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

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Name of the Course : MAD & PWA Lab

Course Code : ITL604

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

Faculty Incharge : Mrs. Kajal Joseph.

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

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

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: Flutter : FreshMenu / PWA : Ecommerce Website **Roll No. 35**

Lab Objectives:

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

Lab Outcomes:

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

Index

Sr.No	Experiment Title	LO	DOP	DOS	Grade
1.	To install and configure the Flutter Environment	LO1	17/01/2024	24/01/204	15M
2.	To design Flutter UI by including common widgets.	LO2	24/01/2024	31/01/2024	15M
3.	To include icons, images, fonts in Flutter app	LO2	31/01/2024	07/02/2024	14M
4.	To create an interactive Form using form widget	LO2	07/02/2024	14/02/2024	14M
5.	To apply navigation, routing and gestures in Flutter App	LO2	14/02/2024	21/02/2024	14M
6.	To Connect Flutter UI with fireBase database	LO3	21/02/2024	06/03/2024	14M
7.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4	06/03/2024	29/03/2024	15M
8.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA	LO5	13/03/2024	29/03/2024	15M
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5	20/03/2024	29/03/2024	15M
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5	27/03/2024	29/03/2024	15M
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6	27/03/204	29/03/2024	15M
12.	Assignment-1	LO1,LO2 ,LO3	28/01/2024	05/03/2024	5M
13.	Assignment-2	LO4,LO5 ,LO6	14/03/2024	21/03/2024	4M

MAD & PWA Lab
Journal

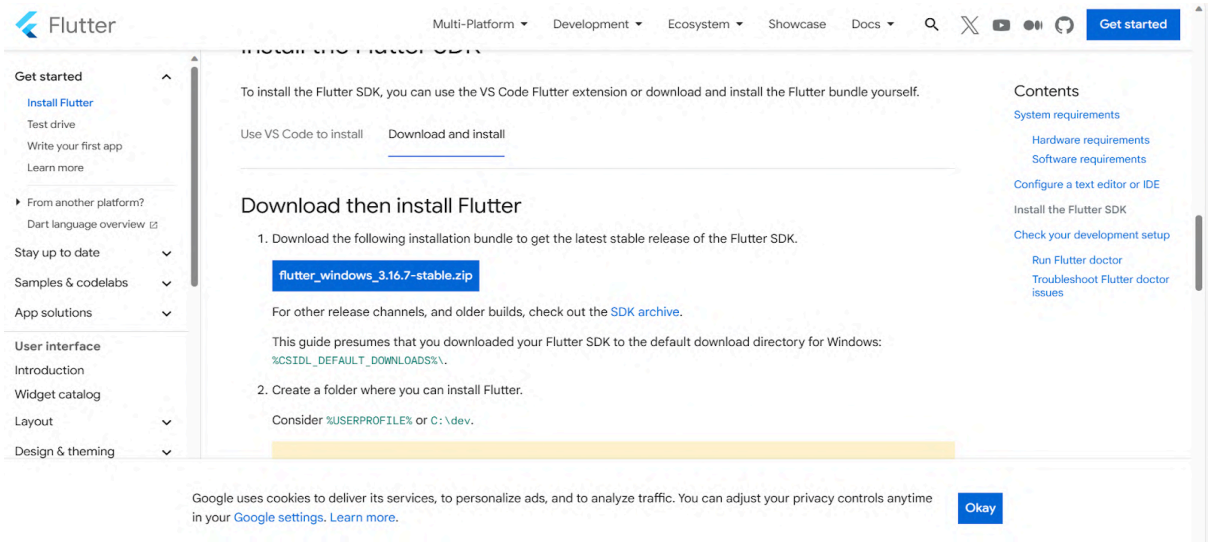
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	35
Name	SEJAL MAURYA
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15M

EXPERIMENT NO - 01

Installation and Configuration of Flutter Environment.

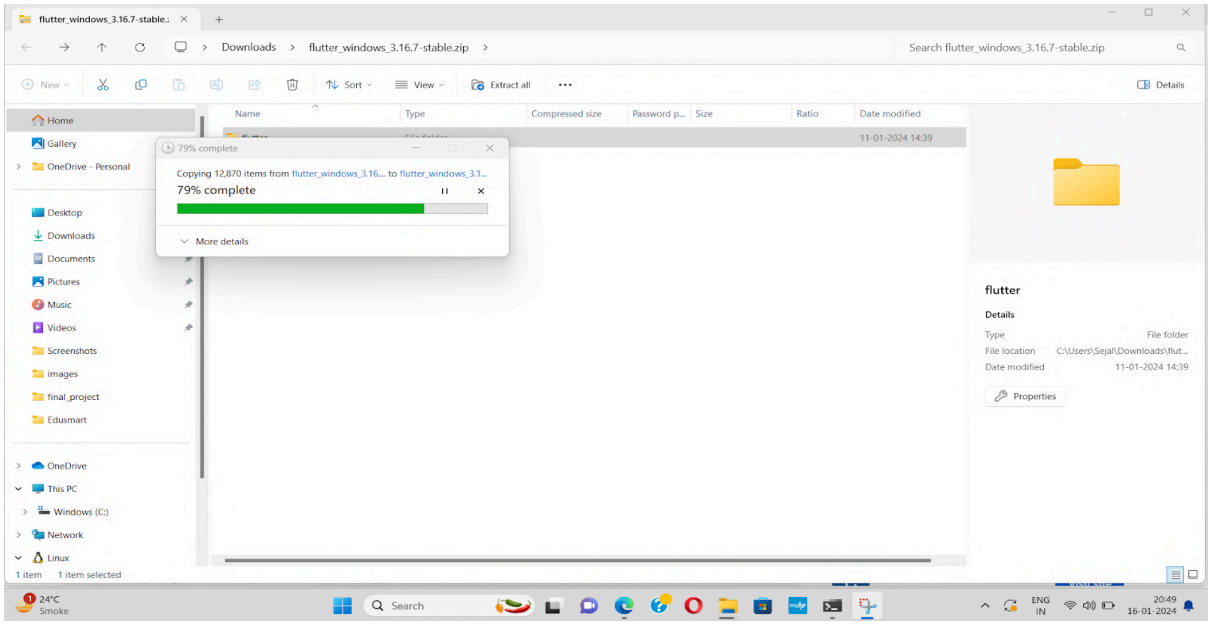
Steps

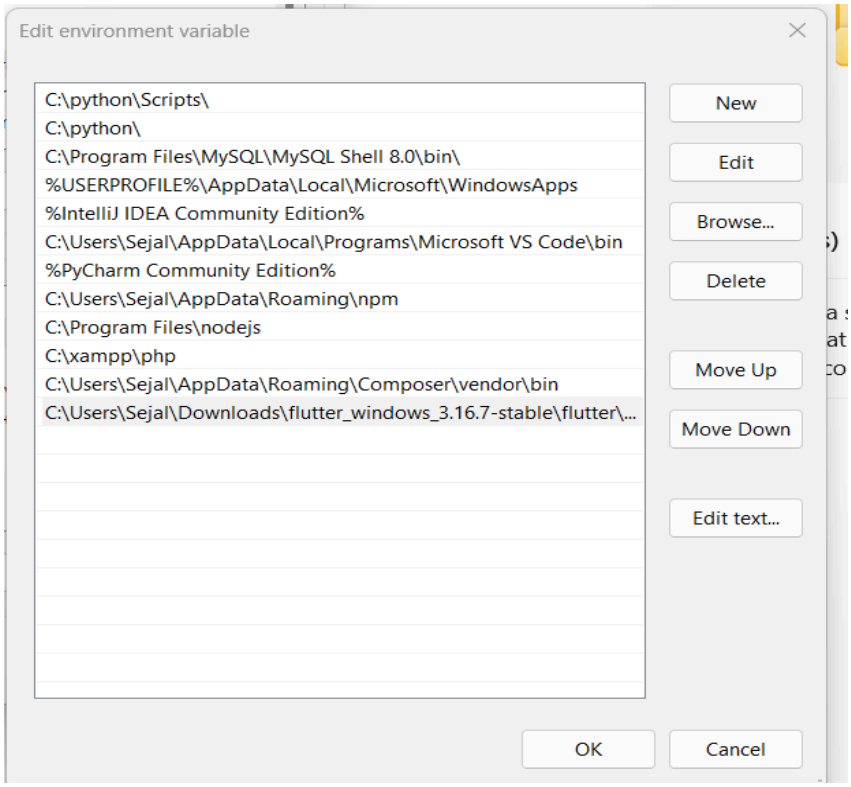
Step 1: Download the installation bundle of the Flutter Software Development Kit for windows.



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

After unzip the file

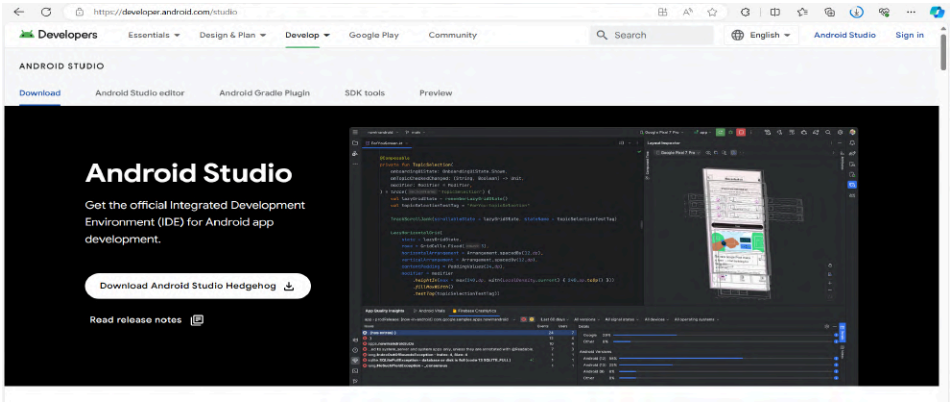


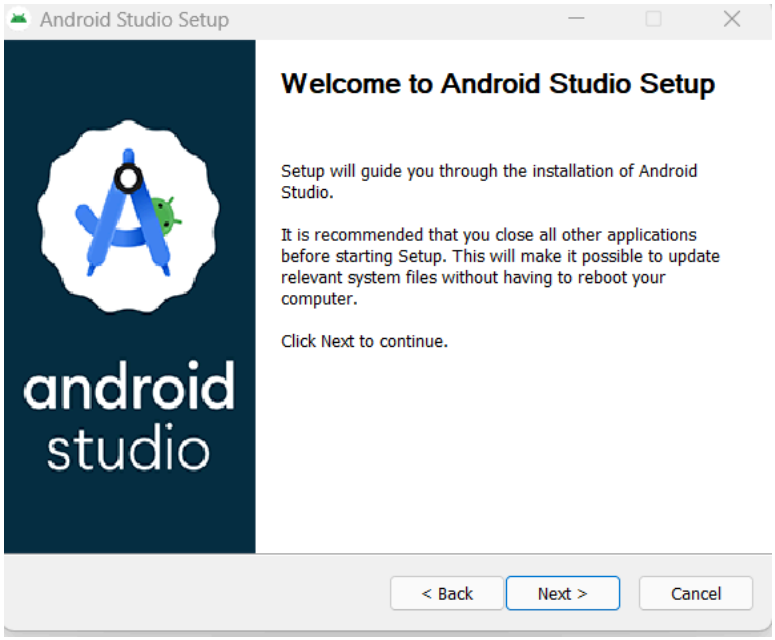


Now, run the $ flutter command in command prompt.

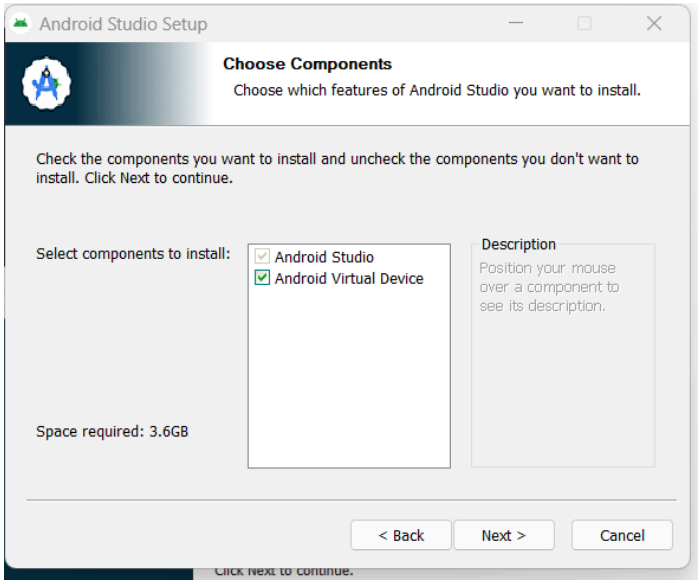
```
C:\Users\Sejal>flutter
Manage your Flutter app development.

Common commands:

  flutter create <output directory>
    Create a new Flutter project in the specified directory.

  flutter run [options]
    Run your Flutter application on an attached device or in an emulator.

Usage: flutter <command> [arguments]

Global options:
-h, --help                  Print this usage information.
-v, --verbose               Noisy logging, including all shell commands executed.
                            If used with "--help", shows hidden options. If used with "flutter doctor", shows additional
                            diagnostic information. (Use "-vv" to force verbose logging in those cases.)
-d, --device-id             Target device id or name (prefixes allowed).
    --version               Reports the version of this tool.
    --enable-analytics      Enable telemetry reporting each time a flutter or dart command runs.
    --disable-analytics     Disable telemetry reporting each time a flutter or dart command runs, until it is
                            re-enabled.
    --suppress-analytics    Suppress analytics reporting for the current CLI invocation.

Available commands:
```

```
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sejal>dart pub cache repair
  ╔═══════════════════════════════════════════════════════════════════╗
  ║ The Dart tool uses Google Analytics to report feature usage statistics
  ║ and to send basic crash reports. This data is used to help improve the
  ║ Dart platform and tools over time.
  ║
  ║ To disable reporting of analytics, run:
  ║
  ║   dart --disable-analytics
  ╚═══════════════════════════════════════════════════════════════════╝

^CTerminate batch job (Y/N)?
^CThe syntax of the command is incorrect.

C:\Users\Sejal>dart pub cache repair
Reinstalled 18 packages.

C:\Users\Sejal>flutter doctor
Found an existing Pub cache at C:\Users\Sejal\AppData\Local\Pub\Cache.
It can be repaired by running `dart pub cache repair`.
It can be reset by running `dart pub cache clean`.
```





Project Title: Flutter : FreshMenu / PWA : Ecommerce Website **Roll No. 35**

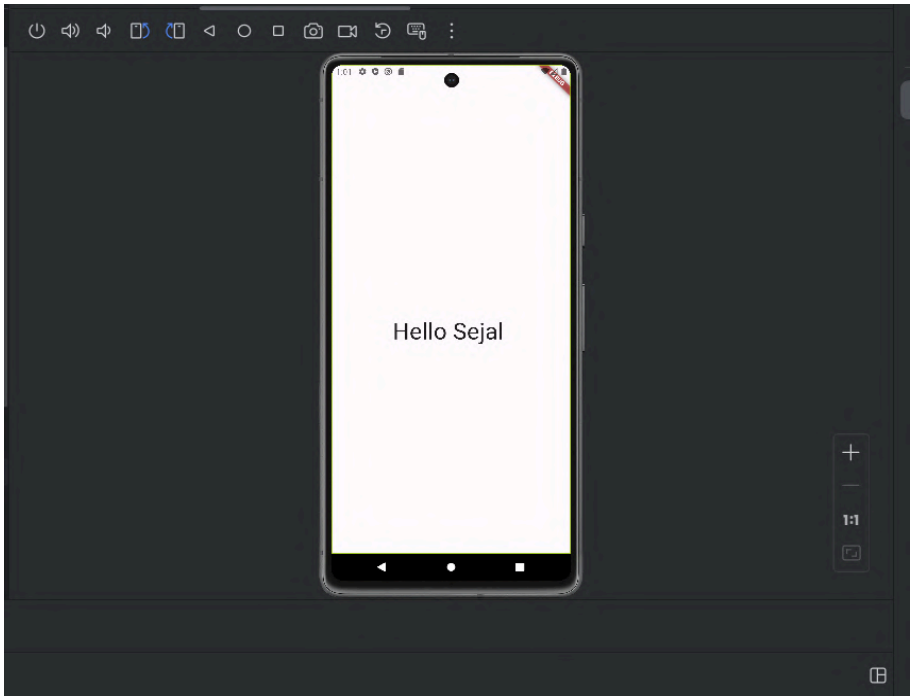


Code And Output :

```
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(
     home: Scaffold(
       appBar: AppBar(

       ),
       body: Container(
         child: Center(
           child: Text("Hello Sejal ",
               style: TextStyle(
               fontSize: 40
           ),),
         ),
       ),

   ),
   );
   }
}
```

:

Project Title: Flutter : FreshMenu / PWA : Ecommerce Website **Roll No. 35**



Conclusion - Successfully installed Android Studio

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	35
Name	SEJAL MAURYA
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15M

Experiment No -02
Exploring Common Widgets

Aim - To design flutter UI using common widgets
Theory -

Container: The Container widget is a versatile widget that can contain other widgets and apply various styling options, such as padding, margin, color, border, and alignment. It's often used as a layout element to structure the UI.

Text: The Text widget is used to display textual content within the UI. It supports styling options like font size, font weight, color, alignment, and text decoration.

Row and Column: These are layout widgets used to arrange child widgets horizontally (Row) or vertically (Column). They allow you to create flexible layouts and organize widgets in rows or columns.

AppBar: The AppBar widget is a material design widget used to create a customizable app bar at the top of the screen. It typically contains actions, titles, icons, and other UI elements relevant to the current screen or application.

Scaffold: The Scaffold widget is a layout widget that provides a framework for implementing basic material design layouts and features, such as app bars, drawers, bottom navigation bars, floating action buttons, and more.

IconButton: These are button widgets used to trigger actions or navigate to different parts of the application. They provide various styles and configurations for buttons, including text buttons, raised buttons, icon buttons, and more.

Code And Output :

```
import 'package:flutter/material.dart';
import 'package:food/screen/category.dart';
// Define a class to represent each food item
class FoodItem {
  final String imagePath;
  final String foodName;
  final String description;
  final String price;
  final String category;
  final bool isVeg; // New property to indicate if the item is vegetarian
  final Color dotColor; // New property to hold the color of the dot
  final String offer; // New property to hold the offer information
  final String additionalText; // New property to hold additional text below the price
  FoodItem({
    required this.imagePath,
    required this.foodName,
    required this.description,
    required this.price,
    required this.category,
    required this.isVeg,
    required this.dotColor, // Initialize dotColor
    required this.offer, // Initialize offer
    required this.additionalText, // Initialize additionalText
  });
}
class ItemWidgets2 extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: DefaultTabController(
          length: 6,
          child: Scaffold(
            backgroundColor: Colors.black,
            appBar: AppBar(
              backgroundColor: Colors.black,
```

```
        title: Text('Food Items'),
        bottom: TabBar(
          isScrollable: true,
          indicatorColor: Colors.orange,
          labelColor: Colors.orange,
          labelStyle: TextStyle(fontSize: 20),
          tabs: [
            Tab(text: 'Beverages and Cakes'),
            Tab(text: 'Breakfast'),
            Tab(text: 'Lunch'),
            Tab(text: 'Noddles'),
            Tab(text: 'Sandwitch'),
            Tab(text: 'Soup and Salad'),
          ],
        ),
      ),
      body: Stack(
        children: [
          TabBarView(
            children: [
              _buildTabView('Beverages and Cakes'),
              _buildTabView('Breakfast'),
              _buildTabView('Lunch'),
              _buildTabView('Noddles'),
              _buildTabView('Sandwitch'),
              _buildTabView('Soup and Salad'),
            ],
          ),
  Widget _buildTabView(String category) {
final List<FoodItem> foodItems = [

  FoodItem(
    imagePath: 'assets/images/mintchass.jpg',
    foodName: 'RedVelvet Cake',
    description: 'Special cake',
    price: '\₹199',
    category: 'Beverages and Cakes',
```

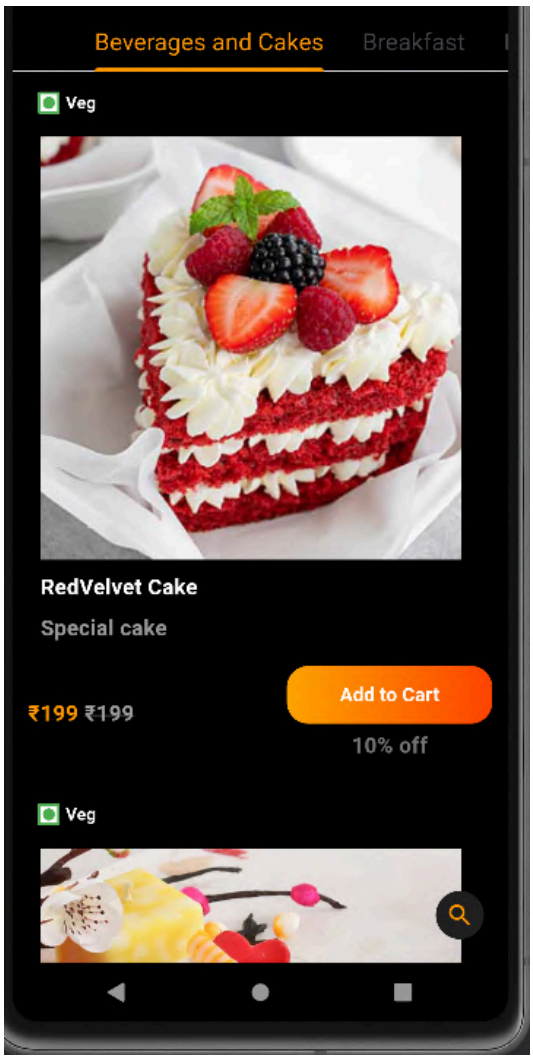
```
      isVeg: true,
      dotColor: Colors.green,
      offer: '10% off', // Example offer
      additionalText: '\₹199', // Example additional text
    ),
     FoodItem(
      imagePath: 'assets/images/hakkanoddles.jpg',
      foodName: 'Hakka Noddles',
      description: '',
      price: '\₹149',
      category: 'Noddles',
      isVeg: true,
      dotColor: Colors.green,
      offer: '10% off', // Example offer
      additionalText: '\₹199', // Example additional text
    ),
    // Add more food items here
  ];
List<FoodItem> filteredItems =
  foodItems.where((item) => item.category == category).toList();
  return ListView.builder(
    itemCount: filteredItems.length,
    itemBuilder: (context, index) {
      return Container(
        margin: EdgeInsets.symmetric(vertical: 8, horizontal: 13),
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(20),
          color: Colors.black,
          boxShadow: [
            BoxShadow(
              color: Colors.black.withOpacity(0.4),
              spreadRadius: 1,
              blurRadius: 8,
            ),
          ],
        ),
        child: Column(
```

```
crossAxisAlignment: CrossAxisAlignment.start,
children: [
 Padding(
  padding: const EdgeInsets.all(8.0)
 InkWell(
  onTap: () {
   Navigator.pushNamed(context, "singleItemPage");
  },
  child: Container(
   margin: EdgeInsets.all(10),
   child: Image.asset(
    filteredItems[index].imagePath,
    width: 350,
    height: 350,
    fit: BoxFit.cover,
   ),
  ),
 ),
 Padding(
  padding: EdgeInsets.only(left: 10, bottom: 8),
  child: Text(
   filteredItems[index].foodName,
   style: TextStyle(
    fontSize: 18,
    fontWeight: FontWeight.bold,
    color: Colors.white,
   ),
  ),
 ),
 Padding(
  padding: EdgeInsets.only(left: 10, bottom: 8),
  child: Text(
   filteredItems[index].description,
   style: TextStyle(
    fontSize: 18,
    fontWeight: FontWeight.bold,
    color: Colors.white60,
```

```
                ),
              ),
            ),
            Padding(
              padding: EdgeInsets.symmetric(vertical: 10),
              child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  Row(
                    children: [
                      Text(
                        filteredItems[index].price,
                        style: TextStyle(
                          fontWeight: FontWeight.bold,
                          fontSize: 18,
                          color: Colors.orange,
                        ),
                      ),
                      SizedBox(width: 5), // Adding some space between the price and
the additional text
                      Text(
                        filteredItems[index].additionalText, // Additional text
                        style: TextStyle(
                          fontSize: 18,
                          fontWeight: FontWeight.bold,
                          decoration: TextDecoration.lineThrough,
                          decorationColor: Colors.grey,
                          color: Colors.grey,
                        ),

                      ),
                    ],
                  ),


              SizedBox(height: 5), // Adding some space between the button and the
                      Text(
```

```
                filteredItems[index].offer, // Offer text
                style: TextStyle(
                  color: Colors.grey[600],
                  fontSize: 18,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ],

}
```

Output :



Conclusion - I learnt about the common widgets in flutter like container, rows,columns ,images etc and successfully implemented them.

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	35
Name	SEJAL MAURYA
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	14M

Experiment No -03
Exploring Flutter Widgets

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

Code and Output :

1) Button the Button widget is not a specific widget, but rather a category of widgets that are used to handle user interaction by triggering actions when pressed. Some commonly used button widgets include:
Elevated Button , Textfield Button, Outlined button etc

2.) Textfield with Icon

In Flutter, a TextField widget is used to allow users to input text. It is a fundamental part of many forms and input-based user interfaces. TextField provides a text input area where users can enter and edit text, and it comes with various customization options.

3.Image

This widget holds the image which can fetch it from multiple sources like from the asset folder or directly from the URL.
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.

4.Gesture Detection: To make an image interactive like a button, you need to detect user gestures such as taps. Flutter provides gesture detection widgets like GestureDetector .These widgets allow you to listen for various touch events like taps, swipes, and drags.I used this widget to make image as a button.

5.Icon Button: Flutter provides the IconButton widget, which combines an icon with a tappable area, making it easy to create interactive icons that respond to user taps. The IconButton widget is commonly used for actions like navigation, opening menus, submitting forms, etc.

Code -

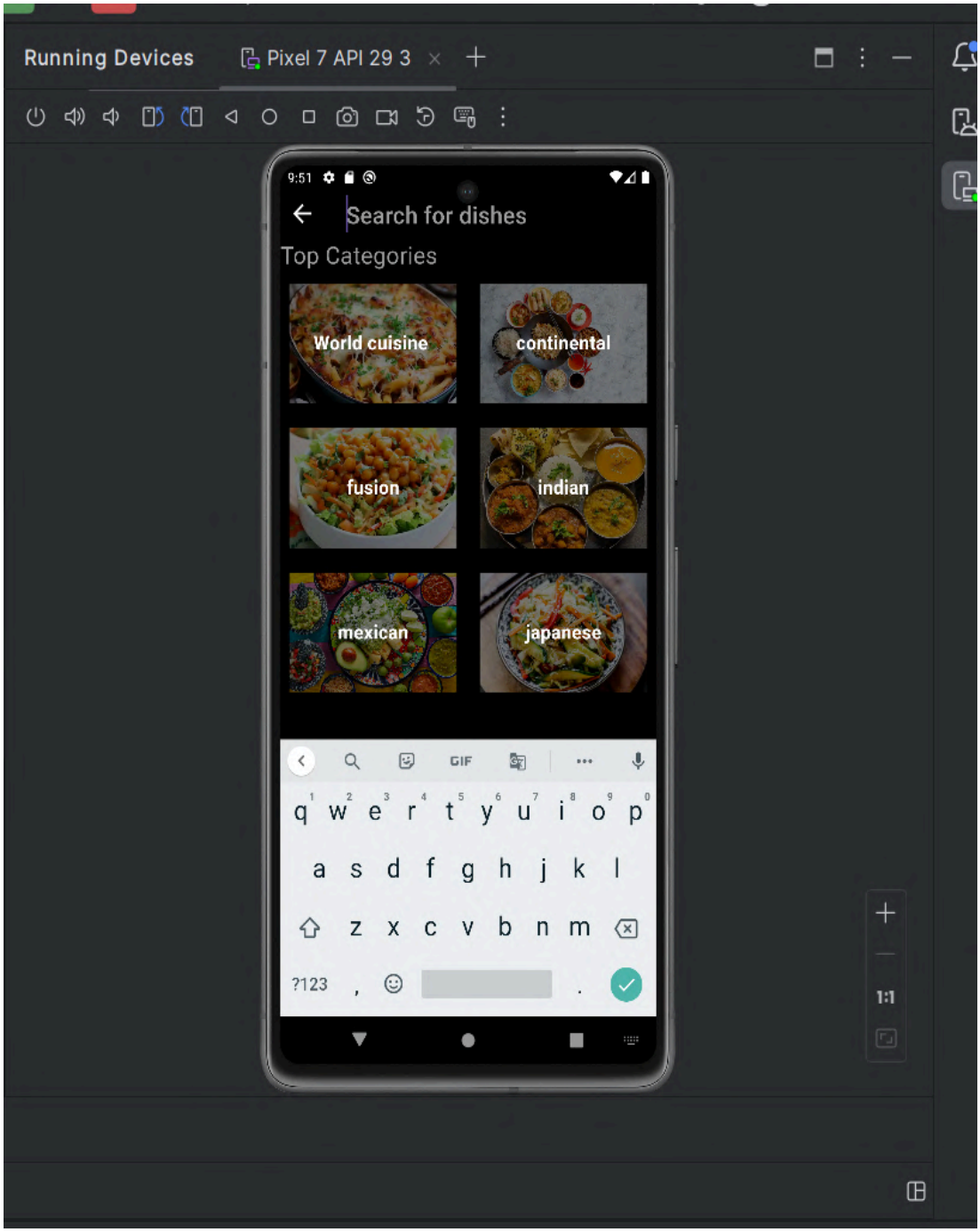
```
import 'package:flutter/material.dart';

class CateGory extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     backgroundColor: Colors.black,
```

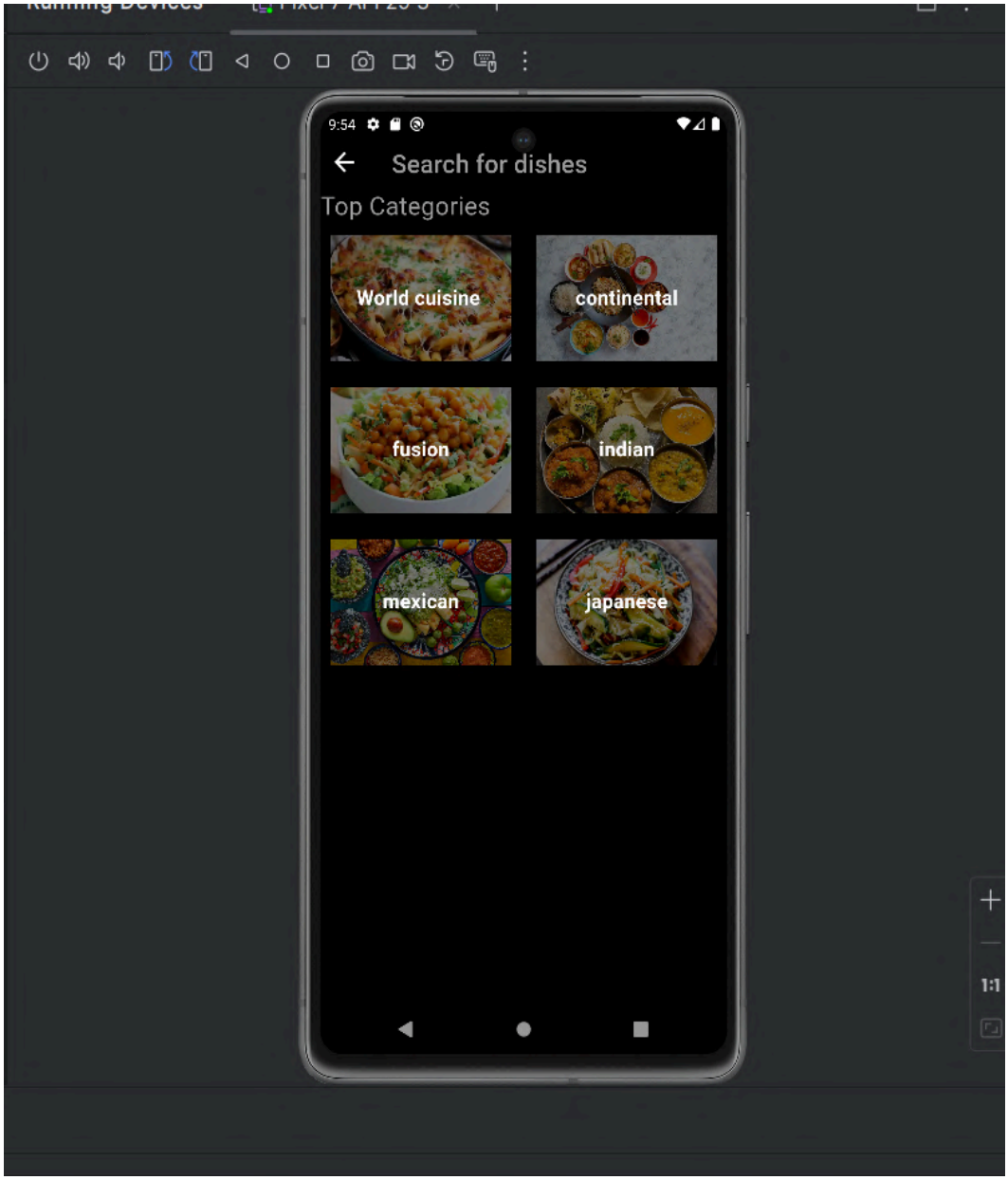
```
appBar: AppBar(
  backgroundColor: Colors.black,
  leading: Row(
    children: [
      IconButton(
        icon: Icon(Icons.arrow_back, size: 30.0, color: Colors.white),
        onPressed: () {
          Navigator.pop(context);
        },
      ),
    ],
  ),
  title: TextField(
    style: TextStyle(color: Colors.white,fontSize: 25),
    decoration: InputDecoration(
      hintText: 'Search for dishes',
      hintStyle: TextStyle(color: Colors.grey,fontSize:25),
      border: InputBorder.none,
    ),
    onChanged: (value) {
      // Implement your search functionality here
      print('Search query: $value');
    },
  ),
  centerTitle: true,
  bottom: PreferredSize(
    preferredSize: Size.fromHeight(30.0),
    child: Align(
      alignment: Alignment.centerLeft,
      child: Text(
        'Top Categories',
        style: TextStyle(color: Colors.grey, fontSize: 25),
      ),
    ),
  ),
),
body: Container(
```

```
      child: Padding(
        padding: const EdgeInsets.all(10.0),
        child: GridView(
          gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
            crossAxisCount: 2,
            mainAxisSpacing: 25,
            crossAxisSpacing: 25,
            childAspectRatio: 3 / 2, // Adjust the aspect ratio for rectangles
          ),
          children: [
            // Rectangular grid items with images and text
            buildGridItem('World cuisine ', 'assets/images/first.jpg'),
            buildGridItem('continental', 'assets/images/second.jpg'),
            buildGridItem('fusion', 'assets/images/fusion.jpg'),
            buildGridItem('indian', 'assets/images/forth.jpg'),
            buildGridItem('mexican', 'assets/images/mexian.jpg'),
            buildGridItem('japanese', 'assets/images/japaneese.jpg'),
          ],
        ),
      ),
    ),
  );
}

          Center(
            child: Text(
              text,
              style: TextStyle(
                color: Colors.white,
                fontSize: 20,
                fontWeight: FontWeight.bold,
              ),
            ),
          ),
        ],
      ),
    ),
```

```
    );
  }
}
```

OUTPUT :


Running Devices
Pixel 7 API 29 3
9:51
Search for dishes
Top Categories
World cuisine
continental
fusion
indian
mexican
japanese
GIF
1:1



Conclusion -

I learnt about the widgets in flutter like container, buttons,icons ,images etc and successfully implemented them.

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	35
Name	SEJAL MAURYA
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	14M

Experiment No -04
Exploring Form Widgets

Aim - To create interactive forms using form widgets.
Theory -

Obscure Text -the obscureText property is often used in text input fields like TextFormField to obscure the entered text, typically for password fields or any other sensitive information.

TextField: The TextField widget is used to collect user input via text entry. It supports various input types, such as text, numbers, passwords, and more, and provides options for customization and validation.

HintStyle - The style of the hint text displayed in form fields using the hintStyle property of the InputDecoration class. This property allows you to define the text style for the hint text, including its color, font size, font weight, and more.

TextInputType - This property is used to indicate to the system what type of input is expected from the user, and it adjusts the keyboard layout accordingly.

1.TextInputType.text: The default keyboard type for general text input.

2.TextInputType.number: A keyboard optimized for numeric input. It typically includes digits, a decimal point, and symbols for arithmetic operations.

3.TextInputType.phone: A keyboard optimized for entering phone numbers. It may include digits, plus sign, and symbols commonly used in phone numbers.

4.TextInputType.emailAddress: A keyboard optimized for entering email addresses. It includes characters commonly used in email addresses, such as "@" and ".".

Code and Output :

```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:food/screen/login_page.dart';
import 'package:food/screen/logincheck.dart';
 // Assuming this is your home page

class NewPage extends StatefulWidget {
  @override
  _NewPageState createState() => _NewPageState();
}

class _NewPageState extends State<NewPage> {
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  final TextEditingController _nameController = TextEditingController();
  final TextEditingController _phoneNumberController = TextEditingController();
  signUp(BuildContext context, String email, String password ,String name ,String
phoneno) async {
    if (email.isEmpty || password.isEmpty || name.isEmpty || phoneno.isEmpty ) {
      return 'Please enter your email and password';
    } else {
      try {
        final UserCredential userCredential = await
FirebaseAuth.instance.createUserWithEmailAndPassword(email: email,
password: password);
        // Store additional user information if needed
        // For example, you can store the user's name and phone number
        // You can access these values using _nameController.text and
_phoneNumberController.text
        // Then store them using userCredential.user.updateProfile() or store them
in Firestore, for example
        Navigator.push(context, MaterialPageRoute(builder: (context) =>
LoginPage1()));
      } on FirebaseAuthException catch (ex) {
        return ex.code.toString();
      }
```

```
  }
 }
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   backgroundColor: Colors.black,
   appBar: AppBar(
    backgroundColor: Colors.black,
    leading: IconButton(
     icon: Icon(Icons.arrow_back, size: 30.0, color: Colors.white),
     onPressed: () {
      Navigator.pop(context);
     },
    ),
    title: Text(
     'Sign Up',
     textAlign: TextAlign.center,
     style: TextStyle(color: Colors.white, fontSize: 30),
    ),
    centerTitle: true,
   ),
   body: Column(
    children: [
     SizedBox(height: 20),
     buildTextField(_emailController, "Email"),
     SizedBox(height: 20),
     buildTextField(_nameController, "Name"),
     SizedBox(height: 20),
     buildTextField(_phoneNumberController, "10 Digit Mobile Number"),
     SizedBox(height: 20),
     buildTextField(_passwordController, "Password", obscureText: true),
     SizedBox(height: 40),
     Expanded(
      child: Align(
       alignment: Alignment.bottomCenter,
       child: GestureDetector(
        onTap: () {
```

```
                signUp(context, _emailController.text,
_passwordController.text,_nameController.text,_phoneNumberController.text);
              },
              child: Container(
                height: 50,
                color: Colors.orange[900],
                child: Center(
                  child: Text(
                    'Sign Up',
                    style: TextStyle(color: Colors.white, fontSize: 30),
                  ),
                ),
              ),
            ),
          ),
        ),
      ],
    ),
  );
 }

  Widget buildTextField(TextEditingController controller, String hintText, {bool
obscureText = false}) {
    return Padding(
      padding: EdgeInsets.symmetric(horizontal: 5),
      child: TextField(
        controller: controller,
        style: TextStyle(color: Colors.white, fontSize: 20),
        keyboardType: TextInputType.text,
        obscureText: obscureText,
        decoration: InputDecoration(
          hintText: hintText,
          hintStyle: TextStyle(color: Colors.grey[800], fontSize: 20, fontWeight:
FontWeight.w400),
          focusedBorder: UnderlineInputBorder(
            borderSide: BorderSide(color: Colors.white),
          ),
```

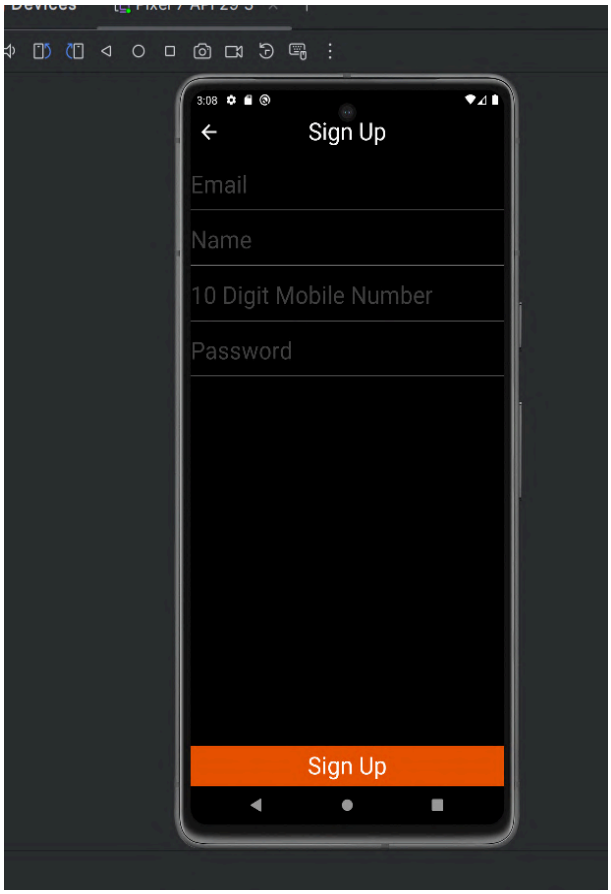
```
        ),
      ),
    );
  }
}
```

LoginPage.dart

```
import 'package:flutter/material.dart';
class LoginPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        backgroundColor: Colors.black,
        appBar: AppBar(
          backgroundColor: Colors.black,
          leading: Row(
            children: [
              IconButton(
                icon: Icon(Icons.arrow_back, size: 30.0, color: Colors.white,),
onPressed: () {Navigator.pop(context);},
              ),
            ],
          ),
          title: Text('Welcome to Fres',style: TextStyle(color: Colors.white ,fontSize:
25),),
        ),
        body: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            SizedBox(height: 15.0), // Add space between text and image
            Container(
              color: Colors.black, // Set your desired background color
              child: Image.asset(
                'assets/images/logo.png', // Replace with your image URL
                width: 200.0, // Set your desired width
```

```
                height: 200.0, // Set your desired height
                colorBlendMode: BlendMode.clear,
              ),
            ),
          ],
        ),

      ),
    );
  }
}
```

Output :



Conclusion - I learnt about the form widgets and successfully implemented them.

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	35
Name	SEJAL MAURYA
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	14M

Experiment No -05
Navigation ,Routing and Gestures

Aim - To Apply navigation ,routing and gestures in Flutter App.

Theory -

Navigation:

- Navigation refers to moving between different screens or pages within a Flutter app.
- Flutter provides a Navigator widget to manage navigation between routes.
- Routes are identified by a unique string, which is used to push or pop routes onto/from the navigation stack.

Routing:

- Routing in Flutter refers to the process of defining and managing the routes within your app.
- You can define routes using a MaterialApp widget's routes property or by using named routes with the Navigator.
- With named routes, you define a mapping of route names to builder functions that return the corresponding widgets for each route.

Gestures:

Flutter provides gesture recognizers like GestureDetector for handling user gestures such as taps, swipes, etc. Here's a simple example of detecting a tap gesture.

Code And Output :
Foodinfo.dart

```
import 'package:flutter/material.dart';
import 'package:food/cart/cart2.dart';
import 'package:food/screen/category.dart';
import 'package:food/widgets/item_widgets.dart';
import 'package:food/widgets/item_widgets2.dart';
import 'package:food/widgets/item_widget2.dart';
import 'package:food/widgets/item_widgets3.dart';
import 'package:food/widgets/item_widgets4.dart';
import 'package:food/widgets/item_widgets5.dart';
import 'package:food/widgets/item_widgets6.dart';
import 'package:food/widgets/item_widgets7.dart';

class FoodInfo extends StatefulWidget {
  @override
  _FoodInfoState createState() => _FoodInfoState();
}

class _FoodInfoState extends State<FoodInfo> {
  int _currentIndex = 0;

  // Define the list of headings for the containers
  List<String> containerHeadings = [
    "Beverages and Cakes",
    "Breakfast",
    "Lunch",
    "Noddles",
    "Sandwitch",
    "Soup and Salad",
  ];

  // List of routes corresponding to each tab
  List<Widget> _tabRoutes = [
    // Add your routes here for each tab
```

```
    ItemWidget2(), // For example
    // Dummy route for demonstration, replace with your desired routes
    ItemWidgets3(),
    ItemWidgets4(),
    ItemWidgets5(),
    ItemWidgets6(),
    ItemWidgets7(),
  ];

  @override
  Widget build(BuildContext context) {
    return DefaultTabController(
      length: 6,
      child: Scaffold(
        backgroundColor: Colors.black,
        appBar: AppBar(
          backgroundColor: Colors.black,
          iconTheme: IconThemeData(
            color: Colors.white, // Change the color of the drawer icon here
          ),
          actions: [
            Padding(
              padding: const EdgeInsets.symmetric(horizontal: 10.0),
              child: Container(
                width: MediaQuery.of(context).size.width * 0.7,
                child: TextField(
                  style: TextStyle(color: Colors.white),
                  decoration: InputDecoration(
                    hintText: 'Menu is for ....',
                    hintStyle: TextStyle(color: Colors.white.withOpacity(0.5), fontSize:
25),
                    border: InputBorder.none,
                  ),
                  cursorColor: Colors.white,
                  // Add any search functionality here
                ),
              ),
```

```
          ),
        ],
      ),
      drawer: Drawer(
        backgroundColor: Colors.black,
        child: ListView(
          children: [
            ListTile(
              title: const Text("Address",style: TextStyle(color: Colors.white),),
              onTap: () {},
            ),
            Divider(
              color: Colors.white,
              thickness: 1,
            ),
            ListTile(
              title: const Text("Personal Information",style: TextStyle(color:
Colors.white),),
              onTap: () {},
            ),
            Divider(
              color: Colors.white,
              thickness: 1,
            ),
            ListTile(
              title: const Text("View Cart",style: TextStyle(color: Colors.white),),
              onTap: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(builder: (context) => (CartView1())),
                );


              }
              ,
            ),
          ],
```

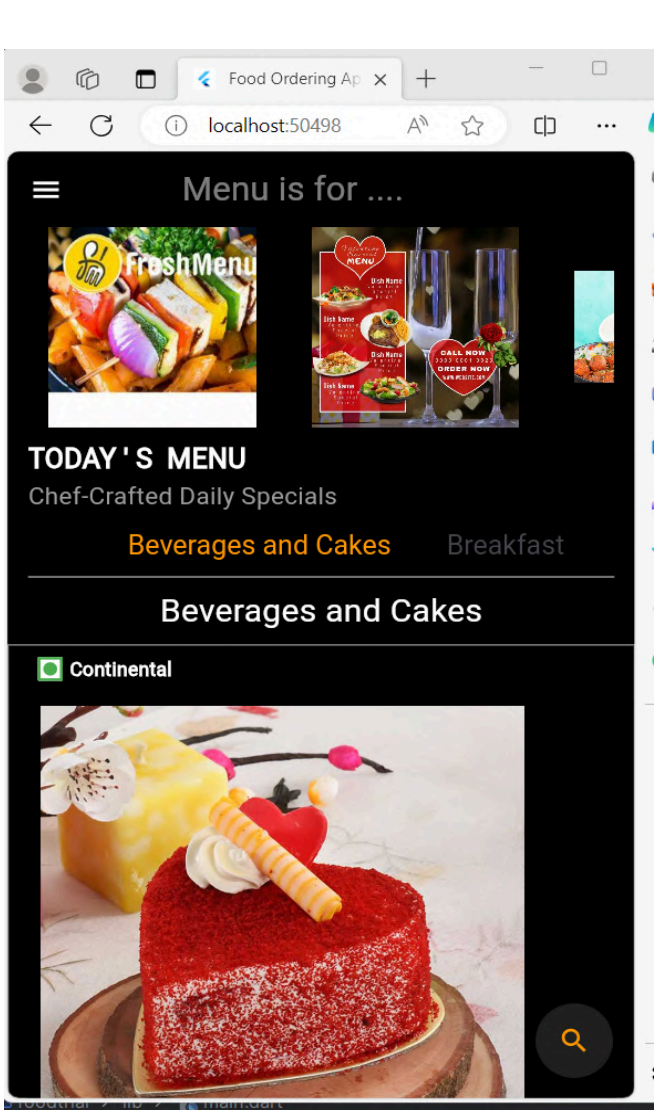
```
  ),
),
body: SafeArea(
  child: Stack(
    children: [
      Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 15.0),
            child: SingleChildScrollView(
              scrollDirection: Axis.horizontal,
              child: Row(
                children: [
                  SizedBox(height: 50),
                  Padding(
                    padding: EdgeInsets.symmetric(horizontal: 15),
                    child: Image(
                      image: AssetImage('assets/images/upper1.jpg'),
                      width: 150,
                      height: 150,
                      fit: BoxFit.cover,
                    ),
                  ),
                  SizedBox(width: 10),
                  Padding(
                    padding: EdgeInsets.symmetric(horizontal: 15),
                    child: Image(
                      image: AssetImage('assets/images/valentine.jpg'),
                      width: 150,
                      height: 150,
                      fit: BoxFit.contain,
                    ),
                  ),
                  SizedBox(width: 10),
                  // Add more images here
                  Padding(
```

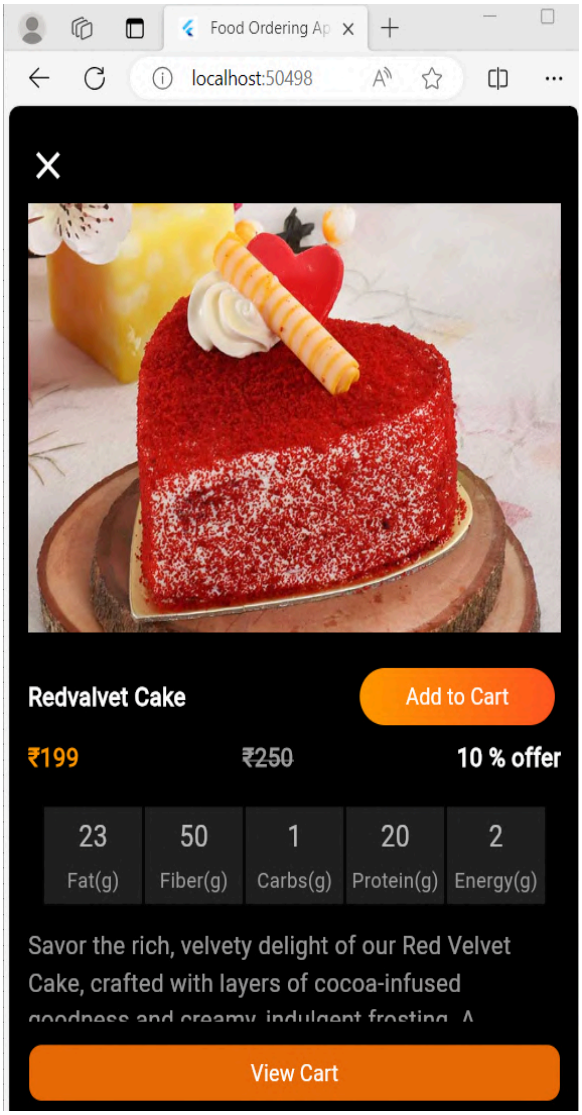
```
                    padding: EdgeInsets.symmetric(horizontal: 15),
                    child: Image(
                      image: AssetImage('assets/images/dinner.jpg'),
                      width: 150,
                      height: 150,
                      fit: BoxFit.contain,
                    ),
                  ),
                  // Add more images as needed
                ],
              ),
            ),
          ),
          SizedBox(height: 10),
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 15.0),
            child: Text(
              "TODAY ' S  MENU",
              style: TextStyle(color: Colors.white, fontSize: 20, fontWeight:
FontWeight.bold),
            ),
          ),
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 15.0),
            child: Text(
              "Chef-Crafted Daily Specials",
              style: TextStyle(color: Colors.grey, fontSize: 18),
            ),
          ),
          Padding(
            padding: EdgeInsets.symmetric(horizontal: 15),
            child: TabBar(
              isScrollable: true,
              indicator: BoxDecoration(),
              labelStyle: TextStyle(fontSize: 20,color: Colors.orange),
              labelPadding: EdgeInsets.symmetric(horizontal: 20),
              tabs: [
```

```
      for (var heading in containerHeadings)
        Tab(text: heading),
    ],
    onTap: (index) {
      setState(() {
        _currentIndex = index;
      });
      // Navigate to corresponding page
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => _tabRoutes[index]),
      );
    },
  ),
),
Flexible(
  flex:1,
  child:TabBarView(
    children: [
      for (var i = 0; i < containerHeadings.length; i++)
        _currentIndex == i
            ? Container(
          color: Colors.black,
          child: Column(
            children: [
              Padding(
                padding: const EdgeInsets.all(8.0),
                child: Text(
                  containerHeadings[i],
                  style: TextStyle(color: Colors.white, fontSize: 24),
                ),
              ),
              Expanded(
                child: Container(
                  decoration: BoxDecoration(
                    border: Border.all(color: Colors.grey),
                  ),
```

```
                            child: ItemWidgets(),
                          ),
                        ),
                      ],
                    ),
                  )
                      : Container(),
                ],
              ),
            ),
          ],
        ),
        Positioned(
          bottom: 16,
          right: 16,
          child: FloatingActionButton(
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => (CateGory())),
              );
            },
            child: Icon(Icons.search, color: Colors.orange),
            backgroundColor: Colors.grey[900], // Change the color as needed
            shape: CircleBorder(), // Make the button circular
          ),
        ),
      ],
    ),
  ),
),
);
}
}
```

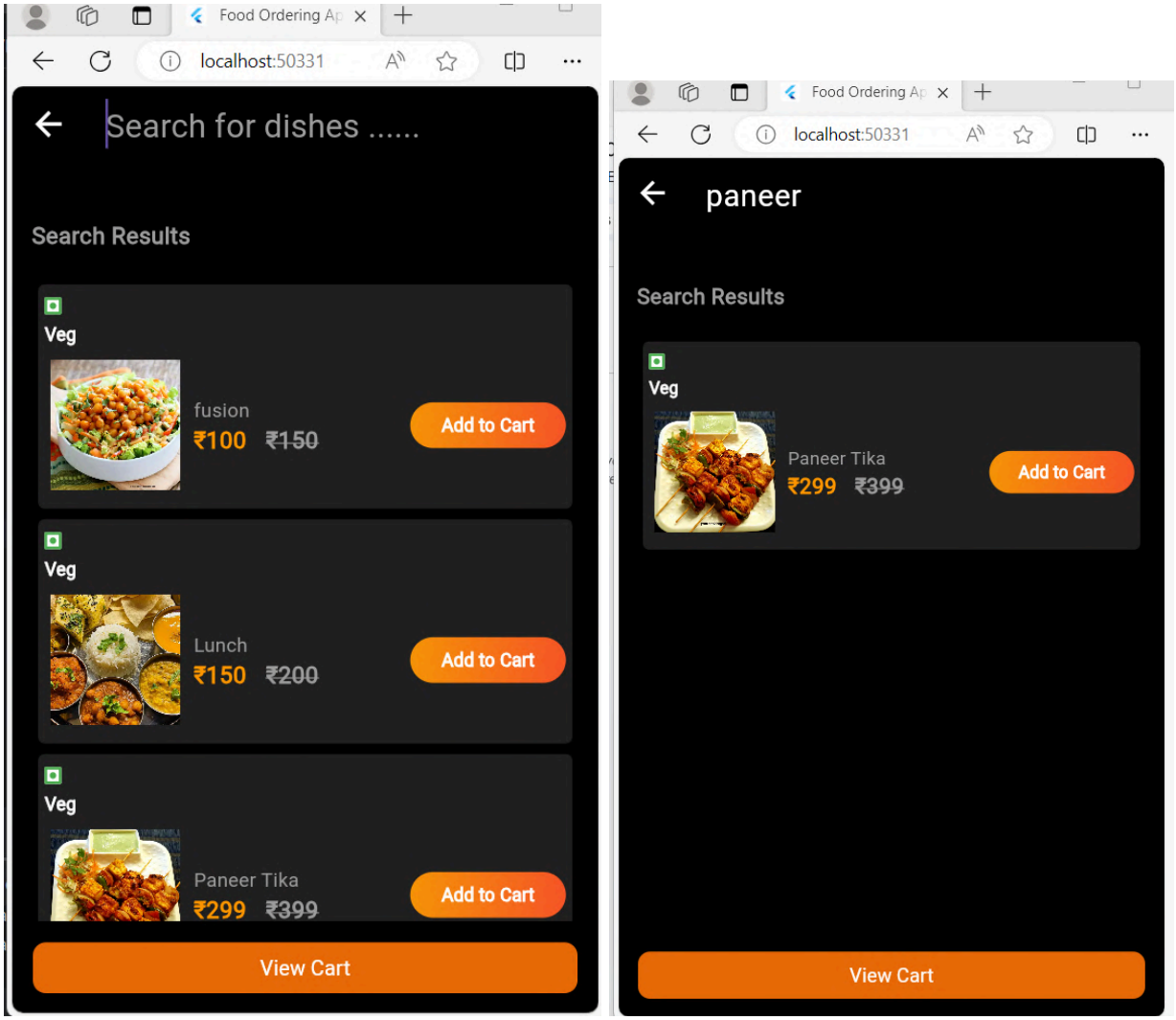
Output





After click on the image I change from this page to another page using the navigation

Other Functionality - like search button is also working
Here when I am searching for any specific food it will show the results here.



Conclusion - In this experiment I learned how to navigate from one page to another using Navigation.push() and successfully implemented it.

MAD & PWA Lab
Journal

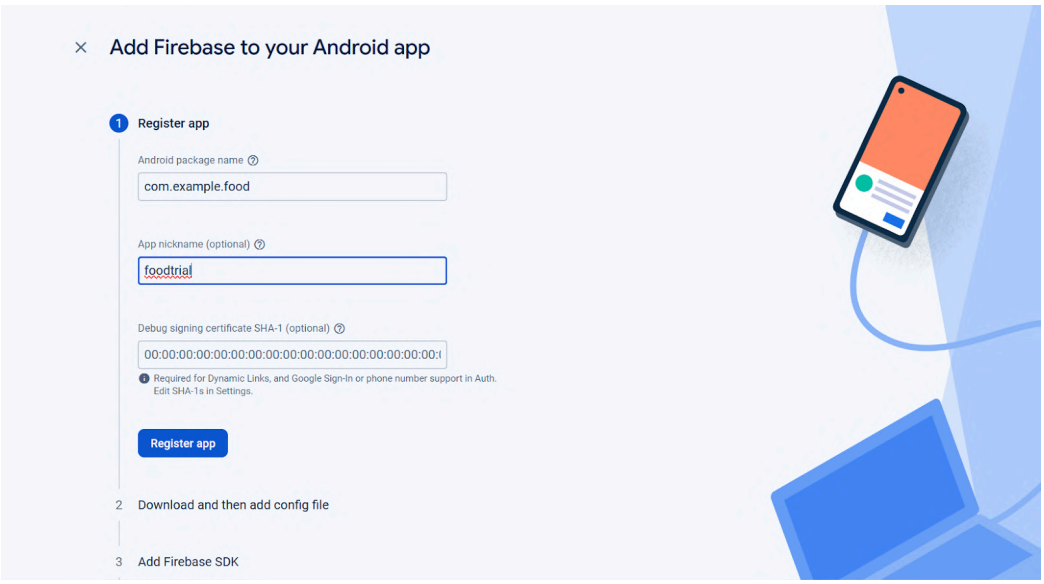
Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	35
Name	SEJAL MAURYA
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	14M

Experiment No -06
Firebase Database

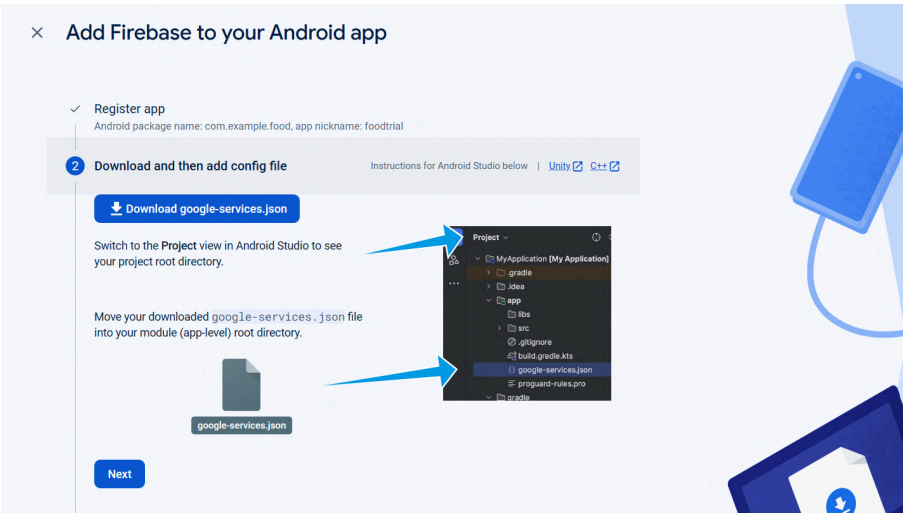
Aim - To connect flutter UI with Firebase database.

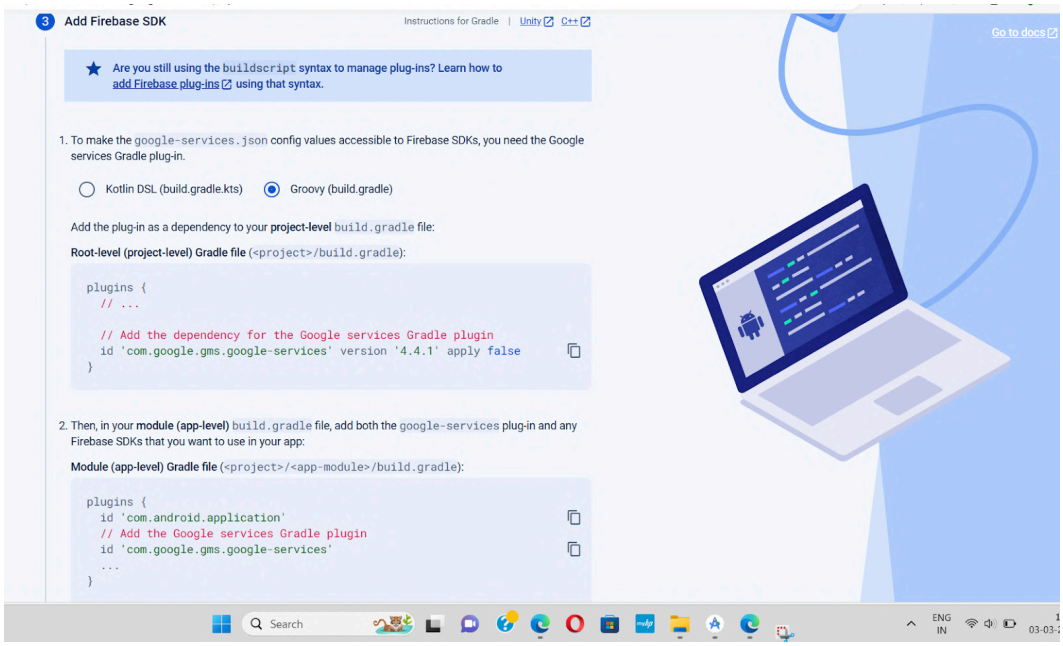
Theory

Firebase Realtime Database is a powerful and easy-to-use solution for developers looking to add real-time data synchronization and offline support to their web and mobile applications. It simplifies the process of building real-time collaborative features and allows developers to focus on creating engaging user experiences.



Download the google-services.json

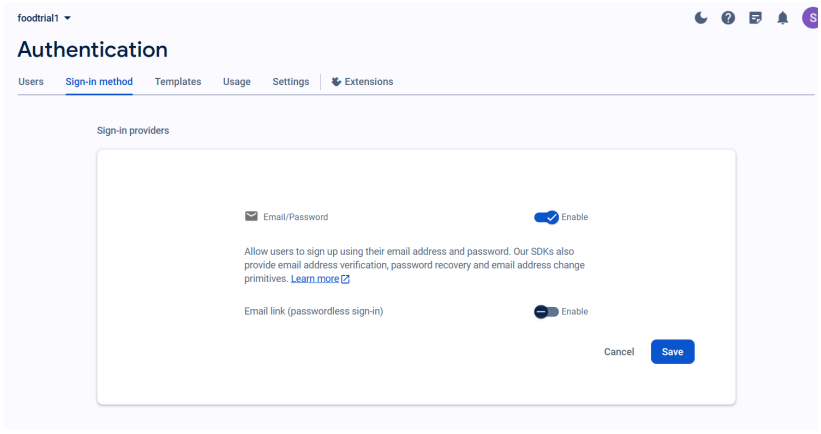




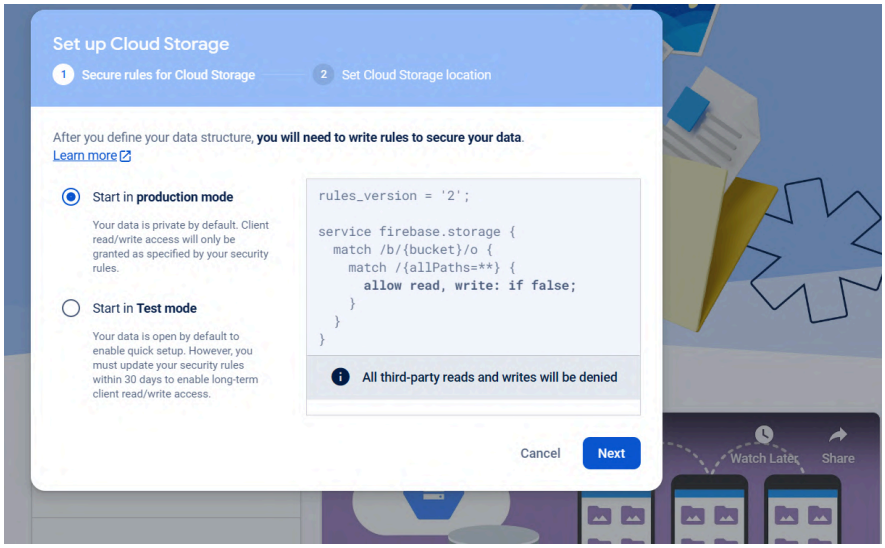
Add dependencies in project-level gradle and app-level gradle.

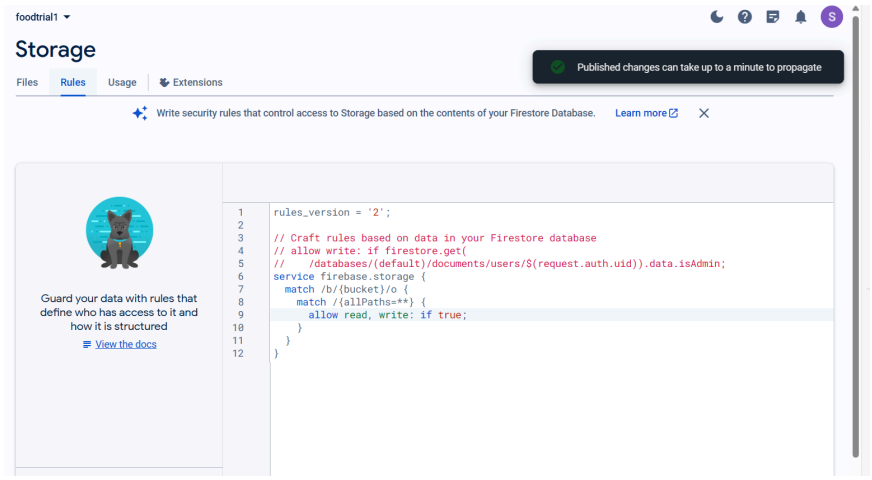
```
dependencies {
    classpath 'com.google.gms:google-services:4.4.1'
    classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
}
}
```

```
dependencies {
    implementation platform('com.google.firebase:firebase-bom:32.7.3')
    implementation 'com.google.firebase:firebase-analytics'
}
```



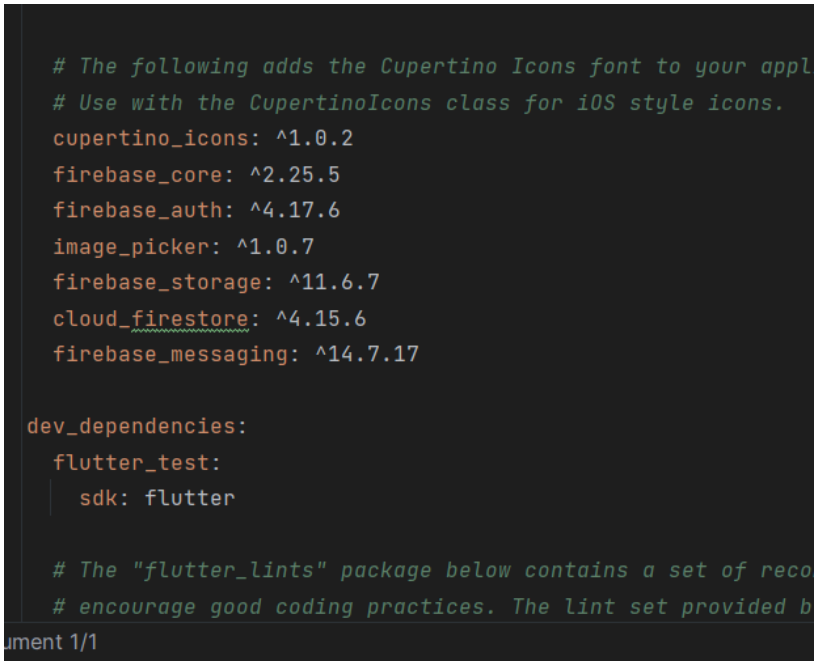
Project Title: Flutter : FreshMenu / PWA : Ecommerce Website **Roll No. 35**






```dart
import 'package:food/extra/cartpage.dart';
void main()async{
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
      options: FirebaseOptions(
          apiKey:"AIzaSyASrFb41V8qhwpAFuYA9hkzzpzIl-sjjj4",
          appId: "1:116578659366:android:df8de54c41f6bdee97cae7",
          messagingSenderId: "116578659366",
          projectId: "foodtrial1-e8b96")
  );
  runApp(Myapp());
}
```

Add these dependencies in pubspec.yml



```
  # The following adds the Cupertino Icons font to your appl
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.2
  firebase_core: ^2.25.5
  firebase_auth: ^4.17.6
  image_picker: ^1.0.7
  firebase_storage: ^11.6.7
  cloud_firestore: ^4.15.6
  firebase_messaging: ^14.7.17

dev_dependencies:
  flutter_test:
    sdk: flutter

  # The "flutter_lints" package below contains a set of reco
  # encourage good coding practices. The lint set provided b
ument 1/1
```

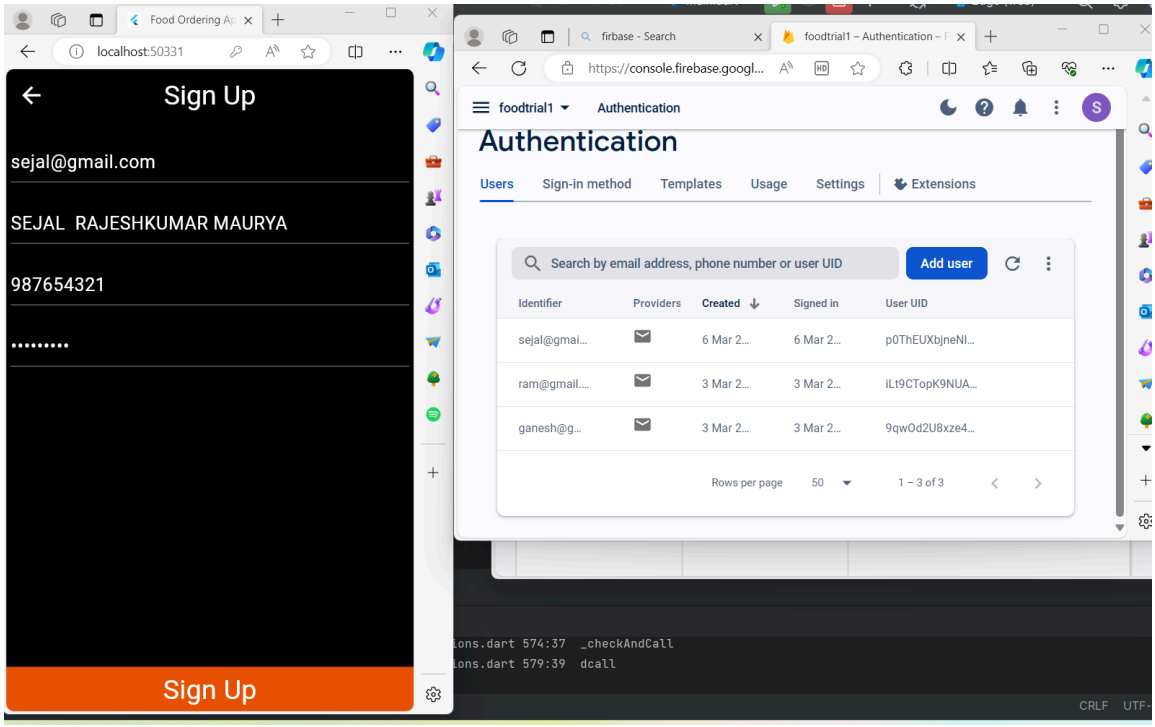
Code -

```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:food/screen/foodinfo.dart';

class LoginPage1 extends StatefulWidget {
  @override
  _LoginPageState createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage1> {
  TextEditingController _emailController = TextEditingController();
  TextEditingController _passwordController = TextEditingController();
  final FirebaseAuth _auth = FirebaseAuth.instance;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      appBar: AppBar(
        backgroundColor: Colors.black,
        leading: Row(
          children: [
            IconButton(
```

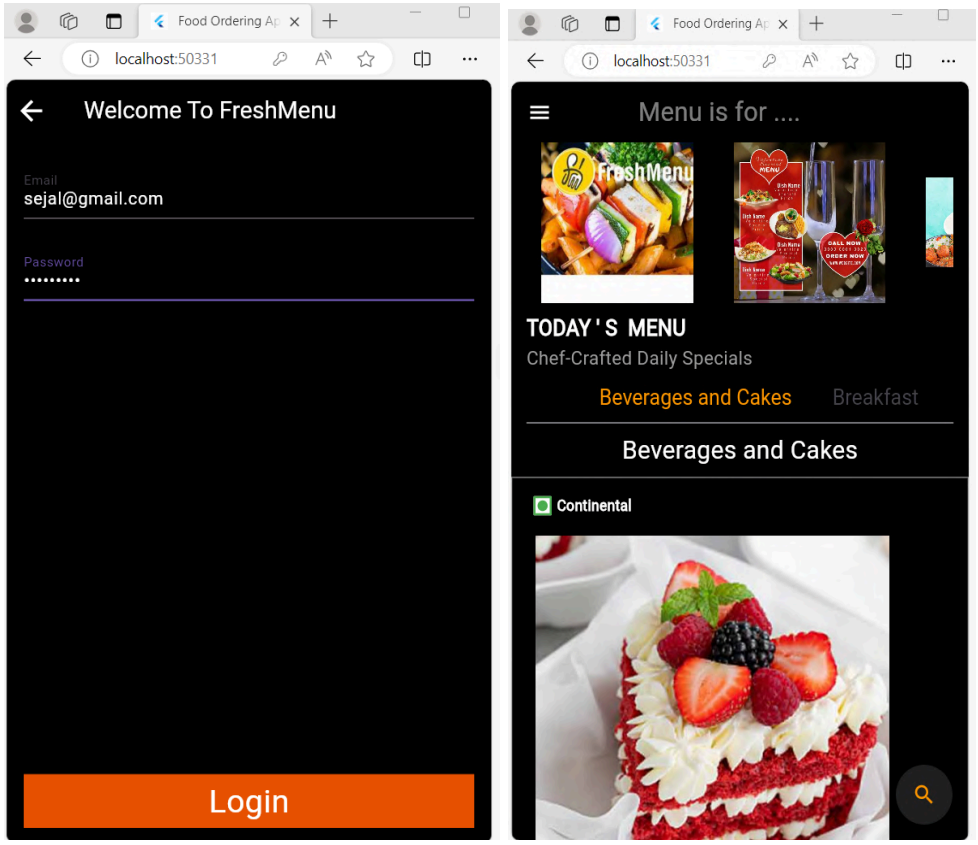
```
          icon: Icon(Icons.arrow_back, size: 30.0, color: Colors.white,),
          onPressed: () {Navigator.pop(context);},
        ),
      ],
    ),
    title: Text('Welcome To FreshMenu', style: TextStyle(color: Colors.white),),
  ),
  body: Padding(
    padding: const EdgeInsets.all(16.0),
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        TextField(
          style: TextStyle(color: Colors.white),
          controller: _emailController,
          decoration: InputDecoration(
            labelText: 'Email',
          ),
        ),
        SizedBox(height: 20),
        TextField(
          style: TextStyle(color: Colors.white),
          controller: _passwordController,
          decoration: InputDecoration(
            labelText: 'Password',
          ),
          obscureText: true,
        ),
        SizedBox(height: 20),
        Expanded(
          child: Align(
            alignment: Alignment.bottomCenter,
            child: GestureDetector(
              onTap: _login,
              child: Container(
                height: 50,
                color: Colors.orange[900],
                child: Center(
                  child: Text(
                    'Login',
```

```
                    style: TextStyle(color: Colors.white, fontSize: 30),
                  ),
                ),
              ),
            ),
          ),
        ),
      ],
    ),
  ),
 );
}

void _login() async {
 try {
  final String email = _emailController.text.trim();
  final String password = _passwordController.text.trim();

  // Sign in with email and password
  final UserCredential userCredential =
  await _auth.signInWithEmailAndPassword(
   email: email,
   password: password,
  );

  // Check if the user is authenticated
  if (userCredential.user != null) {
   // Navigate to the home page if authentication is successful
   Navigator.pushReplacement(
    context,
    MaterialPageRoute(builder: (context) => FoodInfo()),
   );
  }
 } catch (e) {
  // Handle errors such as invalid credentials, network errors, etc.
  print('Error: $e');
  // You can show a snackbar or dialog to display the error to the user
 }
}
}
```

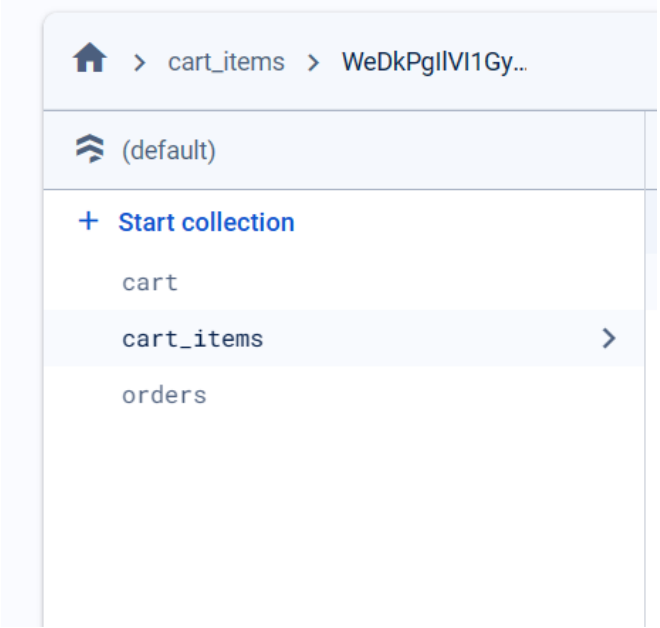
Authentication



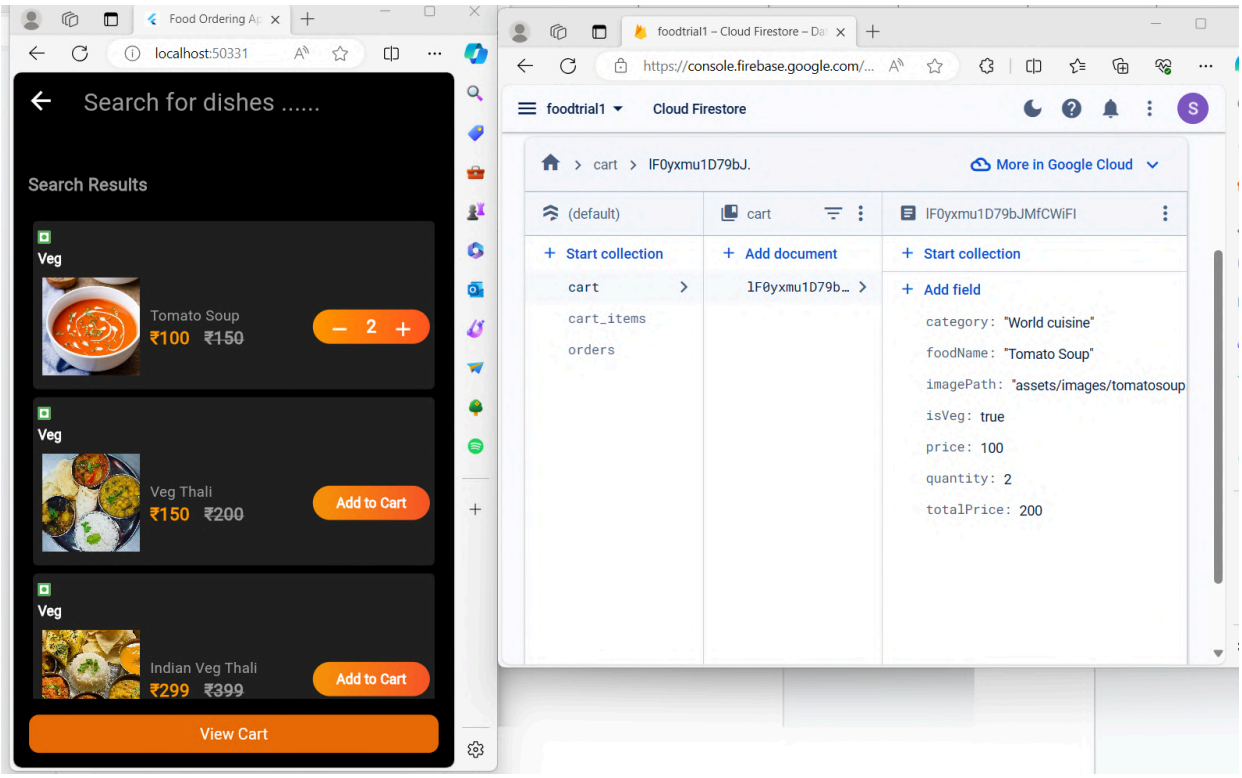
And then when login into the page with right email and password I can moved to the home page



Other database are cart ,cart items, orders

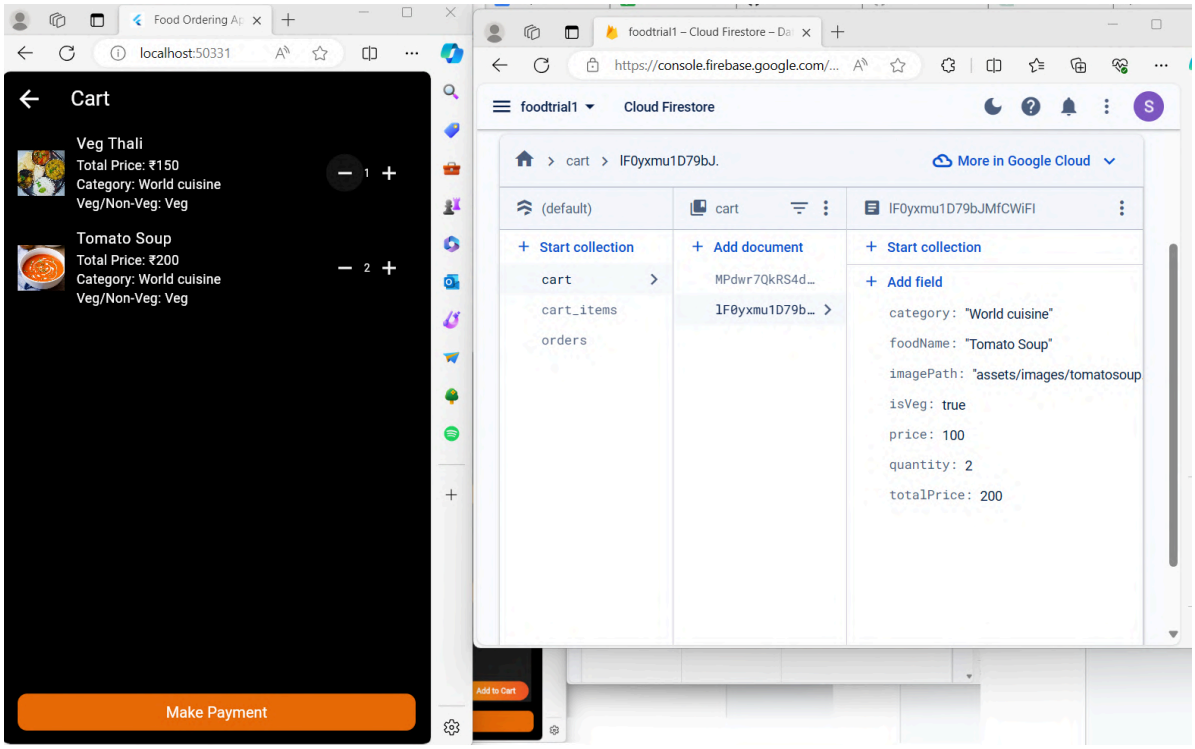


When I added the food items into the cart it stores into the cart database
Here the data is fetch from flutter UI to database

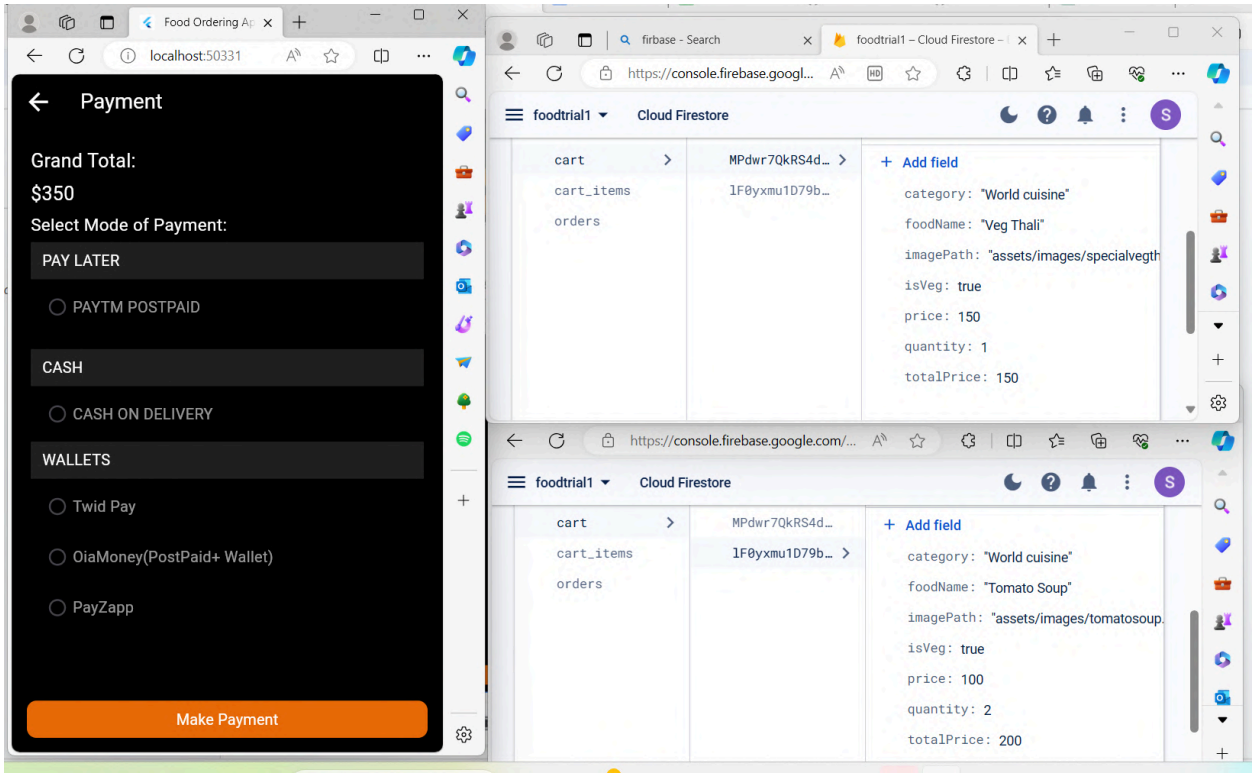


Project Title: Flutter : FreshMenu / PWA : Ecommerce Website **Roll No. 35**

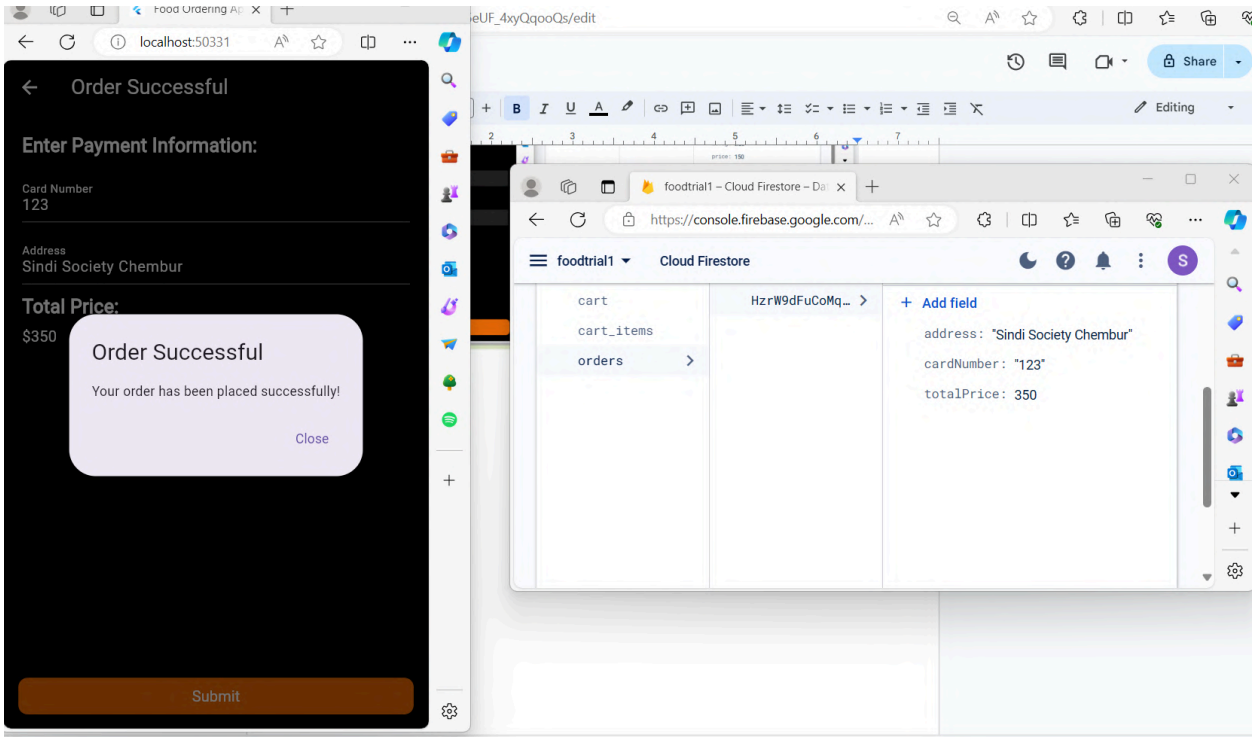
Here the data is fetch from the cloud store to Flutter UI and If I will update the items in flutter UI it also updated in Firebase



Here the data is fetched and after adding the total price it reflects on the payment page. So that the user need not to add calculate the total payment.



Order related information is also stored in the order database.



Conclusion - I learnt about the firebase and successfully implemented it in my flutter project.

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.	35
Name	SEJAL MAURYA
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	15M

EXPERIMENT NO -07

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
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 PWAruns 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 o ers 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 e efficiently. Their interaction does not depend on the browser user uses. This eventually improves the chances of notifying the user

regarding your services, o ers, 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.

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.
- 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 errored by service workers.

Code And Output
Index1.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">
<link rel="manifest" href="manifest.json">
<title>JKS</title>

<link rel="stylesheet" type="text/css" href="css/settings.css">
<!--Features Section-->
<section class="feature_wrap padding-half" id="specialities">
  <div class="container">
    <div class="row">
     <div class="col-md-12 text-center">
        <h2 class="heading ">Our   Specialities</h2>
        <hr class="heading_space">
      </div>
    </div>
    <div class="row">
      <div class="col-md-3 col-sm-6 feature text-center">
        <i class="icon-glass"></i>
        <h3><a href="order.html">Dinner &amp; Dessert</a></h3>
        <p> Enjoy Delicious Food!</p>
```

```
      </div>
      <div class="col-md-3 col-sm-6 feature text-center">
        <i class="icon-coffee"></i>
        <h3><a href="order.html">Breakfast</a></h3>
        <p> Enjoy Delicious Food!</p>
      </div>
      <div class="col-md-3 col-sm-6 feature text-center">
        <i class="icon-glass"></i>
        <h3><a href="order.html">Ice Shakes</a></h3>
        <p> Enjoy Delicious Food!</p>
      </div>
      <div class="col-md-3 col-sm-6 feature text-center">
        <i class="icon-coffee"></i>
        <h3><a href="order.html">Beverges</a></h3>
        <p> Enjoy Delicious Food!</p>
      </div>
    </div>

  </div>
</section>
<!-- Food Gallery -->
<section id="gallery" class="padding">
  <div class="container">
      <div class="row">
     <div class="col-md-12 text-center">
        <h2 class="heading ">Delicious   Food</h2>
        <hr class="heading_space">
        <div class="work-filter">
          <ul class="text-center">
            <li><a href="javascript:;" data-filter="all" class="active filter">All Food</a></li>
            <li><a href="javascript:;" data-filter=".starters" class="filter">Starters</a></li>
            <li><a href="javascript:;" data-filter=".drinks" class="filter">Drinks &
Beverges</a></li>
            <li><a href="javascript:;" data-filter=".dinner" class="filter"> Dinner</a></li>
            <li><a href="javascript:;" data-filter=".lunch" class="filter">Breakfast &
Lunch</a></li>
          </ul>
        </div>
      </div>
    </div>
```

```
        <!--Footer-->
<footer class="padding-top bg_black">
  <div class="container">
    <div class="row">
      <div class="col-md-3 col-sm-6 footer_column">
        <h4 class="heading">Who are we?</h4>
        <hr class="half_space">
        <p class="half_space"></p>
        <p>Launched in Mumbai, JKS has grown from a home project to one of the largest
food aggregators in the world. We are present in 24 countries and 10000+ cities
globally, enabling our vision of better food for more people. We not only connect people
to food in every context but work closely with restaurants to enable a sustainable
ecosystem.</p>
      </div>
      <div class="col-md-3 col-sm-6 footer_column">
        <div class="copyright clearfix">
          <p>Copyright &copy; 2020 JKS. All Right Reserved</p>
          <ul class="social_icon">
              <li><a href="#" class="facebook"><i class="icon-facebook5"></i></a></li>
              <li><a href="#" class="twitter"><i class="icon-twitter4"></i></a></li>
              <li><a href="#" class="google"><i class="icon-google"></i></a></li>
            </ul>
        </div>
      </div>
    </div>
  </div>
</footer>

<a href="#" id="back-top"><i class="fa fa-angle-up fa-2x"></i></a>
<script src="js/revolution.extension.video.min.js"></script>
</body>
</html>
```

Manifest.json

```
{
    "name":"foodorderingapp",
    "short_name":"app",
    "start_url":"index1.html",
    "display":"standalone",
    "background_color":"#5900b3",
```

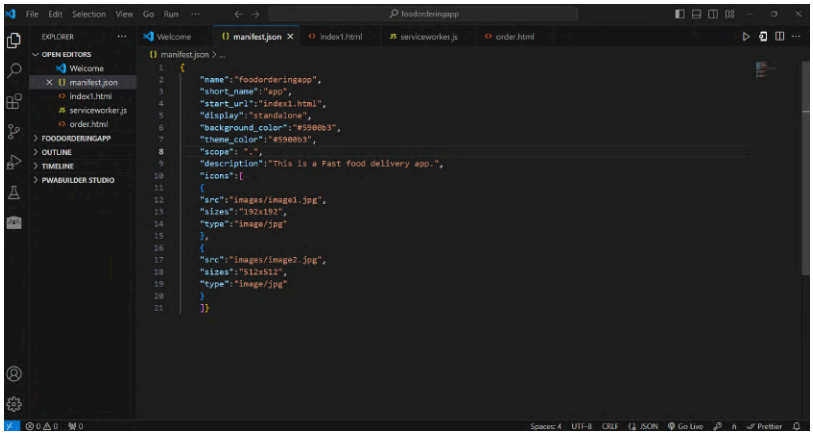
```
"theme_color":"#5900b3",
"scope": ".",
"description":"This is a Fast food delivery app.",
"icons":[
{
"src":"images/image1.jpg",
"sizes":"192x192",
"type":"image/jpg"
},
{
"src":"images/image2.jpg",
"sizes":"512x512",
"type":"image/jpg"
}
]}
```

Serviceworker.json

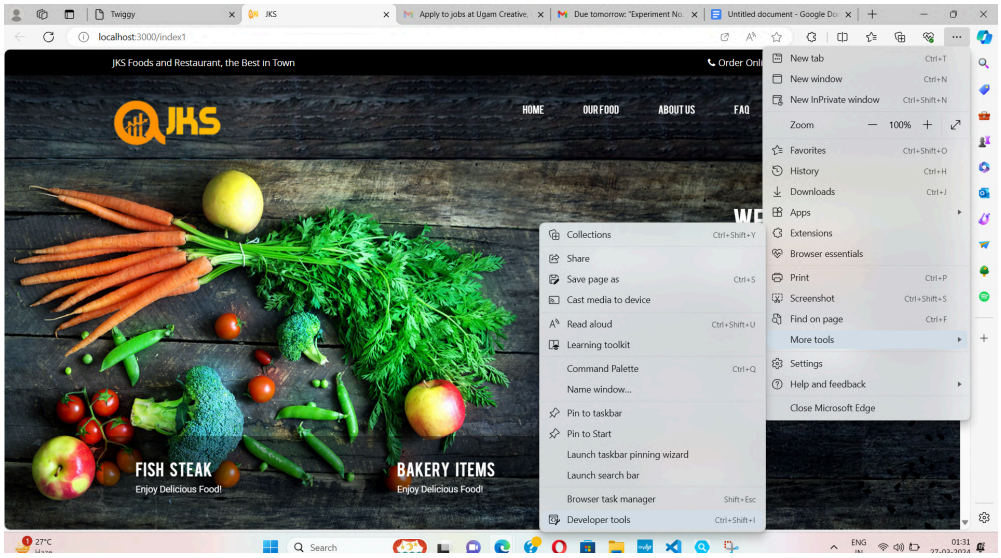
```
var staticCacheName = "foodorderingapp";
self.addEventListener("install", function (e) {
e.waitUntil(
caches.open(staticCacheName).then(function (cache) {
return cache.addAll(["/"]);
})
);
});
self.addEventListener("fetch", function (event) {
console.log(event.request.url);
event.respondWith(
caches.match(event.request).then(function (response) {
return response || fetch(event.request);
})
);
});
```

Project Title: Flutter : FreshMenu / PWA : Ecommerce Website **Roll No. 35**

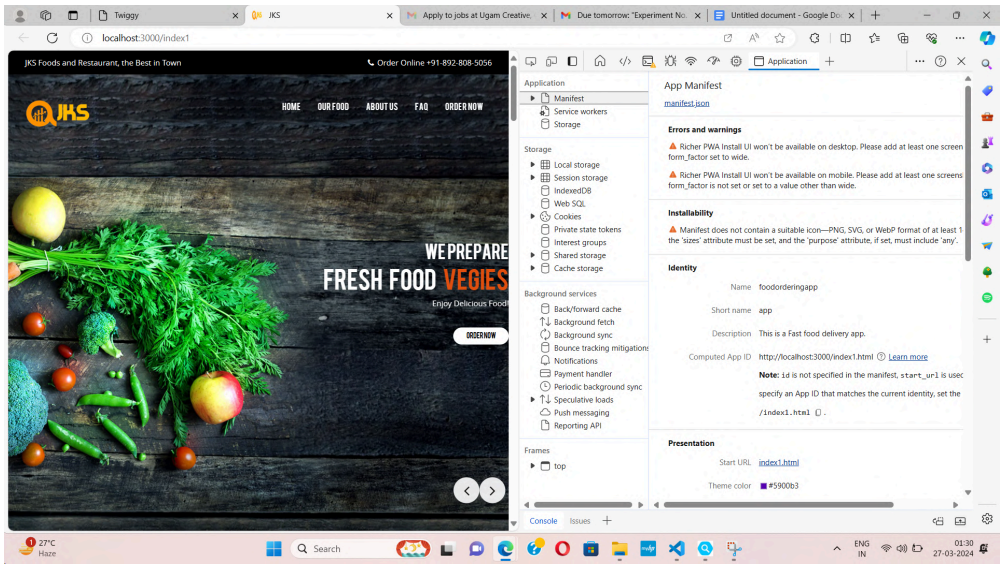
Folder Structure



Start the server

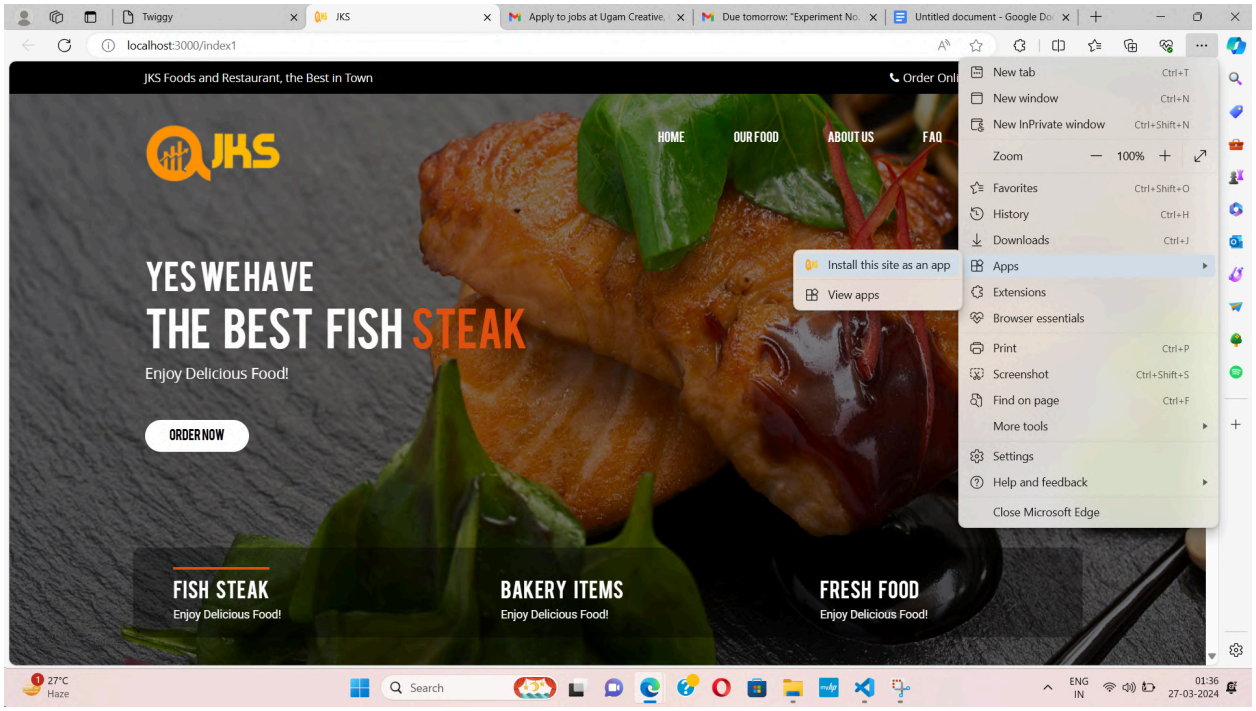


Application

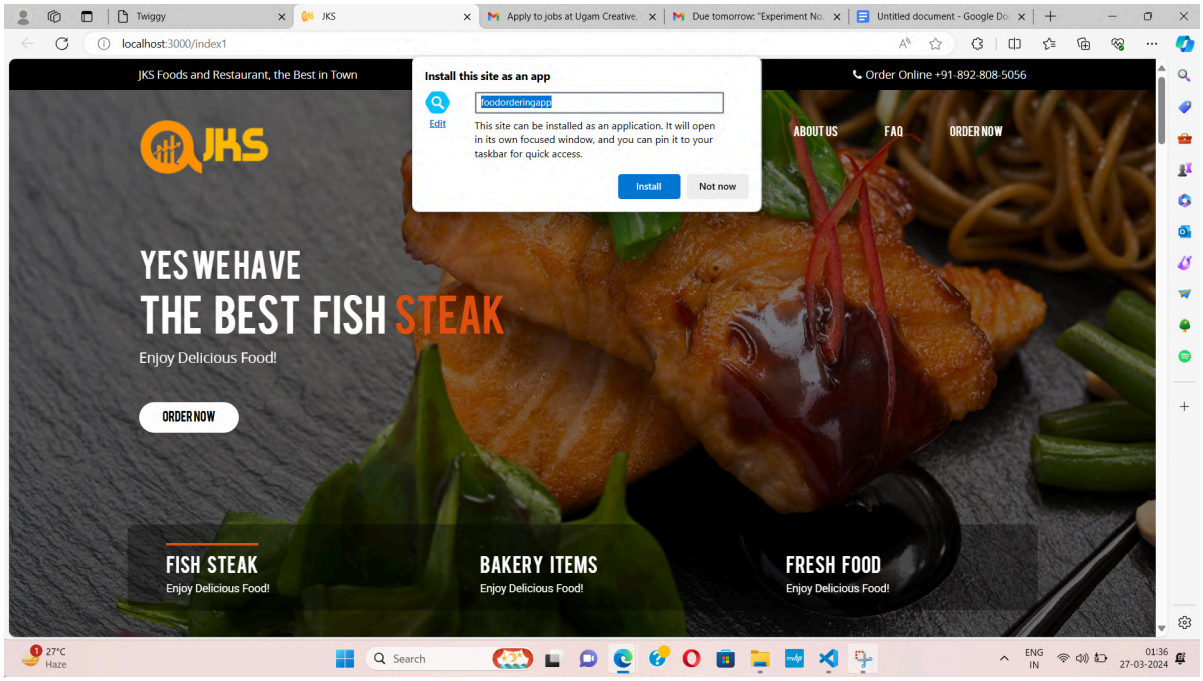


Project Title: Flutter : FreshMenu / PWA : Ecommerce Website **Roll No. 35**

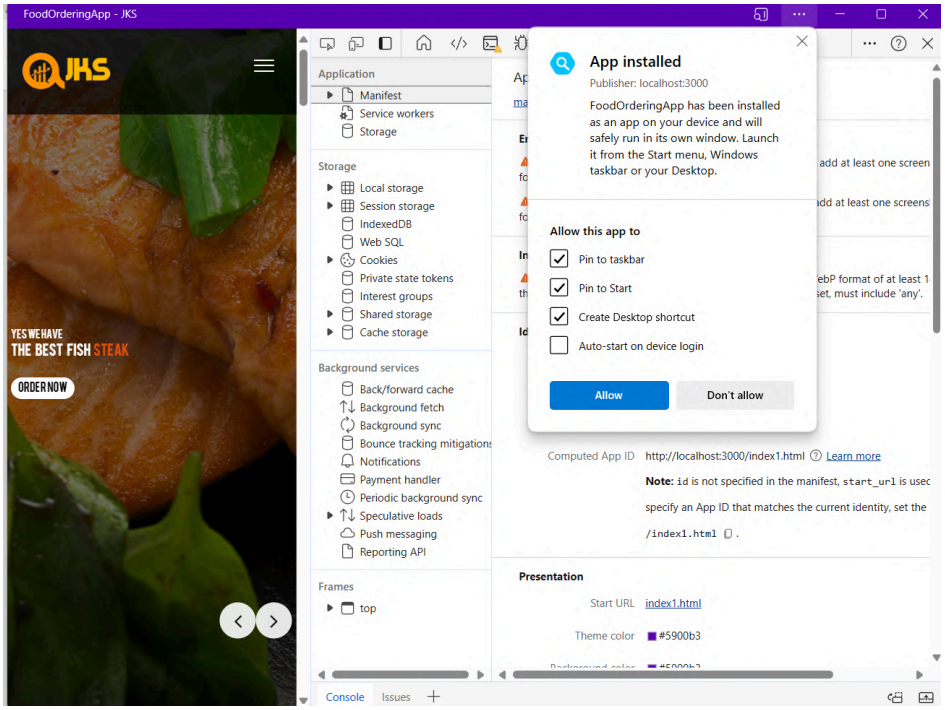
Click on 3 dots—>App—>installApp



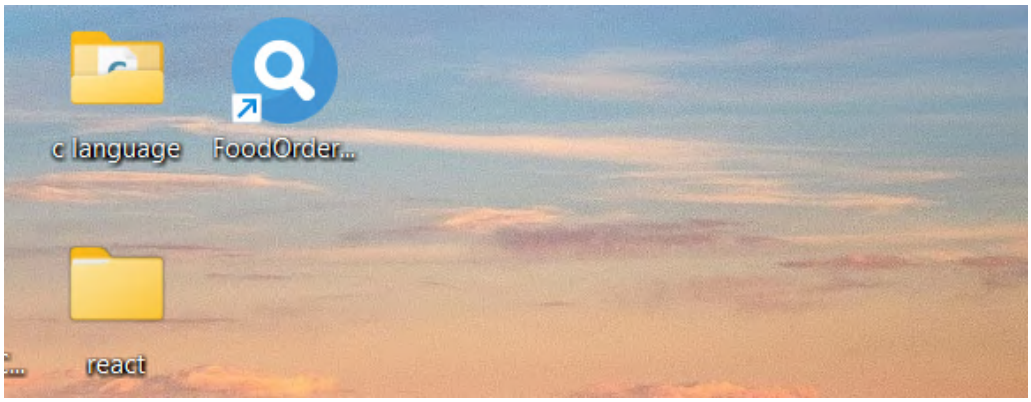
Click on the install



Project Title: Flutter : FreshMenu / PWA : Ecommerce Website **Roll No. 35**



Desktop Icon



Conclusion - I added a home screen feature that was successfully implemented. I created a Shortcut icon to open our website's app locally ,I also added an image to the app icon.

MAD & PWA Lab
Journal

Experiment No.	08
Experiment Title.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA
Roll No.	35
Name	SEJAL MAURYA
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15M

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 dev

Installation

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

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

service-worker.js

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

Activation

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

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

service-worker.js

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

Code And Output :

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>ShoppingMart</title>
    <link rel="stylesheet" href="css/style.css" />
    <link
      rel="stylesheet"

href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,
FILL,GRAD@20..48,100..700,0..1,-50..200"
    />
  </head>
  <body>
    <header>
      <div class="logo_container">
        <a href="#"
          ><img
            class="myntra_home"
            src="images/myntra_logo.webp"
            alt="Myntra Home"
        /></a>
          <div class="search_bar">
        <span class="material-symbols-outlined search_icon">search</span>
        <input
          class="search_input"
          placeholder="Search for products, brands and more"
        />
      </div>
```

```
  <div class="action_bar">
    <div class="action_container">
      <span class="material-symbols-outlined action_icon">person</span>
      <span class="action_name">Profile</span>
    </div>
      <span class="action_name">Bag</span>
      <span class="bag-item-count">0</span>
    </a>
  </div>
</header>
<main>
  <div class="items-container"></div>
</main>
<footer>
  <div class="footer_container">
    <div class="footer_column">
      <h3>ONLINE SHOPPING</h3>


      <a href="#">Men</a>
      <a href="#">Women</a>
      <a href="#">Kids</a>
      <a href="#">Home & Living</a>
      <a href="#">Beauty</a>
      <a href="#">Gift Card</a>
      <a href="#">Myntra Insider</a>
    </div
      <div class="footer_column">
      <h3>ONLINE SHOPPING</h3
      <a href="#">Men</a>
      <a href="#">Women</a>
      <a href="#">Kids</a>
      <a href="#">Home & Living</a>
      <a href="#">Beauty</a>
      <a href="#">Gift Card</a>
      <a href="#">Myntra Insider</a>
    </div>
  </div>
  <hr />
  <div class="copyright">© 2023 www.myntra.com. All rights reserved.</div>
```

```
    </footer>
    <script src="data/items.js"></script>
    <script src="scripts/index.js"></script>
    <script src="app.js"></script>
  </body>
</html>
```

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

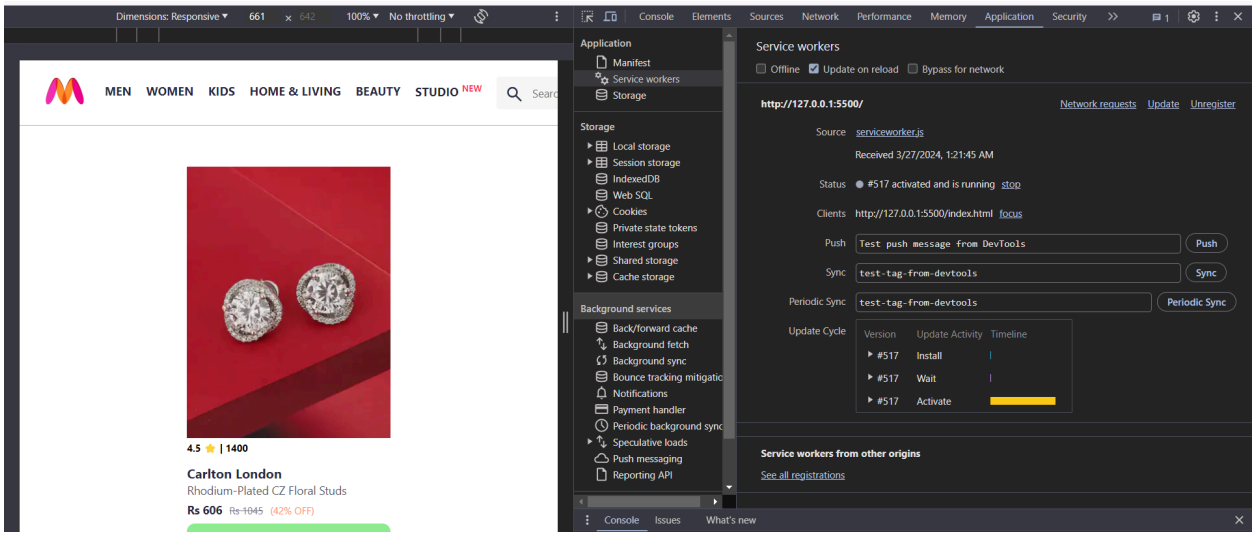
```
const cacheName = "ecommerce-pwa-v1";
const assetsToCache = ["/", "/index.html", "../css/style.css", "/app.js"];
self.addEventListener("install", (event) => {
  event.waitUntil(
    caches.open(cacheName).then((cache) => {
      return cache.addAll(assetsToCache);
    })
  );
});
self.addEventListener("activate", (event) => {
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
```

```
        cacheNames
          .filter((name) => {
            return name !== cacheName;
          })
          .map((name) => {
            return caches.delete(name);
          })
      );
    })
  );
});
```

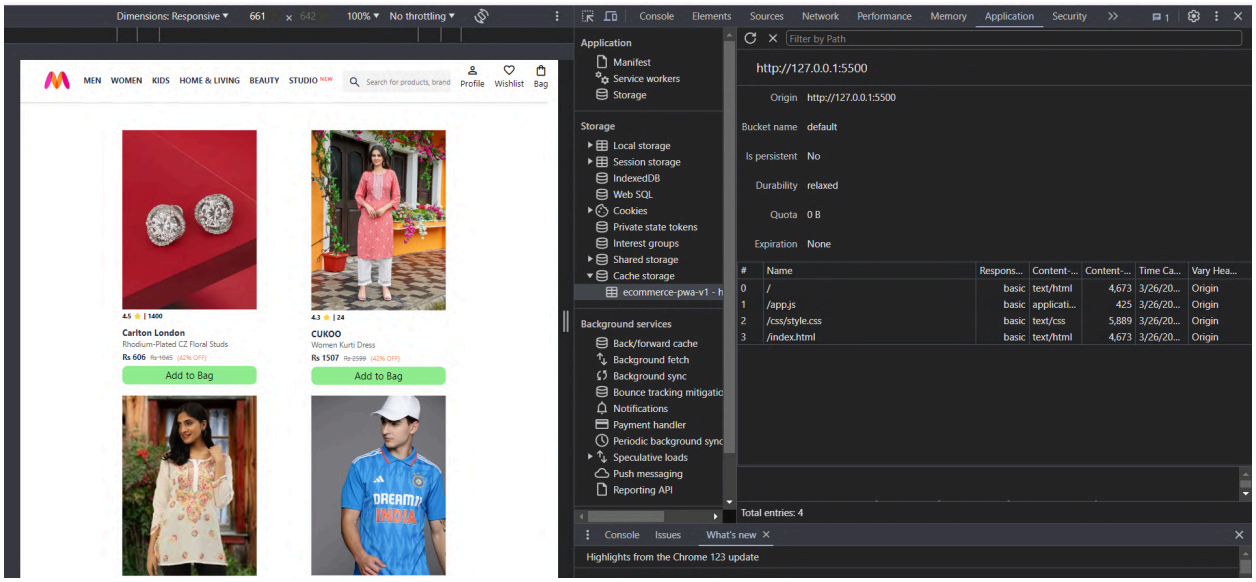
Steps for Execution

- 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 : FreshMenu / PWA : Ecommerce Website **Roll No. 35**



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.	35
Name	SEJAL MAURYA
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15M

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.

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.

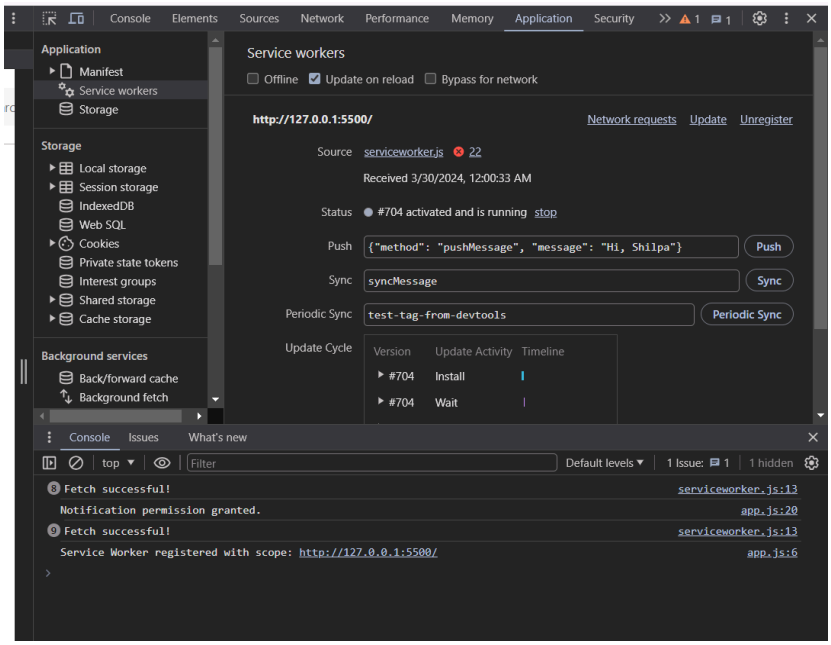
In the following code block is in sw.js file. You can handle push notifications with this event. In this example, I kept it simple. We send an object that has “method” and “message” properties. If the method value is “pushMessage”, we open the information notification with the “message” property.

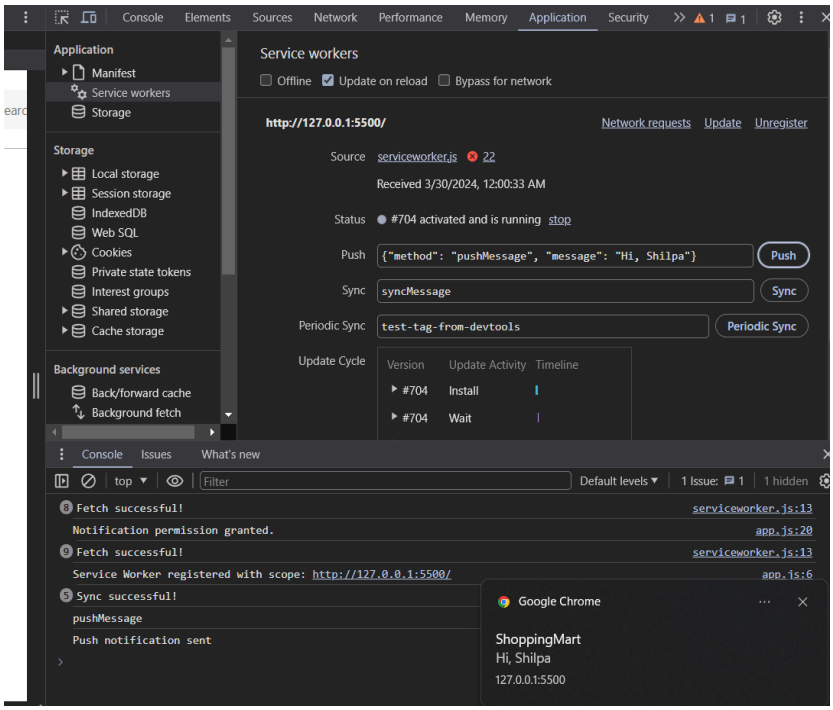
```
//serviceworker
self.addEventListener("install", function (event) {
  event.waitUntil(preLoad());
});

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.request));
});

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

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

```
var preLoad = function () {
  return caches.open("offline").then(function (cache) {
    // caching index and important routes
    return cache.addAll(["/index.html", "/images/10.jpg", "/css/style.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(function (cache) {
    return fetch(request).then(function (response) {
      return cache.put(request, response.clone()).then(function () {
        return response;
      });
    });
  });
};
```

Project Title: Flutter : FreshMenu / PWA : Ecommerce Website **Roll No. 35**

Output





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.	35
Name	SEJAL MAURYA
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15M

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

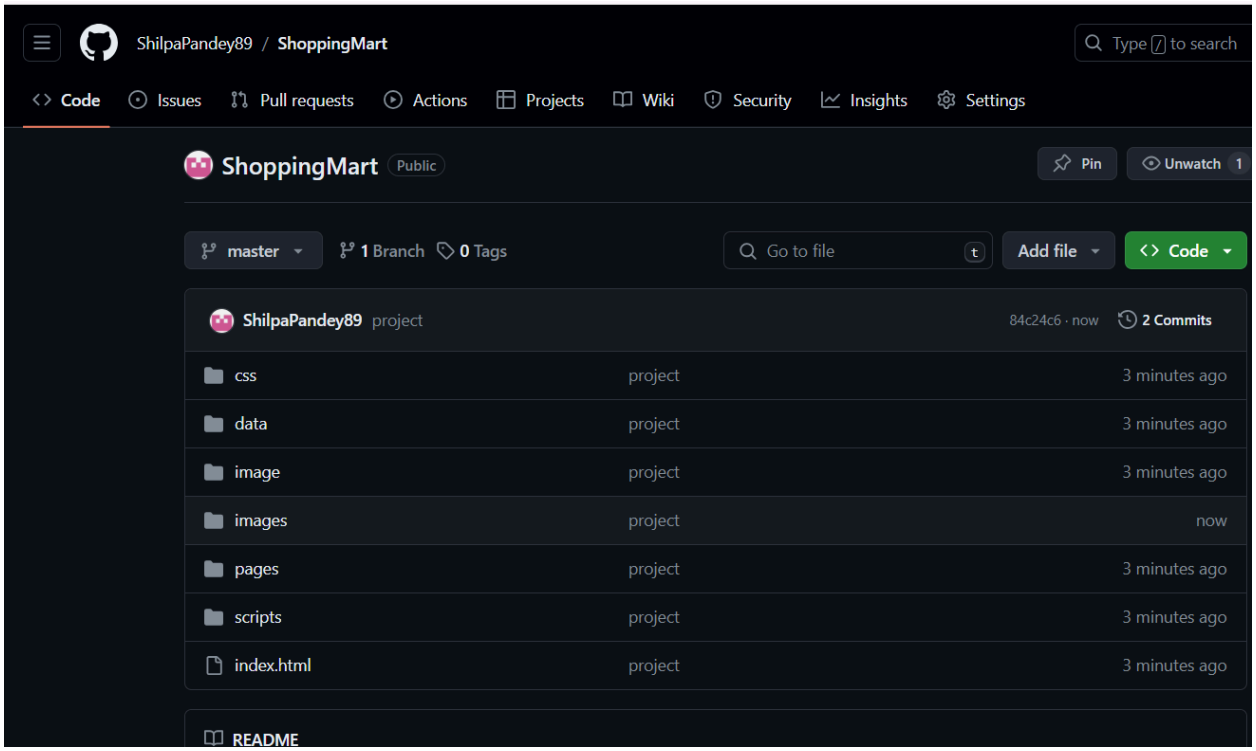
1. Hosted by Google. Enough said.
2. Authentication, Cloud Messaging, and a whole lot of other handy services will be available to you.
3. A real-time database will be available to you, which can store 1 GB of data.
4. You'll also have access to a blob store, which can store another 1 GB of data.
5.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

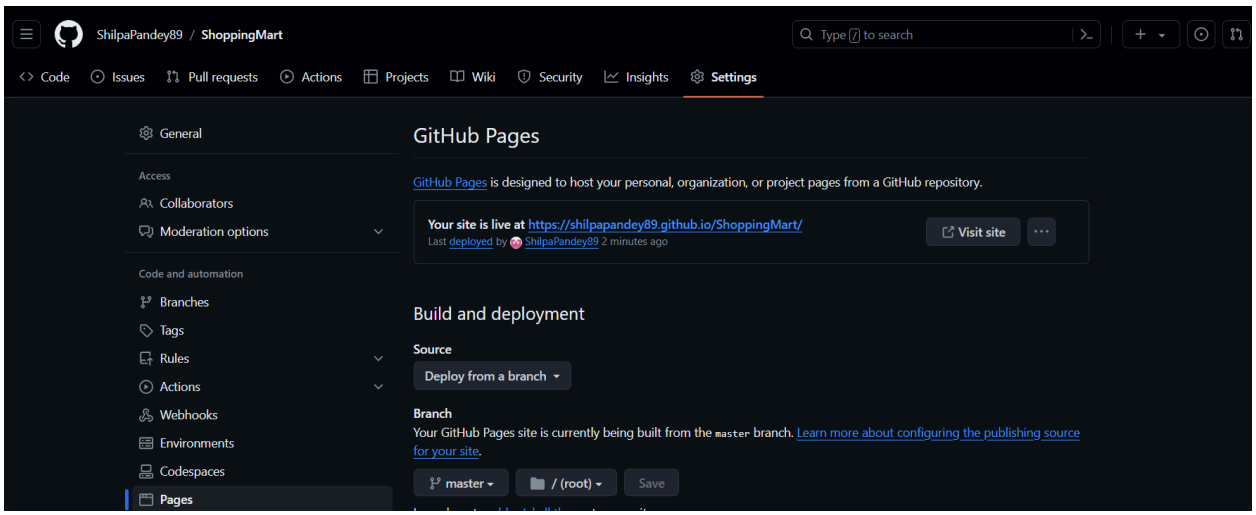
Cons

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.
3. No in-built support for any static site generator.

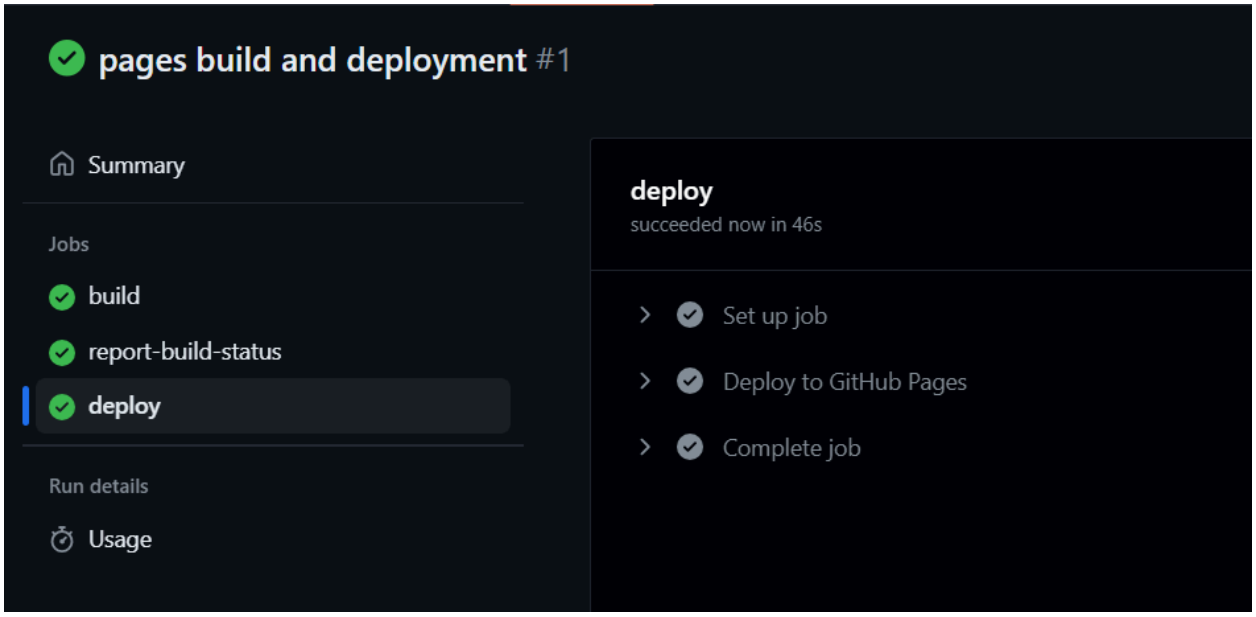
Implementation:

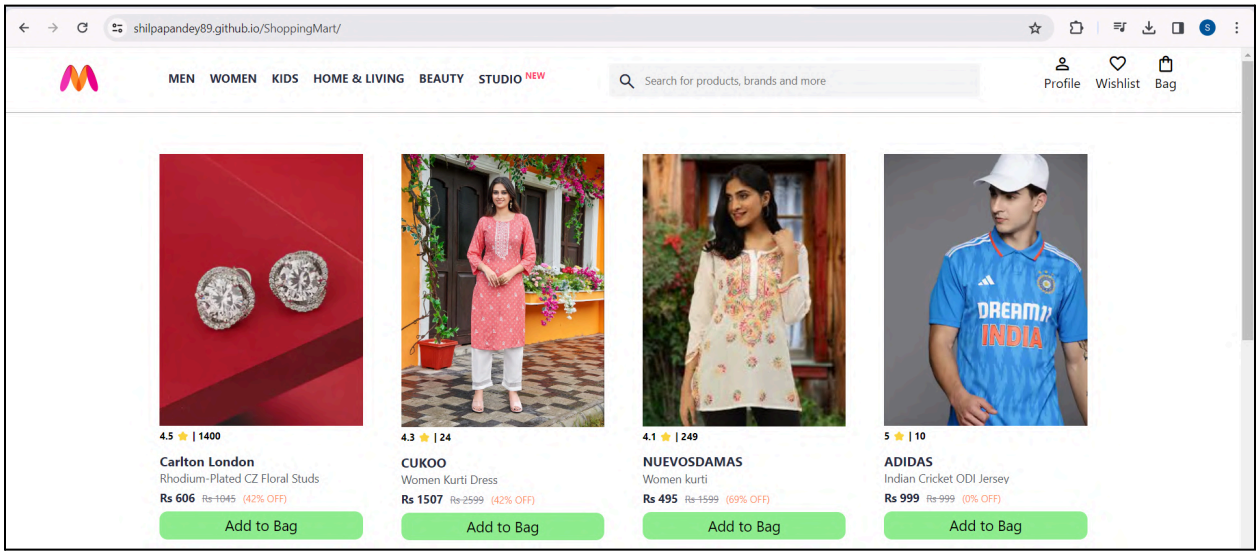
Project Title: Flutter : FreshMenu / PWA : Ecommerce Website **Roll No. 35**





Project Title: Flutter : FreshMenu / PWA : Ecommerce Website **Roll No. 35**





Link to GitHub repository:
https://github.com/ShilpaPandey89/ShoppingMart
Hosted Link
https://shilpapandey89.github.io/ShoppingMart/

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.	35
Name	SEJAL MAURYA
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	15M

‘

Experiment 11

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

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

Google Lighthouse :

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

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

Key Features and Audit Metrics

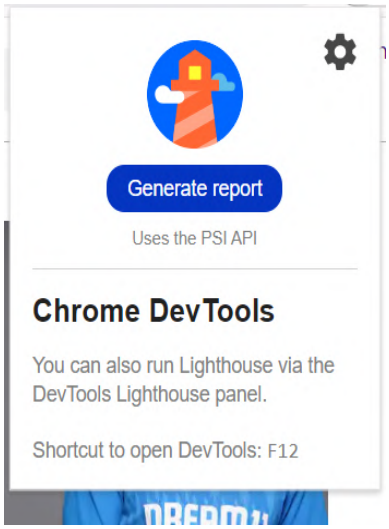
Google Lighthouse has the option of running the Audit for Desktop as well as mobile version of your page(s). The top metrics that will be measured in the Audit are:

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

PWA checklist laid down by Google which includes Service Worker implementation(s), viewport handling, offline functionality, performance in script-disabled environments, etc.

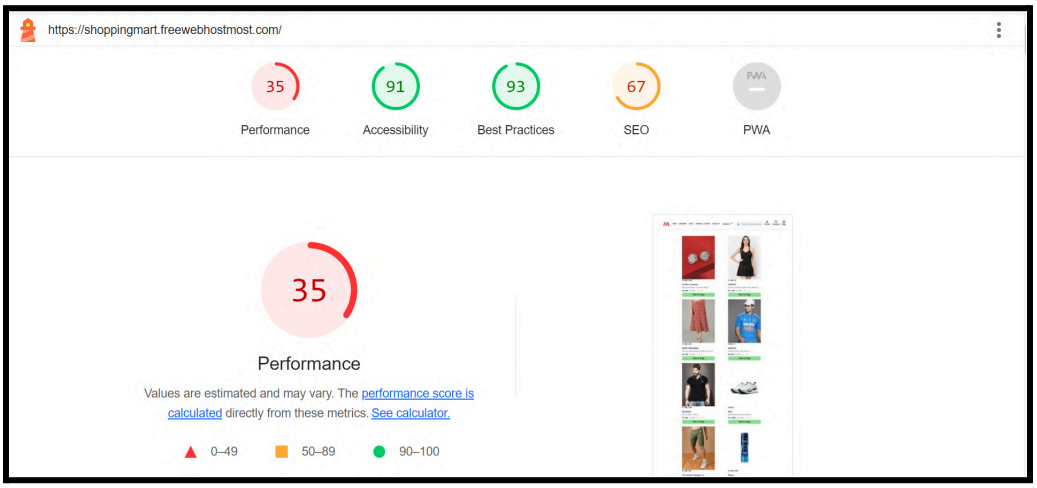
3. **Accessibility:** As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics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

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



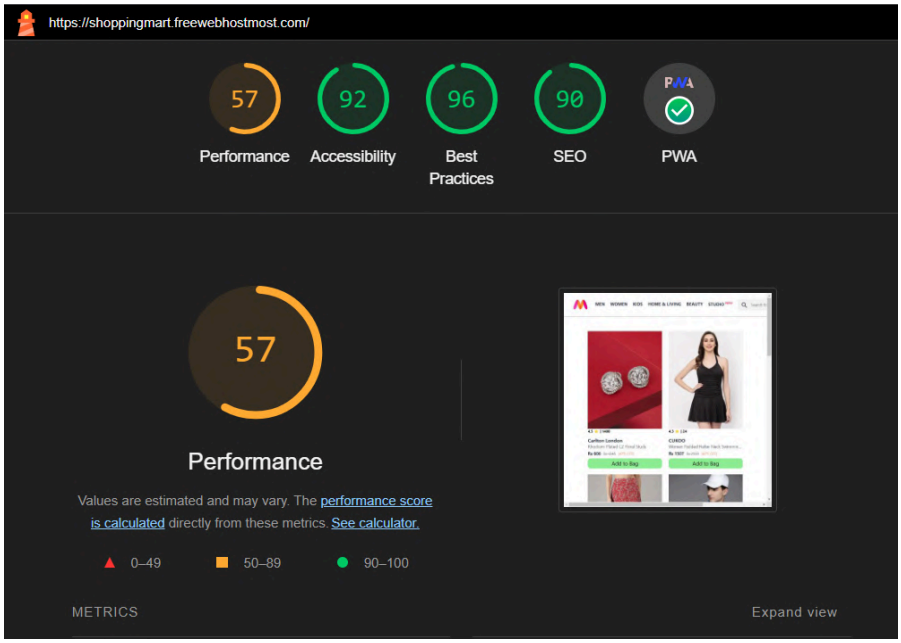
manifest.json

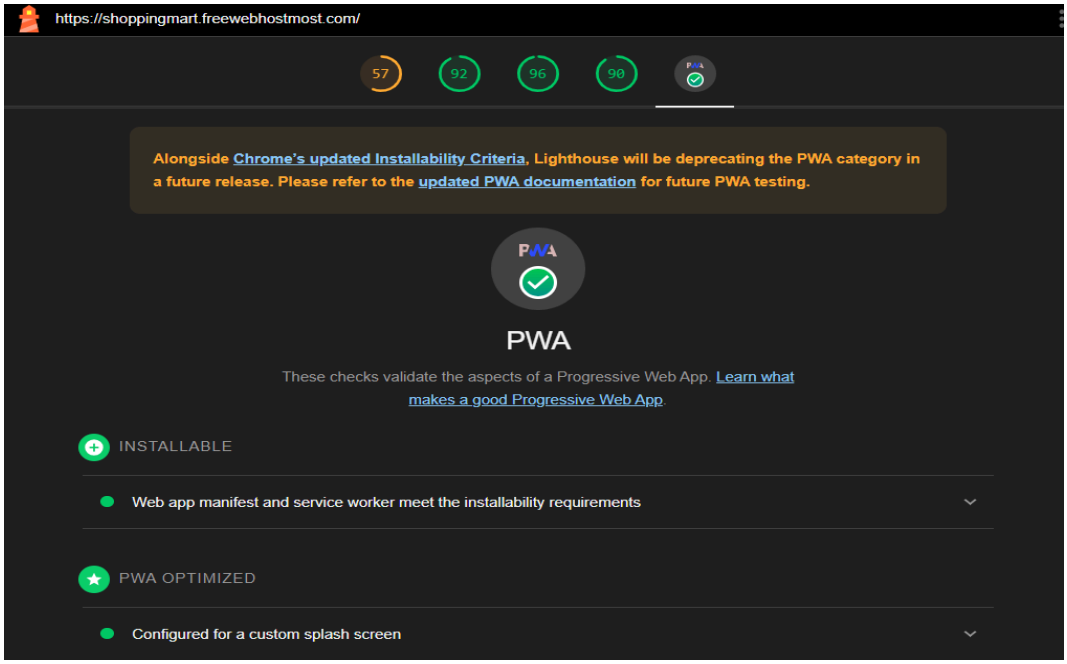
```
{
  "name": "shoppingMart",
  "start_url": "index.html",
  "display": "standalone",
  "background_color": "#5900b3",
  "theme_color": "black",
  "scope": ".",
  "description": "This is an Ecommerce app.",
  "icons": [
    {
      "src": "image/logo.jpg",
      "sizes": "192x192",
      "type": "image/png"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

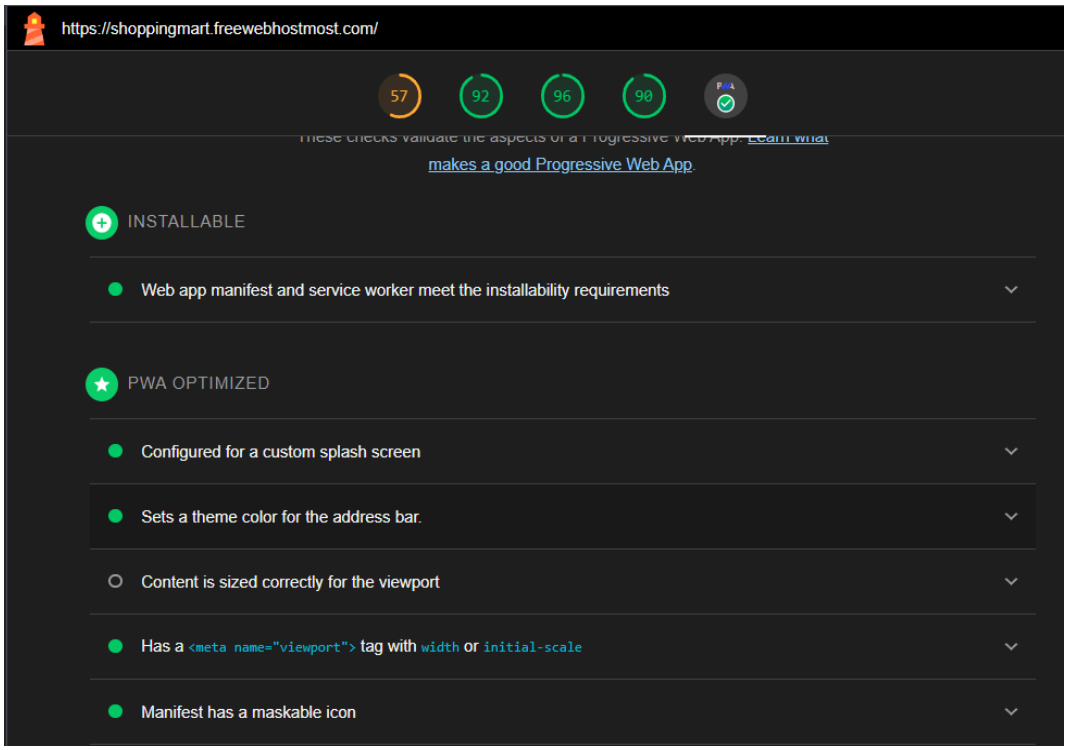


Changes in manifest.json

```
{
  "name": "shoppingMart",
  "start_url": "index.html",
  "display": "standalone",
  "background_color": "#5900b3",
  "theme_color": "black",
  "scope": ".",
  "description": "This is an Ecommerce.",
  "icons": [
    {
      "src": "image/logo.jpg",
      "sizes": "192x192",
      "type": "image/png",
      "purpose": "any maskable"
    },
    {
      "src": "image/CompressJPEG.online_512x512_image.jpg",
      "sizes": "512x512",
      "type": "image/png",
      "purpose": "any maskable"
    }
  ]
}
```







Conclusion: Thus we 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. 2. Widget Tree and Composition: Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree. 3. State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable. 4. Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.
Roll No.	35
Name	SEJAL MAURYA
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	5M

SEJAL MAURYA
D15A
35 (Flutter)

Assignment -01

Q1) flutter Overview :
Explain the key features and advantages of using Flutter for mobile app development. Discuss how the flutter framework differs from traditional and why it gained popularity in the developer community.

Ans flutter is an open-source UI (User Interface) software development toolkit created by Google. flutter uses dart as its programming language.

- Key features of flutter

1) Single Codebase - flutter enables the development of both ios and Android applications using a single codebase. This saves time and effort compared to maintaining separate codebase for each platform.

2) Hot Reload - One of Flutter's standout feature is hot reload, allowing developers to instantly view the changes in the code without restarting the entire programm application.

3) Rich Widget library - flutter provides an extensive set of customizable widgets, which are building blocks of the user interface. This enables developers to create highly expressive UIs.

4) Expressive UI - Flutter allows developers to

describe the UI in a simple and expressive manner.

- Advantages of using flutter

① Faster Development – With a single codebase and hot reload feature flutter significantly reduce the development time.

② Easy to learn → Developers with experience in languages like Java, Kotlin, Swift or Javascript can quickly pick up dart, the programming language used in flutter.

③ Cost-effectiveness → Developing for both ios and Android with one codebase reduces the development and maintenance costs.

④ Community And Support → Flutter growing community and active support from google contribute to a wealth of resources, documentation and third-party package, making problem-solving and development easier.

- Difference between Flutter and Traditional Approaches

	Traditional Approach	Flutter
1) Single Codebase	It requires separate codebase for each platform (Swift for ios and Java or Kotlin for Android)	It provides a single codebase for both ios & Android reducing time and effort.

2)	Language And framework	Uses platform specific languages and frameworks.	It utilizes Dart programming language and the Flutter framework for both platforms
3)	Development Environment.	Requires separate development environment IDEs for both & tools for ios & Android.	It allows developers to use a single development for both platform
4)	Hot Reload	It requires recompilation and relaunching the entire app to see code changes.	It offers hot reload allowing developers to instantly view changes in the app without restarting.

Flutter Has Grained Popularity because of its features shown below -

1) Cross - Platform Development - Flutter's ability to develop for both ios and Android with a single codebase.

2) Productive and faster Development - The hot reload feature allows developers to see instant changes without relaunching the entire app.

3) Performance - The compilation to native ARA code results in high performance.

4) Easy to learn - It has easy learning curve for developer familiar with dart.

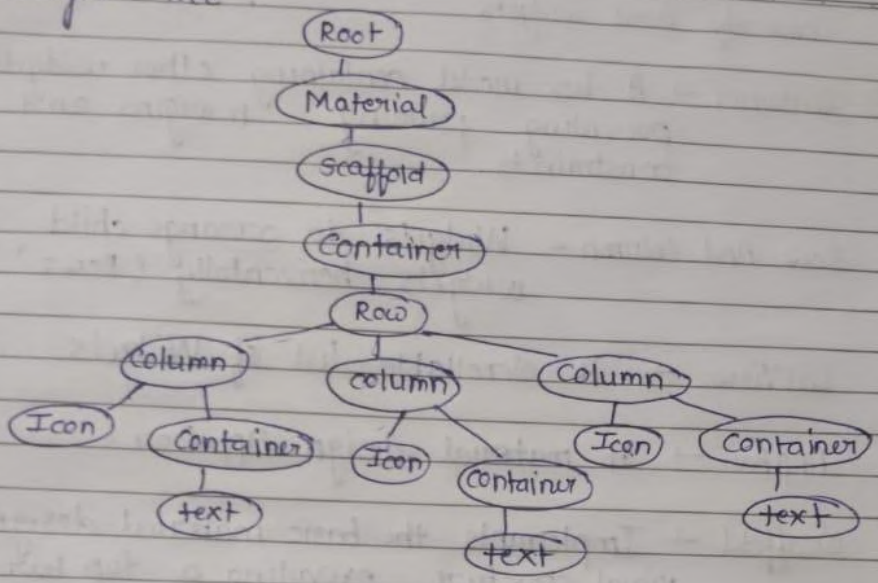
Q2. Widget Tree And Composition: Describe the concept of the widget tree in Flutter Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree.

Ans. • Flutter containes a set of widget which helps to make a expressive User interface.
- A widget describes what the view of app would look like in a given state.
- The user interface contains a several widgets. those several widgets are combined to form a widget tree.
- The widget tree can also divide into 2 small tree
  1. Stateless widget tree
  2. Stateful widget tree.

1. Stateless widget - A widget that is used by user to create a static application.

2. Statefull widget → A stateful widget that is used to create a dynamic application

Widget Tree



- Widget composition to build a complex user interface. Widget composition in flutter provides a powerful technique used to build complex user interface by combining and nesting smaller, reusable widgets. This modular approach makes it easier to manage, maintain, and understand codebase.

① Modularity and Reusability
Widgets in flutter are modular building blocks. Developer creates custom widgets that encapsulate specific UI elements or functionalities.

② Nested Widgets Structures
Widgets can be nested within each other to create hierarchical structures, allowing for composition of complex UIs.

Commonly used widgets

① Container → A box model containing other widgets, providing padding, margins and constraints.

② Row And Column - Widgets to arrange child widgets horizontally (Rows)

③ ListView → A Scrollable list of Widgets

④ AppBar → A material design App Bar.

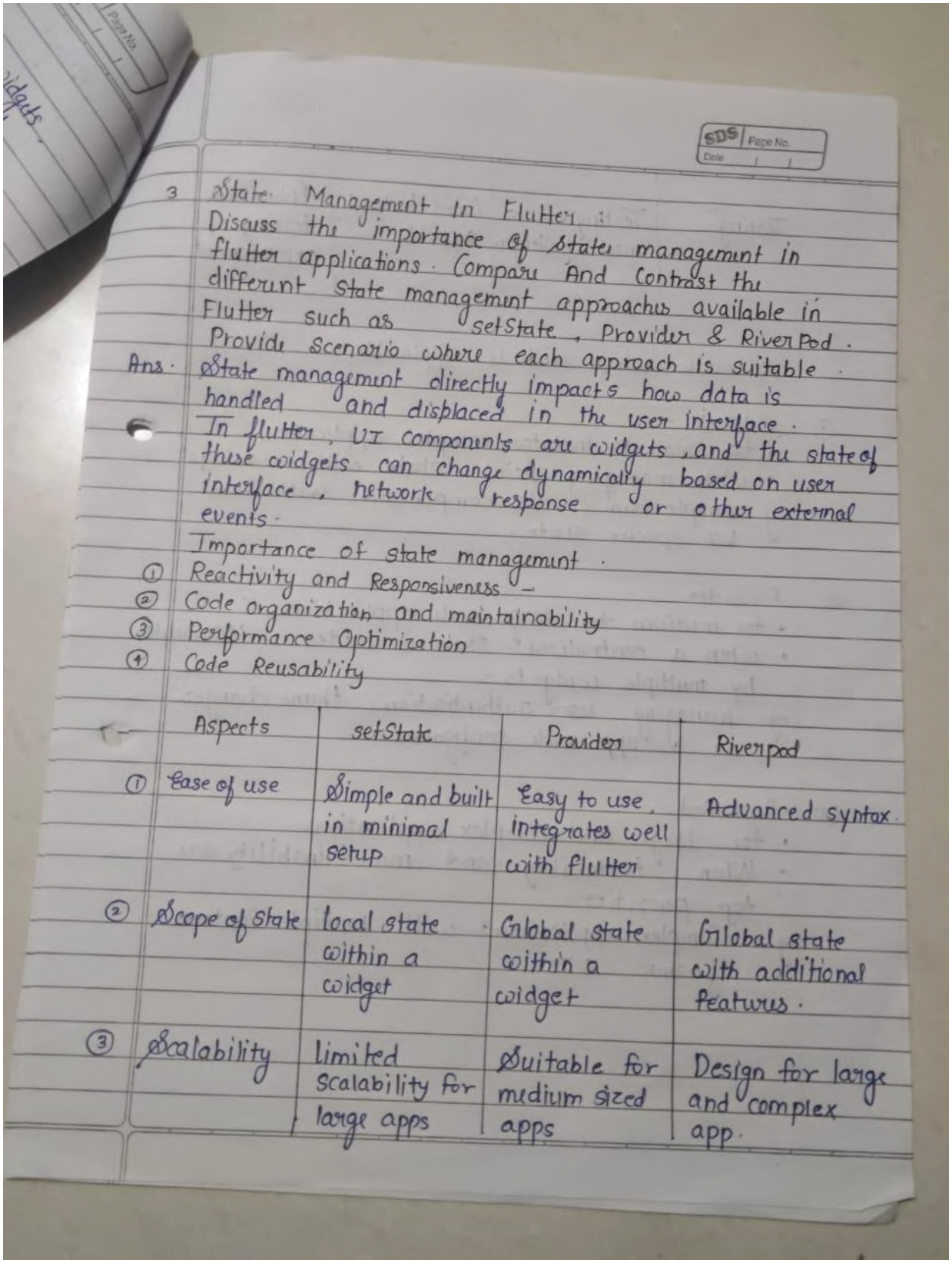
⑤ Scaffold → Implements the basic material design visual structure, providing a top bar, body, bottom navigation etc.

⑥ Text → Display a run of text

⑦ Image → Displays an image from the network and local assets.

⑧ TextField → A widget that allows the user to enter text.

⑨ RaisedButton / ElevatedButton → A button that, when pressed, can trigger an action or event



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 & RiverPod. Provide scenario where each approach is suitable.

Ans. State management directly impacts how data is handled and displaced in the user interface. In flutter, UI components are widgets, and the state of these widgets can change dynamically based on user interface, network response or other external events.

Importance of state management.

① Reactivity and Responsiveness -
② Code organization and maintainability
③ Performance Optimization
④ Code Reusability

	Aspects	setState	Provider	Riverpod
①	Ease of use	Simple and built in minimal setup	Easy to use, integrates well with flutter	Advanced syntax.
②	Scope of state	local state within a widget	Global state within a widget	Global state with additional features.
③	Scalability	limited scalability for large apps	Suitable for medium sized apps	Design for large and complex app.

Testing	Testing can be more challengin	Good testability support	Enhanced testing experienced

Scenario where each is applicable.

① SetState →
- for small to madatory complex application
- when managing local state within a widget.

eg Simple form, UI components with local UI specific state.

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

eg. managing user authentication, thimu changes or app-wide configuration.

③ RiverPod
- for large and complex applications.
- When testability and maintainability are top priorities.

eg. Complex applications with multiple feature dynamic UIs

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

Ans IndexedDB is a client-side storage mechanism in web browsers that allows web applications to store large amounts of structured data. When used in conjunction with a service worker, IndexedDB provides a way for web application to store data offline and perform background task efficiently.

1) Offline Data Storage → Service workers can intercept request and cache responses for offline use. IndexedDB provides a structured storage solution for storing this cached data persistently on the client's device.

2) Background Data Processing → Service worker can run in the background even when the web application is not active. IndexedDB allows service workers to efficiently manage and manipulate data in the background.

5) Data Persistence - IndexedDB data persists across sessions and page reloads, providing a reliable storage solution for web application

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

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

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

- Data Synchronization

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

② Cloud fire-store
- It notifies clients, when data changes, allowing for seamless real-time updates.

③ Authentication
- If user sign in or out an one device, the the authentication state is automically reflected on other devices.

MAD & PWA Lab
Journal

Experiment No.	Assignment-2
Assignment 2 Questions	1. Define Progressive Web App (PWA) and explain its significance in modern web development.Discuss the key characteristics that differentiate PWAs from traditional mobile apps 2. Define responsive web design and explain its importance in the context of Progressive Web Apps.Compare and contrast responsive, fluid, and adaptive web design approaches. 3. Describe the lifecycle of Service Workers, including registration, installation, and activation phases. 4. Explain the use of IndexedDB in the Service Worker for data storage.
Roll No.	35
Name	SEJAL MAURYA
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4:Understand various PWA frameworks and their requirements LO5: Design and Develop a responsive User Interface by applying PWA Design techniques LO6:Develop and Analyze PWA Features and deploy it over app hosting solutions
Grade:	4M

SEJAL MAURYA
DISN
35

Batch - B

SDS

ASSIGNMENT - 02
PWA

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

Ans PWA

A progressive Web app (PWA) is a type of web application that utilizes modern web technologies to provide user with an app-like experience directly through their web browsers

Significance of PWA in modern Web development

1) Improved User experience - PWAs offers a fast reliable and engaging user experience, similar to native mobile apps. This includes feature like smooth navigation, quick loading times etc.

2) Cross-Platform Compatibility - PWAs work seamlessly accross different devices and platform including desktops, laptops & tablets.

3) Cost Effectiveness - Developing and maintaing a single PWA codebase is often more cost-effective than creating separate version for different platforms.

key characteristics differentiating PWA from traditional mobile Apps :

① Development Technology - PWAs built using web technologies (HTML, CSS, javascript).
Traditional mobile apps developed using platform specific languages (Swift, kotlin).

② Platform → PWAs coorks on any device with a standards compliant bracoser. While traditional mobile Apps platform specific (iOS, Android, etc.

③ Performance - PWA optimized for fast loading times and smooth navigation.
Traditional mobiles performance may vary depending on platform & app design.

④ Security - PWAs are served over HTTPS to ensure data security & integrity. protecting user's sensitive information from unauthorized access.

3) Moderate initial efforts for development	It Requires multiple layouts potentially more maintenance for developmnt
4) Requires less maintenance as changes to the website can be applied universally across all devices	4) Require more maintence as change to the website.

Fluid

① Fluid web design involves creating web sites that use proportional units like percentage instead of fixed unit like pixels for layout

2) It focus on creating design that scale smoothy across differunt screen sizes

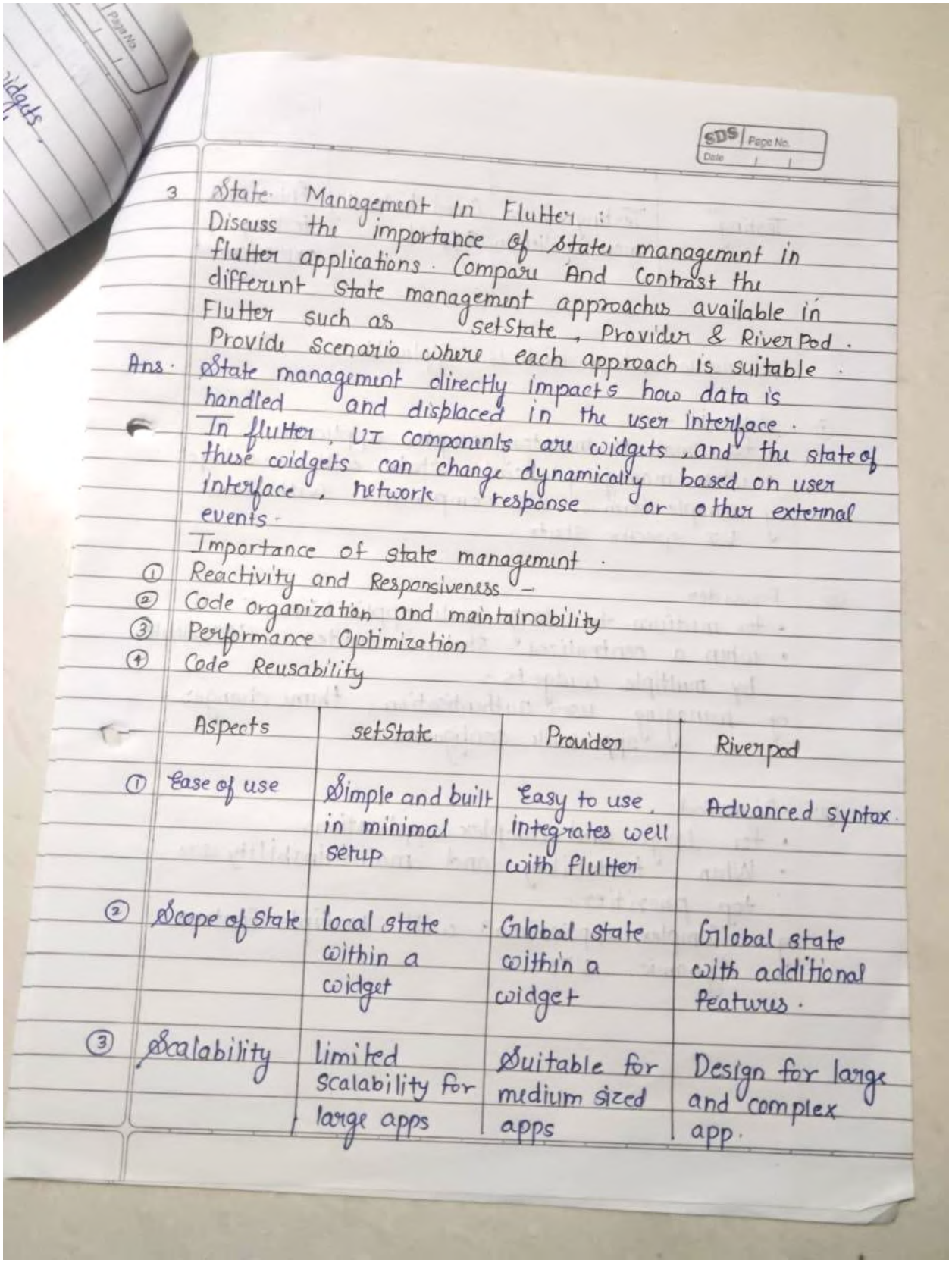
3) In a fluid design, elemunts like text blocks and images expand or contrast smoothly as the browser window is resized.

Q2) 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

Ans 1) Responsive Web design is an approach to Web design that ensures web pages render well on a variety of devices and window or screen sizes. It involves using flexible layout, images and media queries to adapt the content to different screen sizes.

2) In the context of progressive web apps, responsive web design is crucial because PWAs aims to provide a seamless user experience across various devices and scree sizes. Since PWAs are designed to be responsive by nature. Whether it's a desktop, tablet, or smartphone, without sacrificing usability or functionality

	Responsive Web Design	Adaptive Web Design
①	It offers flexible layout adjustments across devices	① It provides predefined layouts tailored for specific screen sizes
②	User experience is consistent across a wide range of devices	② User experience is tailored for different device categories



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 & RiverPod. Provide scenario where each approach is suitable.

Ans. State management directly impacts how data is handled and displaced in the user interface. In flutter, UI components are widgets, and the state of these widgets can change dynamically based on user interface, network response or other external events.

Importance of state management.

① Reactivity and Responsiveness -
② Code organization and maintainability
③ Performance Optimization
④ Code Reusability

	Aspects	setState	Provider	Riverpod
①	Ease of use	Simple and built in minimal setup	Easy to use, integrates well with flutter	Advanced syntax.
②	Scope of state	local state within a widget	Global state within a widget	Global state with additional features.
③	Scalability	limited scalability for large apps	Suitable for medium sized apps	Design for large and complex app.

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

Ans IndexedDB is a client-side storage mechanism in web browsers that allows web applications to store large amounts of structured data. When used in conjunction with a service worker, IndexedDB provides a way for web application to store data offline and perform background ~~task~~ efficiently.

1) Offline Data Storage → Service workers can intercept request and cache responses for offline use. IndexedDB provides a structured storage solution for storing this cached data persistently on the client's device.

2) Background Data Processing → Service worker can run in the background even when the web application is not active. IndexedDB allows service workers to efficiently manage and manipulate data in the background.

5) Data Persistence - IndexedDB data persists across sessions and page reloads, providing a reliable storage solution for web application