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 **Shravani R. Pore** 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/D15B **A.Y.:** 23-24

Faculty Incharge : Mrs. Kajal Joseph.

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

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

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: Flutter: Twitter(X) clone / Pwa: Ashop Ecom **Roll No. 45**

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

MAD & PWA Lab
Journal

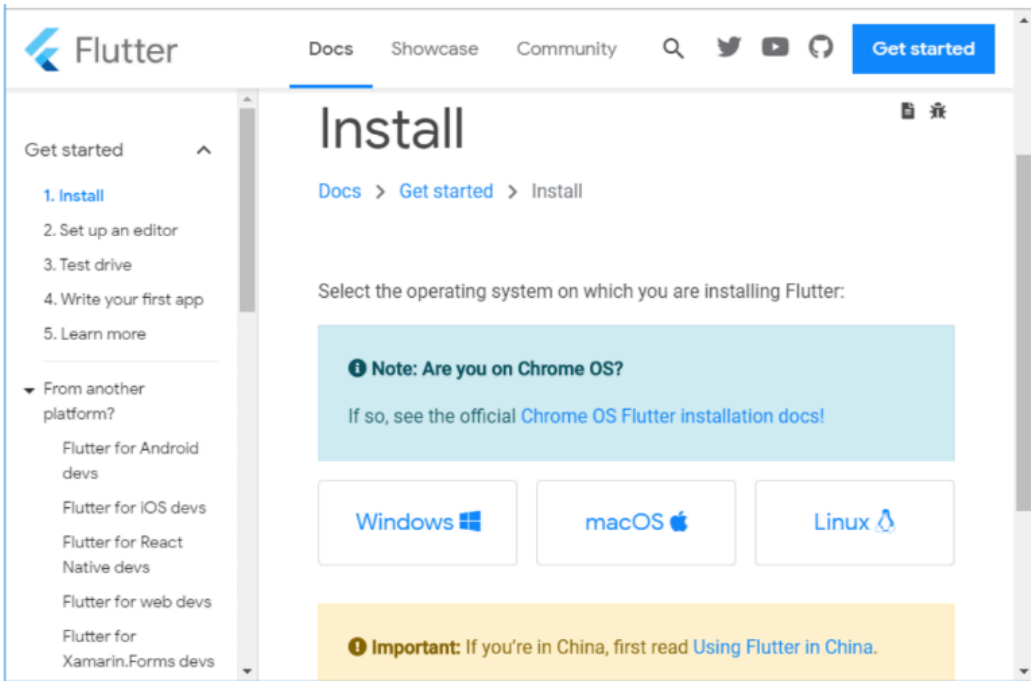
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	45
Name	Shravani R. Pore
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15

Aim:

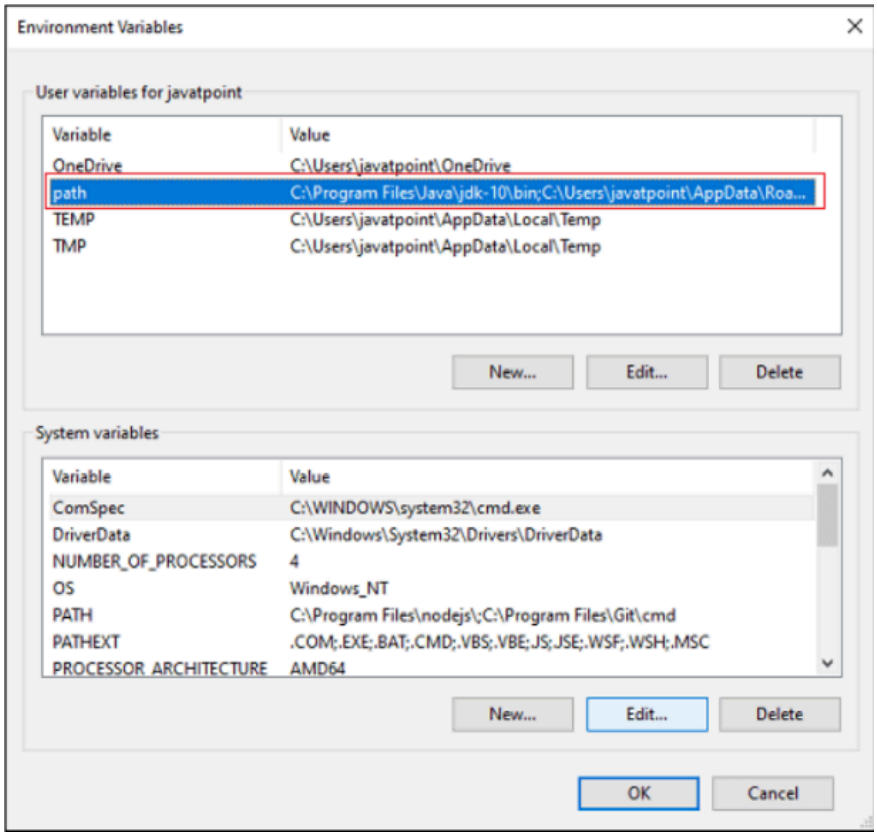
1.Installation and configuration of flutter environment

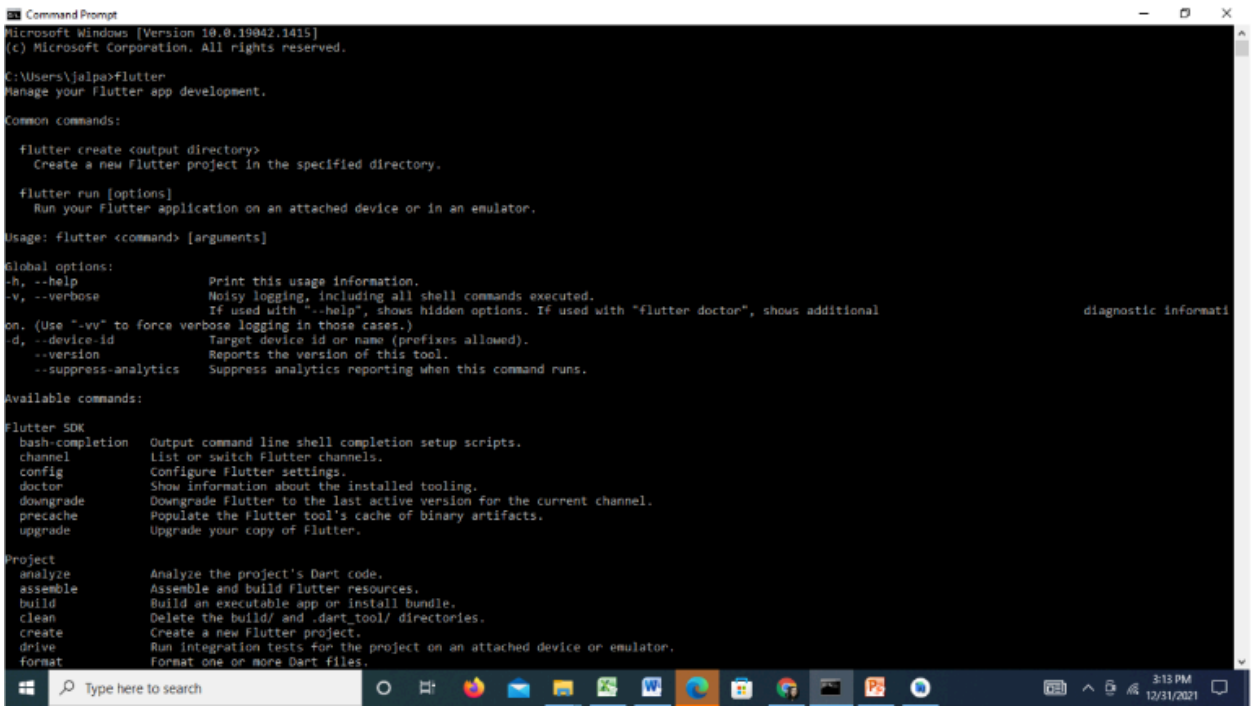
2.Create a 'HELLO WORLD APP' using flutter

Installation:



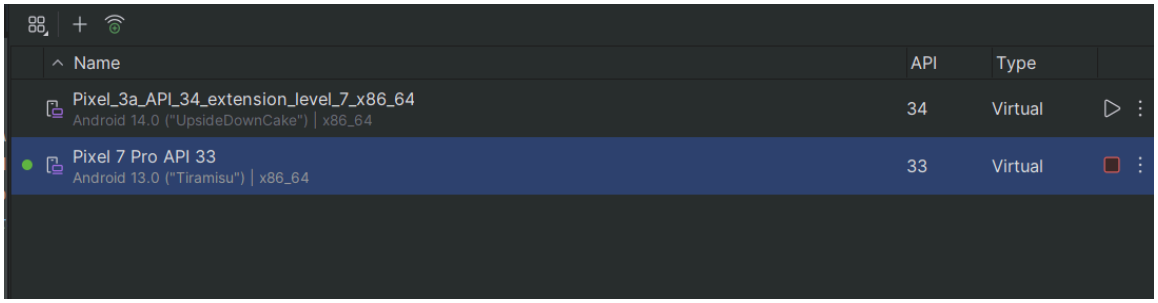
Project Title: Flutter: Twitter(X) clone / Pwa: Ashop Ecom **Roll No. 45**



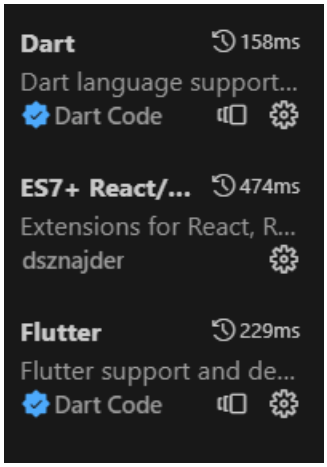


Setup:

Created a new virtual device from tools->device manager.



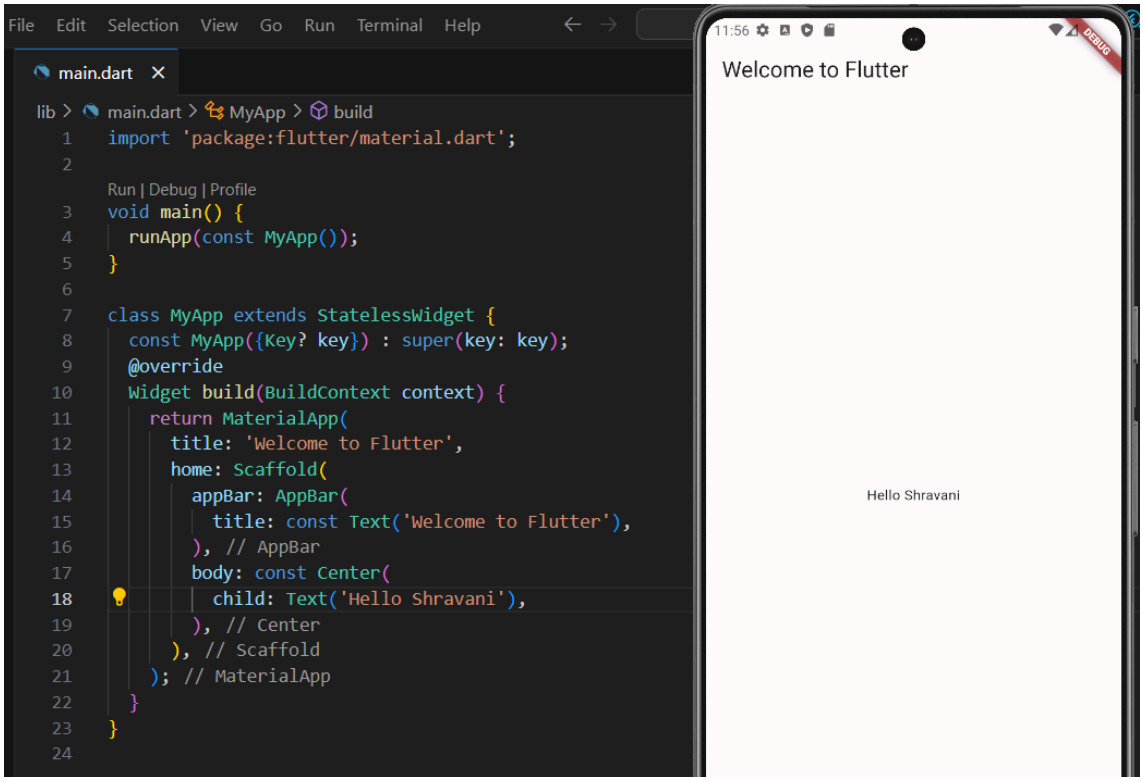
Downloaded flutter and dart extension in vs code.



First application:

Project Title: Flutter: Twitter(X) clone / Pwa: Ashop Ecom **Roll No. 45**

Code and output.



```dart
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 Shravani'),
        ), // Center
      ), // Scaffold
    ); // MaterialApp
  }
}
```

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	45
Name	Shravani R. Pore
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

MAD PWA LAB 2

AIM:
To design Flutter UI by including common widgets.

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

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

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

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

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

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

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

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

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

TextField:
Allows users to input text, providing a UI for text entry.

SYNTAX:
Syntaxes for some of common widgets are as follows:
1.Text

```
Text(
“Text Content”,
style:TextStyle(
```

```
//style properties
)
)
```

2.Container

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

3.TextButton

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

WIDGET AND PROPERTIES:

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

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

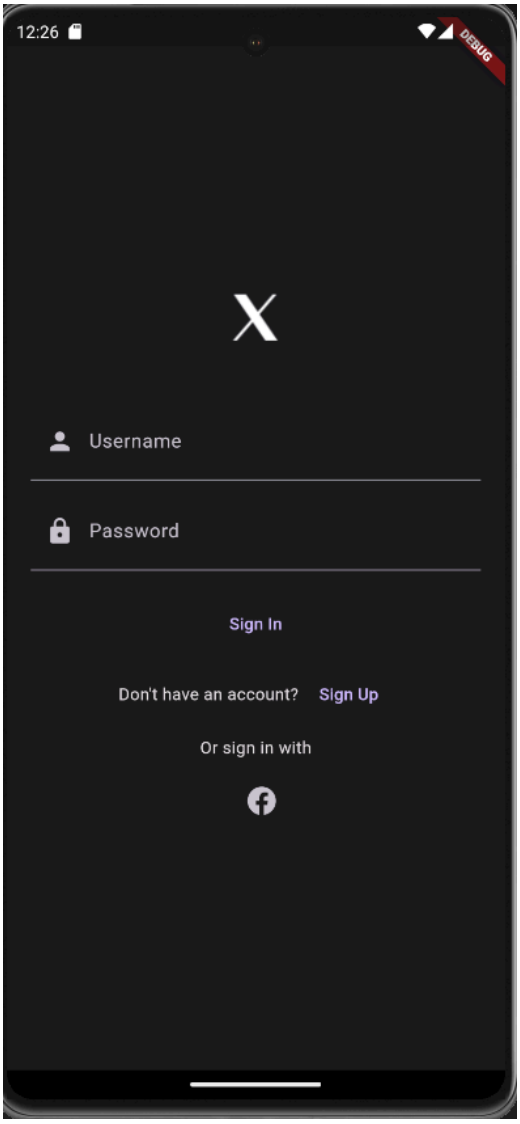
CODE:

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

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData.dark(),
      home: LoginPage(),
    );
  }
}
```

```
class LoginPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Image.asset(
              'assets/Xicon.png', // Replace with your logo image path
              height: 100,
              width: 100,
            ),
            SizedBox(height: 20),
            TextField(
              decoration: InputDecoration(
                labelText: 'Username',
                prefixIcon: Icon(Icons.person),
              ),
            ),
            SizedBox(height: 10),
            TextField(
              decoration: InputDecoration(
                labelText: 'Password',
                prefixIcon: Icon(Icons.lock),
              ),
              obscureText: true,
            ),
            SizedBox(height: 20),
            TextButton(
              onPressed: () {
              },
              child: Text('Sign In'),
            ),
            SizedBox(height: 10),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text('Don\'t have an account?'),
                SizedBox(width: 5),
                TextButton(
                  onPressed: () {
```

```
                  // Navigate to the sign-up page (not implemented in this example)
                },
                child: Text('Sign Up'),
              ),
            ],
          ),
          SizedBox(height: 10),
          Text('Or sign in with'),
          SizedBox(height: 10),
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              SizedBox(width: 10),
              IconButton(
                icon: FaIcon(FontAwesomeIcons.facebook),
                onPressed: () {
                },
              ),
            ],
          ),
        ],
      ),
    ),
  );
 }
}
```

OUTPUT:



CONCLUSION:

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

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	45
Name	Shravani R. Pore
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

MAD PWA LAB 3

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

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

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

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

SYNTAX:
1.Icons:
Icon(Icons.star);

If we are using FontAwesomeIcons Library:
FaIcon(FontAwesomeIcons.star);
(Also this library needs to be added to pubspec.yaml file to be able to be used.)

2.Images:
Loading image from assets
Image.asset('assets/image.png');

Loading image from the internet
Image.network('https://example.com/image.jpg');

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

(This font family needs to be added to the pubspec.yaml file to be able to be used.)

WIDGETS AND PROPERTIES:

The widgets we focussed in this lab are:

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

CODE:

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Profile Page'),
      ),
      body: Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          // Header image
          Container(
            height: 170,
            decoration: const BoxDecoration(
              image: DecorationImage(
                image: NetworkImage(
                    'https://images.shiksha.com/mediadata/images/1553752427phpvP6G9K.png'),
                fit: BoxFit.cover,
              ),
            ),
            child: const Stack(
              children: [
                // Profile picture overlapping header image
                Positioned(
                  top: 100,
                  left: 20,
                  child: CircleAvatar(
                    radius: 50,
                    backgroundImage: NetworkImage(

'https://resize.indiatvnews.com/en/resize/newbucket/1080_-/2023/06/untitled-design-2023-06-20
t120715-1687246152.jpg'),
```

```
          ),
        ),
      ],
    ),
  ),
  // Bio container
  Container(
    padding: const EdgeInsets.all(20),
    height: 250,
    child: const Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Row(
          children: [
            Icon(Icons.account_circle),
            SizedBox(width: 10),
            Text(
              'Shravani',
              style:
                  TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
            ),
          ],
        ),
        SizedBox(height: 10),
        Row(
          children: [
            Icon(Icons.info),
            SizedBox(width: 10),
            Expanded(
              child: Text(
                'My flutter project',
                style: TextStyle(fontSize: 16),
              ),
            ),
          ],
        ),
        SizedBox(height: 10),
        Row(
          children: [
            Icon(Icons.location_on),
            SizedBox(width: 10),
            Text(
              'Chembur',
              style: TextStyle(fontSize: 16),
```

```
                ),
              ],
            ),
            SizedBox(height: 10),
            Row(
              children: [
                Icon(FontAwesomeIcons.birthdayCake),
                SizedBox(width: 10),
                Text(
                  'Born 1 Aug 2003',
                  style: TextStyle(fontSize: 16),
                ),
                SizedBox(width: 20), // Adjust the width as needed
                Icon(FontAwesomeIcons.calendarAlt),
                SizedBox(width: 10),
                Text(
                  'Joined 1 Feb 2024',
                  style: TextStyle(fontSize: 16),
                ),
              ],
            ),
          ],
        ),
      ),
      // Tweets
      Expanded(
        child: ListView.builder(
          itemCount: 10, // Number of tweets
          itemBuilder: (context, index) {
            return Card(
              margin: const EdgeInsets.symmetric(
                  horizontal: 10,
                  vertical: 10), // Increased vertical margin
              elevation: 3, // Adds a shadow
              child: ListTile(
                title: Text('Tweet $index'),
                subtitle: Text(
                    'Date: ${DateTime.now().toString()}'), // Example of displaying date, replace with
actual tweet date
              ),
            );
          },
        ),
      ),
```

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

OUTPUT:

CONCLUSION:

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

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	45
Name	Shravani R. Pore
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

MAD PWA LAB 4

AIM:
To create an interactive Form using form widget

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

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

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

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

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

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

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

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

WIDGET AND PROPERTIES:

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

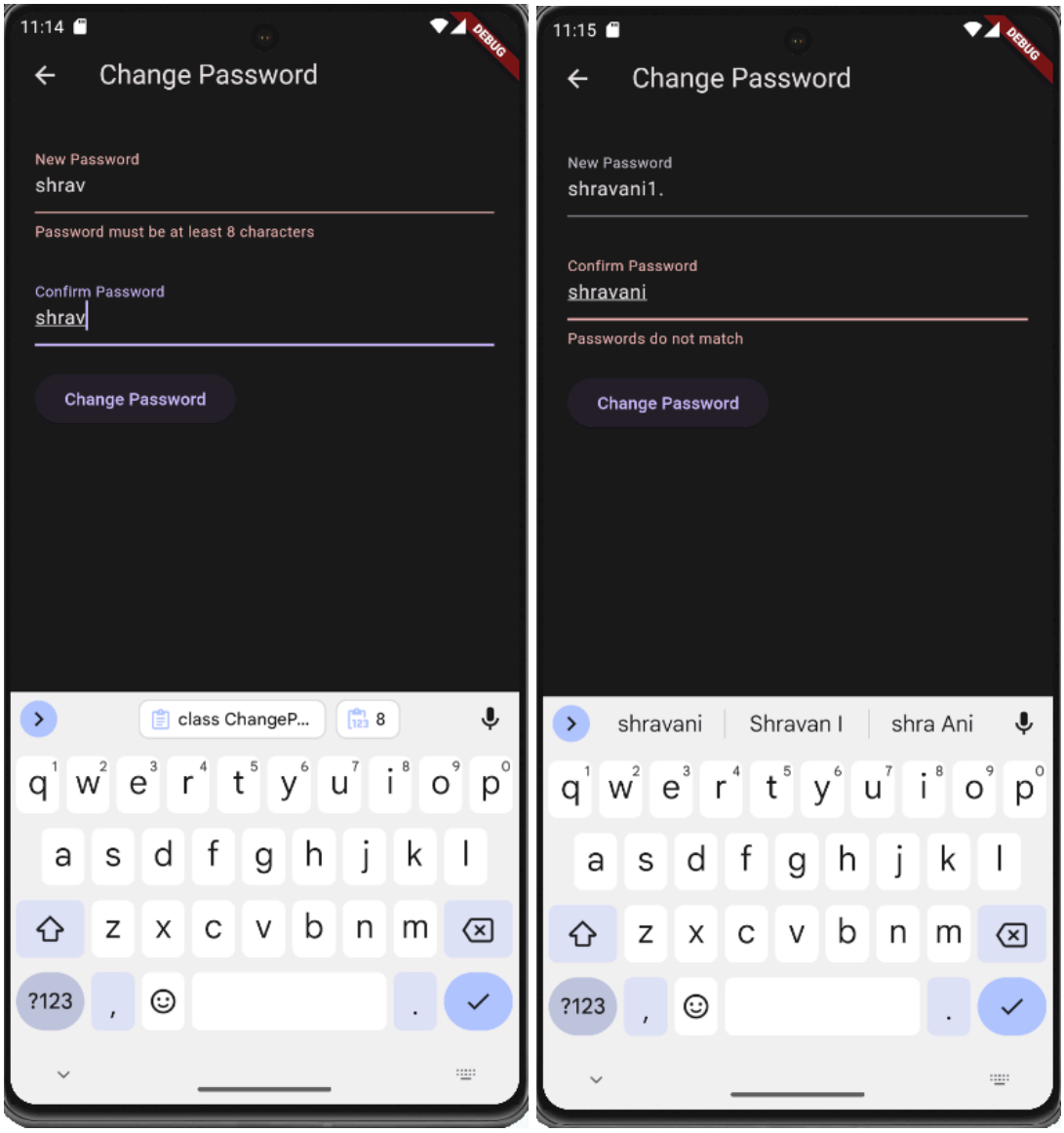
CODE:

```
class ChangePasswordPage extends StatefulWidget {
  @override
  _ChangePasswordPageState createState() => _ChangePasswordPageState();
}

class _ChangePasswordPageState extends State<ChangePasswordPage> {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  TextEditingController _passwordController = TextEditingController();
  TextEditingController _confirmPasswordController =
TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Change Password'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Form(
          key: _formKey,
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              TextFormField(
                controller: _passwordController,
                decoration: InputDecoration(
                  labelText: 'New Password',
                ),
                validator: (value) {
                  if (value!.isEmpty) {
                    return 'Password is required';
                  }
                  if (value.length < 8) {
```

```
                    return 'Password must be at least 8 characters';
                  }
                  if (!value.contains(RegExp(r'[!@#$%^&*(),.?":{}|<>]')))
{
                    return 'Password must contain at least one symbol';
                  }
                  return null;
                },
              ),
              SizedBox(height: 20),
              TextFormField(
                controller: _confirmPasswordController,
                decoration: InputDecoration(
                  labelText: 'Confirm Password',
                ),
                validator: (value) {
                  if (value != _passwordController.text) {
                    return 'Passwords do not match';
                  }
                  return null;
                },
              ),
              SizedBox(height: 20),
              ElevatedButton(
                onPressed: () {
                  if (_formKey.currentState!.validate()) {
                    Navigator.pushNamed(context, '/password');
                    ScaffoldMessenger.of(context).showSnackBar(
                      SnackBar(
                        content: Text('Password changed successfully.'),
                      ),
                    );
                  }
                },
                child: Text('Change Password'),
              ),
            ],
          ),
        ),
      ),
```

```
    );
  }

  @override
  void dispose() {
    _passwordController.dispose();
    _confirmPasswordController.dispose();
    super.dispose();
  }
}
```

OUTPUT:



CONCLUSION:

Learnt about form widget and applied it in my project.

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	45
Name	Shravani R. Pore
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

MAD PWA LAB 5

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

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

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

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

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

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

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

);

WIDGET AND PROPERTIES:

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

CODE:

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

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData.dark(),
      home: LoginPage(),
      routes: {
        '/profile': (context) =>
            const ProfilePage(), // Route to the profile page
        '/edit_profile': (context) =>
            ChangePasswordPage(), // Route to the password page
      },
    );
  }
}

class LoginPage extends StatelessWidget {
  const LoginPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
```

```
            Image.asset(
              'assets/Xicon.png', // Replace with your logo image path
              height: 100,
              width: 100,
            ),
            const SizedBox(height: 20),
            const TextField(
              decoration: InputDecoration(
                labelText: 'Username',
                prefixIcon: Icon(Icons.person),
              ),
            ),
            const SizedBox(height: 10),
            const TextField(
              decoration: InputDecoration(
                labelText: 'Password',
                prefixIcon: Icon(Icons.lock),
              ),
              obscureText: true,
            ),
            const SizedBox(height: 20),
            TextButton(
              onPressed: () {
                Navigator.pushNamed(
                    context, '/profile'); // Navigate to profile page
              },
              child: const Text('Sign In'),
            ),
            const SizedBox(height: 10),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                const Text('Don\'t have an account?'),
                const SizedBox(width: 5),
                TextButton(
                  onPressed: () {
                    // Navigate to the sign-up page (not implemented in
this example)
                  },
                  child: const Text('Sign Up'),
```

```
              ),
            ],
          ),
          const SizedBox(height: 10),
          const Text('Or sign in with'),
          const SizedBox(height: 10),
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              const SizedBox(width: 10),
              IconButton(
                icon: const FaIcon(FontAwesomeIcons.facebook),
                onPressed: () {},
              ),
            ],
          ),
        ],
      ),
    ),
  );
 }
}

class ProfilePage extends StatelessWidget {
  const ProfilePage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Profile Page'),
      ),
      drawer: Drawer(
        child: ListView(
          padding: EdgeInsets.zero,
          children: [
            const UserAccountsDrawerHeader(
              accountName: Text('Shravani'), // Replace with actual
username
              accountEmail: null,
```

```
              currentAccountPicture: CircleAvatar(
                backgroundImage: NetworkImage(

'https://resize.indiatvnews.com/en/resize/newbucket/1080_-/2023/06/untitle
d-design-2023-06-20t120715-1687246152.jpg',
                ),
              ),
              decoration: BoxDecoration(
                color: Color.fromARGB(255, 52, 52, 52),
              ),
            ),
            ListTile(
              title: const Text('Profile'),
              onTap: () {
                Navigator.pop(context);
              },
            ),
            ListTile(
              title: const Text('Edit Profile'),
              onTap: () {
                Navigator.pop(context);
                Navigator.pushNamed(
                    context, '/edit_profile'); // Navigate to password
page
              },
            ),
          ],
        ),
      ),
      body: Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          // Header image
          Container(
            height: 170,
            decoration: const BoxDecoration(
              image: DecorationImage(
                image: NetworkImage(

'https://images.shiksha.com/mediadata/images/1553752427phpvP6G9K.png',
```

```
                ),
                fit: BoxFit.cover,
              ),
            ),
            child: const Stack(
              children: [
                // Profile picture overlapping header image
                Positioned(
                  top: 100,
                  left: 20,
                  child: CircleAvatar(
                    radius: 50,
                    backgroundImage: NetworkImage(

'https://resize.indiatvnews.com/en/resize/newbucket/1080_-/2023/06/untitle
d-design-2023-06-20t120715-1687246152.jpg',
                    ),
                  ),
                ),
              ],
            ),
          ),
          // Bio container
          Container(
            padding: const EdgeInsets.all(20),
            height: 250,
            child: const Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Row(
                  children: [
                    Icon(Icons.account_circle),
                    SizedBox(width: 10),
                    Text(
                      'Shravani',
                      style:
                          TextStyle(fontSize: 24, fontWeight:
FontWeight.bold),
                    ),
                  ],
```

```
            ),
            SizedBox(height: 10),
            Row(
              children: [
                Icon(Icons.info),
                SizedBox(width: 10),
                Expanded(
                  child: Text(
                    'My flutter project',
                    style: TextStyle(fontSize: 16),
                  ),
                ),
              ],
            ),
            SizedBox(height: 10),
            Row(
              children: [
                Icon(Icons.location_on),
                SizedBox(width: 10),
                Text(
                  'Chembur',
                  style: TextStyle(fontSize: 16),
                ),
              ],
            ),
            SizedBox(height: 10),
            Row(
              children: [
                Icon(FontAwesomeIcons.birthdayCake),
                SizedBox(width: 10),
                Text(
                  'Born 1 Aug 2003',
                  style: TextStyle(fontSize: 16),
                ),
                SizedBox(width: 20), // Adjust the width as needed
                Icon(FontAwesomeIcons.calendarAlt),
                SizedBox(width: 10),
                Text(
                  'Joined 1 Feb 2024',
                  style: TextStyle(fontSize: 16),
```

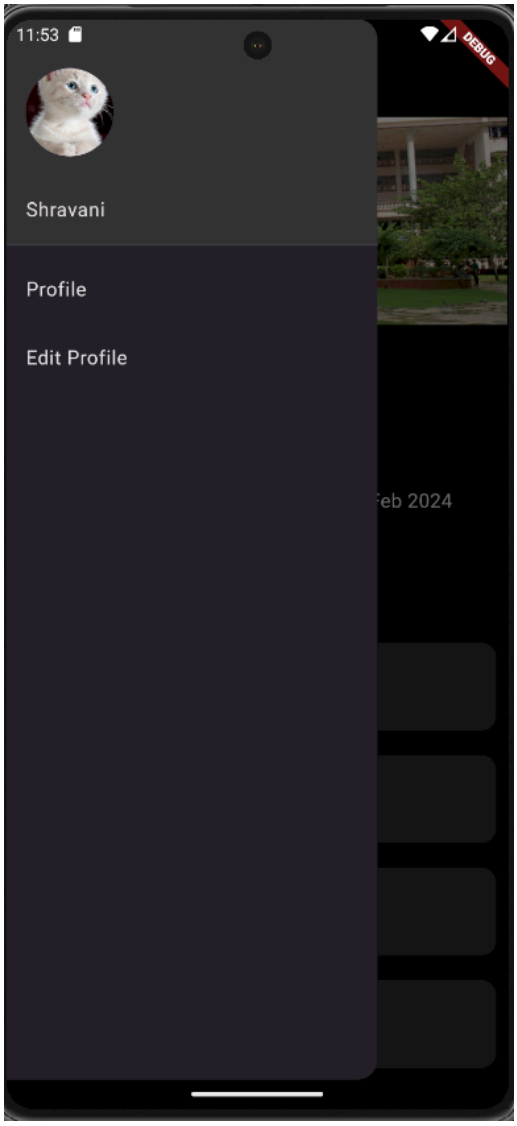
```
                    ),
                  ],
                ),
              ],
            ),
          ),
          // Tweets
          Expanded(
            child: ListView.builder(
              itemCount: 10, // Number of tweets
              itemBuilder: (context, index) {
                return Card(
                  margin: const EdgeInsets.symmetric(
                      horizontal: 10,
                      vertical: 10), // Increased vertical margin
                  elevation: 3, // Adds a shadow
                  child: ListTile(
                    title: Text('Tweet $index'),
                    subtitle: Text(
                        'Date: ${DateTime.now().toString()}'), // Example
of displaying date, replace with actual tweet date
                  ),
                );
              },
            ),
          ),
        ],
      ),
    );
  }
}

// class PasswordPage extends StatelessWidget {
//   const PasswordPage({Key? key}) : super(key: key);

//   @override
//   Widget build(BuildContext context) {
//     return Scaffold(
//       appBar: AppBar(
//         title: const Text('Password Page'),
```

```
//       ),
//       body: Center(
//         child: Text('Password Page'),
//       ),
//     );
//   }
// }

class ChangePasswordPage extends StatefulWidget {
  @override
  _ChangePasswordPageState createState() => _ChangePasswordPageState();
}

class _ChangePasswordPageState extends State<ChangePasswordPage> {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  TextEditingController _passwordController = TextEditingController();
  TextEditingController _confirmPasswordController =
TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Change Password'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Form(
          key: _formKey,
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              TextFormField(
                controller: _passwordController,
                decoration: InputDecoration(
                  labelText: 'New Password',
                ),
                validator: (value) {
                  if (value!.isEmpty) {
                    return 'Password is required';
```

```
                }
                if (value.length < 8) {
                  return 'Password must be at least 8 characters';
                }
                if (!value.contains(RegExp(r'[!@#$%^&*(),.?":{}|<>]')))
{
                  return 'Password must contain at least one symbol';
                }
                return null;
              },
            ),
            SizedBox(height: 20),
            TextFormField(
              controller: _confirmPasswordController,
              decoration: InputDecoration(
                labelText: 'Confirm Password',
              ),
              validator: (value) {
                if (value != _passwordController.text) {
                  return 'Passwords do not match';
                }
                return null;
              },
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: () {
                if (_formKey.currentState!.validate()) {
                  Navigator.pushNamed(context, '/password');
                  ScaffoldMessenger.of(context).showSnackBar(
                    SnackBar(
                      content: Text('Password changed successfully.'),
                    ),
                  );
                }
              },
              child: Text('Change Password'),
            ),
          ],
        ),
```

```
        ),
      ),
    );
  }

  @override
  void dispose() {
    _passwordController.dispose();
    _confirmPasswordController.dispose();
    super.dispose();
  }
}
```

OUTPUT:



Clicking on any of these options would navigate to those pages. We have used router, navigation and gestures in this

CONCLUSION:

Learned about navigation, routing and gestures.

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	45
Name	Shravani R. Pore
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	15

MAD PWA LAB 6

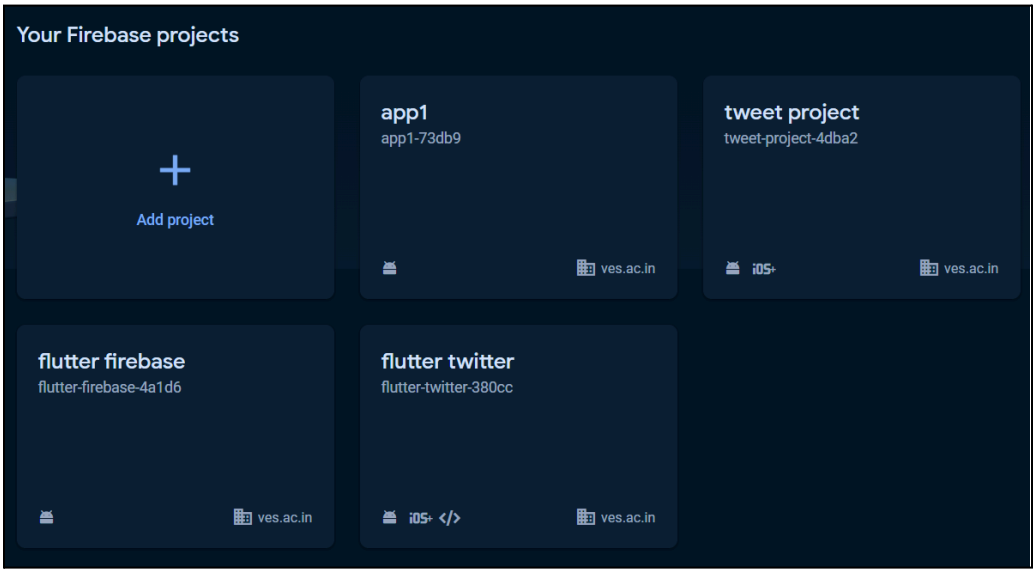
AIM:

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

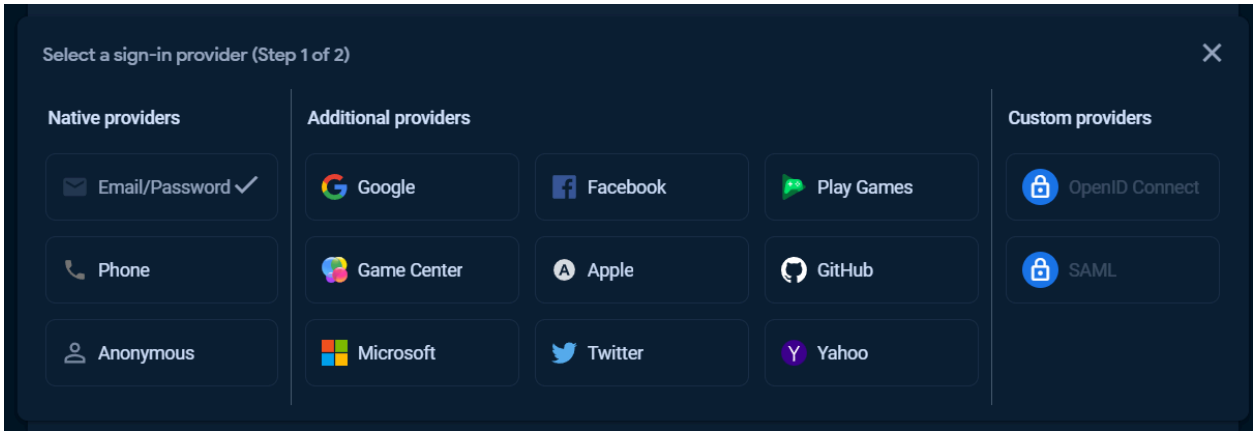
THEORY / STEPS:

(Wherever testing mode is available, I have opted for testing mode.)

1.Create a project in firebase console.



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



3. Similarly enable firestore database.

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

4.Installing firebase CLI

npm install -g firebase-tools

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

firebase login

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

6. Activation of flutter fire cli

Dart pub global activate flutterfire_cli

7. Configuration of flutterfire

Flutterfire configure

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

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

9. Configured applications will be now available in firebase.

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

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

11. We add dependencies:

```
cupertino_icons: ^1.0.2
  firebase_core: ^2.25.4
  firebase_auth: ^4.17.4
  cloud_firestore: 4.15.5
  provider:
```

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

CODE:
Auth_gate.dart

```
// ignore: unused_import
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:twitter/main.dart';

class AuthGate extends StatelessWidget {
```

```
  static final GlobalKey<NavigatorState> navigatorKey = GlobalKey();
  const AuthGate({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        body: StreamBuilder(
      stream: FirebaseAuth.instance.authStateChanges(),
      builder: (context, snapshot) {
        //user is logged in
        if (snapshot.hasData) {
          return ProfilePage();
        }
        //user is not logged in
        else {
          return LoginPage();
        }
      },
    ));
  }
}
```

Auth_service.dart

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

class AuthService extends ChangeNotifier {
  final FirebaseAuth _firebaseAuth = FirebaseAuth.instance;

  //sign user in
  Future<UserCredential> signInWithEmailandPassword(
      String email, String password) async {
    try {
      //signin
      UserCredential userCredential =
          await _firebaseAuth.signInWithEmailAndPassword(
        email: email,
        password: password,
      );
```

```
      return userCredential;
    } on FirebaseAuthException catch (e) {
      //catch errors
      throw Exception(e.code);
    }
  }

  Future<void> changePassword(String newPassword) async {
    try {
      // Get the currently logged-in user
      User? user = _firebaseAuth.currentUser;

      // Update the password
      await user!.updatePassword(newPassword);

      // Sign out the user after changing the password
      await signOut();

      // You may choose to navigate the user to the sign-in page or any
other page after password change
    } catch (e) {
      // Handle any errors that occur during password change
      print('Error changing password: $e');
      throw Exception(e);
    }
  }

  //create a new user
  Future<UserCredential> signUpWithEmailandPassword(
      String email, String password) async {
    try {
      //signin
      UserCredential userCredential =
          await _firebaseAuth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );
      return userCredential;
    } on FirebaseAuthException catch (e) {
      //catch errors
```

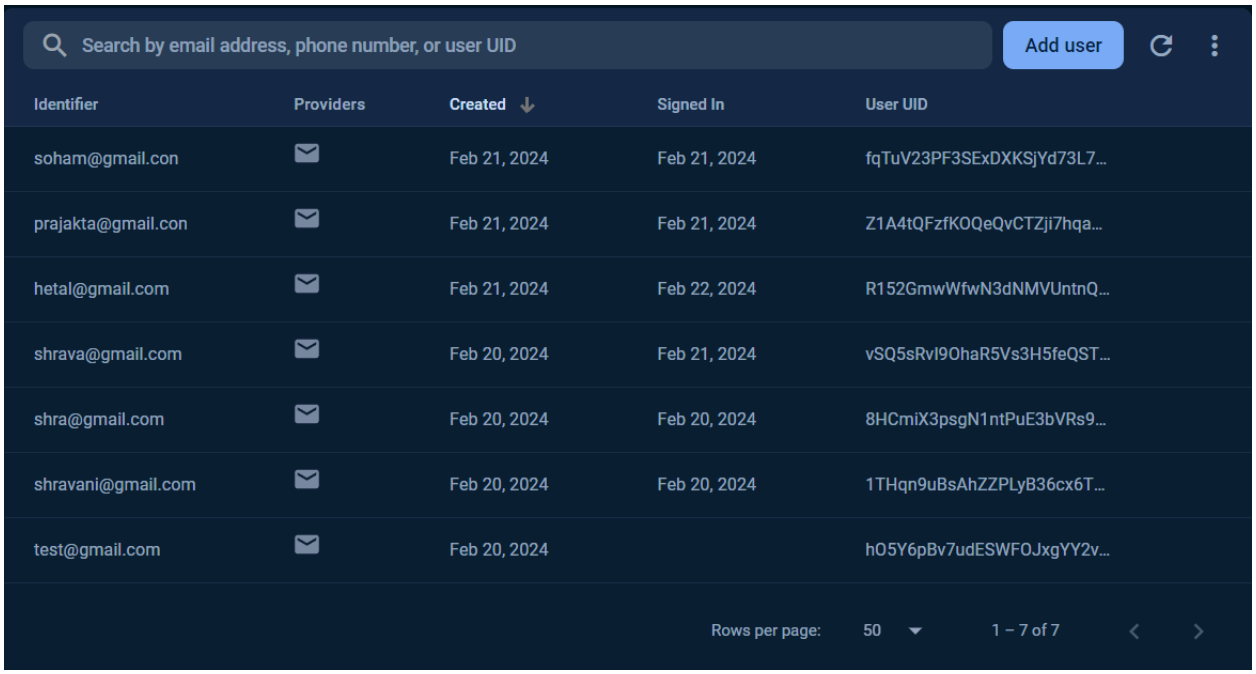
```
      throw Exception(e.code);
    }
  }

  //sign user out
  Future<void> signOut() async {
    return await FirebaseAuth.instance.signOut();
  }
}

//saving tweets
Future<void> saveTweet(String tweetText) async {
  try {
    String uid = FirebaseAuth.instance.currentUser!.uid;
    await FirebaseFirestore.instance
        .collection('users')
        .doc(uid)
        .collection('tweets')
        .add({
      'text': tweetText,
      'timestamp': DateTime.now(),
    });
    print('Tweet added successfully');
  } catch (e) {
    print('Error adding tweet: $e');
  }
}
```

The functions defined here are called in the pages files. Also some code is added to main file itself and not in the Auth_service file.

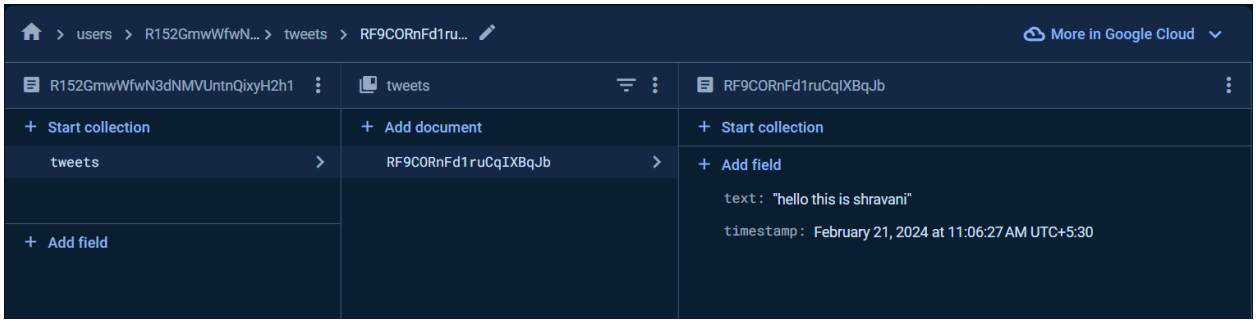
Project Title: Flutter: Twitter(X) clone / Pwa: Ashop Ecom **Roll No. 45**

OUTPUT:

Authentication:



Tweets in firestore database:



With this connection we are able to achieve the following:

- **Saving email and password in database.**
- **Saving tweets and their timestamps in the database under current user's uid.**
- **Fetching tweets from the database to display on the profile page.**
- **Updating the values in the database for password in the authentication database.**

CONCLUSION:

Connected firebase with twitter clone project and handled related errors.

MAD & PWA Lab
Journal

Experiment No.	07
Experiment Title.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.
Roll No.	45
Name	Shravani R. Pore
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	15

MAD PWA LAB 7

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

Theory:
A regular web app is accessed through a web browser and requires an active internet connection to function. These apps often have limited offline capabilities and may not offer a seamless user experience on mobile devices.

On the other hand, a Progressive Web App (PWA) is a type of web application that leverages modern web technologies to provide a more app-like experience to users. PWAs are designed to work across various platforms and devices, including desktops, tablets, and smartphones. They can be accessed through a web browser like regular web apps but offer additional features and capabilities, such as offline access, push notifications, and the ability to install them on the user's device.

To enable the "add to homescreen" feature for a PWA, you need to include metadata in the Web app manifest file. The Web app manifest is a JSON file that provides information about the PWA, such as its name, icons, colors, start URL, display mode, and more. By defining these properties in the manifest file, you can ensure that the PWA is displayed correctly when users add it to their device's homescreen.

Key features of a PWA that contribute to its app-like experience include:

- Responsive Design: PWAs are built with responsive design principles, ensuring that they look and work well across different screen sizes and devices.
- Offline Access: PWAs can cache assets and data, allowing users to access content even when they are offline or have a poor internet connection.
- Fast Loading: PWAs are optimized for speed and performance, delivering fast loading times and smooth navigation.
- Engagement: PWAs can send push notifications to users, increasing user engagement and retention.
- Installable: Users can add PWAs to their device's homescreen for easy access, similar to native apps.

Overall, PWAs combine the best of web and app experiences, offering a lightweight, fast, and engaging way for users to interact with web content on various devices.

Code:

Serviceworker.js

```
var staticCacheName = "pwa";

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

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

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

Manifest.json

```
{
  "name": "To-Do List",
  "short_name": "PWA",
  "start_url": "index.html",
  "display": "standalone",
  "background_color": "#5900b3",
  "theme_color": "black",
  "scope": "./",
  "description": "This is a PWA tutorial.",
  "icons": [
    {
      "src": "C:\\Users\\Shravani
Pore\\Downloads\\ToDoList-main\\ToDoList-main\\pwaimage12.png",
      "sizes": "192x192",
```

```
      "type": "image/png",
      "purpose": "any maskable"
    },
    {
      "src": "C:\\Users\\Shravani
Pore\\Downloads\\ToDoList-main\\ToDoList-main\\pwaimage1.png",
      "sizes": "512x512",
      "type": "image/png",
      "purpose": "any maskable"
    }
  ]
}
```

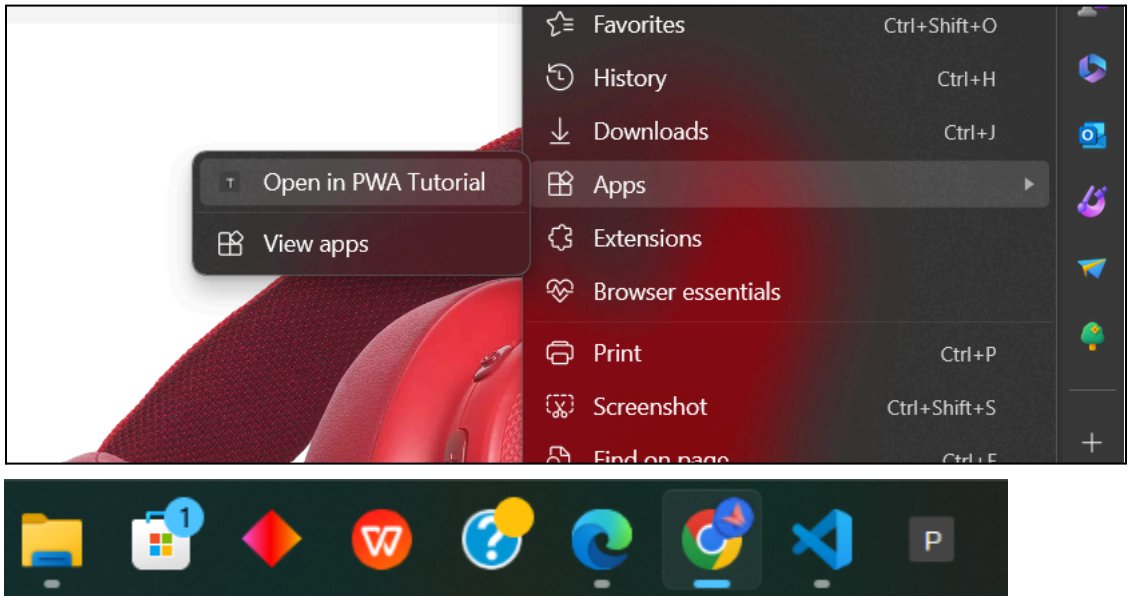
Add following in script of applicatio:

```
<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>
```

Now run the app on the browser and on the right hand side three dots will appear. Click on those dots and go to more options -> developer tools -> applications

Then go to apps section in the three dots menu and install the app:

Output:



Conclusion:

Successfully added meta data to my Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.

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.	45
Name	Shravani R. Pore
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

MAD PWA LAB 8

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.

Things we can 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.

Things we can't do with service worker:

- 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.

Code:

Service-worker.js:

```
var cacheName = "Headphone";
self.addEventListener("install", (event) => {
  event.waitUntil(
    caches.open(cacheName).then((cache) => {
return cache.addAll([
  "/",
  "/index.html",
  // "/index.js",
  "css/grid.css",
  "/product-detail.html",
  "/products.html",
]);
    })
  );
});
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);
          })
      );
    })
```

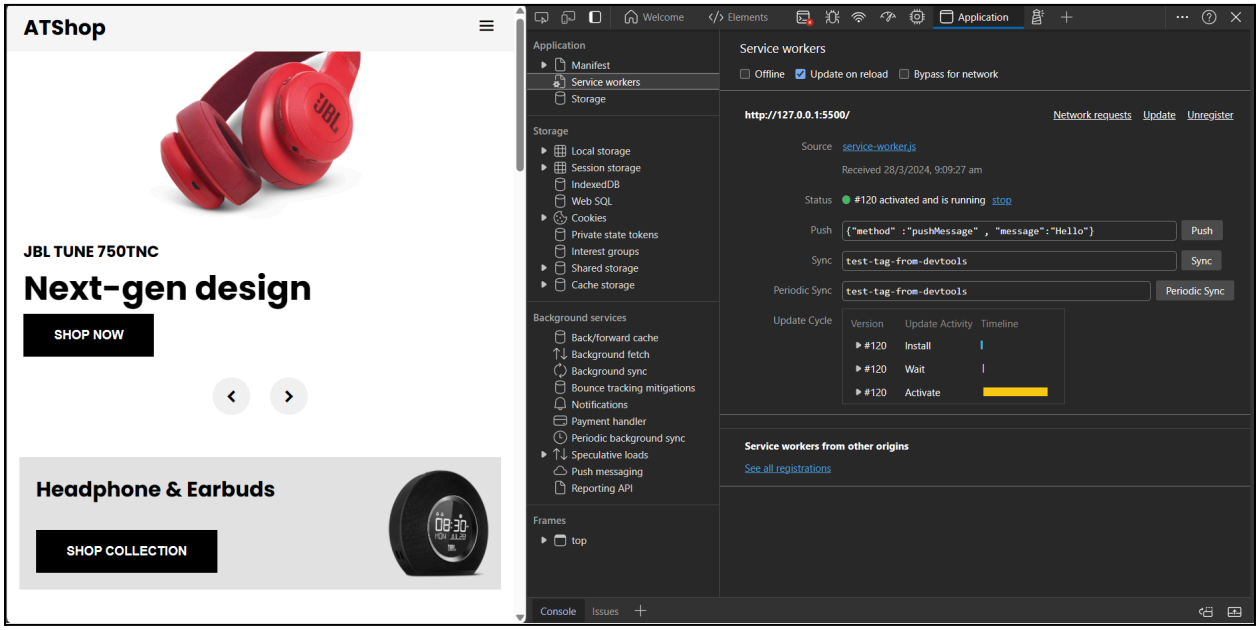
```
  );
});
```

App.js:

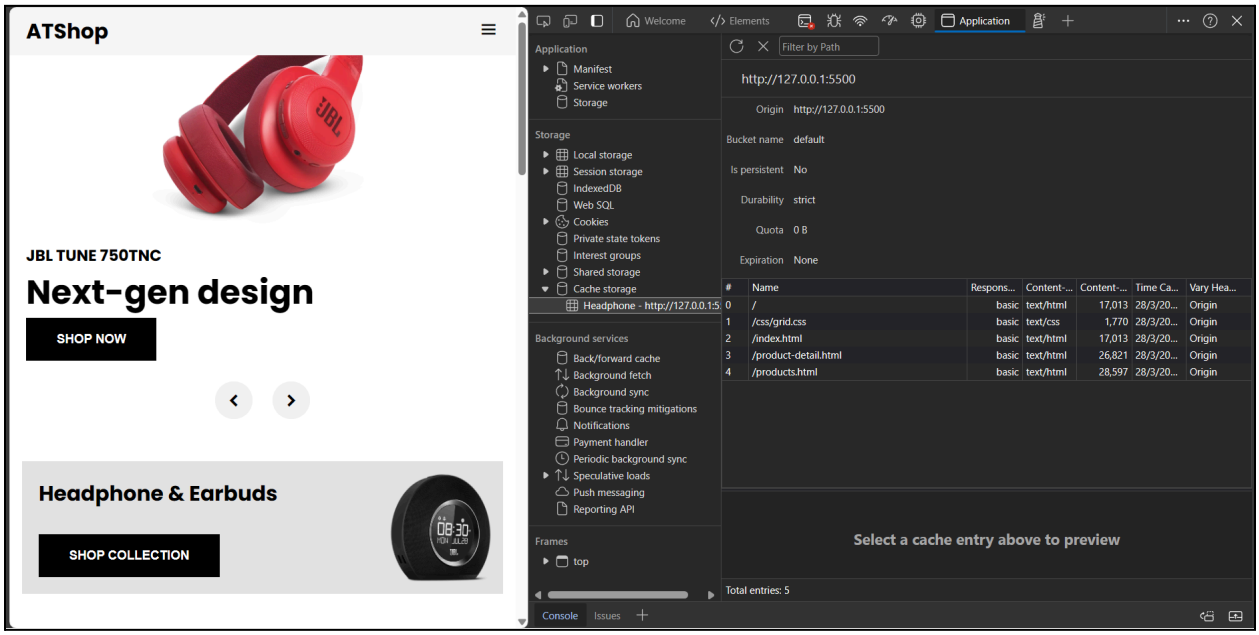
```
if ("serviceWorker" in navigator) {
  window.addEventListener("load", () => {
    navigator.serviceWorker
      .register("/service-worker.js")
      .then((registration) => {
        console.log(
          "Service Worker registered with scope:",
          registration.scope
        );
      })
      .catch((error) => {
        console.error("Service Worker registration failed:", error);
      });
  });
}
if ("Notification" in window) {
  Notification.requestPermission().then(function (permission) {
    if (permission === "granted") {
      console.log("Notification permission granted.");
    } else {
      console.warn("Notification permission denied.");
    }
  });
}
```

Output:

Three dots on right corner ->more tools -> developer tools -> Applications -> Service workers :



Three dots on right corner ->more tools -> developer tools -> Applications -> Cache storage :



Conclusion:

Successfully implemented a code and register a service worker, and completed the installation and activation process for a new service worker for the E-commerce PWA.

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.	45
Name	Shravani R. Pore
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

MAD PWA Lab 11

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.

Performance:

- First Contentful Paint (FCP): Measures how long it takes for the first content to be painted on the screen.
- Speed Index: Calculates how quickly content is visually displayed during page load.
- Largest Contentful Paint (LCP): Indicates the render time of the largest content element visible in the viewport.
- Time to Interactive (TTI): Measures how long it takes for the page to become fully interactive.

Accessibility:

- A11y: Evaluates the accessibility of the web page based on best practices and standards such as WCAG (Web Content Accessibility Guidelines).
- Color Contrast: Checks if text and background colors have sufficient contrast for readability.
- Keyboard Navigation: Assesses if all interactive elements are accessible via keyboard navigation.

Best Practices:

- Avoids Application Cache: Recommends not using the deprecated Application Cache.
- Uses HTTPS: Encourages using HTTPS to ensure secure communication between the server and the user's browser.
- No Broken Links: Checks for broken links on the page.

SEO:

- Meta Tags: Checks for the presence of important meta tags like title, description, and viewport.
- Structured Data: Evaluates if structured data markup is correctly implemented for search engines.
- Mobile Friendly: Assesses if the page is mobile-friendly and responsive.

Progressive Web App (PWA):

- PWA Checklist: Checks if the web page meets the criteria to be considered a Progressive Web App, such as having a service worker and a web app manifest.

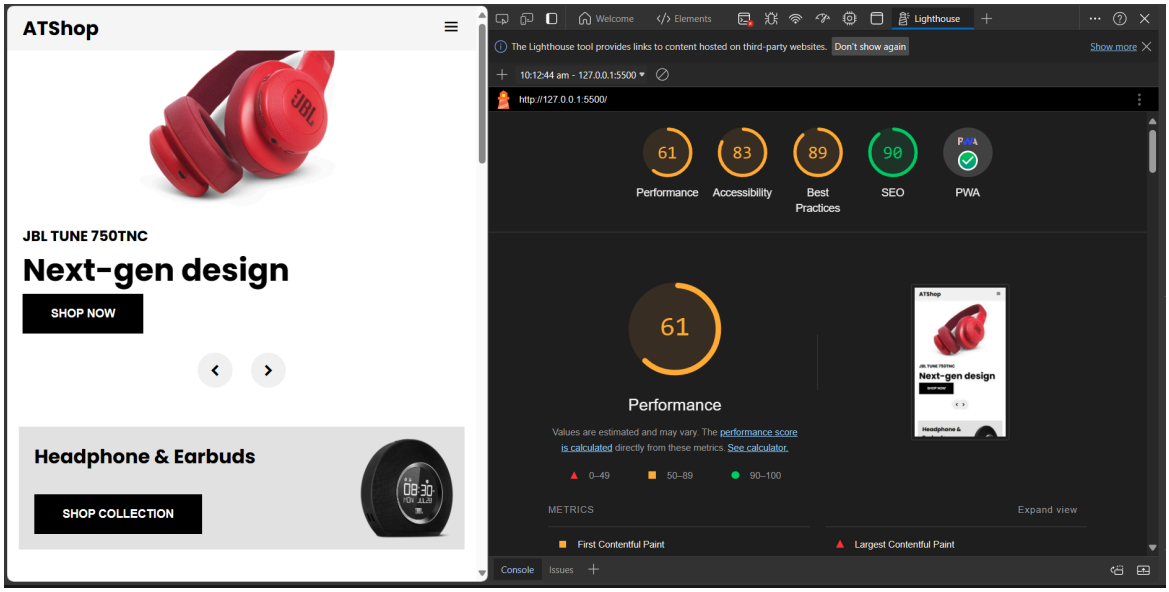
Changes in code:

```
{
  "name": "Ecom",
  "short_name": "PWA",
  "start_url": "index.html",
  "display": "standalone",
  "background_color": "#5900b3",
  "theme_color": "black",
  "scope": ".",
  "description": "This is a PWA tutorial.",
  "icons": [
    {
      "src": "images\\tuat.jpg",
      "sizes": "192x192",
      "type": "image/png",
      "purpose": "any"
    },
    {
      "src": "images\\a.jpg",
      "sizes": "512x512",
      "type": "image/png",
      "purpose": "maskable"
    }
  ]
}
```

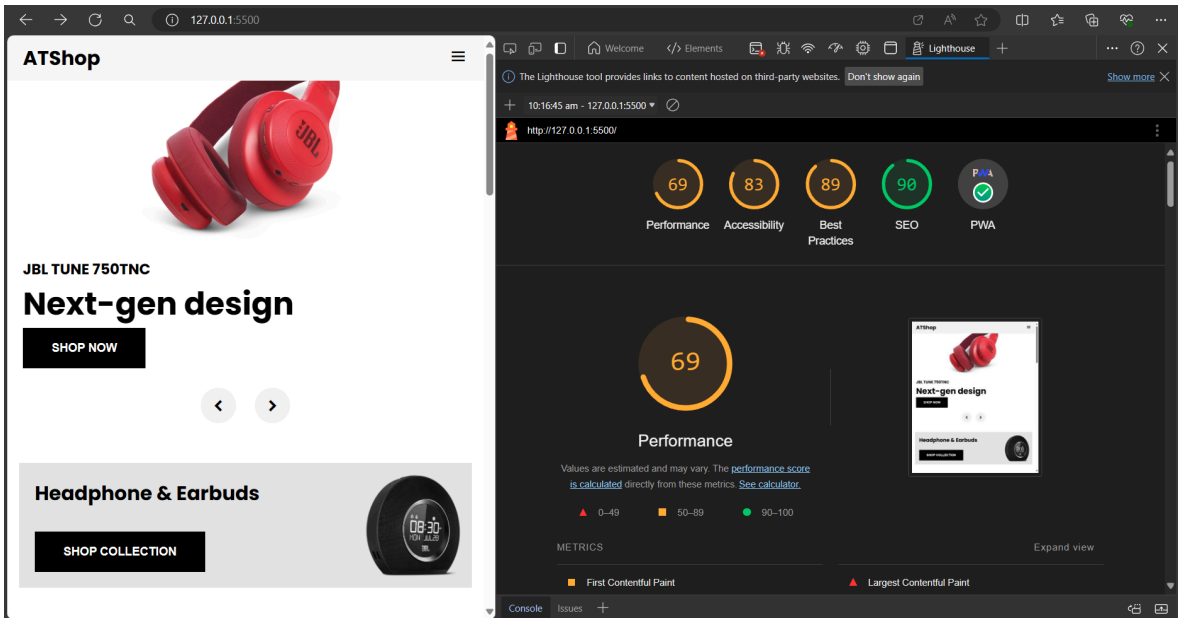
Output:

Three dots on right corner ->more tools -> developer tools -> Lighthouse -> device (mobile) -> Analyze page load

Project Title: Flutter: Twitter(X) clone / Pwa: Ashop Ecom **Roll No. 45**



Three dots on right corner ->more tools -> developer tools -> Lighthouse -> device (mobile) -> Analyze page load



Conclusion:

Successfully implemented a code and register a service worker, and completed the installation and activation process for a new service worker for the E-commerce PWA.

MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	45
Name	Shravani R. Pore
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

MAD PWA LAB 10

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

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

GitHub Pages provides the following key features:

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

Reasons for favoring this over Firebase:

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

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

Pros

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

Cons

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

Firebase

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

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

Pros

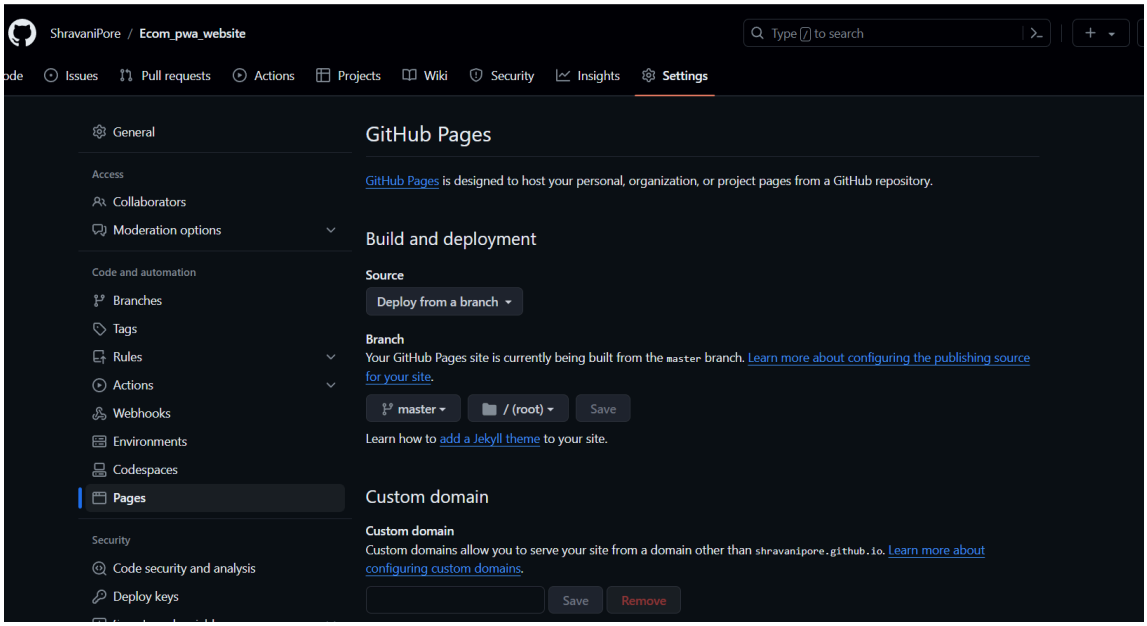
1. Hosted by Google. Enough said.
2. Authentication, Cloud Messaging, and a whole lot of other handy services will be available to you.
3. A real-time database will be available to you, which can store 1 GB of data.
4. You'll also have access to a blob store, which can store another 1 GB of data.
5.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

Cons

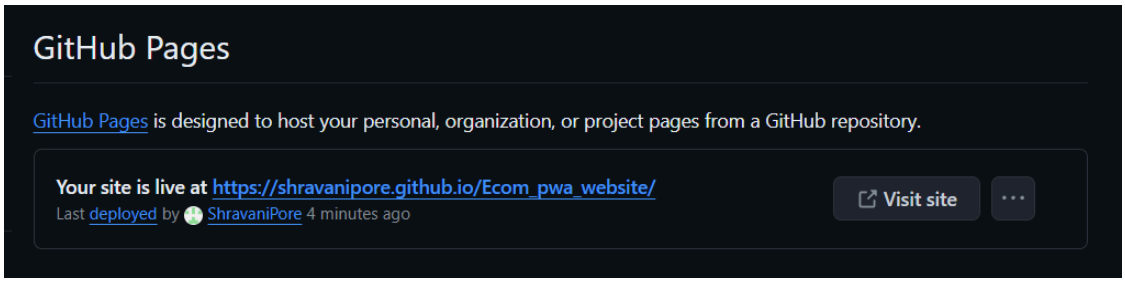
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator.

Steps:

1. **Create a repository.**
2. **Push e-commerce website into the repository.**
3. **Go to settings.**
4. **Click on Pages**

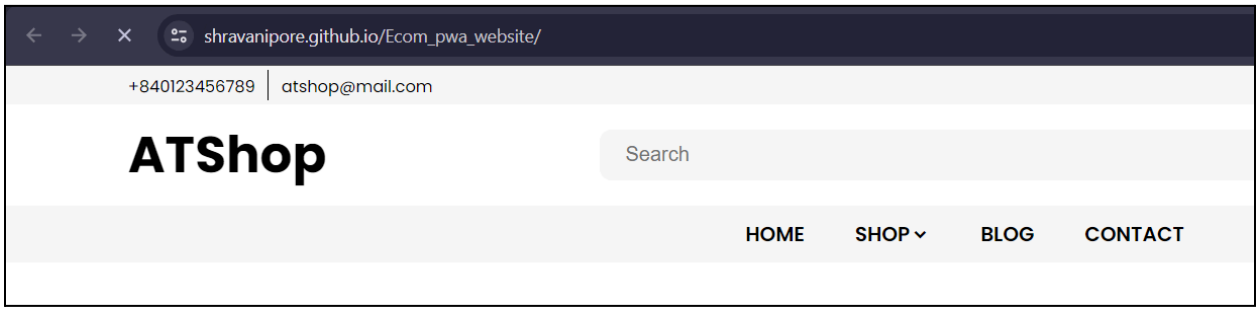


5. **Select the branch you want to deploy your project from.**
6. **After reloading a link will appear at the top of the screen.**



7. **Click on visit site.**

Output:



Conclusion:

Deployed Ecommerce PWA to GitHub Pages.

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	45
Name	Shravani R. Pore
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	15

MAD PWA LAB 11

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.

Performance:

- First Contentful Paint (FCP): Measures how long it takes for the first content to be painted on the screen.
- Speed Index: Calculates how quickly content is visually displayed during page load.
- Largest Contentful Paint (LCP): Indicates the render time of the largest content element visible in the viewport.
- Time to Interactive (TTI): Measures how long it takes for the page to become fully interactive.

Accessibility:

- A11y: Evaluates the accessibility of the web page based on best practices and standards such as WCAG (Web Content Accessibility Guidelines).
- Color Contrast: Checks if text and background colors have sufficient contrast for readability.
- Keyboard Navigation: Assesses if all interactive elements are accessible via keyboard navigation.

Best Practices:

- Avoids Application Cache: Recommends not using the deprecated Application Cache.
- Uses HTTPS: Encourages using HTTPS to ensure secure communication between the server and the user's browser.
- No Broken Links: Checks for broken links on the page.

SEO:

- Meta Tags: Checks for the presence of important meta tags like title, description, and viewport.
- Structured Data: Evaluates if structured data markup is correctly implemented for search engines.
- Mobile Friendly: Assesses if the page is mobile-friendly and responsive.

Progressive Web App (PWA):

- PWA Checklist: Checks if the web page meets the criteria to be considered a Progressive Web App, such as having a service worker and a web app manifest.

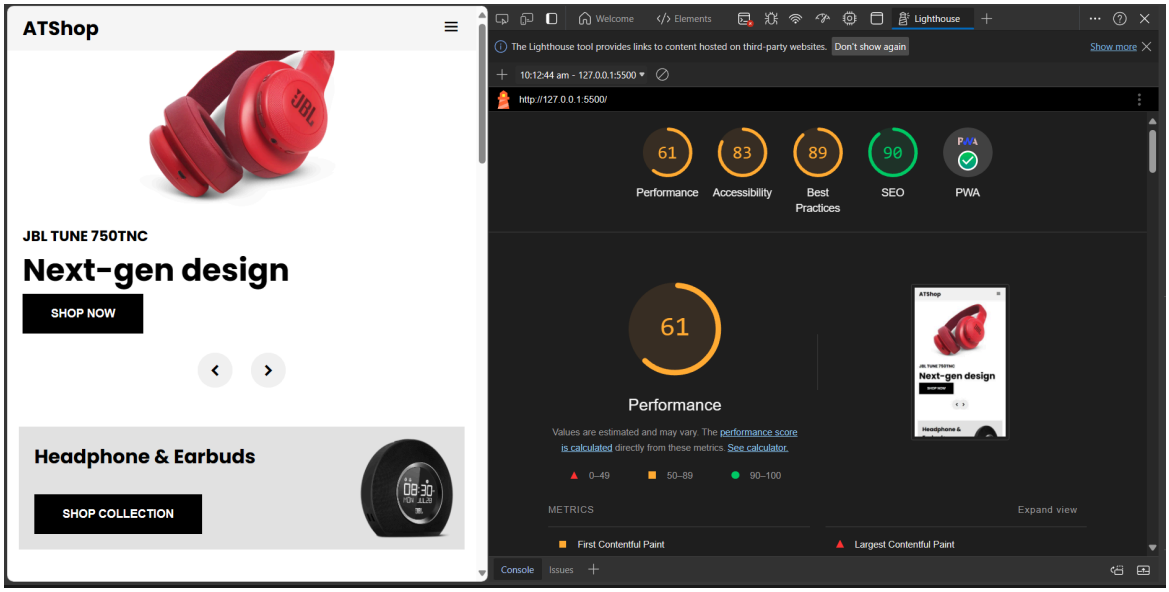
Changes in code:

```
{
  "name": "Ecom",
  "short_name": "PWA",
  "start_url": "index.html",
  "display": "standalone",
  "background_color": "#5900b3",
  "theme_color": "black",
  "scope": ".",
  "description": "This is a PWA tutorial.",
  "icons": [
    {
      "src": "images\\tuat.jpg",
      "sizes": "192x192",
      "type": "image/png",
      "purpose": "any"
    },
    {
      "src": "images\\a.jpg",
      "sizes": "512x512",
      "type": "image/png",
      "purpose": "maskable"
    }
  ]
}
```

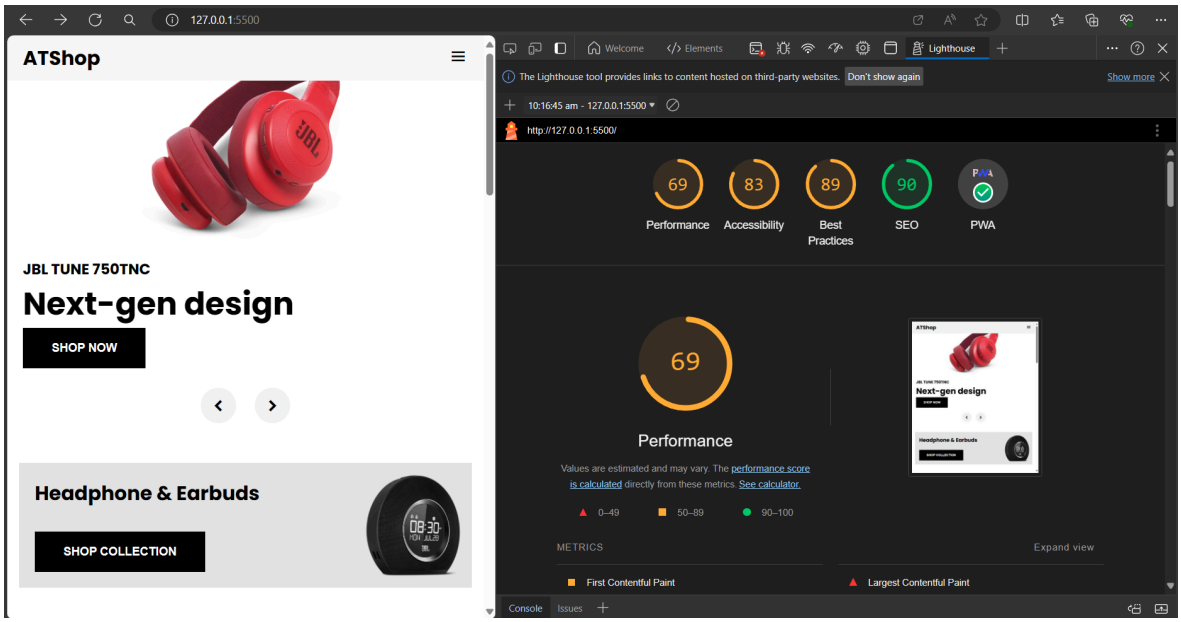
Output:

Three dots on right corner ->more tools -> developer tools -> Lighthouse -> device (mobile) -> Analyze page load

Project Title: Flutter: Twitter(X) clone / Pwa: Ashop Ecom **Roll No. 45**



Three dots on right corner ->more tools -> developer tools -> Lighthouse -> device (mobile) -> Analyze page load



Conclusion:

Successfully implemented a code and register a service worker, and completed the installation and activation process for a new service worker for the E-commerce PWA.

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.	45
Name	Shravani R. Pore
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	5

MAD PWA

Shravani R. Pore
DISA-45

Flutter Assignment - 1

Q1. Flutter overview - Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from tradition and why it has gained popularity in the developer community.

→ Key features of Flutter :

1. Single codebase for multiple platforms - Flutter allows developers to write code and deploy it on both iOS and Android platforms.
2. Hot Reload :
   Enables developers to instantly see the results of code changes they make.
3. Expressive UI :
   Developers have the flexibility to create expressive & flexible UI.
4. Integration with other tools :
   Flutter can easily integrate with other popular development tools and frameworks.

Advantages :

1. Flutter development :
   Uses single codebase for multiple platforms.
2. Consistent UI across platforms.
3. Cost efficiency.
4. Development & maintaining single codebase reduce development cost & resources.

Difference from Traditional Approach :

1. Traditional approach uses a hierarchial structure for UI components, whereas Flutter uses a widget based approach

2. Flutter compiles to native ARM code providing performance comparable to native applications.
3. 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 & adoption by major companies.

Q2. Widget Tree and Composition.
Describe the concept of widget tree in Flutter. Explain how widget composition is used to build, complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree.

→ Widget Tree:

1. It is a hierarchial structure of widgets that defines the user interface of an application.
2. Every visual element, from simple components to complex layouts is represented by a widget.
3. Widget can be categorized in two types:
   a) Stateless widget.
      It is immutable and cannot change over time. Eg: Image, text.
   b) Stateful widget
      Widget that can change its state over time Eg: buttons, forms,

Widget composition

1. It involves combining multiple simple widgets to create

more complex & compound widgets.
2. This composability is a powerful concept that allows developers to build sophisticated user interfaces by nesting widgets within each other.

Used widgets

1. Container: Box model for padding, margin & decoration.
2. Column & row: layout widget for arranging children vertically or horizontally.
3. Stack: overlapping widgets, allowing them to be layered on top of each other.
4. Listview: Scrollable list of widgets.
5. Gridview: Scrollable grid of widgets.
6. Appbar: A material design app bar typically at top of screen.
7. Appbar: App bar typically at top of screen.
8. Text field: Input field for text.
9. Button widgets: For user actions.

Q2. State management in Flutter:
Discuss the importance of state management in Flutter apps. Compare and contrast the different state management approaches available in Flutter such as setstate, Provider ~~available in Flutter such as setstate~~, Provid and riverpod. Provide scenario where each approach is suitable.

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

Setstate	Provider	Riverpod
1. Builtin flutter method.	1. External package named (provider!)	1. ~~Global state with additional features~~ 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 redundency.	4. Balances simplicity & reliability.	4. Emphasizes readability & clean syntax.
5. Testing can be challenging	5. Good testability support.	5. Enhanced testing experience.
6. Widely used & well established.	6. Widely adopted well supported.	6. Gaining popularity growing community.

Scenarios where above states are applicable:

i) setState:

→ For small to moderately complex applications.

→ When managing local state within a widget.

ii) Provider:

→ Medium to large sized applications.

→ When centralized state is needed, accessible by multiple widgets.

iii) Riverpod:
→ Large & complex applications.
→ When testability & maintainability are top priorities.

04. 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. firebase console → create a new project
2. Add firebase SDK by including dependencies in pubspec.yaml
dependencies:
firebase_core: ^version
firebase_auth: ^version
cloud_firebase: ^version
3. Run flutter pub get.
4. Initialize flutter by calling
'firebase.initializeApp()' in main.

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

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

Benefit of using firebase as backend.

1. Realtime database.
→ offers realtime NoSQL database.
2. Authentication.
→ Secure & easy to implement solution for user authentication.
3. Cloud Firestore
→ secure & easy to store scalable NoSQL, store & sync in realtime
4. Hosting.
→ Firebase hosting provides a simple & efficient way to deploy & host web applications.

Data Synchronization.:

1. Realtime database :
→ when data changes on one client it triggers events that automatically updates data on other clients.
2. Cloud Firestore:
Notifies client when data changes allowing for seamless realtime updates.
3. Authentication:
If user signin 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.	45
Name	Shravani R. Pore
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4:Understand various PWA frameworks and their requirements LO5: Design and Develop a responsive User Interface by applying PWA Design techniques LO6:Develop and Analyze PWA Features and deploy it over app hosting solutions
Grade:	4

Project Title: Flutter: Twitter(X) clone / Pwa: Ashop Ecom **Roll No. 45**

Shravani Pore
D15A - 45

PWA Assignment

Q1. Define progressive web App (PWA) and explain its significance in modern web development. Discuss the key characteristics that differentiate PWA from trad. mobile app.

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

i) Cross platform compatibility.
→ PWAs work across different devices & platforms.

ii) Responsive design
→ Built with this principle

iii) Offline functionality

iv) App like experience.

v) Fast performance

vi) Discoverability
→ Discoverable through search engines & can be ~~too~~ shared easily

Overall PWAs combine the best software of web & mobile app offering a compelling alternative to traditional native app development

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 & respond to the users device & screen size.

It uses a combination of flexible grids & layouts, images & CSS media queries to ensure that a website takes and function optimally on any device. Whether its a desktop, tablet or smartphone.

– Responsive vs fluid vs adaptive.

Feature	Responsive	~~Adaptibility~~ fluid	Adaptive.
1. layout	Flexibility, adapt to any screensize	continually adjust based on relative units.	Uses multiple fixed width layouts.
2. Development	Moderate	less effort for basic layout.	More efforts to create multiple layout.
3. Control.	Good control over overall layout.	less control over element & appearance one extreme sizes.	High control over layout for each category.
4. Suitability for PWAs.	Ideas foundation due to its versatility.	can be used but may require adjustment for optimal experience.	Lets common for PWAs due to developement overhead.

Q3. Describe the lifecycle of service workers including representation, installation & activation phase.

→ The lifecycle of service workers involves several key phases.

1. Registration :
Developers register a service worker script in the main .js file of the web application using the 'navigation servicewaker. register ( )' method. This tells the browser to install and service worker for the specified scope.

2. Installation :
During the installation phase, the browser downloads the service workes script specified in the registration step the service worker is then installed in the background. But it does not yet control any pages or intercept network request.

3. Activation :
After the service worker is installed, it goes through the activation phase. During activation, the service worker can start controlling pages & intercepting network requests.

Throughout its life cycle, a service worker operates independently of any specific web page making it a powerful tool for building offline capable, resilient web applications with enhanced performance & user experience.

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

→ IndexedDB is a low level API for client-side storage of significant amounts of structured data including files / blogs.

It is used in service workers as:

1) Persistent storage - IndexedDB provides persistent storage meaning data stored in indexedDB persists even when the browser in closed or the device is restarted.

2) Asynchronous Operation : IndexedDB operates asynchronously allowing the service workes to perform database operations without blocking the main thread.

3) Key value storage : Stores data in keyword value pairs but also supports more complex data structures like objects & allays.

4) Querying & indexing : IndexedDB supports indexing enabling efficient querying of stored data.

5) Data Transactions : IndexedDB supports transactions ensure data integrity, by providing automicity, consistency, isolation & durability (ACID) properties.

Overall, indexedDB in service workers enables enhanced performance & delives seamless user experience.