Vivekanand Education Society's Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai
Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

Department of Information Technology

CERTIFICATE

This is to certify that **Mrudula Gotmare** 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: Flutter- Google meet clone/ PWA - Netflix website **Roll No. 23**

Year/Sem/Class **:** D15A/D15B **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- Google meet clone/ PWA - Netflix website **Roll No. 23**

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

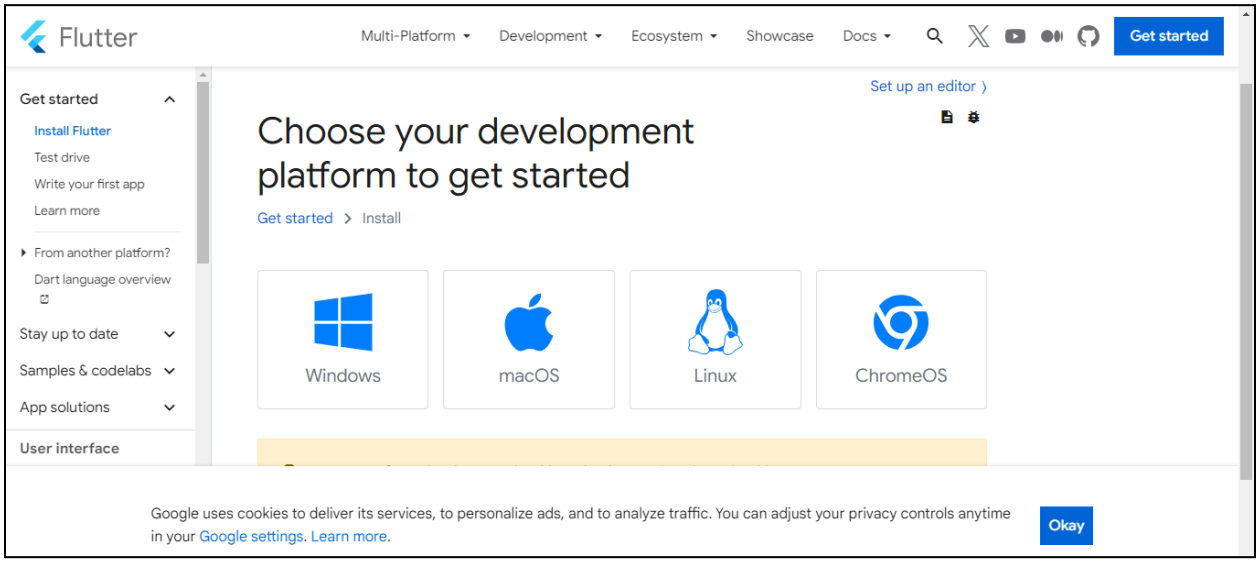
Sr. No	Experiment Title	LO	DOP	DOS	Grade
1.	To install and configure the Flutter Environment	LO1	16/1	23/1	15
2.	To design Flutter UI by including common widgets.	LO2	23/1	30/1	15
3.	To include icons, images, fonts in Flutter app	LO2	30/1	6/2	15
4.	To create an interactive Form using form widget	LO2	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

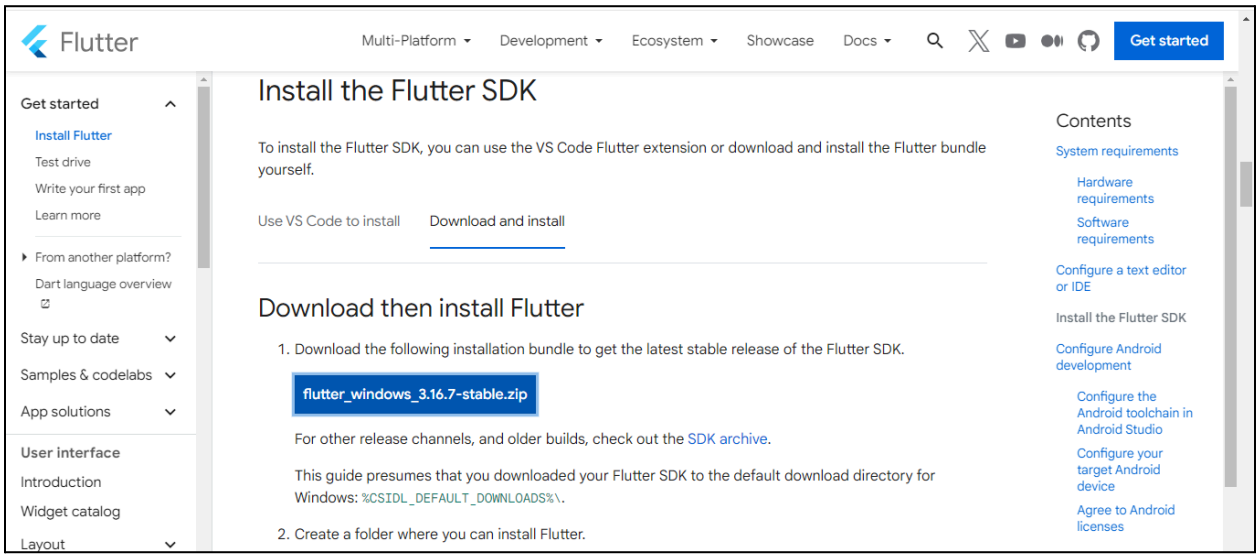
MAD & PWA Lab
Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	23
Name	Mrudula Gotmare
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15

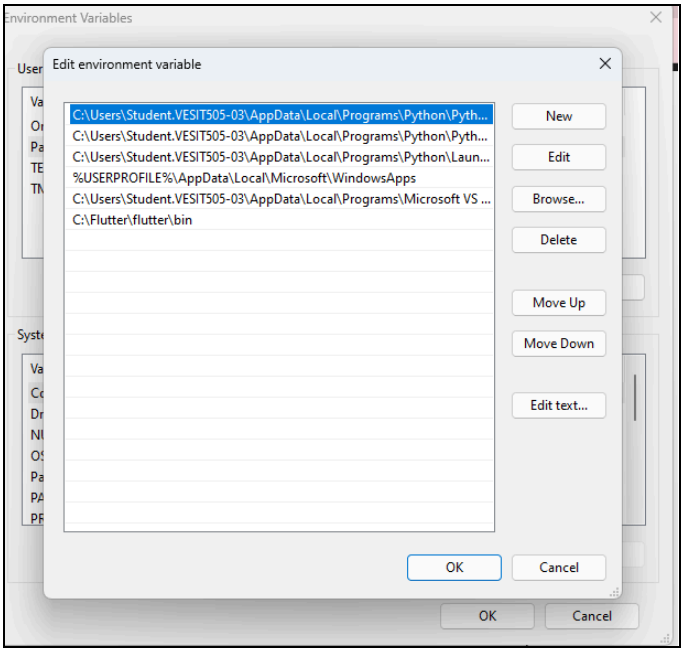
Project Title: Flutter- Google meet clone/ PWA - Netflix website **Roll No. 23**

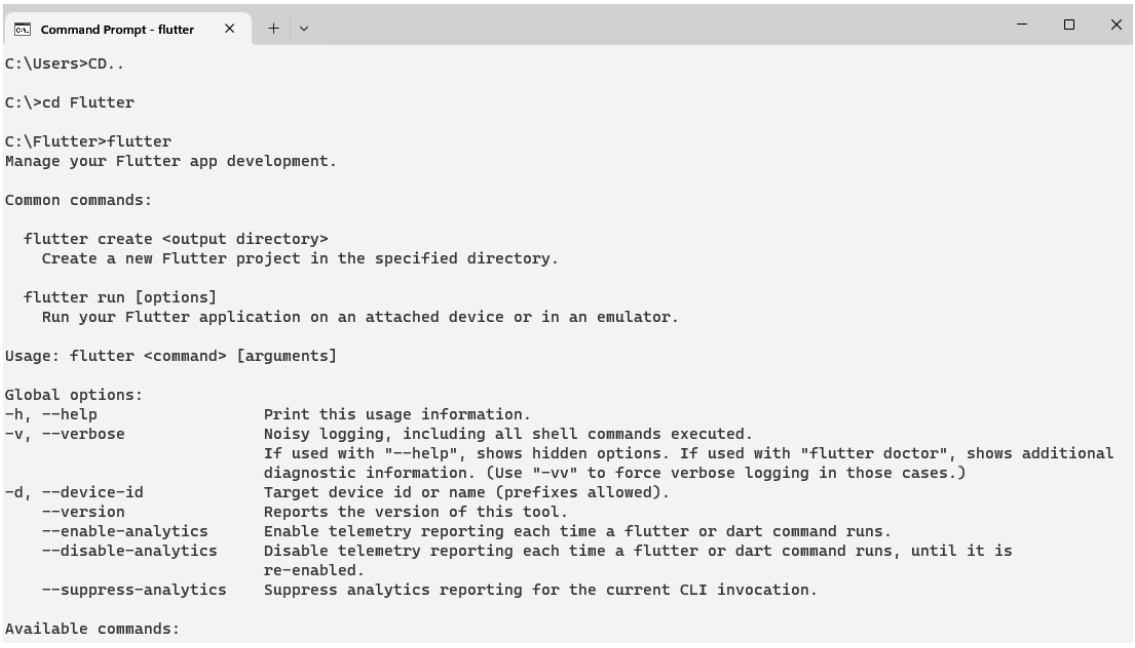
Aim : Installation and Configuration of Flutter Environment



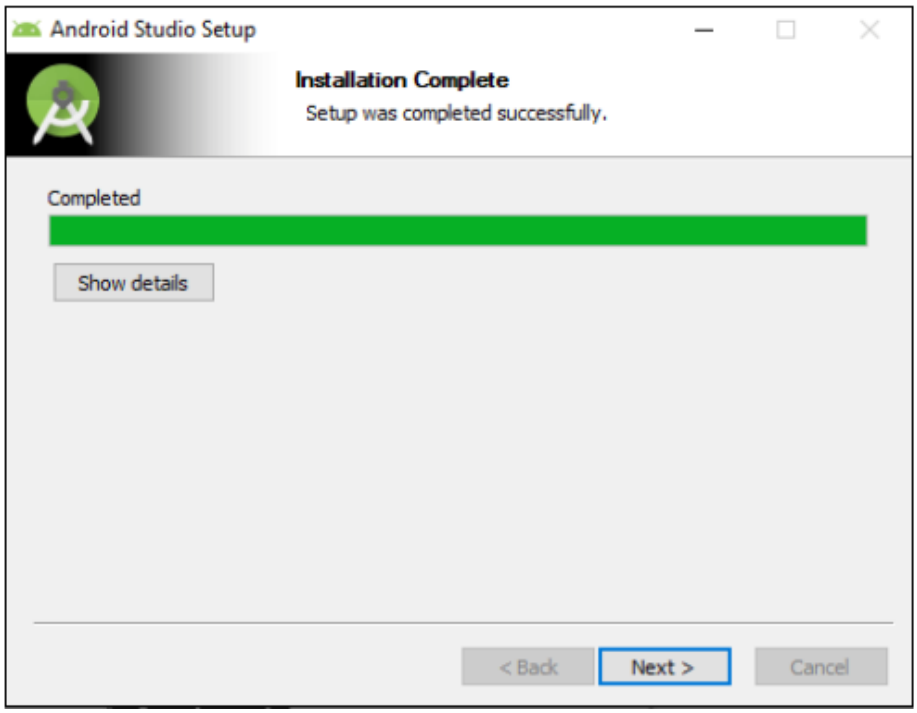


Project Title: Flutter- Google meet clone/ PWA - Netflix website **Roll No. 23**

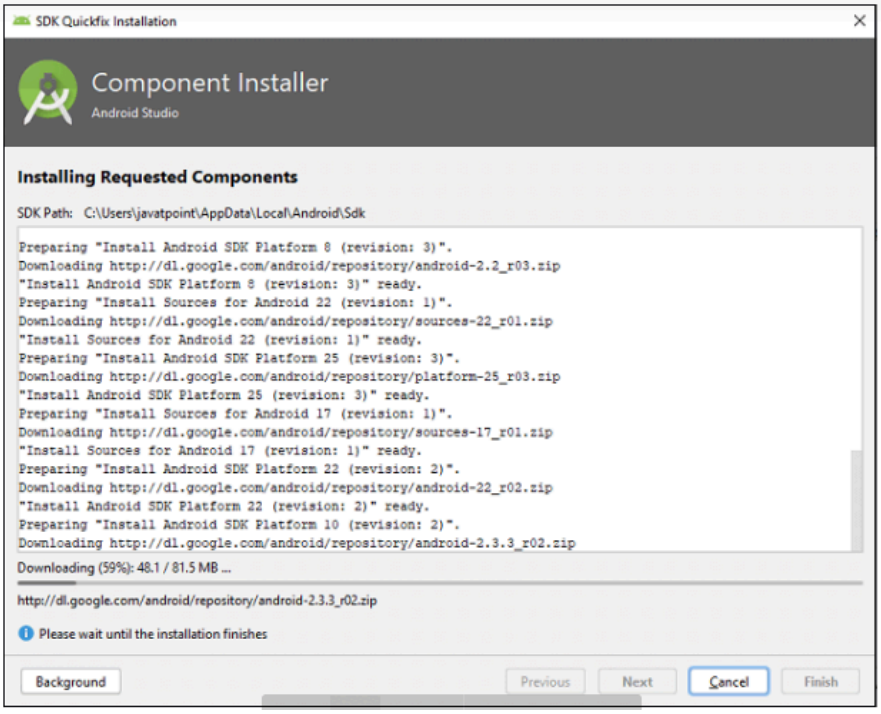


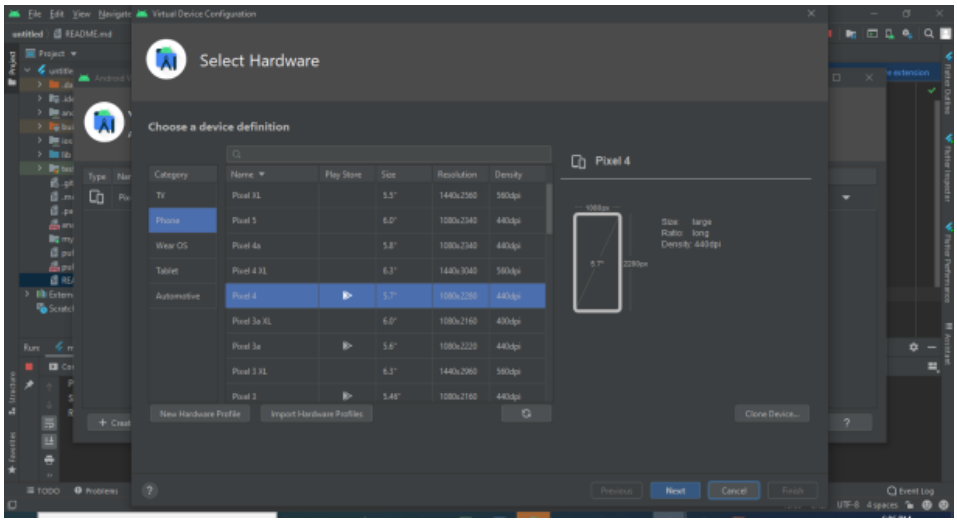


```
C:\Flutter\flutter>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.16.7, on Microsoft Windows [Version 10.0.22621.3007], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[!] Android toolchain - develop for Android devices (Android SDK version 33.0.2)
    X cmdline-tools component is missing
      Run `path/to/sdkmanager --install "cmdline-tools;latest"`
      See https://developer.android.com/studio/command-line for more details.
    X Android license status unknown.
      Run `flutter doctor --android-licenses` to accept the SDK licenses.
      See https://flutter.dev/docs/get-started/install/windows#android-setup for more details.
[√] Chrome - develop for the web
[√] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.4.3)
[√] Android Studio (version 2023.1)
[√] VS Code (version 1.85.1)
[√] Connected device (4 available)
[√] Network resources

! Doctor found issues in 1 category.

C:\Flutter\flutter>
```

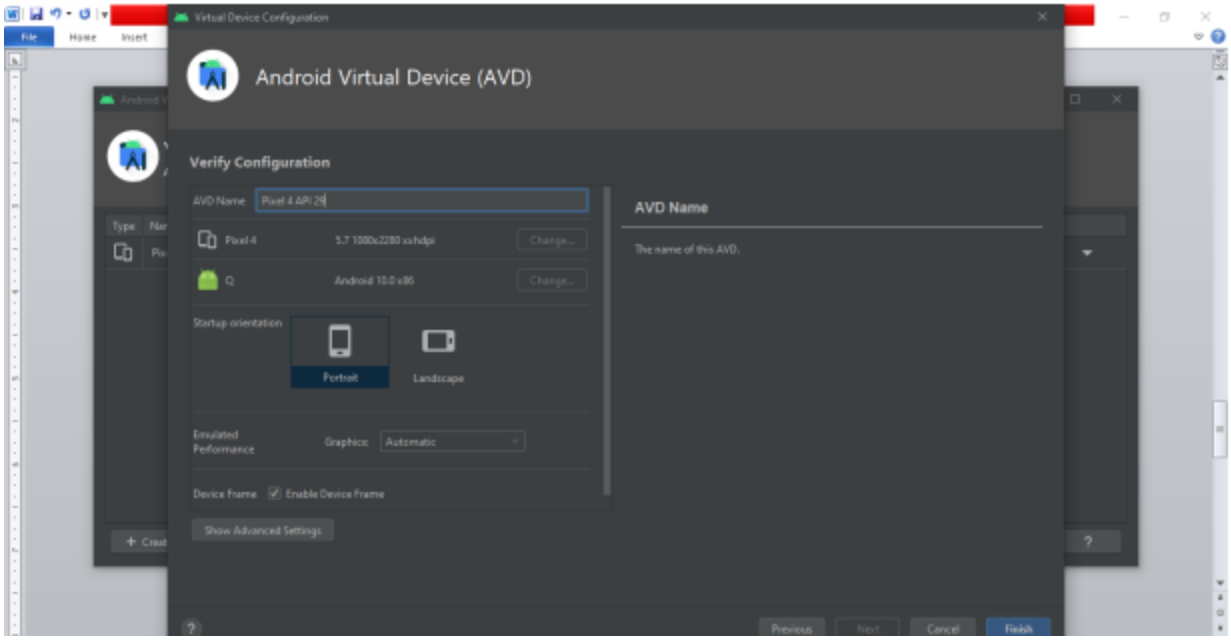


Project Title: Flutter- Google meet clone/ PWA - Netflix website **Roll No. 23**



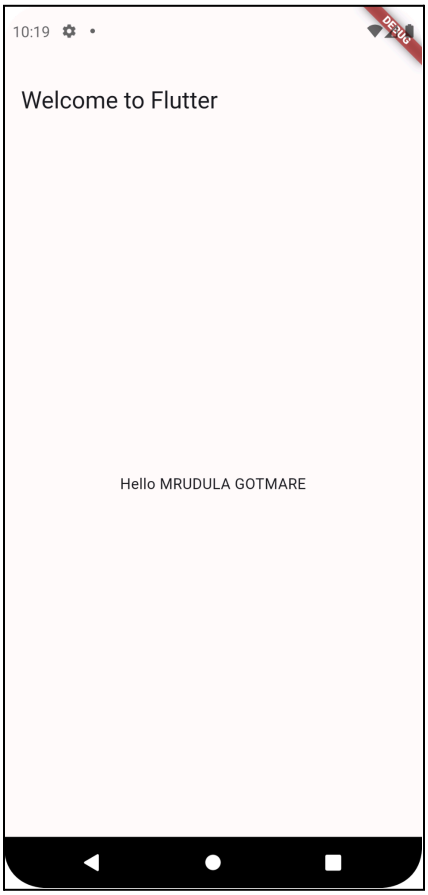


Project Title: Flutter- Google meet clone/ PWA - Netflix website **Roll No. 23**




```
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 MRUDULA GOTMARE'),
       ),
     ),
   );
 }
}
```

Project Title: Flutter- Google meet clone/ PWA - Netflix website **Roll No. 23**



MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	23
Name	Mrudula Gotmare
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Aim : Exploring Flutter Widgets.

Theory :

Each element on a screen of the Flutter app is a widget. The view of the screen completely depends upon the choice and sequence of the widgets used to build the apps. And the structure of the code of an apps is a tree of widgets.

Types of Widgets:

There are broadly two types of widgets in the flutter:

1. Stateless Widget – The widgets whose state can not be altered once they are built are called stateless widgets. These widgets are immutable once they are built i.e any amount of change in the variables, icons, buttons, or retrieving data can not change the state of the app.

2. Stateful Widget – The widgets whose state can be altered once they are built are called stateful Widgets. These states are mutable and can be changed multiple times in their lifetime. This simply means the state of an app can change multiple times with different sets of variables, inputs, data.

Description of the widgets used are as follows:

Scaffold – Implements the basic material design visual layout structure.
Text – To write anything on the screen.
Column – Arranges it's children widgets in a vertical array.
Center – To provide center alignment to other widgets.
Image – Displays an image from assets bundle.
CustomButton – Represents a custom button widget, custom styling and behavior

Code :

//ContactList.dart

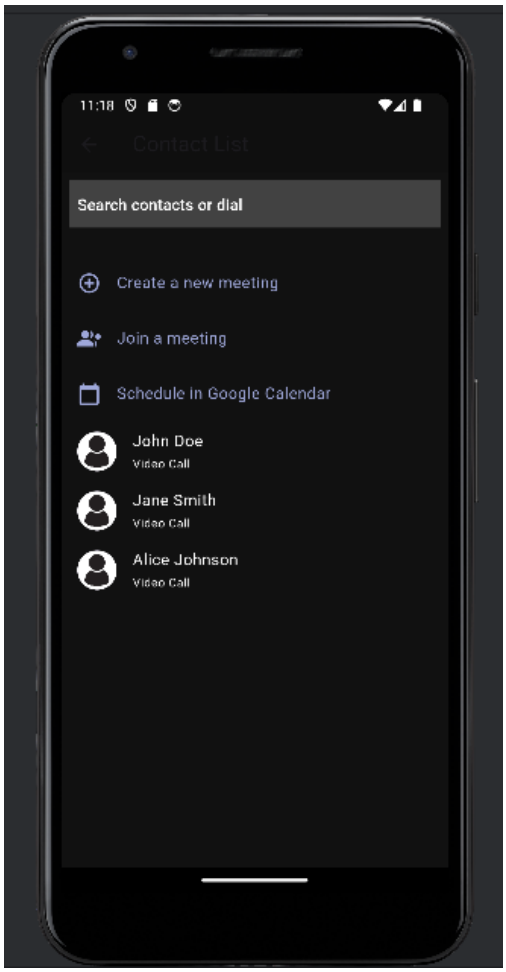
```
import 'package:flutter/material.dart';

class ContactListPage extends StatefulWidget {
  @override
  _ContactListPageState createState() => _ContactListPageState();
}

class _ContactListPageState extends State<ContactListPage> {
  late List<Contact> contacts;
  late List<Contact> filteredContacts;
```

```
TextEditingController searchController = TextEditingController();

@override
void initState() {
  super.initState();
  // Sample list of contacts
  contacts = [
    Contact(name: 'John Doe', imageUrl: 'assets/Images/profile.jpg', date: '8/10/2023'),
    Contact(name: 'Jane Smith', imageUrl: 'assets/Images/profile.jpg', date: '7/9/2023'),
    Contact(name: 'Alice Johnson', imageUrl: 'assets/Images/profile.jpg', date: '6/9/2023'),
    // Add more contacts as needed
  ];
  filteredContacts = contacts;
  searchController.addListener(() {
    filterContacts(searchController.text);
  });
}

void filterContacts(String query) {
  List<Contact> filteredList = [];
  if (query.isNotEmpty) {
    for (var contact in contacts) {
      if (contact.name.toLowerCase().contains(query.toLowerCase())) {
        filteredList.add(contact);
      }
    }
  } else {
    filteredList = contacts;
  }
  setState(() {
    filteredContacts = filteredList;
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Contact List'),
      backgroundColor: Colors.black12,
    ),
    body: Column(
      children: [
```

```
Padding(
  padding: const EdgeInsets.all(8.0),
  child: Container(
    decoration: BoxDecoration(
      //borderRadius: BorderRadius.circular(50),
      color: Colors.grey[800],
    ),
    child: Padding(
      padding: const EdgeInsets.symmetric(horizontal: 8.0),
      child: TextField(
        controller: searchController,
        style: TextStyle(color: Colors.white),
        decoration: InputDecoration(
          hintText: 'Search contacts or dial',
          hintStyle: TextStyle(color: Colors.white),
          border: InputBorder.none,
          labelStyle: TextStyle(color: Colors.white),
        ),
      ),
    ),
  ),
),
SizedBox(height: 20),
Expanded(
  child: ListView.builder(
    itemCount: filteredContacts.length + 3, // Add 3 for additional list items
    itemBuilder: (context, index) {
      // Check if the index is within the range of contacts
      if (index < 3) {
        // Render additional list items
        return ListTile(
          leading: Icon(
            index == 0
                ? Icons.add_circle_outline // Icon for "Create a new meeting"
                : index == 1
                ? Icons.group_add // Icon for "Join a meeting"
                : Icons.calendar_today, // Icon for "Schedule in Google Calendar"
            color: Colors.indigo[200], // Set icon color to indigo
          ),
          title: Text(
            index == 0
                ? 'Create a new meeting'
                : index == 1
                ? 'Join a meeting'
```

```
                    : 'Schedule in Google Calendar',
                style: TextStyle(color: Colors.indigo[200]),
              ),
              onTap: () {
                // Handle tapping on each list item
                if (index == 0) {
                  // Handle "Create a new meeting" action
                } else if (index == 1) {
                  // Handle "Join a meeting" action
                } else {
                  // Handle "Schedule in Google Calendar" action
                }
              },
            );
          } else {
            // Render contact list items
            int contactIndex = index - 3; // Adjust index for contact list
            return ListTile(
              leading: CircleAvatar(
                backgroundImage: AssetImage(filteredContacts[contactIndex].imageUrl),
              ),
              title: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    filteredContacts[contactIndex].name,
                    style: TextStyle(color: Colors.white),
                  ),
                  SizedBox(height: 2),
                  Text(
                    'Video Call',
                    style: TextStyle(color: Colors.white, fontSize: 12),
                  ),
                ],
              ),
            );
          }
        },
      ),
    ),
  ],
 ),

);
```

```
  }
}

// Contact class to represent a contact with name, image URL, and date
class Contact {
  final String name;
  final String imageUrl;
  final String date;

  Contact({
    required this.name,
    required this.imageUrl,
    required this.date,
  });
}
```



MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	23
Name	Mrudula Gotmare
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- Google meet clone/ PWA - Netflix website **Roll No. 23**

Aim : To include, icon, images, font in Flutter app.

Theory :

Flutter images -

When you create an app in Flutter, it includes both code and assets (resources). An asset is a file, which is bundled and deployed with the app and is accessible at runtime. The asset can include static data, configuration files, icons, and images. The Flutter supports many image formats, such as JPEG, WebP, PNG, GIF, animated WebP/GIF, BMP, and WBMP.

Flutter Icon -

Flutter provides an Icon Widget to create icons in our applications. We can create icons in Flutter, either using inbuilt icons or with the custom icons. Flutter provides the list of all icons in the Icons class. In this article, we are going to learn how to use Flutter icons in the application.

Flutter font -

The text widget has its own properties like font style, font size, font-weight, etc. The Flutter team has set a few default properties before making things easier for the user to create apps. So, customizing text is all about editing these properties as we want, to get the desired output.

Code :

Home_screen

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

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

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

class _LoginScreenState extends State<LoginScreen> {
  final AuthMethods _authMethods = AuthMethods();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(
```

```
children: [
  const SizedBox(height: 40),
  Container(
    width: double.infinity,
    padding: EdgeInsets.all(10), // Add padding for spacing
    decoration: BoxDecoration(
      color: Color.fromARGB(255, 47, 46, 46),
      //border: Border.all(color: Colors.lightBlue),
      borderRadius: BorderRadius.circular(50), // Add border color
    ),
    alignment: Alignment.center,
    child: Text(
      'For SAMSUNG Galaxy',
      style: TextStyle(
        fontSize: 18,
        fontWeight: FontWeight.bold,
      ),
    ),
  ),
  const SizedBox(height: 30),
  Expanded(
    child: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          const SizedBox(height: 50),
          Image.asset('assets/images/bg_img1.jpg'),
          // Add a SizedBox with desired height
          // Add a SizedBox for spacing
        ],
      ),
    ),
  ),
  Expanded(
    child: Align(
      alignment: Alignment.bottomRight,
      child: SizedBox(
        width: 150,
        child: CustomButton(
          text: 'New',
          onPressed: () async {
            bool res = await _authMethods.signInWithGoogle(context);
            if (res) {
              Navigator.pushNamed(context, '/home');
```

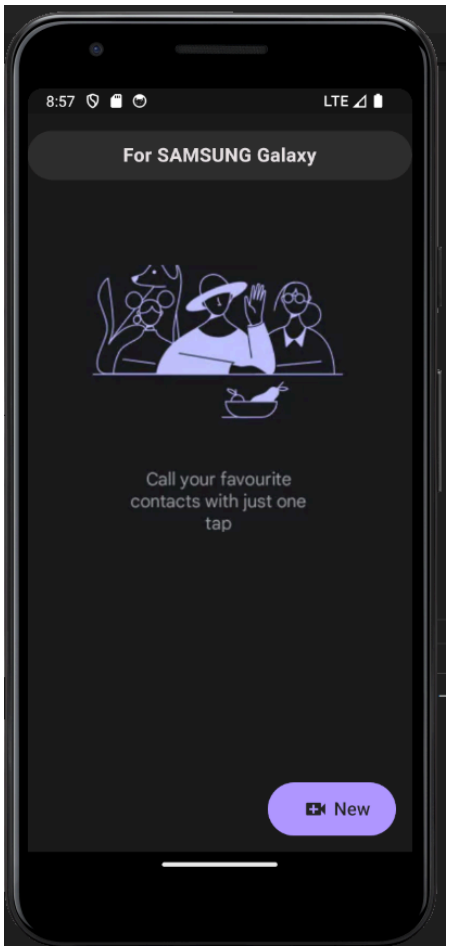
```
                  }
                },
              ),

            ),
          ),
        ),

      ],
    ),
  );
 }
}
```

Custom_button

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

class CustomButton extends StatelessWidget {
 final String text;
 final VoidCallback onPressed;

 const CustomButton({
  Key? key,
  required this.text,
  required this.onPressed,
 }) : super(key: key);

 @override
 Widget build(BuildContext context) {
  return Padding(
   padding: const EdgeInsets.all(15.0),
   child: ElevatedButton(
    onPressed: onPressed,
    style: ElevatedButton.styleFrom(
     primary: Color.fromARGB(255, 175, 151, 255),
     minimumSize: const Size(
      20,
      50,
     ),
    ),
    child: Row(
     mainAxisAlignment: MainAxisAlignment.end,
     children: [
      Icon(
```

```
            Icons.video_call,
            color: const Color.fromARGB(255, 39, 38, 38), // Set icon color to dark grey
          ),
          SizedBox(width: 5), // Add space between icon and text
          Text(
            text,
            style: const TextStyle(
              fontSize: 17,
              color: Color.fromARGB(255, 39, 38, 38),
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```

Output :



MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	23
Name	Mrudula Gotmare
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- Google meet clone/ PWA - Netflix website **Roll No. 23**

Aim : To create an interactive Form using form widget

Theory :

Flutter Forms

Forms are an integral part of all modern mobile and web applications. It is mainly used to interact with the app as well as gather information from the users. They can perform many tasks, which depend on the nature of your business requirements and logic, such as authentication of the user, adding user, searching, filtering, ordering, booking, etc.

Some Properties of Form Widget

- key: A GlobalKey that uniquely identifies the Form. You can use this key to interact with the form, such as validating, resetting, or saving its state.
- child: The child widget that contains the form fields. Typically, this is a Column, ListView, or another widget that allows you to arrange the form fields vertically.
- autovalidateMode: An enum that specifies when the form should automatically validate its fields.

Code :

//login_screen.dart

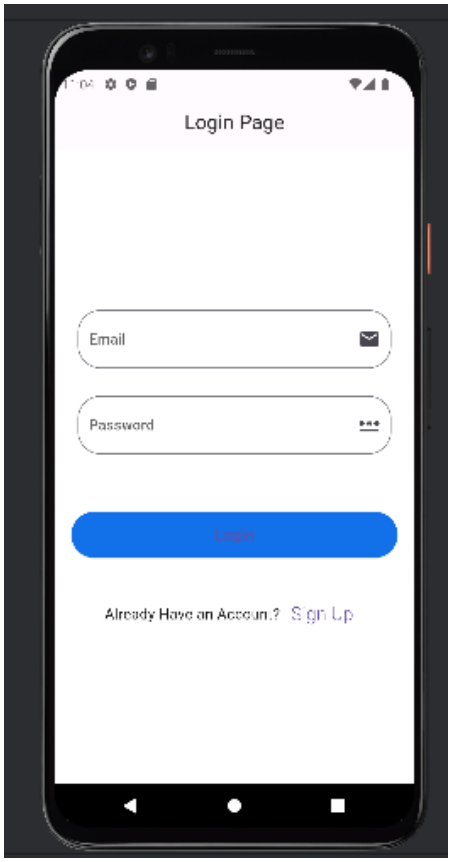
```
import 'package:flutter/material.dart';
import 'package:gmeet_clone/Utils/UIhelper.dart';
import 'package:gmeet_clone/widgets/custom_button.dart';

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

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

  class _LoginScreenState extends State<LoginScreen> {
  TextEditingController emailController=TextEditingController();
  TextEditingController passwordController=TextEditingController();

    @override
    Widget build(BuildContext context) {
      return Scaffold(
        appBar: AppBar(
          title: Text("Login Page"),
          centerTitle: true,
        ),
        body: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
```

```
        UIhelper.CustomTextField(emailController, "Email", Icons.mail, false),
        UIhelper.CustomTextField(passwordController, "Password", Icons.password, true),
        const SizedBox(height: 30),
        CustomButton(text: "Login", onPressed: () { }),
          const SizedBox(height: 20),
          Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
              Text("Already Have an Account?", style: TextStyle(fontSize: 16),),
              TextButton(onPressed: (){}, child: Text("Sign Up",style: TextStyle(fontSize: 20,
fontWeight: FontWeight.w300)))
            ]
          )
      ],),
    );
  }
 }
```

//UIhelper.dart

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

class UIhelper{
  static CustomTextField(TextEditingController controller, String text, IconData iconData,bool
toHide){
    return Padding(
      padding: const EdgeInsets.symmetric(horizontal: 25, vertical: 15),
      child: TextField(
        controller: controller,
        obscureText: toHide,
        decoration: InputDecoration(
          hintText: text,
          suffixIcon: Icon(iconData),
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(25)
          )
        ),
      ),
    );
  }
}
```

Project Title: Flutter- Google meet clone/ PWA - Netflix website **Roll No. 23**



MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	23
Name	Mrudula Gotmare
Class	D15A/D15B
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- Google meet clone/ PWA - Netflix website **Roll No. 23**

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

Theory :

1. **Navigation and routing**
    a. In the MeetingCode widget's build method, a TransparentButton widget with the text 'Join Meeting' is defined.
    b. When this button is pressed, it triggers the onPressed callback, which navigates to the CallPage.
    c. Navigation is handled using Navigator.of(context).push(MaterialPageRoute(builder: (context)=> CallPage(callID: "1",userId: widget.userId, meetingCode: meetingCode)));.
    d. This code pushes a new route onto the Navigator's stack, which creates a new CallPage widget with the specified parameters (callID, userId, and meetingCode).

2. **Gestures**
    a. The 'Dismiss' text at the bottom of the MeetingCode widget is wrapped with a GestureDetector.
    b. When this text is tapped, it triggers the onTap callback, which calls Navigator.pop(context).
    c. Navigator.pop(context) removes the top route from the Navigator's stack, effectively closing the current screen and returning to the previous screen.

Code :

//meeting-code.dart

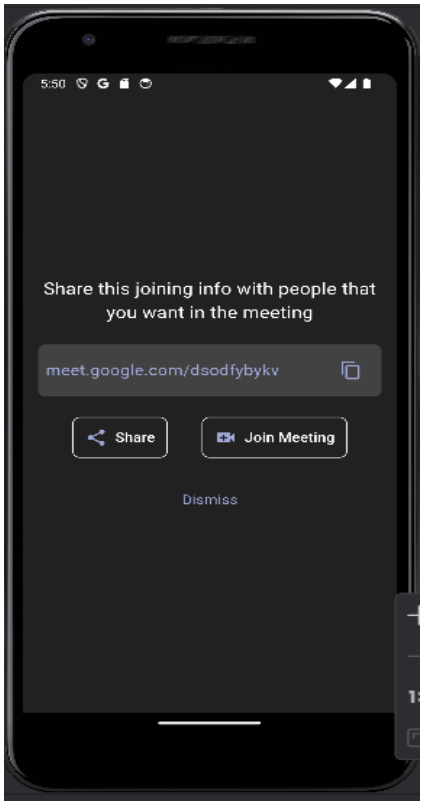
```
import 'package:flutter/material.dart';
import 'dart:math';

import 'package:gmeet_clone/screens/call.dart';

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

  @override
  State<MeetingCode> createState() => _MeetingCodeState();
}

class _MeetingCodeState extends State<MeetingCode> {
  late String meetingCode;

  @override
  void initState() {
    super.initState();
    // Generate a random meeting code
```

```
  generateMeetingCode();
 }

 void generateMeetingCode() {
  // Example meeting code "meet.google.com/uyz-vjvj-mbp"
  setState(() {
   meetingCode = 'meet.google.com/${_generateRandomCode()}';
  });
 }

 String _generateRandomCode() {
  const characters = 'abcdefghijklmnopqrstuvwxyz';
  final random = Random();
  String code = '';
  for (int i = 0; i < 10; i++) {
   code += characters[random.nextInt(characters.length)];
  }
  return code;
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   backgroundColor: Colors.grey[900], // Set the background color to grey
   body: Center(
    child: Padding(
     padding: const EdgeInsets.all(16.0),
     child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      crossAxisAlignment: CrossAxisAlignment.stretch,
      children: [
       Text(
        'Share this joining info with people that you want in the meeting',
        style: TextStyle(
         color: Colors.white,
         fontSize: 20,
        ),
        textAlign: TextAlign.center,
       ),
       SizedBox(height: 24),
       Container(
        padding: const EdgeInsets.all(8.0),
        decoration: BoxDecoration(
         color: Colors.grey[800],
```

```
            borderRadius: BorderRadius.circular(8.0),
          ),
          child: Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [
              Text(
                meetingCode ?? 'Generating...',
                style: TextStyle(
                  color: Colors.indigo[200],
                  fontSize: 18,
                ),
                textAlign: TextAlign.left,
              ),
              IconButton(
                onPressed: () {
                  // Implement copy functionality
                },
                icon: Icon(
                  Icons.content_copy,
                  color: Colors.indigo[200],
                ),
              ),
            ],
          ),
        ),
        SizedBox(height: 25),
        Row(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: [
            TransparentButton(
              text: 'Share',
              icon: Icons.share,
              onPressed: () {
                // Implement share functionality
              },
            ),
            TransparentButton(
              text: 'Join Meeting',
              icon: Icons.video_call,
              onPressed: () {
                // Implement join meeting functionality
                Navigator.of(context).push(MaterialPageRoute(builder: (context)=>
CallPage(callID: "1",userId: widget.userId, meetingCode: meetingCode)));
              },
```

```
              ),
            ],
          ),
          SizedBox(height: 40),
          Center(
            child: GestureDetector(
              onTap: () {
                Navigator.pop(context); // Navigate back to the ContactList page
              },
              child: Text(
                'Dismiss',
                style: TextStyle(
                  color: Colors.indigo[200],
                  fontSize: 16,
                  decoration: TextDecoration.underline,
                ),
              ),
            ),
          ),
        ],
      ),
    ),
  );
 }
}

class TransparentButton extends StatelessWidget {
 final String text;
 final IconData? icon;
 final VoidCallback onPressed;

 const TransparentButton({
  required this.text,
  this.icon,
  required this.onPressed,
 });

 @override
 Widget build(BuildContext context) {
  return Container(
   decoration: BoxDecoration(
    border: Border.all(color: Colors.white),
    borderRadius: BorderRadius.circular(8.0),
```

```
      ),
      child: TextButton(
        onPressed: onPressed,
        child: Row(
          mainAxisSize: MainAxisSize.min,
          children: [
            if (icon != null) Icon(icon, color: Colors.indigo[200]),
            SizedBox(width: icon != null ? 8.0 : 0), // Add some spacing if there's an icon
            Text(
              text,
              style: TextStyle(
                color: Colors.white,
                fontSize: 16,
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```



MAD & PWA Lab
Journal

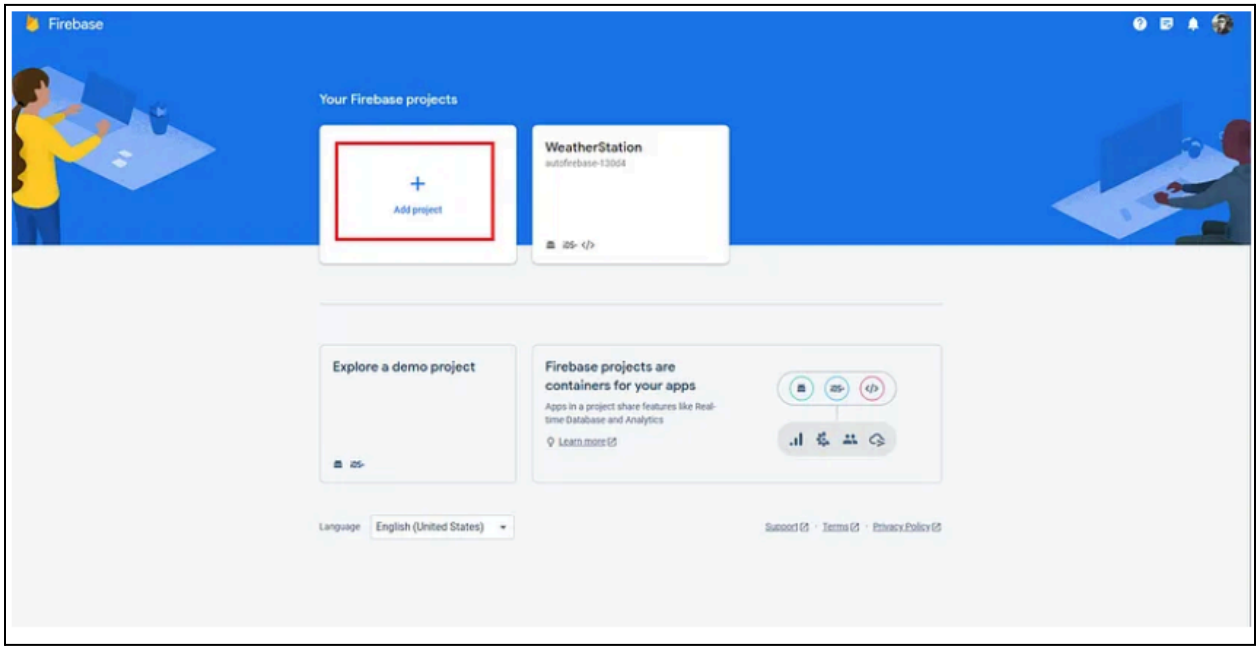
Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	23
Name	Mrudula Gotmare
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	15

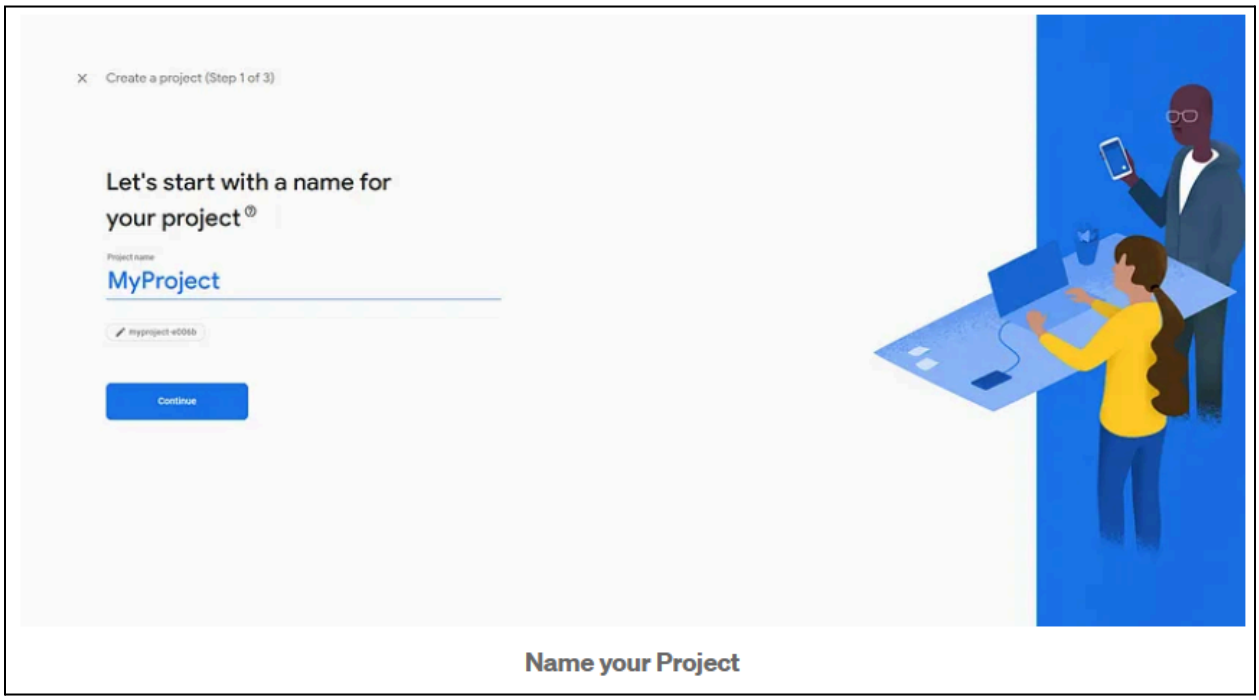
Project Title: Flutter- Google meet clone/ PWA - Netflix website **Roll No. 23**

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

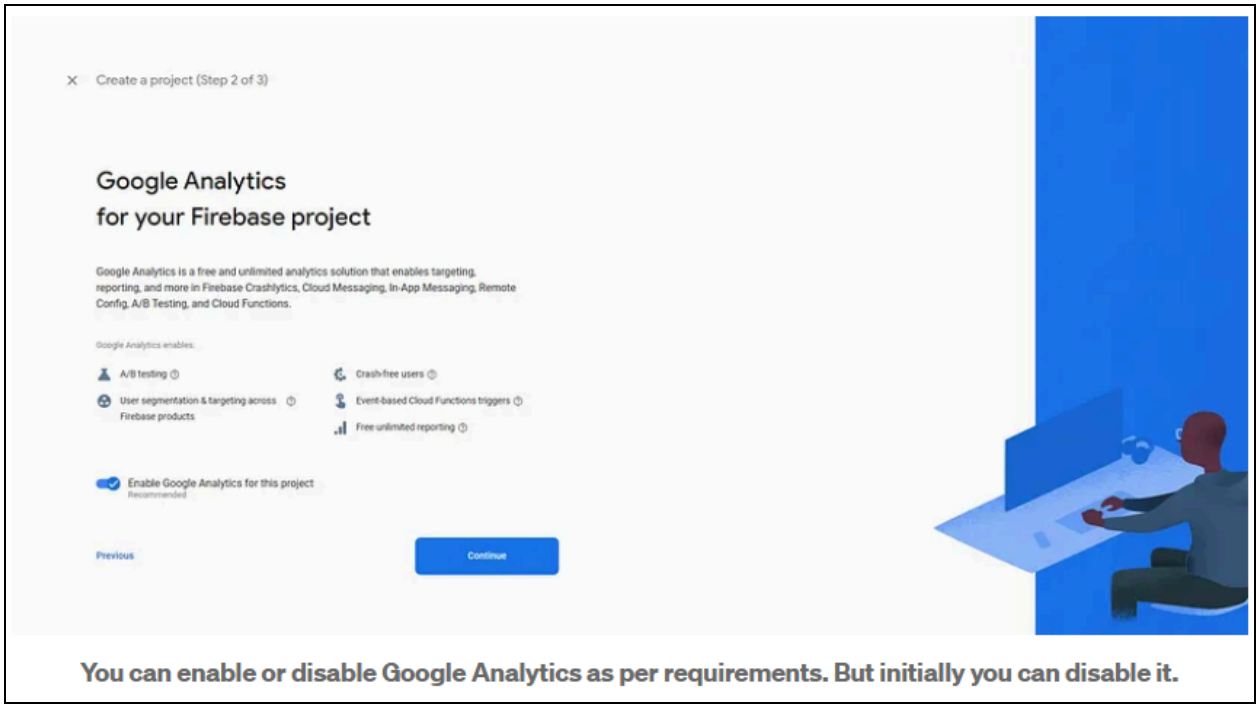
Theory :

First and foremost, you need to setup a Firebase project. For that, you need to go to *https://console.firebase.google*.com and tap on Add project.

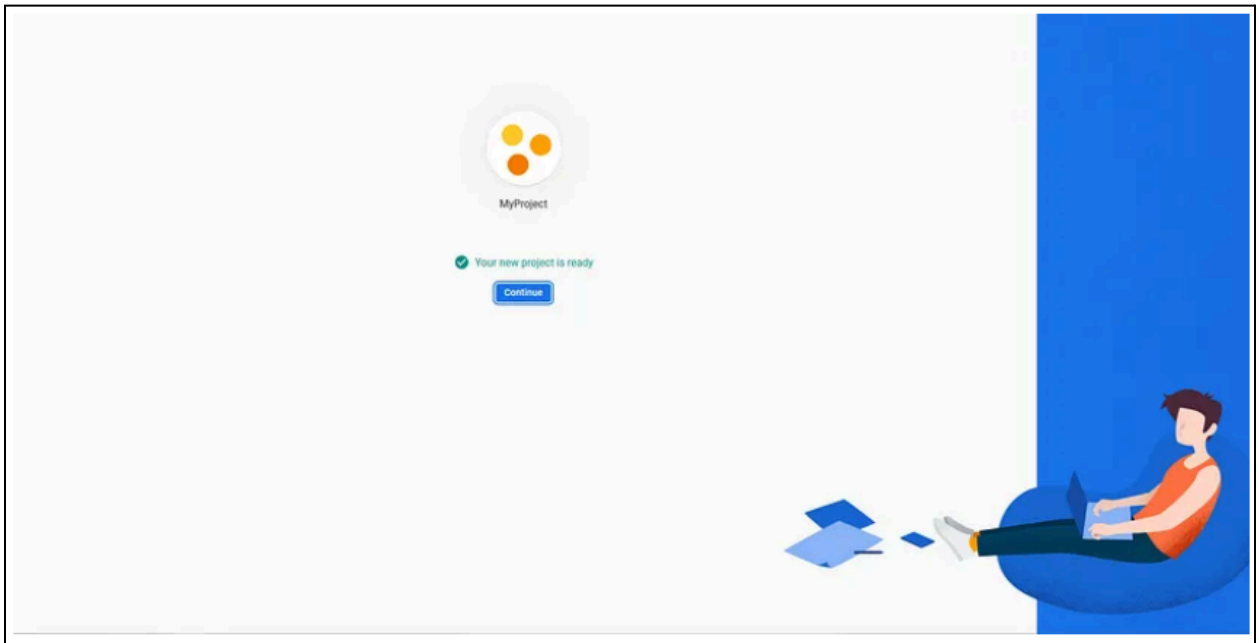




Name your Project



You can enable or disable Google Analytics as per requirements. But initially you can disable it.



Voila, your Firebase project is ready. You will be presented with the following screen.

As we have successfully created a Firebase project we can continue with installing Firebase CLI. Firebase CLI differs from OS to OS.

To Install Firebase CLI in Windows,

1. standalone binary for CLI (for new developers, who are unfamiliar with Node.js).

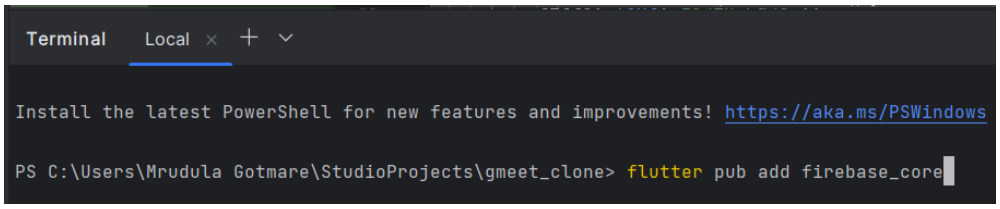
2. npm (Node Package Manager). [Head over the download page and download the LTS installer and execute it. To check if installation is successful or not, you can run: node -v and npm -v command in CMD prompt].

```
C:\Users\Mrudula Gotmare>node -v
v18.17.1
```

3. After installing npm, you need to install Firebase CLI via npm using the following commands:
    a. **npm install -g firebase-tools**
    b. **dart pub global activate flutterfire_cli**

4. Use the FlutterFire CLI to configure your Flutter apps to connect to Firebase.
    a. **flutterfire configure**

5. From your Flutter project directory, run the following command to install the core plugin:



6. From your Flutter project directory, run the following command to ensure that your Flutter app's Firebase configuration is up-to-date:

7. In your lib/main.dart file, import the Firebase core plugin and the configuration file you generated earlier:

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

8. Also in your lib/main.dart file, initialize Firebase using the DefaultFirebaseOptions object exported by the configuration file:

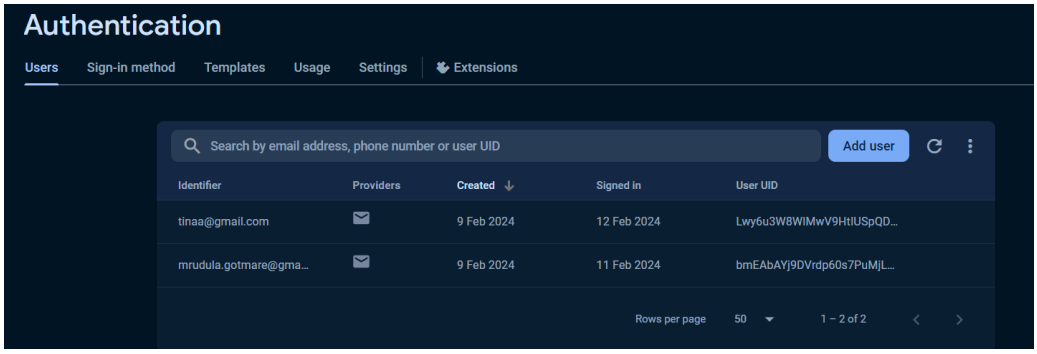
```
await Firebase.initializeApp(
  options: DefaultFirebaseOptions.currentPlatform,
```

```
);
```

9. Rebuild your Flutter application:

```
PS C:\Users\Mrudula Gotmare\StudioProjects\gmeet_clone> flutter run
```

10. So, the firebase is successfully connected to the project.



Conclusion : In this article, you learned how to set up and ready our Flutter applications to be used with Firebase. Flutter has official support for Firebase with the FlutterFire set of libraries.

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.	23
Name	Mrudula Gotmare
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	15

Project Title: Flutter- Google meet clone/ PWA - Netflix website **Roll No. 23**

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

Theory :

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.

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.

Code:

//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"
        }
]
}
```

//serviceworker.js

```
var staticCacheName = "pwa";
```

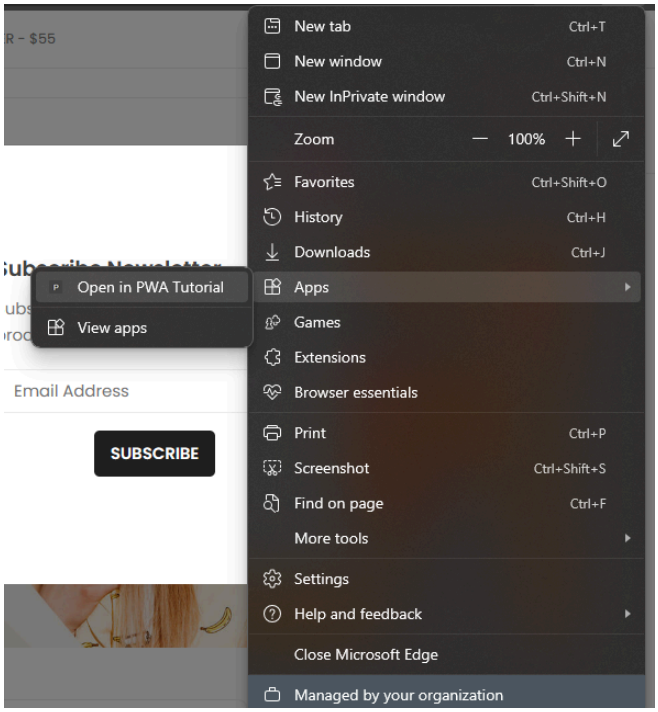
```
self.addEventListener("install", function (e) {
e.waitUntil(
        caches.open(staticCacheName).then(function (cache) {
        return cache.addAll(["/"]);
        })
);
});

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

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

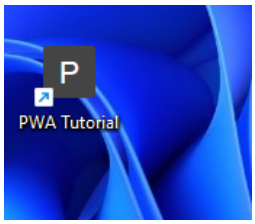
//index.html

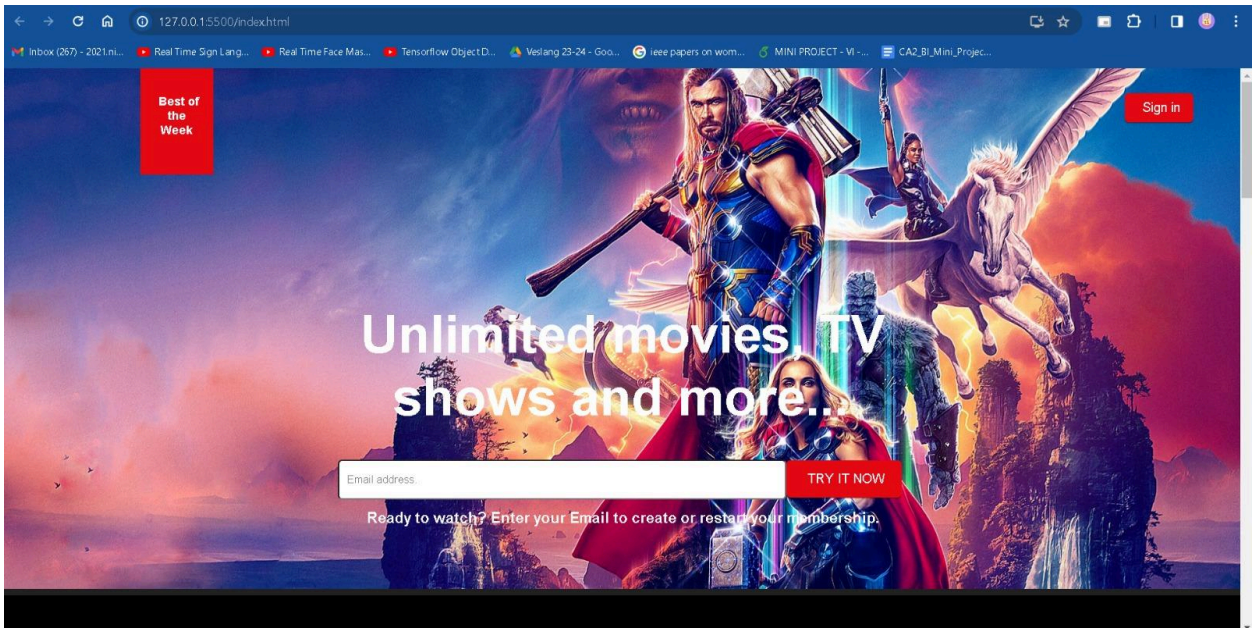
```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Anon - eCommerce Website</title>

  <!--
    - favicon
  -->
  <link rel="shortcut icon" href="./assets/images/logo/favicon.ico" type="image/x-icon">

  <!--
    - custom css link
  -->
  <link rel="stylesheet" href="./assets/css/style-prefix.css">

  <!--
    - google font link
  -->
  <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@300;400;500;600;700;800;900&
display=swap"
    rel="stylesheet">
  <!-- Manifest File link -->
  <link rel="manifest" href="manifest.json">

</head>

<body>
<script src="./assets/js/script.js"></script>
  <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>

</body>
</html>
```

Project Title: Flutter- Google meet clone/ PWA - Netflix website **Roll No. 23**







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.	23
Name	Mrudula Gotmare
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

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.

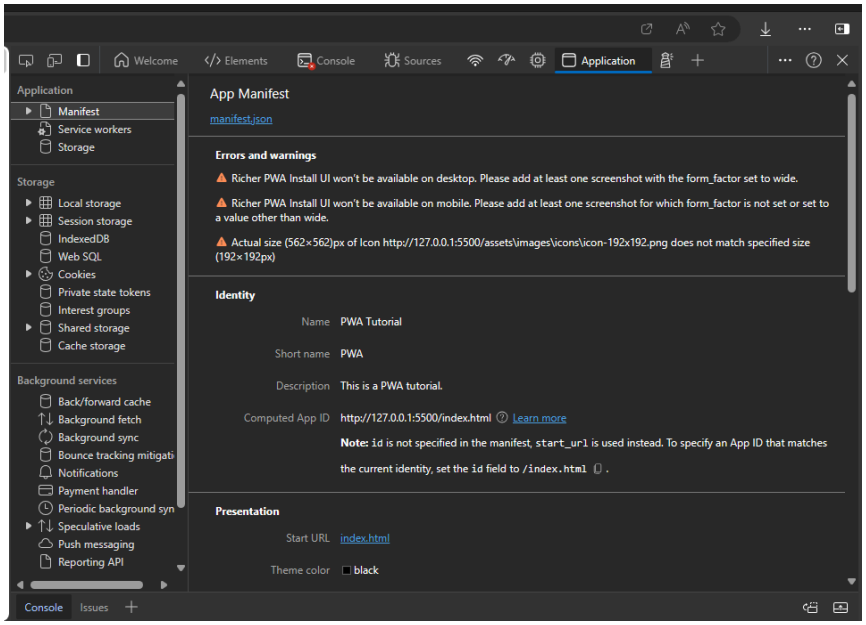
What can we do with Service Workers?
You can dominate Network Traffic
You can manage all network traffic of the page and do any manipulations. For example, when the page requests a CSS file, you can send plain text as a response or when the page requests an HTML file, you can send a png file as a response. 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.

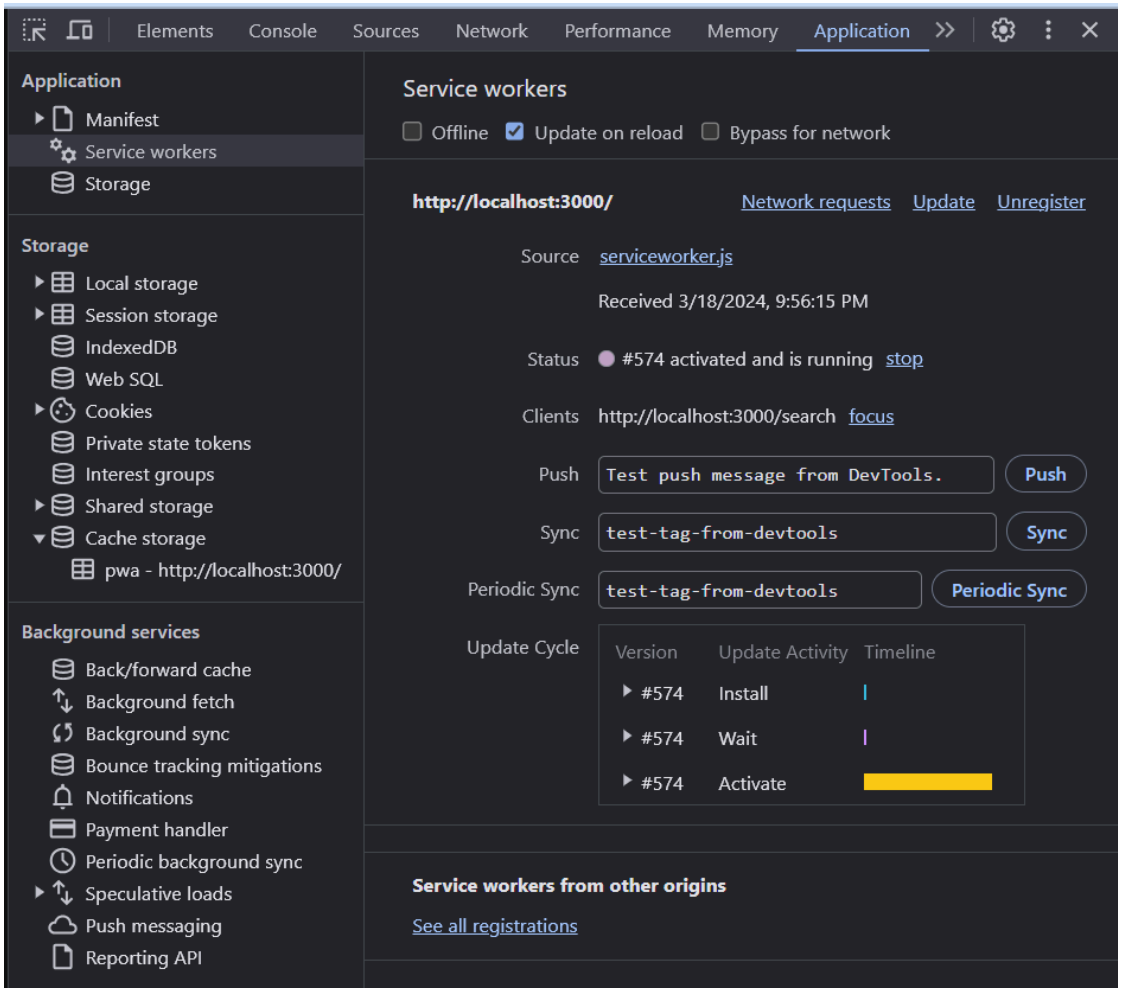
What can’t we do with Service Workers?
You can’t access the Window
You can’t access the window, therefore, You can’t manipulate DOM elements. But, you can communicate to the window through post Message and manage processes that you want.

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

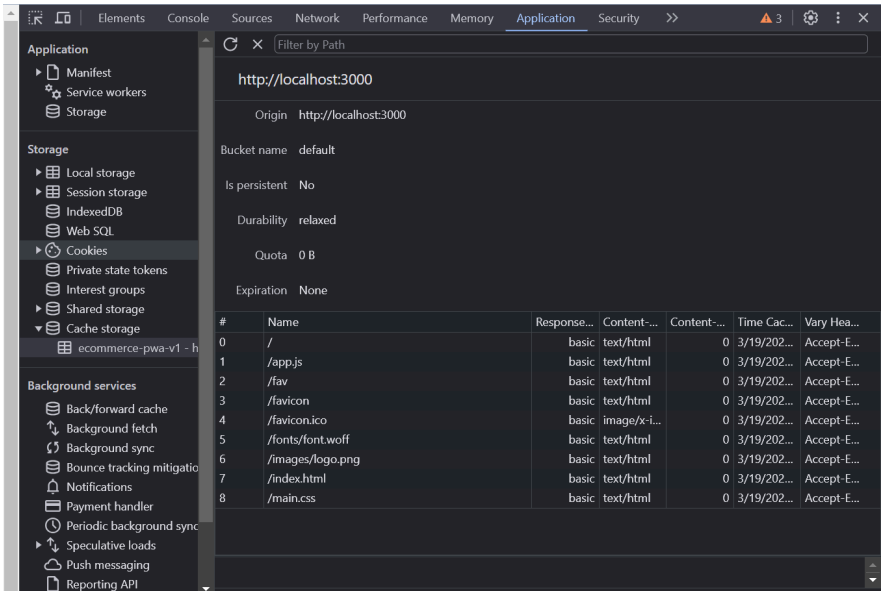
Project Title: Flutter- Google meet clone/ PWA - Netflix website **Roll No. 23**

Screenshots :





Project Title: Flutter- Google meet clone/ PWA - Netflix website **Roll No. 23**



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.	23
Name	Mrudula Gotmare
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

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

- **CacheFirst** - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.

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

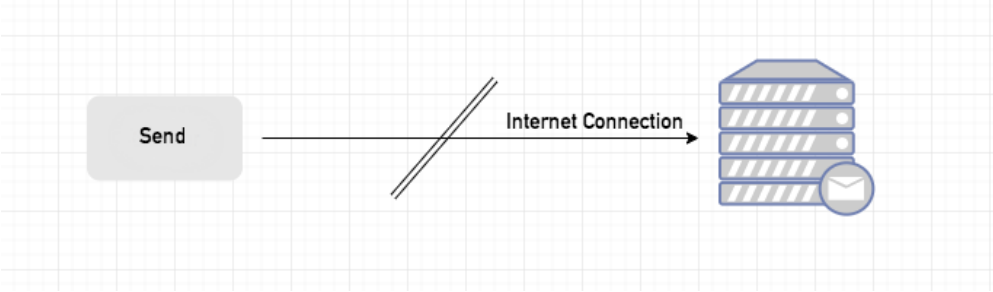
```
self.addEventListener("fetch", function (event) {
    const req = event.request;
    const url = new URL(req.url);

    if (url.origin === location.origin) {
        event.respondWith(cacheFirst(req));
    }
    else {
        event.respondWith(networkFirst(req));
    }
});

async function cacheFirst(req) {
    return await caches.match(req) || fetch(req);
}

async function networkFirst(req) {
    const cache = await caches.open("pwa-dynamic");
    try {
        const res = await fetch(req);
        cache.put(req, res.clone());
        return res;
    } catch (error) {
        const cachedResponse = await cache.match(req);
        return cachedResponse || await caches.match("./noconnection.json");
    }
}
```

Sync Event

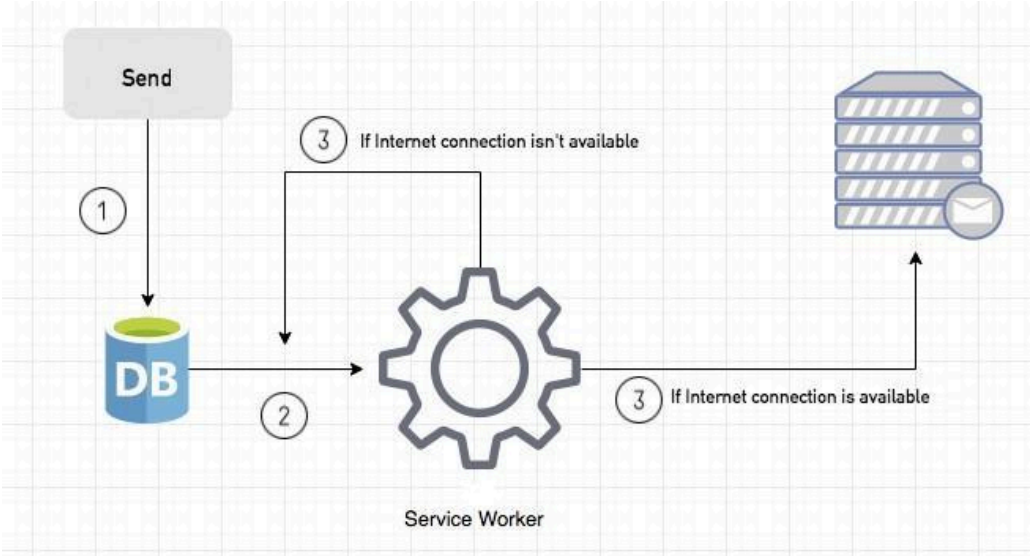
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

Here is a job for the Background Sync.

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



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



case.

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

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

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

Event Listener for Background Sync Registration

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

Event Listener for sw.js

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

Push Event

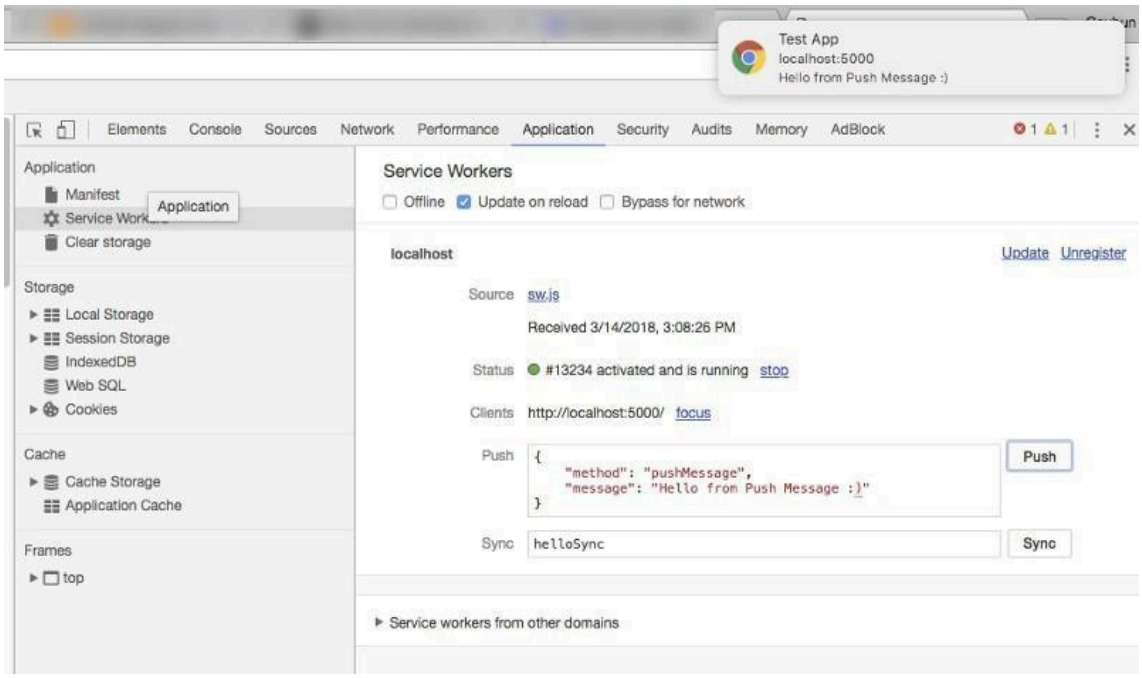
This is the event that handles push notifications that are received from the server. You can apply any method with received data.

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

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

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

You can use Application Tab from Chrome Developer Tools for testing push notification.



code:

serviceworker.js:

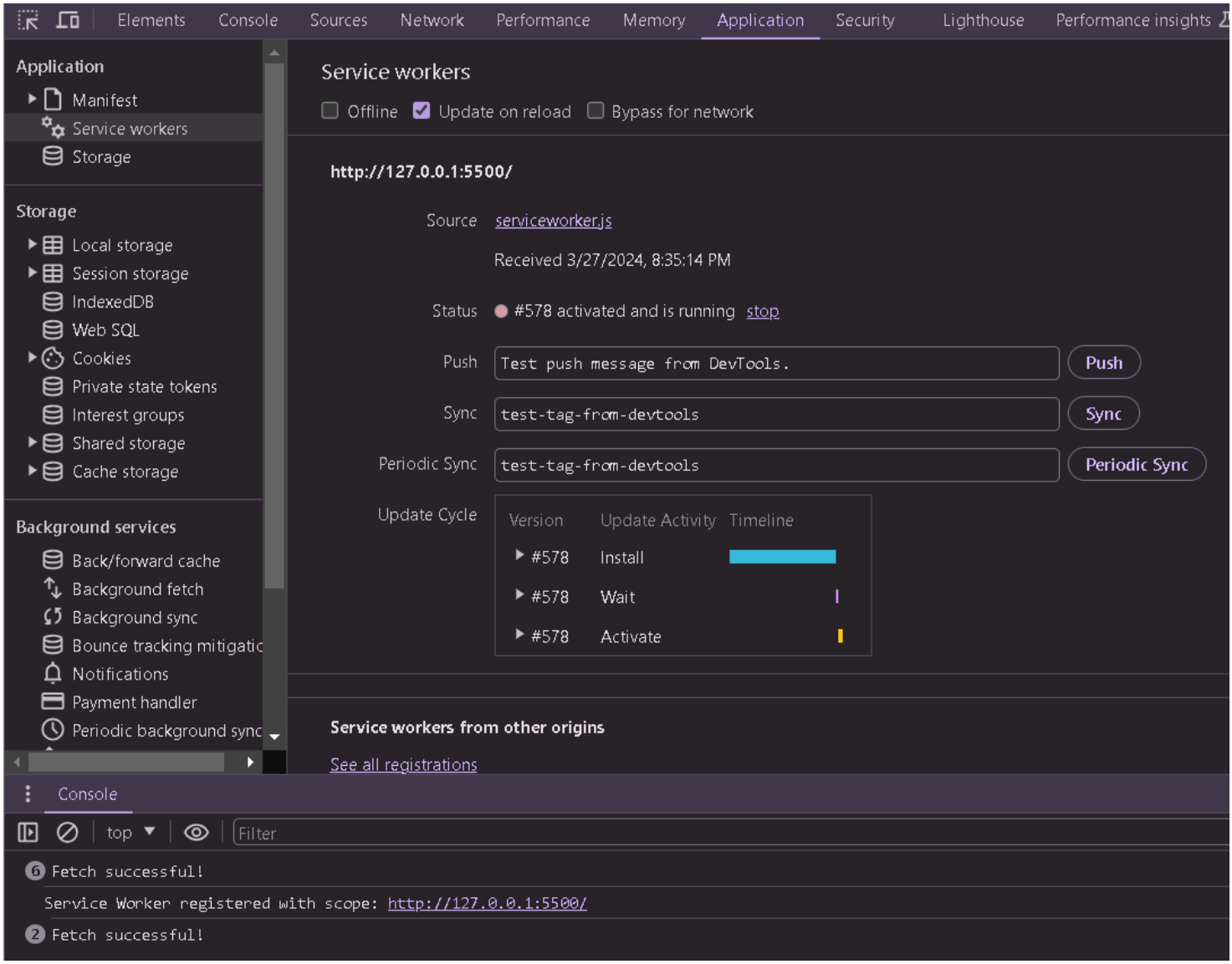
```
self.addEventListener("install", function (event) {
    event.waitUntil(preLoad());
    });

//Fetch event Listener
    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));
    });

//Sync event listener
    self.addEventListener("sync", (event) => {
    if (event.tag === "syncMessage") {
    console.log("Sync successful!");
    }
```

```
    });

//Push event listener
    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("Avengers!!!Assemble", {
    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',
        '/netflixstyles.css',
        '/app.js',
        '/icon-192x192.jpg',
        '/icon-512x512.jpg',
        '/manifest.json',
        '/serviceworker.js'
    ]);
    });
    };
    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);
    });
    });
```

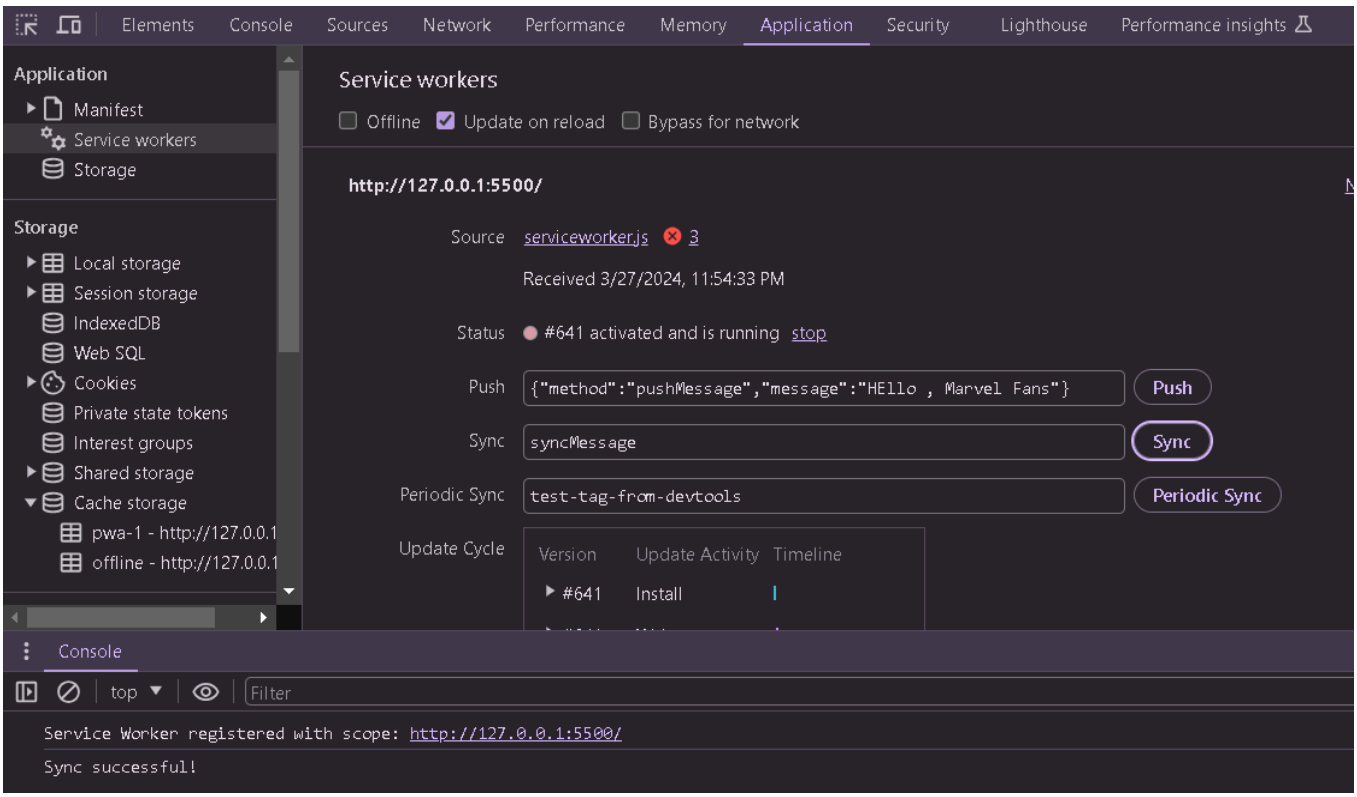
```
};
var returnFromCache = function (request) {
return caches.open("offline").then(function (cache) {
return cache.match(request).then(function (matching) {
if (!matching || matching.status == 404) {
return cache.match("offline.html");
} else {
return matching;
}
});
});
};
var addToCache = function (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:

1. Fetch

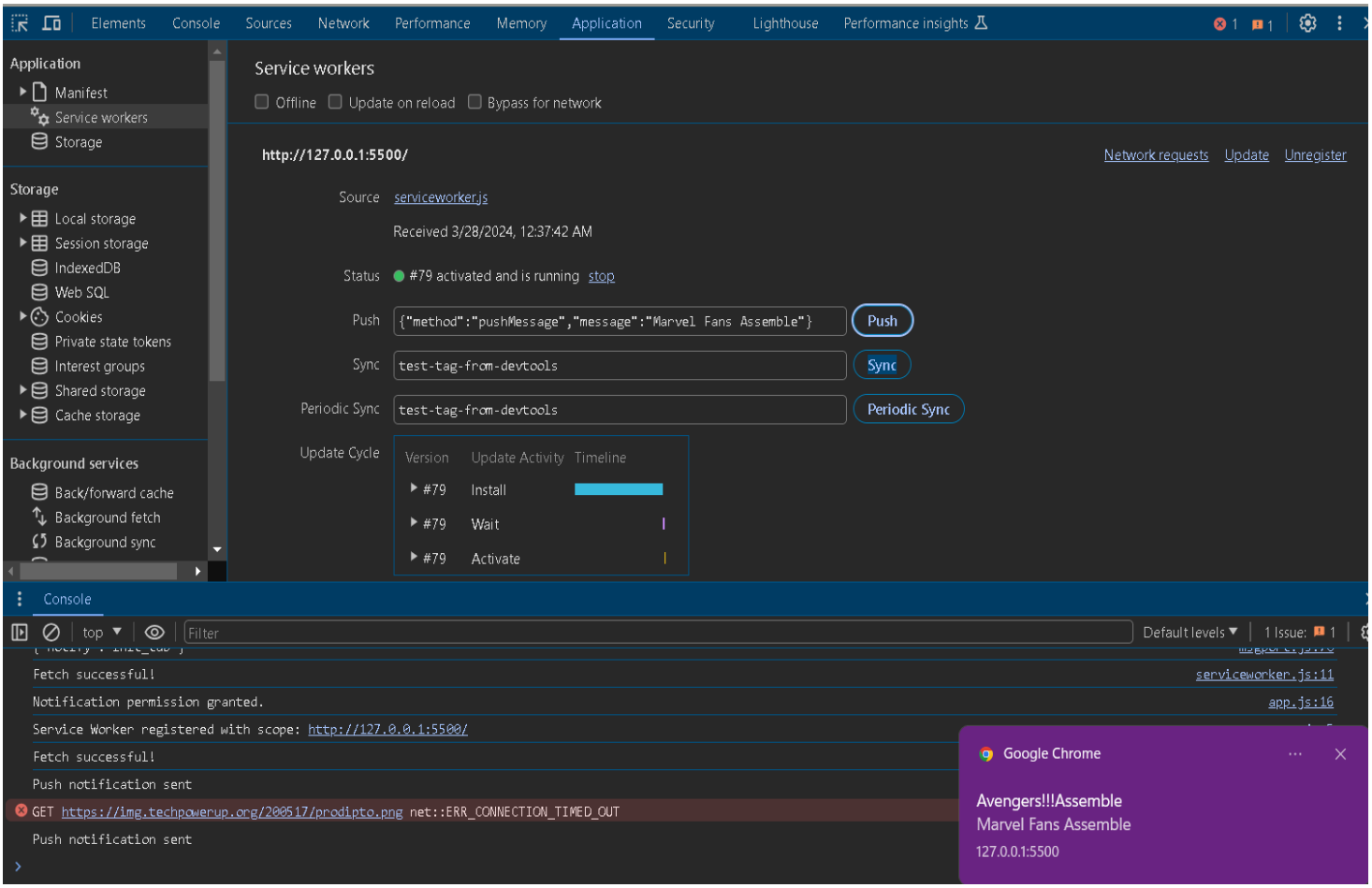

Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Performance insights
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
IndexedDB
Web SQL
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigatic
Notifications
Payment handler
Periodic background sync
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Source
serviceworker.js
Received 3/27/2024, 8:35:14 PM
Status
#578 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
#578
Install
#578
Wait
#578
Activate
Service workers from other origins
See all registrations
Console
top
Filter
Fetch successful!
Service Worker registered with scope: http://127.0.0.1:5500/
Fetch successful!

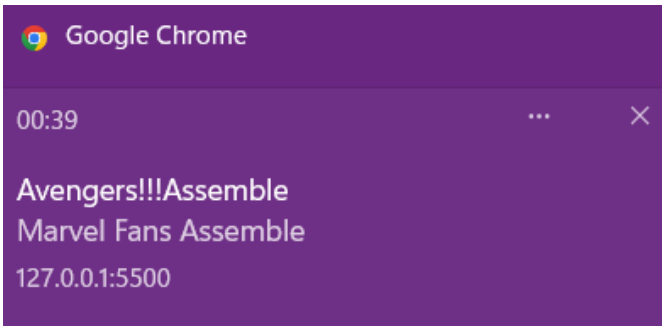
2. Sync



3. Push Notification:

Project Title: Flutter- Google meet clone/ PWA - Netflix website **Roll No. 23**





MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	23
Name	Mrudula Gotmare
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

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

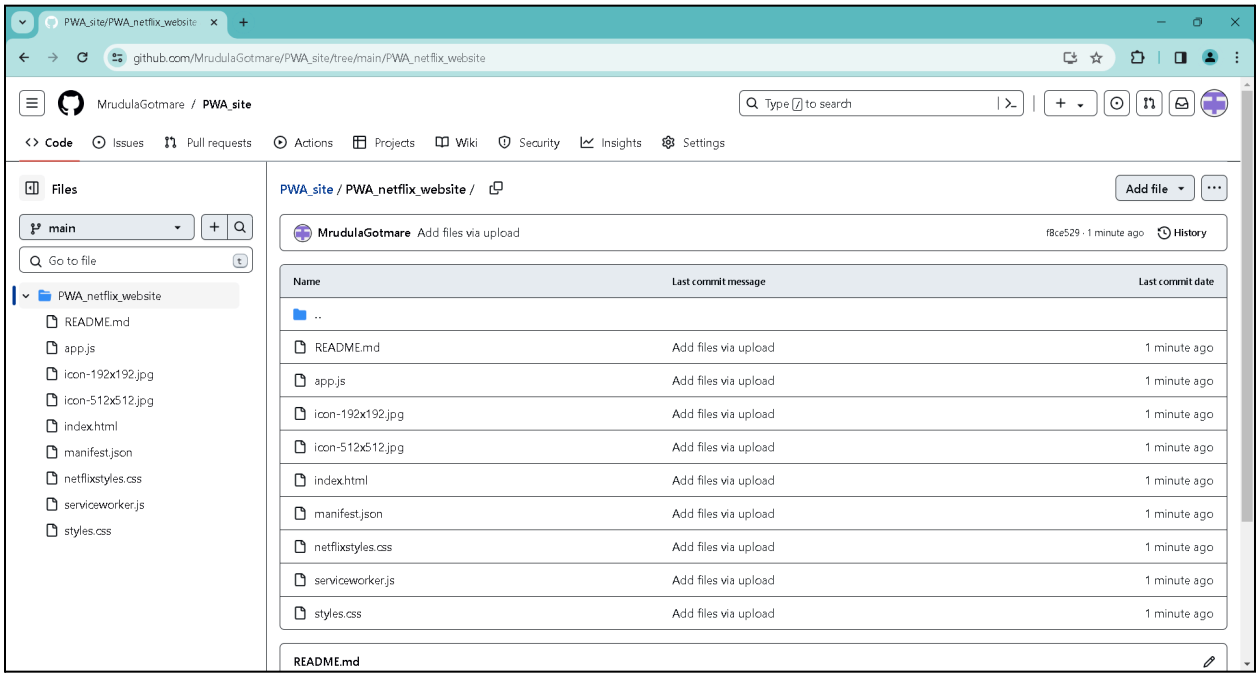
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator.

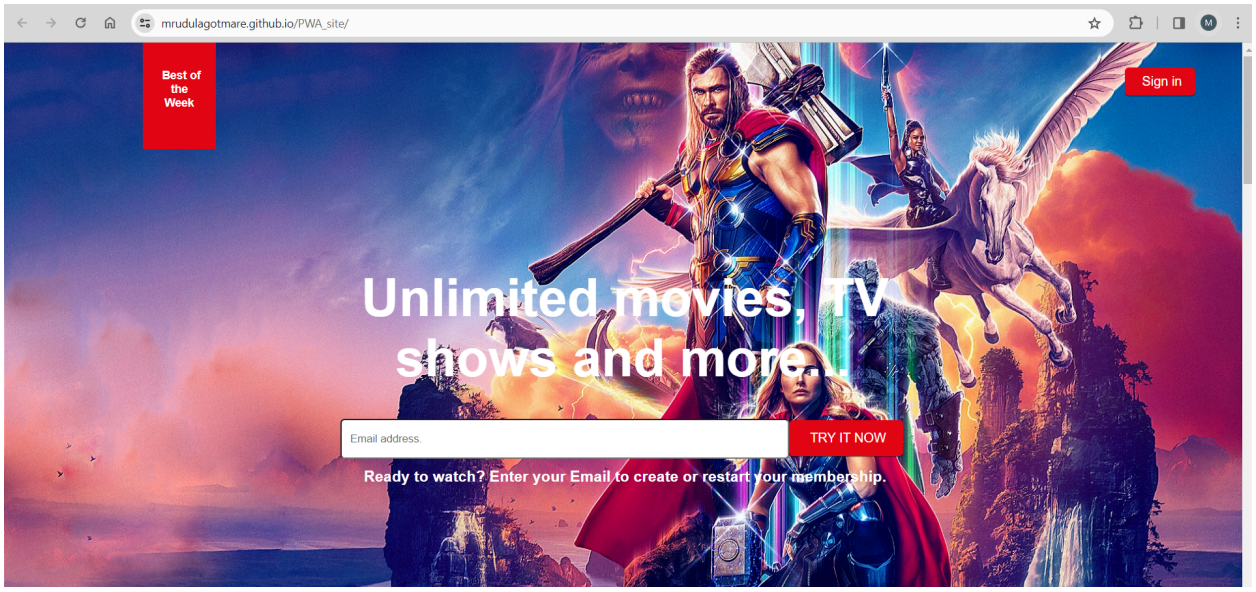
Link to our GitHub repository:

Github repo : https://github.com/MrudulaGotmare/PWA_site
Pages link : https://mrudulagotmare.github.io/PWA_site/

Project Title: Flutter- Google meet clone/ PWA - Netflix website **Roll No. 23**

Github Screenshot:





MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	23
Name	Mrudula Gotmare
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	15

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

Theory :

Google Lighthouse :

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

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

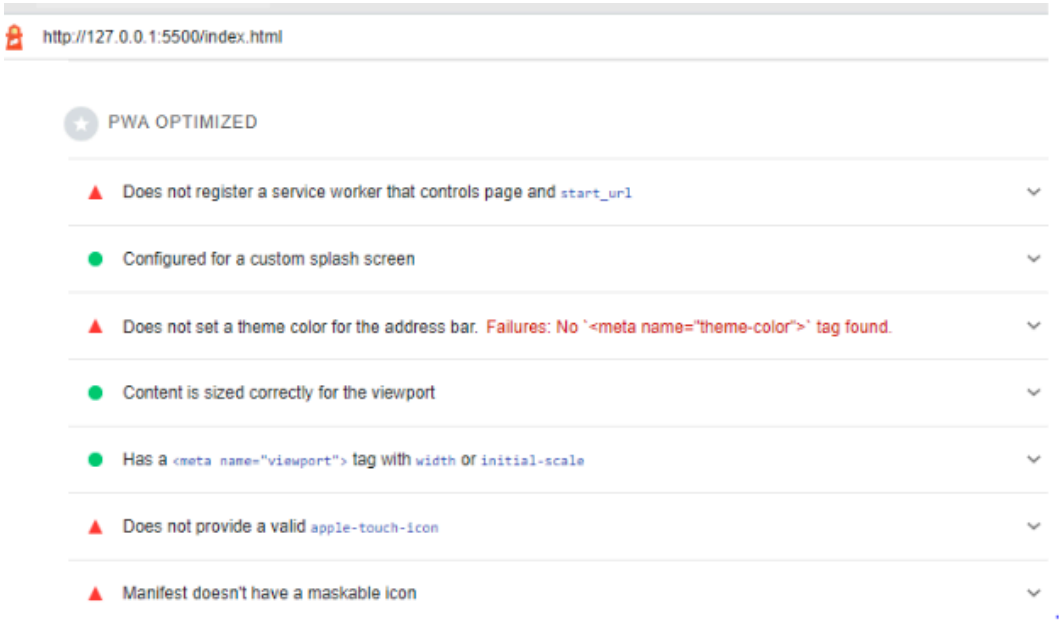
Key Features and Audit Metrics

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

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

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

Changes made to the code :



For theme color add a meta tag in index.html-

<meta name="theme-color" content="#FFFFF"/>

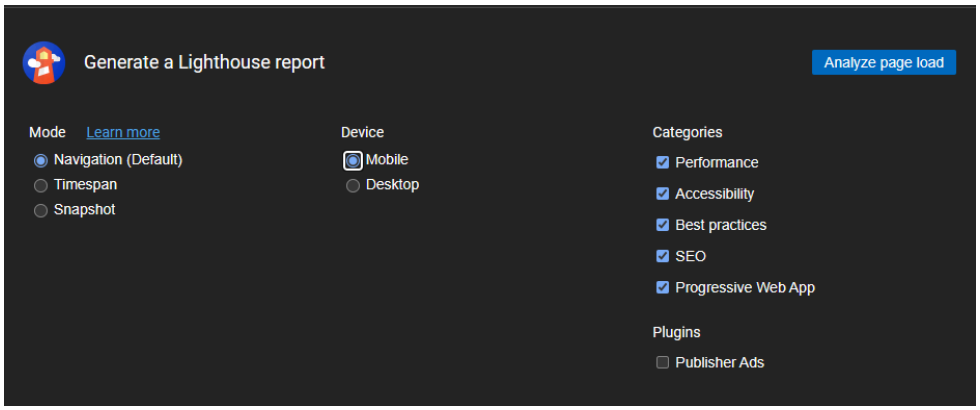
For a maskable icon add "purpose": "any maskable" to the icons in manifest.json file For apple touch icon add the following meta tag in index.html-

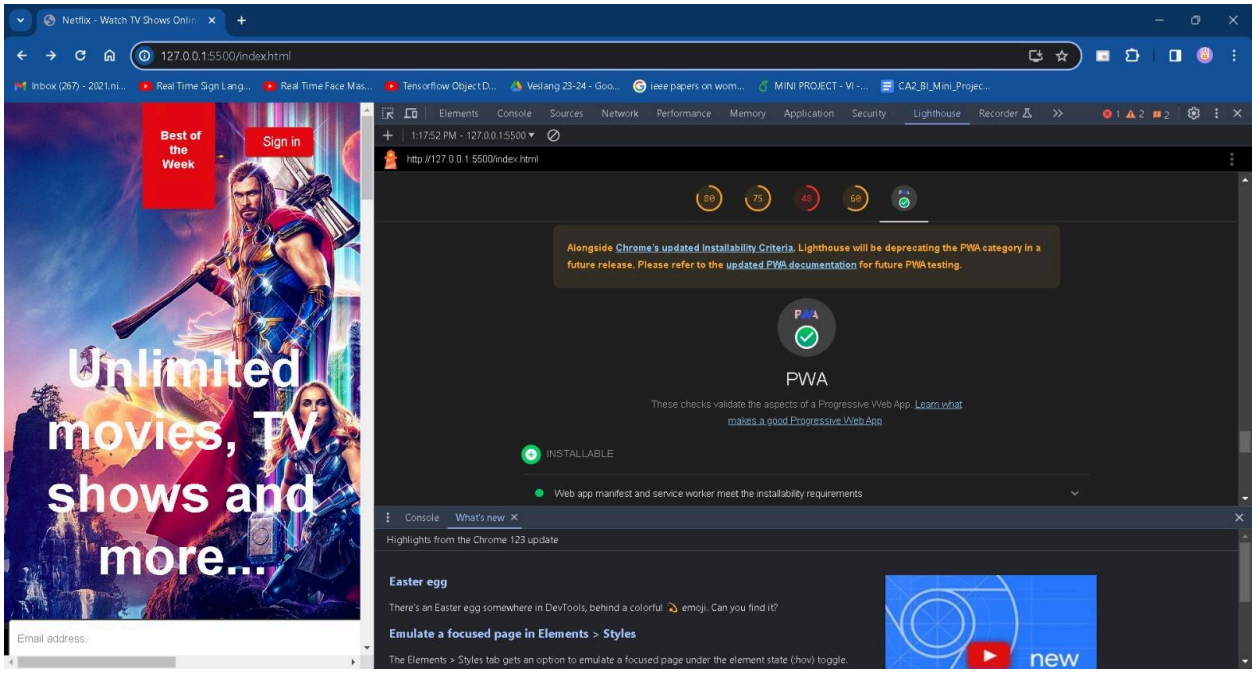
Changes in manifest.json

```
{
	"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":"assets\\images\\icons\\icon-192x192.png",
	"sizes":"192x192",
	"type":"image/png",
	"purpose":"any"
	},

	{
	"src":"assets\\images\\icons\\icon-512x512.png",
	"sizes":"512x512",
	"type":"image/png",
	"purpose":"maskable"
	}
]
}
```

Project Title: Flutter- Google meet clone/ PWA - Netflix website **Roll No. 23**

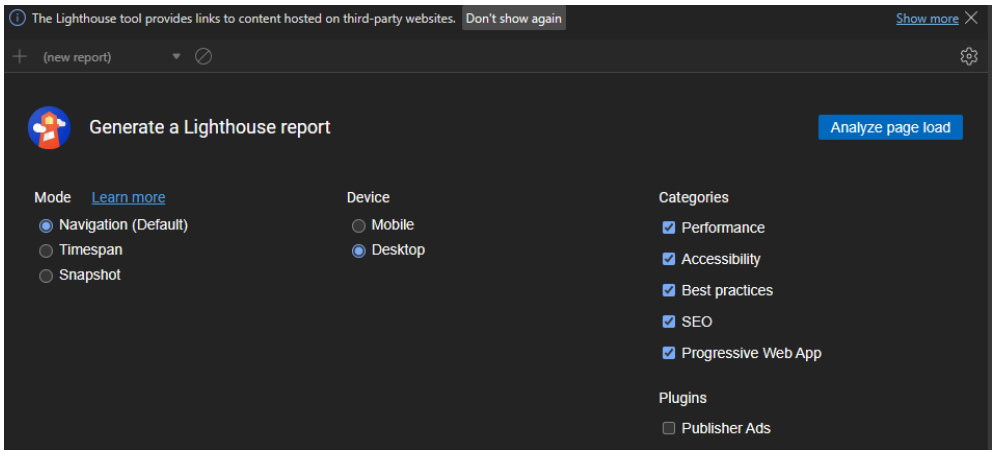
For mobile

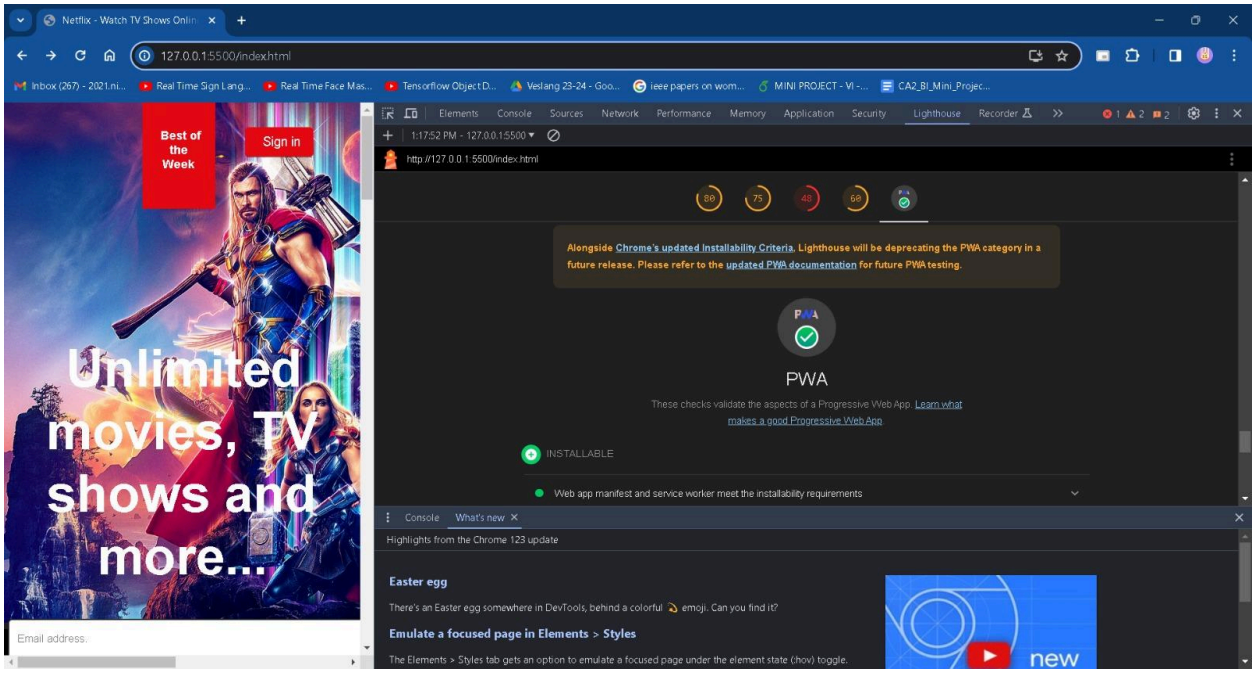




Project Title: Flutter- Google meet clone/ PWA - Netflix website **Roll No. 23**

For desktop





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.	23
Name	Mrudula Gotmare
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

Name:- Mrudula Gotmare
Class:- D15A
Roll no: 23
DATE:

MPL Assignment-1

1. Explain the key features and advantages of using Flutter for mobile app.

→ Flutter is a popular open-source UI software development toolkit created by Google for building natively compiled applications for mobile, web and desktop from a single codebase. Here are key features and advantages of using Flutter.

a) Single Codebase - allows you to write code once and deploy it on both ios and android platforms.

b) Hot Reload - Developers can see the results of code change in real time.

c) Rich widgets Library - Provides a comprehensive set of customizable widgets.

d) High performance - Flutter apps are compiled to native ARM code, providing high performance and smooth animations.

e) Expressive UI - Allows for the creation of expressive and flexible UI designs, making it easy to implement.

DATE:

2. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community.

→ Flutter differs from traditional approaches in several ways:

- Flutter is different than most other options for building mobile apps because it doesn't rely on web browser technology nor the set of widgets that ship with each device. Instead, Flutter uses its own high-performance rendering engine to draw widgets.

Here are some reasons that why Flutter is gaining popularity with developers.

a) Dart programming - It uses dart programming language. which is compiled into native code, giving flutter applications a significant performance advantage.

b) Easy to use - There are widgets, which allow developers to see the code in a readable format. Save time on UI development.

c) Support - The community is more active, which means more resources are available for flutter development.

DATE:

3. Widget Tree concept.

→ The widget tree is precisely what it sounds like - a tree structure where each node is a widget. This structure determines how your app's UI is organized and displayed. Widgets are arranged hierarchically, forming a parent-child relationship.

4. Explain How widget composition is assigned to build complex UI.

→ Widget composition is a fundamental concept in Flutter that allows developers to build complex UI by combining and nesting simple widgets together. Here's how widget composition works.

- Basic widgets
- Nesting widgets
- Custom widgets
- Widget Trees

5. Provide examples of commonly used widgets and their roles in creating a widget tree.

→ a) Text - A text widget holds some text to display on the screen.

```
eg:-  new Text(
        'Hello', textAlign : TextAlign.center,
        style : new TextStyle( fontWeight : bold ),
      )
```

DATE:

b) Button

eg:- new FlatButton(
child: Text("click here"),
onPressed: () {};
);

c) Image

eg: Image.asset('assets/computer.png')

6. Discuss the importance of state management in Flutter.

→ State management in Flutter is a critical concept for building robust, responsive and effective mobile application. It refers to the management and manipulation of data within an app to ensure that the UI accurately reflect the current state of the application. Flutter, as a reactive framework, offers various approaches to handling and updating state data.

7. Compare and contrast different state management approach available in Flutter such as setState, Provider, and RiverPod.

→ In Flutter, managing application state effectively is crucial for building reactive, scalable, and maintainable UIs. Here's a breakdown of the prominent approaches.

a) setState - The most fundamental approach, ideal for managing simple, widget-specific state. Data is directly in widget itself.

DATE:

b) Provider - A popular dependency injection and state management package that builds upon InheritedWidgets. Provides a clear structure for managing global and scoped state.

c) Riverpod - A dependency injection and state management library inspired by Redux, offering reactive state management and dependency injection. Emphasizes immutability and minimizes boilerplate.

8. Explain the process of integrating Firebase with a Flutter application.

→
- Ensure you have the Flutter CLI installed and a Firebase project created.
- Install the necessary FlutterFire plugins using `flutter pub add` commands.
- Register your app in the Firebase console.
- Import the appropriate plugins in dart code.
- Initialize Firebase in your main.dart file.

9. Discuss the benefits of using Firebase as a backend solution.

→
- Fast and global hosting -
- Flexible databases.
- Built-in authentication.
- Real-time communication.
- Analytics and insights.

DATE:

10. Discuss the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.

→ Several Firebase services are popular choices:

a) Firebase authentication - Enables secure user login and management using various methods like email/ password.

b) Cloud Firestore - Flexible, NoSQL database that scales efficiently with your app.

c) cloud storage - secure storage for various data types like user-generated content.

Data synchronization overview:

a) Cloud Firestore - Employs a publish-subscribe model with real-time updates. Clients listen for changes on specific documents.

b) Cloud storage - Files are uploaded/ downloaded directly, and clients manage synchronization logic if needed.

c) Firebase Realtime Database: Clients connect and listen for changes on specific database nodes.

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.	23
Name	Mrudula Gotmare
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

Name:- Mrudula Gotmare
Class:- D15A
Roll no:- 23 DATE:

PWA Assignment - 2

1. Define Progressive Web App (PWA) and explain it's significance in modern web development. Discuss the key charachteristics that differentiate PWAs from traditional mobile apps.

→ A Progressive Web App (PWA) is a website that looks and behaves as if it is a mobile app. PWAs are built to take advantage of native mobile device feature. It is buit with regular web technologies like html, css, and Javascript.

It's significance in modern web development is:
- Reduced development cost - By using existing web skills and a single codebase, PWA are quicker and cheaper to develop.

- Wider Reach - It works on any device with a modern browser, eliminating need for app store retrieval

- Easy updates - Updates to PWA happens automatically in the background, ensuring latest version.

- Improved user engagement - Features like push notifications and offline functionality enhance user experience.

8 - 4

DATE:

	PWA	Traditional Mobile App
Technology	• Web Technologies (HTML, CSS, Javascript)	• Native code (platform-specific).
Installation	• Installed from browser "Add to home" prompt.	• Downloaded from app stores.
Discoverability	• Accessible through search engines.	• Reliant on app store discovery.
Updatability	• Automatic background update	• Requires manual update.

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.

→ Responsive web design (RWD) is a web development approach that ensures a website adjusts its layout and functionality based on the device it's being viewed on. This creates an optimal user experience for desktops, tablets, smartphones, and any screen size in between.

Importance for PWA:-

- consistent user experience : Users expect a smooth experience regardless of the device they use. RWD ensures the PWA looks and functions well on all screens.

DATE:

- Improved Accessibility - RWD makes PWA accessible to a wider audience using devices with varying screen sizes and capabilities.

- Search Engine Optimization - Google and other search engines favor websites that offer a good user experience on all devices.

While RWD is a umbrella term, there can be confusion with other related design approaches.

- Responsive Web Design (RWD): RWD uses a single codebase that adapts to different devices.

- Fluid Web Design: A specific layout within RWD that uses percentages and relative units to define element sizes. This allows elements to expand and contract based on the screen size. Fluid design is a core principle used to achieve RWD.

- Adaptive Web Design: This approach involves multiple fixed width layouts for different device categories. It requires more development work compared to RWD.

DATE:

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

→ A key player in this PWA universe is the "service worker". The service worker is a Javascript File that runs on a seperate thread apart from the one in which your usual websit Javascript file run.

Three phases of Lifecycle
a) Registration phase
b) Installation phase
c) Activation phase.

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

b) Installation
Once the service worker is succesfully 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 service worker will be installed:

DATE:

- A new service worker file
- A modified service worker file.

c) Activation

Once the installation phase is successful, the next phase is the activation phase. The service worker does not move into active state immediately. It can move into the activated state only in below cases:

- None of the pages use the service worker and are closed.
- There is no other service worker active on that page.

4. Explain the use of IndexedDB in the service worker for data storage.

— IndexedDB is a browser API for storing large amounts of structured data on the client-side. It functions like a NoSQL database, allowing you to store key-value pairs and organise data into object stores with indexes for efficient retrieval.

Benefits of using IndexedDB with service worker:

- Offline data access - Improves user experience by allowing interaction with the PWA even without an internet connection.
- Faster load times - Cached data retrieval from IndexedDB can be served much faster than fetching it from the server again.

DATE:

- Improved reliability - Reduces dependences on a constant network connection, making the PWA more reliable.
- Security - IndexedDB provides a secure storage mechanism within the browser's sandbox environment.
- Data management - PWA should have mechanisms to manage storage space and avoid excessive data accumulation in IndexedDB.

Project Title: Flutter- Google meet clone/ PWA - Netflix website **Roll No. 23**