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.

Since 1962

Department of Information Technology

CERTIFICATE

This is to certify that **_Tejas Shailendra Rokade** 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

Project Title:Lunching and Glowing MAD /PWA Lab **Roll no.- 50**

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

Faculty Incharge : Mrs. Kajal Joseph.

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

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

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title:Lunching and Glowing MAD /PWA Lab **Roll no.- 50**

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	19/01	26/01	15
2.	To design Flutter UI by including common widgets.	LO2	26/01	02/02	15
3.	To include icons, images, fonts in Flutter app	LO2	02/02	09/02	15
4.	To create an interactive Form using form widget	LO2	09/02	16/02	15
5.	To apply navigation, routing and gestures in Flutter App	LO2	16/02	23/02	15
6.	To Connect Flutter UI with fireBase database	LO3	23/02	08/03	15
7.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4	08/03	15/03	15
8.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA	LO5	15/03	22/03	15
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5	22/03	31/03	15
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5	29/03	31/03	15
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6	29/03	31/03	15
12.	Assignment-1	LO1,LO2 ,LO3	28/01	05/02	5
13.	Assignment-2	LO4,LO5 ,LO6	14/03	21/03	4

MAD & PWA Lab
Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll no.	50
Name	Tejas Shailendra Rokade
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15

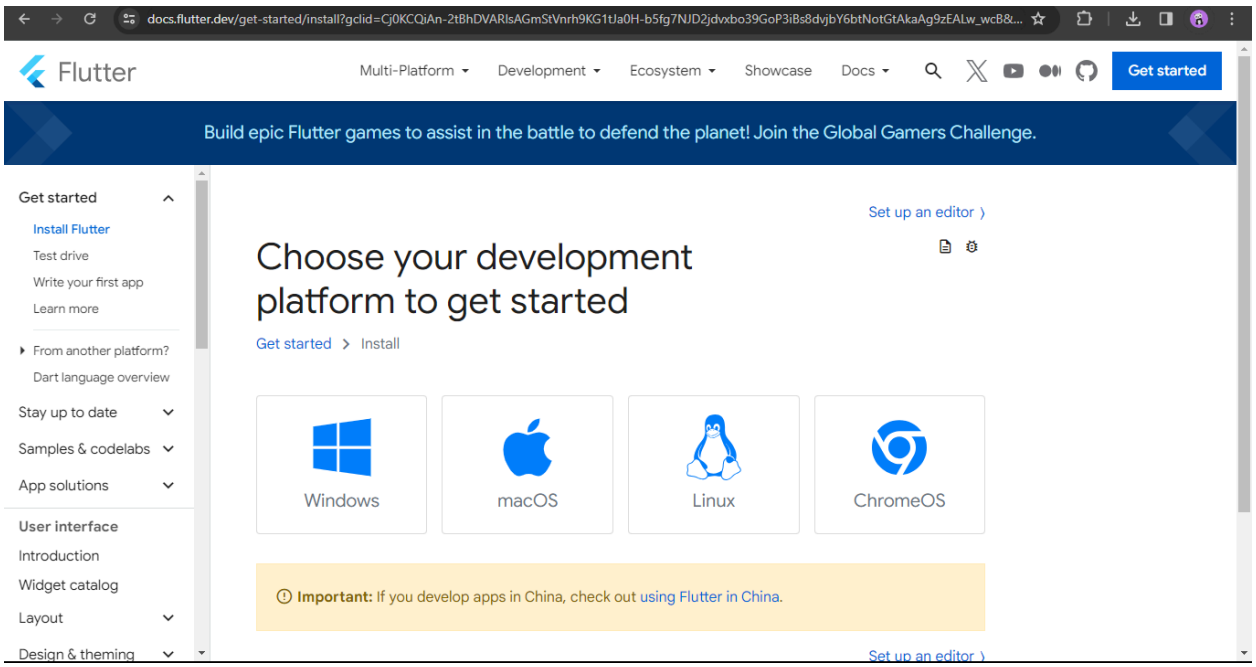
Name - Tejas Shailendra Rokade
Div - D15A Roll no - 50
Batch - C

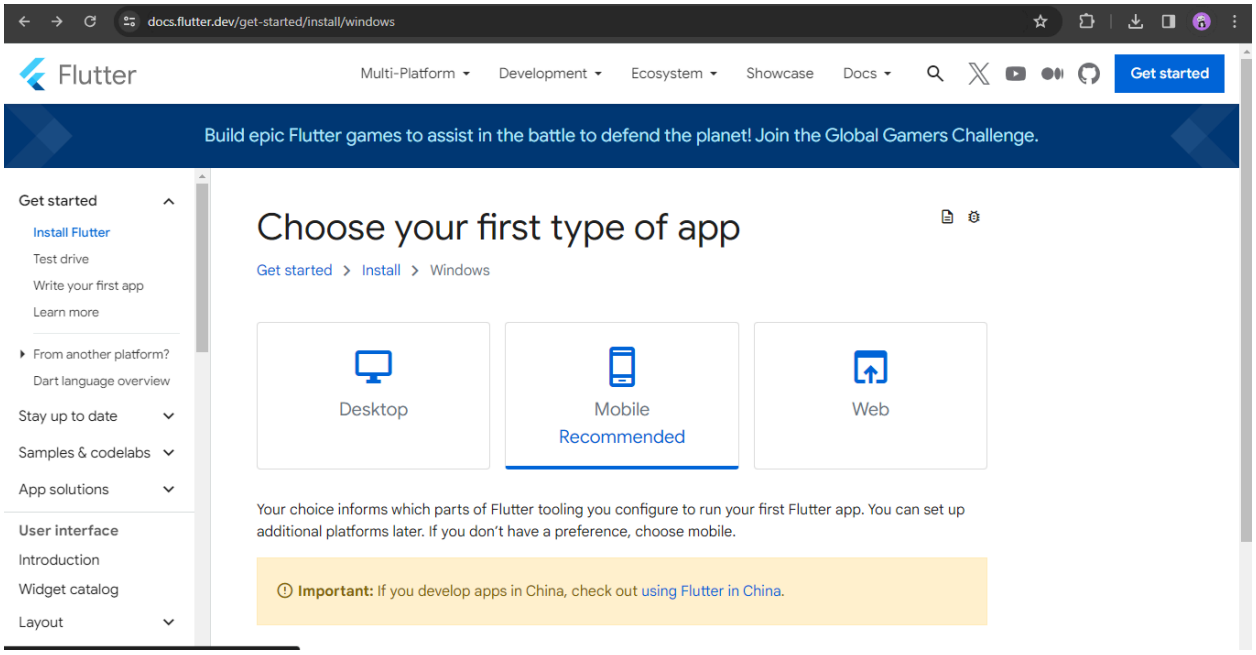
Experiment - 1

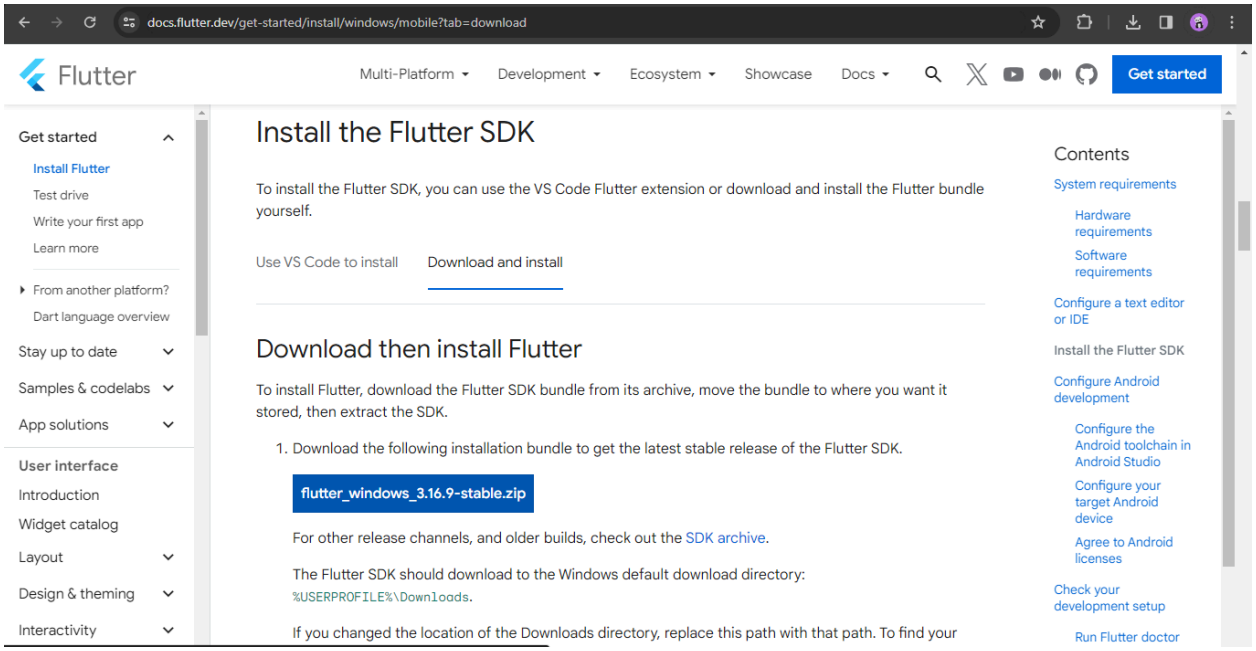
Aim - Installation and Configuration of Flutter Environment.

Pre Requisites: Android Studio Hedgehog, Visual Studio Code and Flutter package.I highly recommend watching this video, for Setting up your Flutter Environment and your Virtual Device: https://youtu.be/ZSWfgxrxN0M?feature=shared

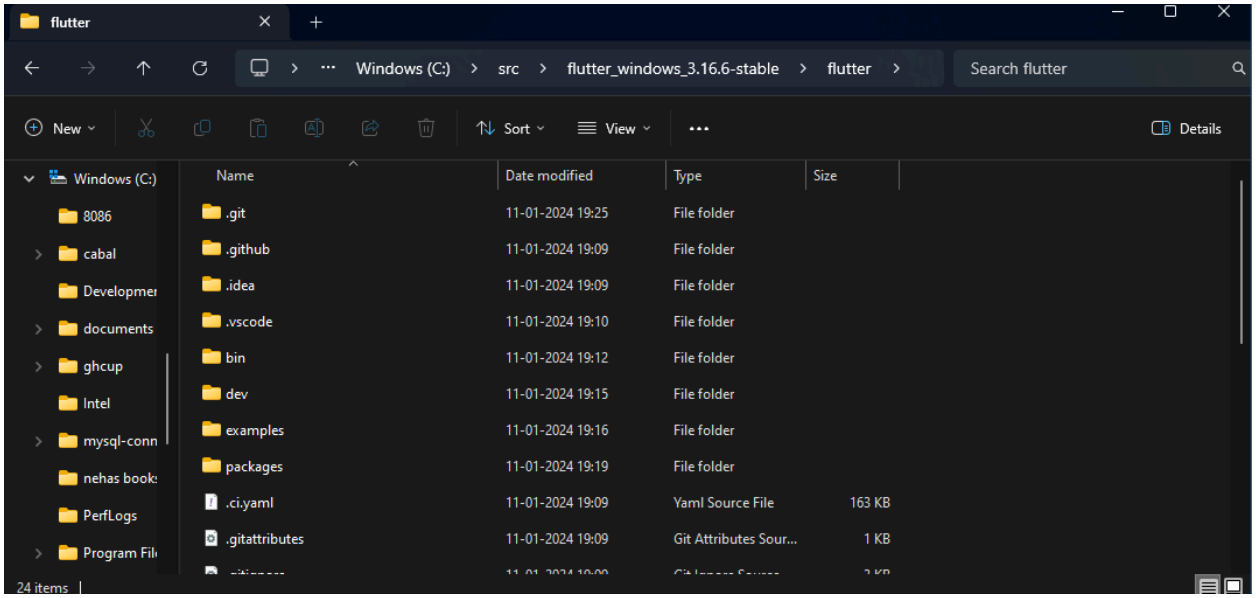
Installation -



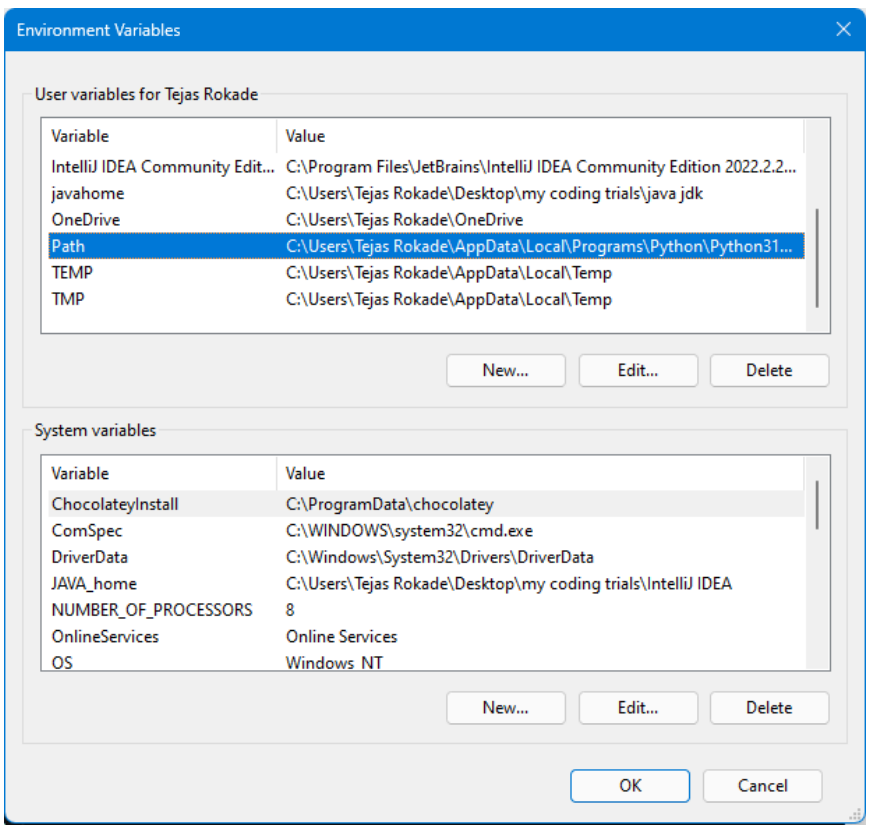




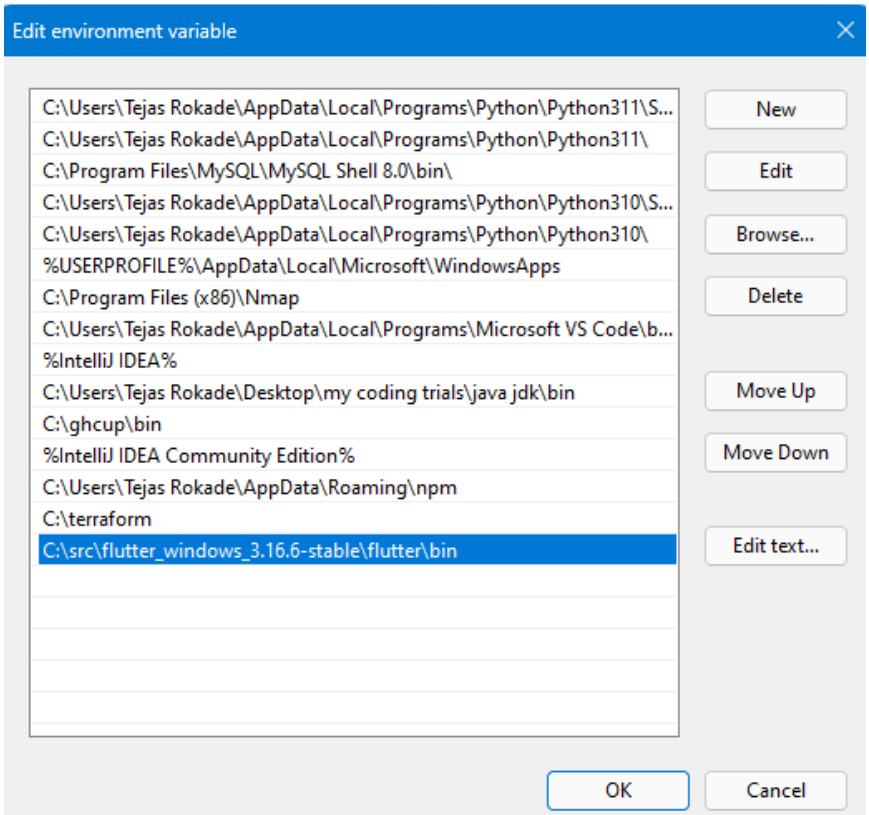
Extract the files in the directory -



Setup path in Environment variables -



Create new variable and insert the path to bin folder and apply -



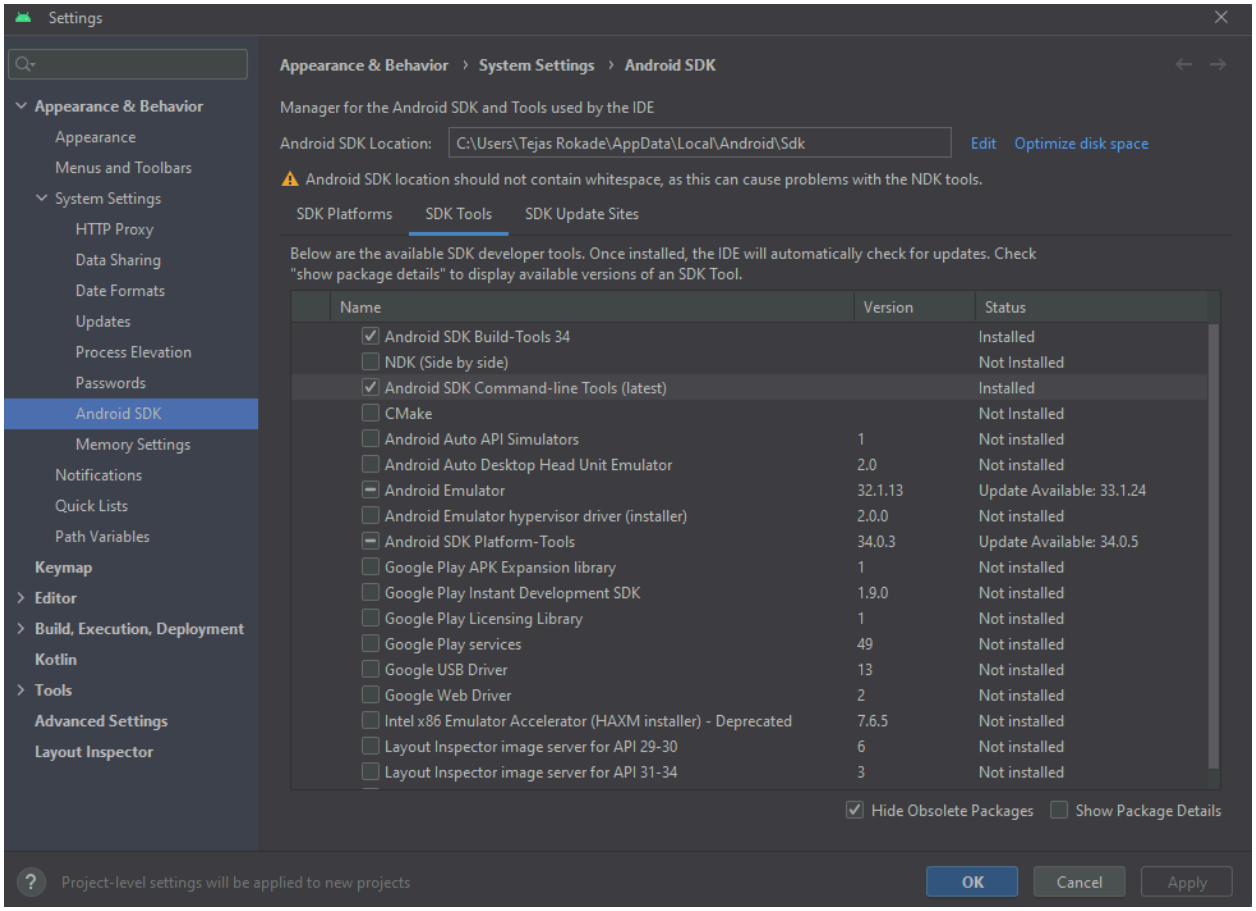
Then on CMD -

```
C:\Users\Tejas Rokade>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 d
                            diagnostic information. (Use "-vv" to force verbose logging in those
-d, --device-id             Target device id or name (prefixes allowed).
    --version               Reports the version of this tool.
```

```
C:\Users\Tejas Rokade>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.16.6, on Microsoft Windows [Version 10.0.22621.3007], locale en-IN)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[✓] Chrome - develop for the web
[✓] Visual Studio - develop Windows apps (Visual Studio Build Tools 2019 16.11.24)
[✓] Android Studio (version 2022.2)
[✓] VS Code (version 1.85.2)
[✓] Connected device (3 available)
[✓] Network resources

• No issues found!

C:\Users\Tejas Rokade>
```

The error for Android toolchain will occur if havent installed android SDK Command Line Tools in Android Studios SDK Tools.

And IF issue of android licenses occurs run Flutter –android-licenses command in the prompt and type y+enter till process is complete

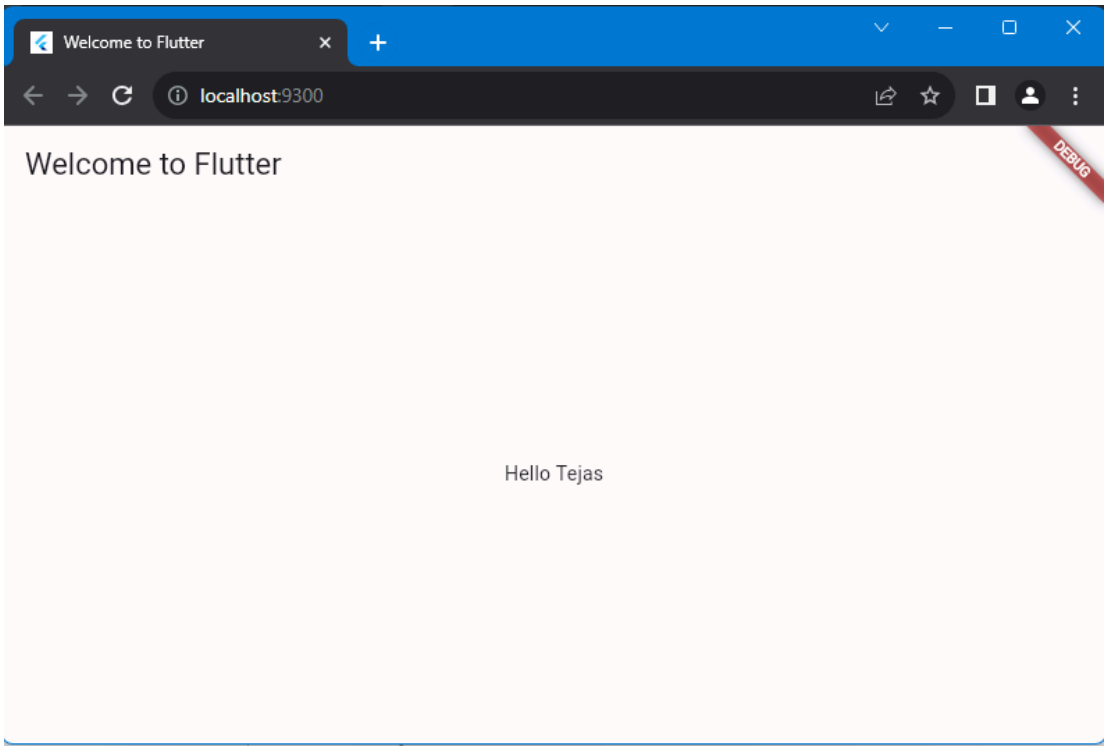


Code -

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

```
void main() {
runApp(const MyApp());
}
class MyApp extends StatelessWidget {
const MyApp({Key? key}) : super(key: key);
@override
Widget build(BuildContext context) {
return MaterialApp(
title: 'Welcome to Flutter',
home: Scaffold(
appBar: AppBar(
title: const Text('Welcome to Flutter'),
),
body: const Center(
child: Text('Hello Tejas''),
),
),
);
}
}
```

Output -



MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll no.	50
Name	Tejas Shailendra Rokade
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Name - Tejas Shailendra Rokade
Div - D15A Roll no - 50
Batch - C

Experiment - 2

Aim - To design Flutter UI by including common widgets.

Theory -

In Flutter, widgets are the building blocks of the user interface (UI). Everything in Flutter is a widget, from the simplest elements like text and buttons to more complex structures like entire screens or even the entire application itself. Widgets are used to create the visual representation of the app and to define its behavior.

There are two main types of widgets in Flutter: StatelessWidget and StatefulWidget.

StatelessWidget:

- A StatelessWidget is immutable, meaning its properties cannot change once it's been instantiated.
- It does not store or manage any mutable state.
- The UI of a StatelessWidget is purely based on its configuration

StatefulWidget:

- A StatefulWidget is mutable and can change its internal state during its lifetime.
- It consists of two separate classes: one for the immutable widget and one for the mutable state.
- When the state changes, the widget rebuilds its UI.

Category of widgets -

Basic UI Elements:

- Text: Displays a paragraph of text.
- Image: Displays an image.
- Icon: Displays a material design icon.
- Container: A box model to contain other widgets.

- Layout Widgets:
  - Row and Column: Arrange children in a horizontal (Row) or vertical (Column) array.
  - Stack: Overlays widgets on top of each other.
  - Expanded: Takes up available space along a parent widget.
  - ListView and GridView: Scrollable lists and grids.
- Material Design Widgets:
  - AppBar: Represents the top app bar.
  - Drawer: A slide-in menu from the edge of the screen.
  - BottomNavigationBar: A material design bottom navigation bar.

- Input Widgets:
  - TextField: Accepts user input.
  - Checkbox and Radio: For selecting options.
  - Slider and Switch: Allows users to make selections on a continuous range or toggle a setting.
- Scrolling Widgets:
  - SingleChildScrollView: Allows a single child to be scrolled.
  - ListView, GridView, and PageView: Support scrolling through a list, grid, or pages.
- Navigation Widgets:
  - Navigator: Manages a stack of route objects and transitions between them.
  - MaterialPageRoute: Represents a full page that slides in from the right.
- Animation Widgets:
  - AnimatedContainer, AnimatedOpacity: Animate the properties of a widget.
  - Hero: Animates a widget transition between two screens.
- State Management Widgets:
  - StatefulWidget and StatelessWidget: Fundamental building blocks for managing state.
  - Provider, Bloc, Riverpod: External packages for more advanced state management.
- Form Widgets:
  - Form: A container for form fields, validation, and submission.
  - TextFormField: An input field within a form.
- Async Widgets:
  - FutureBuilder and StreamBuilder: Build widgets based on asynchronous data.
- Custom Widgets:
  - Developers can create their own custom widgets by extending existing ones or combining basic widgets to encapsulate specific functionality

Code -

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

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

class FoodDeliveryApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Food Delivery App',
     theme: ThemeData(
       primarySwatch: Colors.blue,
       visualDensity: VisualDensity.adaptivePlatformDensity,
```

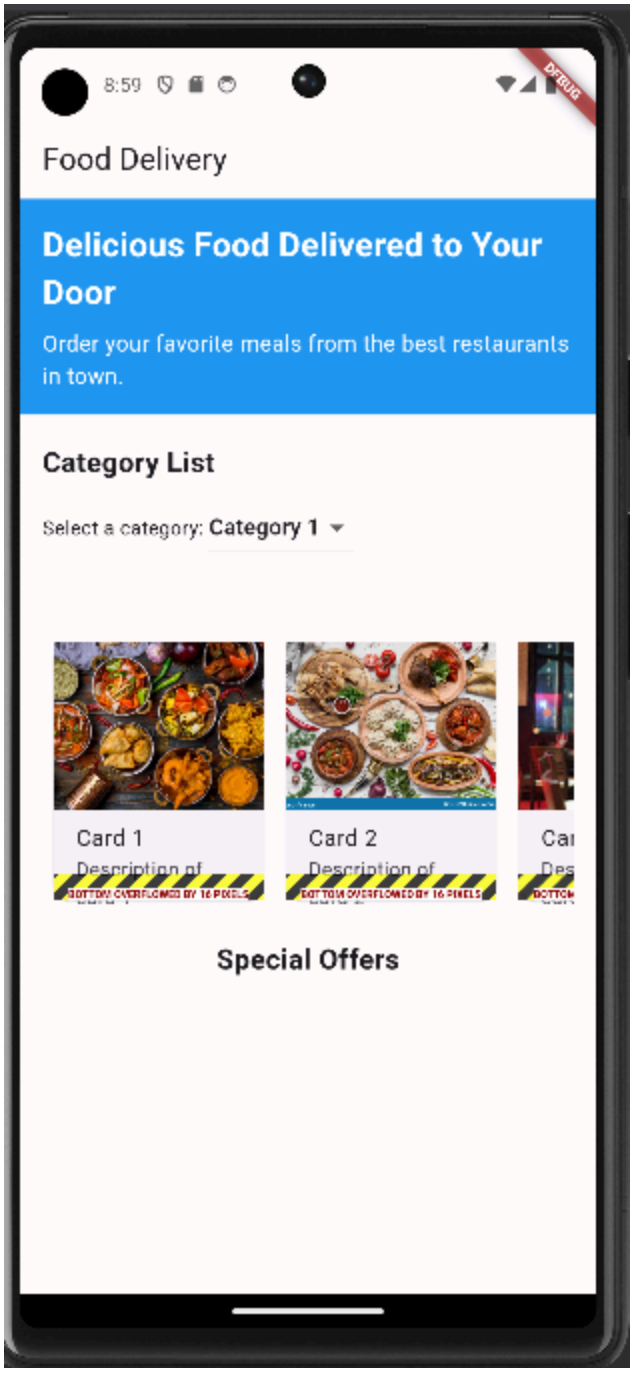
```
    ),
    home: HomePage(),
  );
 }
}

class HomePage extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Food Delivery'),
    ),
    body: Column(
      children: [
        Header(),
        SizedBox(height: 20.0),
        CategoryList(),
        SizedBox(height: 20.0),
        SpecialOffer(),
      ],
    ),
  );
 }
}

class Header extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Container(
    padding: EdgeInsets.all(16.0),
    decoration: BoxDecoration(
      color: Colors.blue,
    ),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(
          'Delicious Food Delivered to Your Door',
          style: TextStyle(
            color: Colors.white,
            fontSize: 24.0,
            fontWeight: FontWeight.bold,
          ),
```

```
        ),
        SizedBox(height: 8.0),
        Text(
          'Order your favorite meals from the best restaurants in town.',
          style: TextStyle(
            color: Colors.white,
            fontSize: 16.0,
          ),
        ),
      ],
    ),
  );
}
}

class CategoryList extends StatefulWidget {
 @override
 _CategoryListState createState() => _CategoryListState();
}

class _CategoryListState extends State<CategoryList> {
 final List<String> categories = ['Category 1', 'Category 2', 'Category 3'];
 String selectedCategory = 'Category 1'; // Default category

 @override
 Widget build(BuildContext context) {
  return Container(
    padding: EdgeInsets.symmetric(horizontal: 16.0),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(
          'Category List',
          style: TextStyle(
            fontSize: 20.0,
            fontWeight: FontWeight.bold,
          ),
        ),
        SizedBox(height: 8.0),
        Row(
          children: [
            Text('Select a category: '),
            DropdownButton<String>(
              value: selectedCategory,
```

```
        onChanged: (String? newValue) {
          // Update the selected category when the dropdown changes
          if (newValue != null) {
            setState(() {
              selectedCategory = newValue;
            });
          }
        },
        items: categories.map((String category) {
          return DropdownMenuItem<String>(
            value: category,
            child: Text(category),
          );
        }).toList(),
      ),
    ],
  ),
  SizedBox(height: 50.0),
  // Updated section with 4 cards and images
  Container(
    height: 200.0,
    child: ListView.builder(
      scrollDirection: Axis.horizontal,
      itemCount: 4, // Number of cards
      itemBuilder: (context, index) {
        return Card(
          margin: EdgeInsets.all(8.0),
          child: Container(
            width: 150.0, // Customize the card width as needed
            child: Column(
              children: [
                // Add an Image widget with the image path
                Image.asset(
                  'assets/card_image_${index + 1}.jpeg',
                  height: 120.0,
                  width: 150.0,
                  fit: BoxFit.cover,
                ),
                ListTile(
                  title: Text('Card ${index + 1}'),
                  subtitle: Text('Description of card ${index + 1}'),
                  // Add onTap callback if needed
                ),
              ],
```

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

class SpecialOffer extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   // Implement your special offer section here
   // For example, you can use a GridView with special offer cards
   return Container(
     padding: EdgeInsets.symmetric(horizontal: 16.0),
     child: Text(
       'Special Offers',
       style: TextStyle(
         fontSize: 20.0,
         fontWeight: FontWeight.bold,
       ),
     ),
   );
 }
}
```

Output -


8:59
DEBUG
Food Delivery
Delicious Food Delivered to Your Door
Order your favorite meals from the best restaurants in town.
Category List
Select a category: Category 1
Card 1
Card 2
Special Offers

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll no.	50
Name	Tejas Shailendra Rokade
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Name - Tejas Shailendra Rokade
Div - D15A Roll no - 50
Batch - C

Experiment - 3

Aim - To include fonts and image in the flutter app.

Theory -

In essence, Flutter widgets serve as fundamental building blocks for crafting the user interface of a Flutter application. They can be broadly classified into two categories: `StatelessWidget`, representing immutable aspects of the UI, and `StatefulWidget`, representing mutable components that can undergo changes over time.

Some noteworthy Flutter widgets include:
1. **Scaffold**: Provides the basic structure for a Flutter app, incorporating layout elements like AppBar, BottomNavigationBar, and a body for primary content.
2. **Container**: A versatile box model utilized for layout, padding, margin, decoration, and constraints, capable of containing other widgets.
3. **Row & Column**: Widgets designed for arranging child widgets horizontally (Row) or vertically (Column), pivotal for creating flexible and responsive layouts.
4. **Text**: Used for displaying text on the screen, with support for various styling options such as font size, color, and alignment.
5. **TextField**: Captures user input, such as text, numbers, or passwords, with the `onChanged` property enabling dynamic updates based on user input.
6. **Buttons**: Various button widgets like `ElevatedButton` or `TextButton` trigger actions when pressed, providing a means for user interaction.
7. **Forms**: The `Form` widget manages a group of `TextFormField` widgets, facilitating input validation and submission.
8. **Icons**: The `Icon` widget displays icons from libraries, enhancing visual elements and conveying meaning through symbols.

Key Design Principles emphasized include:
- **Consistency**: Utilizing common widgets fosters a consistent design language throughout the app.
- **Responsive Layouts**: Widgets like `Row` and `Column` assist in creating responsive and flexible layouts, adapting to different screen sizes.
- **User Input Handling**: `TextField` and `Form` widgets facilitate proper handling, ensuring data integrity and validation.
- **Interactive Elements**: Buttons and icons contribute to interactivity and user engagement within the app.
- **Visual Styling**: The `Container` widget and styling properties of other widgets allow for visual customization and theming.

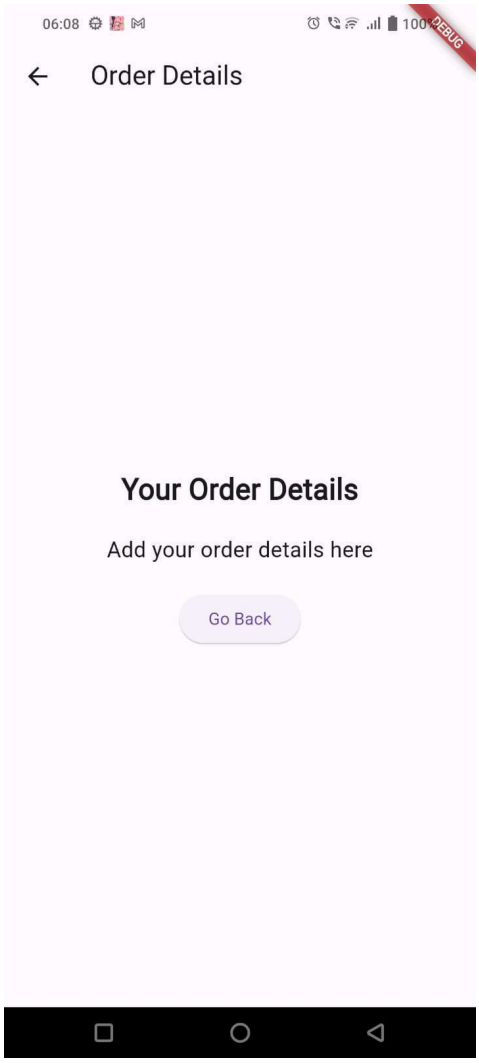
Code -

For Image and card creation -

```
class _CategoryListState extends State<CategoryList> {
 final List<String> categories = ['Category 1', 'Category 2', 'Category 3'];
 String selectedCategory = 'Category 1'; // Default category

 @override
 Widget build(BuildContext context) {
  return Container(
    padding: EdgeInsets.symmetric(horizontal: 16.0),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(
          'Category List',
          style: TextStyle(
            fontSize: 20.0,
            fontWeight: FontWeight.bold,
          ),
        ),
        SizedBox(height: 8.0),
        Row(
          children: [
            Text('Select a category: '),
            DropdownButton<String>(
              value: selectedCategory,
              onChanged: (String? newValue) {
                // Update the selected category when the dropdown changes
                if (newValue != null) {
                  setState(() {
                    selectedCategory = newValue;
                  });
                }
              },
              items: categories.map((String category) {
                return DropdownMenuItem<String>(
                  value: category,
                  child: Text(category),
                );
              }).toList(),
            ),
          ],
```

```
        ),
        SizedBox(height: 50.0),
        // Updated section with 4 cards and images
        Container(
          height: 200.0,
          child: ListView.builder(
            scrollDirection: Axis.horizontal,
            itemCount: 4, // Number of cards
            itemBuilder: (context, index) {
              return Card(
                margin: EdgeInsets.all(8.0),
                child: Container(
                  width: 150.0, // Customize the card width as needed
                  child: Column(
                    children: [
                      // Add an Image widget with the image path
                      Image.asset(
                        'assets/card_image_${index + 1}.jpeg',
                        height: 120.0,
                        width: 150.0,
                        fit: BoxFit.cover,
                      ),
                      ListTile(
                        title: Text('Card ${index + 1}'),
                        subtitle: Text('Description of card ${index + 1}'),
                        // Add onTap callback if needed
                      ),
                    ],
                  ),
                ),
              );
            },
          ),
        ),
      ],
    ),
  );
}
}
```

Output -



Conclusion -

Through this experiment we understood how to insert images and fonts into the flutter app and how to use them and call them from the assets folder .

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll no.	50
Name	Tejas Shailendra Rokade
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Name - Tejas Shailendra Rokade
Div - D15A Roll no - 50
Batch - C

Experiment - 4

Aim - To create an interactive Form using form widget

Theory -

Interactive Form Creation in Flutter
Flutter offers an array of powerful widgets and techniques to build adaptable and user-friendly forms that seamlessly integrate with your app's design and functionality. This guide delves into the key concepts and strategies involved in crafting interactive forms:
Fundamental Widgets:

- Form: The overarching container that encompasses form fields and manages validation. Create a unique GlobalKey<FormState> for global access and validation.
- FormField: A base class for input widgets, providing styling, validation, and error handling. Choose the appropriate concrete widget based on the input type (e.g., TextField, Checkbox, DropdownButton).
- TextInputAction: Set the textInputAction property on TextField to control the on-screen keyboard's behavior (e.g., TextInputAction.next, TextInputAction.done).
- FocusNode: Control keyboard focus navigation between form fields or widgets within a field (e.g., for multi-line text editing). Use FocusNode objects with FocusManager for management.

Dynamic Form Building:

- ListViews: Dynamically render form fields by creating a ListView.builder that builds FormField instances based on data (e.g., from a list or JSON). Customize field types and validation based on data properties.
- State Management: Employ state management solutions like Bloc, Provider, or raw setState to handle form state effectively. Store form data, validation errors, and other state information dynamically.

User Interaction and Validation:

- Input Validation: Implement validation rules within FormField or its descendant widgets using a validator callback. Provide clear error messages for invalid input to guide users.
- Focus Management: Use FocusNode to control keyboard focus flow, enabling a natural form-filling experience. Consider using packages like auto_animated for field auto-focusing when entering the screen.
- Interactive UI: Incorporate widgets like Checkbox, DropdownButton, Radio, and custom interactive elements to offer a variety of input options and enhance user engagement.

Additional Considerations:

- Performance: For large forms, consider techniques like lazy loading or pagination to manage memory and rendering overhead.

- Accessibility: Ensure your forms are accessible to users with disabilities by following WCAG guidelines and using appropriate semantic elements.
- Styling: Customize form aesthetics using Flutter's rich customization options to match your app's design and provide visual feedback (e.g., underline active fields, highlight errors)

Code -

Login page -

```
import 'package:flutter/material.dart';
import 'package:lunching/Signuppage.dart';
import 'package:lunching/main.dart';

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

 static const tmail = "E-mail";
 static const tpassword = "Password";
 static const tforgetpassword = "Forget Password?";
 static const tFormHeight = 50.0;

 @override
 Widget build(BuildContext context) {
  return Scaffold(
    body: SingleChildScrollView(
      child: Container(
        padding: const EdgeInsets.symmetric(vertical: tFormHeight - 10),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            Image.asset(
              'assets/card_image_1.jpeg',
              height: 120.0,
              width: 150.0,
              fit: BoxFit.cover,
            ),
            Text("Lunching", style: Theme.of(context).textTheme.headline2),
            Text("The best of best", style: Theme.of(context).textTheme.headline3),
            Form(
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.center,
                children: [
                  TextFormField(
                    decoration: InputDecoration(
                      prefixIcon: Icon(Icons.person_outline_outlined),
```

```
    labelText: tmail,
    hintText: tmail,
    border: OutlineInputBorder(),
  ),
),
const SizedBox(height: 20),
TextFormField(
  obscureText: true,
  decoration: InputDecoration(
    prefixIcon: Icon(Icons.lock_outline),
    labelText: tpassword,
    hintText: tpassword,
    border: OutlineInputBorder(),
    suffixIcon: IconButton(
      onPressed: () {},
      icon: Icon(Icons.remove_red_eye),
    ),
  ),
),
const SizedBox(height: 20),
Align(
  alignment: Alignment.centerRight,
  child: TextButton(
    onPressed: () {},
    child: const Text(tforgetpassword),
  ),
),
SizedBox(
  width: double.infinity,
  child: ElevatedButton(
    onPressed: () {
      Navigator.of(context).push(
        MaterialPageRoute(builder: (context) => HomePage()),
      );
    },
    child: Text("LOG IN"),
  ),
),
const SizedBox(height: 20),
Text("OR"),
const SizedBox(height: 20),
TextButton(
  onPressed: () {},
  child: Text.rich(
```

```
                TextSpan(
                  text: "Don't have an account?",
                  style: Theme.of(context).textTheme.bodyText1,
                  children: const [

                    TextSpan(
                      text: "Register",
                      style: TextStyle(color: Colors.blue),
                    ),
                  ],
                ),
              ),
            ),
            const SizedBox(height: 20),
            SizedBox(
              width: double.infinity,
              child: ElevatedButton(
                onPressed: () {
                  Navigator.of(context).push(
                    MaterialPageRoute(builder: (context) => Signuppage()),
                  );
                },
                child: Text("Sign Up"),
              ),
            ),
          ],
        ),
      ),
    ),
  ),
  );
 }
}
```

Sign up page -

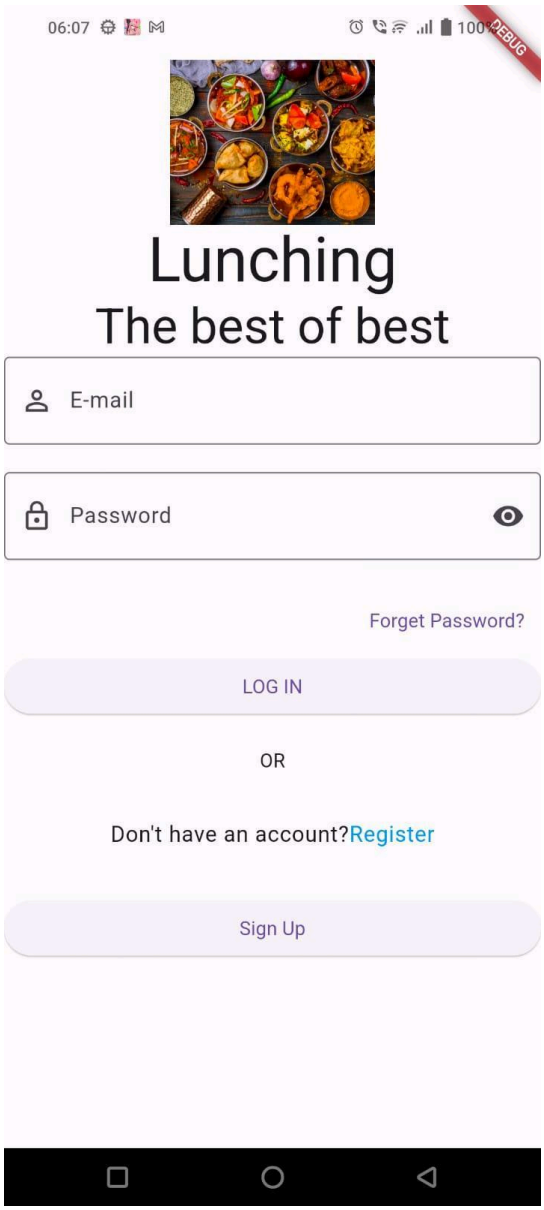
```
import 'package:flutter/material.dart';
import 'package:lunching/main.dart';

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

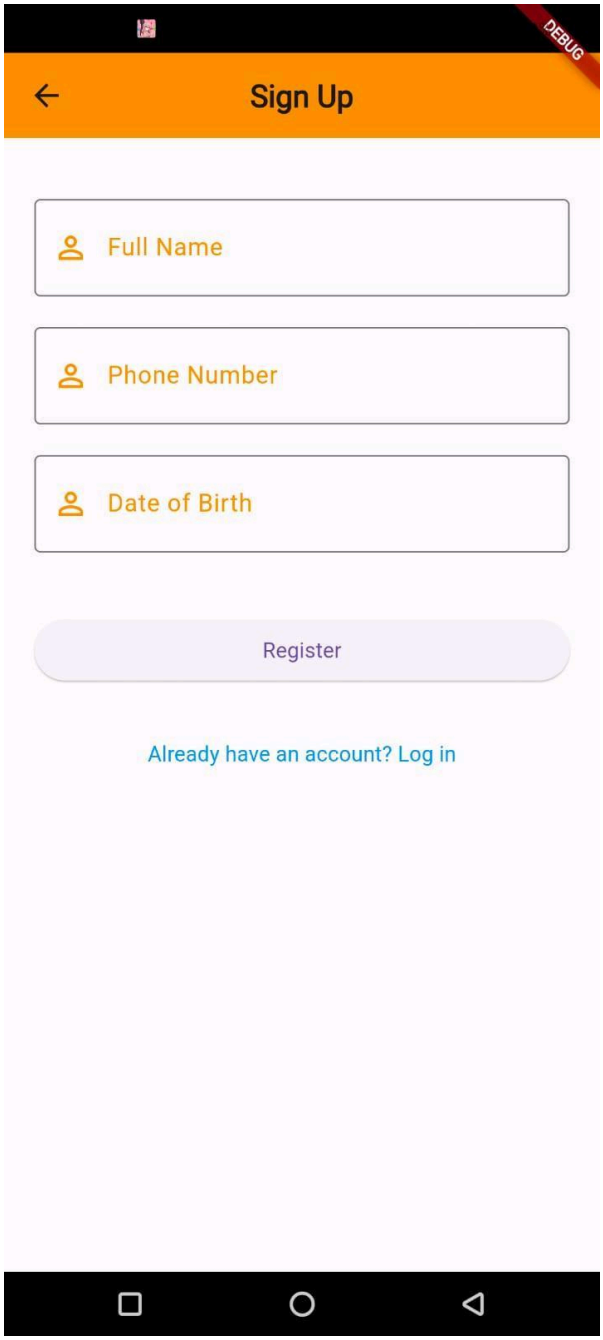
```
static const tFullName = "Full Name";
static const tPhoneNumber = "Phone Number";
static const tDateOfBirth = "Date of Birth";

@override
Widget build(BuildContext context) {
  return SafeArea(
    child: Scaffold(
      appBar: AppBar(
        title: Text(
          'Sign Up',
          style: TextStyle(
            fontWeight: FontWeight.bold,
            fontSize: 20,
          ),
        ),
        centerTitle: true,
        backgroundColor: Colors.orange,
      ),
      body: SingleChildScrollView(
        child: Container(
          padding: const EdgeInsets.all(20.0),
          child: Column(
            children: [
              Container(
                padding: const EdgeInsets.symmetric(vertical: 20),
                child: Form(
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      TextFormField(
                        decoration: _inputDecoration(tFullName),
                      ),
                      const SizedBox(height: 20),
                      TextFormField(
                        decoration: _inputDecoration(tPhoneNumber),
                        keyboardType: TextInputType.phone,
                      ),
                      const SizedBox(height: 20),
                      TextFormField(
                        decoration: _inputDecoration(tDateOfBirth),
                        keyboardType: TextInputType.datetime,
                      ),
                      const SizedBox(height: 40),
```

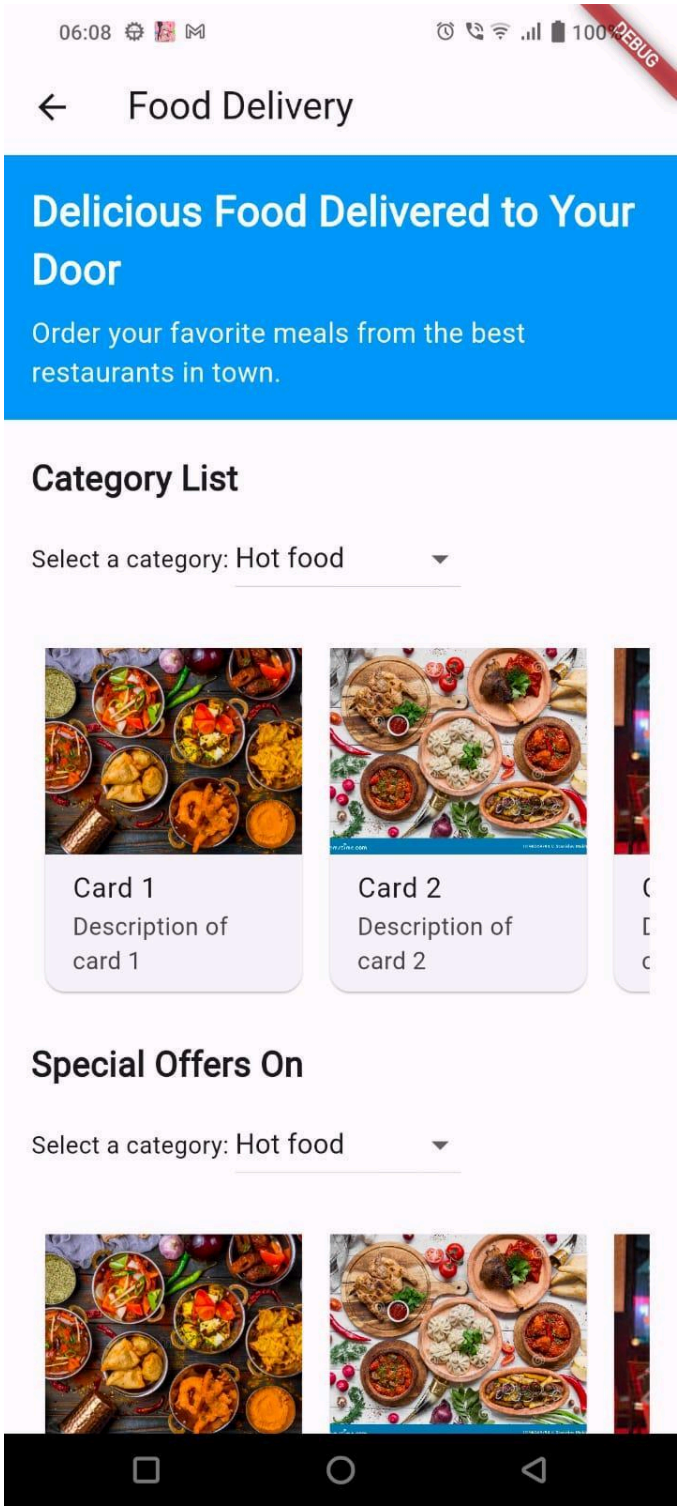
```
                SizedBox(
                  width: double.infinity,
                  child: ElevatedButton(
                    onPressed: () {
                      Navigator.of(context).push(
                        MaterialPageRoute(builder: (context) => HomePage()),
                      );
                    },
                    child: Text("Register"),
                  ),
                ),
              ],
            ),
          ),
        ),
        TextButton(
          onPressed: () {
            Navigator.of(context).pop(); // Navigate back to the login page
          },
          child: Text(
            "Already have an account? Log in",
            style: TextStyle(color: Colors.blue),
          ),
        ),
      ],
    ),
  ),
),
),
);
}

InputDecoration _inputDecoration(String labelText) {
  return InputDecoration(
    labelText: labelText,
    border: OutlineInputBorder(),
    prefixIcon: Icon(
      Icons.person_outline_outlined,
      color: Colors.orange,
    ),
    labelStyle: TextStyle(color: Colors.orange),
    focusedBorder: OutlineInputBorder(
      borderSide: BorderSide(width: 2.0, color: Colors.orange),
    ),
```

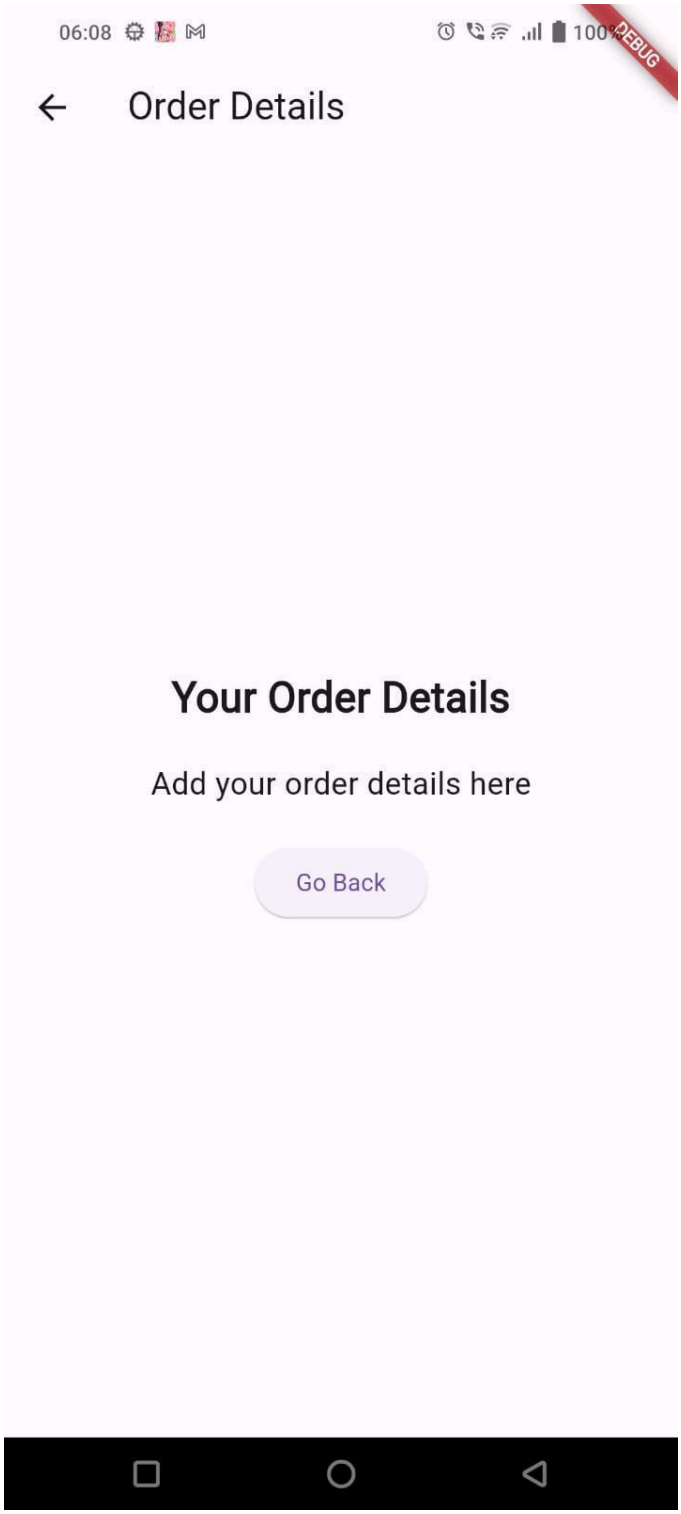
```
  );
 }
}
```

Screenshot -


06:07
DEBUG
Lunching
The best of best
E-mail
Password
Forget Password?
LOG IN
OR
Don't have an account?Register
Sign Up


DEBUG
Sign Up
Full Name
Phone Number
Date of Birth
Register
Already have an account? Log in


06:08
100%
DEBUG
Food Delivery
Delicious Food Delivered to Your Door
Order your favorite meals from the best restaurants in town.
Category List
Select a category: Hot food
Card 1
Description of card 1
Card 2
Description of card 2
Special Offers On
Select a category: Hot food



Conclusion -
Through this experiment we understood how to create interactive form using widgets in flutter and how to validate data input in the Text fields.We also learned dynamic Form building.

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll no.	50
Name	Tejas Shailendra Rokade
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Name - Tejas Shailendra Rokade
Div - D15A Roll no - 50
Batch - C

Experiment - 5

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

Theory -

Navigation and Routing in Flutter:
Flutter offers two fundamental approaches to manage navigation and routing:
1. Navigator:
   - The built-in Navigator widget provides imperative methods for pushing, popping, and replacing routes.
   - Use Navigator.push() to move forward in the navigation stack, Navigator.pop() to go back, and Navigator.popUntil() to clear the stack up to a certain point.
   - Suitable for small to medium-sized apps with straightforward navigation structures.
2. Router Packages:
   - Packages like go_router and routemaster offer declarative routing paradigms, handling deep linking and more complex navigation scenarios.
   - Define routes as data structures, and the router handles navigation seamlessly.
   - Useful for larger apps or ones with dynamic navigation requirements.

Choosing the Right Approach:
Consider these factors when selecting a navigation system:
- App size and complexity
- Deep linking needs
- Preference for imperative or declarative routing styles

Example with Navigator:
Dart
```
// First screen (home)
ElevatedButton(
  onPressed: () => Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => SecondScreen()),
  ),
  child: const Text('Go to Second Screen'),
),

// Second screen
ElevatedButton(
  onPressed: () => Navigator.pop(context),
```

```
  child: const Text('Go Back'),
),
```

Use code with caution. Learn more
content_copy
Example with go_router:
Dart

```
// main.dart
final myRouter = GoRouter(
  initialLocation: '/',
  routes: [
    GoRoute(
      path: '/',
      builder: (context, state) => Home(),
      routes: [
        GoRoute(
          path: '/second',
          builder: (context, state) => SecondScreen(),
        ),
      ],
    ),
  ],
);

// Second screen
ElevatedButton(
  onPressed: () => context.go('/'),
  child: const Text('Go Back'),
),
```

Use code with caution. Learn more
content_copy
Gestures in Flutter:
Flutter's rich gesture system allows you to create interactive and intuitive experiences. Here are some key types:

- Taps: Detect single, double, or long taps using GestureDetector or InkWell.
- Drags: Handle drags, swipes, and pan gestures using RawGestureDetector or GestureDetector.
- Scrolls: Use ListView, GridView, or custom scrollable widgets.
- Other gestures: Use packages like flutter_slidable for swipe/delete actions, pinch_zoom for pinch-to-zoom, and more.

Tips for Effective Gestures:

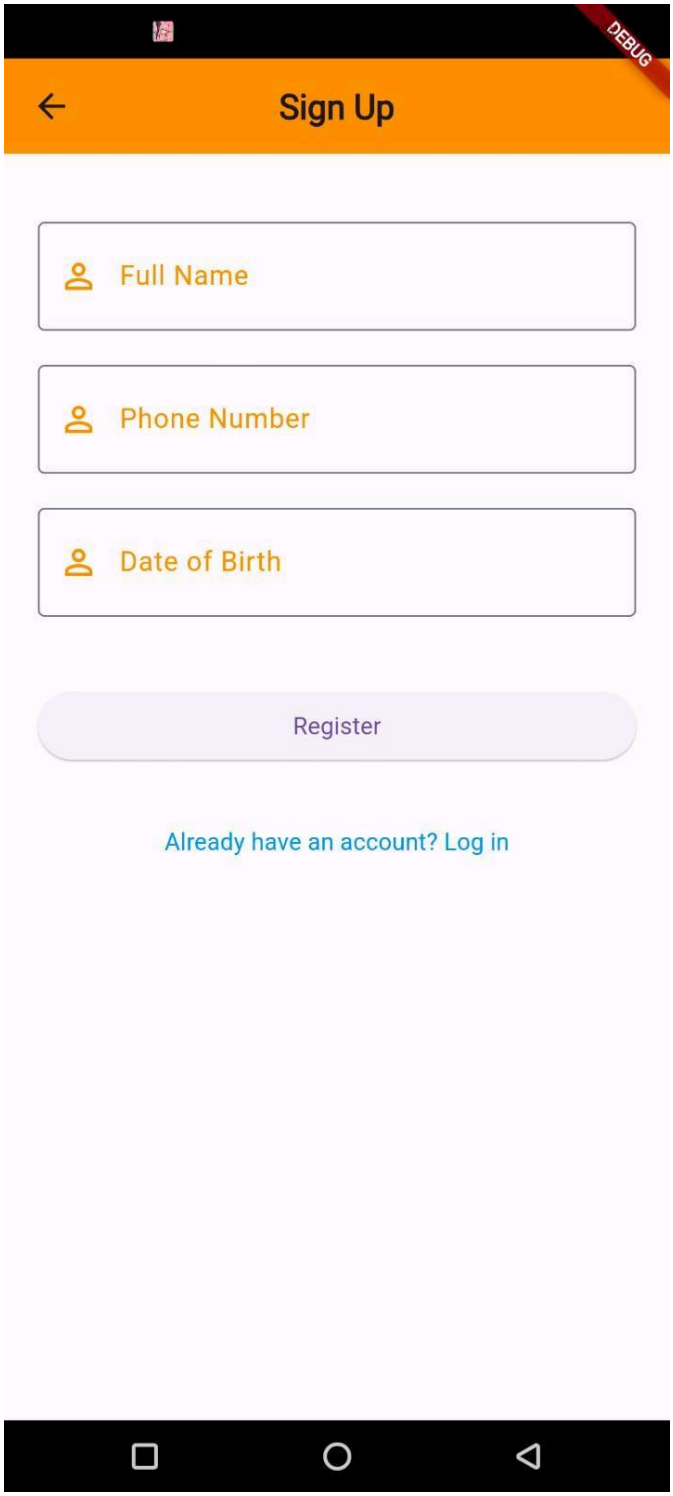
- Provide visual feedback (e.g., animations, sound effects) on gesture interaction.
- Test gestures on different devices and screen sizes.

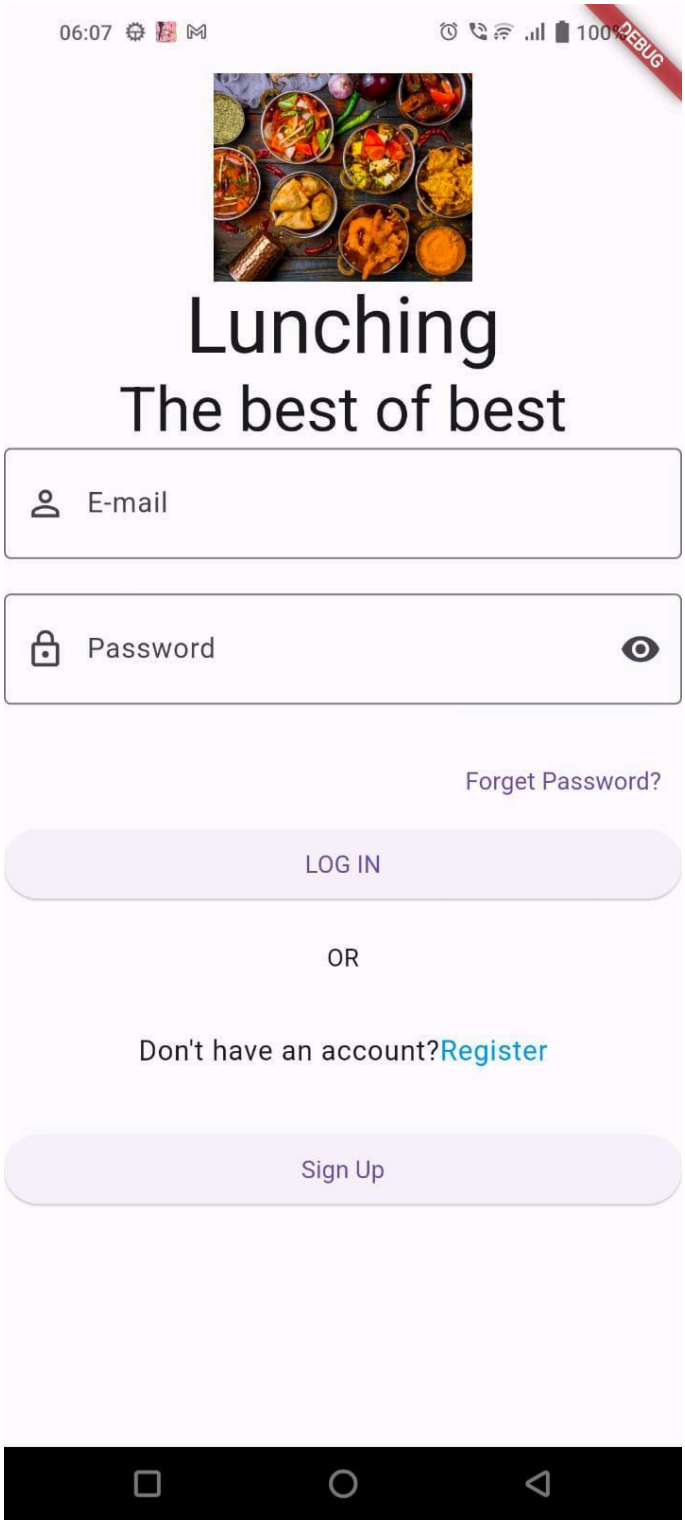
- Prioritize accessibility by considering users with different abilities.

Example with Taps:
Dart
```
GestureDetector(
  onTap: () => print('Tapped!'),
  child: Container(
    color: Colors.blue,
    child: const Text('Tap me'),
  ),
),
```

Use code with caution. Learn more
content_copy
Example with Drags:
Dart
```
RawGestureDetector(
  onPanDown: (details) => print('Drag started'),
  onPanUpdate: (details) => print('Dragging (dx: ${details.delta.dx}, dy: ${details.delta.dy})'),
  onPanEnd: (details) => print('Drag ended'),
  child: Container(
    color: Colors.green,
    child: const Text('Drag me'),
  ),
),
```

Screenshot -


DEBUG
Sign Up
Full Name
Phone Number
Date of Birth
Register
Already have an account? Log in



Conclusion -

Through this experiment we understood implementation of navigator , haptics and gestures for implementing navigation from one tab to another tab as per need and gestures to recognise and for events to occur in the flutter application.

MAD & PWA Lab Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll no.	50
Name	Tejas Shailendra Rokade
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	15

Name - Tejas Shailendra Rokade
Div - D15A Roll no - 50
Batch - C

Experiment - 6

Aim - To Connect Flutter UI with fireBase database

Theory -

Firebase is a popular mobile and web application development platform powered by Google, offering a suite of tools and services to streamline various aspects of app development. In Flutter, Google's UI toolkit for building natively compiled applications for mobile, web, and desktop from a single codebase, Firebase integration is seamless and efficient.

Firebase provides several services that cater to different aspects of app development:

1. Authentication: Firebase Authentication offers simple SDKs and ready-to-use UI libraries to authenticate users across various platforms. With Firebase Authentication, developers can implement authentication mechanisms such as email/password, phone number, social logins (Google, Facebook, Twitter, etc.), and more into their Flutter apps with ease.

2. Realtime Database: Firebase Realtime Database is a cloud-hosted NoSQL database that allows developers to store and synchronize data between users in real-time. It's particularly useful for applications that require real-time updates, such as chat apps, collaborative tools, and multiplayer games. Flutter developers can seamlessly integrate Firebase Realtime Database into their apps to store and sync structured data.

3. Cloud Firestore: Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform. It offers more powerful querying capabilities, real-time data synchronization, and offline support compared to the Realtime Database. Firestore is commonly used in Flutter apps for its scalability, real-time updates, and robust querying capabilities.

4. Cloud Storage: Firebase Cloud Storage provides secure and scalable object storage for storing user-generated content such as images, videos, and documents. It offers SDKs for easy integration into Flutter apps, allowing developers to upload, download, and manage files seamlessly.

5. Cloud Functions: Firebase Cloud Functions allows developers to run backend code in response to events triggered by Firebase features and HTTPS requests. Developers can write serverless functions in Node.js, Python, Java, Go, or TypeScript, and deploy them directly from

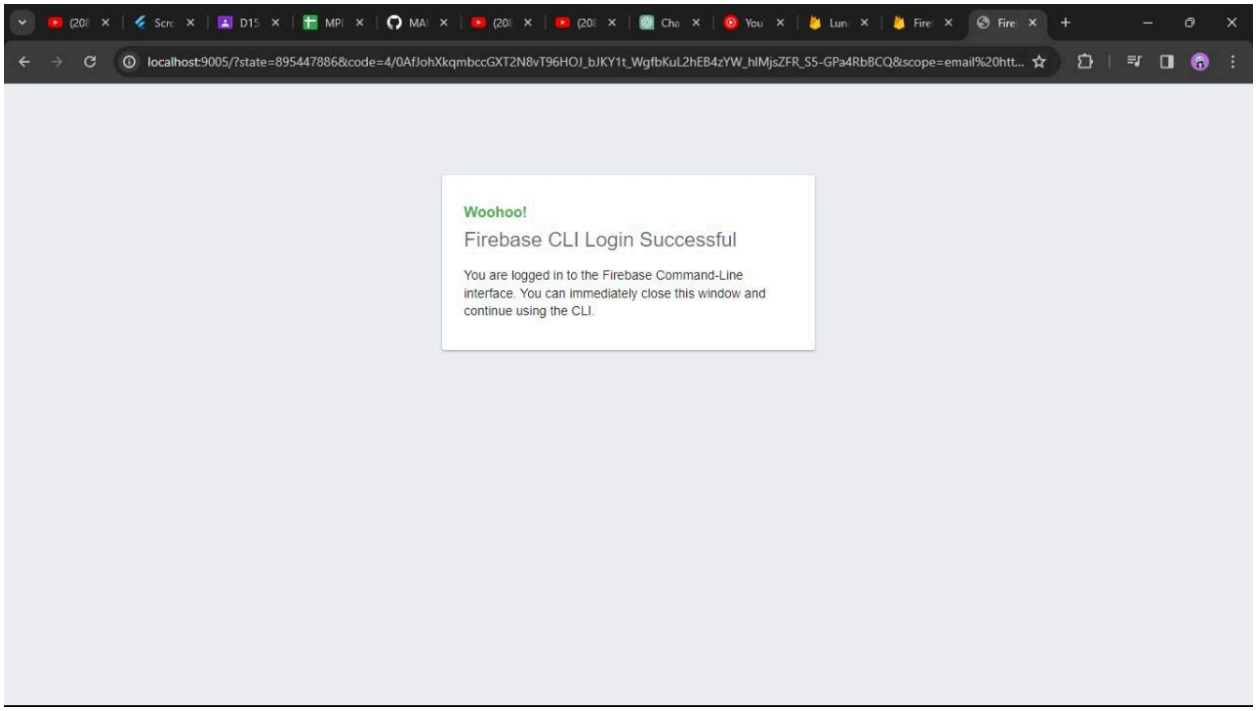
the Firebase CLI. Cloud Functions are often used to automate tasks, process data, and integrate with other Google Cloud services.

6. Cloud Messaging: Firebase Cloud Messaging (FCM) enables developers to send notifications and messages to users across platforms, including Android, iOS, and web. With FCM, Flutter developers can engage users with targeted messages, notifications, and updates, improving user retention and engagement.

7. Remote Config: Firebase Remote Config allows developers to customize the behavior and appearance of their Flutter apps without requiring an app update. It enables A/B testing, feature flag management, and dynamic content personalization, empowering developers to deliver tailored experiences to their users.

8. Analytics: Firebase Analytics provides comprehensive app usage and event tracking capabilities, allowing developers to gain insights into user behavior, app performance, and user engagement. Flutter developers can easily integrate Firebase Analytics into their apps to track user interactions, measure app performance, and make data-driven decisions.

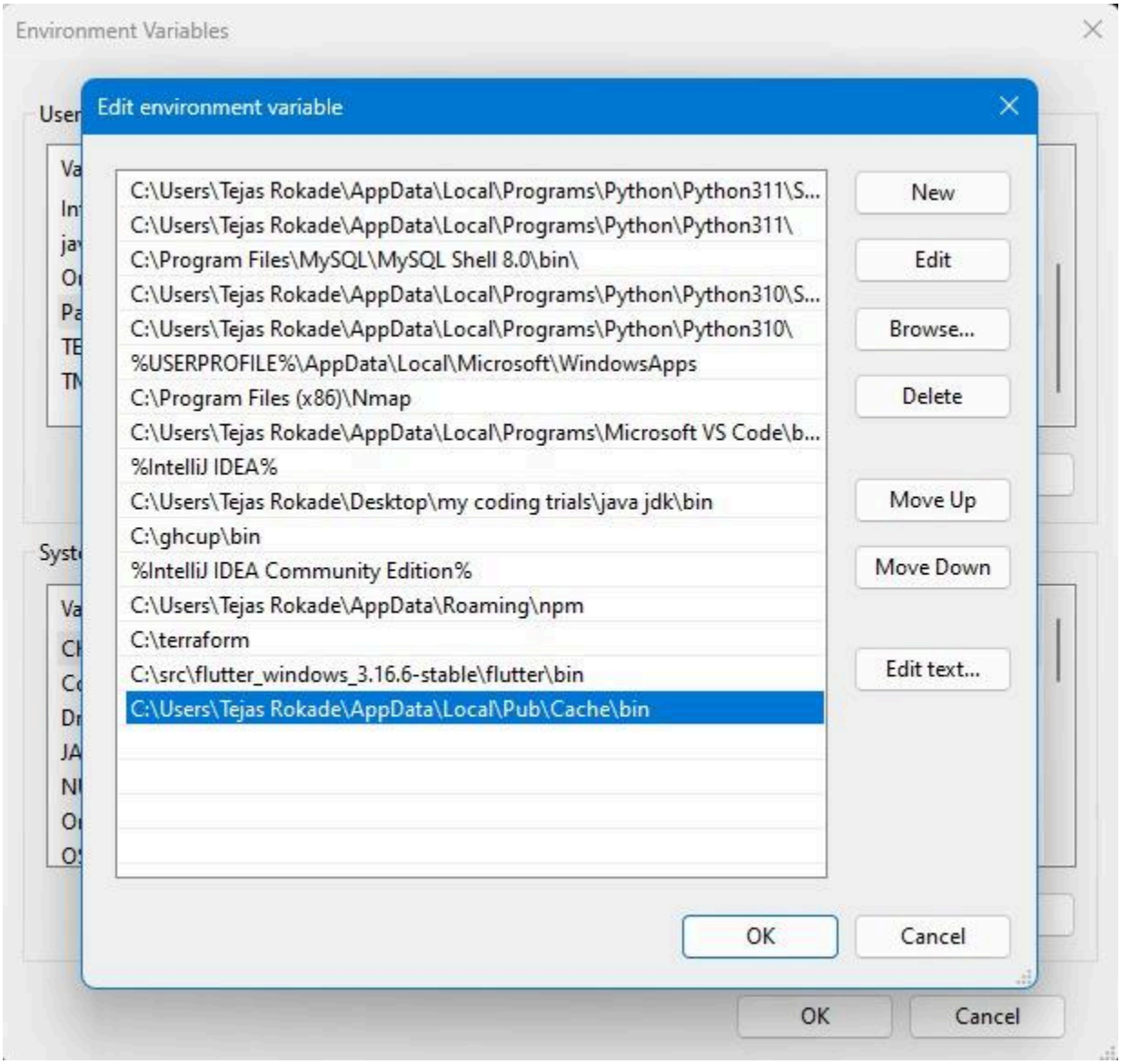
These are just a few of the many services offered by Firebase for Flutter app development. By leveraging Firebase's powerful tools and services, Flutter developers can build high-quality, feature-rich apps more efficiently, saving time and effort in the development process.



```
C:\WINDOWS\system32\cmd.

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

C:\Users\Tejas Rokade>npm install -g firebase-tools

added 640 packages in 3m

65 packages are looking for funding
  run `npm fund` for details

C:\Users\Tejas Rokade>firebase login
i  Firebase optionally collects CLI and Emulator Suite usage and error reporting information to help improve our product
s. Data is collected in accordance with Google's privacy policy (https://policies.google.com/privacy) and is not used to
 identify you.

? Allow Firebase to collect CLI and Emulator Suite usage and error reporting information? No

Visit this URL on this device to log in:
https://accounts.google.com/o/oauth2/auth?client_id=563584335869-fgrhgmd47bqnekij5i8b5pr03ho849e6.apps.googleusercontent
.com&scope=email%20openid%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloudplatformprojects.readonly%20https%3A%2F%2Fwww
.googleapis.com%2Fauth%2Ffirebase%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform&response_type=code&state=89
5447886&redirect_uri=http%3A%2F%2Flocalhost%3A9005

Waiting for authentication...

+  Success! Logged in as tejasrok007@gmail.com

C:\Users\Tejas Rokade>dart pub global activate flutterfire_cli
  ╔═══════════════════════════════════════════════════════════╗
  ═ The Dart tool uses Google Analytics to report feature usage statistics  ═
```

```
C:\Users\Tejas Rokade>dart pub global activate flutterfire_cli
  ╔═══════════════════════════════════════════════════════════╗
  ═ 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                                              ═
  ═                                                                         ═
  ═══════════════════════════════════════════════════════════╝

+ ansi_styles 0.3.2+1s... (3.1s)
+ args 2.4.2
+ async 2.11.0
+ boolean_selector 2.1.1
+ characters 1.3.0
+ ci 0.1.0
+ cli_util 0.3.5 (0.4.1 available)
+ clock 1.1.1
+ collection 1.18.0
+ dart_console 1.2.0
+ deep_pick 0.10.0 (1.0.0 available)
```

```
+ test_api 0.7.0
+ tint 2.0.1
+ typed_data 1.3.2
+ uri 1.0.0
+ win32 5.2.0
+ xml 6.5.0
+ yaml 3.1.2
Building package executables... (11.7s)
Built flutterfire_cli:flutterfire.
Installed executable flutterfire.
Warning: Pub installs executables into C:\Users\Tejas Rokade\AppData\Local\Pub\Cache\bin, which is not on your path.
You can fix that by adding that directory to your system's "Path" environment variable.
A web search for "configure windows path" will show you how.
Activated flutterfire_cli 0.2.7.

C:\Users\Tejas Rokade>
```

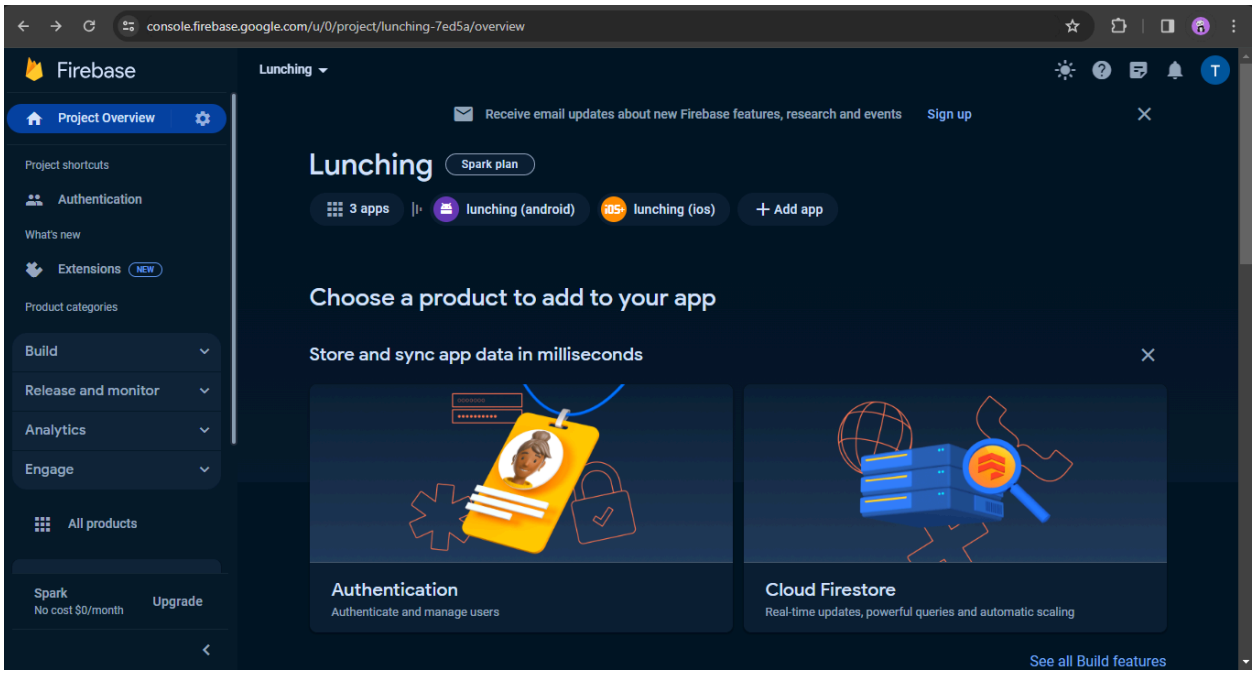


Command Prompt - dart pub

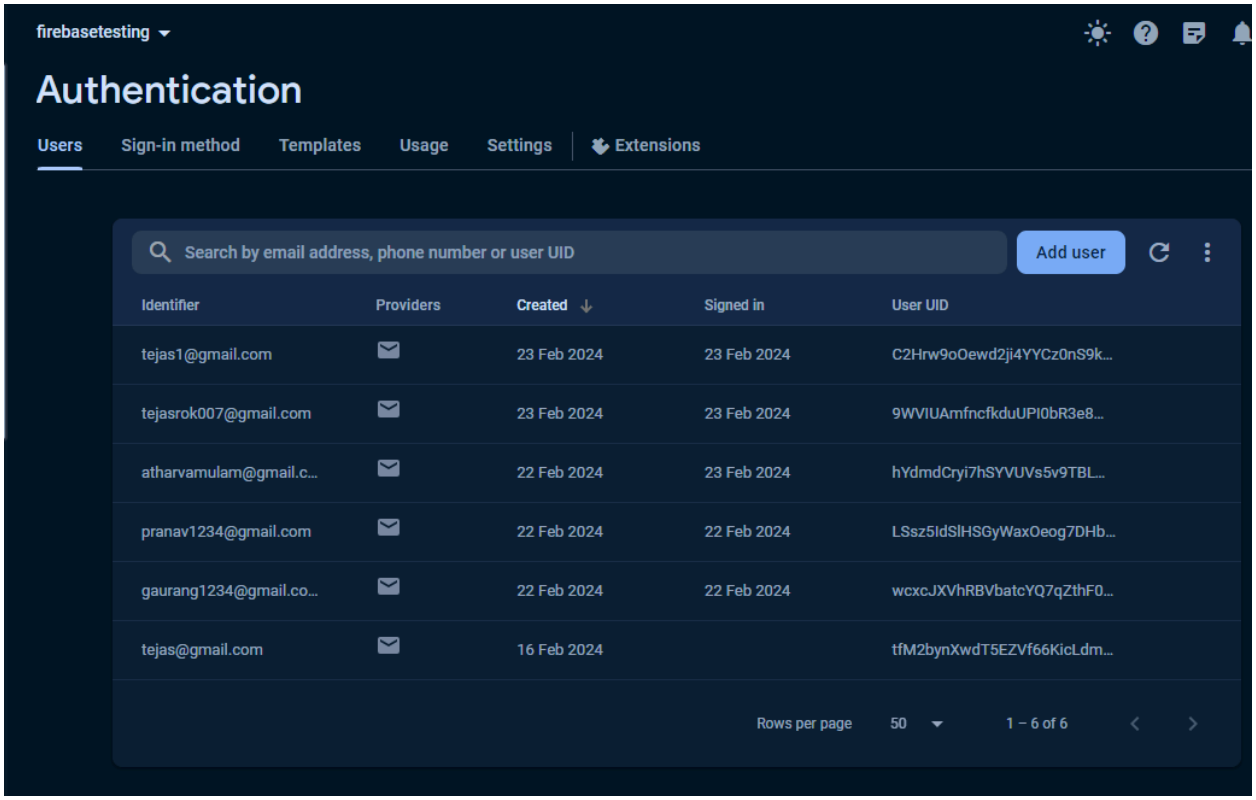
```
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Tejas Rokade>dart pub global activate flutterfire_cli
Package flutterfire_cli is currently active at version 0.2.7.
The package flutterfire_cli is already activated at newest available version.
To recompile executables, first run `dart pub global deactivate flutterfire_cli`.
Installed executable flutterfire.
Activated flutterfire_cli 0.2.7.

C:\Users\Tejas Rokade>
```

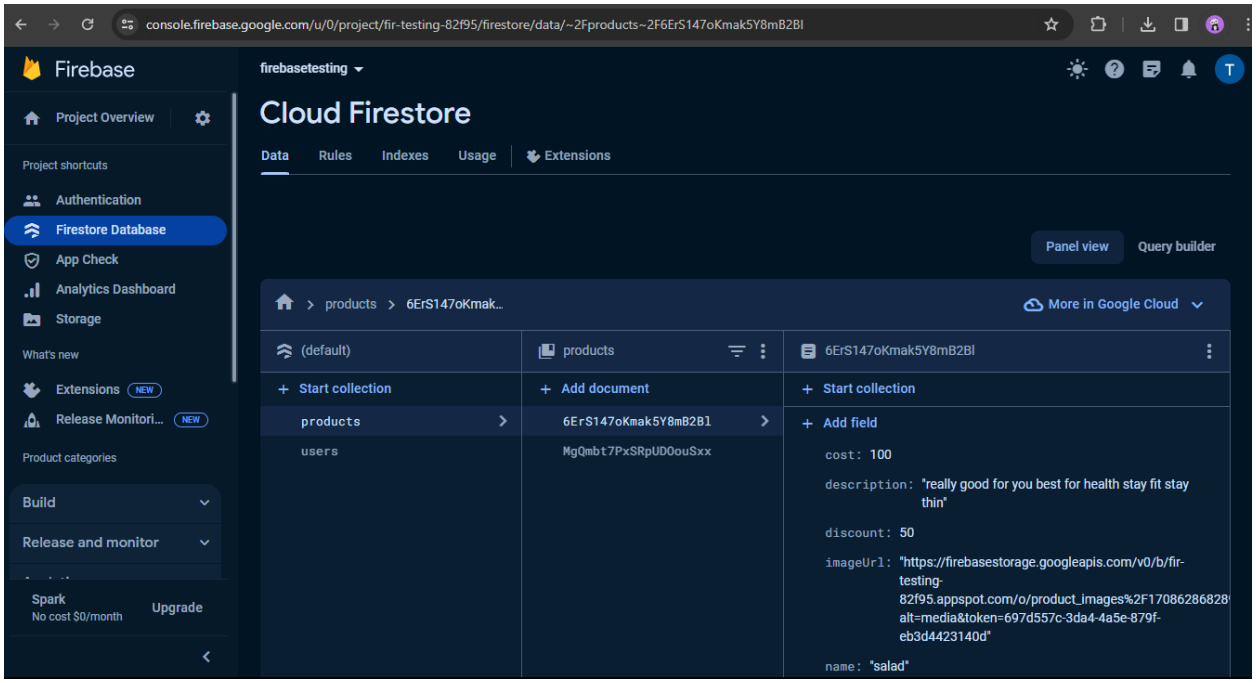
Terminal: Local

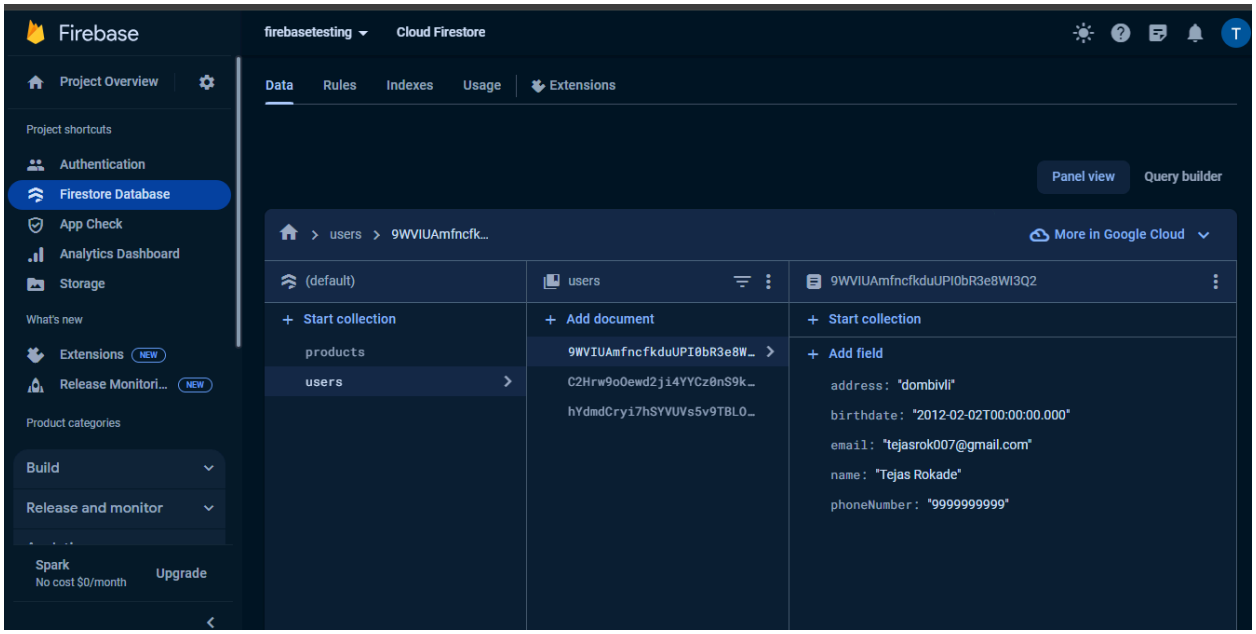
```
Firebase configuration file lib\firebase_options.dart generated successfully with the following Firebase apps:

Platform  Firebase App Id
web       1:239334465731:web:ba8fad3e01d5f2fe200276
android   1:239334465731:android:0996f1103a910d57200276
ios       1:239334465731:ios:f56a7d0376c143c4200276

Learn more about using this file and next steps from the documentation:
 > https://firebase.google.com/docs/flutter/setup
PS C:\Users\Tejas Rokade\StudioProjects\lunching>
```

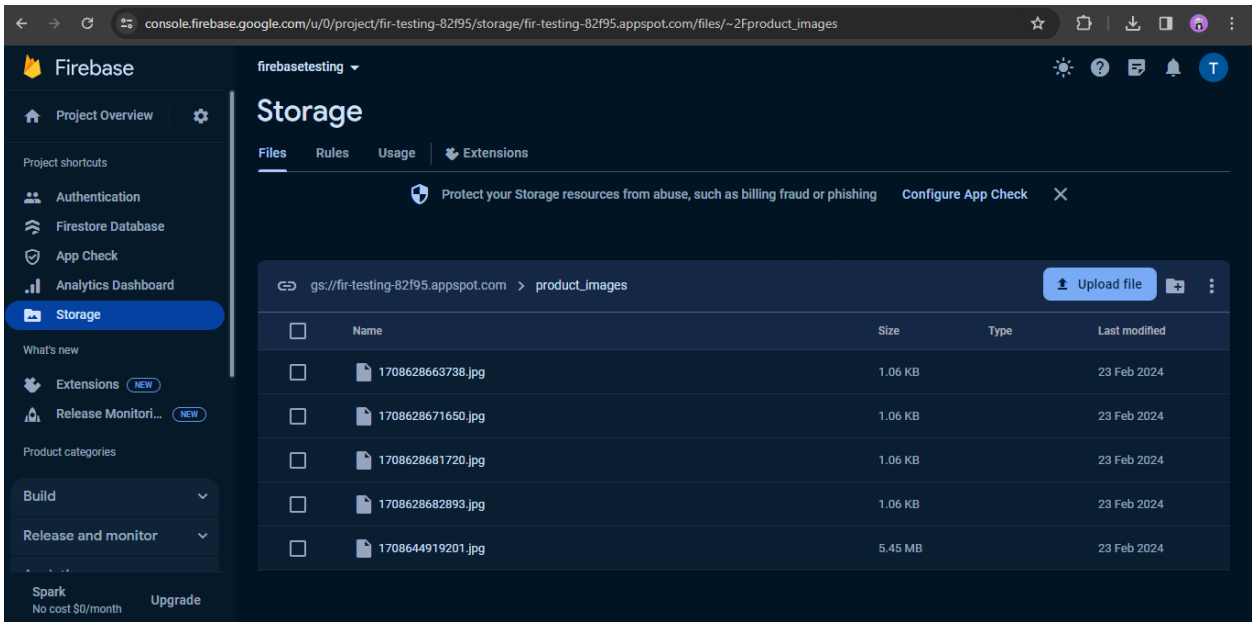


Project Title:Lunching and Glowing MAD /PWA Lab **Roll no.- 50**









Code -

```
void main() async {
 WidgetsFlutterBinding.ensureInitialized();
 await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform,);
 runApp(MyApp());
}
class loginscreen extends StatefulWidget {
 const loginscreen({Key? key}) : super(key: key);
 @override
 State<loginscreen> createState() => _loginscreenState();
}
```

```
class _loginscreenState extends State<loginscreen> {
 final TextEditingController _emailController = TextEditingController();
 final TextEditingController _passwordController = TextEditingController();
 @override
 void dispose(){
   _emailController.dispose();
   _passwordController.dispose();
   super.dispose();
 }
 void loginUser() async{
   // Implementation of loginUser method
   String res = await AuthMethods().loginUser(email: _emailController.text, password: _passwordController.text);
   if(res == 'success'){
     Navigator.of(context as BuildContext).push(
       MaterialPageRoute(
         builder: (context) => HomePage(),
       ),
     );
   }
   else{
     // Show authentication failed message
     ScaffoldMessenger.of(context).showSnackBar(
       SnackBar(
         content: Text('Authentication failed'),
       ),
     );
   }
 }
 void navigateSignup(){
   // Implementation of navigateSignup method
   Navigator.of(context as BuildContext).push(
     MaterialPageRoute(builder: (context) => SignupPage()),
   );
 }
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     body: SafeArea(
       child: Container(
         padding: const EdgeInsets.symmetric(horizontal: 32),
         child: Column(
           crossAxisAlignment: CrossAxisAlignment.center,
           children: [
             const Text(
               'Lunching',
               style: TextStyle(
                 fontSize: 24,
                 fontWeight: FontWeight.bold,
                 color: Colors.red,
               ),
             ),
             const SizedBox(height: 20),
             // Image.network(
             //   imageUrl,
```

```
          //   height: 150, // Adjust height as needed
          //   width: double.infinity,
          //   fit: BoxFit.cover,
          // ),
          Image.asset('assets/card_image_1.jpeg',height: 200,width:double.infinity ,fit: BoxFit.cover,),
          const SizedBox(height: 24),
          InputTextField(
            textEditingController: _emailController,
            hintText: 'Email',
            textInputType: TextInputType.emailAddress,
          ),
          const SizedBox(height: 24),
          InputTextField(
            textEditingController: _passwordController,
            hintText: 'Password',
            textInputType: TextInputType.text,
            isPass: true,
          ),
          const SizedBox(height: 24),
          ElevatedButton(
            onPressed: loginUser,
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.green, // Set button color to green
            ),
            child: const Text('Log In'),
          ),
          const SizedBox(height: 20),
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text("Don't have an account?"),
              GestureDetector(
                onTap: navigateSignup,
                child: Text(
                  " Sign up",
                  style: TextStyle(
                    fontWeight: FontWeight.bold,
                    color: Colors.blue, // Keep signup link color blue
                  ),
                ),
              )
            ],
          )
        ],
      ),
    ),
  ),
);
}
}
class InputTextField extends StatelessWidget {
 final TextEditingController textEditingController;
 final String hintText;
 final TextInputType textInputType;
```

```
 final bool isPass;

 const InputTextField({
   Key? key,
   required this.textEditingController,
   required this.hintText,
   required this.textInputType,
   this.isPass = false,
 }) : super(key: key);

 @override
 Widget build(BuildContext context) {
   return TextField(
     controller: textEditingController,
     keyboardType: textInputType,
     obscureText: isPass,
     decoration: InputDecoration(
       hintText: hintText,
       border: OutlineInputBorder(),
     ),
   );
 }
}
class DatabaseMethods {
 Future<void> addUserDetail(Map<String, dynamic> userInfoMap, String id) async {
   return await FirebaseFirestore.instance
       .collection('users')
       .doc(id)
       .set(userInfoMap);
 }
}
class SignupPage extends StatefulWidget {
 @override
 _SignupPageState createState() => _SignupPageState();
}
class _SignupPageState extends State<SignupPage> {
 final FirebaseAuth _auth = FirebaseAuth.instance;
 final DatabaseMethods _databaseMethods = DatabaseMethods();
 final TextEditingController _emailController = TextEditingController();
 final TextEditingController _passwordController = TextEditingController();
 final TextEditingController _nameController = TextEditingController();
 final TextEditingController _addressController = TextEditingController();
 final TextEditingController _phoneNumberController = TextEditingController();
 DateTime? _selectedDate;
 Future<void> _signUpWithEmailAndPassword() async {
   try {
     final UserCredential userCredential = await _auth.createUserWithEmailAndPassword(
       email: _emailController.text.trim(),
       password: _passwordController.text.trim(),
     );
     await _databaseMethods.addUserDetail({
       'email': _emailController.text.trim(),
       'name': _nameController.text.trim(),
       'address': _addressController.text.trim(),
```

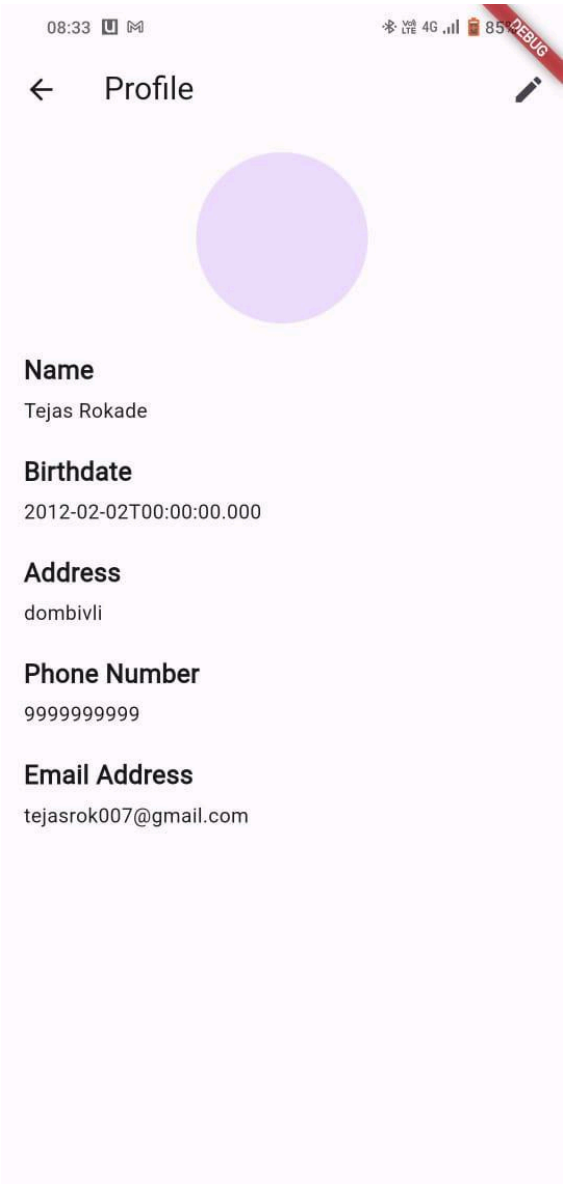
```
      'phoneNumber': _phoneNumberController.text.trim(),
      'birthdate': _selectedDate != null ? _selectedDate!.toIso8601String() : null,
    }, userCredential.user!.uid);
  } catch (e) {
    print("Failed to register with email and password: $e");
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: Text("Registration Error"),
          content: Text("Failed to register with email and password."),
          actions: <Widget>[
            TextButton(
              child: Text("OK"),
              onPressed: () {
                Navigator.of(context).pop();
              },
            ),
          ],
        );
      },
    );
  }
}
Future<void> _selectDate(BuildContext context) async {
  final DateTime? picked = await showDatePicker(
    context: context,
    initialDate: DateTime.now(),
    firstDate: DateTime(1900),
    lastDate: DateTime.now(),
  );
  if (picked != null && picked != _selectedDate)
    setState(() {
      _selectedDate = picked;
    });
}
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("Signup"),
    ),
    body: SingleChildScrollView(
      child: Padding(
        padding: EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            TextField(
              controller: _emailController,
              decoration: InputDecoration(
                labelText: 'Email',
                border: OutlineInputBorder(
```

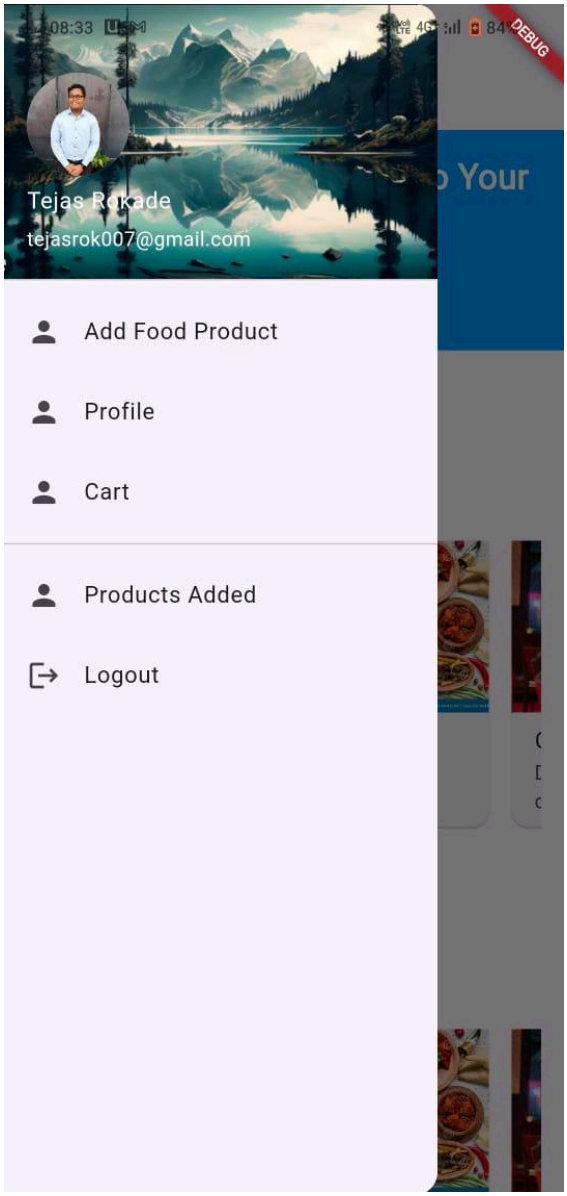
```
      borderSide: BorderSide(color: Colors.lightBlue),
    ),
  ),
),
SizedBox(height: 16.0),
TextField(
  controller: _passwordController,
  obscureText: true,
  decoration: InputDecoration(
    labelText: 'Password',
    border: OutlineInputBorder(
      borderSide: BorderSide(color: Colors.lightBlue),
    ),
  ),
),
SizedBox(height: 16.0),
TextField(
  controller: _nameController,
  decoration: InputDecoration(
    labelText: 'Name',
    border: OutlineInputBorder(
      borderSide: BorderSide(color: Colors.lightBlue),
    ),
  ),
),
SizedBox(height: 16.0),
TextField(
  controller: _addressController,
  decoration: InputDecoration(
    labelText: 'Address',
    border: OutlineInputBorder(
      borderSide: BorderSide(color: Colors.lightBlue),
    ),
  ),
),
SizedBox(height: 16.0),
TextField(
  controller: _phoneNumberController,
  decoration: InputDecoration(
    labelText: 'Phone Number',
    border: OutlineInputBorder(
      borderSide: BorderSide(color: Colors.lightBlue),
    ),
  ),
),
SizedBox(height: 16.0),
GestureDetector(
  onTap: () => _selectDate(context),
  child: AbsorbPointer(
    child: TextFormField(
      decoration: InputDecoration(
        labelText: 'Birthdate',
        border: OutlineInputBorder(
          borderSide: BorderSide(color: Colors.lightBlue),
```

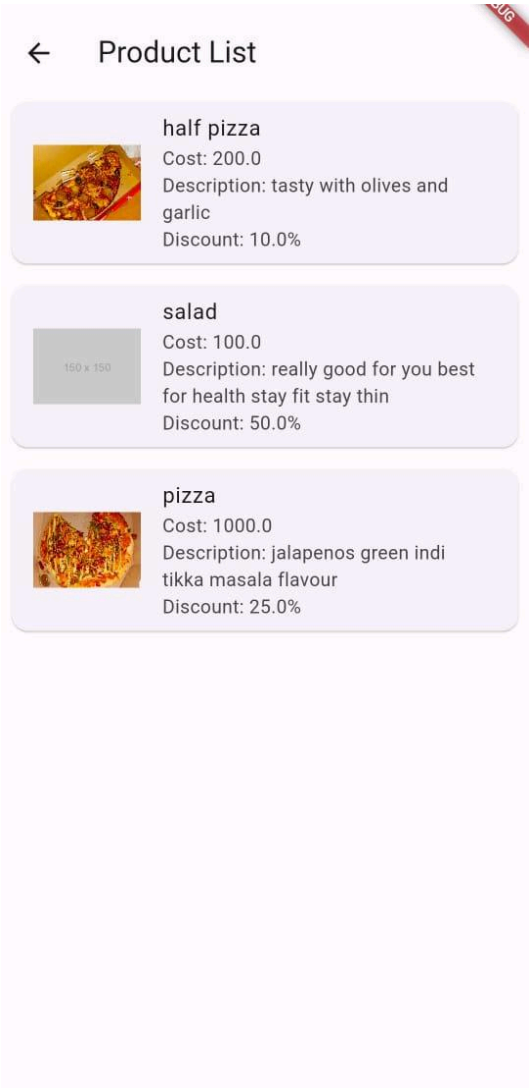
```
                ),
                suffixIcon: Icon(Icons.calendar_today),
              ),
              controller: TextEditingController(
                text: _selectedDate != null
                    ? '${_selectedDate!.day}/${_selectedDate!.month}/${_selectedDate!.year}'
                    : '',
              ),
            ),
          ),
        ),
        SizedBox(height: 32.0),
        ElevatedButton(
          onPressed: _signUpWithEmailAndPassword,
          style: ElevatedButton.styleFrom(
            foregroundColor: Colors.white, backgroundColor: Colors.lightGreen,
          ),
          child: Text('Sign Up'),
        ),
      ],
    ),
  ),
),
);
}
}

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

 @override
 Widget build(BuildContext context) {
   return Drawer(
     child: ListView(
       padding: EdgeInsets.zero,
       children: [
         StreamBuilder(
           stream: FirebaseAuth.instance.authStateChanges(),
           builder: (BuildContext context, AsyncSnapshot<User?> snapshot) {
             if (snapshot.connectionState == ConnectionState.waiting) {
               return CircularProgressIndicator(); // Show loading indicator while fetching user data
             }

             if (!snapshot.hasData || snapshot.data == null) {
               return Text('No user signed in'); // Handle the case where no user is signed in
             }

             String? userId = snapshot.data!.uid;

             return StreamBuilder(
               stream: FirebaseFirestore.instance
                   .collection('users')
                   .doc(userId)
                   .snapshots(),
```

```
        builder: (BuildContext context, AsyncSnapshot<DocumentSnapshot> snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return CircularProgressIndicator(); // Show loading indicator while fetching data
          }

          if (snapshot.hasError) {
            return Text('Error: ${snapshot.error}');
          }

          if (!snapshot.hasData || !snapshot.data!.exists) {
            return Text('No data found'); // Handle the case where user data is not available
          }

          // Access user data from snapshot
          Map<String, dynamic> userData = snapshot.data!.data() as Map<String, dynamic>;
          String accountName = userData['name'] ?? ''; // Get account name from user data
          String accountEmail = userData['email'] ?? ''; // Get account email from user data
          return UserAccountsDrawerHeader(
            accountName: Text(accountName),
            accountEmail: Text(accountEmail),
            currentAccountPicture: CircleAvatar(
              backgroundImage: AssetImage('assets/Profile_image_1.jpg'),
            ),
            decoration: BoxDecoration(
              color: Colors.blueAccent,
              // Add background image if needed
              image: DecorationImage(
                image: AssetImage('assets/background_image_1.jpg'),
                fit: BoxFit.cover,
              ),
            ),
          );
        },
      );
    },
  ),
  ListTile(
    leading: Icon(Icons.person),
    title: Text('Add Food Product'),
    onTap: () {
      Navigator.of(context).push(
        MaterialPageRoute(builder: (context) => AddProductPage()),
      );
    },
  ),
  ListTile(
    leading: Icon(Icons.person),
    title: Text('Profile'),
    onTap: () {
      Navigator.of(context).push(
        MaterialPageRoute(builder: (context) => ProfilePage()),
      );
    },
  ),
```

```
        ListTile(
          leading: Icon(Icons.person),
          title: Text('Cart'),
          onTap: () {
            Navigator.of(context).push(
              MaterialPageRoute(builder: (context) => CartPage()),
            );
          },
        ),
        Divider(),
        ListTile(
          leading: Icon(Icons.person),
          title: Text('Products Added'),
          onTap: () {
            Navigator.of(context).push(
              MaterialPageRoute(builder: (context) => ProductList()),
            );
          },
        ),
        ListTile(
          leading: Icon(Icons.logout),
          title: Text('Logout'),
          onTap: () {
            // Logout functionality
            // Example:
            // Navigator.of(context).pop(); // Close drawer
            // Navigator.of(context).pushReplacement(
            //   MaterialPageRoute(builder: (context) => LoginScreen()),
            // );
          },
        ),
      ],
    ),
  );
}
}
```


08:33
4G
85
DEBUG
Profile
Name
Tejas Rokade
Birthdate
2012-02-02T00:00:00.000
Address
dombivli
Phone Number
9999999999
Email Address
tejasrok007@gmail.com


DEBUG
Tejas Rokade
tejasrok007@gmail.com
Add Food Product
Profile
Cart
Products Added
Logout



Conclusion -

Through this experiment we understood implementation of navigator , haptics and gestures for implementing navigation from one tab to another tab as per need and gestures to recognise and for events to occur in the flutter application.

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.	50
Name	Tejas Shailendra Rokade
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	15

Name - Tejas Shailendra Rokade
Div - D15A Roll no - 50
Batch - C

Experiment - 7

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

Theory -

It's clear that PWAs offer several advantages over traditional web apps, particularly in terms of providing a more native-like experience, ease of access, faster services, engaging features, and real-time data access.
PWAs leverage modern web technologies and principles to deliver a more app-like experience to users while maintaining the flexibility and accessibility of web apps. By incorporating features such as service workers, PWAs can offer offline capabilities, faster loading times, and seamless updates without requiring users to download or install new versions manually.
The key features of PWAs that distinguish them from regular web apps include:
1. Progressive Enhancement: PWAs are built with progressive improvement principles, ensuring that they work for every user regardless of the browser or device they are using.

2. Responsive Design: PWAs adapt to various screen sizes and orientations, providing a consistent user experience across desktops, mobile devices, and tablets.

3. App-like Experience: PWAs behave like native apps in terms of interaction and navigation, providing a smoother and more immersive user experience.

4. Automatic Updates: PWAs can receive updates automatically from the server, ensuring that users always have access to the latest features and content without needing to manually update the app.

5. Security: PWAs are served over HTTPS to ensure secure connections and prevent unauthorized access or tampering of data.

6. Search Engine Indexing: PWAs are identified as applications and can be indexed by search engines, improving discoverability and visibility.

7. Reactivatable: PWAs make it easy for users to reactivate the application and receive notifications, enhancing user engagement and retention.

Overall, PWAs represent a significant advancement in web app development, offering a compelling alternative to traditional native apps with their enhanced performance, user experience, and accessibility across various platforms and devices.

Code -
Index.html

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

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="theme-color" content="#4285f4">
  <meta name="viewport"
       content="width=device-width,
              initial-scale=1">
  <meta http-equiv="X-UA-Compatible"
       content="ie=edge">
  <title>Glowing - Reveal The Beauty of Skin</title>

  <link rel="shortcut icon" href="./favicon.svg" type="image/svg+xml">

  <link rel="stylesheet" href="./assets/css/style.css">

  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link
href="https://fonts.googleapis.com/css2?family=Urbanist:wght@400;500;600;700;800&display=
swap" rel="stylesheet">


  <link rel="preload" as="image" href="./assets/images/logo.png">
  <link rel="preload" as="image" href="./assets/images/hero-banner-1.jpg">
  <link rel="preload" as="image" href="./assets/images/hero-banner-2.jpg">
  <link rel="preload" as="image" href="./assets/images/hero-banner-3.jpg">
  <link rel="manifest" href="manifest.json">
  <script src="serviceworker.js"></script>
</head>

<body id="top">
  <script>
                // Add event listener to execute code when page loads
                window.addEventListener('load', () => {
```

```
                // Call registerSW function when page loads
                registerSW();
        });

        // Register the Service Worker
        async function registerSW() {
                // Check if browser supports Service Worker
                if ('serviceWorker' in navigator) {
                        try {
                                // Register the Service Worker named 'serviceworker.js'
                                await navigator.serviceWorker.register('serviceworker.js');
                        }
                        catch (e) {
                                // Log error message if registration fails
                                console.error('ServiceWorker registration failed: ', e);
                        }
                }
        }
        if ('Notification' in window) {
                Notification.requestPermission().then(function (permission) {
                        if (permission === 'granted') {
                                console.log('Notification permission granted.');
                        } else {
                                console.warn('Notification permission denied.');
                        }
                });
        }
    </script>
<header class="header">

  <div class="alert">
    <div class="container">
      <p class="alert-text">Free Shipping On All U.S. Orders $50+</p>
    </div>
  </div>

  <div class="header-top" data-header>
    <div class="container">

      <button class="nav-open-btn" aria-label="open menu" data-nav-toggler>
        <span class="line line-1"></span>
        <span class="line line-2"></span>
        <span class="line line-3"></span>
      </button>
```

```
<div class="input-wrapper">
  <input type="search" name="search" placeholder="Search product" class="search-field">

  <button class="search-submit" aria-label="search">
    <ion-icon name="search-outline" aria-hidden="true"></ion-icon>
  </button>
</div>

<a href="#" class="logo">
  <img src="./assets/images/logo.png" width="179" height="26" alt="Glowing">
</a>

<div class="header-actions">

  <button class="header-action-btn" aria-label="user">
    <ion-icon name="person-outline" aria-hidden="true" aria-hidden="true"></ion-icon>
  </button>

  <button class="header-action-btn" aria-label="favourite item">
    <ion-icon name="star-outline" aria-hidden="true" aria-hidden="true"></ion-icon>

    <span class="btn-badge">0</span>
  </button>

  <button class="header-action-btn" aria-label="cart item">
    <data class="btn-text" value="0">$0.00</data>

    <ion-icon name="bag-handle-outline" aria-hidden="true" aria-hidden="true"></ion-icon>

    <span class="btn-badge">0</span>
  </button>

</div>

<nav class="navbar">
  <ul class="navbar-list">

    <li>
      <a href="#home" class="navbar-link has-after">Home</a>
    </li>

    <li>
      <a href="#collection" class="navbar-link has-after">Collection</a>
```

```
          </li>

          <li>
            <a href="#shop" class="navbar-link has-after">Shop</a>
          </li>

          <li>
            <a href="#offer" class="navbar-link has-after">Offer</a>
          </li>

          <li>
            <a href="#blog" class="navbar-link has-after">Blog</a>
          </li>

        </ul>
      </nav>

    </div>
  </div>

</header>
<div class="sidebar">
  <div class="mobile-navbar" data-navbar>
    <div class="wrapper">
      <a href="#" class="logo">
        <img src="./assets/images/logo.png" width="179" height="26" alt="Glowing">
      </a>
      <button class="nav-close-btn" aria-label="close menu" data-nav-toggler>
        <ion-icon name="close-outline" aria-hidden="true"></ion-icon>
      </button>
    </div>
    <ul class="navbar-list">
      <li>
        <a href="#home" class="navbar-link" data-nav-link>Home</a>
      </li>
      <li>
        <a href="#collection" class="navbar-link" data-nav-link>Collection</a>
      </li>
      <li>
        <a href="#shop" class="navbar-link" data-nav-link>Shop</a>
      </li>
      <li>
        <a href="#offer" class="navbar-link" data-nav-link>Offer</a>
      </li>
```

```
      <li>
        <a href="#blog" class="navbar-link" data-nav-link>Blog</a>
      </li>
    </ul>
  </div>
  <div class="overlay" data-nav-toggler data-overlay></div>
</div>
<main>
  <article>
    <section class="section hero" id="home" aria-label="hero" data-section>
      <div class="container">

        <ul class="has-scrollbar">

          <li class="scrollbar-item">
            <div class="hero-card has-bg-image" style="background-image:
url('./assets/images/hero-banner-1.jpg')">

              <div class="card-content">

                <h1 class="h1 hero-title">
                  Reveal The <br>
                  Beauty of Skin
                </h1>

                <p class="hero-text">
                  Made using clean, non-toxic ingredients, our products are designed for everyone.
                </p>

                <p class="price">Starting at $7.99</p>

                <a href="#" class="btn btn-primary">Shop Now</a>

              </div>

            </div>
          </li>

          <li class="scrollbar-item">
            <div class="hero-card has-bg-image" style="background-image:
url('./assets/images/hero-banner-2.jpg')">

              <div class="card-content">
```

```
            <h1 class="h1 hero-title">
              Reveal The <br>
              Beauty of Skin
            </h1>

            <p class="hero-text">
              Made using clean, non-toxic ingredients, our products are designed for everyone.
            </p>

            <p class="price">Starting at $7.99</p>

            <a href="#" class="btn btn-primary">Shop Now</a>

          </div>

        </div>
      </li>

      <li class="scrollbar-item">
        <div class="hero-card has-bg-image" style="background-image:
url('./assets/images/hero-banner-3.jpg')">

          <div class="card-content">
            <h1 class="h1 hero-title">
              Reveal The <br>
              Beauty of Skin
            </h1>
            <p class="hero-text">
              Made using clean, non-toxic ingredients, our products are designed for everyone.
            </p>
            <p class="price">Starting at $7.99</p>
            <a href="#" class="btn btn-primary">Shop Now</a>
          </div>
        </div>
      </li>
    </ul>

  </div>
</section>
```

```
<!--
  - #COLLECTION
-->

<section class="section collection" id="collection" aria-label="collection" data-section>
  <div class="container">

    <ul class="collection-list">

      <li>
        <div class="collection-card has-before hover:shine">

          <h2 class="h2 card-title">Summer Collection</h2>

          <p class="card-text">Starting at $17.99</p>

          <a href="#" class="btn-link">
            <span class="span">Shop Now</span>

            <ion-icon name="arrow-forward" aria-hidden="true"></ion-icon>
          </a>

          <div class="has-bg-image" style="background-image:
url('./assets/images/collection-1.jpg')"></div>

        </div>
      </li>

      <li>
        <div class="collection-card has-before hover:shine">

          <h2 class="h2 card-title">What’s New?</h2>

          <p class="card-text">Get the glow</p>

          <a href="#" class="btn-link">
            <span class="span">Discover Now</span>

            <ion-icon name="arrow-forward" aria-hidden="true"></ion-icon>
          </a>

          <div class="has-bg-image" style="background-image:
url('./assets/images/collection-2.jpg')"></div>
```

```
          </div>
        </li>

        <li>
          <div class="collection-card has-before hover:shine">

            <h2 class="h2 card-title">Buy 1 Get 1</h2>

            <p class="card-text">Starting at $7.99</p>

            <a href="#" class="btn-link">
              <span class="span">Discover Now</span>

              <ion-icon name="arrow-forward" aria-hidden="true"></ion-icon>
            </a>

            <div class="has-bg-image" style="background-image:
url('./assets/images/collection-3.jpg')"></div>

          </div>
        </li>

      </ul>

    </div>
  </section>




  <!--
    - #SHOP
  -->

  <section class="section shop" id="shop" aria-label="shop" data-section>
    <div class="container">

      <div class="title-wrapper">
        <h2 class="h2 section-title">Our Bestsellers</h2>

        <a href="#" class="btn-link">
          <span class="span">Shop All Products</span>
```

```
    <ion-icon name="arrow-forward" aria-hidden="true"></ion-icon>
  </a>
</div>

<ul class="has-scrollbar">

  <li class="scrollbar-item">
    <div class="shop-card">

      <div class="card-banner img-holder" style="--width: 540; --height: 720;">
        <img src="./assets/images/product-01.jpg" width="540" height="720" loading="lazy"
          alt="Facial cleanser" class="img-cover">

        <span class="badge" aria-label="20% off">-20%</span>

        <div class="card-actions">

          <button class="action-btn" aria-label="add to cart">
            <ion-icon name="bag-handle-outline" aria-hidden="true"></ion-icon>
          </button>

          <button class="action-btn" aria-label="add to whishlist">
            <ion-icon name="star-outline" aria-hidden="true"></ion-icon>
          </button>

          <button class="action-btn" aria-label="compare">
            <ion-icon name="repeat-outline" aria-hidden="true"></ion-icon>
          </button>

        </div>
      </div>

      <div class="card-content">

        <div class="price">
          <del class="del">$39.00</del>

          <span class="span">$29.00</span>
        </div>

        <h3>
          <a href="#" class="card-title">Facial cleanser</a>
        </h3>
```

```
        <div class="card-rating">

          <div class="rating-wrapper" aria-label="5 start rating">
            <ion-icon name="star" aria-hidden="true"></ion-icon>
            <ion-icon name="star" aria-hidden="true"></ion-icon>
            <ion-icon name="star" aria-hidden="true"></ion-icon>
            <ion-icon name="star" aria-hidden="true"></ion-icon>
            <ion-icon name="star" aria-hidden="true"></ion-icon>
          </div>

          <p class="rating-text">5170 reviews</p>

        </div>

      </div>

    </div>
  </li>

  <li class="scrollbar-item">
    <div class="shop-card">

      <div class="card-banner img-holder" style="--width: 540; --height: 720;">
        <img src="./assets/images/product-02.jpg" width="540" height="720" loading="lazy"
          alt="Bio-shroom Rejuvenating Serum" class="img-cover">

        <div class="card-actions">

          <button class="action-btn" aria-label="add to cart">
            <ion-icon name="bag-handle-outline" aria-hidden="true"></ion-icon>
          </button>

          <button class="action-btn" aria-label="add to whishlist">
            <ion-icon name="star-outline" aria-hidden="true"></ion-icon>
          </button>

          <button class="action-btn" aria-label="compare">
            <ion-icon name="repeat-outline" aria-hidden="true"></ion-icon>
          </button>

        </div>
      </div>
```

```
            <div class="card-content">

              <div class="price">
                <span class="span">$29.00</span>
              </div>

              <h3>
                <a href="#" class="card-title">Bio-shroom Rejuvenating Serum</a>
              </h3>

              <div class="card-rating">

                <div class="rating-wrapper" aria-label="5 start rating">
                  <ion-icon name="star" aria-hidden="true"></ion-icon>
                  <ion-icon name="star" aria-hidden="true"></ion-icon>
                  <ion-icon name="star" aria-hidden="true"></ion-icon>
                  <ion-icon name="star" aria-hidden="true"></ion-icon>
                  <ion-icon name="star" aria-hidden="true"></ion-icon>
                </div>

                <p class="rating-text">5170 reviews</p>

              </div>

            </div>

          </div>
        </li>
```

…….. WHole code on github -https://github.com/tejasrok007/tppwaexp7

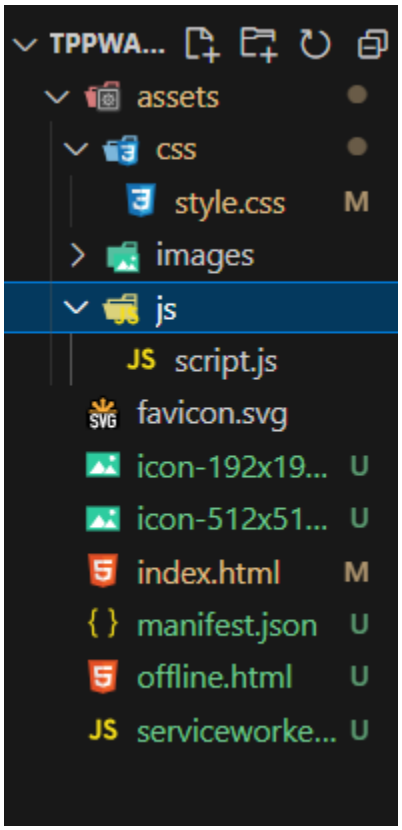
……

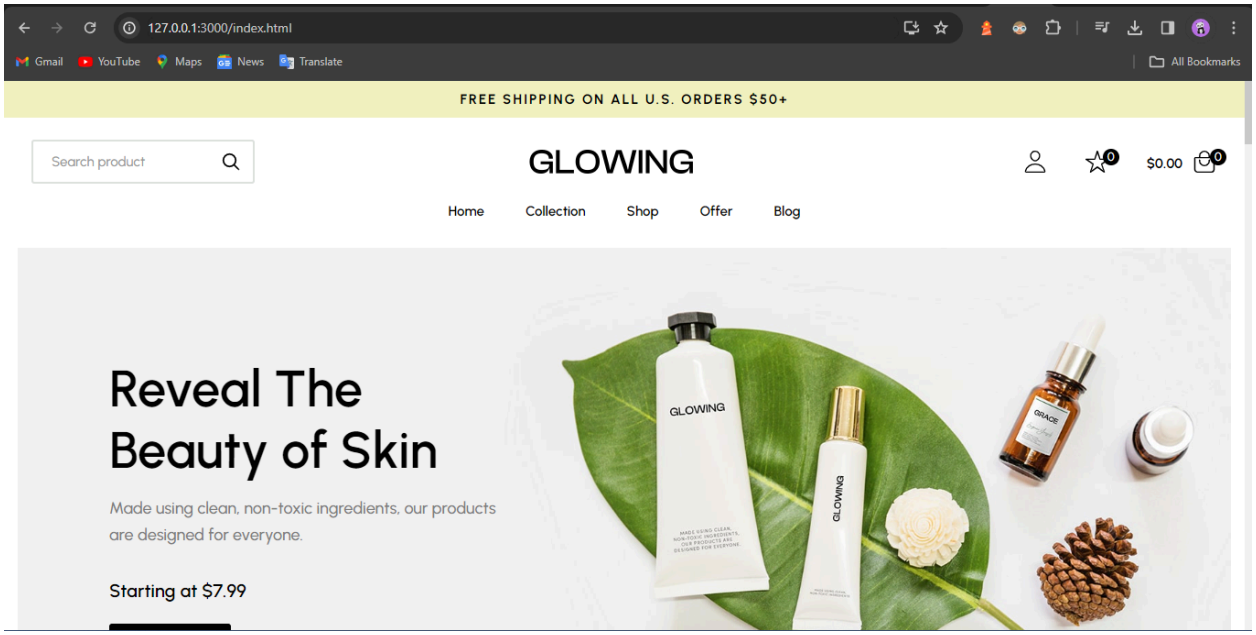
```
<script>
   // Add event listener to execute code when page loads
   window.addEventListener('load', () => {
     // Call registerSW function when page loads
     registerSW();
   });

   // Register the Service Worker
   async function registerSW() {
     // Check if browser supports Service Worker
     if ('serviceWorker' in navigator) {
       try {
         // Register the Service Worker named 'serviceworker.js'
         await navigator.serviceWorker.register('serviceworker.js');
```

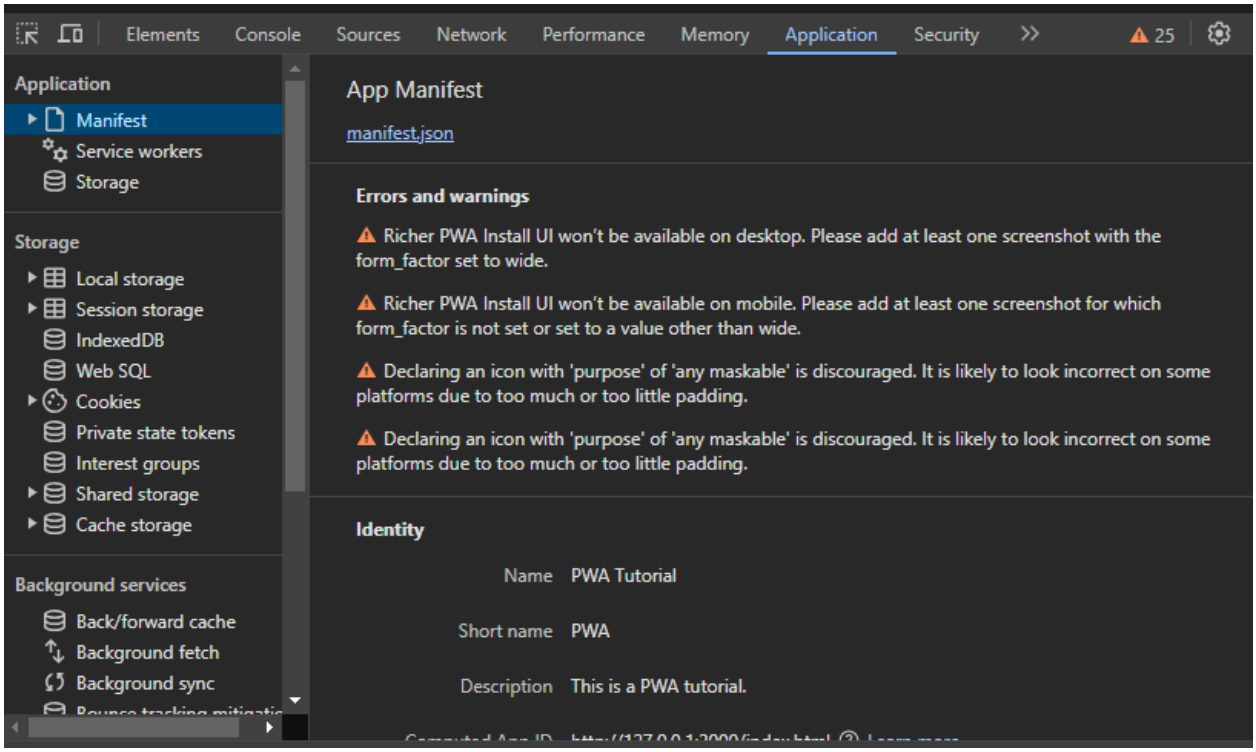
```
      }
      catch (e) {
        // Log error message if registration fails
        console.error('ServiceWorker registration failed: ', e);
      }
    }
  }
  if ('Notification' in window) {
    Notification.requestPermission().then(function (permission) {
      if (permission === 'granted') {
        console.log('Notification permission granted.');
      } else {
        console.warn('Notification permission denied.');
      }
    });
  }
</script>
```

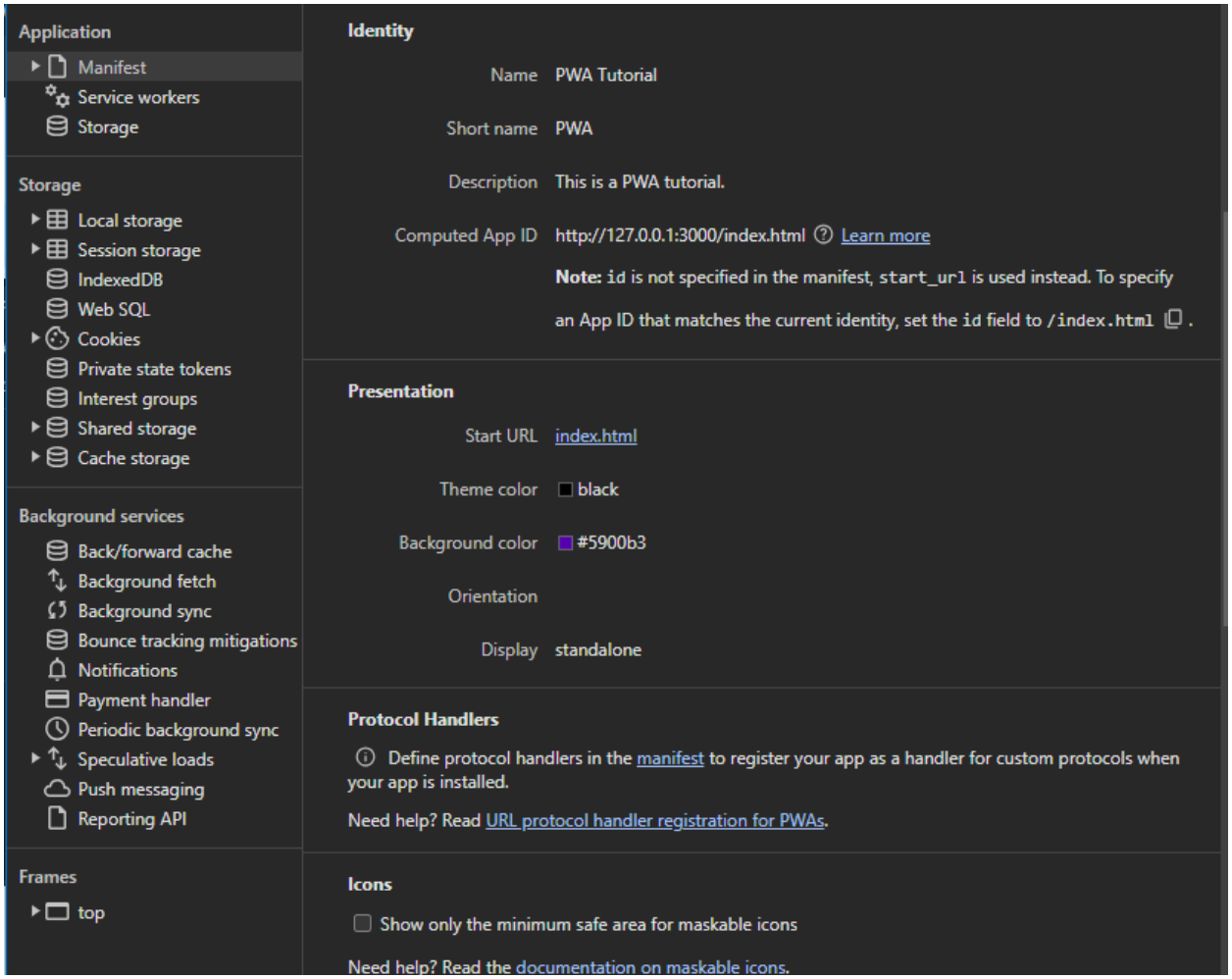
```
{
  "name":"PWA Tutorial",
  "short_name":"PWA",
  "start_url":"index.html",
  "display":"standalone",
  "background_color":"#5900b3",
  "theme_color":"black",
  "scope": ".",
  "description":"This is a PWA tutorial.",
  "icons":[
  {
  "src":"icon-192x192.png",
  "sizes":"192x192",
  "type":"image/png",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

}

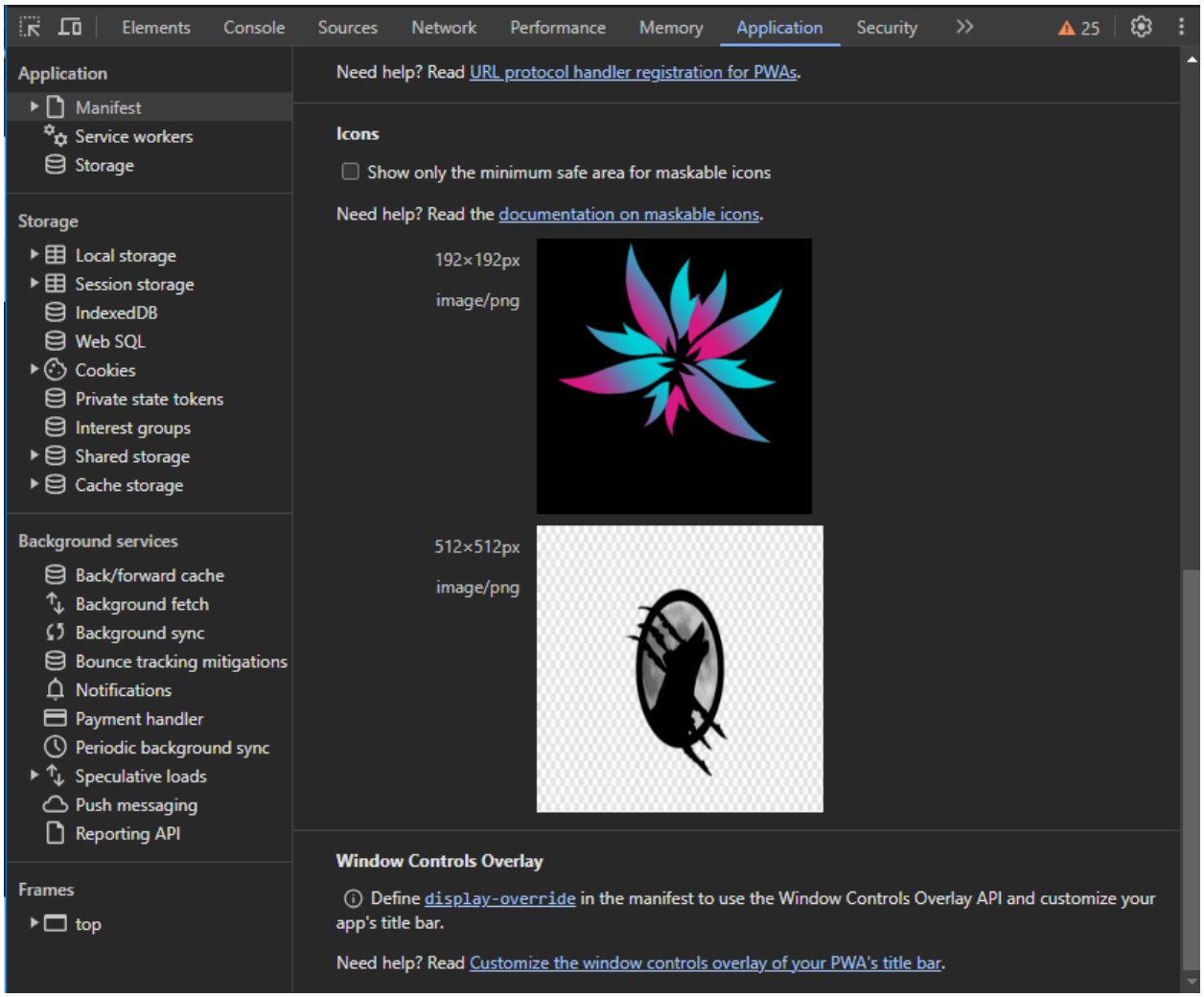

TPPWA...
assets
css
style.css M
images
js
script.js
favicon.svg
icon-192x19... U
icon-512x51... U
index.html M
manifest.json U
offline.html U
serviceworke... U

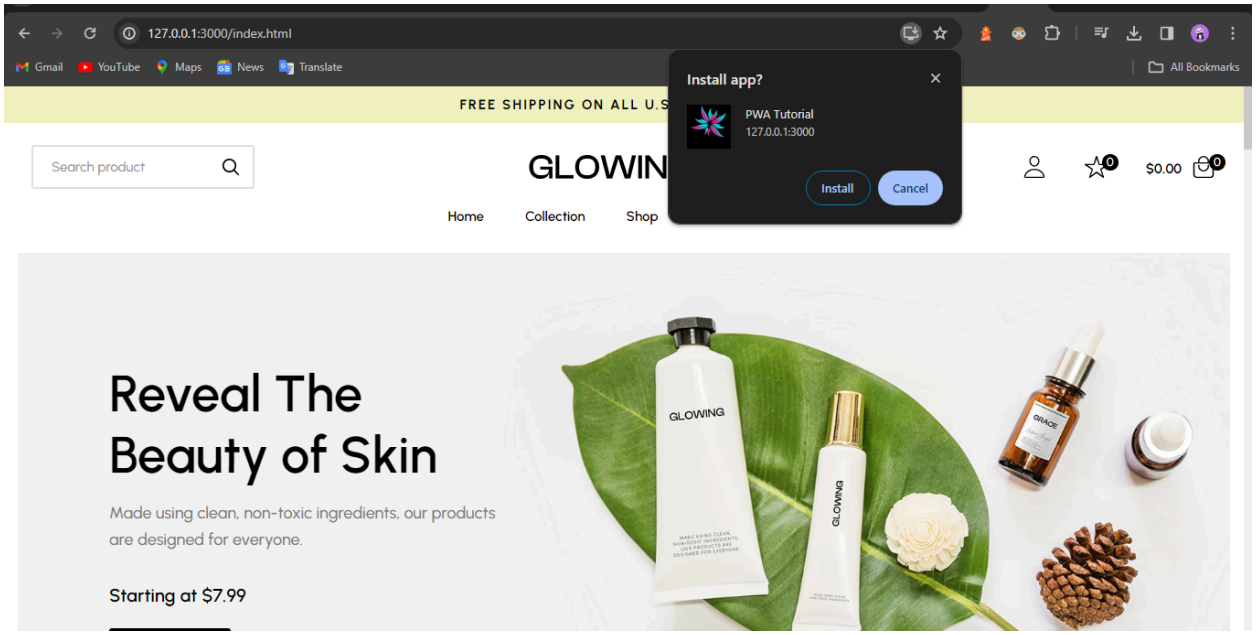

127.0.0.1:3000/index.html
Gmail
YouTube
Maps
News
Translate
All Bookmarks
FREE SHIPPING ON ALL U.S. ORDERS $50+
Search product
GLOWING
$0.00
Home
Collection
Shop
Offer
Blog
Reveal The Beauty of Skin
Made using clean, non-toxic ingredients, our products are designed for everyone.
Starting at $7.99
GLOWING
GLOWING
GRACE


Elements
Console
Sources
Network
Performance
Memory
Application
Security
25
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
IndexedDB
Web SQL
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Background services
Back/forward cache
Background fetch
Background sync
App Manifest
manifest.json
Errors and warnings
Richer PWA Install UI won't be available on desktop. Please add at least one screenshot with the form_factor set to wide.
Richer PWA Install UI won't be available on mobile. Please add at least one screenshot for which form_factor is not set or set to a value other than wide.
Declaring an icon with 'purpose' of 'any maskable' is discouraged. It is likely to look incorrect on some platforms due to too much or too little padding.
Declaring an icon with 'purpose' of 'any maskable' is discouraged. It is likely to look incorrect on some platforms due to too much or too little padding.
Identity
Name PWA Tutorial
Short name PWA
Description This is a PWA tutorial.

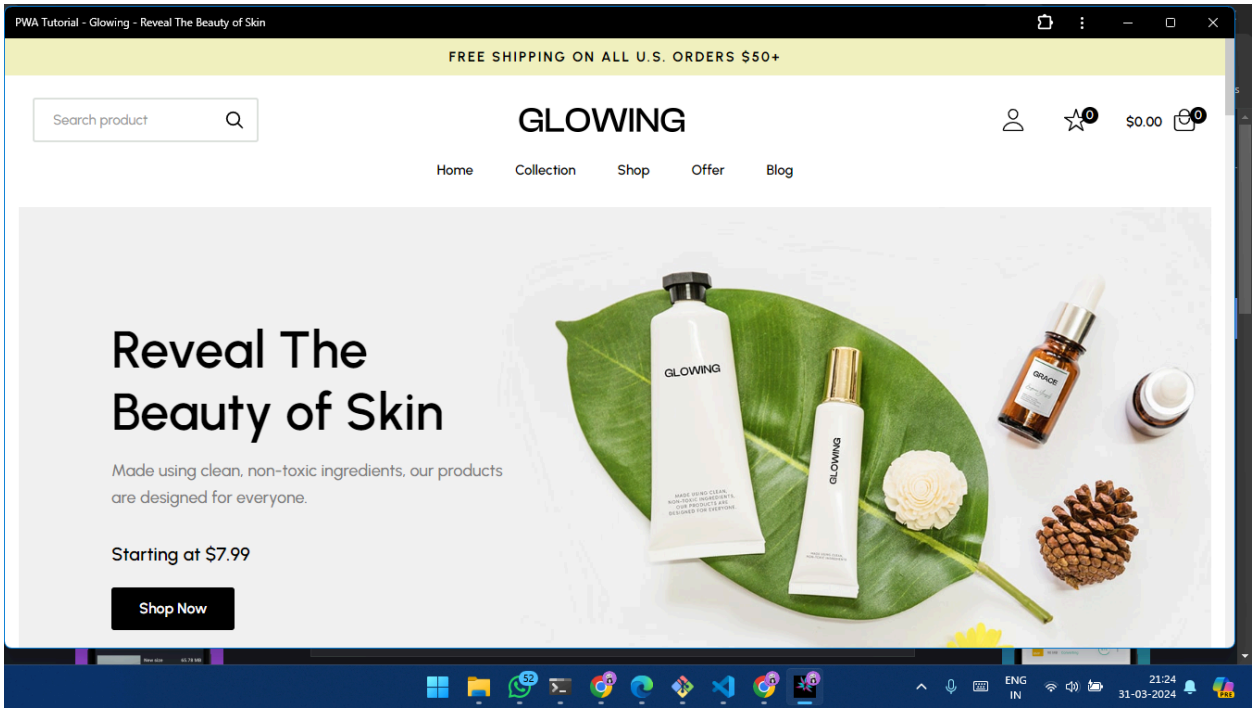

Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
IndexedDB
Web SQL
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigations
Notifications
Payment handler
Periodic background sync
Speculative loads
Push messaging
Reporting API
Frames
top
Identity
Name PWA Tutorial
Short name PWA
Description This is a PWA tutorial.
Computed App ID http://127.0.0.1:3000/index.html Learn more
Note: id is not specified in the manifest, start_url is used instead. To specify an App ID that matches the current identity, set the id field to /index.html .
Presentation
Start URL index.html
Theme color black
Background color #5900b3
Orientation
Display standalone
Protocol Handlers
Define protocol handlers in the manifest to register your app as a handler for custom protocols when your app is installed.
Need help? Read URL protocol handler registration for PWAs.
Icons
Show only the minimum safe area for maskable icons
Need help? Read the documentation on maskable icons.

Project Title:Lunching and Glowing MAD /PWA Lab **Roll no.- 50**





Project Title:Lunching and Glowing MAD /PWA Lab **Roll no.- 50**



Conclusion -

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

MAD & PWA Lab Journal

Experiment No.	08
Experiment Title.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA
Roll no.	50
Name	Tejas Shailendra Rokade
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Name - Tejas Shailendra Rokade
Div - D15A Roll no - 50
Batch - C

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 is a script that operates independently in the background of a browser, acting as a programmable network proxy. It enables developers to control how network requests from a web page are handled. It's crucial to note that Service Workers only function over HTTPS due to security concerns, as they have the capability to intercept network requests and modify responses.

Here's a summary of what you can do and can't do with Service Workers:

What You Can Do:

1. Dominate Network Traffic: You can manage all network traffic of the page and manipulate requests and responses as needed. This includes altering responses for different types of requests.

2. Cache Content: Service Workers allow caching of request/response pairs using the Cache API. This enables offline access to cached content, improving performance and user experience.

3. Manage Push Notifications: You can handle push notifications with Service Workers, allowing you to display messages to users even when the internet connection is broken.

4. Background Sync: Service Workers facilitate background processes, enabling tasks to continue despite interruptions in internet connectivity.

What You Can't Do:

1. Access the Window: Service Workers cannot access the window or manipulate DOM elements directly. Communication with the window is possible through postMessage.

2. Work on Port 80: Service Workers only function on HTTPS, but during development, they can operate on localhost.

Registration:
To install a Service Worker, it must be registered in the main JavaScript code. This registration informs the browser where the Service Worker is located and initiates the installation process in the background.

Installation:
Once registered, the browser attempts to install the Service Worker if it's considered new or updated. This triggers an install event, during which tasks such as precaching parts of the web app can be performed to enhance subsequent loading times.

Overall, Service Workers offer powerful capabilities for enhancing web applications, particularly in terms of offline functionality, network traffic management, and push notifications. However, developers must be mindful of security considerations and the limitations imposed by the Service Worker environment.

Code -
Serviceworker.js

```
const CACHE_NAME = 'my-ecommerce-app-cache-v1';
const urlsToCache = [
  '/',
  'index.html',
  'assets/js/script.js',
  'assets/css/style.css',
  'serviceworker.js',
  'manifest.json',
  'offline.html'
  // Add more files to cache as needed
];

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

```
self.addEventListener('activate', function(event) {
  // Perform activation steps
  event.waitUntil(
    caches.keys().then(function(cacheNames) {
      return Promise.all(
        cacheNames.map(function(cacheName) {
          if (cacheName !== CACHE_NAME) {
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
});
self.addEventListener('activate', async () => {
  if (Notification.permission !== 'granted') {
    try {
      const permission = await Notification.requestPermission();
      if (permission === 'granted') {
        console.log('Notification permission granted.');
      } else {
        console.warn('Notification permission denied.');
      }
    } catch (error) {
      console.error('Failed to request notification permission:', error);
    }
  }
});
```

manifest.js

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

```
    "purpose":"any maskable"
    },
    {
    "src":"icon-512x512.png",
    "sizes":"512x512",
    "type":"image/png",
    "purpose":"any maskable"
    }
]
}
```

Offline.html

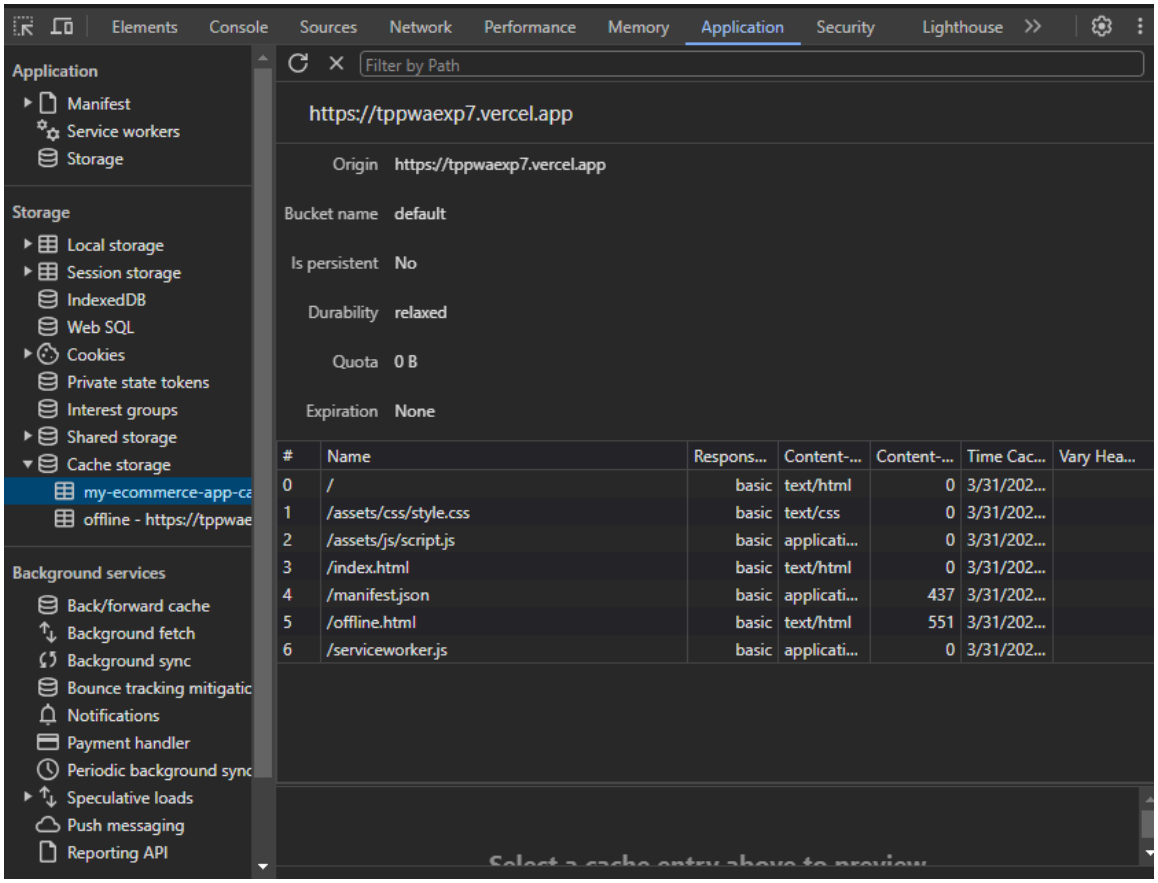
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Offline</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f0f0f0;
      color: #333;
      text-align: center;
      padding: 50px;
    }
    h1 {
      margin-bottom: 20px;
    }
  </style>
</head>
<body>
  <h1>Oops! You're offline.</h1>
  <p>It seems like you're currently offline. Please check your internet connection and try again
later.</p>
</body>
</html>
```

script.js-

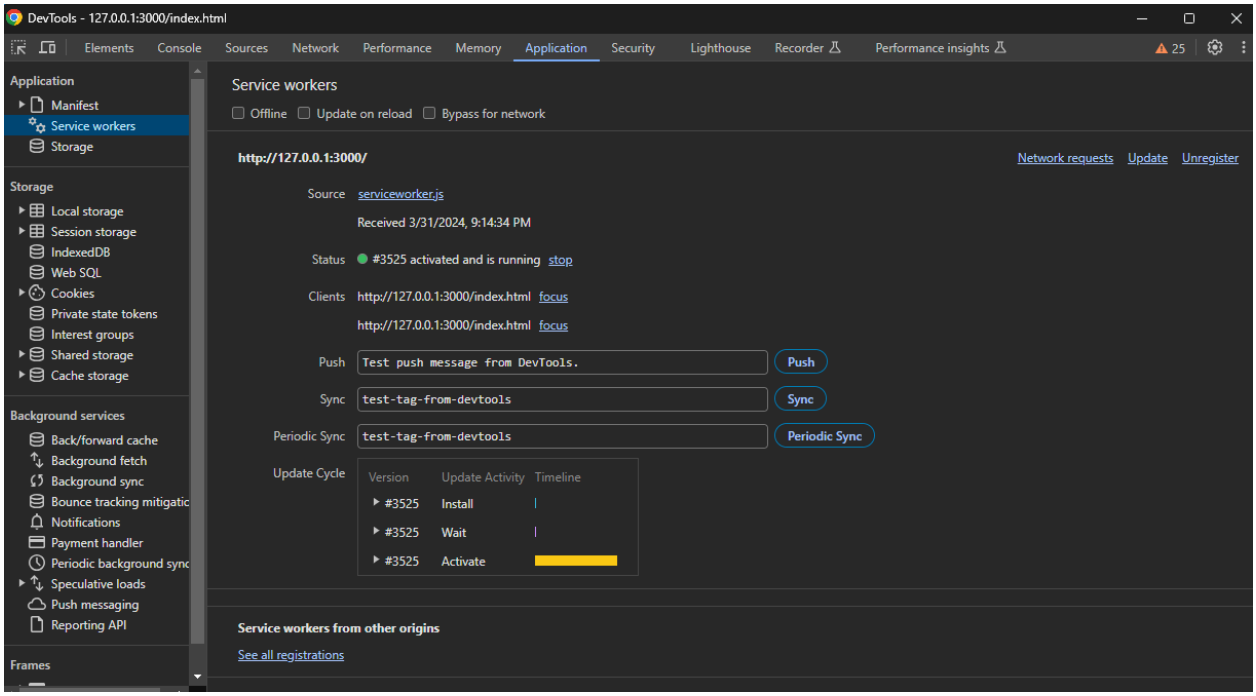
```
'use strict';

const addEventOnElem = function (elem, type, callback) {
  if (elem.length > 1) {
    for (let i = 0; i < elem.length; i++) {
```

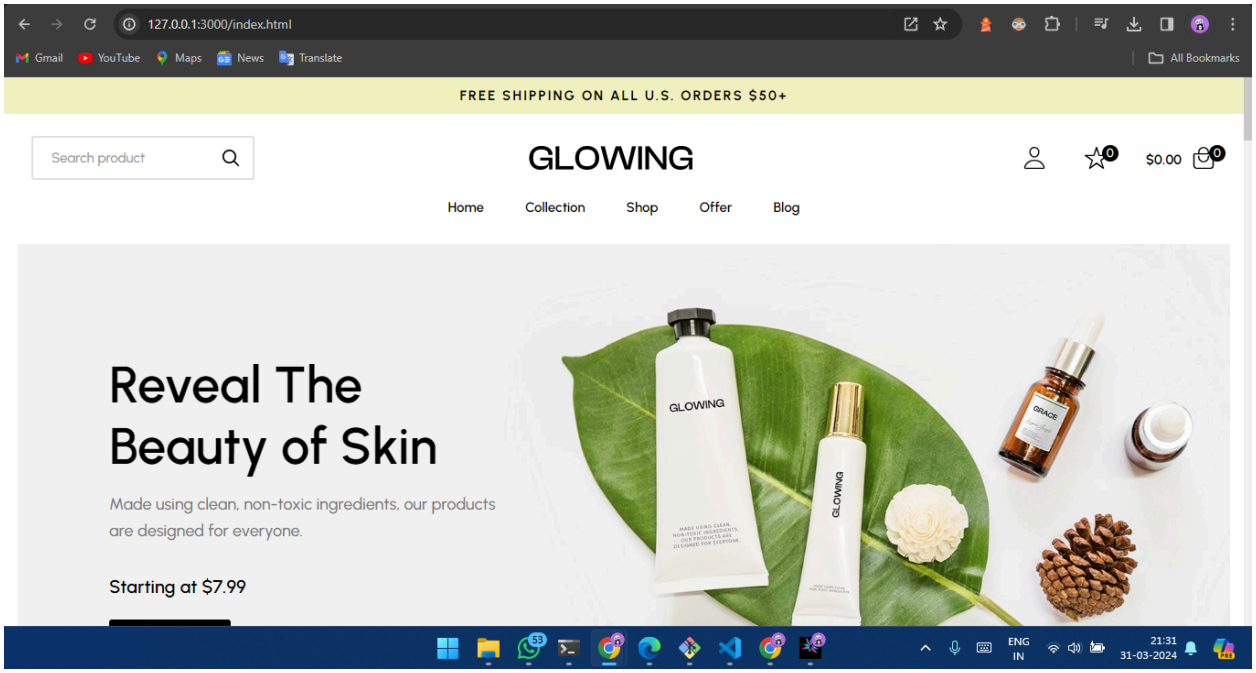
```
      elem[i].addEventListener(type, callback);
    }
  } else {
    elem.addEventListener(type, callback);
  }
}
const navTogglers = document.querySelectorAll("[data-nav-toggler]");
const navbar = document.querySelector("[data-navbar]");
const navbarLinks = document.querySelectorAll("[data-nav-link]");
const overlay = document.querySelector("[data-overlay]");

const toggleNavbar = function () {
  navbar.classList.toggle("active");
  overlay.classList.toggle("active");
}

addEventOnElem(navTogglers, "click", toggleNavbar);

const closeNavbar = function () {
  navbar.classList.remove("active");
  overlay.classList.remove("active");
}

addEventOnElem(navbarLinks, "click", closeNavbar);
const header = document.querySelector("[data-header]");
const backTopBtn = document.querySelector("[data-back-top-btn]");

const headerActive = function () {
  if (window.scrollY > 150) {
    header.classList.add("active");
    backTopBtn.classList.add("active");
  } else {
    header.classList.remove("active");
    backTopBtn.classList.remove("active");
  }
}

addEventOnElem(window, "scroll", headerActive);

let lastScrolledPos = 0;

const headerSticky = function () {
  if (lastScrolledPos >= window.scrollY) {
    header.classList.remove("header-hide");
```

```
  } else {
    header.classList.add("header-hide");
  }

  lastScrolledPos = window.scrollY;
}
addEventOnElem(window, "scroll", headerSticky);
const sections = document.querySelectorAll("[data-section]");
const scrollReveal = function () {
  for (let i = 0; i < sections.length; i++) {
    if (sections[i].getBoundingClientRect().top < window.innerHeight / 2) {
      sections[i].classList.add("active");
    }
  }
}
scrollReveal();
addEventOnElem(window, "scroll", scrollReveal);
```



Project Title:Lunching and Glowing MAD /PWA Lab **Roll no.- 50**





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.	50
Name	Tejas Shailendra Rokade
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Name - Tejas Shailendra Rokade
Div - D15A Roll no - 50
Batch - C

Experiment - 9

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

Theory -
Service Worker is a powerful script that operates in the background of a browser independently, similar to a proxy working on the user's side. It allows developers to track network traffic, manage push notifications, and develop "offline-first" web applications using the Cache API. Here are some important aspects to note about Service Workers:

- Programmable Network Proxy: Service Worker acts as a programmable network proxy, giving developers control over how network requests from their web page are handled.

- HTTPS Requirement: Service Workers only function over HTTPS due to security reasons. They can intercept network requests and modify responses, so running them over insecure connections could pose significant security risks.

- Idle State and Restart: When not in use, a Service Worker becomes idle and restarts when needed next. Developers can't rely on a global state persisting between events. IndexedDB databases can be used to persist and reuse information across restarts.

- Fetch Event: This event allows tracking and managing of page network traffic. Developers can implement strategies such as "cache first" and "network first" approaches. For example:
  - CacheFirst: If the requested resource is cached, return the cached response; otherwise, request a new response from the network.
  - NetworkFirst: Attempt to get an updated response from the network. If successful, cache and return the new response. If unsuccessful, check if the request is cached and return it if available.

- Sync Event: Background Sync API delays a process until the internet connection is stable. For example, in an email client application, if the internet connection is lost while composing an email, Background Sync can ensure the email is sent once the connection is restored.

- Push Event: Handles push notifications received from the server. Developers can apply various actions based on received data. For instance, displaying a notification to the user. The `Notification.requestPermission();` line is necessary to show notifications to the user.

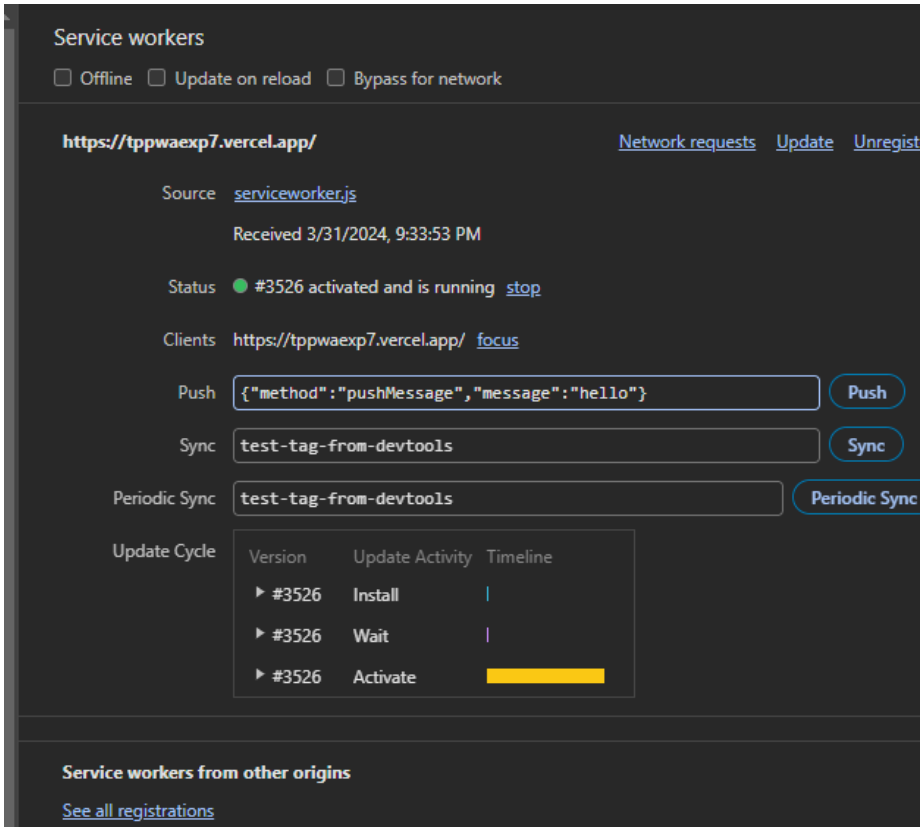
Service Workers offer developers extensive capabilities for enhancing web applications, including offline functionality, network traffic management, and push notifications. However, they require careful implementation due to security considerations and their asynchronous nature.

Code -
Service worker.js

```
const CACHE_NAME = 'my-ecommerce-app-cache-v1';
const urlsToCache = [
  '/',
  'index.html',
  'assets/js/script.js',
  'assets/css/style.css',
  'serviceworker.js',
  'manifest.json',
  'offline.html'
  // Add more files to cache as needed
];

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

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

```
    })
  );
});

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

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

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

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

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

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

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

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


Service workers
Offline
Update on reload
Bypass for network
https://tppwaexp7.vercel.app/
Network requests
Update
Unregist
Source
serviceworker.js
Received 3/31/2024, 9:33:53 PM
Status
#3526 activated and is running
stop
Clients
https://tppwaexp7.vercel.app/
focus
Push
{"method":"pushMessage","message":"hello"}
Push
Sync
test-tag-from-devtools
Sync
Periodic Sync
test-tag-from-devtools
Periodic Sync
Update Cycle
Version
Update Activity
Timeline
#3526
Install
#3526
Wait
#3526
Activate
Service workers from other origins
See all registrations

Project Title:Lunching and Glowing MAD /PWA Lab **Roll no.- 50**

```
Elements  Console  Sources  Network  Performance  Memory  Application  Security  Lighthouse  Recorder  Performance insights   25
top   Filter   Default levels   No Issues   3 hidden
Fetch from cache successful!                                                    serviceworker.js:44
43 Fetch successful!                                                            serviceworker.js:47
Fetch from cache successful!                                                    serviceworker.js:44
50 Fetch successful!                                                            serviceworker.js:47
Fetch from cache successful!                                                    serviceworker.js:44
53 Fetch successful!                                                            serviceworker.js:47
Fetch from cache successful!                                                    serviceworker.js:44
49 Fetch successful!                                                            serviceworker.js:47
Fetch from cache successful!                                                    serviceworker.js:44
41 Fetch successful!                                                            serviceworker.js:47
Fetch from cache successful!                                                    serviceworker.js:44
65 Fetch successful!                                                            serviceworker.js:47
Fetch from cache successful!                                                    serviceworker.js:44
41 Fetch successful!                                                            serviceworker.js:47
Fetch from cache successful!                                                    serviceworker.js:44
56 Fetch successful!                                                            serviceworker.js:47
Fetch from cache successful!                                                    serviceworker.js:44
5 Fetch successful!                                                             serviceworker.js:47
▲ ▸The FetchEvent for "http://127.0.0.1:3000/icon-192x192." resulted in a network error response: an object that was not a Response was passed to respondWith().
47 Fetch successful!                                                            serviceworker.js:47
Fetch from cache successful!                                                    serviceworker.js:44
779 Fetch successful!                                                           serviceworker.js:47
Notification permission granted.                                                index.html:58
86 Fetch successful!                                                            serviceworker.js:47
12 Push notification sent                                                       serviceworker.js:64
›
```

```
Service workers
☐ Offline  ☐ Update on reload  ☐ Bypass for network

http://127.0.0.1:3000/                    Network requests  Update  Unregister

Source        serviceworker.js
              Received 3/31/2024, 9:14:34 PM
Status        ● #3525 activated and is running  stop
Push          {"method":"pushMessage","message":"hello"}   [Push]
Sync          syncMessage                                  [Sync]
Periodic Sync test-tag-from-devtools                       [Periodic Sync]
Update Cycle
  Version    Update Activity  Timeline
  ▸ #3525    Install          |
  ▸ #3525    Wait             |
  ▸ #3525    Activate         ▬

Service workers from other origins
See all registrations
```

```
47 Fetch successful!                                                            serviceworker.js:47
Fetch from cache successful!                                                    serviceworker.js:44
779 Fetch successful!                                                           serviceworker.js:47
Notification permission granted.                                                index.html:58
86 Fetch successful!                                                            serviceworker.js:47
12 Push notification sent                                                       serviceworker.js:64
Sync successful!                                                                serviceworker.js:54
›
```

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.	50
Name	Tejas Shailendra Rokade
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Name - Tejas Shailendra Rokade
Div - D15A Roll no - 50
Batch - C

Experiment - 10

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

Theory -
GitHub Pages and Firebase are both excellent platforms for hosting websites and web applications, each with its own set of features, advantages, and limitations. Here's a comparison of the two:

GitHub Pages:
Pros:
1. Familiar interface for GitHub users.
2. Easy setup process.
3. Supports Jekyll for blogging out of the box.
4. Allows custom domains.

Cons:
1. Website code is public unless using a private repository.
2. No HTTPS support for custom domains (currently).
3. Limited plugin support for Jekyll.

Firebase:
Pros:
1. Hosted by Google, ensuring reliability and scalability.
2. Offers authentication, cloud messaging, and other useful services.
3. Provides a real-time database and blob store.
4. HTTPS support with free SSL certificate provisioned for custom domains.

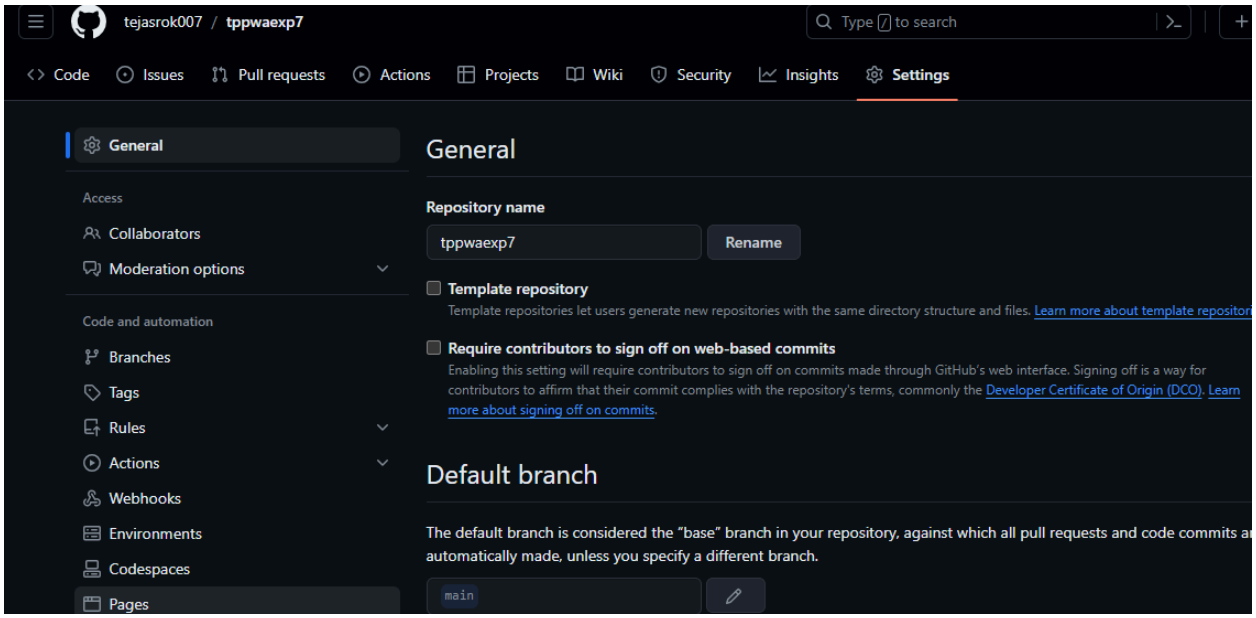
Cons:
1. Limited to 10 GB of data transfer per month.
2. Command-line interface only.

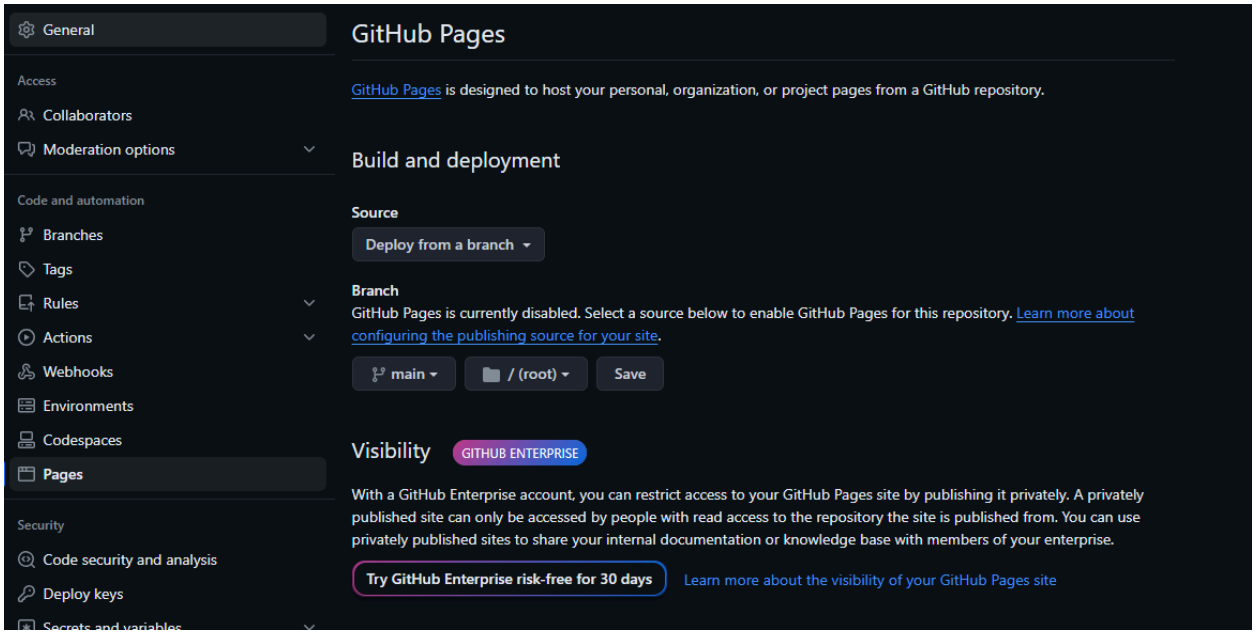
Reasons to Choose GitHub Pages:
1. Free to use and integrates seamlessly with GitHub repositories.
2. Easy setup process, particularly for static websites.
3. Great for hosting blogs with Jekyll.
4. Supports custom domains.

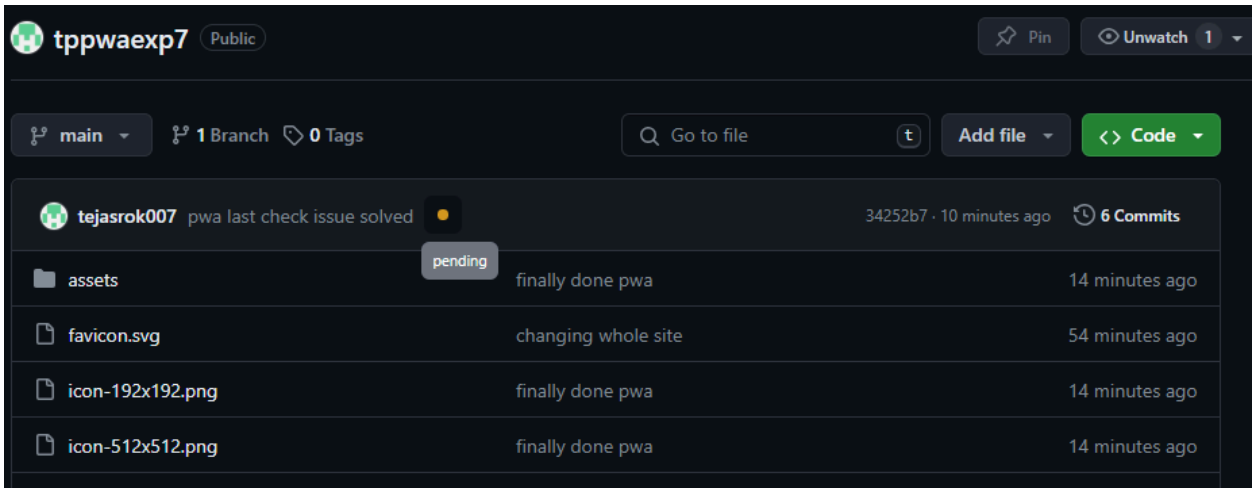
Reasons to Choose Firebase:
1. Real-time backend capabilities make it suitable for dynamic and collaborative applications.
2. Provides a broader range of services beyond hosting, such as authentication and cloud messaging.
3. HTTPS support and free SSL certificates for custom domains.
4. Hosted by Google, ensuring reliability and scalability.

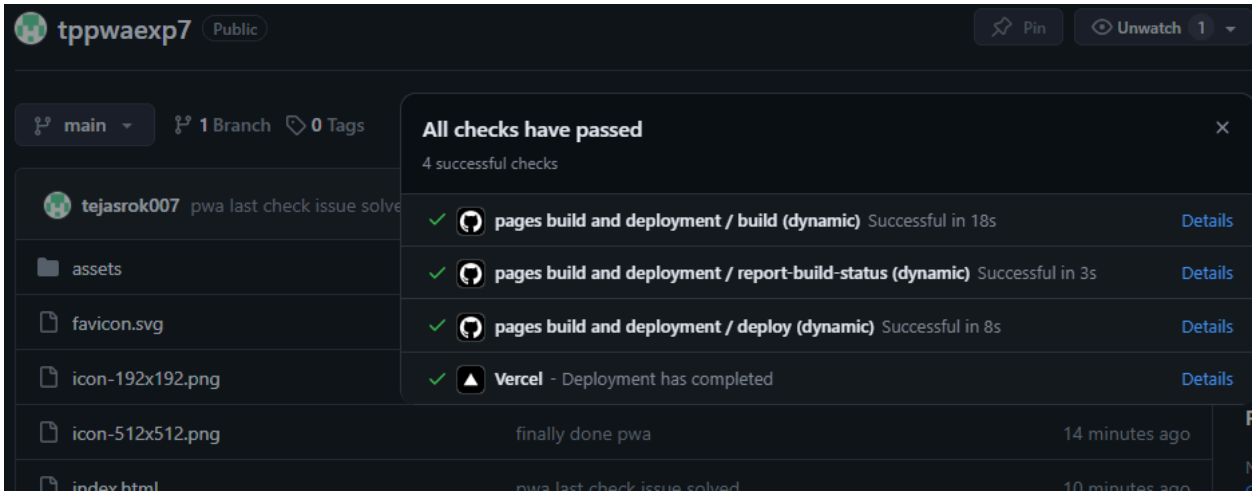
In summary, GitHub Pages is a great choice for static websites, particularly for users already familiar with GitHub, while Firebase offers a more comprehensive platform for building dynamic and real-time applications with additional backend services. Your choice should depend on the specific needs and requirements of your project.

Screenshot -

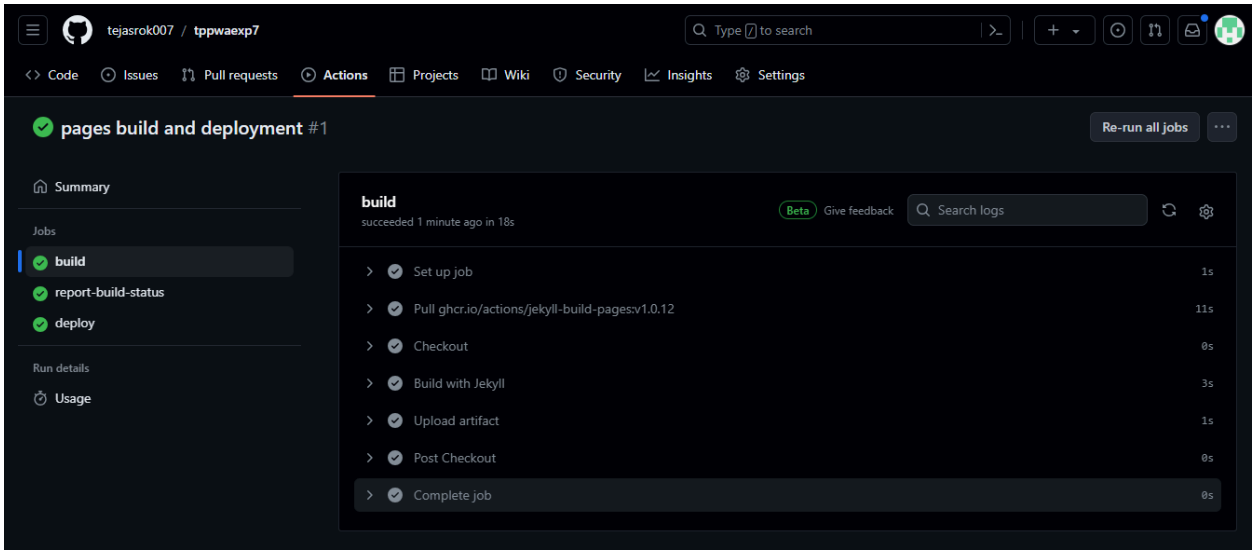


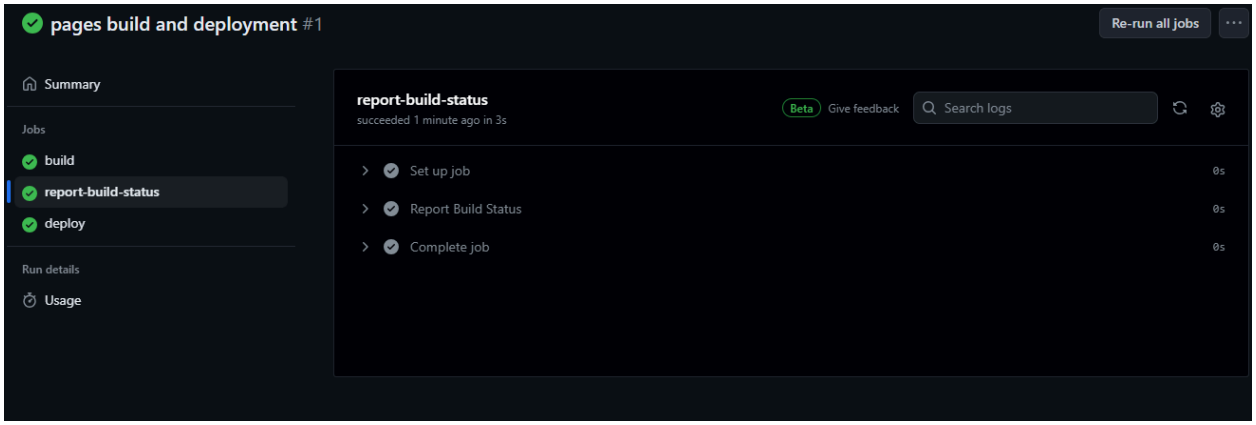

General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
Security
Code security and analysis
Deploy keys
GitHub Pages
GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.
Build and deployment
Source
Deploy from a branch
Branch
GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. Learn more about configuring the publishing source for your site.
main
/ (root)
Save
Visibility
GITHUB ENTERPRISE
With a GitHub Enterprise account, you can restrict access to your GitHub Pages site by publishing it privately. A privately published site can only be accessed by people with read access to the repository the site is published from. You can use privately published sites to share your internal documentation or knowledge base with members of your enterprise.
Try GitHub Enterprise risk-free for 30 days
Learn more about the visibility of your GitHub Pages site

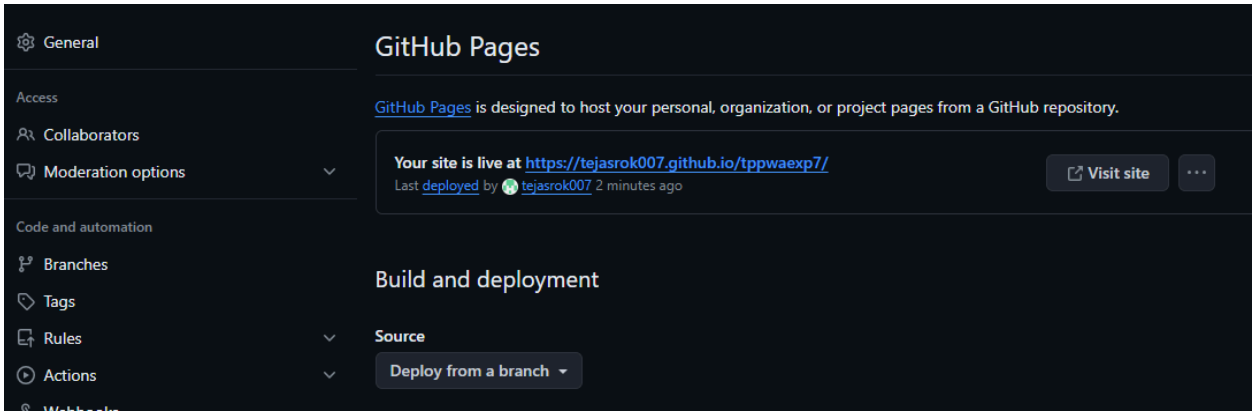

tppwaexp7
Public
Pin
Unwatch 1
main
1 Branch
0 Tags
Go to file
t
Add file
Code
tejasrok007 pwa last check issue solved
pending
34252b7 · 10 minutes ago
6 Commits
assets
finally done pwa
14 minutes ago
favicon.svg
changing whole site
54 minutes ago
icon-192x192.png
finally done pwa
14 minutes ago
icon-512x512.png
finally done pwa
14 minutes ago

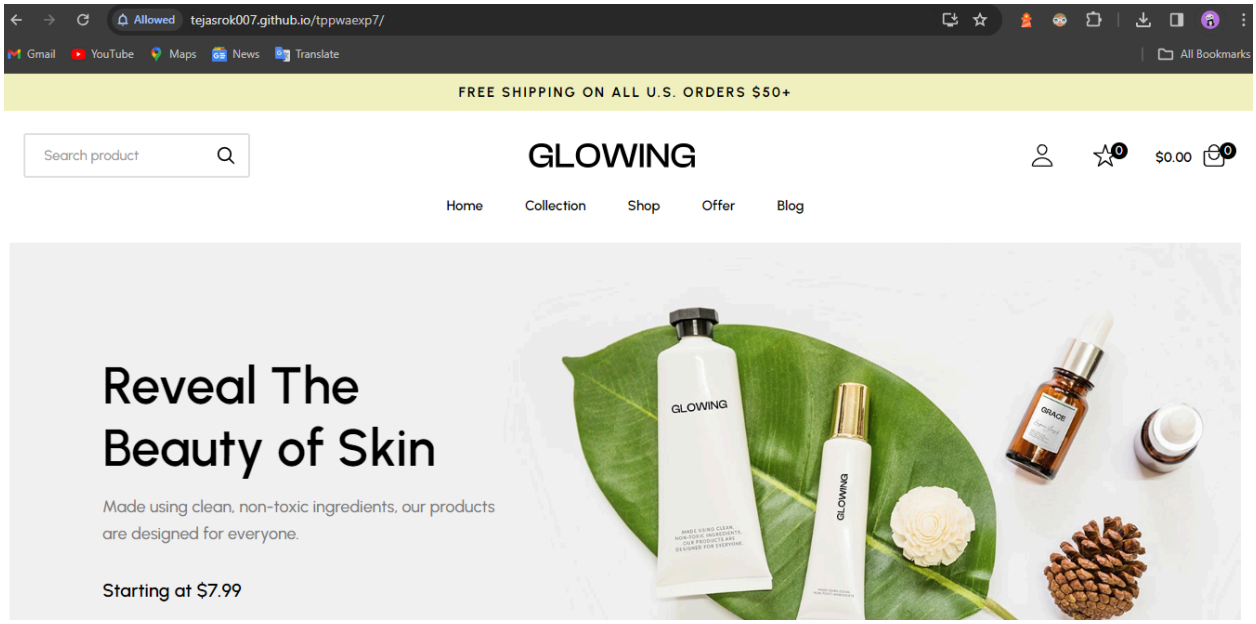

tppwaexp7
Public
Pin
Unwatch 1
main
1 Branch
0 Tags
All checks have passed
4 successful checks
tejasrok007 pwa last check issue solve
pages build and deployment / build (dynamic) Successful in 18s
Details
pages build and deployment / report-build-status (dynamic) Successful in 3s
Details
pages build and deployment / deploy (dynamic) Successful in 8s
Details
Vercel - Deployment has completed
Details
assets
favicon.svg
icon-192x192.png
icon-512x512.png
finally done pwa
14 minutes ago

Project Title:Lunching and Glowing MAD /PWA Lab **Roll no.- 50**









Github - https://tejasrok007.github.io/tppwaexp7/

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.	50
Name	Tejas Shailendra Rokade
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	15

Name - Tejas Shailendra Rokade
Div - D15A Roll no - 50
Batch - C

Experiment - 11

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

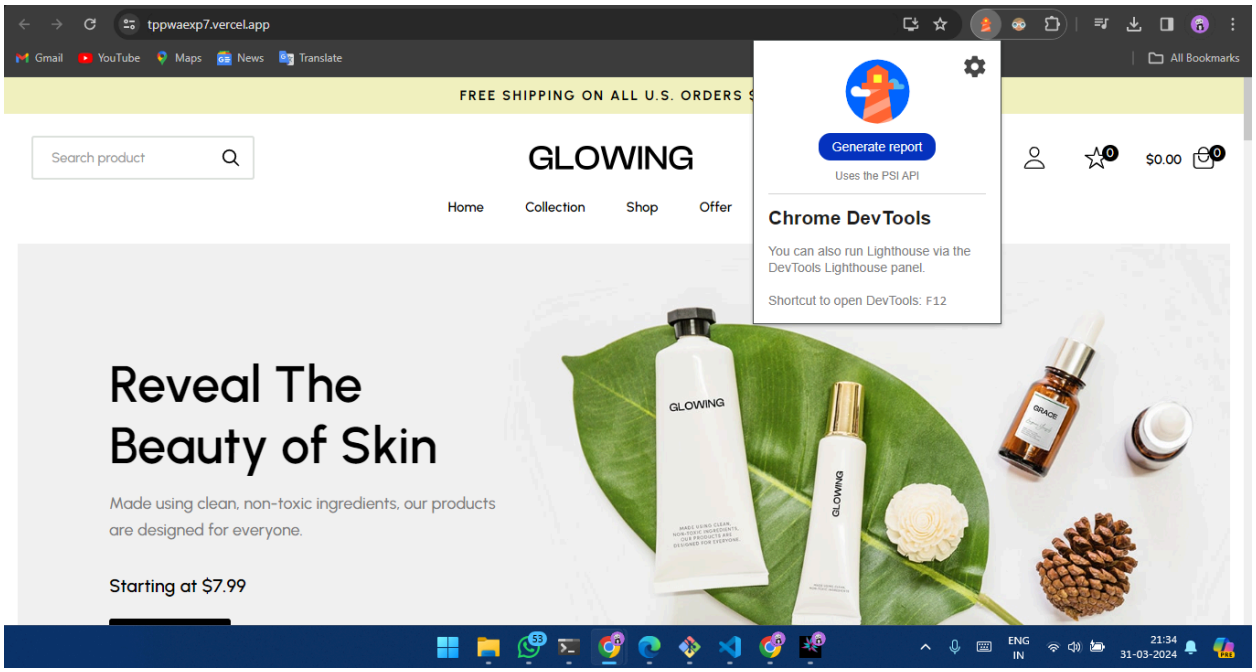
Theory -
Google Lighthouse is a powerful open-source tool designed to improve the quality of web pages by providing comprehensive audits across various metrics. Here's a breakdown of its features:

1. Performance:
- Provides valuable metrics to optimize site performance, including opportunities for improvement.
- Offers insights into render-blocking resources such as stylesheets, scripts, and HTML imports.
- Combines data from both real-world usage (Field Data) and simulated conditions (Lab Data) for a comprehensive analysis.
2. Accessibility:
- Highlights potential improvements to enhance accessibility and user experience.
- Ensures that websites are easily navigable and usable for all users, including those with disabilities.
- Improving accessibility can also positively impact search engine rankings.
3. Best Practices:
- Offers recommendations to improve overall site performance and user experience, particularly for mobile sites.
- Provides suggestions for adhering to best practices in web development and design.
4. SEO:
- Evaluates a website's optimization for search engine rankings.
- Provides essential tools to analyze and improve a page's visibility in search engine results.
- Covers fundamental aspects of SEO, though not all factors may be considered.
5. Progressive Web Applications (PWAs):
- Assess whether a website meets the criteria for being classified as a Progressive Web App.
- Focuses on key factors such as service worker registration, enabling push notifications, and offline capabilities.
- Helps developers enhance the user experience and engagement with modern web app features.

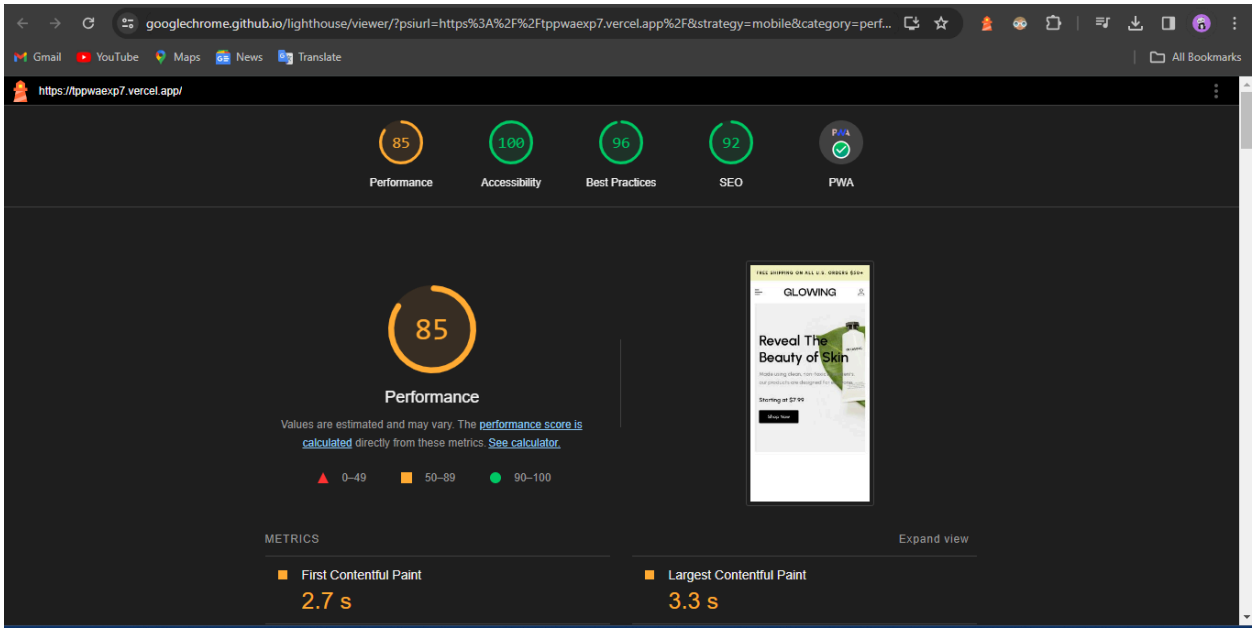
Overall, Google Lighthouse offers a comprehensive suite of metrics and recommendations to help developers optimize their websites for performance, accessibility, SEO, and PWA features. By addressing the issues identified in the Lighthouse report, developers can improve their site's user experience, search engine visibility, and overall quality.

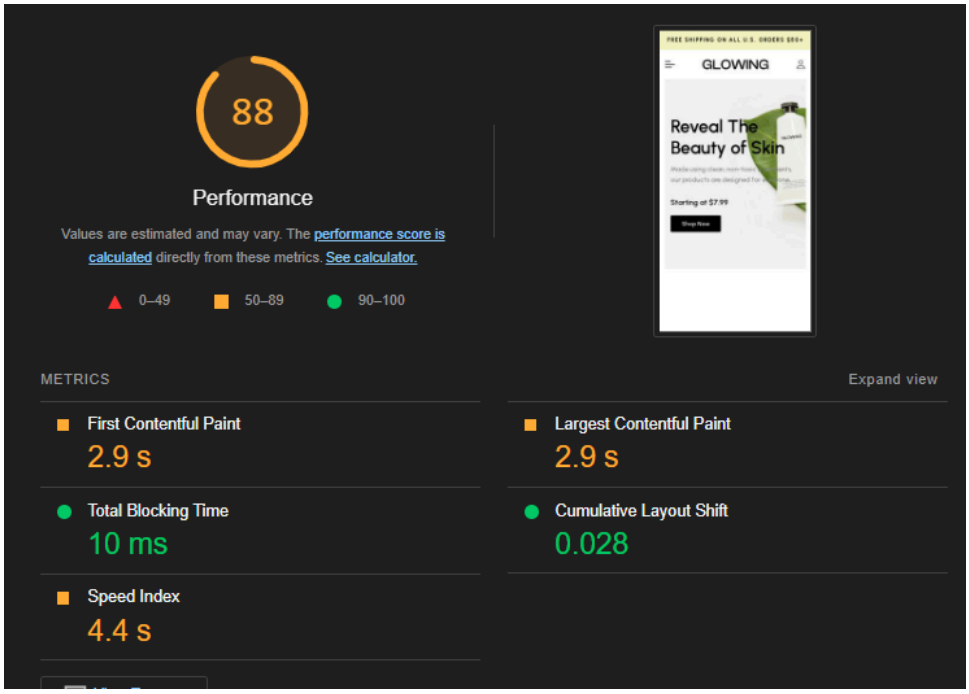
```
{
   "name":"PWA Tutorial",
   "short_name":"PWA",
   "start_url":"index.html",
   "display":"standalone",
   "background_color":"#5900b3",
   "theme_color":"black",
   "scope": ".",
   "description":"This is a PWA tutorial.",
   "icons":[
   {
   "src":"icon-192x192.png",
   "sizes":"192x192",
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


ScreenShot -



Project Title:Lunching and Glowing MAD /PWA Lab **Roll no.- 50**





Project Title:Lunching and Glowing MAD /PWA Lab **Roll no.- 50**

DIAGNOSTICS

- Eliminate render-blocking resources — Potential savings of 840 ms
- Largest Contentful Paint element — 2,870 ms
- Defer offscreen images — Potential savings of 36 KiB
- Serve images in next-gen formats — Potential savings of 196 KiB
- Serve static assets with an efficient cache policy — 23 resources found
- Avoid an excessive DOM size — 1,071 elements
- Initial server response time was short — Root document took 280 ms
- Avoids enormous network payloads — Total size was 597 KiB
- Avoid chaining critical requests — 4 chains found
- JavaScript execution time — 0.2 s
- Minimizes main-thread work — 0.8 s
- Minimize third-party usage — Third-party code blocked the main thread for 0 ms

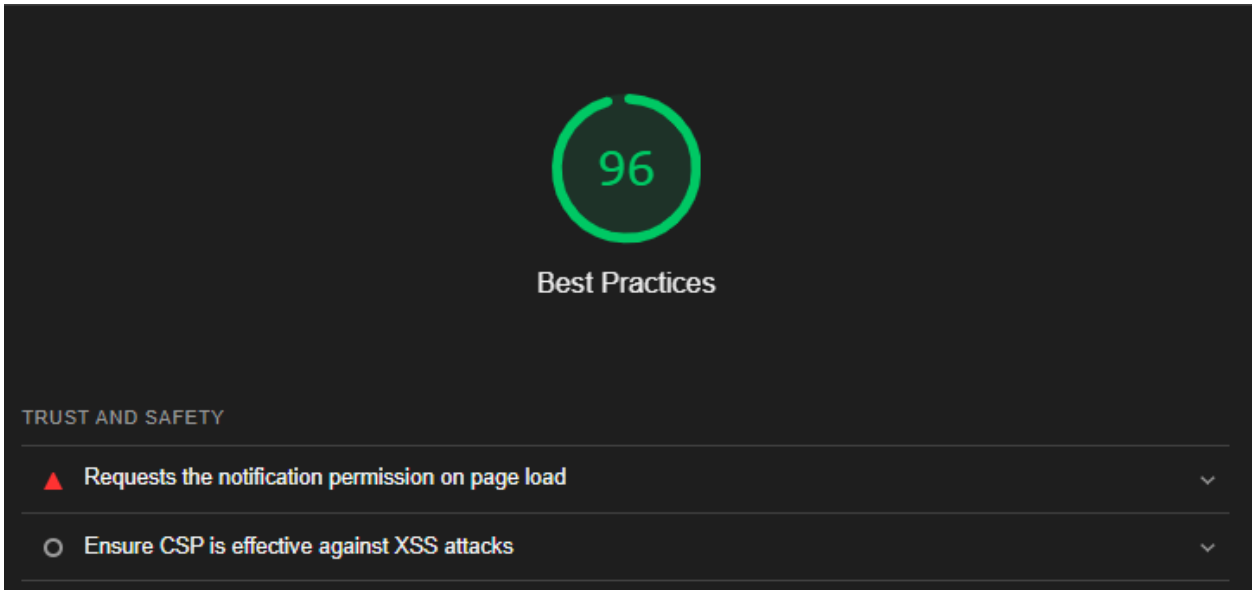
100

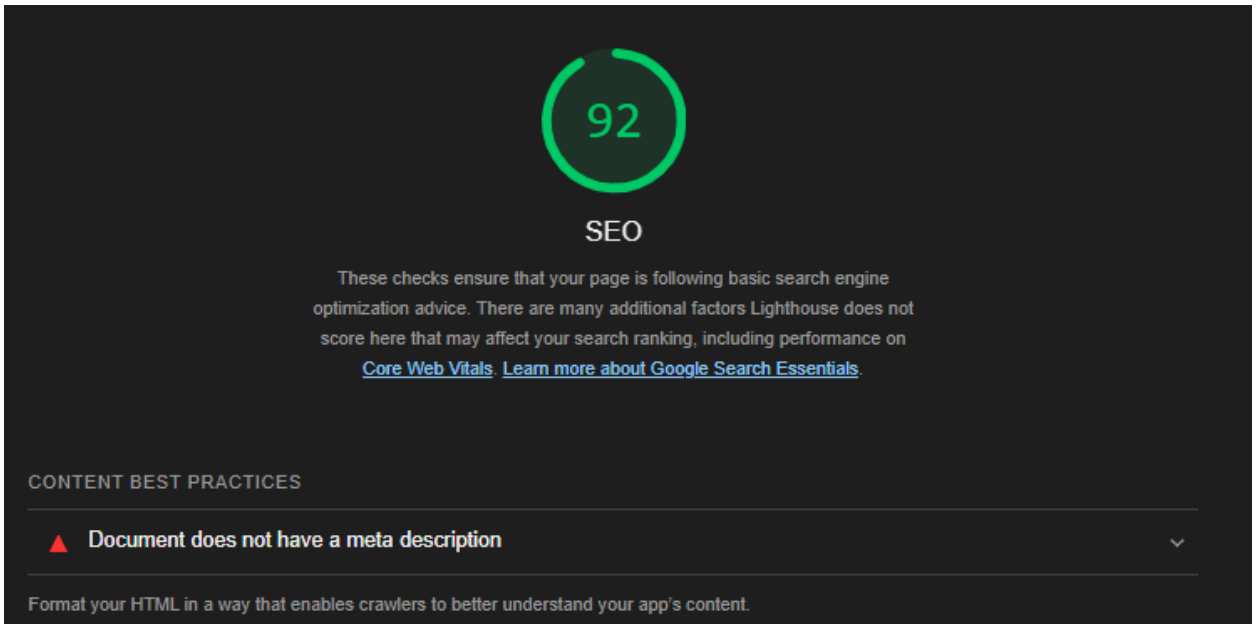
Accessibility

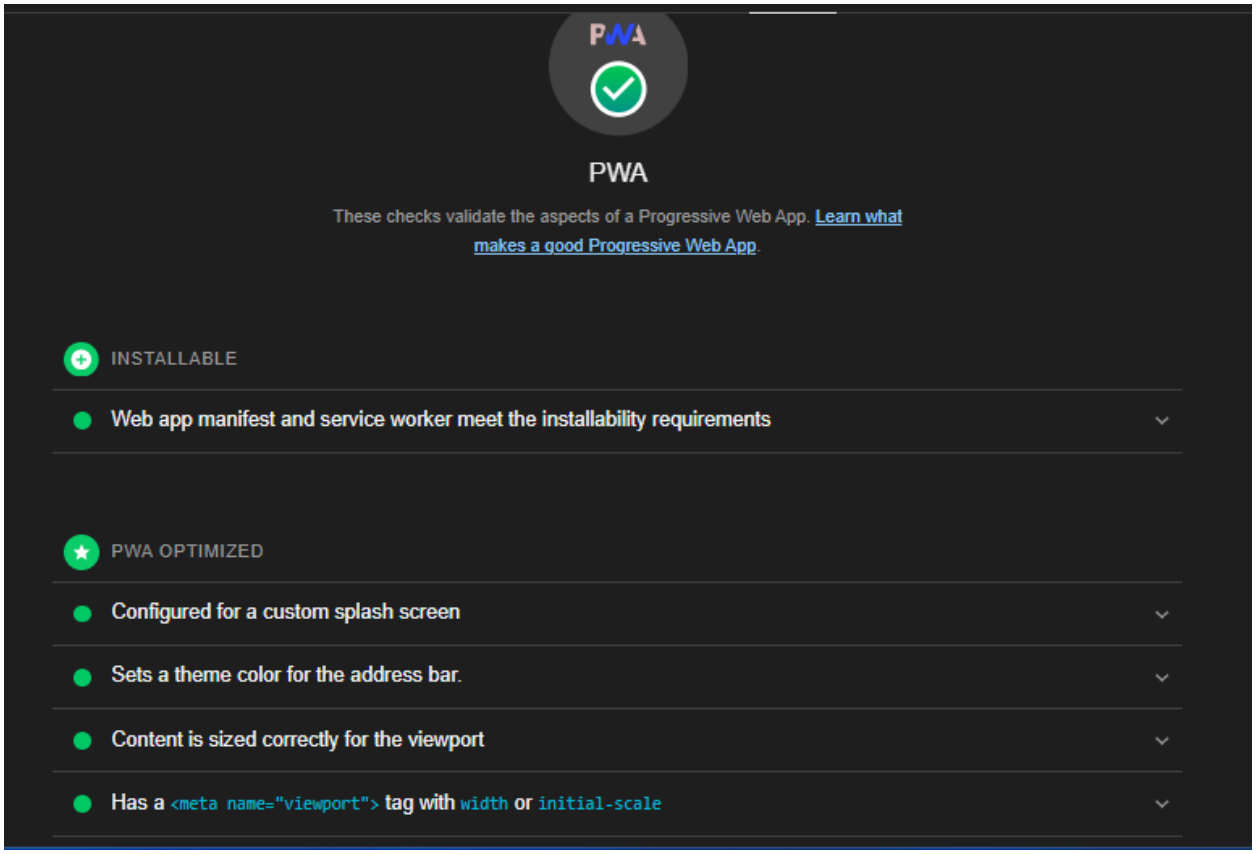
These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so manual testing is also encouraged.

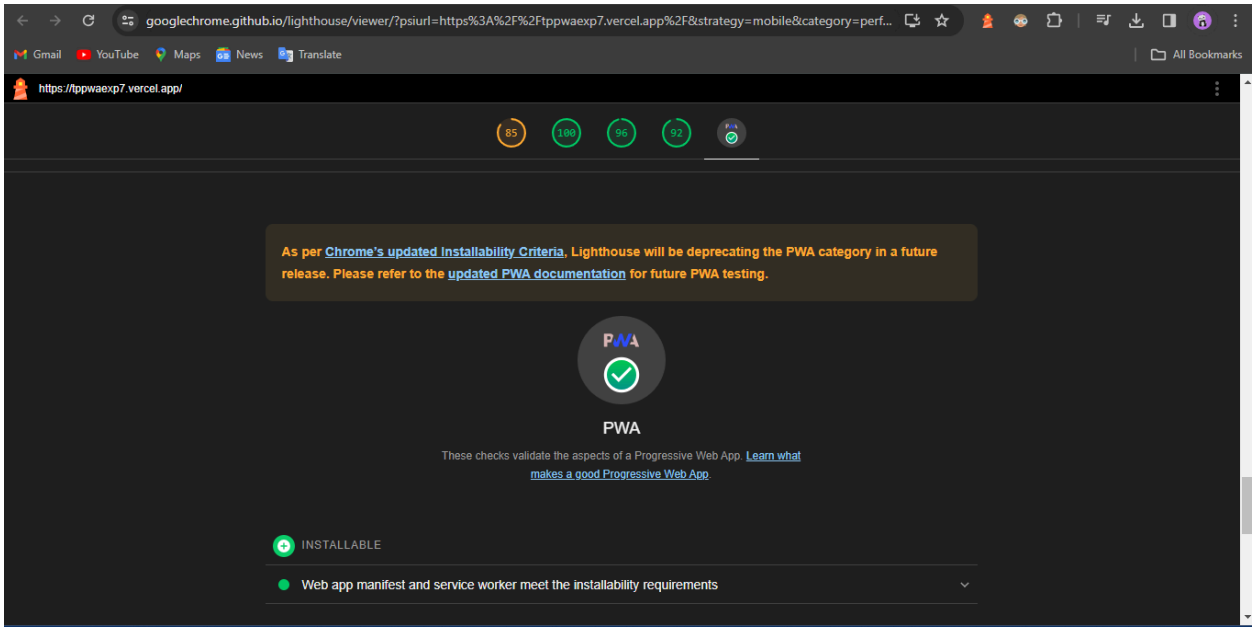
ADDITIONAL ITEMS TO MANUALLY CHECK (10) Hide

- Interactive controls are keyboard focusable
- Interactive elements indicate their purpose and state
- The page has a logical tab order
- Visual order on the page follows DOM order
- User focus is not accidentally trapped in a region


96
Best Practices
TRUST AND SAFETY
Requests the notification permission on page load
Ensure CSP is effective against XSS attacks


92
SEO
These checks ensure that your page is following basic search engine optimization advice. There are many additional factors Lighthouse does not score here that may affect your search ranking, including performance on Core Web Vitals. Learn more about Google Search Essentials.
CONTENT BEST PRACTICES
Document does not have a meta description
Format your HTML in a way that enables crawlers to better understand your app's content.





Conclusion -

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

MAD & PWA Lab Journal

Experiment No.	Assignment-1
Assignment 1 Questions	1. Flutter Overview:Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community. 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.	50
Name	Tejas Shailendra Rokade
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	5

/

Name- Tejas Shailendra Rokade
Div- D15A Rollno- 50 Batch- C

Flutter

Assignment- 1

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

Ans Flutter is Google's software development kit for building iOS and Android apps. Flutter allows you to create cross-platform apps that provide native performance. Apps created with flutter feature beautiful and intuitive design and are able to run animations smoothly. Flutter also increases mobile development speed, helping lower costs.

Key features and Advantages-

① Single Codebase - Code written once can be deployed to iOS and android platforms. This not only reduces development time but also ensures consistent behaviour across different devices.

② Hot Reload - Flutter's hot reload feature allows developers to instantly see the impact of change made to the code without restarting app. Enabling quicker iterations and debugging.

③ Rich Widget Library - Flutter provides comprehensive set of highly customizable widgets for building UIs.

④ Performance - Flutter compiles to native ARM code, resulting in high-performance applications. Use of Dart language and Skia graphics engine contributes smooth animations and fast execution.

⑤ Widget - based Architecture - Flutter's architecture revolves around widgets making it modular and easy to organize code. Developers can compose complex UIs by combining smaller, reusable widgets.

⑥ Open Source - Flutter encourages collaboration and contributions from the community. This leads to continuous improvement, regular updates, and wealth of third-part packages and plugins.

⑦ Cross-Platform Development - Flutter supports cross-platform development for mobile, web and Desktop applications. This versatility allows developers to reach a broader audience with minimal extra effort.

Traditional Approach- ① Developers had to maintain different codebase for iOS and android apps. using languages like Java, swift, Obj-C, etc
② Change in code used to require rebuilding and restarting the entire application, leading to slower development cycles.
③ Achieving a consistent UI across platforms might require using different UI component which was a problem.
④ Implementing custom animations may be challenging, and achieving a highly expressive UI can be time consuming.
Such traditional approach have been tackled and these are some factor that makes Flutter framework different from traditional approaches. ~~And~~ It's key features and advantages is the reason why it has gained popularity in the developer community.

Q2) Widget tree and Composition - Describe the concept of the widget tree in flutter. Explain, how widget composition is used to build complex UI. Provide examples of commonly used widgets and their roles in creating a widget tree.

Ans Widget Composition is a powerful concept in Flutter that involves combining multiple smaller widgets to create more complex and sophisticated UI. Instead of creating large and monolithic UI components, developers can break down the UI into smaller, reusable widgets and compose them hierarchically to form a comprehensive UI. Some of it's benefits are -

① Reusability of code -
② Modularity
③ Readability and Maintainability
④ Flexibility
⑤ Scalability
⑥ Customization.

• ② Widget Tree is a fundamental concept that represents the hierarchy of UI elements in application. Everything in Flutter is a widget, from simple elements to more complex structures like entire screens or even entire applications itself.

• Widget tree is organized in a hierarchical structure, with each widget encapsulating a part of the UI. Structure begins with root widget, typically representing entire application. This root widget can have child widget, and each child widget can further have its own children forming a tree-like structures.

• key points about Widget Tree - ① Hierarchy ② Composition ③ Reusability ④ Immutability ⑤ Hot Reload.

There are 2 types of widget tree ① Stateless ② Stateful widget tree.

Example of commonly used widget

① Material App - Represents root widget of the application.

```
MaterialApp ( home: MyHomePage(),)
```

② Scaffold - Provides a basic structure for the visual elements of an app, such as app bar, body and bottom navigation.

```
Scaffold (
  AppBar: AppBar (
    title: Text('My App'),
  ),
  body: Mybody(),
  )
```

③ Column/Row - Arranges child widgets vertically or horizontally

```
Column (
  Children: [
    Text ('First Child'),
    Text ('Second Child'),
  ],
)
```

④ Container - Basic box model that can contain other widgets and define prop. like padding, margin, etc.

```
Container (
  Margin: EdgeInsets.all (16.0),
  Padding: EdgeInsets.symmetric(horizontal: 20.0, vertical: 10.0),
  ),
  child: Text('Styled Container'),
  )
```

Q3) State Management in Flutter - Discuss the importance of state management in flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable.

Ans State Management is a critical aspect of flutter development, playing a pivotal role in ensuring a responsive and effecient user interface. Effective state management strategies is key to building robust and maintainable applications.

Importance of the state management in flutter:

① Reactivity and UI Updates - Flutter follows reactive programming model, and state management is essential for reflecting changes in application state.

② Performance Optimization - Efficient state management minimizes unnecessary widget rebuilds, improving performance by avoiding redundant UI updates.

③ Code Organization and Maintainability - Well-managed state promotes clean code architecture, making it easier to understand, maintain and scale app.

④ Scalability - As application grow in complexity, proper state management becomes crucial for maintaining a manageable and organized databases.

⑤ Testing - Proper state management facilitates unit testing and makes it easier to write test cases for different application states. This ensures reliability and stability ~~for~~ of the application under various scenarios.

Comparison of state Management approaches -

① setState(): - The simplest and built in way to manage local state with a widget.

- Use case - • Suitable for small to medium-sized applications.
  • Ideal for managing simple UI ~~bas~~ specific states.
- Pros - • Straightforward & easy to use.
  • No external dependencies.
- Cons - • Limited to the scope of a single widget.
  • May lead to widget rebuilds higher in widget tree.

③ Readability and Maintainability

② Provider - A popular state management solution that provides global & easily accessible state.

- Use case - . Good for managing global state, dependency injection, and sharing data between widgets.
- Pros - . Simple & easy to implement.
  . Offers good performance and efficiency.
  . Well-suited for dependency injection.
- Cons - . can be considered overkill for small applications.
  . Limited support for complex asynchronous operations.

③ Riverpod - An extension of provider that introduces improvements in terms of readability, scalability and testability.

- Use Case - . offers improved syntax and advanced features compared to Provider.
- Pros - . Provides improved readability and testability.
  . More flexible in terms of state management.
- Cons - slightly steeper learning curve compared to Provider.

Scenarios

① SetState() - example - Basic calculator app where state changes are isolated to a single screen.

② Provider - ex. - A weather app with multiple screens, where the chosen location and temperature unit need to be shared across different widgets.

③ Riverpod - ex. An e-commerce app with complex user applications, shopping cart management, and real-time updates, where riverpod's advanced features that can be beneficial.

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

Ans Integrating firebase with flutter applications-

1) Create firebase project
2) Register your app - by clicking on Add app and choose the appropriate platform.
3) Add firebase SDK to flutter project-
   Add flutter SDK dependencies to 'pubspec.yaml' file
   dependencies:
      firebase_core: ^1.10.6
      firebase_auth: ^4.10.6
4) Initialize firebase ~~with~~ in Your App:
   'Firebase.initializeApp()' in the 'main()' function.
5) Use firebase services in your App

Benefits of using firebase as a backend solution:

1) Real-time database
2) Authentication
3) Cloud firestore:
4) Cloud functions:
5) Cloud storage:
6) Easy integration
7) Scalability
8) Authentication providers
9) Analytics and Crash Reporting
10) Cloud functions
11) Firebase hosting.

Data Synchronization in flutter-

Firebase cloud ~~sto~~ firestore employs a real-time synchronization model to ensure that data changes are instantly reflected across all connected devices. The process involves following key concepts.

1) Collections and Documents - Data in firestore organized into collections which can contain multiple documents. Each document is a set of key-val pairs.
2) Listeners and observables - Flutter developers can set up listeners to

changes in a particular collection or document. These listeners are notified whenever the data in firestore is modified.

3) Real-time updates - when the data changes in ~~flutt~~ ~~fire~~ ~~base~~ firestore database, the changes are instantly pushed ~~to~~ to all connected clients that have set up listeners for that particular data. This real-time update mechanism ensures that the UI reflects the most current data.

4) Offline Support - Firestore provides offline support, allowing users to interact with the app even when offline. Data modifications made while offline are stored locally and synced with the server once the device is online.

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.	50
Name	Tejas Shailendra Rokade
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4:Understand various PWA frameworks and their requirements LO5: Design and Develop a responsive User Interface by applying PWA Design techniques LO6:Develop and Analyze PWA Features and deploy it over app hosting solutions
Grade:	4

Tejas Shailendra Rokade
D15A 50 Batch-C

PWA - Assignment -2

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

Ans A progressive Web App (PWA) is a type of web application that uses modern web capabilities to deliever an app-like experience to users. PWAs are built using standard web technologies such as HTML, CSS, & Js but are designed to provide a more native-like experience for users, especially on mobile devices.

Key characteristics of Progressive Web Apps are -

① Progressive Enhancement - They are built using progressive enhancement principles, meaning they ~~look~~ provide a basic experience for all users and then enhance based on the capabilities of the user's device and browser.

② Responsive design - PWAs are designed to feel and behave like native mobile apps. They include multiple features such as animations, gestures, and transition to provide a more immersive user experience.

③ Reliability - PWAs are designed to be reliable, even when users are offline or has slow or unreliable internet connection. They cache content and resources to ensure that the app remains functional.

④ Fast performance -

⑤ Discoverability - PWAs are easily discoverable and shareable, as they are built using standard web technologies and can be accessed via a URL.

⑥ Security - PWAs are served over HTTPs to ensure security.

⑦ Cross-platform compatibility -

The significance of PWAs in modern web development lies in their ability to bridge the gap between web and native app experiences. By combining the reach and accessibility of the web with functionality and engagement of native apps. They allow developers to create fast, reliable and engaging experience that work seamlessly

Sundaram
FOR EDUCATIONAL USE

across a wide range of devices and platforms, helping to improve user engagement, retention, and conversion rates. PWAs are easier and more cost effective to develop and maintain compared to native apps, making them an attractive option for businesses and developers looking to reach a broad audience with their applications.

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 Responsive Web design is a web design approach that ensure a Website adapts its layout and content to seamlessly function and display well across various screen sizes, from desktops to smartphones.

Importance for PWAs -

- foundation for App-like Experience - RWD ensures proper layout and navigation across devices, mimicking the adaptability of native apps.
- Accessibility - A responsive design make PWA accessible to wider audience using diverse devices.
- SEO benefits - Responsive design is a positive ranking factor in search engine algo, boosting the discoverability of PWAs.

① Responsive Web design v/s Fluid v/s Adaptive

Feature	Responsive	Fluid	Adaptive
Layout	① Flexible, adapts to any screen size.	① Continuously adjusts based on relative units	① Uses multiple - fixed width layouts.
Development Effort	② Moderate	② Less effort for basic layout.	② More effects to create Multiple Layout

Control	③ Good control over overall layout	③ Less control over element appearance on extreme sizes.	③ High control over layout for each device category.
Suitability for PWAs	④ Ideal foundation due to its versatility	④ Can be used but may require adjustment for optimal experience.	④ Less common for PWAs due to development overhead.

Q3) Describe the lifecycle of service workers, including registration, installation and activation phases.

Ans Service workers are crucial components of progressive web Apps (PWAs) that enable features such as offline support, push notifications, and background synchronization. The lifecycle of service worker involves several key phases: registration, installation and activation.

① Registration.– • 1st phase of lifecycle. Occurs in the main Javascript file of web application.

• Registration is done using the 'navigator.serviceWorker.register()' method, which takes the path to service worker file as a ~~file~~ parameter.

• This method returns a promise that resolves to a service worker registration object, which can be used to interact with service worker.

② Installation – • Once service worker registered, browser downloads and starts installation process of service worker script.

• During installation, the service worker script is executed, and the 'install' event is fired. Inside 'install' event handler, developers can define what resources should be cached for offline access using techniques like caching strategies or precaching.

• Installation process is critical for setting up the service worker's initial ~~code~~ cache and preparing it to take control of web application.

Sundaram
FOR EDUCATIONAL USE

③ Activation - • After installation phase, Activation phase. It occurs when service worker is ready to take control of the web application.
• During activation, 'activate' event is fired, giving service worker an opportunity to clean up ~~and~~ any old caches or perform other tasks necessary for application functionality.
• Developers can use the 'activate' event handler to manage cache cleanup, versioning and other activation related tasks.
• Service remains active until it is explicitly unregistered or updated.
It's important to note that service workers run separately from main web page and operate on a different thread, allowing them to perform tasks like network interception and caching without blocking UI.

Q4) ~~IndexedDB and service workers form a powerful duo for progressive web design Apps. Here's how indexed DB is used within a service worker for data storage.~~
Explain the use of indexed DB in the service worker for data storage.

Ans IndexedDB is a low-level API for client-side storage of significant amounts of structured data, including files and blobs. It's particularly useful for storing data in web applications that need to work offline or with large datasets that may not fit comfortably in memory.
① ~~Operating~~ • Persistent ~~storage~~ Storage - Service workers themselves are temporary by nature. IndexedDB provides persistent storage on the user's device, allowing PWAs to save data even when offline.
• Structured Data - IndexedDB stores data in a structured format using object stores and keys, making it ideal for complex data models used by PWAs.
• Offline Data storage - Service workers are background run separately from web pages and enable features like push notifications and offline functionality. Indexedb provides a way for service workers to store data locally on the user's device, allowing web applications to work offline and improve performance by caching frequently accessed data

FOR EDUCATIONAL USE

- Asynchronous Database operations - Indexeddb operates asynchronously meaning that database operations like adding, retrieving, update, and deleting data do not block the main thread of the web application. This asynchronous nature is well suited for service workers, which need to handle multiple tasks simultaneously without affecting the user experience.

③ Structured data Storage - Indexeddb stores data in key-value pairs within object stores, similar to how data is organized in a database. Service workers can create, access, and manage object stores to store different types of data required by the web application, such as user preferences, cached content or application state.

④ Data Caching and Retrieval - Service workers can use indexeddb to cache resources like HTML, CSS, JS and the images when the user in online. This cached data can then be retrieved and served from the local indexeddb storage when the user is offline, providing a seamless experience even without an internet connection.

Indexeddb is used in service workers for efficient and asynchronous data storage, enabling web applications to function offline, improve performance, and provide a better user experien..

Sundaram FOR EDUCATIONAL USE