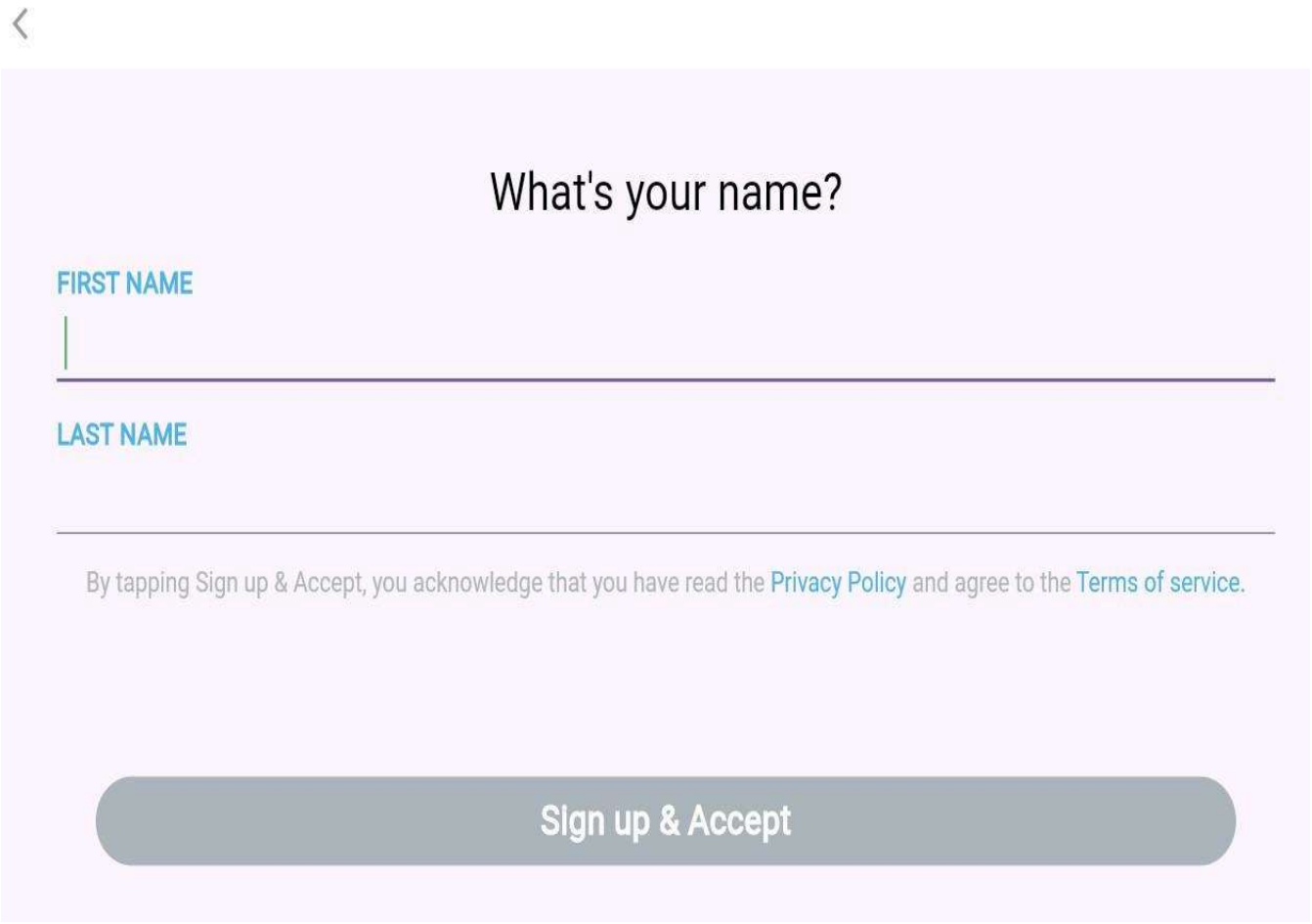
D



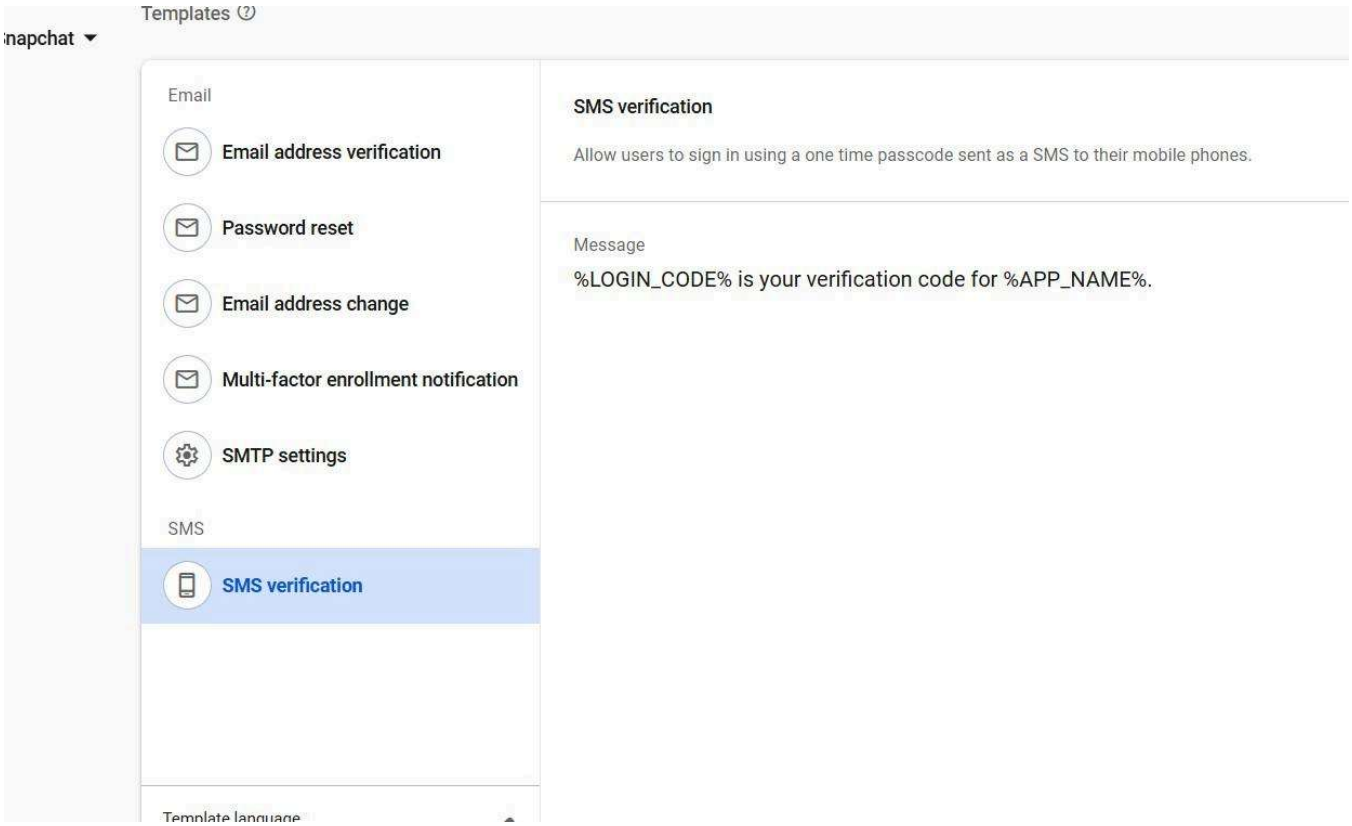
D


Log in
USERNAME OR EMAIL
PASSWORD
Forgot your password?
Log in

D


What's your name?
FIRST NAME
LAST NAME
By tapping Sign up & Accept, you acknowledge that you have read the Privacy Policy and agree to the Terms of service.
Sign up & Accept

D


napchat
Templates
Email
Email address verification
Password reset
Email address change
Multi-factor enrollment notification
SMTP settings
SMS
SMS verification
Template language
SMS verification
Allow users to sign in using a one time passcode sent as a SMS to their mobile phones.
Message
%LOGIN_CODE% is your verification code for %APP_NAME%.

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.	**63**
Name	**UTEKAR DIKSHA ABHINAY**
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

Experiment No. 7

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

Theory:

- **Regular Web App**

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

- **Progressive Web App**

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

- **Difference between PWAs vs. Regular Web Apps:**

  1. **Native Experience:** Though a PWA runs on web technologies (HTML, CSS, JavaScript) like a Regular web app, it gives user experience like a native mobile application. It can use most native device features, including push notifications, without relying on the browser or any other entity. It offers a seamless and integrated user experience that it is quite tough for one to differentiate between a PWA and a Native application by considering its look and feel.

  2. **Ease of Access:** Unlike other mobile apps, PWAs do not demand longer download time and make memory space available for installing the applications. The PWAs can be shared and installed by a link, which cuts down the number of steps to install and use. These applications can easily keep an app icon on the user's home screen, making the app easily accessible to the users and helps the brands remain in the users' minds, and improving the chances of interaction.

  3. **Faster Services:** PWAs can cache the data and serve the user with text stylesheets, images, and other web content even before the page loads completely. This lowers the waiting time for the end-users and helps the brands improve the user engagement and retention rate, which eventually adds value to their business.

  4. **Engaging Approach:** As already shared, the PWAs can employ push notifications and other native device features more efficiently. Their interaction does not depend on the browser user uses. This eventually improves the chances of notifying the user regarding your services, offers, and other options related to your brand and keeping them hooked to your brand.

5. **Updated Real:** Time Data Access: Another plus point of PWAs is that these apps get updated on their own. They do not demand the end-users to go to the App Store or other such platforms to download the update and wait until installed.

6. **Discoverable**: PWAs reside in web browsers. This implies higher chances of optimizing them as per the Search Engine Optimization (SEO) criteria and improving the Google rankings like that in websites and other web apps.

7. **Lower Development Cost:** Progressive web apps can be installed on the user device like a native device, but it does not demand submission on an App Store. This makes it far more cost-effective than native mobile applications while offering the same set of functionalities.

- **The main features are:**

1. Progressive — They work for every user, regardless of the browser chosen because they are built at the base with progressive improvement principles.
2. Responsive — They adapt to the various screen sizes: desktop, mobile, tablet, or dimensions that can later become available.
3. Updated — Information is always up-to-date thanks to the data update process offered by service workers.
4. Secure — Exposed over HTTPS protocol to prevent the connection from displaying information or altering the contents.
5. Searchable — They are identified as “applications” and are indexed by search engines.
6. Reactivable — Make it easy to reactivate the application thanks to capabilities such as web notifications.
7. Installable — They allow the user to “save” the apps that he considers most useful with the corresponding icon on the screen of his mobile terminal (home screen) without having to face all the steps and problems related to the use of the app store.
8. Linkable — Easily shared via URL without complex installations.
9. Offline — Once more it is about putting the user before everything, avoiding the usual error message in case of weak or no connection. The PWA are based on two particularities: first of all the ‘skeleton’ of the app, which recalls the page structure, even if its contents do not respond and its elements include the header, the page layout, as well as an illustration that signals that the page is loading.

Code:

1. **Manifest.json**

```
{
    "name": "Simple PWA",
    "short_name": "PWA",
    "start_url": "index.html",
    "display": "standalone",
    "background_color": "#ffffff",
    "theme_color": "#000000",
    "description": "A simple Progressive Web App that explains what a PWA is.",
    "icons": [
        {
            "src": "pwa-banner.png",
            "sizes": "192x192",
            "type": "image/png"
        }
    ]
}
```

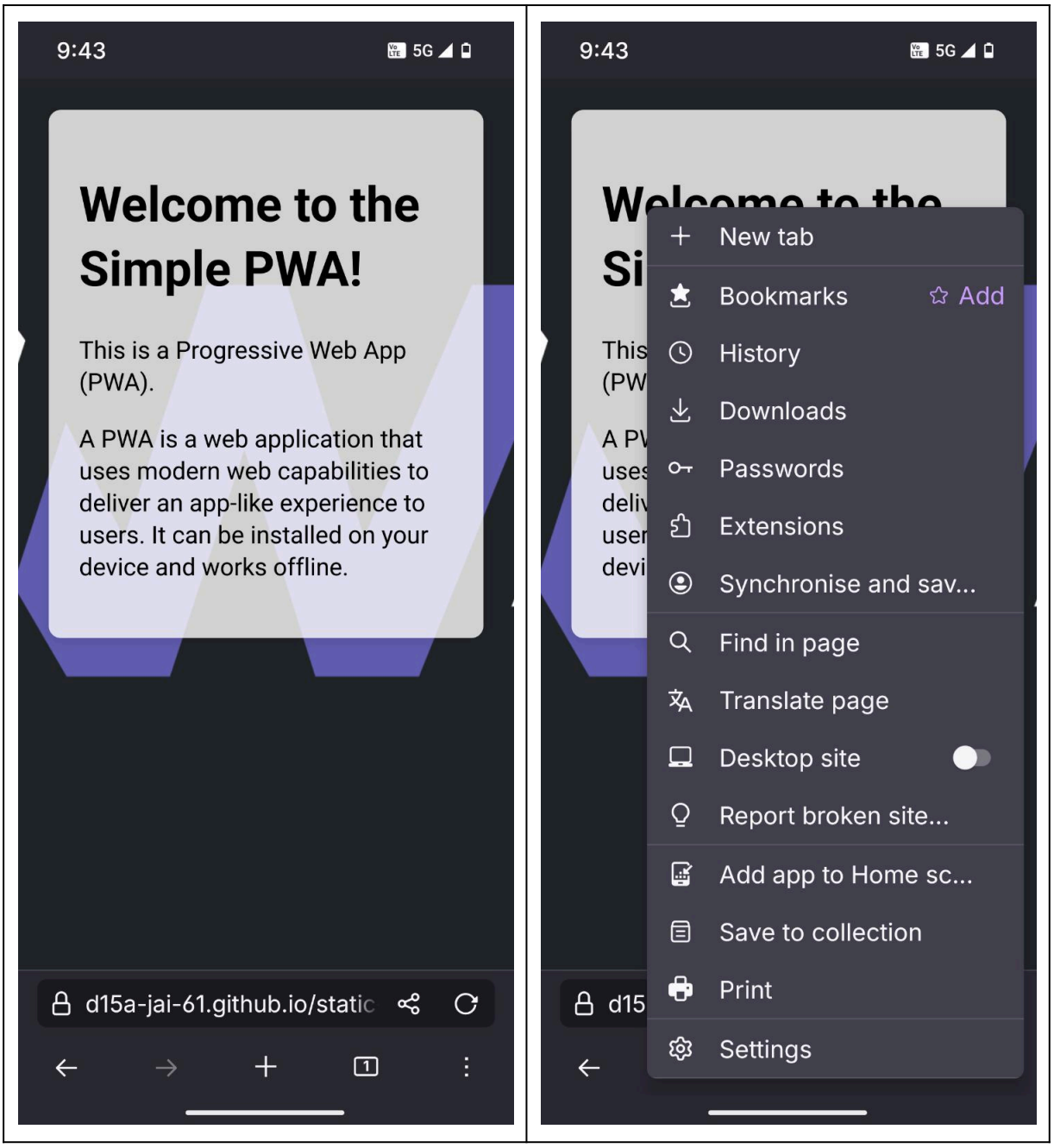
2. **Service-worker.js**

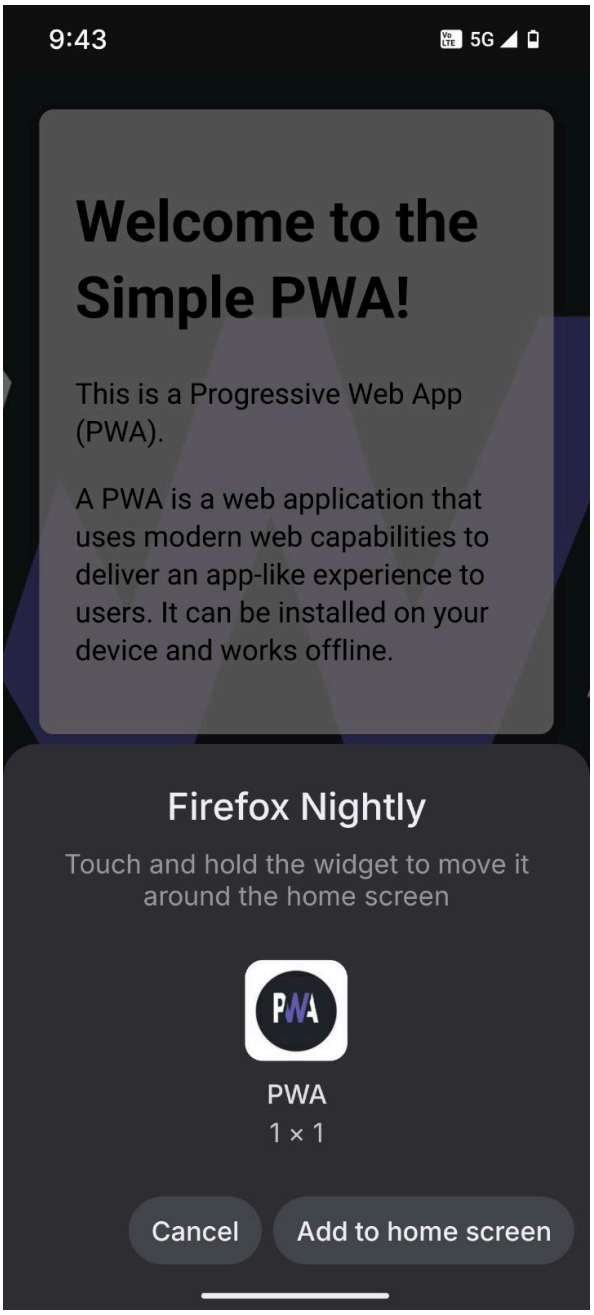
```
const CACHE_NAME = 'simple-pwa-cache-v1';
const urlsToCache = [
    'index.html',
    'offline-form.html',
    'styles.css',
    'manifest.json',
    'pwa-banner.png',
];

// Install the service worker and cache assets
self.addEventListener('install', (event) => {
    event.waitUntil(
        caches.open(CACHE_NAME).then((cache) => {
            console.log('[Service Worker] Caching essential assets');
            return cache.addAll(urlsToCache);
        })
    );
});

// Fetch event - serve cached assets or fetch from network
self.addEventListener('fetch', (event) => {
    event.respondWith(
        caches.match(event.request).then((response) => {
            if (response) {
                console.log('[Service Worker] Returning cached resource:',
```

```
event.request.url);
                return response;
            }
            console.log('[Service Worker] Fetching from network:', event.request.url);
            return fetch(event.request);
        })
    );
});

// Sync event - retry offline form submissions
self.addEventListener('sync', (event) => {
    if (event.tag === 'sync-form') {
        console.log('[Service Worker] Sync event triggered: Submitting offline form
data...');
        event.waitUntil(syncFormData());
    }
});

function syncFormData() {
    return getFormDataFromIndexedDB().then((formData) => {
        if (formData) {
            console.log('[Service Worker] Syncing form data...');
            console.log('[Service Worker] Form Data:', formData);

            // Instead of sending to an API, let's store it back in IndexedDB or localStorage
for now
            saveFormDataLocally(formData);

            // Clear the form data from IndexedDB after "syncing"
            clearFormDataFromIndexedDB();
        }
    });
}

function getFormDataFromIndexedDB() {
    return new Promise((resolve, reject) => {
        const request = indexedDB.open('offlineFormData', 1);
        request.onsuccess = function() {
            const db = request.result;
            const tx = db.transaction('formData', 'readonly');
            const store = tx.objectStore('formData');
            const data = store.getAll();  // Get all stored form data
            data.onsuccess = function() {
                if (data.result.length > 0) {
                    resolve(data.result[0]);  // Return the first form data entry
                } else {
                    resolve(null);  // No data found
                }
            };
            data.onerror = reject;
        };
        request.onerror = reject;
    });
}
```

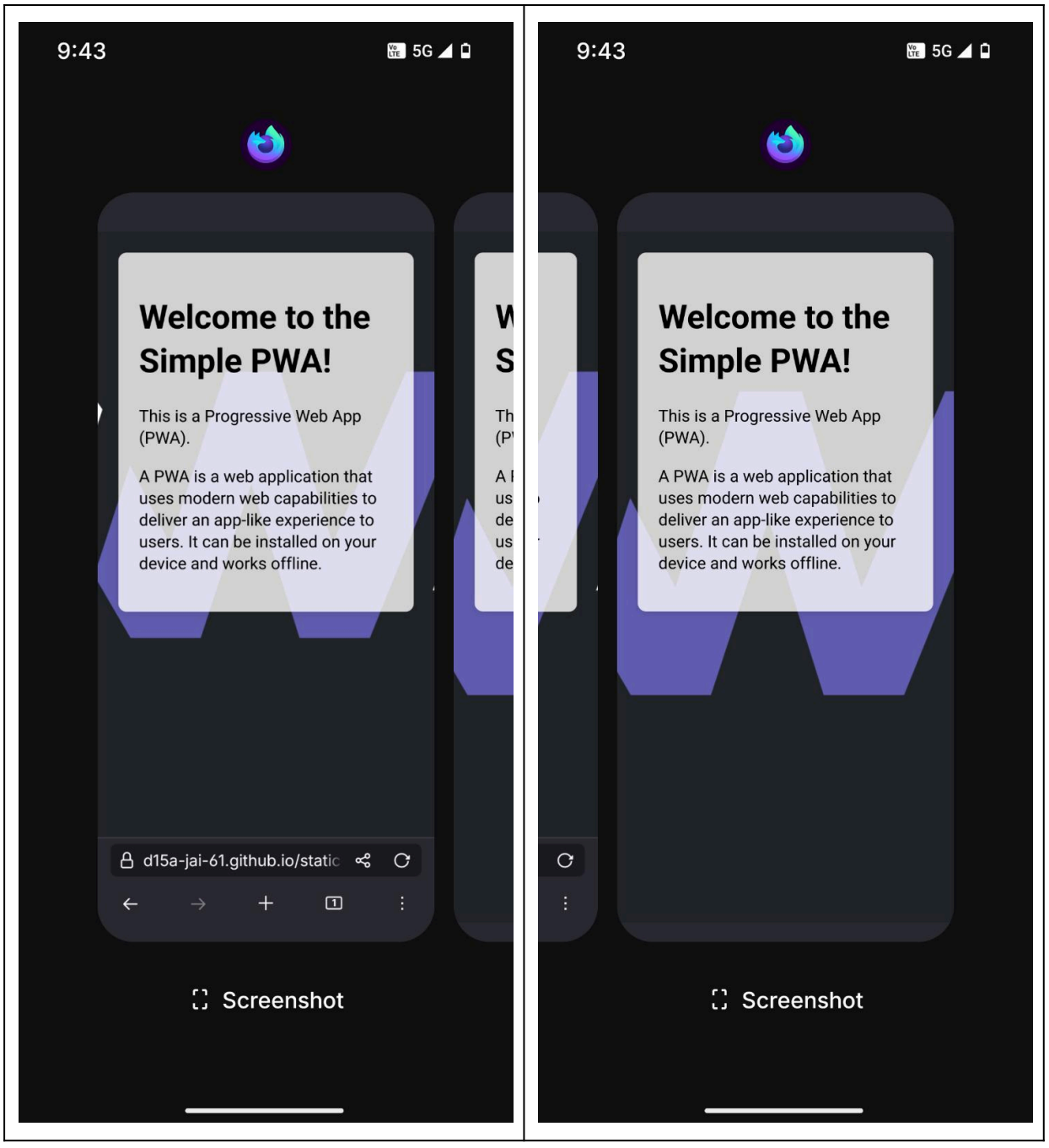
```
function saveFormDataLocally(formData) {
    // Save form data back into IndexedDB (simulating pushing data back into the app)
    const request = indexedDB.open('offlineFormData', 1);
    request.onsuccess = function() {
        const db = request.result;
        const tx = db.transaction('formData', 'readwrite');
        const store = tx.objectStore('formData');
        store.add(formData);  // Push data back into IndexedDB
        tx.oncomplete = function() {
            console.log('[Service Worker] Form data saved locally to IndexedDB:',
formData);
        };
    };
}

function clearFormDataFromIndexedDB() {
    const request = indexedDB.open('offlineFormData', 1);
    request.onsuccess = function() {
        const db = request.result;
        const tx = db.transaction('formData', 'readwrite');
        const store = tx.objectStore('formData');
        store.clear();  // Clear the form data from IndexedDB after syncing
        console.log('[Service Worker] Form data cleared from IndexedDB.');
    };
}

// Push event - handle push notifications (you can modify this part as per your needs)
self.addEventListener('push', (event) => {
    const options = {
        body: event.data.text(),
        icon: 'pwa-banner.png',  // Use pwa-banner.png for the notification icon
        badge: 'badge.png',      // Optional: You can use a badge if needed
    };

    event.waitUntil(
        self.registration.showNotification('New Content Available!', options)
    );
});

// Handle notification click
self.addEventListener('notificationclick', (event) => {
    event.notification.close();
    event.waitUntil(
        clients.openWindow('https://d15a-jai-61.github.io/static-pwa/') // Open your app's
homepage or a specific page
    );
});
```

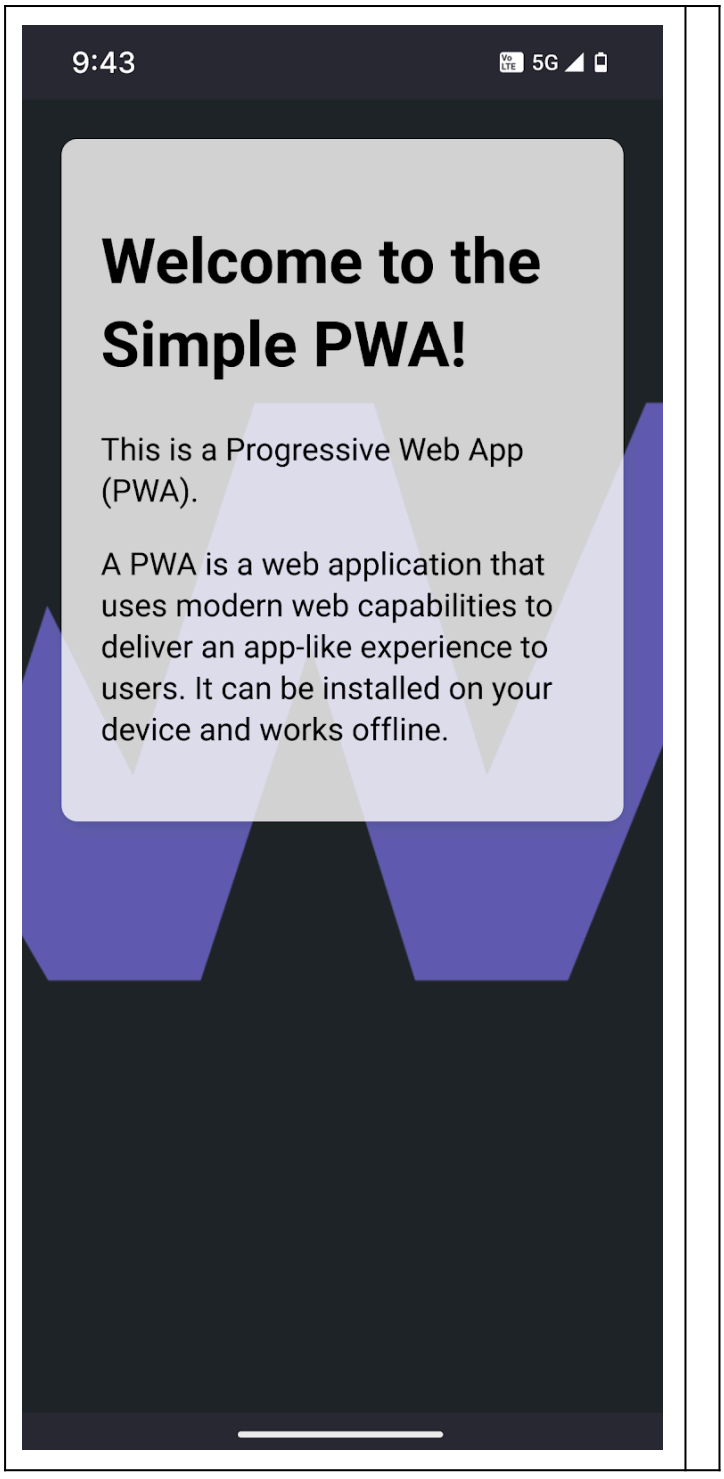
Output:


9:43
5G
Welcome to the Simple PWA!
This is a Progressive Web App (PWA).
A PWA is a web application that uses modern web capabilities to deliver an app-like experience to users. It can be installed on your device and works offline.
d15a-jai-61.github.io/static
9:43
5G
New tab
Bookmarks
Add
History
Downloads
Passwords
Extensions
Synchronise and sav...
Find in page
Translate page
Desktop site
Report broken site...
Add app to Home sc...
Save to collection
Print
Settings


9:43
5G
Welcome to the Simple PWA!
This is a Progressive Web App (PWA).
A PWA is a web application that uses modern web capabilities to deliver an app-like experience to users. It can be installed on your device and works offline.
Firefox Nightly
Touch and hold the widget to move it around the home screen
PWA
1 × 1
Cancel
Add to home screen

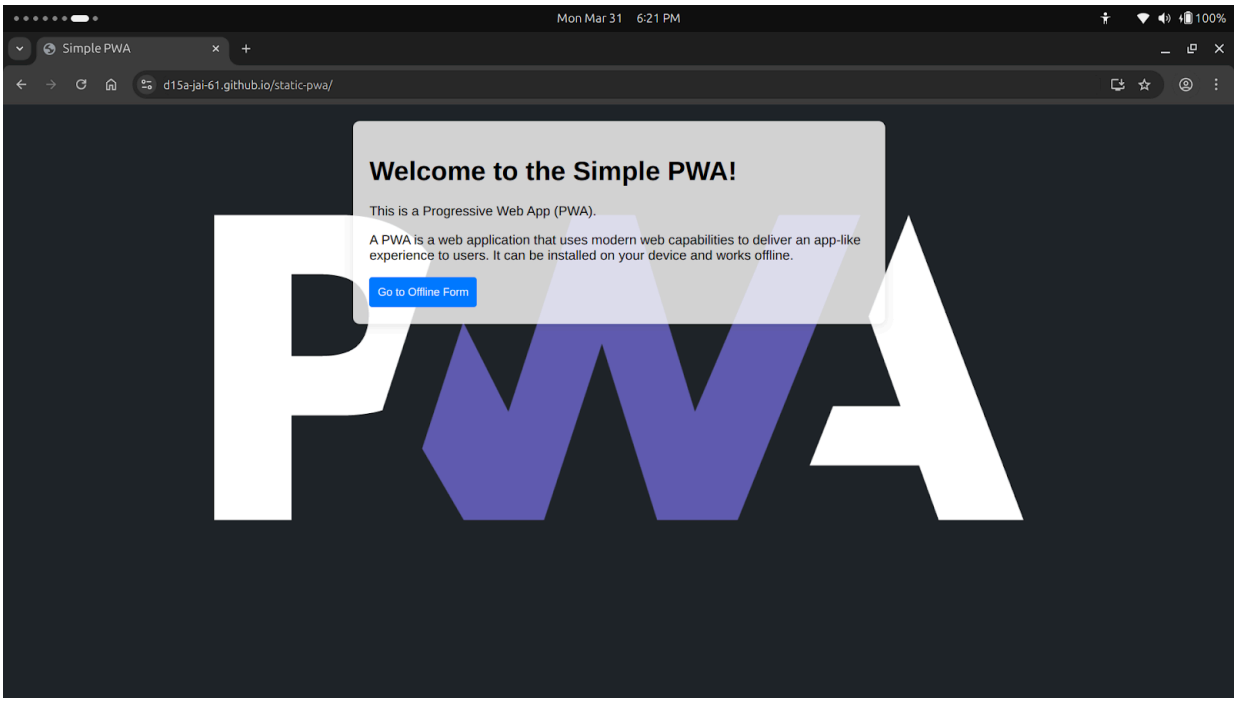

9:43
5G
S
K
PWA

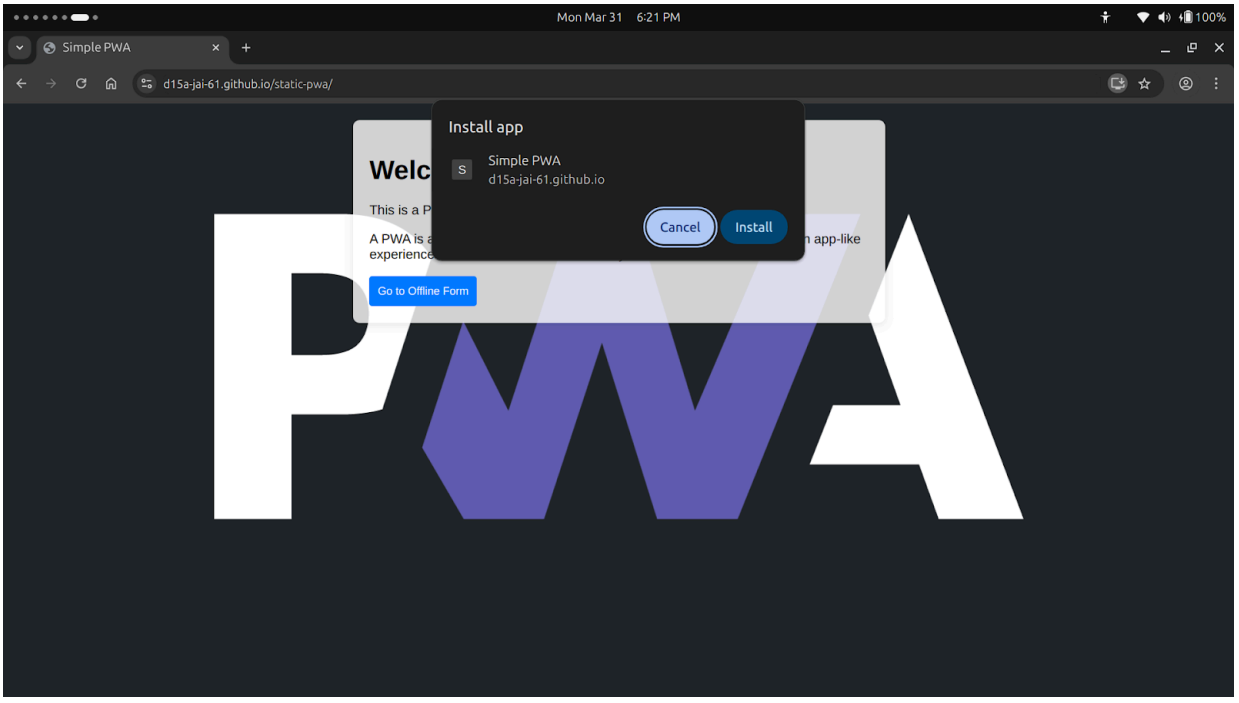

9:43
5G
Welcome to the Simple PWA!
This is a Progressive Web App (PWA).
A PWA is a web application that uses modern web capabilities to deliver an app-like experience to users. It can be installed on your device and works offline.
d15a-jai-61.github.io/static
Screenshot
9:43
5G
Welcome to the Simple PWA!
This is a Progressive Web App (PWA).
A PWA is a web application that uses modern web capabilities to deliver an app-like experience to users. It can be installed on your device and works offline.
Screenshot


9:43
5G
Welcome to the Simple PWA!
This is a Progressive Web App (PWA).
A PWA is a web application that uses modern web capabilities to deliver an app-like experience to users. It can be installed on your device and works offline.

NAME: Diksha Utekar Jai Talreja Vedang Wajge

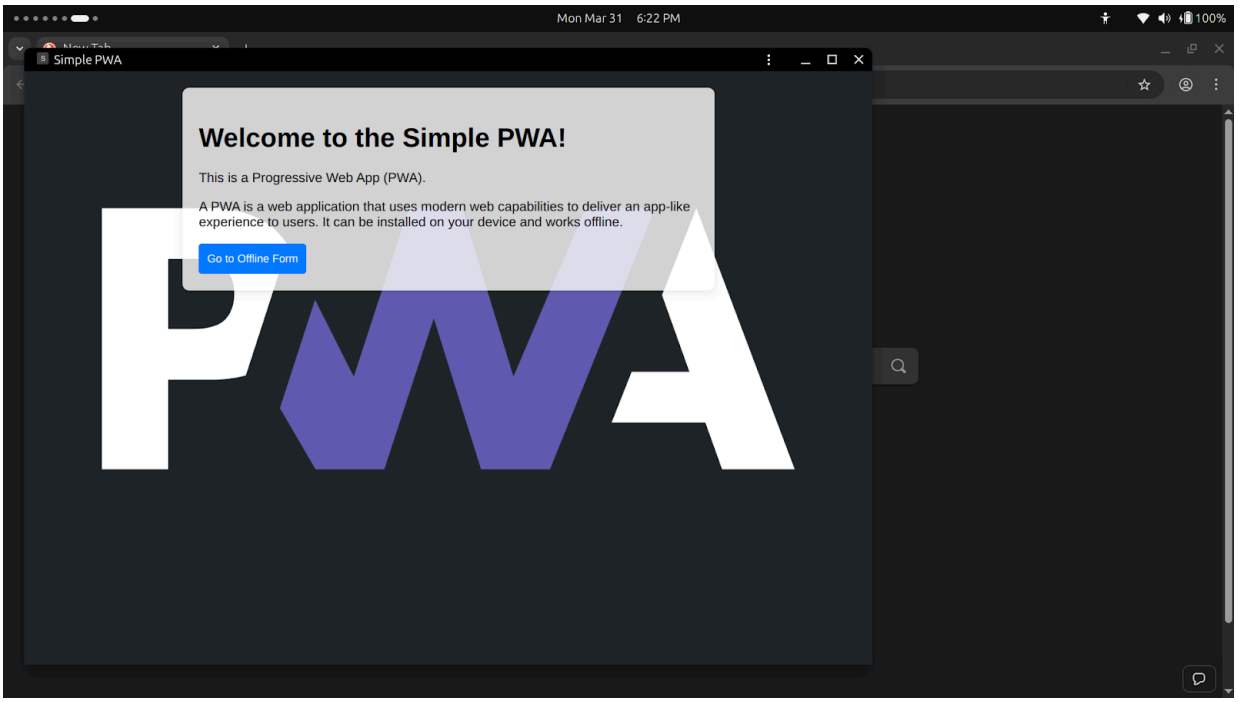
D15A- 63/59/62

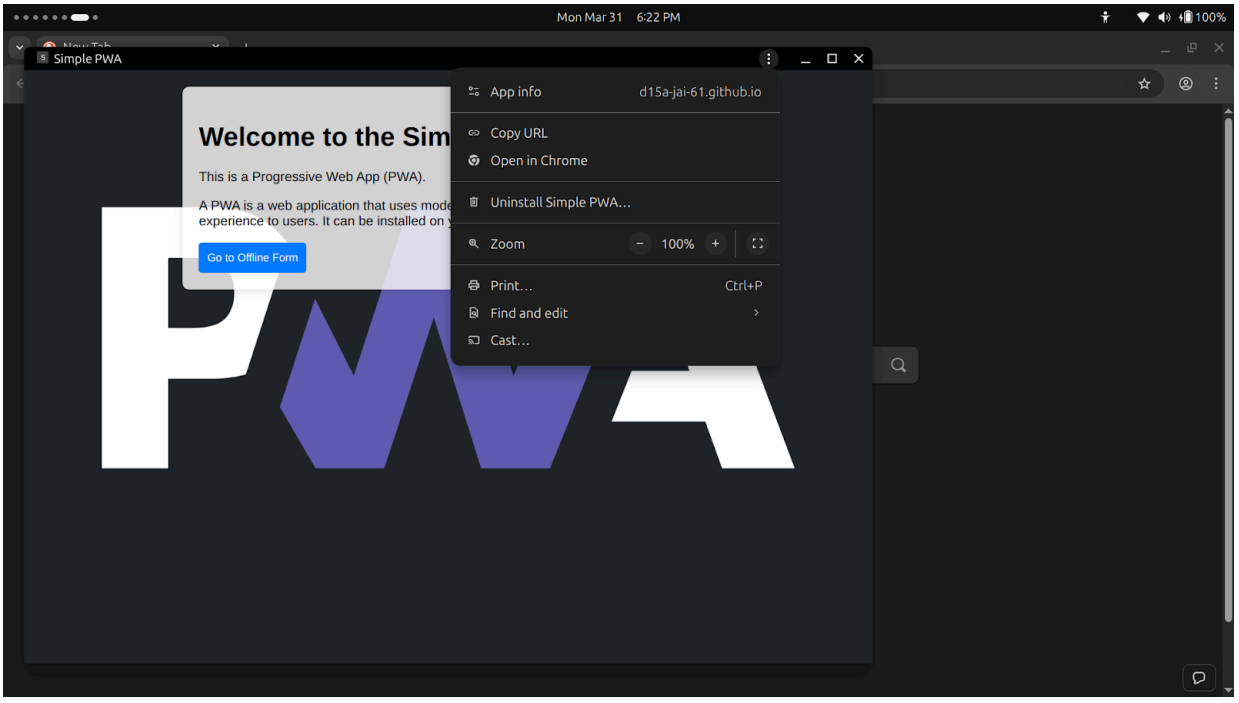


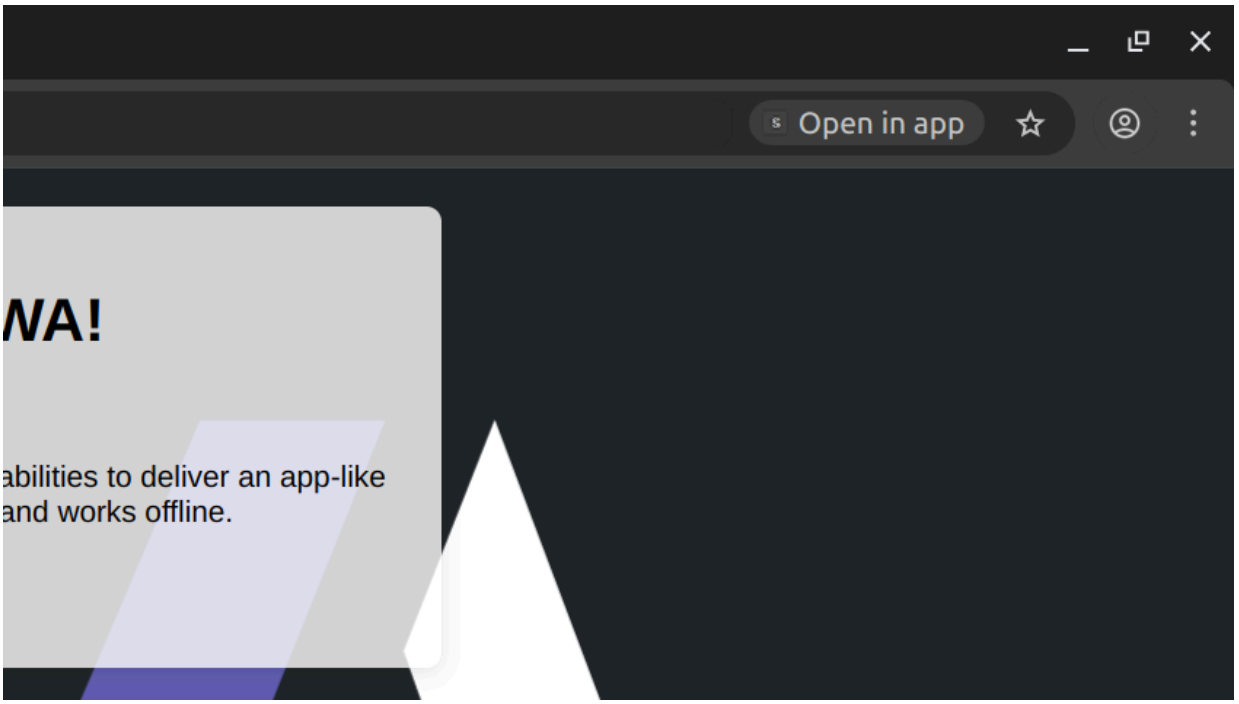


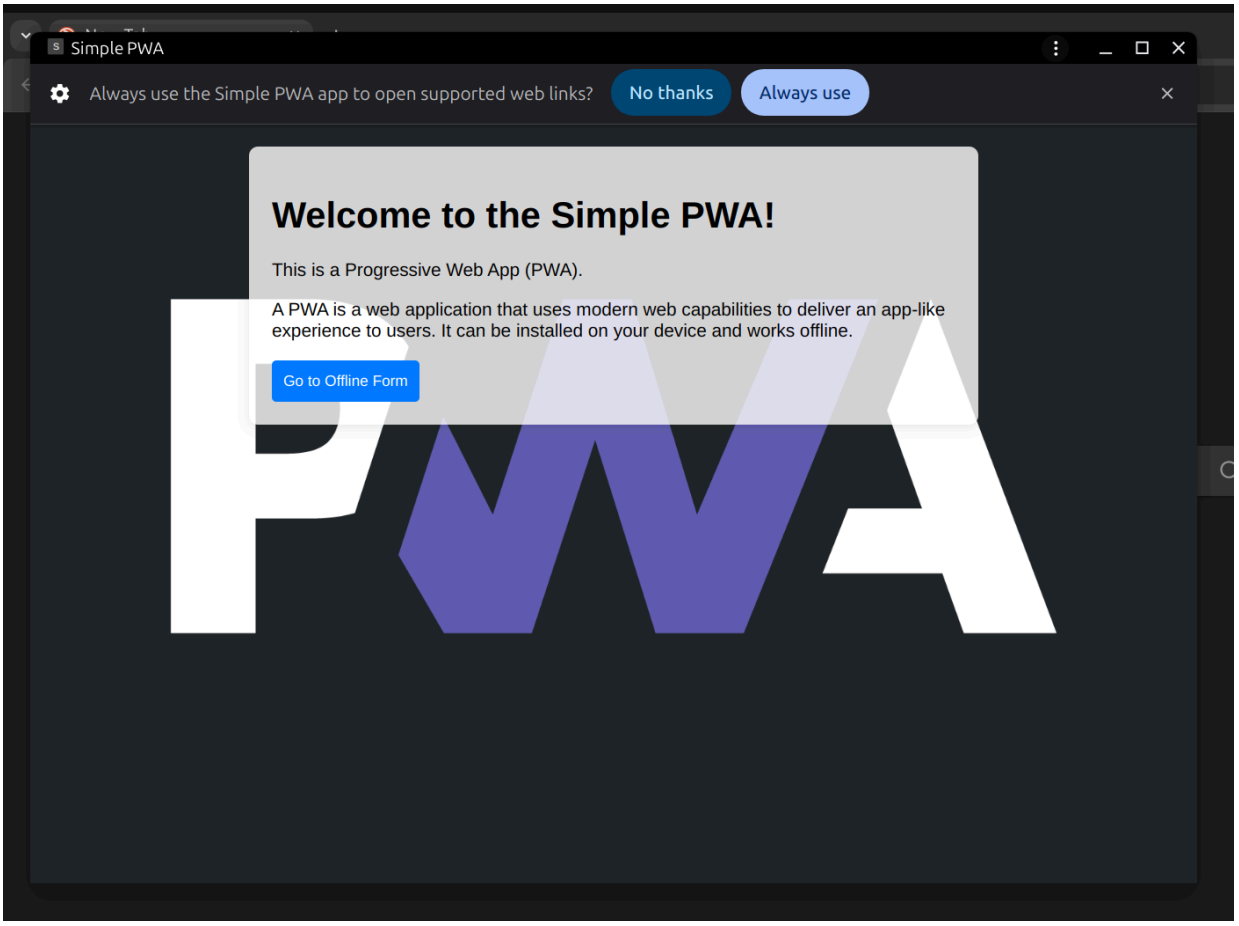
NAME: Diksha Utekar Jai Talreja Vedang Wajge

D15A- 63/59/62






Open in app
NA!
abilities to deliver an app-like
and works offline.


Simple PWA
Always use the Simple PWA app to open supported web links?
No thanks
Always use
Welcome to the Simple PWA!
This is a Progressive Web App (PWA).
A PWA is a web application that uses modern web capabilities to deliver an app-like experience to users. It can be installed on your device and works offline.
Go to Offline Form
PWA

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.	**63**
Name	**UTEKAR DIKSHA ABHINAY**
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Name: Diksha Utekar Jai Talreja Vedang Wajge D15A- 63/59/62

Experiment 8

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 dominate **Network Traffic**
  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.

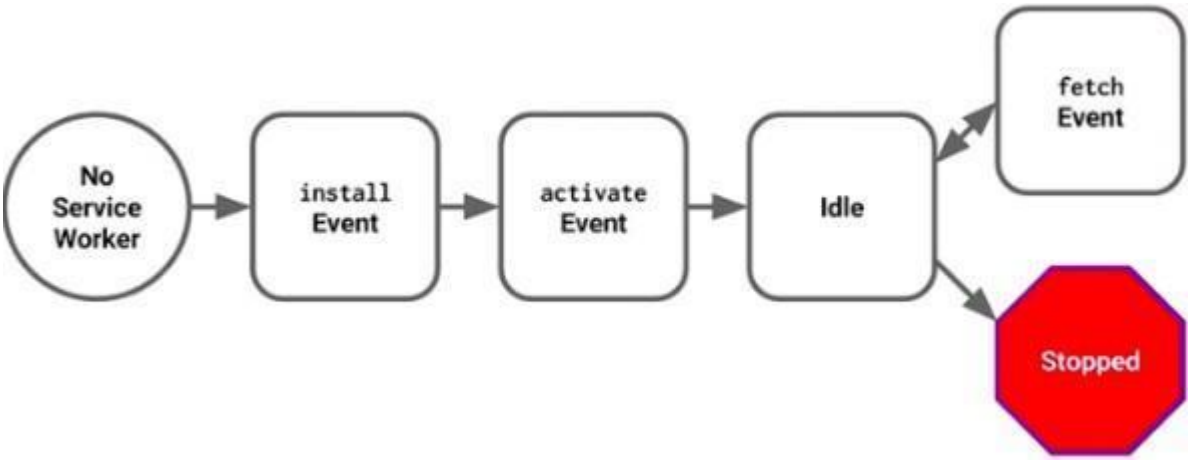
What can’t we do with Service Workers?

- 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.

Service Worker Cycle



A service worker goes through three steps in its life cycle:

- Registration
- Installation
- Activation

Registration
To install a service worker, you need to register it in your main JavaScript code. Registration tells the browser where your service worker is located, and to start installing it in the background. Let's look at an example:

main.js

```
if ('serviceWorker' in navigator) { navigator.serviceWorker.register('/service-worker.js')
  .then(function(registration) {
    console.log('Registration successful, scope is:', registration.scope);
  })
  .catch(function(error) {
    console.log('Service worker registration failed, error:', error);
  });
}
```

This code starts by checking for browser support by examining **navigator.serviceWorker**. The service worker is then registered with navigator.serviceWorker.register, which returns a promise that resolves when the service worker has been successfully registered. The scope of the service worker is then logged with registration.scope. If the service worker is already installed, navigator.serviceWorker.register returns the registration object of the

currently active service worker.

The scope of the service worker determines which files the service worker controls, in other words, from which path the service worker will intercept requests. The default scope is the location of the service worker file, and extends to all directories below. So if service-worker.js is located in the root directory, the service worker will control requests from all files at this domain.

You can also set an arbitrary scope by passing in an additional parameter when registering. For example:

main.js

```
navigator.serviceWorker.register('/service-worker.js', {
  scope: '/app/'
});
```

In this case we are setting the scope of the service worker to /app/, which means the service worker will control requests from pages like /app/, /app/lower/ and /app/lower/lower, but not from pages like /app or /, which are higher.

If you want the service worker to control higher pages e.g. /app (without the trailing slash) you can indeed change the scope option, but you'll also need to set the Service-Worker-Allowed HTTP Header in your server config for the request serving the service worker script.

main.js

```
navigator.serviceWorker.register('/app/service-worker.js', { scope: '/app'
});
```

Installation

Once the browser registers a service worker, installation can be attempted. This occurs if the service worker is considered to be new by the browser, either because the site currently doesn't have a registered service worker, or because there is a byte difference between the new service worker and the previously installed one.

A service worker installation triggers an install event in the installing service worker. We can include an install event listener in the service worker to perform some task when the service worker installs. For instance, during the install, service workers can precache parts of a web app so that it loads instantly the next time a user opens it (see caching the application shell). So, after that first load, you're going to benefit from instant repeat loads and your time to interactivity is going to be even better in those cases. An example of an installation event listener looks like this:

service-worker.js

```
// Listen for install event, set callback
self.addEventListener('install', function(event) {
  // Perform some task
});
```

Activation

Once a service worker has successfully installed, it transitions into the activation stage. If there are any open pages controlled by the previous service worker, the new service worker enters a waiting state. The new service worker only activates when there are no longer any pages loaded that are still using the old service worker. This ensures that only one version of the service worker is running at any given time.

When the new service worker activates, an activate event is triggered in the activating service worker. This event listener is a good place to clean up outdated caches (see the Offline Cookbook for an example).

service-worker.js

```
self.addEventListener('activate', function(event) {
  // Perform some task
});
```

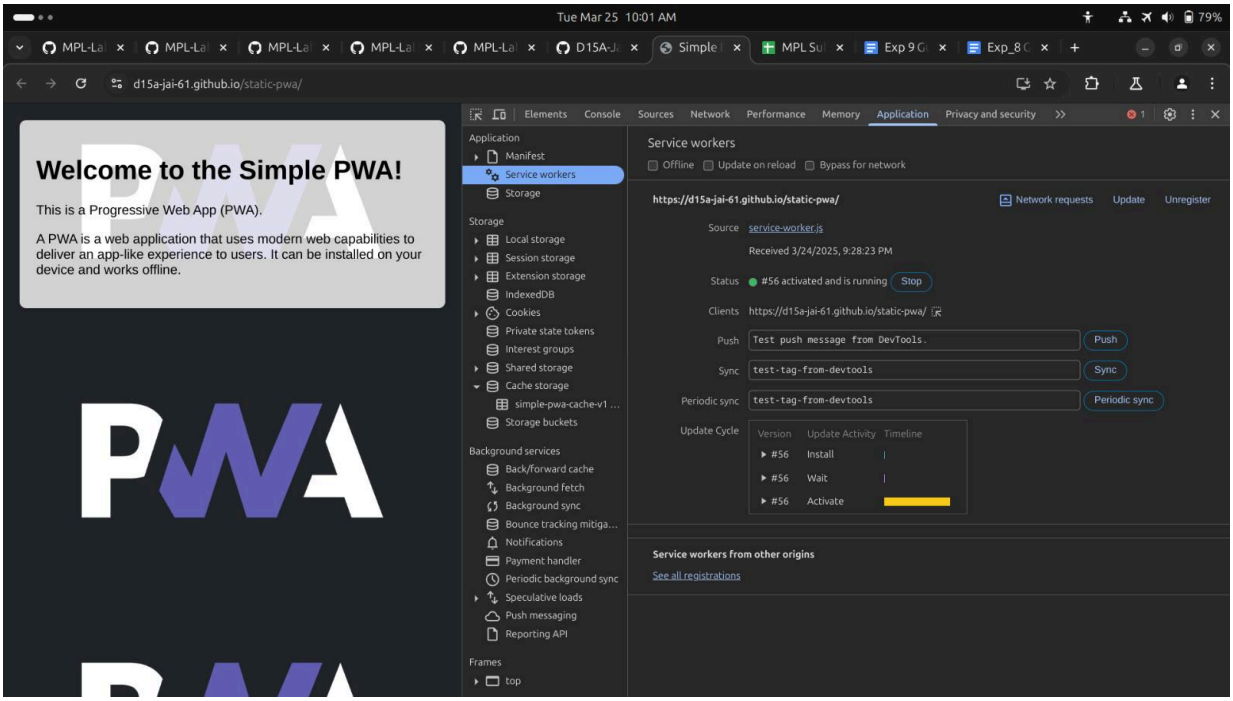
Once activated, the service worker controls all pages that load within its scope, and starts listening for events from those pages. However, pages in your app that were loaded before the service worker activation will not be under service worker control. The new service worker will only take over when you close and reopen your app, or if the service worker calls **clients.claim()**. Until then, requests from this page will not be intercepted by the new service worker. This is intentional as a way to ensure consistency in your site.

Code under index.html:

```
<script>
  if ('serviceWorker' in navigator) {
    window.addEventListener('load', function() {
      navigator.serviceWorker.register('service-worker.js').then(function(registration) {
        console.log('Service Worker registered with scope:', registration.scope);
      }, function(err) {
        console.log('Service Worker registration failed:', err);
      });
    });
  }
</script>
```

Name: Diksha Utekar Jai Talreja Vedang Wajge D15A- 63/59/62

Output:



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.	**63**
Name	**UTEKAR DIKSHA ABHINAY**
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Experiment 9

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.

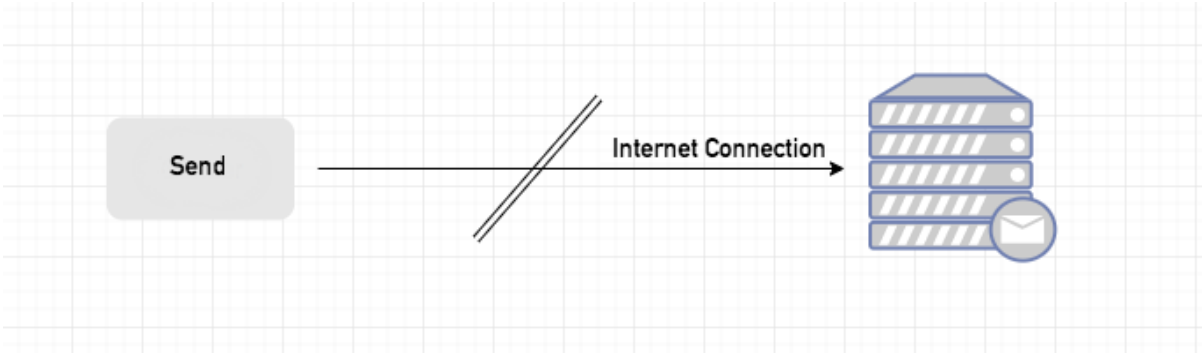
```
self.addEventListener("fetch", function (event) {
    const req = event.request;
    const url = new URL(req.url);
```

Sync Event

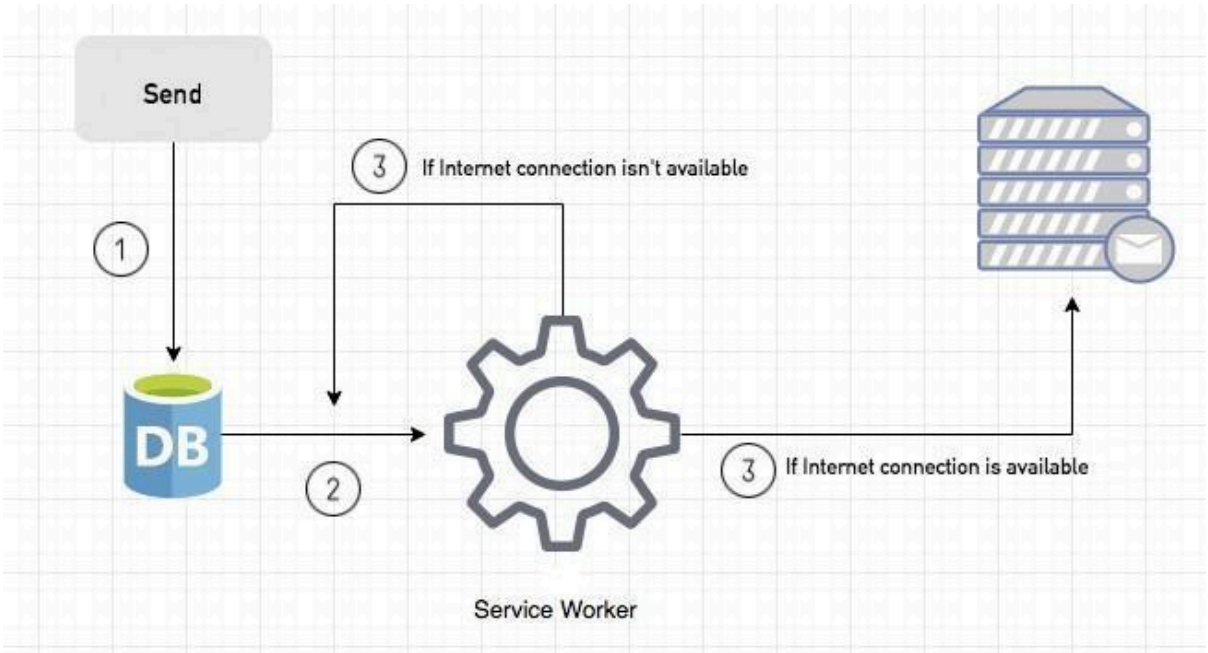
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

Here is a job for the Background Sync.

The following view shows the classical process of sending email to us. If the Internet Connection is broken, we can't send any content to Mail Server.



Here, you can create any scenario for yourself. A sample is in the following for this case.



1. When we click the "send" button, email content will be saved to IndexedDB.
2. Background Sync registration.
3. **If the Internet connection is available**, all email content will be read and sent to Mail Server.
   If the Internet connection is unavailable, the service worker waits until the connection is available even though

the window is closed. When it is available, email content will be sent to Mail Server.

You can see the working process within the following code block.

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:

service-worker.js :

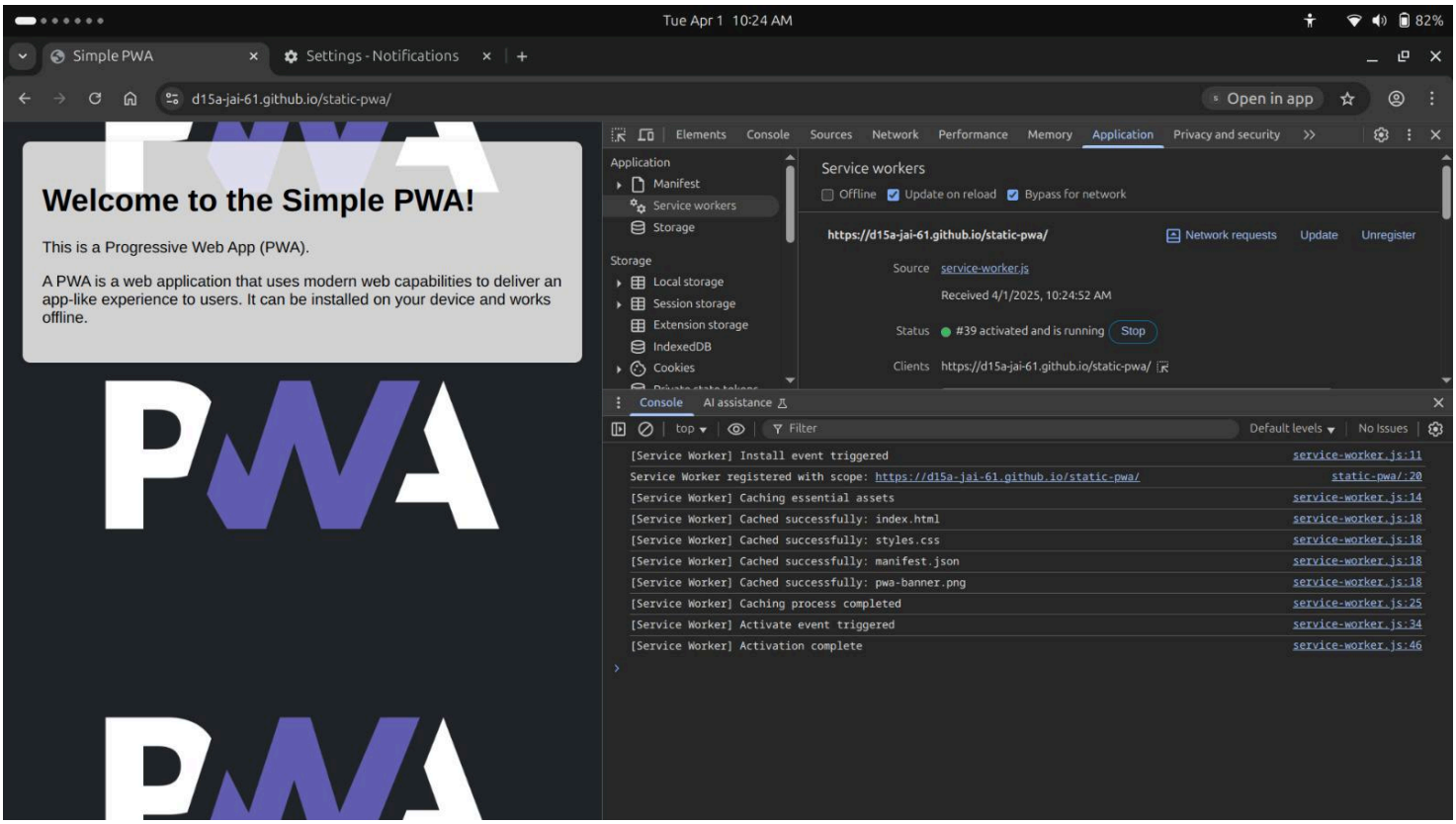
```
const CACHE_NAME = 'static-pwa-cache-v1';
const urlsToCache = [
   'index.html',
   'styles.css',
   'manifest.json',
   'pwa-banner.png',
];

// Install event - Cache essential assets
self.addEventListener('install', (event) => {
   console.log('[Service Worker] Install event triggered');
   event.waitUntil(
      caches.open(CACHE_NAME).then(cache => {
         console.log('[Service Worker] Caching essential assets');
         return Promise.all(
            urlsToCache.map(url =>
               cache.add(url).then(() =>
                  console.log([Service Worker] Cached successfully: ${url})
               ).catch(error =>
                  console.error([Service Worker] Failed to cache: ${url}, error)
               )
            )
         );
```

```
        }).then(() => {
            console.log('[Service Worker] Caching process completed');
            return self.skipWaiting();
        })
        .catch(error => console.error('[Service Worker] Caching failed:', error))
    );
});

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

// Fetch event - Serve cached assets or fetch from network
self.addEventListener('fetch', (event) => {
    console.log([Service Worker] Fetch event triggered for: ${event.request.url});
    if (event.request.method !== 'GET') return;

    event.respondWith(
        fetch(event.request).then(networkResponse => {
            console.log([Service Worker] Network response for ${event.request.url});
            return caches.open(CACHE_NAME).then(cache => {
                cache.put(event.request, networkResponse.clone());
                return networkResponse;
            });
        }).catch(error => {
            console.log([Service Worker] Network failed, trying cache for ${event.request.url});
            return caches.match(event.request).then(cachedResponse => {
                if (cachedResponse) {
                    console.log([Service Worker] Serving from cache: ${event.request.url});
                    return cachedResponse;
                }
                if (event.request.mode === 'navigate') {
                    return caches.match(OFFLINE_URL);
                }
                return new Response('Offline', { status: 503, statusText: 'Service Unavailable' });
            });
        })
    );
```

```
});

// Sync event - Handle background sync
self.addEventListener('sync', (event) => {
    console.log([Service Worker] Sync event triggered: ${event.tag});
    if (event.tag === 'sync-demo') {
        event.waitUntil(
            handleSync().catch(error => {
                console.error('[Service Worker] Sync failed:', error);
            })
        );
    }
});

async function handleSync() {
    console.log('[Service Worker] Starting sync...');
    await new Promise(resolve => setTimeout(resolve, 2000));
    console.log('[Service Worker] Sync task completed successfully!');
}

// Push event - Handle push notifications
self.addEventListener('push', (event) => {
    console.log('[Service Worker] Push event received');
    const options = {
        body: event.data ? event.data.text() : 'New update available!',
        icon: 'pwa-banner.png',
        badge: 'badge.png',
        actions: [
            { action: 'open', title: 'View Now' },
            { action: 'dismiss', title: 'Dismiss' }
        ]
    };
    event.waitUntil(
        self.registration.showNotification('New Notification', options)
        .then(() => console.log('[Service Worker] Push notification displayed successfully'))
        .catch((error) => console.error('[Service Worker] Failed to display push notification:', error))
    );
});

// Handle notification click
self.addEventListener('notificationclick', (event) => {
    console.log([Service Worker] Notification clicked: ${event.notification.title});
    event.notification.close();
    if (event.action === 'open') {
        console.log('[Service Worker] Opening application');
        event.waitUntil(clients.openWindow('https://your-static-site.com'));
    } else {
        console.log('[Service Worker] Notification dismissed');
    }
});
```

OUTPUT:

Fetch event

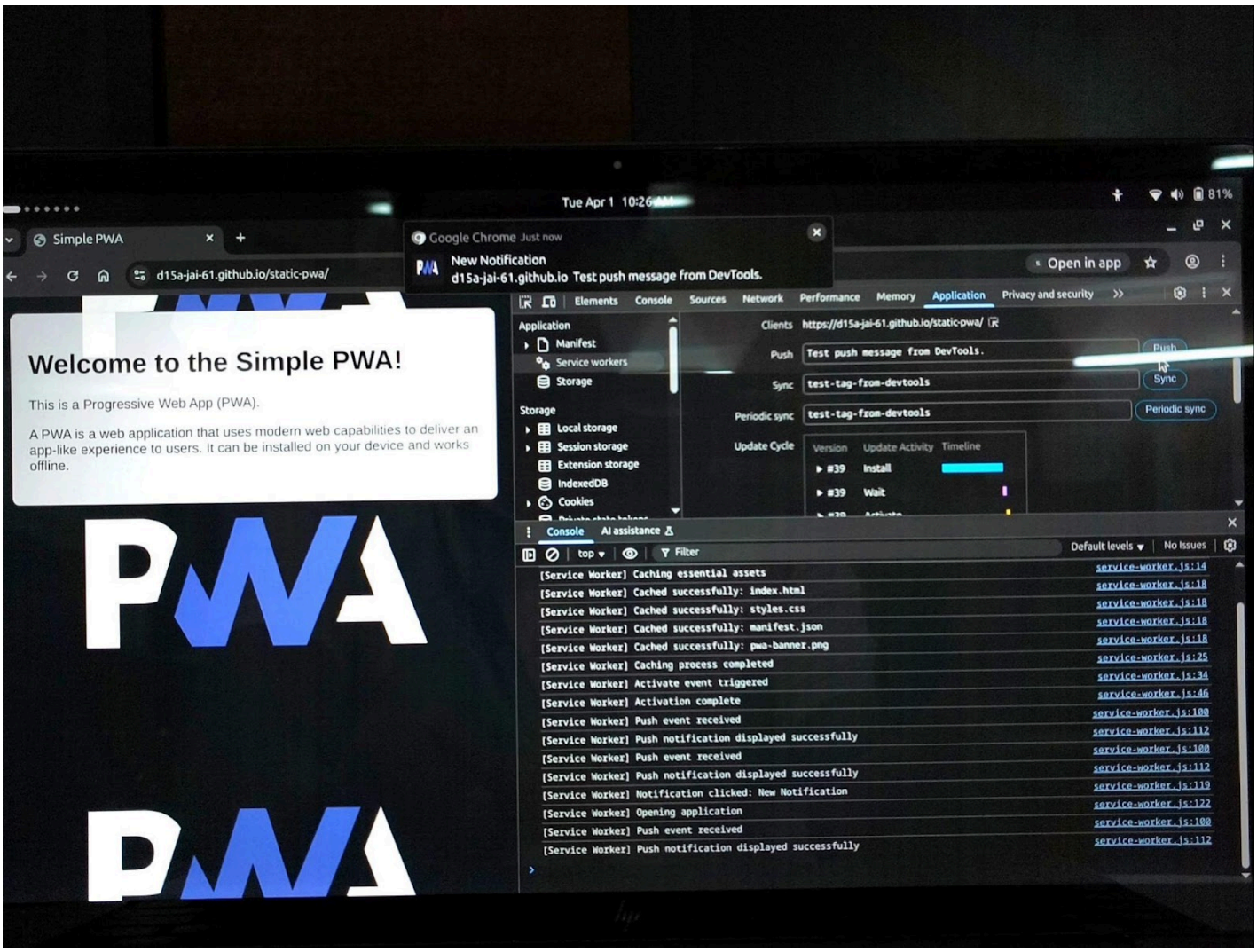

Tue Apr 1 10:24 AM
82%
Simple PWA
Settings - Notifications
d15a-jai-61.github.io/static-pwa/
Open in app
Welcome to the Simple PWA!
This is a Progressive Web App (PWA).
A PWA is a web application that uses modern web capabilities to deliver an app-like experience to users. It can be installed on your device and works offline.
Elements
Console
Sources
Network
Performance
Memory
Application
Privacy and security
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Service workers
Offline
Update on reload
Bypass for network
https://d15a-jai-61.github.io/static-pwa/
Network requests
Update
Unregister
Source service-worker.js
Received 4/1/2025, 10:24:52 AM
Status #39 activated and is running
Stop
Clients https://d15a-jai-61.github.io/static-pwa/
Console
AI assistance
top
Filter
Default levels
No Issues
[Service Worker] Install event triggered
service-worker.js:11
Service Worker registered with scope: https://d15a-jai-61.github.io/static-pwa/
static-pwa/:20
[Service Worker] Caching essential assets
service-worker.js:14
[Service Worker] Cached successfully: index.html
service-worker.js:18
[Service Worker] Cached successfully: styles.css
service-worker.js:18
[Service Worker] Cached successfully: manifest.json
service-worker.js:18
[Service Worker] Cached successfully: pwa-banner.png
service-worker.js:18
[Service Worker] Caching process completed
service-worker.js:25
[Service Worker] Activate event triggered
service-worker.js:34
[Service Worker] Activation complete
service-worker.js:46

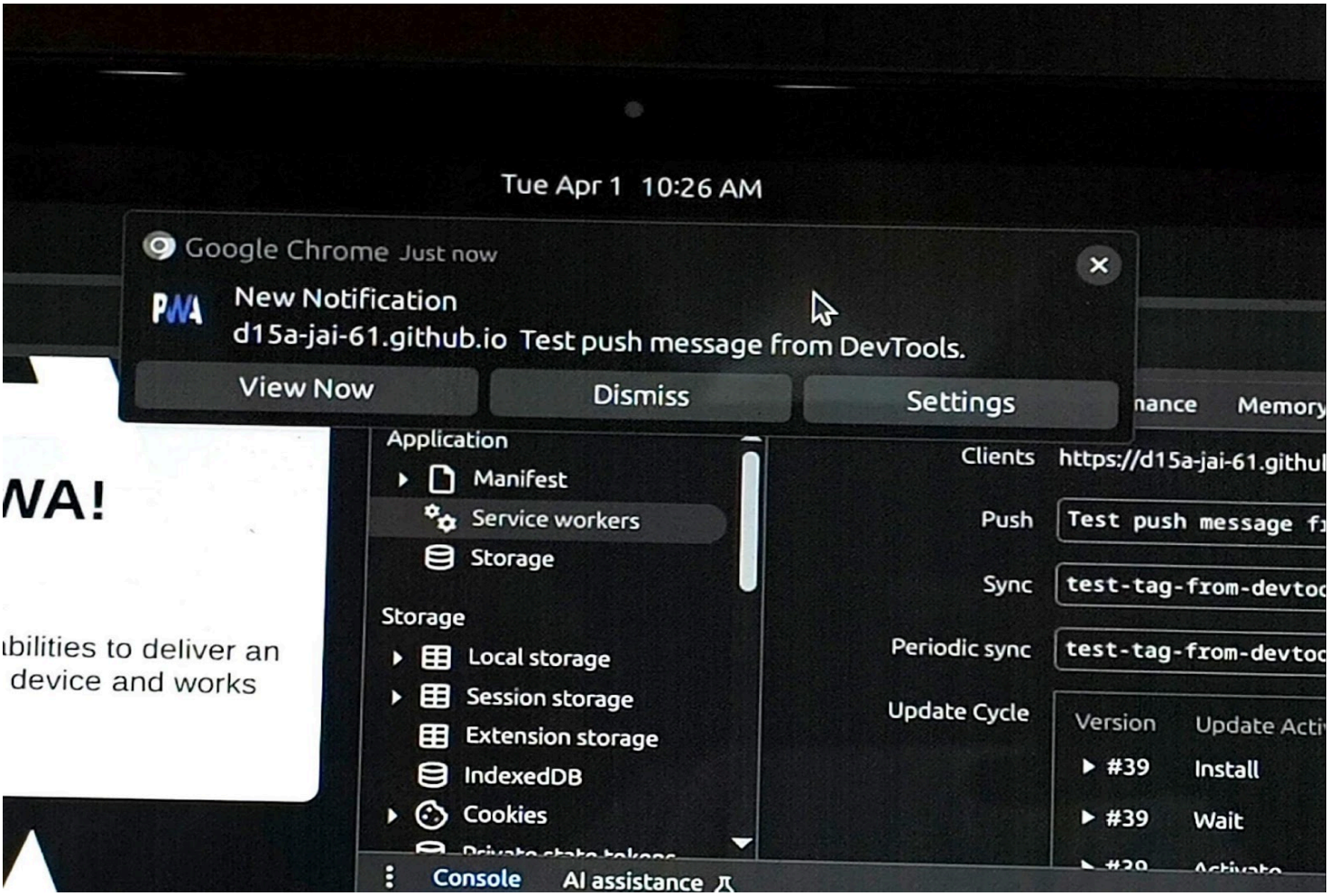
Push event

Console AI assistance

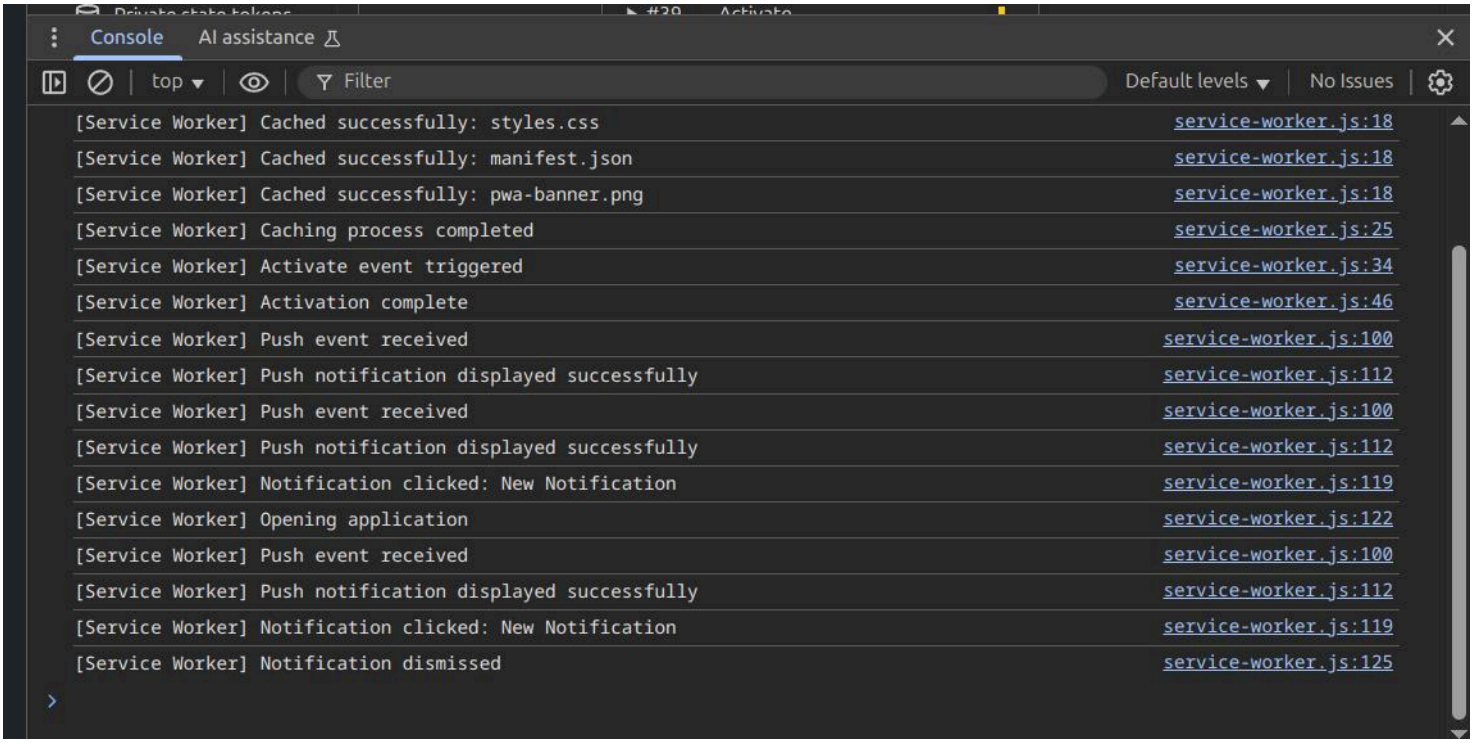
top Filter Default levels No Issues

```
[Service Worker] Install event triggered                                service-worker.js:11
Service Worker registered with scope: https://d15a-jai-61.github.io/static-pwa/  static-pwa/:20
[Service Worker] Caching essential assets                               service-worker.js:14
[Service Worker] Cached successfully: index.html                        service-worker.js:18
[Service Worker] Cached successfully: styles.css                        service-worker.js:18
[Service Worker] Cached successfully: manifest.json                     service-worker.js:18
[Service Worker] Cached successfully: pwa-banner.png                    service-worker.js:18
[Service Worker] Caching process completed                              service-worker.js:25
[Service Worker] Activate event triggered                               service-worker.js:34
[Service Worker] Activation complete                                    service-worker.js:46
[Service Worker] Push event received                                    service-worker.js:100
[Service Worker] Push notification displayed successfully               service-worker.js:112
```





After dismissing the notification


Console
AI assistance
top
Filter
Default levels
No Issues
[Service Worker] Cached successfully: styles.css service-worker.js:18
[Service Worker] Cached successfully: manifest.json service-worker.js:18
[Service Worker] Cached successfully: pwa-banner.png service-worker.js:18
[Service Worker] Caching process completed service-worker.js:25
[Service Worker] Activate event triggered service-worker.js:34
[Service Worker] Activation complete service-worker.js:46
[Service Worker] Push event received service-worker.js:100
[Service Worker] Push notification displayed successfully service-worker.js:112
[Service Worker] Push event received service-worker.js:100
[Service Worker] Push notification displayed successfully service-worker.js:112
[Service Worker] Notification clicked: New Notification service-worker.js:119
[Service Worker] Opening application service-worker.js:122
[Service Worker] Push event received service-worker.js:100
[Service Worker] Push notification displayed successfully service-worker.js:112
[Service Worker] Notification clicked: New Notification service-worker.js:119
[Service Worker] Notification dismissed service-worker.js:125

Sync event


[Service Worker] Opening application service-worker.js:122
[Service Worker] Push event received service-worker.js:100
[Service Worker] Push notification displayed successfully service-worker.js:112
[Service Worker] Notification clicked: New Notification service-worker.js:119
[Service Worker] Notification dismissed service-worker.js:125
[Service Worker] Sync event triggered: test-tag-from-devtools service-worker.js:82

MAD & PWA Lab

Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	**63**
Name	**UTEKAR DIKSHA ABHINAY**
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Experiment 10

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

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:

1. Blogging with Jekyll
2. Custom URL
3. Automatic Page Generator

Reasons for favoring this over Firebase:

1. Free to use
2. Right out of github
3. Quick to set up

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

Pros

1. Very familiar interface if you are already using GitHub for your projects.
2. Easy to set up. Just push your static website to the gh-pages branch and your website is ready.
3. Supports Jekyll out of the box.
4. 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

1. The code of your website will be public, unless you pay for a private repository.
2. Currently, there is no support for HTTPS for custom domains. It's probably coming soon though.
3. 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:

1. 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.
2. 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.
3. 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:

1. Realtime backend made easy
2. Fast and responsive

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

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

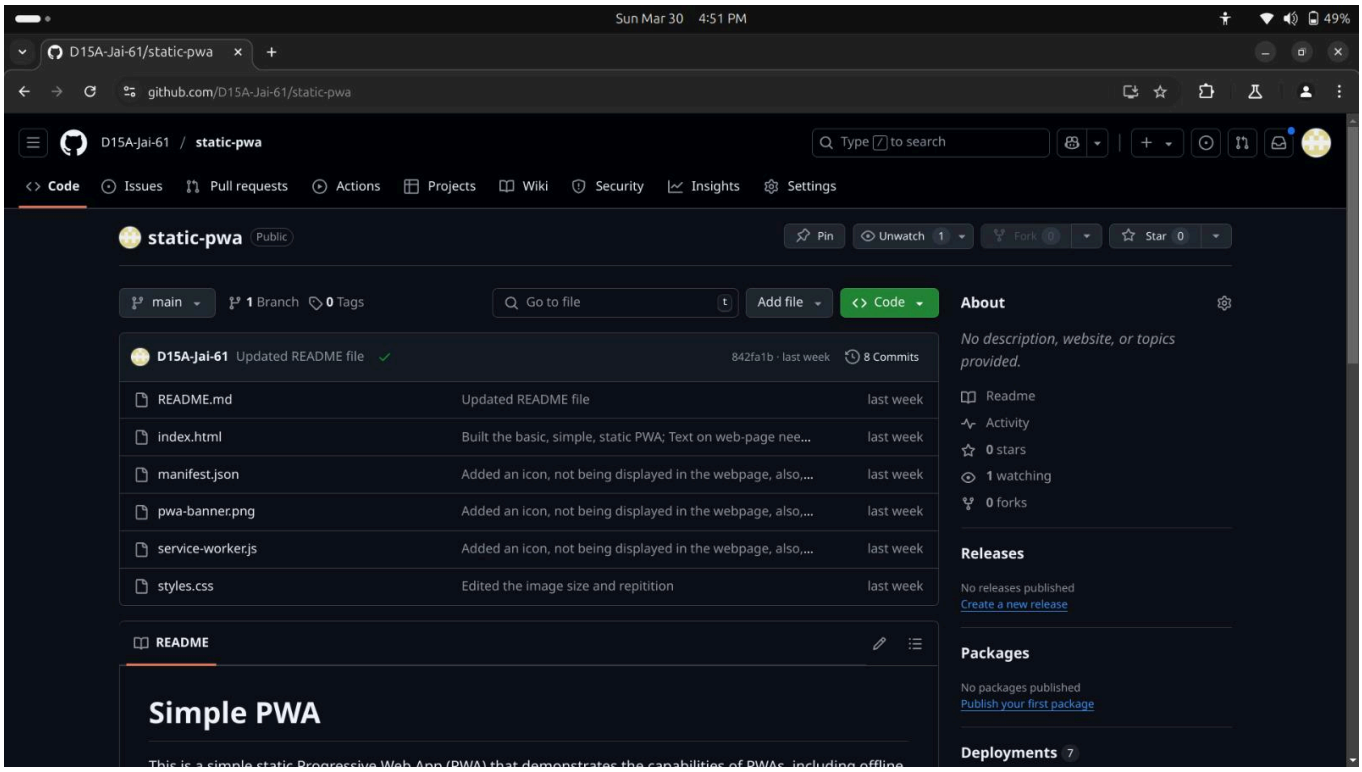
Cons

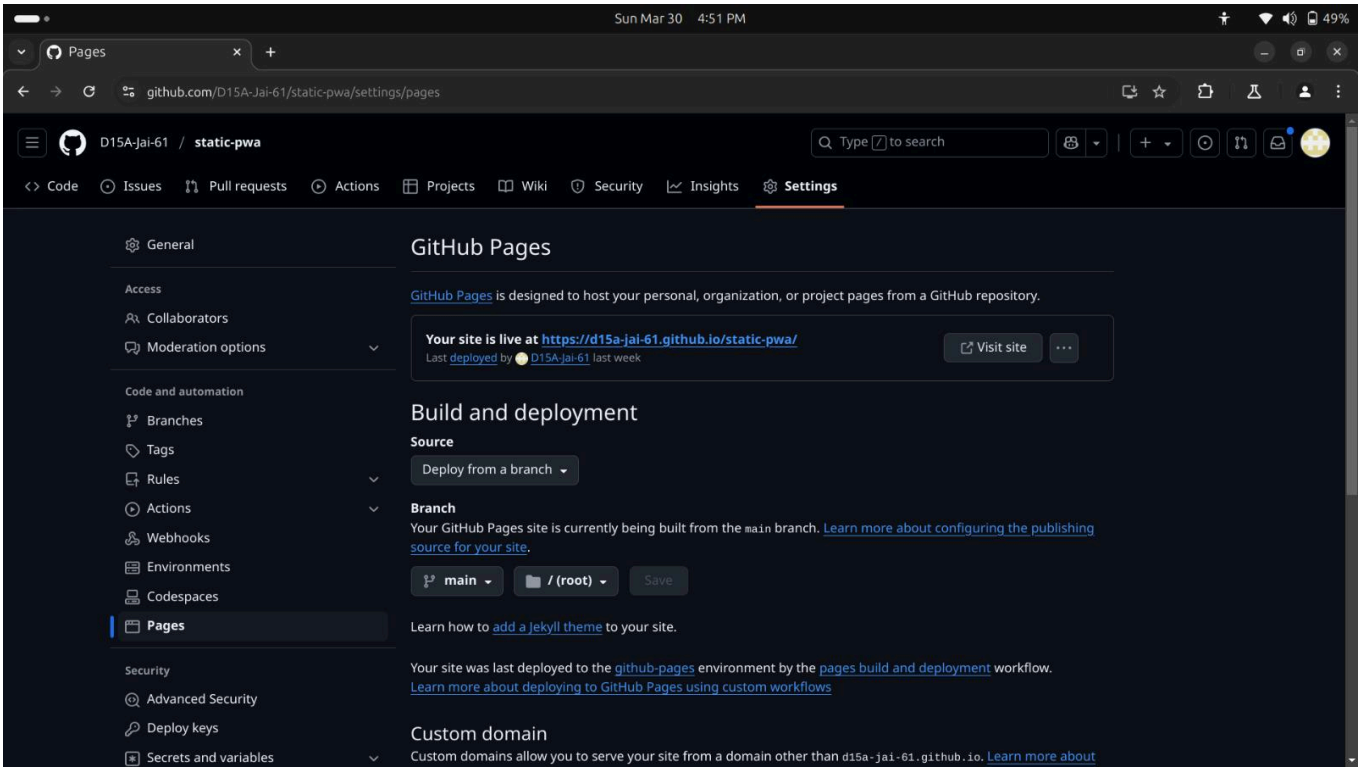
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator.

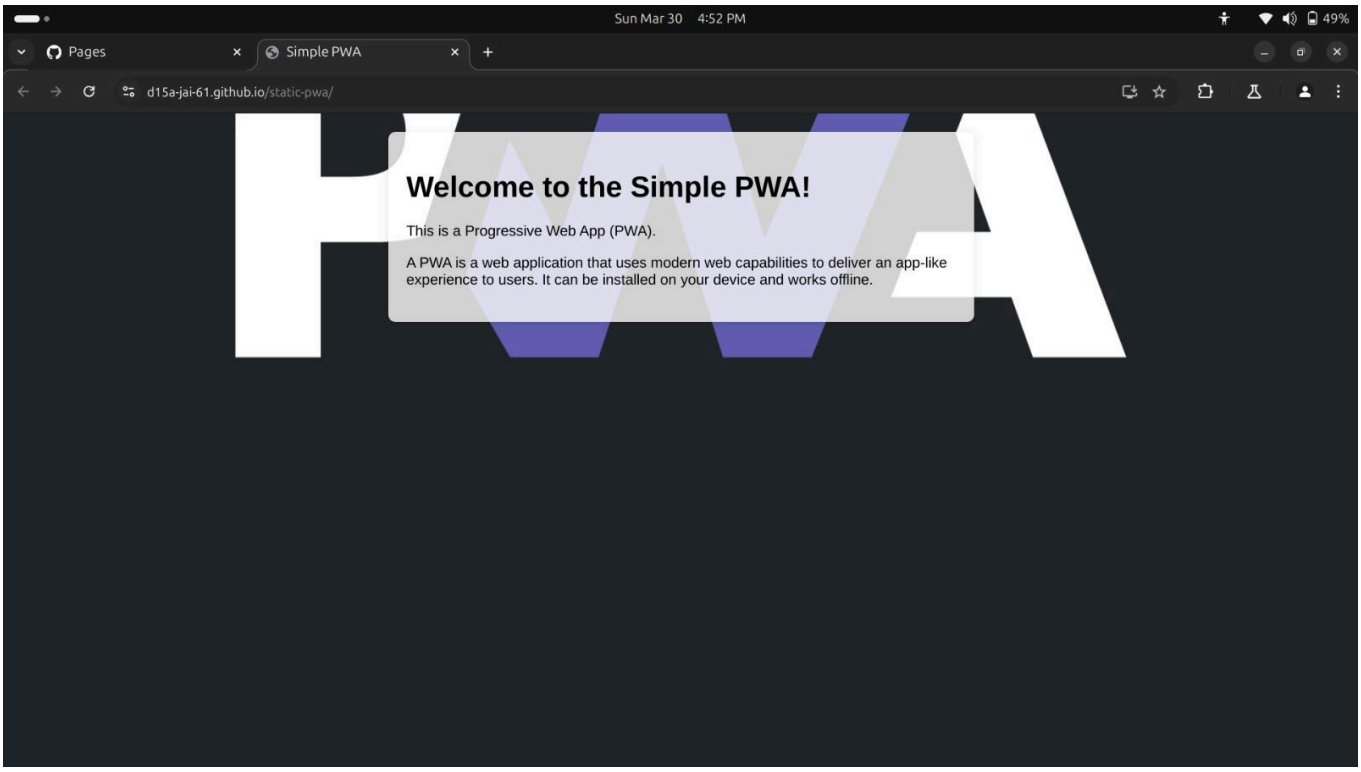
Link to our GitHub repository:

https://github.com/D15A-Jai-61/static-pwa.git

Github Screenshot:


Sun Mar 30 4:51 PM
D15A-Jai-61/static-pwa
github.com/D15A-Jai-61/static-pwa
D15A-Jai-61 / static-pwa
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
static-pwa
Public
Pin
Unwatch 1
Fork 0
Star 0
main
1 Branch
0 Tags
Go to file
Add file
Code
D15A-Jai-61 Updated README file
842fa1b · last week
8 Commits
README.md Updated README file last week
index.html Built the basic, simple, static PWA; Text on web-page nee... last week
manifest.json Added an icon, not being displayed in the webpage, also,... last week
pwa-banner.png Added an icon, not being displayed in the webpage, also,... last week
service-worker.js Added an icon, not being displayed in the webpage, also,... last week
styles.css Edited the image size and repitition last week
README
Simple PWA
About
No description, website, or topics provided.
Readme
Activity
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Deployments 7


Sun Mar 30 4:51 PM
Pages
github.com/D15A-Jai-61/static-pwa/settings/pages
D15A-Jai-61 / static-pwa
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
Security
Advanced Security
Deploy keys
Secrets and variables
GitHub Pages
GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.
Your site is live at https://d15a-jai-61.github.io/static-pwa/
Last deployed by D15A-Jai-61 last week
Visit site
Build and deployment
Source
Deploy from a branch
Branch
Your GitHub Pages site is currently being built from the main branch. Learn more about configuring the publishing source for your site.
main
/ (root)
Save
Learn how to add a Jekyll theme to your site.
Your site was last deployed to the github-pages environment by the pages build and deployment workflow.
Learn more about deploying to GitHub Pages using custom workflows
Custom domain
Custom domains allow you to serve your site from a domain other than d15a-jai-61.github.io. Learn more about


Sun Mar 30 4:52 PM
49%
Pages
Simple PWA
d15a-jai-61.github.io/static-pwa/
Welcome to the Simple PWA!
This is a Progressive Web App (PWA).
A PWA is a web application that uses modern web capabilities to deliver an app-like experience to users. It can be installed on your device and works offline.

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	**63**
Name	**UTEKAR DIKSHA ABHINAY**
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

Experiment 11

Aim :

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

Theory :

Reference : https://www.semrush.com/blog/google-lighthouse/

Google Lighthouse :

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

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

Key Features and Audit Metrics

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

Performance: This score is an aggregation of how the page fared in aspects such as (but not limited to) loading speed, time taken for loading for basic frame(s), displaying meaningful content to the user, etc. To a layman, this score is indicative of how decently the site performs, with a score of 100 meaning that you figure in the 98th percentile, 50 meaning that you figure in the 75th percentile and so on.

PWA Score (Mobile): Thanks to the rise of Service Workers, app manifests, etc., a lot of modern web applications are moving towards the PWA paradigm, where the

objective is to make the application behave as close as possible to native mobile applications. Scoring points are based on the Baseline PWA checklist laid down by Google which includes Service Worker implementation(s), viewport handling, offline functionality, performance in script-disabled environments, etc.

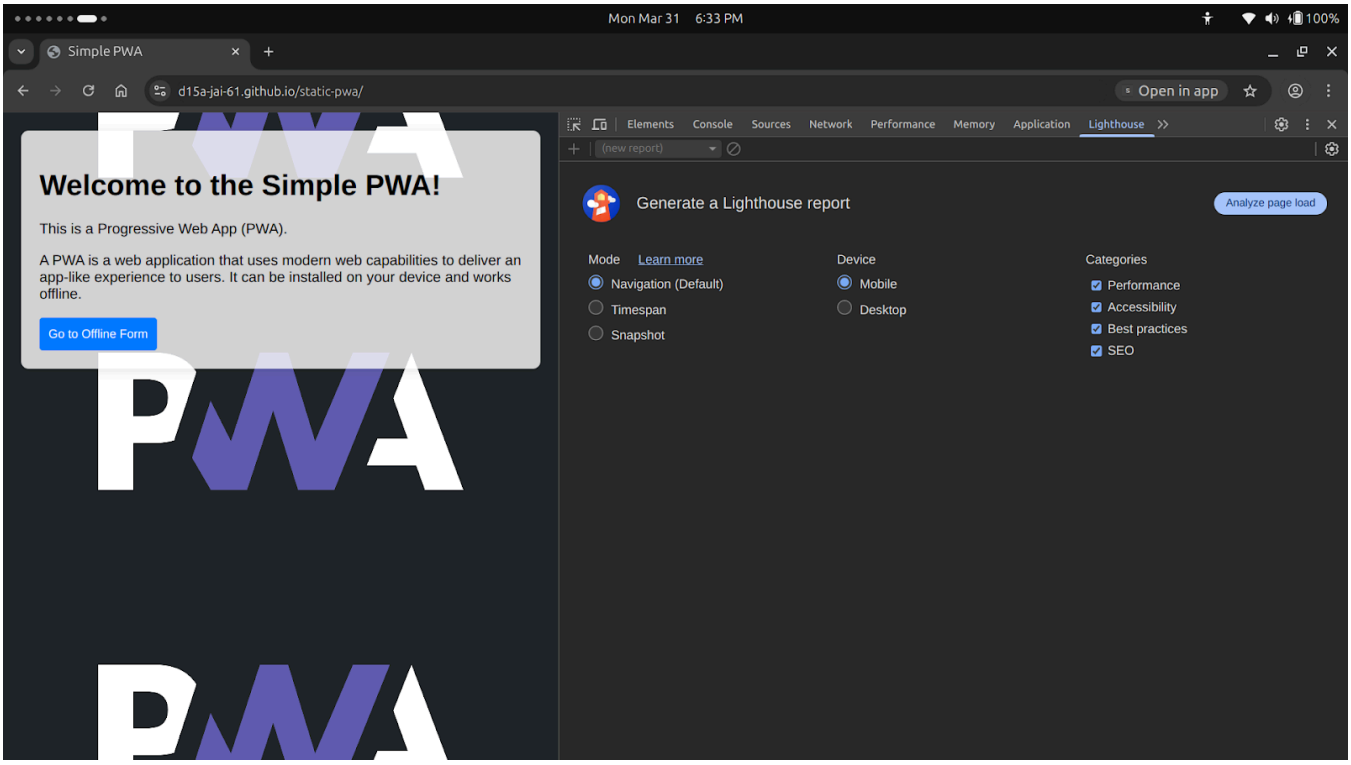
Accessibility: As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics though, Accessibility metrics score on a pass/fail basis i.e. if all possible elements of the page are not screen-reader friendly (HTML5 introduced features that would make pages easy to interpret for screen readers used by visually challenged people like tag names, tags such as <section>, <article>, etc.), you get a 0 on that score. The aggregate of these scores is your Accessibility metric score.

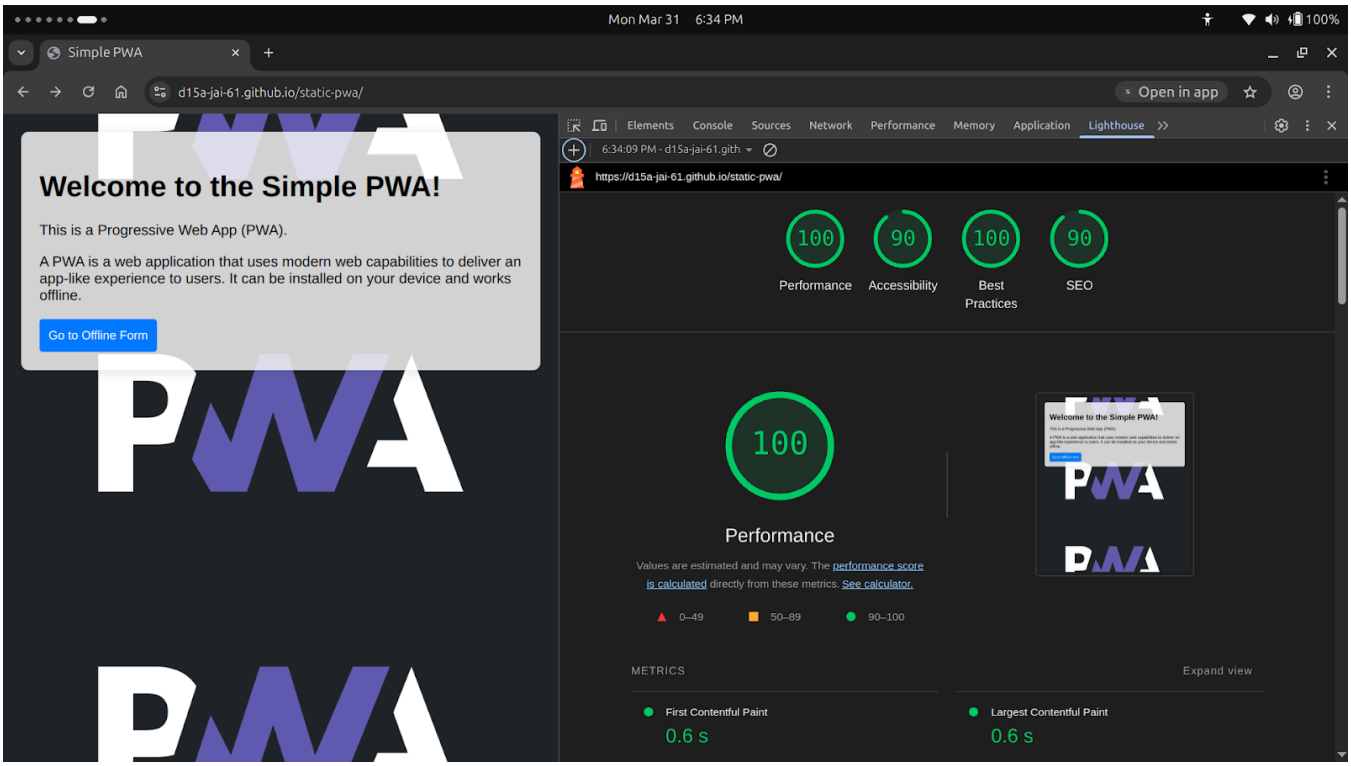
Best Practices: As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS

Avoiding the use of deprecated code elements like tags, directives, libraries, etc. Password input with paste-into disabled

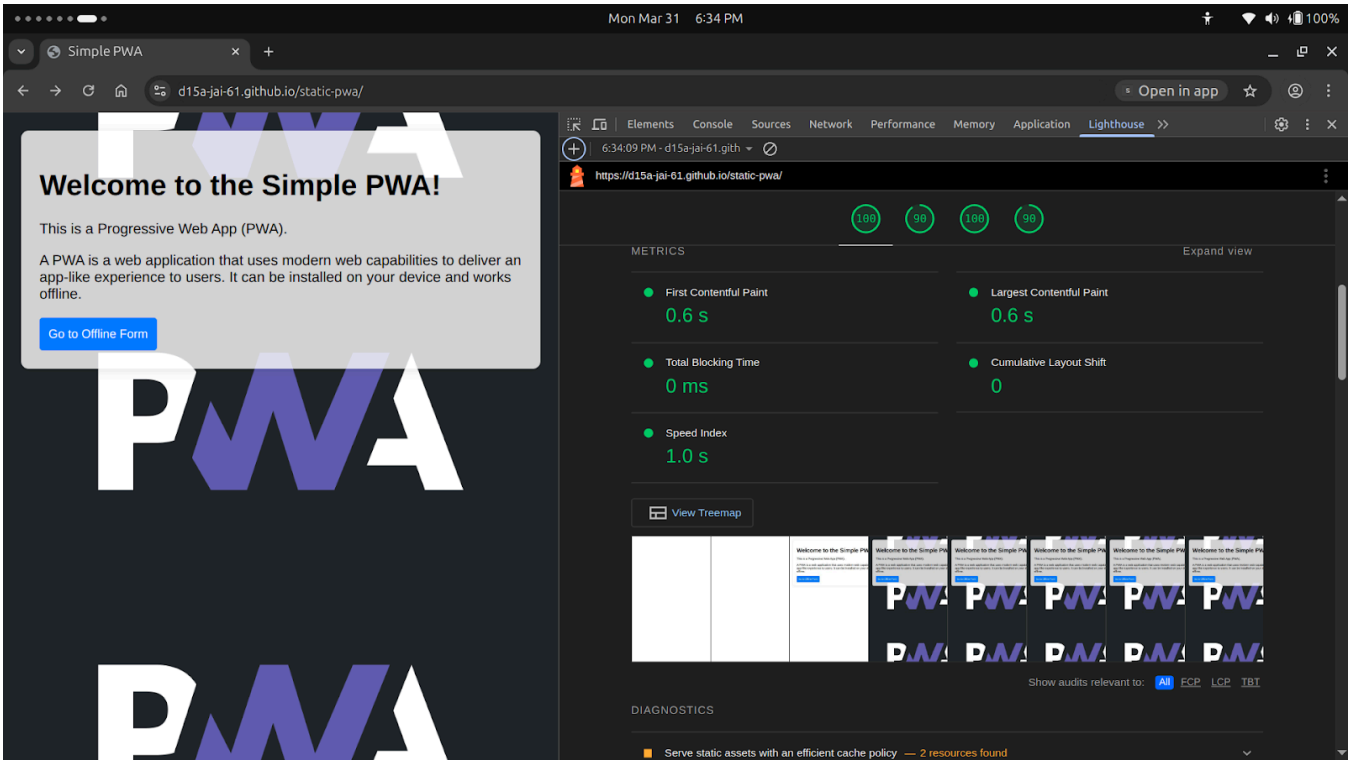
Geo-Location and cookie usage alerts on load, etc.

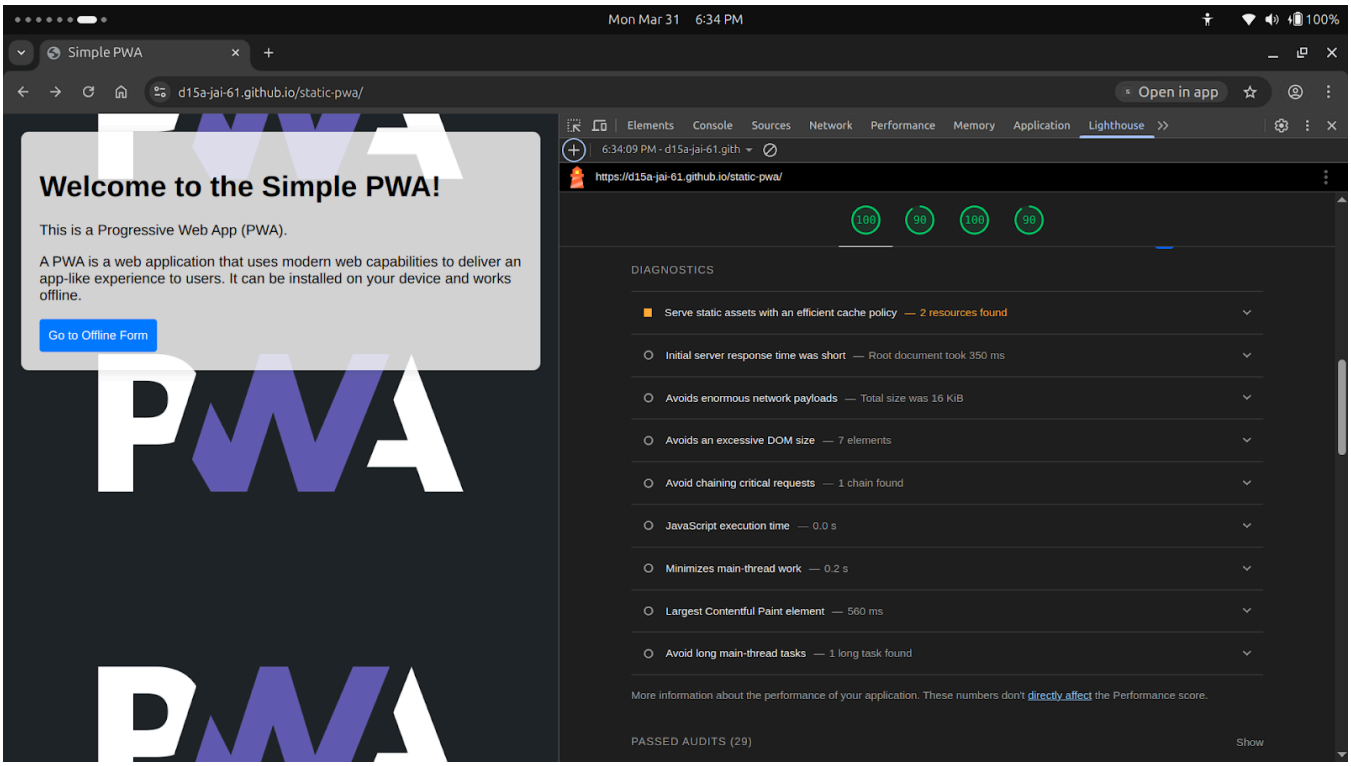
Diksha Utekar Jai Talreja Vedang Wajge D15A-63/59/62



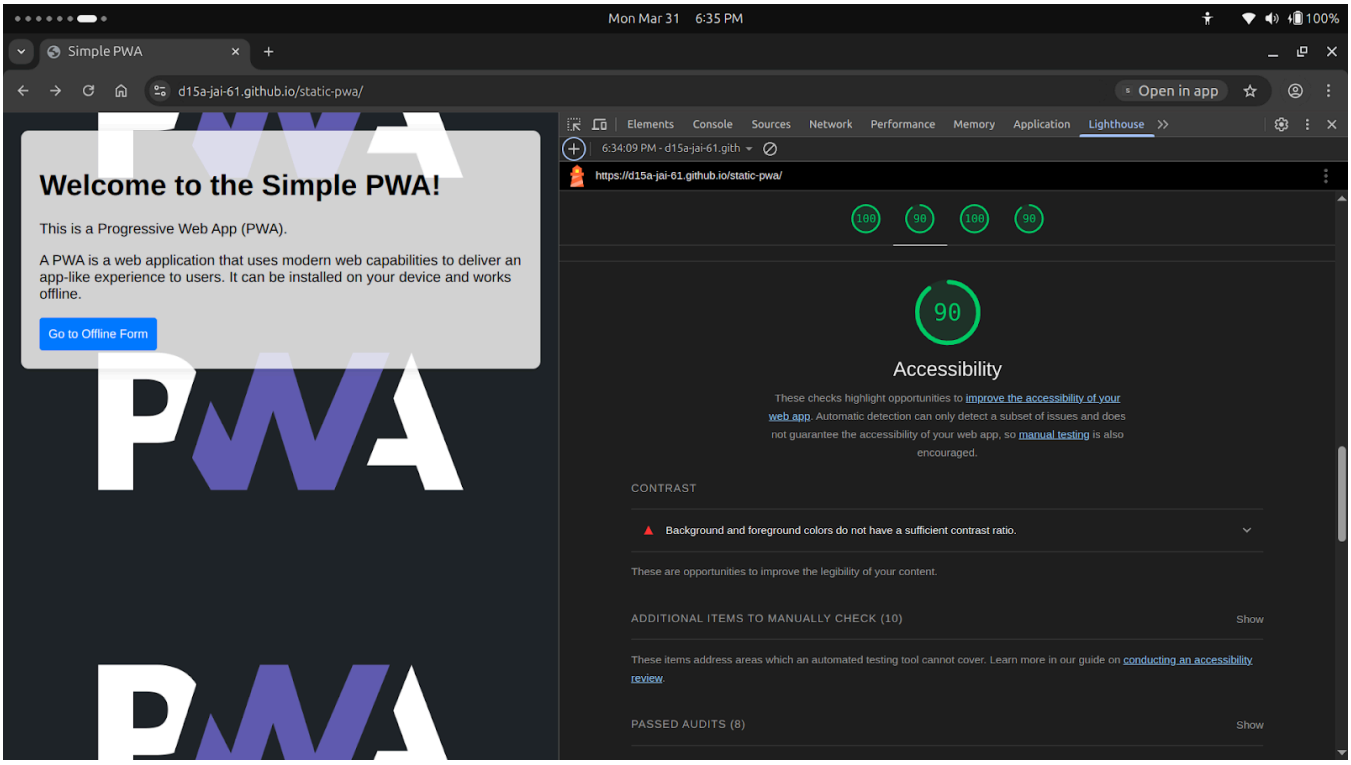


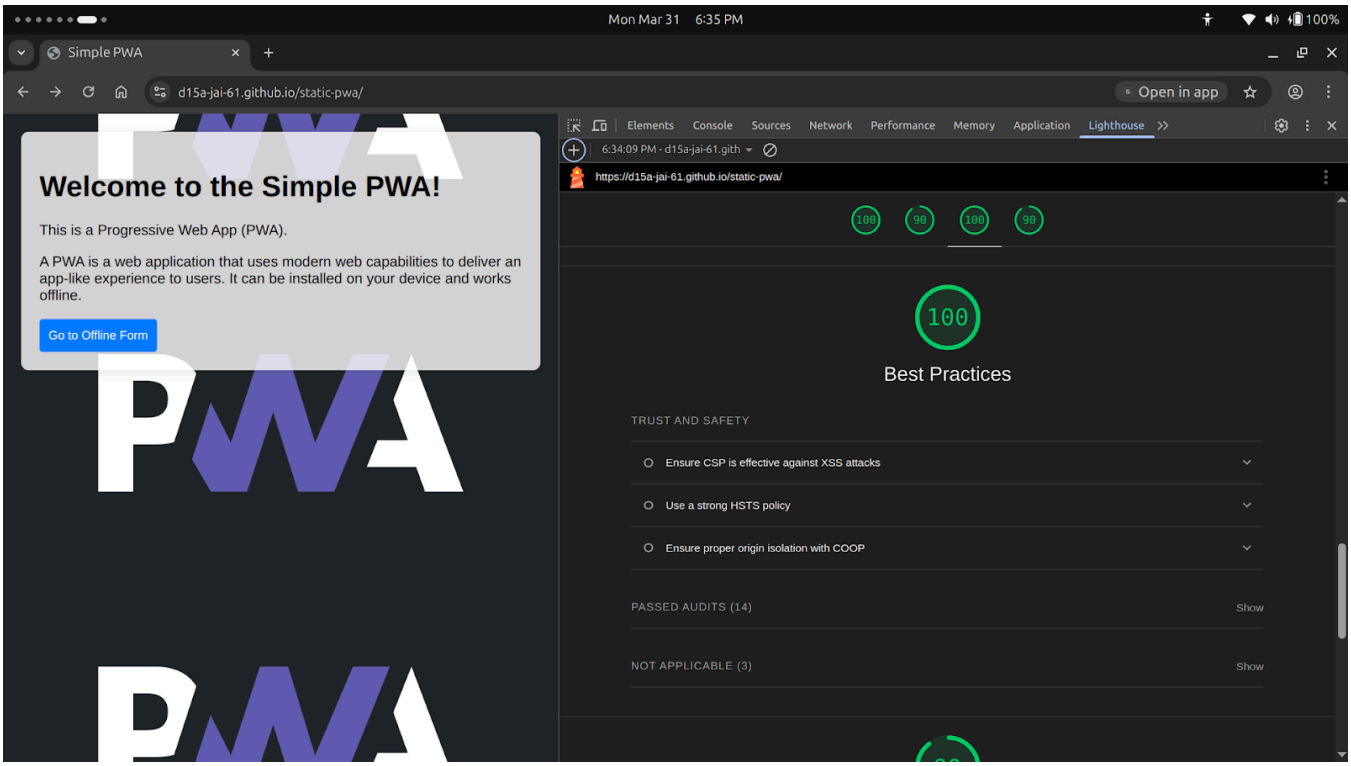
Diksha Utekar Jai Talreja Vedang Wajge D15A-63/59/62

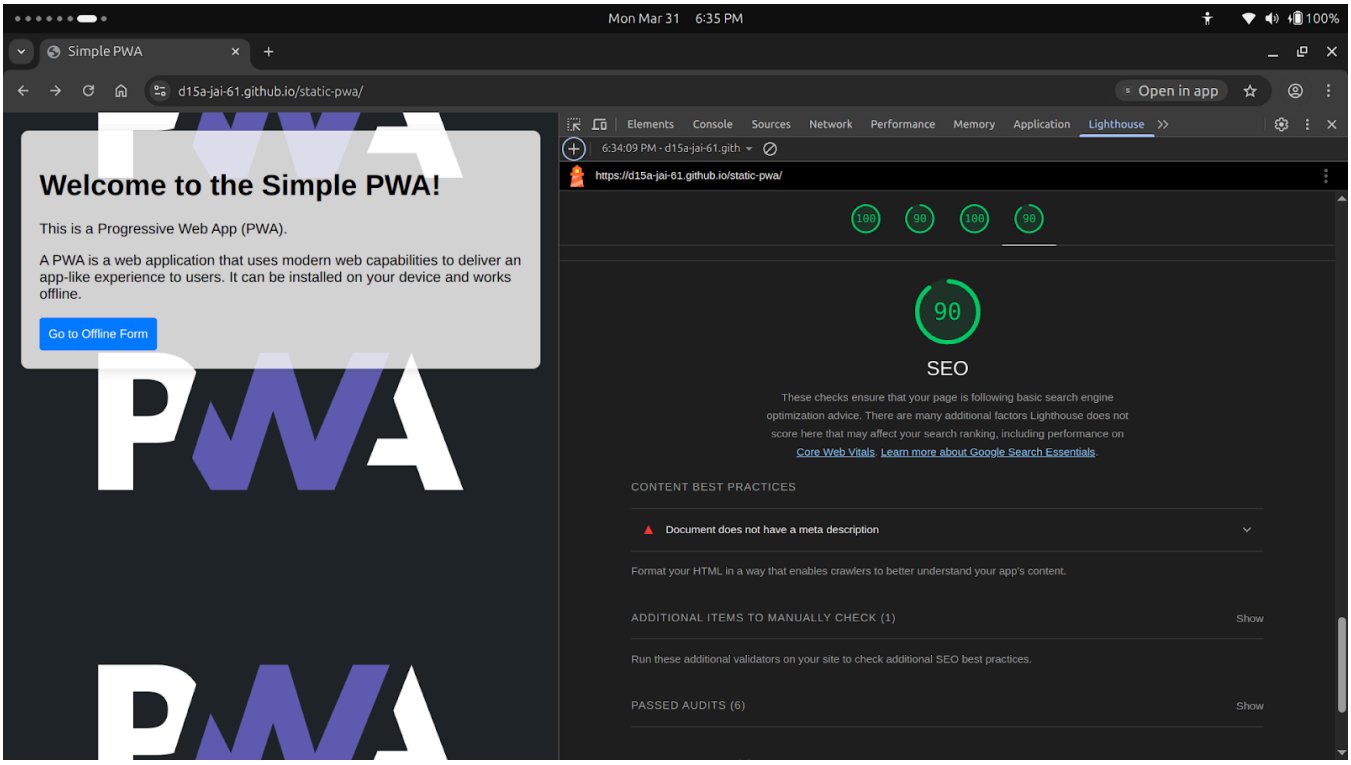




Diksha Utekar Jai Talreja Vedang Wajge D15A-63/59/62







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

MAD & PWA Lab
Journal

Experiment No.	Assignment-1
Assignment 1 Questions	1. Flutter Overview:Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community. 2. Widget Tree and Composition: Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree. 3. State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable. 4. Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.
Roll No.	63
Name	UTEKAR DIKSHA ABHINAY
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.	63
Name	UTEKAR DIKSHA ABHINAY
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:	