# 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 **ANUJ CHITARI** 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: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**

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

**Year/Sem/Class :** D15A/D15B **A.Y.:** 23-24

**Faculty Incharge** : Mrs. Kajal Joseph.

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

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

**Programme Outcomes**: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

**Program specific Outcomes**

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

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

**Project Title: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**

**Lab Objectives:**

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

**Lab Outcomes:**

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

# Index

| Sr. No | Experiment Title | LO | DOP | DOS | Grade |
|---|---|---|---|---|---|
| **1.** | To install and configure the Flutter Environment | LO1 | 16/1 | 23/1 | |
| **2.** | To design Flutter UI by including common widgets. | LO2 | 23/1 | 30/1 | |
| **3.** | To include icons, images, fonts in Flutter app | LO2 | 30/1 | 6/2 | |
| **4.** | To create an interactive Form using form widget | LO2 | 6/2 | 13/2 | |
| **5.** | To apply navigation, routing and gestures in Flutter App | LO2 | 13/2 | 20/2 | |
| **6.** | To Connect Flutter UI with fireBase database | LO3 | 20/2 | 5/3 | |
| **7.** | To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”. | LO4 | 5/3 | 12/3 | |
| **8.** | To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA | LO5 | 12/3 | 19/3 | |
| **9.** | To implement Service worker events like fetch, sync and push for E-commerce PWA | LO5 | 19/3 | 26/3 | |
| **10.** | To study and implement deployment of Ecommerce PWA to GitHub Pages. | LO5 | 26/3 | 2/4 | |
| **11.** | To use google Lighthouse PWA Analysis Tool to test the PWA functioning. | LO6 | 5/3 | 12/3 | |
| **12.** | Assignment-1 | LO1,LO2 ,LO3 | 2/2 | 5/2 | |
| **13.** | Assignment-2 | LO4,LO5 ,LO6 | 19/3 | 21/3 | |

# MAD & PWA Lab
# Journal

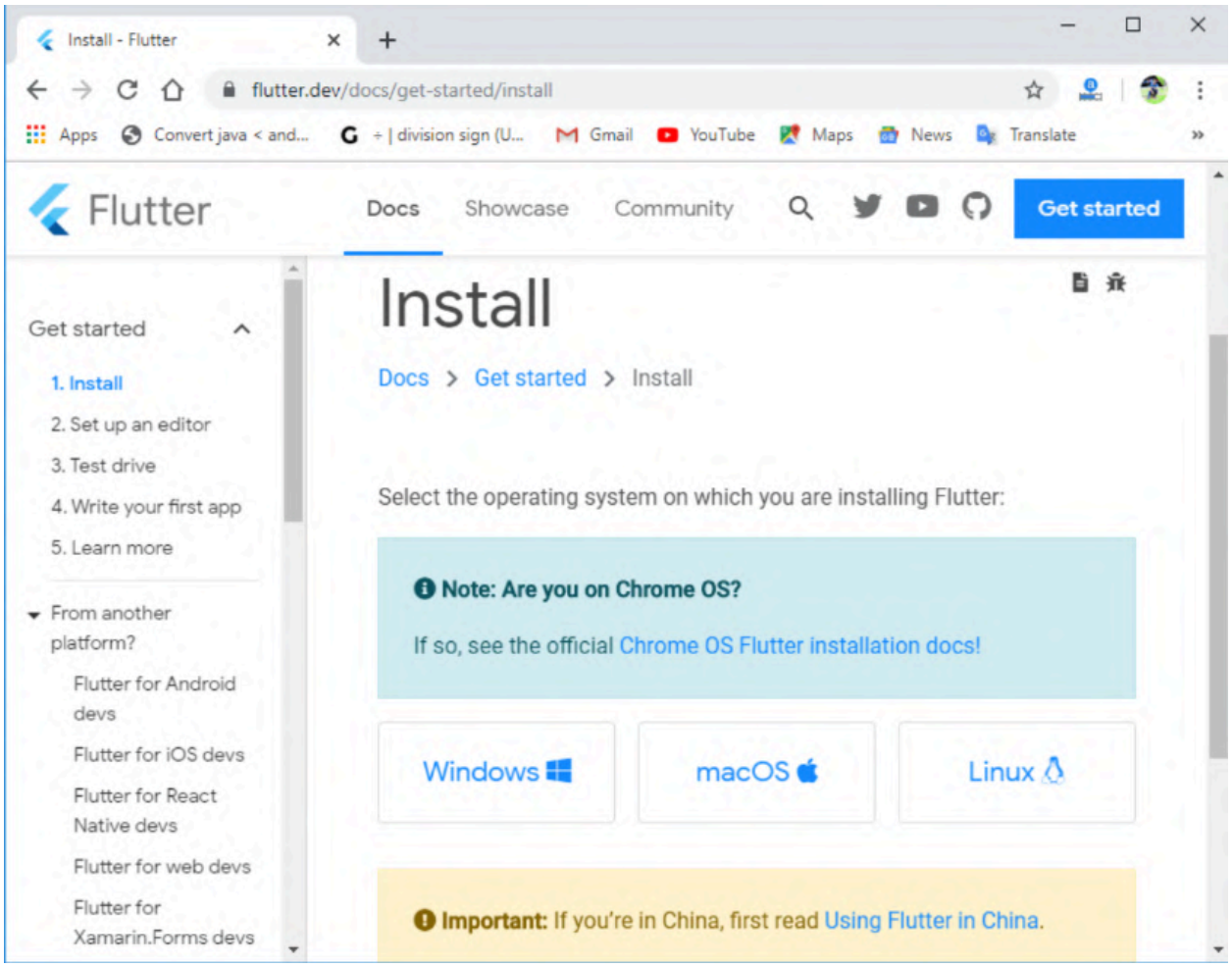
| | |
|---|---|
| Experiment No. | 01 |
| Experiment Title. | To install and configure the Flutter Environment |
| Roll No. | 11 |
| Name | Anuj Chitari |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO1: Understand cross platform mobile application development using Flutter framework |
| Grade: | |

# MAD LAB EXP-1

Name- Anuj Chitari Div-D15A Roll-11

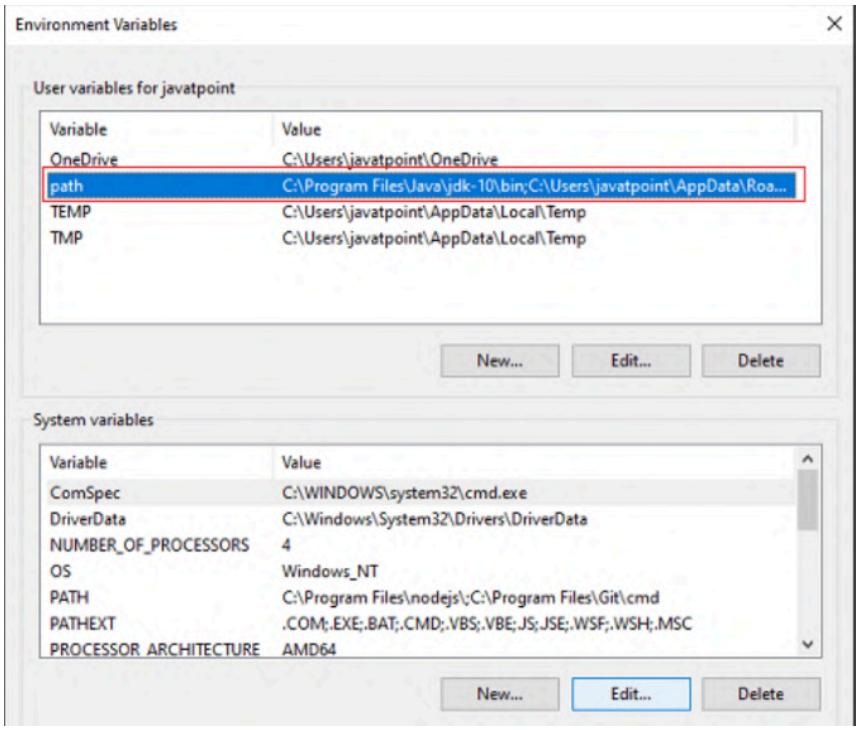
## EXP-1 Installation of Flutter

Step 1- Install the flutter SDK, download the latest Flutter SDK,

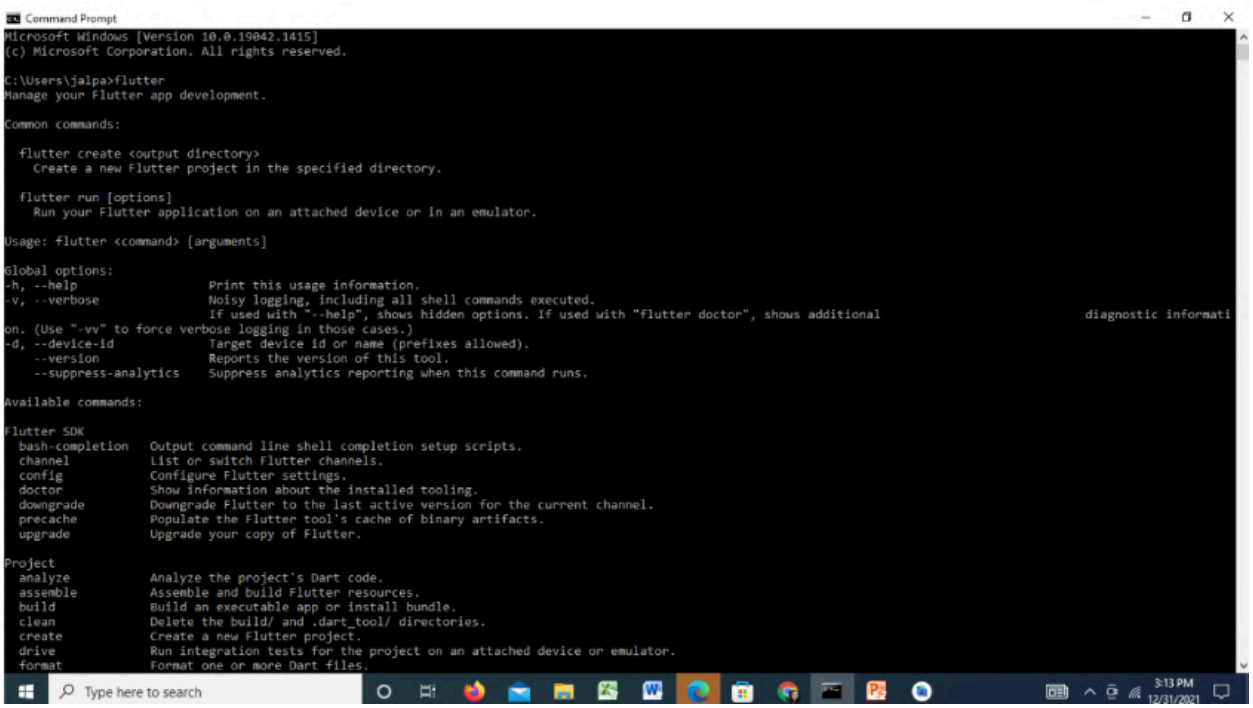



Step 2-When your download is complete, extract the zip file and place it in the desired installation folder or location, for example, C: /Flutter.

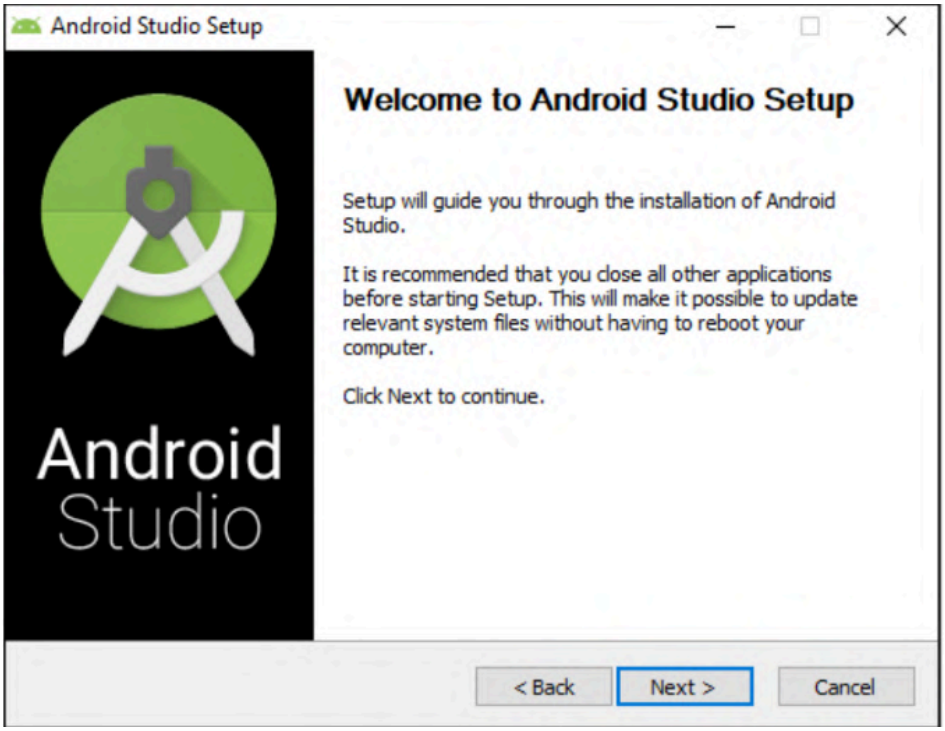
Step 3- Now edit the environment variables.



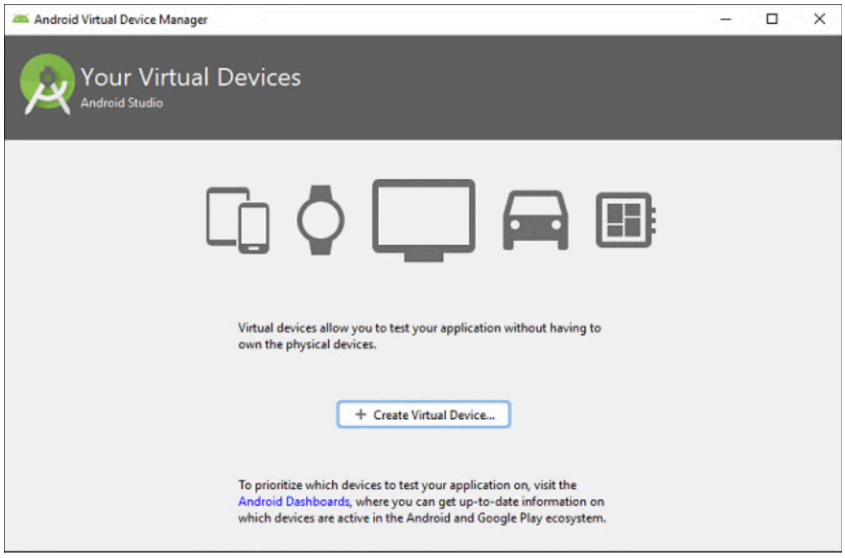

Step 4-Now, run the $ flutter and Flutter doctor command in command prompt.



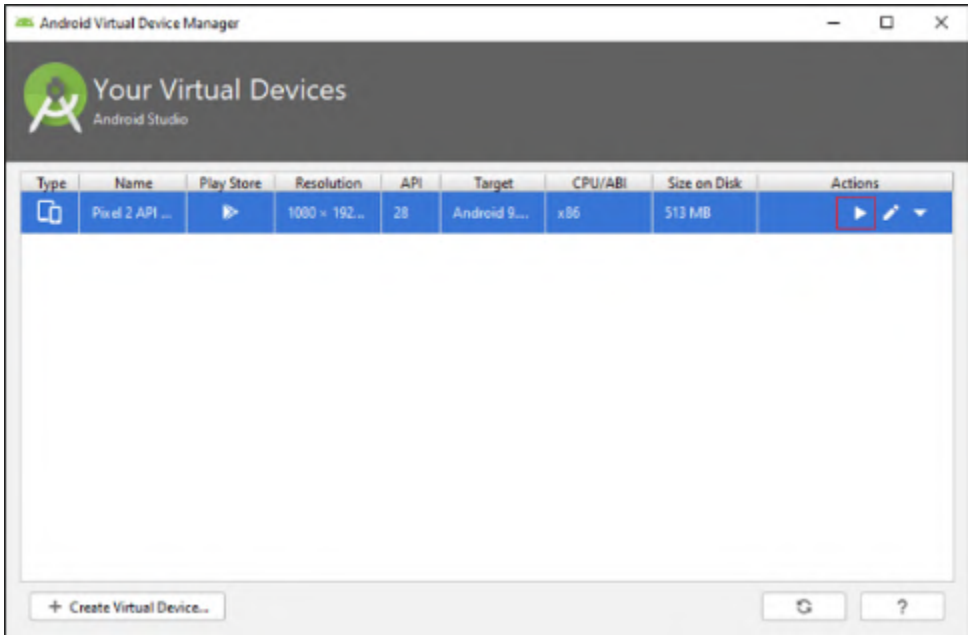
Step 5- Install the Android SDK. If the flutter doctor command does not find the Android SDK tool in your system, then you need first to install the Android Studio IDE.




Step 6- Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application.



Step 7- Select the system image for the latest Android version and click on Next. Now, verify the all AVD configuration. If it is correct, click on Finish. The following screen appears.




Step 8 - Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself.

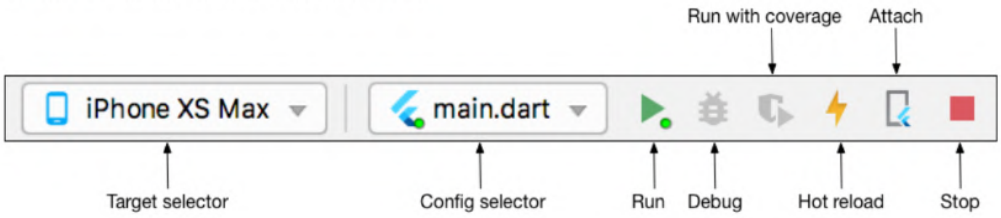
# RUNNING Hello World on Flutter

Step 1-Create the app.

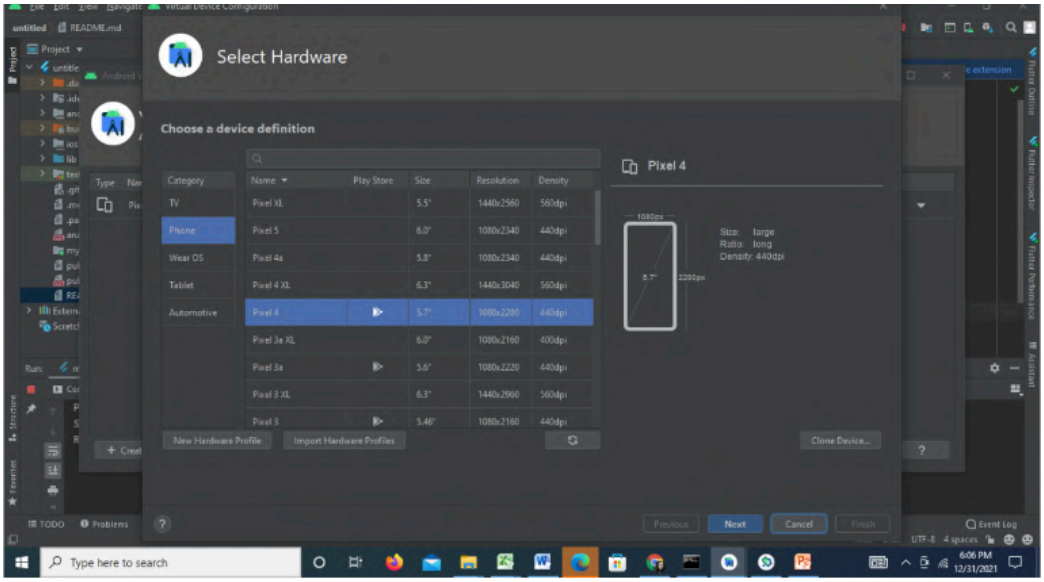
1. Open the IDE and select Create New Flutter Project.
2. Select Flutter Application as the project type. Then click Next.
3. Verify the Flutter SDK path specifies the SDK‟s location (select Install SDK... if the
text field is blank).
4. Enter a project name (for example, myapp). Then click Next.
5.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

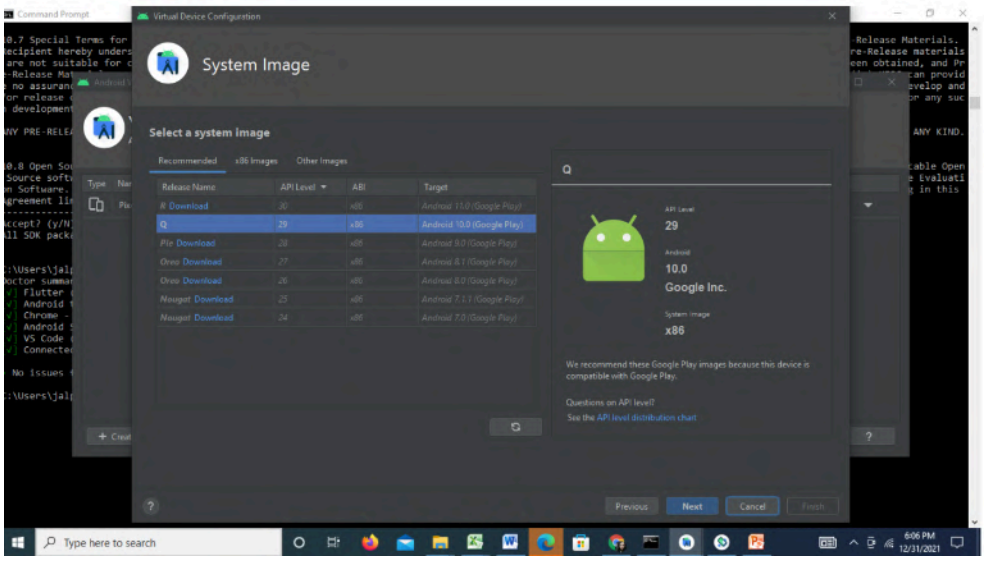
Step 2: Run the app.

Locate the main Android Studio toolbar:




Step 3-





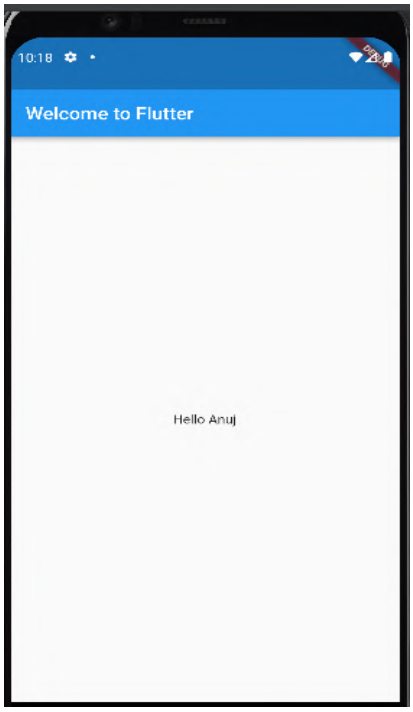

## Step 3 : Creating Hello world app

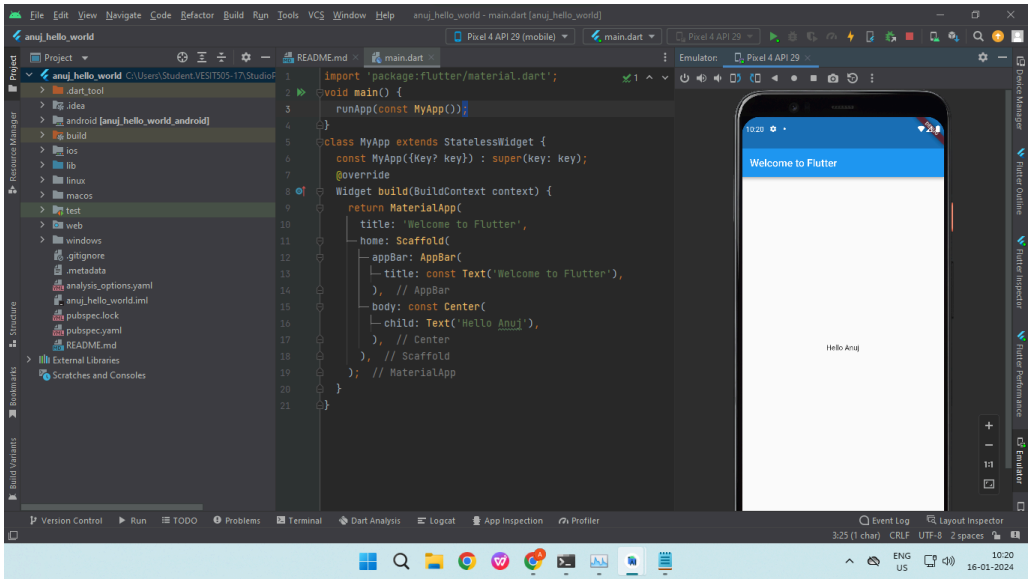
Code-

```
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 Anuj'),
        ),
      ),
    );
  }
}
```

**Project Title: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**

OUTPUT:-




Conclusion- Android studio has been downloaded and installed and Hello Anuj has been displayed.

# MAD & PWA Lab
# Journal

| | |
|---|---|
| Experiment No. | 02 |
| Experiment Title. | To design Flutter UI by including common widgets. |
| Roll No. | 11 |
| Name | Anuj Chitari |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation |
| Grade: | |

# EXP-2

Name- Anuj Chitari
Div: D15A R0ll- 11

**AIM**-: To include icons, images, fonts in Flutter app

**THEORY: 2**This widget holds the image which can fetch it from multiple sources like from the asset folder or directly from the URL. It provides many constructors for loading image, which are given below:

o Image: It is a generic image loader, which is used by ImageProvider.

o asset: It load image from your project asset folder.

o file: It loads images from the system folder.

o memory: It load image from memory.

o network: It loads images from the network.

To add an image in the project, you need first to create an assets folder where you keep your images and then add the below line in pubspec.yaml file.

Assets:
-assets/comp.jpeg

CODE:

```
import 'package:flutter/material.dart';
import 'package:food_panda/admin/home_admin.dart';
import 'package:food_panda/pages/bottomnav.dart';
import 'package:food_panda/pages/onboard.dart';
import 'package:firebase_core/firebase_core.dart';
import 'firebase_options.dart';
```

```
import 'package:food_panda/admin/admin_login.dart';



void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: FirebaseOptions(
    apiKey: 'AIzaSyCT2s9X9jKNBctZbW5dCRGpyqr1Z_-phM8',
    appId: '1:431523531027:android:562d4418509346182b32e6',
    messagingSenderId: '431523531027',
    projectId: 'fooddeliveryapp-5a9ad',
    storageBucket: 'fooddeliveryapp-5a9ad.appspot.com',)
  );
  runApp(MyApp()); // Replace MyApp() with your main application widget
}

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

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // TRY THIS: Try running your application with "flutter run". You'll see
        // the application has a purple toolbar. Then, without quitting the app,
        // try changing the seedColor in the colorScheme below to Colors.green
        // and then invoke "hot reload" (save your changes or press the "hot
        // reload" button in a Flutter-supported IDE, or press "r" if you used
        // the command line to start the app).
        //
        // Notice that the counter didn't reset back to zero; the application
        // state is not lost during the reload. To reset the state, use hot
        // restart instead.
        //
```

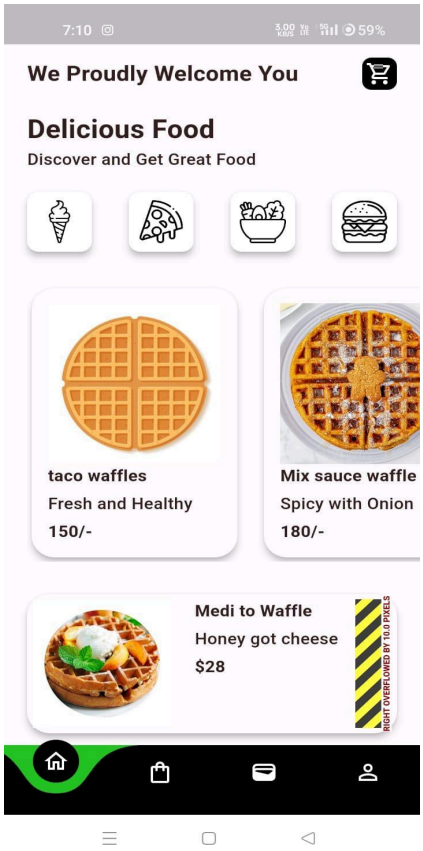
```
        // This works for code too, not just values: Most code changes can be
        // tested with just a hot reload.
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: Onboard(),
      // home: HomeAdmin(),
      // home: BottomNav(),
     // home: AdminLogin(),

    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});

  // This widget is the home page of your application. It is stateful, meaning
  // that it has a State object (defined below) that contains fields that affect
  // how it looks.

  // This class is the configuration for the state. It holds the values (in this
  // case the title) provided by the parent (in this case the App widget) and
  // used by the build method of the State. Fields in a Widget subclass are
  // always marked "final".

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      // This call to setState tells the Flutter framework that something has
      // changed in this State, which causes it to rerun the build method below
      // so that the display can reflect the updated values. If we changed
```

```
    // _counter without calling setState(), then the build method would not be
    // called again, and so nothing would appear to happen.
    _counter++;
  });
}

@override
Widget build(BuildContext context) {
  // This method is rerun every time setState is called, for instance as done
  // by the _incrementCounter method above.
  //
  // The Flutter framework has been optimized to make rerunning build methods
  // fast, so that you can just rebuild anything that needs updating rather
  // than having to individually change instances of widgets.
  return Scaffold(
    appBar: AppBar(
      // TRY THIS: Try changing the color here to a specific color (to
      // Colors.amber, perhaps?) and trigger a hot reload to see the AppBar
      // change color while the other colors stay the same.
      backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      // Here we take the value from the MyHomePage object that was created by
      // the App.build method, and use it to set our appbar title.
      title: Text(widget.title),
    ),
    body: Center(
      // Center is a layout widget. It takes a single child and positions it
      // in the middle of the parent.
      child: Column(
        // Column is also a layout widget. It takes a list of children and
        // arranges them vertically. By default, it sizes itself to fit its
        // children horizontally, and tries to be as tall as its parent.
        //
        // Column has various properties to control how it sizes itself and
        // how it positions its children. Here we use mainAxisAlignment to
        // center the children vertically; the main axis here is the vertical
        // axis because Columns are vertical (the cross axis would be
        // horizontal).
        //
        // TRY THIS: Invoke "debug painting" (choose the "Toggle Debug Paint"
        // action in the IDE, or press "p" in the console), to see the
```

```
        // wireframe for each widget.
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          const Text(
            'You have pushed the button this many times:',
          ),
          Text(
            '$_counter',
            style: Theme.of(context).textTheme.headlineMedium,
          ),
        ],
      ),
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: _incrementCounter,
      tooltip: 'Increment',
      child: const Icon(Icons.add),
    ), // This trailing comma makes auto-formatting nicer for build methods.
  );
}
```

**OUTPUT**

**Project Title: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**



# MAD & PWA Lab
# Journal

| | |
|---|---|
| Experiment No. | 03 |
| Experiment Title. | To include icons, images, fonts in Flutter app |
| Roll No. | 11 |
| Name | Anuj Chitari |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation |
| Grade: | |

**MAD and PWA Lab**

Name: Anuj Chitari
Class: D15A Roll no:11

**Experiment - 3**

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

## Theory:

Fonts:

In Flutter, the TextStyle class is used to define the styling for text within the
Text widget or other widgets that involve displaying text. Here's an overview of
how you can use the TextStyle class to set various font-related properties
fontSize:
You can set the size of the font using the fontSize
property. fontWeight:
The fontWeight property allows you to set the thickness of the characters in the text.
fontStyle:
The fontStyle property lets you specify whether the text should be in normal, italic,
or oblique style.
fontFamily:
You can specify the font family using the fontFamily property. This refers to the
specific font you want to use, and it should be available in your project. decoration:
The decoration property allows you to add decorations to the text, such as underline
or overline.

1. Text

A Text widget holds some text to display on the screen. We can align the text
widget by using textAlign property, and style property allow the customization of
Text that includes font, font weight, font style, letter spacing, color, and many
more.

2. Button

This widget allows you to perform some action on click. Flutter does not allow
you to use the Button widget directly; instead, it uses a type of buttons like a
FlatButton and a RaisedButton.

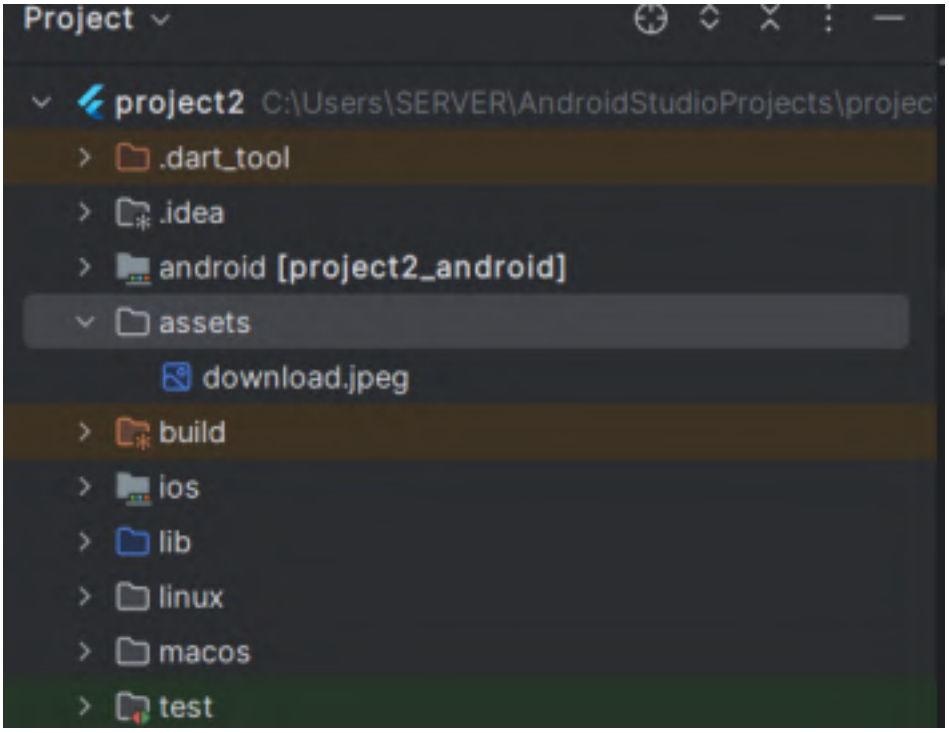
3. Image

This widget holds the image which can fetch it from multiple sources like
from the asset folder or directly from the URL. It provides many constructors for
loading image, which are given below:

o Image: It is a generic image loader, which is used by ImageProvider. o asset: It load image from your project asset folder.
o file: It loads images from the system folder.
o memory: It load image from memory.
o network: It loads images from the network.

To add an image in the project, you need first to create an assets folder where
you keep your images and then add the below line in pubspec.yaml file. assets:
- assets/images

FOLDER FORMAT:

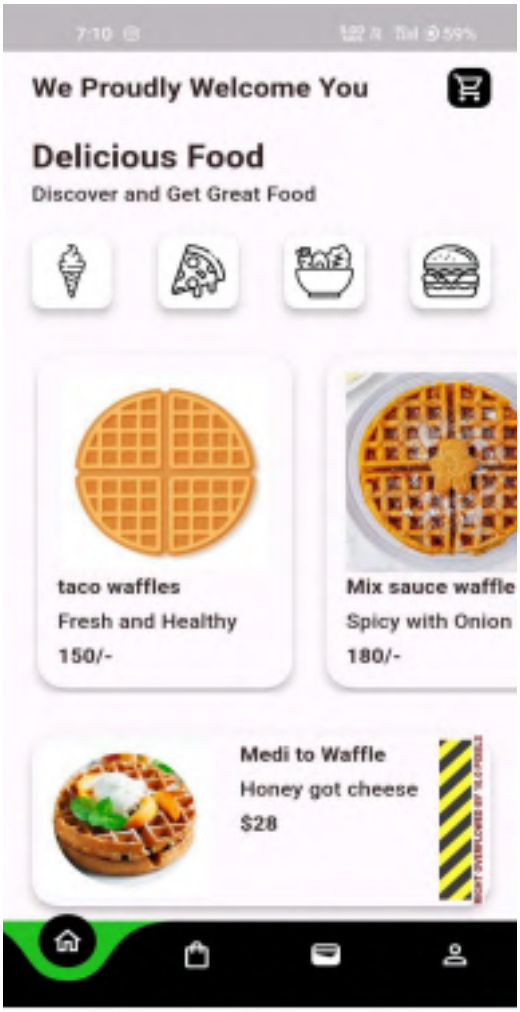



CODE:

```
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 UBEREATS!!'),
),
body: Center(
child: Image.asset('assets/download.jpeg'),
),
),
);
}
}
```

OUTPUT:

**Project Title: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**

# MAD & PWA Lab
# Journal

| | |
|---|---|
| Experiment No. | 04 |
| Experiment Title. | To create an interactive Form using form widget |
| Roll No. | 11 |
| Name | Anuj Chitari |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation |
| Grade: | |

# MAD LAB EXP 4

**Name- Anuj Chitari**
**Div-D15A Roll-11**

**AIM : To create an interactive Form using a form widget**

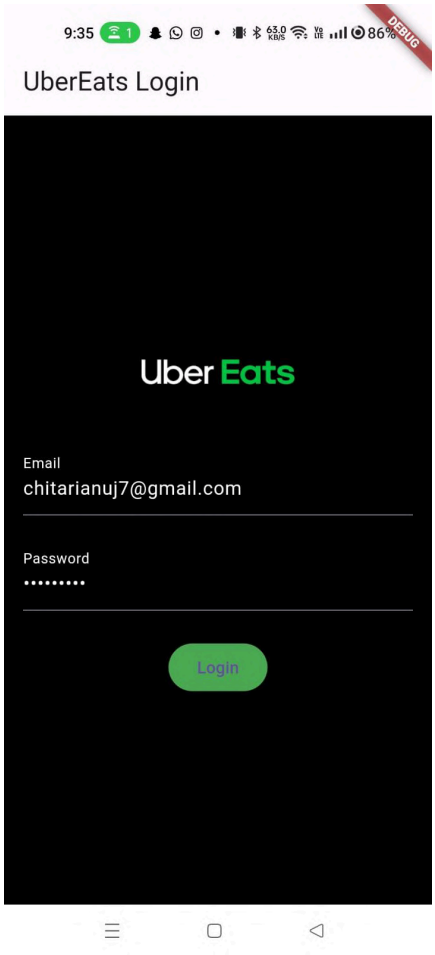
**Code-**

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


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


class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   home: Scaffold(
    appBar: AppBar(
     title: Text('Interactive Form'),
    ),
    body: MyForm(),
   ),
  );
 }
}


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


class _MyFormState extends State<MyForm> {
 final _formKey = GlobalKey<FormState>();
 String _name = '';
```

```
String _email = '';
String _selectedGender = 'Male';
bool _subscribeToNewsletter = false;


@override
Widget build(BuildContext context) {
 return Padding(
  padding: const EdgeInsets.all(16.0),
  child: Form(
   key: _formKey,
   child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
     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 || !value.contains('@')) {
        return 'Please enter a valid email address';
       }
       return null;
      },
      onSaved: (value) {
       _email = value!;
      },
     ),
     DropdownButtonFormField<String>(
      value: _selectedGender,
```

```
 items: ['Male', 'Female', 'Other']
    .map((gender) => DropdownMenuItem(
  value: gender,
  child: Text(gender),
 ))
    .toList(),
 onChanged: (value) {
  setState(() {
   _selectedGender = value!;
  });
 },
 decoration: InputDecoration(labelText: 'Gender'),
),
Row(
 children: [
  Checkbox(
   value: _subscribeToNewsletter,
   onChanged: (value) {
    setState(() {
     _subscribeToNewsletter = value ?? false;
    });
   },
  ),
  Text('Subscribe to Newsletter'),
 ],
),
SizedBox(height: 16),
ElevatedButton(
 onPressed: () {
  if (_formKey.currentState!.validate()) {
   _formKey.currentState!.save();
   // Process the form data, e.g., send it to a server
   print('Name: $_name');
   print('Email: $_email');
   print('Gender: $_selectedGender');
   print('Subscribe to Newsletter: $_subscribeToNewsletter');
  }
 },
 child: Text('Submit'),
),
```

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


9:35
DEBUG
UberEats Login
Uber Eats
Email
chitarianuj7@gmail.com
Password
Login

# MAD & PWA Lab
# Journal

| | |
|---|---|
| Experiment No. | 05 |
| Experiment Title. | To apply navigation, routing and gestures in Flutter App |
| Roll No. | 11 |
| Name | Anuj Chitari |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation |
| Grade: | |

# MAD PWA Lab
# Experiment - 5

**Name: Anuj Chitari**
**Class: D15A Roll no:11**

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

**Theory:**

**Navigation**:
Navigation is a fundamental aspect of mobile app development that involves transitioning between different screens or pages. In Flutter, the Navigator class plays a central role in managing the navigation stack, allowing developers to push and pop routes as users move through the app.

**Navigator Class:**
The Navigator class handles the navigation stack, maintaining a history of routes. The push method adds a new route to the stack, typically triggered by user actions. The pop method removes the current route from the stack, enabling backward Navigation.

**Routing:**
Routing in Flutter involves defining and organizing the paths or routes within the application. Routes represent different screens or pages, and their effective use is crucial for structuring the app's architecture.

**MaterialPageRoute and CupertinoPageRoute:**
MaterialPageRoute is employed in apps following Material Design principles, providing a standard Android-style transition between screens.
CupertinoPageRoute is used for iOS-style designs, ensuring a consistent and native user experience.

**Named Routes:**
Named routes offer a more organized and readable approach to navigation. By assigning names to routes, such as '/details', developers can easily navigate to specific screens using Navigator.pushNamed.

**Gestures in Flutter:**

Gestures enhance user interaction by allowing the app to respond to touch or mouse inputs. In Flutter, the GestureDetector widget is instrumental in recognizing and handling various gestures.

**GestureDetector Widget:**
The GestureDetector widget is used to detect a variety of gestures, including taps, drags, and long presses. By wrapping UI components with GestureDetector, developers can specify callback
functions to execute when specific gestures are detected.

**Gesture Recognizers:**
Flutter provides gesture recognizers for more complex gestures, such as panning, pinching, and swiping. These recognizers, when combined with a GestureDetector, enable the app to respond to a broader range of user inputs.

**InkWell and InkWell:**
The InkWell and InkResponse widgets bring a material ripple effect to touchable UI components. Integrating these widgets enhances the visual feedback during user interactions, contributing to a more polished and intuitive user experience.

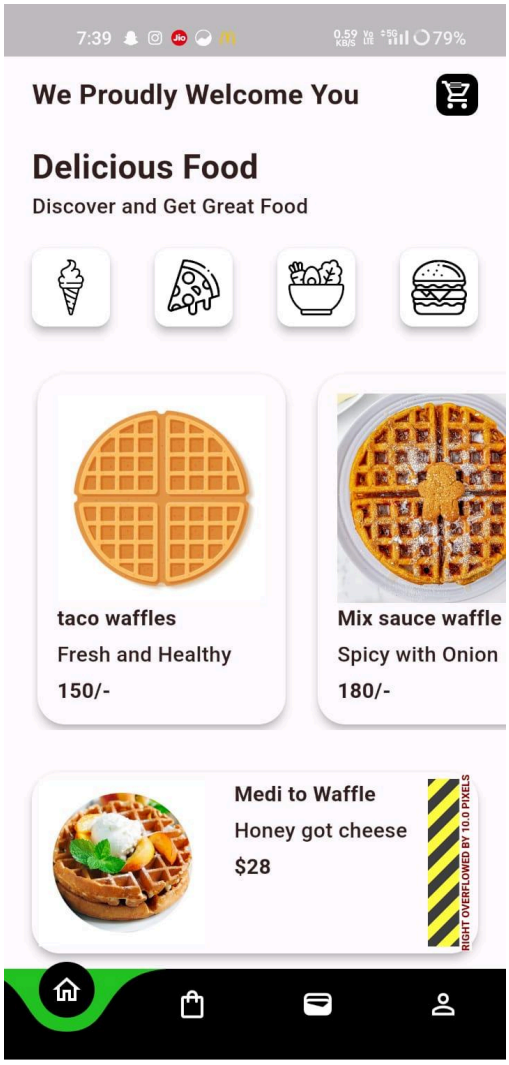
**CODE:** Routing/Navigation and Gesture Detector between **home.dart** and **details.dart**
**Home.dart**

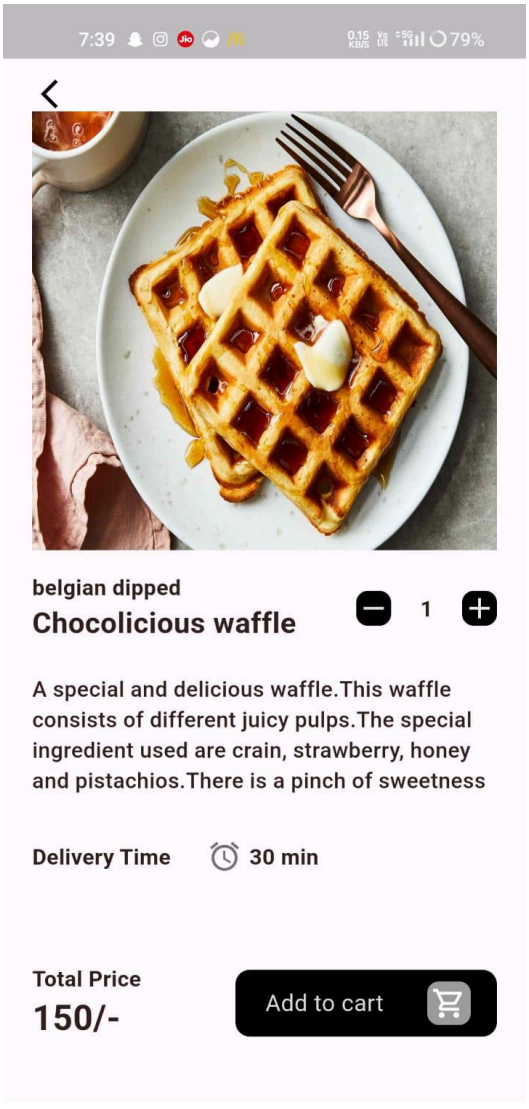
```
Click here to ask Blackbox to help you code faster
import 'package:flutter/material.dart';
import 'package:food_panda/pages/details.dart';
import 'package:food_panda/widget/widget_support.dart';

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(
        scrollDirection: Axis.vertical,
        child: Container(
          margin: const EdgeInsets.only(top: 50.0, left: 20.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  Text("We Proudly Welcome You", style: AppWidget.boldTextFieldStyle()),
                  Container(
                    margin: const EdgeInsets.only(right: 20.0),
                    padding: const EdgeInsets.all(3),
                    decoration: BoxDecoration(
                        color: Colors.black,
                        borderRadius: BorderRadius.circular(8)), // BoxDecoration
                    child: const Icon(
                      Icons.shopping_cart_outlined,
                      color: Colors.white,
                    ), // Icon
                  ) // Container
                ],
              ), // Row
              const SizedBox(
                height: 20.0,
              ), // SizedBox
              Text("Delicious Food", style: AppWidget.HeadlineTextFieldStyle()),
              Text("Discover and Get Great Food",
```

```
Text("Discover and Get Great Food",
    style: AppWidget.LightTextFieldStyle()), // Text
const SizedBox(
  height: 20.0,
), // SizedBox
Container(
    margin: const EdgeInsets.only(right: 20.0),
    child: showItem()), // Container
const SizedBox(
  height: 30.0,
), // SizedBox
SingleChildScrollView(
  scrollDirection: Axis.horizontal,
  child: Row(
    children: [
      GestureDetector(
        onTap: () {
          Navigator.push(context,
              MaterialPageRoute(builder: (context) => Details()));
        },
        child: Container(
          margin: const EdgeInsets.all(4),
          child: Material(
            elevation: 5.0,
            borderRadius: BorderRadius.circular(20),
            child: Container(
              padding: const EdgeInsets.all(14),
              child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Image.asset(
                      "images/wafflee.png",
                      height: 150,
                      width: 150,
                      fit: BoxFit.cover,
                    ), // Image.asset
                    Text("taco waffles",
                        style:
                            AppWidget.semiBoldTextFieldStyle()), // Text
                    const SizedBox(
                      height: 5.0,
                    ), // SizedBox
                    Text("Fresh and Healthy",
                        style: AppWidget.LightTextFieldStyle()), // Text
                    const SizedBox(
                      height: 5.0,
```

**Project Title: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**

**OUTPUT:**



# MAD & PWA Lab
# Journal

| | |
|---|---|
| Experiment No. | 06 |
| Experiment Title. | To Connect Flutter UI with fireBase database |
| Roll No. | 11 |
| Name | Anuj Chitari |
| 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: | |

# MAD and PWA Lab

# Experiment - 6

**Name: Anuj Chitari** **Class: D15A Roll no:04**

# Aim: To Connect Flutter UI with FireBase database Theory:

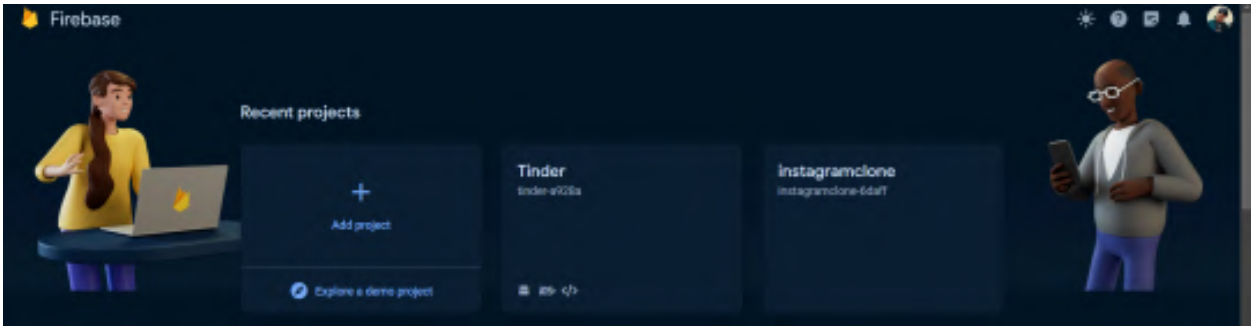
## Prerequisites

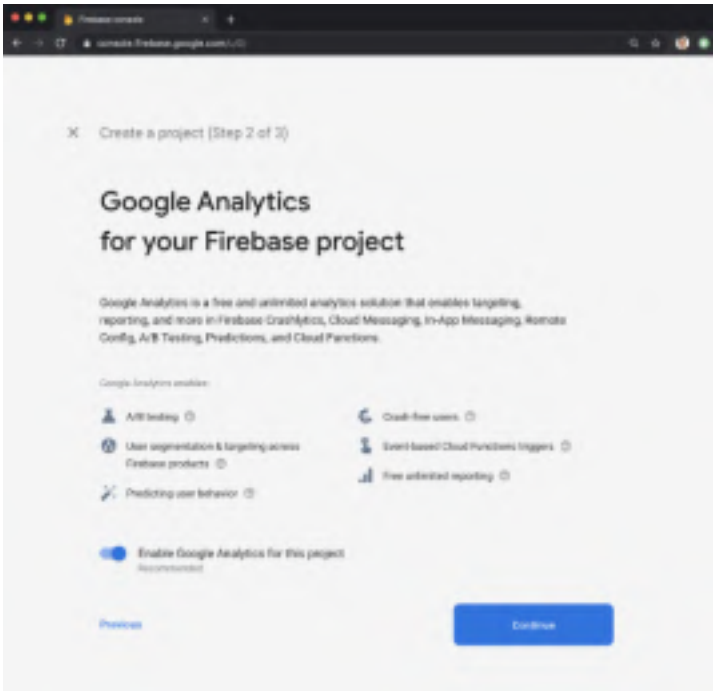
To complete this tutorial, you will need:

- 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.

## Create a 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:




Go to the Firebase Console and create a new project.
Add your Flutter app to the Firebase project:

Register your app in the Firebase project, and follow the instructions to download the configuration files (google-services.json for Android, GoogleService-Info.plist for iOS).

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:

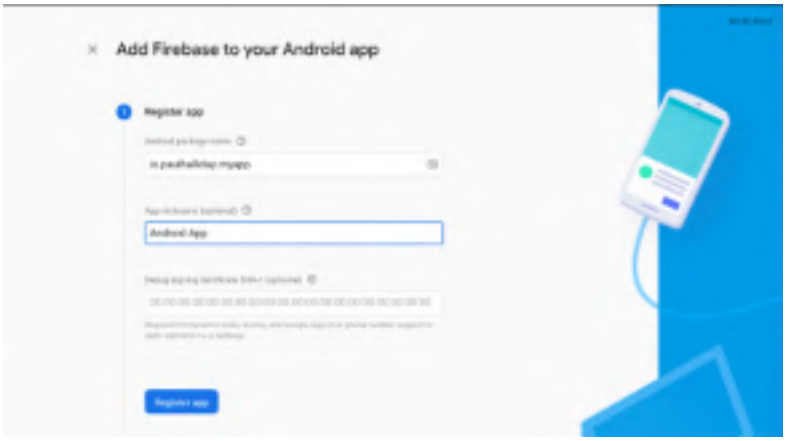
com.example.flutterfirebaseexample

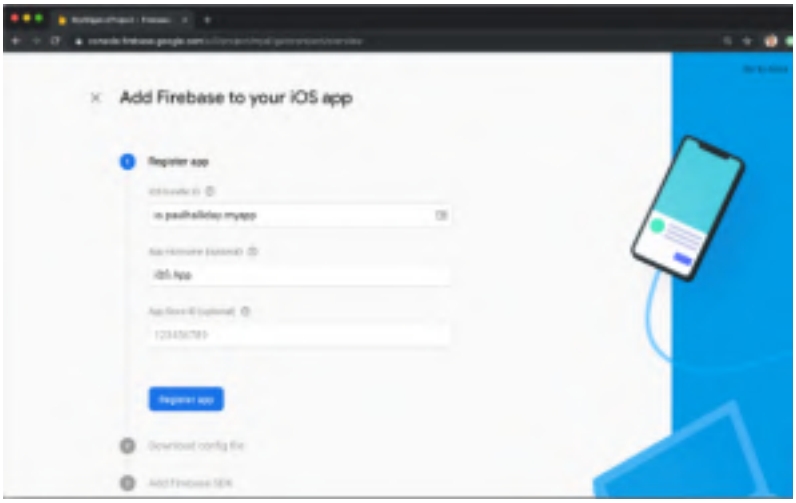
Once you've decided on a name, open android/app/build.gradle in your code editor and update the applicationId to match the Android package name:

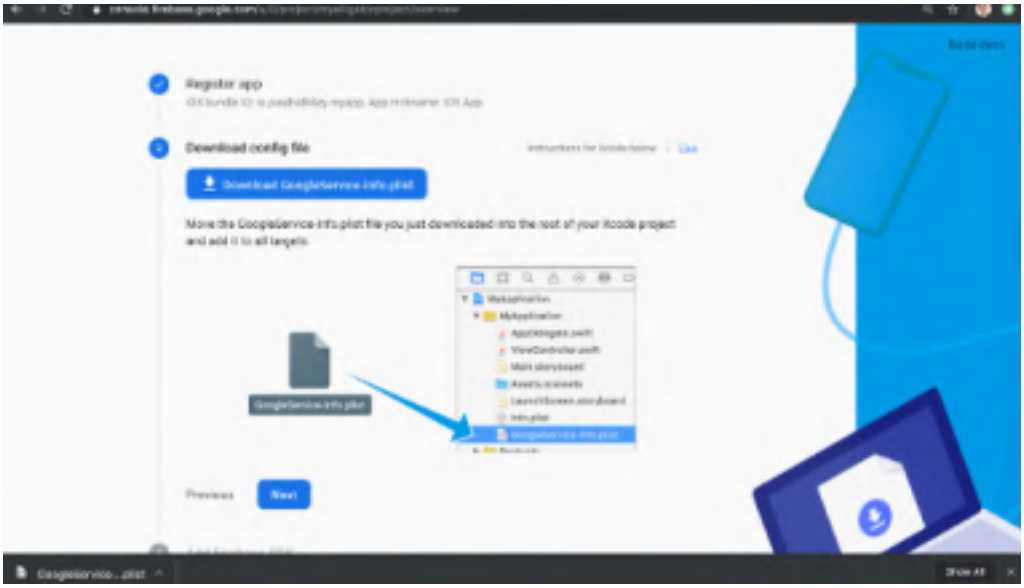
```
android/app/build.gradle
...
defaultConfig {
    // TODO: Specify your own unique Application ID
    (https://developer.android.com/studio/build/application-id.html). applicationId
    'com.example.flutterfirebaseexample'
    ...
}
...
```

## 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:






2. Add Firebase to your Flutter project:
Add Dependencies:

Open your pubspec.yaml file and add the necessary dependencies:
yaml

```yaml
dependencies:
  flutter:
    sdk: flutter

  curved_navigation_bar: ^1.0.3 #latest version
  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.2
  firebase_core: ^2.25.4
  firebase_auth: ^4.17.4
  cloud_firestore: ^4.15.4
  image_picker: ^1.0.7
  firebase_storage: ^11.6.5
  shared_preferences: ^2.2.2
  random_string: ^2.3.1
```

Code:
Signup.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:food_panda/pages/bottomnav.dart';
import 'package:food_panda/pages/login.dart';
import 'package:food_panda/widget/widget_support.dart';


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

  @override
  State<SignUp> createState() => _SignUpState();
}

class _SignUpState extends State<SignUp> {
  String email = "", password = "", name = "";

  TextEditingController namecontroller = new TextEditingController();

  TextEditingController passwordcontroller = new TextEditingController();

  TextEditingController mailcontroller = new TextEditingController();

  final _formkey = GlobalKey<FormState>();

  registration() async {
    if (password != null) {
      try {
        UserCredential userCredential = await FirebaseAuth.instance
            .createUserWithEmailAndPassword(email: email, password: password);

        ScaffoldMessenger.of(context).showSnackBar((SnackBar(
            backgroundColor: Colors.redAccent,
            content: Text(
              "Registered Successfully",
              style: TextStyle(fontSize: 20.0),
            ))));

        // ignore: use_build_context_synchronously
        Navigator.pushReplacement(
            context, MaterialPageRoute(builder: (context) => BottomNav()));
      } on FirebaseException catch (e) {
```

```
      if (e.code == 'weak-password') {
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
            backgroundColor: Colors.orangeAccent,
            content: Text(
              "Password Provided is too Weak",
              style: TextStyle(fontSize: 18.0),
            )));
      } else if (e.code == "email-already-in-use") {
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
            backgroundColor: Colors.orangeAccent,
            content: Text(
              "Account Already exsists",
              style: TextStyle(fontSize: 18.0),
            )));
      }
    }
  }
}


@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Container(
      child: Stack(
        children: [
          Container(
            width: MediaQuery.of(context).size.width,
            height: MediaQuery.of(context).size.height / 2.5,
            decoration: BoxDecoration(
                gradient: LinearGradient(
                    begin: Alignment.topLeft,
                    end: Alignment.bottomRight,
                    colors: [
                    Color(0xFF000000),
                  Color(0xFF000000),
                ])),
          ),
          Container(
            margin:
                EdgeInsets.only(top: MediaQuery.of(context).size.height / 3),
            height: MediaQuery.of(context).size.height / 2,
            width: MediaQuery.of(context).size.width,
```

```
  decoration: BoxDecoration(
      color: Color.fromARGB(255, 18, 17, 17),
      borderRadius: BorderRadius.only(
          topLeft: Radius.circular(40),
          topRight: Radius.circular(40))),
  child: Text(""),
),
Container(
  margin: EdgeInsets.only(top: 60.0, left: 20.0, right: 20.0),
  child: Column(
    children: [
      Center(
          child: Image.asset(
        "images/logo copy.png",
        width: MediaQuery.of(context).size.width / 1.5,
        fit: BoxFit.cover,
      )),
      SizedBox(
        height: 50.0,
      ),
      Material(
        elevation: 5.0,
        borderRadius: BorderRadius.circular(20),
        child: Container(
          padding: EdgeInsets.only(left: 20.0, right: 20.0),
          width: MediaQuery.of(context).size.width,
          height: MediaQuery.of(context).size.height / 1.8,
          decoration: BoxDecoration(
              color: const Color.fromARGB(255, 255, 255, 255),
              borderRadius: BorderRadius.circular(20)),
          child: Form(
            key: _formkey,
            child: Column(
              children: [
                SizedBox(
                  height: 30.0,
                ),
                Text(
                  "Sign up",
                  style: AppWidget.HeadlineTextFieldStyle(),
                ),
                SizedBox(
                  height: 30.0,
```

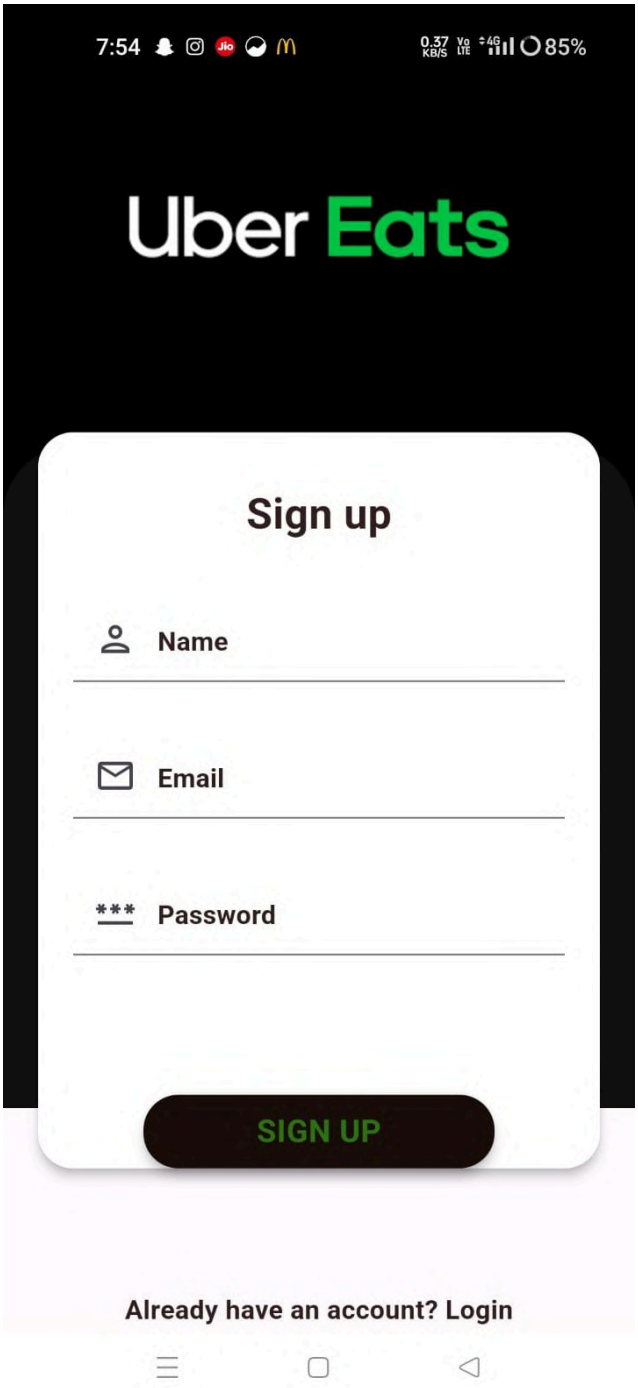
```
),
TextFormField(
  controller: namecontroller,
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please Enter Name';
    }
    return null;
  },
  decoration: InputDecoration(
      hintText: 'Name',
      hintStyle: AppWidget.semiBoldTextFieldStyle(),
      prefixIcon: Icon(Icons.person_outlined)),
),
SizedBox(
  height: 30.0,
),
TextFormField(
  controller: mailcontroller,
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please Enter E-mail';
    }
    return null;
  },
  decoration: InputDecoration(
      hintText: 'Email',
      hintStyle: AppWidget.semiBoldTextFieldStyle(),
      prefixIcon: Icon(Icons.email_outlined)),
),
SizedBox(
  height: 30.0,
),
TextFormField(
  controller: passwordcontroller,
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please Enter Password';
    }
    return null;
  },
  obscureText: true,
  decoration: InputDecoration(
```

```
                    hintText: 'Password',
                    hintStyle: AppWidget.semiBoldTextFieldStyle(),
                    prefixIcon: Icon(Icons.password_outlined)),
                ),
                SizedBox(
                  height: 80.0,
                ),
                GestureDetector(
                  onTap: () async {
                    if (_formkey.currentState!.validate()) {
                      setState(() {
                        email = mailcontroller.text;
                        name = namecontroller.text;
                        password = passwordcontroller.text;
                      });
                    }
                    registration();
                  },
                  child: Material(
                    elevation: 5.0,
                    borderRadius: BorderRadius.circular(20),
                    child: Container(
                      padding: EdgeInsets.symmetric(vertical: 8.0),
                      width: 200,
                      decoration: BoxDecoration(
                          color: Color.fromARGB(255, 26, 14, 10),
                          borderRadius: BorderRadius.circular(20)),
                      child: Center(
                          child: Text(
                        "SIGN UP",
                        style: TextStyle(
                            color: Color.fromARGB(153, 62, 185, 21),
                            fontSize: 18.0,
                            fontFamily: 'Poppins1',
                            fontWeight: FontWeight.bold),
                      )),
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
```

```
                ),
                SizedBox(
                  height: 70.0,
                ),
                GestureDetector(
                    onTap: () {
                      Navigator.push(context,
                          MaterialPageRoute(builder: (context) => LogIn()));
                    },
                    child: Text(
                      "Already have an account? Login",
                      style: AppWidget.semiBoldTextFieldStyle(),
                    ))
              ],
            ),
          )
        ],
      ),
    ),
  );
 }
}
```

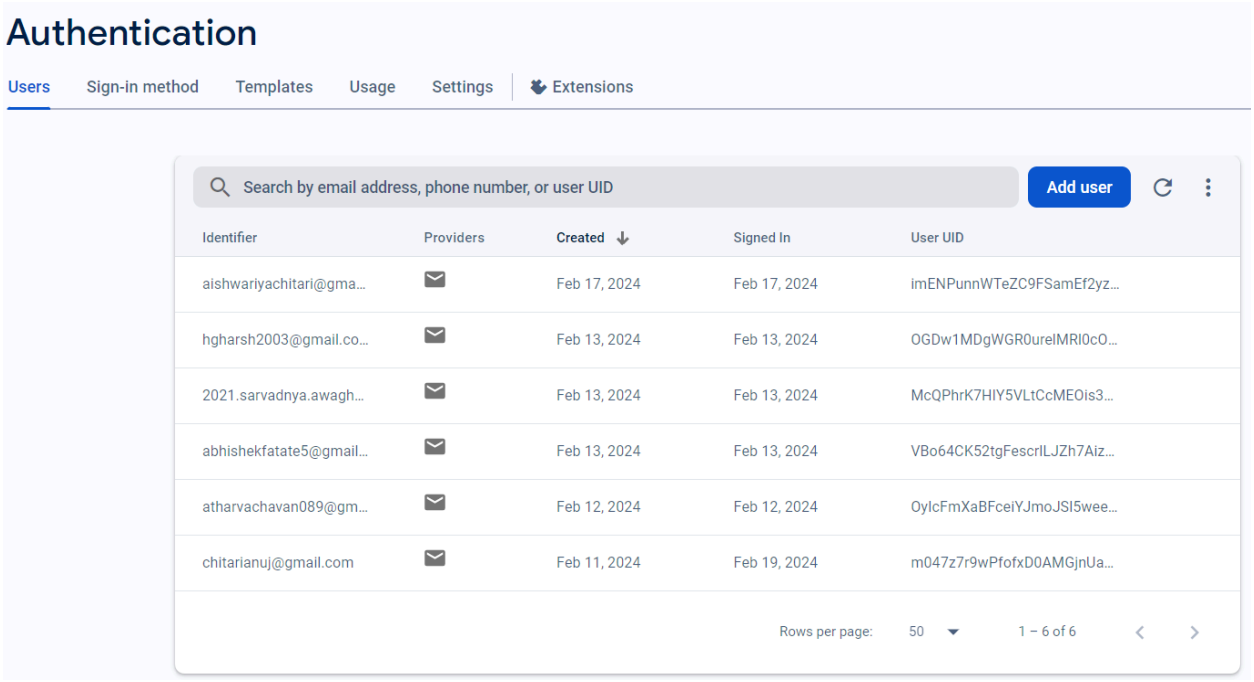
Output:

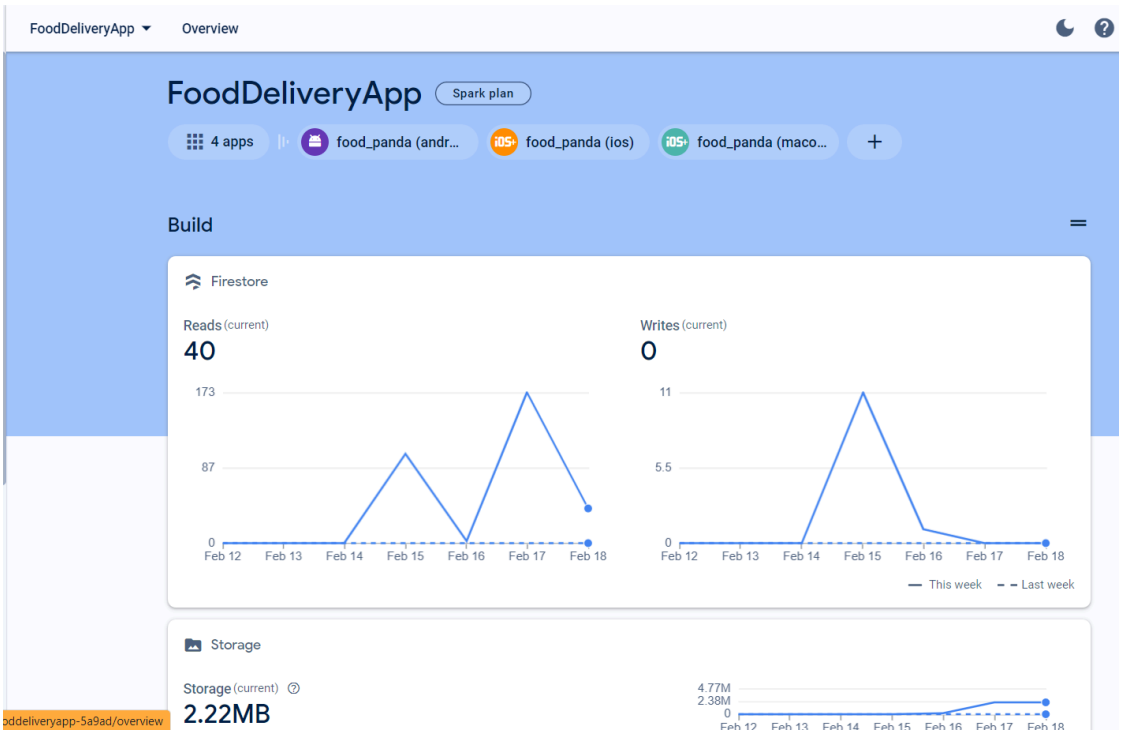
**Project Title: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**



**Project Title: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**

**Project Title: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**



# 2nd DB part

**add_food.dart**

```
import 'dart:io';

import 'package:firebase_storage/firebase_storage.dart';
import 'package:flutter/material.dart';
import 'package:food_panda/service/database.dart';
import 'package:food_panda/widget/widget_support.dart';
import 'package:image_picker/image_picker.dart';
import 'package:random_string/random_string.dart';
// import 'package:sample_flutter/widget/widget_support.dart';

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

  @override
  State<AddFood> createState() => _AddFoodState();
}

class _AddFoodState extends State<AddFood> {
  final List<String> fooditems = ['Ice-cream', 'Burger', 'Salad', 'Pizza'];
  String? value;
  TextEditingController namecontroller = new TextEditingController();
  TextEditingController pricecontroller = new TextEditingController();
  TextEditingController detailcontroller = new TextEditingController();
  final ImagePicker _picker = ImagePicker();
  File? selectedImage;

  Future getImage() async {
    var image = await _picker.pickImage(source: ImageSource.gallery);
    selectedImage = File(image!.path);
    setState(() {});
  }

  uploadItem() async {
    if (selectedImage != null &&
```

```
      namecontroller.text != "" &&
      pricecontroller.text != "" &&
      detailcontroller.text != "") {
    String addId = randomAlphaNumeric(10);
    Reference firebaseStorageRef =
        FirebaseStorage.instance.ref().child("blogImages").child(addId);
    final UploadTask task = firebaseStorageRef.putFile(selectedImage!);

    var downloadUrl = await (await task).ref.getDownloadURL();

    Map<String, dynamic> addItem = {
      "Image": downloadUrl,
      "Name": namecontroller.text,
      "Price": pricecontroller.text,
      "Detail": detailcontroller.text
    };
    await DatabaseMethods().addFoodItem(addItem, value!).then((value) {
      ScaffoldMessenger.of(context).showSnackBar((SnackBar(
          backgroundColor: Colors.redAccent,
          content: Text(
            "Food item added successfully",
            style: TextStyle(fontSize: 20.0),
          ))));
    });
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      leading: GestureDetector(
          onTap: () {
            Navigator.pop(context);
          },
          child: Icon(
            Icons.arrow_back_ios_new_outlined,
            color: Color(0xFF373866),
          )),
      centerTitle: true,
      title: Text(
        "Add Item",
        style: AppWidget.HeadlineTextFieldStyle(),
      ),
```

```
),
body: SingleChildScrollView(
  child: Container(
    margin:
        EdgeInsets.only(left: 20.0, right: 20.0, top: 20.0, bottom: 50.0),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(
          "Upload the Item Picture",
          style: AppWidget.semiBoldTextFieldStyle(),
        ),
        SizedBox(
          height: 20.0,
        ),
        selectedImage == null
            ? GestureDetector(
                onTap: () {
                  getImage();
                },
                child: Center(
                  child: Material(
                    elevation: 4.0,
                    borderRadius: BorderRadius.circular(20),
                    child: Container(
                      width: 150,
                      height: 150,
                      decoration: BoxDecoration(
                        border:
                            Border.all(color: Colors.black, width: 1.5),
                        borderRadius: BorderRadius.circular(20),
                      ),
                      child: Icon(
                        Icons.camera_alt_outlined,
                        color: Colors.black,
                      ),
                    ),
                  ),
                ),
              )
            : Center(
                child: Material(
                  elevation: 4.0,
                  borderRadius: BorderRadius.circular(20),
```

```
            child: Container(
              width: 150,
              height: 150,
              decoration: BoxDecoration(
                border: Border.all(color: Colors.black, width: 1.5),
                borderRadius: BorderRadius.circular(20),
              ),
              child: Image.file(
                selectedImage!,
                fit: BoxFit.cover,
              ),
            ),
          ),
        ),
      ),
SizedBox(
  height: 30.0,
),
Text(
  "Item Name",
  style: AppWidget.semiBoldTextFieldStyle(),
),
SizedBox(
  height: 10.0,
),
Container(
  padding: EdgeInsets.symmetric(horizontal: 20.0),
  width: MediaQuery.of(context).size.width,
  decoration: BoxDecoration(
      color: Color(0xFFececf8),
      borderRadius: BorderRadius.circular(10)),
  child: TextField(
    controller: namecontroller,
    decoration: InputDecoration(
        border: InputBorder.none,
        hintText: "Enter Item Name",
        hintStyle: AppWidget.LightTextFieldStyle()),
  ),
),
SizedBox(
  height: 30.0,
),
Text(
  "Item Price",
  style: AppWidget.semiBoldTextFieldStyle(),
```

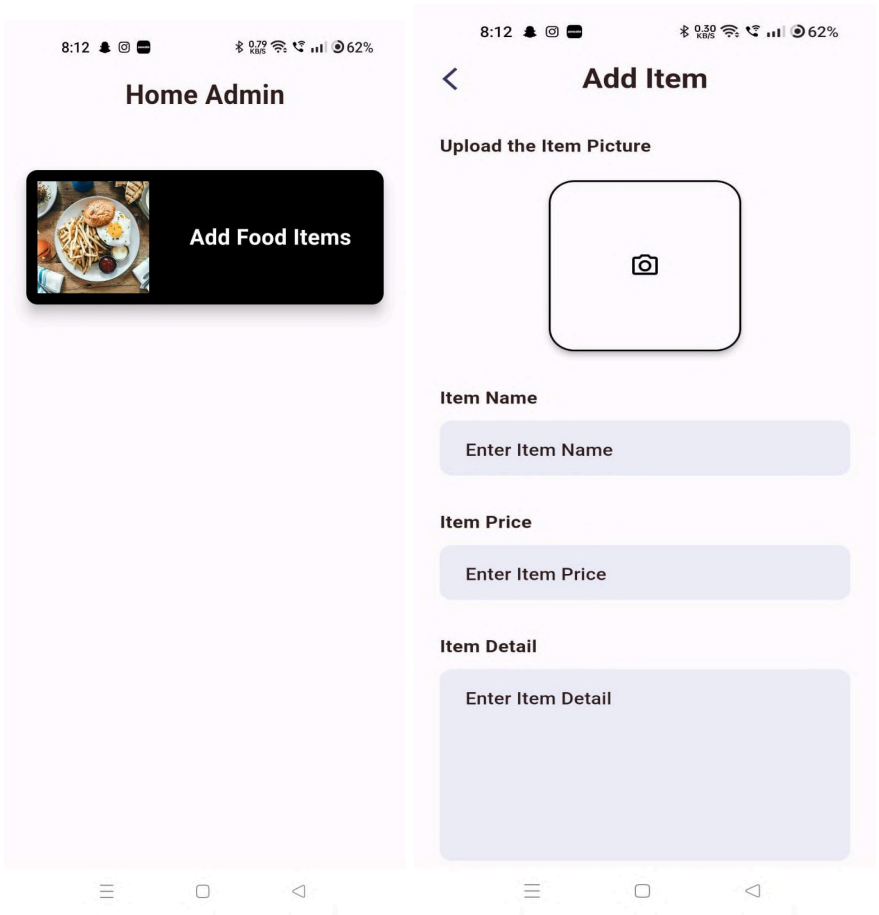
```
),
SizedBox(
  height: 10.0,
),
Container(
  padding: EdgeInsets.symmetric(horizontal: 20.0),
  width: MediaQuery.of(context).size.width,
  decoration: BoxDecoration(
      color: Color(0xFFececf8),
      borderRadius: BorderRadius.circular(10)),
  child: TextField(
    controller: pricecontroller,
    decoration: InputDecoration(
        border: InputBorder.none,
        hintText: "Enter Item Price",
        hintStyle: AppWidget.LightTextFieldStyle()),
  ),
),
SizedBox(
  height: 30.0,
),
Text(
  "Item Detail",
  style: AppWidget.semiBoldTextFieldStyle(),
),
SizedBox(
  height: 10.0,
),
Container(
  padding: EdgeInsets.symmetric(horizontal: 20.0),
  width: MediaQuery.of(context).size.width,
  decoration: BoxDecoration(
      color: Color(0xFFececf8),
      borderRadius: BorderRadius.circular(10)),
  child: TextField(
    maxLines: 6,
    controller: detailcontroller,
    decoration: InputDecoration(
        border: InputBorder.none,
        hintText: "Enter Item Detail",
        hintStyle: AppWidget.LightTextFieldStyle()),
  ),
),
SizedBox(
```

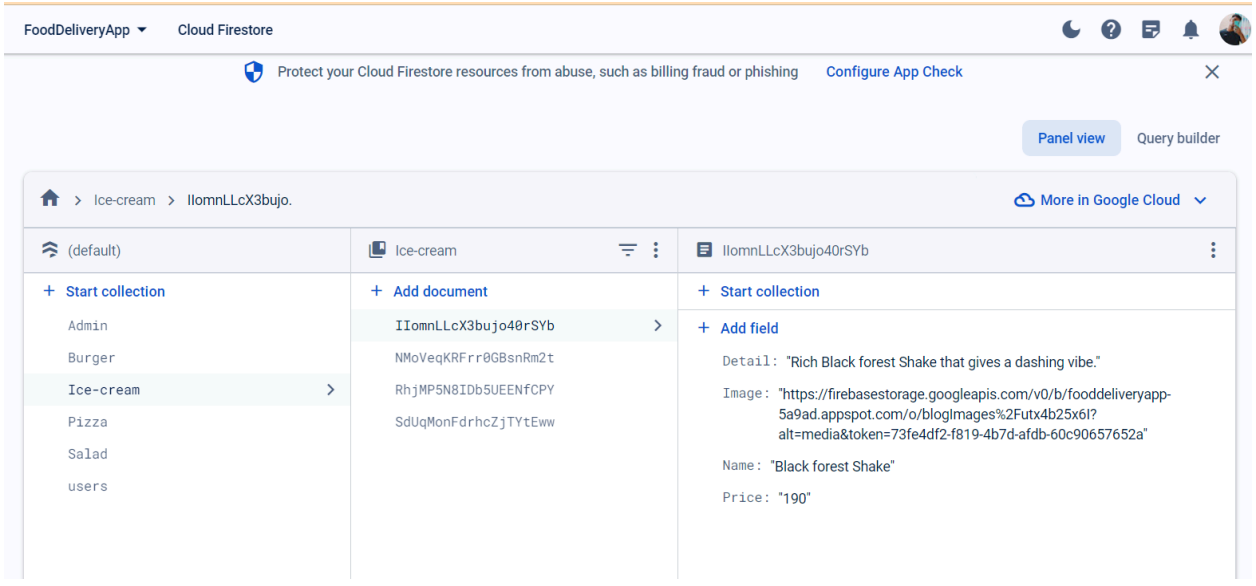
```
  height: 20.0,
),
Text(
  "Select Category",
  style: AppWidget.semiBoldTextFieldStyle(),
),
SizedBox(
  height: 20.0,
),
Container(
  padding: EdgeInsets.symmetric(horizontal: 10.0),
  width: MediaQuery.of(context).size.width,
  decoration: BoxDecoration(
      color: Color(0xFFececf8),
      borderRadius: BorderRadius.circular(10)),
  child: DropdownButtonHideUnderline(
      child: DropdownButton<String>(
    items: fooditems
        .map((item) => DropdownMenuItem<String>(
            value: item,
            child: Text(
              item,
              style:
                  TextStyle(fontSize: 18.0, color: Colors.black),
            )))
        .toList(),
    onChanged: ((value) => setState(() {
          this.value = value;
        })),
    dropdownColor: Colors.white,
    hint: Text("Select Category"),
    iconSize: 36,
    icon: Icon(
      Icons.arrow_drop_down,
      color: Colors.black,
    ),
    value: value,
  )),
),
SizedBox(
  height: 30.0,
),
GestureDetector(
  onTap: () {
```

```
            uploadItem();
          },
          child: Center(
            child: Material(
              elevation: 5.0,
              borderRadius: BorderRadius.circular(10),
              child: Container(
                padding: EdgeInsets.symmetric(vertical: 5.0),
                width: 150,
                decoration: BoxDecoration(
                    color: Colors.black,
                    borderRadius: BorderRadius.circular(10)),
                child: Center(
                  child: Text(
                    "Add",
                    style: TextStyle(
                        color: Colors.white,
                        fontSize: 22.0,
                        fontWeight: FontWeight.bold),
                  ),
                ),
              ),
            ),
          ),
        )
      ],
    ),
  ),
),
);
}
}
```

## OUTPUT:

**Project Title: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**




## home.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:food_panda/pages/details.dart';
```

```
import 'package:food_panda/service/database.dart';
import 'package:food_panda/widget/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;
//Syream is used to get all the items in the form of a stream which can be used to handle it
easily
  Stream? fooditemStream;

  ontheLoad() async {
    fooditemStream = await DatabaseMethods().getFoodItem("Pizza");
    setState(() {});
  }

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

  Widget allItemsVertically() {
    return StreamBuilder(
        stream: fooditemStream,
        builder: (context, AsyncSnapshot snapshot) {
          return snapshot.hasData
              ? ListView.builder(
                  padding: EdgeInsets.zero,
                  itemCount: snapshot.data.docs.length,
                  shrinkWrap: true,
                  scrollDirection: Axis.vertical,
                  itemBuilder: (context, index) {
                    DocumentSnapshot ds = snapshot.data.docs[index];
                    return GestureDetector(
                      onTap: () {
                        Navigator.push(
                            context,
                            MaterialPageRoute(
```

```
            builder: (context) => Details(
                  detail: ds["Detail"],
                  name: ds["Name"],
                  price: ds["Price"],
                  image: ds["Image"],
                )));
      },
      child: Container(
        margin: const EdgeInsets.only(right: 10.0, bottom: 20),
        child: Material(
          elevation: 5.0,
          borderRadius: BorderRadius.circular(20),
          child: Container(
            padding: const EdgeInsets.all(5),
            child: Row(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                ClipRRect(
                  borderRadius: BorderRadius.circular(20),
                  child: Image.network(
                    ds["Image"],
                    height: 120,
                    width: 120,
                    fit: BoxFit.cover,
                  ),
                ),
                const SizedBox(
                  width: 20.0,
                ),
                Column(
                  children: [
                    Container(
                        width:
                            MediaQuery.of(context).size.width /
                                2,
                        child: Text(
                          ds["Name"],
                          style: AppWidget
                              .semiBoldTextFieldStyle(),
                        )),
                    const SizedBox(
                      height: 5.0,
                    ),
                    Container(
```

```
                                width:
                                    MediaQuery.of(context).size.width /
                                        2,
                                child: Text(
                                  "Honey goot cheese",
                                  style:
                                      AppWidget.LightTextFieldStyle(),
                                )),
                            const SizedBox(
                              height: 5.0,
                            ),
                            Container(
                                width:
                                    MediaQuery.of(context).size.width /
                                        2,
                                child: Text(
                                  "\$" + ds["Price"],
                                  style: AppWidget
                                      .semiBoldTextFieldStyle(),
                                ))
                          ],
                        )
                      ],
                    ),
                  ),
                ),
              );
            })
          : CircularProgressIndicator();
    });
}

Widget allItems() {
  return StreamBuilder(
      stream: fooditemStream,
      builder: (context, AsyncSnapshot snapshot) {
        return snapshot.hasData
            ? ListView.builder(
                padding: EdgeInsets.zero,
                itemCount: snapshot.data.docs.length,
                shrinkWrap: true,
                scrollDirection: Axis.horizontal,
                itemBuilder: (context, index) {
```

```
DocumentSnapshot ds = snapshot.data.docs[index];
return GestureDetector(
 onTap: () {
  Navigator.push(
     context,
     MaterialPageRoute(
        builder: (context) => Details(
             detail: ds["Detail"],
             name: ds["Name"],
             price: ds["Price"],
             image: ds["Image"],
           )));
 },
 child: Container(
  margin: const EdgeInsets.all(4),
  child: Material(
   elevation: 5.0,
   borderRadius: BorderRadius.circular(20),
   child: Container(
    padding: const EdgeInsets.all(14),
    child: Column(
       crossAxisAlignment: CrossAxisAlignment.start,
       children: [
        //ClipRRect ,This is used to add border radius to the image
        ClipRRect(
         borderRadius: BorderRadius.circular(20),
         //This gets the image stored in the firebase
         child: Image.network(
          ds["Image"],
          height: 150,
          width: 150,
          fit: BoxFit.cover,
         ),
        ),
        Text(ds["Name"],
           style:
              AppWidget.semiBoldTextFieldStyle()),
        const SizedBox(
         height: 5.0,
        ),
        Text("Fresh and Healthy",
           style: AppWidget.LightTextFieldStyle()),
        const SizedBox(
         height: 5.0,
```

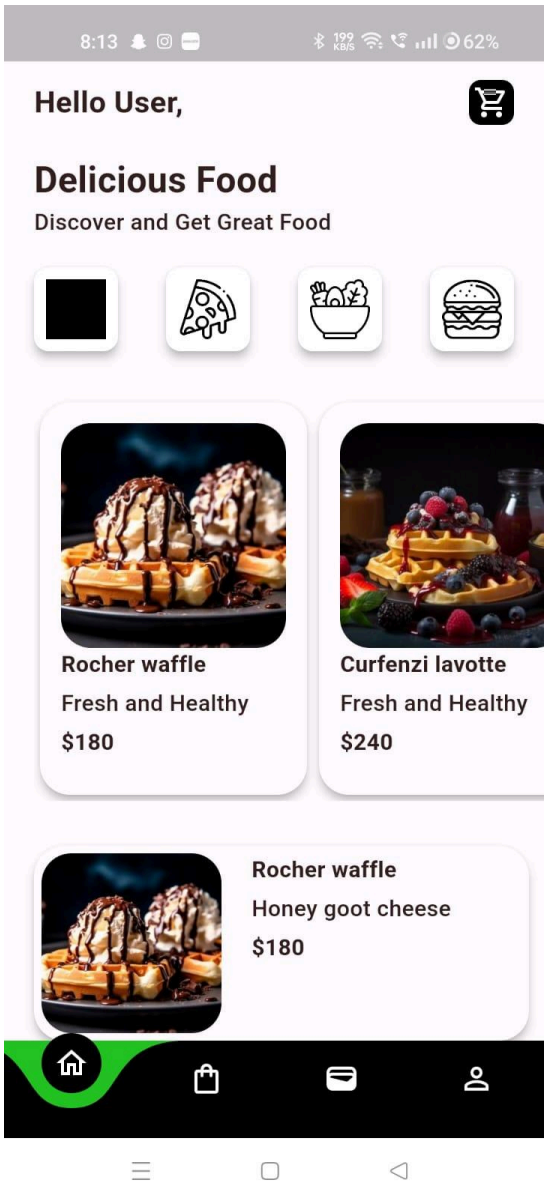
```
                        ),
                        Text(
                          "\$" + ds["Price"],
                          style: AppWidget.semiBoldTextFieldStyle(),
                        )
                      ]),
                ),
              ),
            ),
          );
        })
        : CircularProgressIndicator();
    });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: SingleChildScrollView(
      scrollDirection: Axis.vertical,
      child: Container(
        margin: const EdgeInsets.only(top: 50.0, left: 20.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                Text("Hello User,", style: AppWidget.boldTextFieldStyle()),
                Container(
                  margin: const EdgeInsets.only(right: 20.0),
                  padding: const EdgeInsets.all(3),
                  decoration: BoxDecoration(
                      color: Colors.black,
                      borderRadius: BorderRadius.circular(8)),
                  child: const Icon(
                    Icons.shopping_cart_outlined,
                    color: Colors.white,
                  ),
                )
              ],
            ),
            const SizedBox(
              height: 20.0,
```

```
            ),
            Text("Delicious Food", style: AppWidget.HeadlineTextFieldStyle()),
            Text("Discover and Get Great Food",
               style: AppWidget.LightTextFieldStyle()),
            const SizedBox(
              height: 20.0,
            ),
            Container(
               margin: const EdgeInsets.only(right: 20.0),
               child: showItem()),
            const SizedBox(
              height: 30.0,
            ),
            Container(height: 270, child: allItems()),
                    const SizedBox(
              height: 30.0,
            ),
            allItemsVertically(),
            const SizedBox(
              height: 30.0,
            ),
          ],
        ),
      ),
    ),
  );
}

Widget showItem() {
  return Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [
      GestureDetector(
        onTap: () async {
          icecream = true;
          pizza = false;
          salad = false;
          burger = false;
          fooditemStream = await DatabaseMethods().getFoodItem("Ice-cream");
          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: const EdgeInsets.all(8),
        child: Image.asset(
          "images/shake.png",
          height: 40,
          width: 40,
          fit: BoxFit.cover,
          color: icecream ? Colors.white : Colors.black,
        ),
      ),
    ),
  ),
  GestureDetector(
    onTap: () async {
      icecream = false;
      pizza = true;
      salad = false;
      burger = false;
      fooditemStream = await DatabaseMethods().getFoodItem("Pizza");
      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: const EdgeInsets.all(8),
        child: Image.asset(
          "images/pizza.png",
          height: 40,
          width: 40,
          fit: BoxFit.cover,
          color: pizza ? Colors.white : Colors.black,
        ),
      ),
    ),
  ),
  GestureDetector(
    onTap: () async {
```

```
          icecream = false;
          pizza = false;
          salad = true;
          burger = false;
          fooditemStream = await DatabaseMethods().getFoodItem("Salad");
          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: const EdgeInsets.all(8),
            child: Image.asset(
              "images/salad.png",
              height: 40,
              width: 40,
              fit: BoxFit.cover,
              color: salad ? Colors.white : Colors.black,
            ),
          ),
        ),
      ),
      GestureDetector(
        onTap: () async {
          icecream = false;
          pizza = false;
          salad = false;
          burger = true;
          fooditemStream = await DatabaseMethods().getFoodItem("Burger");
          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: const EdgeInsets.all(8),
            child: Image.asset(
              "images/burger.png",
```

```
            height: 40,
            width: 40,
            fit: BoxFit.cover,
            color: burger ? Colors.white : Colors.black,
          ),
        ),
      ),
    ),
  ],
);
```



# Conclusion:

In this experiment, we have successfully connected firebase database and authenticated using google signin and email and password without flutter application successfully.

# 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. | 11 |
| Name | Anuj Chitari |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO4: Understand various PWA frameworks and their requirements |
| Grade: | |

# MAD and PWA Lab ]

# Experiment - 7

**Name: Anuj Chitari**
**Class: D15A Roll no:11**

**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.

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.

## Code:

```
import Footer from './components/footer/Footer';
import Hero from './components/hero/Hero';
import Navbar from './components/navbar/Navbar';
import Newsletter from './components/newsletter/Newsletter';
import PopularProperties from
'./components/popularProperties/PopularProperties'; import Signin
from './components/signin/Signin';
import Signup from './components/signup/Signup';
import Properties from './components/properties/Properties';
import PropertyDetail from
'./components/propertyDetail/PropertyDetail'; import {
useSelector } from 'react-redux'
import { Routes, Route, Navigate, useLocation } from
'react-router-dom' import { useEffect } from 'react';
import EditProperty from
'./components/editProperty/EditProperty'; import
Yachts from './components/yachts/Yachts';
import YachtDetails from
'./components/yachtDetails/YachtDetails'; import
CreateYacht from
'./components/createYacht/CreateYacht'; import
YachtEdit from './components/yachtEdit/YachtEdit';
import MyProfile from './components/myProfile/MyProfile';
import UpdateProfile from
'./components/updateProfile/UpdateProfile'; import
'./App.css';
import NotFound from './components/notFound/NotFound';
import Layout from './components/Layout';
```

```
import Services from './components/Services/Services';


function App() {
  const { user } = useSelector((state) => state.auth)
  const url = useLocation().pathname

  useEffect(() => {
    url && window.scrollTo(0, 0)
  }, [url])

  return (
    <div>
      <Routes>
        <Route path='/' element={
          <Layout title="DormDine || Home">
            <Navbar />
            <Hero />
            <PopularProperties />
            <Services/>
            <Newsletter />
            <Footer />
          </Layout>
        } />
        <Route path='/signup' element={!user ? <Signup /> :
        <Navigate to='/' />} /> <Route path='/signin'
        element={!user ? <Signin /> : <Navigate to='/' />} />
        <Route path='/properties' element={
          <>
            <Navbar />
            <Properties />
            <Footer />
          </>
        } />
        <Route path='/yachts'
          element={user ? <>
            <Navbar />
            <Yachts />
            <Footer />
          </>
          : <Navigate to='/signin' />} />
```

```
<Route path='/yacht/:id'
  element={user ? <>
    <Navbar />
    <YachtDetails />
    <Footer />
  </>
  : <Navigate to='/signin' />} />
<Route path='/create-yacht'
  element={user ? <>
    <Navbar />
    <CreateYacht />
    <Footer />
  </>
  : <Navigate to='/signin' />} />
<Route path='/yacht-edit/:id'
  element={user ? <>
    <Navbar />
    <YachtEdit />
    <Footer />
  </>
  : <Navigate to='/signin' />} />
<Route
  path='/propertyDetail/:id'
  element={ <>
    <Navbar />
    <PropertyDetail />
    <Footer />
  </>
} />
<Route
  path='/editproperty/:id'
  element={ user ?
    <>
      <Navbar />
      <EditProperty />
      <Footer />
    </>
    : <Navigate to='/signin' />
} />
<Route path='/my-profile' element={
  user ?
```

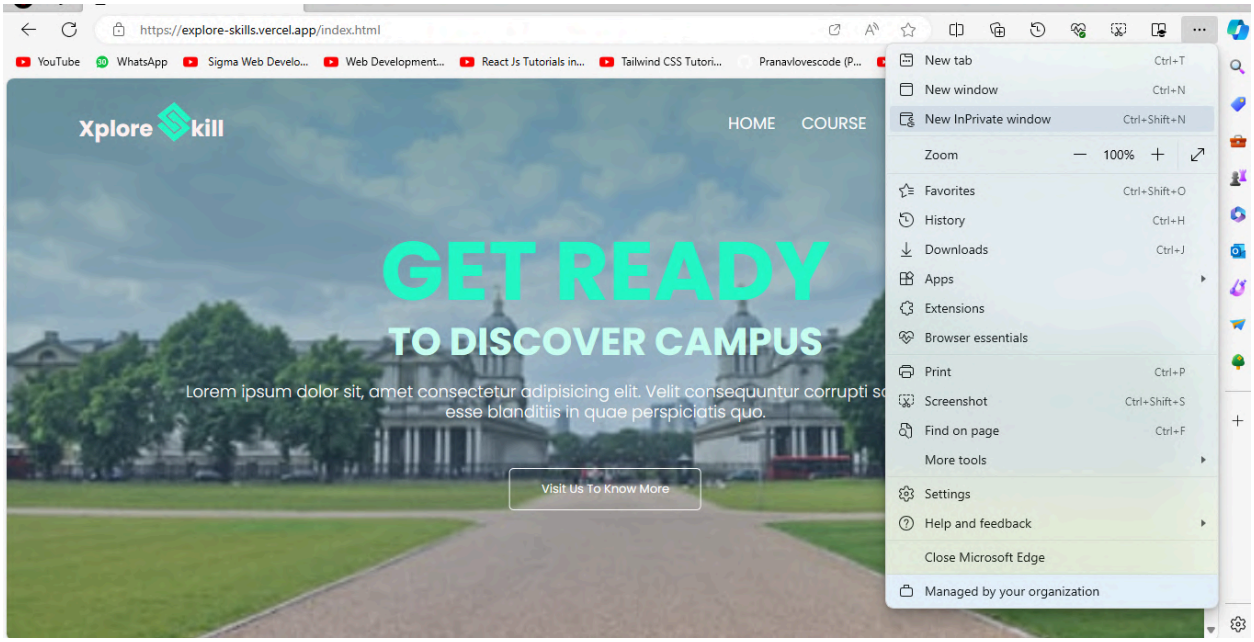
```
          <>
            <Navbar />
            <MyProfile />
            <Footer />
          </>
          : <Navigate to='/signin' />
        } />
        <Route path='/update-profile' element={
          user ?
          <>
            <Navbar />
            <UpdateProfile />
            <Footer />
          </>
          : <Navigate to='/signin' />
        } />
        <Route path='*' element={
          <>
            <Navbar />
            <NotFound />
            <Footer />
          </>
        } />
      </Routes>
    </div>
  );
}

export default App;
```

Open folder in VS code and click go live at

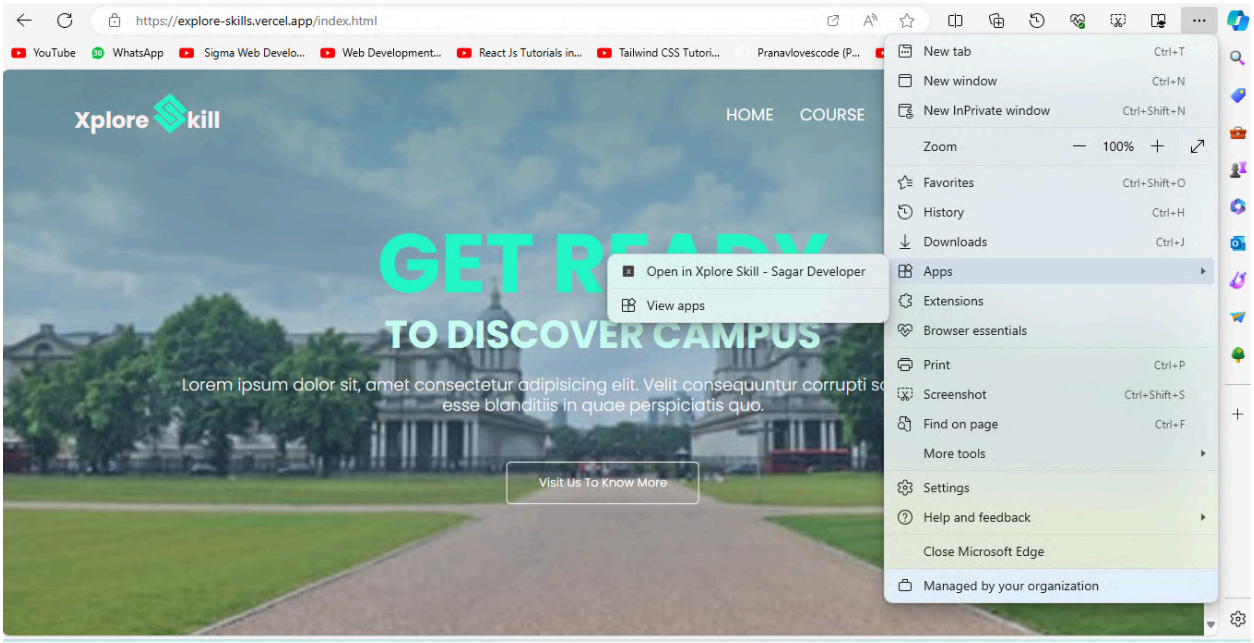
bottom right corner Open your hosted site on

Microsoft Edge

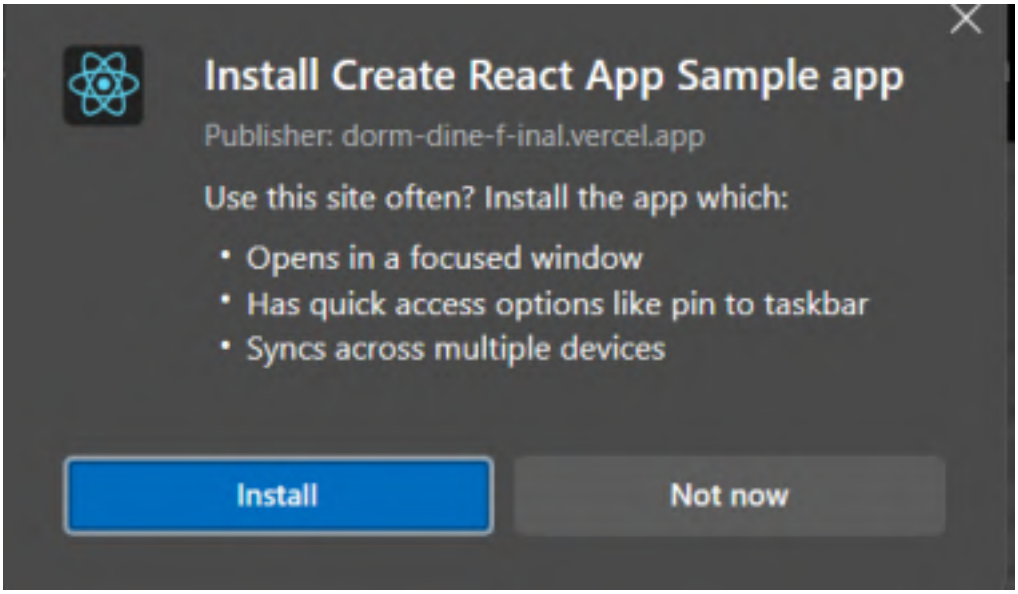
**Project Title: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**




Click on 3 dots
click on Apps
select install app option



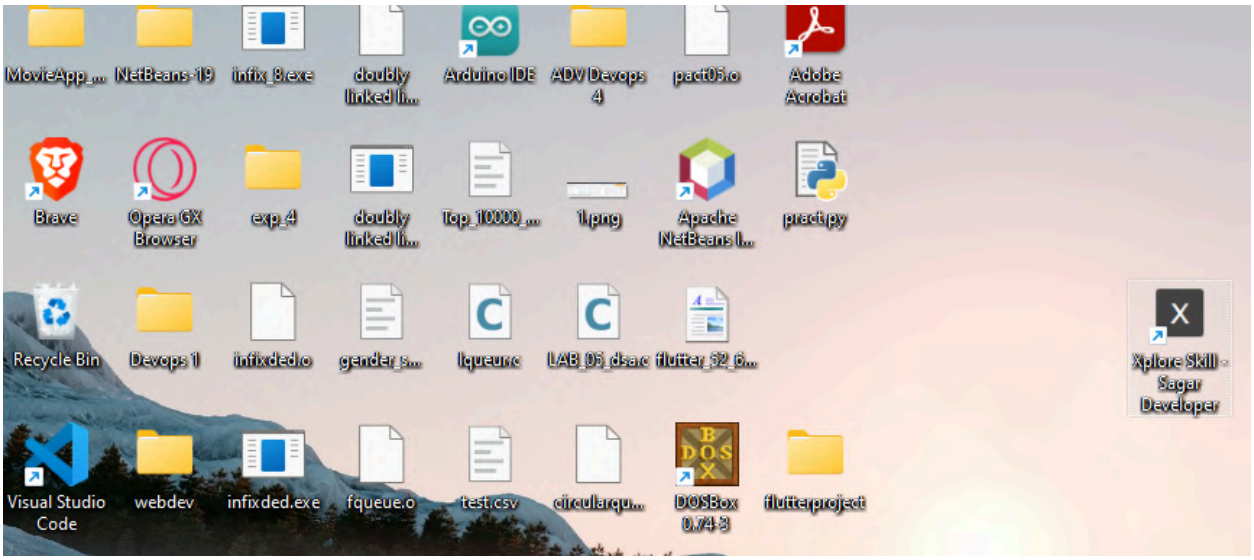

Click on Install



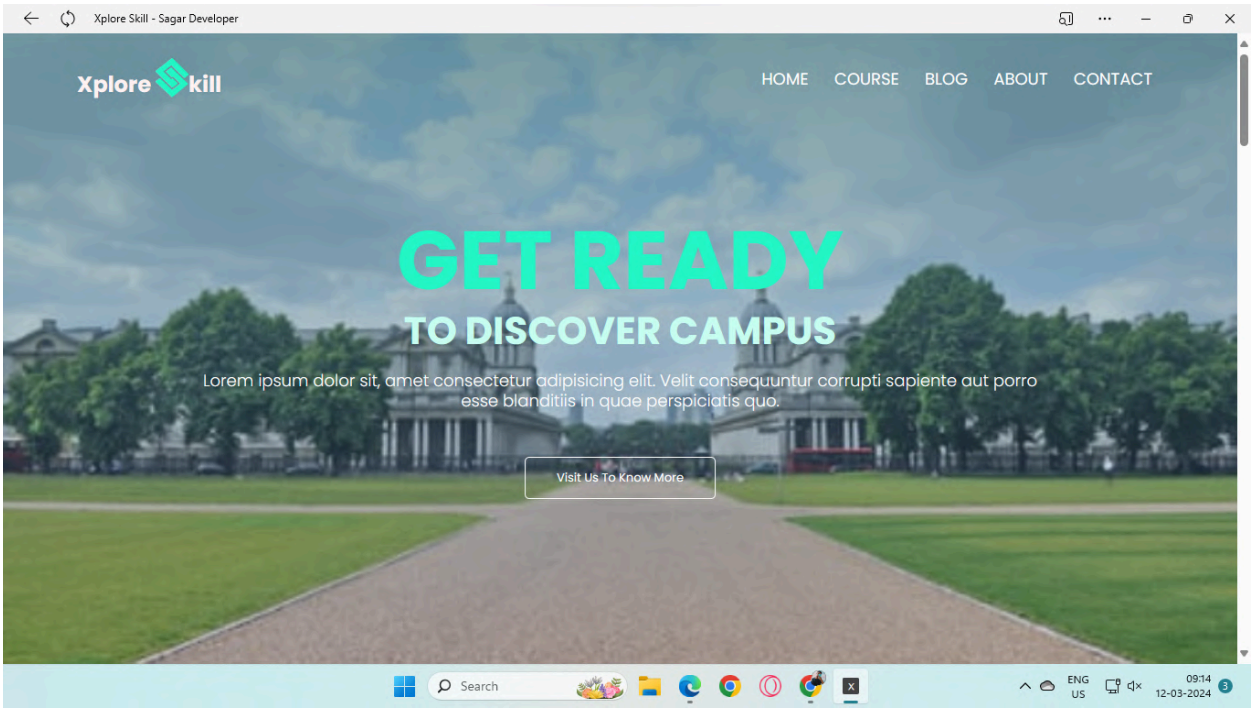

App icon will appear at the bottom

Output:

Desktop App Created Successfully.




Open Desktop App:




# Conclusion:

In this experiment, we have successfully created a basic progressive web app of our web page and installed it in our desktop successfully.

# MAD & PWA Lab
# Journal

| | |
|---|---|
| Experiment No. | 08 |
| Experiment Title. | To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA |
| Roll No. | 11 |
| Name | Anuj Chitari |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO5: Design and Develop a responsive User Interface by applying PWA Design techniques |
| Grade: | |

# MAD and PWA Lab

## Name: Anuj Chitari Class: D15A Roll no:11

## Experiment - 8

## Aim: To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA.

## Theory:

Service Worker

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

Things to note about Service Worker:

A service worker is a programmable network proxy that lets you control how network requests from your page are handled.

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

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.

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:

```
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-work
er.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).

**Code:**

## index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>E-commerce PWA</title>
  <link rel="stylesheet" href="main.css">
</head>
<body>
  <h1>Welcome to Our E-commerce Store</h1>
  <!-- E-commerce content goes here -->
  <script src="app.js"></script>
</body>
</html>
```

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

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Xplore Skill - Sagar Developer</title>
    <link rel="stylesheet" href="main.css">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css">
    <link rel="stylesheet"

href="https://fonts.googleapis.com/css2?family=Poppins:wght@100;200;300;400;600;
700&display=swap">
</head>
<!-- Google tag (gtag.js) -->
<script async
src="https://www.googletagmanager.com/gtag/js?id=G-F8NS5KXZLN"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'G-F8NS5KXZLN');
</script>
<body>
    <script src="app.js"></script>
    <!-- Created BY - Sagar Developer -->
    <section class="header">
        <nav>
            <a href="index.html" class="logo">Xplore
                <i class="fab fa-staylinked"></i>kill
            </a>
            <div class="nav-links" id="navLinks">
                <!-- Reposnive bar open and close -->
                <i class="fa fa-times" onclick="hideMenu()"></i>
                <ul>
                    <li><a href="index.html">Home</a></li>
                    <li><a href="course.html">Course</a></li>
                    <li><a href="blog.html">Blog</a></li>
                    <li><a href="about.html">About</a></li>
                    <li><a href="contact.html">Contact</a></li>
```

```
            </ul>
          </div>
          <i class="fa fa-bars" onclick="showMenu()"></i>

        </nav>

        <div class="text_box">
          <h2>GET READY</h2>
          <p id="headtext">TO DISCOVER CAMPUS</p>
          <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Velit consequuntur
corrupti sapiente aut porro
            <br> esse blanditiis in quae perspiciatis quo.
          </p>
          <a href="#" class="hero_btn">Visit Us To Know More</a>
        </div>
    </section>

    <!-- Course Section Start -->
    <section class="course">
        <div class="PageBlock">
          <div class="verticalLine"></div>
          <div class="Clear"></div>
        </div>
        <h1>EXPLORE OUR 60+ <br> MAJOR PROGRAMS</h1>
        <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit.</p>

        <div class="row">
          <div class="course-col">
            <h3>Undergraduate Programs</h3>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Similique dolor
corporis, commodi nihil quas
              soluta labore quisquam impedi distinctio explicabo aut minima quos
pariatur unde aliquam earum
              laborum velit non.</p>
          </div>
          <div class="course-col">
            <h3>Graduate Programs</h3>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Similique dolor
corporis, commodi nihil quas
              soluta labore quisquam impedit distinctio explicabo aut minima quos
pariatur unde aliquam earum
              laborum velit non.</p>
          </div>
          <div class="course-col">
```

```
            <h3>Adult Learning & Degree Completion</h3>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Similique dolor
corporis, commodi nihil quas
                soluta labore quisquam impedit distinctio explicabo aut minima quos
pariatur unde aliquam earum
                laborum velit non.</p>
          </div>
        </div>
    </section>
    <!-- Course Section End -->

    <!-- Campus Section Start-->

    <section class="campus">
      <div class="PageBlock">
        <div class="verticalLine"></div>
        <div class="Clear"></div>
      </div>
      <h1>TAKE OUR VIRTUAL TOUR</h1>
      <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit.</p>

      <div class="row">
        <div class="campus-col">
          <img src="img/Campus1.png" alt="">
          <div class="layer">
            <h3>DELHI</h3>
          </div>
        </div>
        <div class="campus-col">
          <img src="img/Campus2.png" alt="">
          <div class="layer">
            <h3>HYDERABAD</h3>
          </div>
        </div>
        <div class="campus-col">
          <img src="img/Campus3.png" alt="">
          <div class="layer">
            <h3>MUMBAI</h3>
          </div>
        </div>
      </div>
    </section>
    <!-- Campus Section End -->
```

```
    <!-- Facilities Section Start -->
    <section class="facilities">
      <div class="PageBlock">
        <div class="verticalLine"></div>
        <div class="Clear"></div>
      </div>
      <h1>Our Facilities</h1>
      <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit.</p>

      <div class="row">
        <div class="facilities-col">
          <img src="img/libary.png" alt="">
          <h3>Best Libary</h3>
          <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Distinctio omnis
asperiores atque aperiam.
          </p>
        </div>
        <div class="facilities-col">
          <img src="img/playground.png" alt="">
          <h3>Athletics</h3>
          <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Distinctio omnis
asperiores atque aperiam.
          </p>
        </div>
        <div class="facilities-col">
          <img src="img/food.png" alt="">
          <h3>Tasty and Healthy Food</h3>
          <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Distinctio omnis
asperiores atque aperiam.
          </p>
        </div>
      </div>
    </section>
    <!-- Facilities Section End -->

    <!-- Testimonials Section Start -->
    <section class="testimonials">
      <div class="PageBlock">
        <div class="verticalLine"></div>
        <div class="Clear"></div>
      </div>
      <h1>What Our Student Says</h1>
      <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit.</p>
```

```
        <div class="row">
          <div class="testimonials-col">
            <img src="img/user.png" alt="oo..">
            <div>
              <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Modi maiores
in nostrum rerum voluptatem
                praesentium veritatis alias omnis voluptate nisi aperiam, voluptatum
quibusdam itaque a
                deserunt. In quia repellat maxime.</p>
              <h3>Student Name</h3>
              <i class="fa fa-star"></i>
              <i class="fa fa-star"></i>
              <i class="fa fa-star"></i>
              <i class="fa fa-star"></i>
              <i class="fa fa-star"></i>
            </div>
          </div>
          <div class="testimonials-col">
            <img src="img/user.png" alt="oo..">
            <div>
              <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Modi maiores
in nostrum rerum voluptatem
                praesentium veritatis alias omnis voluptate nisi aperiam, voluptatum
quibusdam itaque a
                deserunt. In quia repellat maxime.</p>
              <h3>Student Name</h3>
              <i class="fa fa-star"></i>
              <i class="fa fa-star"></i>
              <i class="fa fa-star"></i>
              <i class="fa fa-star"></i>
              <i class="fa fa-star-half-alt"></i>
            </div>
          </div>
        </div>
      </section>
      <!-- Testimonials Section End -->

      <!-- Call To Action Section Start -->
      <section class="cta">
        <h1>GET READY FOR A BRIGHT FUTURE</h1>
        <a href="contact.html" class="hero_btn">CONTACT US</a>
      </section>
      <!-- Call To Action Section End -->
```

```
    <!-- Footer Section Start -->
    <section class="footer">
        <hr>
        <h4>About Us</h4>
        <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Esse minima incidunt
odio nam facilis, eligendi
            Itaque fugiat cupiditate consectetur. Aliquid ab deserunt exercitationem, illum
molestiae dolorem.
        </p>

        <div class="icons">
            <i class="fab fa-facebook-f"></i>
            <i class="fab fa-instagram"></i>
            <i class="fab fa-twitter"></i>
            <i class="fab fa-linkedin"></i>
        </div>
        <p>Made with <i class="fas fa-heart"></i> by <a href="index.html">Sagar
Developer</a></p>
        <p>Copyright © 2021 <a href="index.html">Xplore Skill</a>. All Rights
Reserved</p>
    </section>
    <!-- Footer Section End -->

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

</html>
```

## main.css

```
html, body {
    margin: 0;
    font-family: arial,sans-serif;
    min-width: 900px;
    line-height: 14px;
    font-size: 14px;
  }

  * {
    box-sizing: border-box;
  }

  a {
    color: #0066c0;
  }
```

```
a:hover {
  color: #c45500;
}

#banner {
  background: #F6F6F6 url('../img/banner.jpg') no-repeat top left;
  height: 55px;
  background-size: 1920px;
  min-width: 1000px;
  display: block;
}

header {
  background-color: #232f3e;
  height: 99px;
}
```

## app.js

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

**service-worker.js**

```
// service-worker.js
```

```
const cacheName = 'ecommerce-pwa-v1';
```

```
const assetsToCache = [
    '/',
    '/index.html',
    '/main.css',
    '/app.js'
    // Add more files and assets here as needed
];



self.addEventListener('install', event => {
    event.waitUntil(
        caches.open(cacheName)
            .then(cache => {
                return cache.addAll(assetsToCache);
            })
    );
});
```

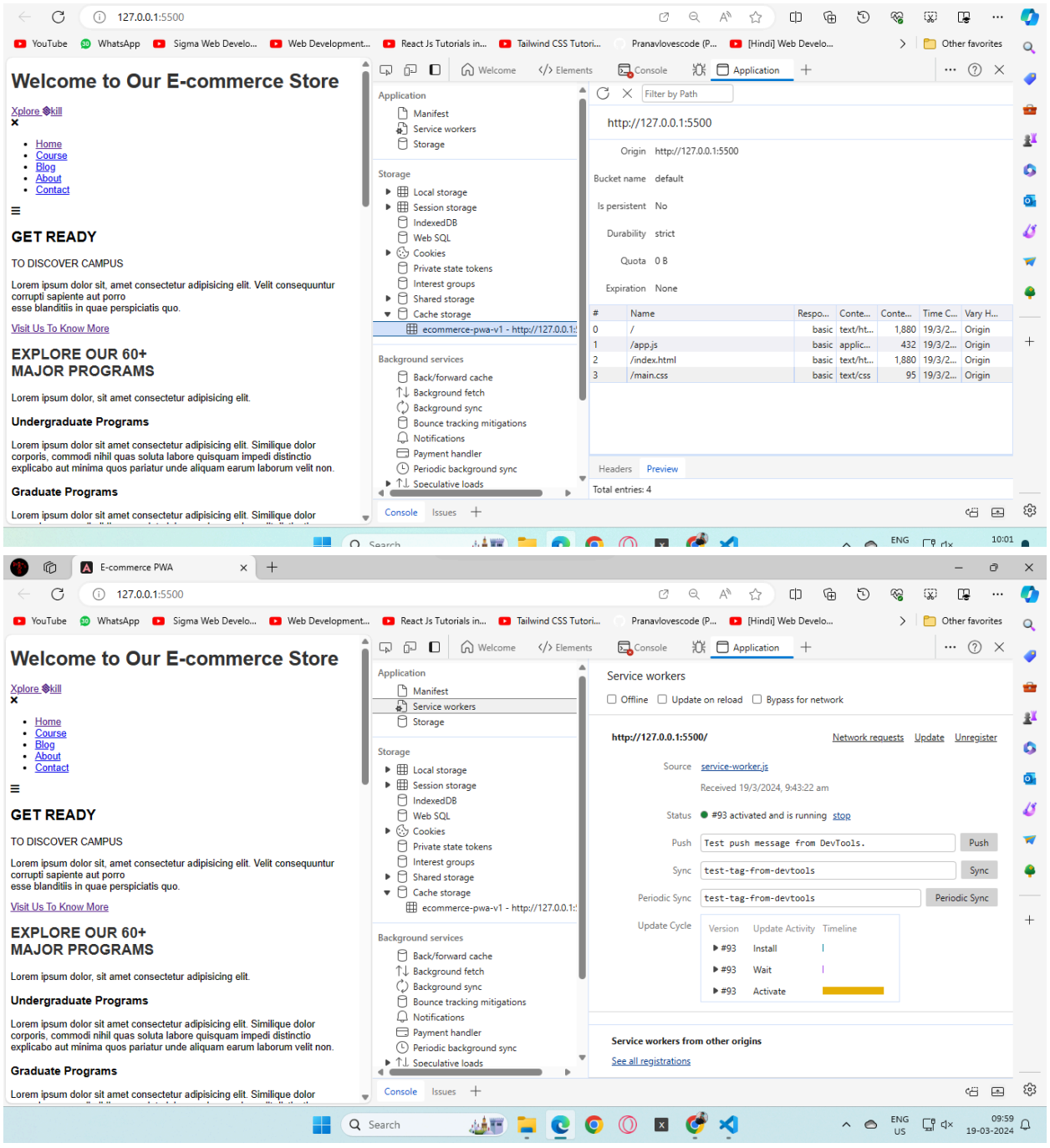
```
self.addEventListener('activate', event => {
  event.waitUntil(
    caches.keys().then(cacheNames => {
      return Promise.all(
        cacheNames.filter(name => {
          return name !== cacheName;
        }).map(name => {
          return caches.delete(name);
        })
      );
```

**Steps for Execution**

**Create a folder and put all 4 files main.css , service-worker.js, app.js, index.html open visual studio**
**install extension Live server**
**open folder in visual studio open index.html**
**on bottom right corner click go Live**
**it will open html page in browser**
**go to developer tools**

**Output:**

**Project Title: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**




# Conclusion:

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

# MAD & PWA Lab
# Journal

| | |
|---|---|
| Experiment No. | 09 |
| Experiment Title. | To implement Service worker events like fetch, sync and push for E-commerce PWA |
| Roll No. | 11 |
| Name | Anuj Chitari |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO5: Design and Develop a responsive User Interface by applying PWA Design techniques |
| Grade: | |

# MAD and PWA Lab

## Name: Anuj Chitari Class: D15A Roll no:11

# Experiment - 9

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

**Theory**:

**Service Worker**

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

Things to note about Service Worker:

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

**Fetch Event**

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

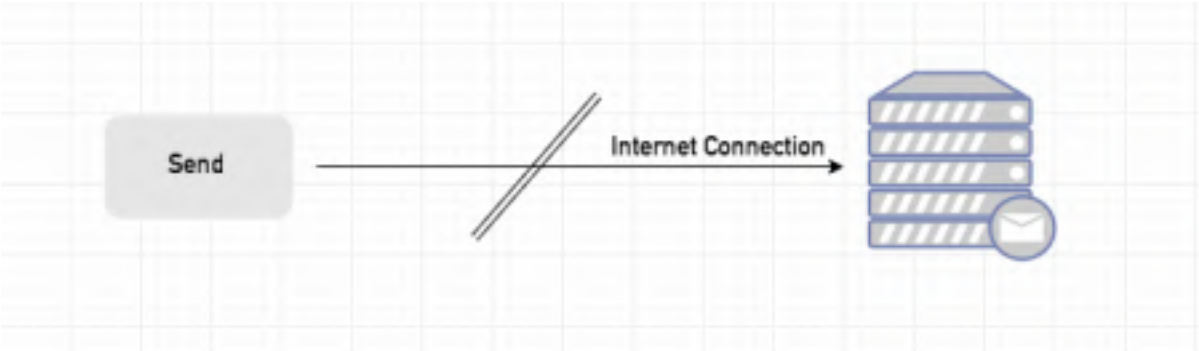
Of course, you can use many different methods but you can find in the following example a “cache first” and “network first” approach. In this example, if the request’s and current location’s origin are the same (Static content is requested.), this is called “cacheFirst” but if you request a targeted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

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

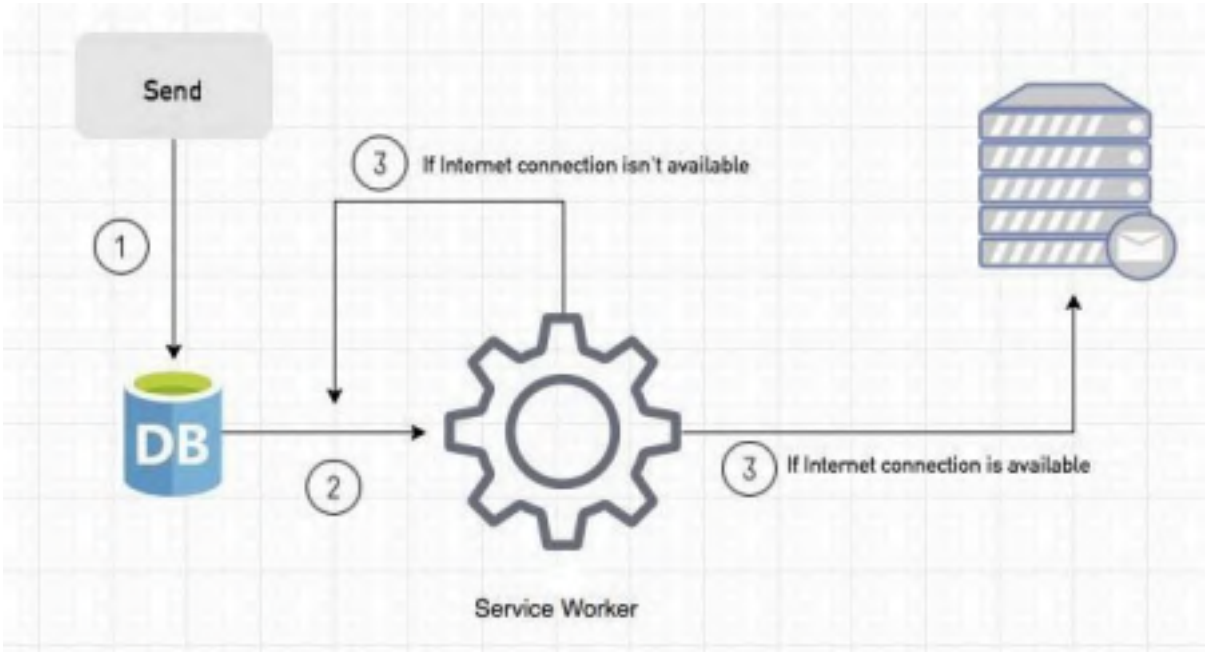
**Sync Event**

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

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




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




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

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

**Push Event**

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

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

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

## Code:

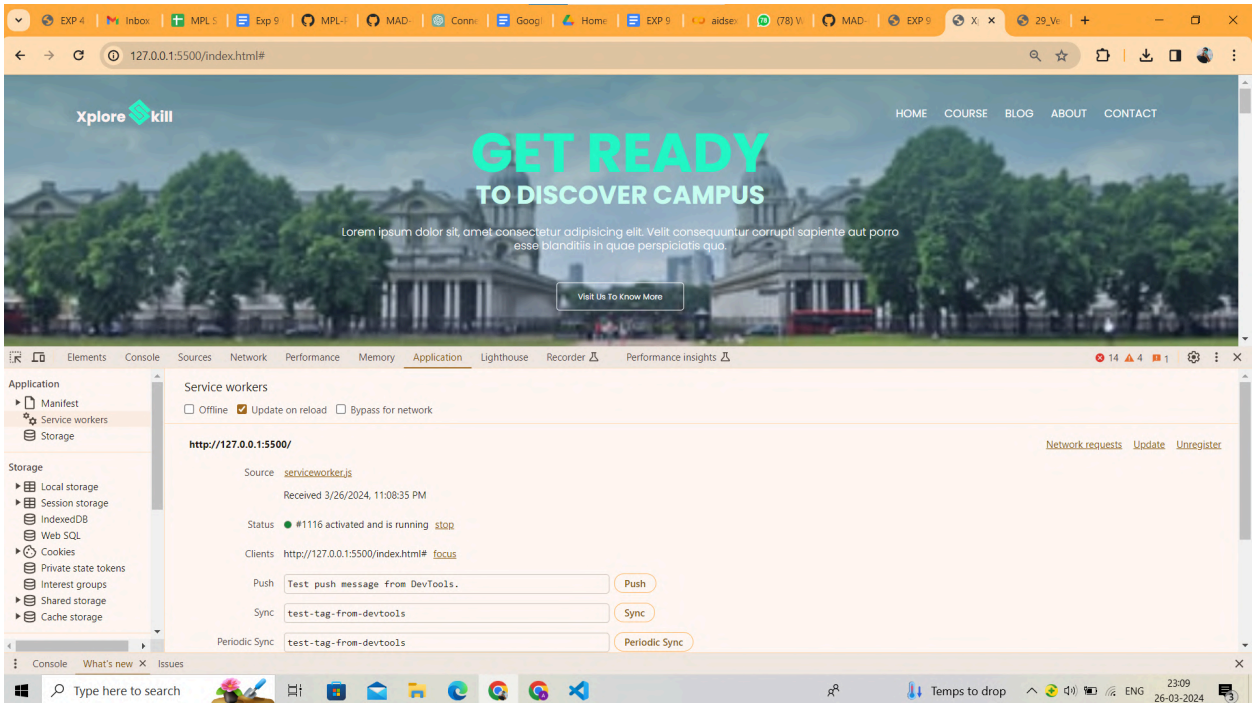
```
//serviceworker
self.addEventListener("install", function (event) {
  event.waitUntil(preLoad());
});
self.addEventListener("fetch", function (event) {
  event.respondWith(
    checkResponse(event.request).catch(function () {
      console.log("Fetch from cache successful!");
      return returnFromCache(event.request);
    })
  );
  console.log("Fetch successful!");
  event.waitUntil(addToCache(event.requesst));
});
self.addEventListener("sync", (event) => {
  if (event.tag === "syncMessage") {
    console.log("Sync successful!");
  }
});
self.addEventListener("push", function (event) {
  if (event && event.data) {
    try {
      var data = event.data.json();
      if (data && data.method === "pushMessage") {
        console.log("Push notification sent");
        self.registration.showNotification("Eco
          mmerce website", { body:
          data.message,
        });
      }
    } catch (error) {
      console.error("Error parsing push data:", error);
```

```
    }
  }
});
var preLoad = function () {
  return
    caches.open("offline").then(funct
    ion (cache) { // caching index
    and important routes
    return cache.addAll([
      "/",
      "/index.html",
      "/about.html",
      "/blog.html",
      "/contact.html",
      "/services.html",
      "/img/slider-img4.jpg",
      "/css/main.css",
    ]);
  });
};
var checkResponse = function (request) {
  return new Promise(function (fulfill, reject) {
    fetch(request)
      .then(function (response) {
        if (response.status !== 404) {
          fulfill(response);
        } else {
          reject(new Error("Response not found"));
        }
      })
      .catch(function (error) {
        reject(error);
      });
  });
};
var returnFromCache = function (request) {
  return
    caches.open("offline").then(function
    (cache) { return
    cache.match(request).then(function
    (matching) { if (!matching ||
    matching.status == 404) {
        return cache.match("offline.html");
      } else {
        return matching;
      }
```

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

**Output:**

Fetch:

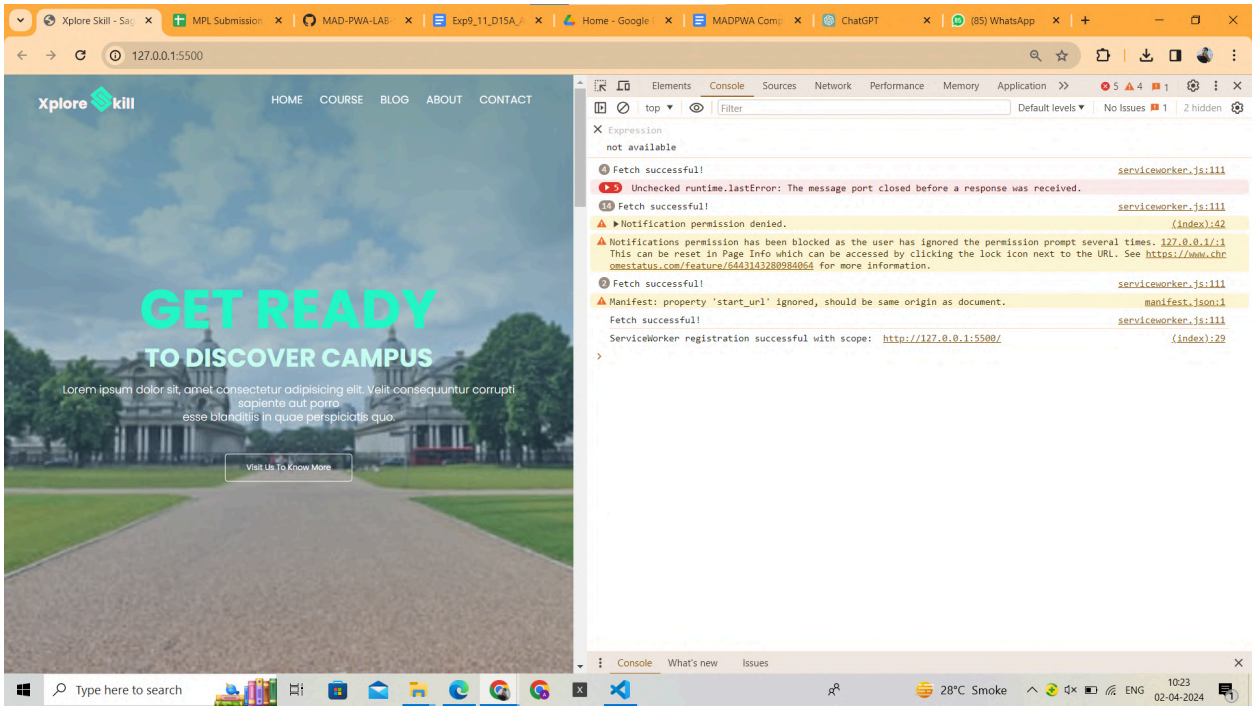


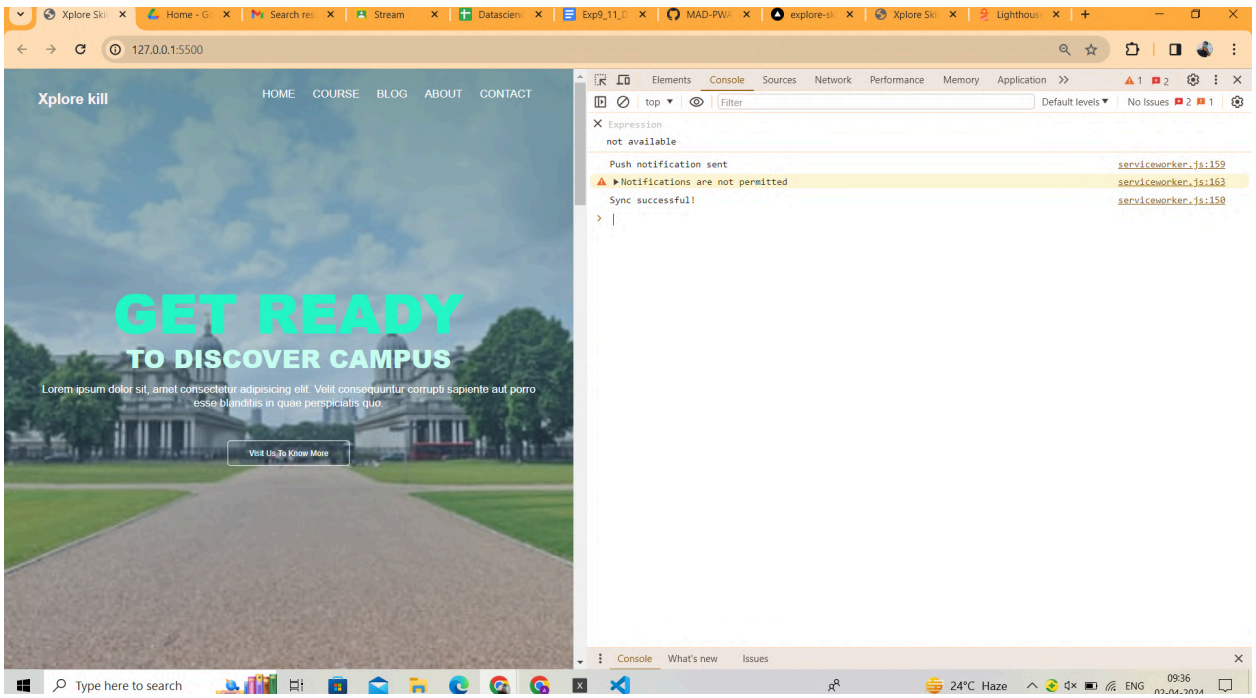

SYNC:

&

PUSH:

**Project Title: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**




Conclusion :

In this experiment, we have successfully implemented service worker events like fetch, sync and push for E-commerce PWA and found out output for above implementation.

# MAD & PWA Lab
# Journal

| | |
|---|---|
| Experiment No. | 10 |
| Experiment Title. | To study and implement deployment of Ecommerce PWA to GitHub Pages. |
| Roll No. | 11 |
| Name | Anuj Chitari |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO5: Design and Develop a responsive User Interface by applying PWA Design techniques |
| Grade: | |

# MAD and PWA Lab

## Name: Anuj Chitari Class: D15A Roll no:04

## Experiment - 10

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

## Theory:

### GitHub Pages

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

GitHub Pages provides the following key features:

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

Reasons for favoring this over Firebase:

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

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

Pros

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

Cons

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

**Firebase**

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

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

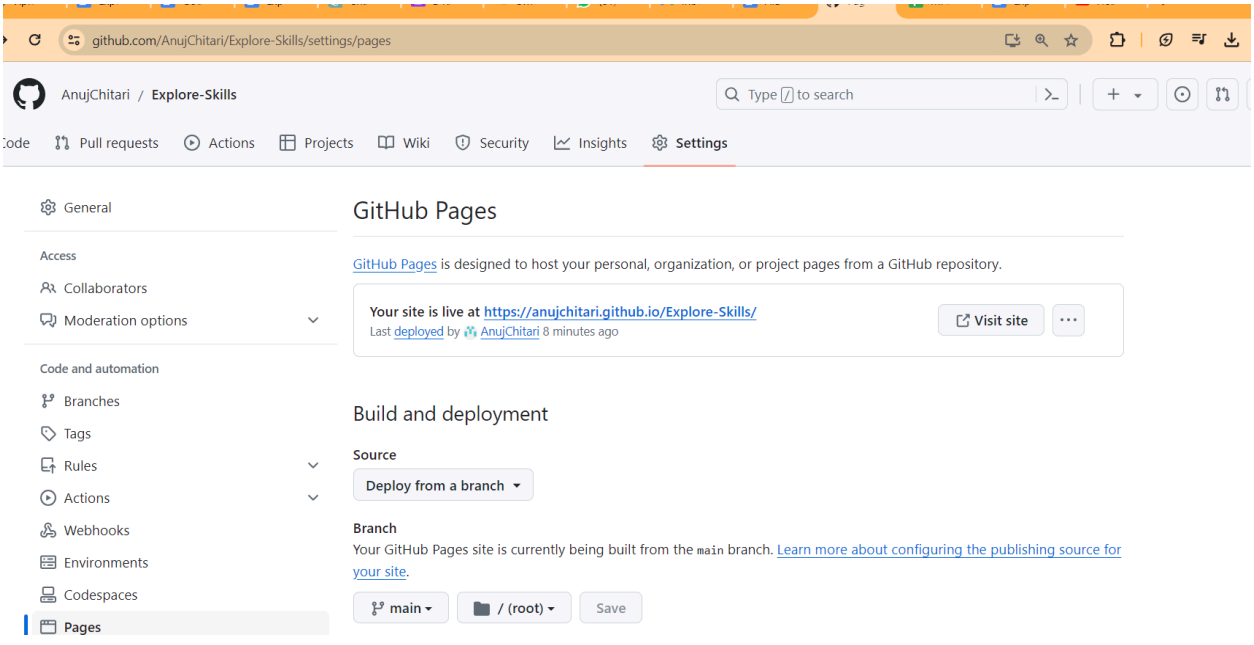
Pros

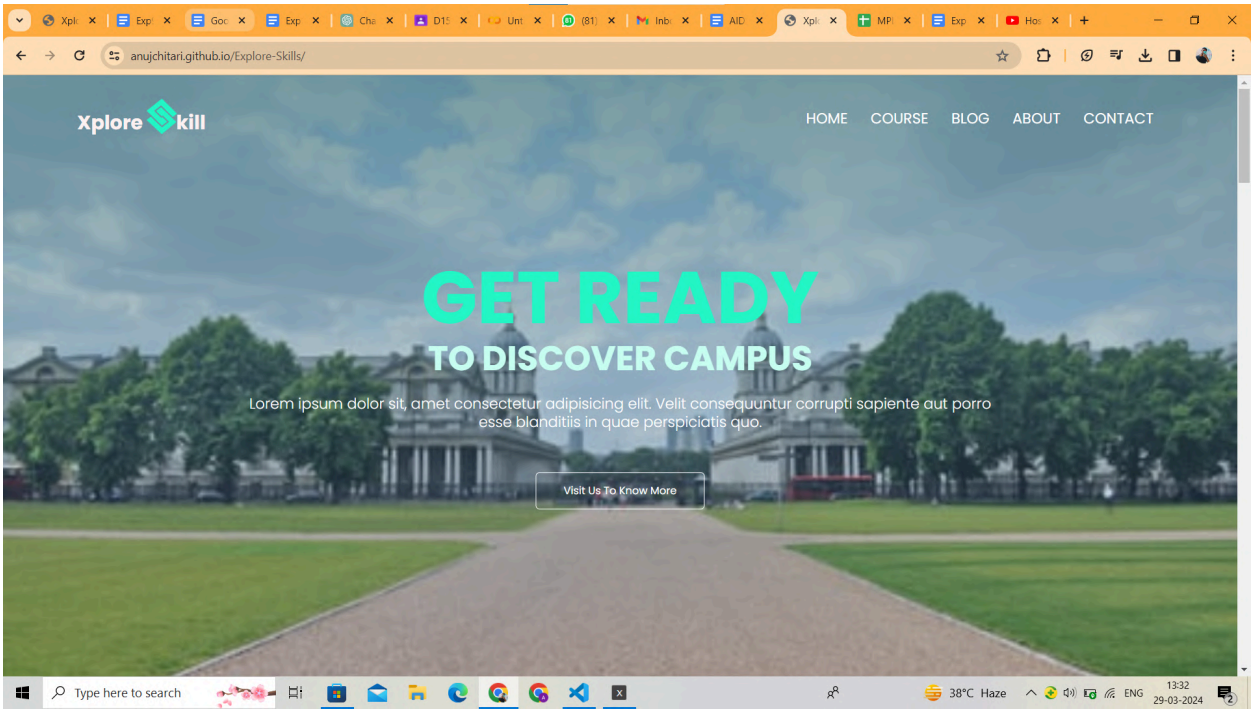
1. Hosted by Google. Enough said.
2. Authentication, Cloud Messaging, and a whole lot of other handy services will be available to you.
3. A real-time database will be available to you, which can store 1 GB of data.
4. You'll also have access to a blob store, which can store another 1 GB of data. 5.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

Cons

1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.

## Implementation:



**Link to our GitHub repository:**
https://github.com/AnujChitari/Explore-Skills

**Hosted Link:** https://anujchitari.github.io/Explore-Skills/

# Conclusion:

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

# MAD & PWA Lab
# Journal

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

# MAD and PWA Lab

## Name: Anuj Chitari Class: D15A Roll no:11

## Experiment - 11

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

## Theory:

Google Lighthouse

Lighthouse is an open-source, automated tool for improving the quality of web pages. You can run it against any web page, public or requiring authentication. It has audits for performance, accessibility, progressive web apps, SEO and more.

You can run Lighthouse in Chrome DevTools, from the command line, or as a Node module. You give Lighthouse a URL to audit, it runs a series of audits against the page, and then it generates a report on how well the page did. From there, use the failing audits as indicators on how to improve the page. Each audit has a reference doc explaining why the audit is important, as well as how to fix it.

Features of Lighthouse

Google Lighthouse gives a breakdown of your site into the accompanying metrics. Here is a brief explanationof each of the aforementioned metrics:

1. Performance

Performance is generally viewed as the most valuable metric given by the Google Lighthouse tool. Like the PageSpeed Insights, the Performance area of the Lighthouse report contains a few helpful metrics you can use to advance your site to climb Google's

rankings. The Performance segment of the Lighthouse report joins the Opportunities, Field Data, Lab Data, and Diagnostics metrics of the PageSpeed Insights tool.

A great example is the opportunities metric as it flags three types of render-blocking URL's namely stylesheets, scripts, and HTML imports. This merged perspective on performance metrics gives an exact and valuable analysis of your site's performance and any progressions you should make to expand your site's exhibition.

2. Accessibility

The first of the new regions of Google Lighthouse is the Accessibility metric. Basically what this metric does is feature potential chances to improve the availability and client experience of your mobile app or website.

Following the accessibility improvement report will guarantee that your clients can without much of a stretch explore and utilize your site. Just as guaranteeing that you have the most obvious opportunity with regards to positioning better on web search engines.

3. Best Practices

Another segment new to Google's analysis tools is the Best Practices metric. This region of the Lighthouse report doesn't carefully give execution related data. However, it will give you recommendations which can improve both your exhibition and client experience, particularly for mobile sites.

4. SEO

The latest and most dynamic of the highlights in Google's Lighthouse instrument is the SEO metric.

PageSpeed Insights doesn't offer this tool. This is why most web designers and SEO specialists prefer to utilize Google Lighthouse to analyze a website. The SEO metric gives fundamental tools to examine your page's streamlining for search engine results rankings. While there are numerous more factors which Lighthouse doesn't consider or quantify, the most essential focuses are secured.

5. Progressive Web Applications

The Progressive Web App area is another of Google's most up to date execution measurements incorporated into its Lighthouse tool. While the meaning of a Progressive Web App (PWA) hasn't been especially clear, Google's documentation expresses that there are a few key variables which make a site a PWA. A great feature of this metric is registering service workers which allow you to enable push notifications on your web app

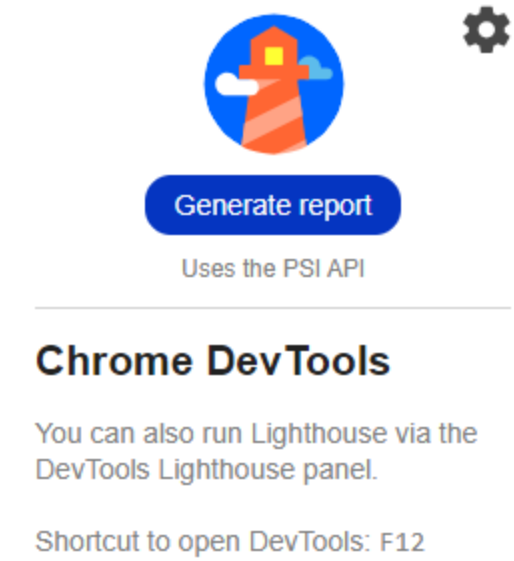
## IMPLEMENTATION:

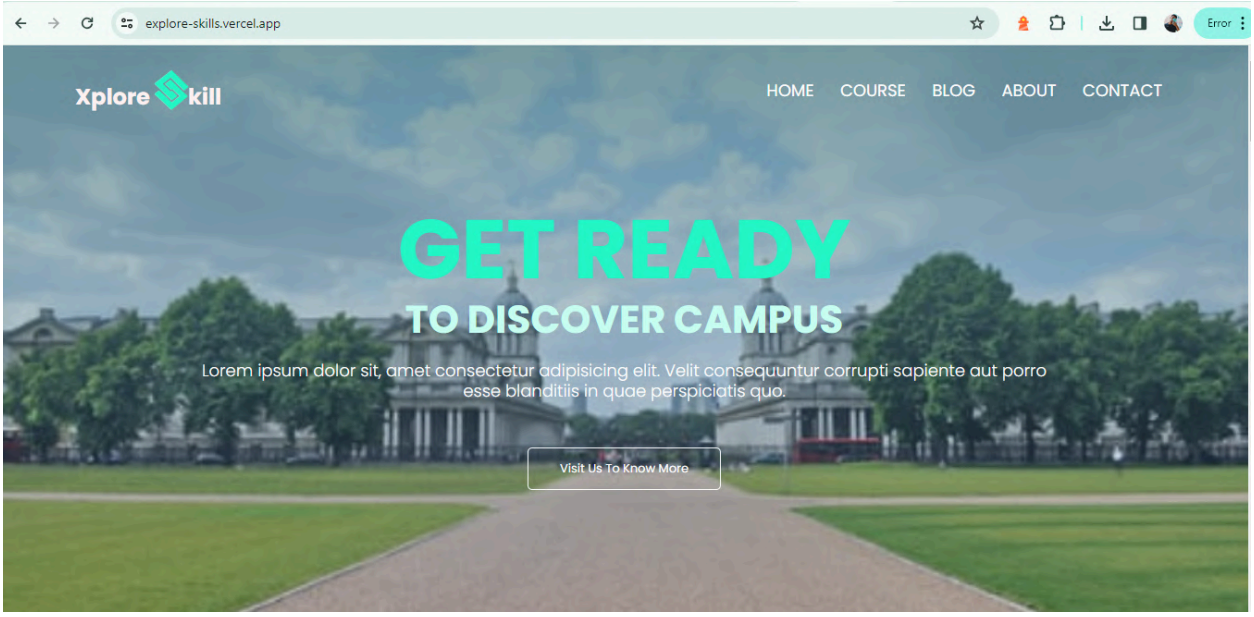
Manifest.json :

```
{
    "short_name": "Xploreskill",
    "name": "Xploreskill Website",
    "icons": [
      {
        "src": "library.png",
        "type": "image/png",
        "sizes": "192x192",
        "purpose": "maskable"
      },
      {
        "src": "library.png",
        "type": "image/png",
        "sizes": "512x512"
      }
    ],
    "start_url": "https://explore-skills.vercel.app/",
    "display": "standalone",
    "theme_color": "#000000",
    "background_color": "#ffffff"
  }
```

**FOR DESKTOP DEVICE**

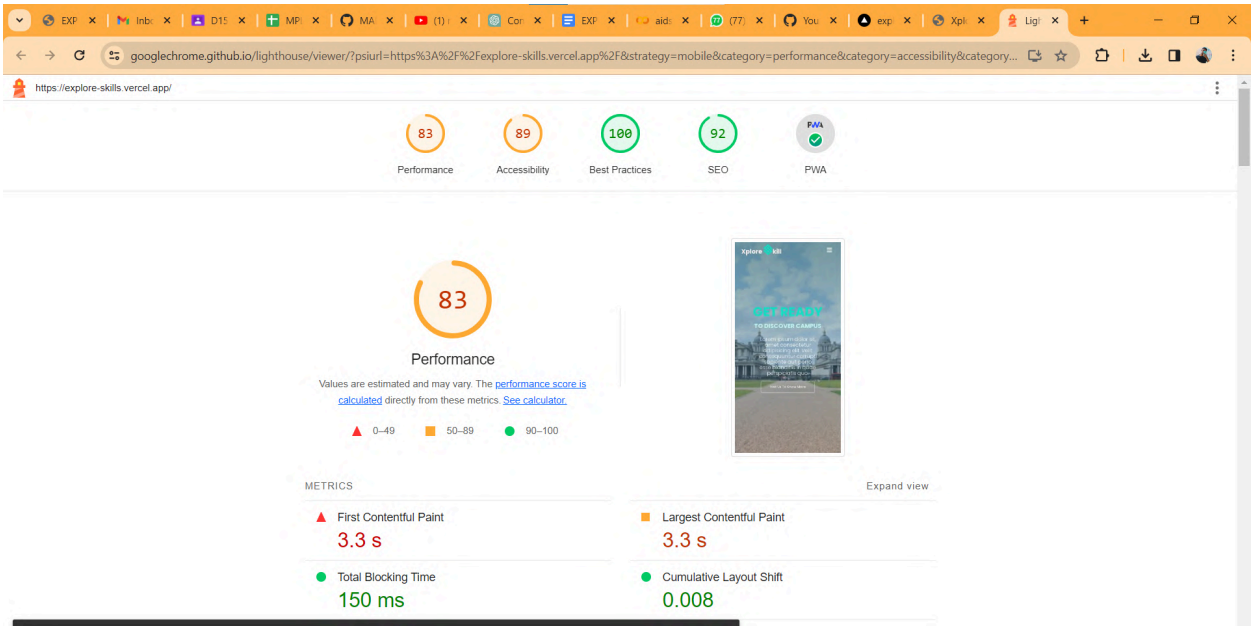
**Project Title: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**

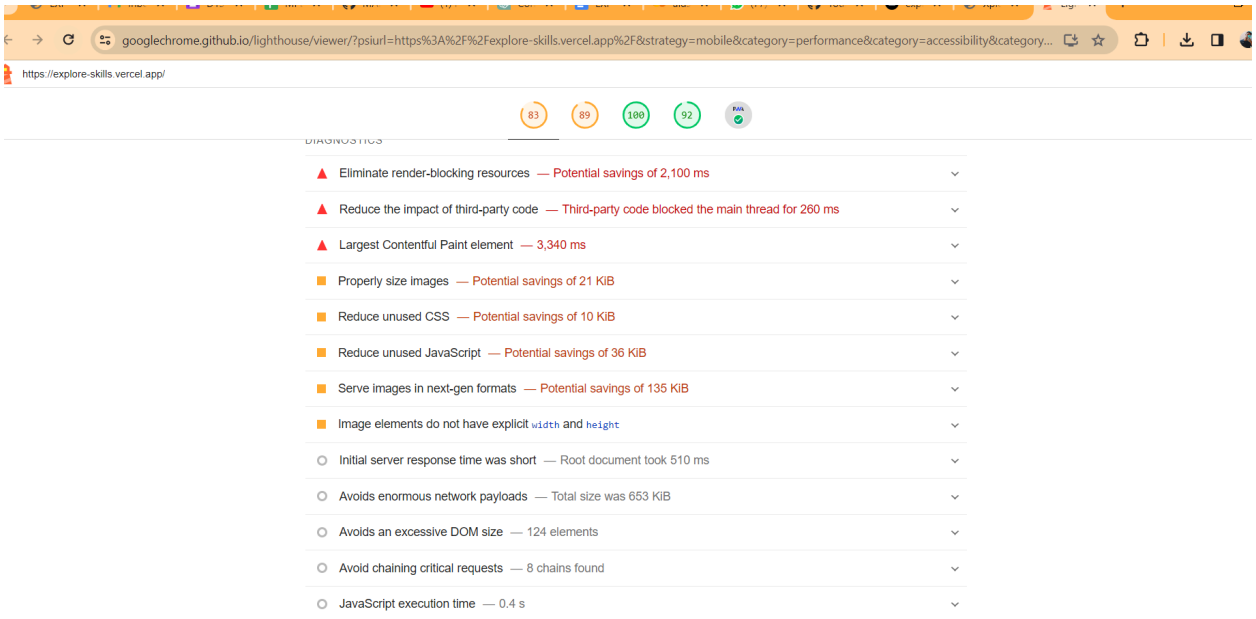


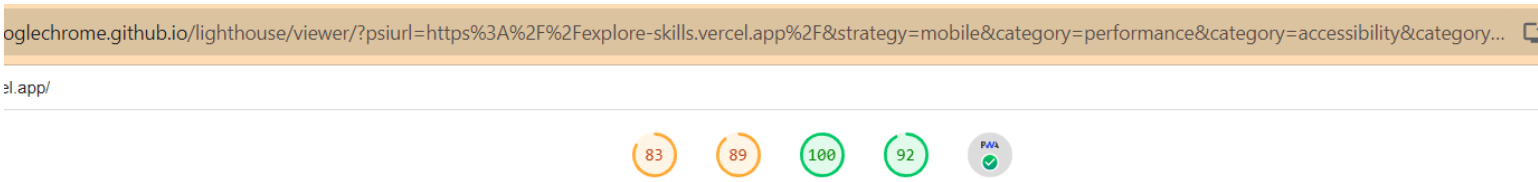

**REPORT GENERATED:**

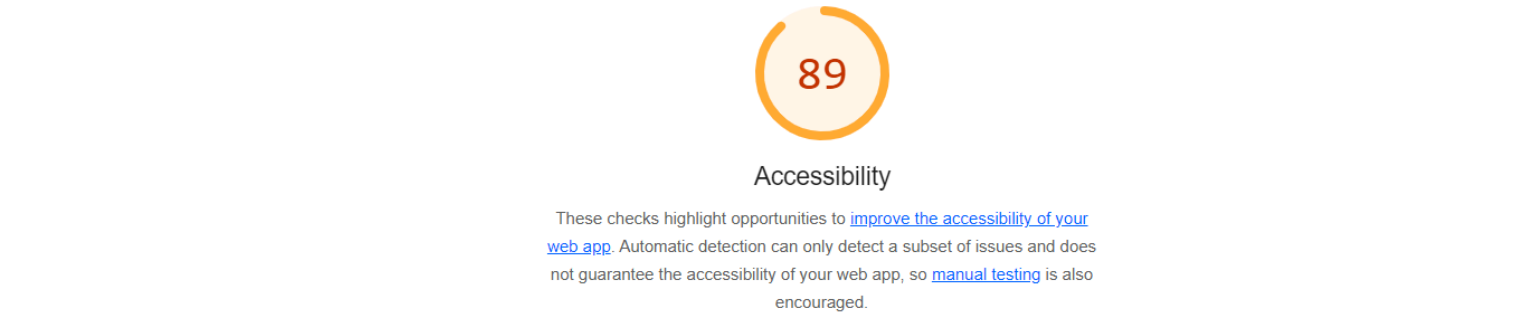
**Project Title: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**

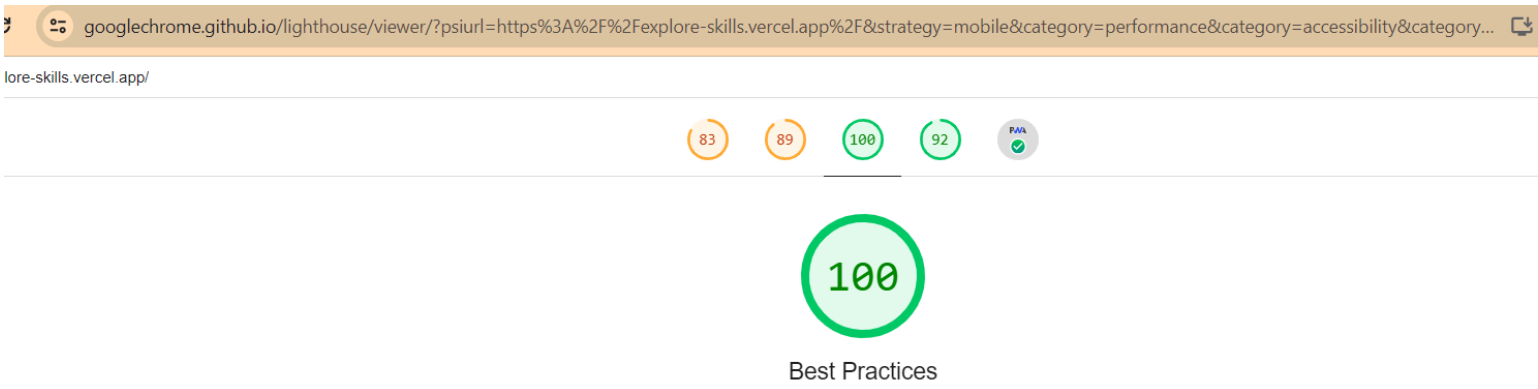


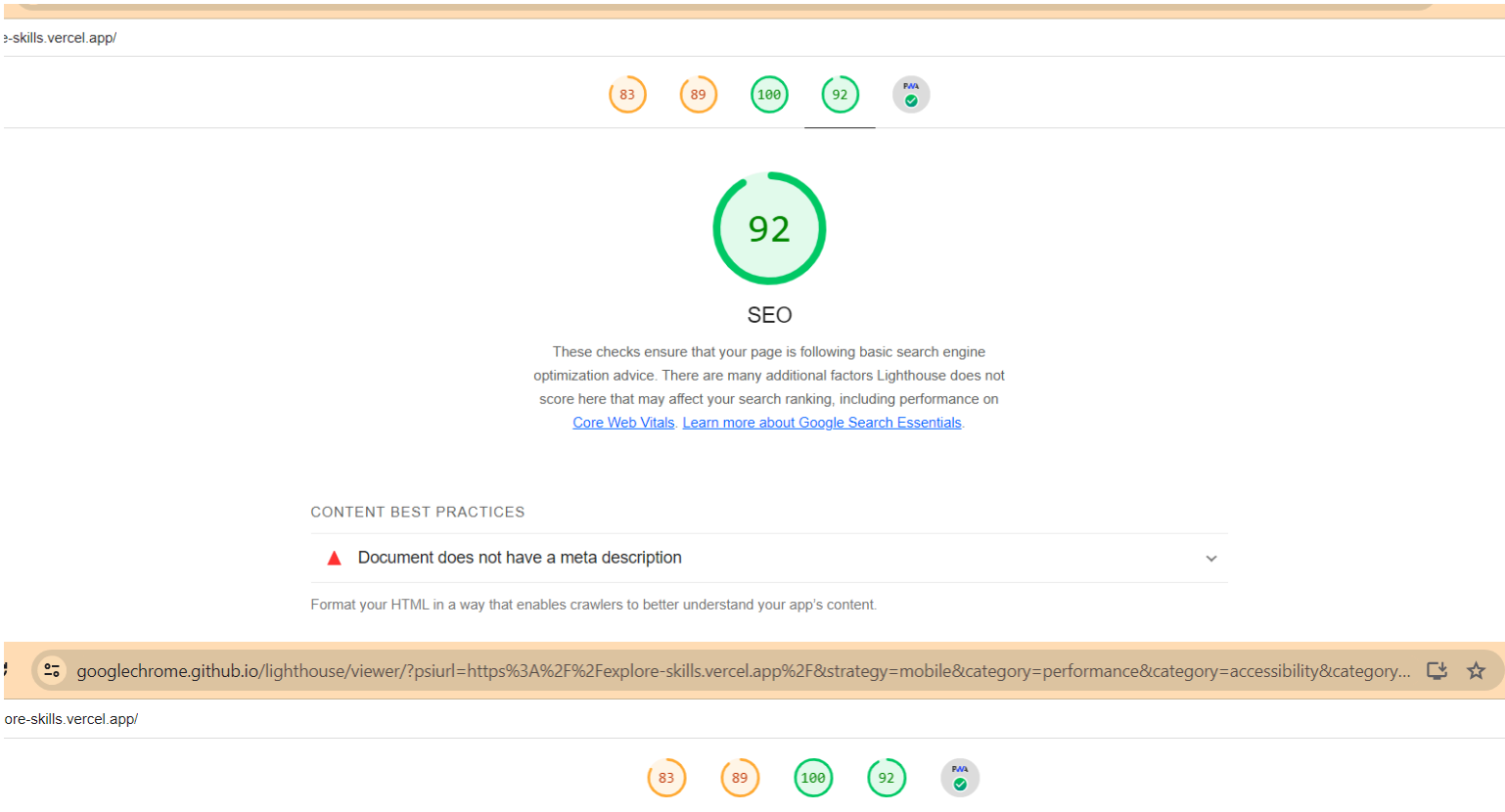
**Project Title: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**



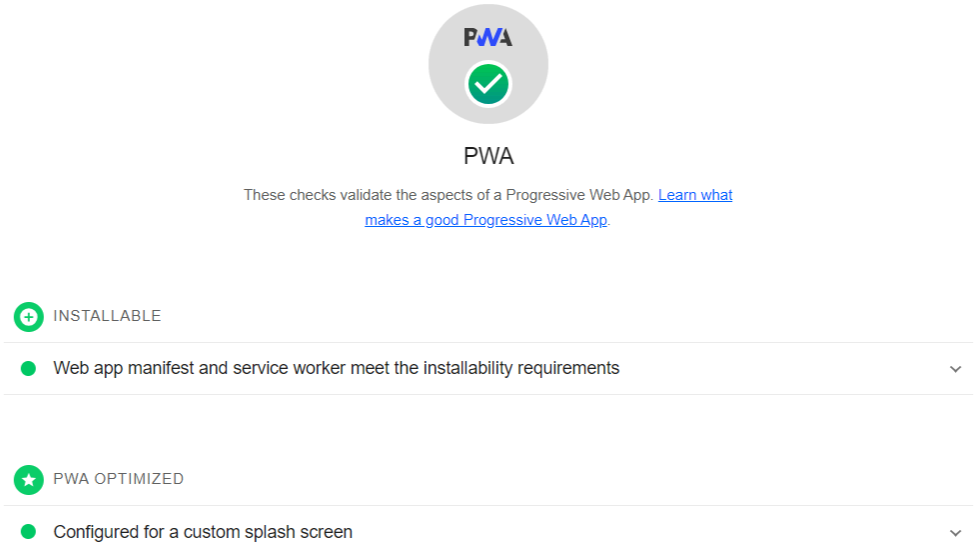



**Project Title: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**

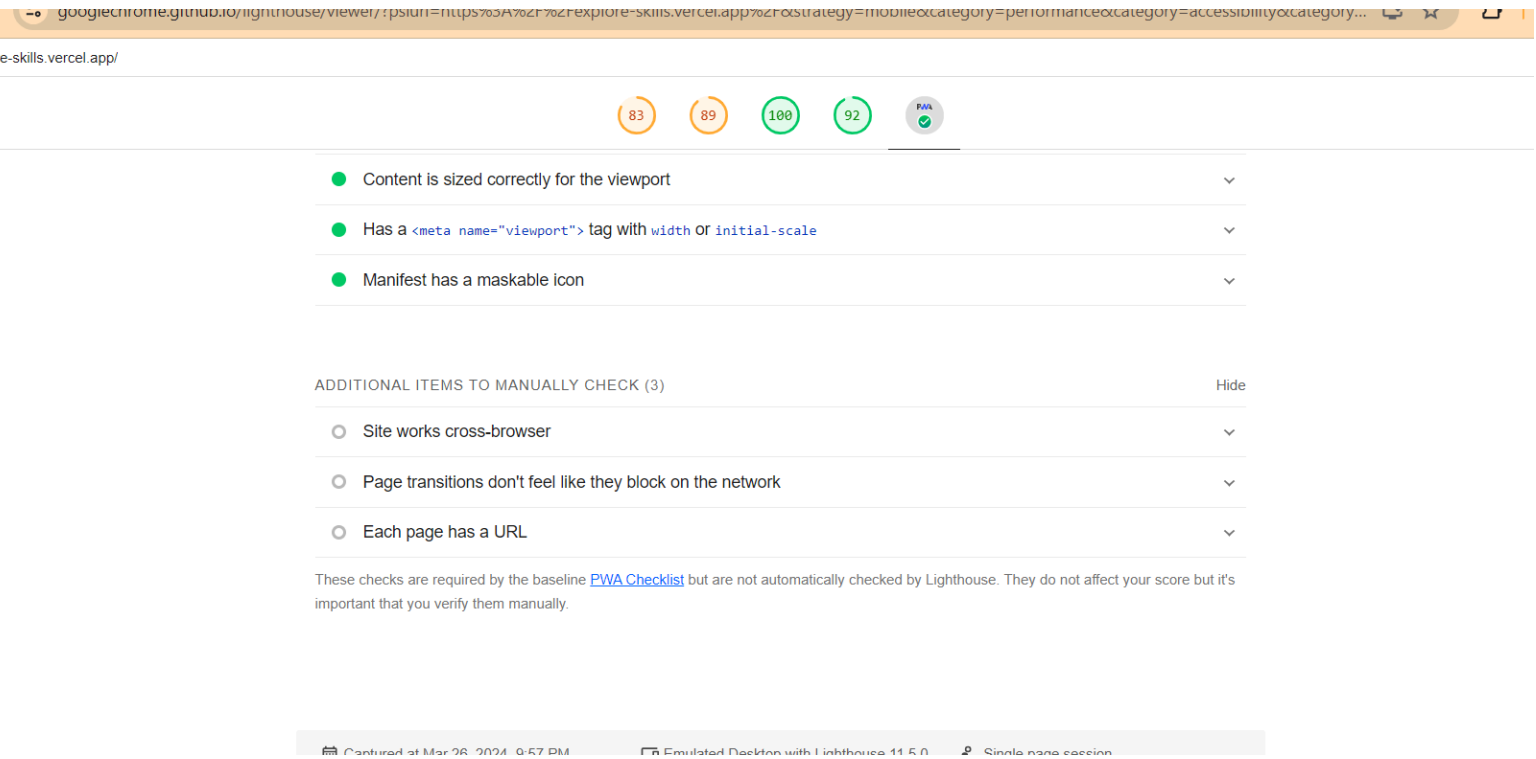








**Project Title: Flutter:UberEats / PWA- ExploreSkill** **Roll No. 11**




# Conclusion:

In this experiment, we have successfully used Google Lighthouse PWA Analysis Tool for testing the PWA functioning.

# MAD & PWA Lab
# Journal

| | |
|---|---|
| Experiment No. | Assignment-1 |
| Assignment 1 Questions | 1. Flutter Overview:Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community.<br>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.<br>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.<br>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. | |
| Name | |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO1: Understand cross platform mobile application development using Flutter framework<br>LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation<br>LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS |
| Grade: | |

Assignment 1 flutter

Date :

Q 1) Flutter Overview:- Explain key features and advantages of using flutter for mobile app development. Discuss how flutter framework differs from traditional approach and why it gained popularity

→ Key features:-

ⓐ Single codebase for multiple platforms:- Flutter allows developer to write code and deploy it on both ios and android.

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

ⓒ Express UI:- Developers have flexibility to create expressive and flexible UI.

ⓓ Integration with other tool. Flutter easily integrate with other popular development tool and framework.

2. Advantages:-

ⓐ Faster Development. Uses single codebase for multiple platform

ⓑ Consistent UI Widget provide a consistent look

Q4) Firebase Integration in flutter. Explain the process of integrating Firebase with Flutter application. Discuss benefit of using Firebase as a backend solution. Highlight Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.

→

Integration:-

1. Go to Firebase console and create new project
2. Add Firebase SDK by including dependencies in pubspec.yaml

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

3. Run flutter pub get
4. Initialize firebase by calling 'firebase.initialiseApp()

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

void main () async {
  Widget FlutterBinding.ensureInitialized();
  await Firebase.initializeApp()
  runApp (myApp());
  }
```

Teacher's Sign.:

Page No.
Date :

and feel across different platform

© Cost Efficiency:-
Developing and maintaining single codebase for both ios and android reduces development cost.

3) Differ from Traditional approach
@ Traditional approach uses a hierarchical structure for UI component, where as Flutter uses a widget based approach
ⓑ Hot Reload allows to see changes made instantly

Flutter popularity is driven by increased productivity, a growing community, cross platform and adoption by companies

Q 2) Widget Tree and Composition: Describe the concept of widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide eg of commonly used widget & their role in widget tree.

→

Widget Tree:

The widget Tree is a hierarchical structure of widget that defines the user interface

Every visual element, from simple component to complex layout, is represented by widget.

Widget are categorized in 2 types

ⓐ Stateless Widget

It is immutable and cannot change over time

ⓑ Stateful Widget

Widget that can change state over time.

Widget Composition

- Widget composition in Flutter involves combining multiple simple widget to create more complex and compound widget.

Teacher's Sign.: _____

Commonly used widget :-

(a) Container :-
A box model for padding, margin and decoration

(b) Column and Row
· layout widget for arranging children vertically or horizontally

(c) Stack
Overlapping widget allowing them to be layered on top of each other.

(d) List View :-
A scrollable grid of widget.

(e) Grid View :-
A scrollable grid of widget.

(f) Text field :-
An input field for user

Q3] State Management in Flutter:
Discuss importance of state management in Flutter application. Compare and contrast different state management approaches available in Flutter, such as setState, Provider scenario wher each approach suitable.

→ State management is crucial in Flutter application because it involves managing the data that can chang over time

| setState | Provider | Riverpod. |
|---|---|---|
| Built in Flutter method | External package named ('provider') | External package ('riverpod') |
| local state within widget | Global state within widget tree | Global state with additional feature |
| limited scalability for large app | Suitable for medium size app | Designed for large and complex app |
| may lead to code redudancy | Balances simplicity and readability | Emphorizes readiability and clean syntax. |
| Testing is challenging | Good testability support | Enhanced testing experience. |

Benefits of using Firebase as Backend:

① Realtime Database
- firebase offer a realtime NoSQL database.

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

③ Cloud firestore:
Firebase's Cloud firestore provide a secure and easy to implement scalable NoSQL db that allows you to store and sync data

Data Synchronization:-

① Realtime Database
when data changes on one client, it triggers event that automatically update data on other client.

② Cloud firestore
It notifies client when data changes, allowing for seamless realtime updates.

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

Scenario where each is applicable.

① Set State
for small to moderately complex application when managing local state within a widget.
eg:- Simple form, UI specific state.

② Provider
For medium to large sized application when a centralized state is needed accessible by multiple widget
eg:- managing user authentication, theme change or appwide configuration

③ Riverpod :-
For large and complex application when testability and maintainability are top priorities
eg:- Complex application with multiple feature dynamic UIs

# 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<br>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.<br>3. Describe the lifecycle of Service Workers, including registration, installation, and activation phases.<br>4. Explain the use of IndexedDB in the Service Worker for data storage. |
| Roll No. | 11 |
| Name | Anuj Chitari |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO4:Understand various PWA frameworks and their requirements<br>LO5: Design and Develop a responsive User Interface by applying PWA Design techniques<br>LO6:Develop and Analyze PWA Features and deploy it over app hosting solutions |
| Grade: | |

· Anuj · Chitari ~~Anurag~~
- DISA - II
- Batch - A.

PWA Assignment (2)

Q(1)

Ans ① PWA :- are type of web appln that combines the best features of traditional web apps.

② PWA are built using web techno. like html, cssf JS, but they can provided a user experience i.e similar to native app.

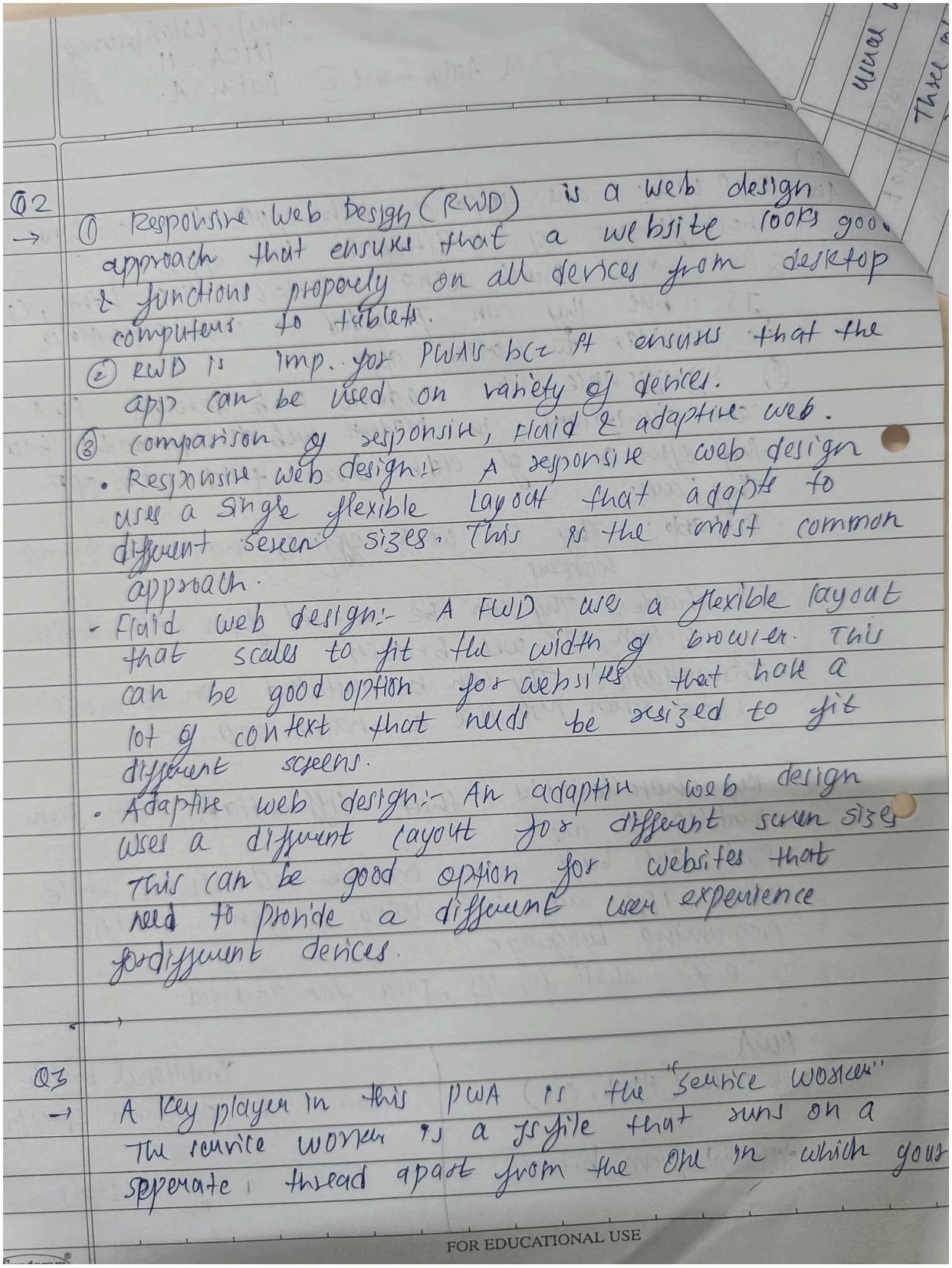
③ significance in modern web develop. PWA are significant in modern web development bcoz they offer a no. of adv. Over trad. web app. They are :-

- Reliable :- They can work offline, thanks to Service workers.
- Reachable :- They can be accessed from any device within a web browser
- Instantiable :- They can be installed on a user's home screen just like a native app

④ key characteristics that differentiate PWA from traditional are...

- PWA are built using ~~mobile~~ techno/web, while native apps are built using platform - specific programing Language
  e.g:- swift for iOS, Java for android.

| PWA | Traditional mobile app |
|---|---|
| Web Tech (Html, css) | Native code platform specific. |
| Installed from browser | Download from store |



Q2
→ ① Responsive web design (RWD) is a web design approach that ensures that a website looks good & functions properly on all devices from desktop computers to tablets.
② RWD is imp. for PWAs bcz it ensures that the app can be used on variety of devices.
③ comparison of responsive, fluid & adaptive web.
- Responsive web design:- A responsive web design uses a single flexible layout that adapts to different screen sizes. This is the most common approach.
- Fluid web design:- A FWD use a flexible layout that scales to fit the width of browser. This can be good option for websites that have a lot of context that needs be resized to fit different screens.
- Adaptive web design:- An adaptive web design uses a different layout for different screen sizes. This can be good option for websites that need to provide a different user experience for different devices.

Q3
→ A key player in this PWA is the "service worker". The service worker is a js file that runs on a seperate thread apart from the one in which your

usual website JS file runs.

Three phases of lifecycle:-

a) Registration phase:-

→ The 1st phase in service worker is lifecycle is registration it to the browser. The registration can be done in 2 ways:-

- You either specify a scope for a service worker has access to.
- You leave it to the default global scope of where the service worker is present.

b) Installation phase:-

→ Once the service worker is successfully registered, it is not ready to be installed. The service worker script is downloaded to the browser & the browser will attempt to install the service worker

- There are few situations in which service worker will be installed:-
  - A new service worker file.
  - A modified service worker file.

c) Activation:-

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

below cases:-
- None of pages use the service worker & closed
- There is no other service worker active on that page.

Q4 explain the use of indexed DB in service worker for data storage.

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

Benifits of using IndexedDB with service worker:-
- Offline data access:- Improves U.I by allowing interaction with PWA even w/o internet connectio

- Faster load times:- cached data retrival from indexed can be served much faster than fetchi it from the service again.