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 **Shubham R. Nakashe** 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

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

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers **:** Mrs. Kajal Jewani.

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

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

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	17/01	24/01	15
2.	To design Flutter UI by including common widgets.	LO2	24/01	31/01	15
3.	To include icons, images, fonts in Flutter app	LO2	31/01	07/02	15
4.	To create an interactive Form using form widget	LO2	07/02	14/02	15
5.	To apply navigation, routing and gestures in Flutter App	LO2	14/02	21/02	15
6.	To Connect Flutter UI with fireBase database	LO3	21/02	06/03	15
7.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4	06/03	29/03	15
8.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA	LO5	13/03	29/03	15
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5	20/03	29/03	15
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5	27/03	29/03	15
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6	27/03	29/03	15
12.	Assignment-1	LO1,LO2 ,LO3	28/01	05/02	05
13.	Assignment-2	LO4,LO5 ,LO6	14/03	21/03	04

MAD & PWA Lab
Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	38
Name	Shubham Raghuvir Nakashe
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15

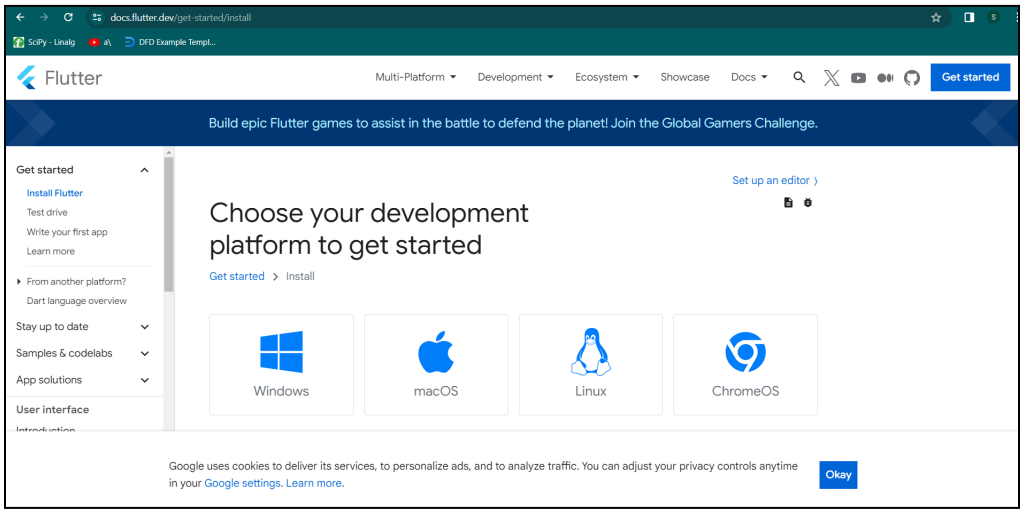
Aim -

To install and configure the Flutter Environment

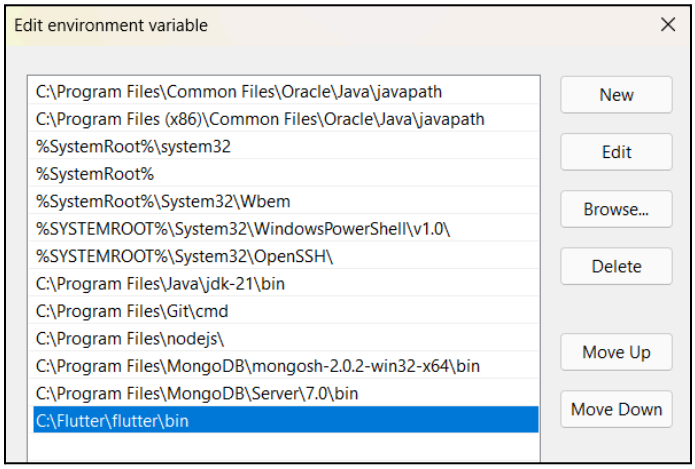
Code -

Installation

1) Installing the flutter from its official website



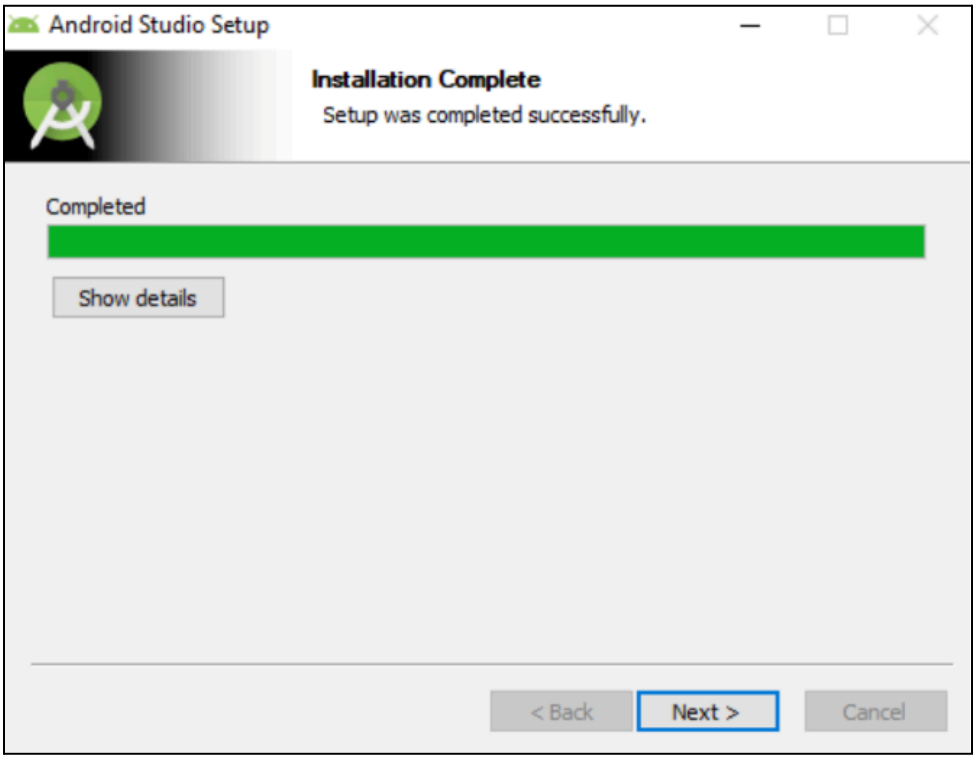
2) Clicking on windows installation and downloading the suitable zip that is compatible with the system
3) Extracting the zip and copying its bin path to configure it in environment variables



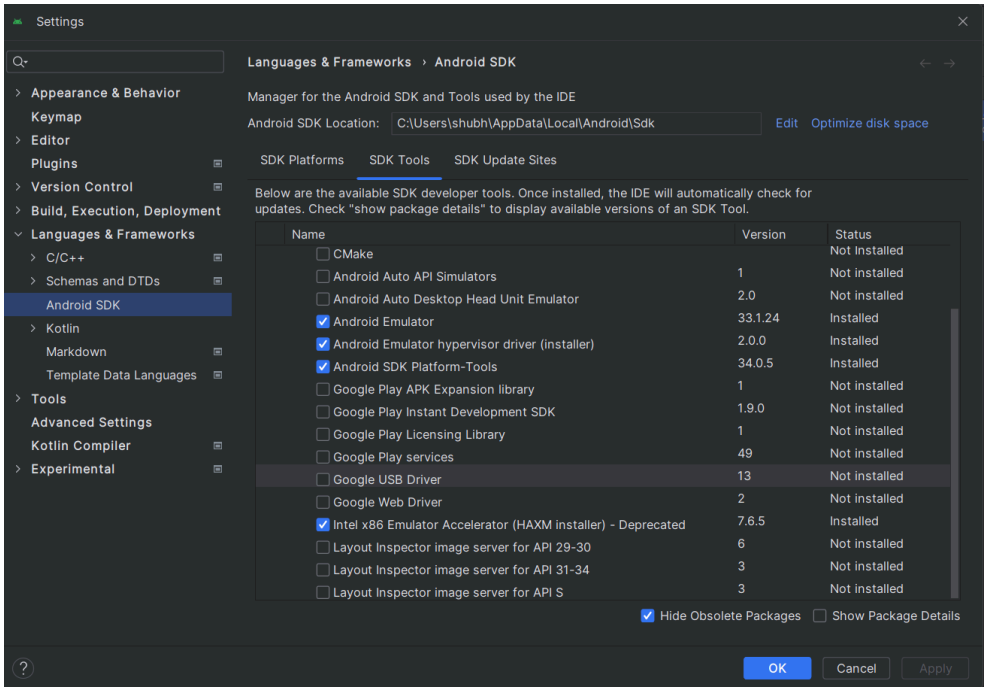
4) Running the “flutter doctor” command in command prompt to check the flutter installation and requirements

```
Command Prompt - flutter d
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Users\shubh>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-US)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[√] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[√] Chrome - develop for the web
[X] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows apps.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[√] Android Studio (version 2023.1)
[√] VS Code (version 1.85.2)
[√] Connected device (3 available)
[√] Network resources

! Doctor found issues in 1 category.
```

5) Installation of android studio



6) Installing the SDK packages in Andriod Studio

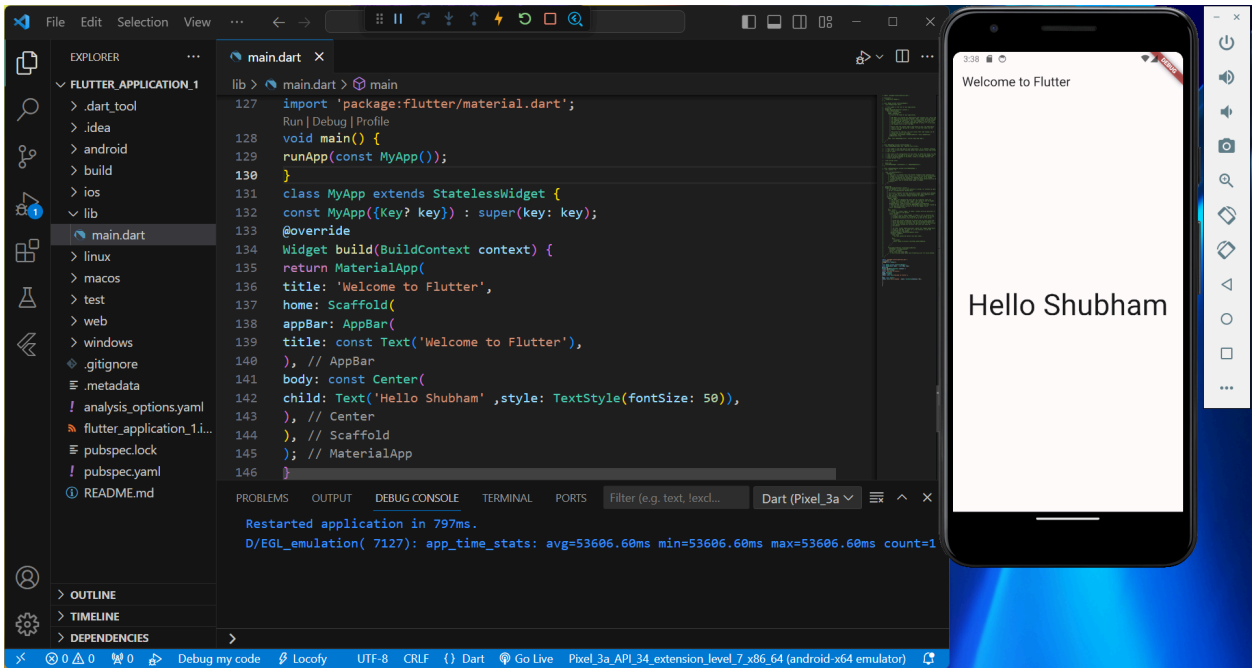


HELLO SHUBHAM CODE -

```
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: const Center(
child: Text('Hello Shubham' ,style: TextStyle(fontSize: 50)),
),
),
);
}
```

}

OUTPUT


File Edit Selection View
EXPLORER
FLUTTER_APPLICATION_1
main.dart
lib > main.dart > main
import 'package:flutter/material.dart';
Run | Debug | Profile
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 Shubham' ,style: TextStyle(fontSize: 50)),
), // Center
), // Scaffold
); // MaterialApp
}
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Restarted application in 797ms.
D/EGL_emulation(7127): app_time_stats: avg=53606.60ms min=53606.60ms max=53606.60ms count=1
OUTLINE
TIMELINE
DEPENDENCIES
Debug my code
Locofy
UTF-8
CRLF
Dart
Go Live
Pixel_3a_API_34_extension_level_7_x86_64 (android-x64 emulator)
Welcome to Flutter
Hello Shubham

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	38
Name	Shubham Raghuvir Nakashe
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

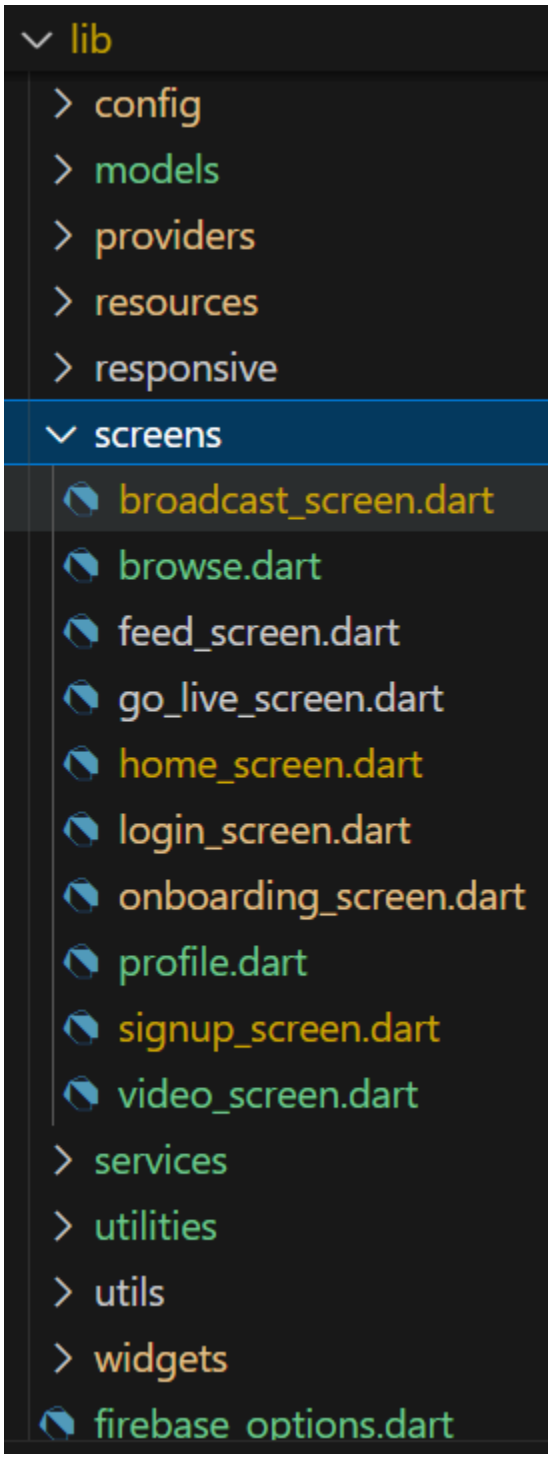
Aim: To design Flutter UI by including common widgets.

Theory:

Some common widgets in Flutter include:

- Container: A versatile widget that can contain other widgets and provides properties for styling, padding, margin, and more.
- Text: Displays a string of text with options for styling such as font size, color, weight, and alignment.
- Row and Column: Used to arrange child widgets horizontally (Row) or vertically (Column).
- Image: Displays an image from various sources such as assets, the internet, or memory.
- AppBar and Scaffold: Widgets for creating an app bar and a basic layout structure for your app.
- Button widgets (ElevatedButton,TextButton,IconButton): Used for user interaction and triggering actions.
- TextField: Allows users to input text with options for customization and validation.

FOLDER STRUCTURE:



CODE:

Profile.dart

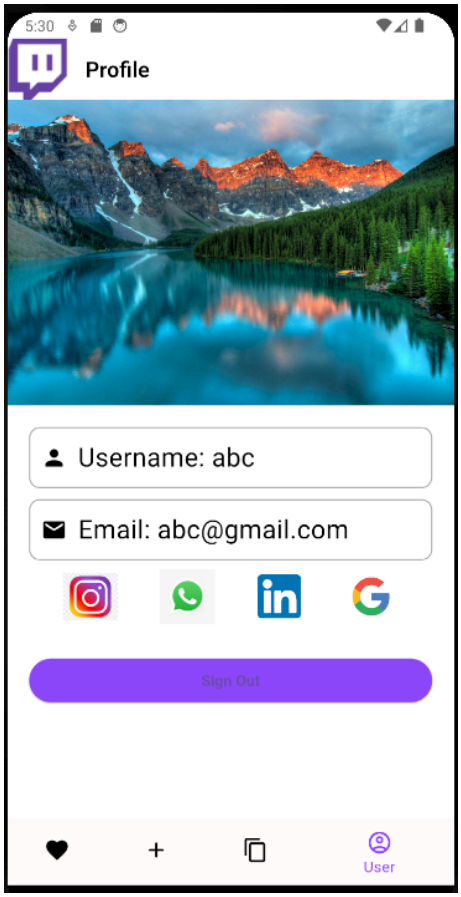
```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:twitch_clone/providers/user_provider.dart';
import 'package:twitch_clone/screens/onboarding_screen.dart';
import 'package:twitch_clone/widgets/custom_button.dart';

class ProfileScreen extends StatelessWidget {
  static const String routeName = '/profile';

  Future<void> signOut(BuildContext context) async {
    try {
      await FirebaseAuth.instance.signOut();
      Navigator.pushReplacementNamed(context, OnBoardingScreen.routeName);
    } catch (e) {
      print("Error signing out: $e");
    }
  }

  @override
  Widget build(BuildContext context) {
    final userProvider = Provider.of<UserProvider>(context);
    return Scaffold(
      appBar: AppBar(
        title: Text('Profile'),
        leading: Container(
          child: Image.asset('assets/twitch-icon.png'),
          width: 12,
          height: 12,
        ),
      ),
      body: Stack(
        children: [
          Container(
            height: MediaQuery.of(context).size.height * 0.35,
            decoration: BoxDecoration(
              image: DecorationImage(
                image: AssetImage('assets/nature2.jpg'),
                fit: BoxFit.cover,
              ),
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(20.0),
```

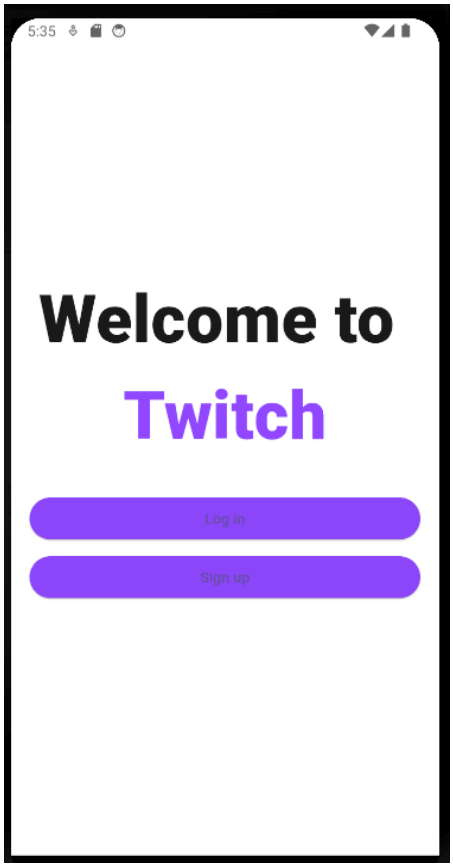
```
child: Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
    SizedBox(height: MediaQuery.of(context).size.height * 0.35),
    Container(
      padding: EdgeInsets.all(10),
      decoration: BoxDecoration(
        border:
            Border.all(color: Colors.grey),
        borderRadius: BorderRadius.circular(
            10),
      ),
      child: Row(
        children: [
          Icon(
            Icons.person,
            size: 24,
            color: Colors.black,
          ),
          SizedBox(width: 10),
          Text(
            'Username: ${userProvider.user.username}',
            style: TextStyle(
              fontSize: 24,
              color: Colors.black,
            ),
          ),
        ],
      ),
    ),
    SizedBox(height: 10),
    Container(
      padding: EdgeInsets.all(10),
      decoration: BoxDecoration(
        border:
            Border.all(color: Colors.grey),
        borderRadius: BorderRadius.circular(
            10),
      ),
      child: Row(
        children: [
          Icon(
            Icons.email,
            size: 24,
            color: Colors.black,
          ),
```

```
        SizedBox(width: 10),
        Text(
          'Email: ${userProvider.user.email}',
          style: TextStyle(
            fontSize: 24,
            color: Colors.black,
          ),
        ),
      ],
    ),
  ),
  Row(
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    children: [
      IconButton(
        icon: Container(
          width: 50,
          height: 50,
          child: Image.asset('assets/instagram_icon.jpg'),
        ),
        onPressed: () {

        },
      ),
      IconButton(
        icon: Container(
          width: 50,
          height: 50,
          child: Image.asset('assets/whatsapp_icon.jpg'),
        ),
        onPressed: () {

        },
      ),
      IconButton(
        icon: Container(
          width: 40,
          height: 40,
          child: Image.asset('assets/linkedin_icon.png'),
        ),
        onPressed: () {
        },
      ),
      IconButton(
        icon: Container(
          width: 50,
```

```
                height: 50,
                child: Image.asset('assets/google_icon.png'),
              ),
              onPressed: () {
              },
            ),
          ],
        ),
        SizedBox(height: 20),
        CustomButton(
          text: 'Sign Out',
          onTap: () => signOut(context),
        ),
      ],
    ),
  ),
 ],
 ),
 );
 }
}
```

OUTPUT:





MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	38
Name	Shubham Raghuvir Nakashe
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: To include icons, images, fonts in Flutter app

Theory:

- Fonts:
  You can use custom fonts in your Flutter app by adding the font files (e.g., `.ttf` or `.otf`) to your project and then specifying them in your `pubspec.yaml` file. Once added, you can use these fonts in your app's text widgets by specifying the font family.

- Images:
  Flutter supports various image formats (e.g., PNG, JPEG, GIF) and provides widgets like `Image` and `AssetImage` to display images in your app. You can load images from assets or network URLs.

- Videos:
  Flutter provides the `video_player` package for playing videos in your app. You can use this package to display and control video playback, including features like play, pause, seek, and fullscreen.

- Icons:
  Flutter includes a set of built-in icons from popular icon packs like Material Icons and Cupertino Icons. You can use these icons in your app using the `Icon` widget. Additionally, you can use custom icons by importing and referencing them in your app.

PUBSPEC.yaml

```
assets:
  - assets/twitch-icon.png
  - assets/nature.jpg
  - assets/whatsapp_icon.jpg
  - assets/linkedin_icon.png
  - assets/nature2.jpg
  - assets/google_icon.png
  - assets/instagram_icon.jpg
```

CODE:

```
import 'package:flutter/material.dart';
import 'package:twitch_clone/models/channel_model.dart';
import 'package:twitch_clone/models/video_model.dart';
import 'package:twitch_clone/screens/video_screen.dart';
import 'package:twitch_clone/services/api_service.dart';

class browse extends StatefulWidget {
  @override
  _browseState createState() => _browseState();
}

class _browseState extends State<browse> {
  List<String> channelIds = [
    'UCeVMnSShP_Iviwkknt83cww',
    'UC1XBh-m27kkgwLAwu_SRJBg',
    'UCq-Fj5jknLsUf-MWSy4_brA',
    'UCBwmMxybNva6P_5VmxjzwqA',
    'UCsooa4yRKGN_zEE8iknghZA',
    'UCOQNJjhXwvAScuELTT_i7cQ',
    'UCBqFKDipsnzvJdt6UT0lMIg',
    'UCPXGFu34px86DdXwocV-bYA',
    'UCX8pnu3DYUnx8qy8V_c6oHg',
    'UClfos9f7uDdoun8ZyE9jYFg',
    'UC7eHZXheF8nVOfwB2PEslMw'

    // Add more channel IDs as needed
  ];
  List<Channel> channels = [];
  bool _isLoading = false;

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

  _initChannels() async {
    List<Channel> loadedChannels = [];
    for (String channelId in channelIds) {
      Channel channel =
          await APIService.instance.fetchChannel(channelId: channelId);
      loadedChannels.add(channel);
    }
    setState(() {
      channels = loadedChannels;
```

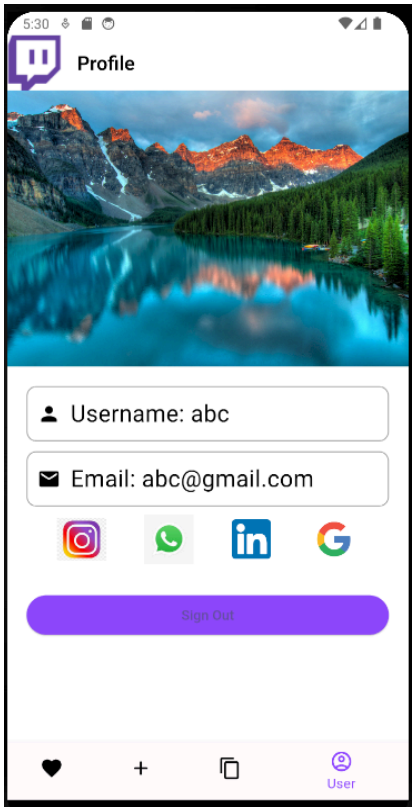
```
  });
 }

 Widget _buildProfileInfo(Channel channel) {
  return Container(
    margin: EdgeInsets.all(20.0),
    padding: EdgeInsets.all(20.0),
    height: 100.0,
    decoration: BoxDecoration(
      color: Colors.white,
      boxShadow: [
        BoxShadow(
          color: Colors.black12,
          offset: Offset(0, 1),
          blurRadius: 6.0,
        ),
      ],
    ),
    child: Row(
      children: <Widget>[
        CircleAvatar(
          backgroundColor: Colors.white,
          radius: 35.0,
          backgroundImage: NetworkImage(channel.profilePictureUrl),
        ),
        SizedBox(width: 12.0),
        Expanded(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.start,
            children: <Widget>[
              Text(
                channel.title,
                style: TextStyle(
                  color: Colors.black,
                  fontSize: 20.0,
                  fontWeight: FontWeight.w600,
                ),
                overflow: TextOverflow.ellipsis,
              ),
              Text(
                '${channel.subscriberCount} subscribers',
                style: TextStyle(
                  color: Colors.grey[600],
                  fontSize: 16.0,
                  fontWeight: FontWeight.w600,
```

```
              ),
              overflow: TextOverflow.ellipsis,
            ),
          ],
        ),
      )
    ],
  ),
 );
}

_buildVideo(Video video) {
 return GestureDetector(
  onTap: () => Navigator.push(
   context,
   MaterialPageRoute(
    builder: (_) => VideoScreen(id: video.id),
   ),
  ),
  child: Container(
   margin: EdgeInsets.symmetric(horizontal: 20.0, vertical: 5.0),
   padding: EdgeInsets.all(10.0),
   height: 140.0,
   decoration: BoxDecoration(
    color: Colors.white,
    boxShadow: [
     BoxShadow(
      color: Colors.black12,
      offset: Offset(0, 1),
      blurRadius: 6.0,
     ),
    ],
   ),
   child: Row(
    children: <Widget>[
     Image(
      width: 150.0,
      image: NetworkImage(video.thumbnailUrl),
     ),
     SizedBox(width: 10.0),
     Expanded(
      child: Text(
       video.title,
       style: TextStyle(
        color: Colors.black,
        fontSize: 18.0,
```

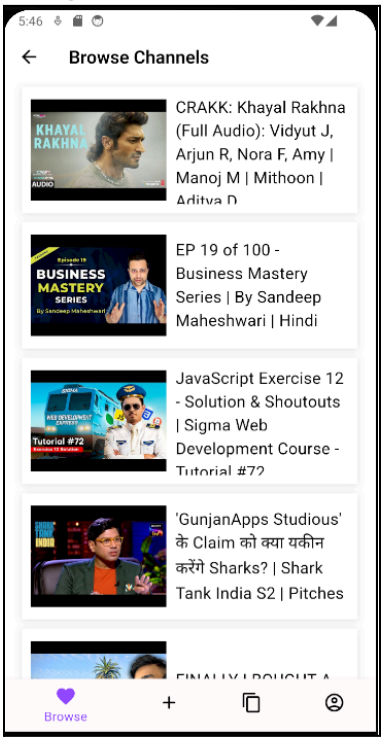
```
            ),
          ),
        ),
      ],
    ),
  ),
 );
}

_loadMoreVideos(Channel channel) async {
 _isLoading = true;
 List<Video> moreVideos = await APIService.instance
     .fetchVideosFromPlaylist(playlistId: channel.uploadPlaylistId);
 List<Video> allVideos = channel.videos..addAll(moreVideos);
 setState(() {
  channel.videos = allVideos;
 });
 _isLoading = false;
}

@override
Widget build(BuildContext context) {
// Flatten the list of videos from all channels
List<Video> allVideos = [];
for (Channel channel in channels) {
 allVideos.addAll(channel.videos);
}

// Shuffle all the videos
allVideos.shuffle();

return Scaffold(
 appBar: AppBar(
  title: Text('Browse Channels'),
 ),
 body: ListView.builder(
  itemCount: allVideos.length,
  itemBuilder: (BuildContext context, int index) {
   Video video = allVideos[index];
   return _buildVideo(video);
  },
 ),
);
}
 Widget _buildChannelWidget(Channel channel) {
 return NotificationListener<ScrollNotification>(
```

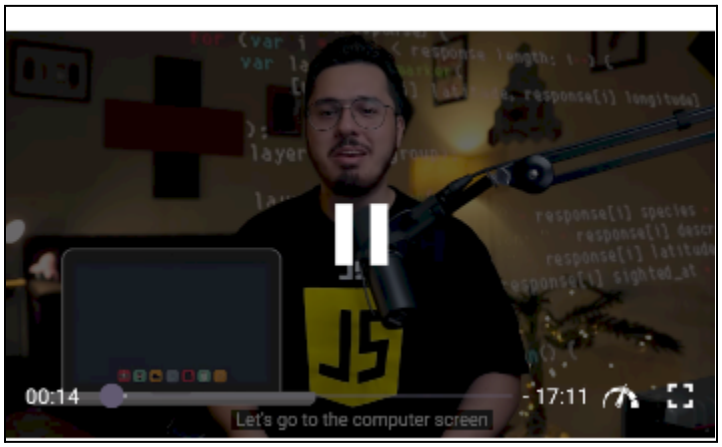
```
    onNotification: (ScrollNotification scrollDetails) {
      if (!_isLoading &&
          channel.videos.length != int.parse(channel.videoCount) &&
          scrollDetails.metrics.pixels ==
              scrollDetails.metrics.maxScrollExtent) {
        _loadMoreVideos(channel);
      }
      return false;
    },
    child: ListView.builder(
      shrinkWrap: true,
      physics: NeverScrollableScrollPhysics(),
      itemCount: 1 + channel.videos.length,
      itemBuilder: (BuildContext context, int index) {
        if (index == 0) {
          return Container(); // You can add additional information for the channel here
        }
        Video video = channel.videos[index - 1];
        return _buildVideo(video);
      },
    ),
  );
}}
```

OUTPUT:



Images and icons (twitch ,whatsapp,instagram,google etc .)





Videos

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	38
Name	Shubham Raghuvir Nakashe
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: To create an interactive Form using form widget

Theory:

In Flutter, a Form widget is used to manage a group of form fields and handle form submission. It provides methods to validate and save form field values.

CODE:

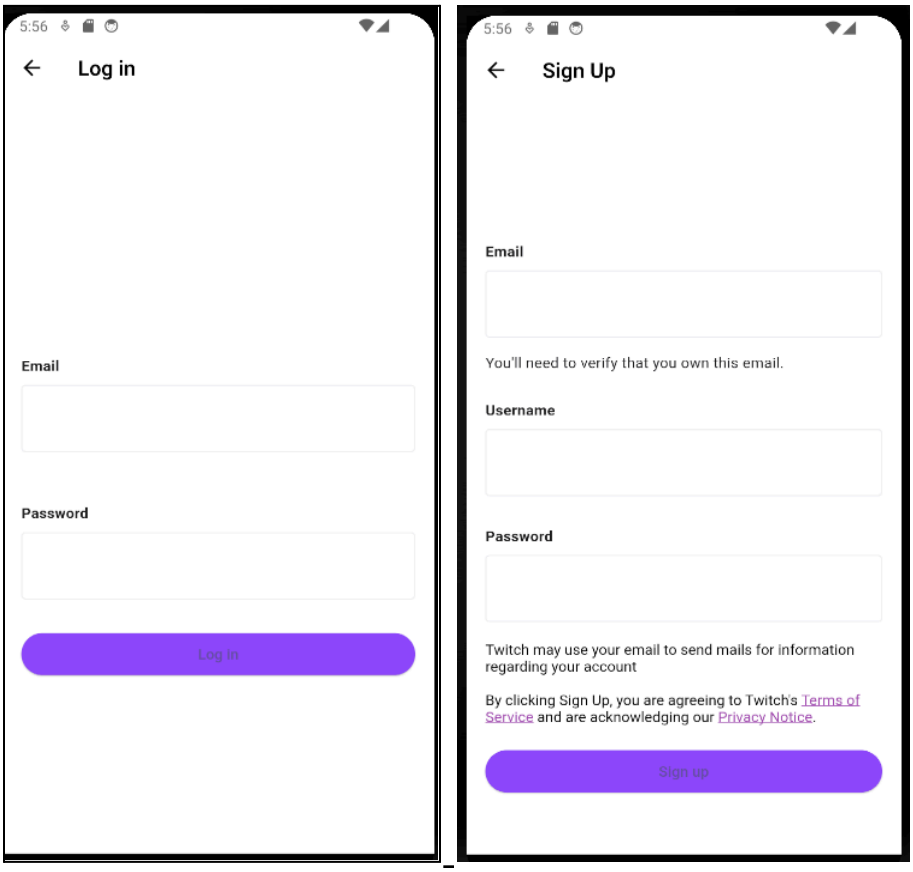
```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:twitch_clone/models/user.dart';
import 'package:twitch_clone/providers/user_provider.dart';
import 'package:twitch_clone/resources/auth_methods.dart';
import 'package:twitch_clone/responsive/responsive.dart';
import 'package:twitch_clone/screens/home_screen.dart';
import 'package:twitch_clone/screens/profile.dart';
import 'package:twitch_clone/widgets/custom_button.dart';
import 'package:twitch_clone/widgets/custom_text_field.dart';
import 'package:twitch_clone/widgets/loading_indicator.dart';

class SignupScreen extends StatefulWidget {
  static const String routeName = '/signup';

  const SignupScreen({super.key});

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

class _SignupScreenState extends State<SignupScreen> {
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  final TextEditingController _usernameController = TextEditingController();
  final AuthMethods _authMethods = AuthMethods();
  bool _isLoading = false;

  void signUpUser() async {
    void signUpUser() async {
      setState(() {
        _isLoading = true;
      });
      bool res = await _authMethods.signUpUser(context, _emailController.text,
          _usernameController.text, _passwordController.text);
      setState(() {
        _isLoading = false;
      });
```

```
    if (res) {
      final userProvider = Provider.of<UserProvider>(context, listen: false);
      userProvider.setUser(User(uid: '', username: _usernameController.text, email: ''));

       Navigator.pushReplacementNamed(context,ProfileScreen.routeName);
    }
  }
}

@override
void dispose() {
  _emailController.dispose();
  _passwordController.dispose();
  _usernameController.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  final size = MediaQuery.of(context).size;
  return Scaffold(
    appBar: AppBar(
      title: const Text('Sign Up'),
    ),
    body: _isLoading
        ? const LoadingIndicator()
        : Responsive(
            child: SingleChildScrollView(
              child: Padding(
                padding: const EdgeInsets.symmetric(horizontal: 18.0),
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    SizedBox(
                      height: size.height * 0.1,
                    ),
                    const Text(
                      "Email",
                      style: TextStyle(
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                    Padding(
                      padding: const EdgeInsets.symmetric(vertical: 8.0),
                      child: CustomTextField(controller: _emailController),
                    ),
```

```
                SizedBox(height: 6),
                Padding(
                  padding: const EdgeInsets.only(bottom: 5.0),
                  child: Text(
                    'You\'ll need to verify that you own this email.',
                    style: TextStyle(fontSize: 14),
                  ),
                ),
                const SizedBox(
                  height: 20,
                ),
                const Text(
                  "Username",
                  style: TextStyle(
                    fontWeight: FontWeight.bold,
                  ),
                ),
                Padding(
                  padding: const EdgeInsets.symmetric(vertical: 8.0),
                  child: CustomTextField(controller: _usernameController),
                ),
                const SizedBox(
                  height: 20,
                ),
                const Text(
                  "Password",
                  style: TextStyle(
                    fontWeight: FontWeight.bold,
                  ),
                ),
                Padding(
                  padding: const EdgeInsets.symmetric(vertical: 8.0),
                  child: CustomTextField(controller: _passwordController),
                ),
                SizedBox(height: 10),
                RichText(
                  text: const TextSpan(
                    text:
                        'Twitch may use your email to send mails for information regarding your
account \n \n',
                    style: TextStyle(color: Colors.black),
                    children: [
                      TextSpan(
                        text: 'By clicking Sign Up',
                        style: TextStyle(color: Colors.black),
                      ),
```

```
                    TextSpan(
                      text: ', you are agreeing to Twitch\'s ',
                    ),
                    TextSpan(
                      text: 'Terms of Service',
                      style: TextStyle(
                          color: Color.fromARGB(255, 162, 72, 177),
                          decoration: TextDecoration.underline),
                    ),
                    TextSpan(
                      text: ' and are acknowledging our ',
                    ),
                    TextSpan(
                      text: 'Privacy Notice',
                      style: TextStyle(
                          color: Color.fromARGB(255, 162, 72, 177),
                          decoration: TextDecoration.underline),
                    ),
                    TextSpan(
                      text: '.',
                    ),
                  ],
                ),
              ),
              const SizedBox(
                height: 20,
              ),
              CustomButton(
                text: "Sign up",
                onTap: signUpUser,
              )
            ],
          ),
        ),
      ),
    ),
  );
 }
}
```

OUTPUT:



LOGIN AND SIGNUP FORM

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	38
Name	Shubham Raghuvir Nakashe
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: To apply navigation, routing and gestures in Flutter App

Theory:

In Flutter, navigation gestures and routing are used to navigate between different screens or pages .

- Navigation: Flutter provides a `Navigator` widget that manages a stack of routes (screens/pages) in your app. You can push a new route onto the stack to navigate to a new screen and pop a route to go back to the previous screen.

- Gestures: Flutter supports various gestures for navigation, such as tapping, swiping, and dragging. These gestures can be used to trigger navigation actions, like going to the next or previous screen.

- Routing: Flutter uses a named routing system, where each route/screen has a unique name. You can define routes in your app's main `MaterialApp` widget using the `routes` property, or you can use the `Navigator` directly to navigate to a specific route.

- Navigation Stack: The `Navigator` manages a stack of routes, which represents the navigation history of your app. When you push a new route, it gets added to the top of the stack, and when you pop a route, it gets removed from the stack.

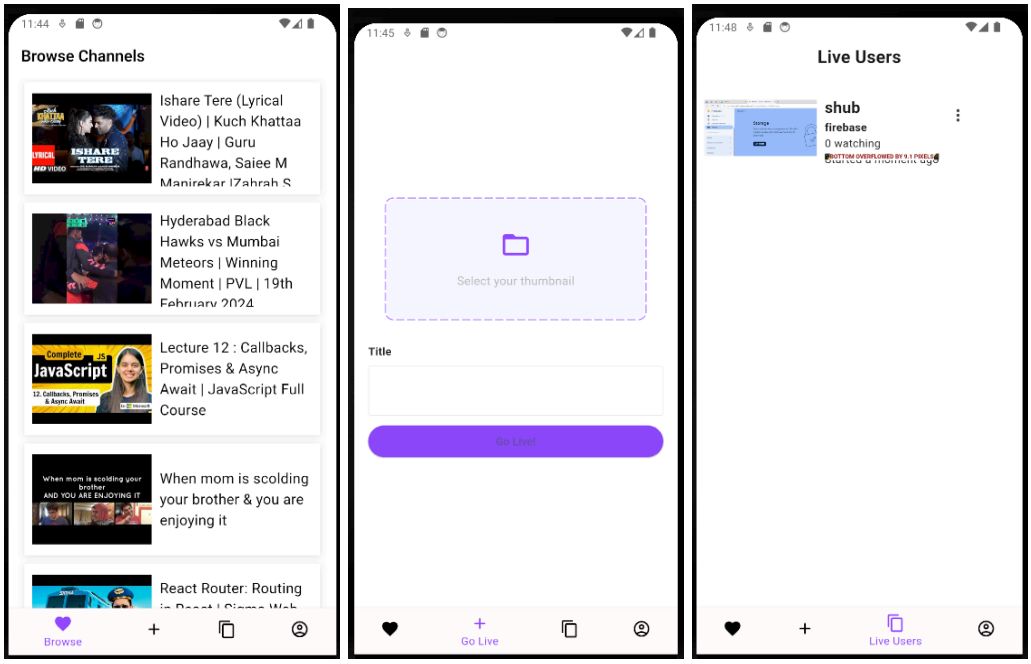
CODE:

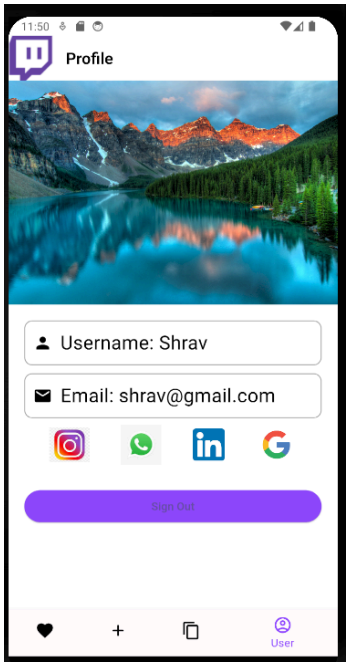
```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:twitch_clone/screens/browse.dart';
import 'package:twitch_clone/screens/feed_screen.dart';
import 'package:twitch_clone/screens/go_live_screen.dart';
import 'package:twitch_clone/screens/login_screen.dart';
import 'package:twitch_clone/screens/onboarding_screen.dart';
import 'package:twitch_clone/screens/profile.dart';
import 'package:twitch_clone/utils/colors.dart';
import 'package:twitch_clone/widgets/custom_button.dart';

class HomeScreen extends StatefulWidget {
  static String routeName = '/home';
```

```
  const HomeScreen({super.key});

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

class _HomeScreenState extends State<HomeScreen> {


  int _page = 0;
  List<Widget> pages = [];

  @override
  void initState() {
    super.initState();
    pages = [
      browse(),
      const GoLiveScreen(),
      const FeedScreen(),
      ProfileScreen(),

    ];
  }

  onPageChanged(int page) {
    setState(() {
      _page = page;
    });
  }

  @override
  Widget build(BuildContext context) {
    // final userProvider = Provider.of<UserProvider>(context);
    return Scaffold(
      bottomNavigationBar: BottomNavigationBar(
        selectedItemColor: buttonColor,
        unselectedItemColor: primaryColor,
        backgroundColor: backgroundColor,
        onTap: onPageChanged,
        currentIndex: _page,
        unselectedFontSize: 12,
        items: const [
          BottomNavigationBarItem(
            icon: Icon(Icons.favorite),
            label: 'Browse',
```

```
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.add_rounded),
          label: 'Go Live',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.copy),
          label: 'Live Users',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.account_circle_outlined),
          label: 'User',
        ),
      ],
    ),
    body: pages[_page],
  );
 }
}
```

OUTPUT:





Navigation from Browse channels → Live Stream → Current live stream —> Profile Page

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	38
Name	Shubham Raghuvir Nakashe
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

Aim: To connect Flutter UI with Firebase Database

Theory:

Step 1: Create a Firebase Project

1. Go to the [Firebase Console](https://console.firebase.google.com/) and create a new project.
2. Follow the on-screen instructions to set up your project.

Step 2: Add Firebase to Your Flutter Project

1. Add the `firebase_core,firebase_auth,firebase_storage,cloud_storage` package to your `pubspec.yaml` file:

```
 yaml
dependencies:
  flutter:
    sdk: flutter
firebase_auth: ^4.14.1
firebase_core: ^2.23.0
firebase_storage: ^11.5.2
```

2. Run `flutter pub get` to install the package.

Step 3: Initialize Firebase in Your Flutter App

1. Import the `firebase_core` package in your Dart code:

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

2. Initialize Firebase in your app's `main` function:

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

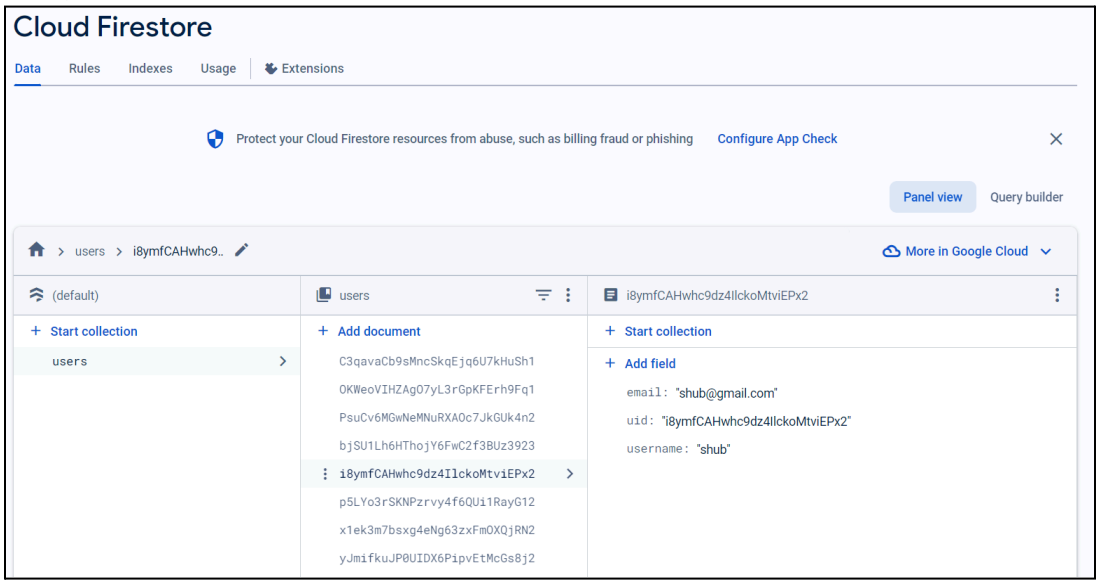
CODE:

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:twitch_clone/firebase_options.dart';
import 'package:twitch_clone/providers/user_provider.dart';
import 'package:twitch_clone/resources/auth_methods.dart';
import 'package:twitch_clone/screens/feed_screen.dart';
import 'package:twitch_clone/screens/home_screen.dart';
import 'package:twitch_clone/screens/login_screen.dart';
import 'package:twitch_clone/screens/onboarding_screen.dart';
import 'package:twitch_clone/screens/profile.dart';
import 'package:twitch_clone/screens/signup_screen.dart';
import 'package:twitch_clone/utils/colors.dart';
import 'package:twitch_clone/models/user.dart' as model;
import 'package:twitch_clone/widgets/loading_indicator.dart';


Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();

    await Firebase.initializeApp(
      options: DefaultFirebaseOptions.currentPlatform,
    );

  runApp(MultiProvider(providers: [
    ChangeNotifierProvider(
      create: (_) => UserProvider(),
    ),
  ], child: const MyApp()));
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Twitch Clone',
      theme: ThemeData.light().copyWith(
          scaffoldBackgroundColor: backgroundColor,
          appBarTheme: AppBarTheme.of(context).copyWith(
              backgroundColor: backgroundColor,
```

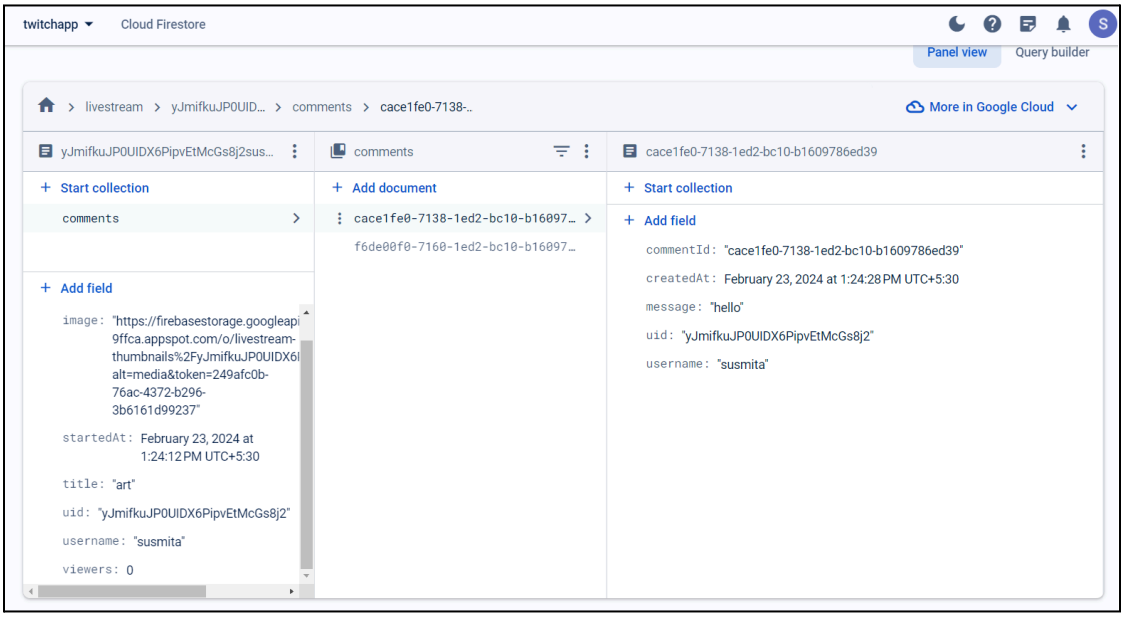
```
          elevation: 0,
          titleTextStyle: const TextStyle(
            color: primaryColor,
            fontSize: 20,
            fontWeight: FontWeight.w600,
          ),
          iconTheme: const IconThemeData(
            color: primaryColor,
          ))),
    routes: {
      OnBoardingScreen.routeName: (context) => const OnBoardingScreen(),
      LoginScreen.routeName: (context) => const LoginScreen(),
      SignupScreen.routeName: (context) => const SignupScreen(),
      HomeScreen.routeName: (context) => const HomeScreen(),
      ProfileScreen.routeName: (context) => ProfileScreen(),
    },
    home: FutureBuilder(
        future: AuthMethods()
            .getCurrentUser(FirebaseAuth.instance.currentUser != null
                ? FirebaseAuth.instance.currentUser!.uid
                : null)
            .then((value) {
          if (value != null) {
            Provider.of<UserProvider>(context, listen: false)
                .setUser(model.User.fromMap(value));
          }
          return value;
        }),
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const LoadingIndicator();
          }

          if (snapshot.hasData) {
            return const HomeScreen();
          }
          return const OnBoardingScreen();
        }),
  );
 }
}
```

OUTPUT:

1) USER DATA IS STORED IN FIREBASE

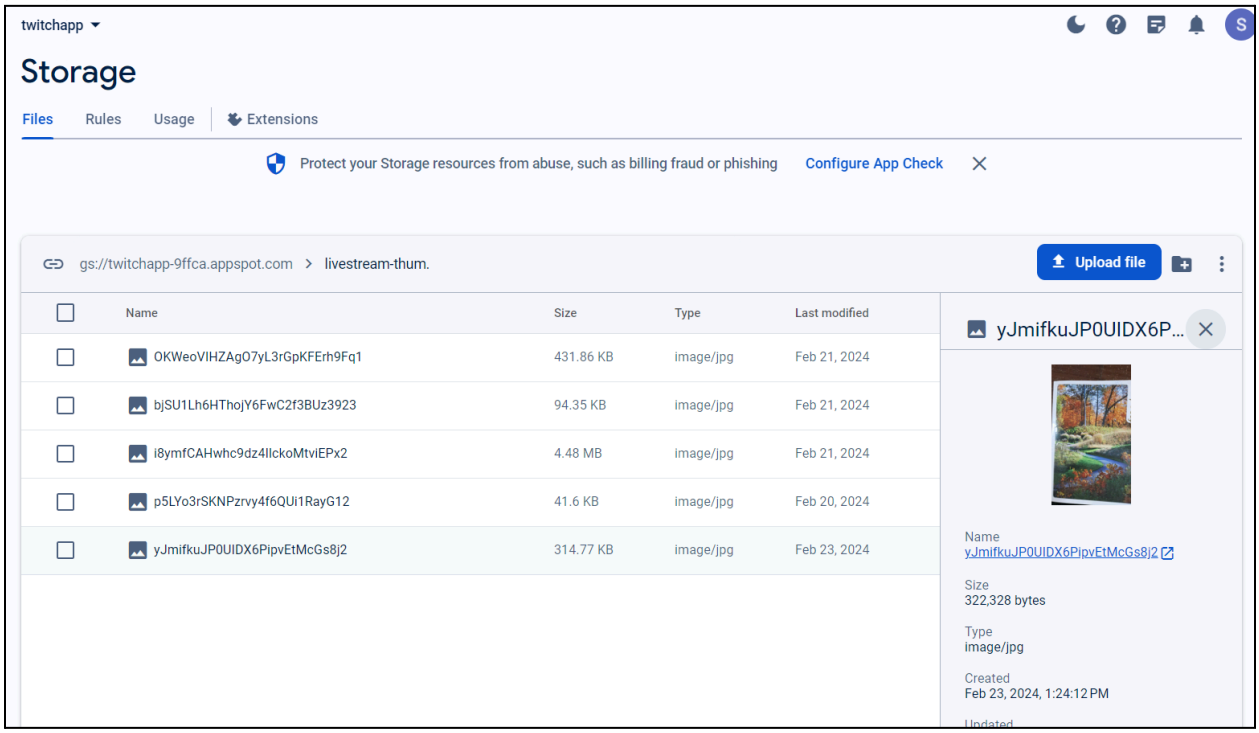


2) LiveStream detail



Comment by users

3) Thumbnail of live stream



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.	38
Name	Shubham Raghuvir Nakashe
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	15

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

Theory:

Regular Web App

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

Progressive Web App

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

Difference between PWAs vs. Regular Web Apps:
A Progressive Web is different and better than a Regular Web app with features like:

1. **Native Experience**

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

2. **Ease of Access**

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

3. **Faster Services**

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

4. **Engaging Approach**

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

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

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

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

The main features are:

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

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

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

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

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

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

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

Linkable — Easily shared via URL without complex installations.

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

CODE AND OUTPUT:

Folder Structure

Icons

Index.html

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

```
<head>
   <!-- Google tag (gtag.js) -->
<script async src="https://www.googletagmanager.com/gtag/js?id=G-CSQNB0SBN9"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'G-CSQNB0SBN9');
</script>
<script>
            window.addEventListener('load', () => {
            registerSW();
            });

            // Register the Service Worker
            async function registerSW() {
            if ('serviceWorker' in navigator) {
                        try {
                        await navigator
                                    .serviceWorker
                                    .register('serviceworker.js');
                        }
                        catch (e) {
                        console.log('SW registration failed');
                        }
            }
            }
</script>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <!-- CSS LINK -->
   <link rel="stylesheet" href="style.css">
   <link rel="manifest"
      href="manifest.json">
   <!-- Fontawesome -->
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.1/css/all.min.css"
integrity="sha512-KfkfwYDsLkIlwQp6LFnl8zNdLGxu9YAA1QvwINks4PhcElQSvqcyVLLD9aMhXd13uQjoXtE
KNosOWaZqXgel0g==" crossorigin="anonymous" referrerpolicy="no-referrer" />
   <title>Twiggy</title>
</head>
<body>
   <!-------------------------------------------- HEADER SECTION -->
   <header class="header" >
      <a href="#" class="logo">
         <img src="./image/pngegg.png" alt="logo">
      </a>
      <nav class="navbar">
         <a href="./index.html" class="active">home</a>
         <a href="./about.html">about</a>
         <a href="./menu.html">menu</a>
         <a href="./products.html">products</a>
         <a href="./review.html">review</a>
```

```
        <a href="./contact.html">contact</a>
        <a href="./blog.html">blog</a>
    </nav>
    <div class="buttons">
        <button id="search-btn">
            <i class="fas fa-search"></i>
        </button>
        <button id="cart-btn">
            <i class="fas fa-shopping-cart"></i>
        </button>
        <button id="menu-btn">
            <i class="fas fa-bars"></i>
        </button>
    </div>
    <div class="search-form">
        <input type="text" class="search-input" id="search-box" placeholder="Search">
        <i class="fas fa-search"></i>
    </div>
    <div class="cart-items-container">
        <div class="cart-item">
            <i class="fas fa-times"></i>
            <img src="./image/pngwing.com (1).png" alt="menu">
            <div class="content">
                <h3>cart item 01</h3>
                <div class="price">Rs 50</div>
            </div>
        </div>
        <div class="cart-item">
            <i class="fas fa-times"></i>
            <img src="./image/pngwing.com (2).png" alt="menu">
            <div class="content">
                <h3>cart item 02</h3>
                <div class="price">Rs 100 </div>
            </div>
        </div>
        <div class="cart-item">
            <i class="fas fa-times"></i>
            <img src="./image/pngwing.com (3).png" alt="menu">
            <div class="content">
                <h3>cart item 03</h3>
                <div class="price">Rs 150 </div>
            </div>
        </div>
        <div class="cart-item">
            <i class="fas fa-times"></i>
            <img src="./image/pngwing.com (4).png" alt="menu">
            <div class="content">
                <h3>cart item 04</h3>
                <div class="price">Rs 200 </div>
            </div>
        </div>
        <a href="#" class="btn">check out </a>
    </div>
</header>
```

```
  <!----------------------------------------------HEADER SECTION -->

  <!----------------------------------------------HOME SECTION -->
  <section class="home" id="home">
    <div class="content">
    <h3>Twiggy</h3>
    <p>Where cravings meet convenience! Explore a world of flavors and order your favorite meals in just a few
taps. Let's make your hunger a thing of the past!"</p>
    <a href="#" class="btn">order now</a>
    </div>
  </section>
  <!----------------------------------------------HOME SECTION -->

  <!----------------------------------------------MENU SECTION -->
  <section class="menu" id="menu">
    <h1 class="heading">our <span>menu</span></h1>
    <div class="box-container">
      <div class="box">

        <div class="box-head">
          <img src="./image/pngwing.com.png" alt="">
          <span class="menu-category">Pizza</span>
          <h3>6 Mini Pizzas</h3>
          <div class="price">Rs 349 <span>Rs 499</span></div>
        </div>
        <div class="box-bottom">
          <a href="#" class="btn">add to cart</a>
        </div>
      </div>
      <div class="box">

        <div class="box-head">
          <img src="./image/pngwing.com (2).png" alt="">
          <span class="menu-category">Burger</span>
          <h3>5 Mini Burgers</h3>
          <div class="price">Rs 249 <span>Rs 375</span></div>
        </div>
        <div class="box-bottom">
          <a href="#" class="btn">add to cart</a>
        </div>
      </div>
      <div class="box">

        <div class="box-head">
          <img src="./image/pngwing.com (1).png" alt="">
          <span class="menu-category">Pizza</span>
          <h3>2 Mixed Pizzas</h3>
          <div class="price">Rs 229 <span>Rs 299</span></div>
        </div>
        <div class="box-bottom">
          <a href="#" class="btn">add to cart</a>
        </div>
      </div>
      <div class="box">
```

```
            <div class="box-head">
               <img src="./image/pngwing.com (3).png" alt="">
               <span class="menu-category">Burger</span>
               <h3>3 Meatball Burgers</h3>
               <div class="price">Rs 299 <span>Rs 449</span></div>
            </div>
            <div class="box-bottom">
               <a href="#" class="btn">add to cart</a>
            </div>
         </div>
      </div>
   </section>
   <!----------------------------------------------MENU SECTION -->


   <!----------------------------------------------PRODUCTS SECTION -->
   <section class="products" id="products">
      <h1 class="heading">our <span>products</span> </h1>
      <div class="box-container">
         <div class="box">
            <div class="box-head">
               <span class="title">mini burger</span>
               <a href="#" class="name">Bacon Burger</a>
            </div>
            <div class="image">
               <img src="./image/pngwing.com (6).png" alt="">
            </div>
            <div class="box-bottom">
               <div class="info">
                  <b class="price">Rs 75</b>
                  <span class="amount">110gr / 300 Cal</span>
               </div>
               <div class="product-btn">
                  <a href="#">
                     <i class="fas fa-plus"></i>
                  </a>
               </div>
            </div>
         </div>
         <div class="box">
            <div class="box-head">
               <span class="title">cheese burger</span>
               <a href="#" class="name">cheese Burger</a>
            </div>
            <div class="image">
               <img src="./image/pngwing.com (4).png" alt="">
            </div>
            <div class="box-bottom">
               <div class="info">
                  <b class="price">Rs 120</b>
                  <span class="amount">140gr / 2500 Cal</span>
               </div>
               <div class="product-btn">
```

```
                <a href="#">
                  <i class="fas fa-plus"></i>
                </a>
              </div>
            </div>
          </div>
          <div class="box">
            <div class="box-head">
              <span class="title">Double burger</span>
              <a href="#" class="name">Double Burger</a>
            </div>
            <div class="image">
              <img src="./image/pngwing.com (5).png" alt="">
            </div>
            <div class="box-bottom">
              <div class="info">
                <b class="price">Rs 240</b>
                <span class="amount">440gr / 600 Cal</span>
              </div>
              <div class="product-btn">
                <a href="#">
                  <i class="fas fa-plus"></i>
                </a>
              </div>
            </div>
          </div>
        </div>
    </section>
    <!----------------------------------------------PRODUCTS SECTION -->

    <!----------------------------------------------ABOUT US SECTION -->
    <section class="about" id="about">
        <h1 class="heading">about <span>us</span> </h1>

        <div class="row">
            <div class="image">
                <img src="./image/pngwing.com (7).png" alt="">
            </div>
            <div class="content">
                <h3>What is the secret receipe of our burgers</h3>
                <div class="paragraph">
                    <p>Our burgers are made with a special blend of freshly ground beef, seasoned to perfection with our
secret spice mix. </p>
                    <p>Each patty is grilled to juicy perfection and served on a toasted brioche bun. </p>
                    <p>Topped with crisp lettuce, ripe tomatoes, melted cheese, and our signature sauce, it's a burger
experience like no other</p>
                </div>
                    <a href="#" class="btn">Learn More</a>
            </div>
        </div>
    </section>
    <!----------------------------------------------ABOUT US SECTION -->

    <!----------------------------------------------REVIEW SECTION -->
```

```
  <section class="review" id="review">
    <h1 class="heading">customer's <span>review</span> </h1>
    <div class="box-container">
      <div class="box">
        <img src="./image/pngwing.com (8).png" alt="quote">
        <p> I'm always impressed by the range of restaurants available on this website. Whether I'm craving
burgers, pizza, or Asian cuisine, I can always find something to satisfy my appetite. The website is reliable, and the
food quality is consistently excellent.</p>
        <img src="./image/pngwing.com (9).png" alt="customer-avatar" class="user">
        <h3>Yash Uskelwar</h3>
        <div class="stars">
          <i class="fas fa-star"></i>
          <i class="fas fa-star"></i>
          <i class="fas fa-star"></i>
          <i class="fas fa-star"></i>
          <i class="fas fa-star-half-alt"></i>
        </div>
      </div>
      <div class="box">
        <img src="./image/pngwing.com (8).png" alt="quote">
        <p>The convenience of ordering food from this website is unmatched. I can order my favorite meals with
just a few clicks and have them delivered to my doorstep in no time. The website is well-designed, making it easy to
navigate, and the delivery is always prompt.</p>
        <img src="./image/pngwing.com (10).png" alt="customer-avatar" class="user">
        <h3>Harshita Dubey</h3>
        <div class="stars">
          <i class="fas fa-star"></i>
          <i class="fas fa-star"></i>
          <i class="fas fa-star"></i>
          <i class="fas fa-star"></i>
          <i class="fas fa-star-half-alt"></i>
        </div>
      </div>
      <div class="box">
        <img src="./image/pngwing.com (8).png" alt="quote">
        <p>I love using this fast food delivery website! It's so easy to order my favorite meals from the comfort of
my home. The food always arrives hot and delicious, and the delivery is always on time.</p>
        <img src="./image/pngwing.com (11).png" alt="customer-avatar" class="user">
        <h3>Sneha Utekar</h3>
        <div class="stars">
          <i class="fas fa-star"></i>
          <i class="fas fa-star"></i>
          <i class="fas fa-star"></i>
          <i class="fas fa-star"></i>
          <i class="fas fa-star-half-alt"></i>
        </div>
      </div>
    </div>
  </section>
  <!----------------------------------------------REVIEW SECTION -->
  <!----------------------------------------------CONTACT SECTION -->
  <section class="contact" id="contact">
    <h1 class="heading">contact <span>us</span> </h1>
    <div class="row">
```

```
        <iframe class="map"
src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d3771.8616688864765!2d72.88367541489297!
3d19.079384587080392!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x3be7b96f91a6c81f%3A0x7c66
614d7f8d15c9!2sKurla%2C%20Mumbai%2C%20Maharashtra!5e0!3m2!1sen!2sin!4v1658505945506!5m2!1sen!2s
in" allowfullscreen="" loading="lazy" referrerpolicy="no-referrer-when-downgrade"></iframe>

        <form>
          <h3>get in touch</h3>
          <div class="inputBox">
            <i class="fas fa-user"></i>
            <input type="text" placeholder="name">
          </div>
          <div class="inputBox">
            <i class="fas fa-envelope"></i>
            <input type="email" placeholder="email">
          </div>
          <div class="inputBox">
            <i class="fas fa-phone"></i>
            <input type="number" placeholder="number">
          </div>
          <input type="submit" class="btn" value="contact now">
        </form>
      </div>
    </section>
    <section class="blog" id="blog">
      <h1 class="heading">our <span>blog</span> </h1>
      <div class="box-container">
        <div class="box-full">
          <div class="image">
            <img src="./image/delicious-burger-on-wooden-board-2022-03-04-05-58-25-utc.jpg" alt="">
          </div>
          <div class="content">
            <a href="#" class="title">how to make burgers</a>
            <span>by admin / 15th Jan, 2024</span>
            <p>A classic cheeseburger starts with a juicy beef patty seasoned with salt and pepper, grilled to
perfection. Toast the burger buns until they're lightly golden and assemble the burger with a layer of crisp lettuce, a
slice of ripe tomato, a few rings of onion, and a slice of your favorite cheese. Add a dollop of ketchup and mustard
for that classic flavor combination. It's simple, delicious, and perfect for any burger lover.

            </p>
            <a href="#" class="btn">read more</a>
          </div>
        </div>
        <div class="box-full">
          <div class="image">
            <img src="./image/delicious-tasty-burgers-on-wooden-background-2021-08-26-15-25-13-utc.jpg"
alt="">
          </div>
          <div class="content">
            <a href="#" class="title">how to make burgers</a>
            <span>by admin / 1st Feb, 2024</span>
            <p>For a BBQ bacon burger, start by grilling or frying your beef patty to your desired level of
doneness. While the patty is cooking, fry up some crispy bacon. Toast your burger buns on the grill or in a pan until
they're lightly golden. Assemble your burger with a generous spoonful of BBQ sauce on the bottom bun, followed
by lettuce, tomato, onion, the beef patty, a slice of cheese, and the crispy bacon. Top it off with the top bun and enjoy
```

```
the smoky, savory flavors of this delicious burger.</p>
                <a href="#" class="btn">read more</a>
            </div>
        </div>
        <div class="box-full">
            <div class="image">
                <img src="./image/handmade-burger-on-dark-background-delicious-blac-2021-10-21-02-27-27-utc.jpg"
alt="">
            </div>
            <div class="content">
                <a href="#" class="title">how to make burgers</a>
                <span>by admin / 21st Feb, 2024</span>
                <p>For a spicy jalapeño burger, start by seasoning your beef patty with chili powder and cayenne
pepper for an extra kick of heat. Grill or fry the patty until it's cooked to your liking. Slice some jalapeños and
remove the seeds for a milder flavor, or leave them in for an extra spicy kick. Toast your burger buns until they're
lightly golden and assemble your burger with a layer of lettuce, the beef patty, sliced jalapeños, a slice of pepper jack
cheese, and a dollop of spicy mayo. It's a fiery and flavorful burger that's sure to please spice lovers.</p>
                <a href="#" class="btn">read more</a>
            </div>
        </div>
    </div>
  </section>
  <section class="footer">
    <div class="search">
        <input type="text" class="search-input" placeholder="Search">
        <button class="btn btn-primary">search</button>
    </div>
    <div class="share">
        <a href="#" class="fab fa-facebook"></a>
        <a href="#" class="fab fa-twitter"></a>
        <a href="#" class="fab fa-instagram"></a>
        <a href="#" class="fab fa-linkedin"></a>
        <a href="#" class="fab fa-pinterest"></a>
    </div>
    <div class="links">
        <a href="#home">home</a>
        <a href="#about">about</a>
        <a href="#menu">menu</a>
        <a href="#products">products</a>
        <a href="#review">review</a>
        <a href="#contact">contact</a>
        <a href="#blog">blog</a>
    </div>
    <div class="credit">
        created by <span>Shubham Nakashe</span> | all rights reserved!
    </div>
  </section>
  <!----------------------------------------------FOOTER SECTION -->

<script src="./script.js"></script>
</body>
</html>
```

serviceworker.js

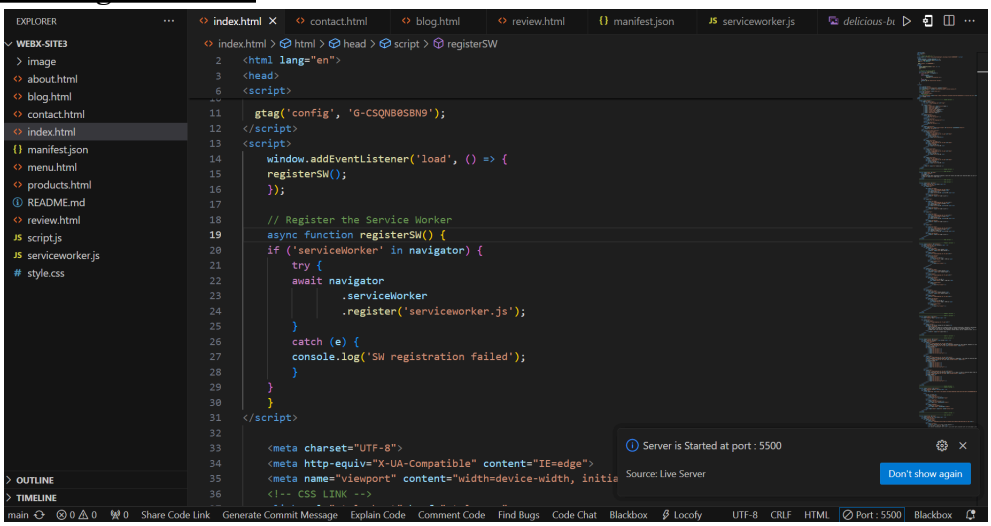
```
var staticCacheName = "Twiggy";

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

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

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

manifest.json

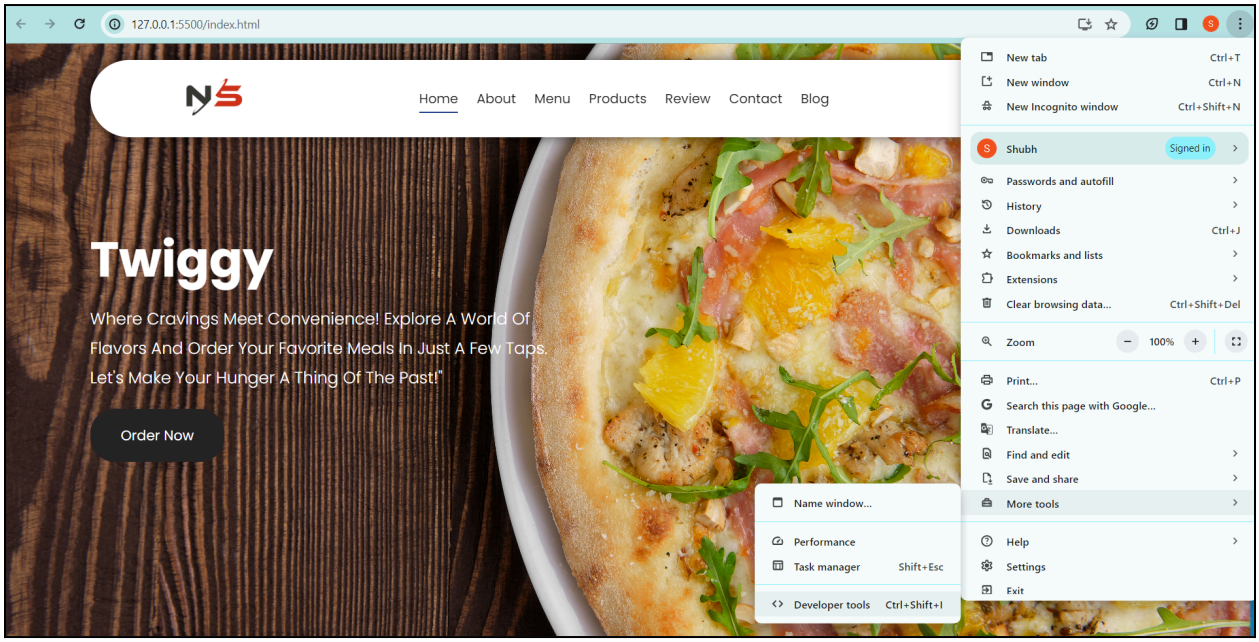
```
{
        "name":"Twiggy",
        "start_url":"index.html",
        "display":"standalone",
        "background_color":"#5900b3",
        "theme_color":"black",
        "scope": ".",
        "description":"This is a Fast food delivery app.",
        "icons":[
        {
        "src":"image/cropped.jpg",
        "sizes":"192x192",
        "type":"image/png"
        },
        {
        "src":"image/CompressJPEG.online_512x512_image.jpg",
        "sizes":"512x512",
        "type":"image/png"
        }
]}
```

Project Title: MAD-PWA **Roll No. 38**

Starting the server

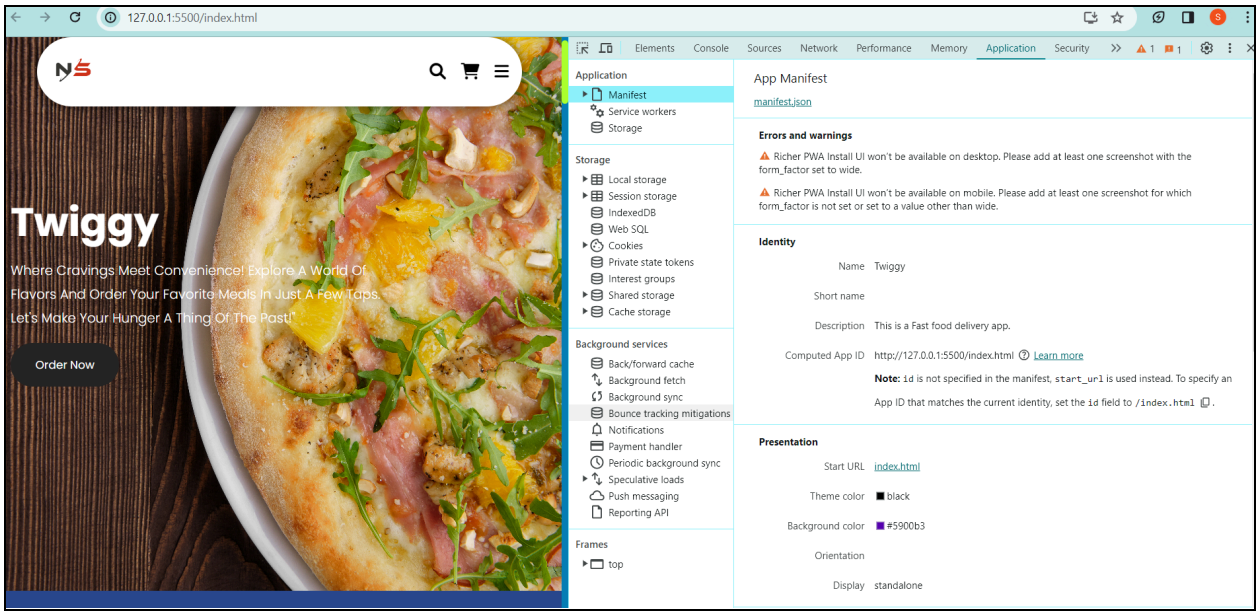


Going to the developers option

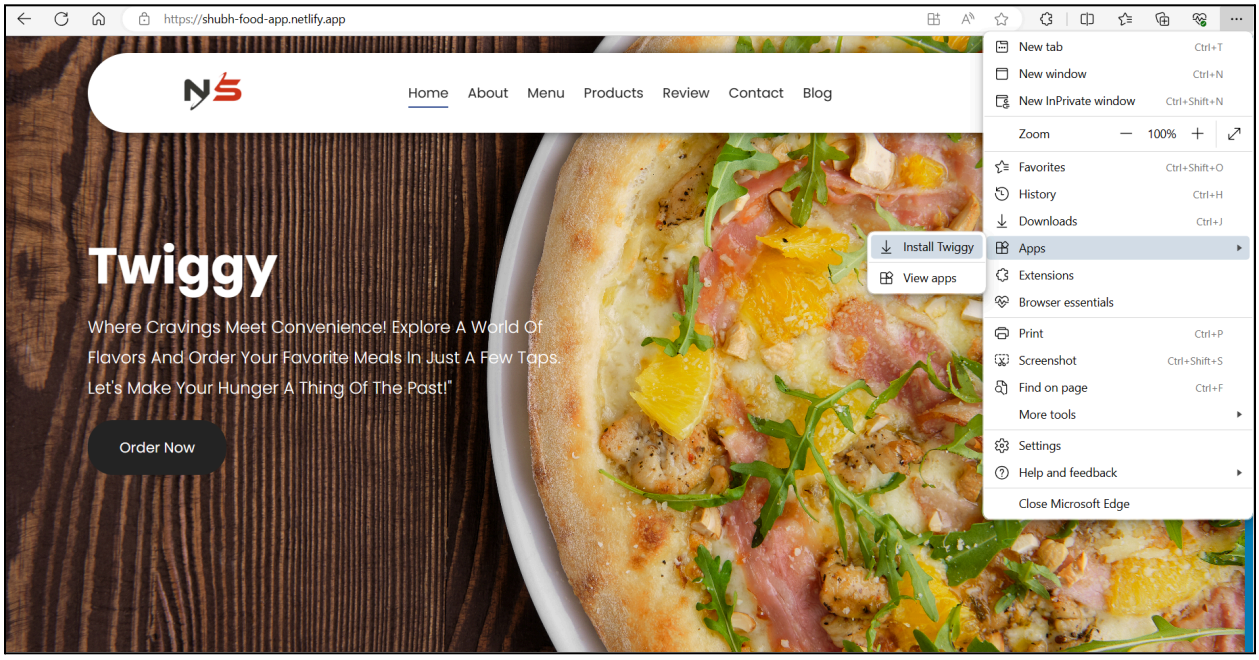


Application

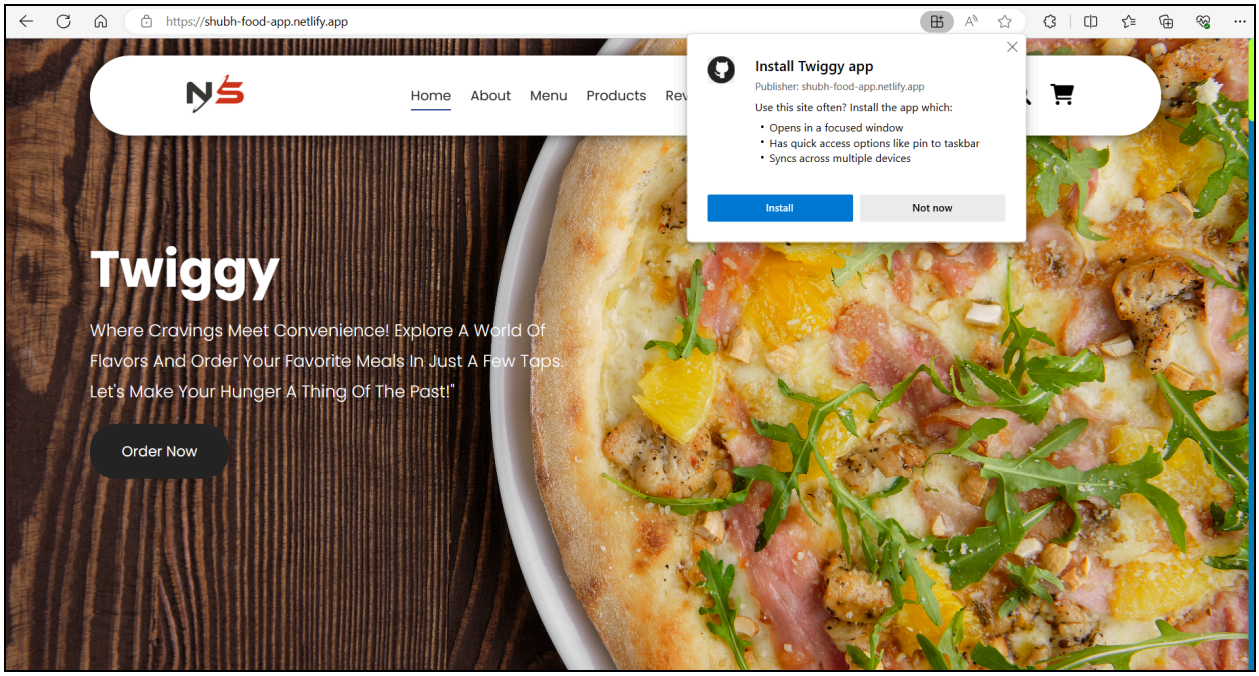
Project Title: MAD-PWA **Roll No. 38**



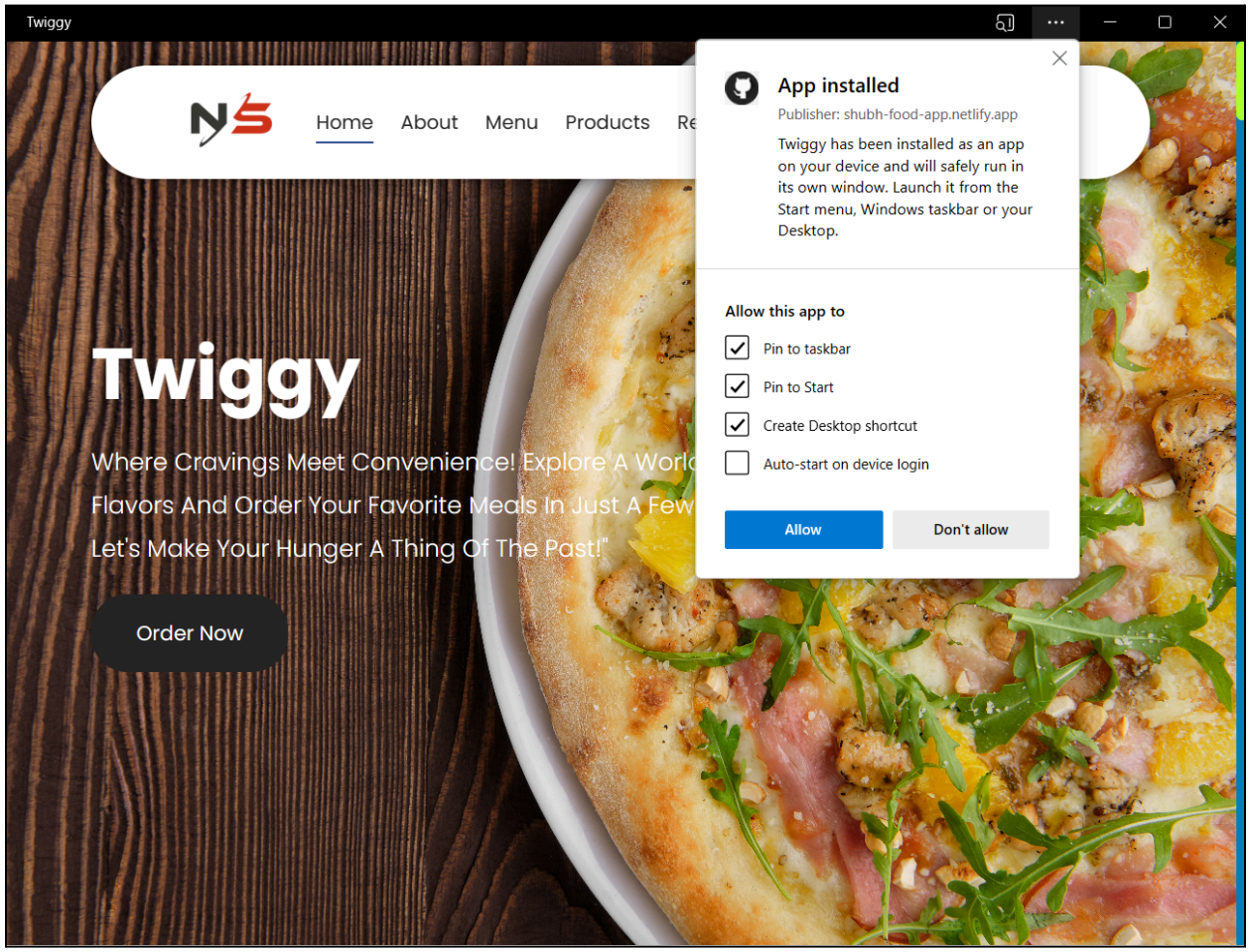
Click on 3 dots —> App —> install App



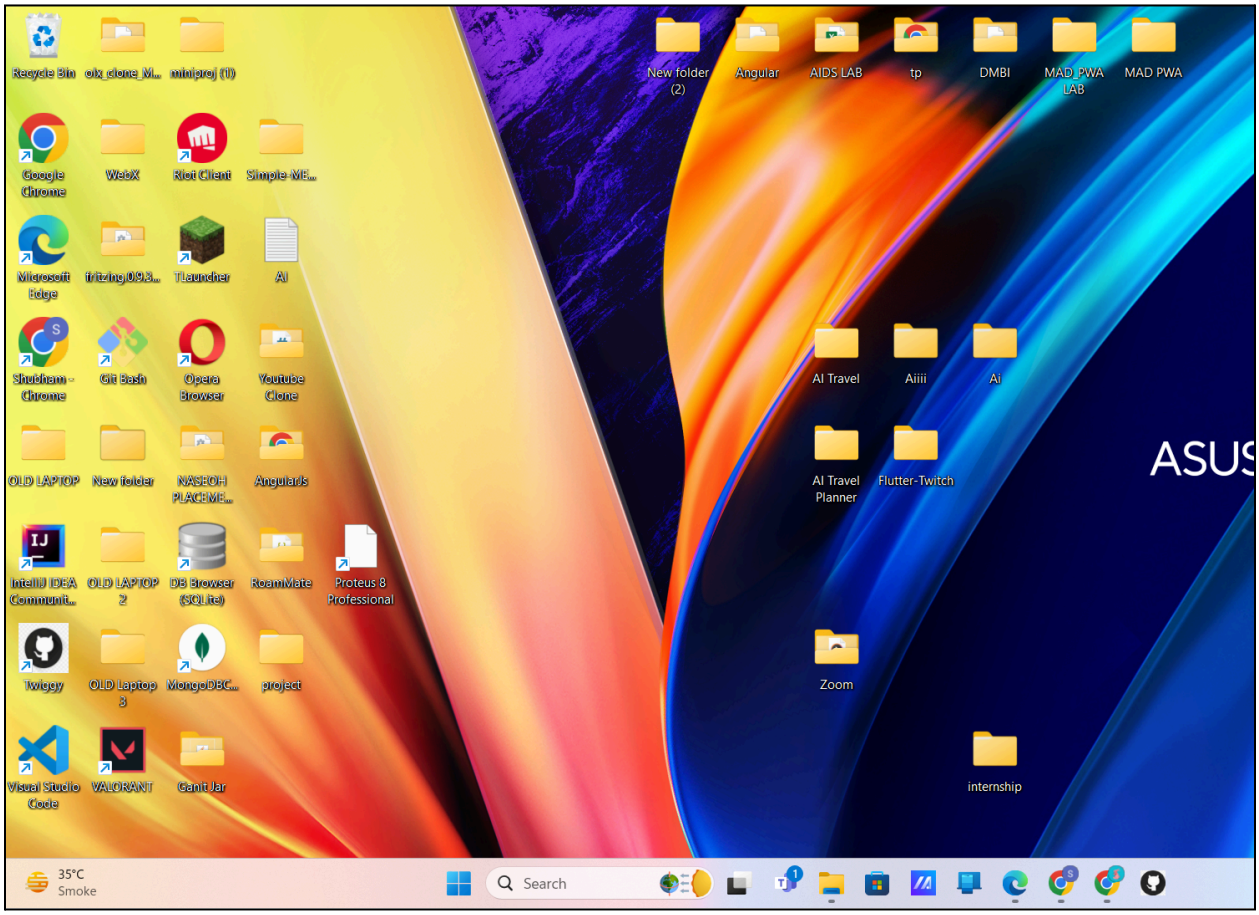
Click on install



Allow for creating desktop shortcut



This is the Twiggy app that is being created



Conclusion:

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

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.	38
Name	Shubham Raghuvir Nakashe
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 code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA.

Theory:

Service Worker

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

Things to note about Service Worker:

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

What can we do with Service Workers?

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

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

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

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

What can’t we do with Service Workers?

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

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

Service Worker Cycle

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

- Registration
- Installation
- Activation

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

main.js

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

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

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

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

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

});

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

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

main.js

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

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

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

service-worker.js

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

Activation
Once a service worker has successfully installed, it transitions into the activation stage. If there are any open pages controlled by the previous service worker, the new service worker enters a waiting state. The new service worker only activates when there are no longer any pages loaded that are still using the old service worker. This ensures that only one version of the service worker is running at any given time.

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

service-worker.js

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

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

CODE AND OUTPUT:

1) Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <!-- Google tag (gtag.js) -->
<script async src="https://www.googletagmanager.com/gtag/js?id=G-CSQNB0SBN9"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'G-CSQNB0SBN9');
</script>

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

   <meta 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="theme-color" content="#4285f4">
   <!-- CSS LINK -->
   <link rel="stylesheet" href="style.css">

   <!-- Fontawesome -->
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.1/css/all.min.css"
integrity="sha512-KfkfwYDsLkIlwQp6LFnl8zNdLGxu9YAA1QvwINks4PhcElQSvqcyVLLD9aMhXd13uQjoXtEKNosOWaZqXgel
0g==" crossorigin="anonymous" referrerpolicy="no-referrer" />
   <title>Twiggy</title>
</head>
<body>
   <!---------------------------------------------- HEADER SECTION -->
   <header class="header" >
      <a href="#" class="logo">
         <img src="./image/pngegg.png" alt="logo">
      </a>
      <nav class="navbar">
```

```
        <a href="./index.html" class="active">home</a>
        <a href="./about.html">about</a>
        <a href="./menu.html">menu</a>
        <a href="./products.html">products</a>
        <a href="./review.html">review</a>
        <a href="./contact.html">contact</a>
        <a href="./blog.html">blog</a>
    </nav>
    <div class="buttons">
        <button id="search-btn">
            <i class="fas fa-search"></i>
        </button>
        <button id="cart-btn">
            <i class="fas fa-shopping-cart"></i>
        </button>
        <button id="menu-btn">
            <i class="fas fa-bars"></i>
        </button>
    </div>
    <div class="search-form">
        <input type="text" class="search-input" id="search-box" placeholder="Search">
        <i class="fas fa-search"></i>
    </div>
    <div class="cart-items-container">
        <div class="cart-item">
            <i class="fas fa-times"></i>
            <img src="./image/pngwing.com (1).png" alt="menu">
            <div class="content">
                <h3>cart item 01</h3>
                <div class="price">Rs 50</div>
            </div>
        </div>
        <div class="cart-item">
            <i class="fas fa-times"></i>
            <img src="./image/pngwing.com (2).png" alt="menu">
            <div class="content">
                <h3>cart item 02</h3>
                <div class="price">Rs 100 </div>
            </div>
        </div>
        <div class="cart-item">
            <i class="fas fa-times"></i>
            <img src="./image/pngwing.com (3).png" alt="menu">
            <div class="content">
                <h3>cart item 03</h3>
                <div class="price">Rs 150 </div>
            </div>
        </div>
        <div class="cart-item">
            <i class="fas fa-times"></i>
            <img src="./image/pngwing.com (4).png" alt="menu">
            <div class="content">
                <h3>cart item 04</h3>
                <div class="price">Rs 200 </div>
            </div>
        </div>
        <a href="#" class="btn">check out </a>
    </div>
</header>
<!----------------------------------------------HEADER SECTION -->
```

```
<!----------------------------------------------HOME SECTION -->
<section class="home" id="home">
   <div class="content">
   <h3>Twiggy</h3>
   <p>Where cravings meet convenience! Explore a world of flavors and order your favorite meals in just a few taps. Let's
make your hunger a thing of the past!"</p>
   <a href="#" class="btn">order now</a>
   </div>
</section>
<!----------------------------------------------HOME SECTION -->

<!----------------------------------------------MENU SECTION -->
<section class="menu" id="menu">
   <h1 class="heading">our <span>menu</span></h1>
   <div class="box-container">
      <div class="box">

         <div class="box-head">
            <img src="./image/pngwing.com.png" alt="">
            <span class="menu-category">Pizza</span>
            <h3>6 Mini Pizzas</h3>
            <div class="price">Rs 349 <span>Rs 499</span></div>
         </div>
         <div class="box-bottom">
            <a href="#" class="btn">add to cart</a>
         </div>
      </div>
      <div class="box">

         <div class="box-head">
            <img src="./image/pngwing.com (2).png" alt="">
            <span class="menu-category">Burger</span>
            <h3>5 Mini Burgers</h3>
            <div class="price">Rs 249 <span>Rs 375</span></div>
         </div>
         <div class="box-bottom">
            <a href="#" class="btn">add to cart</a>
         </div>
      </div>
      <div class="box">

         <div class="box-head">
            <img src="./image/pngwing.com (1).png" alt="">
            <span class="menu-category">Pizza</span>
            <h3>2 Mixed Pizzas</h3>
            <div class="price">Rs 229 <span>Rs 299</span></div>
         </div>
         <div class="box-bottom">
            <a href="#" class="btn">add to cart</a>
         </div>
      </div>
      <div class="box">

         <div class="box-head">
            <img src="./image/pngwing.com (3).png" alt="">
            <span class="menu-category">Burger</span>
            <h3>3 Meatball Burgers</h3>
            <div class="price">Rs 299 <span>Rs 449</span></div>
         </div>
```

```
            <div class="box-bottom">
                <a href="#" class="btn">add to cart</a>
            </div>
        </div>
    </div>
</section>
<!-----------------------------------------------MENU SECTION -->


<!-----------------------------------------------PRODUCTS SECTION -->
<section class="products" id="products">
    <h1 class="heading">our <span>products</span> </h1>
    <div class="box-container">
        <div class="box">
            <div class="box-head">
                <span class="title">mini burger</span>
                <a href="#" class="name">Bacon Burger</a>
            </div>
            <div class="image">
                <img src="./image/pngwing.com (6).png" alt="">
            </div>
            <div class="box-bottom">
                <div class="info">
                    <b class="price">Rs 75</b>
                    <span class="amount">110gr / 300 Cal</span>
                </div>
                <div class="product-btn">
                    <a href="#">
                        <i class="fas fa-plus"></i>
                    </a>
                </div>
            </div>
        </div>
        <div class="box">
            <div class="box-head">
                <span class="title">cheese burger</span>
                <a href="#" class="name">cheese Burger</a>
            </div>
            <div class="image">
                <img src="./image/pngwing.com (4).png" alt="">
            </div>
            <div class="box-bottom">
                <div class="info">
                    <b class="price">Rs 120</b>
                    <span class="amount">140gr / 2500 Cal</span>
                </div>
                <div class="product-btn">
                    <a href="#">
                        <i class="fas fa-plus"></i>
                    </a>
                </div>
            </div>
        </div>
        <div class="box">
            <div class="box-head">
                <span class="title">Double burger</span>
                <a href="#" class="name">Double Burger</a>
            </div>
            <div class="image">
                <img src="./image/pngwing.com (5).png" alt="">
```

```
            </div>
            <div class="box-bottom">
               <div class="info">
                  <b class="price">Rs 240</b>
                  <span class="amount">440gr / 600 Cal</span>
               </div>
               <div class="product-btn">
                  <a href="#">
                     <i class="fas fa-plus"></i>
                  </a>
               </div>
            </div>
         </div>
      </section>
      <!----------------------------------------------PRODUCTS SECTION -->

      <!----------------------------------------------ABOUT US SECTION -->
      <section class="about" id="about">
         <h1 class="heading">about <span>us</span> </h1>

         <div class="row">
            <div class="image">
               <img src="./image/pngwing.com (7).png" alt="">
            </div>
            <div class="content">
               <h3>What is the secret receipe of our burgers</h3>
               <div class="paragraph">
                  <p>Our burgers are made with a special blend of freshly ground beef, seasoned to perfection with our secret spice
mix. </p>
                  <p>Each patty is grilled to juicy perfection and served on a toasted brioche bun. </p>
                  <p>Topped with crisp lettuce, ripe tomatoes, melted cheese, and our signature sauce, it's a burger experience like no
other</p>
               </div>
                  <a href="#" class="btn">Learn More</a>
            </div>
         </div>
      </section>
      <!----------------------------------------------ABOUT US SECTION -->

      <!----------------------------------------------REVIEW SECTION -->
      <section class="review" id="review">
         <h1 class="heading">customer's <span>review</span> </h1>
         <div class="box-container">
            <div class="box">
               <img src="./image/pngwing.com (8).png" alt="quote">
               <p> I'm always impressed by the range of restaurants available on this website. Whether I'm craving burgers, pizza, or
Asian cuisine, I can always find something to satisfy my appetite. The website is reliable, and the food quality is consistently
excellent.</p>
               <img src="./image/pngwing.com (9).png" alt="customer-avatar" class="user">
               <h3>Yash Uskelwar</h3>
               <div class="stars">
                  <i class="fas fa-star"></i>
                  <i class="fas fa-star"></i>
                  <i class="fas fa-star"></i>
                  <i class="fas fa-star"></i>
                  <i class="fas fa-star-half-alt"></i>
               </div>
            </div>
            <div class="box">
```

```
            <img src="./image/pngwing.com (8).png" alt="quote">
            <p>The convenience of ordering food from this website is unmatched. I can order my favorite meals with just a few
clicks and have them delivered to my doorstep in no time. The website is well-designed, making it easy to navigate, and the
delivery is always prompt.</p>
            <img src="./image/pngwing.com (10).png" alt="customer-avatar" class="user">
            <h3>Harshita Dubey</h3>
            <div class="stars">
               <i class="fas fa-star"></i>
               <i class="fas fa-star"></i>
               <i class="fas fa-star"></i>
               <i class="fas fa-star"></i>
               <i class="fas fa-star-half-alt"></i>
            </div>
         </div>
         <div class="box">
            <img src="./image/pngwing.com (8).png" alt="quote">
            <p>I love using this fast food delivery website! It's so easy to order my favorite meals from the comfort of my home.
The food always arrives hot and delicious, and the delivery is always on time.</p>
            <img src="./image/pngwing.com (11).png" alt="customer-avatar" class="user">
            <h3>Sneha Utekar</h3>
            <div class="stars">
               <i class="fas fa-star"></i>
               <i class="fas fa-star"></i>
               <i class="fas fa-star"></i>
               <i class="fas fa-star"></i>
               <i class="fas fa-star-half-alt"></i>
            </div>
         </div>
      </div>
   </section>
   <section class="contact" id="contact">
      <h1 class="heading">contact <span>us</span> </h1>
      <div class="row">
         <iframe class="map"
src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d3771.8616688864765!2d72.88367541489297!3d19.079384
587080392!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x3be7b96f91a6c81f%3A0x7c66614d7f8d15c9!2sKurla%2C
%20Mumbai%2C%20Maharashtra!5e0!3m2!1sen!2sin!4v1658505945506!5m2!1sen!2sin" allowfullscreen="" loading="lazy"
referrerpolicy="no-referrer-when-downgrade"></iframe>

         <form>
            <h3>get in touch</h3>
            <div class="inputBox">
               <i class="fas fa-user"></i>
               <input type="text" placeholder="name">
            </div>
            <div class="inputBox">
               <i class="fas fa-envelope"></i>
               <input type="email" placeholder="email">
            </div>
            <div class="inputBox">
               <i class="fas fa-phone"></i>
               <input type="number" placeholder="number">
            </div>
            <input type="submit" class="btn" value="contact now">
         </form>
      </div>
   </section>
   <section class="blog" id="blog">
      <h1 class="heading">our <span>blog</span> </h1>
      <div class="box-container">
```

```
        <div class="box-full">
          <div class="image">
            <img src="./image/delicious-burger-on-wooden-board-2022-03-04-05-58-25-utc.jpg" alt="">
          </div>
          <div class="content">
            <a href="#" class="title">how to make burgers</a>
            <span>by admin / 15th Jan, 2024</span>
            <p>A classic cheeseburger starts with a juicy beef patty seasoned with salt and pepper, grilled to perfection. Toast
the burger buns until they're lightly golden and assemble the burger with a layer of crisp lettuce, a slice of ripe tomato, a few
rings of onion, and a slice of your favorite cheese. Add a dollop of ketchup and mustard for that classic flavor combination. It's
simple, delicious, and perfect for any burger lover.

            </p>
            <a href="#" class="btn">read more</a>
          </div>
        </div>
        <div class="box-full">
          <div class="image">
            <img src="./image/delicious-tasty-burgers-on-wooden-background-2021-08-26-15-25-13-utc.jpg" alt="">
          </div>
          <div class="content">
            <a href="#" class="title">how to make burgers</a>
            <span>by admin / 1st Feb, 2024</span>
            <p>For a BBQ bacon burger, start by grilling or frying your beef patty to your desired level of doneness. While the
patty is cooking, fry up some crispy bacon. Toast your burger buns on the grill or in a pan until they're lightly golden. Assemble
your burger with a generous spoonful of BBQ sauce on the bottom bun, followed by lettuce, tomato, onion, the beef patty, a slice
of cheese, and the crispy bacon. Top it off with the top bun and enjoy the smoky, savory flavors of this delicious burger.</p>
            <a href="#" class="btn">read more</a>
          </div>
        </div>
        <div class="box-full">
          <div class="image">
            <img src="./image/handmade-burger-on-dark-background-delicious-blac-2021-10-21-02-27-27-utc.jpg" alt="">
          </div>
          <div class="content">
            <a href="#" class="title">how to make burgers</a>
            <span>by admin / 21st Feb, 2024</span>
            <p>For a spicy jalapeño burger, start by seasoning your beef patty with chili powder and cayenne pepper for an
extra kick of heat. Grill or fry the patty until it's cooked to your liking. Slice some jalapeños and remove the seeds for a milder
flavor, or leave them in for an extra spicy kick. Toast your burger buns until they're lightly golden and assemble your burger with
a layer of lettuce, the beef patty, sliced jalapeños, a slice of pepper jack cheese, and a dollop of spicy mayo. It's a fiery and
flavorful burger that's sure to please spice lovers.</p>
            <a href="#" class="btn">read more</a>
          </div>
        </div>
      </div>
    </section>
    <section class="footer">
      <div class="search">
        <input type="text" class="search-input" placeholder="Search">
        <button class="btn btn-primary">search</button>
      </div>
      <div class="share">
        <a href="#" class="fab fa-facebook"></a>
        <a href="#" class="fab fa-twitter"></a>
        <a href="#" class="fab fa-instagram"></a>
        <a href="#" class="fab fa-linkedin"></a>
        <a href="#" class="fab fa-pinterest"></a>
      </div>
      <div class="links">
```

```
            <a href="#home">home</a>
            <a href="#about">about</a>
            <a href="#menu">menu</a>
            <a href="#products">products</a>
            <a href="#review">review</a>
            <a href="#contact">contact</a>
            <a href="#blog">blog</a>
        </div>
        <div class="credit">
            created by <span>Shubham Nakashe</span> | all rights reserved!
        </div>
    </section>
    <!----------------------------------------------FOOTER SECTION -->

<script src="./script.js"></script>
<script src="./app.js"></script>
</body>
</html>
```

2) app.js

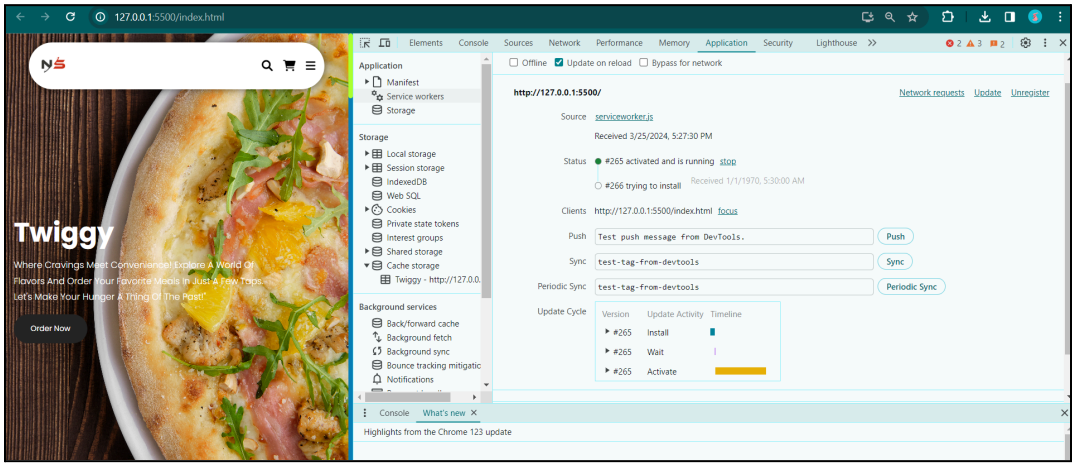
```
if ('serviceWorker' in navigator) {
    window.addEventListener('load', () => {
        navigator.serviceWorker.register('/service-worker.js')
            .then(registration => {
                console.log('Service Worker registered with scope:', registration.scope);
            })
            .catch(error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

3) service-worker.js

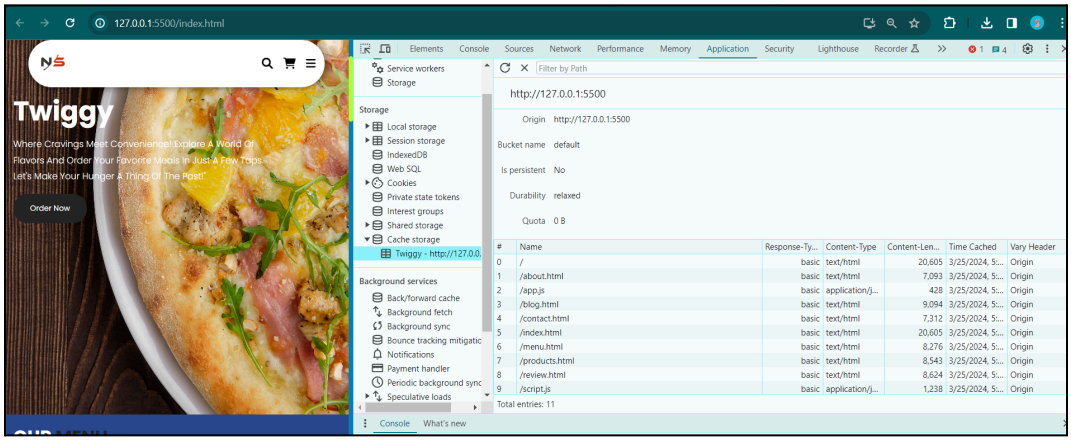
```
var cacheName = "Twiggy";
const assetsToCache = [
    '/',
    '/index.html',
    '/style.css',
    '/app.js',
                '/blog.html',
                '/about.html',
                '/contact.html',
                '/menu.html',
                '/products.html',
                '/review.html',
                '/script.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);
        })
      );
    })
  );
});
```

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



Now scroll down to Cache Storage



Conclusion:

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

MAD & PWA Lab
Journal

Experiment No.	09
Experiment Title.	To implement Service worker events like fetch, sync and push for E-commerce PWA
Roll No.	38
Name	Shubham Raghuvir Nakashe
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.

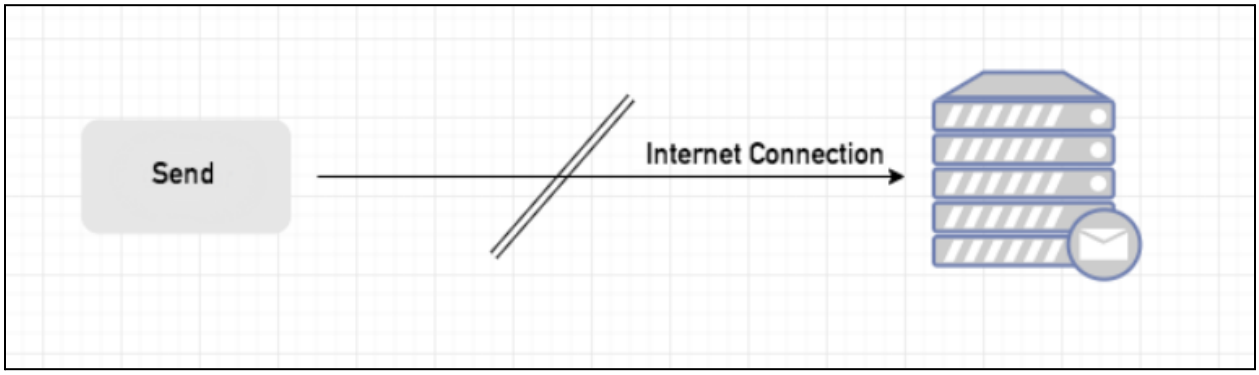
Fetch Event
You can track and manage page network traffic with this event. You can check existing cache, manage “cache first” and “network first” requests and return a response that you want.

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”.

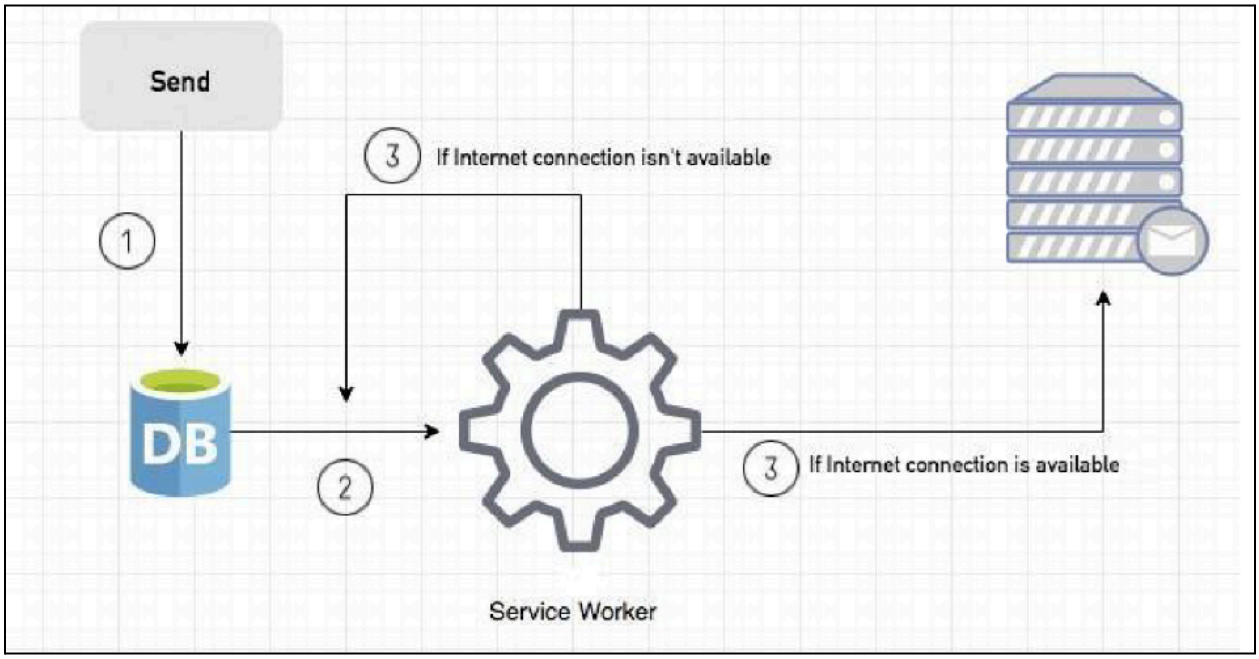
- CacheFirst - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.
- NetworkFirst - In this function, firstly we can try getting an updated response from the network, if this process completed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

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

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



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



1. When we click the "send" button, email content will be saved to IndexedDB.
2. Background Sync registration.
3. If the Internet connection is available, all email content will be read and sent to Mail Server.
   If the Internet connection is unavailable, the service worker waits until the connection is available even though the window is closed. When it is available, email content will be sent to Mail Server.

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

Event Listener for Background Sync Registration

Event Listener for sw.js

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

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

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

CODE AND OUTPUT:

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("Twiggy", {
          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',
      '/style.css',
      '/app.js',
      '/blog.html',
      '/about.html',
      '/contact.html',
      '/menu.html',
      '/products.html',
      '/review.html',
      '/script.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);
      });
  });
};
```

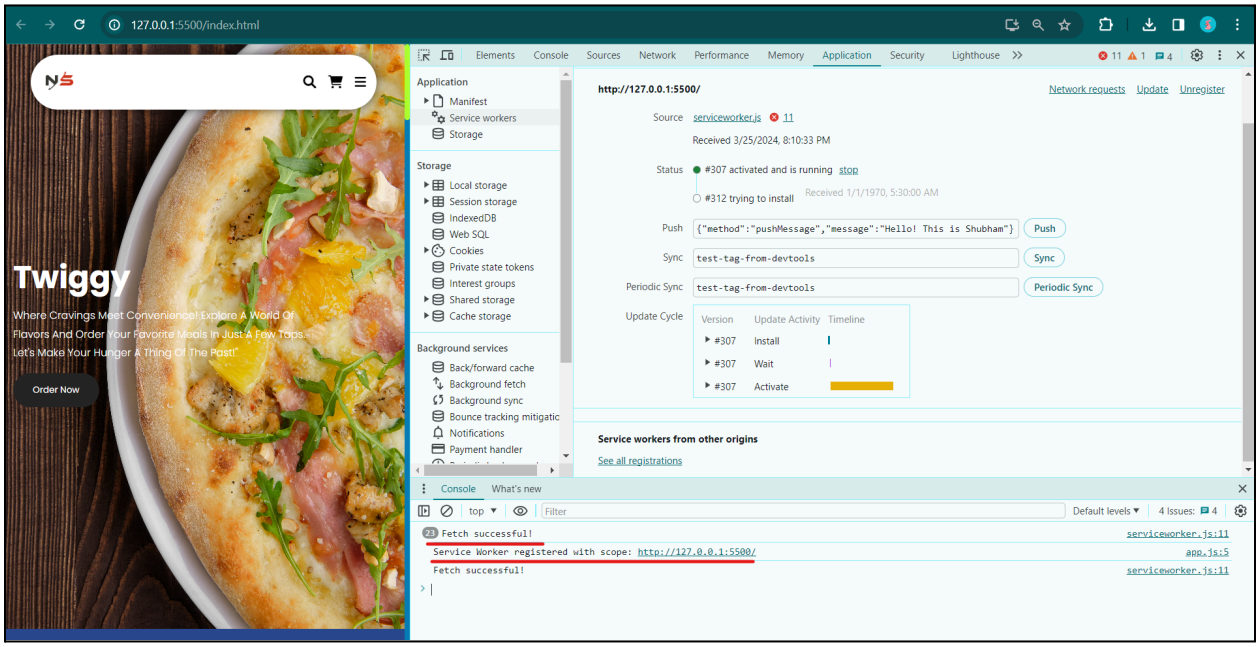
```
var returnFromCache = function (request) {
   return caches.open("offline").then(function (cache) {
      return cache.match(request).then(function (matching) {
         if (!matching || matching.status == 404) {
            return cache.match("offline.html");
         } else {
            return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

app.js

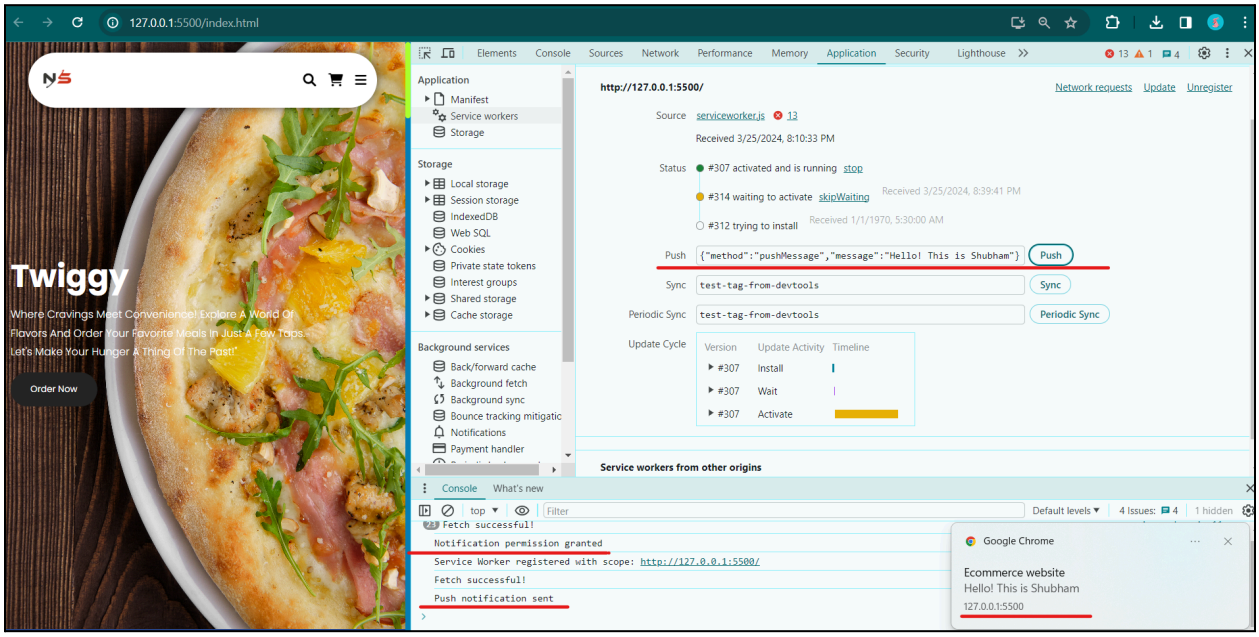
```
if ('serviceWorker' in navigator) {
   window.addEventListener('load', () => {
      navigator.serviceWorker.register('serviceworker.js')
         .then(registration => {
            console.log('Service Worker registered with scope:', registration.scope);
         })
         .catch(error => {
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

<script>
   if ('Notification' in window) {
   Notification.requestPermission().then(function (result) {
      if (result === 'granted') {
         console.log('Notification permission granted');
      } else {
         console.warn('Notification permission denied');
      }
   });
}
```

</script>

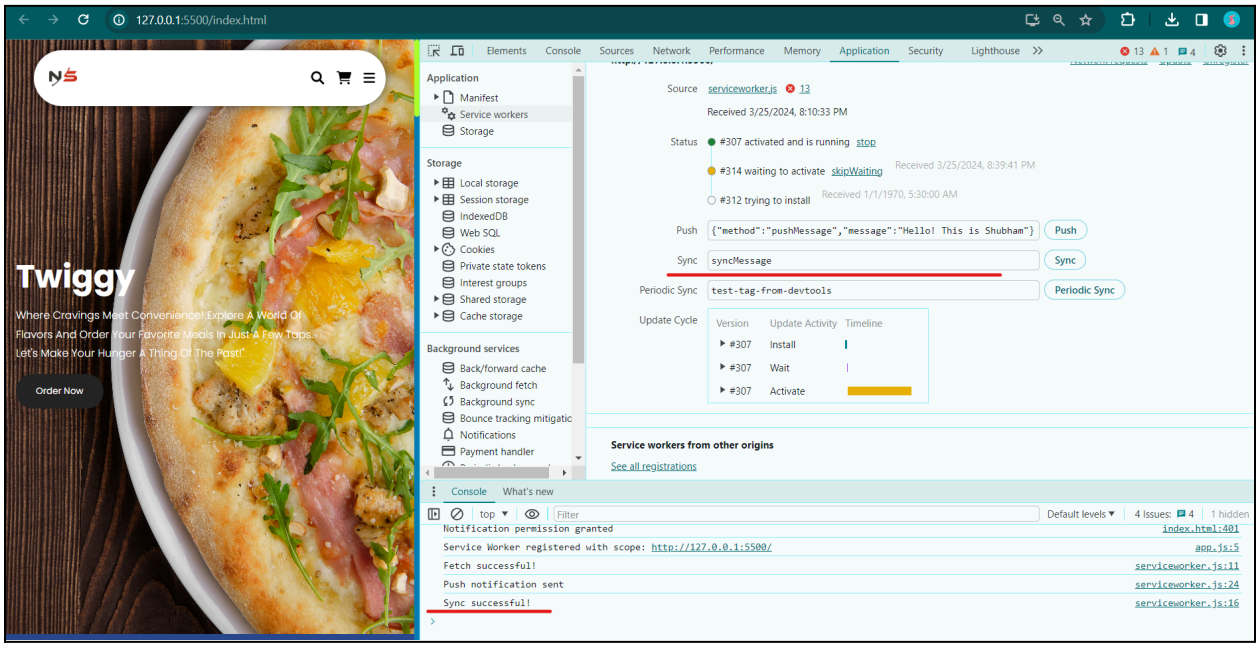
1) Fetch Event



2) Push Event



3) Sync Event



Conclusion:

In this experiment, we have successfully implemented service worker events like fetch, sync and push for my Twiggy 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.	38
Name	Shubham Raghuvir Nakashe
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:
- Blogging with Jekyll
- Custom URL
- Automatic Page Generator

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

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

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

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

Firebase

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

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

- Realtime backend made easy
- Fast and responsive

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

<u>Pros</u>

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

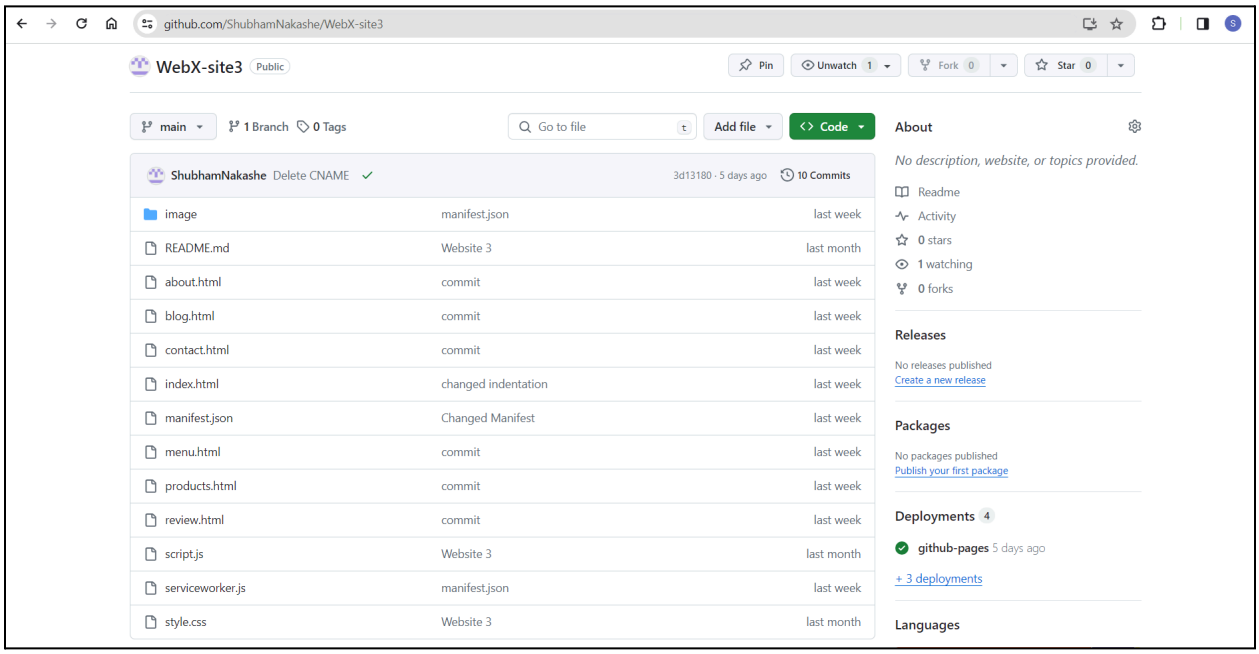
<u>Cons</u>

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

CODE AND OUTPUT:

GITHUB REPOSITORY LINK : GitHub Repo

GITHUB SCREENSHOT :



HOSTED WEBSITE LINK : Hosted Website

Conclusion:

Successfully hosted website on 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.	38
Name	Shubham Raghuvir Nakashe
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:

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

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

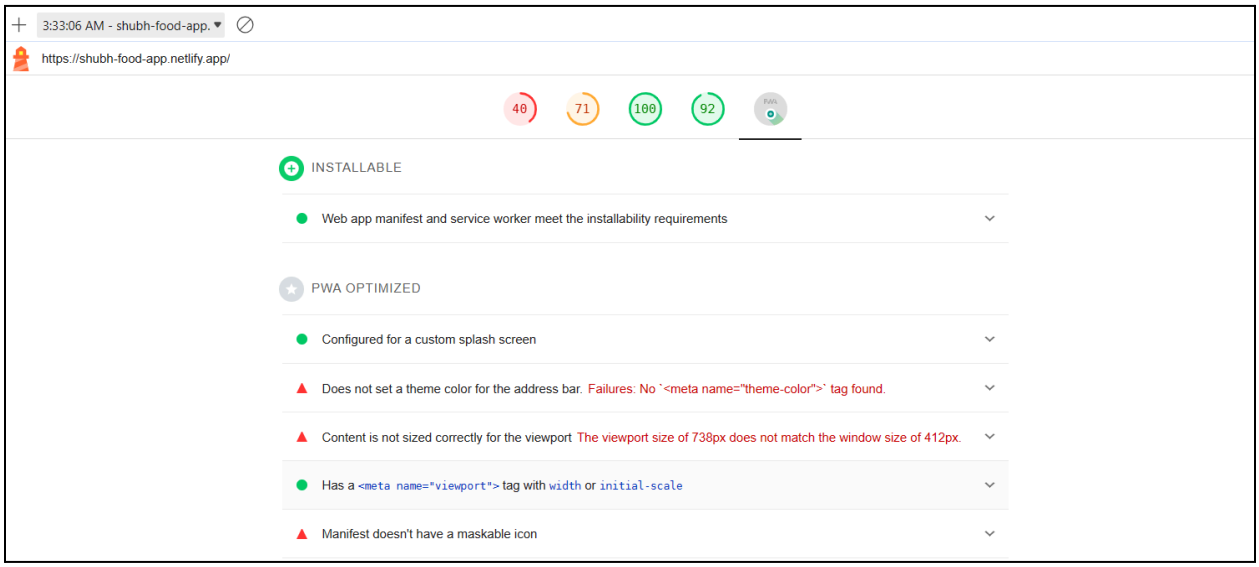
Accessibility: As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics though,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

Best Practices: As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS
Avoiding the use of deprecated code elements like tags, directives, libraries, etc. Password input with paste-into disabled Geo-Location and cookie usage alerts on load, etc.

CODE AND OUTPUT:

manifest.json

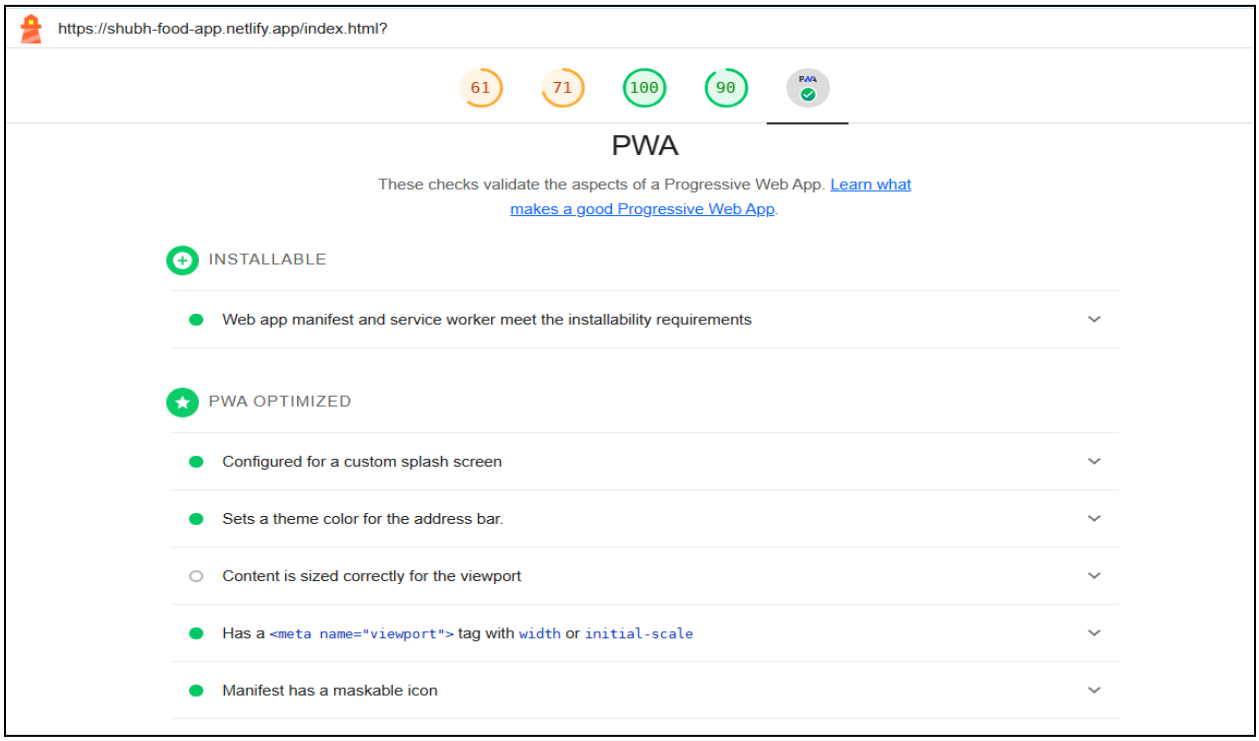
```
{
        "name":"Twiggy",
        "start_url":"index.html",
        "display":"standalone",
        "background_color":"#5900b3",
        "theme_color":"black",
        "scope": ".",
        "description":"This is a Fast food delivery app.",
        "icons":[
        {
        "src":"image/cropped.jpg",
        "sizes":"192x192",
        "type":"image/png"
        },
        {
        "src":"image/CompressJPEG.online_512x512_image.jpg",
        "sizes":"512x512",
        "type":"image/png"
        }
]
}
```



After changes made in manifest.json

```
{
        "name":"Twiggy",
        "start_url":"index.html",
        "display":"standalone",
        "background_color":"#5900b3",
```

```
        "theme_color":"black",
        "scope": ".",
        "description":"This is a Fast food delivery app.",
        "icons":[
        {
        "src":"image/cropped.jpg",
        "sizes":"192x192",
        "type":"image/png",
   "purpose":"any maskable"

        },
        {
        "src":"image/CompressJPEG.online_512x512_image.jpg",
        "sizes":"512x512",
        "type":"image/png",
   "purpose":"any maskable"
        }
]
}
```



Conclusion:

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

MAD & PWA Lab
Journal

Experiment No.	Assignment-1
Assignment 1 Questions	1. Flutter Overview:Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community. 2. Widget Tree and Composition: Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree. 3. State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable. 4. Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.
Roll No.	38
Name	Shubham Raghuvir Nakashe
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:	05

Shubham R. Nakashe_D15A_38

Page No. _____

Date : _____

Assignment - 1 flutter

Q1. 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.

→ 1) Key features of flutter:

ⓐ Single codebase for Multiple Platforms -
Flutter allows developers to write code and deploy it on both iOS and Android platforms.

ⓑ Hot Reload -
This enables developers to instantly see the results of the code changes they make.

ⓒ Expressive UI -
Developers have the flexibility to create expressive and flexible UIs.

ⓓ Integration with other Tools. -
Flutter can easily integrate with other popular development tools and frameworks.

2) Advantages of Flutter

ⓐ Faster Development
Uses single codebase for multiple platforms.

ⓑ consistent UI Across Platforms
Widgets provide a consistent look and feel across different platforms.

ⓒ Cost-Efficiency
Developing and maintaing single codebase

Page No. _____

Date : _____

Q2. Widget Tree and Composition: Describe the concept of 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.

→ Widget Tree.

- The widget Tree is a hierarchical structure of widgets that defines the user interface of an application.
- Every visual element, from simple components to complex layouts, is represented by a widget.
- Widget can be categorized in 2 types.

ⓐ Stateless Widget

It is immutable and cannot change over time.
Eg: images, text.

Ⓑ Stateful Widget.

Widget that can change its state over time
Eg: buttons, forms.

Widget Composition

- Widget composition in Flutter involves combining multiple simple widgets to create more complex and compound widgets.
- This composability is a powerful concept that allows developers to build sophisticated user interfaces by nesting widgets within each other.

Page No.

Date :

Commonly Used Widgets.

ⓐ Container
- A box model for padding, margin and decoration.

ⓑ Column and Row
- Layout widgets for arranging children vertically or horizontally

ⓒ Stack
Overlapping widgets, allowing them to be layered on top of each other.

ⓓ List View
A scrollable list of widgets.

ⓔ Grid View
A scrollable grid of widgets.

ⓕ AppBar
A material design app bar typically at top of screen.

ⓖ Textfield
An input field for users to enter text.

ⓗ Button Widgets
Interactive buttons for user actions.

Page No.
Date :

for both iOS and Android reduces development cost and resources.

3) Differ from Traditional Approach.
   a) Traditional approach uses a hierarchical structure for UI components, whereas flutter uses a widget-based approach.
   b) Flutter compiles to native ARM code, providing performance comparable to native applications.
   c) Hot Reloads allows to see changes made instantly.

Flutter's popularity is driven by increased productivity, a growing community, flexibility in UI design, cross-platform development capabilities, and adoption by major companies.

Page No. _____

Date : _____

⑤ Testing can be more challenging	⑤ Good testability support	⑤ Enhanced testing experience
⑥ Widely used, well established	⑥ Widely adopted well supported	⑥ Gaining popularity growing community

Scenario where each is applicable

① setState.
- for small to moderately complex applications.
- when managing local state within a widget.

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

② Provider
- For medium to large-sized applications.
- when a centralized state is needed, accessible by multiple widgets.

Eg:- Managing user authentication, theme changes or app-wide configuration.

③ Riverpod.
- for large and complex applications.
- when testability and maintainability are top priorities.

Eg:- Complex applications with multiple feature dynamic UIs

Page No. _____

Date : _____

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 dependencies in pubspec.yaml.

```
dependencies:
    firebase_core : ^version
    firebase_auth : ^version
    cloud_firestore : ^version
```

3) Run flutter pub get
4) Initialise firebase by calling 'Firebase.initialiseApp()' in main

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

Void main () async {
Widgets FlutterBinding.ensureInitialized();
await Firebase.initializeApp()
runApp (MyApp());
}
```

Page No. _____

Date : _____

Q3. 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 scenario where each approach is suitable.

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

setState	Provider	Riverpod.
① Built-in Flutter method	① External package named ('provider')	① External package ('riverpod')
② Local state within a widget	② Global state within widget tree	② Global state with additional features.
③ Limited scalability for large apps	③ Suitable for medium sized apps	③ Designed for large and complex apps.
④ May lead to code redundancy	④ Balances simplicity and readability	④ Emphasizes readability and clean syntax.

Page No. _____

Date : _____

Benefits of Using Firebase as Backend :

① Real-time Database
- Firebase offers a real-time NoSQL database.

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

③ Cloud firestore :
Firebase's Cloud firestore provides a secure and easy-to-implement scalable NoSQL database that allows you to store and sync data in real time.

④ Hosting
Firebase Hosting provides a simple and efficient way to deploy and host web applications.

Data Synchronization

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

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.

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.	38
Name	Shubham Raghuvir Nakashe
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:	04

Shubham R. Nakashe
D15A - 38

Assignment 2

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

→

- A Progressive Web App is a type of web application that uses modern web technologies to deliver an app-like experience to users, combining the best features of web and mobile applications.
- PWAs are designed to be fast, reliable and engaging and they can work seamlessly across different devices and platforms.

(8)

Significance in modern web development. (4)

① Cross-Platform Compatability:
PWAs are built using web technologies like HTML, CSS and JavaScript, making them work across different devices and platforms.

② Responsive design:
It is built with responsive design principle

③ Offline functionality.
It can work offline or with poor internet connection.

④ Applike Experience
It offers features such as push notification homescreen installation full screen mode making them feel like native app.

⑤ fast Performance.

Teacher's Sign.: _____

Key Characteristics:

① Resource Consumption:
PWAs generally consume less device storage compared to traditional apps.

② Distribution:
PWA can be shared via a URL making them easier to distribute and discover.

③ Offline Capability
PWA can work offline or with limited connectivity using service workers.

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

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

Teacher's Sign.:

Responsive	Fluid	Adaptive
① Flexible, adapt to any screen size	① continually adjust based on relative units	① user multiple fixed width layouts.
② Moderate effort for development	② less effort for basic layout	② More effort to create multiple layouts.
③ Good control over overall layout	③ less control over element appearance on extreme sizes.	③ High control over layout for each category.
④ Ideal foundation due to its versatility	④ Can be used may require adjustment for optimal Experience	④ Less common for PWA's due to development overhead.

Q3 Describe the lifecycle of Service Workers, including registration, installation and activation phases.

→

For Service worker to start performing its background tasks, we have to make sure it is up and running in the background.

Teacher's Sign.: _____

Date
Page

① Registration Phase

- The Service Worker script is linked in the HTML code and registered using "navigator.serviceWorker.register()".
- Developers define the scope of the Service Worker during registration.
- Scope means the files to which the service worker has access to.

② Installation Phase.

- The browser downloads the Service Worker script and triggers an 'install' event.
- Developers can cache essential resources during this phase using the Cache API.
- Eg:- Caching assets during the install event.

③ Activation Phase.

- Upon installation completion, an 'activate' event is triggered.
- Developers typically clean up old caches during activation and prepare the Service Worker to take control.
- The Service Worker now controls its specified scope, enabling features like offline access and background async.

Teacher's Sign.: _____

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

→

- IndexedDB is a powerful client-side storage API that allows web applications, including those utilizing Service Workers, to store large amounts of structured data persistently in the user's browser.

Here is how IndexedDB is used in the context of a Service Worker for data storage.

① Offline Data Storage.
- Service Workers can intercept network requests and cache responses, enabling PWAs to work offline.
- When the Service Worker fetches data from the network, it can store the fetched data in IndexedDB for future offline use.

② Asynchronous API
- Provides an asynchronous API for data storage and retrieval.
- Do not block the main thread, preventing UI freezes.

③ Structured Data Storage.
- Stores data in structured databases including developers to organize data into object stores and indexes.

- Service Workers can use IndexedDB to store various types of data, including JSON objects. files.

④ Data Synchronization.

- It can be used in conjunction with Service Worker sync events or background Sync APIs to synchronize data with server when the user's device reconnects to the internet.

Teacher's Sign.: _____