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

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

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

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

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers : Mrs. Kajal Joseph.

Email : kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Lab Objectives:

Sr. No.	Lab Objectives
The Lab experiments aims:	
1	Learn the basics of the Flutter framework.
2	Develop the App UI by incorporating widgets, layouts, gestures and animation
3	Create a production ready Flutter App by including files and firebase backend service.
4	Learn the Essential technologies, and Concepts of PWAs to get started as quickly and efficiently as possible
5	Develop responsive web applications by combining AJAX development techniques with the jQuery JavaScript library.
6	Understand how service workers operate and also learn to Test and Deploy PWA.

Lab Outcomes:

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

Index

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

MAD & PWA Lab
Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	29
Name	Sanket Satish More
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

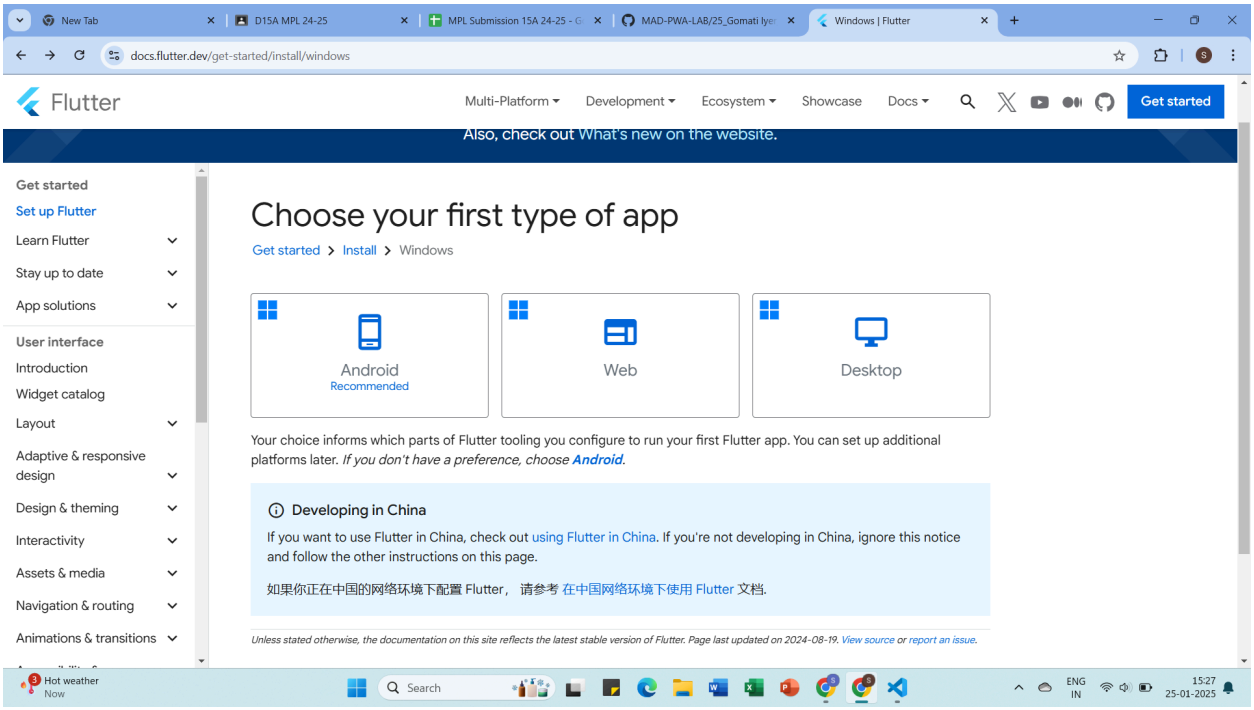
MPL Experiment 1

Sanket More
D15A 29

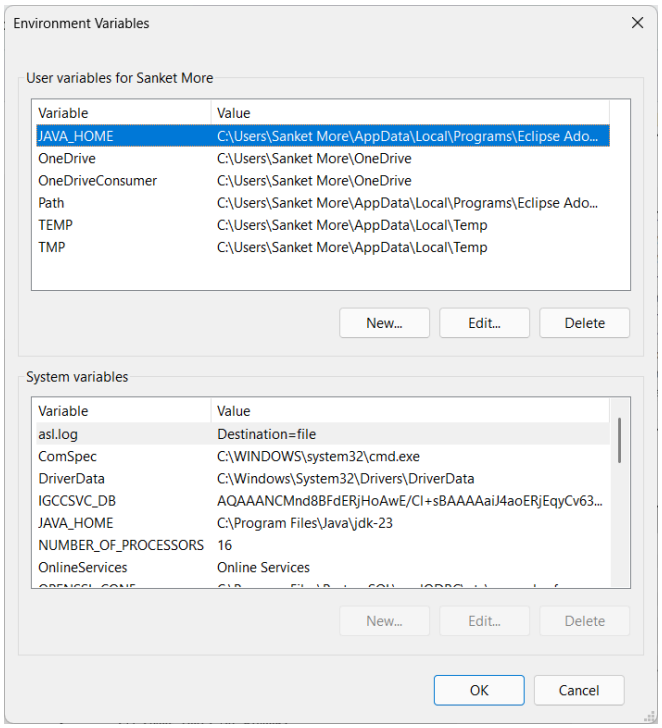
AIM: Installation and configuration of flutter Environment.

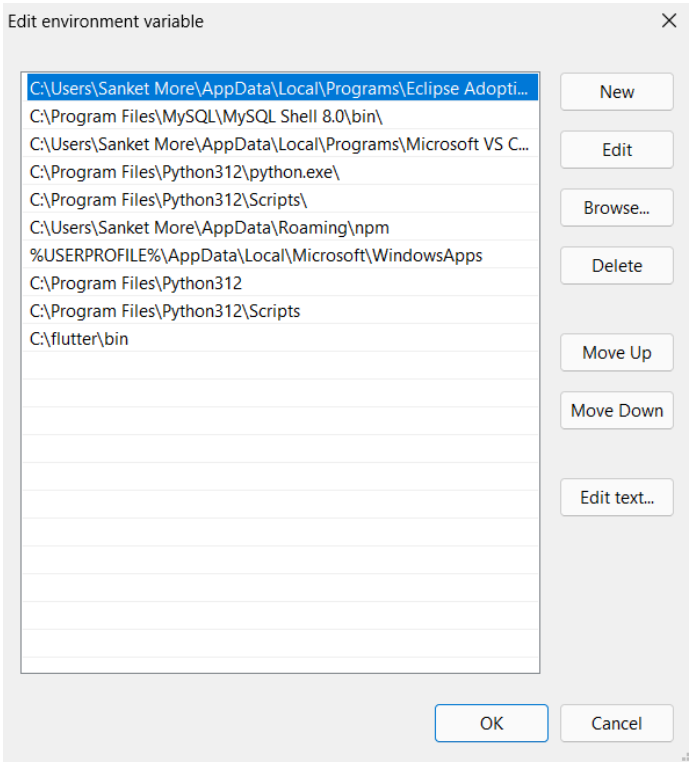
CODE:

Flutter Installation



Setting up Environment Variables



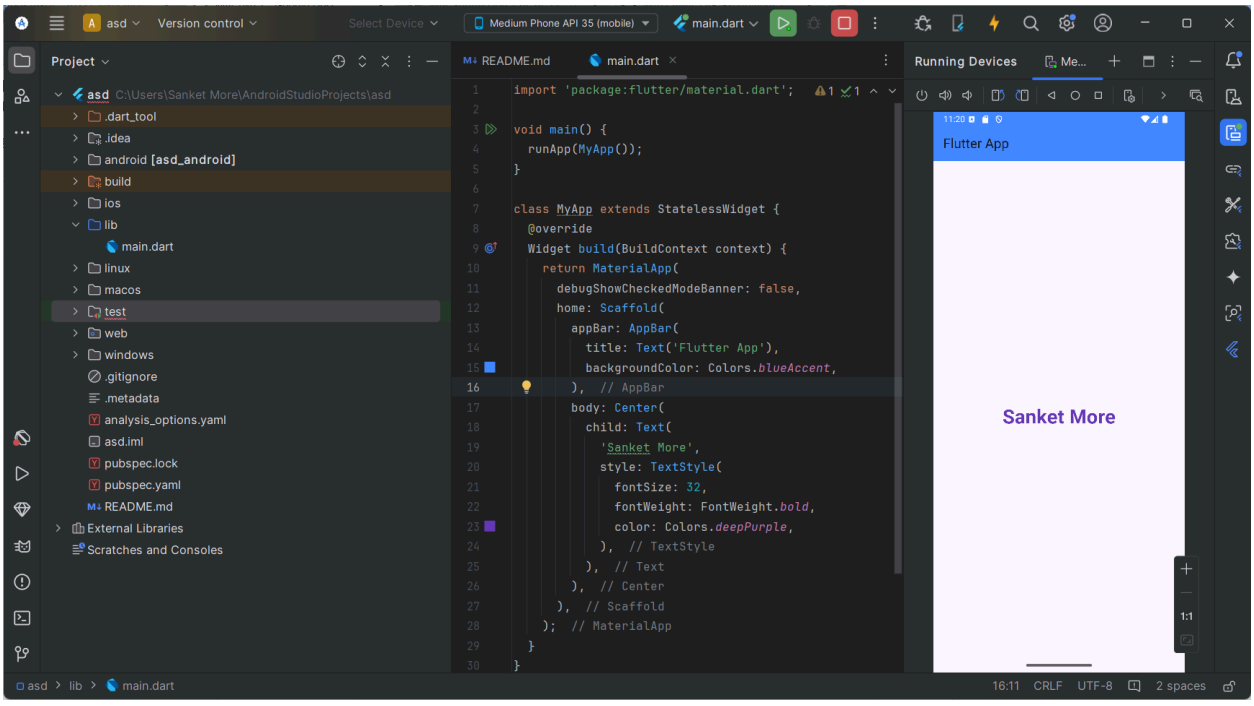


Checking for flutter on cmd:

```
Microsoft Windows [Version 10.0.26100.2894]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sanket More>flutter
Manage your Flutter app development.

Common commands:

  flutter create <output directory>
    Create a new Flutter project in the specified directory.

  flutter run [options]
    Run your Flutter application on an attached device or in an emulator.

Usage: flutter <command> [arguments]

Global options:
-h, --help                  Print this usage information.
-v, --verbose               Noisy logging, including all shell commands executed.
                            If used with "--help", shows hidden options. If used with "flutter doctor", shows additional
                            diagnostic information. (Use "-vv" to force verbose logging in those cases.)
-d, --device-id             Target device id or name (prefixes allowed).
    --version               Reports the version of this tool.
    --enable-analytics      Enable telemetry reporting each time a flutter or dart command runs.
    --disable-analytics     Disable telemetry reporting each time a flutter or dart command runs, until it is
                            re-enabled.
    --suppress-analytics    Suppress analytics reporting for the current CLI invocation.
```

```
C:\Users\Sanket More>flutter --version
Flutter 3.27.3 • channel stable • https://github.com/flutter/flutter.git
Framework • revision c519ee916e (4 days ago) • 2025-01-21 10:32:23 -0800
Engine • revision e672b006cb
Tools • Dart 3.6.1 • DevTools 2.40.2

C:\Users\Sanket More>
```

Flutter Basic Code:



MAD & PWA Lab

Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	29
Name	Sanket Satish More
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

MPL Experiment-2

Sanket More
D15A 29

AIM:To design Flutter UI using common widgets.

Theory:

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

● Stateless Widgets: Do not change their state once built (e.g., Text, Container).

● Stateful Widgets: Can update dynamically based on user interaction (e.g., TextField,Checkbox).

Commonly Used Widgets in Flutter-

(a) Scaffold Widget

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

(b) Container Widget

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

(c) Row and Column Widgets

● Row: Arranges widgets horizontally.

● Column: Arranges widgets vertically.

These two widgets are fundamental for designing layouts in Flutter.

(d) ListView Widget

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

(e) Stack Widget

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

(f) ElevatedButton Widget

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

(g) TextField Widget

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

CODE:-

home.dart

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

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

  @override
  State<HomeScreen> createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  final List<Map<String, dynamic>> _posts = [
    {
      'user': 'user1',
      'imageUrl': 'images/post.jpg',
      'caption': 'First post!',
      'likes': 100,
      'comments': 20,
    },
    {
      'user': 'user2',
      'imageUrl': 'images/post2.jpg',
      'caption': 'Another day, another post.',
      'likes': 50,
      'comments': 10,
    },
    // ... more posts
  ];

  final List<Map<String, dynamic>> _stories = [
    {
      'user': 'user1',
      'imageUrl': 'images/post.jpg', // Replace with your story image URLs
    },
    {
      'user': 'user2',
      'imageUrl': 'images/post2.jpg',
    },
    // ... more stories
  ];

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

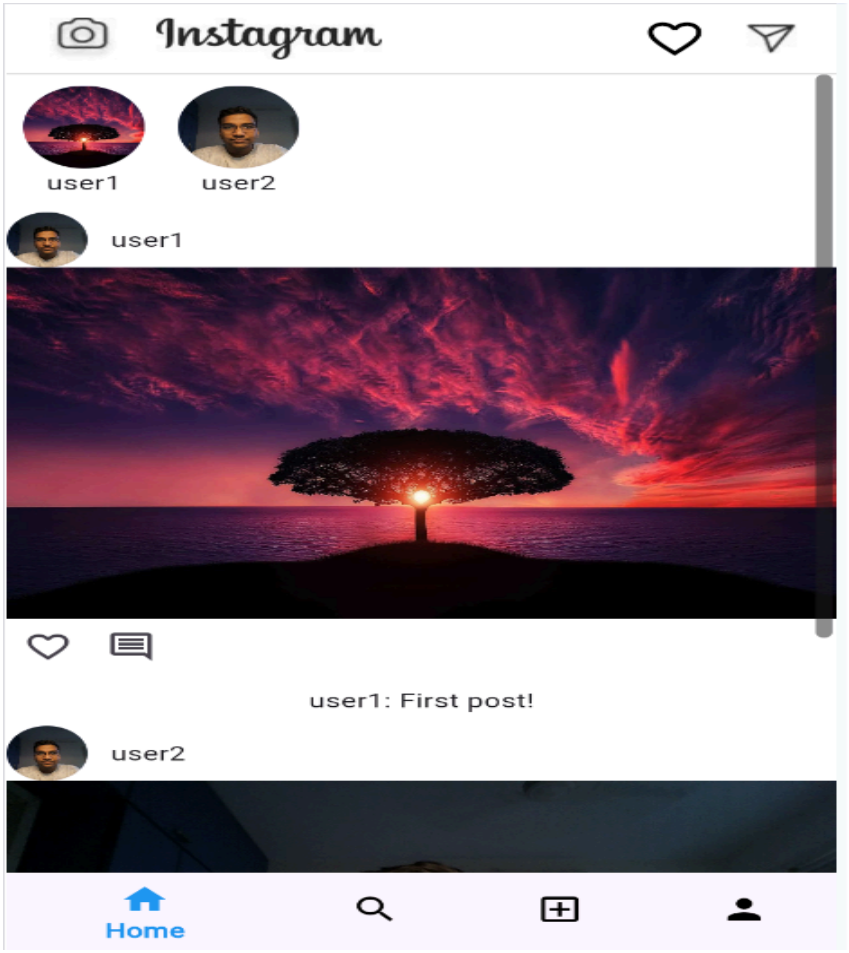
```
    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        elevation: 0,
        toolbarHeight: 50.h,
        title: Image.asset(
          'images/instagram.jpg',
          height: 30.h,
        ),
        leading: Padding(
          padding: EdgeInsets.only(left: 16.w),
          child: Image.asset(
            'images/camera.jpg',
            height: 24.h,
          ),
        ),
        actions: [
          Padding(
            padding: EdgeInsets.only(right: 16.w),
            child: Icon(
              Icons.favorite_border_outlined,
              color: Colors.black,
              size: 28.sp,
            ),
          ),
          Padding(
            padding: EdgeInsets.only(right: 16.w),
            child: Image.asset(
              'images/send.jpg',
              height: 24.h,
            ),
          ),
        ],
        backgroundColor: Colors.white,
        bottom: PreferredSize(
          preferredSize: Size.fromHeight(1.h),
          child: Container(
            height: 1.h,
            color: Colors.grey[300],
          ),
        ),
      ),
      body: SingleChildScrollView(
        // Added SingleChildScrollView
        child: Column(
```

```
          children: [
            // Stories Section
            SizedBox(
              height: 100.h, // Adjust height as needed
              child: ListView.builder(
                scrollDirection: Axis.horizontal,
                itemCount: _stories.length,
                itemBuilder: (context, index) {
                  return StoryWidget(_stories[index]);
                },
              ),
            ),
            // Posts Section
            ListView.builder(
              shrinkWrap: true, // Important for nested ListViews
              physics:
                  const NeverScrollableScrollPhysics(), // Disable scrolling of inner
ListView
              itemCount: _posts.length,
              itemBuilder: (context, index) {
                return PostWidget(_posts[index]);
              },
            ),
          ],
        ),
      ),
    );
  }
}

class StoryWidget extends StatelessWidget {
  final Map<String, dynamic> storyData;

  const StoryWidget(this.storyData, {super.key});

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(8.0),
      child: Column(
        children: [
          CircleAvatar(
            radius: 30.r,
            backgroundImage: AssetImage(storyData['imageUrl'] ??
                'images/default_profile.png'), // Use story image
```

```
          ),
          Text(storyData['user'] ?? ""),
        ],
      ),
    );
  }
}

class PostWidget extends StatelessWidget {
  final Map<String, dynamic> postData;

  const PostWidget(this.postData, {super.key});

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        // User Info
        Row(
          children: [
            CircleAvatar(
              radius: 20.r,
              backgroundImage: const AssetImage('images/post2.jpg'),
            ),
            SizedBox(width: 10.w),
            Text(postData['user'] ?? ""),
          ],
        ),

        // Post Image
        Image.asset(postData['imageUrl'] ?? ""),

        // Post Actions (Likes, Comments)
        Row(
          children: [
            IconButton(
              icon: const Icon(Icons.favorite_border),
              onPressed: () {
                // Handle like
              },
            ),
            IconButton(
              icon: const Icon(Icons.comment_outlined),
              onPressed: () {
                // Handle comment
```

```
              },
            ),
          ],
        ),

        // Caption
        Padding(
          padding: const EdgeInsets.all(8.0),
          child:
              Text('${postData['user'] ?? ""}: ${postData['caption'] ?? ""}'),
        ),
      ],
    );
  }
}
```

OUTPUT:-



add_post_screen.dart:-

```dart
// add_post_screen.dart
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'dart:io';

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

  @override
  State<AddPostScreen> createState() => _AddPostScreenState();
}

class _AddPostScreenState extends State<AddPostScreen> {
  File? _image;
  final TextEditingController _captionController = TextEditingController();
  bool _isLoading = false;
  List<String> _previousImageAssets = [];

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

  Future<void> _loadPreviousImages() async {
    setState(() {
      _previousImageAssets = [
        'images/person.png', // Replace with your actual image paths
        'images/post.jpg', // Make sure these images are in your assets
        'images/cat.png', // Replace with a valid image in your assets
      ];
    });
  }

  Future<void> _pickImage(ImageSource source) async {
    final pickedFile = await ImagePicker().pickImage(source: source);

    setState(() {
      if (pickedFile != null) {
        _image = File(pickedFile.path);
      } else {
        print('No image selected.');
      }
    });
```

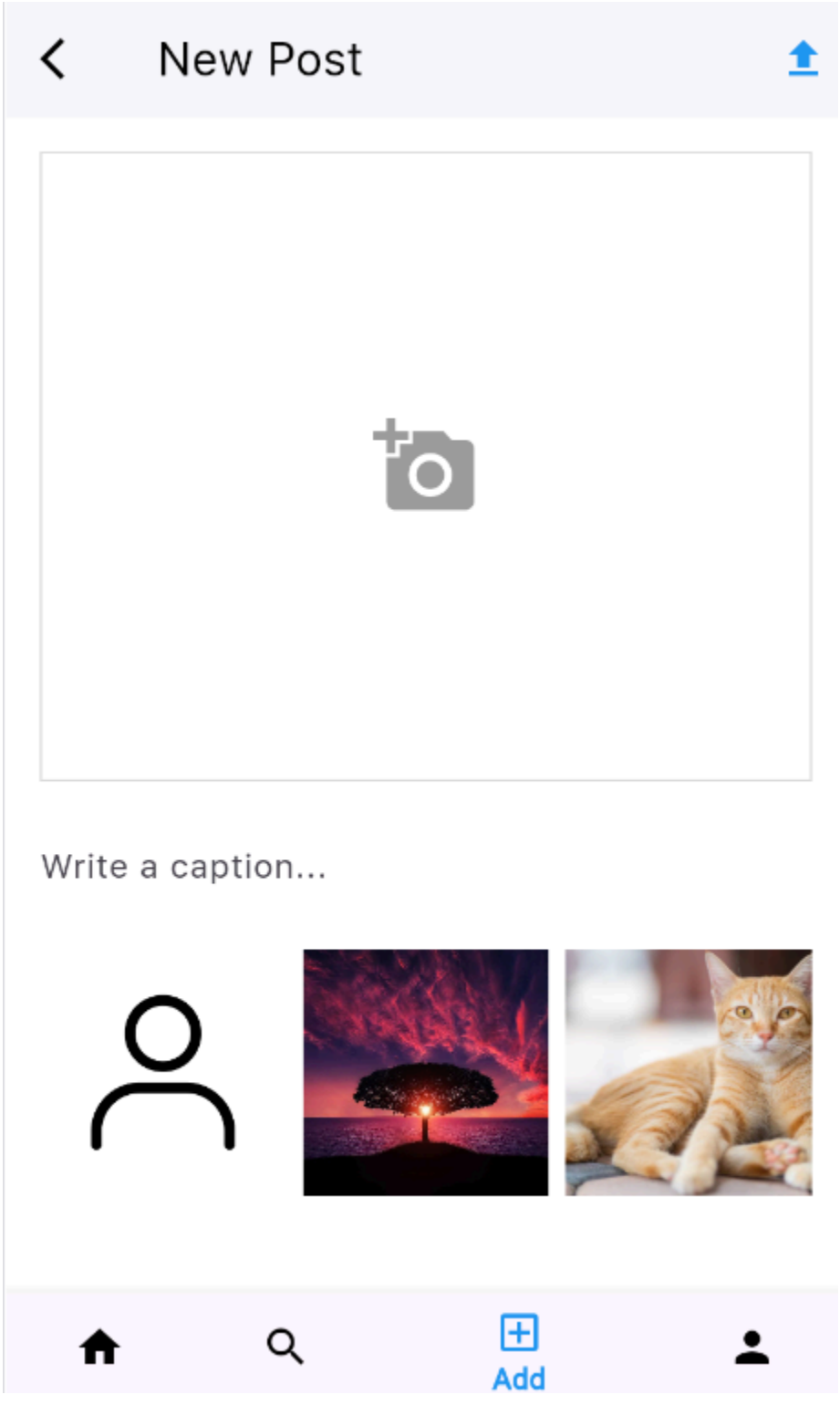
```
}

void _uploadPost() async {
  setState(() {
    _isLoading = true;
  });

  if (_image != null) {
    String caption = _captionController.text;

    // Simulate upload (replace with your actual logic)
    await Future.delayed(const Duration(seconds: 2));

    print("Uploading image: ${_image!.path}");
    print("Caption: $caption");

    setState(() {
      _image = null;
      _captionController.clear();
      _isLoading = false;
    });

    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Post uploaded successfully!')),
    );
  } else {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Please select an image')),
    );
    setState(() {
      _isLoading = false;
    });
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.white,
    appBar: AppBar(
      backgroundColor: Colors.white,
      elevation: 1,
      leading: IconButton(
        icon: const Icon(Icons.arrow_back_ios, color: Colors.black),
        onPressed: () {
```

```
            Navigator.pop(context);
          },
        ),
        title: const Text('New Post',
            style: TextStyle(color: Colors.black, fontFamily: 'SFProText')),
        actions: [
          IconButton(
            onPressed: _isLoading ? null : _uploadPost,
            icon: _isLoading
                ? const CircularProgressIndicator()
                : const Icon(Icons.upload, color: Colors.blue),
          ),
        ],
      ),
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              GestureDetector(
                onTap: () {
                  showModalBottomSheet(
                    context: context,
                    builder: (BuildContext context) {
                      return Column(
                        mainAxisSize: MainAxisSize.min,
                        children: <Widget>[
                          ListTile(
                            leading: const Icon(Icons.photo_library),
                            title: const Text('Photo Library'),
                            onTap: () {
                              _pickImage(ImageSource.gallery);
                              Navigator.pop(context);
                            },
                          ),
                          ListTile(
                            leading: const Icon(Icons.camera_alt),
                            title: const Text('Camera'),
                            onTap: () {
                              _pickImage(ImageSource.camera);
                              Navigator.pop(context);
                            },
                          ),
                        ],
```

```
                );
              },
            );
          },
          child: Container(
            height: 300,
            width: double.infinity,
            decoration: BoxDecoration(
              border: Border.all(color: Colors.grey.shade300),
            ),
            child: _image != null
                ? Image.file(_image!, fit: BoxFit.cover)
                : const Center(
                    child: Icon(Icons.add_a_photo,
                        size: 50, color: Colors.grey)),
          ),
        ),
        const SizedBox(height: 16),
        TextField(
          controller: _captionController,
          style: const TextStyle(fontFamily: 'SFProText'),
          decoration: const InputDecoration(
            hintText: 'Write a caption...',
            hintStyle: TextStyle(fontFamily: 'SFProText'),
            border: InputBorder.none,
          ),
          maxLines: null,
        ),
        const SizedBox(height: 16),
        GridView.builder(
          shrinkWrap: true,
          physics: const NeverScrollableScrollPhysics(),
          gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
            crossAxisCount: 3,
            crossAxisSpacing: 8,
            mainAxisSpacing: 8,
          ),
          itemCount: _previousImageAssets.length,
          itemBuilder: (context, index) {
            return Image.asset(
              _previousImageAssets[index],
              fit: BoxFit.cover,
            );
          },
        ),
```

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

OUTPUT:-



MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	29
Name	Sanket Satish More
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

MPL Experiment-3

Sanket More
D15A 29

AIM:To include images,icons and fonts in flutter app.

Theory:

Using Icons in Flutter

Icons in Flutter can be added using the built-in Material Icons or custom icon packs.

(a) **Material Icons**

Flutter provides a collection of built-in Material Icons, which can be used with the Icon widget.

Eg: Icon(Icons.home, size: 30, color: Colors.blue)

(b) **Custom Icons**

If you need icons that are not available in the Material Icons set, you can use external icon packs like:

- Font Awesome (font_awesome_utter package)
- Custom SVG Icons (utter_svg package)

Eg in pubspec.yaml le -

```
dependencies:
 font_awesome_utter: ^10.5.0
```

In code -

```
import 'package:font_awesome_utter/font_awesome_utter.dart';
IconButton(
 icon: FaIcon(FontAwesomeIcons.heart, color: Colors.red),
 onPressed: () {},
)
```

Adding Images in Flutter

Images can be loaded in Flutter from different sources like assets, network, or memory.

(a) **Using Network Images**

Network images are loaded from an online URL. Example:

Eg: Image.network("https://example.com/sample.jpg", width: 200, height: 150)

(b) **Using Asset Images**

To use images from the local project folder (assets/), follow these steps:

1. Place the image inside the assets/images/ folder.
2. Declare the image in pubspec.yaml:

```
utter:
 assets:
```

- assets/images/sample.png
In code: Image.asset("assets/images/sample.png", width: 200, height: 150)

Adding Custom Fonts in Flutter

Custom fonts improve the visual identity of an app.

Steps to Add a Custom Font:

1. Download the font and place it inside the assets/fonts/ folder.
2. Declare the font in pubspec.yaml:

utter:
fonts:
- family: CustomFont
fonts:
- asset: assets/fonts/CustomFont-Regular.ttf
- asset: assets/fonts/CustomFont-Bold.ttf
weight: 700

In code -

Text(
"Hello, Flutter!",
style: TextStyle(fontFamily: "CustomFont", fontSize: 20, fontWeight: FontWeight.bold),
)

CODE:-

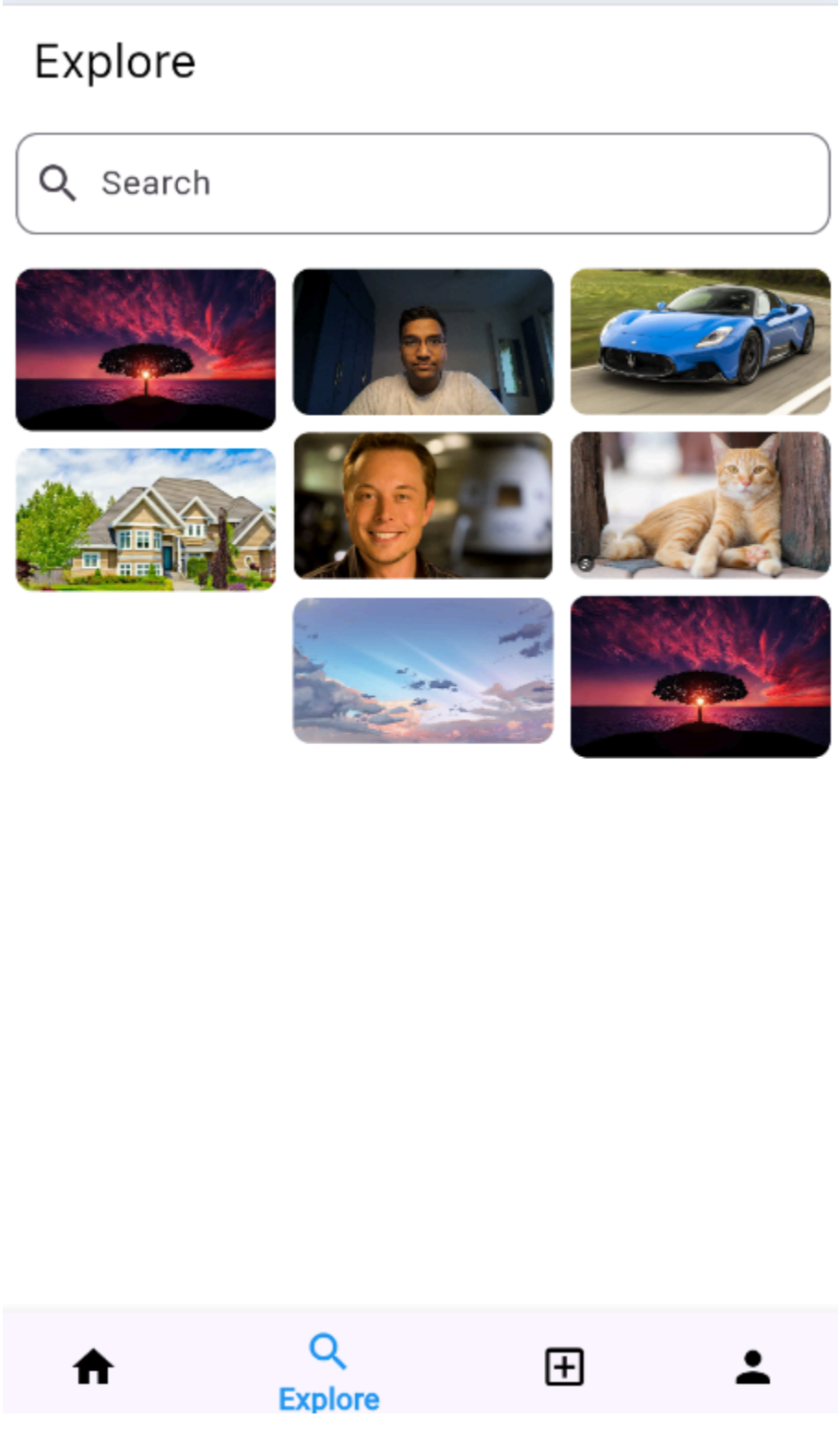
explore.dart

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

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

  @override
  State<ExploreScreen> createState() => _ExploreScreenState();
}

class _ExploreScreenState extends State<ExploreScreen> {
  final List<String> _imageUrls = [
    'images/post.jpg',
    'images/post2.jpg',
    'images/car.png',
    'images/cat.png',
    'images/elon.png',
```

```
    'images/house.png',
    'images/post.jpg',
    'images/sky.png',
  ];

  // Add a TextEditingController for the search bar
  final TextEditingController _searchController = TextEditingController();

  // Add a filtered list of image URLs
  List<String> _filteredImageUrls = [];

  @override
  void initState() {
    super.initState();
    _filteredImageUrls = _imageUrls; // Initialize with all images
  }

  void _filterImages(String query) {
    setState(() {
      if (query.isEmpty) {
        _filteredImageUrls = _imageUrls; // Show all images if search is empty
      } else {
        _filteredImageUrls = _imageUrls
            .where((imageUrl) =>
                imageUrl.toLowerCase().contains(query.toLowerCase()))
            .toList();
      }
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        backgroundColor: Colors.white,
        elevation: 0,
        title: const Text('Explore',
            style: TextStyle(color: Colors.black, fontFamily: 'SFProText')),
        bottom: PreferredSize(
          // Add a search bar below the app bar
          preferredSize: const Size.fromHeight(60.0),
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: TextField(
```

```
              controller: _searchController,
              onChanged: _filterImages, // Filter images on text change
              decoration: InputDecoration(
                hintText: 'Search',
                prefixIcon: const Icon(Icons.search),
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(10.0),
                ),
              ),
            ),
          ),
        ),
      ),
      body: Padding(
        padding: const EdgeInsets.all(8.0),
        child: MasonryGridView.count(
          crossAxisCount: 3,
          mainAxisSpacing: 8.0,
          crossAxisSpacing: 8.0,
          itemCount: _filteredImageUrls.length, // Use the filtered list
          itemBuilder: (context, index) {
            return ClipRRect(
              borderRadius: BorderRadius.circular(8.0),
              child: Image.asset(
                _filteredImageUrls[index], // Use the filtered list
                fit: BoxFit.cover,
              ),
            );
          },
        ),
      ),
    );
  }
}
```

OUTPUT:-



MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	29
Name	Sanket Satish More
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

MPL Experiment - 4

Sanket More
D15A 29

AIM: To create an interactive form using form widget

Theory:

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

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

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

Steps:

1. **Create a new Flutter project or open an existing one.**
    - You can create a new Flutter project using the command: `flutter create form_example`.
2. **Define a Form widget inside a StatefulWidget to manage user input.**
    - Use a `StatefulWidget` to maintain the state of the form, allowing for dynamic validation and input handling.
3. **Use TextFormField for text input fields with validation logic.**
    - Use `TextFormField` for user input, with built-in validation to ensure that the fields are not left empty and that the data is in the correct format.

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

    - Add interactive widgets like `DropdownButton`, `Checkbox`, and `Switch` to collect different types of user input.
5. **Wrap the form inside a SingleChildScrollView to ensure smooth scrolling.**

    - This ensures that when the keyboard appears or the form becomes long, users can still scroll through the form fields.
6. **Implement an ElevatedButton to trigger form validation and submission.**

    - Use an `ElevatedButton` to trigger the validation of the form fields and perform any necessary actions after the form is validated.
7. **Use GlobalKey to manage form validation.**

    - A `GlobalKey<FormState>` is essential to manage the form state, particularly for validation and form submission. The key allows you to validate all fields before submission.

CODE:-

signup.dart

```
import 'dart:io';

import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

class SignupScreen extends StatefulWidget {
  final Function(BuildContext) showLogin;
  const SignupScreen({Key? key, required this.showLogin}) : super(key: key);

  @override
  State<SignupScreen> createState() => _SignupScreenState();
}

class _SignupScreenState extends State<SignupScreen> {
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _usernameController = TextEditingController();
  final TextEditingController _nameController =
      TextEditingController(); // Added name controller
  final TextEditingController _passwordController = TextEditingController();
  final TextEditingController _confirmPasswordController =
```

```
    TextEditingController();

File? _profileImage;

Future<void> _pickImage() async {
  final pickedFile =
      await ImagePicker().pickImage(source: ImageSource.gallery);
  if (pickedFile != null) {
    setState(() {
      _profileImage = File(pickedFile.path);
    });
  }
}

void _signup() {
  // Implement your signup logic here
  print('Name: ${_nameController.text}');
  print('Email: ${_emailController.text}');
  print('Username: ${_usernameController.text}');
  print('Password: ${_passwordController.text}');
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.white, // Instagram background color
    body: SafeArea(
      child: Padding(
        padding: EdgeInsets.symmetric(horizontal: 24.w), // Consistent padding
        child: SingleChildScrollView(
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.stretch, // Align to start
            children: [
              SizedBox(height: 40.h),

              Center(
                // Center the title
                child: Text(
                  'Sign Up',
                  style: TextStyle(
                    fontSize: 24.sp,
                    fontWeight: FontWeight.w500,
                    color: Colors.black,
                  ),
                ),
```

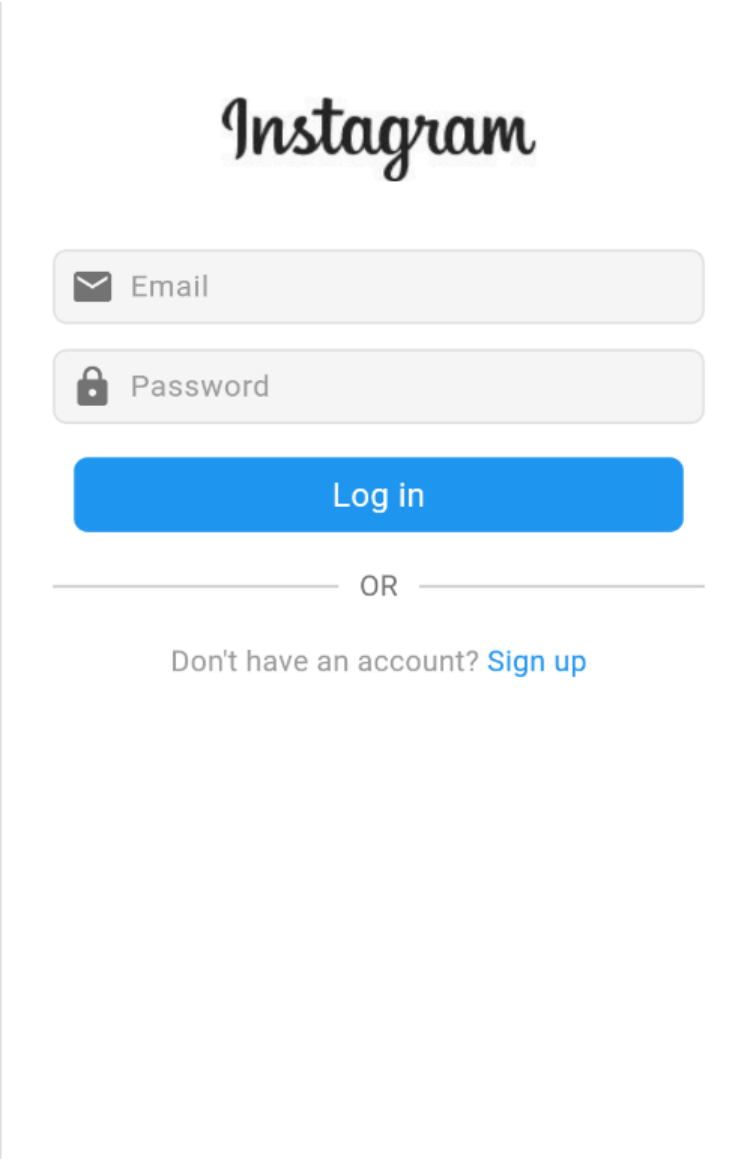
```
            ),

            SizedBox(height: 30.h),

            GestureDetector(
              // Image picker area
              onTap: _pickImage,
              child: CircleAvatar(
                radius: 40.r,
                backgroundColor: Colors.grey[200], // Light background
                backgroundImage: _profileImage != null
                    ? FileImage(_profileImage!)
                    : null,
                child: _profileImage == null
                    ? Icon(
                        Icons.add_a_photo,
                        size: 30.r,
                        color: Colors.grey,
                      )
                    : null,
              ),
            ),

            SizedBox(height: 20.h),

            _buildTextField(_nameController, 'Full Name',
                Icons.person), // Added name field
            SizedBox(height: 15.h),
            _buildTextField(_emailController, 'Email', Icons.email),
            SizedBox(height: 15.h),
            _buildTextField(
                _usernameController, 'Username', Icons.person_outline),
            SizedBox(height: 15.h),
            _buildTextField(_passwordController, 'Password', Icons.lock,
                obscureText: true),
            SizedBox(height: 15.h),
            _buildTextField(
                _confirmPasswordController, 'Confirm Password', Icons.lock,
                obscureText: true),

            SizedBox(height: 25.h),

            _buildSignupButton(), // Styled button

            SizedBox(height: 25.h),
```

```
              _buildOrDivider(),

              SizedBox(height: 25.h),

              _buildLoginLink(), // Instagram-style login link
            ],
          ),
        ),
      ),
    ),
  );
}

Widget _buildOrDivider() {
  return Row(
    children: [
      Expanded(
        child: Divider(
          color: Colors.grey[400],
          thickness: 1,
        ),
      ),
      Padding(
        padding: EdgeInsets.symmetric(horizontal: 10.w),
        child: Text(
          'OR',
          style: TextStyle(
            fontSize: 14.sp,
            color: Colors.grey[600],
          ),
        ),
      ),
      Expanded(
        child: Divider(
          color: Colors.grey[400],
          thickness: 1,
        ),
      ),
    ],
  );
}

Widget _buildSignupButton() {
  return ElevatedButton(
```

```
      onPressed: _signup,
      child: const Text('Sign Up'),
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.blue, // Instagram blue
        minimumSize: Size(double.infinity, 44.h), // Consistent button height
        shape: RoundedRectangleBorder(
          // Rounded corners
          borderRadius: BorderRadius.circular(8.r),
        ),
        textStyle: TextStyle(
          fontSize: 16.sp,
          fontWeight: FontWeight.w500,
        ),
      ),
    );
  }

  Widget _buildLoginLink() {
    return Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Text(
          "Already have an account? ",
          style: TextStyle(fontSize: 14.sp, color: Colors.grey),
        ),
        GestureDetector(
          onTap: () {
            widget.showLogin(context);
          },
          child: Text(
            "Log in",
            style: TextStyle(
              color: Colors.blue,
              fontWeight: FontWeight.w500,
              fontSize: 14.sp,
            ),
          ),
        ),
      ],
    );
  }

  Widget _buildTextField(
      TextEditingController controller, String labelText, IconData icon,
      {bool obscureText = false}) {
```

```
    return Container(
      height: 44.h,
      decoration: BoxDecoration(
        color: Colors.grey[100], // Light gray background
        borderRadius: BorderRadius.circular(8.r), // Rounded corners
        border: Border.all(color: Colors.grey[300]!), // Added border
      ),
      child: TextField(
        controller: controller,
        obscureText: obscureText,
        style: TextStyle(fontSize: 16.sp, color: Colors.black),
        decoration: InputDecoration(
          labelText: labelText,
          labelStyle: TextStyle(color: Colors.grey),
          prefixIcon: Icon(
            icon,
            color: Colors.grey[600],
          ),
          contentPadding:
              EdgeInsets.symmetric(horizontal: 15.w, vertical: 12.h),
          border: InputBorder.none,
          focusedBorder: InputBorder.none,
          enabledBorder: InputBorder.none,
        ),
      ),
    );
  }
}
```

OUTPUT:-



login_screen.dart

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

class LoginScreen extends StatefulWidget {
  final Function(BuildContext) showSignup;
  const LoginScreen({Key? key, required this.showSignup}) : super(key: key);

  @override
  State<LoginScreen> createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  final email = TextEditingController();
  final password = TextEditingController();
```

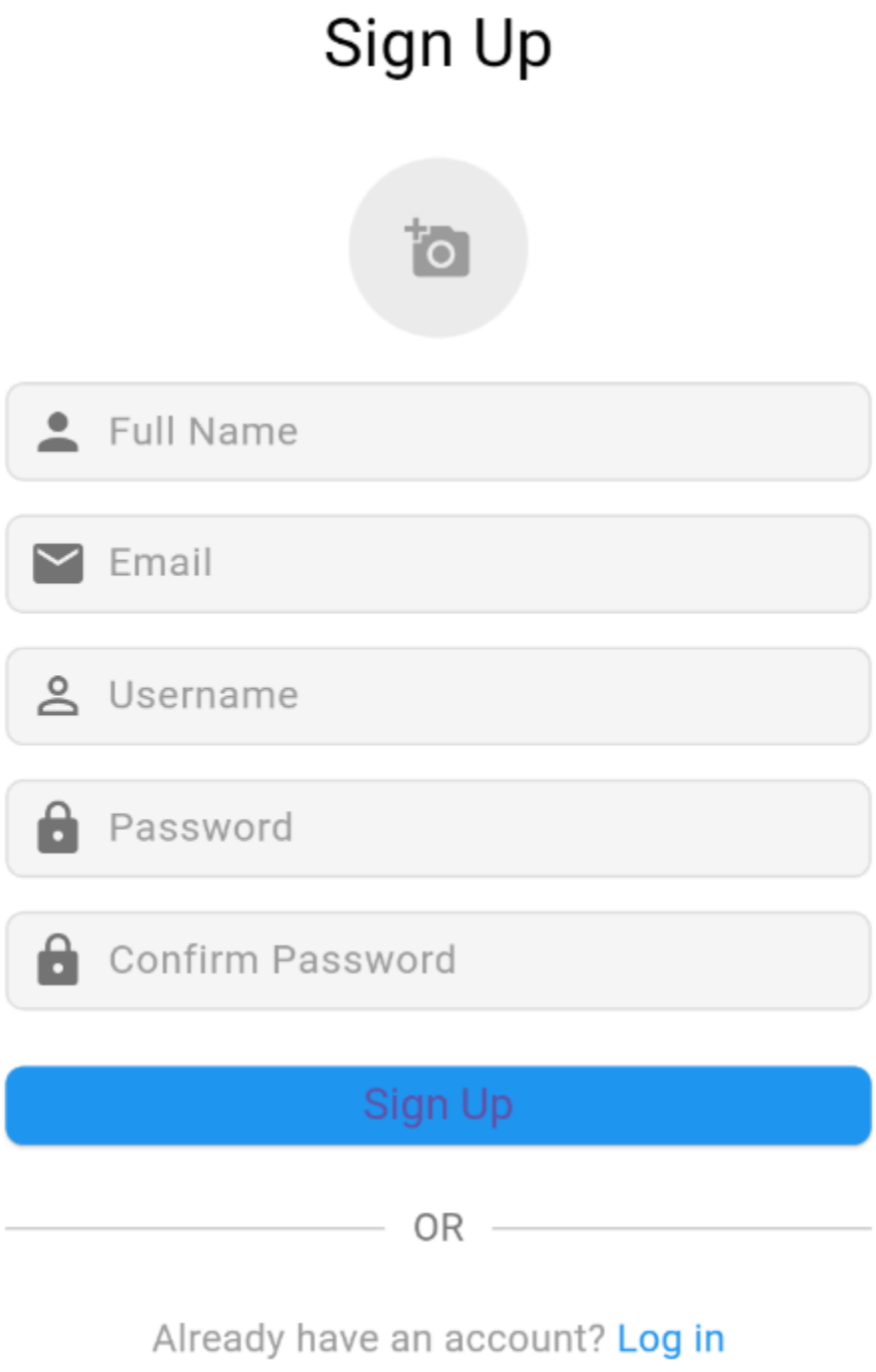
```dart
FocusNode emailFocus = FocusNode();
FocusNode passwordFocus = FocusNode();

@override
void dispose() {
  email.dispose();
  password.dispose();
  emailFocus.dispose();
  passwordFocus.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    resizeToAvoidBottomInset: false,
    backgroundColor: Colors.white,
    body: SafeArea(
      child: Padding(
        padding: EdgeInsets.symmetric(horizontal: 24.w),
        child: SingleChildScrollView(
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.stretch,
            children: [
              SizedBox(height: 60.h),
              Center(
                child: Image.asset(
                  'images/logo.jpg',
                  height: 50.h,
                ),
              ),
              SizedBox(height: 40.h),
              _buildTextField(email, emailFocus, 'Email', Icons.email),
              SizedBox(height: 15.h),
              _buildTextField(
                  password, passwordFocus, 'Password', Icons.lock),
              SizedBox(height: 20.h),
              _buildLoginButton(),
              SizedBox(height: 20.h),
              _buildOrDivider(),
              SizedBox(height: 20.h),
              _buildSignupButton(context),
            ],
          ),
        ),
```

```
      ),
    ),
  );
}

Widget _buildOrDivider() {
  return Row(
    children: [
      Expanded(
        child: Divider(
          color: Colors.grey[400],
          thickness: 1,
        ),
      ),
      Padding(
        padding: EdgeInsets.symmetric(horizontal: 10.w),
        child: Text(
          'OR',
          style: TextStyle(
            fontSize: 14.sp,
            color: Colors.grey[600],
          ),
        ),
      ),
      Expanded(
        child: Divider(
          color: Colors.grey[400],
          thickness: 1,
        ),
      ),
    ],
  );
}

Widget _buildSignupButton(BuildContext context) {
  return Padding(
    padding: EdgeInsets.symmetric(horizontal: 10.w),
    child: Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Text(
          "Don't have an account? ",
          style: TextStyle(
            fontSize: 14.sp,
            color: Colors.grey,
```

```
          ),
        ),
        GestureDetector(
          onTap: () {
            widget.showSignup(context);
          },
          child: Text(
            "Sign up",
            style: TextStyle(
              fontSize: 14.sp,
              color: Colors.blue,
              fontWeight: FontWeight.w500,
            ),
          ),
        ),
      ],
    ),
  );
}

Widget _buildLoginButton() {
  return Padding(
    padding: EdgeInsets.symmetric(horizontal: 10.w),
    child: InkWell(
      onTap: () {
        // Handle login here
      },
      child: Container(
        alignment: Alignment.center,
        width: double.infinity,
        height: 44.h,
        decoration: BoxDecoration(
          color: Colors.blue,
          borderRadius: BorderRadius.circular(8.r),
        ),
        child: Text(
          'Log in',
          style: TextStyle(
            fontSize: 16.sp,
            color: Colors.white,
            fontWeight: FontWeight.w500,
          ),
        ),
      ),
    ),
```

```
    );
  }

  Padding _buildTextField(TextEditingController controller, FocusNode focusNode,
      String labelText, IconData icon) {
    return Padding(
      padding: EdgeInsets.symmetric(horizontal: 0.w),
      child: Container(
        height: 44.h,
        decoration: BoxDecoration(
          color: Colors.grey[100],
          borderRadius: BorderRadius.circular(8.r),
          border: Border.all(color: Colors.grey[300]!),
        ),
        child: TextField(
          style: TextStyle(fontSize: 16.sp, color: Colors.black),
          controller: controller,
          focusNode: focusNode,
          decoration: InputDecoration(
            labelText: labelText,
            labelStyle: TextStyle(color: Colors.grey),
            prefixIcon: Icon(
              icon,
              color: focusNode.hasFocus ? Colors.blue : Colors.grey[600],
            ),
            contentPadding:
                EdgeInsets.symmetric(horizontal: 15.w, vertical: 12.h),
            border: InputBorder.none,
            focusedBorder: InputBorder.none,
            enabledBorder: InputBorder.none,
          ),
        ),
      ),
    );
  }
}
```

OUTPUT:-


Sign Up
Full Name
Email
Username
Password
Confirm Password
Sign Up
OR
Already have an account? Log in

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	29
Name	Sanket Satish More
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

MPL Experiment-5

Sanket More
D15A 29

AIM: To apply navigation and routing in flutter application.

Theory:

Flutter provides tools to handle navigation, routing, and gestures, allowing users to move between screens and interact with the app smoothly. These features help create a user-friendly experience in mobile applications.

1. **Navigation in Flutter**

Navigation is the process of moving between different screens (or pages) in a Flutter app. Flutter uses a stack-based approach for navigation, where new screens are pushed onto the stack and removed when the user navigates back.

Types of Navigation:

- Push Navigation: Moves to a new screen and adds it to the stack.
- Pop Navigation: Removes the current screen and returns to the previous one.
- Named Routes: Uses pre-dened route names to navigate.
- Navigation with Data: Allows passing data between screens when navigating.

2. **Routing in Flutter**

Routing helps in managing different screens eciently. Instead of manually handling each screen transition, Flutter allows dening routes in a structured way.

Types of Routing:

- Direct Routing: Navigates to a specic screen using explicit methods.
- Named Routing: Uses a predened route name to navigate, making the app more organized.

Routing improves app maintainability, especially in apps with multiple screens.

3. **Gestures in Flutter**

Gestures enable user interaction in Flutter applications. Flutter provides built-in gesture detection capabilities for touch-based interactions.

Common Gestures:

- Tap: A single touch interaction.
- Double Tap: Two quick consecutive taps.
- Long Press: Holding a touch for a longer duration.
- Swipe: Moving a nger across the screen.

● Drag: Moving an object by pressing and holding it.
Gestures are essential for making apps interactive and responsive.

4. **Combining Navigation and Gestures**
Navigation and gestures can be combined to enhance user experience. For example:
● Tapping on a button can navigate to another screen.
● Swiping a card can delete an item or move to another page.
● Dragging an element can reposition items within the app.
Navigation, routing, and gestures are fundamental to creating an interactive Flutter application. Navigation allows movement between screens, routing helps manage screens seciently, and gestures enable touch interactions. Mastering these concepts helps in developing dynamic and user-friendly Flutter applications.

CODE:
navigation.dart

```dart
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import '../screen/home.dart';
import '../screen/explore.dart';
import '../screen/add_post_screen.dart';

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

  @override
  State<NavigationScreen> createState() => _NavigationScreenState();
}

class _NavigationScreenState extends State<NavigationScreen> {
  int _selectedIndex = 0;

  static const List<Widget> _widgetOptions = <Widget>[
    HomeScreen(),
    ExploreScreen(),
    AddPostScreen(),
  ];

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

```dart
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: _widgetOptions.elementAt(_selectedIndex),
      ),
      bottomNavigationBar: BottomNavigationBar(
        items: const <BottomNavigationBarItem>[
          BottomNavigationBarItem(
            icon: Icon(Icons.home),
            label: 'Home',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.search),
            label: 'Explore',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.add_box_outlined),
            label: 'Add',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.person),
            label: 'Profile',
          ),
        ],
        currentIndex: _selectedIndex,
        selectedItemColor: Colors.blue, // Highlight color
        unselectedItemColor: Colors.black, // Set unselected item color to black
        selectedLabelStyle:
            const TextStyle(fontWeight: FontWeight.bold), // Bold selected label
        onTap: _onItemTapped,
      ),
    );
  }
}
```

add_post_screen.dart

```dart
// add_post_screen.dart
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'dart:io';

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

  @override
  State<AddPostScreen> createState() => _AddPostScreenState();
}

class _AddPostScreenState extends State<AddPostScreen> {
  File? _image;
  final TextEditingController _captionController = TextEditingController();
  bool _isLoading = false;
  List<String> _previousImageAssets = [];

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

  Future<void> _loadPreviousImages() async {
    setState(() {
      _previousImageAssets = [
        'images/person.png', // Replace with your actual image paths
        'images/post.jpg', // Make sure these images are in your assets
        'images/cat.png', // Replace with a valid image in your assets
      ];
    });
  }

  Future<void> _pickImage(ImageSource source) async {
    final pickedFile = await ImagePicker().pickImage(source: source);

    setState(() {
      if (pickedFile != null) {
        _image = File(pickedFile.path);
      } else {
        print('No image selected.');
      }
    });
```

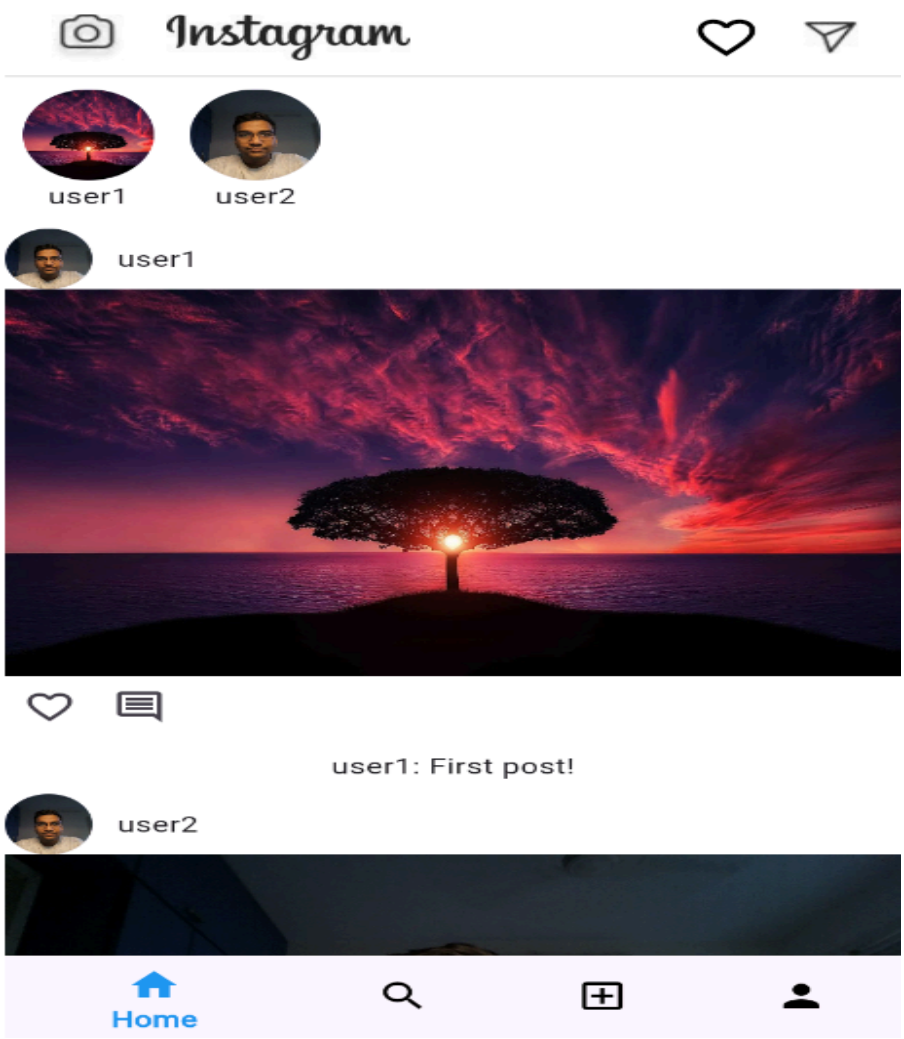
```
}

void _uploadPost() async {
  setState(() {
    _isLoading = true;
  });

  if (_image != null) {
    String caption = _captionController.text;

    // Simulate upload (replace with your actual logic)
    await Future.delayed(const Duration(seconds: 2));

    print("Uploading image: ${_image!.path}");
    print("Caption: $caption");

    setState(() {
      _image = null;
      _captionController.clear();
      _isLoading = false;
    });

    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Post uploaded successfully!')),
    );
  } else {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Please select an image')),
    );
    setState(() {
      _isLoading = false;
    });
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.white,
    appBar: AppBar(
      backgroundColor: Colors.white,
      elevation: 1,
      leading: IconButton(
        icon: const Icon(Icons.arrow_back_ios, color: Colors.black),
        onPressed: () {
```

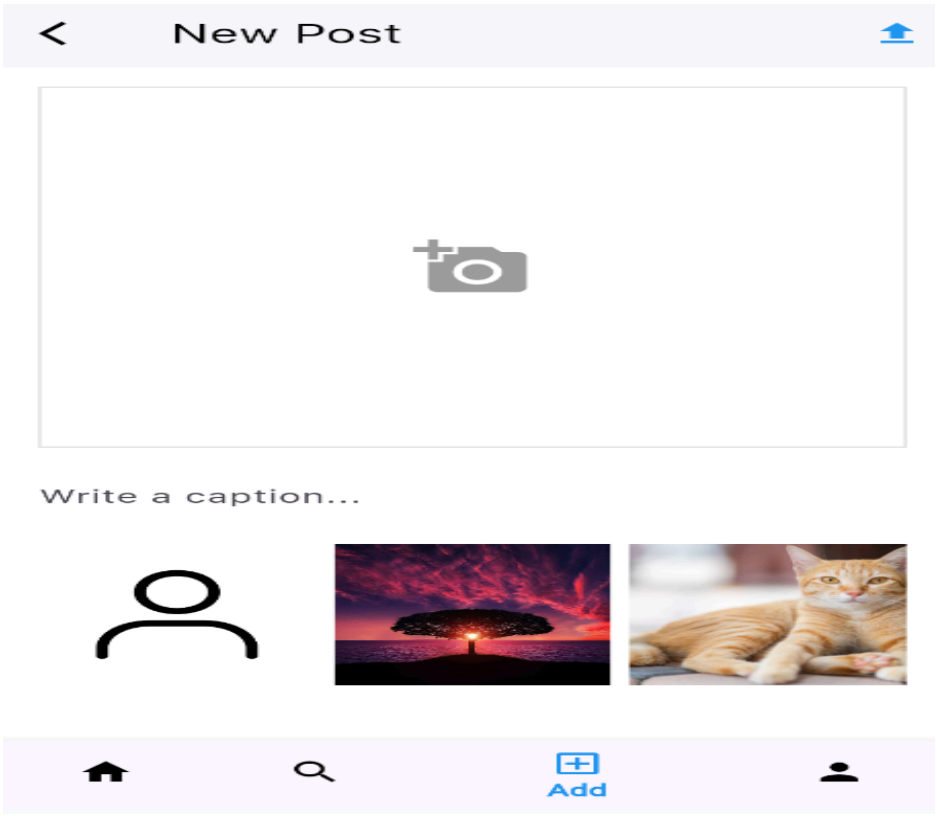
```
            Navigator.pop(context);
          },
        ),
        title: const Text('New Post',
            style: TextStyle(color: Colors.black, fontFamily: 'SFProText')),
        actions: [
          IconButton(
            onPressed: _isLoading ? null : _uploadPost,
            icon: _isLoading
                ? const CircularProgressIndicator()
                : const Icon(Icons.upload, color: Colors.blue),
          ),
        ],
      ),
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              GestureDetector(
                onTap: () {
                  showModalBottomSheet(
                    context: context,
                    builder: (BuildContext context) {
                      return Column(
                        mainAxisSize: MainAxisSize.min,
                        children: <Widget>[
                          ListTile(
                            leading: const Icon(Icons.photo_library),
                            title: const Text('Photo Library'),
                            onTap: () {
                              _pickImage(ImageSource.gallery);
                              Navigator.pop(context);
                            },
                          ),
                          ListTile(
                            leading: const Icon(Icons.camera_alt),
                            title: const Text('Camera'),
                            onTap: () {
                              _pickImage(ImageSource.camera);
                              Navigator.pop(context);
                            },
                          ),
                        ],
```

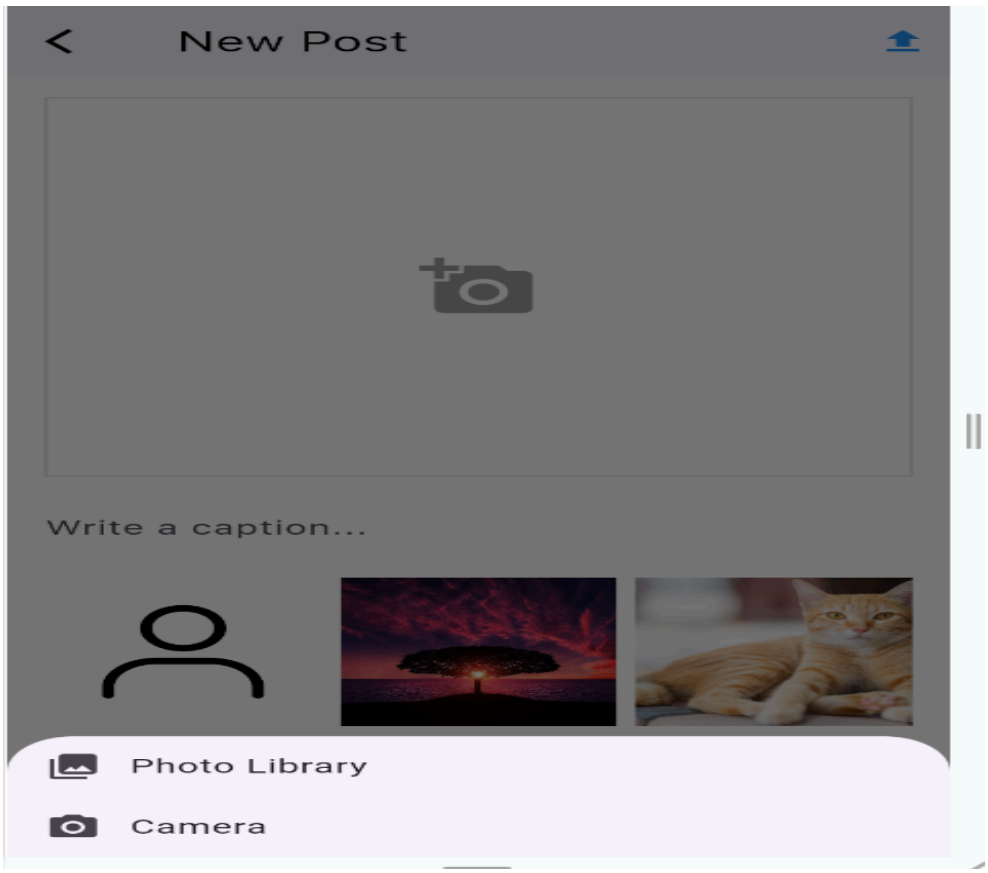
```
                );
              },
            );
          },
          child: Container(
            height: 300,
            width: double.infinity,
            decoration: BoxDecoration(
              border: Border.all(color: Colors.grey.shade300),
            ),
            child: _image != null
                ? Image.file(_image!, fit: BoxFit.cover)
                : const Center(
                    child: Icon(Icons.add_a_photo,
                        size: 50, color: Colors.grey)),
          ),
        ),
        const SizedBox(height: 16),
        TextField(
          controller: _captionController,
          style: const TextStyle(fontFamily: 'SFProText'),
          decoration: const InputDecoration(
            hintText: 'Write a caption...',
            hintStyle: TextStyle(fontFamily: 'SFProText'),
            border: InputBorder.none,
          ),
          maxLines: null,
        ),
        const SizedBox(height: 16),
        GridView.builder(
          shrinkWrap: true,
          physics: const NeverScrollableScrollPhysics(),
          gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
            crossAxisCount: 3,
            crossAxisSpacing: 8,
            mainAxisSpacing: 8,
          ),
          itemCount: _previousImageAssets.length,
          itemBuilder: (context, index) {
            return Image.asset(
              _previousImageAssets[index],
              fit: BoxFit.cover,
            );
          },
        ),
```

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

OUTPUT:-




New Post
Write a caption...
Add


New Post
Write a caption...
Photo Library
Camera

MAD & PWA Lab

Journal

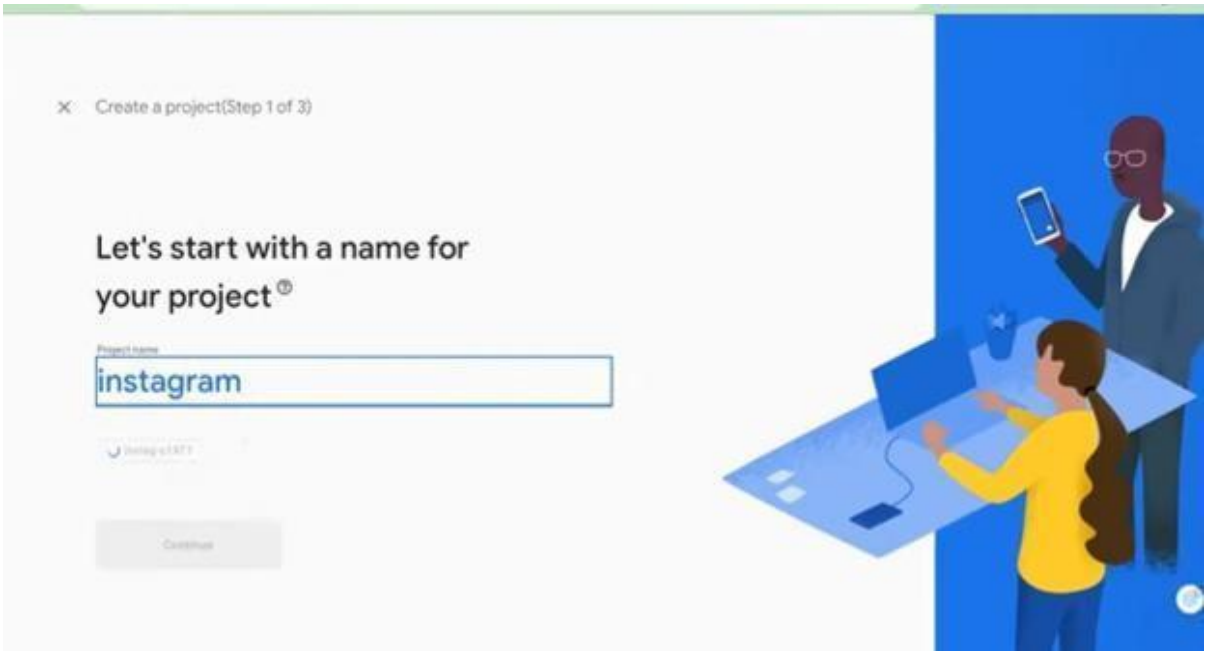
Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	29
Name	Sanket Satish More
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

MPL EXPERIMENT NO:- 6

Name:-Sanket More D15A Roll-no:-29

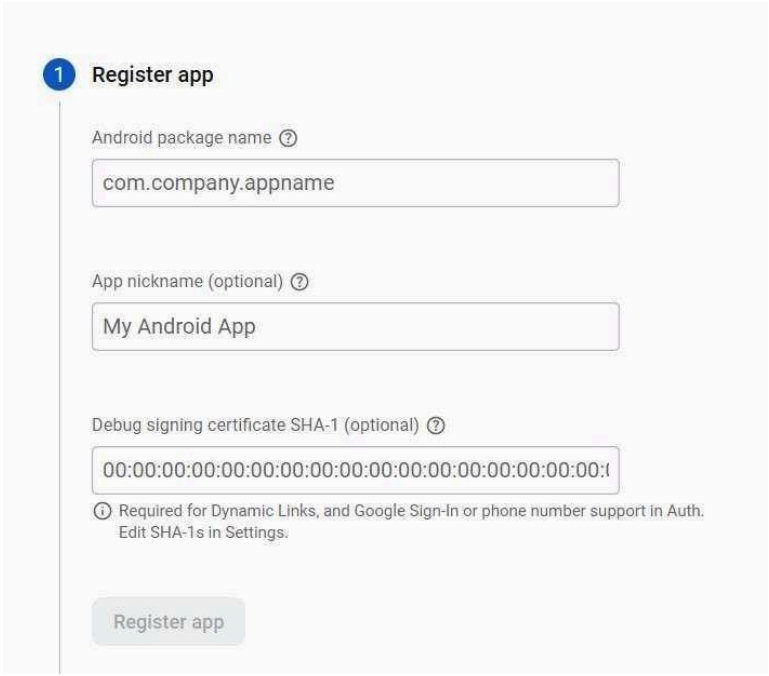
Aim:- To Connect flutter UI with firebase database

Creating a New Firebase Project



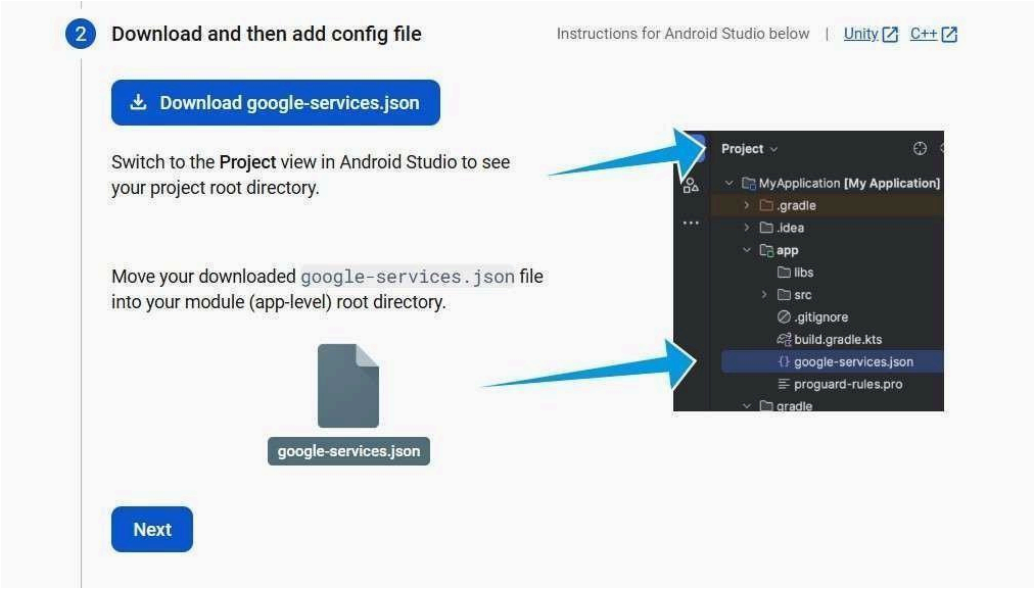
First, log in with your Google account to manage your Firebase projects. From within the Firebase dashboard, select the Create new project button and give it a name

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

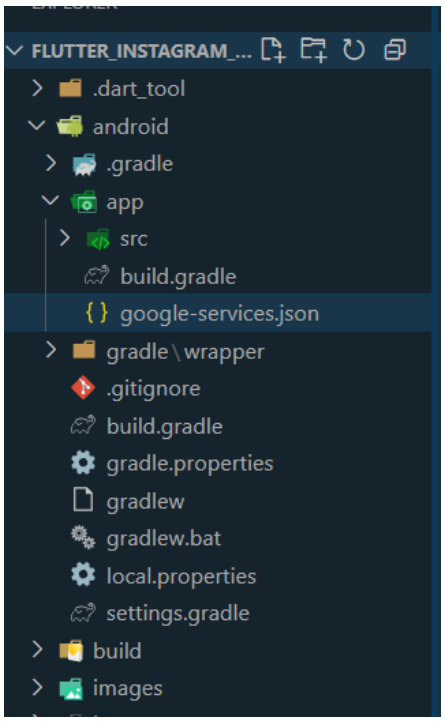


The most important thing here is to match up the Android package name that you choose here with the one inside of our application.

Then download the google-services.json file, that you will get.



put that file in the android folder (root level)



then select the build.gradle.kts (Kotlin DSL) part, and then follow the rest instructions

3 Add Firebase SDK

Instructions for Gradle | Unity | C++

Are you still using the `buildscript` syntax to manage plugins? Learn how to add Firebase plugins using that syntax.

1. To make the `google-services.json` config values accessible to Firebase SDKs, you need the Google services Gradle plugin.

(•) Kotlin DSL (build.gradle.kts) ( ) Groovy (build.gradle)

Add the plugin as a dependency to your **project-level** `build.gradle.kts` file:

Root-level (project-level) Gradle file (`<project>/build.gradle.kts`):

```
plugins {
  // ...

  // Add the dependency for the Google services Gradle plugin
  id("com.google.gms.google-services") version "4.4.2" apply false
}
```

2. Then, in your **module (app-level)** `build.gradle.kts` file, add both the `google-services` plugin and any Firebase SDKs that you want to use in your app:

Module (app-level) Gradle file (`<project>/<app-module>/build.gradle.kts`):

```
plugins {
  id("com.android.application")
  // Add the Google services Gradle plugin
  id("com.google.gms.google-services")
  ...
  }

dependencies {
  // Import the Firebase BoM
  implementation(platform("com.google.firebase:firebase-bom:33.9.0"))

  // TODO: Add the dependencies for Firebase products you want to use
  // When using the BoM, don't specify versions in Firebase dependencies
  // https://firebase.google.com/docs/android/setup#available-libraries
}
```

By using the Firebase Android BoM, your app will always use compatible Firebase library versions. Learn more

4 Next steps

You're all set!

Make sure to check out the documentation to learn how to get started with each Firebase product that you want to use in your app.

You can also explore sample Firebase apps.

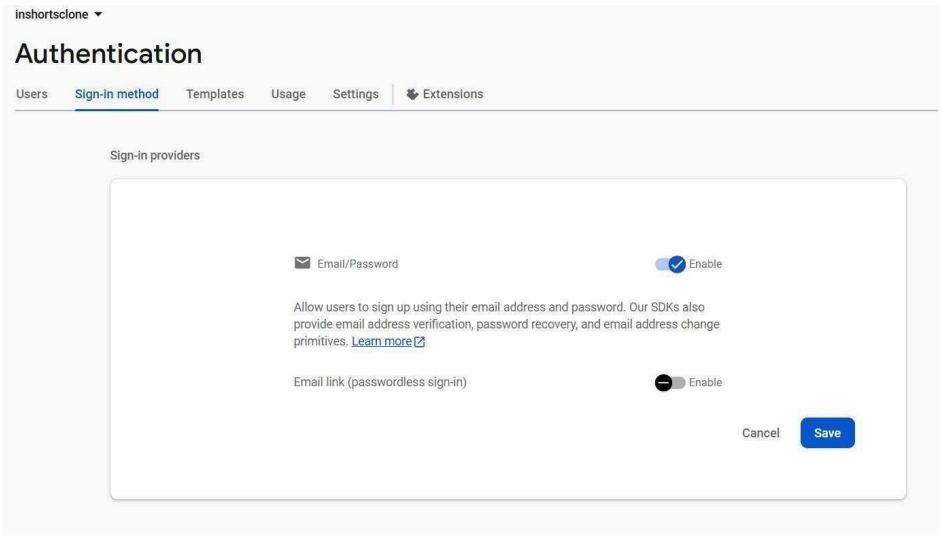
Or, continue to the console to explore Firebase.

Previous **Continue to console**

Generate the firebase_options.dart file, based on the google-services.json file

```dart
class DefaultFirebaseOptions {

  static const FirebaseOptions web = FirebaseOptions(
    apiKey: 'AIzaSyDvhua8saPw1WH4VAdePLF7AoR7RwhhZmA',
    appId: '1:167400934834:web:e3046e038154729c1e68ab',
    messagingSenderId: '167400934834',
    projectId: 'instagram-7555e',
    authDomain: 'instagram-7555e.firebaseapp.com',
    storageBucket: 'instagram-7555e.appspot.com',
  );

  static const FirebaseOptions android = FirebaseOptions(
    apiKey: 'AIzaSyC01Z6y2-6nrBl7HXyxFbqH2_292jEyo4g',
    appId: '1:167400934834:android:9f1c6eb6431a5d481e68ab',
    messagingSenderId: '167400934834',
    projectId: 'instagram-7555e',
    storageBucket: 'instagram-7555e.appspot.com',
  );

  static const FirebaseOptions ios = FirebaseOptions(
    apiKey: 'AIzaSyBlPS99WXA_v15pkf1OGHkT2t2BKqptnp4',
    appId: '1:167400934834:ios:b9d046db16efc3a91e68ab',
    messagingSenderId: '167400934834',
    projectId: 'instagram-7555e',
    storageBucket: 'instagram-7555e.appspot.com',
    iosBundleId: 'com.example.flutterInstagramClone',
```

In your part select the sign-in method and enable it.



Code:- Authenction_logic

```
import 'dart:io';

import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter_instagram_clone/data/firebase_service/firestor.dart'; import
'package:flutter_instagram_clone/data/firebase_service/storage.dart';

import 'package:flutter_instagram_clone/util/exeption.dart';

class Authentication {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  Future<void> Login({
    required String email,
    required String password,
  }) async { try {
      await _auth.signInWithEmailAndPassword( email:
          email.trim(), password: password.trim());
    } on FirebaseException catch (e) {
      throw exceptions(e.message.toString());
    }
  }

  Future<void> Signup({ required
    String email, required String
    password,
    required String passwordConfirme,
    required String username, required
```

```
  String bio,

  required File profile,
}) async { String
  URL;
```

```
try {
  if (email.isNotEmpty && password.isNotEmpty
      && username.isNotEmpty &&
      bio.isNotEmpty) {
    if (password == passwordConfirme) {
      // create user with email and password
      await _auth.createUserWithEmailAndPassword( email:
        email.trim(),
        password: password.trim(),
      );
      // upload profile image on storage

      if (profile != File('')) { URL
        =
          await StorageMethod().uploadImageToStorage('Profile', profile);
      } else { URL =
        '';
      }

      // get information with firestor

      await Firebase_Firestor().CreateUser(
        email: email,
        username: username,
        bio: bio,
        profile: URL == ''
            ? 'https://firebasestorage.googleapis.com/v0/b/instagram-
8a227.appspot.com/o/person.png?alt=media&token=c6fcbe9d-f502-4aa1-8b4b-ec37339e78ab'
            : URL,
      );
    } else {
```

```
            throw exceptions('password and confirm password should be same');
          }
        } else {
          throw exceptions('enter all the fields');
        }
      } on FirebaseException catch (e) {
        throw exceptions(e.message.toString());
      }
    }
  }
```

Login_screen

```
import 'dart:io';

import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter_instagram_clone/data/firebase_service/firestor.dart'; import
'package:flutter_instagram_clone/data/firebase_service/storage.dart';
import 'package:flutter_instagram_clone/util/exeption.dart'; class

Authentication {

  final FirebaseAuth _auth = FirebaseAuth.instance;
  Future<void> Login({
    required String email,
    required String password,
  }) async { try {
      await _auth.signInWithEmailAndPassword( email:
          email.trim(), password: password.trim());
    } on FirebaseException catch (e) {
      throw exceptions(e.message.toString());
    }
  }

  Future<void> Signup({ required
    String email, required String
    password,
    required String passwordConfirme,
```

required String username, *required* String bio,

```
    required File profile,
  }) async { String
    URL; try {
     if (email.isNotEmpty && password.isNotEmpty
         && username.isNotEmpty &&
         bio.isNotEmpty) {
       if (password == passwordConfirme) {
         // create user with email and password
         await _auth.createUserWithEmailAndPassword( email:
           email.trim(),
           password: password.trim(),
         );
         // upload profile image on storage

         if (profile != File('')) { URL =
             await StorageMethod().uploadImageToStorage('Profile', profile);
         } else { URL =
           '';
         }

         // get information with firestor

         await Firebase_Firestor().CreateUser( email:
           email,
           username: username,
           bio: bio,
           profile: URL == ''
               ? 'https://firebasestorage.googleapis.com/v0/b/instagram-
8a227.appspot.com/o/person.png?alt=media&token=c6fcbe9d-f502-4aa1-8b4b-ec37339e78ab'
               : URL,
         );
       } else {
         throw exceptions('password and confirm password should be same');
       }
     } else {
       throw exceptions('enter all the fields');
     }
    } on FirebaseException catch (e) {
     throw exceptions(e.message.toString());
    }
  }
}
```

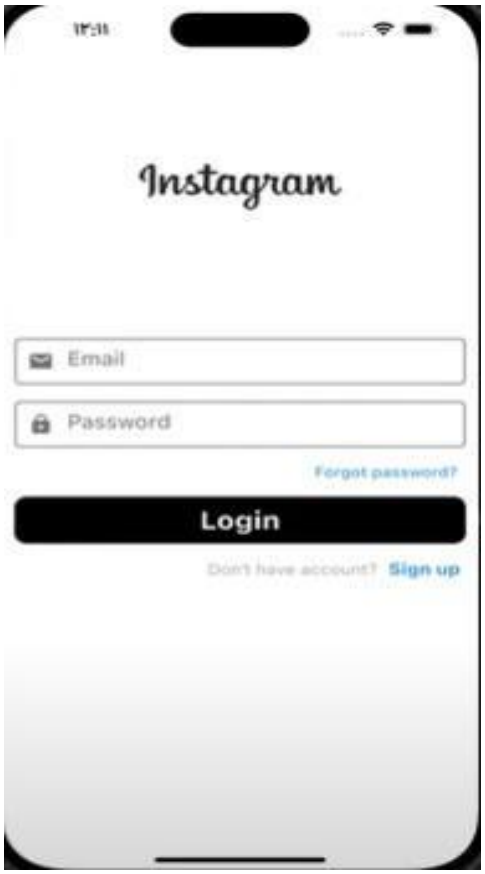
Home_Screen

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_instagram_clone/widgets/post_widget.dart'; import
'package:flutter_screenutil/flutter_screenutil.dart';

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

  @override
  State<HomeScreen> createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  FirebaseAuth _auth = FirebaseAuth.instance;
  FirebaseFirestore _firebaseFirestore = FirebaseFirestore.instance; @override
  Widget build(BuildContext context) { return
    Scaffold( resizeToAvoidBottomInset:
    false, backgroundColor: Colors.white,
    appBar: AppBar(
        centerTitle: true,
        elevation: 0, title:
        SizedBox( width:
        105.w, height:
        28.h,
          child: Image.asset('images/instagram.jpg'),
        ),
        leading: Image.asset('images/camera.jpg'), actions:
        [
          const Icon(
            Icons.favorite_border_outlined,
            color: Colors.black,
            size: 25,
          ),
          Image.asset('images/send.jpg'),
        ],
        backgroundColor: const Color(0xffFAFAFA),
      ),
      body: CustomScrollView(
        slivers: [
          StreamBuilder(
            stream: _firebaseFirestore
                .collection('posts')
                .orderBy('time', descending: true)
                .snapshots(),
            builder: (context, snapshot) { return
              SliverList(
                delegate: SliverChildBuilderDelegate( (context,
                  index) {
                    if (!snapshot.hasData) {
                      return Center(child: CircularProgressIndicator());
                    }
```

```
                return PostWidget(snapshot.data!.docs[index].data());
              },
              childCount:
                  snapshot.data == null ? 0 : snapshot.data!.docs.length,
            ),
          );
        },
      )
    ],
  ),
 );
 }
}
```

OUTPUT :


Instagram
Email
username
bio
Password
PasswordConfirme
Sign up
Don you have account? Login


Instagram
Email
Password
Forgot password?
Login
Don't have account? Sign up

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.	29
Name	Sanket Satish More
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

PWA Experiment -7

Sanket More
D15A 29

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

Theory:-

Regular Web Application

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

Progressive Web Application (PWA)

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

Key Differences Between PWAs and Regular Web Apps

1. Native-Like Experience

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

2. Ease of Access

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

3. Enhanced Performance

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

4. Improved User Engagement

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

5. Real-Time Updates

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

6. Search Engine Optimization (SEO) Benefits

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

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

Advantages and Limitations of PWAs

Advantages:

- **Progressive:** Compatible with all browsers and devices, following the principle of progressive enhancement.
- **Responsive:** Adapts to different screen sizes, including desktops, tablets, and smartphones.
- **App-Like Feel:** Mimics the experience of native applications in navigation and interaction.
- **Always Updated:** Service workers ensure real-time updates without requiring user intervention.
- **Secure:** Delivered over HTTPS, ensuring secure data transfer and protection against cyber threats.
- **SEO-Friendly:** Can be indexed by search engines, enhancing discoverability.
- **Re-Engagement:** Enables push notifications to encourage continued user interaction.

- **Installable:** Allows users to add the app to their home screen without app store downloads.
- **Offline Functionality:** Can function in low or no connectivity conditions using cached content.

Limitations:

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

CODE:

Index.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <!-- Favicon -->
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />

    <!-- Fonts -->
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@500&display=swap"
rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Satisfy&display=swap"
rel="stylesheet">

    <!-- FontAwesome Icons -->
    <script src="https://kit.fontawesome.com/25b9223bba.js"
crossorigin="anonymous"></script>

    <!-- PWA Manifest -->
    <link rel="manifest" href="/manifest.json">
```

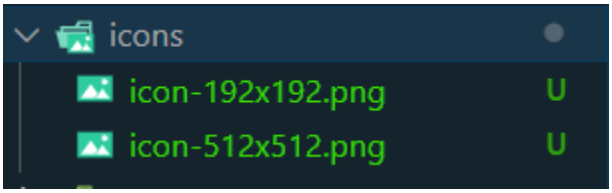
```
    <!-- Theme Color (for PWA UI) -->
    <meta name="theme-color" content="#ffffff">

    <title>React Recipe App</title>
  </head>
  <body>
    <div id="root"></div>

    <!-- Main Script -->
    <script type="module" src="/src/main.jsx"></script>

    <!-- Register Service Worker for PWA -->
    <script>
      if ('serviceWorker' in navigator) {
        window.addEventListener('load', () => {
          navigator.serviceWorker.register('/sw.js').then(registration => {
            console.log('Service Worker registered:', registration);
          }).catch(error => {
            console.log('Service Worker registration failed:', error);
          });
        });
      }
    </script>
  </body>
</html>
```

Manifest.json

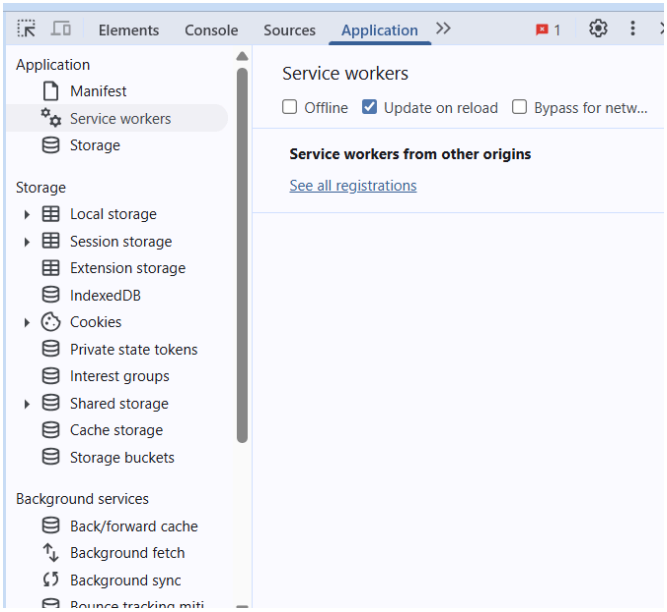
```
{
    "name": "React Recipe App",
    "short_name": "RecipeApp",
    "description": "A Progressive Web App for recipes!",
    "theme_color": "#ffffff",
    "background_color": "#ffffff",
    "display": "standalone",
    "start_url": "/",
    "icons": [
      {
        "src": "/icons/icon-192x192.png",
        "sizes": "192x192",
        "type": "image/png"
      },
      {
        "src": "/icons/icon-512x512.png",
```

```
        "sizes": "512x512",
        "type": "image/png"
      }
    ]
  }
```

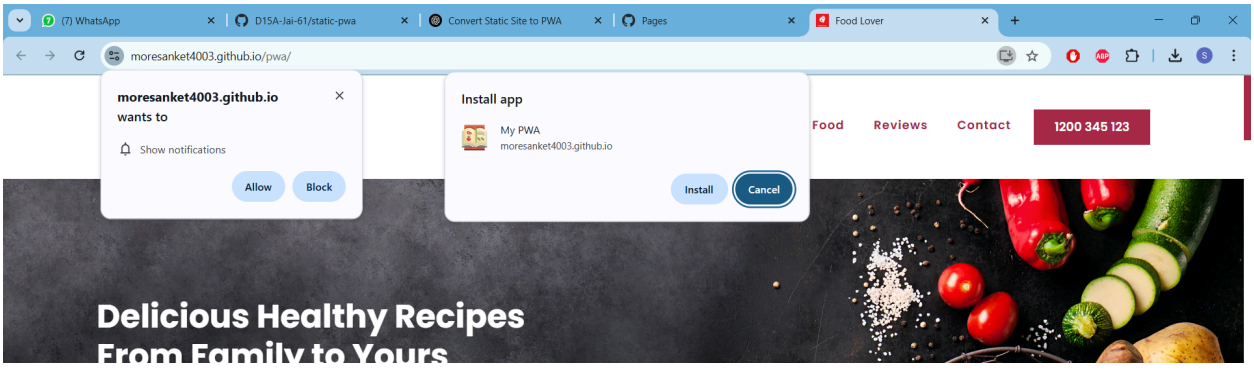
Icons

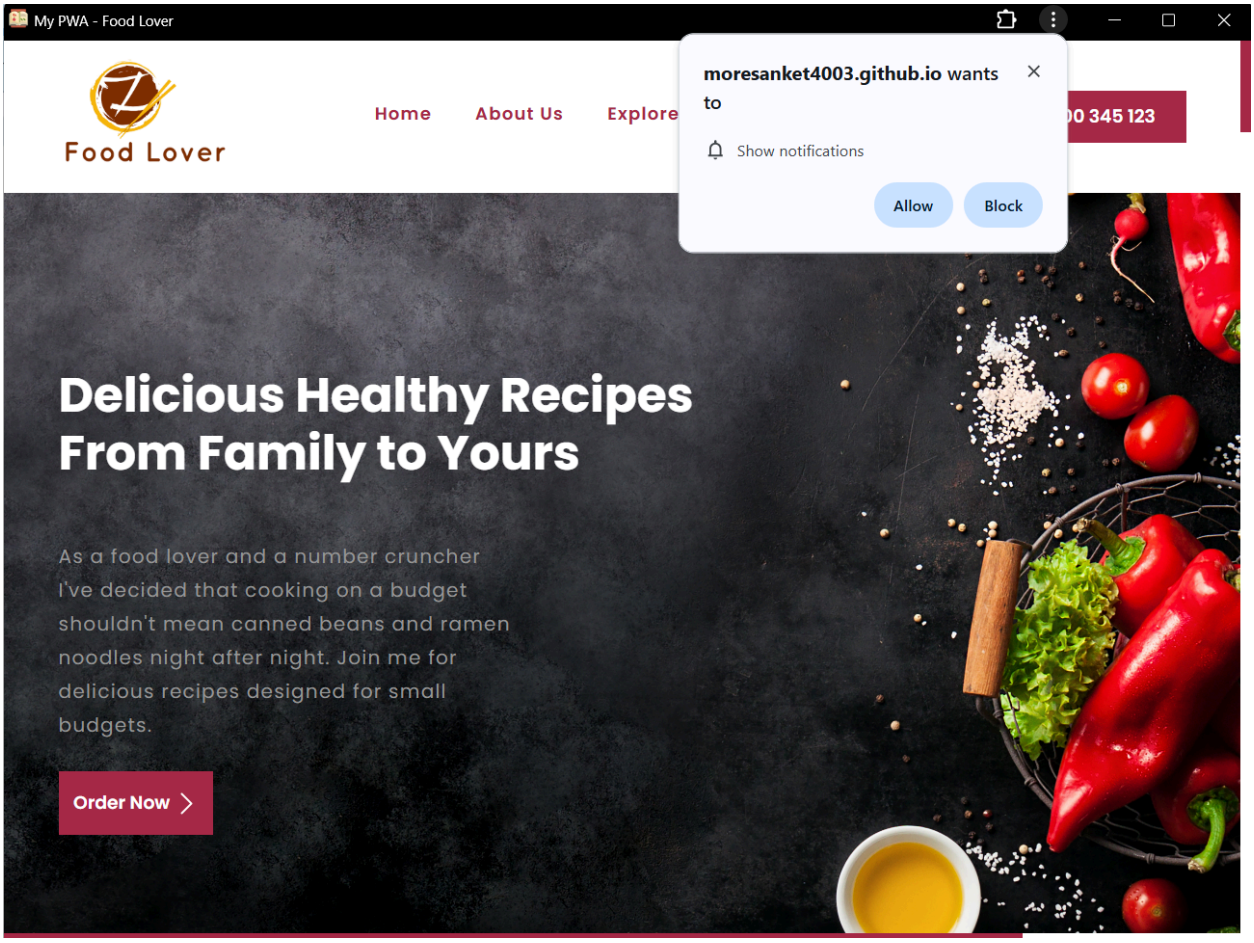


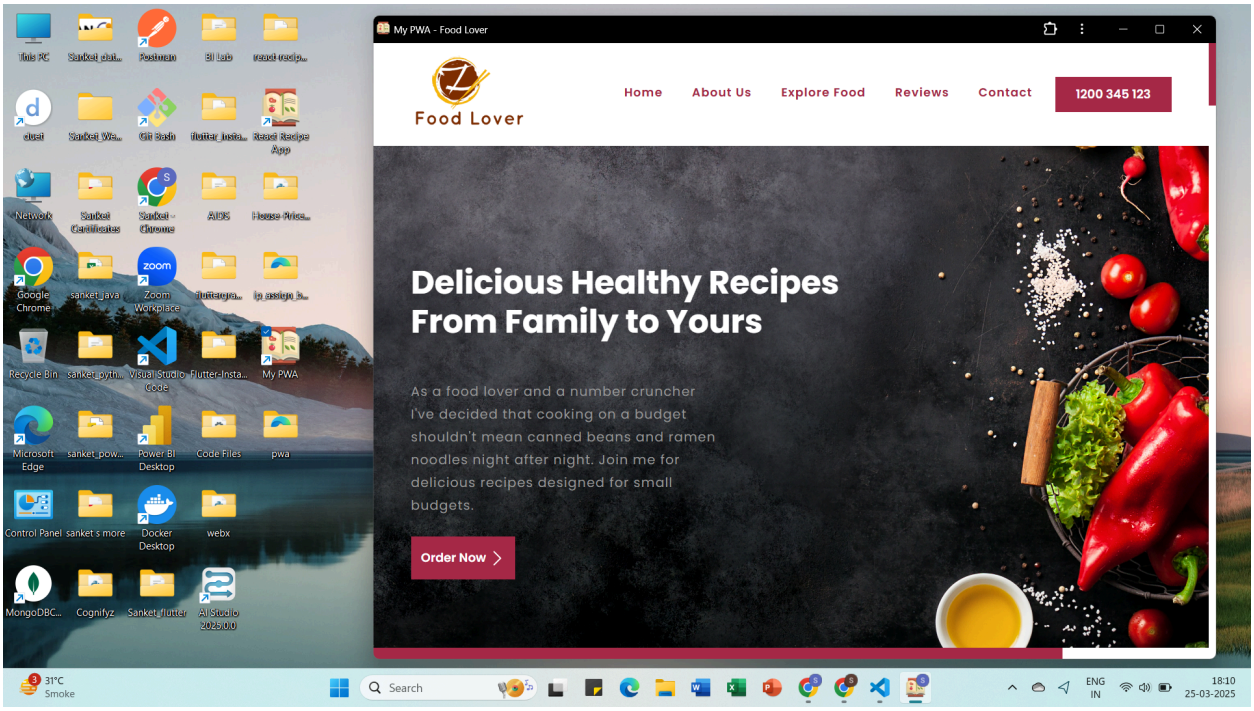
Google Dev



Output:-




My PWA - Food Lover
moresanket4003.github.io wants to
Show notifications
Allow
Block
Food Lover
Home
About Us
Explore
00 345 123
Delicious Healthy Recipes From Family to Yours
As a food lover and a number cruncher I've decided that cooking on a budget shouldn't mean canned beans and ramen noodles night after night. Join me for delicious recipes designed for small budgets.
Order Now


My PWA - Food Lover
Food Lover
Home
About Us
Explore Food
Reviews
Contact
1200 345 123
Delicious Healthy Recipes From Family to Yours
As a food lover and a number cruncher I've decided that cooking on a budget shouldn't mean canned beans and ramen noodles night after night. Join me for delicious recipes designed for small budgets.
Order Now

MAD & PWA Lab
Journal

Experiment No.	08
Experiment Title.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA
Roll No.	29
Name	Sanket Satish More
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

PWA Experiment -8

Sanket More
D15A 29

Theory:-

Understanding Service Workers

A Service Worker is a JavaScript file that runs independently in the background of a browser without direct user interaction. It acts as a proxy, handling network requests, managing push notifications, and enabling offline functionalities using the Cache API.

Key Characteristics of Service Workers:

- They act as a programmable network proxy, allowing control over network requests.
- Service workers function only over HTTPS due to security concerns, preventing potential "man-in-the-middle" attacks.
- They become idle when not in use and restart as needed. To persist data across restarts, IndexedDB can be used.

Capabilities of Service Workers:

- **Network Traffic Management:** Service workers can intercept and manipulate network requests. For example, when a request is made for a CSS file, the service worker can return a plain text response instead.
- **Caching Mechanism:** They store request-response pairs, making offline access possible.
- **Push Notifications:** They enable push notifications, enhancing user engagement.
- **Background Sync:** They allow processes to continue even when there is no active internet connection.

Limitations of Service Workers:

- **No Direct Access to the Window Object:** Service workers cannot manipulate the DOM directly. However, communication with the window is possible through the `postMessage` API.
- **Must Run on HTTPS:** Service workers function only over secure connections, except when being tested on localhost.

Lifecycle of a Service Worker

A service worker undergoes three main stages:

1. **Registration**
2. **Installation**
3. **Activation**

1. Registration

To initialize a service worker, it must be registered in the main JavaScript file. This process informs the browser about the service worker location and initiates its installation.

Example: Registering a Service Worker

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

This snippet first checks if the browser supports service workers. If it does, `navigator.serviceWorker.register()` is called, returning a promise. Upon successful registration, the scope of the service worker is logged.

Defining the Scope: The scope determines which files the service worker controls. By default, it covers all files in the same directory and its subdirectories.

Setting a Custom Scope:

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

In this case, the service worker controls all requests under `/app/`, including subdirectories.

If the service worker needs to cover higher-level directories, the `Service-Worker-Allowed` HTTP header must be configured on the server.

2. Installation

After registration, the browser attempts to install the service worker. If the site lacks an existing service worker or if modifications are detected in the new version, the installation process is triggered.

During installation, service workers can cache essential files to enable faster subsequent page loads.

Example: Installation Event Listener

```
self.addEventListener('install', function(event) {
  // Perform installation tasks like caching resources
});
```

3. Activation

Once installed, the service worker moves to the activation phase. If an older version of the service worker is already in use, the new one enters a waiting state until all instances of the old service worker close.

Example: Activation Event Listener

```
self.addEventListener('activate', function(event) {
  // Clean up outdated caches or perform setup tasks
});
```

Once activated, the new service worker takes control of pages within its scope, intercepting network requests and enabling offline functionalities. However, pages loaded before activation remain unaffected until they are refreshed or reopened.

To immediately apply the new service worker to all active pages, `clients.claim()` can be used.

```
self.addEventListener('activate', function(event) {
  event.waitUntil(clients.claim());
});
```

CODE:-

Changes Made in index.html

```
<body>
```

```
    <script>
        if ("serviceWorker" in navigator) {
          navigator.serviceWorker.register("sw.js")
            .then((reg) => {
```

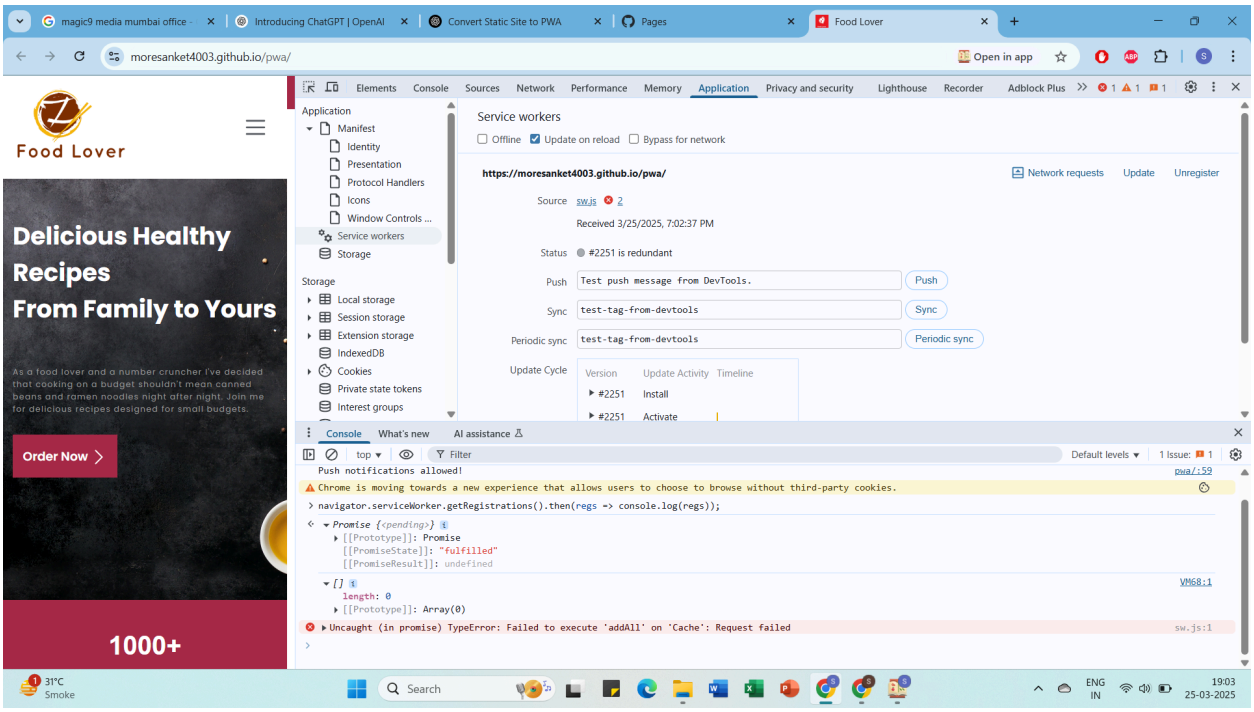
```
            console.log("Service Worker registered!", reg);

            // Background Sync
            if ("SyncManager" in window) {
              navigator.serviceWorker.ready.then((swReg) => {
                return swReg.sync.register("sync-data");
              });
            }

            // Push Notifications
            if ("PushManager" in window) {
              Notification.requestPermission().then((permission) => {
                if (permission === "granted") {
                  console.log("Push notifications allowed!");
                }
              });
            }
          })
          .catch((err) => console.log("Service Worker registration failed!", err));
      }
    </script>
```

sw.js

```
const CACHE_NAME = "pwa-cache-v1";
const urlsToCache = [
  "/index.html",
  "/assets/css/style.css",  // Update with actual CSS filename
  "/assets/images/icon-192x192.png",
  "/assets/images/icon-512x512.png"
];

// Install event: Caches assets
self.addEventListener("install", (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => {
      return cache.addAll(urlsToCache);
    })
  );
});

// Fetch event: Serve cached files when offline
self.addEventListener("fetch", (event) => {
  event.respondWith(
```

```
    caches.match(event.request).then((response) => {
      return response || fetch(event.request);
    })
  );
});

// Sync event: Background sync (requires registration in main script)
self.addEventListener("sync", (event) => {
  if (event.tag === "sync-data") {
    event.waitUntil(syncDataFunction());
  }
});

async function syncDataFunction() {
  console.log("Syncing data in the background...");
  // Example: Send stored requests to server when online
}

// Push event: Handle push notifications
self.addEventListener("push", (event) => {
  const options = {
    body: "New message received!",
    icon: "/assets/images/icon-192x192.png",
    badge: "/assets/images/icon-192x192.png"
  };
  event.waitUntil(
    self.registration.showNotification("PWA Notification", options)
  );
});

// Activate event: Clean up old caches
self.addEventListener("activate", (event) => {
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames
          .filter((cacheName) => cacheName !== CACHE_NAME)
          .map((cacheName) => caches.delete(cacheName))
      );
    })
  );
});
```

OUTPUT:-


Food Lover
moresanket4003.github.io/pwa/
Delicious Healthy Recipes From Family to Yours
As a food lover and a number cruncher I've decided that cooking on a budget shouldn't mean canned beans and ramen noodles night after night. Join me for delicious recipes designed for small budgets.
Order Now
1000+
Service workers
Offline
Update on reload
Bypass for network
https://moresanket4003.github.io/pwa/
Network requests
Update
Unregister
Source sw.js 2
Received 3/25/2025, 7:02:37 PM
Status #2251 is redundant
Push
Test push message from DevTools.
Sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#2251 Install
#2251 Activate
Push notifications allowed!
Chrome is moving towards a new experience that allows users to choose to browse without third-party cookies.
navigator.serviceWorker.getRegistrations().then(regs => console.log(regs));
Uncaught (in promise) TypeError: Failed to execute 'addAll' on 'Cache': Request failed
Search
19:03
25-03-2025

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.	29
Name	Sanket Satish More
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

PWA Experiment -9

Sanket More
D15A 29

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.

Push Event

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

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

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

CODE:-

Changes Made in Index.html

```
<body>

    <script>
        if ("serviceWorker" in navigator) {
          navigator.serviceWorker.register("sw.js")
```

```
            .then((reg) => {
              console.log("Service Worker registered!", reg);

              // Background Sync
              if ("SyncManager" in window) {
                navigator.serviceWorker.ready.then((swReg) => {
                  return swReg.sync.register("sync-data")
                    .then(() => console.log("Sync event registered"))
                    .catch((err) => console.log("Sync registration failed", err));
                });
              }

              // Push Notifications
              if ("PushManager" in window) {
                Notification.requestPermission().then((permission) => {
                  if (permission === "granted") {
                    console.log("Push notifications allowed!");
                  } else {
                    console.log("Push notifications denied!");
                  }
                });
              }
            })
            .catch((err) => console.log("Service Worker registration failed!",
err));
        }

        // Function to send a message to the Service Worker
        function sendMessageToServiceWorker(action) {
          if (navigator.serviceWorker.controller) {
            navigator.serviceWorker.controller.postMessage({ action: action });
          } else {
            console.log("Service Worker is not active yet.");
          }
        }

        // Detect clicks on "Contact Us" navigation link
        document.getElementById("contactUs")?.addEventListener("click", () => {
          sendMessageToServiceWorker("contact_us_nav_clicked");
        });

        // Detect clicks on "Contact Us" button
        document.getElementById("contactUsBtn")?.addEventListener("click", () => {
          sendMessageToServiceWorker("contact_us_btn_clicked");
        });
```

```
      </script>


    <!-- **** Header Section **** -->

    <header>
        <nav class="navbar navbar-expand-lg navigation-wrap">
            <div class="container">
                <a class="navbar-brand" href="#">
                    <img src="assets/images/logo.png" alt="" class="logo">
                </a>
                <button class="navbar-toggler" type="button"
data-bs-toggle="collapse" data-bs-target="#navbarNav"
                    aria-controls="navbarNav" aria-expanded="false"
aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="collapse navbar-collapse" id="navbarNav">
                    <ul class="navbar-nav ms-auto">
                        <li class="nav-item">
                            <a class="nav-link" href="#home">Home</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="#about">About Us</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="#explore-food">Explore
Food</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="#testinomial">Reviews</a>
                        </li>

                        <!-- Contact Us Nav Link -->
                        <li class="nav-item">
                            <a class="nav-link" href="#contact"
id="contactUs">Contact</a>
                        </li>

                        <!-- Contact Us Button -->
                        <li>
                            <button class="main-btn" id="contactUsBtn">1200 345
123</button>
                        </li>
                    </ul>
```

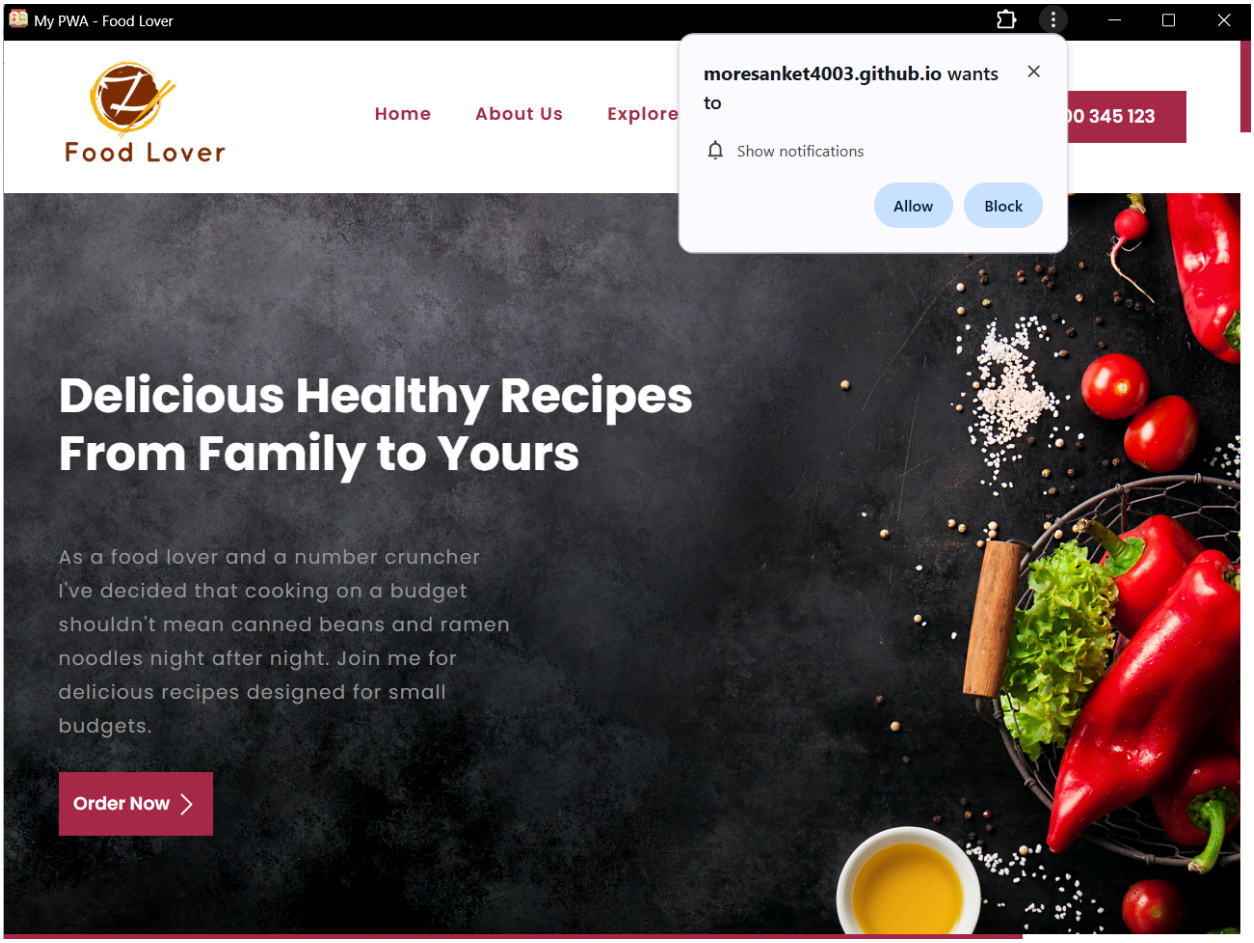
```
                </div>
            </div>
        </nav>
    </header>
```

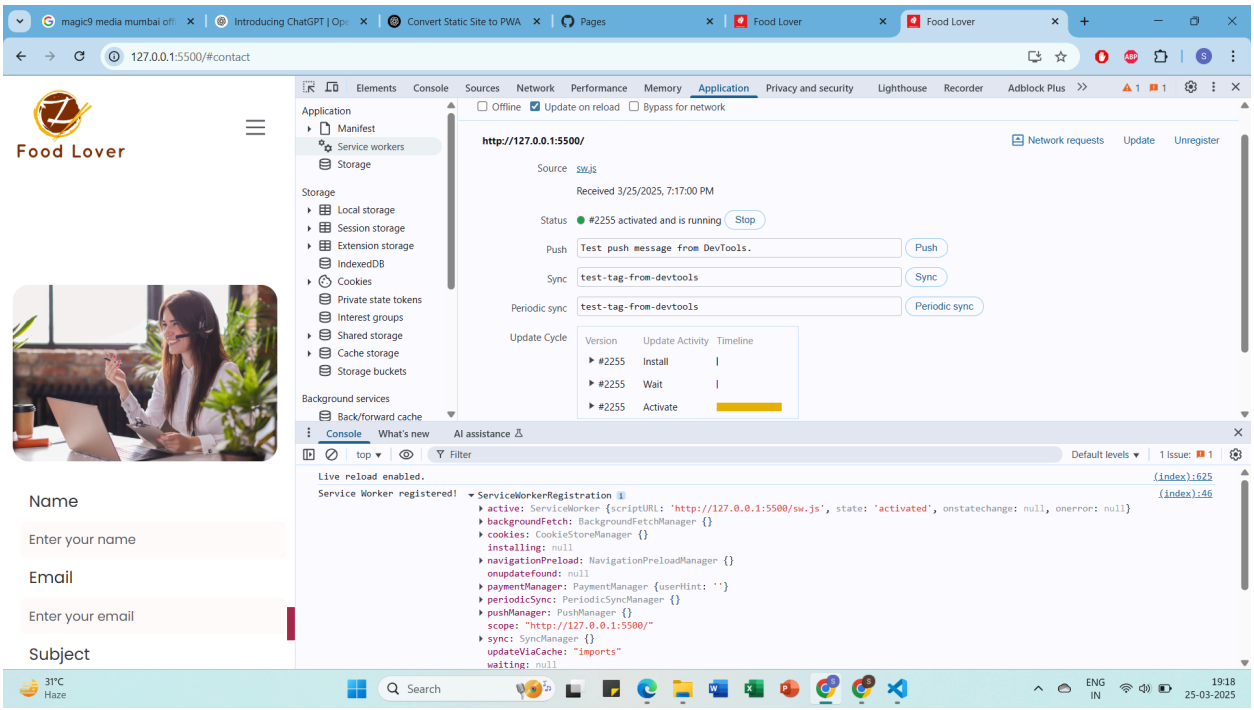
New sw.js

```
const CACHE_NAME = "pwa-cache-v1";
const urlsToCache = [
  "/index.html",
  "/assets/css/style.css",  // Update with actual CSS filename
  "/assets/images/icon-192x192.png",
  "/assets/images/icon-512x512.png"
];

// Install event: Caches assets
self.addEventListener("install", (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => {
      return cache.addAll(urlsToCache);
    })
  );
});

// Fetch event: Serve cached files when offline
self.addEventListener("fetch", (event) => {
  event.respondWith(
    caches.match(event.request).then((response) => {
      return response || fetch(event.request);
    })
  );
});

// Sync event: Background sync (requires registration in main script)
self.addEventListener("sync", (event) => {
  if (event.tag === "sync-data") {
    event.waitUntil(syncDataFunction());
  }
});

async function syncDataFunction() {
  console.log("Syncing data in the background...");
  // Example: Send stored requests to server when online
```

```
}

// Push event: Handle push notifications
self.addEventListener("push", (event) => {
  const options = {
    body: "New message received!",
    icon: "/assets/images/icon-192x192.png",
    badge: "/assets/images/icon-192x192.png"
  };
  event.waitUntil(
    self.registration.showNotification("PWA Notification", options)
  );
});

// Message event: Handle messages from the main thread
self.addEventListener("message", (event) => {
  if (event.data && event.data.action) {
    console.log(`Service Worker: User clicked on ${event.data.action}`);

    // Example: Show a notification when "Contact Us" is clicked
    if (event.data.action === "contact_us_nav_clicked" || event.data.action ===
"contact_us_btn_clicked") {
      self.registration.showNotification("Contact Us Clicked", {
        body: "User clicked the Contact Us button!",
        icon: "/assets/images/icon-192x192.png"
      });
    }
  }
});

// Activate event: Clean up old caches
self.addEventListener("activate", (event) => {
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames
          .filter((cacheName) => cacheName !== CACHE_NAME)
          .map((cacheName) => caches.delete(cacheName))
      );
    })
  );
});
```

OUTPUT:-


My PWA - Food Lover
Food Lover
Home
About Us
Explore
0 345 123
moresanket4003.github.io wants to
Show notifications
Allow
Block
Delicious Healthy Recipes From Family to Yours
As a food lover and a number cruncher I've decided that cooking on a budget shouldn't mean canned beans and ramen noodles night after night. Join me for delicious recipes designed for small budgets.
Order Now


127.0.0.1:5500/#contact
Food Lover
Name
Enter your name
Email
Enter your email
Subject
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
http://127.0.0.1:5500/
Network requests
Update
Unregister
Source sw.js
Received 3/25/2025, 7:17:00 PM
Status #2255 activated and is running Stop
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
#2255 Install
#2255 Wait
#2255 Activate
Console
What's new
AI assistance
Live reload enabled.
Service Worker registered!
ServiceWorkerRegistration
active: ServiceWorker {scriptURL: 'http://127.0.0.1:5500/sw.js', state: 'activated'}
backgroundFetch: BackgroundFetchManager {}
cookies: CookieStoreManager {}
installing: null
navigationPreload: NavigationPreloadManager {}
onupdatefound: null
paymentManager: PaymentManager {userHint: ''}
periodicSync: PeriodicSyncManager {}
pushManager: PushManager {}
scope: "http://127.0.0.1:5500/"
sync: SyncManager {}
updateViaCache: "imports"
waiting: null
31°C Haze
Search
ENG IN
19:18 25-03-2025

```
                    waiting: null
                  ▸ [[Prototype]]: ServiceWorkerRegistration
Sync event registered
Syncing data in the background...
```

```
Live reload enabled.
Service Worker registered! ▸ ServiceWorkerRegistration
Sync event registered
Syncing data in the background...
Push notifications allowed!
```

MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	29
Name	Sanket Satish More
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

PWA Experiment -10

Sanket More

D15A 29

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

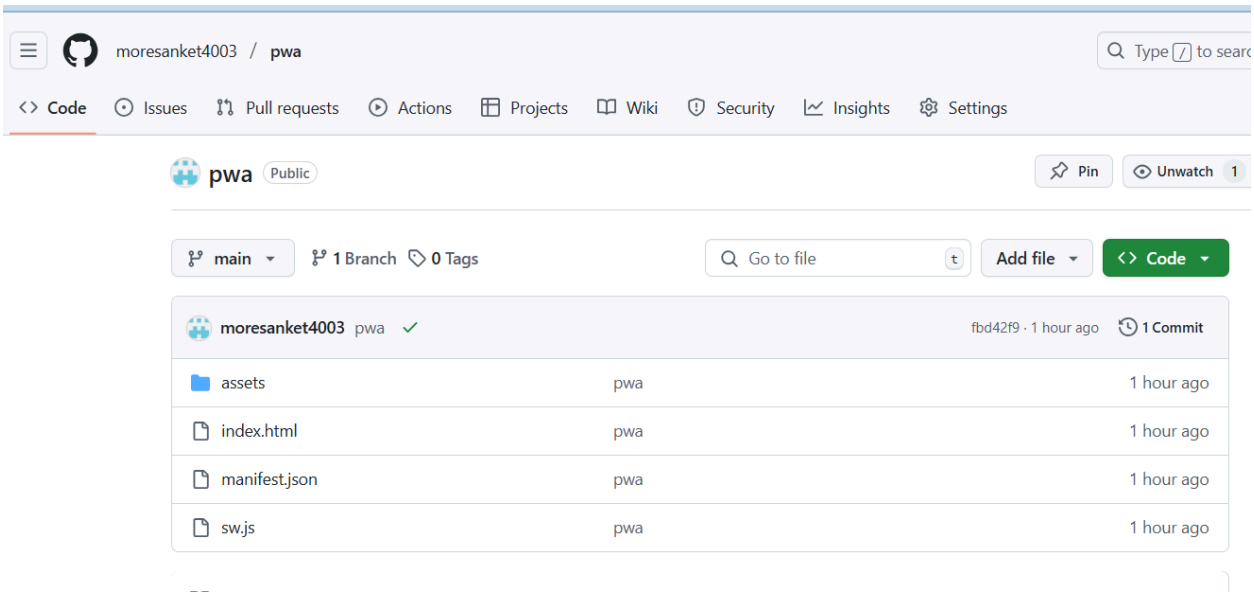
1. Very familiar interface if you are already using GitHub for your projects.
2. Easy to set up. Just push your static website to the gh-pages branch and your website is ready.
3. Supports Jekyll out of the box.
4. Supports custom domains. Just add a file called CNAME to the root of your site, add an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

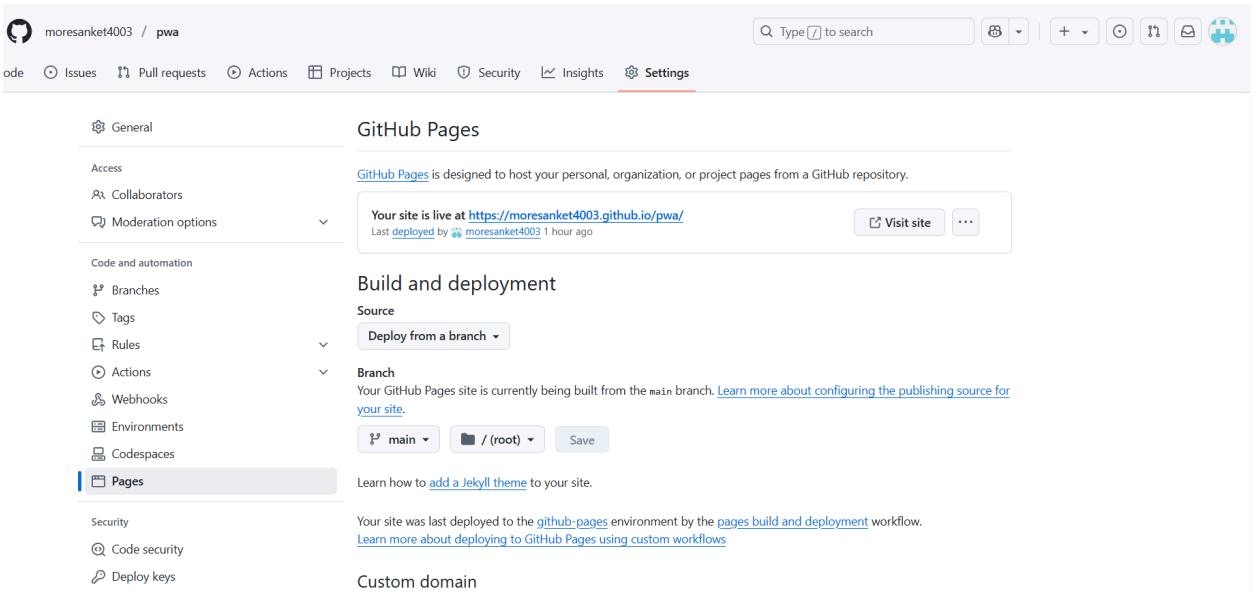
Cons

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.

Link to our GitHub repository:

OUTPUT:-


moresanket4003 / pwa
Type / to searc
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
pwa
Public
Pin
Unwatch 1
main
1 Branch
0 Tags
Go to file
t
Add file
Code
moresanket4003 pwa
fbd42f9 · 1 hour ago
1 Commit
assets pwa 1 hour ago
index.html pwa 1 hour ago
manifest.json pwa 1 hour ago
sw.js pwa 1 hour ago


moresanket4003 / pwa
Type / to search
ode
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
Security
Code security
Deploy keys
GitHub Pages
GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.
Your site is live at https://moresanket4003.github.io/pwa/
Last deployed by moresanket4003 1 hour ago
Visit site
Build and deployment
Source
Deploy from a branch
Branch
Your GitHub Pages site is currently being built from the main branch. Learn more about configuring the publishing source for your site.
main
/ (root)
Save
Learn how to add a Jekyll theme to your site.
Your site was last deployed to the github-pages environment by the pages build and deployment workflow.
Learn more about deploying to GitHub Pages using custom workflows
Custom domain

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	29
Name	Sanket Satish More
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

PWA Experiment 11

Sanket More
D15A 29

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

Theory :

Google Lighthouse :

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

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

Key Features and Audit Metrics

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

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.

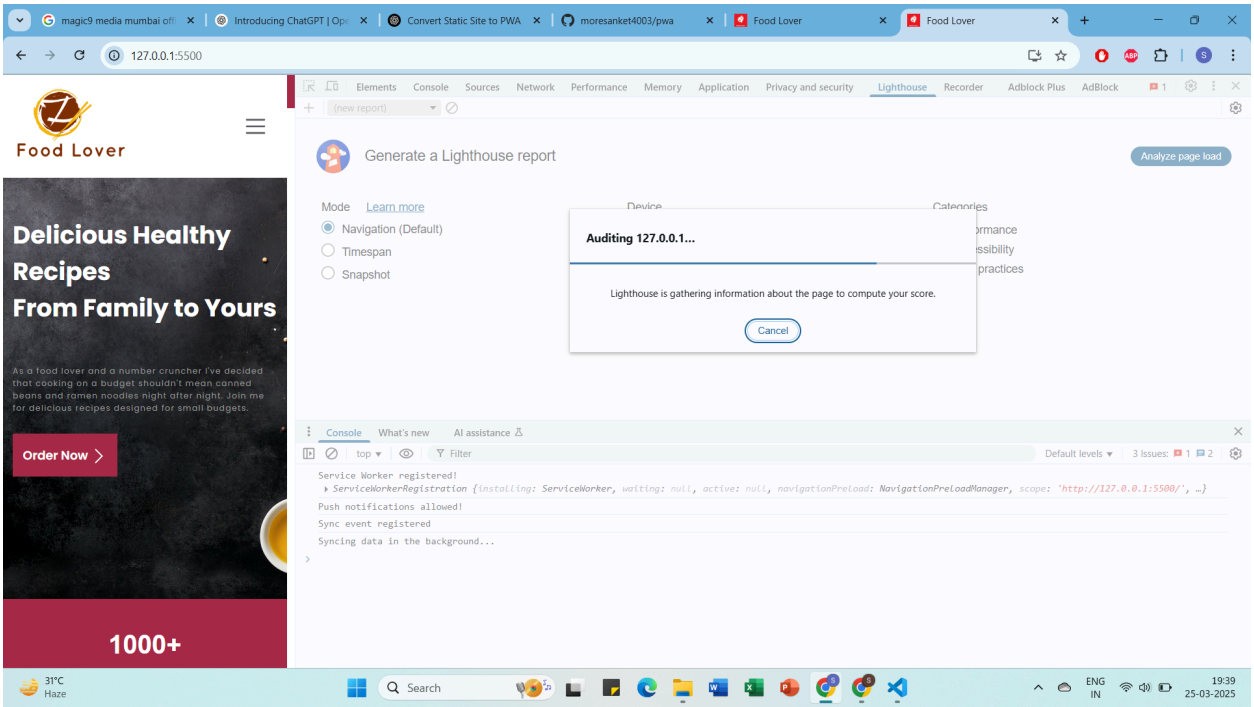
3. **Accessibility:** As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the ‘aria-’ attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics 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.

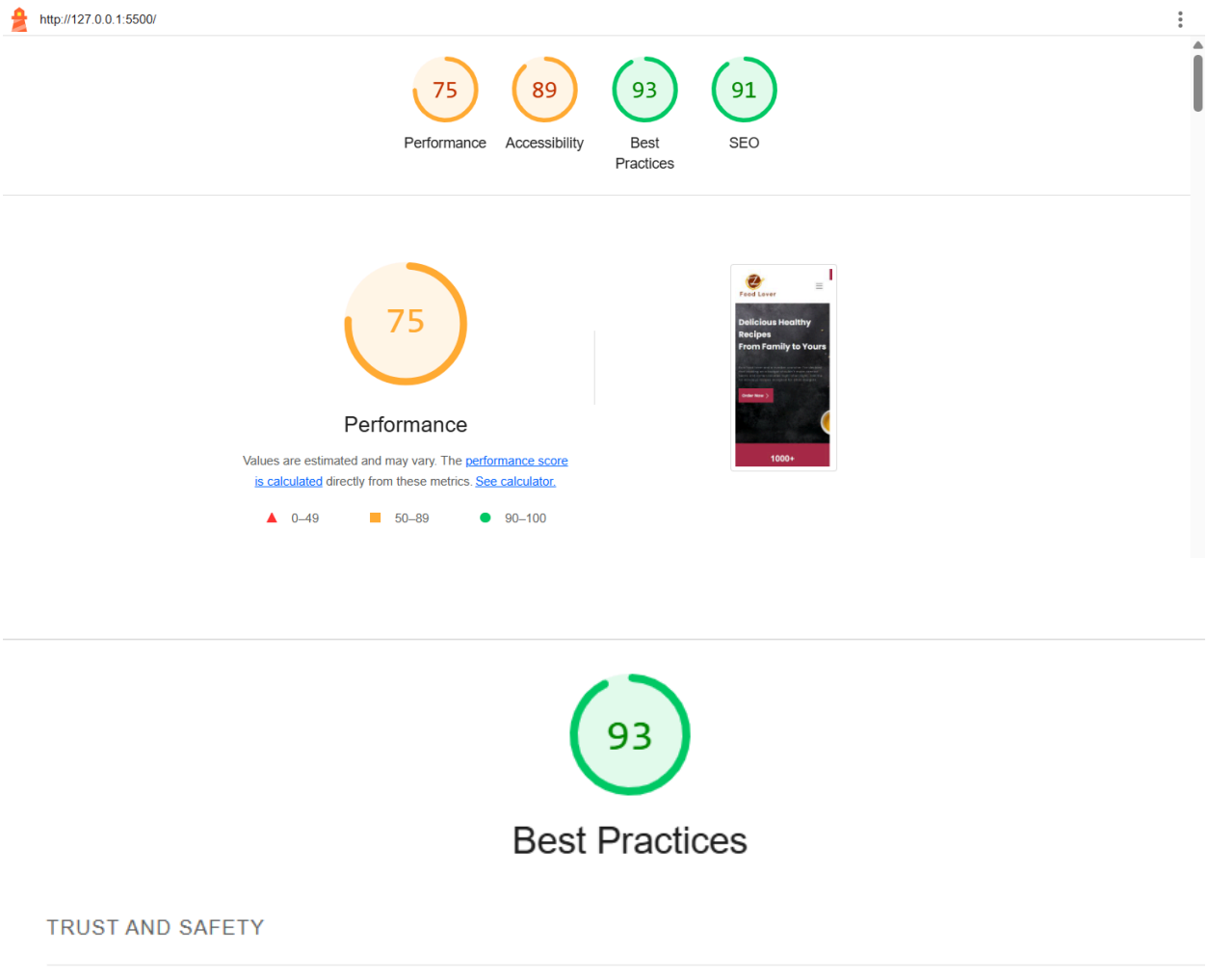
4. **Best Practices:** As any developer would know, there are a number of practices that have been deemed ‘best’ based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS

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.

OUTPUT:-





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

MAD & PWA Lab
Journal

Experiment No.	Assignment-1
Assignment 1 Questions	1. Flutter Overview:Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community. 2. Widget Tree and Composition: Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree. 3. State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable. 4. Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.
Roll No.	29
Name	Sanket Satish More
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

MAD & PWA Lab
Journal

Experiment No.	Assignment-2
Assignment 2 Questions	1. Define Progressive Web App (PWA) and explain its significance in modern web development.Discuss the key characteristics that differentiate PWAs from traditional mobile apps 2. Define responsive web design and explain its importance in the context of Progressive Web Apps.Compare and contrast responsive, fluid, and adaptive web design approaches. 3. Describe the lifecycle of Service Workers, including registration, installation, and activation phases. 4. Explain the use of IndexedDB in the Service Worker for data storage.
Roll No.	29
Name	Sanket Satish More
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4:Understand various PWA frameworks and their requirements LO5: Design and Develop a responsive User Interface by applying PWA Design techniques LO6:Develop and Analyze PWA Features and deploy it over app hosting solutions
Grade:	