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 **Mahek Bhagwani** 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

Name of the Course : MAD & PWA Lab

Course Code : ITL604

Project Title: MAD: Zoom Clone/ PWA:fuzzy **Roll No.6**

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

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers **:** Mrs. Kajal Joseph

Email **:** kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: MAD: Zoom Clone/ PWA:fuzzy **Roll No.6**

Lab Objectives:

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

Lab Outcomes:

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

Index

Sr. No	Experiment Title	LO	DOP	DOS	Grade
1.	To install and configure the Flutter Environment	LO1	16/1	23/1	15
2.	To design Flutter UI by including common widgets.	LO2	23/1	30/1	15
3.	To include icons, images, fonts in Flutter app	LO2	30/1	6/2	15
4.	To create an interactive Form using form widget	LO2	6/2	13/2	15
5.	To apply navigation, routing and gestures in Flutter App	LO2	13/2	20/2	15
6.	To Connect Flutter UI with fireBase database	LO3	20/2	5/3	15
7.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4	5/3	12/3	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	12/3	19/3	15
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5	19/3	26/3	15
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5	26/3	2/4	15
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6	5/3	12/3	15
12.	Assignment-1	LO1,LO2 ,LO3	2/2	5/2	5
13.	Assignment-2	LO4,LO5 ,LO6	19/3	21/3	4

MAD & PWA Lab Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	6
Name	Mahek Bhagwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15M

EXP 1: Installation and Configuration of Flutter Environment.

Install the Flutter SDK

Step 1: Download the installation bundle of the Flutter Software Development Kit for windows. To download Flutter SDK, Go to its official website https://docs.flutter.dev/get-started/install , you will get the following screen.

Step 2: Next, to download the latest Flutter SDK, click on the Windows **icon**. Here, you will find the download link for SDK.

Step 3: When your download is complete, extract the **zip** file and place it in the desired installation folder or location, for example, C: /Flutter.

Step 4: To run the Flutter command in regular windows console, you need to update the system path to include the flutter bin directory. The following steps are required to do this:

Step 4.1: Go to MyComputer properties -> advanced tab -> environment variables. You will get the following screen.

Step 4.2: Now, select path -> click on edit. The following screen appears

Step 4.3: In the above window, click on New->write path of Flutter bin folder in variable value -> ok -> ok -> ok.

Step 5: Now, run the $ **flutter** command in command prompt.

Now, run the $ **flutter doctor** command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation.

Step 6: When you run the above command, it will analyze the system and show its report, as shown in the below image. Here, you will find the details of all missing tools, which required to run Flutter as well as the development tools that are available but not connected with the device.

Step 7: Install the Android SDK. If the flutter doctor command does not find the Android SDK tool in your system, then you need first to install the Android Studio IDE. To install Android Studio IDE, do the following steps.

Step 7.1: Download the latest Android Studio executable or zip file from the official site.

Step 7.2: When the download is complete, open the **.exe** file and run it. You will get the following dialog box.

Step 7.3: Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.

Step 7.4: In the above screen, click Next-> Finish. Once the Finish button is clicked, you need to choose the 'Don't import Settings option' and click OK. It will start the Android Studio.

Step 7.5 run the $ **flutter doctor** command and Run flutter doctor --android-licenses command.

Step 8: Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application.

Step 8.1: To set an Android emulator, go to Android Studio > Tools > Android > AVD Manager and select Create Virtual Device. Or, go to Help->Find Action->Type Emulator in the search box. You will get the following screen.

Step 8.2: Choose your device definition and click on Next.

Step 8.3: Select the system image for the latest Android version and click on Next.

Step 8.4: Now, verify the all AVD configuration. If it is correct, click on Finish. The following screen appears.

Step 8.5: Last, click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen.

Step 9: Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself. Do the following steps to install these plugins.

Step 9.1: Open the Android Studio and then go to File->Settings->Plugins.

Step 9.2: Now, search the Flutter plugin. If found, select Flutter plugin and click install. When you click on install, it will ask you to install Dart plugin as below screen. Click yes to proceed.

Step 9.3: Restart the Android Studio.

Experiment 2: Create a 'Hello World App' using Flutter

Step 1 : Create the app

1. Open the IDE and select **Create New Flutter Project**.
2. Select **Flutter Application** as the project type. Then click **Next**.
3. Verify the Flutter SDK path specifies the SDK‟s location (select **Install SDK…** if the text field is blank).
4. Enter a project name (for example, myapp). Then click **Next**.
5. Click **Finish**.
6. Wait for Android Studio to install the SDK and create the project.

Step 2: Run the app

1. Locate the main Android Studio toolbar:

2. In the **target selector**, select an Android device for running the app. If none are listed as available, select **Tools > AVD Manager** and create one there.

Project Title: Zoom Clone / Fuzzy Roll No:6

3. Click the run icon in the toolbar, or invoke the menu item **Run** > **Run**.

Step 3 : Creating Hello world app

```dart
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'),
        ),  // AppBar
        body: const Center(
          child: Text('Hello Mahek Bhagwani'),
        ),  // Center
      ),  // Scaffold
    );  // MaterialApp
  }
}
```

Project Title: Zoom Clone / Fuzzy Roll No:6

1:1
1,235x2,646 PNG (32-bit color) 717.73 kB
7:54
DEBUG
Welcome to Flutter
Hello Mahek Bhagwani

MAD & PWA Lab Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	6
Name	Mahek Bhagwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15M

Experiment No: 2

Aim: To Design flutter UI by including common widgets.

Code:

```
import 'package:flutter/material.dart';
import 'package:uuid/uuid.dart';
import 'package:zoom_meet/resources/jitsi_meet_methods.dart';
import 'package:zoom_meet/utils/colors.dart';
import 'package:zoom_meet/widgets/home_meeting_button.dart';

class MeetingScreen extends StatelessWidget {
 MeetingScreen({super.key});

 final JitsiMeetMethods _jitsiMeetMethods = JitsiMeetMethods();

 createNewMeeting() async {
   String roomName = const Uuid().v1().substring(0, 10);
   _jitsiMeetMethods.createMeeting(
     roomName: roomName,
     isAudioMuted: true,
     isVideoMuted: true,
   );
 }

 joinMeeting(BuildContext context) {
   Navigator.pushNamed(context, '/videocall');
 }

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       centerTitle: true,
       backgroundColor: backgroundColor,
       title: const Text('Meet & Chat'),
     ),
     body: Column(
       children: [
         const SizedBox(height: 30),
         Row(
           mainAxisAlignment: MainAxisAlignment.center,
           children: [
```

```
            HomeMeetingButton(
              onPressed: createNewMeeting,
              text: 'New Meeting',
              icon: Icons.videocam,
            ),
            HomeMeetingButton(
              onPressed: () => joinMeeting(context),
              text: 'Join Meeting',
              icon: Icons.add_box_rounded,
            ),
            HomeMeetingButton(
              onPressed: () {},
              text: 'Schedule',
              icon: Icons.calendar_today_rounded,
            ),
          ],import 'package:flutter/material.dart';
import 'package:uuid/uuid.dart';
import 'package:zoom_meet/resources/jitsi_meet_methods.dart';
import 'package:zoom_meet/utils/colors.dart';
import 'package:zoom_meet/widgets/home_meeting_button.dart';

class MeetingScreen extends StatelessWidget {
  MeetingScreen({super.key});

  final JitsiMeetMethods _jitsiMeetMethods = JitsiMeetMethods();

  createNewMeeting() async {
    String roomName = const Uuid().v1().substring(0, 10);
    _jitsiMeetMethods.createMeeting(
      roomName: roomName,
      isAudioMuted: true,
      isVideoMuted: true,
    );
  }

  joinMeeting(BuildContext context) {
    Navigator.pushNamed(context, '/videocall');
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        centerTitle: true,
```

```
      backgroundColor: backgroundColor,
      title: const Text('Meet & Chat'),
    ),
    body: Column(
      children: [
        const SizedBox(height: 30),
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            HomeMeetingButton(
              onPressed: createNewMeeting,
              text: 'New Meeting',
              icon: Icons.videocam,
            ),
            HomeMeetingButton(
              onPressed: () => joinMeeting(context),
              text: 'Join Meeting',
              icon: Icons.add_box_rounded,
            ),
            HomeMeetingButton(
              onPressed: () {},
              text: 'Schedule',
              icon: Icons.calendar_today_rounded,
            ),
          ],
        ),
        const Expanded(
          child: Center(
            child: Text(
              'Create/Join Meetings with Just a Click!',
              style: TextStyle(fontWeight: FontWeight.bold, fontSize: 18),
            ),
          ),
        ),
      ],
    ),
  );
 }
}

      ),
      const Expanded(
        child: Center(
          child: Text(
```

```
            'Create/Join Meetings with Just a Click!',
            style: TextStyle(fontWeight: FontWeight.bold, fontSize: 18),
          ),
        ),
      ),
    ],
  ),
);
}
}
```

Output:

Project Title: Zoom Clone/Fuzzy **Roll No:6**

MAD & PWA Lab Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	6
Name	Mahek Bhagwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15M

Project Title: Zoom Clone/Fuzzy **Roll No:6**

Experiment No:3

Aim: To include icons image and fonts in flutter app

Code:

```
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: Center(
child: Image.asset('assets/comp.jpg'),
),
),
);
}
}
```

Output:

Project Title: Zoom Clone/Fuzzy **Roll No:6**

MAD & PWA Lab Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	6
Name	Mahek Bhagwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15M

Experiment No : 4

Aim: To create an interactive Form using form widget

Code:

```
// ignore_for_file: use_build_context_synchronously

import 'package:flutter/material.dart';
import 'package:zoom_meet/resources/auth_methods.dart';
import 'package:zoom_meet/widgets/custom_button.dart';

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

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

class _LoginScreenState extends State<LoginScreen> {
 final AuthMethods _authMethods = AuthMethods();
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     body: Column(
       mainAxisAlignment: MainAxisAlignment.center,
       children: [
         const Text(
           'Create or Join Meeting',
           style: TextStyle(
             fontSize: 24,
             fontWeight: FontWeight.bold,
           ),
         ),
         Padding(
           padding: const EdgeInsets.symmetric(vertical: 40.0),
           child: Image.asset('assets/images/onboarding.jpg'),
         ),
         CustomButton(
           text: 'Google Sign In',
           onPressed: () async {
             bool res = await _authMethods.signInWithGoogle(context);
             if (res) {
               Navigator.pushNamed(context, '/homescreen');
             }
           },
```

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

Output:

Project Title: Zoom Clone **Roll No:6**

MAD & PWA Lab Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	6
Name	Mahek Bhagwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15M

Project Title: Zoom Clone **Roll No:6**

Experiment No:5

Aim:To apply navigation routing and gestures in Flutter App.

Code:
Meeting History Page:

```
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:zoom_meet/resources/firestore_methods.dart';
import 'package:zoom_meet/utils/colors.dart';

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

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       centerTitle: true,
       backgroundColor: backgroundColor,
       title: const Text('Meeting History'),
     ),
     body: StreamBuilder(
         stream: FirestoreMethods().meetingHistory,
         builder: (context, snapshot) {
           if (snapshot.connectionState == ConnectionState.waiting) {
             return const Center(child: CircularProgressIndicator());
           }

           return ListView.builder(
             itemCount: (snapshot.data! as dynamic).docs.length,
             itemBuilder: (context, index) => ListTile(
               title: Text(
                 'Room Id: ${(snapshot.data! as dynamic).docs[index]['meetingName']}',
               ),
               subtitle: Text(
                 'Joined On: ${DateFormat.yMMMMd().format((snapshot.data! as
dynamic).docs[index]['createdAt'].toDate())}',
               ),
               trailing: IconButton(
                 onPressed: () {
                   FirestoreMethods().deleteMeetingHistory(snapshot.data!.docs[index].id);
                 },
                 icon: const Icon(Icons.delete),
```

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

Settings Screen:

```
import 'package:flutter/material.dart';
import 'package:zoom_meet/resources/auth_methods.dart';
import 'package:zoom_meet/utils/colors.dart';
import 'package:zoom_meet/widgets/custom_button.dart';

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

 @override
 Widget build(BuildContext context) {
   final AuthMethods _authMethods = AuthMethods();
   return Scaffold(
     appBar: AppBar(
       centerTitle: true,
       backgroundColor: backgroundColor,
       title: const Text('Settings'),
     ),
     body: Column(
       mainAxisAlignment: MainAxisAlignment.center,
       crossAxisAlignment: CrossAxisAlignment.center,
       children: [
         CircleAvatar(
           radius: 60,
           backgroundImage: NetworkImage(_authMethods.user.photoURL!),
         ),
         const SizedBox(height: 20),
         Text(
           _authMethods.user.displayName!,
           style: const TextStyle(
             fontWeight: FontWeight.bold,
             fontSize: 24,
           ),
         ),
         Text('Email: ${_authMethods.user.email}'),
         const SizedBox(height: 40),
```

```
        CustomButton(
          text: 'Log Out',
          onPressed: () => AuthMethods().logOut(),
        ),
      ],
    ),
  );
}
}
```

Outputs:

Project Title: Zoom Clone **Roll No:6**

Project Title: Zoom Clone **Roll No:6**

Conclusion: In this experiment I learnt about navigation and routing in flutter and successfully implemented it in my project/application.

MAD & PWA Lab Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	6
Name	Mahek Bhagwani
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:	15M

EXPERIMENT NO: 06

Aim: To Connect Flutter UI with firebase database

Theory:

Firebase is a Backend-as-a-Service (BaaS) app development platform that provides hosted backend services such as a realtime database, cloud storage, authentication, crash reporting, machine learning, remote configuration, and hosting for your static files.

CODE:

Firebase auth:

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

class FirestoreMethods {
 final FirebaseFirestore _firestore = FirebaseFirestore.instance;
 final FirebaseAuth _auth = FirebaseAuth.instance;
 Stream<QuerySnapshot<Map<String, dynamic>>> get meetingHistory => _firestore
     .collection('users')
     .doc(_auth.currentUser!.uid)
     .collection('meetings')
     .orderBy('createdAt', descending: true)
     .snapshots();
 void addToMeetingHistory(String meetingName) async {
   try {
     await _firestore
         .collection('users')
         .doc(_auth.currentUser!.uid)
         .collection('meetings')
         .add(
       {
         'meetingName': meetingName,
         'createdAt': DateTime.now(),
       },
     );
   } catch (e) {
     print(e);
   }
 }

 void deleteMeetingHistory(String id) async {
```

```
  try {
    await _firestore
        .collection('users')
        .doc(_auth.currentUser!.uid)
        .collection('meetings')
        .doc(id)
        .delete();
  } catch (e) {}
 }
}
```

Jitsi meet package methods:

```
import 'package:jitsi_meet_flutter_sdk/jitsi_meet_flutter_sdk.dart';
import 'package:zoom_meet/resources/auth_methods.dart';
import 'package:zoom_meet/resources/firestore_methods.dart';

class JitsiMeetMethods {
 final AuthMethods _authMethods = AuthMethods();
 final FirestoreMethods _firestoreMethods = FirestoreMethods();
 void createMeeting({
   required String roomName,
   required bool isAudioMuted,
   required bool isVideoMuted,
   String userName = '',
 }) async {
   try {
     String name;
     if (userName.isEmpty) {
       name = _authMethods.user.displayName!;
     } else {
       name = userName;
     }
     var options = JitsiMeetConferenceOptions(
       room: roomName,
       userInfo: JitsiMeetUserInfo(
         displayName: name,
         email: _authMethods.user.email,
         avatar: _authMethods.user.photoURL,
       ),
       configOverrides: {
         "startWithAudioMuted": isAudioMuted,
         "startWithVideoMuted": isVideoMuted,
       },
     );
```

```
    _firestoreMethods.addToMeetingHistory(roomName);
    var jitsiMeet = JitsiMeet();
    await jitsiMeet.join(options);
  } catch (e) {
    print(e.toString());
  }
 }
}
```

Output:

Project Title: Zoom Clone **Roll No:6**

Project Title: Zoom Clone **Roll No:6**

Project Title: Zoom Clone **Roll No:6**

Firebase Console:

Firebase Authentication:

I have created a firebase database syncing to my project
This includes all the user details like user id user name profile photo.

Then in users you can check their meeting history that which users created how many meetings it will store the meeting id

Project Title: Zoom Clone **Roll No:6**

Ahead after clicking on a particular meeting you can get more information about that particular meeting like meeting name, when was it created how many minutes the meeting continued

Conclusion: Here, Successfully Connected Flutter UI with Firebase database

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.	6
Name	Mahek Bhagwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	15M

Project Title: Fuzzy **Roll No:6**

EXPERIMENT NO: 07

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.

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

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

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

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

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

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

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

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

Linkable — Easily shared via URL without complex installations.

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

Weaknesses refer to:
IOS support from version 11.3 onwards;
Greater use of the device battery;

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

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
    <meta name="description" content="fuzzy" />
    <meta name="keywords" content="fuzzy" />
    <meta name="author" content="fuzzy" />
    <link rel="manifest" href="manifest.json" />
    <link rel="icon" href="assets/images/logo/favicon.png"
type="image/x-icon" />
    <title>fuzzy</title>
    <link rel="apple-touch-icon"
href="assets/images/logo/favicon.png" />
    <meta name="theme-color" content="#122636" />
    <meta name="apple-mobile-web-app-capable" content="yes"
/>
    <meta name="apple-mobile-web-app-status-bar-style"
content="black" />
    <meta name="apple-mobile-web-app-title" content="fuzzy"
/>
    <meta name="msapplication-TileImage"
content="assets/images/logo/favicon.png" />
```

```
    <meta name="msapplication-TileColor" content="#FFFFFF"
/>
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <!--Google font-->
    <link rel="preconnect"
href="https://fonts.googleapis.com/" />
    <link rel="preconnect"
href="https://fonts.gstatic.com/" crossorigin />
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght
@100;200;300;400;500;600;700;800;900&amp;display=swap"
rel="stylesheet" />

    <!-- iconsax css -->
    <link rel="stylesheet" type="text/css"
href="assets/css/vendors/iconsax.css" />

    <!-- bootstrap css -->
    <link rel="stylesheet" id="rtl-link" type="text/css"
href="assets/css/vendors/bootstrap.min.css" />

    <!-- swiper css -->
    <link rel="stylesheet" type="text/css"
href="assets/css/vendors/swiper-bundle.min.css" />

    <!-- Theme css -->
    <link rel="stylesheet" id="change-link" type="text/css"
href="assets/css/style.css" />
  </head>

  <body class="auth-body dark">
    <!-- onboarding section start -->
    <section class="section-b-space">
      <div class="swiper intro slider-1">
```

```
        <div class="swiper-wrapper">
          <div class="swiper-slide">
            <div class="theme-logo pb-3">
              <img class="img-fluid logo-img"
src="assets/images/logo/logo.png" alt="logo" />
            </div>
            <div class="onboarding-design">
              <img class="img-fluid design-img"
src="assets/images/onboarding/design1.png" alt="bg-design"
/>

              <img class="img-fluid slider-img1"
src="assets/images/onboarding/1.png" alt="slider-1" />

              <img class="img-fluid vector1"
src="assets/images/onboarding/vector1.png" alt="v1" />
              <img class="img-fluid vector2"
src="assets/images/onboarding/vector2.png" alt="v2" />
              <img class="img-fluid vector3"
src="assets/images/onboarding/vector3.png" alt="v3" />
            </div>
            <div class="product-details">
              <h1>Office Furniture</h1>
              <span></span>
              <p>The best products that suits your
lifestyle with visually appealing designs.</p>
              <div class="redirate-btn">
                <a href="login.html" class="next-arrow">
                  <i class="iconsax right-arrow" data-
icon="arrow-right"></i>
                </a>
              </div>
            </div>
          </div>
```

```
          <div class="swiper-slide">
            <div class="theme-logo pb-3">
              <img class="img-fluid logo-img"
src="assets/images/logo/logo.png" alt="logo" />
            </div>
            <div class="onboarding-design">
              <img class="img-fluid design-img"
src="assets/images/onboarding/design1.png" alt="bg-design"
/>

              <img class="img-fluid slider-img2"
src="assets/images/onboarding/2.png" alt="slider-2" />

              <img class="img-fluid vector1"
src="assets/images/onboarding/vector1.png" alt="v1" />
              <img class="img-fluid vector2"
src="assets/images/onboarding/vector2.png" alt="v2" />
              <img class="img-fluid vector3"
src="assets/images/onboarding/vector3.png" alt="v3" />
            </div>
            <div class="product-details">
              <h1>Relaxing Furniture</h1>
              <span></span>
              <p>The best funiture that fits your House and
workspace.</p>
              <div class="redirate-btn">
                <a href="login.html" class="next-arrow">
                  <i class="iconsax right-arrow" data-
icon="arrow-right"></i>
                </a>
              </div>
            </div>
          </div>

          <div class="swiper-slide">
```

```
            <div class="theme-logo pb-3">
              <img class="img-fluid logo-img"
src="assets/images/logo/logo.png" alt="logo" />
            </div>
            <div class="onboarding-design">
              <img class="img-fluid design-img"
src="assets/images/onboarding/design1.png" alt="bg-design"
/>

              <img class="img-fluid slider-img3"
src="assets/images/onboarding/3.png" alt="slider-3" />

              <img class="img-fluid vector1"
src="assets/images/onboarding/vector1.png" alt="v1" />
              <img class="img-fluid vector2"
src="assets/images/onboarding/vector2.png" alt="v2" />
              <img class="img-fluid vector3"
src="assets/images/onboarding/vector3.png" alt="v3" />
              <div class="product-details">
                <h1>Home Decor</h1>
                <span></span>
                <p>The best payment method connects your
money to friends, family, brands, and experiences.</p>
                <div class="redirate-btn">
                  <a href="login.html" class="next-arrow">
                    <i class="iconsax right-arrow" data-
icon="arrow-right"></i>
                  </a>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </section>
```

```
    <!-- onboarding section end -->

    <!-- pwa install app popup start -->
    <div class="offcanvas offcanvas-bottom addtohome-popup
theme-offcanvas" tabindex="-1" id="offcanvas">
      <button type="button" class="btn-close text-reset"
data-bs-dismiss="offcanvas" aria-label="Close"></button>
      <div class="offcanvas-body small">
        <div class="app-info">
          <img src="assets/images/logo/48.png" class="img-
fluid" alt="" />
          <div class="content">
            <h4>fuzzy app</h4>
            <a href="#">www.fuzzy-app.com</a>
          </div>
        </div>
        <a href="#!" class="btn theme-btn install-app btn-
inline home-screen-btn m-0" id="installapp">Add to Home
Screen</a>
      </div>
    </div>
    <!-- pwa install app popup start -->

    <!-- swiper js -->
    <script src="assets/js/swiper-bundle.min.js"></script>
    <script src="assets/js/custom-swiper.js"></script>

    <!-- iconsax js -->
    <script src="assets/js/iconsax.js"></script>

    <!-- bootstrap js -->
    <script
src="assets/js/bootstrap.bundle.min.js"></script>
```

```html
    <!-- homescreen popup js -->
    <script src="assets/js/homescreen-popup.js"></script>

    <!-- PWA offcanvas popup js -->
    <script src="assets/js/offcanvas-popup.js"></script>

    <!-- script js -->
    <script src="assets/js/script.js"></script>
      <script async
src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js?client=ca-pub-5645451029544050"
     crossorigin="anonymous"></script>
     <script src="/app.js"></script>
  </body>
</html>
```

OUTPUT:

Open folder in VS code and click go live at bottom right corner

Open your hosted site on Microsoft Edge

Click on 3 dots
Click on Apps
Select install app option

Desktop App created Successfully

Open Desktop App

Conclusion: Here, We have Successfully created a basic progressive web app of our web page and installed it inour desktop successfully.

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.	6
Name	Mahek Bhagwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15M

Project Title: Fuzzy **Roll No:6**

Experiment No:08

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.

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('/ser
  vice-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.

Code

Index.html

manifest.json M | app.js U | service-worker.js U | index.html M

index.html > html > body.auth-body.dark > script

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="fuzzy" />
    <meta name="keywords" content="fuzzy" />
    <meta name="author" content="fuzzy" />
    <link rel="manifest" href="manifest.json" />
    <link rel="icon" href="assets/images/logo/favicon.png" type="image/x-icon" />
    <title>fuzzy</title>
    <link rel="apple-touch-icon" href="assets/images/logo/favicon.png" />
    <meta name="theme-color" content="#122636" />
    <meta name="apple-mobile-web-app-capable" content="yes" />
    <meta name="apple-mobile-web-app-status-bar-style" content="black" />
    <meta name="apple-mobile-web-app-title" content="fuzzy" />
    <meta name="msapplication-TileImage" content="assets/images/logo/favicon.png" />
    <meta name="msapplication-TileColor" content="#FFFFFF" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

```html
</head>

<body class="auth-body dark">
  <!-- onboarding section start -->
  <section class="section-b-space">
    <div class="swiper intro slider-1">
      <div class="swiper-wrapper">
        <div class="swiper-slide">
          <div class="theme-logo pb-3">
            <img class="img-fluid logo-img" src="assets/images/logo/logo.png" alt="logo" />
          </div>
          <div class="onboarding-design">
            <img class="img-fluid design-img" src="assets/images/onboarding/design1.png" alt="b

            <img class="img-fluid slider-img1" src="assets/images/onboarding/1.png" alt="slider

            <img class="img-fluid vector1" src="assets/images/onboarding/vector1.png" alt="v1"
            <img class="img-fluid vector2" src="assets/images/onboarding/vector2.png" alt="v2"
            <img class="img-fluid vector3" src="assets/images/onboarding/vector3.png" alt="v3"
          </div>
          <div class="product-details">
            <h1>Office Furniture</h1>
            <span></span>
```

```html
<!-- pwa install app popup start -->
<div class="offcanvas offcanvas-bottom addtohome-popup theme-offcanvas" tabindex="-1" id="offc
  <button type="button" class="btn-close text-reset" data-bs-dismiss="offcanvas" aria-label="C
  <div class="offcanvas-body small">
    <div class="app-info">
      <img src="assets/images/logo/48.png" class="img-fluid" alt="" />
      <div class="content">
        <h4>fuzzy app</h4>
        <a href="#">www.fuzzy-app.com</a>
      </div>
    </div>
    <a href="#!" class="btn theme-btn install-app btn-inline home-screen-btn m-0" id="installa
  </div>
</div>
<!-- pwa install app popup start -->

<!-- swiper js -->
<script src="assets/js/swiper-bundle.min.js"></script>
<script src="assets/js/custom-swiper.js"></script>

<!-- iconsax js -->
<script src="assets/js/iconsax.js"></script>
```

Style.css

```css
@-webkit-keyframes fireworkLine {
  0% {
    right: 20%;
    -webkit-transform: scale(0, 0);
            transform: scale(0, 0);
  }
  25% {
    right: 20%;
    width: 6px;
    -webkit-transform: scale(1, 1);
            transform: scale(1, 1);
  }
  35% {
    right: 0;
    width: 35%;
  }
  70% {
    right: 0;
    width: 4px;
    -webkit-transform: scale(1, 1);
            transform: scale(1, 1);
  }
  100% {
    right: 0;
    -webkit-transform: scale(0, 0);
            transform: scale(0, 0);
```

```css
@keyframes fireworkLine {
  0% {
    right: 20%;
    -webkit-transform: scale(0, 0);
            transform: scale(0, 0);
  }
  25% {
    right: 20%;
    width: 6px;
    -webkit-transform: scale(1, 1);
            transform: scale(1, 1);
  }
  35% {
    right: 0;
    width: 35%;
  }
  70% {
    right: 0;
    width: 4px;
    -webkit-transform: scale(1, 1);
            transform: scale(1, 1);
  }
  100% {
    right: 0;
    -webkit-transform: scale(0, 0);
```

App.js

manifest.json M | app.js U | service-worker.js U | index.html M | style.css 5

app.js > ...

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

Service-Worker.js

manifest.json M | app.js U | service-worker.js U | index.html M

service-worker.js > assetsToCache

```js
const cacheName = 'fuzzy_app';
const assetsToCache = [
'/',
'/index.html',
'/assets/css/style.css',
'/app.js'
]
self.addEventListener('install', event => {
event.waitUntil(
caches.open(cacheName)
.then(cache => {
return cache.addAll(assetsToCache);
})
);
});
self.addEventListener('activate', event => {
event.waitUntil(
caches.keys().then(cacheNames => {
return Promise.all(
cacheNames.filter(name => {
return name !== cacheName;
}).map(name => {
return caches.delete(name);
})
);
})
```

OUTPUT:

127.0.0.1:5500/index.html
FUZZY
Office Furniture
The best products that suits your lifestyle with visually appealing designs.
Welcome
Elements
Console
Sources
Network
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
IndexedDB
Web SQL
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
fuzzy_app - http://127
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigati
Notifications
Payment handler
Periodic background syn
Speculative loads
Push messaging
Reporting API
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Network requests
Update
Unregister
Source
service-worker.js
Received 1/4/2024, 1:12:49 pm
Status
#351 activated and is running
stop
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic Sync
test-tag-from-devtools
Periodic Sync
Update Cycle
Version
Update Activity
Timeline
#351
Install
#351
Wait
#351
Activate
Service workers from other origins
See all registrations
Console
Issues
3D View
top
Filter
Default levels
13

Welcome
Elements
Console
Sources
Network
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
IndexedDB
Web SQL
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigati
Notifications
Payment handler
Periodic background syn
Speculative loads
Push messaging
Reporting API
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Network requests
Update
Source
service-worker.js
Received 1/4/2024, 1:12:49 pm
Status
#351 activated and is running
stop
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic Sync
test-tag-from-devtools
Periodic Sync
Update Cycle
Version
Update Activity
Timeline
#351
Install
#351
Wait
#351
Activate
Service workers from other origins
See all registrations

Cache Storage

Conclusion: In this experiment, we have registered a service worker, and completed the install and activation process for a new service worker for the E-commerce PW

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.	6
Name	Mahek Bhagwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15M

Project Title: Fuzzy **Roll No:6**

Experiment No: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.

Code:

Service-Worker.js

```
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.requesst));
   });
   self.addEventListener("sync", (event) => {
   if (event.tag === "syncMessage") {
   console.log("Sync successful!");
   }
   });
   self.addEventListener("push", function (event) {
   if (event && event.data) {
   try {
   var data = event.data.json();
   if (data && data.method === "pushMessage") {
   console.log("Push notification sent");
   self.registration.showNotification("Ecommerce website", {
   body: data.message,
   });
   }
   } catch (error) {
   console.error("Error parsing push data:", error);
   }
   }
   });
   var preLoad = function () {
   return caches.open("offline").then(function (cache) {
```

```
// caching index and important routes
return cache.addAll([
"/",
"/index.html",
"/landing.html",
"/profile.html",
"/shop.html",
"/cart.html",
"/css/main.css",
]);
});
};
var checkResponse = function (request) {
return new Promise(function (fulfill, reject) {
fetch(request)
.then(function (response) {
if (response.status !== 404) {
fulfill(response);
} else {
reject(new Error("Response not found"));
}
})
.catch(function (error) {
reject(error);
});
});
};
var returnFromCache = function (request) {
return caches.open("offline").then(function (cache) {
return cache.match(request).then(function (matching) {
if (!matching || matching.status == 404) {
return cache.match("offline.html");
} else {
return matching;
}
});
});
```

```
};
var addToCache = function (request) {
return caches.open("offline").then(function (cache) {
return fetch(request).then(function (response) {
return cache.put(request, response.clone()).then(function () {
return response;
});
});
});
};
```

Output:

Offline

Conclusion : In this experiment, we have successfully implemented service worker events like fetch, sync and push for E-commerce 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.	6
Name	Mahek Bhagwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15M

Project Title: Fuzzy **Roll No:6**

Experiment No. 10

Aim:

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

Theory:

GitHub Pages

Public web pages are freely hosted and easily published. Public webpages hosted directly from your GitHub repository. Just edit, push, and your changes are live.

GitHub Pages provides the following key features:

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

Reasons for favoring this over Firebase:

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

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

Pros

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

Cons

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

Firebase

The Realtime App Platform. Firebase is a cloud service designed to power real-time, collaborative applications. Simply add the Firebase library to your application to gain access to a shared data structure; any changes you make to that data are automatically synchronized with the Firebase cloud and with other clients within milliseconds.

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

Pros

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

Cons

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

Link to our GitHub repository:https://github.com/maheek07/Fuzzy-app

Hosted Link: https://shamaila02.github.io/PWA_Fuzzy/

Github Screenshot:

Conclusion: In this experiment we have Successfully deployed the Ecommerce PWA to GitHub Pages.

MAD & PWA Lab Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	6
Name	Mahek Bhagwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	15M

Project Title: Fuzzy **Roll No:6**

Experiment 11

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

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

Google Lighthouse :

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

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

Key Features and Audit Metrics

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

1. **Performance:** This score is an aggregation of how the page fared in aspects such as (but not limited to) loading speed, time taken for loading for basic frame(s), displaying meaningful content to the user, etc. To a layman, this score is indicative of how decently the site performs, with a score of 100 meaning that you figure in the 98th percentile, 50 meaning that you figure in the 75th percentile and so on.
2. **PWA Score (Mobile):** Thanks to the rise of Service Workers, app manifests, etc., a lot of modern web applications are moving towards the PWA paradigm, where the objective is to make the application behave as close as possible to native mobile applications. Scoring

points are based on the Baseline PWA checklist laid down by Google which includes Service Worker implementation(s), viewport handling, offline functionality, performance in script-disabled environments, etc.

3. **Accessibility:** As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics though, Accessibility metrics score on a pass/fail basis i.e. if all possible elements of the page are not screen-reader friendly (HTML5 introduced features that would make pages easy to interpret for screen readers used by visually challenged people like tag names, tags such as <section>, <article>, etc.), you get a 0 on that score. The aggregate of these scores is your Accessibility metric score.
4. **Best Practices:** As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS

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

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

Code&Implementation:

Index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="fuzzy" />
    <meta name="keywords" content="fuzzy" />
    <meta name="author" content="fuzzy" />
    <link rel="manifest" href="manifest.json" />
    <link rel="icon" href="assets/images/logo/favicon.png" type="image/x-icon" />
    <title>fuzzy</title>
    <link rel="apple-touch-icon" href="assets/images/logo/favicon.png" />
    <meta name="theme-color" content="#122636" />
    <meta name="apple-mobile-web-app-capable" content="yes" />
    <meta name="apple-mobile-web-app-status-bar-style" content="black" />
    <meta name="apple-mobile-web-app-title" content="fuzzy" />
    <meta name="msapplication-TileImage" content="assets/images/logo/favicon.png" />
    <meta name="msapplication-TileColor" content="#FFFFFF" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

Style.css

```css

body {
    font-family: "Poppins", sans-serif;
    max-width: 600px;
    width: 100%;
    margin: 0 auto;
    height: 100vh;
    background-color: rgba(var(--white), 1);
}
body::-webkit-scrollbar {
    width: 0;
}

h1 {
    font-weight: 600;
    font-size: 20px;
    line-height: 29px;
    margin-bottom: 0;
}

h2 {
    font-size: 16px;
    font-weight: 600;
    margin-bottom: 0;
}
```

Script.js

```
document.getElementById("loginForm").addEventListen
er("submit",(event)=>{
    event.preventDefault()
})

firebase.auth().onAuthStateChanged((user)=>{
    if(user){
        location.replace("/landing.html")
    }
})

function login(){
                            const       email       =
document.getElementById("email").value
                          const      password      =
document.getElementById("password").value

firebase.auth().signInWithEmailAndPassword(email,
password)
    .catch((error)=>{
          document.getElementById("error").innerHTML
= error.message
    })
}

function signUp(){
                        const      email_login      =
```

```
document.getElementById("email_login").value
                    const     password_login     =
document.getElementById("password_login").value

firebase.auth().createUserWithEmailAndPassword(emai
l, password)
    .catch((error) => {
          document.getElementById("error").innerHTML
= error.message
    });
}

function forgotPass(){
                          const      email       =
document.getElementById("email").value
    firebase.auth().sendPasswordResetEmail(email)
    .then(() => {
        alert("Reset link sent to your email id")
    })
    .catch((error) => {
          document.getElementById("error").innerHTML
= error.message
    });
}
```

Manifest.json

```
{
  "name": "Fuzzy app",
  "short_name": "Fuzzy app",
```

```
  "lang": "en-US",
  "start_url": ".",
  "display": "standalone",
  "background_color": "white",
  "theme_color": "#122636",
  "description": "fuzzy app",
  "icons": [
    {
      "src": "assets/images/logo/512.png",
      "sizes": "144x144",
      "type": "image/png",
      "purpose": "any"
    },

    {
      "src": "assets/images/logo/512.png",
      "sizes": "512x512",
      "type": "image/png",
      "purpose": "any maskable"
    },
    {
      "src": "assets/images/icon.png",
      "sizes": "144x144",
      "type": "image/png",
      "purpose": "any maskable"
    }

  ]
```

Output:

For Desktop Devices

For Mobile Devices

Report Generated

53 88 93 100

88

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so manual testing is also encouraged.

ARIA

- Elements with `role="dialog"` or `role="alertdialog"` do not have accessible names.

These are opportunities to improve the usage of ARIA in your application which may enhance the experience for users of assistive technology, like a screen reader.

NAMES AND LABELS

- Links do not have a discernible name

These are opportunities to improve the semantics of the controls in your application. This may enhance the experience for users of assistive technology, like a screen reader.

53 88 93 100

93

Best Practices

USER EXPERIENCE

- Serves images with low resolution

GENERAL

- Browser errors were logged to the console

TRUST AND SAFETY

- Ensure CSP is effective against XSS attacks

PASSED AUDITS (12) Show

NOT APPLICABLE (2) Show

100

SEO

These checks ensure that your page is following basic search engine optimization advice. There are many additional factors Lighthouse does not score here that may affect your search ranking, including performance on Core Web Vitals. Learn more about Google Search Essentials.

ADDITIONAL ITEMS TO MANUALLY CHECK (1) Hide

- Structured data is valid

Run these additional validators on your site to check additional SEO best practices.

PASSED AUDITS (12) Show

NOT APPLICABLE (2) Show

PWA

These checks validate the aspects of a Progressive Web App. Learn what makes a good Progressive Web App.

INSTALLABLE

- Web app manifest and service worker meet the installability requirements

PWA OPTIMIZED

- Configured for a custom splash screen
- Sets a theme color for the address bar.
- Content is sized correctly for the viewport
- Has a `<meta name="viewport">` tag with `width` or `initial-scale`
- Manifest has a maskable icon

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

MAD & PWA Lab Journal

Experiment No.	Assignment-1
Assignment 1 Questions	1. Flutter Overview:Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community. 2. Widget Tree and Composition: Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree. 3. State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable. 4. Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.
Roll No.	6
Name	Mahek Bhagwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	5

MAD

Maher P. Bhagvani
DISA 6

Assignment-1

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.

Flutter is a popular open-source UI framework e by Google for building natively compiled applications for mobile, web and desktop from a single codeba. It's key features and advantages include

1. UI customization :- Flutter offers a rich set o customizable widgets that allow developers to c visually appealing and highly interactive user in
2. High performance :- Flutter's apps are compil to native machine code using the Dart platfor
3. Cross-platform :- Flutter ensures consistent ap behaviour and apperance across different platforms, eliminating the need for platform specific tweaks or adjustments.

Flutter differs from traditional mobile develop in several ways:

1. Single codebase :- Traditional approaches o involve maintaining seperate codebases for i and android using platform-specific langua
2. Development Tools :- Flutter provides its o set of development tools, including the Flutter SDK, Dart programming languag

FOR EDUCATIONAL USE

for example if you can compose a complex butt
by combining a 'Container' widget for styling
a 'Gesture Detector' for handling user interactio
and a 'Text' widget for displaying the button
label. This composition allows you to create a
custom button widget with specific functionali
and appearance.

Commonly used widgets:-

1. Container:- Used for styling and positioning
widgets. It can contain child widgets and de
properties like padding & margin.
2. Column / Row:- Organize child widgets
vertically (column) or horizontally (Row) allo
for flexible layout structures.
3. List View:- A scrollable list of widgets
4. Grid View:- A scrollable grid of widget
5. App Bar:- A material design app bar t
typically at top of screen.
6. Button widgets:- Interactive buttons

FOR EDUCATIONAL USE

and flutter plugin for popular integrated development environments.

Flutter has gained popularity in the developer community due to its ability to address common pain points in mobile app development, such as code reuse, UI consistency and development efficiency.

Q2 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

→ In flutter, a widget tree is a hierarchical structure of widgets that defines the user interface. Widgets are the building blocks of flutter applications, representing UI elements or components. The widget tree begins with the root widget and branches out into nested widgets forming a structure that defines the entire UI.

Widget composition in flutter involves combining multiple smaller widgets to create more complex and reusable UI components.

FOR EDUCATIONAL USE

Q3 State management in flutter
Discuss the importance of state manag
in flutter applications. Compare and cont
the different state management approac
available in flutter, such as setstate, P
and Riverpod. Provide scenario where ea
approach is suitable

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

SetState	Provider	Riverpod
1. Built-in flutter method	1. External package named (provider)	1. External pac (riverpod)
2. local state within a widget	2. Global state within tree	2. Global state additional feat
3. Limited scalability for large apps	3. Suitable for medium sized apps	3. Designed for and complex
4. May lead to code redundancy	4. Balance simplicity and readability	4. Emphasizes rea and clean syn

FOR EDUCATIONAL USE

Q4 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.

→ Integration:

1. Go to firebase console and create a new project
2. Add firebase SDK by including depend in pubsec.yaml

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

3. Run flutter pubget
4. Initialise firebase by calling 'firebase.initialise.App()' in main

import 'package:firebase_core/firebase.d

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

Benefits of using firebase as Backend

1. Real-time Database.
   - Firebase offers a realtime NO SQL data

2 Authentication
   - Provides a secure and easy-to-implem solution for user authentication.

3. Cloud firestore
   Firebase's cloud firestore provides a s and easy-to implement scalable NO database that allows you to store sync data in real time

4. Hosting
   Firebase Hosting provides a simple efficient way to display and host web application

FOR EDUCATIONAL USE

Data synchronization:

1. Real-time Database
When data changes on one client. It triggers events that automatically update data on other client.

2. Cloud Firestore:
It notifies clients, when data changes allowing for seamless real time updates

3. Authentication:
If user sign in or out on one device the authentication state is automatically reflected on other devices

FOR EDUCATIONAL USE

Scenarios where each is applicable:-

1. SetState:-
- For small to moderately complex applic
- Managing local state within a widget.
Eg:- Simple forms, UI components with UI specific state.

2. Provider:-
- For medium to large-sized applicat
- When a centralized state is needed accessible by multiple widgets.
Eg:- Managing user authentication, them changes or app-wide configuration.

3. Riverpod:-
- For large and complex applications
- When testability and maintainabilit top priorities.
Eg:- complex applications with multiple dynamic UIs.

FOR EDUCATIONAL USE

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.	6
Name	Mahek Bhagwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4:Understand various PWA frameworks and their requirements LO5: Design and Develop a responsive User Interface by applying PWA Design techniques LO6:Develop and Analyze PWA Features and deploy it over app hosting solutions
Grade:	4M

Name:- Mahek P. Bhagwani
D15A 6

PWA ASSIGNMENT-2

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

→ A Progressive Web App (PWA) is a type of web application that utilizes modern web technologies to provide a user experience similar to that of native mobile apps. PWA's are designed to work on any platform that uses a standard compliant browser, such as smartphones, tablets, and desktops. They offer several significant advantages in modern web development:-

1. Cross-Platform compatibility:- PWA's are built with web technologies like HTML, CSS and Javascript making them compatible across various platforms and devices.
2. Offline functionality:- One of the key features of PWA's is their ability to work offline or with limited connectivity.
3. App-like Experience:- PWA's provide an app-like experience with features such as push notifications, home screen icons, and full-screen mode, blurring the line between web apps and native mobile apps.

Key features functionalities that differ PWA from other mobile apps:

1. Easy installation:- PWA can be installed directly from the browser ~~through~~ an without going through an app store.
2. Offline functionality:- PWA's can function offline or with limited connectivity by caching content and data.
3. Security:- PWA's are served over HTTPS, ensuring data security and preventing unauthorized access.
4. Cross-platform compatibility:- PWA's work on any device with modern browser including smartphones tablets and desktops eliminating the need for separate versions for different platforms.

Q2 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.

→ Responsive web design is an approach to web design aimed at creating websites that provide an optimal viewing experience across a wide range of devices and screen sizes, from desktop computers to smartphones

and tablets. It involves using flexible layouts images and CSS media queries. In context of PWA's responsive web designs is crucial because PWA's are designed to work seamlessly across various devices and platforms. By ensuring that the web app's layout and functionality adjust to different screen sizes, responsive design helps provide a consistent and user-friendly experience regardless of the device being used.

Q3 Describe the lifecycle of service workers, including registration, installation, and activation phases.

A key player in the PWA universe is the "Service worker". The service worker is a Javascript file that runs on a separate thread apart from the one in which your usual website Javascript files run. The service worker is always up and is listening to the incoming responses and the outgoing requests.

3 Phases of Lifecycle

For the service worker to start performing its background tasks, we have to make sure it is up and running in the background. The phases through which the service worker goes through can be broadly classified into:

1. Registration Phase :- The first phase in the service worker's lifecycle is registering it to the browser. The registration can be done in two different ways:
- You either specify a scope for the service worker.
- You leave it to the default global scope of where the service worker file is present.

2. Installation :- what we previously did was registering the service worker with browser, but not install it. Once the service worker is successfully registered, it is not ready to be installed. The service worker script is downloaded to the browser and the browser will attempt to install the service worker.

There are few situations in which the service worker will be installed.
1. A new service worker file.
2. A modified service worker file.
The installation is automatic and once it is installed. the install event is fired and we can listen to this event to do certain tasks.

3. Activation :- Once the installation phase is successful, the next phase is the Activation phase. The service worker is now in the installed state but not yet active. The service worker does not move into the active state immediately after installation automatically.
There are a few ways to force the activation of the new service worker.
- One can shift + reload and hard refresh page.
- close the tab / navigate to another page not handled by the service worker
- Go to the service worker tab in the developers tools and click on skip waiting.

Q4 Explain the use of IndexedDB in the Service worker for data storage.

→ IndexedDB is a browser-based database that allows web applications including progressive

Web Apps (PWA's) to store and retrieve large amounts of structured data.

1. Initialization:- PWA's typically initialize IndexedDB when the application loads or when certain conditions are met. This involves opening a connection to the database.

2. Data storage:- PWA's can then store data in the IndexedDB by adding, updating or deleting records within the defined object stores.

3. Data Retrival:- When needed, PWA's can query the IndexedDB to retrive data based on certain criteria. Queries can be performed using indexes for faster retrieval.

4. Offline support:- One of the key benefits of using IndexedDB in PWA's is its support for offline data storage. PWA's can store data locally in IndexedDB allowing users to access and interact with the app even when they're offline.

5. Database structure:- Object stores are similar to tables in relational databases and are used to store data objects.

Sundaram

FOR EDUCATIONAL USE