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.

Since 1962

Department of Information Technology

CERTIFICATE

This is to certify that **Prachit P. Paralikar** 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 **2023-2024.**

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

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

Year/Sem/Class : D15A **A.Y.:** 23-24

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers : Mrs. Kajal Jewani.

Email : kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

Lab Objectives:

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

Lab Outcomes:

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

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

Index

Sr. No	Experiment Title	LO	DOP	DOS	Grade
1.	To install and configure the Flutter Environment	LO1	17/1/2024	24/1/24	15
2.	To design Flutter UI by including common widgets.	LO2	24/1/2024	31/1/2024	15
3.	To include icons, images, fonts in Flutter app	LO2	31/1/2024	7/2/2024	13
4.	To create an interactive Form using form widget	LO2	7/2/2024	14/2/2024	13
5.	To apply navigation, routing and gestures in Flutter App	LO2	14/2/2024	21/2/2024	13
6.	To Connect Flutter UI with fireBase database	LO3	21/2/2024	6/3/2024	13
7.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4	6/3/2024	13/3/2024	15
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	13/3/2024	20/3/2024	15
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5	20/3/2024	27/3/2024	15
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5	27/3/2024	01/04/2024	15
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6	27/3/2024	01/04/2024	15
12.	Assignment-1	LO1,LO2,LO3	28/01/2024	05/02/2024	5
13.	Assignment-2	LO4,LO5,LO6	14/03/2024	21/3	4

MAD & PWA Lab Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	43
Name	Prachit P. Paralikar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15

Name - Prachit Paralikar Class - D15A Roll No. - 43 EXP 1

Aim -

To install and configure the Flutter Environment

Code -

Installation

1) Installing the flutter from its official website

2) Clicking on windows installation and downloading the suitable zip that is compatible with the system
3) Extracting the zip and copying its bin path to configure it in environment variables

4) Running the “flutter doctor” command in command prompt to check the flutter installation and requirements

5) Installation of android studio

6) Installing the SDK packages in Andriod Studio

<u>HELLO Prachit CODE -</u>

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

```
void main() {
runApp(const MyApp());
}
class MyApp extends StatelessWidget {
const MyApp({Key? key}) : super(key: key);
@override
Widget build(BuildContext context) {
return MaterialApp(
title: 'Welcome to Flutter',
home: Scaffold(
appBar: AppBar(
title: const Text('Welcome to Flutter'),
),
body: const Center(
child: Text('Hello Shubham' ,style:
TextStyle(fontSize: 50)), ),
),
);
}
}
```

OUTPUT

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

MAD & PWA Lab Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	43
Name	Prachit P. Paralikar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

AIM:
To design Flutter UI by including common widgets.

THEORY:
Some common widgets used in Flutter are as follows:
MaterialApp:
Represents the overall structure of a Flutter application, providing settings for the app.

Container:
A box model allowing customization of size, padding, margin, and decoration for its child widget.

Column and Row:
Column: Arranges children vertically.
Row: Arranges children horizontally.

Text:
Displays text with specific styles using the TextStyle class.

Image:
Displays images from various sources, including network URLs or local assets.

ListView:
Displays a scrolling list of widgets, allowing users to scroll through content.

GestureDetector:
Captures user gestures like taps, double taps, and swipes, enabling interactive behavior.

Stack:
Overlaps widgets, allowing them to be positioned on top of each other.

AppBar:
Represents the app bar at the top of the screen, typically containing the app's title and actions.

TextField:
Allows users to input text, providing a UI for text entry.
SYNTAX:
Syntaxes for some of common widgets are as follows:
1.Text

```
Text(
“Text Content”,
style:TextStyle(
//style properties
)
)
```

2.Container

```
Container(
//style like
Width: 100,
Height: 100,
child://child widget to which container contains
)
```

3.TextButton

```
TextButton(
Event: action,
child: Text('Button placeholder'),
)
```

WIDGET AND PROPERTIES:

The widgets used in the following code are MaterialApp, Scaffold, Padding, Column, TextField, SizedBox, TextButton, Icon, Text, EdgeInsets.

- Scaffold is used to define the basic visual structure of the screen.
- SizedBox creates a box of a specified height.
- EdgeInsets provide padding between content and the box.

CODE:

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

import 'package:messenger_clone/helperfunctions/sharedpref_helper.dart';

import 'package:messenger_clone/services/auth.dart';

import 'package:messenger_clone/services/database.dart';

import 'package:messenger_clone/views/chatscreen.dart';

import 'package:messenger_clone/views/signin.dart';

import 'package:messenger_clone/views/stories_screen.dart'; // Import the Stories
```

screen

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



class Home extends StatefulWidget {

  @override

  _HomeState createState() => _HomeState();

}



class _HomeState extends State<Home> {

  bool isSearching = false;

  String? myName, myProfilePic, myUserName, myEmail;

  Stream? usersStream, chatRoomsStream;



  TextEditingController searchUsernameEditingController =

      TextEditingController();
```

```
getMyInfoFromSharedPreference() async {

  myName = await SharedPreferenceHelper().getDisplayName();

  myProfilePic = await SharedPreferenceHelper().getUserProfileUrl();

  myUserName = await SharedPreferenceHelper().getUserName();

  myEmail = await SharedPreferenceHelper().getUserEmail();

  setState(() {});

}



getChatRoomIdByUsernames(String a, String b) {

  if (a.substring(0, 1).codeUnitAt(0) > b.substring(0, 1).codeUnitAt(0)) {

    return "$b\_$a";

  } else {

    return "$a\_$b";

  }
```

```
  }



  onSearchBtnClick() async {

    isSearching = true;

    setState(() {});

    usersStream = await DatabaseMethods()

        .getUserByUserName(searchUsernameEditingController.text);



    setState(() {});

  }



  Widget chatRoomsList() {

    return StreamBuilder(

      stream: chatRoomsStream,

      builder: (context, snapshot) {
```

```
      return snapshot.hasData

        ? ListView.builder(

            itemCount: snapshot.data!.docs.length,

            shrinkWrap: true,

            itemBuilder: (context, index) {

              DocumentSnapshot ds = snapshot.data!.docs[index];

              return ChatRoomListTile(

                  ds["lastMessage"], ds.id, myUserName!);

            })

        : Center(child: CircularProgressIndicator());

    },

  );

}



Widget searchListUserTile(
```

```
    {required String profileUrl,

    required String name,

    required String username,

    required String email}) {

   return GestureDetector(

    onTap: () {

      var chatRoomId = getChatRoomIdByUsernames(myUserName!, username);

      Map<String, dynamic> chatRoomInfoMap = {

        "users": [myUserName, username]

      };

      DatabaseMethods().createChatRoom(chatRoomId, chatRoomInfoMap);

      Navigator.push(

          context,

          MaterialPageRoute(

              builder: (context) => ChatScreen(username, name)));
```

```
    },
    child: Container(
      margin: EdgeInsets.symmetric(vertical: 8),
      child: Row(
        children: [
          CircleAvatar(
            radius: 20,
            backgroundImage: NetworkImage(profileUrl),
          ),
          SizedBox(width: 12),
          Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [Text(name), Text(email)])
        ],
      ),
```

```
    ),

  );

}



Widget searchUsersList() {

  return StreamBuilder(

    stream: usersStream,

    builder: (context, snapshot) {

      return snapshot.hasData

          ? ListView.builder(

              itemCount: snapshot.data!.docs.length,

              shrinkWrap: true,

              itemBuilder: (context, index) {

                DocumentSnapshot ds = snapshot.data!.docs[index];

                return searchListUserTile(
```

```
                profileUrl: ds["imgUrl"],

                name: ds["name"],

                email: ds["email"],

                username: ds["username"]);

            },

          )

        : Center(

            child: CircularProgressIndicator(),

          );

    },

  );

}



getChatRooms() async {

  chatRoomsStream = await DatabaseMethods().getChatRooms();
```

```
  setState(() {});

}



onScreenLoaded() async {

  await getMyInfoFromSharedPreference();

  getChatRooms();

}



@override

void initState() {

  onScreenLoaded();

  super.initState();

}



@override
```

```
Widget build(BuildContext context) {

  return Scaffold(

    appBar: AppBar(

      title: Text("Chats"),

      actions: [

        InkWell(

          onTap: () {

            AuthMethods().signOut(context).then((s) {

              Navigator.pushReplacement(

                  context, MaterialPageRoute(builder: (context) => SignIn()));

            });

          },

          child: Padding(

            padding: const EdgeInsets.all(8.0),

            child: Icon(Icons.exit_to_app),
```

```
        ),

      )

    ],

  ),

  body: Container(

    margin: EdgeInsets.symmetric(horizontal: 20),

    child: Column(

      children: [

        Row(

          children: [

            isSearching

                ? GestureDetector(

                    onTap: () {

                      isSearching = false;

                      searchUsernameEditingController.text = "";
```

```
            setState(() {});

          },

          child: Padding(

            padding: EdgeInsets.only(right: 12),

            child: Icon(Icons.arrow_back)),

        )

      : Container(),

    Expanded(

      child: Container(

        margin: EdgeInsets.symmetric(vertical: 16),

        padding: EdgeInsets.symmetric(horizontal: 16),

        decoration: BoxDecoration(

          border: Border.all(

            color: Colors.grey,

            width: 1,
```

```
    ),
    borderRadius: BorderRadius.circular(24),
  ),
  child: Row(
    children: [
      Expanded(
        child: TextField(
          controller: searchUsernameEditingController,
          decoration: InputDecoration(
            border: InputBorder.none,
            hintText: "Search",
          ),
        ),
      ),
      GestureDetector(
```

```
                onTap: () {
                  if (searchUsernameEditingController.text != "") {
                    onSearchBtnClick();
                  }
                },
                child: Icon(Icons.search),
              ),
            ],
          ),
        ),
      ),
    ],
  ),
  isSearching ? searchUsersList() : chatRoomsList(),
],
```

```
    ),
  ),
  bottomNavigationBar: BottomAppBar(
    child: Row(
      mainAxisAlignment: MainAxisAlignment.spaceAround,
      children: [
        IconButton(
          icon: Icon(Icons.chat),
          onPressed: () {
            // Add functionality for chat
          },
        ),
        IconButton(
          icon: Icon(Icons.view_headline),
          onPressed: () {
```

```
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => StoriesScreen()),
            );
          },
        ),
      ],
    ),
  ),
);
}
}


class ChatRoomListTile extends StatefulWidget {
  final String lastMessage, chatRoomId, myUsername;
```

```
  ChatRoomListTile(this.lastMessage, this.chatRoomId, this.myUsername);




  @override

  _ChatRoomListTileState createState() => _ChatRoomListTileState();

}




class _ChatRoomListTileState extends State<ChatRoomListTile> {

  String profilePicUrl = "", name = "", username = "";




  getThisUserInfo() async {

    username =

        widget.chatRoomId.replaceAll(widget.myUsername, "").replaceAll("_", "");

    QuerySnapshot querySnapshot = await DatabaseMethods().getUserInfo(username);

    print(

        "something bla bla ${querySnapshot.docs[0].id} ${querySnapshot.docs[0]["name"]}
${querySnapshot.docs[0]["imgUrl"]}");
```

```
    name = "${querySnapshot.docs[0]["name"]}";

    profilePicUrl = "${querySnapshot.docs[0]["imgUrl"]}";

    setState(() {});

  }



  @override

  void initState() {

    getThisUserInfo();

    super.initState();

  }



  @override

  Widget build(BuildContext context) {

    return GestureDetector(

      onTap: () {
```

```
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => ChatScreen(username, name)));
    },
    child: Container(
      margin: EdgeInsets.symmetric(vertical: 8),
      child: Row(
        children: [
        CircleAvatar(
          radius: 20,
          backgroundImage: NetworkImage(profilePicUrl),
        ),
        SizedBox(width: 12),
        Column(
```

```
            crossAxisAlignment: CrossAxisAlignment.start,

            children: [

              Text(

                name,

                style: TextStyle(fontSize: 16),

              ),

              SizedBox(height: 3),

              Text(widget.lastMessage)

            ],

          )

        ],

      ),

  }

}
```

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

OUTPUT:

CONCLUSION:

Learned about basic widgets used in flutter.Learned about syntaxes for the same.

MAD & PWA Lab Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	43
Name	Prachit P. Paralikar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	13

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

THEORY:

1. Icons:

Flutter provides the Icon widget for displaying material design icons. We can also use icon packages like font_awesome_flutter for additional icon options.

2.Images:

We can use the Image widget to display images in your Flutter app. Images can be loaded from assets, the internet, or other sources.

3.Fonts:

To use custom fonts in Flutter, you need to include the font files (usually .ttf or .otf) in your project and then reference them in your Flutter code.

SYNTAX:

1.Icons:

```
 actions: [
        IconButton(
          icon: Icon(Icons.more_vert),
          onPressed: () {
            // Handle more options
          },
        ),
      ],
    ),
```

2.Images:

```
 void _getImageAndUpload() async {
   final pickedFile = await _picker.pickImage(source: ImageSource.gallery);
   if (pickedFile != null) {
     // Valid file path obtained
     print('File Path: ${pickedFile.path}');

     // Check if the file exists
     File imageFile = File(pickedFile.path);
     if (await imageFile.exists()) {
       setState(() {
         _imageFile = imageFile;
       });

       String? downloadURL = await DatabaseMethods().uploadImage(_imageFile!);

       if (downloadURL != null) {
         // Image uploaded successfully
         print('Image uploaded. Download URL: $downloadURL');
```

```
        // Update the UI with the new story URL and timestamp
        setState(() {
          _stories.add(downloadURL);
          _storyTimestamps.add(DateTime.now());
        });
      } else {
        // Image upload failed
        print('Image upload failed.');
      }
    } else {
      // File does not exist
      print('Selected file does not exist.');
    }
  } else {
    // File picker returned null
    print('No file selected.');
  }
}
```

3.Fonts:

```
// Loading custom font
Text(
'Custom Font Text',
style: TextStyle(
fontFamily: 'Font Family',
fontSize: 16,
),
);
```

<u>WIDGETS AND PROPERTIES:</u>

The widgets we focussed in this lab are:

- Images- We added the X logo image on the page.
- Icons- We added a facebook icon on the page from the FontAwesomeIcons library along with person and lock icons for username and password field.
- Fonts

<u>CODE:</u>

```
import 'dart:io';

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

```
import 'package:image_picker/image_picker.dart';

import 'package:messenger_clone/services/database.dart';

import 'dart:convert';


class StoriesScreen extends StatefulWidget {

  @override

  _StoriesScreenState createState() => _StoriesScreenState();

}


class _StoriesScreenState extends State<StoriesScreen> {

  List<String> _stories = [];

  List<DateTime> _storyTimestamps = [];


  File? _imageFile;

  final ImagePicker _picker = ImagePicker();


  @override

  void initState() {

    super.initState();

    _loadStories();

  }
```

```
void _loadStories() async {
  // Implement logic to fetch stories from Firestore
  // For now, I'm just setting some placeholder URLs and timestamps
  setState(() {
    _stories = [
      'https://example.com/image1.jpg',
      'https://example.com/image2.jpg',
      // Add more story URLs here
    ];
    _storyTimestamps = [
      DateTime.now().subtract(Duration(hours: 3)), // Example timestamp
      DateTime.now().subtract(Duration(minutes: 30)), // Example timestamp
      // Add more timestamps here
    ];
  });
}

void _getImageAndUpload() async {
  final pickedFile = await _picker.pickImage(source: ImageSource.gallery);
  if (pickedFile != null) {
    // Valid file path obtained
    print('File Path: ${pickedFile.path}');
```

```
// Check if the file exists
File imageFile = File(pickedFile.path);
if (await imageFile.exists()) {
  setState(() {
    _imageFile = imageFile;
  });

  String? downloadURL = await DatabaseMethods().uploadImage(_imageFile!);

  if (downloadURL != null) {
    // Image uploaded successfully
    print('Image uploaded. Download URL: $downloadURL');
    // Update the UI with the new story URL and timestamp
    setState(() {
      _stories.add(downloadURL);
      _storyTimestamps.add(DateTime.now());
    });
  } else {
    // Image upload failed
    print('Image upload failed.');
  }
```

```
    } else {
      // File does not exist
      print('Selected file does not exist.');
    }
  } else {
    // File picker returned null
    print('No file selected.');
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(_storyTimestamps.isNotEmpty
          ? 'Stories (${_storyTimestamps.length}) -
${_storyTimestamps.first.hour}:${_storyTimestamps.first.minute}'
          : 'Stories'),
    ),
    body: _stories.isEmpty
        ? Center(
            child: Text('No stories available'),
          )
```

```
: ListView.builder(
    itemCount: _stories.length,
    scrollDirection: Axis.horizontal,
    itemBuilder: (context, index) {
      return Padding(
        padding: EdgeInsets.symmetric(horizontal: 8.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          mainAxisAlignment: MainAxisAlignment.start,
          children: [
            Container(
              width: 100,
              height: 180,
              decoration: BoxDecoration(
                borderRadius: BorderRadius.circular(8.0),
                image: DecorationImage(
                  image: NetworkImage(_stories[index]),
                  fit: BoxFit.cover,
                ),
              ),
            ),
            SizedBox(height: 8.0),
```

```
                // Show username only when a story is available
                Text(
                  'My Story', // Display username or other info here
                  style: TextStyle(
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ],
            ),
          );
        },
      ),
    floatingActionButton: FloatingActionButton(
      onPressed: _getImageAndUpload,
      child: Icon(Icons.camera_alt),
    ),
  );
}
}
```

OUTPUT:

CONCLUSION:

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

1. Learnt about fonts, images and icons in flutter.
2. Imported a new library for icons and added it to the pubspec.yaml file. 3. Faced errors in adding image due to usage of assets syntax instead of network syntax while using image from browser.

MAD & PWA Lab Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	43
Name	Prachit P. Paralikar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	13

Prachit Paralikar
D15A_ 43

EXPT 4

AIM:
To create an interactive Form using form widget

THEORY:
Grouping Form Fields: The Form widget allows us to group multiple form fields together. This grouping is important for managing the state of the form and handling form submission.

State Management: The Form widget manages the state of the form fields within it. It automatically keeps track of the current value of each form field, their validation status, and any errors associated with them.

Validation: The Form widget provides built-in support for form field validation. We can specify validation logic for each form field, and the Form widget will automatically validate the form when it's submitted. Validation ensures that the user input meets certain criteria, such as required fields, valid email addresses, or password length.

Error Reporting: If a form field fails validation, the Form widget displays error messages associated with the invalid fields. These error messages are typically displayed below the corresponding form fields to inform us about what went wrong.

Submission Handling: When the user submits the form, the Form widget can trigger a callback function, allowing us to handle form submission. This callback function can perform actions such as sending data to a server, saving data locally, or navigating to another screen.

GlobalKey: To access the state of a Form widget, we typically use a GlobalKey<FormState>. This key allows us to interact with the Form widget programmatically, such as triggering form validation or resetting the form fields.

Nested Forms: We can nest Form widgets within each other to create complex forms with different sections or groups of fields. Each nested Form manages its own set of form fields independently.

SYNTAX:
```
Form(
  key: _formKey, // GlobalKey<FormState>
  child: Column(
    children: [
      // Form fields go here
    ],
  ),
)
```

WIDGET AND PROPERTIES:

In this experiment we have focussed on the form widget. We have used form widget for the change password page and have given different validation criteria for the submission of the form. Thus we have used properties of form widget like validator and decoration.

CODE:

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

import 'package:flutter/material.dart';

import 'package:google_sign_in/google_sign_in.dart';

import 'package:messenger_clone/helperfunctions/sharedpref_helper.dart';

import 'package:messenger_clone/services/database.dart';

import 'package:messenger_clone/views/home.dart';

import 'package:shared_preferences/shared_preferences.dart';



class AuthMethods {

  final FirebaseAuth auth = FirebaseAuth.instance;



  getCurrentUser() async {

    return await auth.currentUser;

  }



  Future<void> signInWithGoogle(BuildContext context) async {

    final FirebaseAuth _firebaseAuth = FirebaseAuth.instance;

    final GoogleSignIn _googleSignIn = GoogleSignIn();
```

```
try {
  final GoogleSignInAccount? googleSignInAccount =
      await _googleSignIn.signIn();

  if (googleSignInAccount != null) {
    final GoogleSignInAuthentication googleSignInAuthentication =
        await googleSignInAccount.authentication;

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

    try {
      final UserCredential result =
          await _firebaseAuth.signInWithCredential(credential);
      final User? userDetails = result.user;

      if (userDetails != null) {
        SharedPreferenceHelper().saveUserEmail(userDetails.email ??
```

```
"");

            SharedPreferenceHelper().saveUserId(userDetails.uid ?? "");

            SharedPreferenceHelper().saveUserName(

                userDetails?.email?.replaceAll("@gmail.com", "") ?? "");

            SharedPreferenceHelper()

                .saveDisplayName(userDetails.displayName ?? "");

            SharedPreferenceHelper()

                .saveUserProfileUrl(userDetails.photoURL ?? "");



            Map<String, dynamic> userInfoMap = {

              "email": userDetails.email,

              "username": userDetails.email!.replaceAll("@gmail.com", ""),

              "name": userDetails.displayName,

              "imgUrl": userDetails.photoURL

            };



            await DatabaseMethods().addUserInfoToDB(

                userDetails.uid!, userInfoMap); // Fixed this line

            Navigator.pushReplacement(

                context, MaterialPageRoute(builder: (context) => Home()));

          } else {
```

```
            print('User details are null');
          }
        } catch (e) {
          print('Error signing in with credential: $e');
        }
      } else {
        print('Google sign-in account is null');
      }
    } catch (e) {
      print('Error signing in with Google: $e');
    }
  }

  Future<void> signOut(BuildContext context) async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    prefs.clear();
    await auth.signOut();
    // Navigate to the sign-in screen or perform any other required action
  }
}
```

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

OUTPUT:

CONCLUSION:

Learnt about form widget and applied it in my project.

MAD & PWA Lab Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	43
Name	Prachit P. Paralikar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	13

Prachit Paralikar
D15A
43

MPL EXPT 5

AIM:
To apply navigation, routing and gestures in Flutter App

THEORY:
Navigation:
Movement between different screens or pages in a Flutter app.
Achieved using widgets like Navigator or MaterialApp. Triggered by user actions like tapping buttons or selecting items.

Gesture:
User's physical interaction with the touchscreen, like tapping or swiping.
Detected using widgets like GestureDetector.
Used to capture user input and trigger actions or events.

Routing:
Defines and manages navigation paths within the app. Done using Navigator widget and routes parameter of MaterialApp. Maps routes to screens or widgets, facilitating structured navigation.

SYNTAX:
Navigation:
```
Navigator.push(
  context,
  MaterialPageRoute(builder: (context) =>
NextScreen()), );
```

Routing:
```
MaterialApp(
  routes: {
    '/next': (context) => NextScreen(),
  },
  // Main app content
);
```

Gestures:
```
GestureDetector(
  onTap: () {
    // Handle tap gesture
  },
  child: Container(
```

```
    // Widget content
  ),
);
```

WIDGET AND PROPERTIES:
In this experiment we have focussed on the navigation, gesture and routing widgets and we are using various properties like onpressed, pushnamed, etc.

CODE:

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

import 'package:messenger_clone/helperfunctions/sharedpref_helper.dart';

import 'package:messenger_clone/services/auth.dart';

import 'package:messenger_clone/services/database.dart';

import 'package:messenger_clone/views/chatscreen.dart';

import 'package:messenger_clone/views/signin.dart';

import 'package:messenger_clone/views/stories_screen.dart'; // Import the Stories
screen

import 'package:cloud_firestore/cloud_firestore.dart';



class Home extends StatefulWidget {

  @override

  _HomeState createState() => _HomeState();

}



class _HomeState extends State<Home> {

  bool isSearching = false;

  String? myName, myProfilePic, myUserName, myEmail;

  Stream? usersStream, chatRoomsStream;
```

```
TextEditingController searchUsernameEditingController =
    TextEditingController();

getMyInfoFromSharedPreference() async {
  myName = await SharedPreferenceHelper().getDisplayName();
  myProfilePic = await SharedPreferenceHelper().getUserProfileUrl();
  myUserName = await SharedPreferenceHelper().getUserName();
  myEmail = await SharedPreferenceHelper().getUserEmail();
  setState(() {});
}

getChatRoomIdByUsernames(String a, String b) {
  if (a.substring(0, 1).codeUnitAt(0) > b.substring(0, 1).codeUnitAt(0)) {
    return "$b\_$a";
  } else {
    return "$a\_$b";
  }
}

onSearchBtnClick() async {
  isSearching = true;
  setState(() {});
```

```
  usersStream = await DatabaseMethods()
      .getUserByUserName(searchUsernameEditingController.text);


  setState(() {});
}


Widget chatRoomsList() {
  return StreamBuilder(
    stream: chatRoomsStream,
    builder: (context, snapshot) {
      return snapshot.hasData
          ? ListView.builder(
              itemCount: snapshot.data!.docs.length,
              shrinkWrap: true,
              itemBuilder: (context, index) {
                DocumentSnapshot ds = snapshot.data!.docs[index];
                return ChatRoomListTile(
                    ds["lastMessage"], ds.id, myUserName!);
              })
          : Center(child: CircularProgressIndicator());
    },
  );
}
```

```
Widget searchListUserTile(
    {required String profileUrl,
    required String name,
    required String username,
    required String email}) {
  return GestureDetector(
    onTap: () {
      var chatRoomId = getChatRoomIdByUsernames(myUserName!, username);
      Map<String, dynamic> chatRoomInfoMap = {
        "users": [myUserName, username]
      };
      DatabaseMethods().createChatRoom(chatRoomId, chatRoomInfoMap);
      Navigator.push(
          context,
          MaterialPageRoute(
              builder: (context) => ChatScreen(username, name)));
    },
    child: Container(
      margin: EdgeInsets.symmetric(vertical: 8),
      child: Row(
        children: [
          CircleAvatar(
```

```
            radius: 20,
            backgroundImage: NetworkImage(profileUrl),
          ),
          SizedBox(width: 12),
          Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [Text(name), Text(email)])
        ],
      ),
    ),
  );
}

Widget searchUsersList() {
  return StreamBuilder(
    stream: usersStream,
    builder: (context, snapshot) {
      return snapshot.hasData
          ? ListView.builder(
            itemCount: snapshot.data!.docs.length,
            shrinkWrap: true,
            itemBuilder: (context, index) {
              DocumentSnapshot ds = snapshot.data!.docs[index];
```

```
            return searchListUserTile(
              profileUrl: ds["imgUrl"],
              name: ds["name"],
              email: ds["email"],
              username: ds["username"]);
          },
        )
      : Center(
          child: CircularProgressIndicator(),
        );
    },
  );
}

getChatRooms() async {
  chatRoomsStream = await DatabaseMethods().getChatRooms();
  setState(() {});
}

onScreenLoaded() async {
  await getMyInfoFromSharedPreference();
  getChatRooms();
}
```

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

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("Chats"),
      actions: [
        InkWell(
          onTap: () {
            AuthMethods().signOut(context).then((s) {
              Navigator.pushReplacement(
                  context, MaterialPageRoute(builder: (context) => SignIn()));
            });
          },
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Icon(Icons.exit_to_app),
```

```
        ),
      )
    ],
  ),
  body: Container(
    margin: EdgeInsets.symmetric(horizontal: 20),
    child: Column(
      children: [
        Row(
          children: [
            isSearching
                ? GestureDetector(
                    onTap: () {
                      isSearching = false;
                      searchUsernameEditingController.text = "";
                      setState(() {});
                    },
                    child: Padding(
                        padding: EdgeInsets.only(right: 12),
                        child: Icon(Icons.arrow_back)),
                  )
                : Container(),
            Expanded(
```

```
child: Container(
  margin: EdgeInsets.symmetric(vertical: 16),
  padding: EdgeInsets.symmetric(horizontal: 16),
  decoration: BoxDecoration(
    border: Border.all(
      color: Colors.grey,
      width: 1,
    ),
    borderRadius: BorderRadius.circular(24),
  ),
  child: Row(
    children: [
      Expanded(
        child: TextField(
          controller: searchUsernameEditingController,
          decoration: InputDecoration(
            border: InputBorder.none,
            hintText: "Search",
          ),
        ),
      ),
      GestureDetector(
        onTap: () {
```

```
                    if (searchUsernameEditingController.text != "") {
                      onSearchBtnClick();
                    }
                  },
                  child: Icon(Icons.search),
                ),
              ],
            ),
          ),
        ),
      ],
    ),
    isSearching ? searchUsersList() : chatRoomsList(),
  ],
),
),
bottomNavigationBar: BottomAppBar(
  child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceAround,
    children: [
      IconButton(
        icon: Icon(Icons.chat),
        onPressed: () {
```

```
            // Add functionality for chat
          },
        ),
        IconButton(
          icon: Icon(Icons.view_headline),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => StoriesScreen()),
            );
          },
        ),
      ],
    ),
  ),
);
}
}


class ChatRoomListTile extends StatefulWidget {
  final String lastMessage, chatRoomId, myUsername;
  ChatRoomListTile(this.lastMessage, this.chatRoomId, this.myUsername);
```

```
  @override
  _ChatRoomListTileState createState() => _ChatRoomListTileState();
}


class _ChatRoomListTileState extends State<ChatRoomListTile> {
  String profilePicUrl = "", name = "", username = "";


  getThisUserInfo() async {
    username =
        widget.chatRoomId.replaceAll(widget.myUsername, "").replaceAll("_", "");
    QuerySnapshot querySnapshot = await DatabaseMethods().getUserInfo(username);
    print(
        "something bla bla ${querySnapshot.docs[0].id} ${querySnapshot.docs[0]["name"]}
${querySnapshot.docs[0]["imgUrl"]}");
    name = "${querySnapshot.docs[0]["name"]}";
    profilePicUrl = "${querySnapshot.docs[0]["imgUrl"]}";
    setState(() {});
  }


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

```
@override
Widget build(BuildContext context) {
  return GestureDetector(
    onTap: () {
      Navigator.push(
          context,
          MaterialPageRoute(
              builder: (context) => ChatScreen(username, name)));
    },
    child: Container(
      margin: EdgeInsets.symmetric(vertical: 8),
      child: Row(
        children: [
          CircleAvatar(
            radius: 20,
            backgroundImage: NetworkImage(profilePicUrl),
          ),
          SizedBox(width: 12),
          Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Text(
```

```
                name,
                style: TextStyle(fontSize: 16),
              ),
              SizedBox(height: 3),
              Text(widget.lastMessage)
            ],
          )
        ],
      ),
    ),
  );
}
}
```

OUTPUT:

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

CONCLUSION:

Learned about navigation, routing and gestures.

MAD & PWA Lab Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	43
Name	Prachit P. Paralikar
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	13

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

Prachit P. Paralikar
D15A 43

MPL EXPT 6

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

THEORY / STEPS:
(Wherever testing mode is available, I have opted for testing mode.) 1.Create a project in firebase console.

2.Go to authentication section inside the project we have created and enable email/password verification under sign-in method.

3. Similarly enable firestore database.

(Further part is done using Flutter CLI i.e Flutter command line interface) Following are the commands for the same

4.Installing firebase CLI

npm install -g firebase-tools

5. We have to connect our project with the google account we have made out firebase project on.

firebase login

It would open a browser where we can choose our account.

6. Activation of flutter fire cli

Dart pub global activate flutterfire_cli

7. Configuration of flutterfire

Flutterfire configure

This shows us our firebase projects and we can select our project that we created for this application. I.e app1

8. It would then show us the platforms that we want to make our application

compatible for. 9. Configured applications will be now available in firebase.

10. We check the flutter initialization in main function of out application.

```
future:
Firebase.initializeApp(),
```

11. We add dependencies:

```
cupertino_icons: ^1.0.2
```

```
  firebase_core: ^2.25.4
  firebase_auth: ^4.17.4
  cloud_firestore: 4.15.5
  provider:
```

These should be compatible with the flutter SDK version we are using. If we get an error with any of these we can try running:
Flutter pub upgrade
Flutter pub get

CODE:
Auth

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:google_sign_in/google_sign_in.dart';
import
'package:messenger_clone/helperfunctions/sharedpref_helper.dar
t';
import 'package:messenger_clone/services/database.dart';
import 'package:messenger_clone/views/home.dart';
import 'package:shared_preferences/shared_preferences.dart';

class AuthMethods {
  final FirebaseAuth auth = FirebaseAuth.instance;

  getCurrentUser() async {
    return await auth.currentUser;
  }

  Future<void> signInWithGoogle(BuildContext context) async {
    final FirebaseAuth _firebaseAuth = FirebaseAuth.instance;
    final GoogleSignIn _googleSignIn = GoogleSignIn();

    try {
      final GoogleSignInAccount? googleSignInAccount =
          await _googleSignIn.signIn();

      if (googleSignInAccount != null) {
```

```
        final GoogleSignInAuthentication
googleSignInAuthentication =
            await googleSignInAccount.authentication;

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

        try {
          final UserCredential result =
              await
_firebaseAuth.signInWithCredential(credential);
          final User? userDetails = result.user;

          if (userDetails != null) {

SharedPreferenceHelper().saveUserEmail(userDetails.email ??
"");

SharedPreferenceHelper().saveUserId(userDetails.uid ?? "");
            SharedPreferenceHelper().saveUserName(
                userDetails?.email?.replaceAll("@gmail.com",
"") ?? "");
            SharedPreferenceHelper()
                .saveDisplayName(userDetails.displayName ??
"");
            SharedPreferenceHelper()
                .saveUserProfileUrl(userDetails.photoURL ??
"");

            Map<String, dynamic> userInfoMap = {
              "email": userDetails.email,
              "username":
userDetails.email!.replaceAll("@gmail.com", ""),
              "name": userDetails.displayName,
```

```
              "imgUrl": userDetails.photoURL
            };

            await DatabaseMethods().addUserInfoToDB(
                userDetails.uid!, userInfoMap); // Fixed this
line
            Navigator.pushReplacement(
                context, MaterialPageRoute(builder: (context)
=> Home()));
          } else {
            print('User details are null');
          }
        } catch (e) {
          print('Error signing in with credential: $e');
        }
      } else {
        print('Google sign-in account is null');
      }
    } catch (e) {
      print('Error signing in with Google: $e');
    }
  }

  Future<void> signOut(BuildContext context) async {
    SharedPreferences prefs = await
SharedPreferences.getInstance();
    prefs.clear();
    await auth.signOut();
    // Navigate to the sign-in screen or perform any other
required action
  }
}
```

OUTPUT:
Authentication:

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

Chats in firestore database:

With this connection we are able to achieve the following:

- **Saving email and password in database.**
- **Saving chats and their timestamps in the database under current user's uid.**

- **Fetching chats from the database to display on the profile page.**
- **Updating the values in the database for password in the authentication database.**

CONCLUSION:

Connected firebase with twitter clone project and handled related errors.

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.	43
Name	Prachit P. Paralikar
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	15

PRACTICAL 7

Name : Prachit P. Paralikar Class: D15A Roll No: 43

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

Theory:

Regular Web App

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

Progressive Web App

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

Difference between PWAs vs. Regular Web Apps:

A Progressive Web is different and better than a Regular Web app with features like:

1. **Native Experience**

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

2. **Ease of Access**

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

3. **Faster Services**

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

4. **Engaging Approach**

As already shared, the PWAs can employ push notifications and other native device features more efficiently. Their interaction does not depend on the browser user uses. This eventually improves the chances of notifying the user regarding your services, offers, and other options related to your brand and keeping them hooked to your brand. In simpler words, PWAs let you maintain the user engagement and retention rate.

5. **Updated Real-Time Data Access**
Another plus point of PWAs is that these apps get updated on their own. They do not demand the end-users to go to the App Store or other such platforms to download the update and wait until installed.
In this app type, the web app developers can push the live update from the server, which reaches the apps residing on the user's devices automatically. Therefore, it is easier for the mobile app developer to provide the best of the updated functionalities and services to the end-users without forcing them to update their app.

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

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

The main features are:

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

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

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

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

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

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

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

Linkable — Easily shared via URL without complex installations.

Offline — Once more it is about putting the user before everything, avoiding the usual error message in case of weak or no connection. The PWA are based on two particularities: first of all the ‘skeleton’ of the app, which recalls the page structure, even if its contents do not respond and its elements include the header, the page layout, as well as an illustration that signals that the page is loading.

CODE AND OUTPUT:

Folder Structure

Name	Date modified	Type	Size
.firebase	03-02-2024 15:46	File folder	
.git	03-02-2024 15:27	File folder	
images	03-02-2024 15:27	File folder	
.firebaserc	03-02-2024 15:45	FIREBASERC File	1 KB
.gitattributes	03-02-2024 15:27	txtfile	1 KB
.gitignore	03-02-2024 15:45	txtfile	2 KB
firebase.json	03-02-2024 15:45	JSON File	1 KB
index	20-03-2024 10:51	Microsoft Edge HTM...	12 KB
manifest.json	20-03-2024 10:53	JSON File	1 KB
merge.py	21-03-2024 14:43	PY File	1 KB
New Text Document	20-03-2024 10:53	Text Document	0 KB
script	03-02-2024 15:27	JavaScript File	1 KB
serviceworker	20-03-2024 10:54	JavaScript File	1 KB
styles	03-02-2024 15:27	Cascading Style Shee...	9 KB

Index.html

```
<!DOCTYPE html>
<html>
   <head>
```

```
        <title>Spa Bonjour</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="preconnect" href="https://fonts.gstatic.com">
        <link
href="https://fonts.googleapis.com/css2?family=Kaushan+Script&family=Poppins:wght
@500&display=swap" rel="stylesheet">
        <script
src="https://cdn.jsdelivr.net/gh/cferdinandi/smooth-scroll/dist/smooth-scroll.polyfills.min.j
s"></script>
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css"
/>
        <link rel="stylesheet" href="styles.css">

    </head>
    <script>
          // Add event listener to execute code when page loads
          window.addEventListener('load', () => {
                    // Call registerSW function when page loads
                    registerSW();
          });

          // Register the Service Worker
          async function registerSW() {
                    // Check if browser supports Service Worker
                    if ('serviceWorker' in navigator) {
                              try {
                                        // Register the Service Worker named 'serviceworker.js'
                                        await navigator.serviceworker.register('serviceworker.js');
                              }
                              catch (e) {
                                        // Log error message if registration fails
                                        console.log('SW registration failed');
                              }
                    }
          }
</script>

<!-- Google tag (gtag.js) -->
<script async
src="https://www.googletagmanager.com/gtag/js?id=G-0PRXPB85YQ"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());
```

```
  gtag('config', 'G-0PRXPB85YQ');
</script>
   <body>
      <!---1. Banner/Home Section-->
      <section id="banner">
        <img src="images/logo.png" alt="logo" class="logo">
        <div class="banner-text">
        <h1>Hair Salon</h1>
        <p>Style your hair is Style your life</p>
        <div class="banner-btn">
          <a href="#"><span></span> Find Out</a>
          <a href="#"><span></span> Read More</a>
        </div>
        </div>
        <div id="sideNav">
            <nav>
                <ul>
                    <li><a href="#banner">HOME</a></li>
                    <li><a href="#feature">FEATURES</a></li>
                    <li><a href="#service">SERVICES</a></li>
                    <li><a href="#testimonial">TESTIMONIALS</a></li>
                    <li><a href="#contact">CONTACT US</a></li>
                </ul>
            </nav>
        </div>
        <div id="menuBtn">
            <img src="images/menu.png" alt="menu logo" id="menu">
        </div>
      </section>

      <!--- 2. features Section-->
      <section id="feature">
          <div class="title-text">
              <p>Features</p>
              <h1>WHY CHOOSE US</h1>
          </div>
          <div class="feature-box">
              <div class="features">
                  <h1>Experienced Staff</h1>
                  <div class="features-desc">
                      <div class="feature-icon"><i class="fa fa-shield"></i></div>
                      <div class="feature-text"><p>Lorem ipsum dolor sit amet, consectetur
adipisicing elit.
                          Necessitatibus cumque harum tenetur nam nisi, ducimus, a totam
```

```
veniam magni
                    atque nihil error, sapiente quibusdam at corporis sit quae dolore
quidem!</p>
                 </div>
              </div>
              <h1>Pre-Booking Online</h1>
              <div class="features-desc">
                 <div class="feature-icon"><i class="fa fa-check-square-o"></i></div>
                 <div class="feature-text"><p>Lorem ipsum dolor sit amet, consectetur
adipisicing elit.
                    Necessitatibus cumque harum tenetur nam nisi, ducimus, a totam
veniam magni
                    atque nihil error, sapiente quibusdam at corporis sit quae dolore
quidem!</p>
                 </div>
              </div>
              <h1>Affordable Cost</h1>
              <div class="features-desc">
                 <div class="feature-icon"><i class="fa fa-inr"></i></div>
                 <div class="feature-text"><p>Lorem ipsum dolor sit amet, consectetur
adipisicing elit.
                    Necessitatibus cumque harum tenetur nam nisi, ducimus, a totam
veniam magni
                    atque nihil error, sapiente quibusdam at corporis sit quae dolore
quidem!</p>
                 </div>
              </div>
           </div>
           <div class="feature-img">
              <img src="images/barber-man.jpg" alt="barber-man">
           </div>
        </div>
     </section>

      <!--- 3. Service Section-->
      <section id="service">
         <div class="title-text">
            <p>Services</p>
            <h1>WE PROVIDE BETTER</h1>
         </div>
         <div class="service-box">
            <div class="single-service"><img id="img" src="images/pic-1.jpg" alt="pic1">
               <div class="overlay"></div>
               <div class="service-desc">
                  <h3>Hair Cutting</h3><hr>
```

```
                <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Ullam ipsa
quod sit! Earum quis fuga ut ratione amet necessitatibus minus voluptates.</p>
            </div>
        </div>
        <div class="single-service"><img id="img" src="images/pic-2.jpg" alt="pic1">
            <div class="overlay"></div>
            <div class="service-desc">
                <h3>Hair Styling</h3><hr>
                <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Ullam ipsa
quod sit! Earum quis fuga ut ratione amet necessitatibus minus voluptates.</p>
            </div>
        </div>
        <div class="single-service"><img id="img" src="images/pic-3.jpg" alt="pic1">
            <div class="overlay"></div>
            <div class="service-desc">
                <h3>Hair Dying</h3><hr>
                <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Ullam ipsa
quod sit! Earum quis fuga ut ratione amet necessitatibus minus voluptates.</p>
            </div>
        </div>
        <div class="single-service"><img id="img" src="images/pic-4.jpg" alt="pic1">
            <div class="overlay"></div>
            <div class="service-desc">
                <h3>Dry Shampooing</h3><hr>
                <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Ullam ipsa
quod sit! Earum quis fuga ut ratione amet necessitatibus minus voluptates.</p>
            </div>
        </div>
    </div>
</section>

<!---4. Testimonials Section-->
<section id="testimonial">
    <div class="title-text">
        <p>Testimonials</p>
        <h1>WHAT OUR CLIENTS SAYS</h1>
    </div>
    <div class="testimonial-row">
        <div class="testimonial-col">
            <div class="user">
                <img src="images/img-1.jpg" src="image">
                <div class="user-info">
                    <h4>Ken Pollard<i class="fa fa-twitter"></i></h4>
                    <small>@kenpollard</small>
                </div>
```

```
                </div>
                <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit.
                    Voluptatum odit et ratione dolorem delectus nobis, sunt consectetur,
                    quos eveniet nihil harum? Deserunt nam ex quo commodi, facilis
laudantium repellendus dolorem!</p>
            </div>
            <div class="testimonial-col">
                <div class="user">
                    <img src="images/img-2.jpg" src="image">
                    <div class="user-info">
                        <h4>Sabiya Tabassum<i class="fa fa-twitter"></i></h4>
                        <small>@nerd__fswd</small>
                    </div>
                </div>
                <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit.
                    quos eveniet nihil harum? Deserunt nam ex quo commodi, facilis
laudantium repellendus dolorem!</p>
            </div>
            <div class="testimonial-col">
                <div class="user">
                    <img src="images/img-3.jpg" src="image">
                    <div class="user-info">
                        <h4>Suriya<i class="fa fa-twitter"></i></h4>
                        <small>@im.suriya</small>
                    </div>
                </div>
                <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit.
                    Voluptatum odit et ratione dolorem delectus nobis, sunt consectetur,
                    quos eveniet nihil harum? Deserunt nam ex quo commodi, facilis
laudantium repellendus dolorem!</p>
            </div>
        </div>
    </section>

    <!---5. Contact Us section-->
    <section id="contact">
        <img src="images/footer-img.png" alt="footer img" class="footer-img">
        <div class="title-text">
            <p>Contact Us</p>
            <h1>Visit Shop Today</h1>
        </div>
        <div class="footer-row">
            <div class="footer-left">
                <h1>Opening Hours</h1>
                <p><i class="fa fa-clock-o"></i>Mon - Fri : 9:00 AM - 9:00 PM</p>
```

```
                <p><i class="fa fa-clock-o"></i>Sat - Sun : 10:00 AM - 11:00 PM</p>
            </div>
            <div class="footer-right">
                <h1>Get in Touch</h1>
                <p><i class="fa fa-map-marker"></i>#30 ABC Colony, XYZ City, IN</p>
                <p><i class="fa fa-paper-plane"></i>example@website.in</p>
                <p><i class="fa fa-map-marker"></i>+01 1123456782</p>
            </div>
        </div>
        <div class="social-links">
            <a href="#"><i class="fa fa-facebook"></i></a>
            <a href="#"><i class="fa fa-twitter"></i></a>
            <a href="#"><i class="fa fa-instagram"></i></a>
            <a href="#"><i class="fa fa-youtube-play"></i></a><br>
            <small> © All Copyrights are reserved to nerd__fswd.</small>
        </div>
    </section>
    <script src="script.js" type="text/javascript"></script>
<script type="module">
  // Import the functions you need from the SDKs you need
  import { initializeApp } from "https://www.gstatic.com/firebasejs/10.8.0/firebase-app.js";
  import { getAnalytics } from
"https://www.gstatic.com/firebasejs/10.8.0/firebase-analytics.js";
  // TODO: Add SDKs for Firebase products that you want to use
  // https://firebase.google.com/docs/web/setup#available-libraries

  // Your web app's Firebase configuration
  // For Firebase JS SDK v7.20.0 and later, measurementId is optional
  const firebaseConfig = {
    apiKey: "AIzaSyCt0d-RSB3hgKMUU20QbbZA7CNRCkrXaMg",
    authDomain: "barber-985c6.firebaseapp.com",
    projectId: "barber-985c6",
    storageBucket: "barber-985c6.appspot.com",
    messagingSenderId: "3049260321",
    appId: "1:3049260321:web:b3a79fc45245ff78986a60",
    measurementId: "G-X7FB372LM2"
  };

  // Initialize Firebase
  const app = initializeApp(firebaseConfig);
  const analytics = getAnalytics(app);
</script>
   </body>

</html>
```

serviceworker.js

```
var staticCacheName = "pwa";

self.addEventListener("install", function (e) {
e.waitUntil(
	caches.open(staticCacheName).then(function (cache) {
	return cache.addAll(["/"]);
	})
);
});

self.addEventListener("fetch", function (event) {
console.log(event.request.url);

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

manifest.json

```
{
	"name":"PWA Tutorial",
	"short_name":"PWA",
	"start_url":"index.html",
	"display":"standalone",
	"background_color":"#5900b3",
	"theme_color":"black",
	"scope": ".",
	"description":"This is a PWA tutorial.",
	"icons":[
	{
	"src":"images/icon-192x192.png",
	"sizes":"192x192",
	"type":"image/png"
	},
	{
	"src":"images/icon-512x512.png",
	"sizes":"512x512",
```

```
        "type":"image/png"
        }
]
}
```

Starting the server

Going to the developers option

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

Application

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

Click on 3 dots —> App —> install App

Allow for creating desktop shortcut

Conclusion:

- Thus “add to homescreen feature was successfully implemented”
- We created a shortcut icon to open our websites app locally

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.	43
Name	Prachit P. Paralikar
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

PRACTICAL 8

Name : Prachit P. Paralikar **Class: D15A** **Roll No: 43**

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

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

Things to note about Service Worker:

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

What can we do with Service Workers?

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

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

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

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

What can’t we do with Service Workers?

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

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

Service Worker Cycle

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

- Registration
- Installation
- Activation

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

main.js

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

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

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

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

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

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

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

main.js

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

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

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

service-worker.js

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

Activation
Once a service worker has successfully installed, it transitions into the activation stage. If there are any open pages controlled by the previous service worker, the new service

worker enters a waiting state. The new service worker only activates when there are no longer any pages loaded that are still using the old service worker. This ensures that only one version of the service worker is running at any given time.

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

service-worker.js

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

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

CODE AND OUTPUT:

1) Index.html

```
<!DOCTYPE html>
<html>
   <head>
      <title>Spa Bonjour</title>
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <link rel="preconnect" href="https://fonts.gstatic.com">
      <link
href="https://fonts.googleapis.com/css2?family=Kaushan+Script&family=Poppins:wght
@500&display=swap" rel="stylesheet">
      <script
src="https://cdn.jsdelivr.net/gh/cferdinandi/smooth-scroll/dist/smooth-scroll.polyfills.min.j
s"></script>
      <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css"
/>
      <link rel="stylesheet" href="styles.css">

   </head>
   <script>
         // Add event listener to execute code when page loads
```

```
        window.addEventListener('load', () => {
                // Call registerSW function when page loads
                registerSW();
        });

        // Register the Service Worker
        async function registerSW() {
                // Check if browser supports Service Worker
                if ('serviceWorker' in navigator) {
                        try {
                                // Register the Service Worker named 'serviceworker.js'
                                await navigator.serviceworker.register('serviceworker.js');
                        }
                        catch (e) {
                                // Log error message if registration fails
                                console.log('SW registration failed');
                        }
                }
        }
</script>

<!-- Google tag (gtag.js) -->
<script async
src="https://www.googletagmanager.com/gtag/js?id=G-0PRXPB85YQ"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'G-0PRXPB85YQ');
</script>
   <body>
      <!---1. Banner/Home Section-->
      <section id="banner">
        <img src="images/logo.png" alt="logo" class="logo">
        <div class="banner-text">
        <h1>Hair Salon</h1>
        <p>Style your hair is Style your life</p>
        <div class="banner-btn">
          <a href="#"><span></span> Find Out</a>
          <a href="#"><span></span> Read More</a>
        </div>
        </div>
        <div id="sideNav">
           <nav>
```

```
            <ul>
                <li><a href="#banner">HOME</a></li>
                <li><a href="#feature">FEATURES</a></li>
                <li><a href="#service">SERVICES</a></li>
                <li><a href="#testimonial">TESTIMONIALS</a></li>
                <li><a href="#contact">CONTACT US</a></li>
            </ul>
        </nav>
    </div>
    <div id="menuBtn">
        <img src="images/menu.png" alt="menu logo" id="menu">
    </div>
  </section>

  <!--- 2. features Section-->
  <section id="feature">
      <div class="title-text">
          <p>Features</p>
          <h1>WHY CHOOSE US</h1>
      </div>
      <div class="feature-box">
          <div class="features">
              <h1>Experienced Staff</h1>
              <div class="features-desc">
                  <div class="feature-icon"><i class="fa fa-shield"></i></div>
                  <div class="feature-text"><p>Lorem ipsum dolor sit amet, consectetur
adipisicing elit.
                      Necessitatibus cumque harum tenetur nam nisi, ducimus, a totam
veniam magni
                      atque nihil error, sapiente quibusdam at corporis sit quae dolore
quidem!</p>
                  </div>
              </div>
              <h1>Pre-Booking Online</h1>
              <div class="features-desc">
                  <div class="feature-icon"><i class="fa fa-check-square-o"></i></div>
                  <div class="feature-text"><p>Lorem ipsum dolor sit amet, consectetur
adipisicing elit.
                      Necessitatibus cumque harum tenetur nam nisi, ducimus, a totam
veniam magni
                      atque nihil error, sapiente quibusdam at corporis sit quae dolore
quidem!</p>
                  </div>
              </div>
              <h1>Affordable Cost</h1>
```

```
                <div class="features-desc">
                    <div class="feature-icon"><i class="fa fa-inr"></i></div>
                    <div class="feature-text"><p>Lorem ipsum dolor sit amet, consectetur
adipisicing elit.
                        Necessitatibus cumque harum tenetur nam nisi, ducimus, a totam
veniam magni
                        atque nihil error, sapiente quibusdam at corporis sit quae dolore
quidem!</p>
                    </div>
                </div>
            </div>
            <div class="feature-img">
                <img src="images/barber-man.jpg" alt="barber-man">
            </div>
        </div>
    </section>

    <!--- 3. Service Section-->
    <section id="service">
        <div class="title-text">
            <p>Services</p>
            <h1>WE PROVIDE BETTER</h1>
        </div>
        <div class="service-box">
            <div class="single-service"><img id="img" src="images/pic-1.jpg" alt="pic1">
                <div class="overlay"></div>
                <div class="service-desc">
                    <h3>Hair Cutting</h3><hr>
                    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Ullam ipsa
quod sit! Earum quis fuga ut ratione amet necessitatibus minus voluptates.</p>
                </div>
            </div>
            <div class="single-service"><img id="img" src="images/pic-2.jpg" alt="pic1">
                <div class="overlay"></div>
                <div class="service-desc">
                    <h3>Hair Styling</h3><hr>
                    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Ullam ipsa
quod sit! Earum quis fuga ut ratione amet necessitatibus minus voluptates.</p>
                </div>
            </div>
            <div class="single-service"><img id="img" src="images/pic-3.jpg" alt="pic1">
                <div class="overlay"></div>
                <div class="service-desc">
                    <h3>Hair Dying</h3><hr>
                    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Ullam ipsa
```

```
quod sit! Earum quis fuga ut ratione amet necessitatibus minus voluptates.</p>
                </div>
            </div>
            <div class="single-service"><img id="img" src="images/pic-4.jpg" alt="pic1">
                <div class="overlay"></div>
                <div class="service-desc">
                    <h3>Dry Shampooing</h3><hr>
                    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Ullam ipsa
quod sit! Earum quis fuga ut ratione amet necessitatibus minus voluptates.</p>
                </div>
            </div>
        </div>
    </section>

    <!---4. Testimonials Section-->
    <section id="testimonial">
        <div class="title-text">
            <p>Testimonials</p>
            <h1>WHAT OUR CLIENTS SAYS</h1>
        </div>
        <div class="testimonial-row">
            <div class="testimonial-col">
                <div class="user">
                    <img src="images/img-1.jpg" src="image">
                    <div class="user-info">
                        <h4>Ken Pollard<i class="fa fa-twitter"></i></h4>
                        <small>@kenpollard</small>
                    </div>
                </div>
                <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit.
                    Voluptatum odit et ratione dolorem delectus nobis, sunt consectetur,
                    quos eveniet nihil harum? Deserunt nam ex quo commodi, facilis
laudantium repellendus dolorem!</p>
                        <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit.
                    quos eveniet nihil harum? Deserunt nam ex quo commodi, facilis
laudantium repellendus dolorem!</p>
            </div>
            <div class="testimonial-col">
                <div class="user">
                    <img src="images/img-3.jpg" src="image">
                    <div class="user-info">
                        <h4>Suriya<i class="fa fa-twitter"></i></h4>
                        <small>@im.suriya</small>
                    </div>
                  </div>
```

```
        </section>

        <!---5. Contact Us section-->
        <section id="contact">

            </div>
            <div class="footer-row">
                <div class="footer-left">
                    <h1>Opening Hours</h1>
                    <p><i class="fa fa-clock-o"></i>Mon - Fri : 9:00 AM - 9:00 PM</p>
                    <p><i class="fa fa-clock-o"></i>Sat - Sun : 10:00 AM - 11:00 PM</p>
                </div>
                <div class="footer-right">
                    <h1>Get in Touch</h1>
                    <p><i class="fa fa-map-marker"></i>#30 ABC Colony, XYZ City, IN</p>
                    <p><i class="fa fa-paper-plane"></i>example@website.in</p>
                    <p><i class="fa fa-map-marker"></i>+01 1123456782</p>
                </div>
            </div>
            <div class="social-links">
                <a href="#"><i class="fa fa-facebook"></i></a>
                <a href="#"><i class="fa fa-twitter"></i></a>
                <a href="#"><i class="fa fa-instagram"></i></a>
                <a href="#"><i class="fa fa-youtube-play"></i></a><br>
                <small> © All Copyrights are reserved to nerd__fswd.</small>
            </div>
        </section>
       <script src="script.js" type="text/javascript"></script>
<script type="module">
  // Import the functions you need from the SDKs you need
  import { initializeApp } from "https://www.gstatic.com/firebasejs/10.8.0/firebase-app.js";
  import { getAnalytics } from
"https://www.gstatic.com/firebasejs/10.8.0/firebase-analytics.js";
  // TODO: Add SDKs for Firebase products that you want to use
  // https://firebase.google.com/docs/web/setup#available-libraries

  // Your web app's Firebase configuration
  // For Firebase JS SDK v7.20.0 and later, measurementId is optional
  const firebaseConfig = {
    apiKey: "AIzaSyCt0d-RSB3hgKMUU20QbbZA7CNRCkrXaMg",
    authDomain: "barber-985c6.firebaseapp.com",
    projectId: "barber-985c6",
    storageBucket: "barber-985c6.appspot.com",
    messagingSenderId: "3049260321",
    appId: "1:3049260321:web:b3a79fc45245ff78986a60",
```

```
  measurementId: "G-X7FB372LM2"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
</script>
  </body>

</html>
```

2) script.js

```
// Function to register the Service Worker
async function registerServiceWorker() {
    if ('serviceWorker' in navigator) {
        try {
            // Register the Service Worker named 'serviceworker.js'
            const registration = await
navigator.serviceWorker.register('serviceworker.js');
            console.log('Service Worker registered successfully:', registration);
        } catch (error) {
            console.error('Service worker registration failed:', error);
        }
    } else {
        console.log('Service workers are not supported in this browser.');
    }
}

// Add event listener to execute code when page loads
window.addEventListener('load', () => {
    // Call function to register Service Worker
    registerServiceWorker();

    // Rest of your existing script
    var menuBtn = document.getElementById('menuBtn');
    var sideNav = document.getElementById('sideNav');
    var menu = document.getElementById('menu');

    sideNav.style.right = "-250px";

    menuBtn.onclick = function(){
        if(sideNav.style.right == "-250px"){
            sideNav.style.right="0";
```

```
                menu.src = "images/close.png";
            }
            else{
                sideNav.style.right="-250px";
                menu.src = "images/menu.png";
            }
        }

        // All animations will take exactly 1000ms
        var scroll = new SmoothScroll('a[href*="#"]', {
            speed: 1000,
            speedAsDuration: true
        });
    });
```

3) service-worker.js

```
var staticCacheName = "pwa-v1"; // Update cache name to indicate a new version

// Installation
self.addEventListener("install", function (e) {
    console.log("Service Worker installed");
    e.waitUntil(
        caches.open(staticCacheName).then(function (cache) {
            console.log("Opened cache");
            return cache.addAll([
                "/", // Add paths to cache here
                "/index.html",
                "/styles.css",
                "/script.js",
                "/images/logo.png",
                // Add other static assets you want to cache
            ]);
        })
    );
});

// Activation
self.addEventListener("activate", function (e) {
    console.log("Service Worker activated");

    // Remove old caches
    e.waitUntil(
        caches.keys().then(function (cacheNames) {
            return Promise.all(
```

```
            cacheNames.filter(function (cacheName) {
                return cacheName !== staticCacheName;
            }).map(function (cacheName) {
                console.log("Deleting old cache:", cacheName);
                return caches.delete(cacheName);
            })
        );
    })
  );
});

// Fetch
self.addEventListener("fetch", function (event) {
  console.log("Fetch event for ", event.request.url);
  event.respondWith(
    caches.match(event.request).then(function (response) {
        if (response) {
            console.log("Found ", event.request.url, " in cache");
            return response;
        }
        console.log("Network request for ", event.request.url);
        return fetch(event.request).then(function (response) {
            // Add fetched responses to cache
            return caches.open(staticCacheName).then(function (cache) {
                console.log("Caching new resource:", event.request.url);
                cache.put(event.request.url, response.clone());
                return response;
            });
        });
    }).catch(function (error) {
        // Handle errors gracefully
        console.error("Error fetching data from cache: ", error);
    })
  );
});
```

Now Right click —> Inspect —> Application —> Service workers

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

Now scroll down to Cache Storage

#	Name	Response-...	Content-T...	Content-L...	Time Cach...	Vary Header
0	/	basic	text/html	13,296	26/3/2024...	Origin
1	/images/banner.jpg	basic	image/jpeg	457,629	26/3/2024...	Origin
2	/images/barber-man.jpg	basic	image/jpeg	137,380	26/3/2024...	Origin
3	/images/footer-img.png	basic	image/png	68,452	26/3/2024...	Origin
4	/images/img-1.jpg	basic	image/jpeg	71,492	26/3/2024...	Origin
5	/images/img-2.jpg	basic	image/jpeg	71,685	26/3/2024...	Origin
6	/images/img-3.jpg	basic	image/jpeg	52,489	26/3/2024...	Origin
7	/images/logo.png	basic	image/png	5,169	26/3/2024...	Origin
8	/images/menu.png	basic	image/png	1,542	26/3/2024...	Origin

Conclusion:

- Registered a service worker, and completed the installation and activation process for a new service worker.
- Also checked the Cache Storage

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.	43
Name	Prachit P. Paralikar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

PRACTICAL 09

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

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

- CacheFirst - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.
- NetworkFirst - In this function, firstly we can try getting an updated response from the network, if this process completed successfully, the new response will be cached and returned. But if this process fails, we check whether the request has been cached before or not. If a cache exists, it is returned to the page, but if not, this is up to you. You can return dummy content or information messages to the page.

Sync Event

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

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

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

1. When we click the "send" button, email content will be saved to IndexedDB.
2. Background Sync registration.
3. If the Internet connection is available, all email content will be read and sent to

Mail Server.
If the Internet connection is unavailable, the service worker waits until the connection is available even though the window is closed. When it is available, email content will be sent to Mail Server.

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

Event Listener for Background Sync Registration

Event Listener for sw.js

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

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

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

Code:
service-worker.js:

```
var staticCacheName = "pwa";
self.addEventListener("install", function (event) {
    event.waitUntil(preLoad());
});
self.addEventListener("fetch", function (event) {
    event.respondWith(
        checkResponse(event.request).catch(function () {
            console.log("Fetch from cache successful!");
            return returnFromCache(event.request);
        })
    );
    console.log("Fetch successful!");
    event.waitUntil(addToCache(event.request));
```

```
});
self.addEventListener("sync", (event) => {
    if (event.tag === "syncMessage") {
        console.log("Sync successful!");
    }
});
self.addEventListener("push", function (event) {
    if (event && event.data) {
        var data = event.data.json();
        if (data.method == "pushMessage") {
            console.log("Push notification requested");
            if (Notification.permission === "granted") {
                event.waitUntil(
                    self.registration.showNotification("To Do List", {
                        body: data.message,
                    }).catch(function(error) {
                        console.error("Error showing notification:",
error);
                    })
                );
            } else if (Notification.permission === "denied") {
                console.error("Notification permission denied.");
                // Handle the case where notification permission is
denied, such as showing a message to
                //the user.
            } else {
                Notification.requestPermission().then(function(permission)
{
                    if (permission === "granted") {
                        console.log("Notification permission granted,
showing notification");
                        self.registration.showNotification("To Do List", {
                            body: data.message,
                        });
                    } else {
                        console.error("Notification permission denied.");
                        // Handle the case where notification permission
is denied after requesting.
                    }
                });
```

```
            }
        }
    }
});

var filesToCache = ["/", "/index.html"];
var preLoad = function () {
    return caches.open("index").then(function (cache) {
        return cache.addAll(filesToCache);
    });
};
var checkResponse = function (request) {
    return new Promise(function (fulfill, reject) {
        fetch(request)
        .then(function (response) {
            if (response.status !== 404) {
                fulfill(response);
            } else {
                reject();
            }
        })
        .catch(reject);
    });
};
var addToCache = function (request) {
    return caches.open("index").then(function (cache) {
        return fetch(request).then(function (response) {
            return cache.put(request, response);
        });
    });
};
var returnFromCache = function (request) {
    return caches.open("index").then(function (cache) {
        return cache.match(request).then(function (matching) {
            if (!matching || matching.status == 404) {
                return cache.match("index.html");
            } else {
                return matching;
            }
        });
```

```
    });
};
self.addEventListener("activate", (event) => {
    event.waitUntil(
        caches.keys().then((cacheNames) => {
            return Promise.all(
                cacheNames
                .filter((name) => {
                    return name !== staticCacheName;
                })
                .map((name) => {
                    return caches.delete(name);
                })
            );
        })
    );
});
```

Script,js

```
// Function to register the Service Worker
async function registerServiceWorker() {
    if ('serviceWorker' in navigator) {
        try {
            // Register the Service Worker named 'serviceworker.js'
            const registration = await
navigator.serviceWorker.register('serviceworker.js');
            console.log('Service Worker registered successfully:',
registration);
        } catch (error) {
            console.error('Service worker registration failed:', error);
        }
    } else {
        console.log('Service workers are not supported in this browser.');
    }
}
if ('Notification' in window) {
    Notification.requestPermission().then(function(permission) {
```

```
        if (permission === "granted") {
            console.log("Notification permission granted.");
        } else {
            console.warn("Notification permission denied.");
        }
    });
}

// Add event listener to execute code when page loads
window.addEventListener('load', () => {
    // Call function to register Service Worker
    registerServiceWorker();

    // Rest of your existing script
    var menuBtn = document.getElementById('menuBtn');
    var sideNav = document.getElementById('sideNav');
    var menu = document.getElementById('menu');

    sideNav.style.right = "-250px";

    menuBtn.onclick = function(){
        if(sideNav.style.right == "-250px"){
            sideNav.style.right="0";
            menu.src = "images/close.png";
        }
        else{
            sideNav.style.right="-250px";
            menu.src = "images/menu.png";
        }
    }

    // All animations will take exactly 1000ms
    var scroll = new SmoothScroll('a[href*="#"]', {
        speed: 1000,
        speedAsDuration: true
    });
});
```

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

Output:
Fetch:

Push:

Sync:

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

Conclusion:

In this experiment, we have successfully implemented service worker events like fetch, sync and push for my Spa-Bonjour PWA and found out output for above implementation.

MAD & PWA Lab Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	43
Name	Prachit P. Paralikar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Practical 10

Name : Prachit P. Paralikar **Class: D15A** **Roll No: 43**

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:
- Blogging with Jekyll
- Custom URL
- Automatic Page Generator

Reasons for favoring this over Firebase:
- Free to use
- Right out of github
- Quick to set up

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

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

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

Firebase

The Realtime App Platform. Firebase is a cloud service designed to power real-time,

collaborative applications. Simply add the Firebase library to your application to gain access to a shared data structure; any changes you make to that data are automatically synchronized with the Firebase cloud and with other clients within milliseconds.

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

- Realtime backend made easy
- Fast and responsive

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

<u>Pros</u>

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

<u>Cons</u>

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

CODE AND OUTPUT:

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

GITHUB REPOSITORY LINK : Github -repo

GITHUB SCREENSHOT :

HOSTED WEBSITE LINK: https://prachit707.github.io/spa-bonjour/

MAD & PWA Lab Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	43
Name	Prachit P. Paralikar
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	15

PRACTICAL 11

Name : Prachit P. Paralikar **Class: D15A** **Roll No: 43**

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

Theory:

Google Lighthouse :

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

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

Key Features and Audit Metrics

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

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

PWA Score (Mobile): Thanks to the rise of Service Workers, app manifests, etc., a lot of modern web applications are moving towards the PWA paradigm, where the objective is to make the application behave as close as possible to native mobile applications. Scoring points are based on the Baseline PWA checklist laid down by Google which includes Service Worker implementation(s), viewport handling, offline functionality, performance in script-disabled environments, etc.

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

Best Practices: As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many

such points, including but not limited to:Use of HTTPS
Avoiding the use of deprecated code elements like tags, directives, libraries, etc.
Password input with paste-into disabled Geo-Location and cookie usage alerts on load, etc.

CODE AND OUTPUT:

manifest.json

```
{
        "name":"PWA Tutorial",
        "short_name":"PWA",
        "start_url":"index.html",
        "display":"standalone",
        "background_color":"#5900b3",
        "theme_color":"black",
        "scope": ".",
        "description":"This is a PWA tutorial.",
        "icons":[
        {
        "src":"images/logo1.png",
        "sizes":"192x192",
        "type":"image/png"

        },
        {
        "src":"images/logo2.png",
        "sizes":"512x512",
        "type":"image/png"
        }
]
}
```

After changes made in manifest.json

```
{
        "name":"PWA Tutorial",
        "short_name":"PWA",
        "start_url":"index.html",
        "display":"standalone",
        "background_color":"#5900b3",
        "theme_color":"black",
        "scope": ".",
        "description":"This is a PWA tutorial.",
        "icons":[
        {
        "src":"images/logo1.png",
        "sizes":"192x192",
        "type":"image/png",
        "purpose":"any maskable"

        },
        {
        "src":"images/logo2.png",
```

```
        "sizes":"512x512",
        "type":"image/png",
        "purpose":"any maskable"

        }
]
}
```

Conclusion:

- We analyzed the website with the help of lighthouse tool and found some issues with the website
- Hence we made changes in the manifest.json file we added "purpose":"any maskable"
  for maskable error face
- Also added a meta tag in index.html to resolve the theme error faced

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.	43
Name	Prachit P. Paralikar
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:	5

Prachit P. Pralkar
DISA 43
MAD Lab

Assignment-1

Q1 Flutter Overview - Explain key features of Flutter and advantages. Discuss how Flutter framework differs from traditional approaches and why it has gained popularity

Key Features of Flutter:

1) Single Codebase for multiple Platforms -
Flutter allows to write codes and deploy it on both IOS and Android.

2) Hot Reload -
This enables developers to see immediately the changes they have made

3) Expressive UI -
Developers have flexibility to create expressive and flexible UIs

4) Integrn with other tools -
Flutter can easily integrate with other popular development tools and frameworks

Advantages of Flutter -

1) Faster Development
Use single codebase for diff. platforms

2) Consistent UI across platforms
Widgets provide a consistent look and feel across platforms

3) Cost efficiency
Single codebase for both IOS and Android reduces cost

Difference from traditional Approach

a) Traditional approaches rely on a hierarchical structure for UI components whereas Flutter uses a widget-based approach

b) Flutter compiles the native ARM code, providing performance comparable to native applications

Flutter popularity is driven by increased productivity, a growing community, flexibility in UI and adoption by major companies

Q2 Widget Tree and Composition: Describe the concept of widget tree in Flutter. Explain how widget composition is used to build complex UI. Provide example of commonly used widgets

→ Widget Tree:

1) Widget Tree is a hierarchical structure of widgets that defines the UI of an app

2) Every visual element, from simple component to complex layouts, is represented by a widget

3) Widgets are (2) types:

a) Stateless Widget

It is immutable and cannot change over time. Ex: Images, Text

b) Stateful Widget

Widget that can change over time

Ex - Buttons, forms

Widget Composition

1) Widget Composition in Flutter involves combining multiple simple widgets to create more complex and compound widgets

2) This composability is a powerful concept that allows developers to develop sophisticated UI by nesting widgets

Commonly Used Widgets:

1) Container: A box model for padding margin and decoration

2) Column and Row: Layout widgets for arranging children vertically or horizontally.

3) List View: A scrollable list of widgets

4) Grid View: A scrollable grid of widgets

5) AppBar: A material design app bar typically at top of screen.

6) TextField: An input field for users to enter text

7) Button Widgets: Interactive Buttons for user actions

Q3 State Management in Flutter

Discuss importance of state management in Flutter. Compare and contrast the diff. state management approaches available in Flutter. Provide scenario where each approach is suitable

FOR EDUCATIONAL USE

Project Title: Flutter: Messenger Clone/PWA: Spa-Bonjour **Roll No: 43**

→ • State management is crucial in Flutter because it involves managing the data that can change over time.
• Flutter is reactive, meaning the UI rebuilds when the underlying data changes

Set State	Provider	RiverPod
1) Built-in Flutter method	1) External package named ('Provider')	1) External package ('riverpod')
2) Local state within a widget	2) Global state within widget tree	2) Global state with additional features
3) Limited scability for large apps	3) Suitable for medium sized apps	3) Designed for large and complex app
4) Make lead to code redundancy	4) Balances simplicity and readability	4) Emphasizes readability & dev synchr
5) Testing can be more challenging	5) Good testing support	5) Enchance of testing experience.
6) Widely used well established	6) Widely adopted well supported	6) Gaining popularity.

Sundaram
FOR EDUCATIONAL USE

Scenario where each is used:

1) set State:
- For small to moderately complex applications
- when managing local state within a widget

Eg - Simple forms, UI components with local UI-specific state.

2) Provider
- For medium to large-sized apps.
- When a centralized state is needed, accessible by multiple widgets

Ex - Managing user authentication, theme changes

3) Riverpod
- For large and complex applications
- when testability and maintainability are top priorities

Ex - Complex applications with multiple feature dynamics UIs

Q4 Firebase Integration in Flutter: Explain the process. Discuss the benefits of using Firebase as a backened solution. Highlight backened services used in Flutter and brief overview of data synchronization

→ Integration:

1) Go to Firebase Console, create a new project.

2) Add Firebase SDK by including dependencies in pubspec.yaml.

dependencies:
  firebase_core: ^version
  firebase_auth: ^version
  cloud_firebase: ^version

3) Run flutter pub get
4) Initialize firebase by calling: Firebase.initializeApp()

```
import 'package:firebase_core/firebase_core.dart

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

Benefits of Firebase as Backend:
1) Real-time Database
  Firebase offers real-time NoSQL database.

2) Authentication
  Provides a secure way to implement authentication

3) Hosting: Simple and efficient way to deploy web apps

Prathit P. Pawalkar

Data Synchronization

1) Real time Database
- when Data changes on one client, it triggers event that automatically updates data on other clients

2) Cloud Firebase:
It notifies client when data changes, allowing for seamless real-time updates

3) Authentication:
If user signs in or out on one device, the authentication state is automatically reflected on other devices

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.	43
Name	Prachit P. Paralikar
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:	4

Prachi P. Parlikar
DISA 43
PWA Lab

Assignment - 2

Q1 Define Progressive Web App (PWA) and explain significance in modern web development. Discuss the key characteristics that differentiate PWA from traditional mobile App

→ A PWA is a type of web application that uses modern web technologies to provide a native app like experience to users. They are designed to work on any platforms that uses a standards-compliant browser, such as desktop and mobile devices.

Key characteristics that differentiate PWA:

1) Cross platform compatibility -
PWAs work across different devices and platforms.

2) Responsive Design.
It is built with responsive design principle.

3) Offline Functionality
It can work offline or with poor internet connection

4) App like Experience
It offers features such as push notifications, home screen installation, full screen mode making them feel like native app.

5) Fast Performance

6) Recoverability = It
It is discoverable through search engines and non.

Sundaram
FOR EDUCATIONAL USE

be shared easily.

Overall PWA combines best features of web and mobile app, offering a compelling alternative to traditional ~~web~~ app.

Q2 Define responsive web design and explain its importance in the context of PWA. Compare and contrast responsive fluid and adaptive web design approach.

→ Responsive web design is an approach to web design that aims to create web pages that adjust and respond to the user's device and screen size. It uses a combination of flexible grids and layouts, images and CSS to ensure that a website looks and is functionally optimal on any device, whether it is a desktop, tablet or smartphone.

Responsive vs fluid vs adaptive:

Feature	Responsive	Fluid	Adaptive
1) Layout	Flexibly, adapts to any screen size	Continually adjusts based on relative units	Uses multiple fixed width layouts.

FOR EDUCATIONAL USE

	Feature	Responsive	Fluid	Adaptive
2)	Development/ effort	Moderate	less effort for base layout	More efforts to create multiple layouts
3)	Control	Good control over overall layout	less control over element appearance of extreme sizes	High control over layout for each category
4)	Usability for PWAs	Ideas fundn. due to its versatility	can be used but may require adjustment for optimal requirements.	Lets common for PWA's due to developing overhead.

Q3 Describe the life cycle of service workers including registration, installation and activation phase.

→ The life cycle of service workers involves:

1) Registration: Developers register a service-worker script in the main JS file of web application using the navigator service workers register() method.

2) Installation: During the installation phase, the browser downloads the service worker script

FOR EDUCATIONAL USE

specified in the registration step. The service worker is then installed in the background, but it doesn't get control any pages.

3) Activation: After the service-worker is installed, it goes through the activation phase. During activation, the service-worker can start controlling pages.

Throughout it's life cycle, a service-worker operates independently of any specific web page making it a powerful tool for building web applications.

Q4 Explain the use of indexed DB in the service-worker for data storage.

→ Indexed DB is a low level API for client-side storage of significant amounts of structured data, including files/blobs.

It is used in service worker as:
1) Persistent storage: Indexed DB provides persistent storage, meaning data stored in indexed DB persists even when the browser is closed or the device is restarted.

2) Asynchronous operation - Indexed DB operates

FOR EDUCATIONAL USE

Prachit P. Parulekar

Prachit P. Parulekar

asynchronously, allowing the service worker to perform database operations without blocking the main thread.

3) Key-value storage: Stores data in key-value pairs also supports more complex data structures like object and array.

4) Querying and indexing: Indexed DB supports indexing enabling efficient querying of stored data.

5) Data transcn: Indexed DB transactions ensure data integrity by providing ACID properties (atomicity, consistency, ~~integrity~~ isolation and durability).

Overall indexed DB in service workers enables enhanced performance and deliver seamless user experience.