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 **Himanshu Nilesh Lohote** of **D15A** semester **VI,** have successfully completed necessary experiments in the **MAD & PWA Lab** under my supervision in **VES Institute of Technology** during the academic year **2023-2024.**

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Project Title: MAD APP : Zomato Clone **Roll No. 32**
PWA APP: E - Commerce PWA App

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

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

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers : Mrs. Kajal Jewani.

Email : kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

PO10) Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write

effective reports and design documentation, make effective presentations and give and receive clear instructions.

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

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

Program specific Outcomes

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

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

Project Title: MAD APP : Zomato Clone **Roll No. 32**
PWA APP: E - Commerce PWA App

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

MAD & PWA Lab

Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	32
Name	Himanshu Nilesh Lohote
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15

MAD & PWA LAB
EXP- 1

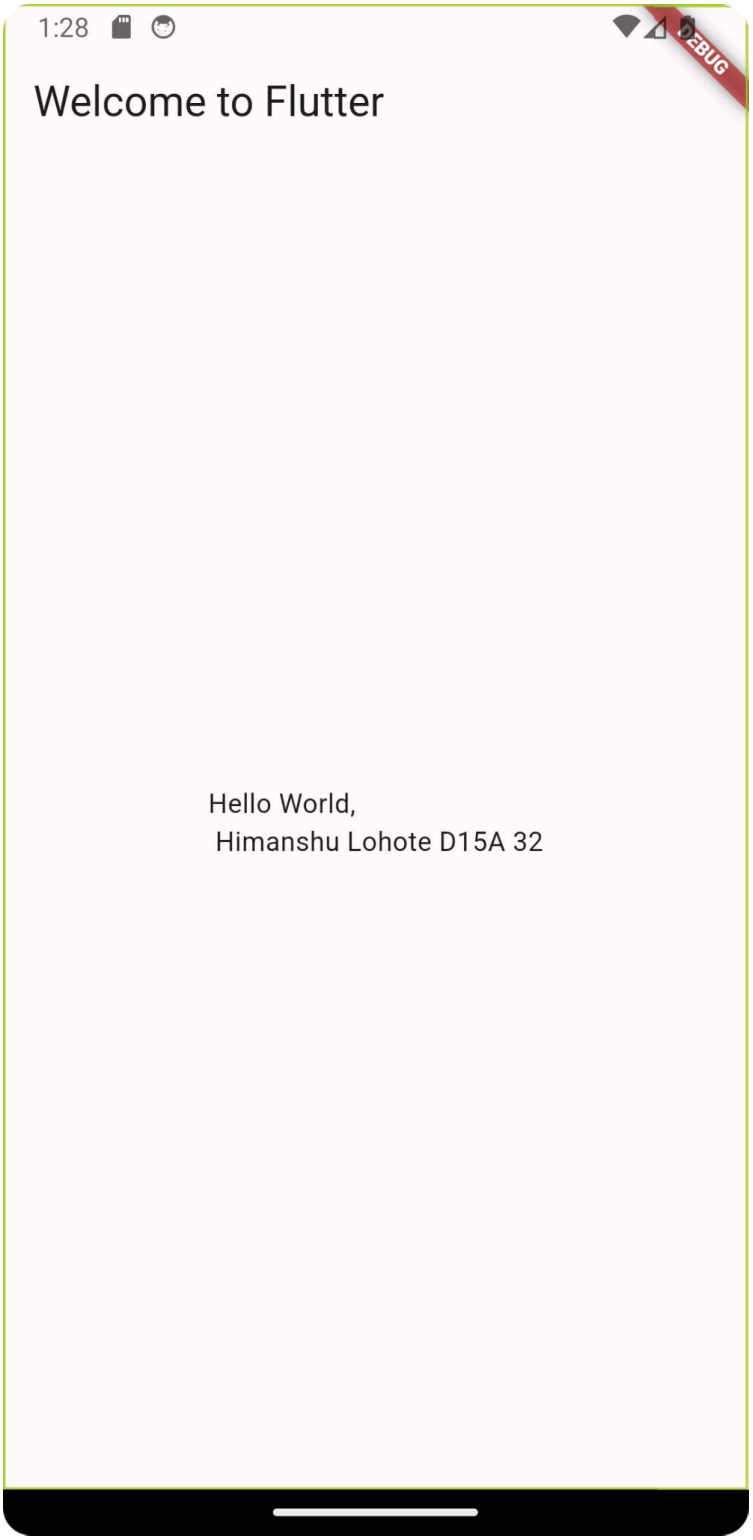
Name : Himanshu Lohote Class: D15A
Roll No. 32

Aim: Introduction of Flutter, Understanding Widget Lifecycle Events,Dart Basics, Widget Tree and Element Tree, Basics of Flutter installation, Flutter Hello World App.

Code:

```
import 'package:flutter/material.dart';
void main() {
 runApp(const MyApp());
}
class MyApp extends StatelessWidget {
 const MyApp({Key? key}) : super(key: key);
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Welcome to Flutter',
     home: Scaffold(
       appBar: AppBar(
         title: const Text('Welcome to Flutter'),
       ),
       body: const Center(
         child: Text('Hello World, \n Himanshu Lohote D15A 32'),
       ),
     ),
   );
 }
```

Output:


1:28
DEBUG
Welcome to Flutter
Hello World,
Himanshu Lohote D15A 32

MAD & PWA Lab

Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	32
Name	Himanshu Nilesh Lohote
Class	D15A
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
EXP- 2

Name : Himanshu Lohote

Class: D15A
Roll No. 32

Aim: To design Flutter UI by including common widgets

Theory:

Stateful Widget vs. Stateless Widget:

- Flutter UI can be built using either stateful or stateless widgets.
- Stateful widgets are dynamic and can change their state over time, while stateless widgets are static and don't change once built.
- In this code, the Home widget is a stateful widget because it maintains the state of selected food categories.

Material Design:

- Flutter follows the Material Design guidelines for creating visually appealing and consistent UI across platforms.
- Widgets like Scaffold, AppBar, Text, Container, etc., are part of the Material Design framework in Flutter.

Gesture Detection:

- Flutter provides the GestureDetector widget to detect gestures like taps, swipes, etc., on UI elements.
- In this code, GestureDetector is used to handle taps on food category icons and navigate to the details screen.

Navigation:

- Flutter's navigation system allows moving between different screens or routes within an app.
- The Navigator class is used for navigation management, and Navigator.push is used to navigate to a new route.

Layout Management:

- Flutter provides various layout widgets (Row, Column, SingleChildScrollView, etc.) to arrange UI elements in different ways.

- In this code, Column and Row widgets are used to vertically and horizontally align UI elements, respectively.

Styling:

- Flutter allows styling UI elements using various properties like TextStyle, BoxDecoration, etc.
- In this code, text styles are defined using the AppWidget class, which presumably contains predefined styles for text.

Image Handling:

- Flutter supports image rendering using the Image.asset widget, which loads images from the app's asset bundle.
- Images can be resized and fit using properties like height, width, and fit.

State Management:

- State management is crucial for handling dynamic data and updating UI accordingly.
- In this code, the state variables (icecream, pizza, salad, burger) are updated using the setState method to trigger UI updates when a food category is selected.

Code:

```
import 'package:flutter/material.dart';
import 'package:fooddeliveryapp/pages/details.dart';
import 'package:fooddeliveryapp/widgets/widget_support.dart';

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

  @override
  State<Home> createState() => _HomeState();
}

class _HomeState extends State<Home> {
  bool icecream = false, pizza = false, salad = false, burger = false;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SingleChildScrollView(
        child: Container(
          margin: EdgeInsets.only(top: 50.0, left: 20.0),
          child: Column(
```

```
crossAxisAlignment: CrossAxisAlignment.start,
children: [
 Row(
  mainAxisAlignment: MainAxisAlignment.spaceBetween,
  children: [
   Text("Hello Himanshu Lohote,",
     style: AppWidget.boldTextFeildStyle()),
   Container(
    margin: EdgeInsets.only(right: 20.0),
    padding: EdgeInsets.all(3),
    decoration: BoxDecoration(
      color: Colors.black,
      borderRadius: BorderRadius.circular(8)),
    child: Icon(
     Icons.shopping_cart_outlined,
     color: Colors.white,
    ),
   )
  ],
 ),
 SizedBox(
  height: 20.0,
 ),
 Text("Delicious Food", style: AppWidget.HeadlineTextFeildStyle()),
 Text("Discover and Get Great Food",
    style: AppWidget.LightTextFeildStyle()),
 SizedBox(
  height: 20.0,
 ),
 Container(
    margin: EdgeInsets.only(right: 20.0), child: showItem()),
 SizedBox(
  height: 30.0,
 ),
 SingleChildScrollView(
  scrollDirection: Axis.horizontal,
  child: Row(
   children: [
    GestureDetector(
     onTap: () {
      Navigator.push(context,
        MaterialPageRoute(builder: (context) => Details()));
     },
     child: Container(
```

```
      margin: EdgeInsets.all(4),
      child: Material(
        elevation: 5.0,
        borderRadius: BorderRadius.circular(20),
        child: Container(
          padding: EdgeInsets.all(14),
          child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Image.asset(
                  "images/salad2.png",
                  height: 150,
                  width: 150,
                  fit: BoxFit.cover,
                ),
                Text("Veggie Taco Hash",
                    style:
                        AppWidget.semiBoldTextFeildStyle()),
                SizedBox(
                  height: 5.0,
                ),
                Text("Fresh and Healthy",
                    style: AppWidget.LightTextFeildStyle()),
                SizedBox(
                  height: 5.0,
                ),
                Text(
                  "\$25",
                  style: AppWidget.semiBoldTextFeildStyle(),
                )
              ]),
        ),
      ),
    ),
  ),
  SizedBox(
    width: 15.0,
  ),
  Container(
    margin: EdgeInsets.all(4),
    child: Material(
      elevation: 5.0,
      borderRadius: BorderRadius.circular(20),
      child: Container(
```

```
            padding: EdgeInsets.all(14),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Image.asset(
                  "images/salad4.png",
                  height: 150,
                  width: 150,
                  fit: BoxFit.cover,
                ),
                Text("Mix Veg Salad",
                    style: AppWidget.semiBoldTextFeildStyle()),
                SizedBox(
                  height: 5.0,
                ),
                Text("Spicy with Onion",
                    style: AppWidget.LightTextFeildStyle()),
                SizedBox(
                  height: 5.0,
                ),
                Text(
                  "\$28",
                  style: AppWidget.semiBoldTextFeildStyle(),
                )
              ]),
        ),
      ),
    ),
  ],
 ),
),
SizedBox(
 height: 30.0,
),
Container(
 margin: EdgeInsets.only(right: 20.0),
 child: Material(
  elevation: 5.0,
  borderRadius: BorderRadius.circular(20),
  child: Container(
   padding: EdgeInsets.all(5),
   child: Row(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
```

```
            Image.asset(
              "images/salad4.png",
              height: 120,
              width: 120,
              fit: BoxFit.cover,
            ),
            SizedBox(
              width: 20.0,
            ),
            Column(
              children: [
                Container(
                    width: MediaQuery.of(context).size.width / 2,
                    child: Text(
                      "Mediterranean Chickpea Salad",
                      style: AppWidget.semiBoldTextFeildStyle(),
                    )),
                SizedBox(
                  height: 5.0,
                ),
                Container(
                    width: MediaQuery.of(context).size.width / 2,
                    child: Text(
                      "Honey goot cheese",
                      style: AppWidget.LightTextFeildStyle(),
                    )),
                SizedBox(
                  height: 5.0,
                ),
                Container(
                    width: MediaQuery.of(context).size.width / 2,
                    child: Text(
                      "\$28",
                      style: AppWidget.semiBoldTextFeildStyle(),
                    ))
              ],
            )
          ],
        ),
      ),
    ),
  ),
),
SizedBox(
  height: 30.0,
```

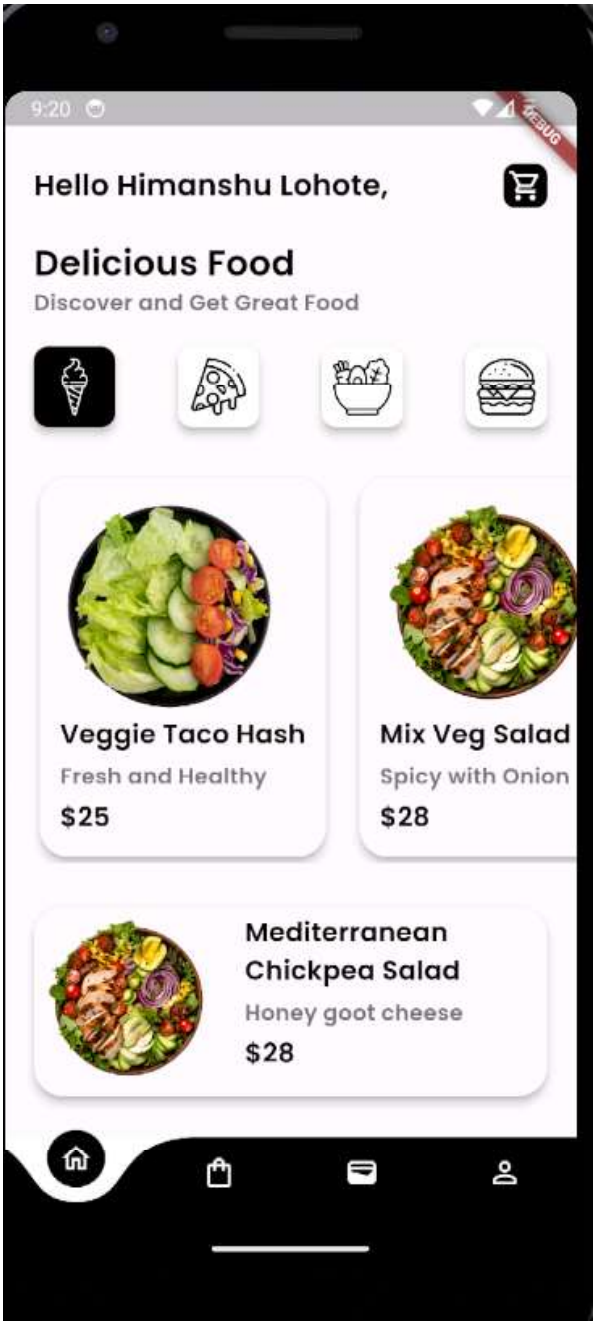
```
),
Container(
  margin: EdgeInsets.only(right: 20.0),
  child: Material(
    elevation: 5.0,
    borderRadius: BorderRadius.circular(20),
    child: Container(
      padding: EdgeInsets.all(5),
      child: Row(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Image.asset(
            "images/salad2.png",
            height: 120,
            width: 120,
            fit: BoxFit.cover,
          ),
          SizedBox(
            width: 20.0,
          ),
          Column(
            children: [
              Container(
                  width: MediaQuery.of(context).size.width / 2,
                  child: Text(
                    "Veggie Taco Hash",
                    style: AppWidget.semiBoldTextFeildStyle(),
                  )),
              SizedBox(
                height: 5.0,
              ),
              Container(
                  width: MediaQuery.of(context).size.width / 2,
                  child: Text(
                    "Honey goot cheese",
                    style: AppWidget.LightTextFeildStyle(),
                  )),
              SizedBox(
                height: 5.0,
              ),
              Container(
                  width: MediaQuery.of(context).size.width / 2,
                  child: Text(
                    "\$28",
```

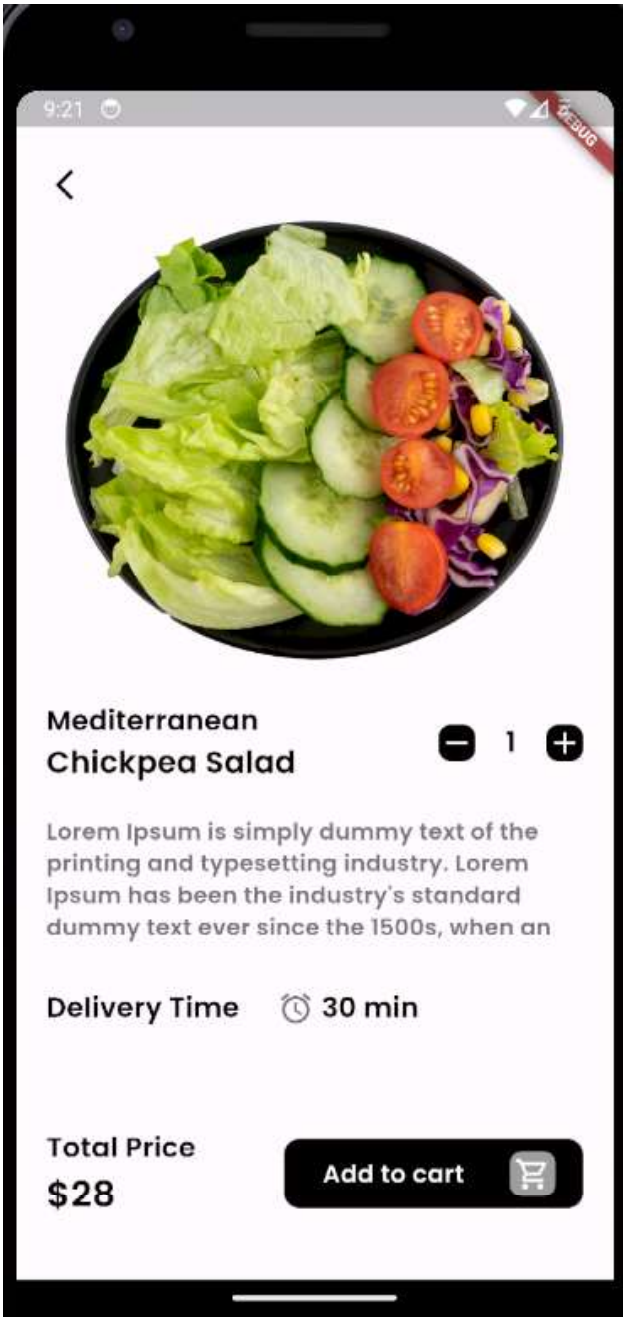
```
                        style: AppWidget.semiBoldTextFeildStyle(),
                      ))
                ],
              )
            ],
          ),
        ),
      ),
    ],
  ),
 ),
),
);
}

Widget showItem() {
 return Row(
  mainAxisAlignment: MainAxisAlignment.spaceBetween,
  children: [
   GestureDetector(
    onTap: () {
     icecream = true;
     pizza = false;
     salad = false;
     burger = false;
     setState(() {});
    },
    child: Material(
     elevation: 5.0,
     borderRadius: BorderRadius.circular(10),
     child: Container(
      decoration: BoxDecoration(
        color: icecream ? Colors.black : Colors.white,
        borderRadius: BorderRadius.circular(10)),
      padding: EdgeInsets.all(8),
      child: Image.asset(
       "images/ice-cream.png",
       height: 40,
       width: 40,
       fit: BoxFit.cover,
       color: icecream ? Colors.white : Colors.black,
      ),
     ),
```

```
    ),
  ),
  GestureDetector(
    onTap: () {
      icecream = false;
      pizza = true;
      salad = false;
      burger = false;
      setState(() {});
    },
    child: Material(
      elevation: 5.0,
      borderRadius: BorderRadius.circular(10),
      child: Container(
        decoration: BoxDecoration(
            color: pizza ? Colors.black : Colors.white,
            borderRadius: BorderRadius.circular(10)),
        padding: EdgeInsets.all(8),
        child: Image.asset(
          "images/pizza.png",
          height: 40,
          width: 40,
          fit: BoxFit.cover,
          color: pizza ? Colors.white : Colors.black,
        ),
      ),
    ),
  ),
  GestureDetector(
    onTap: () {
      icecream = false;
      pizza = false;
      salad = true;
      burger = false;
      setState(() {});
    },
    child: Material(
      elevation: 5.0,
      borderRadius: BorderRadius.circular(10),
      child: Container(
        decoration: BoxDecoration(
            color: salad ? Colors.black : Colors.white,
            borderRadius: BorderRadius.circular(10)),
        padding: EdgeInsets.all(8),
```

```
            child: Image.asset(
              "images/salad.png",
              height: 40,
              width: 40,
              fit: BoxFit.cover,
              color: salad ? Colors.white : Colors.black,
            ),
          ),
        ),
      ),
      GestureDetector(
        onTap: () {
          icecream = false;
          pizza = false;
          salad = false;
          burger = true;
          setState(() {});
        },
        child: Material(
          elevation: 5.0,
          borderRadius: BorderRadius.circular(10),
          child: Container(
            decoration: BoxDecoration(
                color: burger ? Colors.black : Colors.white,
                borderRadius: BorderRadius.circular(10)),
            padding: EdgeInsets.all(8),
            child: Image.asset(
              "images/burger.png",
              height: 40,
              width: 40,
              fit: BoxFit.cover,
              color: burger ? Colors.white : Colors.black,
            ),
          ),
        ),
      ),
    ],
  );
 }
}
```

Output:


9:20
DEBUG
Hello Himanshu Lohote,
Delicious Food
Discover and Get Great Food
Veggie Taco Hash
Fresh and Healthy
$25
Mix Veg Salad
Spicy with Onion
$28
Mediterranean
Chickpea Salad
Honey goot cheese
$28


9:21
DEBUG
Mediterranean
Chickpea Salad
1
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an
Delivery Time
30 min
Total Price
$28
Add to cart

MAD & PWA Lab

Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	32
Name	Himanshu Nilesh Lohote
Class	D15A
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
EXP- 3

Name : Himanshu Lohote

Class: D15A
Roll No. 32

Aim: To include icons, images, fonts in Flutter app

Theory:

In Flutter, you can include icons, images, and custom fonts to enhance the visual appeal and functionality of your app. Here's a brief overview of how you can integrate these assets into your Flutter application:

1. Icons:
   Flutter provides a wide range of icons through its `Icons` class, which includes material design icons. To use these icons, you can simply import the `material_icons` package in your `pubspec.yaml` file. For example:
   ```yaml
   dependencies:
     flutter:
       sdk: flutter
     cupertino_icons: ^1.0.0
   ```

   Then, in your Dart code, you can use these icons directly:
   ```dart
   Icon(Icons.favorite);
   ```

   Additionally, Flutter allows you to use custom icons. You can add custom icon files (like SVG or PNG) to your project and reference them using the `Image.asset` widget.

2. Images:

To include images in your Flutter app, you can place the image files (e.g., PNG, JPEG) in the project's `assets` directory. Then, you need to specify these assets in your `pubspec.yaml` file:

```yaml
flutter:
  assets:
    - assets/image1.png
    - assets/image2.png
```

After adding the assets, you can use them in your Flutter code with the `Image.asset` widget:

```dart
Image.asset('assets/image1.png');
```

Flutter also supports loading images from the network using the `Image.network` widget.

3. Fonts:

To include custom fonts in your Flutter app, you first need to add the font files (e.g., TTF, OTF) to your project's `fonts` directory. Then, specify these fonts in your `pubspec.yaml` file:

```yaml
flutter:
  fonts:
    - family: CustomFont
      fonts:
        - asset: fonts/custom_font.ttf
```

After adding the fonts, you can use them in your Flutter application by specifying the font family in the `style` property of text widgets:

```dart
```

```
  Text(
    'Custom Font Example',
    style: TextStyle(
      fontFamily: 'CustomFont',
      fontSize: 16.0,
    ),
  );
```

Make sure to adjust the font family name to match the one specified in the `pubspec.yaml` file.

By following these steps, you can easily integrate icons, images, and custom fonts into your Flutter app, enhancing its visual appearance and user experience.

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('MAD PWA Lab - Exp 3
\nHimanshu Lohote D15A 32'),
```

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

Output:

MAD & PWA Lab

Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	32
Name	Himanshu Nilesh Lohote
Class	D15A
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
EXP- 4

Name : Himanshu Lohote **Class: D15A**
Roll No. 32

Aim: To create an interactive Form using form widget

Code:

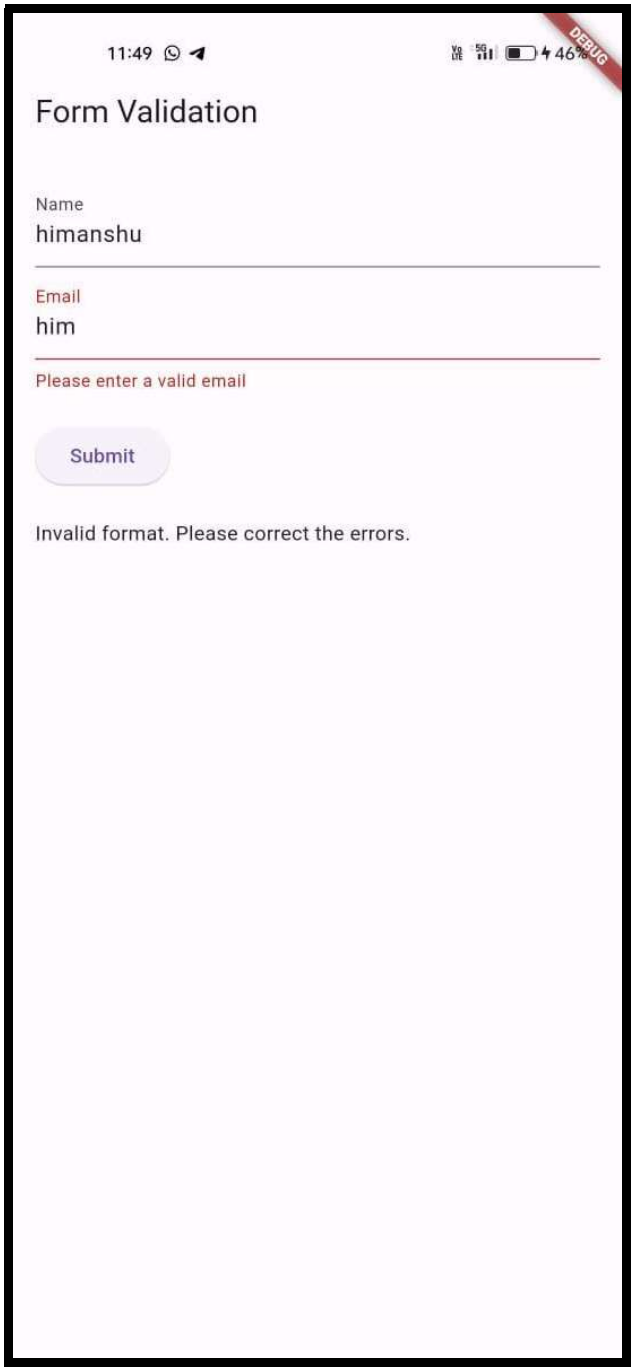
```dart
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Form Validation',
      home: MyForm(),
    );
  }
}

class MyForm extends StatefulWidget {
  @override
  _MyFormState createState() => _MyFormState();
}

class _MyFormState extends State<MyForm> {
```

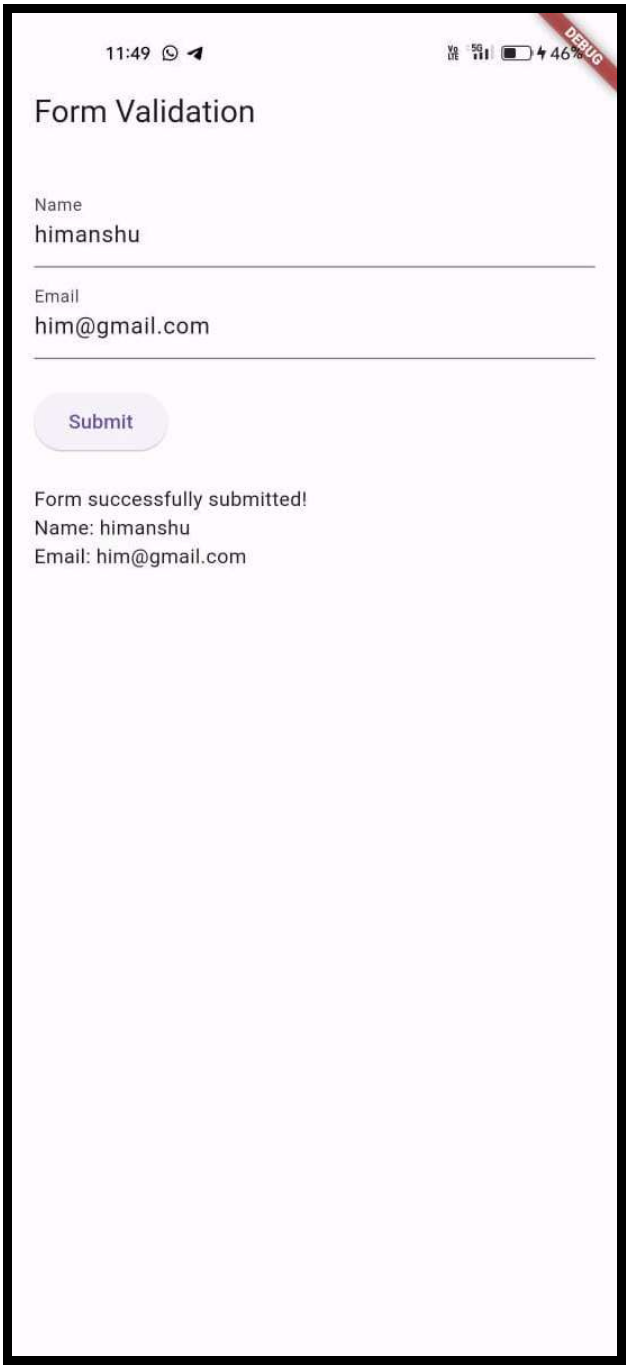
```
  final _formKey = GlobalKey<FormState>();
  String _name = '';
  String _email = '';
  String _message = '';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Form Validation'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Form(
          key: _formKey,
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: <Widget>[
              TextFormField(
                decoration: InputDecoration(labelText:
'Name'),
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Please enter your name';
                  }
                  return null;
                },
                onSaved: (value) {
                  _name = value!;
                },
              ),
              TextFormField(
```

```
                decoration: InputDecoration(labelText:
'Email'),
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Please enter your email';
                  }
                  // This is a basic email validation
                  if (!value.contains('@')) {
                    return 'Please enter a valid email';
                  }
                  return null;
                },
                onSaved: (value) {
                  _email = value!;
                },
              ),
              SizedBox(height: 20),
              ElevatedButton(
                onPressed: () {
                  _submitForm();
                },
                child: Text('Submit'),
              ),
              SizedBox(height: 20),
              Text(_message),
            ],
          ),
        ),
      ),
    );
  }
```

```
  void _submitForm() {
    if (_formKey.currentState!.validate()) {
      _formKey.currentState!.save();
      // You can handle form submission here
      setState(() {
        _message = 'Form successfully submitted!\nName:
$_name\nEmail: $_email';
      });
    } else {
      setState(() {
        _message = 'Invalid format. Please correct the
errors.';
      });
    }
  }
}
```

Output:


11:49
DEBUG
Form Validation
Name
himanshu
Email
him
Please enter a valid email
Submit
Invalid format. Please correct the errors.


11:49
DEBUG
Form Validation
Name
himanshu
Email
him@gmail.com
Submit
Form successfully submitted!
Name: himanshu
Email: him@gmail.com

MAD & PWA Lab

Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	32
Name	Himanshu Nilesh Lohote
Class	D15A
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
EXP- 5

Name : Himanshu Lohote **Class: D15A**
Roll No. 32

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

Code:

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Navigation and Gestures',
      initialRoute: '/',
      routes: {
        '/': (context) => FirstScreen(),
        '/second': (context) => SecondScreen(),
      },
    );
  }
}

class FirstScreen extends StatelessWidget {
  @override
```

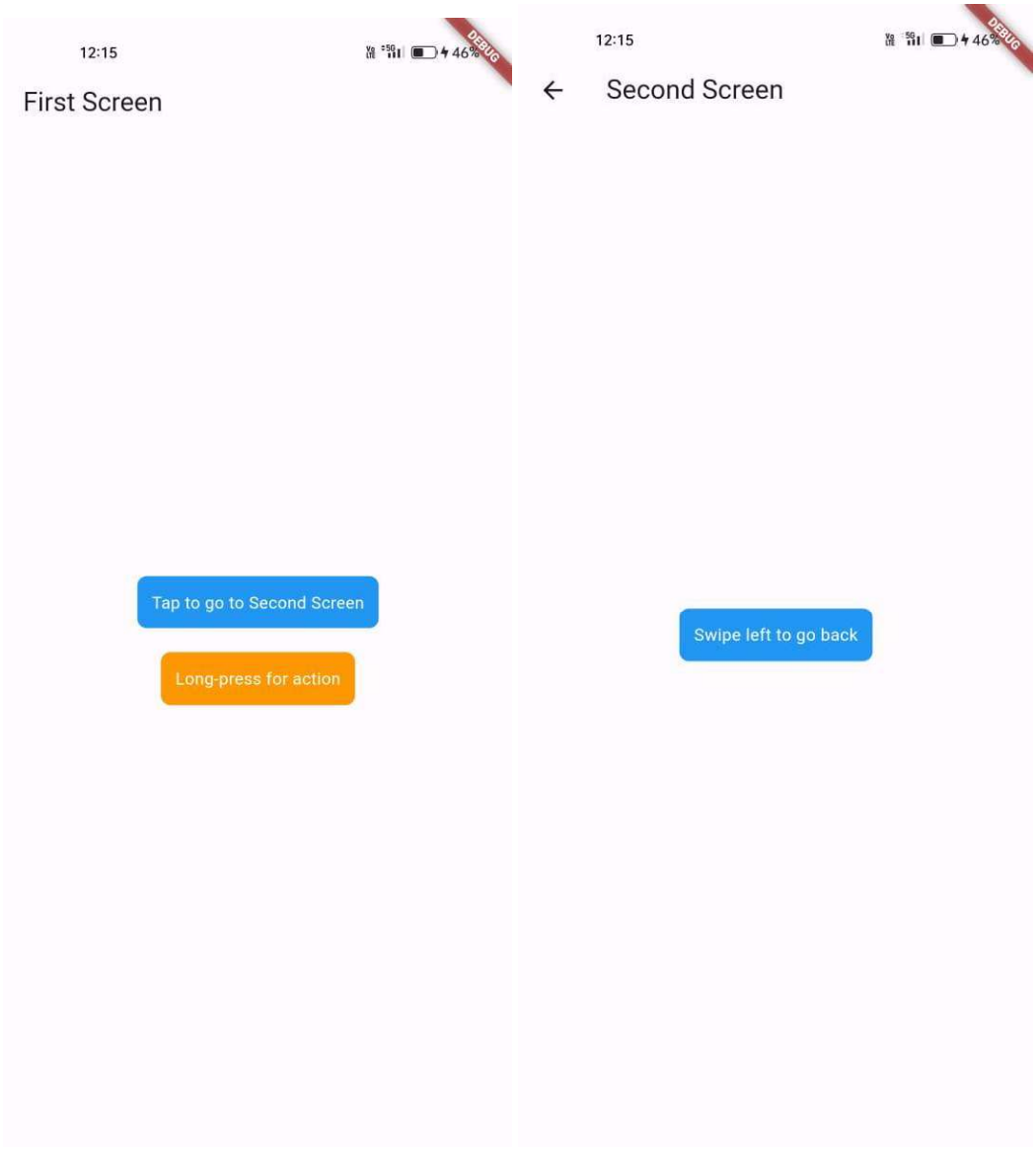
```
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('First Screen')),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            GestureDetector(
              onTap: () {
                Navigator.pushNamed(context, '/second');
              },
              child: Container(
                padding: EdgeInsets.all(12.0),
                decoration: BoxDecoration(
                  color: Colors.blue,
                  borderRadius:
BorderRadius.circular(8.0),
                ),
                child: Text(
                  'Tap to go to Second Screen',
                  style: TextStyle(color: Colors.white),
                ),
              ),
            ),
            SizedBox(height: 20),
            GestureDetector(
              onLongPress: () {
                // Perform any action here

ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(content: Text('Long press
detected')),
```

```
                );
              },
              child: Container(
                padding: EdgeInsets.all(12.0),
                decoration: BoxDecoration(
                  color: Colors.orange,
                  borderRadius:
BorderRadius.circular(8.0),
                ),
                child: Text(
                  'Long-press for action',
                  style: TextStyle(color: Colors.white),
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}

class SecondScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Second Screen')),
      body: Center(
        child: GestureDetector(
          onHorizontalDragEnd: (details) {
            // Detect swipe gesture to navigate back
            if (details.primaryVelocity! < 0) {
```

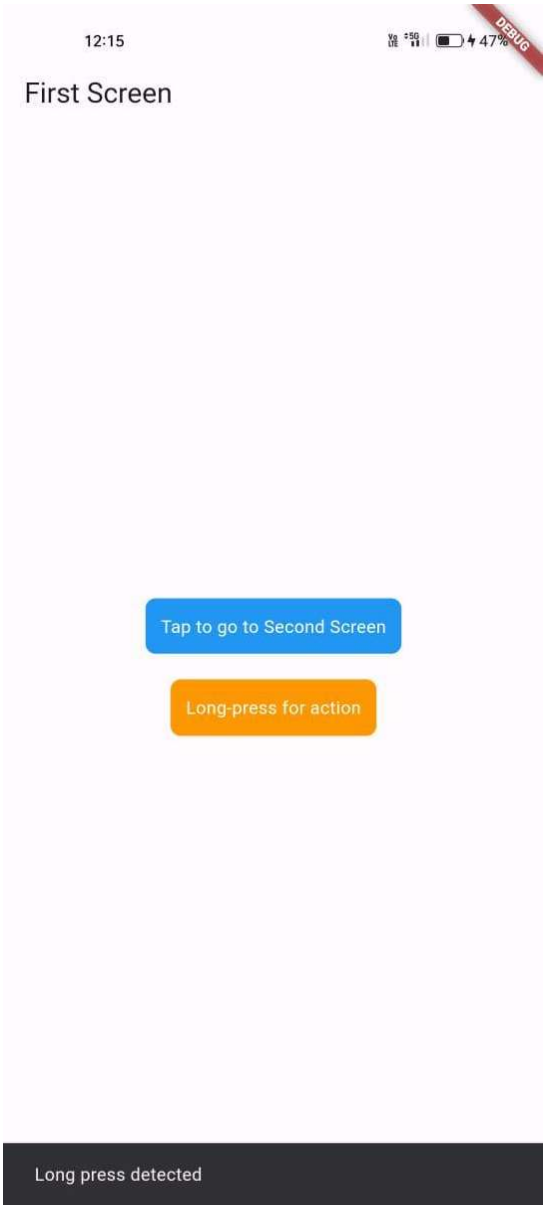
```
              Navigator.pop(context);
            }
          },
          child: Container(
            padding: EdgeInsets.all(12.0),
            decoration: BoxDecoration(
              color: Colors.blue,
              borderRadius: BorderRadius.circular(8.0),
            ),
            child: Text(
              'Swipe left to go back',
              style: TextStyle(color: Colors.white),
            ),
          ),
        ),
      ),
    );
  }
}
```

Output:

Navigation, Tap and Swipe Gesture:



Long Press Gesture:



MAD & PWA Lab

Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	32
Name	Himanshu Nilesh Lohote
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

MAD & PWA LAB
EXP- 6

Name: Himanshu Lohote **Class: D15A**
Roll No. 32

Set Up Firebase with Flutter for iOS and Android Apps

Firebase is a great backend solution for anyone that wants to use authentication, databases, cloud functions, ads, and countless other features within an app.

In this article, you will create a Firebase project for iOS and Android platforms using Flutter.

Prerequisites

To complete this tutorial, you will need:

- A Google account to use Firebase.
- Developing for iOS will require XCode.
- To download and install Flutter.
- To download and install Android Studio and Visual Studio Code.
- It is recommended to install plugins for your code editor:
  - `Flutter` and `Dart` plugins installed for Android Studio.
  - `Flutter` extension installed for Visual Studio Code.

Creating a New Flutter Project

This tutorial will require the creation of an example Flutter app.

Once you have your environment set up for Flutter, you can run the following to create a new application:

```
flutter create fooddeliveryapp
```

Navigate to the new project directory:

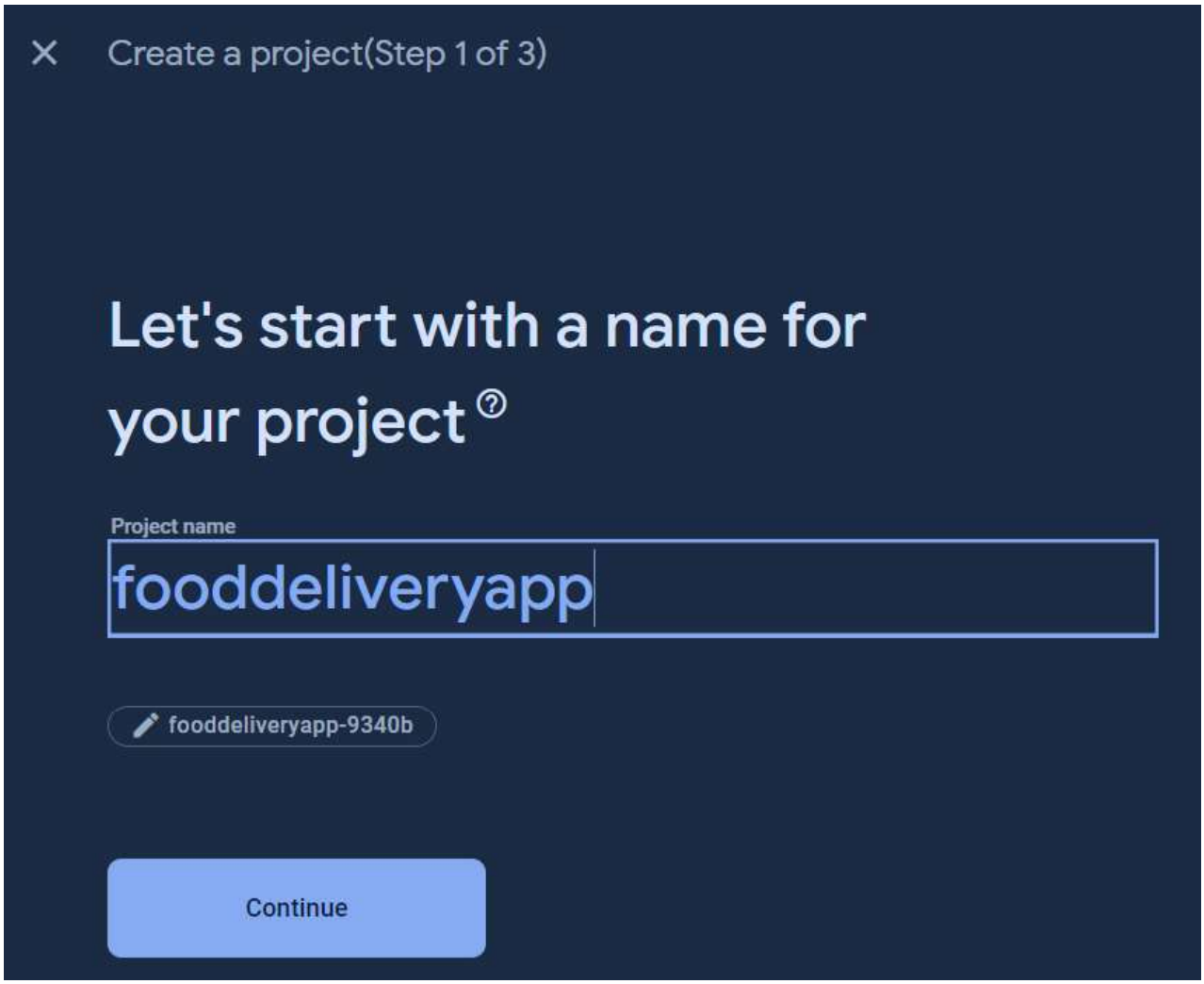
```
cd fooddeliveryapp
```

Using `flutter create` will produce a demo application that will display the number of times a button is clicked.

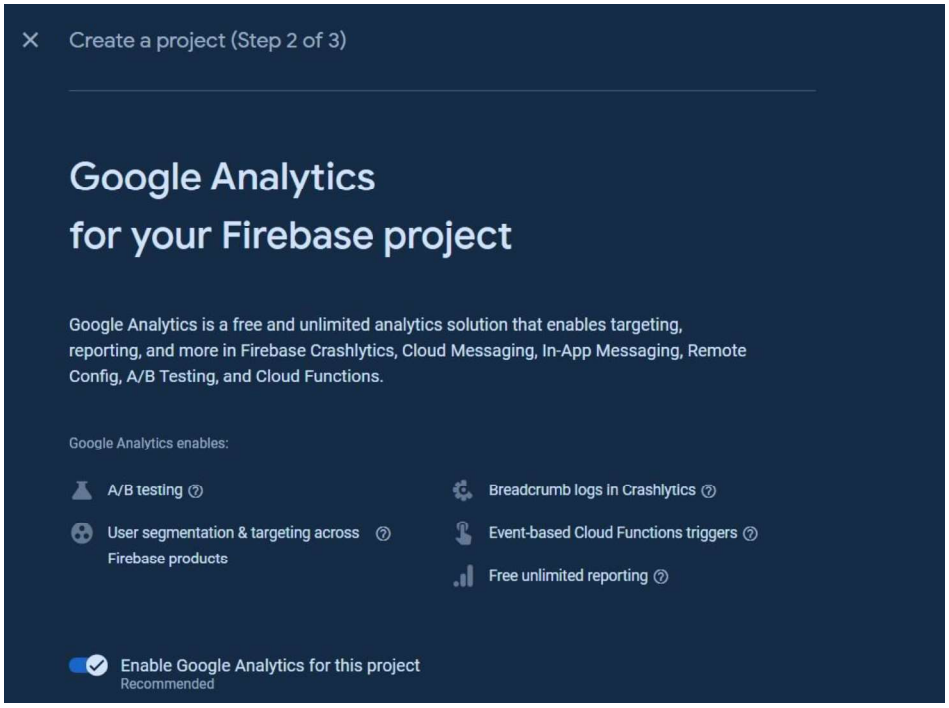
Now that we've got a Flutter project up and running, we can add Firebase.

Creating a New Firebase Project

First, log in with your Google account to manage your Firebase projects. From within the Firebase dashboard, select the Create new project button and give it a name:

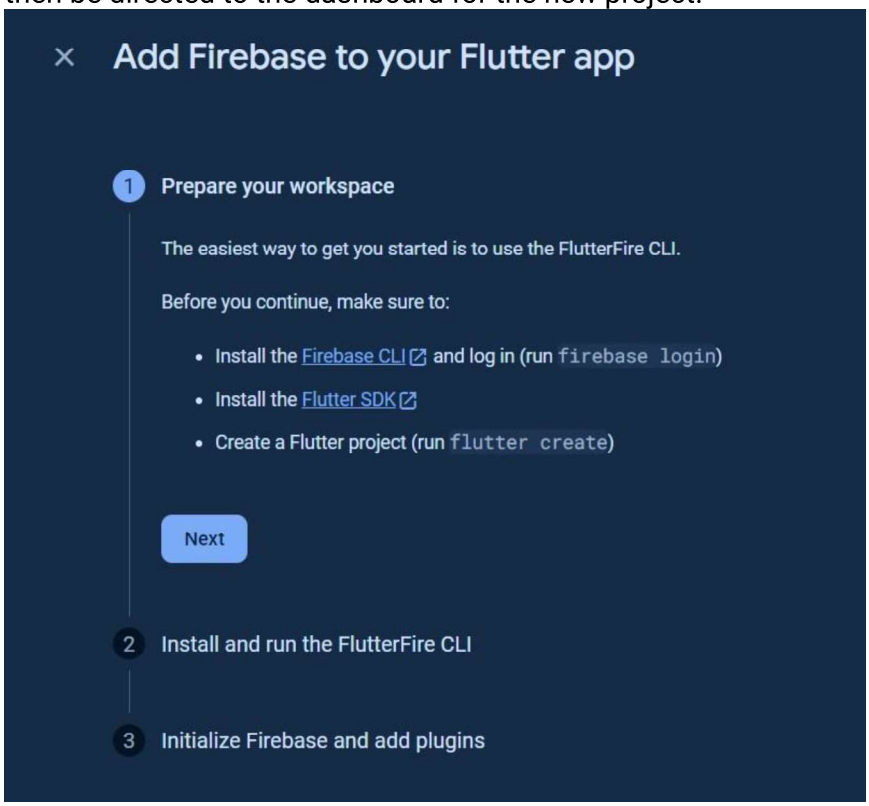


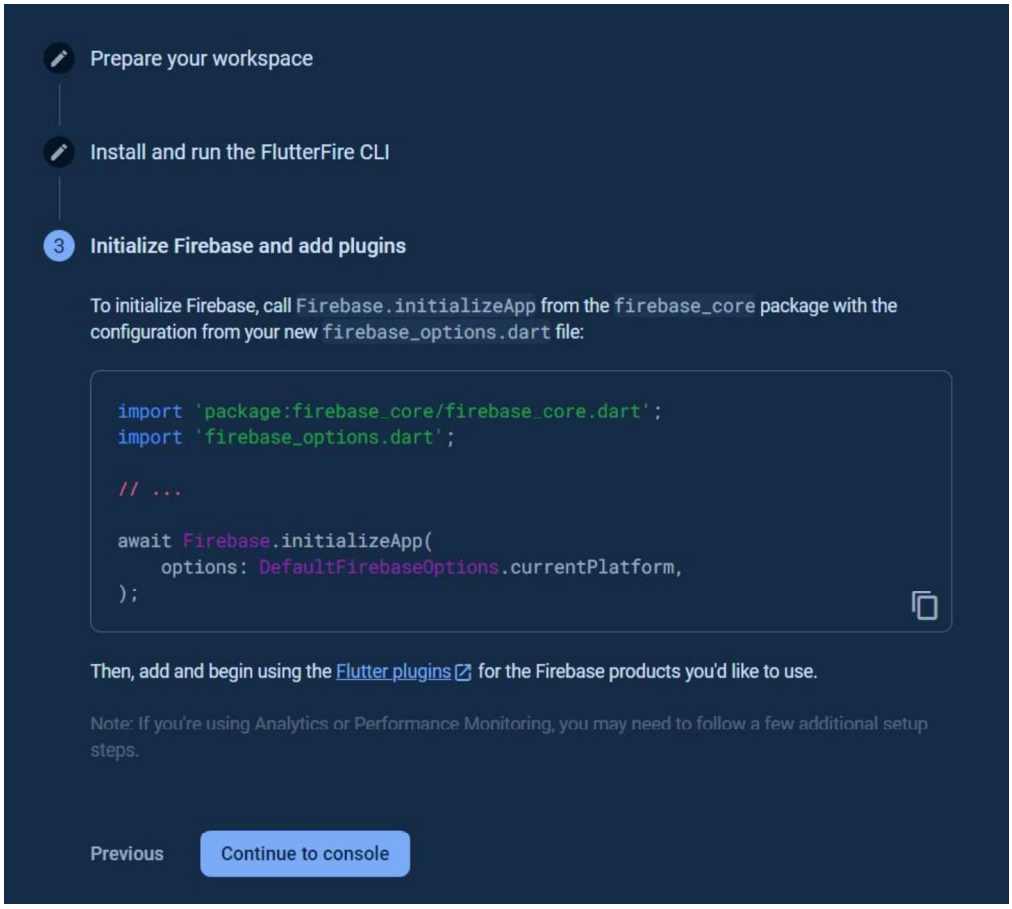
Next, we're given the option to enable Google Analytics. This tutorial will not require Google Analytics, but you can also choose to add it to your project.



if you choose to use Google Analytics, you will need to review and accept the terms and conditions prior to project creation.

After pressing Continue, your project will be created and resources will be provisioned. You will then be directed to the dashboard for the new project.





Adding Android support:

Registering the App

The most important thing here is to match up the Android package name that you choose here with the one inside of our application.

The structure consists of at least two segments. A common pattern is to use a domain name, a company name, and the application name:

`com.example.fooddeliveryapp`

Once you've decided on a name, open `android/app/build.gradle` in your code editor and update the `applicationId` to match the Android package name:

android/app/build.gradle

```
...
defaultConfig {
    // TODO: Specify your own unique Application ID
(https://developer.android.com/studio/build/application-id.html).
    applicationId 'com.example. fooddeliveryapp
'
    ...
```

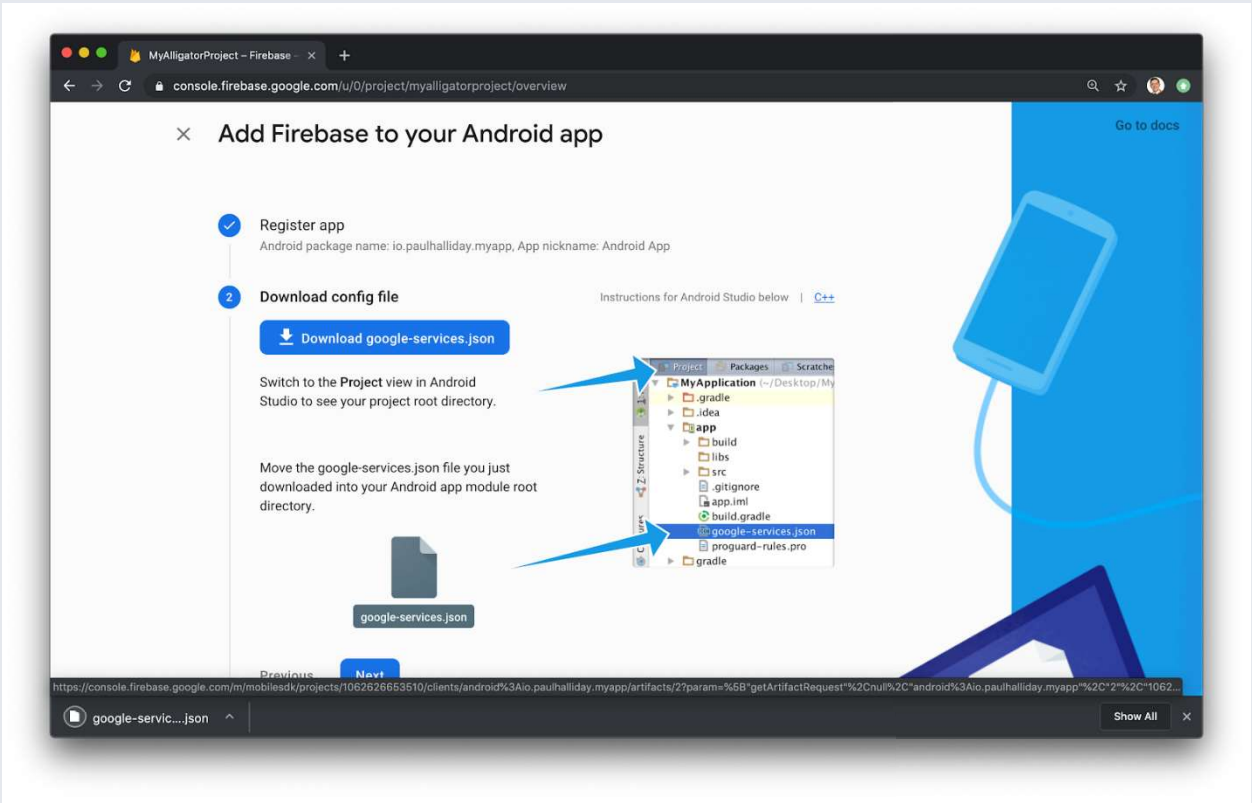
```
}
...
```

You can skip the app nickname and debug signing keys at this stage. Select Register app to continue.

Downloading the Config File

The next step is to add the Firebase configuration file into our Flutter project. This is important as it contains the API keys and other critical information for Firebase to use.

Select Download `google-services.json` from this page:



Next, move the `google-services.json` file to the `android/app` directory within the Flutter project.

Adding the Firebase SDK

We'll now need to update our Gradle configuration to include the Google Services plugin.

Open `android/build.gradle` in your code editor and modify it to include the following:

android/buiild.gradle

```
buildscript {
  repositories {
    // Check that you have the following line (if not, add it):
```

```
    google()  // Google's Maven repository
  }
  dependencies {
    ...
    // Add this line
    classpath 'com.google.gms:google-services:4.3.6'
  }
}

allprojects {
  ...
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
    ...
  }
}
```

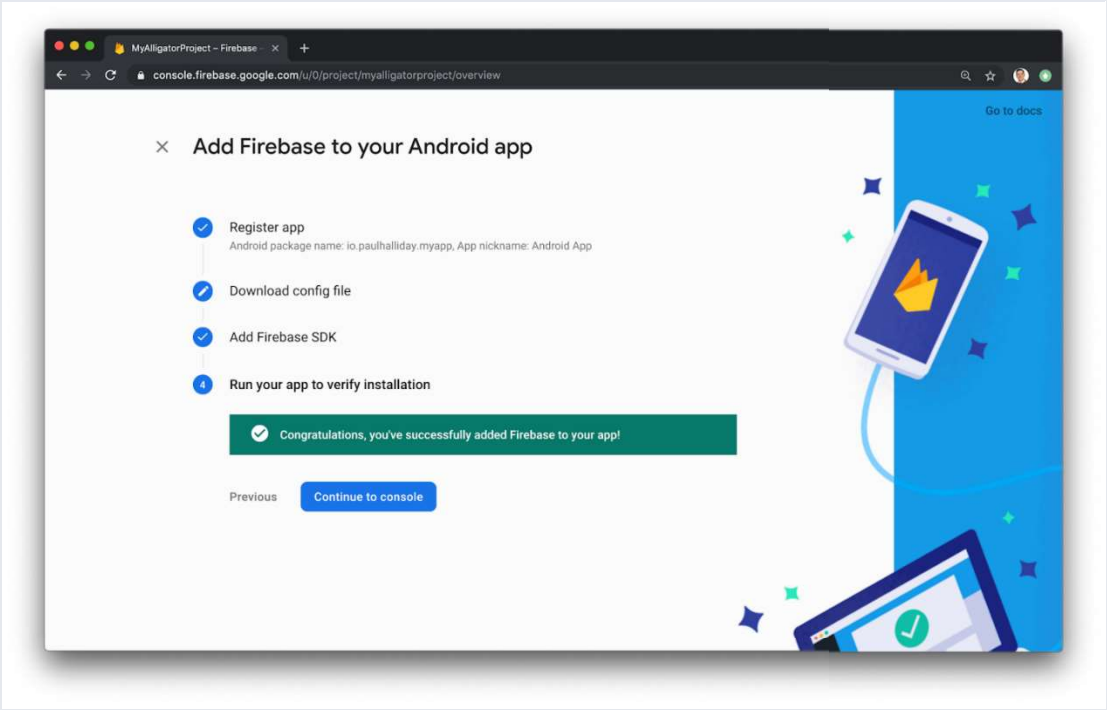
Finally, update the app level file at `android/app/build.gradle` to include the following:

android/app/build.gradle

```
apply plugin: 'com.android.application'
// Add this line
apply plugin: 'com.google.gms.google-services'

dependencies {
  // Import the Firebase BoM
  implementation platform('com.google.firebase:firebase-bom:28.0.0')

  // Add the dependencies for any other desired Firebase products
  // https://firebase.google.com/docs/android/setup#available-libraries
}
```

With this update, we're essentially applying the Google Services plugin as well as looking at how other Flutter Firebase plugins can be activated such as Analytics.

From here, run your application on an Android device or simulator. If everything has worked correctly, you should get the following message in the dashboard:



My Dependencies used in Food Delivery app using Firebase in Pubspec.yaml

```
# versions available, run  flutt
dependencies:
  flutter:
    sdk: flutter
  curved_navigation_bar: ^1.0.3
  firebase_core: ^2.25.4
  firebase_auth: ^4.17.4
  cloud_firestore: ^4.15.4
  flutter_stripe: ^10.0.0
  random_string: ^2.3.1
  shared_preferences: ^2.2.2
  image_picker: ^1.0.7
  firebase_storage: ^11.6.5


  # The following adds the Cuper
  # Use with the CupertinoIcons
  cupertino_icons: ^1.0.2
  http: ^1.2.0

dev_dependencies:
  flutter_test:
    sdk: flutter
```

Conclusion

In this experiment, we learned how to set up and ready our Flutter applications to be used with Firebase.

Flutter has official support for Firebase with the FlutterFire set of libraries.

MAD & PWA Lab

Journal

Experiment No.	07
Experiment Title.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.
Roll No.	32
Name	Himanshu Nilesh Lohote
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	15

MAD & PWA LAB
EXP- 7

Name: Himanshu Lohote

Class: D15A
Roll No. 32

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

Theory: -

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

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

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

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

3. Faster Services

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

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

5. Updated Real-Time Data Access
Another plus point of PWAs is that these apps get updated on their own. They do not demand the end-users to go to the App Store or other such platforms to download the update and wait until installed.
In this app type, the web app developers can push the live update from the server, which reaches the apps residing on the user's devices automatically. Therefore, it is easier for the mobile app developer to provide the best of the updated functionalities and services to the end-users without forcing them to update their app.
6. Discoverable
PWAs reside in web browsers. This implies higher chances of optimizing them as per the Search Engine Optimization (SEO) criteria and improving the Google rankings like that in websites and other web apps.

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

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

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

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

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

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

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

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

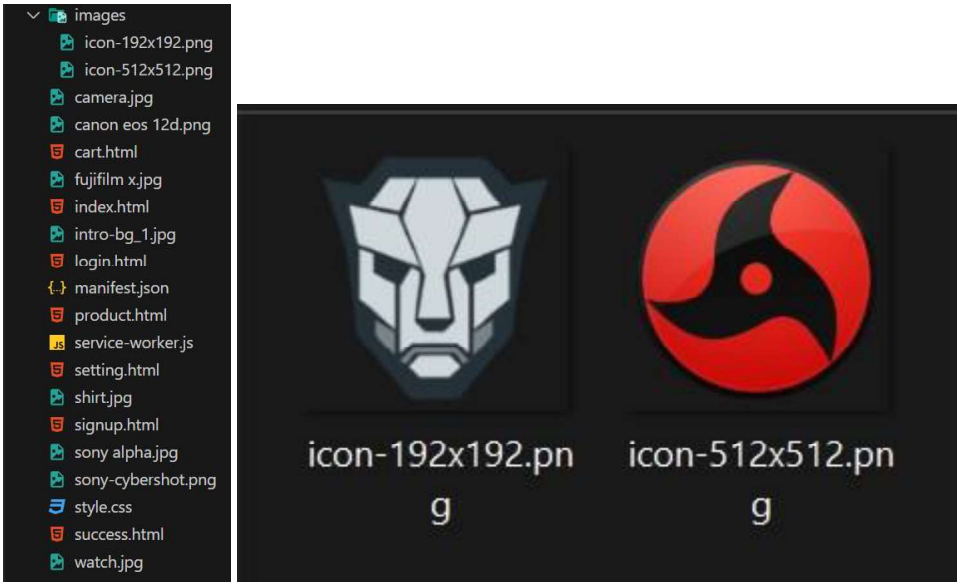
Reactivable — Make it easy to reactivate the application thanks to capabilities such as web notifications.
Installable — They allow the user to “save” the apps that he considers most useful with the corresponding icon on the screen of his mobile terminal (home screen) without having to face all the steps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

Linkable — Easily shared via URL without complex installations.

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

Weaknesses refer to:
IOS support from version 11.3 onwards;
Greater use of the device battery;
Not all devices support the full range of PWA features (same speech for iOS and Android operating systems);
It is not possible to establish a strong re-engagement for iOS users (URL scheme, standard web notifications);
Support for offline execution is however limited;
Lack of presence on the stores (there is no possibility to acquire traffic from that channel);
There is no “body” of control (like the stores) and an approval process;
Limited access to some hardware components of the devices;
Little flexibility regarding “special” content for users (eg loyalty programs, loyalty, etc.).

Folder Structure and icon size



Index.html

```
<!DOCTYPE html>
<html>

<head>
    <meta name="apple-mobile-web-app-status-bar" content="#aa7700">
    <meta name="theme-color" content="black">
    <link rel="manifest" href="manifest.json">
    <script src="service-worker.js"></script>
    <title>
        Index
    </title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.cs
s">

    <!--jQuery library-->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></
script>

    <!--Latest compiled and minified JavaScript-->
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js">
</script>
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="style.css">
</head>

<body>

    <nav class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle"
data-toggle="collapse" data-target="#mynavbar">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="index.html">Lifestyle
Store</a>

            </div>
            <div class="collapse navbar-collapse" id="mynavbar">
                <ul class="nav navbar-nav navbar-right">
                    <li>
                        <a href="signup.html">
                            <span class="glyphicon glyphicon-user" />
Sign-Up </a>
                    </li>
                    <li>
                        <a href="login.html">
                            <span class="glyphicon glyphicon-log-in" />
Login </a>
                    </li>
                </ul>
            </div>

        </div>

    </nav>
    <div class="banner-image">
        <div class="container">
            <div class="banner-content" style="margin-left :25%">
```

```
                <h1>We sell lifestyle</h1>
                <p>Flat 40% OFF on premium brands</p> <br>
                <a href="product.html" class="btn btn-danger btn-lg
active">Shop Now</a>
            </div>
        </div>
    </div>
    <footer>
        <div class="container">
            <p style="text-align:center;">Copyright © Lifestyle Store. All
Rights Reserved and Contact Us: +91 90000
                00000 </p>
        </div>
    </footer>
    <script>
        // Add event listener to execute code when page loads
        window.addEventListener('load', () => {
            // Call registerSW function when page loads
            registerSW();
        });

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

```
</body>

</html>
```

Manifest.json

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

Service-worker.json

```
// service-worker.js

const CACHE_NAME = 'my-ecommerce-app-cache-v1';
const urlsToCache = [
  '/',
```

```
  'cart.html',
  'index.html',
  'product.html',
  'shop.html',
  'style.css',
  'success.html',
  'service-worker.js',
  'manifest.json'

  // Add more files to cache as needed
];

self.addEventListener('install', function(event) {
    event.waitUntil(
      caches.open(CACHE_NAME)
        .then(function(cache) {
          console.log('Opened cache');
          return cache.addAll(urlsToCache)
            .catch(function(error) {
              console.error('Cache.addAll error:', error);
            });
        })
    );
  });

self.addEventListener('activate', function(event) {
  // Perform activation steps
  event.waitUntil(
    caches.keys().then(function(cacheNames) {
      return Promise.all(
        cacheNames.map(function(cacheName) {
          if (cacheName !== CACHE_NAME) {
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
});
```

Product.html

```
<!DOCTYPE html>
<html>

    <head>
    <title>
        product
    </title>
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.cs
s" >

        <!--jQuery library-->
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></
script>

        <!--Latest compiled and minified JavaScript-->
        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js">
</script>
        <meta name="viewport" content="width=device-width,
initial-scale=1">
        <link rel = "stylesheet" href = "style.css">
    </head>
        <body>
            <nav class = "navbar navbar-inverse navbar-fixed-top">
                    <div class ="container">
                        <div class ="navbar-header">
                            <button type="button" class ="navbar-toggle"
data-toggle="collapse" data-target="#mynavbar">
                                <span class="icon-bar"></span>
                                <span class="icon-bar"></span>
                                <span class="icon-bar"></span>
                            </button>
                            <a class="navbar-brand"
href="index.html">Lifestyle Store</a>

                        </div>
```

```
                    <div class="collapse navbar-collapse"
id="mynavbar">
                        <ul class="nav navbar-nav navbar-right">
                            <li>
                                <a href="cart.html">
                                    <span class="glyphicon
glyphicon-shopping-cart">  Cart </span> </a>
                            </li>
                            <li>
                                <a href="setting.html">
                                    <span class="glyphicon
glyphicon-user">  Setting</span>  </a>
                            </li>
                            <li>
                                <a href="index.html">
                                    <span class="glyphicon
glyphicon-log-out">    Logout</span></a>
                            </li>
                        </ul>
                    </div>

                </div>

            </nav>

            <div class =" container" style="margin-top: 5%;">

            <div class ="jumbotron">

            <h1> Welcome  to our Lifestyle Store!  </h1>
            <p>We have the best cameras, watches and shirts for you.
No need to hunt
            around, we have all in one place.  </p>
            </div>

            <div class="row text-center">
            <div class="  col-md-3 col-sm-6  thumbnail " >
            <img src="canon eos 12d.png">
            <div class="caption">
            <h2>Canon EOS 12D</h2>
```

```
<p>Full Zoom Quality</p>

</div>
<div class=" btn btn-primary  btn-block btn-md"> 29000
</div>
</div>


<div class="  col-md-3 col-sm-6  thumbnail " >
<img src="sony-cybershot.png">
<div class="caption">
<h2>Sony CyberShot</h2>
<p>48MP Camera</p>

</div>
<div class=" btn btn-primary  btn-block btn-md"> 19000
</div>
</div>


<div class="  col-md-3 col-sm-6  thumbnail " >
<img src="sony alpha.jpg">
<div class="caption">
<h2>Sony Alpha</h2>
<p>Flagship Camera</p>

</div>
<div class=" btn btn-primary  btn-block btn-md"> 50000
</div>
</div>

<div class="  col-md-3 col-sm-6  thumbnail " >
<img src="fujifilm x.jpg">
<div class="caption">
<h2>Fujifilm X</h2>
<p>Best off all</p>
</div>

<div class=" btn btn-primary  btn-block btn-md"> 29000
</div>
```

```
</div>

</div>


<div class="row text-center">

<div class="  col-md-3 col-sm-6  thumbnail " >
<img src="watch.jpg">
<div class="caption">
<h2>ROLEX</h2>
<p>Best Watch</p>
</div>
<div class=" btn btn-primary  btn-block btn-md"> 200000
</div>
</div>


<div class="  col-md-3 col-sm-6  thumbnail " >
<img src="watch.jpg">
<div class="caption">
<h2>Sonata</h2>
<p>Cheapest Watch</p>
</div>
<div class=" btn btn-primary  btn-block btn-md"> 2000
</div>
</div>


<div class="  col-md-3 col-sm-6  thumbnail " >
<img src="watch.jpg">
     <div class="caption">
    <h2>TIMEX by Titan</h2>
    <p>Premium watch</p>
    </div>
    <div class=" btn btn-primary  btn-block btn-md"> 20000
</div>

</div>
```

```
                <div class="  col-md-3 col-sm-6  thumbnail " >
                <img src="watch.jpg">
                <div class="caption">
                <h2>JACOB&CO. WATCHES </h2>
                <p>Most Expensive</p>
                </div>
                <div class=" btn btn-primary  btn-block btn-md"> 2000000
                </div>

                </div>
                </div>



                <div class="row text-center">

                <div class="  col-md-3 col-sm-6  thumbnail " >
                <img src="shirt.jpg">
                  <div class="caption">
                <h2>Raymond</h2>
                <p>Blue Shirt</p>
                </div>
                <div class=" btn btn-primary  btn-block btn-md">2000
</div>

                </div>

                <div class="  col-md-3 col-sm-6  thumbnail " >
                <img src="shirt.jpg">
                 <div class="caption">
                <h2>Levi</h2>
                <p>Formals</p>
                </div>
                <div class=" btn btn-primary  btn-block btn-md">3000</div>
                </div>

                <div class="  col-md-3 col-sm-6  thumbnail " >
                <img src="shirt.jpg">
                  <div class="caption">
```

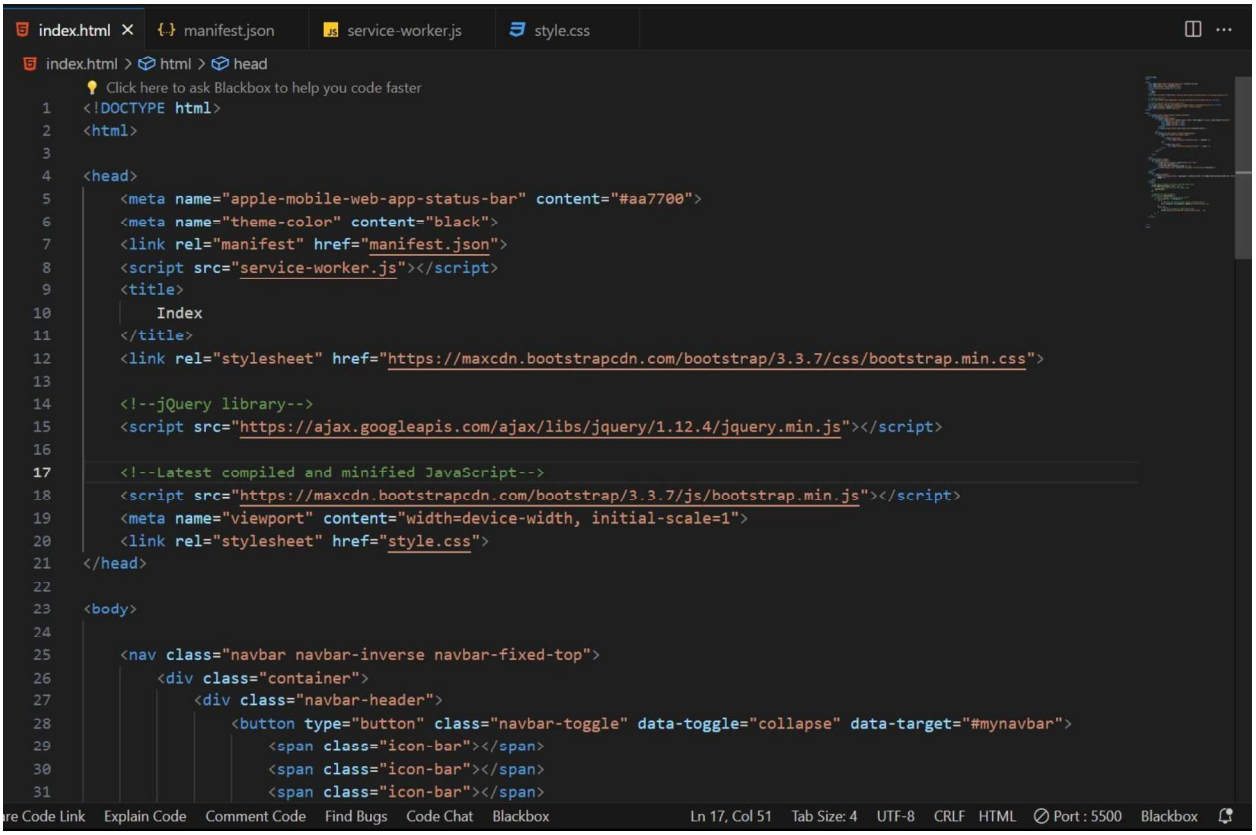
```
                <h2>Denim</h2>
                <p>Blue Shirt</p>
                </div>
                <div class=" btn btn-primary  btn-block btn-md">SHIRTS
</div>
                </div>

                <div class="  col-md-3 col-sm-6  thumbnail " >
                <img src="shirt.jpg">
                  <div class="caption">
                <h2>Cambridge</h2>
                <p>Original Shirts and Formals</p>
                </div>
                <div class=" btn btn-primary  btn-block btn-md">SHIRTS
</div>
                </div>
                </div>
                </div>

                <footer style="margin-top: 5%; margin-bottom:.5%;">
                                <div class="container" >
                                    <p
style="text-align:center;">Copyright © Lifestyle Store. All Rights
Reserved and Contact Us: +91 90000 00000 </p>
                                </div>
                    </footer>



        </body>
    </html>
```

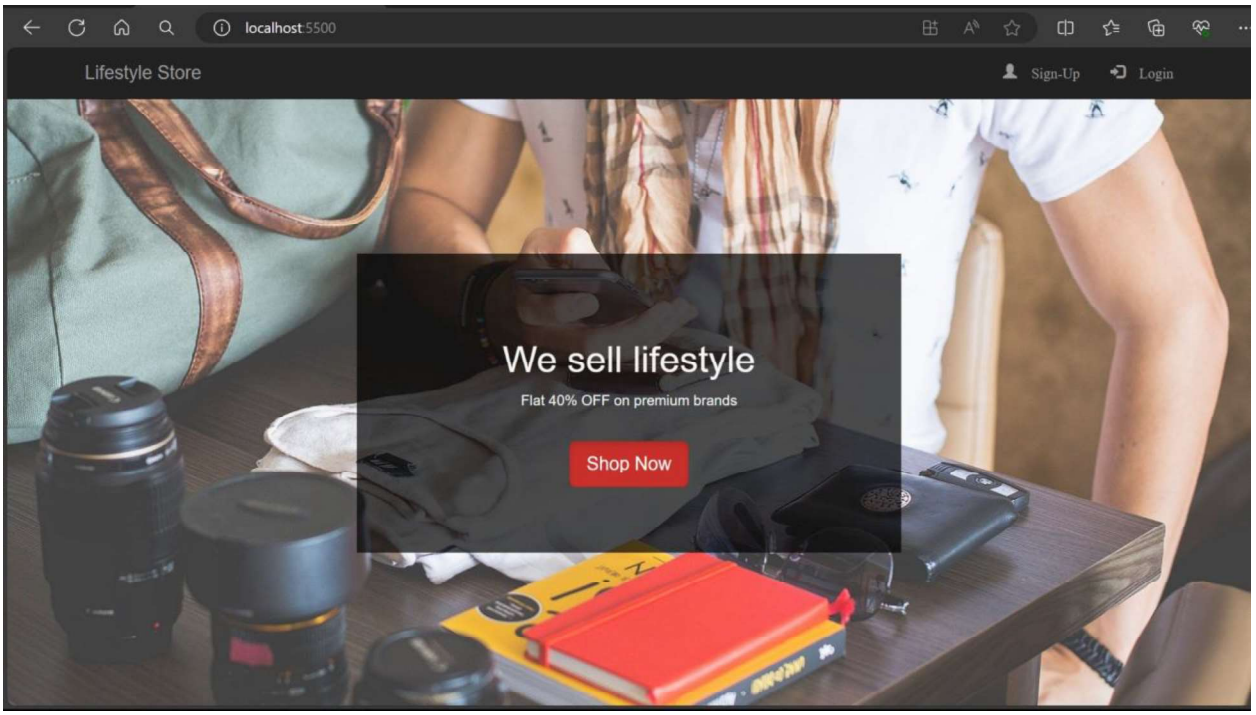
Style.css

```
.banner-image
{padding-top: 75px;
 padding-bottom: 50px;
text-align: center;
color: #f8f8f8;
 background: url(intro-bg_1.jpg) no-repeat center center;
```

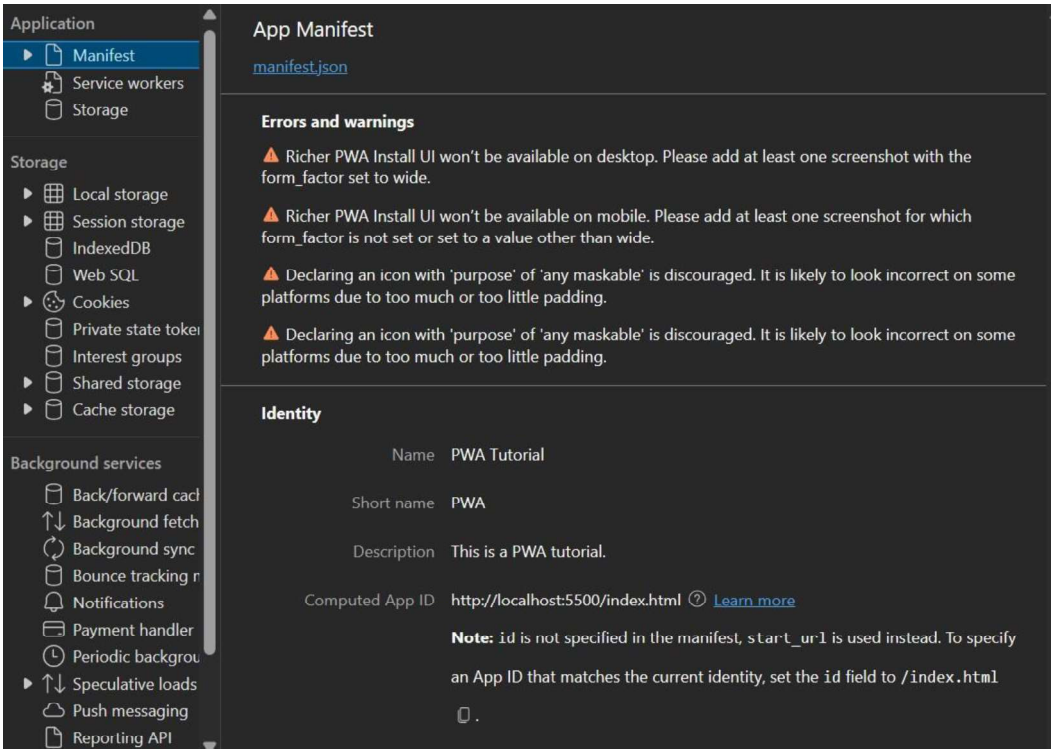
```
background-size: cover;
}
.banner-content{
  position: relative;

 padding-top: 6%;
 padding-bottom: 6%;

 margin-top: 12%;
 margin-bottom: 12%;
 background-color: rgba(0, 0, 0, 0.7);
 width: 50%;
 text-align:center;
}
footer
{
padding: 10px 0;
background-color: #101010;
color:#9d9d9d;
bottom: 0;
width: 100%;
}
.container{
  width:90%;
 margin:auto;
 overflow:hidden;

}
```

Starting the Server:



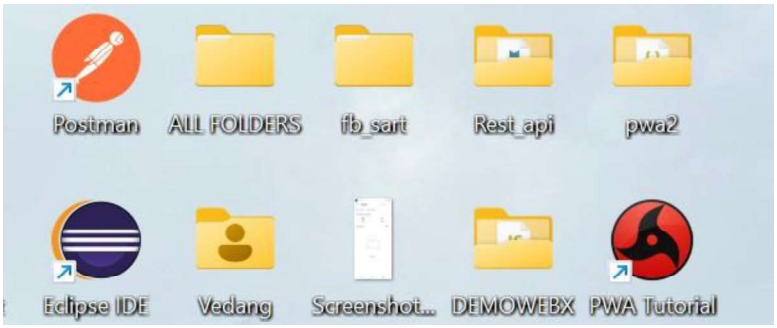


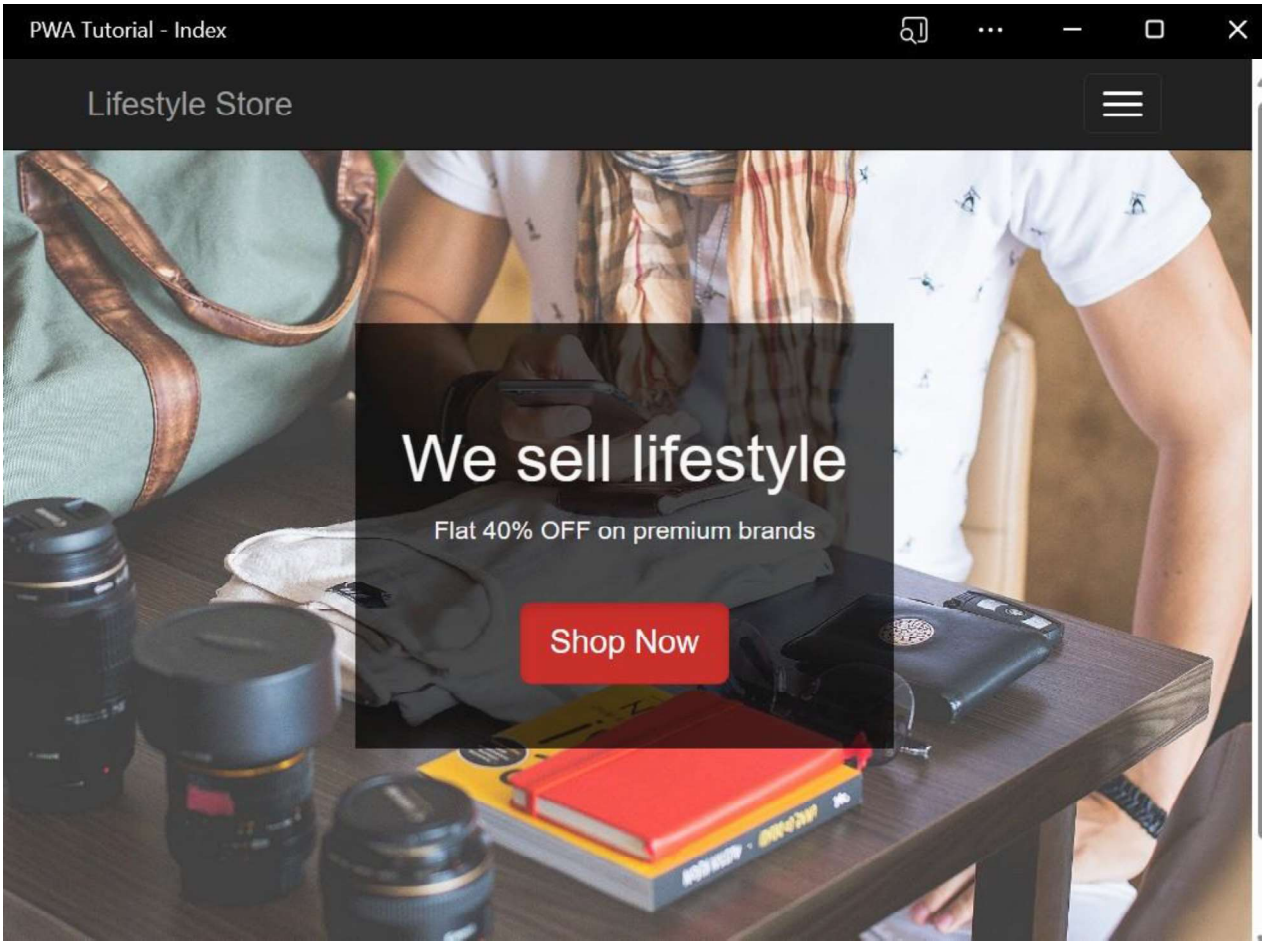
Now go to developer options -> Application->Manifest



Now to install PWA , click on Three dots(...) -> Apps -> Install PWA

Now On the Desktop





Conclusion : Hence We wrote meta data of our Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”. And it is currently added Successfully on the Desktop

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.	32
Name	Himanshu Nilesh Lohote
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

MAD & PWA LAB
EXP- 8

Name: Himanshu Lohote Class: D15A
Roll No. 32

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.

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.

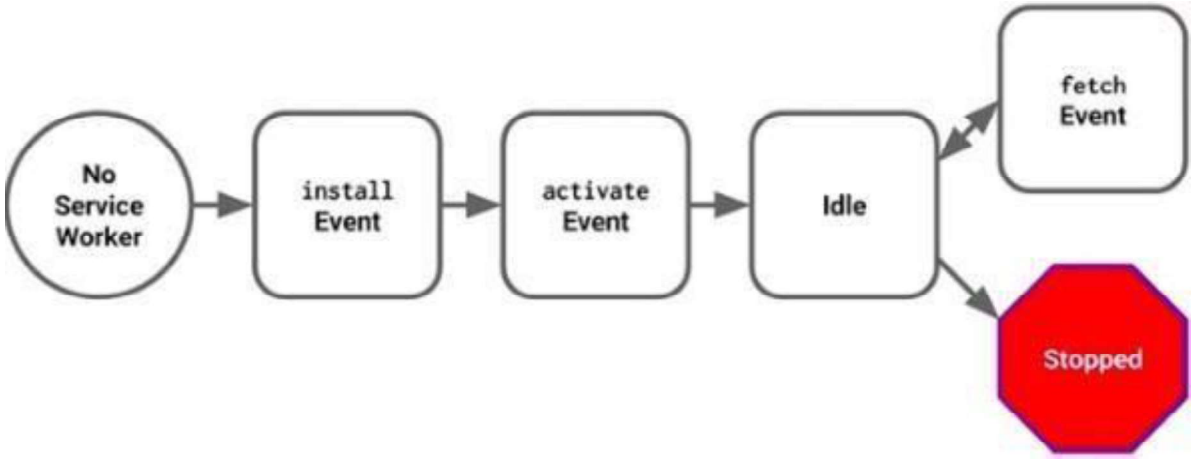
- You can **Continue**
  Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

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 in service-worker.js

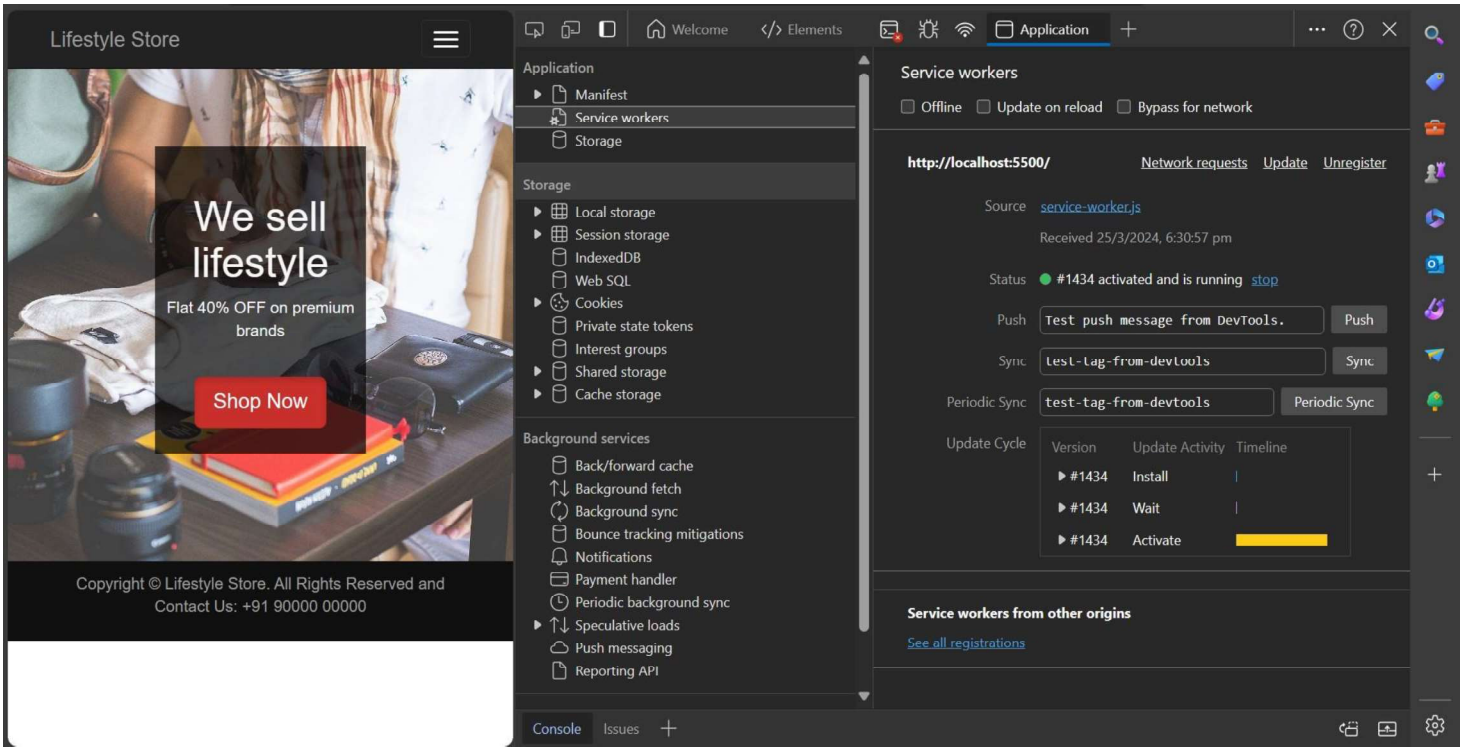
```
self.addEventListener('install', function(event) {
    event.waitUntil(
      caches.open(CACHE_NAME)
        .then(function(cache) {
          console.log('Opened cache');
          return cache.addAll(urlsToCache)
            .catch(function(error) {
              console.error('Cache.addAll error:', error);
            });
        })
    );
  });

self.addEventListener('activate', function(event) {
  // Perform activation steps
  event.waitUntil(
    caches.keys().then(function(cacheNames) {
      return Promise.all(
        cacheNames.map(function(cacheName) {
          if (cacheName !== CACHE_NAME) {
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
});
```

Code in index.html

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

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

Output:



Conclusion: Hence We Successfully Registered our Service Worker on the Progressive Web App and it is activated as well as running

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.	32
Name	Himanshu Nilesh Lohote
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

MAD & PWA LAB
EXP- 9

Name: Himanshu Lohote

Class: D15A
Roll No. 32

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

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

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

But if this process fails, we check whether the request has been cached before or not. If a cache exists, it is returned to the page, but if not, this is up to you. You can return dummy content or information messages to the page.

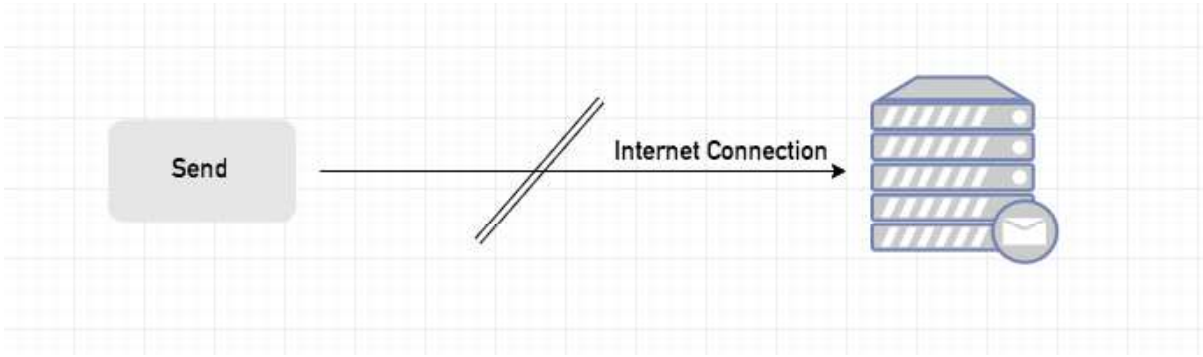
```
self.addEventListener("fetch", function (event) {
    const req = event.request;
    const url = new URL(req.url);

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

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

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

Sync Event

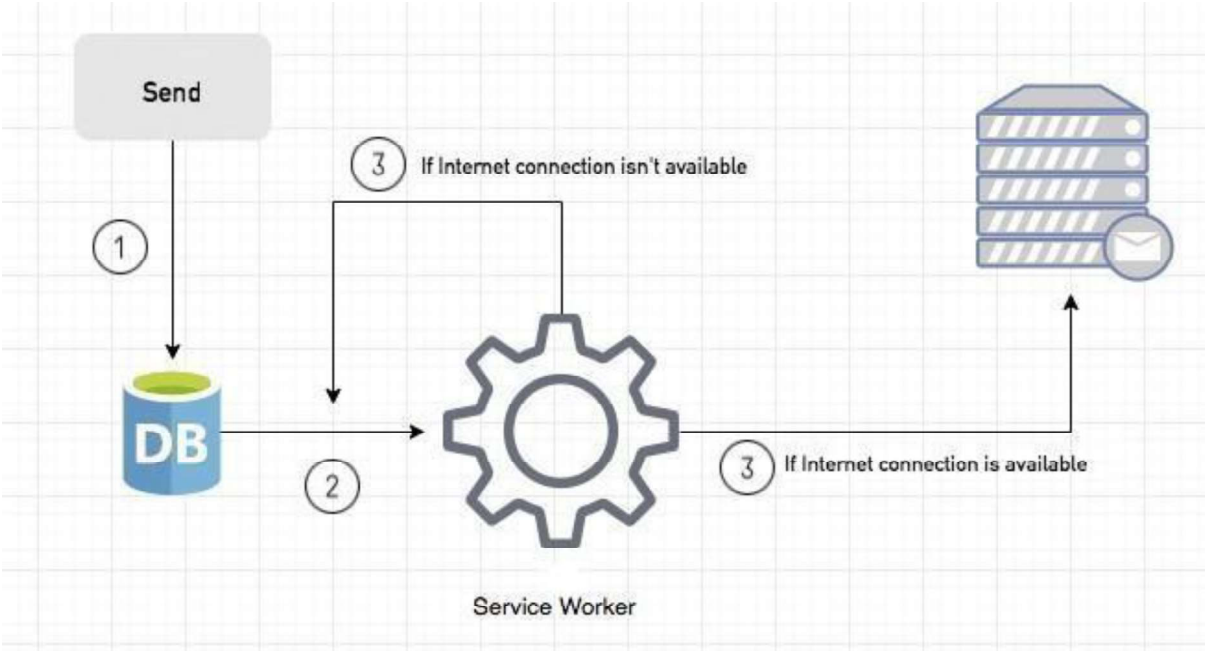
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

Here is a job for the Background Sync.

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



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



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

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

Event Listener for Background Sync Registration

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

Event Listener for sw.js

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

Push Event

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

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

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

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

You can use Application Tab from Chrome Developer Tools for testing push notification
Aim: To implement Service worker events like fetch, sync and push for E-commerce PWA.

Theory:
Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

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

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

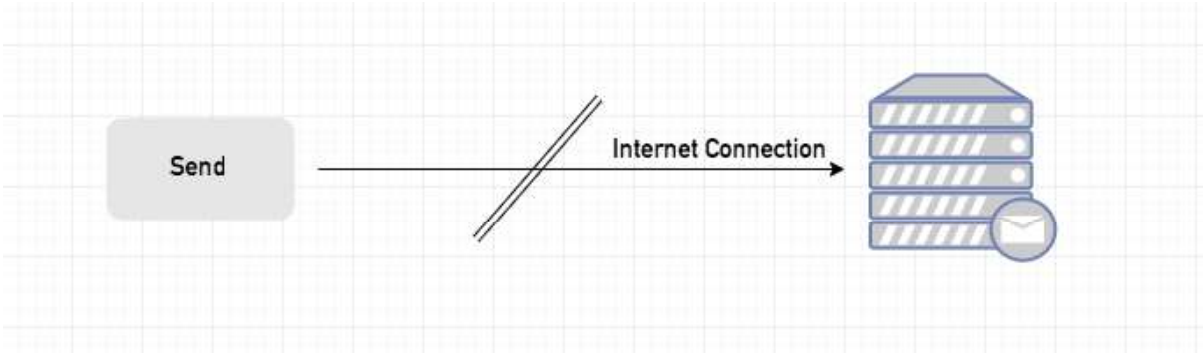
```
self.addEventListener("fetch", function (event) {
    const req = event.request;
    const url = new URL(req.url);
```

Sync Event

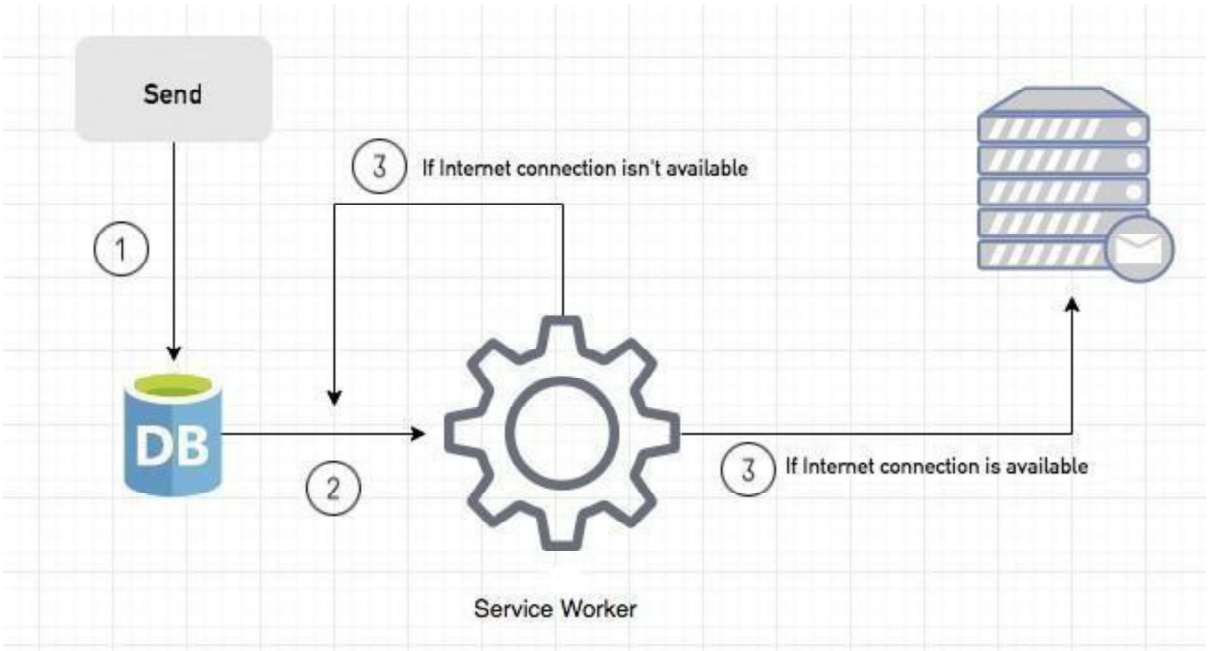
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

Here is a job for the Background Sync.

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



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



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

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

Event Listener for Background Sync Registration

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

Event Listener for sw.js

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

Push Event

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

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

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

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

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

Code:

In index.html

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

This code sends notification permission to your Device , and click on allow to send push notification

service-worker.js

```
// service-worker.js


const CACHE_NAME = 'my-ecommerce-app-cache-v1';
const urlsToCache = [
  '/',
  'cart.html',
  'index.html',
  'product.html',
  'shop.html',
  'style.css',
  'success.html',
  'service-worker.js',
  'manifest.json',
  'offline.html'
  // Add more files to cache as needed
];


self.addEventListener('install', function(event) {
  event.waitUntil(
    caches.open(CACHE_NAME)
```

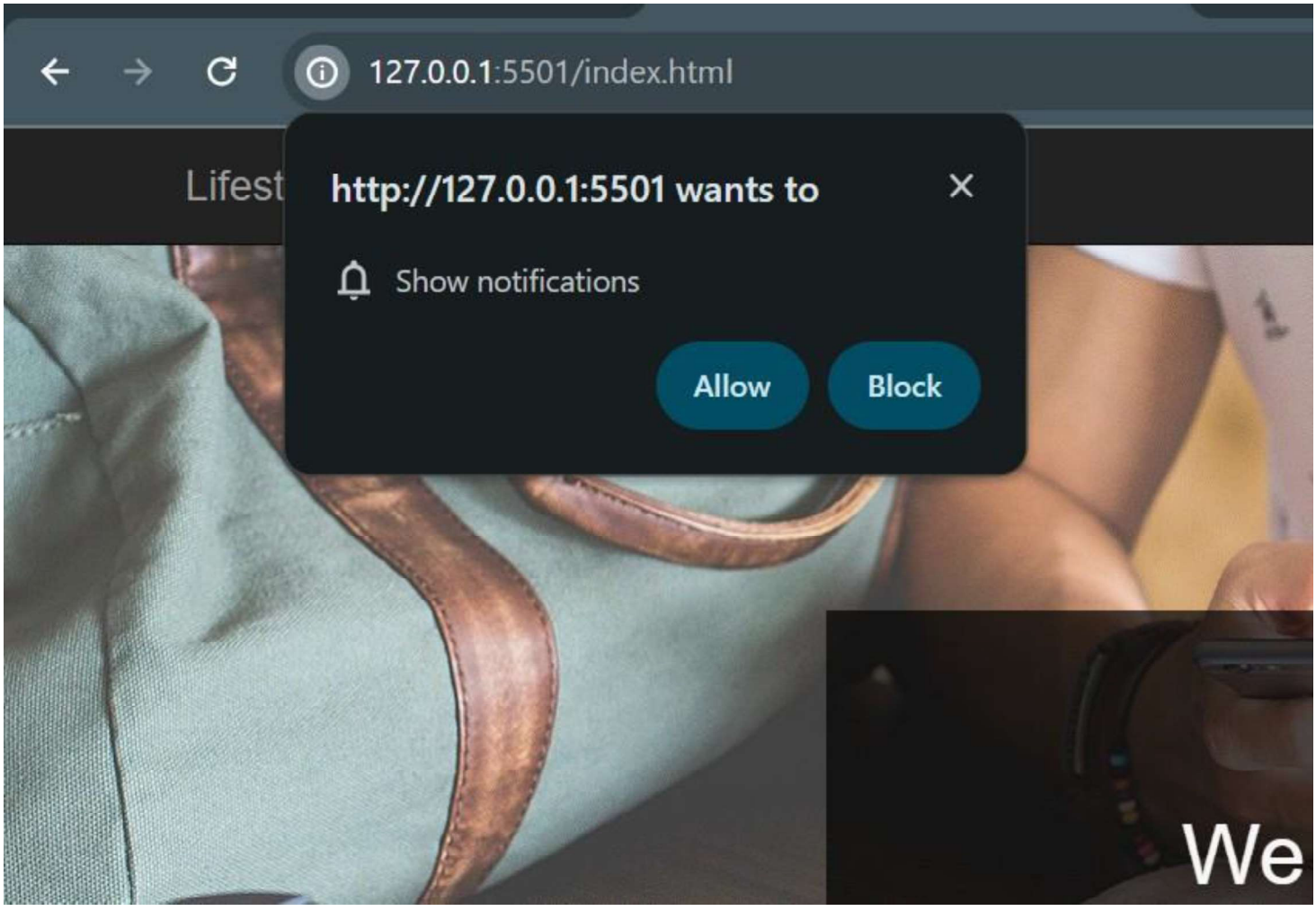
```
      .then(function(cache) {
        console.log('Opened cache');
        return cache.addAll(urlsToCache)
          .catch(function(error) {
            console.error('Cache.addAll error:', error);
          });
      })
  );
});

self.addEventListener('activate', function(event) {
  // Perform activation steps
  event.waitUntil(
    caches.keys().then(function(cacheNames) {
      return Promise.all(
        cacheNames.map(function(cacheName) {
          if (cacheName !== CACHE_NAME) {
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
});

// Fetch event listener
self.addEventListener("fetch", function (event) {
  event.respondWith(checkResponse(event.request).catch(function () {
    console.log("Fetch from cache successful!");
    return returnFromCache(event.request);
  }));
  console.log("Fetch successful!");
  event.waitUntil(addToCache(event.request));
```

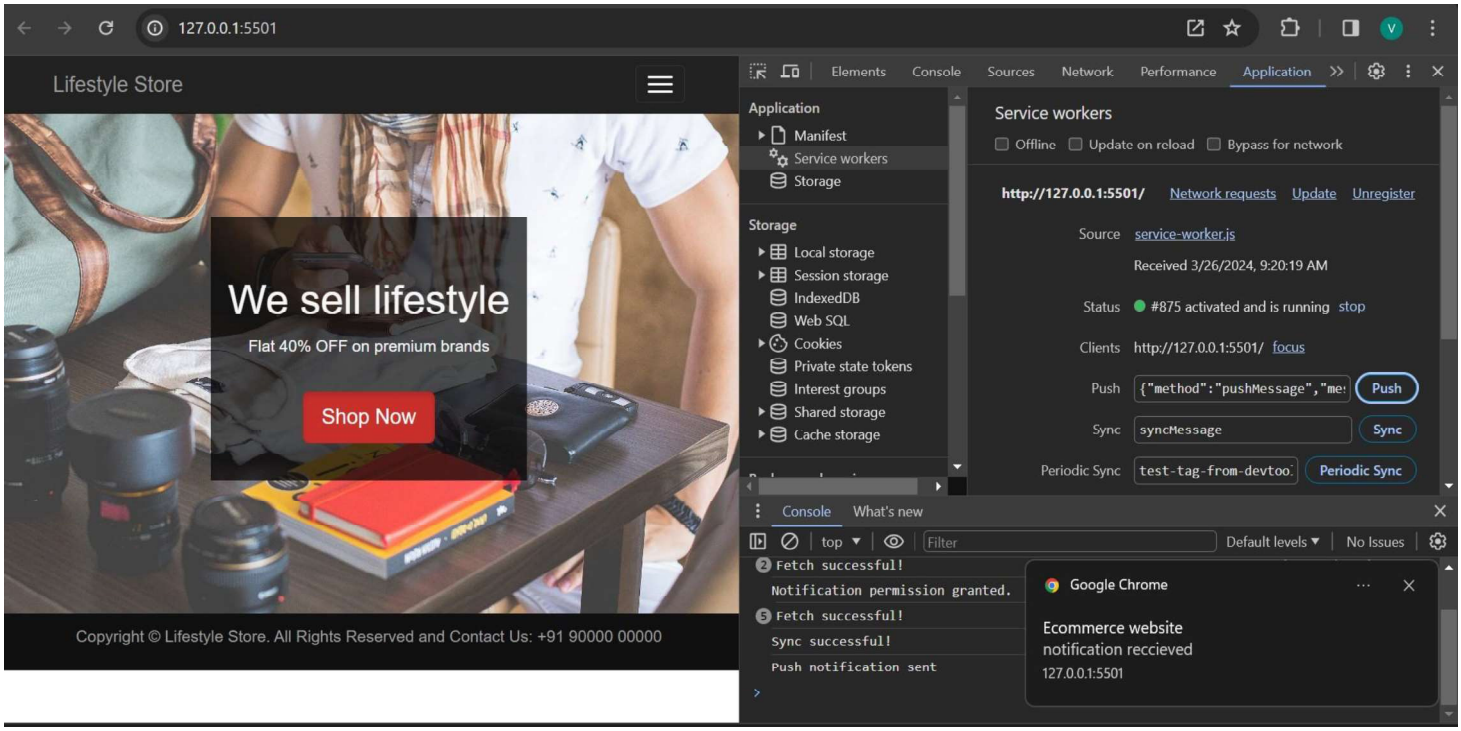
```
});

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

// Push event listener
self.addEventListener("push", function (event) {
  if (event && event.data) {
    try {
      var data = event.data.json();
      if (data && data.method === "pushMessage") {
        console.log("Push notification sent");
        self.registration.showNotification("Ecommerce website", { body: data.message
});
      }
    } catch (error) {
      console.error("Error parsing push data:", error);
    }
  }
});

self.addEventListener('activate', async () => {
  if (Notification.permission !== 'granted') {
    try {
      const permission = await Notification.requestPermission();
      if (permission === 'granted') {
        console.log('Notification permission granted.');
      } else {
        console.warn('Notification permission denied.');
      }
```

```
    } catch (error) {
      console.error('Failed to request notification permission:', error);
    }
  }
});

var checkResponse = function (request) {
  return new Promise(function (fulfill, reject) {
    fetch(request)
      .then(function (response) {
        if (response.status !== 404) {
          fulfill(response);
        } else {
          reject(new Error("Response not found"));
        }
      })
      .catch(function (error) {
        reject(error);
      });
  });
};

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

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

Output:





Conclusion: Hence we implemented methods like fetch,sync,and push on the service worker , and if we push the message, it says “notification recieved” on the desktop. So the push, sync , and fetch method is implemented successfully

MAD & PWA Lab

Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	32
Name	Himanshu Nilesh Lohote
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

MAD & PWA LAB
EXP- 10

Name: Himanshu Lohote Class: D15A
Roll No. 32

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

Theory:

GitHub Pages

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

GitHub Pages provides the following key features:
1. Blogging with Jekyll
2. Custom URL
3. Automatic Page Generator

Reasons for favoring this over Firebase:
1. Free to use
2. Right out of github
3. Quick to set up

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

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

Cons
1. The code of your website will be public, unless you pay for a private repository.

2. Currently, there is no support for HTTPS for custom domains. It's probably coming soon though.
3. Although Jekyll is supported, plug-in support is rather spotty.

Firebase

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

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

Pros

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

Cons

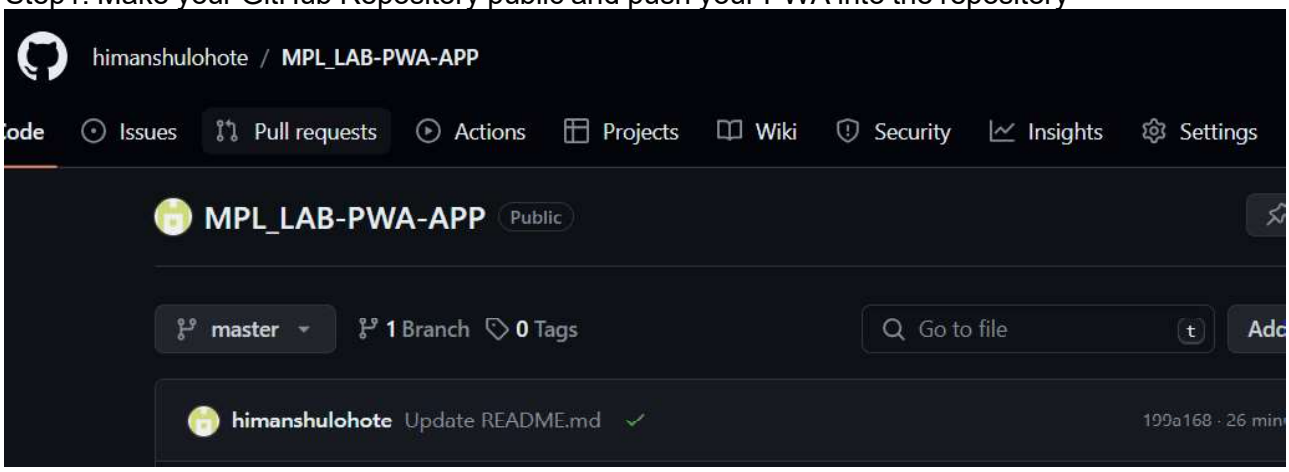
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator.

Link for GitHub:

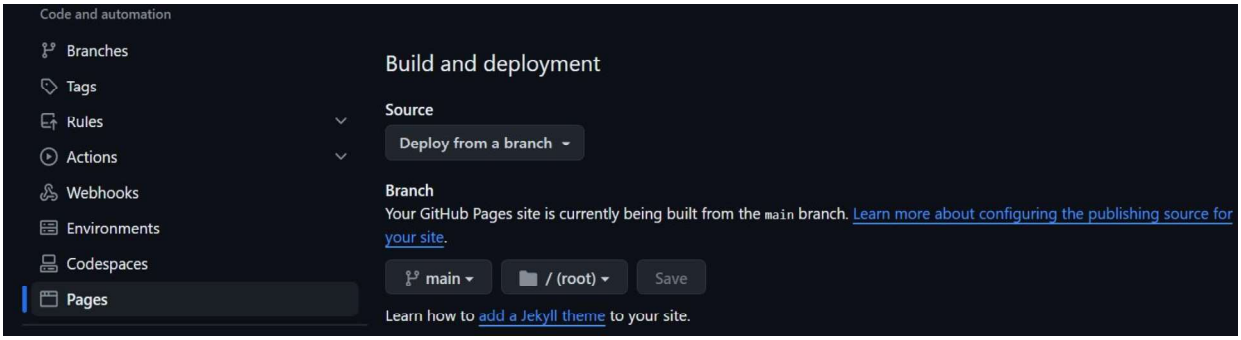
https://github.com/himanshulohote/MPL_LAB-PWA-APP

GitHub Screenshots:

Step1: Make your GitHub Repository public and push your PWA into the repository

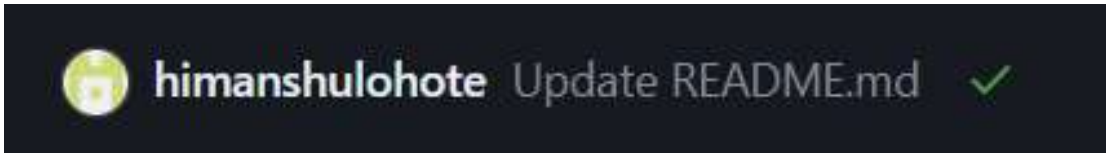


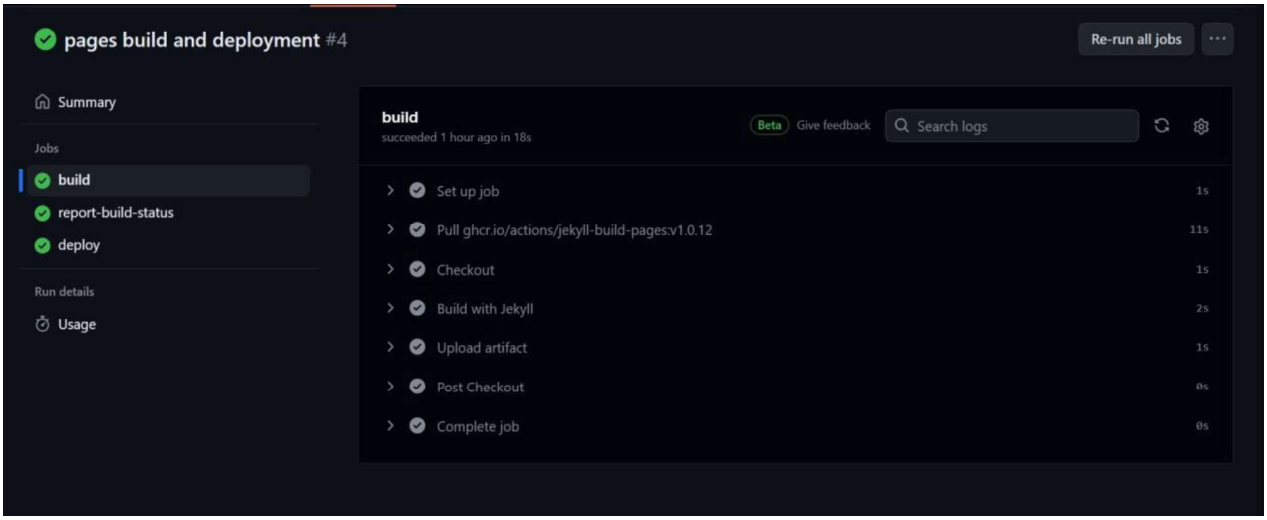
Step 2: Go to settings -> pages and choose your root directory and save it



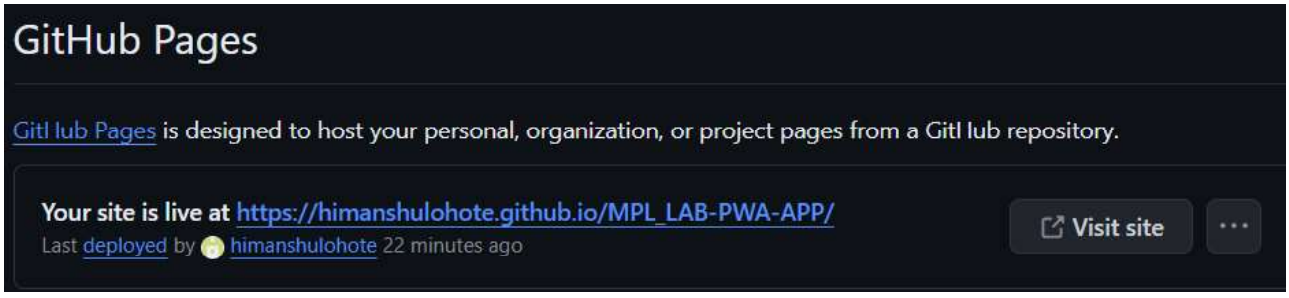
Step 3: Now go to your Code and you will see a small circle near your recent commit (Mine is finished deploying so i am getting a tick-mark sign)

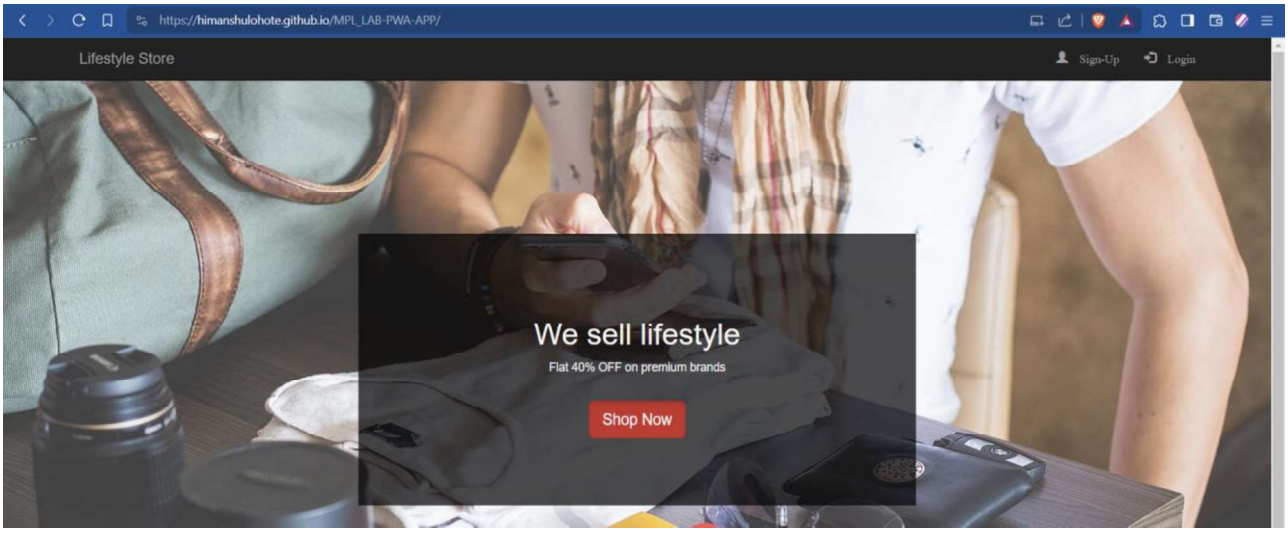
On clicking Logs of all the deployment is shown for convenience





Step 4: Go to Settings -> Pages again, and you will see the pages has been deployed and a link is given





Conclusion: Hence, we deployed our E-Commerce Progressive Web App Successfully 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.	32
Name	Himanshu Nilesh Lohote
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	15

MAD & PWA LAB
EXP- 11

Name: Himanshu Lohote Class: D15A
Roll No. 32

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

Theory:

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

Google Lighthouse:

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

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

Key Features and Audit Metrics

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

1. **Performance:** This score is an aggregation of how the page fared in aspects such as (but not limited to) loading speed, time taken for loading for basic frame(s), displaying meaningful content to the user, etc. To a layman, this score is indicative of how decently the

site performs, with a score of 100 meaning that you figure in the 98th percentile, 50 meaning that you figure in the 75th percentile and so on.

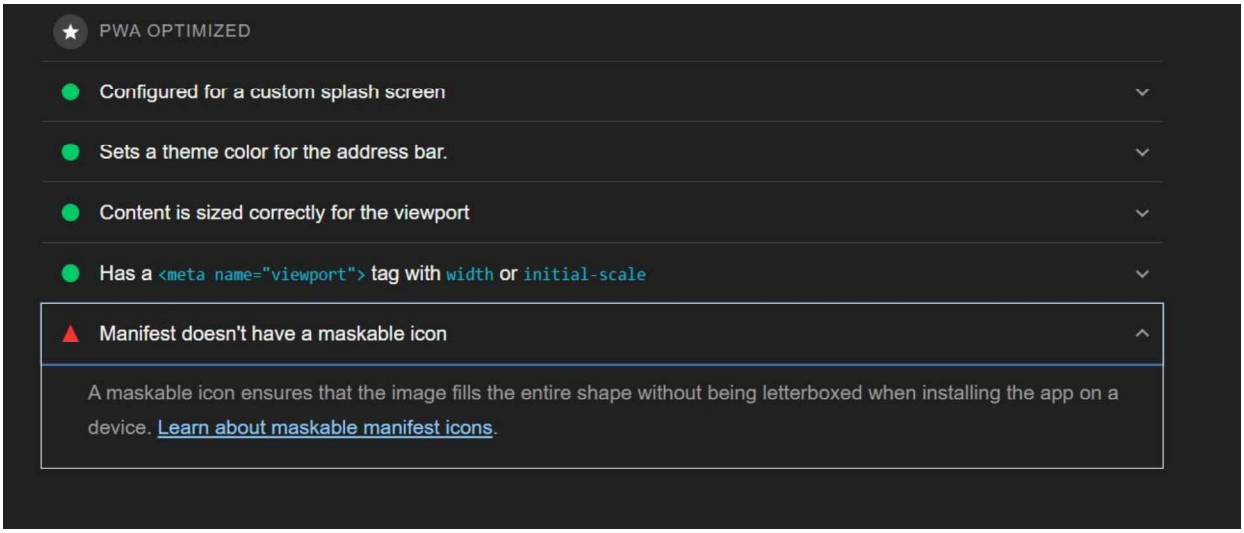
2. **PWA Score (Mobile):** Thanks to the rise of Service Workers, app manifests, etc., a lot of modern web applications are moving towards the PWA paradigm, where the objective is to make the application behave as close as possible to native mobile applications. Scoring points are based on the Baseline PWA checklist laid down by Google which includes Service Worker implementation(s), viewport handling, offline functionality, performance in script-disabled environments, etc.
3. **Accessibility:** As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics though, Accessibility metrics score on a pass/fail basis i.e. if all possible elements of the page are not screen-reader friendly (HTML5 introduced features that would make pages easy to interpret for screen readers used by visually challenged people like tag names, tags such as <section>, <article>, etc.), you get a 0 on that score. The aggregate of these scores is your Accessibility metric score.
4. **Best Practices:** As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS

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

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

Output:

Before



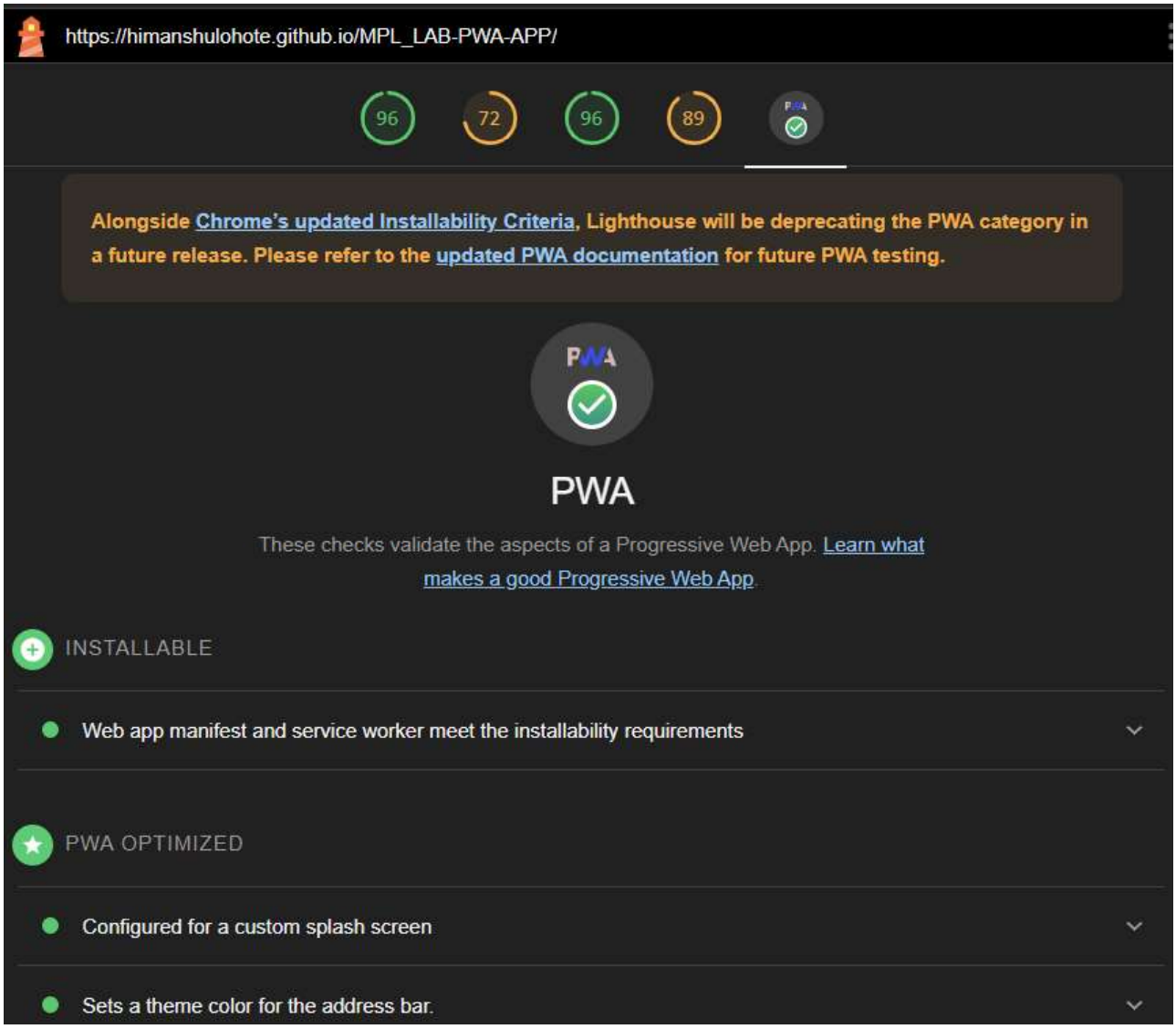
We encountered an issue here , it says “Manifest does not have a maskable icon”

Changes made to the code:

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

```
    }
]
}
```

After:



Conclusion: Hence by making some changes to the code, we did google lighthouse analysis and our PWA is Fully Optimized and ready to go

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.	32
Name	Himanshu Nilesh Lohote
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	5

30/1/2024

Himanshu Lohote Class : D15A

Batch B Roll No : 32

MAD & PWA Lab

Assignment - 1

Q.1] Flutter Overview:

A) • Flutter is a cross platform UI toolkit developed by Google, allowing developers to create natively compiled applications for mobile web and desktop from a single codebase, key features and advantages include:

- Hot Reload : Flutters Hot reload feature enables developers to instantly view changes made to the code without restarting the app, speeding up the development process.
- Single Codebase : Developers can write code and deploy it across multiple platforms, reducing development time
- Performance : flutter apps are compile directly to ARM resulting in high performance.

• Unlike traditional approaches like native development or hybrid frameworks, Flutter offers a more efficient and flexible solution by proving a unified environment.

Its popularity stems from its ability to deliver high-quality native experiences across multiple platforms while offering a fast and productive development workflow

Q.2] Widget tree and composition:

- In Flutter, the UI is built using a hierarchical structure called the widget tree. Widgets are the building blocks of Flutter UI.
- Widget composition refers to the process of combining these individual widgets to create more complex UI's.
- Commonly used widgets include:
  - Container
  - Row/column
  - Text
  - Image
  - List/view

Q.3] State Management in Flutter:

State management is crucial in Flutter application to manage and update the UI based on changes in data or user interactions.

- State management is essential in Flutter apps to keep the UI and data synchronized across different screens and interactions.

Flutter offers various approaches for this:

1. Setstate:

- Simple: Built-in method for updating widget state and triggering UI rebuilds.
- Suitable for: Small apps or simple UI changes within a single widget.
- Examples: Toggling switches, updating text fields.

2. Provider:

- Efficient: Uses Inherited Widget for scalable state management and dependency injection.
- Suitable for: Medium to large appls
- Examples: Users authencation, API data fetching.

3. Riverpod

- Advance: Extension of Provider with

improved scalability and testability.
Suitable for: Large apps.
Examples: Large widget trees, complex business Logic, rigorous unit testing.

- Choose 'setState' for simplicity, Provider for medium-sized projects and Riverpod for large-scale applications with complex state management demands.

Q.4] Firebase integration in Flutter:

- Integrating Firebase with Flutter app includes:
  1) Adding Firebase to project - Add the Firebase SDK to Flutter project by configuring ~~the~~ the pubspec.yaml and adding necessary config.
  2) Using Firebase services: like Firebase authentication for user authentication, Cloud messaging for notifications and Analytics for tracking App usage and engagement
  3) Data synchronization: like Firestore solution.

The benifits of Firebase include: its scalability reliability, ease of use, also firebase offers free tier which is benficial for developers.

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.	32
Name	Himanshu Nilesh Lohote
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4:Understand various PWA frameworks and their requirements LO5: Design and Develop a responsive User Interface by applying PWA Design techniques LO6:Develop and Analyze PWA Features and deploy it over app hosting solutions
Grade:	4

Himanshu Lohote
class : D15A

Batch - B
Roll No. 32

MAD & PWA Lab

Assignment 2

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

A)
- A Progressive Web App (PWA) is a type of web app built using using web technologies that ~~provides~~ a user experience similar to that of native mobile apps.
- Significance in Moodern Web Development
- cross platform compatibility : PWAs work Seamlessly across different devices and platforms.
- Offline capability : ~~PWAs work Seamlessly across different devices~~ and.
- Offine capabilities : PWAs can function offline or with a poor internet connection, enhancing user experience
- Improved performance : PWAs are fast and responsive offering a smooth user experience.
- Engaging user experience : PWAs can be added to user's home screen giving them feel of a native app

- Cost-effective development: PWAs eliminate the need to develop separate apps for different platforms, reducing development costs.

Key characteristics:

1. Installation: PWA do not require installation from a play store.
2. Updates: PWA are automatically updated.
3. Offline Access:
4. Cross-platform compatibility.
5. Discoverability

Q.2) Define responsive web design ~~and explain~~ and explain its importance in context of PWA.

A) - Responsive web design is an approach to web design aimed at creating sites to provide an optimal viewing and interaction experience across a wide range of devices from desktop computers to mobile phones.
- Ensures the PWAs adapt to various screen sizes and ~~dee~~ devices, providing consisting UX.
- Responsive WD uses flexible layouts and media queries to adapt

the layout to the viewing environment
- Ensures that content responds dynamically to changes in screen size and orientation.

2. Fluid Web Design: It is similar to responsive design but focuses more on fluid grids and elements that resize smoothly.
- Adapts seamlessly to any screen size.
- ~~Maintaining~~ maintaining proportionality across different devices.

3. Adaptive Web Design: It involves creating multiple fixed layout based on common device widths.

Q.3) Describe lifecycle of service workers including registration installation and activation phases.

A)
- Registration of service worker is done by using `navigator.serviceWorker.register()` method.
- Installation - The browser downloads and installs the service worker script.
- One installed the service worker is activated, allowing it to control pages or resources specified during registration.

Q.4) Explain use of IndexedDB in Service Worker for data storage

A) • IndexedDB is a low-level API for client-side storage of significant amounts of structured data
• It allows PWAs to store data locally
• Service workers can utilize IndexedDB for storing data fetched from the network. This allows PWA's to cache data for offline use and efficiently manage resources.

Advantages:
• Enables PWAs to provide seamless offline experiences by storing locally.
• Improves performance by reducing the need to fetch data from network repeatedly.