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 Prajjwal Pandey of **D15A/D15B** semester **VI,** have successfully completed necessary experiments in the **MAD & PWA Lab** under my supervision in **VES Institute of Technology** during the academic year **2024-2025.**

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Project Title: **Roll No.**

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: **Roll No.**

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

Project Title: **Roll No.**

Index

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

Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	32
Name	Prajjwal Pandey
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

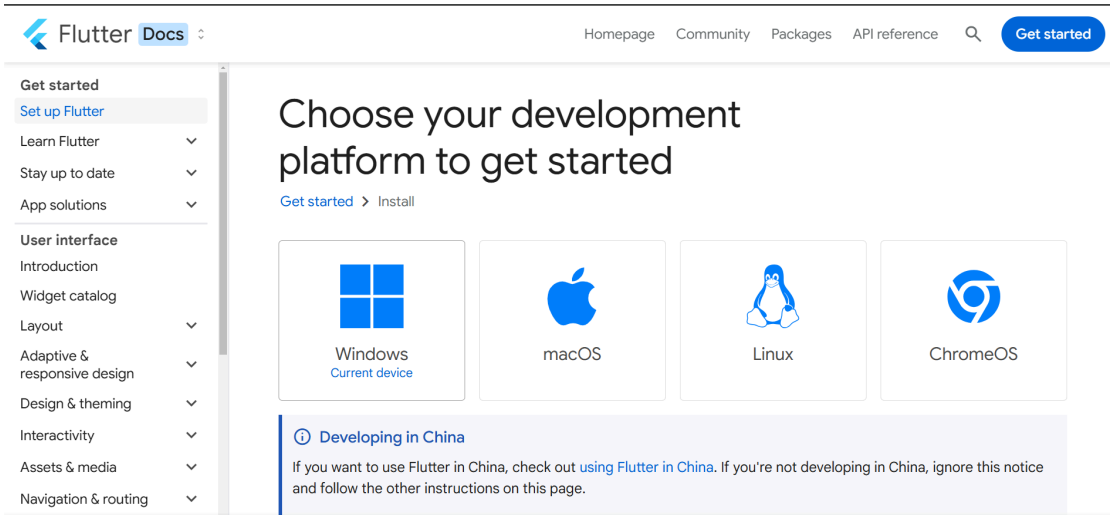
MPL EXPERIMENT 1

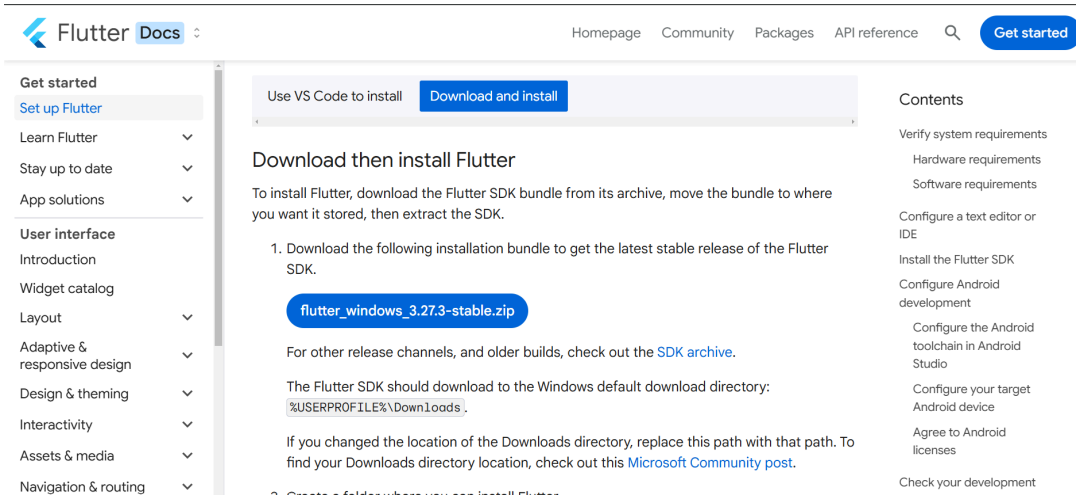
Prajjwal Pandey
D15A / 32

Aim: Installation and Configuration of Flutter Environment.

Step 1: Install Flutter

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

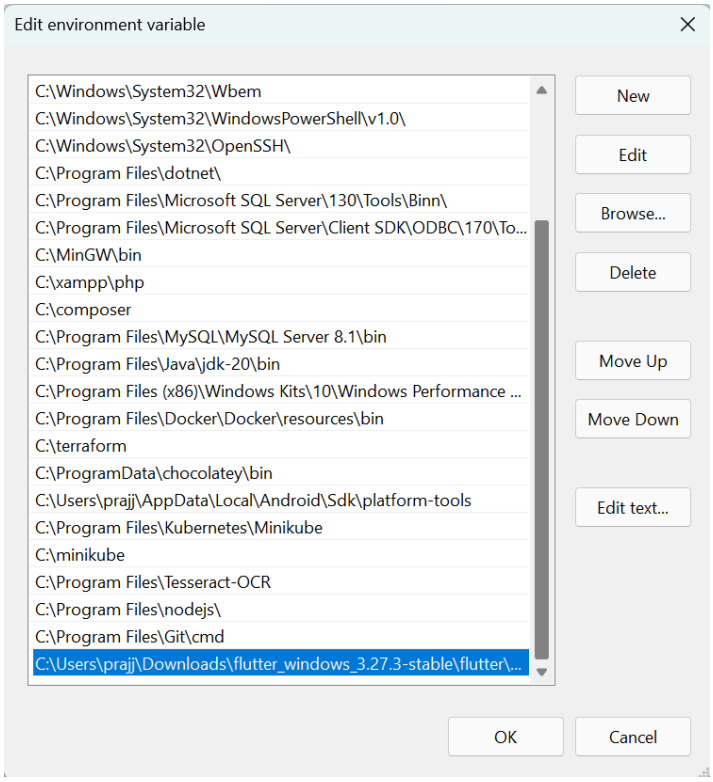




Extract the downloaded zip file to a preferred location on your computer (e.g., C:\src\flutter for Windows).

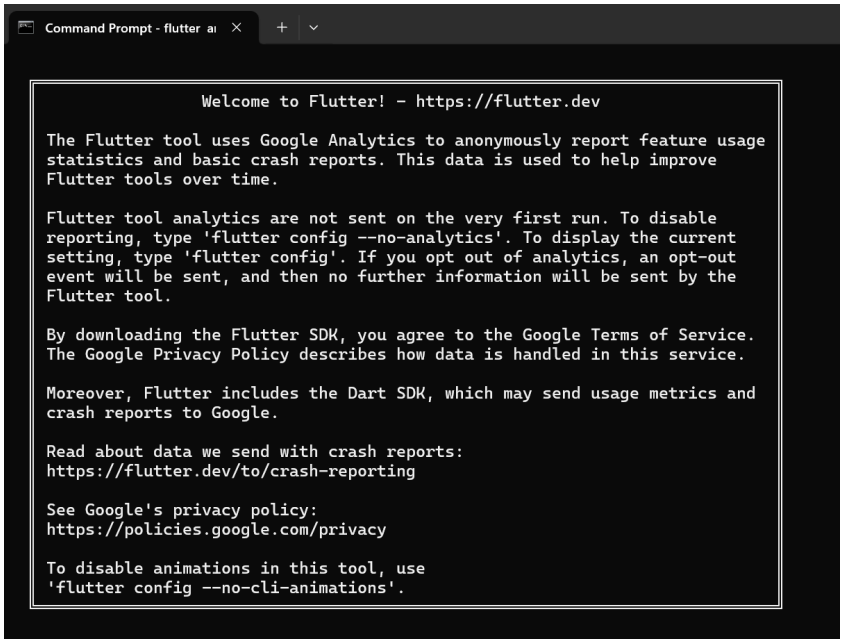
- Add Flutter to the PATH
- Locate the flutter\bin directory in the extracted Flutter folder.

- Add this directory to your system's PATH environment variable.



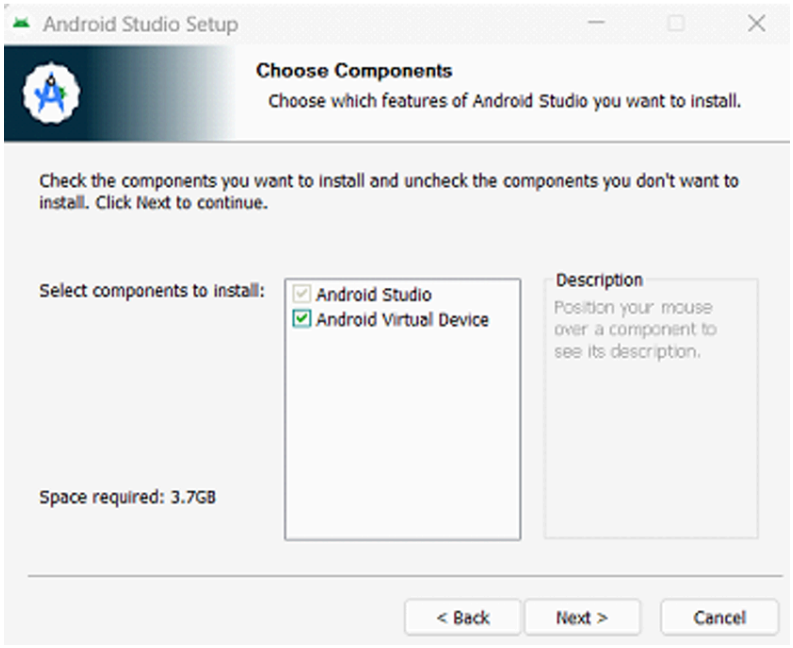
Verify the Installation

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

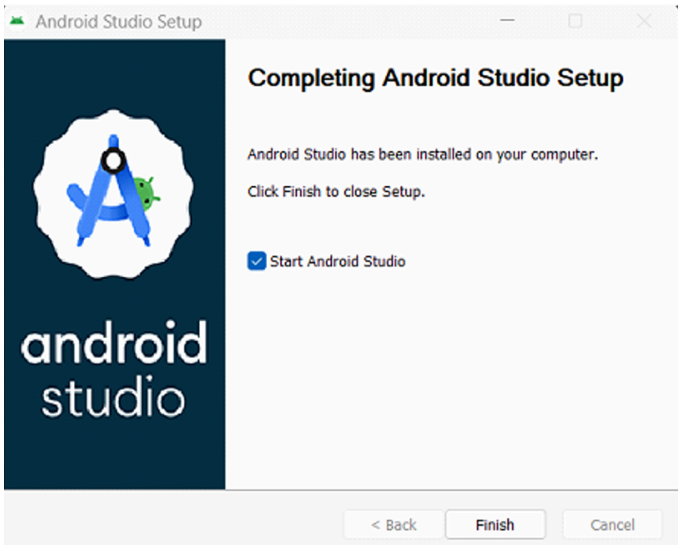


Step 2: Install Android Studio

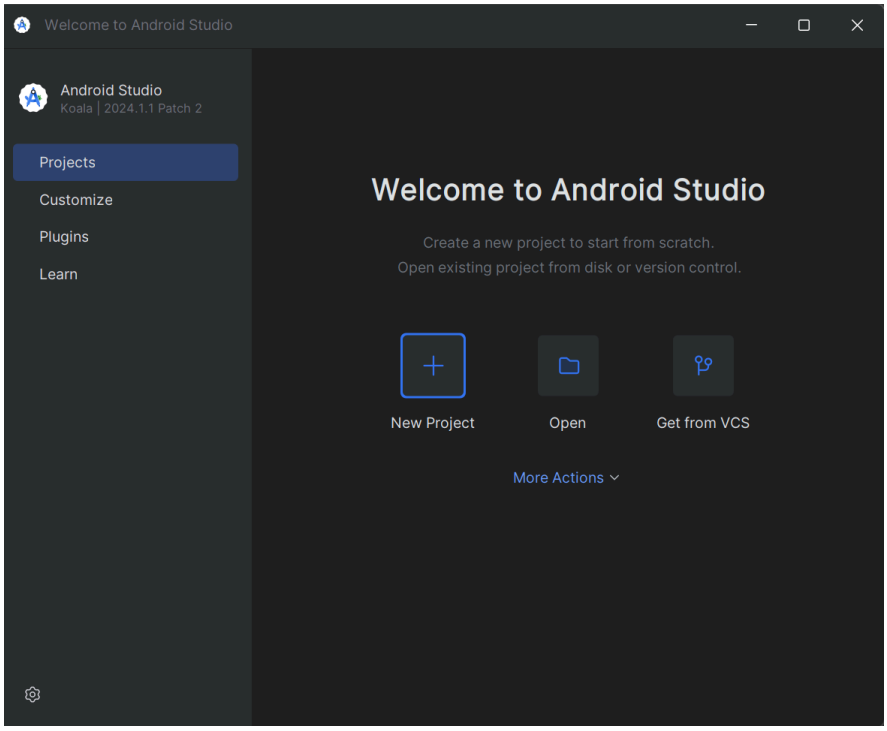
- Download Android Studio
- Go to the Android Studio website. (https://developer.android.com/studio)
- Download the installer for your operating system ((Windows, macOS, or Linux)).



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

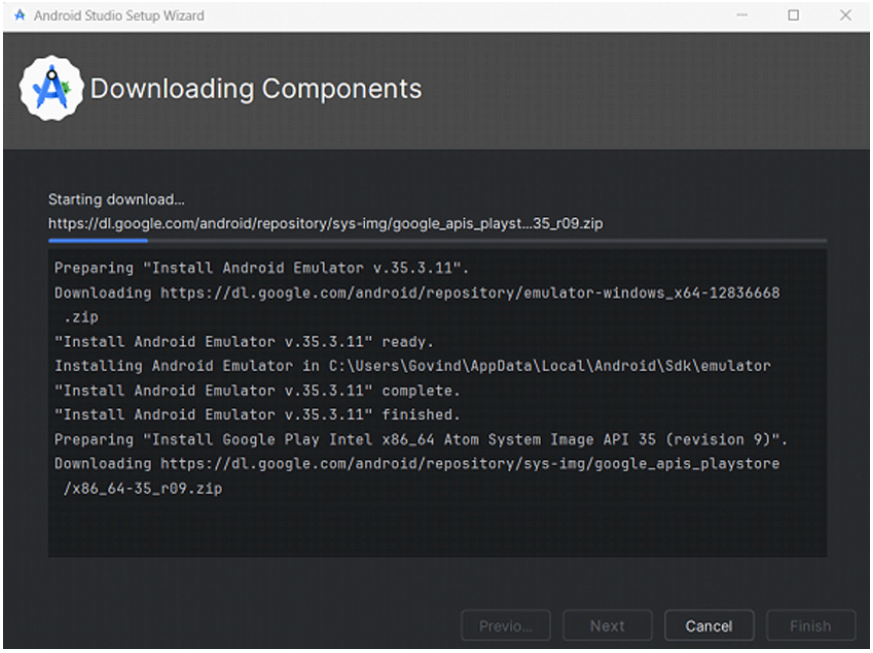


- Install Android SDK Tools
- Open Android Studio.



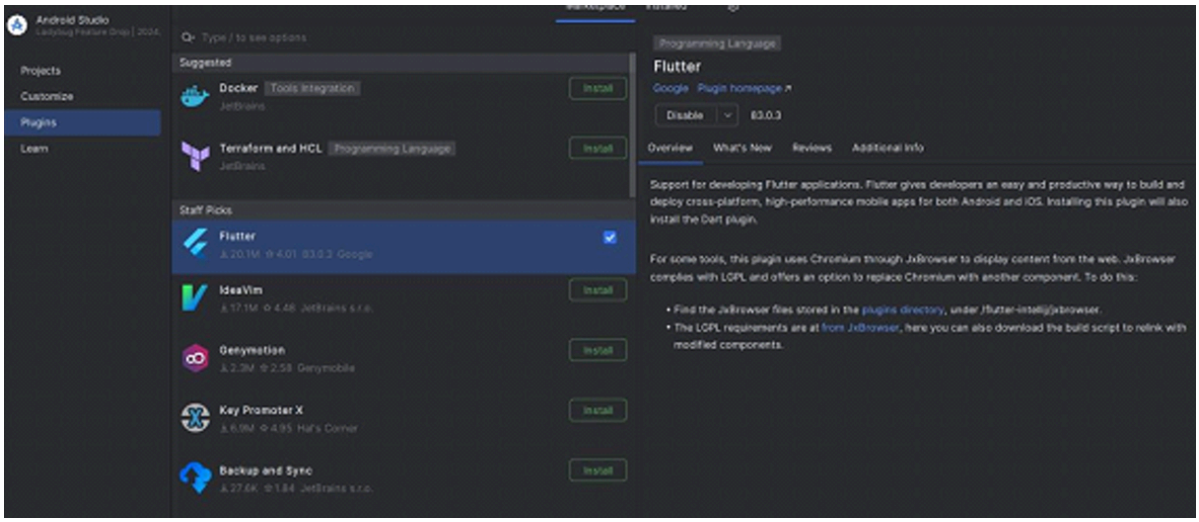
- Go to Settings/Preferences > Language & Framework > Android SDK.
- Select the latest Android API level.
- Ensure "Android SDK Platform" and "Android Virtual Device (AVD)" are selected.

Click "Apply" and wait for the components to install.



Step 3: Connect Flutter with Android Studio

- Install Flutter and Dart Plugins
- Open Android Studio. Go to File > Settings (Windows/Linux) > Plugins.
- Search for "Flutter" and click "Install." Dart will be installed automatically.
- Restart Android Studio.



Step 4: Create a New Flutter Project

- Click on New Flutter Project.
- Enter project details and select the Flutter SDK path.
- Click "Finish" to create the project.

Code :

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: Text("My Flutter App")),
        body: Center(child: Text("Hello, Prajjwal!")),
      ),
    );
  }
}
```

Screenshot:

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

Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	32
Name	Prajjwal Pandey
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

MPL Experiment 02

Prajjwal Pandey
D15A / 32

AIM:

To design a Flutter UI by incorporating common widgets used in mobile application development.

Introduction:

Flutter provides a rich set of widgets to create flexible and visually appealing UIs. In this experiment, we design the user interface of our application using various Flutter widgets, ensuring a smooth user experience.

Common Widgets Used:

Below are the key widgets implemented in the app, along with the pages where they have been used:

1. Scaffold

- **Pages:** Landing Page, Login Page, Signup Page, Home Page, Profile Page
- **Usage:** Provides the basic structure, including an app bar and body layout.

2. Container

- **Pages:** Landing Page, Home Page, Profile Page
- **Usage:** Used for layout customization, including backgrounds and padding.

3. Column & Row

- **Pages:** Landing Page, Login Page, Signup Page, Profile Page
- **Usage:** Aligns widgets vertically (Column) and horizontally (Row) for structured UI design.

4. Text

- **Pages:** Landing Page, Login Page, Signup Page, Home Page, Profile Page
- **Usage:** Displays static and dynamic text content.

5. TextField

- **Pages:** Login Page, Signup Page, Profile Page
- **Usage:** Captures user input for authentication and profile details.

6. ElevatedButton

- **Pages:** Landing Page, Login Page, Signup Page, Profile Page
- **Usage:** Implements user actions like login, signup, and logout.

7. ListView

- **Pages:** Home Page, Discover Page
- **Usage:** Shows a list of live streams, reels, or other content.

CODE:

// landing_page.dart, LoginPage(), SignupPage.dart()

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

import 'package:shared_preferences/shared_preferences.dart';

import 'package:twitch_mpl_lab/pages/home_page.dart';

import 'package:firebase_auth/firebase_auth.dart';



class LandingPage extends StatelessWidget {

  final VoidCallback onLoginSuccess;

  LandingPage({required this.onLoginSuccess});



  Future<void> _setLoginStatus(BuildContext context) async {

    SharedPreferences prefs = await SharedPreferences.getInstance();

    await prefs.setBool('isLoggedIn', true);

    onLoginSuccess();
```

```
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (context) => HomePage()),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: [
          // Background Image
          Positioned.fill(
            child: Image.asset(
              'assets/images/landingbg.png', // Make sure this image is placed in assets folder
              fit: BoxFit.cover,
            ),
          ),
          // Gradient Overlay for better readability
          Positioned.fill(
            child: Container(
              decoration: BoxDecoration(
                gradient: LinearGradient(
```

```
                colors: [Colors.black.withOpacity(0.5),
Colors.black.withOpacity(0.7)],
                begin: Alignment.topCenter,
                end: Alignment.bottomCenter,
              ),
            ),
          ),
        ),
        // Content
        Center(
          child: Padding(
            padding: EdgeInsets.symmetric(horizontal: 20),
            child: Column(
              mainAxisSize: MainAxisSize.min,
              children: [
                Text(
                  "There's something for you on Twitch",
                  style: TextStyle(
                    fontSize: 24,
                    fontWeight: FontWeight.bold,
                    color: Colors.white,
                  ),
                  textAlign: TextAlign.center,
                ),
```

```
                SizedBox(height: 40),
                // Buttons Row
                Row(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: [
                    // Log In Button (White with Opacity)
                    ElevatedButton(
                      onPressed: () {
                        Navigator.push(
                          context,
                          MaterialPageRoute(
                            builder: (context) =>
LoginPage(onLoginSuccess: () => _setLoginStatus(context)),
                          ),
                        );
                      },
                      style: ElevatedButton.styleFrom(
                        backgroundColor: Colors.white.withOpacity(0.7),
                        foregroundColor: Colors.black,
                        padding: EdgeInsets.symmetric(horizontal: 40,
vertical: 15),
                      ),
                      child: Text("Log In"),
                    ),
```

```
                    SizedBox(width: 20), // Spacing between buttons
                    // Sign Up Button (Solid White)
                    ElevatedButton(
                      onPressed: () {
                        Navigator.push(
                          context,
                          MaterialPageRoute(
                            builder: (context) =>
SignupPage(onSignupSuccess: () => _setLoginStatus(context)),
                          ),
                        );
                      },
                      style: ElevatedButton.styleFrom(
                        backgroundColor: Colors.white,
                        foregroundColor: Colors.black,
                        padding: EdgeInsets.symmetric(horizontal: 40,
vertical: 15),
                      ),
                      child: Text("Sign Up"),
                    ),
                  ],
                ),
              ],
            ),
```

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


class LoginPage extends StatefulWidget {
  final VoidCallback onLoginSuccess;

  LoginPage({required this.onLoginSuccess});

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


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

  Future<void> _login() async {
```

```
    try {
      await _auth.signInWithEmailAndPassword(
        email: _emailController.text.trim(),
        password: _passwordController.text.trim(),
      );
      widget.onLoginSuccess();
    } catch (e) {
      print("Login failed: $e");
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("Login failed: ${e.toString()}")),
      );
    }
  }


  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Log In")),
      body: Padding(
        padding: EdgeInsets.all(20),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
```

```
            TextField(
              controller: _emailController,
              decoration: InputDecoration(labelText: "Email"),
            ),
            TextField(
              controller: _passwordController,
              decoration: InputDecoration(labelText: "Password"),
              obscureText: true,
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: _login,
              child: Text("Log In"),
            ),
          ],
        ),
      ),
    );
  }
}


class SignupPage extends StatefulWidget {
  final VoidCallback onSignupSuccess;
```

```
  SignupPage({required this.onSignupSuccess});

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

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

  Future<void> _signup() async {
    try {
      await _auth.createUserWithEmailAndPassword(
        email: _emailController.text.trim(),
        password: _passwordController.text.trim(),
      );
      widget.onSignupSuccess();
    } catch (e) {
      print("Signup failed: $e");
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("Signup failed: ${e.toString()}")),
```

```
    );
  }
}


@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text("Sign Up")),
    body: Padding(
      padding: EdgeInsets.all(20),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          TextField(
            controller: _emailController,
            decoration: InputDecoration(labelText: "Email"),
          ),
          TextField(
            controller: _passwordController,
            decoration: InputDecoration(labelText: "Password"),
            obscureText: true,
          ),
          SizedBox(height: 20),
```

```
            ElevatedButton(
              onPressed: _signup,
              child: Text("Sign Up"),
            ),
          ],
        ),
      ),
    );
  }
}
```

// discover_page.dart

```
@override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      body: SafeArea(
        child: Column(
          children: [
            // Search bar
            Padding(
              padding: const EdgeInsets.all(16.0),
              child: Container(
```

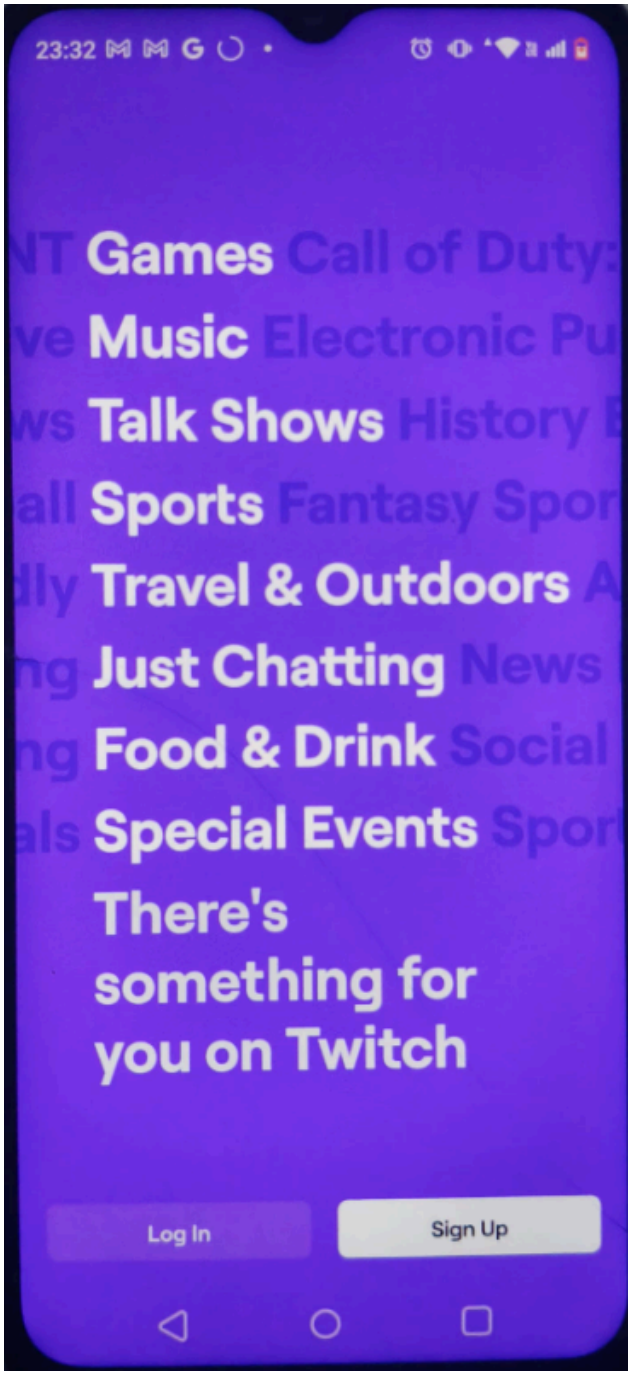
```
        height: 48,
        decoration: BoxDecoration(
          color: const Color(0xFF303030),
          borderRadius: BorderRadius.circular(8),
        ),
        child: Row(
          children: [
            const SizedBox(width: 12),
            Icon(Icons.search, color: Colors.grey[400], size: 24),
            const SizedBox(width: 12),
          ],
        ),
      ),
    ),


    // Tab navigation
    Padding(
      padding: const EdgeInsets.symmetric(horizontal: 16.0),
      child: Row(
        children: [
          Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
```

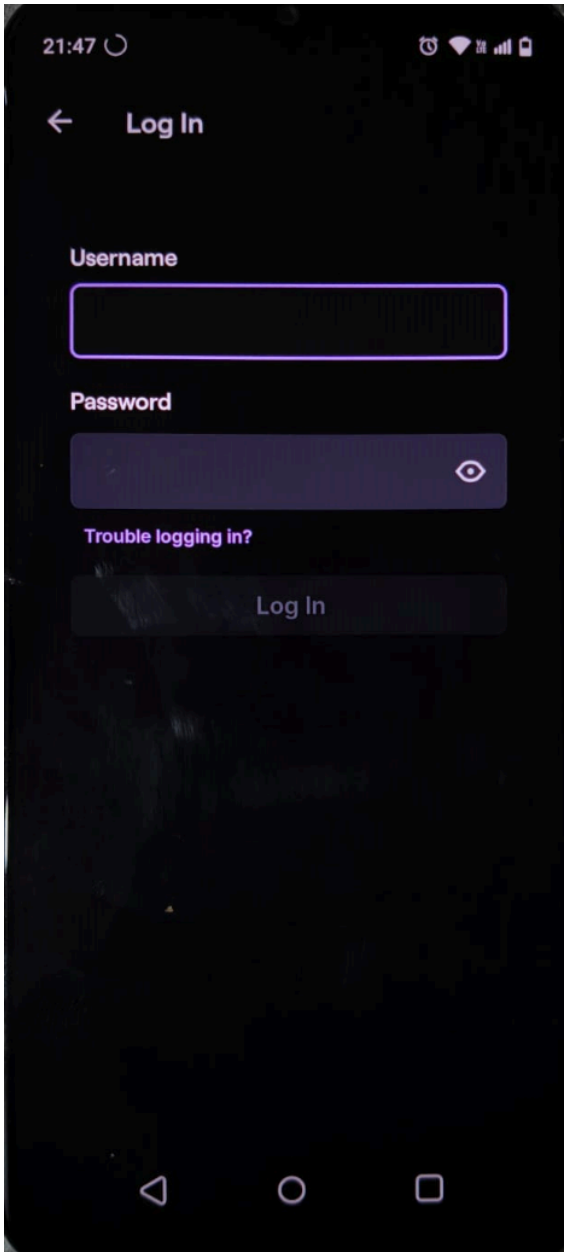
```
                    const Text(
                      'Categories',
                      style: TextStyle(
                        color: Color(0xFFB19DD0), // Light purple
                        fontSize: 18,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                    const SizedBox(height: 4),
                    Container(
                      width: 124,
                      height: 3,
                      color: const Color(0xFF9147FF), // Twitch purple
                    ),
                  ],
                ),
                const SizedBox(width: 24),
                const Text(
                  'Live Channels',
                  style: TextStyle(
                    color: Colors.white,
                    fontSize: 18,
                    fontWeight: FontWeight.bold,
```

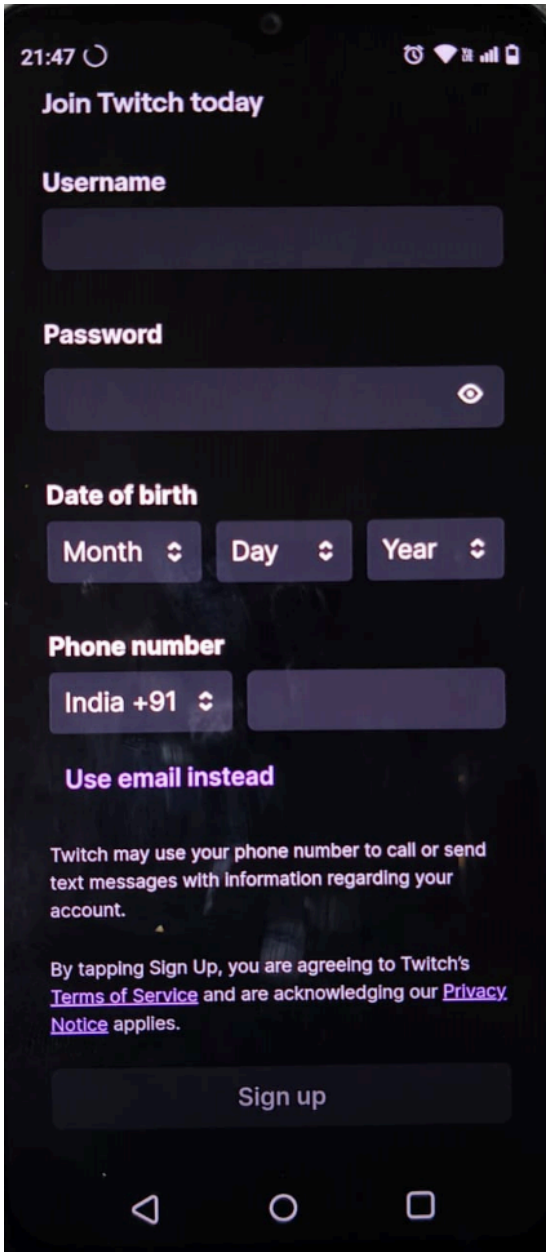
```
                    ),
                  ),
                  const Spacer(),
                  Icon(Icons.tune, color: Colors.grey[400]),
                ],
              ),
            ),

            const SizedBox(height: 16),

            // Game categories grid
            Expanded(
              child: isLoading
                  ? const Center(
                      child: CircularProgressIndicator(
                        color: Color(0xFF9147FF), // Twitch purple
                      ),
                    )
                  : GridView.builder(
                      padding: const EdgeInsets.symmetric(horizontal: 16),
                      itemCount: categories.length,
                      gridDelegate: const
SliverGridDelegateWithFixedCrossAxisCount(
                        crossAxisCount: 3,
```

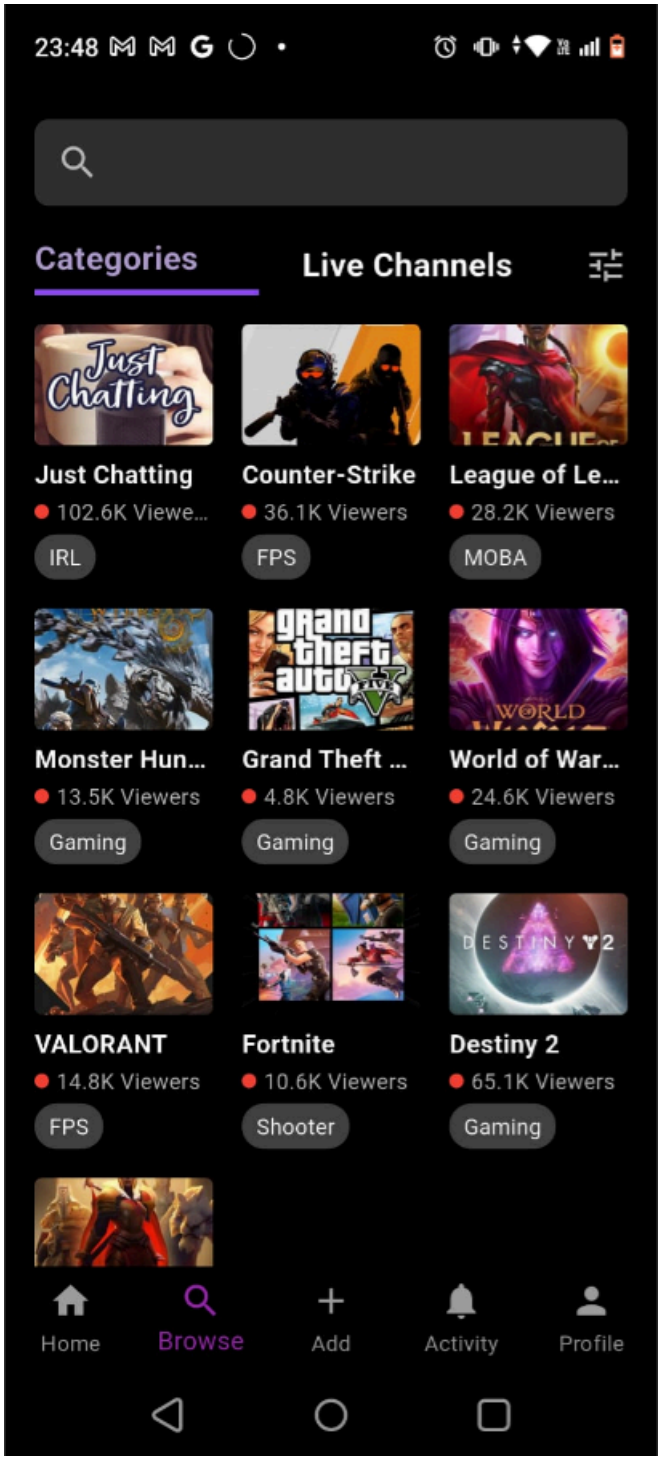
```
                        mainAxisSpacing: 16,
                        crossAxisSpacing: 16,
                        childAspectRatio: 0.7,
                      ),
                      itemBuilder: (context, index) {
                        final category = categories[index];
                        return _buildCategoryCard(
                          category['title']!,
                          category['image']!,
                          category['viewers']!,
                          category['categoryTag']!,
                        );
                      },
                    ),
            ),


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

OUTPUT:


23:32
Games
Music
Talk Shows
Sports
Travel & Outdoors
Just Chatting
Food & Drink
Special Events
There's something for you on Twitch
Log In
Sign Up


21:47
Log In
Username
Password
Trouble logging in?
Log In


21:47
Join Twitch today
Username
Password
Date of birth
Month
Day
Year
Phone number
India +91
Use email instead
Twitch may use your phone number to call or send text messages with information regarding your account.
By tapping Sign Up, you are agreeing to Twitch's Terms of Service and are acknowledging our Privacy Notice applies.
Sign up


23:48
Categories
Live Channels
Just Chatting
Just Chatting
102.6K Viewe...
IRL
Counter-Strike
36.1K Viewers
FPS
League of Le...
28.2K Viewers
MOBA
Monster Hun...
13.5K Viewers
Gaming
Grand Theft ...
4.8K Viewers
Gaming
World of War...
24.6K Viewers
Gaming
VALORANT
14.8K Viewers
FPS
Fortnite
10.6K Viewers
Shooter
Destiny 2
65.1K Viewers
Gaming
Home
Browse
Add
Activity
Profile

Conclusion:

By using these common widgets, we were able to create an interactive and visually appealing UI in Flutter. These widgets allow for flexible customization and efficient UI design for mobile applications.

Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	32
Name	Prajjwal Pandey
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

MPL Experiment 03

Prajjwal Pandey

D15A / 32

AIM: Including Icons, Images, and Fonts in Flutter App

Icons Used and Their Pages:

1. **Home Page**
    - **Icons.home**: Navigation bar
    - **Icons.search**: Search functionality
    - **Icons.notifications**: Notification icon
    - **Icons.person**: Profile section
2. **Discover Page**
    - **Icons.videocam**: Live stream indicator
    - **Icons.chat**: Chat option for streams
3. **Profile Page**
    - **Icons.settings**: Settings option
    - **Icons.logout**: Logout button

Images Used and Their Pages:

1. **Landing Page**
    - **assets/images/landing_bg.png**: Background image for landing screen
2. **Home Page**
    - **assets/images/reel_placeholder.png**: Placeholder for reel thumbnails
    - **assets/images/user_avatar.png**: Default profile avatar
3. **Profile Page**
    - **assets/images/profile_banner.png**: Profile banner

Fonts Used and Their Pages:

1. **General App Theme**
    - **Poppins** (Regular, Medium, Bold): Used across all pages for titles and body text
    - **Roboto** (Regular): Used for button labels and smaller text
2. **Landing Page & Authentication Screens**
    - **Poppins Bold**: Main heading on landing page
3. **Profile Page**
    - **Poppins Medium**: Used for username and profile details

Implementation Details:

- **Icons**: Implemented using `Icons` class from Flutter's material package.
- **Images**: Added in `assets/` folder and listed in `pubspec.yaml` under `flutter -> assets`.
- **Fonts**: Defined in `pubspec.yaml` under `flutter -> fonts` section.

CODE:

//categories_page.dart

```
@override

  Widget build(BuildContext context) {

    return Scaffold(

      backgroundColor: Colors.black,

      body: SafeArea(

        child: Column(

          children: [

            // Search bar

            Padding(

              padding: const EdgeInsets.all(16.0),

              child: Container(

                height: 48,

                decoration: BoxDecoration(

                  color: const Color(0xFF303030),

                  borderRadius: BorderRadius.circular(8),

                ),
```

```
            child: Row(
              children: [
                const SizedBox(width: 12),
                Icon(Icons.search, color: Colors.grey[400], size: 24),
                const SizedBox(width: 12),
              ],
            ),
          ),
        ),

        // Tab navigation
        Padding(
          padding: const EdgeInsets.symmetric(horizontal: 16.0),
          child: Row(
            children: [
              Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  const Text(
                    'Categories',
                    style: TextStyle(
                      color: Color(0xFFB19DD0), // Light purple
                      fontSize: 18,
```

```
                    fontWeight: FontWeight.bold,
                  ),
                ),
                const SizedBox(height: 4),
                Container(
                  width: 124,
                  height: 3,
                  color: const Color(0xFF9147FF), // Twitch purple
                ),
              ],
            ),
            const SizedBox(width: 24),
            const Text(
              'Live Channels',
              style: TextStyle(
                color: Colors.white,
                fontSize: 18,
                fontWeight: FontWeight.bold,
              ),
            ),
            const Spacer(),
            Icon(Icons.tune, color: Colors.grey[400]),
          ],
```

```
              ),
            ),

            const SizedBox(height: 16),

            // Game categories grid
            Expanded(
              child: isLoading
                  ? const Center(
                      child: CircularProgressIndicator(
                        color: Color(0xFF9147FF), // Twitch purple
                      ),
                    )
                  : GridView.builder(
                      padding: const EdgeInsets.symmetric(horizontal: 16),
                      itemCount: categories.length,
                      gridDelegate: const
SliverGridDelegateWithFixedCrossAxisCount(
                        crossAxisCount: 3,
                        mainAxisSpacing: 16,
                        crossAxisSpacing: 16,
                        childAspectRatio: 0.7,
                      ),
                      itemBuilder: (context, index) {
```

```
                        final category = categories[index];
                        return _buildCategoryCard(
                          category['title']!,
                          category['image']!,
                          category['viewers']!,
                          category['categoryTag']!,
                        );
                      },
                    ),
              ),


            ],
          ),
        ),
      );
  }


  Widget _buildCategoryCard(String title, String imageUrl, String viewers,
String categoryTag) {
    return GestureDetector(
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(
```

```
          builder: (context) => GameDetailsPage(
            title: title,
            imageUrl: imageUrl,
          ),
        ),
      );
    },
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        // Game thumbnail
        Expanded(
          child: ClipRRect(
            borderRadius: BorderRadius.circular(4),
            child: Image.network(
              imageUrl,
              fit: BoxFit.cover,
              width: double.infinity,
              loadingBuilder: (context, child, loadingProgress) {
                if (loadingProgress == null) return child;
                return Container(
                  color: Colors.grey[800],
                  child: Center(
```

```
                  child: CircularProgressIndicator(
                    color: const Color(0xFF9147FF),
                    value: loadingProgress.expectedTotalBytes != null
                        ? loadingProgress.cumulativeBytesLoaded /
                            loadingProgress.expectedTotalBytes!
                        : null,
                  ),
                ),
              );
            },
            errorBuilder: (context, error, stackTrace) {
              return Container(
                color: Colors.grey[800],
                child: const Center(
                  child: Icon(Icons.error, color: Colors.white),
                ),
              );
            },
          ),
        ),
      ),

      const SizedBox(height: 6),
```

```
          // Game title
          Text(
            title,
            style: const TextStyle(
              color: Colors.white,
              fontSize: 14,
              fontWeight: FontWeight.bold,
            ),
            maxLines: 1,
            overflow: TextOverflow.ellipsis,
          ),

          const SizedBox(height: 2),

          // Viewer count - FIX: Wrap in a SizedBox with width constraint
to prevent overflow
          SizedBox(
            width: 98, // Set a fixed width that fits within the card
            child: Row(
              children: [
                Container(
                  width: 8,
                  height: 8,
```

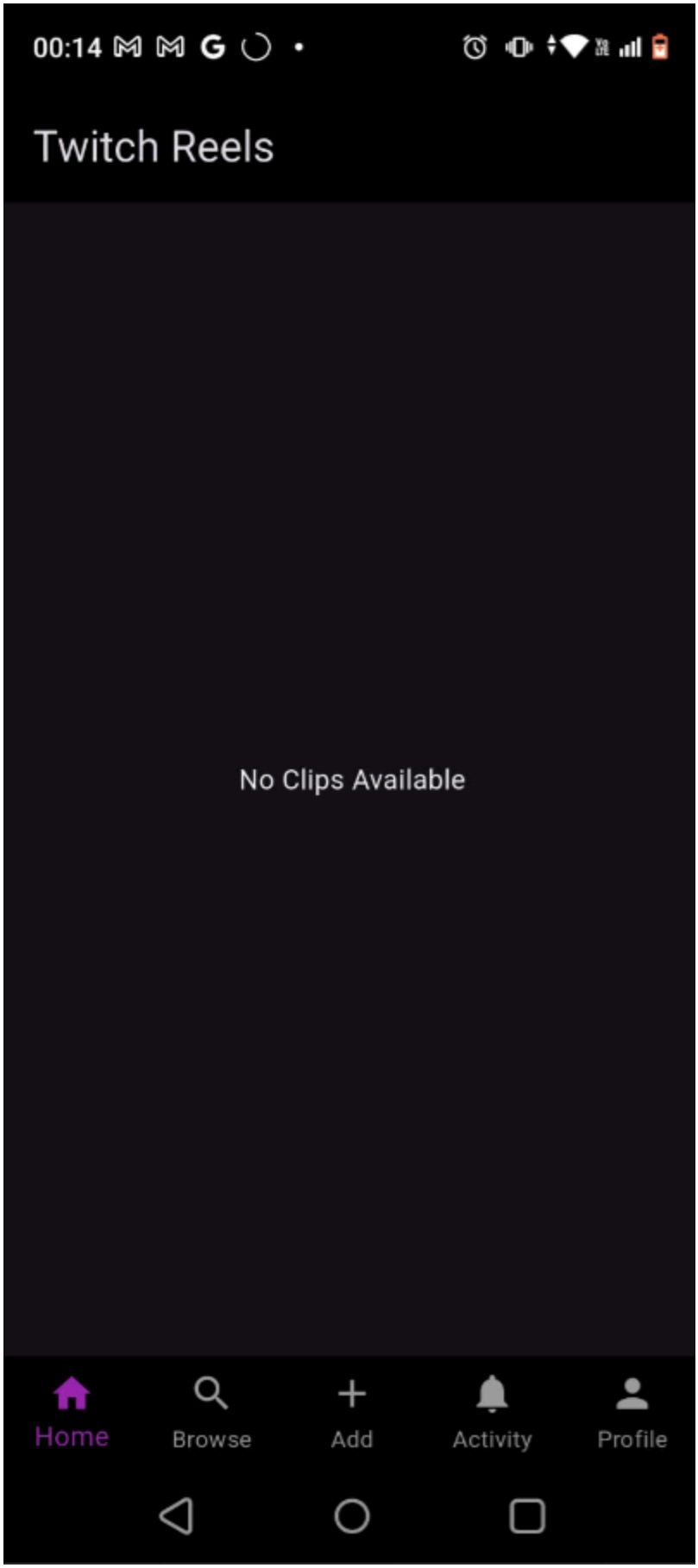
```
              decoration: const BoxDecoration(
                color: Colors.red,
                shape: BoxShape.circle,
              ),
            ),
            const SizedBox(width: 4),
            Flexible(
              child: Text(
                '$viewers Viewers',
                style: TextStyle(
                  color: Colors.grey[400],
                  fontSize: 12,
                ),
                maxLines: 1,
                overflow: TextOverflow.ellipsis,
              ),
            ),
          ],
        ),
      ),

      const SizedBox(height: 4),
```

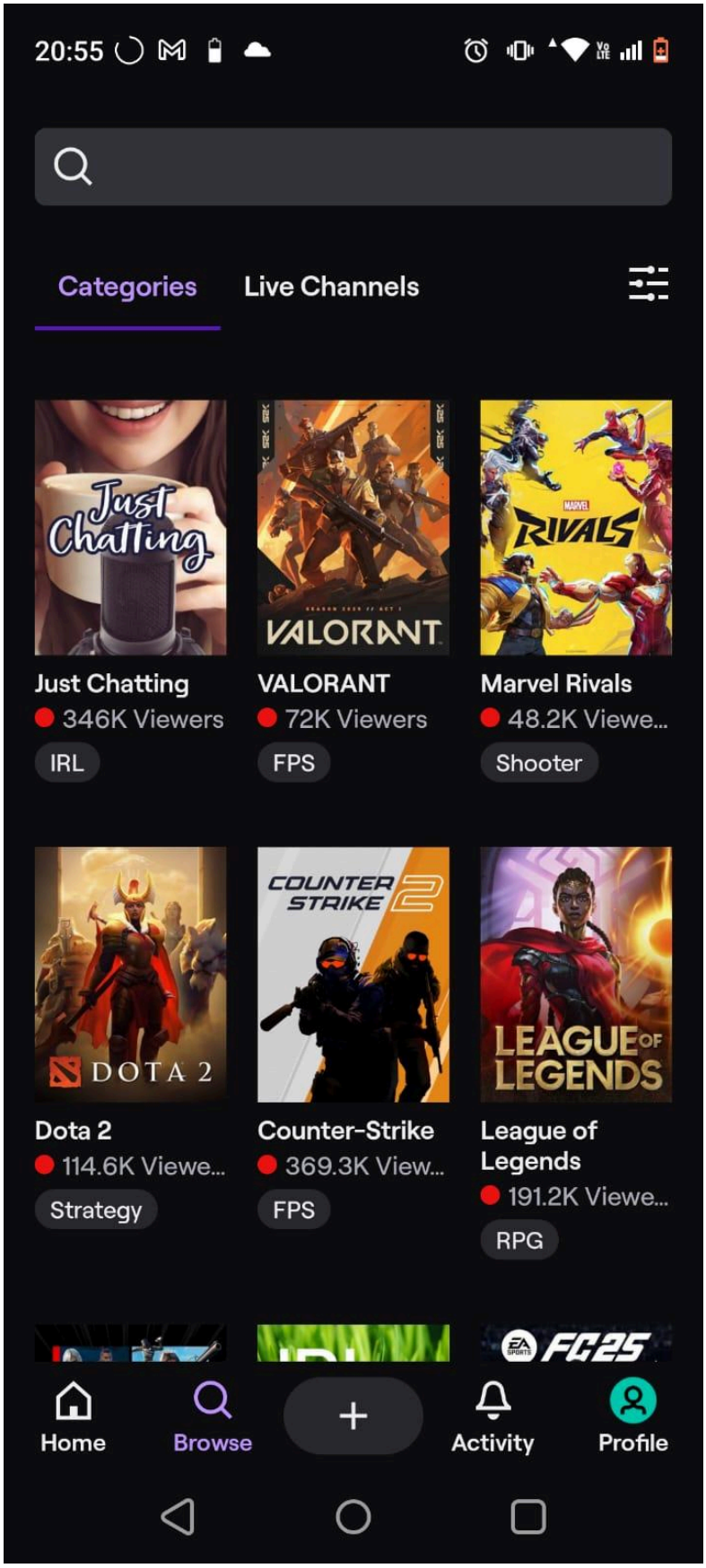
```
          // Category tag
          Container(
            padding: const EdgeInsets.symmetric(horizontal: 8, vertical:
4),
            decoration: BoxDecoration(
              color: Colors.grey[800],
              borderRadius: BorderRadius.circular(12),
            ),
            child: Text(
              categoryTag,
              style: const TextStyle(
                color: Colors.white,
                fontSize: 12,
              ),
            ),
          ),
        ],
      ),
    );
  }

}
```

//home_page.dart

```
@override
 Widget build(BuildContext context) {
   return Scaffold(
     body: _pages[_selectedIndex],
     bottomNavigationBar: BottomNavigationBar(
       type: BottomNavigationBarType.fixed,
       backgroundColor: Colors.black,
       selectedItemColor: Colors.purple,
       unselectedItemColor: Colors.grey,
       currentIndex: _selectedIndex,
       onTap: _onItemTapped,
       items: [
         BottomNavigationBarItem(icon: Icon(Icons.home), label: "Home"),
         BottomNavigationBarItem(icon: Icon(Icons.search), label:
"Browse"),
         BottomNavigationBarItem(icon: Icon(Icons.add), label: "Add"),
         BottomNavigationBarItem(icon: Icon(Icons.notifications), label:
"Activity"),
         BottomNavigationBarItem(icon: Icon(Icons.person), label:
"Profile"),
       ],
```

```
        ),

      );

    }

}
```


00:14
Twitch Reels
No Clips Available
Home
Browse
Add
Activity
Profile


20:55
Categories
Live Channels
Just Chatting
346K Viewers
IRL
VALORANT
72K Viewers
FPS
Marvel Rivals
48.2K Viewe...
Shooter
Dota 2
114.6K Viewe...
Strategy
Counter-Strike
369.3K View...
FPS
League of Legends
191.2K Viewe...
RPG
Home
Browse
Activity
Profile


23:55
Video Player
Gaules
Follow
Dia 4 ESL Pro League Season 21 Stage ...
ZeneTTi_: PAREI FAMILIAAAAAAAAA
bigzera99: era pra quebrar os caras
fortown: Não quer ganhar quita da lobby, é melhor.
LucasBarbosssa: Eu mesmo não
mrmundodosnerds: mas na moral pq foi abrindo um de cada vez
ajato1: @llauqstv segunda acaba a mamata, voltam minhas aulas, aí é loucura demais.
Slow Mode
Send a message

Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	32
Name	Prajjwal Pandey
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

MPL Experiment 04

Prajjwal Pandey

D15A / 32

Aim: To Create an Interactive Form Using the Form Widget

Description of Form Pages:

1. Login Page:

- **Purpose:** Allows users to enter credentials and log in.
- **Widgets Used:**
  - `Form`: Wraps the input fields to enable validation.
  - `TextFormField`: For entering username and password.
  - `ElevatedButton`: For submitting the login form.
  - `InkWell`: For navigating to the sign-up page.
  - `Snackbar`: Displays validation messages.

2. Signup Page:

- **Purpose:** Allows new users to register by entering personal details.
- **Widgets Used:**
  - `Form`: Manages input validation.
  - `TextFormField`: Fields for username, email, password, and phone number.
  - `DropdownButtonFormField`: For selecting gender.
  - `CheckboxListTile`: For accepting terms and conditions.
  - `ElevatedButton`: Submits the form after validation.
  - `Navigator.push`: Redirects users after successful sign-up.

3. Profile Update Page:

- **Purpose:** Allows users to edit their profile details.
- **Widgets Used:**
  - `Form`: Ensures structured data entry.
  - `TextFormField`: For updating username, email, and phone number.
  - `ImagePicker`: Enables users to upload a profile picture.
  - `DropdownButtonFormField`: Lets users update their preferences.
  - `SwitchListTile`: Toggles notification preferences.

- `ElevatedButton`: Saves updates.

Key Features & Functionalities Implemented:

- **Validation:**
  - Used `validator` property in `TextFormField` to validate empty fields and input formats.
  - Displayed error messages using `SnackBar` or inline text.
- **State Management:**
  - Used `GlobalKey<FormState>` to validate and submit forms.
- **Navigation Between Forms:**
  - Utilized `Navigator.push` and `Navigator.pop` for seamless transitions.
- **User Experience Enhancements:**
  - Added proper padding, margins, and icons for better UI.
  - Included loaders and visual feedback on button presses.

CODE :

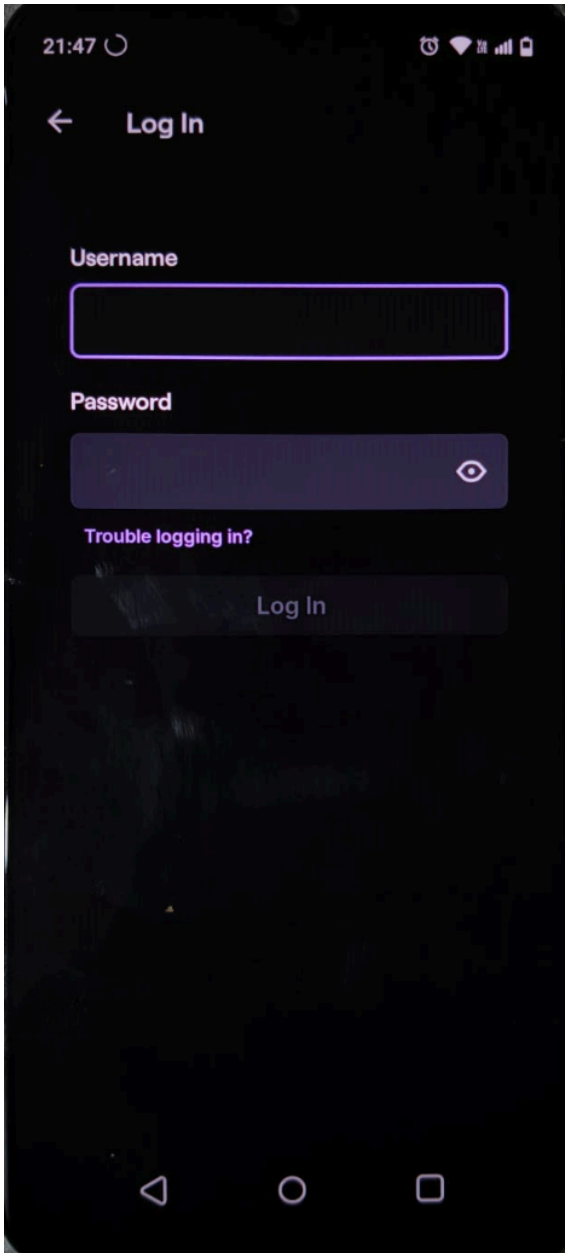
```
class LoginPage extends StatefulWidget {
  final VoidCallback onLoginSuccess;

  LoginPage({required this.onLoginSuccess});

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

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

  Future<void> _login() async {
    try {
      await _auth.signInWithEmailAndPassword(
        email: _emailController.text.trim(),
        password: _passwordController.text.trim(),
      );
      widget.onLoginSuccess();
```

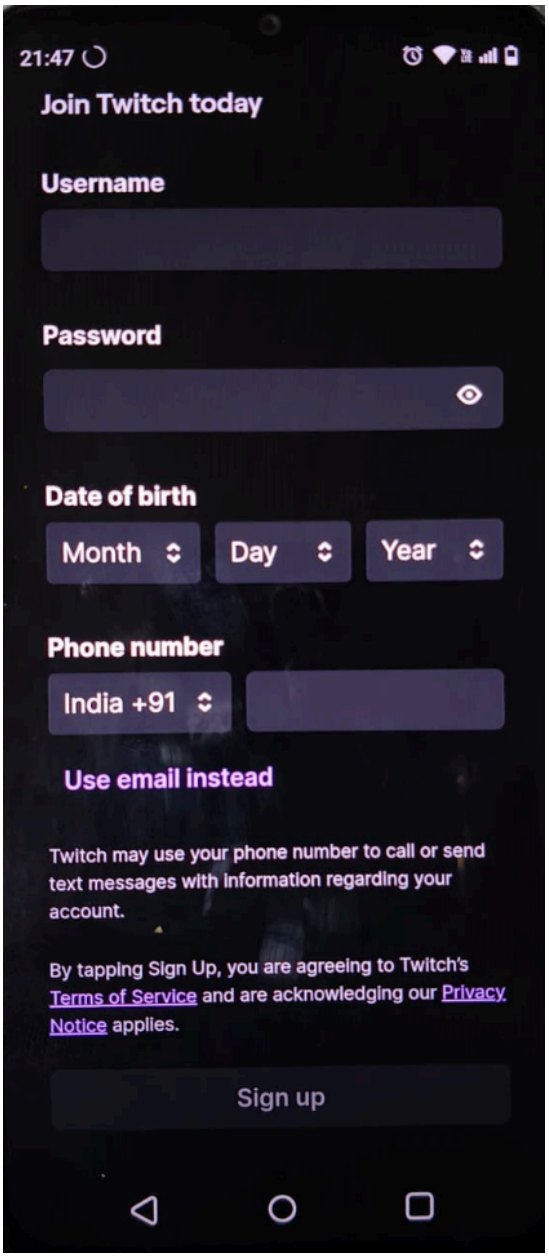
```
    } catch (e) {
      print("Login failed: $e");
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("Login failed: ${e.toString()}")),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Log In")),
      body: Padding(
        padding: EdgeInsets.all(20),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            TextField(
              controller: _emailController,
              decoration: InputDecoration(labelText: "Email"),
            ),
            TextField(
              controller: _passwordController,
              decoration: InputDecoration(labelText: "Password"),
              obscureText: true,
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: _login,
              child: Text("Log In"),
            ),
          ],
        ),
      ),
    );
  }
}

class SignupPage extends StatefulWidget {
  final VoidCallback onSignupSuccess;
```

```
  SignupPage({required this.onSignupSuccess});

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

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

  Future<void> _signup() async {
    try {
      await _auth.createUserWithEmailAndPassword(
        email: _emailController.text.trim(),
        password: _passwordController.text.trim(),
      );
      widget.onSignupSuccess();
    } catch (e) {
      print("Signup failed: $e");
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("Signup failed: ${e.toString()}")),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Sign Up")),
      body: Padding(
        padding: EdgeInsets.all(20),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            TextField(
              controller: _emailController,
              decoration: InputDecoration(labelText: "Email"),
```

```
            ),
            TextField(
              controller: _passwordController,
              decoration: InputDecoration(labelText: "Password"),
              obscureText: true,
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: _signup,
              child: Text("Sign Up"),
            ),
          ],
        ),
      ),
    );
  }
}
```

OUTPUT:


21:47
Log In
Username
Password
Trouble logging in?
Log In


21:47
Join Twitch today
Username
Password
Date of birth
Month
Day
Year
Phone number
India +91
Use email instead
Twitch may use your phone number to call or send text messages with information regarding your account.
By tapping Sign Up, you are agreeing to Twitch's Terms of Service and are acknowledging our Privacy Notice applies.
Sign up

Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	32
Name	Prajjwal Pandey
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

MPL Experiment 05

Prajjwal Pandey

D15A / 32

Aim: Applying Navigation, Routing, and Gestures in Flutter App

Introduction: Navigation and routing in Flutter allow users to move between different screens, while gestures provide interactive functionalities like tapping, swiping, and dragging. This experiment demonstrates how these features are used in a Flutter app.

Implementation Details:

1. Navigation and Routing

Pages Implemented:

- **Landing Page** - Initial screen with options to navigate to login and signup pages.
- **Login Page** - Form for user authentication.
- **Signup Page** - Form for new user registration.
- **Home Page** - Main screen after authentication, displaying content.
- **Profile Page** - User profile with logout functionality.

Navigation Methods Used:

Named Routes:

```
 MaterialApp(
  initialRoute: '/',
  routes: {
    '/': (context) => LandingPage(),
    '/login': (context) => LoginPage(),
    '/signup': (context) => SignupPage(),
    '/home': (context) => HomePage(),
    '/profile': (context) => ProfilePage(),
  },
);
```

-

Push and Pop Navigation:

```
 Navigator.push(
```

```
  context,
  MaterialPageRoute(builder: (context) => HomePage()),
);
 Navigator.pop(context);
```

-

---

2. Gesture Detection

Gestures Implemented:

Tap Gesture: Used for button clicks.

```
 GestureDetector(
  onTap: () {
    print("Tapped!");
  },
  child: Container(
    color: Colors.blue,
    child: Text("Tap Me"),
  ),
);
```

-

Swipe Gesture: Implemented in a carousel or dismissible widgets.

```
 Dismissible(
  key: Key(item.toString()),
  onDismissed: (direction) {
    setState(() {
      items.remove(item);
    });
  },
  child: ListTile(title: Text(item)),
);
```

CODE:

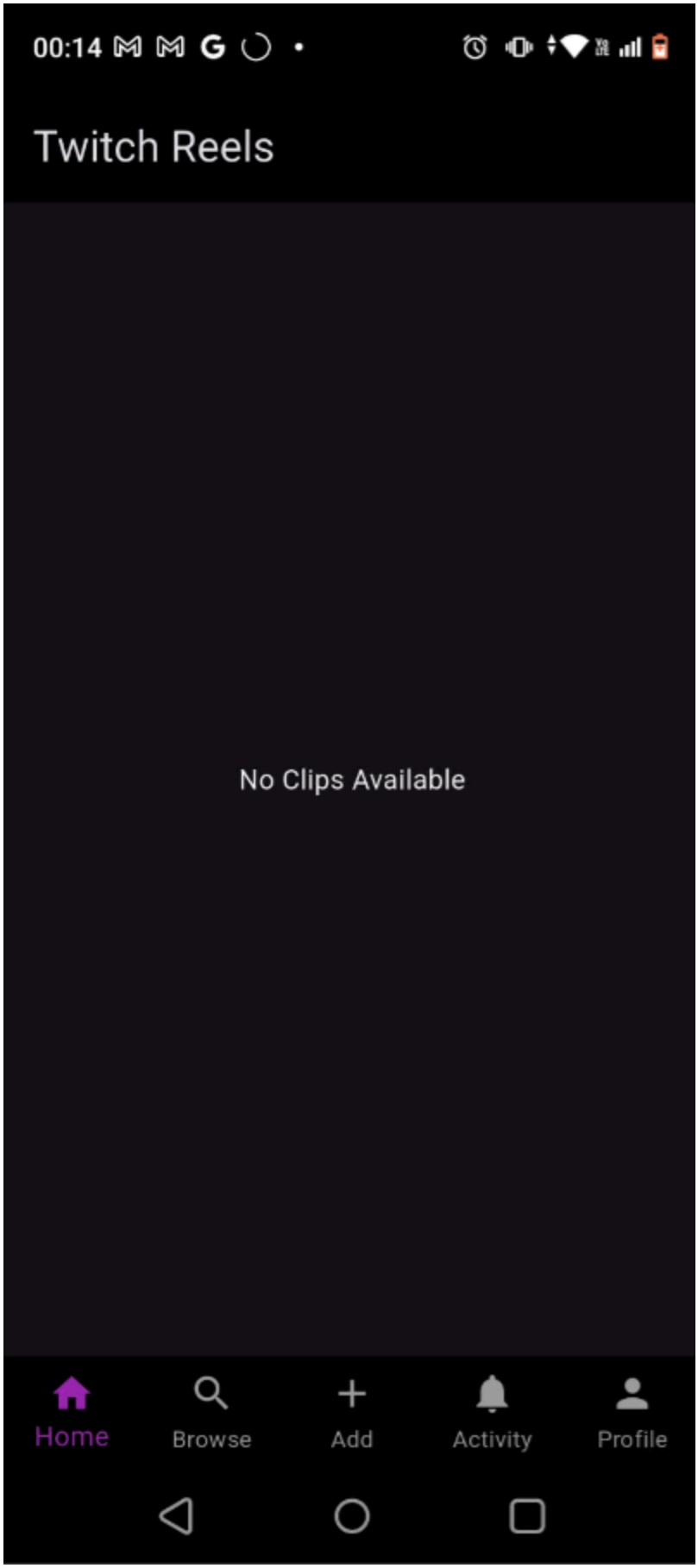
```dart
import 'package:flutter/material.dart';

import 'package:twitch_mpl_lab/pages/categories_page.dart';
```

```dart
import 'package:twitch_mpl_lab/pages/profile_page.dart';

import 'package:twitch_mpl_lab/pages/reels_page.dart';



class HomePage extends StatefulWidget {

  @override

  _HomePageState createState() => _HomePageState();

}



class _HomePageState extends State<HomePage> {

  int _selectedIndex = 0;



  final List<Widget> _pages = [

    ReelsPage(),

    TwitchCategoriesPage(),

    Center(child: Text("Add Content", style: TextStyle(color:
Colors.white))),

    Center(child: Text("Activity", style: TextStyle(color:
Colors.white))),

    ProfilePage(),

  ];



  void _onItemTapped(int index) {

    setState(() {
```

```
      _selectedIndex = index;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: _pages[_selectedIndex],
      bottomNavigationBar: BottomNavigationBar(
        type: BottomNavigationBarType.fixed,
        backgroundColor: Colors.black,
        selectedItemColor: Colors.purple,
        unselectedItemColor: Colors.grey,
        currentIndex: _selectedIndex,
        onTap: _onItemTapped,
        items: [
          BottomNavigationBarItem(icon: Icon(Icons.home), label:
"Home"),
          BottomNavigationBarItem(icon: Icon(Icons.search), label:
"Browse"),
          BottomNavigationBarItem(icon: Icon(Icons.add), label:
"Add"),
          BottomNavigationBarItem(icon: Icon(Icons.notifications),
label: "Activity"),
```

```
          BottomNavigationBarItem(icon: Icon(Icons.person), label:
"Profile"),

        ],

      ),

    );

  }

}
```

OUTPUT :


00:14
Twitch Reels
No Clips Available
Home
Browse
Add
Activity
Profile

23:48

Categories | Live Channels

Just Chatting
102.6K Viewe...
IRL

Counter-Strike
36.1K Viewers
FPS

League of Le...
28.2K Viewers
MOBA

Monster Hun...
13.5K Viewers
Gaming

Grand Theft ...
4.8K Viewers
Gaming

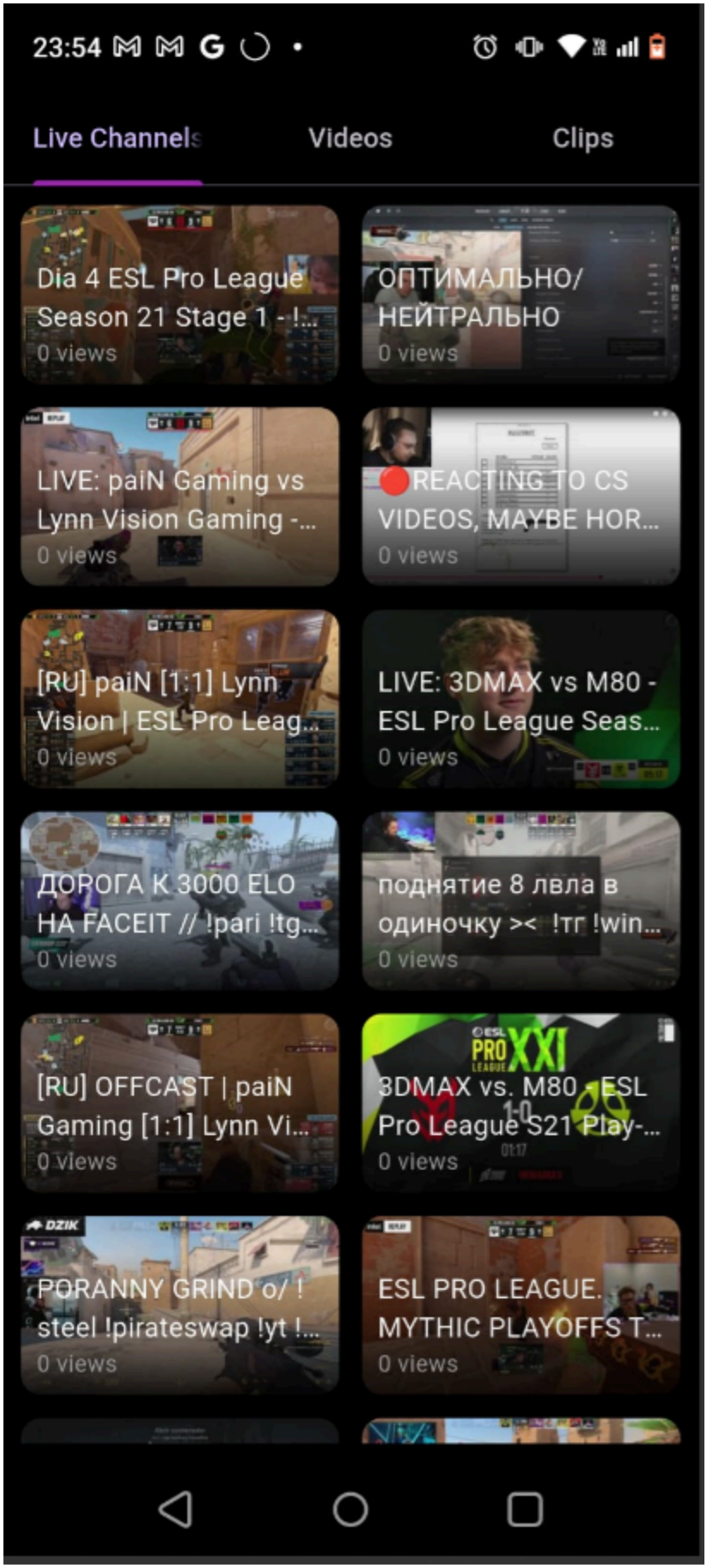
World of War...
24.6K Viewers
Gaming

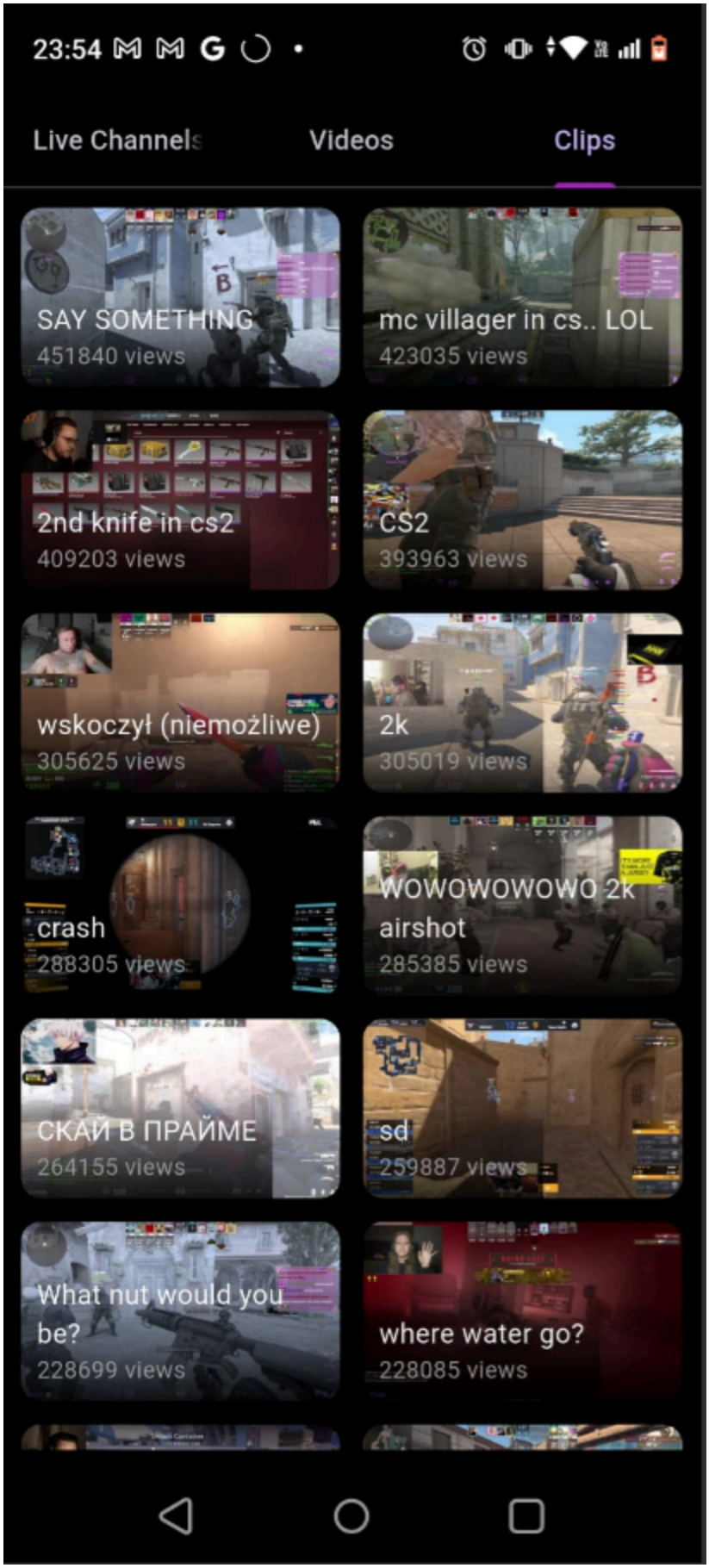
VALORANT
14.8K Viewers
FPS

Fortnite
10.6K Viewers
Shooter

Destiny 2
65.1K Viewers
Gaming

Home | Browse | Add | Activity | Profile


23:54
Live Channels
Videos
Clips
Dia 4 ESL Pro League Season 21 Stage 1 - !...
0 views
ОПТИМАЛЬНО/ НЕЙТРАЛЬНО
0 views
LIVE: paiN Gaming vs Lynn Vision Gaming -...
0 views
🔴REACTING TO CS VIDEOS, MAYBE HOR...
0 views
[RU] paiN [1:1] Lynn Vision | ESL Pro Leag...
0 views
LIVE: 3DMAX vs M80 - ESL Pro League Seas...
0 views
ДОРОГА К 3000 ELO НА FACEIT // !pari !tg...
0 views
поднятие 8 лвла в одиночку >< !тг !win...
0 views
[RU] OFFCAST | paiN Gaming [1:1] Lynn Vi...
0 views
3DMAX vs. M80 - ESL Pro League S21 Play-...
0 views
PORANNY GRIND o/ ! steel !pirateswap !yt !...
0 views
ESL PRO LEAGUE. MYTHIC PLAYOFFS T...
0 views


23:54
Live Channels
Videos
Clips
SAY SOMETHING
451840 views
mc villager in cs.. LOL
423035 views
2nd knife in cs2
409203 views
CS2
393963 views
wskoczył (niemożliwe)
305625 views
2k
305019 views
crash
288305 views
WOWOWOWOWO 2k airshot
285385 views
СКАЙ В ПРАЙМЕ
264155 views
sd
259887 views
What nut would you be?
228699 views
where water go?
228085 views

23:55

Video Player

Gaules Follow

Dia 4 ESL Pro League Season 21 Stage ...

ZeneTTi_: PAREI FAMILIAAAAAAAAA

bigzera99: era pra quebrar os caras

fortown: Não quer ganhar quita da lobby, é melhor.

LucasBarbosssa: Eu mesmo não

mrmundodosnerds: mas na moral pq foi abrindo um de cada vez

ajato1: @llauqstv segunda acaba a mamata, voltam minhas aulas, aí é loucura demais.

Slow Mode

Send a message

23:56

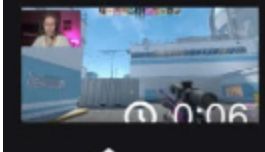
Video Player

biskolata new version!!
Counter-Strike | 51.1K views...

two bear friends
Counter-Strike | 16.7K view...

you can't park there..
Counter-Strike | 13K views |...

2nd kill in 18th round gg
Counter-Strike | 6.6K views ...



she's trained
Counter-Strike | 4.7K views ...

Home Browse Activity Profile

Project Title: **Roll No.**

MAD & PWA Lab
Journal

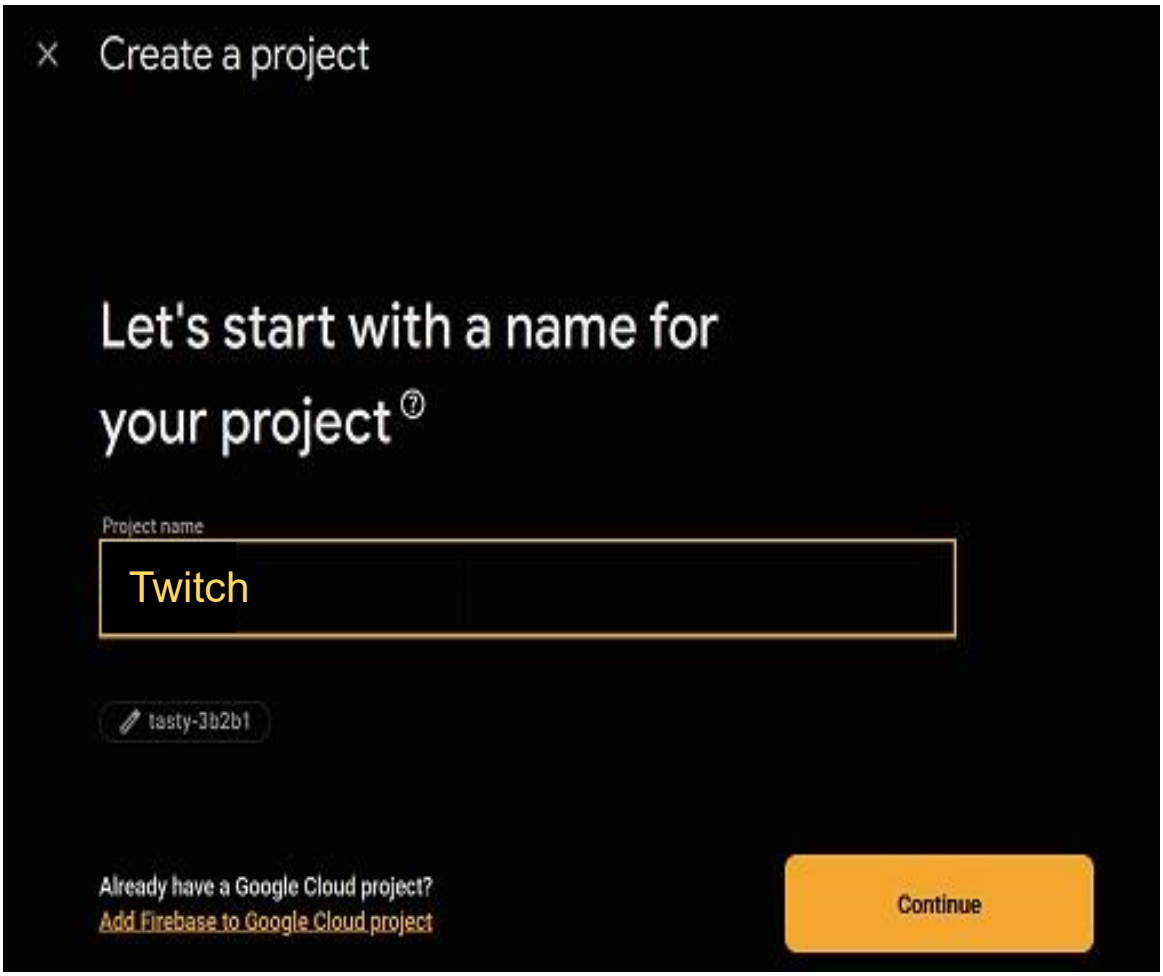
Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	32
Name	Prajjwal Pandey
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

MPL EXPERIMENT-6

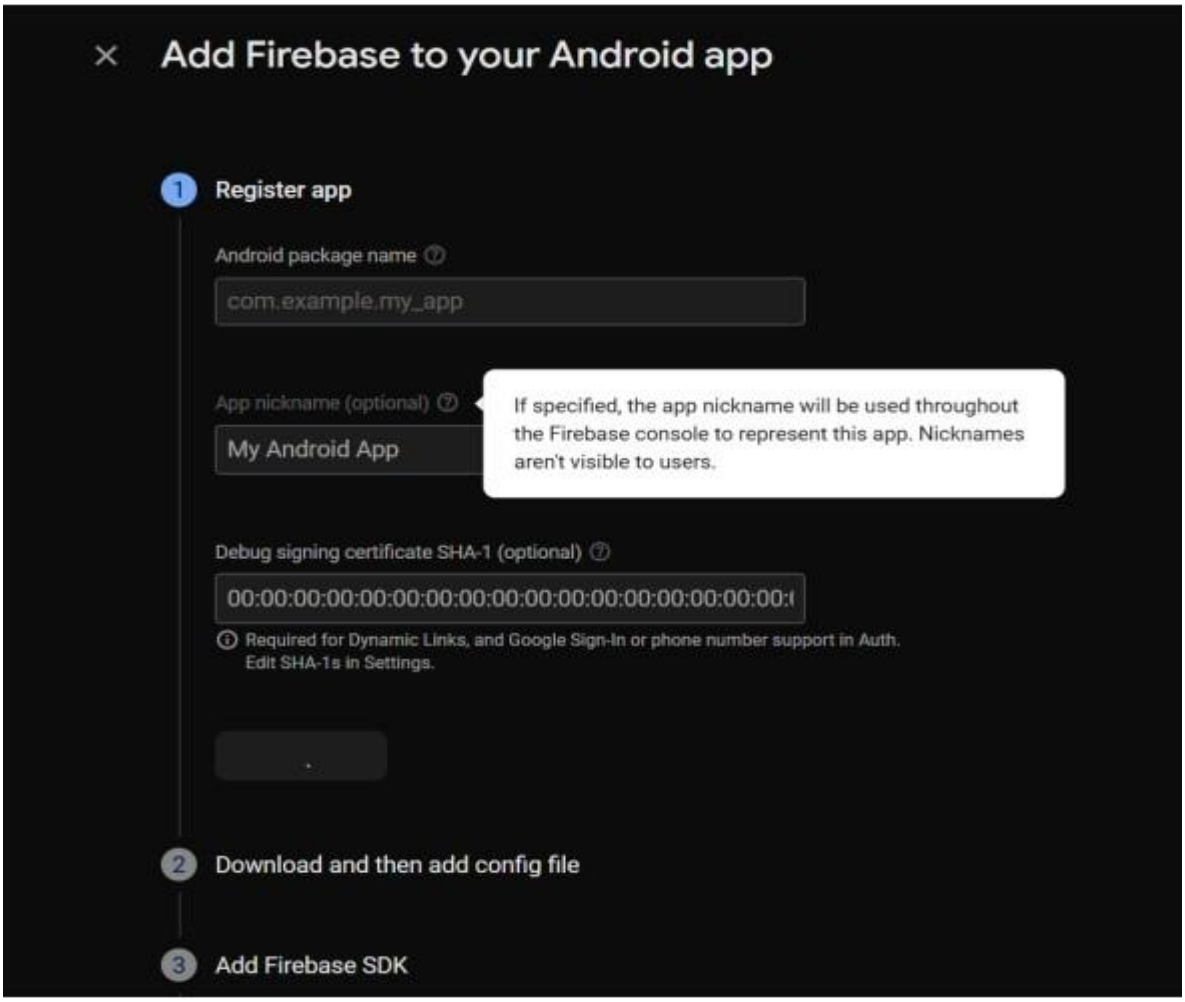
Prajjwal Pandey
D15A / 32

Aim: To Connect Flutter UI with Firebase Database.

Step-1:
First, log in with your Google account to manage your Firebase projects. From within the Firebase dashboard, select the Create new project button and give it a name



In order to add Android support to our Flutter application, select the Android logo from the dashboard. This brings us to the following screen:



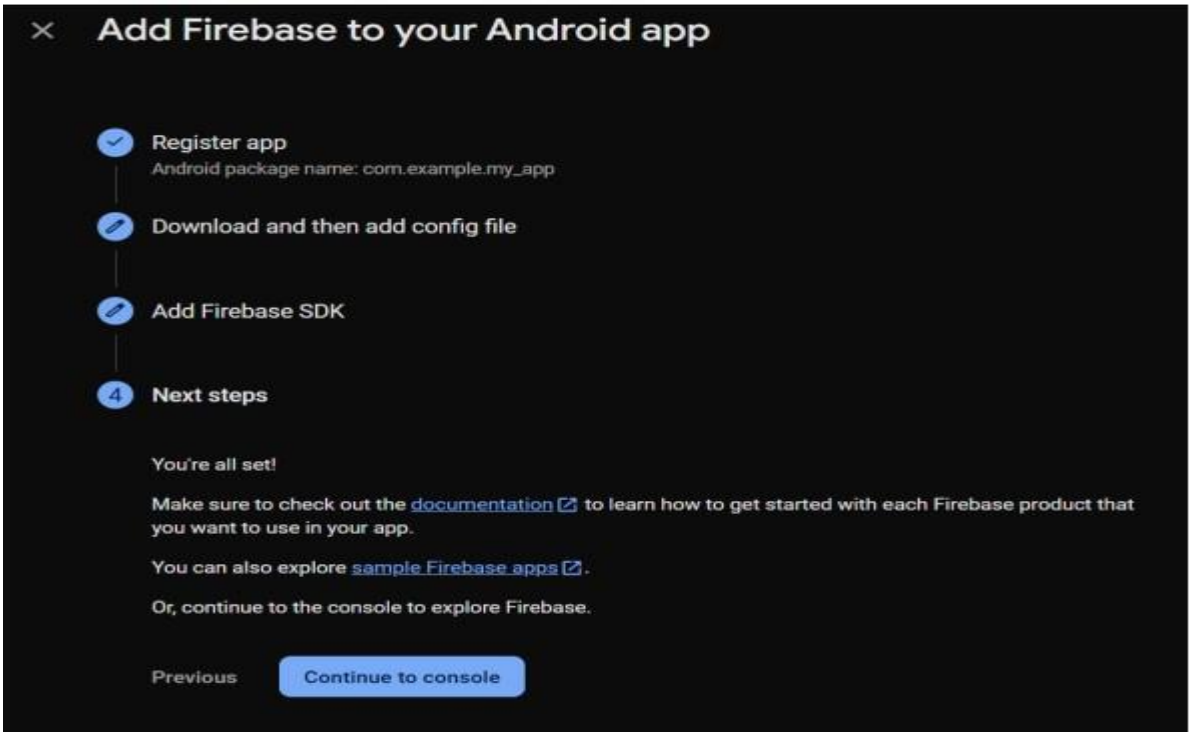
The most important thing here is to match up the Android package name that you choose here with the one inside of our application.

Downloading the Config File

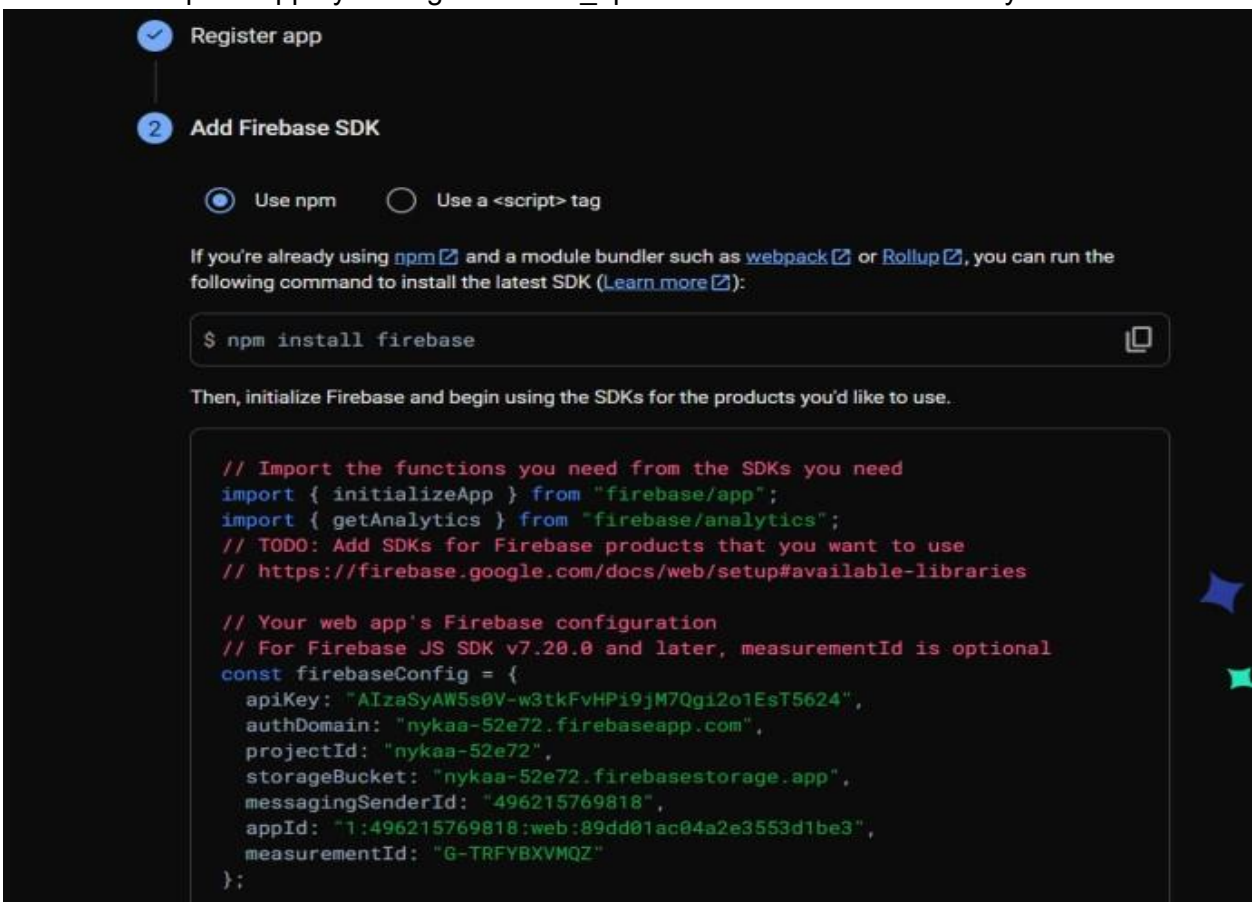
The next step is to add the Firebase configuration file into our Flutter project. This is important as it contains the API keys and other critical information for Firebase to use.

Select Download **google-services.json** from this page
In android/build.gradle adddependencies { classpath
'com.google.gms.google-services:4.4.2'
}

In app/build.gradle adddependencies{ implementation
platform('com.google.firebase:firebase-bom:33.9.0')
}



Now let us setup the app by adding a firebase_option.dart file in the lib directory similar to :



In pubspec.yaml:

```
dependencies: flutter:
sdk: flutter
# The following adds the Cupertino Icons font to your application.
# Use with the CupertinoIcons class for iOS style icons.
cupertino_icons: ^1.0.8
firebase_core: ^3.11.0 firebase_auth:
^5.4.2 cloud_firestore: ^5.6.3
firebase_storage: ^12.4.2
```

Firebase connection Code:
(options is imported from the file: firebase_options.dart)

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();   await
Firebase.initializeApp(     options:
DefaultFirebaseOptions.currentPlatform,
  );
  CloudinaryService.init();
runApp(     MultiProvider(
providers: [
       ChangeNotifierProvider(create: (context) => CartProvider()),
     ],
     child: MyApp(),
   ),
  );
}
```

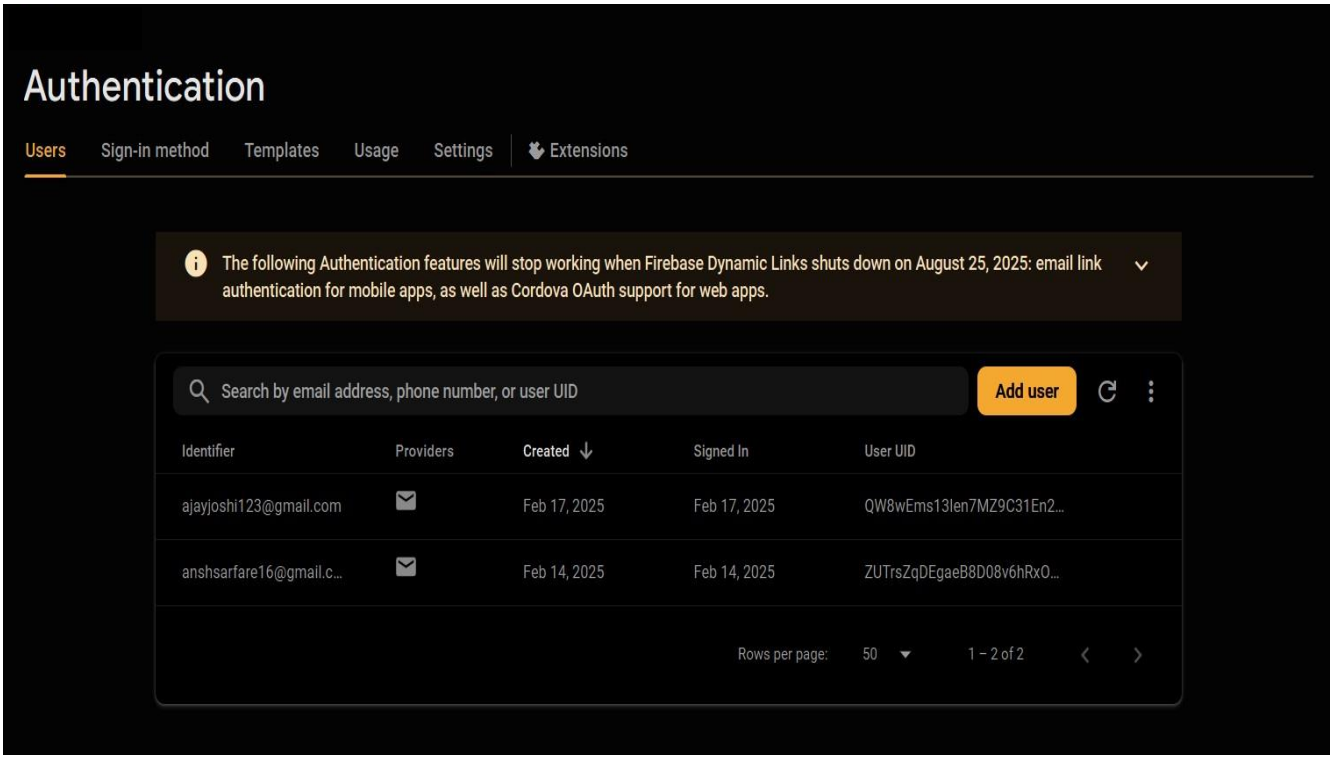
Authentication code:

```
import 'package:firebase_auth/firebase_auth.dart'; import
'package:cloud_firestore/cloud_firestore.dart';

class AuthService {   final FirebaseAuth _auth =
FirebaseAuth.instance;
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;

  Future<User?> registerWithEmailAndPassword(String email, String password, String
username) async {
    try {
      UserCredential credential = await _auth.createUserWithEmailAndPassword(
email: email,         password: password,
```

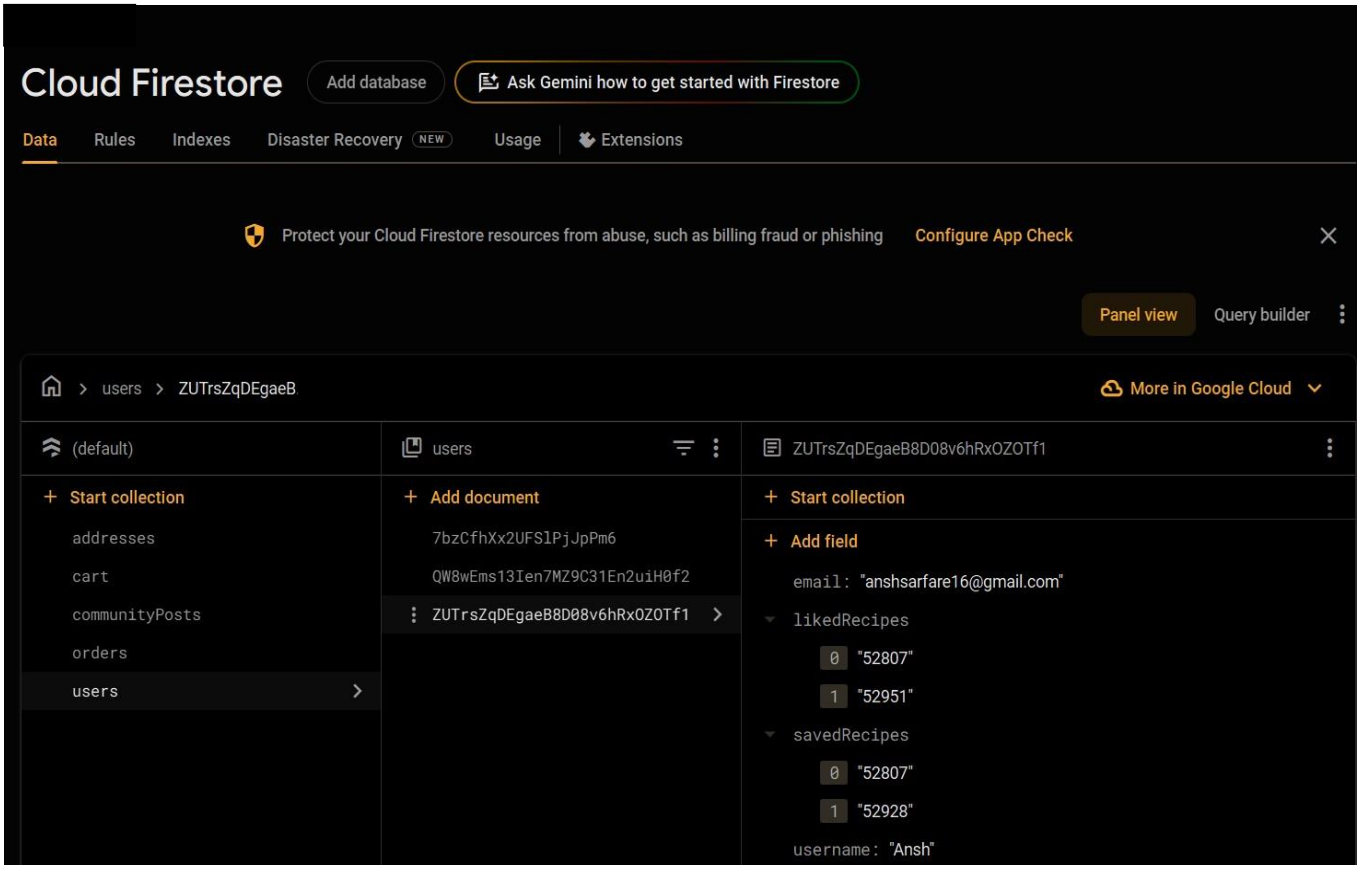
```
      );
      User? user = credential.user;
       if (user != null) {         await
_firestore.collection('users').doc(user.uid).set({
          'email': email,
          'username': username, // Add username field
          'likedRecipes': [],
          'savedRecipes': [],
        });
      }

      return user;     }
catch (e) {
print(e.toString());
return null;
    }
  }

  Future<User?> signInWithEmailAndPassword(String email, String password) async {
try {
      UserCredential credential = await _auth.signInWithEmailAndPassword(
email: email,         password: password,
      );
      User? user = credential.user;
       if (user != null) {         final userDoc = await
_firestore.collection('users').doc(user.uid).get();         if (!userDoc.exists) {
await _firestore.collection('users').doc(user.uid).set({
            'email': email,
            'likedRecipes': [],
            'savedRecipes': [],
          });
        }
      }
      return user;     }
catch (e) {
print(e.toString());
return null;
    }
  }

  Stream<User?> authStateChanges() {     return
FirebaseAuth.instance.authStateChanges();
  }
```

```
  // Sign out
  Future<void> signOut() async {
await _auth.signOut();
  }

  // Get the currently logged-in user
User? getCurrentUser() {     return
_auth.currentUser;
  }
}
```

Authentication:



Firestore Database:


Cloud Firestore
Add database
Ask Gemini how to get started with Firestore
Data
Rules
Indexes
Disaster Recovery
NEW
Usage
Extensions
Protect your Cloud Firestore resources from abuse, such as billing fraud or phishing
Configure App Check
Panel view
Query builder
users
ZUTrsZqDEgaeB
More in Google Cloud
(default)
users
ZUTrsZqDEgaeB8D08v6hRxOZOTf1
Start collection
addresses
cart
communityPosts
orders
users
Add document
7bzCfhXx2UFSlPjJpPm6
QW8wEms13Ien7MZ9C31En2uiH0f2
ZUTrsZqDEgaeB8D08v6hRxOZOTf1
Start collection
Add field
email: "anshsarfare16@gmail.com"
likedRecipes
0 "52807"
1 "52951"
savedRecipes
0 "52807"
1 "52928"
username: "Ansh"

Project Title: **Roll No.**

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.	32
Name	Prajjwal Pandey
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

Experiment 7: Adding Metadata to Web App Manifest for PWA

Ronak Katariya 23

Prajjwal Pandey 32

Snehal Patil 38

Objective:

To write metadata for an eCommerce Progressive Web App (PWA) in a Web App Manifest file to enable the "Add to Home Screen" 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:
A Progressive Web is different and better than a Regular Web app with features like:
1. Native Experience
Though a PWA runs on web technologies (HTML, CSS, JavaScript) like a Regular web app, it gives user experience like a native mobile application. It can use most native device features, including push notifications, without relying on the browser or any other entity. It offers a seamless and integrated user experience that it is quite tough for one to differentiate between a PWA and a Native application by considering its look and feel.

2. Ease of Access
Unlike other mobile apps, PWAs do not demand longer download time and make memory space available for installing the applications. The PWAs can be shared and installed by a link, which cuts down the number of steps to install and use. These applications can easily keep an app

icon on the user's home screen, making the app easily accessible to the users and helps the brands remain in the users' minds, and improving the chances of interaction.

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

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. In simpler words, PWAs let you maintain the user engagement and retention rate.

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.
In this app type, the web app developers can push the live update from the server, which reaches the apps residing on the user's devices automatically. Therefore, it is easier for the mobile app developer to provide the best of the updated functionalities and services to the end-users without forcing them to update their app.
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.

Pros and cons of the Progressive Web App
The main features are:

Progressive — They work for every user, regardless of the browser chosen because they are built at the base with progressive improvement principles.

Responsive — They adapt to the various screen sizes: desktop, mobile, tablet, or dimensions that can later become available.

App-like — They behave with the user as if they were native apps, in terms of interaction and navigation.

Updated — Information is always up-to-date thanks to the data update process offered by service workers.

Secure — Exposed over HTTPS protocol to prevent the connection from displaying information or altering the contents.

Searchable — They are identified as “applications” and are indexed by search engines.

Reactivable — Make it easy to reactivate the application thanks to capabilities such as web notifications.
Installable — They allow the user to “save” the apps that he considers most useful with the corresponding icon on the screen of his mobile terminal (home screen) without having to face all the steps and problems related to the use of the app store.

Linkable — Easily shared via URL without complex installations.

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.

Weaknesses refer to:
IOS support from version 11.3 onwards;
Greater use of the device battery;
Not all devices support the full range of PWA features (same speech for iOS and Android operating systems);
It is not possible to establish a strong re-engagement for iOS users (URL scheme, standard web notifications);
Support for offline execution is however limited;
Lack of presence on the stores (there is no possibility to acquire traffic from that channel);
There is no “body” of control (like the stores) and an approval process;
Limited access to some hardware components of the devices;
Little flexibility regarding “special” content for users (eg loyalty programs, loyalty, etc.).

Requirements:

- A code editor (e.g., VS Code, Sublime Text)
- A basic eCommerce web app
- A browser supporting PWA features (e.g., Chrome, Edge)

Procedure:

1. **Create the Manifest File:**
    - In the root directory of your eCommerce web app, create a file named `manifest.json`.
    - Add the following metadata:

```
{
  "name": "My eCommerce App",
  "short_name": "ShopApp",
  "description": "A fast and secure eCommerce PWA",
  "start_url": "/index.html",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#2196F3",
  "icons": [
    {
      "src": "/icons/icon-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/icons/icon-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

2. **Link the Manifest File to HTML:**

- In the `<head>` section of `index.html`, add:

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

4.
5. **Serve the App and Test Installation:**
    - Host the application using a local server (e.g., `Live Server` extension in VS Code or Node.js `http-server`).
    - Open the app in a PWA-compatible browser.
    - Check for the "Add to Home Screen" prompt or manually install it from the browser menu.
6. **Verify in DevTools:**
    - Open Chrome DevTools (`F12` or `Ctrl+Shift+I`).
    - Go to the "Application" tab → "Manifest".
    - Ensure all metadata is correctly loaded.

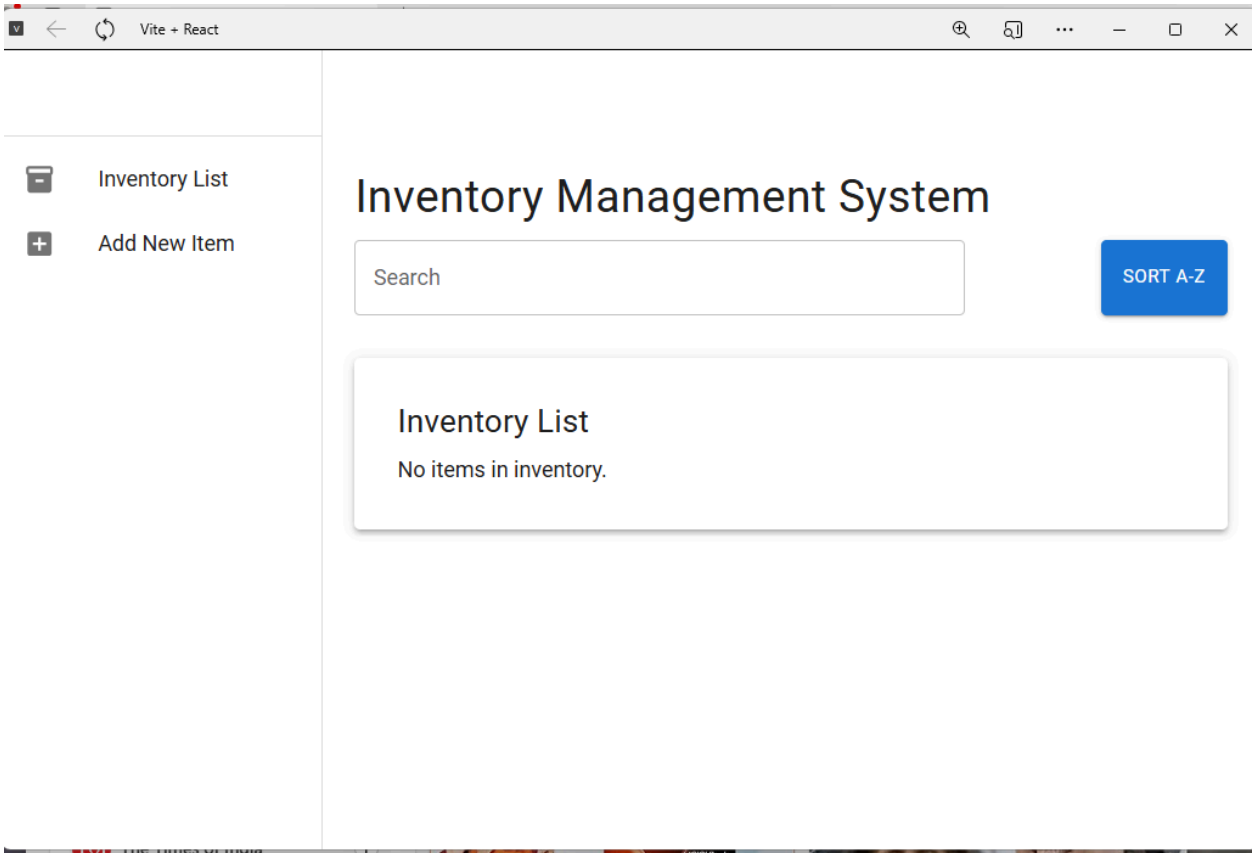
Observations:

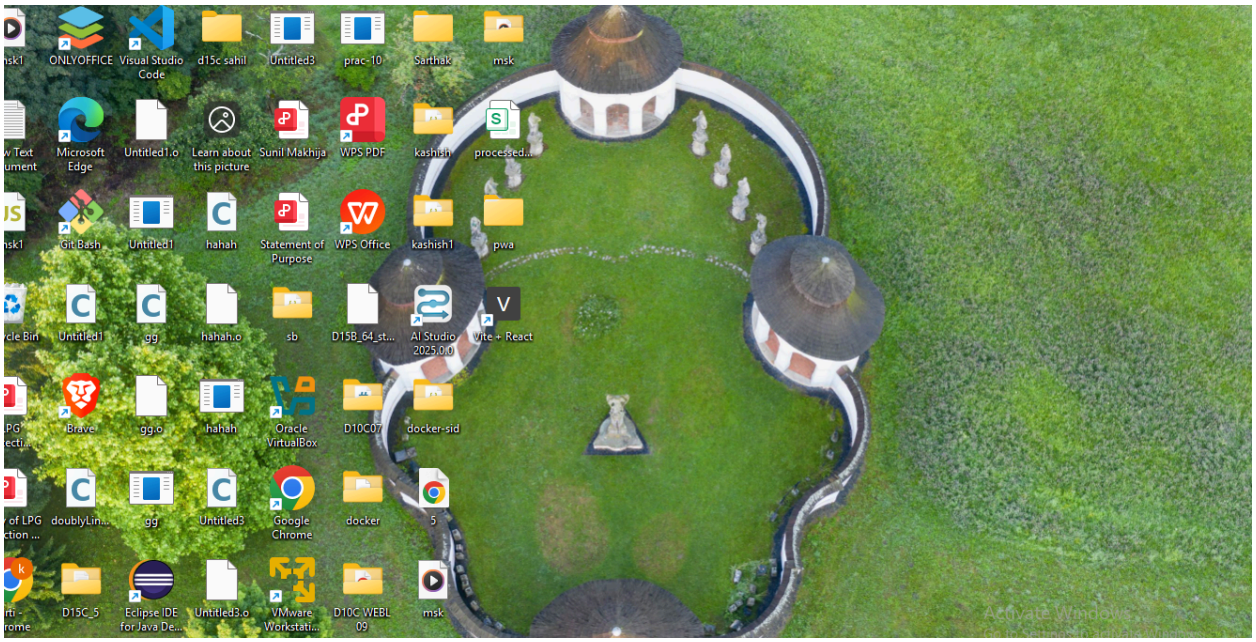
- The manifest file enables the PWA installation.
- The app runs in standalone mode without a browser UI.
- The defined icons and theme colors appear as expected.

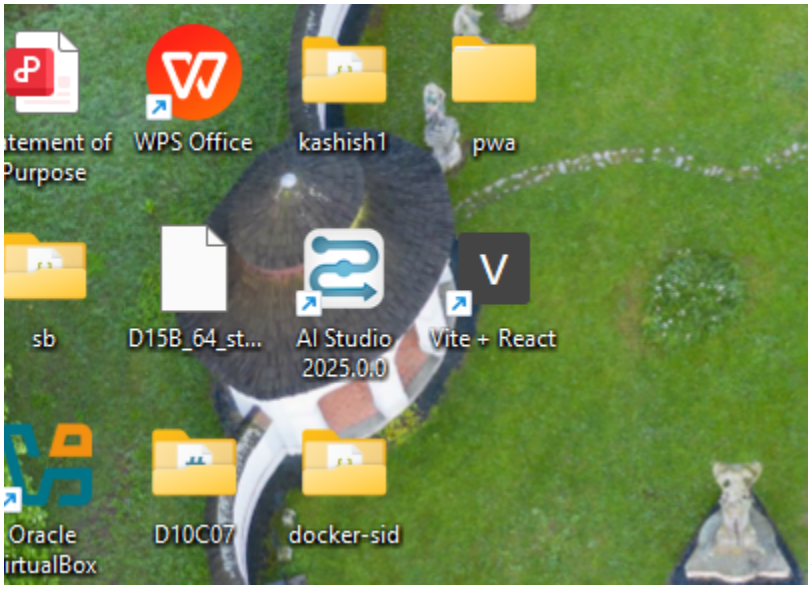
Conclusion:

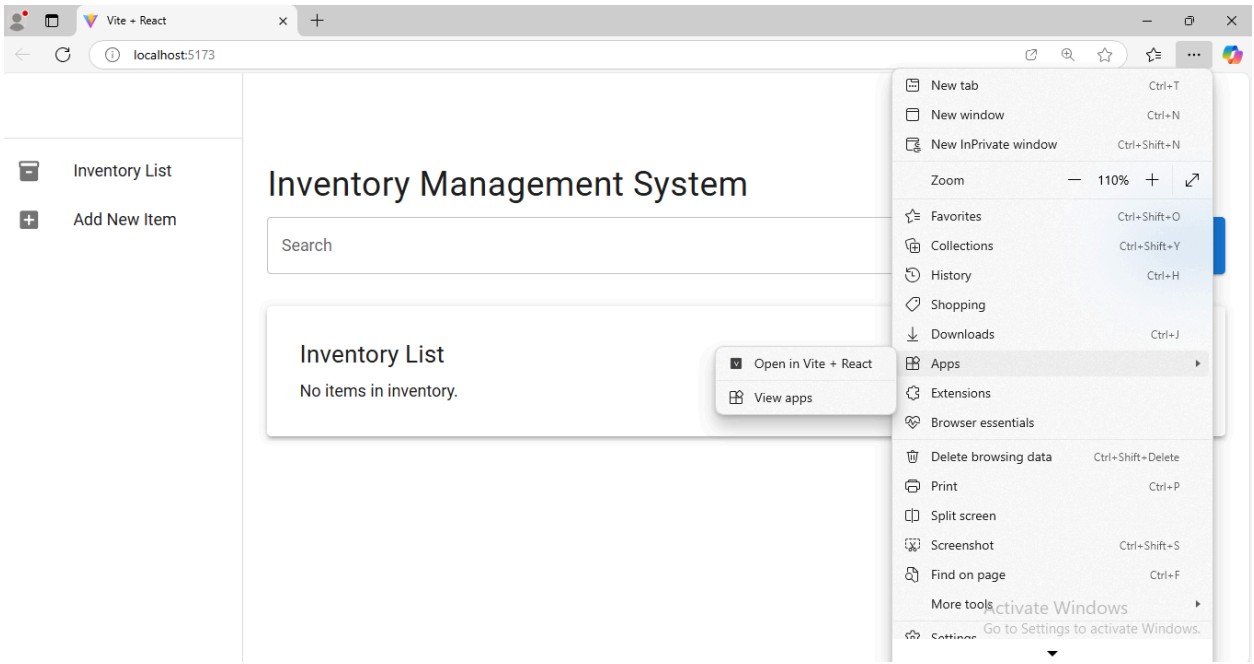
By adding a Web App Manifest file with proper metadata, the eCommerce PWA supports the "Add to Home Screen" feature, improving the user experience with an app-like interface.

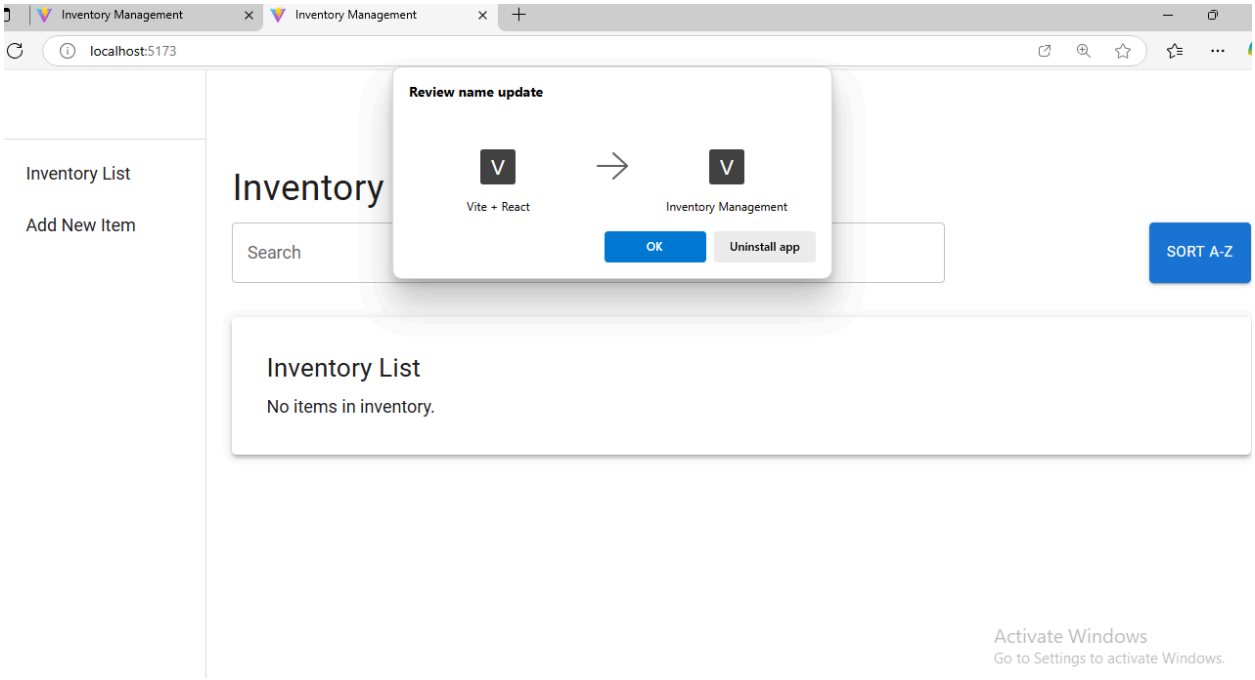
Screenshots:






WPS Office
kashish1
pwa
sb
D15B_64_st...
AI Studio 2025.0.0
Vite + React
Oracle VirtualBox
D10C07
docker-sid
\


Vite + React
localhost:5173
Inventory List
Add New Item
Inventory Management System
Search
Inventory List
No items in inventory.
Open in Vite + React
View apps
New tab Ctrl+T
New window Ctrl+N
New InPrivate window Ctrl+Shift+N
Zoom 110%
Favorites Ctrl+Shift+O
Collections Ctrl+Shift+Y
History Ctrl+H
Shopping
Downloads Ctrl+J
Apps
Extensions
Browser essentials
Delete browsing data Ctrl+Shift+Delete
Print Ctrl+P
Split screen
Screenshot Ctrl+Shift+S
Find on page Ctrl+F
More tools
Activate Windows
Go to Settings to activate Windows.


Inventory Management
Inventory Management
localhost:5173
Review name update
Vite + React
Inventory Management
OK
Uninstall app
Inventory List
Add New Item
Inventory
Search
SORT A-Z
Inventory List
No items in inventory.
Activate Windows
Go to Settings to activate Windows.

Project Title: **Roll No.**

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.	32
Name	Prajjwal Pandey
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Ronak Katariya 23
Prajjwal Pandey 32
Snehal Patil 38

Experiment - 8 : Registering and Activating a Service Worker for E-commerce PWA

Objective:
To code and register a service worker and complete the install and activation process for an eCommerce Progressive Web App (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.

A service worker follows three main lifecycle phases:

1. **Installation** - The service worker is downloaded and installed.
2. **Activation** - The service worker takes control of the pages and manages caching.
3. **Fetching and Events Handling** - The service worker intercepts network requests and serves cached responses when necessary.

Requirements:

- A code editor (e.g., VS Code, Sublime Text)
- A basic eCommerce web app
- A browser that supports service workers (e.g., Chrome, Edge, Firefox)

Procedure:

1. Creating the Service Worker File

- In the root directory of the eCommerce PWA, create a file named `service-worker.js`.
- Add the following code to set up basic caching:

```
const CACHE_NAME = 'ecommerce-pwa-v1';
const urlsToCache = [
  '/',
  '/index.html',
  '/styles.css',
  '/script.js',
  '/icons/icon-192x192.png',
  '/icons/icon-512x512.png'
];

// Install event - Caching files
self.addEventListener('install', (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => {
      return cache.addAll(urlsToCache);
    })
  );
});

// Activate event - Cleanup old caches
self.addEventListener('activate', (event) => {
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.filter((cache) => cache !== CACHE_NAME).map((cache) =>
caches.delete(cache))
```

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

// Fetch event - Serve files from cache
self.addEventListener('fetch', (event) => {
  event.respondWith(
    caches.match(event.request).then((response) => {
      return response || fetch(event.request);
    })
  );
});
```

---

2. Registering the Service Worker

- In the `index.html` or `app.js` file, register the service worker with the following code:

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

---

3. Testing the Service Worker

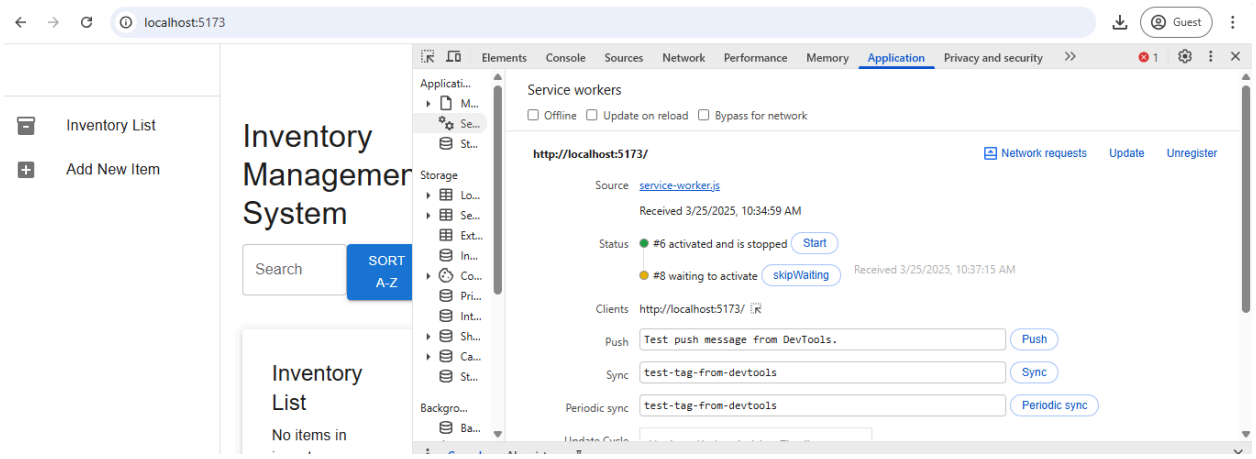
- Host the application on a local server (e.g., using `Live Server` in VS Code or `http-server` in Node.js).
- Open the web application in a browser.
- Open Chrome DevTools (`F12` or `Ctrl+Shift+I`).

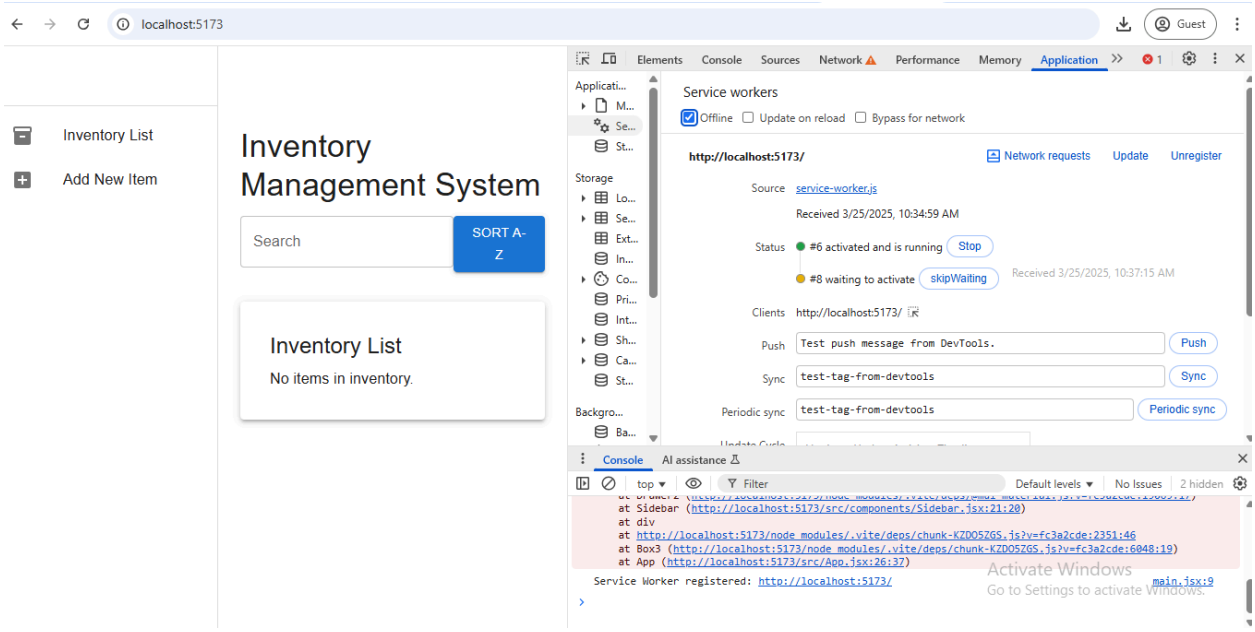
- Go to the "Application" tab → "Service Workers".
- Verify the service worker is installed and activated.

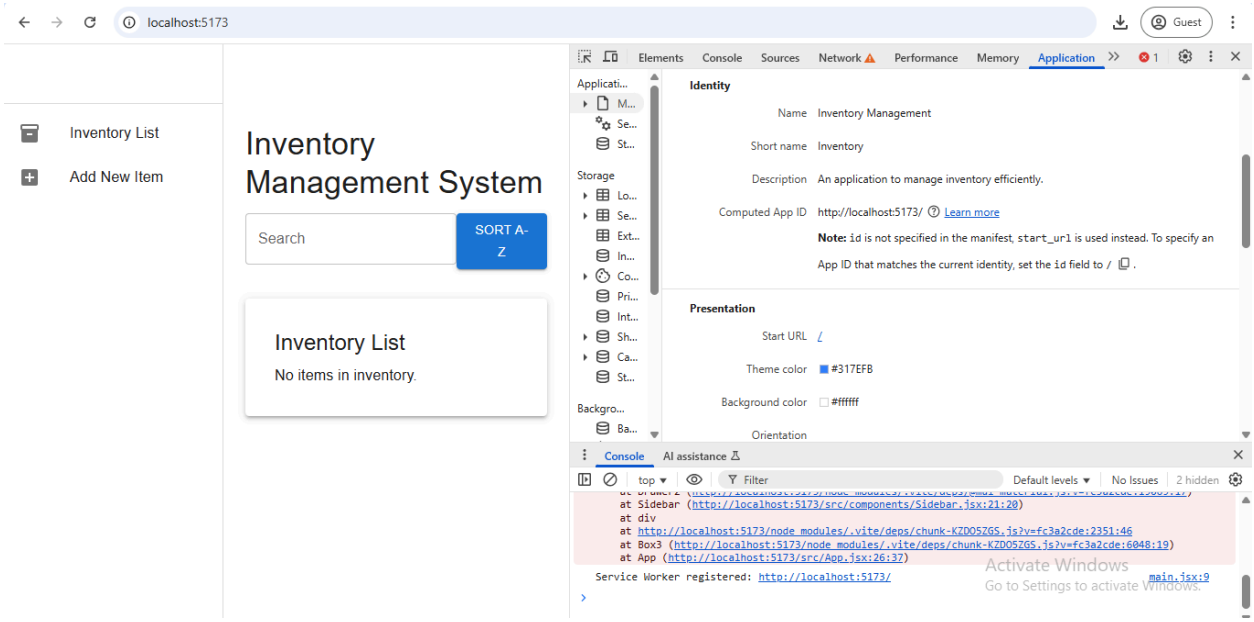
Observations:

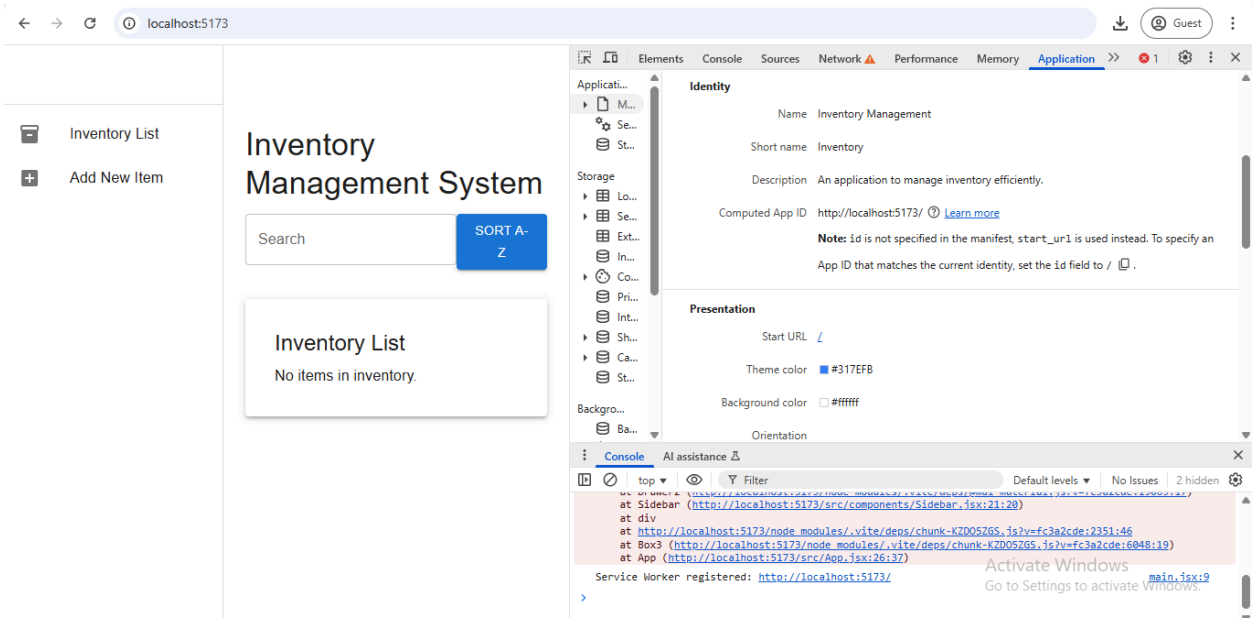
- The service worker successfully caches essential files.
- The app works offline using cached files.
- The activation process ensures that old caches are removed.

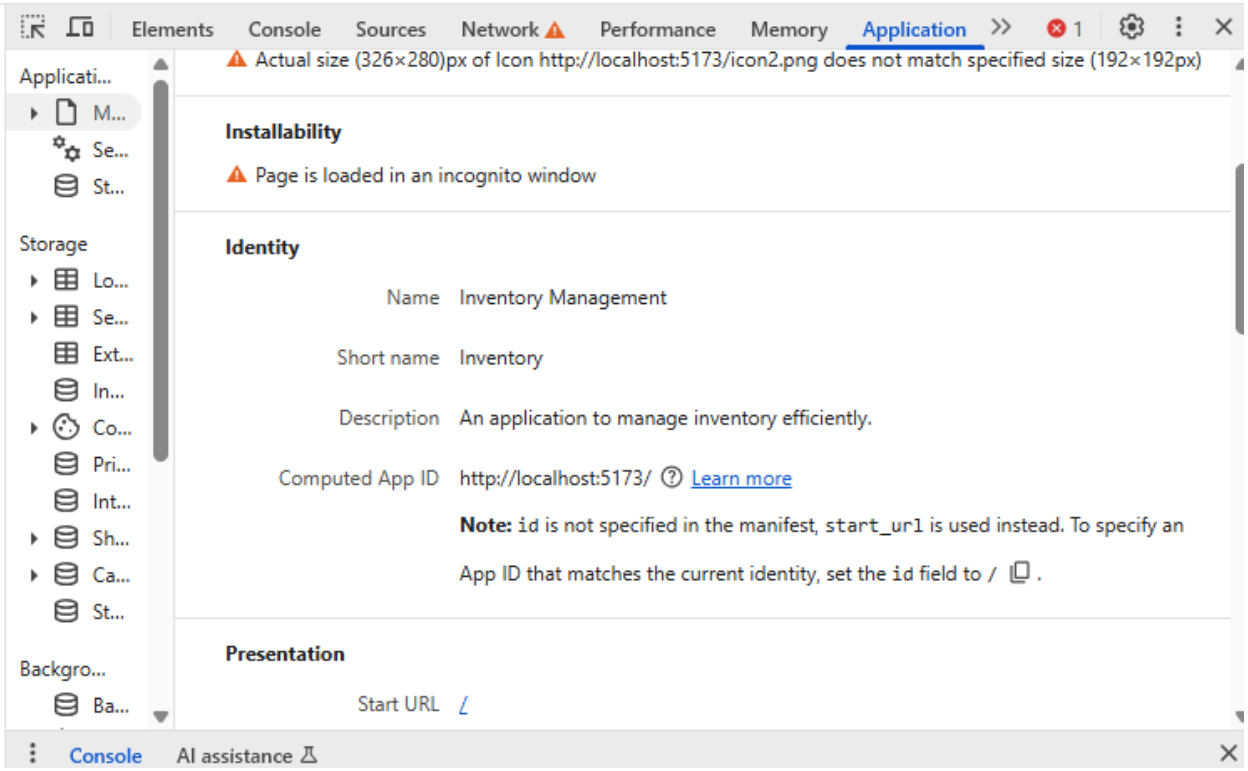
Conclusion:
By coding and registering a service worker, we successfully enabled caching and offline support for the eCommerce PWA, improving performance and reliability.

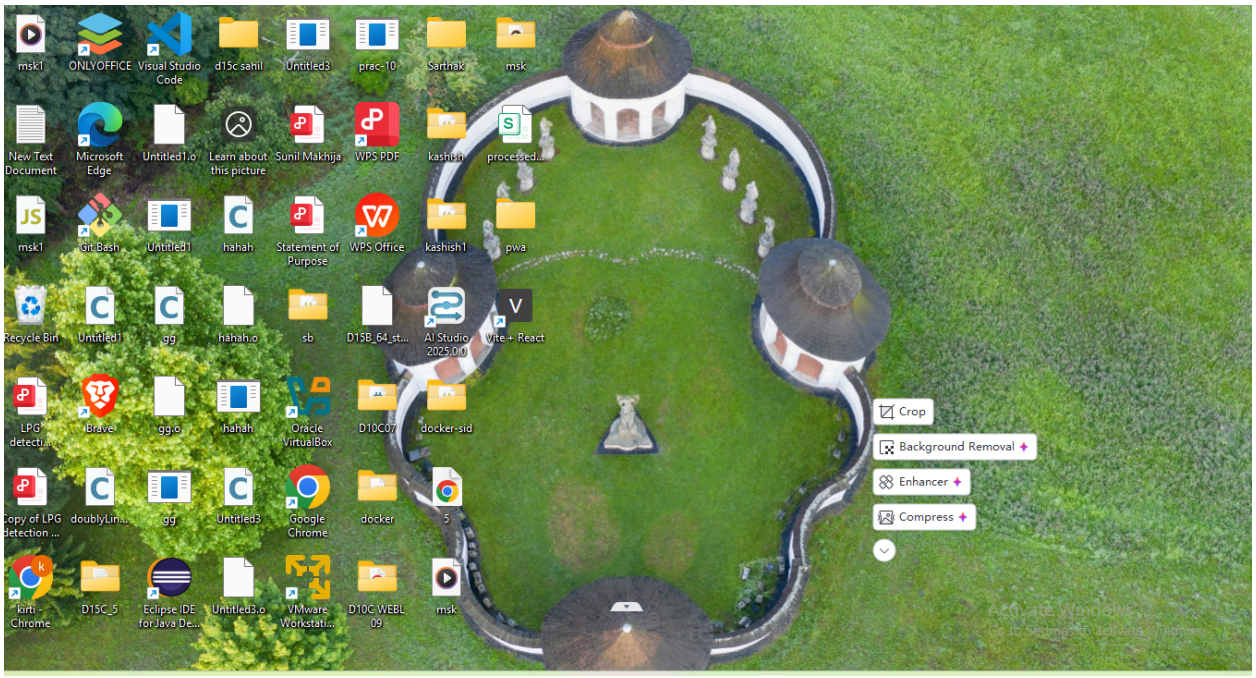



localhost:5173
Guest
Inventory List
Add New Item
Inventory Management System
Search
SORT A-Z
Inventory List
No items in inventory.
Elements
Console
Sources
Network
Performance
Memory
Application
Service workers
Offline
Update on reload
Bypass for network
http://localhost:5173/
Network requests
Update
Unregister
Source
service-worker.js
Received 3/25/2025, 10:34:59 AM
Status
#6 activated and is running
Stop
#8 waiting to activate
skipWaiting
Received 3/25/2025, 10:37:15 AM
Clients
http://localhost:5173/
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Console
AI assistance
Filter
Default levels
No Issues
2 hidden
at Sidebar (http://localhost:5173/src/components/Sidebar.jsx:21:20)
at div
at http://localhost:5173/node_modules/.vite/deps/chunk-KZDO5ZGS.js?v=fc3a2cde:2351:46
at Box3 (http://localhost:5173/node_modules/.vite/deps/chunk-KZDO5ZGS.js?v=fc3a2cde:6048:19)
at App (http://localhost:5173/src/App.jsx:26:37)
Service Worker registered: http://localhost:5173/
main.jsx:9
Activate Windows
Go to Settings to activate Windows.


localhost:5173
Guest
Inventory List
Add New Item
Inventory Management System
Search
SORT A-Z
Inventory List
No items in inventory.
Elements
Console
Sources
Network
Performance
Memory
Application
Identity
Name
Inventory Management
Short name
Inventory
Description
An application to manage inventory efficiently.
Computed App ID
http://localhost:5173/
Learn more
Note: id is not specified in the manifest, start_url is used instead. To specify an App ID that matches the current identity, set the id field to /.
Presentation
Start URL
/
Theme color
#317EFB
Background color
#ffffff
Orientation
Console
AI assistance
Filter
Default levels
No Issues
2 hidden
at Sidebar (http://localhost:5173/src/components/Sidebar.jsx:21:20)
at div
at http://localhost:5173/node_modules/.vite/deps/chunk-KZDO5ZGS.js?v=fc3a2cde:2351:46
at Box3 (http://localhost:5173/node_modules/.vite/deps/chunk-KZDO5ZGS.js?v=fc3a2cde:6048:19)
at App (http://localhost:5173/src/App.jsx:26:37)
Service Worker registered: http://localhost:5173/
main.jsx:9
Activate Windows
Go to Settings to activate Windows.


localhost:5173
Guest
Inventory List
Add New Item
Inventory Management System
Search
SORT A-Z
Inventory List
No items in inventory.
Elements
Console
Sources
Network
Performance
Memory
Application
Identity
Name Inventory Management
Short name Inventory
Description An application to manage inventory efficiently.
Computed App ID http://localhost:5173/ Learn more
Note: id is not specified in the manifest, start_url is used instead. To specify an App ID that matches the current identity, set the id field to /.
Presentation
Start URL /
Theme color #317EFB
Background color #ffffff
Orientation
Console
AI assistance
Service Worker registered: http://localhost:5173/
Activate Windows
Go to Settings to activate Windows.


Elements
Console
Sources
Network
Performance
Memory
Application
Actual size (326×280)px of Icon http://localhost:5173/icon2.png does not match specified size (192×192px)
Installability
Page is loaded in an incognito window
Identity
Name Inventory Management
Short name Inventory
Description An application to manage inventory efficiently.
Computed App ID http://localhost:5173/ Learn more
Note: id is not specified in the manifest, start_url is used instead. To specify an App ID that matches the current identity, set the id field to /.
Presentation
Start URL /
Storage
Backgro...
Console
AI assistance


msk1
ONLYOFFICE
Visual Studio Code
d15c sahil
Untitled3
prac-10
Sarthak
msk
New Text Document
Microsoft Edge
Untitled1.o
Learn about this picture
Sunil Makhija
WPS PDF
kashish
processed...
msk1
Git Bash
Untitled1
hahah
Statement of Purpose
WPS Office
kashish1
pwa
Recycle Bin
Untitled1
gg
hahah.o
sb
D15B_64_st...
AI Studio 2025.0.0
Vite + React
LPG detecti...
Brave
gg.o
hahah
Oracle VirtualBox
D10C07
docker-sid
Copy of LPG detection ...
doublyLin...
gg
Untitled3
Google Chrome
docker
5
kirti - Chrome
D15C_5
Eclipse IDE for Java De...
Untitled3.o
VMware Workstati...
D10C WEBL 09
msk
Crop
Background Removal
Enhancer
Compress
Activate Windows
Go to Settings to activate Windows.

Project Title: **Roll No.**

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.	32
Name	Prajjwal Pandey
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

Ronak Katariya 23
Prajjwal Pandey 32
Snehal Patil 38

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

Theory:
Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

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

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

But if this process fails, we check whether the request has been cached before or not. If a cache exists, it is returned to the page, but if not, this is up to you. You can return dummy content or information messages to the page.

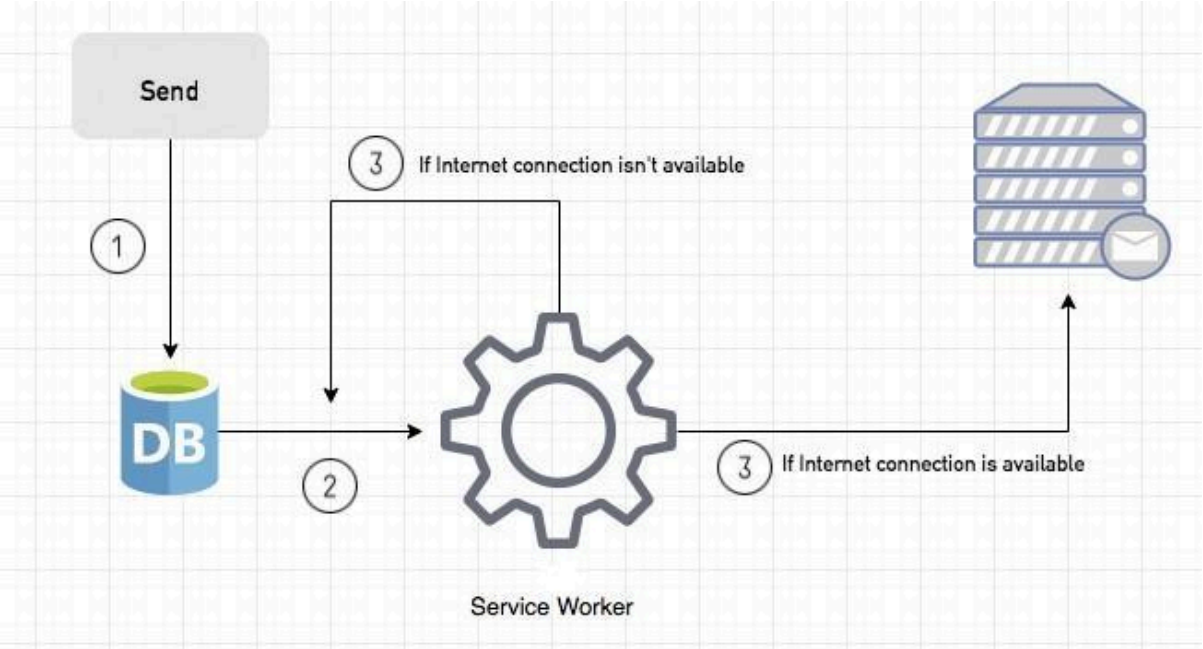
Sync Event

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

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.

Event Listener for Background Sync Registration

```
document.querySelector("button").addEventListener("click", async () => {
    var swRegistration = await navigator.serviceWorker.register("sw.js");
    swRegistration.sync.register("helloSync").then(function () {
        console.log("helloSync success [main.js]");
    });
});
```

Event Listener for sw.js

```
self.addEventListener('sync', event => {
    if (event.tag == 'helloSync') {
        console.log("helloSync [sw.js]");
    }
});
```

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.

```
self.addEventListener('push', event => {
    if (event && event.data) {
        var data = event.data.json();
        if (data.method === "pushMessage") {
            event.waitUntil(self.registration.showNotification("Test App", {
                body: data.message
            }));
        }
    }
});
```

CODE:

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Stopwatch</title>
    <link rel="stylesheet" href="style.css">
    <link rel="manifest" href="/manifest.json">
  </head>
  <body>
    <div class="container">
      <h1>Stopwatch</h1>
      <p class="time">
        <span id="minutes">00</span>:<span id="seconds">00</span>:<span id="tens">00</span>
      </p>
      <ul id="lapList"></ul>
      <button id="start">Start</button>
      <button id="stop">Stop</button>
      <button id="reset">Reset</button>
      <button id="lap">Lap</button>
    </div>
    <script src="stopwatch.js"></script>
    <script>
      if ('serviceWorker' in navigator) {
        window.addEventListener('load', () => {
          navigator.serviceWorker.register('/service-worker.js')
            .then(registration => {
              console.log('ServiceWorker registered with scope: ', registration.scope);
            })
            .catch(error => {
              console.log('ServiceWorker registration failed: ', error);
            });
        });
      }

      // Request Notification Permission and Show Popup
      if ('Notification' in window) {
        Notification.requestPermission().then(permission => {
          if (permission === 'granted') {
            console.log('Notification permission granted');
            showNotification();
```

```
        }
      });
    }

    function showNotification() {
      if (Notification.permission === 'granted') {
        const options = {
          body: 'Welcome Snehal, Ronak, Prajjwal !',
          icon: '/images/icon.png',
          badge: '/images/icon.png'
        };

        // Display the notification
        new Notification('Welcome Snehal, Ronak, Prajjwal ', options);
      }
    }
  </script>
 </body>
</html>
```

Manifest.json

```
{
  "name": "PWA",
  "short_name": "RPS",
  "description": "A simple and elegant stopwatch application.",
  "start_url": "/index.html",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#000000",
  "icons": [
    {
      "src": "/images/icon2.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/images/icon.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
 }
```

serviceworker.js

```
const CACHE_NAME = 'prodigy-stopwatch-cache-v1';
const urlsToCache = [
  '/',
  '/index.html',
  '/style.css',
  '/stopwatch.js',
  '/images/icon2.png',
  '/images/icon.png'
];

// Install Service Worker and Cache Files
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(cache => {
        console.log('Opened cache');
        return cache.addAll(urlsToCache);
      })
  );
});

// Cache and Return Requests
self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request).then(response => {
      return response || fetch(event.request)
        .then(networkResponse => {
          if (!networkResponse || networkResponse.status !== 200) {
            return networkResponse;
          }
          return caches.open(CACHE_NAME).then(cache => {
            cache.put(event.request, networkResponse.clone());
            return networkResponse;
          });
        }).catch(() => {
          if (event.request.mode === 'navigate') {
            return caches.match('/index.html');
          }
        });
    })
  );
});
```

```
// Activate & Remove Old Caches
self.addEventListener('activate', event => {
  const cacheWhitelist = [CACHE_NAME];
  event.waitUntil(
    caches.keys().then(cacheNames => {
      return Promise.all(
        cacheNames.map(cacheName => {
          if (!cacheWhitelist.includes(cacheName)) {
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
});
```

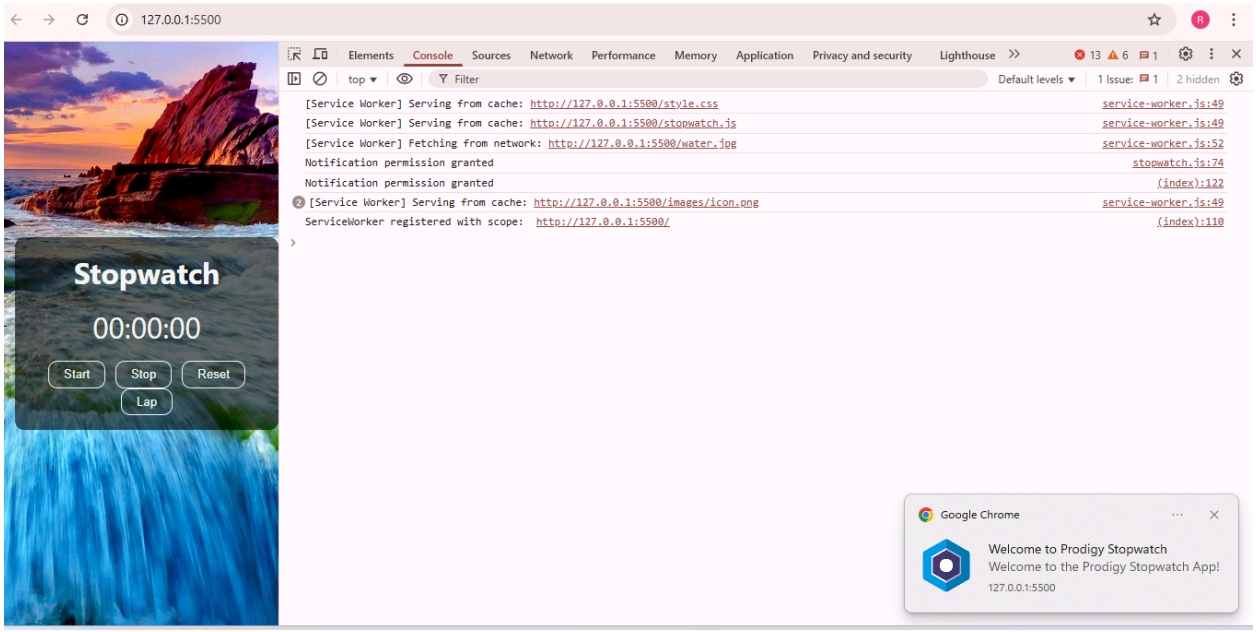
stopwatch.js

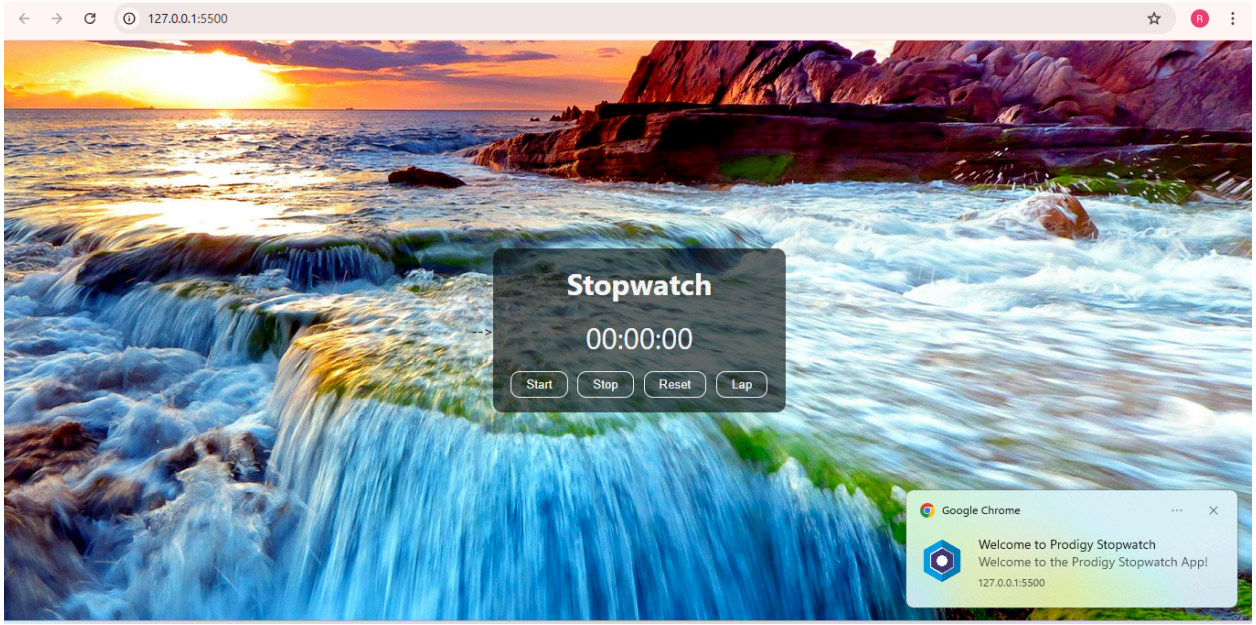
```
window.onload = function () {
    let minutes = 0;
    let seconds = 0;
    let tens = 0;
    let appendMinutes = document.querySelector('#minutes');
    let appendTens = document.querySelector('#tens');
    let appendSeconds = document.querySelector('#seconds');
    let startBtn = document.querySelector('#start');
    let stopBtn = document.querySelector('#stop');
    let resetBtn = document.querySelector('#reset');
    let lapBtn = document.querySelector('#lap');
    let lapList = document.querySelector('#lapList');
    let Interval;

    const startTimer = () => {
      tens++;
      if (tens <= 9) {
        appendTens.innerHTML = '0' + tens;
      }
      if (tens > 9) {
        appendTens.innerHTML = tens;
      }

      if (tens > 99) {
        seconds++;
        appendSeconds.innerHTML = '0' + seconds;
        tens = 0;
```

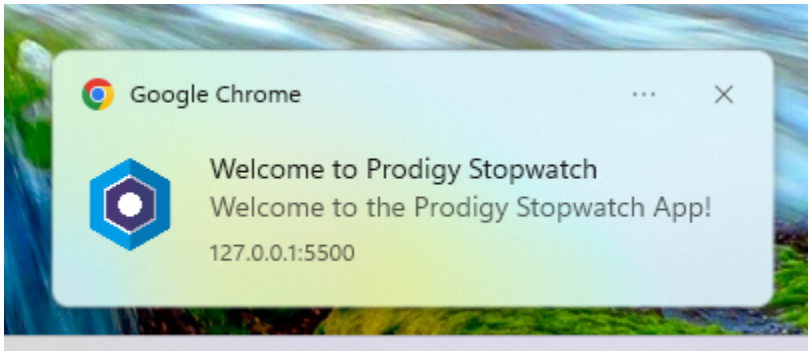
```
      appendTens.innerHTML = '0' + 0;
    }

    if (seconds > 9) {
      appendSeconds.innerHTML = seconds;
    }

    if (seconds > 59) {
      minutes++;
      appendMinutes.innerHTML = '0' + minutes;
      seconds = 0;
      appendSeconds.innerHTML = '0' + 0;
    }
  };

  startBtn.onclick = () => {
    clearInterval(Interval);
    Interval = setInterval(startTimer, 10);
  };

  stopBtn.onclick = () => {
    clearInterval(Interval);
  };

  resetBtn.onclick = () => {
    clearInterval(Interval);
    tens = '00';
    seconds = '00';
    minutes = '00';
    appendTens.innerHTML = tens;
    appendSeconds.innerHTML = seconds;
    appendMinutes.innerHTML = minutes;
    lapList.innerHTML = '';
  };

  lapBtn.onclick = () => {
    var li = document.createElement('li');
    li.innerHTML = appendMinutes.innerHTML + ':' + appendSeconds.innerHTML + '.' +
appendTens.innerHTML;
    lapList.appendChild(li);
  };
};

// Request Notification Permission when the app is loaded
```

```
if ('Notification' in window && 'serviceWorker' in navigator) {
  Notification.requestPermission().then(permission => {
    if (permission === 'granted') {
      console.log('Notification permission granted');
    }
  });
}
```

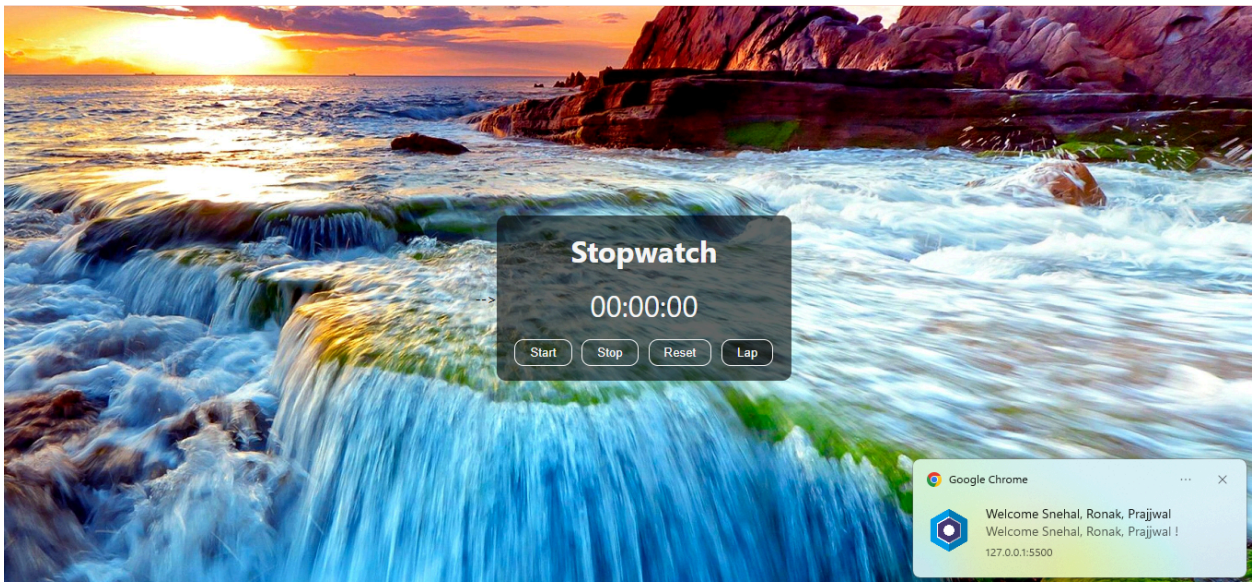
Screenshots:

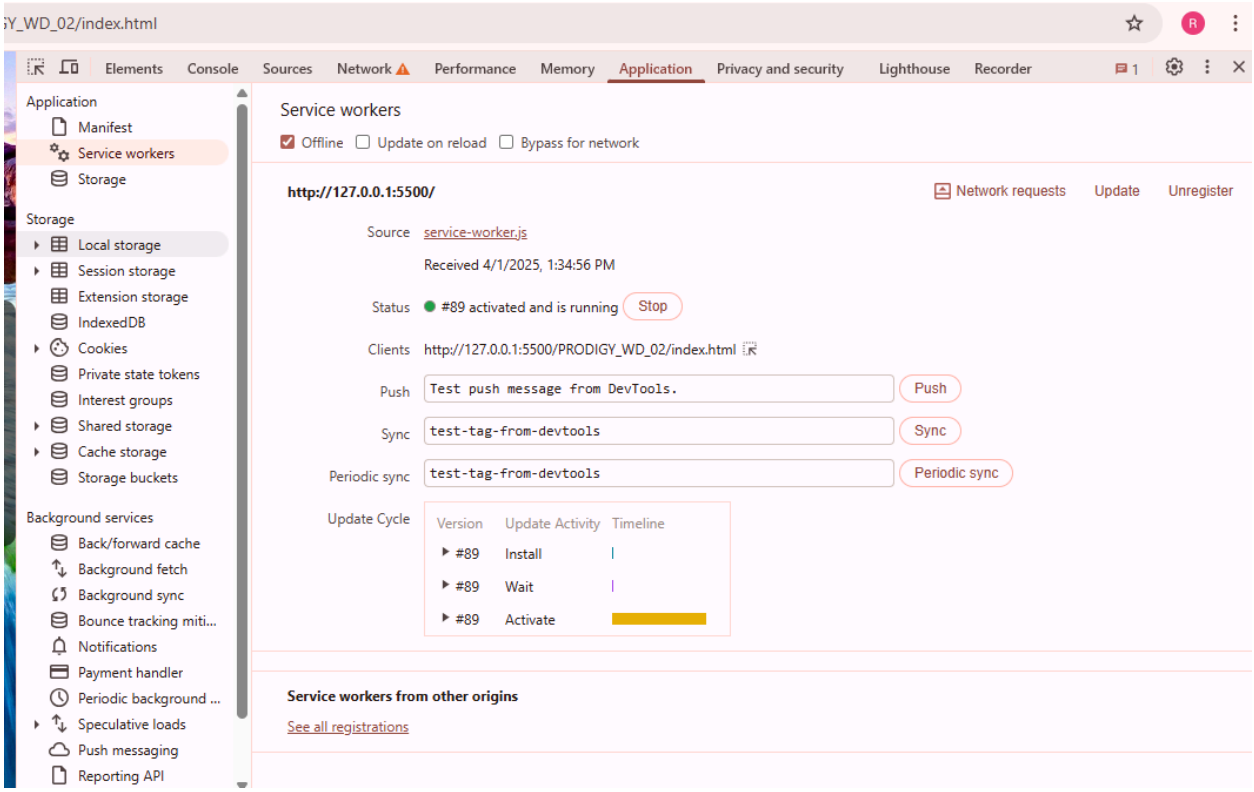


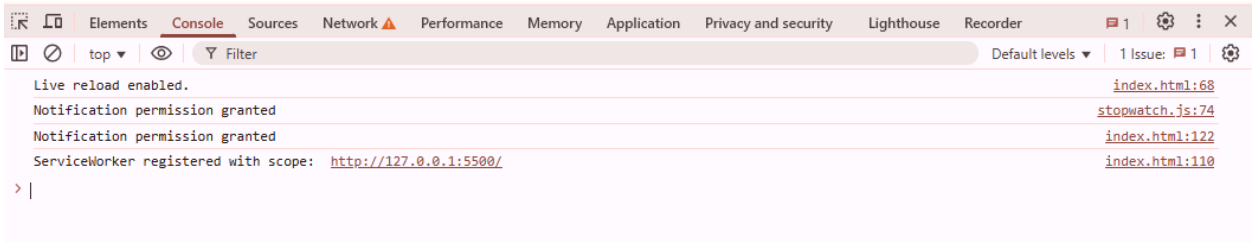





```
Notification permission granted                                  stopwatch.js:74
Notification permission granted                                  (index):122
ServiceWorker registered with scope:  http://127.0.0.1:5500/     (index):110
```




Y_WD_02/index.html
Elements
Console
Sources
Network
Performance
Memory
Application
Privacy and security
Lighthouse
Recorder
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking miti...
Notifications
Payment handler
Periodic background ...
Speculative loads
Push messaging
Reporting API
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Network requests
Update
Unregister
Source
service-worker.js
Received 4/1/2025, 1:34:56 PM
Status
#89 activated and is running
Stop
Clients
http://127.0.0.1:5500/PRODIGY_WD_02/index.html
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#89
Install
#89
Wait
#89
Activate
Service workers from other origins
See all registrations


Elements
Console
Sources
Network
Performance
Memory
Application
Privacy and security
Lighthouse
Recorder
top
Filter
Default levels
1 Issue:
Live reload enabled.
index.html:68
Notification permission granted
stopwatch.js:74
Notification permission granted
index.html:122
ServiceWorker registered with scope:
http://127.0.0.1:5500/
index.html:110

Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	32
Name	Prajjwal Pandey
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Experiment No.
10 MAD & PWA
Lab

Ronak Katariya 23
Prajjwal Pandey 32
Snehal Patil 38

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/ronak03rsk/PRODIGY_WD_02

Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	32
Name	Prajjwal Pandey
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

Experiment: Using Google Lighthouse for PWA Analysis

Objective:
To use Google Lighthouse PWA Analysis Tool to test the Progressive Web App (PWA) functionality.

Theory:
Google Lighthouse is an open-source automated tool used to audit and analyze web applications for performance, accessibility, best practices, SEO, and Progressive Web App (PWA) capabilities. Lighthouse evaluates whether a web app meets PWA standards, such as:

- Service worker registration for offline support
- Web App Manifest inclusion
- HTTPS security compliance
- Fast and responsive performance
- Installability and app-like user experience

Lighthouse assigns a score based on these criteria, helping developers optimize their PWA.

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:

1. **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.
2. **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.
3. **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.
4. **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.

Requirements:

- Google Chrome browser
- A PWA-enabled web application
- Chrome DevTools or Lighthouse CLI

Procedure:

1. Open Lighthouse in Chrome DevTools

- Launch the PWA in Google Chrome.
- Open DevTools using `F12` or `Ctrl+Shift+I`.
- Navigate to the **Lighthouse** tab.

2. Configure Lighthouse Audit

- Select the **Progressive Web App** category.
- Ensure other relevant categories like Performance and Best Practices are checked.
- Choose the mode:

    - **Mobile** (default) for testing mobile performance.
    - **Desktop** for desktop evaluation.
- Click **Generate Report**.

3. Analyze the Results

- Lighthouse generates a PWA score based on:
    - **Fast and reliable:** Checks service worker caching.
    - **Installable:** Verifies manifest and service worker.
    - **PWA Optimizations:** Ensures proper web app experience.
- Click on individual sections to view recommendations for improvements.

4. Running Lighthouse via CLI (Optional)

Install Lighthouse CLI using Node.js:

```
npm install -g lighthouse
```

-

Run an audit on a PWA URL:

```
lighthouse https://your-pwa-url.com --view
```

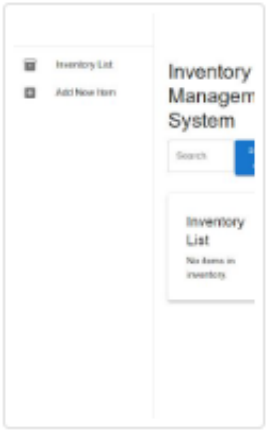
-

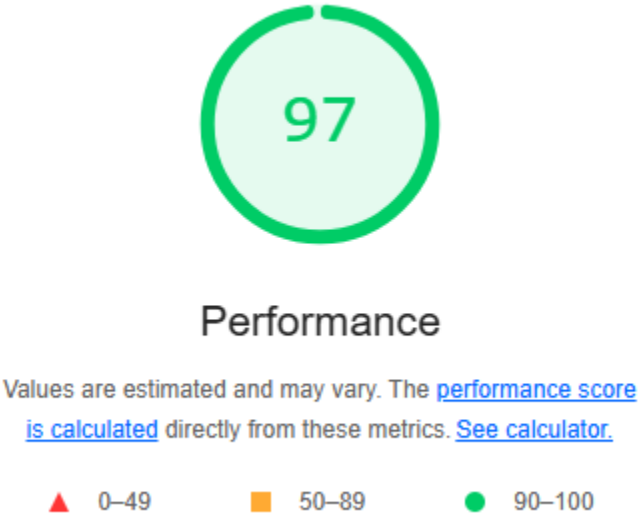
Observations:

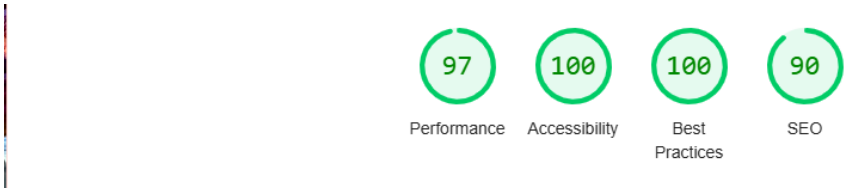
- The PWA is analyzed based on Lighthouse criteria.
- Identified areas needing improvement (e.g., caching strategies, manifest completeness, accessibility fixes).
- Scores indicate overall PWA compliance and optimization level.

Conclusion:
By using Google Lighthouse, we effectively evaluated the PWA's functionality, installability, and performance, enabling improvements for a better user experience.

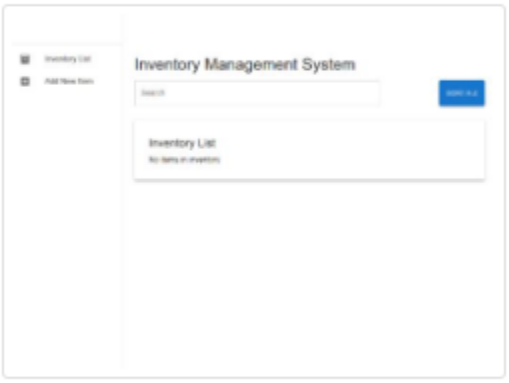
Mobile View:

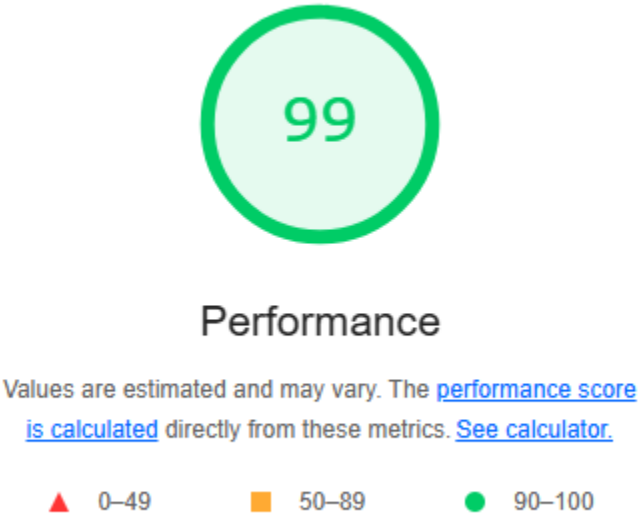


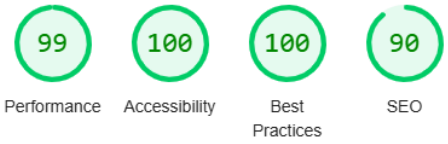




Desktop View:






99
Performance
100
Accessibility
100
Best Practices
90
SEO

Project Title: **Roll No.**

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.	32
Name	Prajjwal Pandey
Class	D15A/D15B
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:	

Project Title: **Roll No.**

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.	32
Name	Prajjwal Pandey
Class	D15A/D15B
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:	