Vivekanand Education Society's Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai
Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

Department of Information Technology

CERTIFICATE

This is to certify that **Rujuta Vinit Medhi** 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 **2024-2025.**

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: **Roll No.**

Year/Sem/Class **:** D15A/D15B **A.Y.:** 24-25

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers **:** Mrs. Kajal Joseph.

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: **Roll No.**

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

Project Title: **Roll No.**

Index

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

Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	27
Name	Rujuta Vinit Medhi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

Project Title: BigBasket **RollNo. 27**

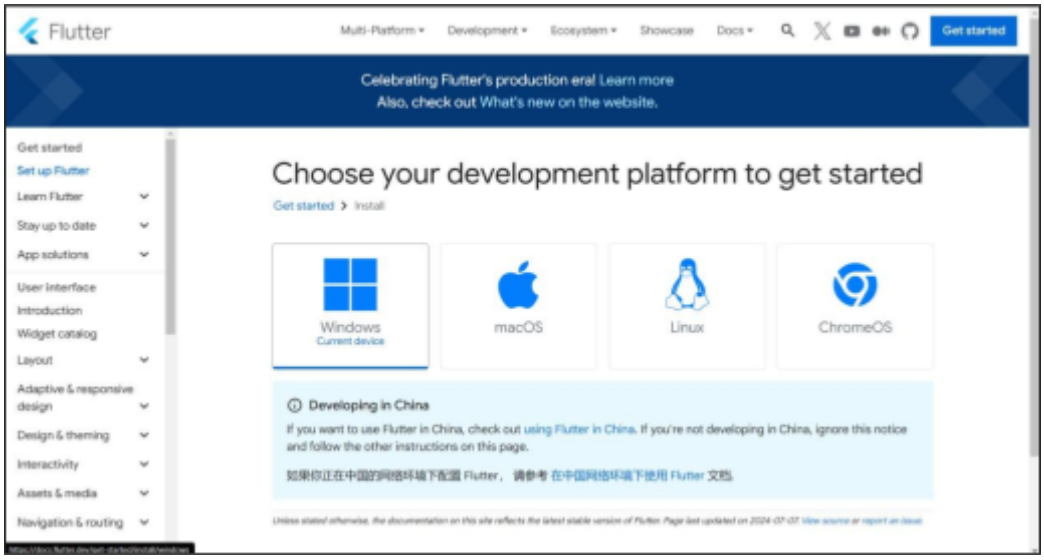
MAD & PWA Lab Journal

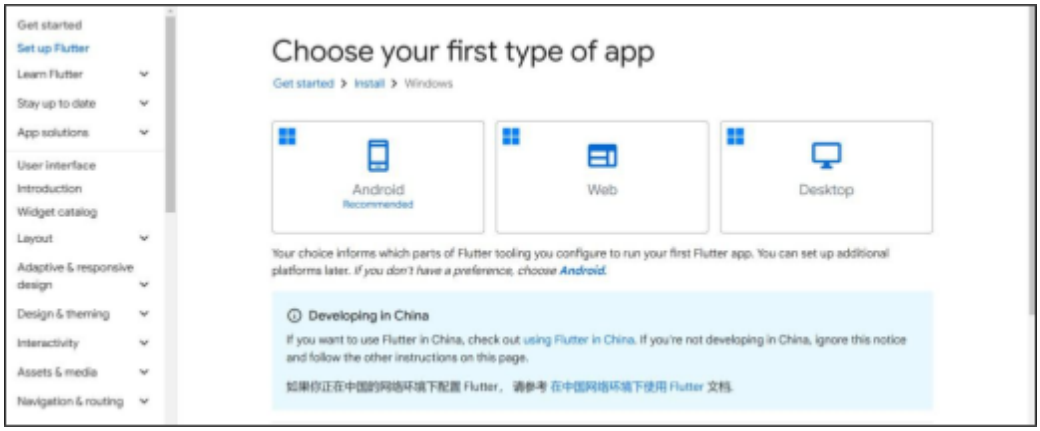
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	27
Name	Rujuta Medhi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

AIM : Installation and Configuration of Flutter Environment.

Step 1: Install Flutter : Download Flutter SDK:

- Go to the Flutter website.
- Download the Flutter SDK for your operating system





Step 2: Extract Flutter SDK: Extract the downloaded .zip file to your desired directory

Step 3: Add Flutter to PATH

- Search for "Environment Variables" in the Start menu.
- Under System Properties → Advanced → Environment Variables, find the Path variable under "System Variables."
- Add the Flutter SDK's bin directory (e.g., C:\flutter\bin) to the PATH.

Now, select path -> click on edit. The following screen appears

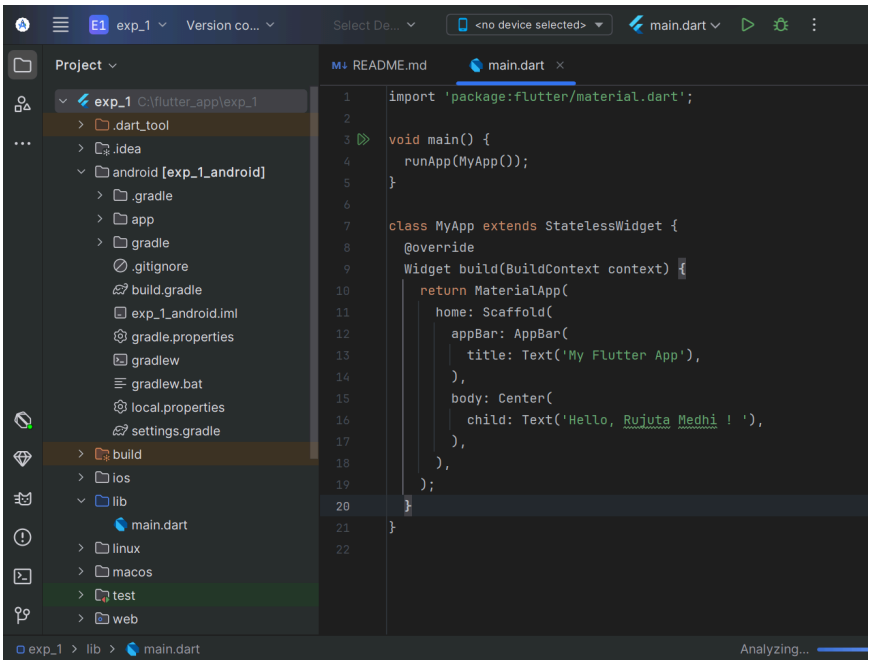
In the above window, click on New->write path of Flutter bin folder in variable value -> ok -> ok -> ok.

Step 4: Now, run the $ flutter command in command prompt.

```
Microsoft Windows [Version 10.0.22631.4602]
(c) Microsoft Corporation. All rights reserved.

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

Common commands:

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

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

Usage: flutter <command> [arguments]

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

Flutter is installed successfully

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



Step 6: Run Flutter Doctor command

```
C:\Users\rugved>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.22631.4602], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[√] Android toolchain - develop for Android devices (Android SDK version 35.0.0)
[√] Chrome - develop for the web
[√] Visual Studio - develop Windows apps (Visual Studio Build Tools 2022 17.11.0)
[√] Android Studio (version 2024.1)
[√] VS Code (version 1.96.4)
[√] Connected device (3 available)
[√] Network resources

• No issues found!
```

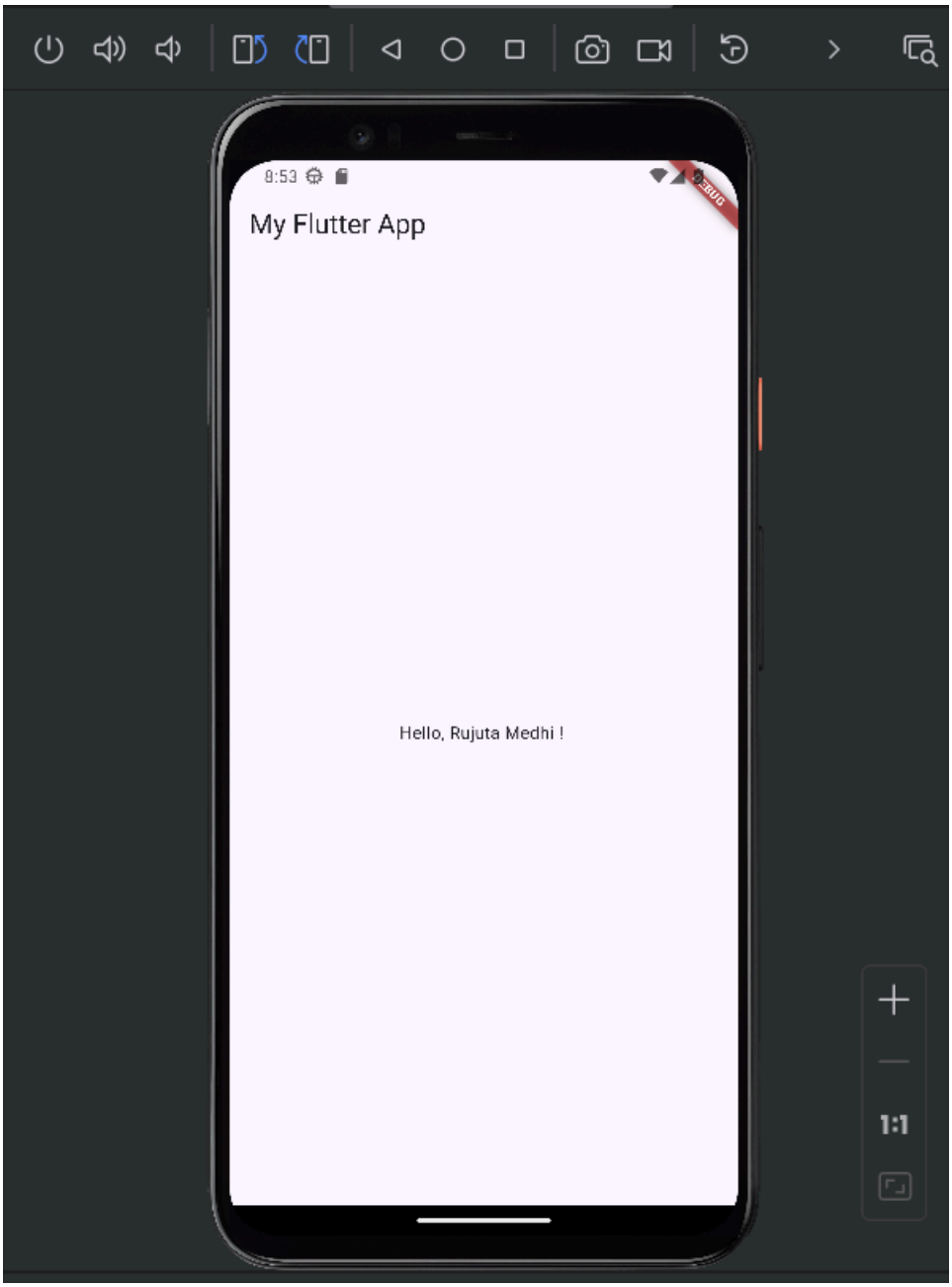
Step 7: Create a new virtual device

Choose your device definition and click on Next.
Select the system image for the latest Android version and click on Next. Now, verify the all AVD configuration. If it is correct, click on Finish. The following screen

.

Once virtual device is created writing a hello name program in flutter:

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('My Flutter App'),
        ),  // AppBar
        body: Center(
          child: Text('Hello, Rujuta Medhi ! '),
        ),  // Center
      ),  // Scaffold
    );  // MaterialApp
  }
}
```


8:53
DEBUG
My Flutter App
Hello, Rujuta Medhi !
1:1

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	27
Name	Rujuta Medhi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Project Title: BigBasket RollNo. 27

MAD & PWA Lab Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	27
Name	Rujuta Medhi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

AIM : To design Flutter UI by including common widgets.

THEORY:

Designing a Flutter UI using **common widgets** involves leveraging the foundational widgets provided by Flutter to build interactive, responsive, and visually appealing user interfaces. These common widgets allow developers to construct the basic building blocks of the app interface, such as text, buttons, images, forms, lists, and more. Here's a breakdown of how Flutter UI design incorporates common widgets:

1. Container

- **Role**: A container is a fundamental widget in Flutter, used to create a rectangular box with various properties like padding, margin, decoration, color, and alignment. It can be used to wrap other widgets and customize their appearance.
- **Usage**: Containers are useful for aligning, decorating, and sizing widgets. For example, it can be used to create a card-like structure, set background colors, or add rounded corners.

2. Row and Column

- **Role**: These are layout widgets used for arranging children widgets either horizontally (Row) or vertically (Column).
- **Usage**: These are commonly used to create flexible layouts. Rows and columns are used for simple linear layouts, but they can also be combined with other layout widgets to create more complex structures (e.g., GridView, Flex).

3. Text

- **Role**: Displays a string of text.
- **Usage**: The Text widget is used to display static text on the screen. It can be customized with various styles (font size, color, weight, etc.).

4. TextField and Form

- **Role**: TextField allows users to enter text, while Form and TextFormField are used for form validation and handling text input.
- **Usage**: Common for building login forms, registration forms, or any interactive form where user input is required.

5. Buttons (ElevatedButton, TextButton, IconButton)

- **Role**: Buttons are used to perform an action when pressed.
- **Usage**: These are the primary way to handle user interaction in Flutter. You can customize their color, shape, padding, and interaction effects.

6. Font

- **Role**: A widget used to display and customize the text within the application.
- **Usage**: The font is applied to text widgets to define their appearance, including font family, size, weight, and style. It's essential for customizing the typographic design of your app, whether you're using default fonts, custom fonts, or fonts from Google Fonts. You can apply different fonts to various text elements to enhance readability and visual appeal.

PROGRAM:

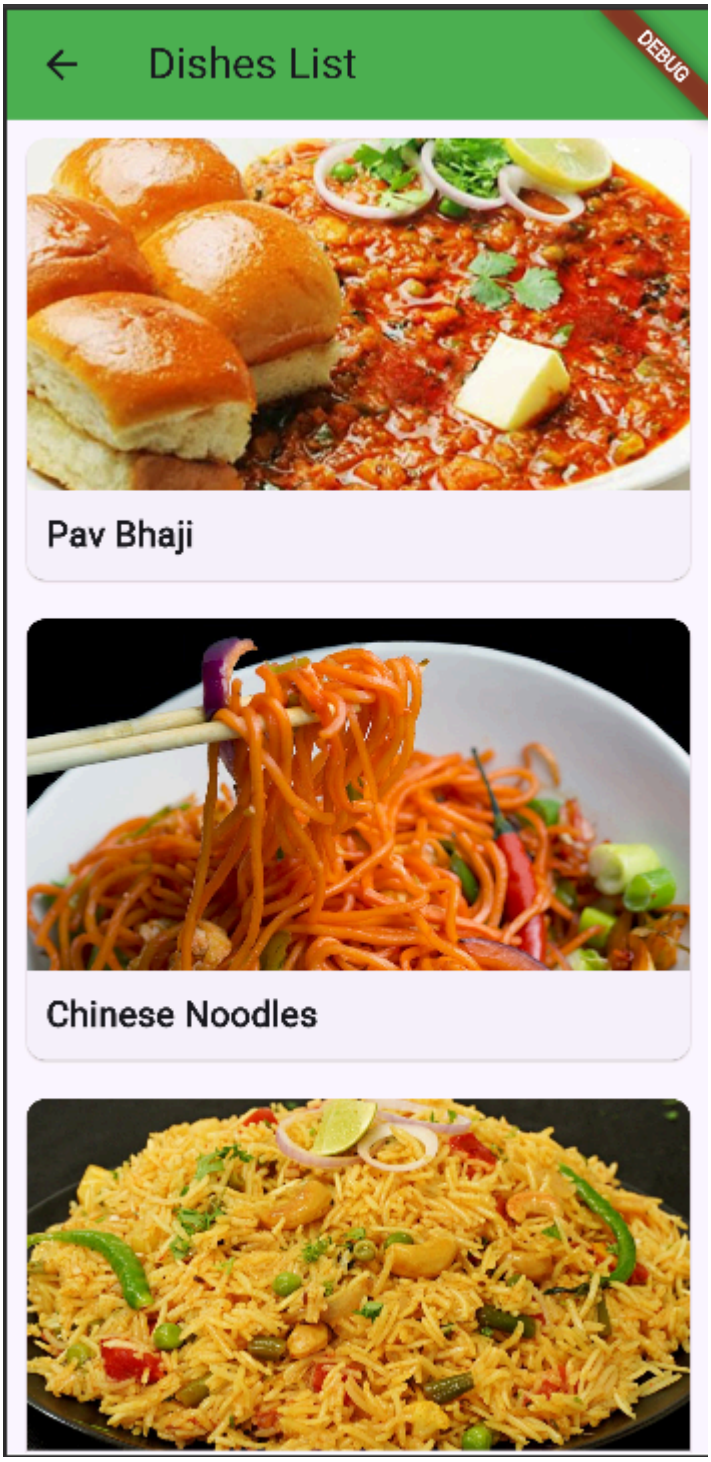
Dishes.dart

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

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

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

  @override
  Widget build(BuildContext context) {
    SystemChrome.setSystemUIOverlayStyle(
      const SystemUiOverlayStyle(
        statusBarColor: Colors.green,
        statusBarIconBrightness: Brightness.light,
      ),
    );

    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primaryColor: Colors.green,
        scaffoldBackgroundColor: Colors.white,
      ),
      home: const HomePage(),
    );
  }
}

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

  final List<Map<String, dynamic>> dishes = const [
    {
      "name": "Pav Bhaji",
      "ingredients": [
        {"name": "Potatoes", "weight": "500g"},
        {"name": "Tomatoes", "weight": "250g"},
        {"name": "Onions", "weight": "150g"},
```

```
      {"name": "Pav Bhaji Masala", "weight": "2 tbsp"},
      {"name": "Butter", "weight": "50g"},
      {"name": "Capsicum", "weight": "100g"},
      {"name": "Garlic", "weight": "10g"},
      {"name": "Green Peas", "weight": "100g"}
    ],
    "image": "assets/pavbhaji.png"
  },
  {
    "name": "Chinese Noodles",
    "ingredients": [
      { "name": "Noodles", "weight": "200g" },
      { "name": "Capsicum", "weight": "100g" },
      { "name": "Carrots", "weight": "100g" },
      { "name": "Spring Onions", "weight": "50g" },
      { "name": "Soy Sauce", "weight": "2 tbsp" },
      { "name": "Vinegar", "weight": "1 tbsp" },
      { "name": "Garlic", "weight": "10g" },
      { "name": "Oil", "weight": "2 tbsp" }
    ],
"image":"assets/noodles.png"
  },
  {
    "name": "Pulao",
    "ingredients": [
      {"name": "Basmati Rice", "weight": "250g"},
      {"name": "Carrots", "weight": "100g"},
      {"name": "Green Peas", "weight": "100g"},
      {"name": "Beans", "weight": "100g"},
      {"name": "Onions", "weight": "100g"},
      {"name": "Cumin Seeds", "weight": "1 tsp"},
      {"name": "Ghee", "weight": "2 tbsp"},
      {"name": "Bay Leaf", "weight": "1 leaf"}
    ],
    "image": "assets/poolao.png"
  },
];

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Dishes List'),
      backgroundColor: Colors.green,
    ),
    body: ListView.builder(
```

```
itemCount: dishes.length,
itemBuilder: (context, index) {
 final dish = dishes[index];

 return GestureDetector(
  onTap: () {
   Navigator.push(
    context,
    MaterialPageRoute(
     builder: (context) => DishDetails(dish: dish),
    ),
   );
  },
  child: Card(
   margin: const EdgeInsets.all(10),
   shape: RoundedRectangleBorder(
    borderRadius: BorderRadius.circular(10),
   ),
   child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
     ClipRRect(
      borderRadius:
      const BorderRadius.vertical(top: Radius.circular(10)),
      child: Image.network(
       dish['image'] ?? '',
       width: double.infinity,
       height: 180,
       fit: BoxFit.cover,
       errorBuilder: (context, error, stackTrace) => Container(
        height: 180,
        color: Colors.grey[300],
        child: const Center(child: Icon(Icons.broken_image)),
       ),
      ),
     ),
     Padding(
      padding: const EdgeInsets.all(10),
      child: Text(
       dish['name'] ?? 'Unknown Dish',
       style: const TextStyle(
        fontSize: 18,
        fontWeight: FontWeight.bold,
       ),
      ),
     ),
```

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



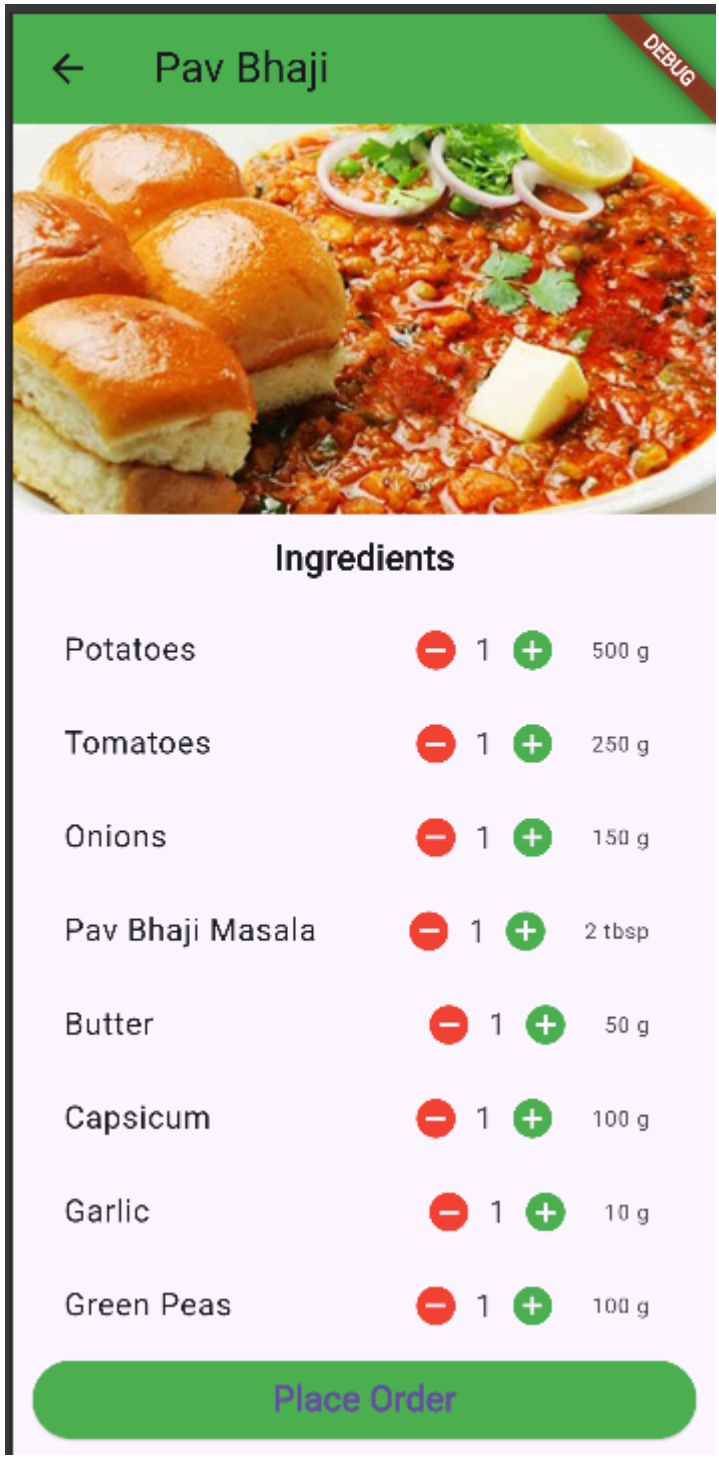
Prod_details.dart

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Product Detail',
      theme: ThemeData(
        primarySwatch: Colors.pink,
        elevatedButtonTheme: ElevatedButtonThemeData(
          style: ElevatedButton.styleFrom(
            backgroundColor: Colors.green,
            foregroundColor: Colors.white,
          ),
        ),
      ),
      home: const ProductDetailPage(),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: IconButton(
          icon: const Icon(Icons.arrow_back),
          onPressed: () {},
        ),
        title: const Text('Ice cream'),
      ),
      body: SingleChildScrollView(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Image.network(
```

```
'https://hebbkx1anhila5yf.public.blob.vercel-storage.com/image-1hitW4RCVecC6jelz0Hlz1YCWy7YbG
.png',
            height: 300,
            width: double.infinity,
            fit: BoxFit.cover,
          ),
          Padding(
            padding: const EdgeInsets.all(16.0),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                const Text(
                  "Kwality Wall\'s Choco-Mint Oreo Brownie Fudge", // Fixed this line
                  style: TextStyle(
                    fontSize: 20,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                const SizedBox(height: 8),
                Row(
                  children: [
                    const Icon(Icons.star, color: Colors.amber, size: 20),
                    const Text(' 4.5'),
                    const Spacer(),
                    Text(
                      '₹255.20',
                      style: TextStyle(
                        fontSize: 20,
                        fontWeight: FontWeight.bold,
                        color: Theme.of(context).primaryColor,
                      ),
                    ),
                  ],
                ),
                const SizedBox(height: 24),
                const Text(
                  'About the Product',
                  style: TextStyle(
                    fontSize: 18,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                const SizedBox(height: 8),
                const Text(
```

```
            'A delicious blend of chocolate and mint ice cream with Oreo cookie pieces and brownie fudge swirls. Perfect for dessert lovers who enjoy the combination of refreshing mint with rich chocolate.',
            style: TextStyle(
              fontSize: 16,
              color: Colors.grey,
            ),
          ),
          const SizedBox(height: 24),
          const Text(
            'How to Use',
            style: TextStyle(
              fontSize: 18,
              fontWeight: FontWeight.bold,
            ),
          ),
          const SizedBox(height: 8),
          const Text(
            'Store in freezer. Allow to soften slightly before serving for best taste and texture.',
            style: TextStyle(
              fontSize: 16,
              color: Colors.grey,
            ),
          ),
          const SizedBox(height: 24),
          Row(
            children: [
              Expanded(
                child: ElevatedButton(
                  onPressed: () {},
                  style: ElevatedButton.styleFrom(
                    padding: const EdgeInsets.symmetric(vertical: 16),
                  ),
                  child: const Text(
                    'Add',
                    style: TextStyle(fontSize: 16),
                  ),
                ),
              ),
            ],
          ),
        ],
      ),
    ),
  ],
),
```

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


Pav Bhaji
DEBUG
Ingredients
Potatoes 1 500 g
Tomatoes 1 250 g
Onions 1 150 g
Pav Bhaji Masala 1 2 tbsp
Butter 1 50 g
Capsicum 1 100 g
Garlic 1 10 g
Green Peas 1 100 g
Place Order

Order Form

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        textTheme: GoogleFonts.poppinsTextTheme(Theme.of(context).textTheme),
      ),
      home: const RegistrationForm(),
    );
  }
}

class RegistrationForm extends StatefulWidget {
  const RegistrationForm({Key? key}) : super(key: key);

  @override
  _RegistrationFormState createState() => _RegistrationFormState();
}

class _RegistrationFormState extends State<RegistrationForm> {
final _formKey = GlobalKey<FormState>();
final _controllers = List.generate(5, (_) => TextEditingController());
final _focusNodes = List.generate(5, (_) => FocusNode());

@override
void dispose() {
  for (var controller in _controllers) {
    controller.dispose();
  }
  for (var node in _focusNodes) {
```

```
    node.dispose();
  }
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Container(
      decoration: const BoxDecoration(
        gradient: LinearGradient(
          begin: Alignment.topLeft,
          end: Alignment.bottomRight,
          colors: [Color(0xFFE8F5E9), Colors.white],
        ),
      ),
      child: SafeArea(
        child: CustomScrollView(
          slivers: [
            SliverAppBar(
              floating: true,
              backgroundColor: Colors.transparent,
              elevation: 0,
              flexibleSpace: FlexibleSpaceBar(
                title: Text(
                  'Order Form',
                  style: GoogleFonts.poppins(
                    color: Colors.green[800],
                    fontWeight: FontWeight.w600,
                  ),
                ),
              ),
            ),
            SliverToBoxAdapter(
              child: Padding(
                padding: const EdgeInsets.all(24.0),
                child: Form(
                  key: _formKey,
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      Text(
                        'Please fill out the form to order',
                        style: GoogleFonts.poppins(
                          fontSize: 16,
                          color: Colors.green[800],
```

```
                    fontWeight: FontWeight.w500,
                  ),
                ),
                const SizedBox(height: 24),
                ..._buildFormFields(),
                const SizedBox(height: 32),
                _buildSubmitButton(),
              ],
            ),
          ),
        ),
      ),
    ),
  ),
 );
}

List<Widget> _buildFormFields() {
  final labels = ['First Name', 'Last Name', 'Address', 'Landmark', 'Mobile Number'];
  final icons = [Icons.person, Icons.person, Icons.home, Icons.location_on, Icons.phone];

  return List.generate(
    5,
      (index) => Padding(
     padding: const EdgeInsets.only(bottom: 16),
     child: TextFormField(
      controller: _controllers[index],
      focusNode: _focusNodes[index],
      decoration: _getInputDecoration(labels[index], icons[index]),
      validator: (value) {
       if (value == null || value.isEmpty) {
        return 'Please enter ${labels[index].toLowerCase()}';  // Ensures the field is required
       }
       if (index == 4 && value.length != 10) {
        return 'Mobile number must be 10 digits'; // Mobile number validation
       }
       return null;
      },
      onFieldSubmitted: (_) {
       if (index < 4) {
        FocusScope.of(context).requestFocus(_focusNodes[index + 1]);
       }
      },
      keyboardType: index == 4 ? TextInputType.phone : TextInputType.text,
```

```
      ),
    ),
  );
}

InputDecoration _getInputDecoration(String label, IconData icon) {
  return InputDecoration(
    labelText: label,
    prefixIcon: Icon(icon, color: Colors.green[800]),
    labelStyle: TextStyle(color: Colors.green[800]),
    enabledBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.green[200]!),
    ),
    focusedBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.green[800]!, width: 2),
    ),
    errorBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.red[400]!),
    ),
    focusedErrorBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.red[400]!, width: 2),
    ),
    filled: true,
    fillColor: Colors.white,
  );
}

Widget _buildSubmitButton() {
  return SizedBox(
    width: double.infinity,
    child: ElevatedButton(
      onPressed: _submitForm,
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.green[800],
        padding: const EdgeInsets.symmetric(vertical: 16),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(12),
        ),
      ),
      child: Text(
        'SUBMIT',
        style: GoogleFonts.poppins(
```

```
          color: Colors.white,
          fontSize: 18,
          fontWeight: FontWeight.w600,
        ),
      ),
    ),
  );
}

void _submitForm() {
  if (_formKey.currentState!.validate()) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text(
          'Order Successful!',
          style: GoogleFonts.poppins(),
        ),
        backgroundColor: Colors.green[800],
      ),
    );
  }
}
}
```

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	27
Name	Rujuta Medhi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Project Title: BigBasket RollNo. 27

MAD & PWA Lab Journal

Experiment No.	03
Experiment Title.	To include image, fonts and icons in flutter app.
Roll No.	27
Name	Rujuta Medhi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

AIM : To include image, fonts and icons in flutter app.

THEORY:

Including Images, Fonts, and Icons in a Flutter App (Theory)

Flutter allows developers to **enhance UI/UX** by including **images, custom fonts, and icons** efficiently. Below is an overview of how each of these assets can be integrated into a Flutter application.

1. Adding Images in Flutter

Flutter supports **both local and network images** for UI components.

Local Images

- Images can be stored in the **assets** folder of the project.
- The `pubspec.yaml` file needs to be updated to include the images.
- Supported formats: **PNG, JPG, GIF, SVG** (with additional packages like `flutter_svg`).

Network Images

- Images can be loaded from the internet using URLs.
- Cached network images can be used for better performance.

2. Adding Custom Fonts in Flutter

Custom fonts improve branding and UI aesthetics.

- **Download and store** font files in the **assets/fonts/** directory.
- Declare them in `pubspec.yaml` under the `fonts` section.
- Assign the font to specific text styles using the `TextStyle` widget.

3. Using Icons in Flutter

Icons enhance UI clarity and usability. Flutter supports **built-in icons and custom icons**.

Built-in Icons (Material Icons)

- Flutter provides a rich set of **Material Icons** through the `Icons` class.
- These icons do not require additional setup.

Custom Icons (SVG, PNG, Font Icons)

- Icons can be added as **SVG** (using `flutter_svg` package) or **PNG** images.
- Font-based icon sets like **FontAwesome** or **Custom Icon Fonts** can be integrated.

PROGRAM:

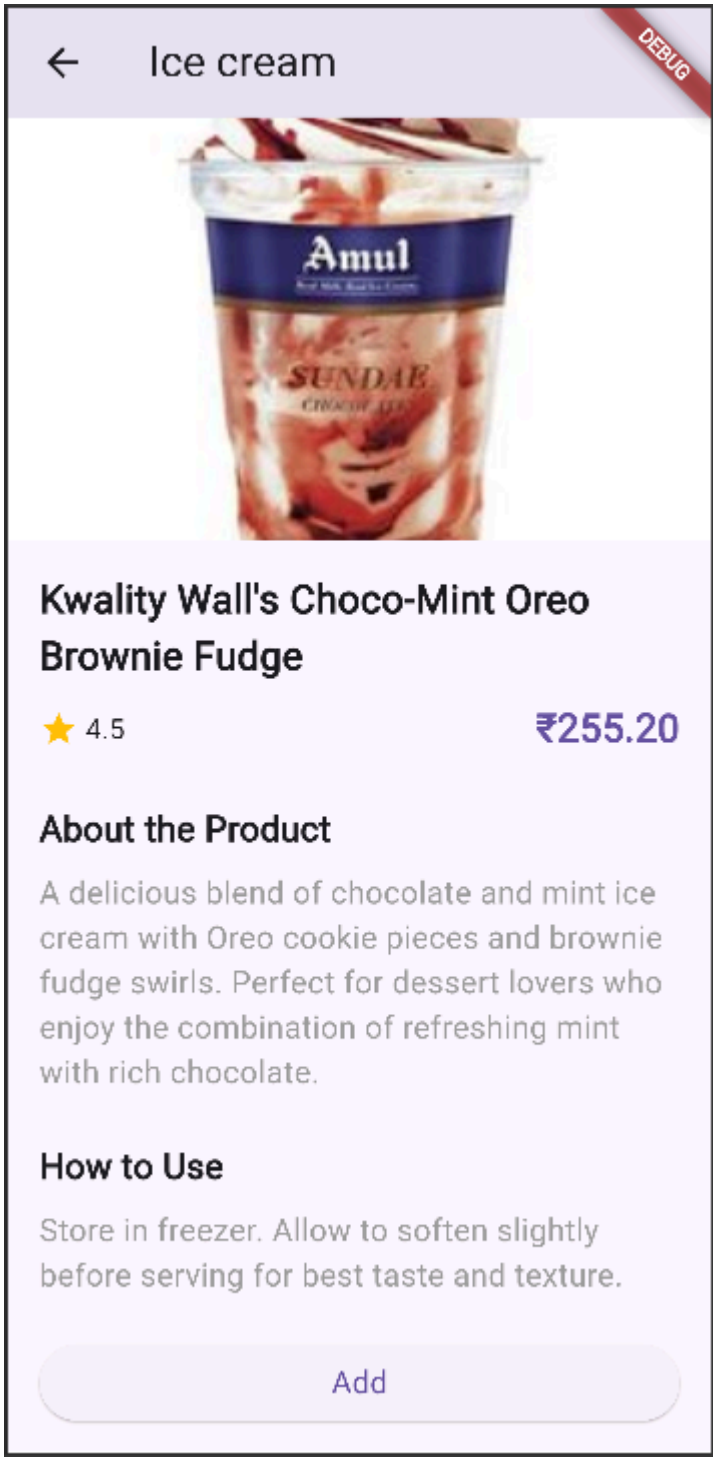
Prod_details.dart

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Product Detail',
      theme: ThemeData(
        primarySwatch: Colors.pink,
        elevatedButtonTheme: ElevatedButtonThemeData(
          style: ElevatedButton.styleFrom(
            backgroundColor: Colors.green,
            foregroundColor: Colors.white,
          ),
        ),
      ),
      home: const ProductDetailPage(),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: IconButton(
          icon: const Icon(Icons.arrow_back),
          onPressed: () {},
        ),
        title: const Text('Ice cream'),
      ),
      body: SingleChildScrollView(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
```

```
children: [
  Image.network(
    'assets/sunday.jpg',
    height: 300,
    width: double.infinity,
    fit: BoxFit.cover,
  ),
  Padding(
    padding: const EdgeInsets.all(16.0),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        const Text(
          "Kwality Wall\'s Choco-Mint Oreo Brownie Fudge", // Fixed this line
          style: TextStyle(
            fontSize: 20,
            fontWeight: FontWeight.bold,
          ),
        ),
        const SizedBox(height: 8),
        Row(
          children: [
            const Icon(Icons.star, color: Colors.amber, size: 20),
            const Text(' 4.5'),
            const Spacer(),
            Text(
              '₹255.20',
              style: TextStyle(
                fontSize: 20,
                fontWeight: FontWeight.bold,
                color: Theme.of(context).primaryColor,
              ),
            ),
          ],
        ),
        const SizedBox(height: 24),
        const Text(
          'About the Product',
          style: TextStyle(
            fontSize: 18,
            fontWeight: FontWeight.bold,
          ),
        ),
        const SizedBox(height: 8),
        const Text(
```

```
            'A delicious blend of chocolate and mint ice cream with Oreo cookie pieces and brownie fudge swirls. Perfect for dessert lovers who enjoy the combination of refreshing mint with rich chocolate.',
            style: TextStyle(
              fontSize: 16,
              color: Colors.grey,
            ),
          ),
          const SizedBox(height: 24),
          const Text(
            'How to Use',
            style: TextStyle(
              fontSize: 18,
              fontWeight: FontWeight.bold,
            ),
          ),
          const SizedBox(height: 8),
          const Text(
            'Store in freezer. Allow to soften slightly before serving for best taste and texture.',
            style: TextStyle(
              fontSize: 16,
              color: Colors.grey,
            ),
          ),
          const SizedBox(height: 24),
          Row(
            children: [
              Expanded(
                child: ElevatedButton(
                  onPressed: () {},
                  style: ElevatedButton.styleFrom(
                    padding: const EdgeInsets.symmetric(vertical: 16),
                  ),
                  child: const Text(
                    'Add',
                    style: TextStyle(fontSize: 16),
                  ),
                ),
              ),
            ],
          ),
        ],
      ),
    ),
  ],
),
```

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


Ice cream
DEBUG
Amul
SUNDAE
Kwality Wall's Choco-Mint Oreo Brownie Fudge
4.5
₹255.20
About the Product
A delicious blend of chocolate and mint ice cream with Oreo cookie pieces and brownie fudge swirls. Perfect for dessert lovers who enjoy the combination of refreshing mint with rich chocolate.
How to Use
Store in freezer. Allow to soften slightly before serving for best taste and texture.
Add

Order Form

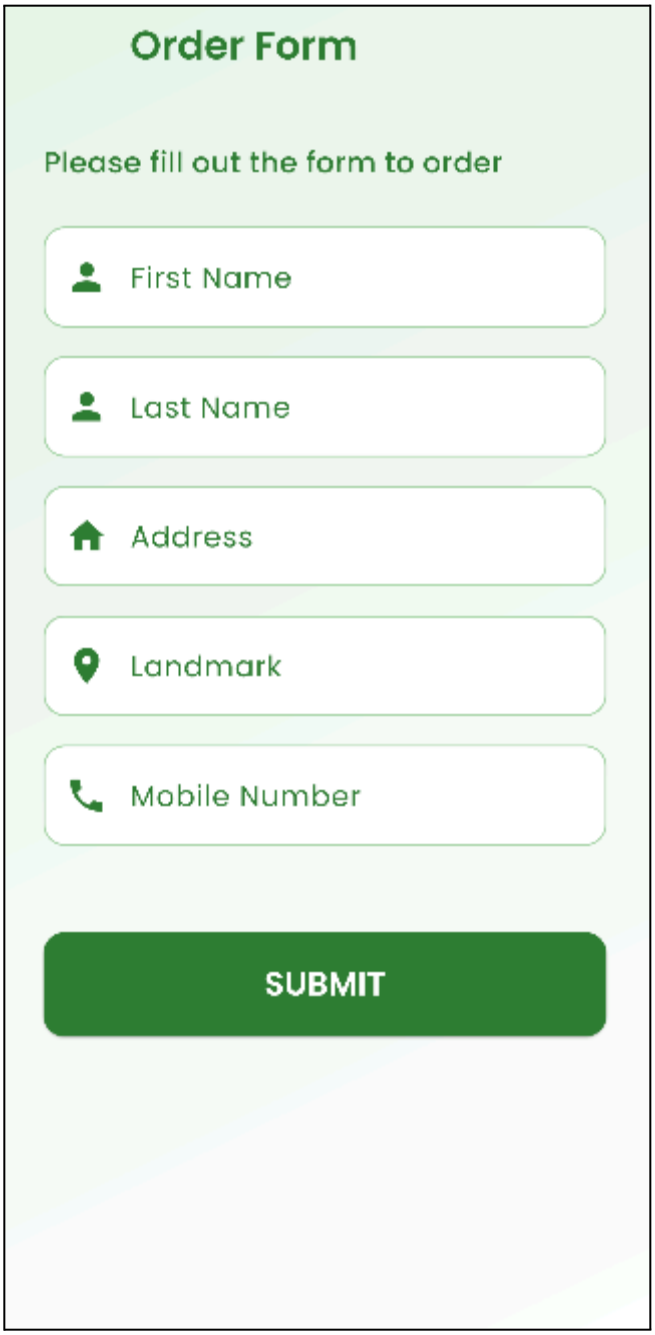
```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        textTheme: GoogleFonts.poppinsTextTheme(Theme.of(context).textTheme),
      ),
      home: const RegistrationForm(),
    );
  }
}

class RegistrationForm extends StatefulWidget {
  const RegistrationForm({Key? key}) : super(key: key);

  @override
  _RegistrationFormState createState() => _RegistrationFormState();
}

class _RegistrationFormState extends State<RegistrationForm> {
final _formKey = GlobalKey<FormState>();
final _controllers = List.generate(5, (_) => TextEditingController());
final _focusNodes = List.generate(5, (_) => FocusNode());

@override
void dispose() {
  for (var controller in _controllers) {
    controller.dispose();
  }
  for (var node in _focusNodes) {
    node.dispose();
```

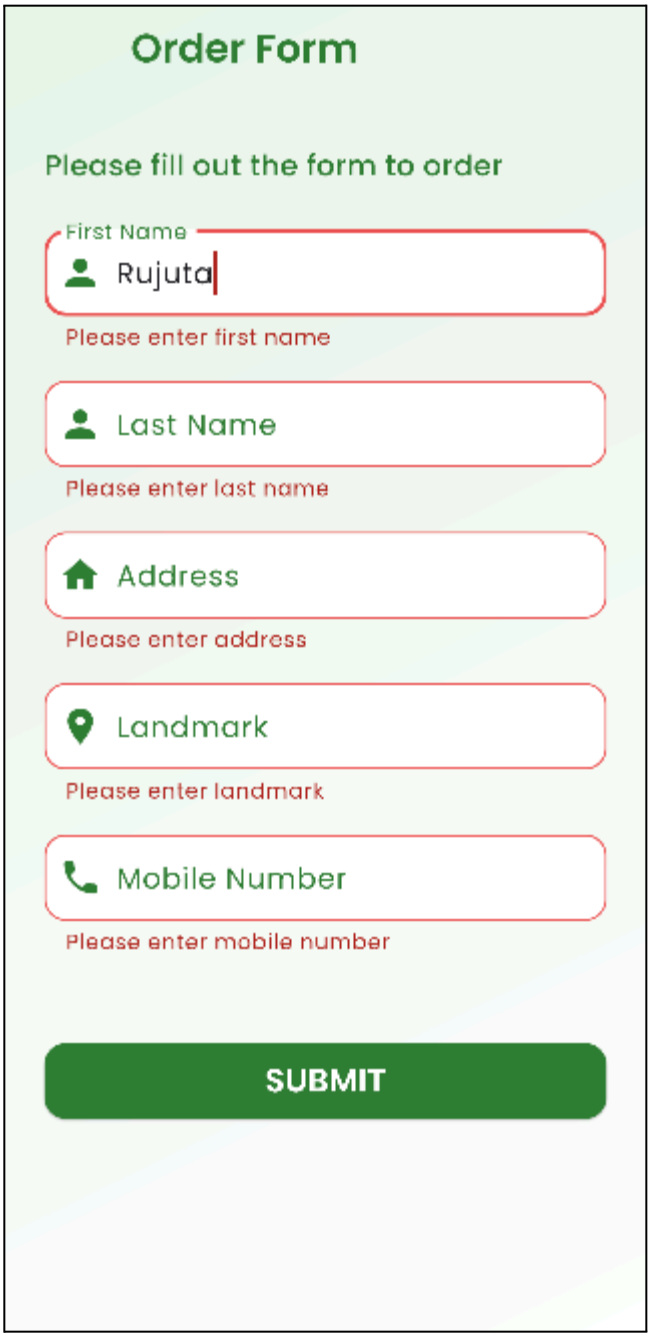
```
  }
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Container(
      decoration: const BoxDecoration(
        gradient: LinearGradient(
          begin: Alignment.topLeft,
          end: Alignment.bottomRight,
          colors: [Color(0xFFE8F5E9), Colors.white],
        ),
      ),
      child: SafeArea(
        child: CustomScrollView(
          slivers: [
            SliverAppBar(
              floating: true,
              backgroundColor: Colors.transparent,
              elevation: 0,
              flexibleSpace: FlexibleSpaceBar(
                title: Text(
                  'Order Form',
                  style: GoogleFonts.poppins(
                    color: Colors.green[800],
                    fontWeight: FontWeight.w600,
                  ),
                ),
              ),
            ),
            SliverToBoxAdapter(
              child: Padding(
                padding: const EdgeInsets.all(24.0),
                child: Form(
                  key: _formKey,
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      Text(
                        'Please fill out the form to order',
                        style: GoogleFonts.poppins(
                          fontSize: 16,
                          color: Colors.green[800],
                          fontWeight: FontWeight.w500,
```

```
                  ),
                ),
                const SizedBox(height: 24),
                ..._buildFormFields(),
                const SizedBox(height: 32),
                _buildSubmitButton(),
              ],
            ),
          ),
        ),
      ),
    ),
  ),
 );
}

List<Widget> _buildFormFields() {
  final labels = ['First Name', 'Last Name', 'Address', 'Landmark', 'Mobile Number'];
  final icons = [Icons.person, Icons.person, Icons.home, Icons.location_on, Icons.phone];

  return List.generate(
    5,
      (index) => Padding(
     padding: const EdgeInsets.only(bottom: 16),
     child: TextFormField(
      controller: _controllers[index],
      focusNode: _focusNodes[index],
      decoration: _getInputDecoration(labels[index], icons[index]),
      validator: (value) {
       if (value == null || value.isEmpty) {
        return 'Please enter ${labels[index].toLowerCase()}';  // Ensures the field is required
       }
       if (index == 4 && value.length != 10) {
        return 'Mobile number must be 10 digits'; // Mobile number validation
       }
       return null;
      },
      onFieldSubmitted: (_) {
       if (index < 4) {
        FocusScope.of(context).requestFocus(_focusNodes[index + 1]);
       }
      },
      keyboardType: index == 4 ? TextInputType.phone : TextInputType.text,
     ),
```

```
    ),
  );
}

InputDecoration _getInputDecoration(String label, IconData icon) {
  return InputDecoration(
    labelText: label,
    prefixIcon: Icon(icon, color: Colors.green[800]),
    labelStyle: TextStyle(color: Colors.green[800]),
    enabledBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.green[200]!),
    ),
    focusedBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.green[800]!, width: 2),
    ),
    errorBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.red[400]!),
    ),
    focusedErrorBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.red[400]!, width: 2),
    ),
    filled: true,
    fillColor: Colors.white,
  );
}

Widget _buildSubmitButton() {
  return SizedBox(
    width: double.infinity,
    child: ElevatedButton(
      onPressed: _submitForm,
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.green[800],
        padding: const EdgeInsets.symmetric(vertical: 16),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(12),
        ),
      ),
      child: Text(
        'SUBMIT',
        style: GoogleFonts.poppins(
          color: Colors.white,
```

```
          fontSize: 18,
          fontWeight: FontWeight.w600,
        ),
      ),
    ),
  );
}

void _submitForm() {
  if (_formKey.currentState!.validate()) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text(
          'Order Successful!',
          style: GoogleFonts.poppins(),
        ),
        backgroundColor: Colors.green[800],
      ),
    );
  }
}
}
```


Order Form
Please fill out the form to order
First Name
Last Name
Address
Landmark
Mobile Number
SUBMIT


Order Form
Please fill out the form to order
First Name
Rujuta
Please enter first name
Last Name
Please enter last name
Address
Please enter address
Landmark
Please enter landmark
Mobile Number
Please enter mobile number
SUBMIT

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	27
Name	Rujuta Medhi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Project Title: BigBasket RollNo. 27

MAD & PWA Lab Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	27
Name	Rujuta Medhi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

AIM : To create an interactive Form using form widget.

THEORY:

- MaterialApp: Sets up the basic structure of the app, including theme and home screen.
- StatelessWidget: A widget that doesn't have mutable state, used for the root app (MyApp).
- StatefulWidget: A widget with mutable state, used for the registration form (RegistrationForm).
- GlobalKey<FormState>: A key to identify and validate the form.
- TextEditingController: Manages text input and allows text fields to be modified programmatically.
- FocusNode: Tracks the focus state of text fields for navigating between fields.
- Scaffold: Provides a structure for the visual layout of the app, including app bar, body, and bottom sheets.
- SafeArea: Ensures the UI elements don't overlap with system UI areas like notches or status bars.
- CustomScrollView: Allows custom scrolling behavior for multiple slivers.
- SliverAppBar: A collapsible app bar that responds to scrolling.
- FlexibleSpaceBar: Provides a flexible area in the SliverAppBar that can expand or contract.
- SliverToBoxAdapter: Used to add regular widgets (like padding) in a sliver-based layout.
- TextFormField: A form field that allows the user to input text with validation and custom styling.
- InputDecoration: Customizes the appearance of text fields (e.g., label text, icons, borders).
- TextInputType: Specifies the type of input expected in the text field (e.g., phone number or text).
- ElevatedButton: A Material Design button with elevated appearance, used for submitting the form.
- SnackBar: A temporary message that appears at the bottom of the screen, typically to show feedback to the user.
- GoogleFonts.poppins: Loads and applies the Poppins font style from Google Fonts to the text.

PROGRAM:

```
Login and Register
import 'package:flutter/material.dart';
import 'package:bigbasket_website/home.dart'; //
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
```

```
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Login Screen',
      theme: ThemeData(
        primarySwatch: Colors.red,
      ),
      home: const LoginScreen(),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SingleChildScrollView(
        child: Column(
          children: [
            // Top image section
            Container(
              height: 250,
              decoration: const BoxDecoration(
                image: DecorationImage(
                  image: NetworkImage(
                      'assets/loginnn.png'),
                  fit: BoxFit.cover,
                ),
              ),
              child: const Padding(
                padding: EdgeInsets.all(24.0),
                child: Align(
                  alignment: Alignment.bottomLeft,

                ),
              ),
            ),
            // Login form section
            Padding(
              padding: const EdgeInsets.all(24.0),
```

```
child: Column(
  crossAxisAlignment: CrossAxisAlignment.stretch,
  children: [
    const Text(
      'Login',
      style: TextStyle(
        fontSize: 24,
        fontWeight: FontWeight.bold,
      ),
    ),
    const SizedBox(height: 24),
    // Social login buttons
    Row(
      children: [
        Expanded(
          child: ElevatedButton.icon(
            onPressed: () {},
            icon: const Icon(Icons.facebook, color: Colors.white),
            label: const Text('FACEBOOK',
                style: TextStyle(color: Colors.white)),
            style: ElevatedButton.styleFrom(
              backgroundColor: const Color(0xFF3B5998),
              padding: const EdgeInsets.symmetric(vertical: 12),
            ),
          ),
        ),
        const SizedBox(width: 16),
        Expanded(
          child: OutlinedButton.icon(
            onPressed: () {},
            icon: const Icon(Icons.g_mobiledata,
                color: Colors.black87),
            label: const Text('GOOGLE',
                style: TextStyle(color: Colors.black87)),
            style: OutlinedButton.styleFrom(
              padding: const EdgeInsets.symmetric(vertical: 12),
            ),
          ),
        ),
      ],
    ),
    const SizedBox(height: 24),
    const Text(
      'or',
```

```
  textAlign: TextAlign.center,
  style: TextStyle(color: Colors.grey),
),
const SizedBox(height: 24),
// Email field
TextField(
  decoration: InputDecoration(
    hintText: 'Email',
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(4),
    ),
  ),
),
const SizedBox(height: 16),
// Password field
TextField(
  obscureText: true,
  decoration: InputDecoration(
    hintText: 'Password',
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(4),
    ),
  ),
),
const SizedBox(height: 16),
// Remember me checkbox
Row(
  children: [
    Checkbox(
      value: false,
      onChanged: (bool? value) {},
    ),
    const Text('Remember me'),
    const Spacer(),
    TextButton(
      onPressed: () {},
      child: const Text('TERMS & CONDITIONS',
          style: TextStyle(color: Colors.green)),
    ),
  ],
),
const SizedBox(height: 24),
// Login button
ElevatedButton(
```

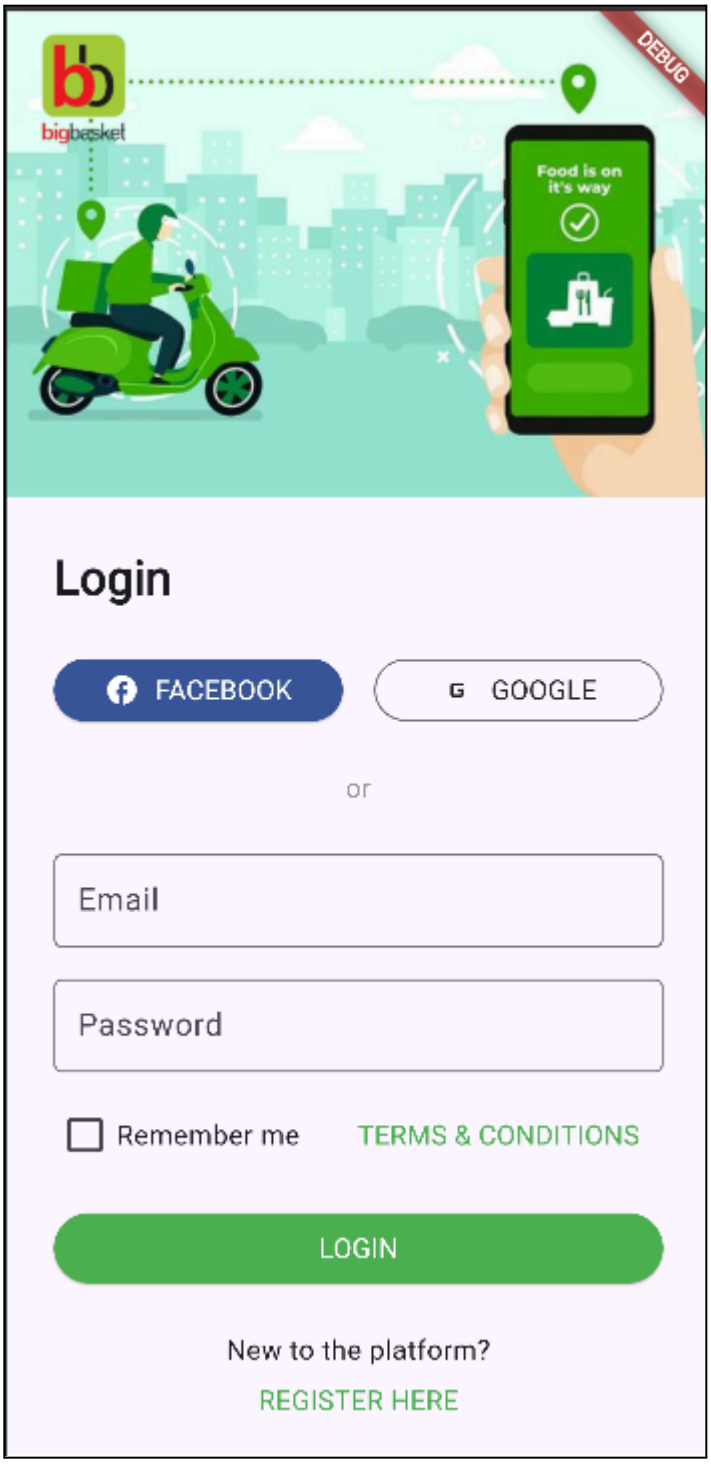
```
                onPressed: () {
                  Navigator.pushReplacement(
                    context,
                    MaterialPageRoute(builder: (context) => const HomePage()), // Navigate to
HomePage
                  );
                },
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.green,
                  padding: const EdgeInsets.symmetric(vertical: 16),
                ),
                child: const Text('LOGIN', style: TextStyle(color: Colors.white)),
              ),

              const SizedBox(height: 24),
              // Bottom text
              const Text(
                'New to the platform?',
                textAlign: TextAlign.center,
              ),
              TextButton(
                onPressed: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(builder: (context) => const RegisterScreen()),
                  );
                },
                child: const Text('REGISTER HERE', style: TextStyle(color: Colors.green)),
              ),

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

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

 @override
```

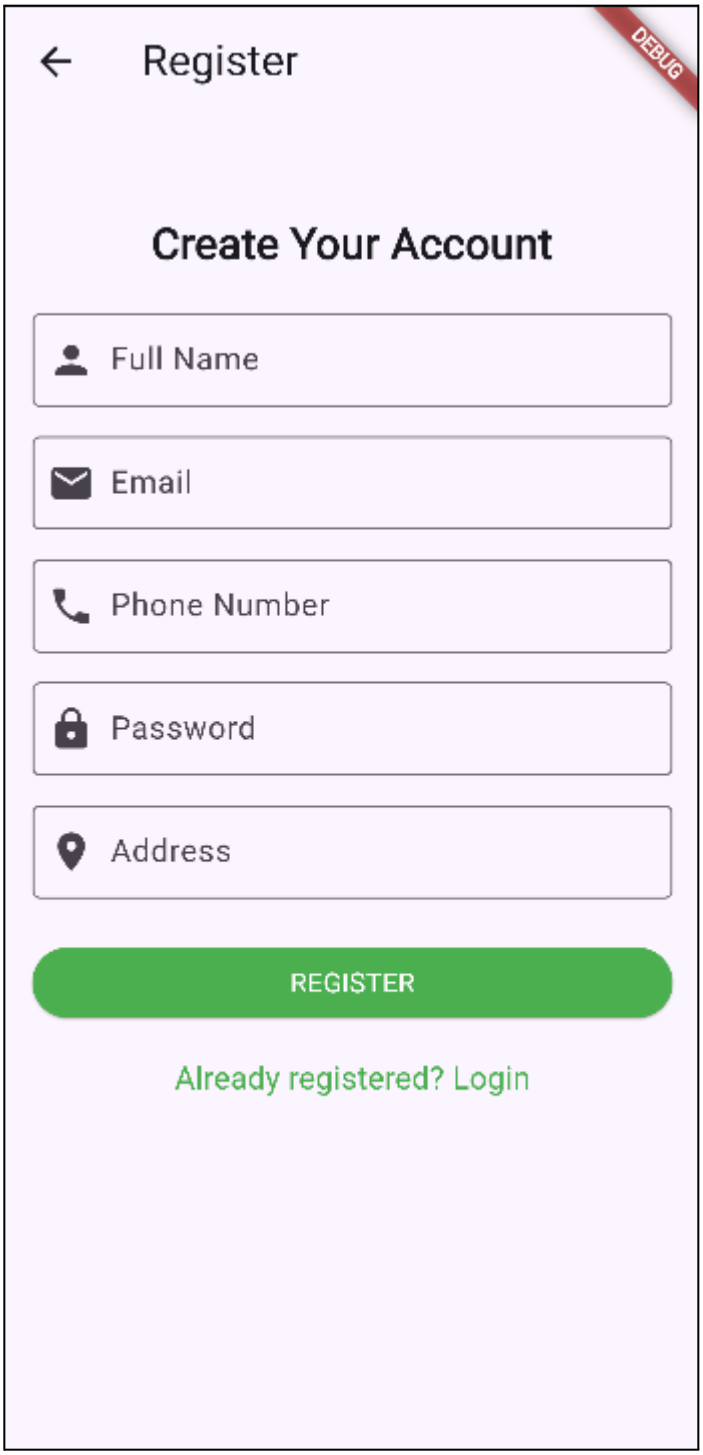
```
  _RegisterScreenState createState() => _RegisterScreenState();
}



class _RegisterScreenState extends State<RegisterScreen> {
  final _formKey = GlobalKey<FormState>();

  // Controllers to store user input
  final TextEditingController _nameController = TextEditingController();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _phoneController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  final TextEditingController _addressController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Register')),
      body: SingleChildScrollView(
        child: Center(
          child: Padding(
            padding: const EdgeInsets.only(top: 50.0, left: 16.0, right: 16.0), // Top padding added
            child: Container(
              constraints: const BoxConstraints(maxWidth: 400), // Limits form width for centering
              child: Form(
                key: _formKey,
                child: Column(
                  mainAxisSize: MainAxisSize.min, // Keeps form compact
                  children: [
                    const Text(
                      'Create Your Account',
                      style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold),
                      textAlign: TextAlign.center,
                    ),
                    const SizedBox(height: 20),

                    // Name Field
                    TextFormField(
                      controller: _nameController,
                      decoration: const InputDecoration(
                        labelText: 'Full Name',
                        border: OutlineInputBorder(),
                        prefixIcon: Icon(Icons.person),
```

```
  ),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your name';
    }
    return null;
  },
),
const SizedBox(height: 15),

// Email Field
TextFormField(
  controller: _emailController,
  keyboardType: TextInputType.emailAddress,
  decoration: const InputDecoration(
    labelText: 'Email',
    border: OutlineInputBorder(),
    prefixIcon: Icon(Icons.email),
  ),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your email';
    }
    if (!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(value)) {
      return 'Enter a valid email';
    }
    return null;
  },
),
const SizedBox(height: 15),

// Phone Number Field
TextFormField(
  controller: _phoneController,
  keyboardType: TextInputType.phone,
  decoration: const InputDecoration(
    labelText: 'Phone Number',
    border: OutlineInputBorder(),
    prefixIcon: Icon(Icons.phone),
  ),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your phone number';
    }
```

```
    if (value.length != 10) {
      return 'Phone number must be 10 digits';
    }
    return null;
  },
),
const SizedBox(height: 15),

// Password Field
TextFormField(
  controller: _passwordController,
  obscureText: true,
  decoration: const InputDecoration(
    labelText: 'Password',
    border: OutlineInputBorder(),
    prefixIcon: Icon(Icons.lock),
  ),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter a password';
    }
    if (value.length < 6) {
      return 'Password must be at least 6 characters long';
    }
    return null;
  },
),
const SizedBox(height: 15),

// Address Field
TextFormField(
  controller: _addressController,
  decoration: const InputDecoration(
    labelText: 'Address',
    border: OutlineInputBorder(),
    prefixIcon: Icon(Icons.location_on),
  ),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your address';
    }
    return null;
  },
),
```

```
                const SizedBox(height: 25),

                // Register Button
                SizedBox(
                  width: double.infinity, // Makes the button take the full width of the parent
                  child: ElevatedButton(
                    onPressed: () {
                      if (_formKey.currentState!.validate()) {
                        ScaffoldMessenger.of(context).showSnackBar(
                          const SnackBar(content: Text('Registration Successful!')),
                        );
                        Navigator.pop(context);
                      }
                    },
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.green,
                      padding: const EdgeInsets.symmetric(vertical: 16),
                    ),
                    child: const Text('REGISTER', style: TextStyle(color: Colors.white)),
                  ),
                ),

                const SizedBox(height: 15),
                // Already Registered? Login Text
                TextButton(
                  onPressed: () {
                    // Navigate to login screen
                    Navigator.pop(context); // Replace with correct navigation if needed
                  },
                  child: const Text(
                    'Already registered? Login',
                    style: TextStyle(fontSize: 16, color: Colors.green),
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
}
```


DEBUG
bigbasket
Food is on it's way
Login
FACEBOOK
GOOGLE
or
Email
Password
Remember me
TERMS & CONDITIONS
LOGIN
New to the platform?
REGISTER HERE


Register
DEBUG
Create Your Account
Full Name
Email
Phone Number
Password
Address
REGISTER
Already registered? Login

Order Form

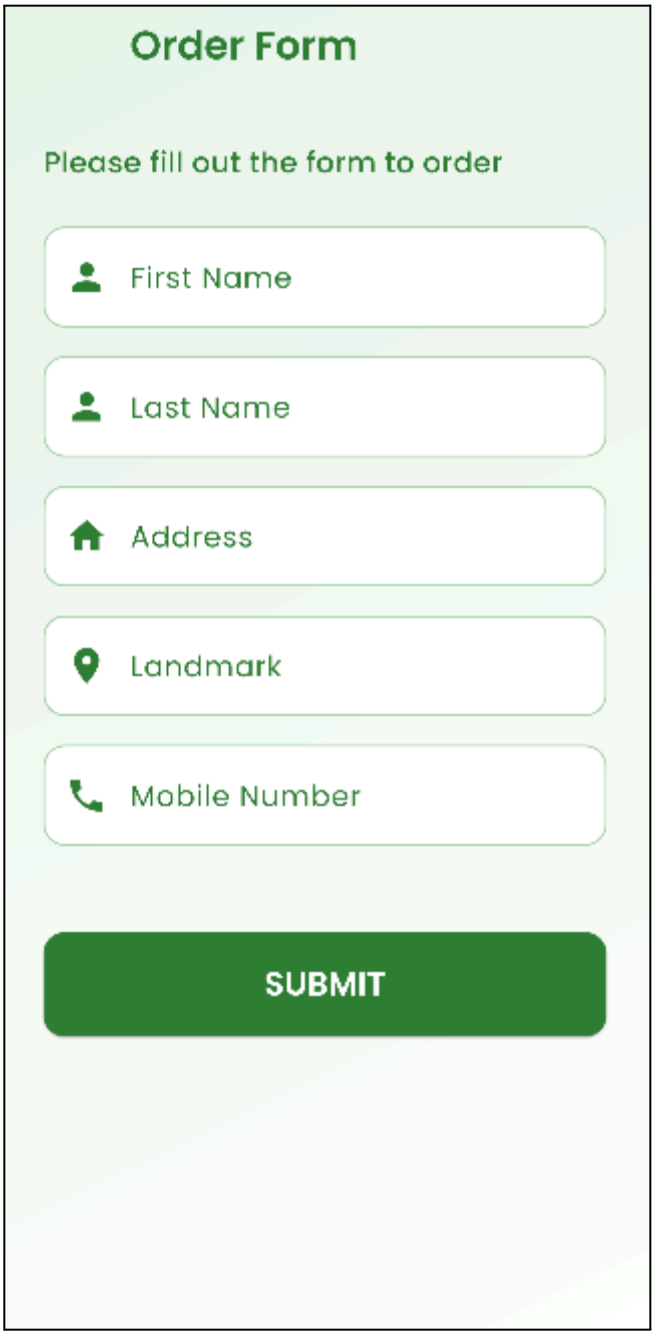
```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        textTheme: GoogleFonts.poppinsTextTheme(Theme.of(context).textTheme),
      ),
      home: const RegistrationForm(),
    );
  }
}

class RegistrationForm extends StatefulWidget {
  const RegistrationForm({Key? key}) : super(key: key);

  @override
  _RegistrationFormState createState() => _RegistrationFormState();
}

class _RegistrationFormState extends State<RegistrationForm> {
final _formKey = GlobalKey<FormState>();
final _controllers = List.generate(5, (_) => TextEditingController());
final _focusNodes = List.generate(5, (_) => FocusNode());

@override
void dispose() {
  for (var controller in _controllers) {
    controller.dispose();
  }
  for (var node in _focusNodes) {
    node.dispose();
  }
```

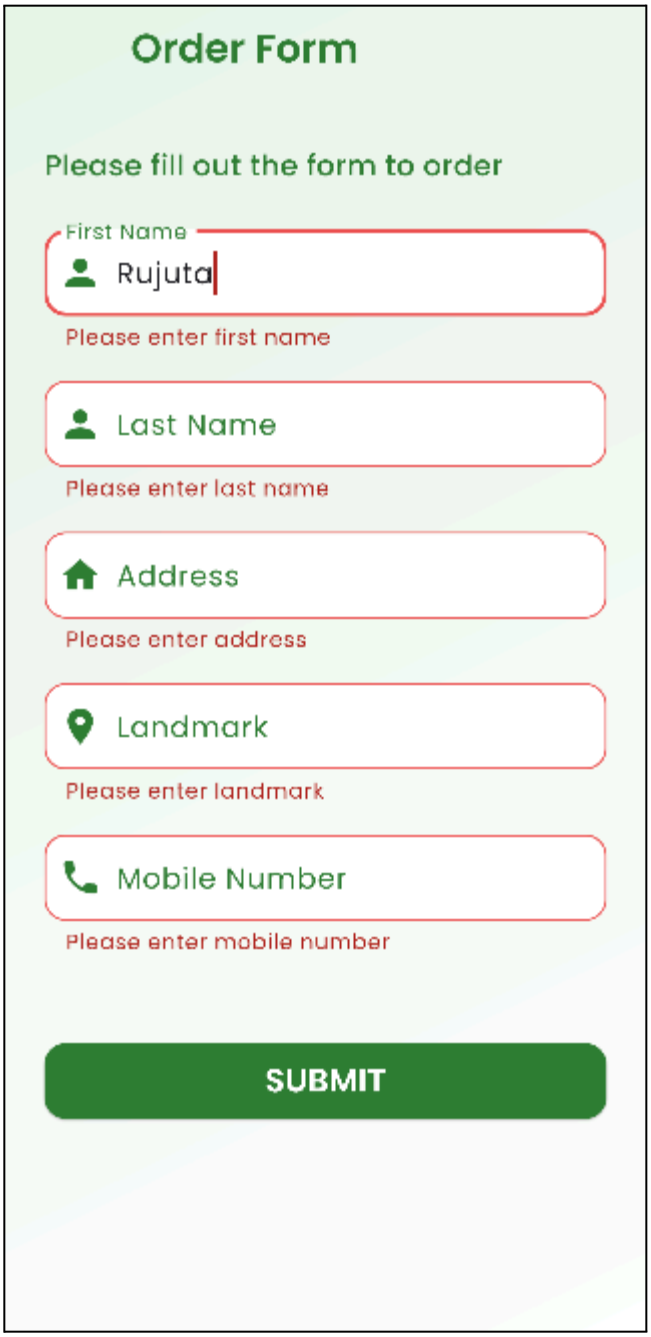
```
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Container(
      decoration: const BoxDecoration(
        gradient: LinearGradient(
          begin: Alignment.topLeft,
          end: Alignment.bottomRight,
          colors: [Color(0xFFE8F5E9), Colors.white],
        ),
      ),
      child: SafeArea(
        child: CustomScrollView(
          slivers: [
            SliverAppBar(
              floating: true,
              backgroundColor: Colors.transparent,
              elevation: 0,
              flexibleSpace: FlexibleSpaceBar(
                title: Text(
                  'Order Form',
                  style: GoogleFonts.poppins(
                    color: Colors.green[800],
                    fontWeight: FontWeight.w600,
                  ),
                ),
              ),
            ),
            SliverToBoxAdapter(
              child: Padding(
                padding: const EdgeInsets.all(24.0),
                child: Form(
                  key: _formKey,
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      Text(
                        'Please fill out the form to order',
                        style: GoogleFonts.poppins(
                          fontSize: 16,
                          color: Colors.green[800],
```

```
                    fontWeight: FontWeight.w500,
                  ),
                ),
                const SizedBox(height: 24),
                ..._buildFormFields(),
                const SizedBox(height: 32),
                _buildSubmitButton(),
              ],
            ),
          ),
        ),
      ),
    ),
  ),
 );
}

List<Widget> _buildFormFields() {
  final labels = ['First Name', 'Last Name', 'Address', 'Landmark', 'Mobile Number'];
  final icons = [Icons.person, Icons.person, Icons.home, Icons.location_on, Icons.phone];

  return List.generate(
    5,
      (index) => Padding(
     padding: const EdgeInsets.only(bottom: 16),
     child: TextFormField(
      controller: _controllers[index],
      focusNode: _focusNodes[index],
      decoration: _getInputDecoration(labels[index], icons[index]),
      validator: (value) {
       if (value == null || value.isEmpty) {
        return 'Please enter ${labels[index].toLowerCase()}';  // Ensures the field is required
       }
       if (index == 4 && value.length != 10) {
        return 'Mobile number must be 10 digits'; // Mobile number validation
       }
       return null;
      },
      onFieldSubmitted: (_) {
       if (index < 4) {
        FocusScope.of(context).requestFocus(_focusNodes[index + 1]);
       }
```

```
        },
        keyboardType: index == 4 ? TextInputType.phone : TextInputType.text,
      ),
    ),
  );
}

InputDecoration _getInputDecoration(String label, IconData icon) {
  return InputDecoration(
    labelText: label,
    prefixIcon: Icon(icon, color: Colors.green[800]),
    labelStyle: TextStyle(color: Colors.green[800]),
    enabledBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.green[200]!),
    ),
    focusedBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.green[800]!, width: 2),
    ),
    errorBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.red[400]!),
    ),
    focusedErrorBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.red[400]!, width: 2),
    ),
    filled: true,
    fillColor: Colors.white,
  );
}

Widget _buildSubmitButton() {
  return SizedBox(
    width: double.infinity,
    child: ElevatedButton(
      onPressed: _submitForm,
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.green[800],
        padding: const EdgeInsets.symmetric(vertical: 16),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(12),
        ),
```

```
      ),
      child: Text(
        'SUBMIT',
        style: GoogleFonts.poppins(
          color: Colors.white,
          fontSize: 18,
          fontWeight: FontWeight.w600,
        ),
      ),
    ),
  );
}

void _submitForm() {
  if (_formKey.currentState!.validate()) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text(
          'Order Successful!',
          style: GoogleFonts.poppins(),
        ),
        backgroundColor: Colors.green[800],
      ),
    );
  }
}
}
```


Order Form
Please fill out the form to order
First Name
Last Name
Address
Landmark
Mobile Number
SUBMIT


Order Form
Please fill out the form to order
First Name
Rujuta
Please enter first name
Last Name
Please enter last name
Address
Please enter address
Landmark
Please enter landmark
Mobile Number
Please enter mobile number
SUBMIT

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	27
Name	Rujuta Medhi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Project Title: BigBasket RollNo. 27

MAD & PWA Lab Journal

Experiment No.	05
Experiment Title.	: To apply navigation, routing and gestures in Flutter App
Roll No.	27
Name	Rujuta Medhi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

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

THEORY:

Navigation, Routing, and Gestures in Flutter

Flutter provides powerful tools for managing **navigation, routing, and gestures** to create seamless user experiences.

1. Navigation in Flutter

Navigation in Flutter refers to moving between different screens (or pages). Flutter uses a **stack-based navigation system**, where each new screen is placed on top of a stack and removed when navigating back.

- **Pushing a new screen** adds it to the stack.
- **Popping a screen** removes it from the stack and returns to the previous one.

Navigation can be managed **imperatively** (by manually pushing and popping screens) or **declaratively** (by defining routes in advance).

2. Routing in Flutter

Routing is a structured way to manage navigation by defining screen paths in advance. There are two main types of routing:

- **Imperative Routing**: Directly navigating to a screen using function calls.
- **Named Routing**: Predefining route names and managing navigation using these names.

Routing helps in organizing the app structure, especially in large applications where multiple screens exist.

3. Gestures in Flutter

Gestures refer to touch-based interactions like **taps, swipes, long presses, and pinches**. Flutter provides a **gesture recognition system** to detect user interactions with widgets.

Common Gestures in Flutter:

- **Tap**: Detects single or double taps on the screen.
- **Swipe**: Identifies left, right, up, or down swipes.
- **Long Press**: Triggers an action when a user holds down on an element.
- **Drag**: Recognizes dragging movements, useful for draggable UI elements.

Gestures enhance user experience by making apps more interactive and intuitive.

PROGRAM:

```
Login and Register
import 'package:flutter/material.dart';
import 'package:bigbasket_website/home.dart'; //
void main() {
  runApp(const MyApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Login Screen',
      theme: ThemeData(
        primarySwatch: Colors.red,
      ),
      home: const LoginScreen(),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SingleChildScrollView(
        child: Column(
          children: [
            // Top image section
            Container(
              height: 250,
              decoration: const BoxDecoration(
                image: DecorationImage(
                  image: NetworkImage(
                      'assets/loginnn.png'),
                  fit: BoxFit.cover,
                ),
              ),
              child: const Padding(
```

```
    padding: EdgeInsets.all(24.0),
    child: Align(
      alignment: Alignment.bottomLeft,

    ),
  ),
),
// Login form section
Padding(
  padding: const EdgeInsets.all(24.0),
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.stretch,
    children: [
      const Text(
        'Login',
        style: TextStyle(
          fontSize: 24,
          fontWeight: FontWeight.bold,
        ),
      ),
      const SizedBox(height: 24),
      // Social login buttons
      Row(
        children: [
          Expanded(
            child: ElevatedButton.icon(
              onPressed: () {},
              icon: const Icon(Icons.facebook, color: Colors.white),
              label: const Text('FACEBOOK',
                  style: TextStyle(color: Colors.white)),
              style: ElevatedButton.styleFrom(
                backgroundColor: const Color(0xFF3B5998),
                padding: const EdgeInsets.symmetric(vertical: 12),
              ),
            ),
          ),
          const SizedBox(width: 16),
          Expanded(
            child: OutlinedButton.icon(
              onPressed: () {},
              icon: const Icon(Icons.g_mobiledata,
                  color: Colors.black87),
              label: const Text('GOOGLE',
                  style: TextStyle(color: Colors.black87)),
```

```
        style: OutlinedButton.styleFrom(
          padding: const EdgeInsets.symmetric(vertical: 12),
        ),
      ),
    ),
  ],
),
const SizedBox(height: 24),
const Text(
  'or',
  textAlign: TextAlign.center,
  style: TextStyle(color: Colors.grey),
),
const SizedBox(height: 24),
// Email field
TextField(
  decoration: InputDecoration(
    hintText: 'Email',
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(4),
    ),
  ),
),
const SizedBox(height: 16),
// Password field
TextField(
  obscureText: true,
  decoration: InputDecoration(
    hintText: 'Password',
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(4),
    ),
  ),
),
const SizedBox(height: 16),
// Remember me checkbox
Row(
  children: [
    Checkbox(
      value: false,
      onChanged: (bool? value) {},
    ),
    const Text('Remember me'),
    const Spacer(),
```

```
              TextButton(
                onPressed: () {},
                child: const Text('TERMS & CONDITIONS',
                    style: TextStyle(color: Colors.green)),
              ),
            ],
          ),
          const SizedBox(height: 24),
          // Login button
          ElevatedButton(
            onPressed: () {
              Navigator.pushReplacement(
                context,
                MaterialPageRoute(builder: (context) => const HomePage()), // Navigate to
HomePage
              );
            },
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.green,
              padding: const EdgeInsets.symmetric(vertical: 16),
            ),
            child: const Text('LOGIN', style: TextStyle(color: Colors.white)),
          ),

          const SizedBox(height: 24),
          // Bottom text
          const Text(
            'New to the platform?',
            textAlign: TextAlign.center,
          ),
          TextButton(
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => const RegisterScreen()),
              );
            },
            child: const Text('REGISTER HERE', style: TextStyle(color: Colors.green)),
          ),

        ],
      ),
    ),
  ],
```

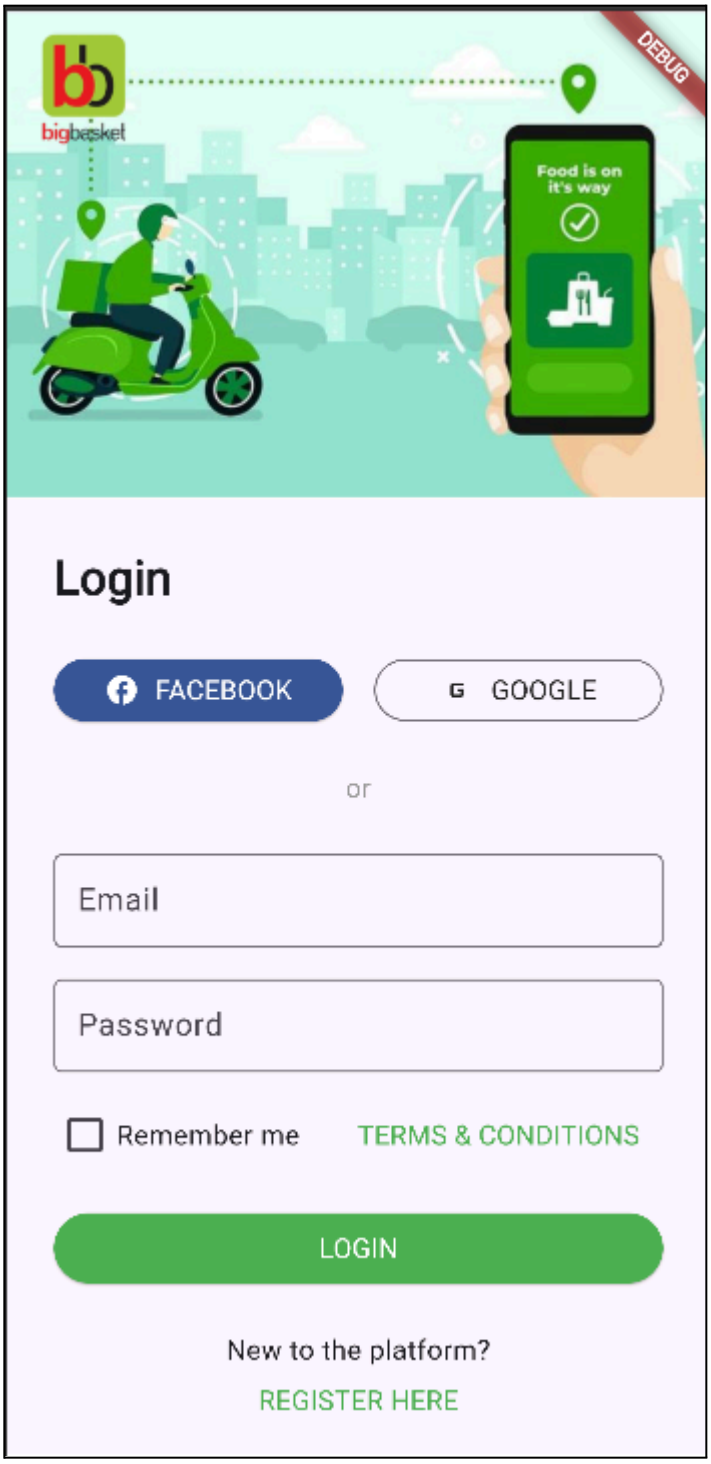
```
        ),
      ),
    );
  }
}

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

  @override
  _RegisterScreenState createState() => _RegisterScreenState();
}



class _RegisterScreenState extends State<RegisterScreen> {
  final _formKey = GlobalKey<FormState>();

  // Controllers to store user input
  final TextEditingController _nameController = TextEditingController();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _phoneController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  final TextEditingController _addressController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Register')),
      body: SingleChildScrollView(
        child: Center(
          child: Padding(
            padding: const EdgeInsets.only(top: 50.0, left: 16.0, right: 16.0), // Top padding added
            child: Container(
              constraints: const BoxConstraints(maxWidth: 400), // Limits form width for centering
              child: Form(
                key: _formKey,
                child: Column(
                  mainAxisSize: MainAxisSize.min, // Keeps form compact
                  children: [
                    const Text(
                      'Create Your Account',
                      style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold),
                      textAlign: TextAlign.center,
```

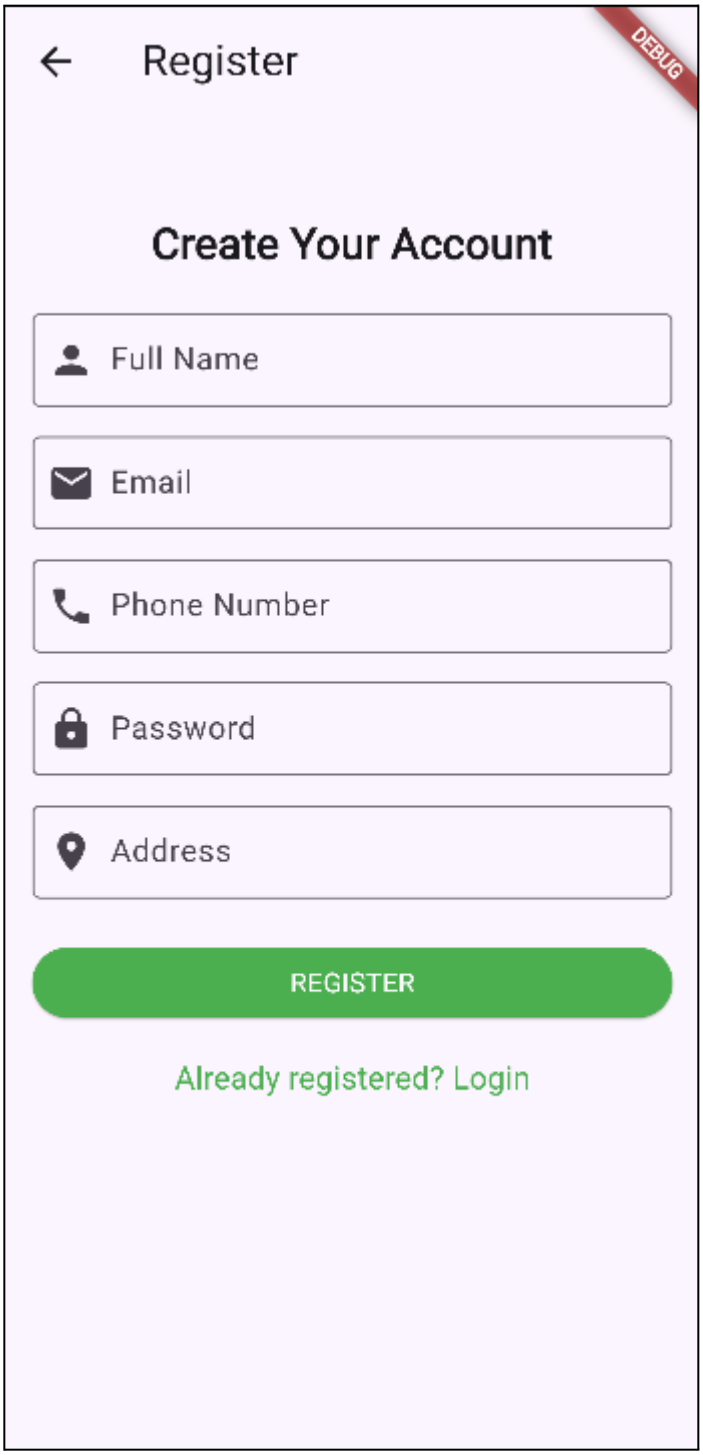
```
),
const SizedBox(height: 20),

// Name Field
TextFormField(
  controller: _nameController,
  decoration: const InputDecoration(
    labelText: 'Full Name',
    border: OutlineInputBorder(),
    prefixIcon: Icon(Icons.person),
  ),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your name';
    }
    return null;
  },
),
const SizedBox(height: 15),

// Email Field
TextFormField(
  controller: _emailController,
  keyboardType: TextInputType.emailAddress,
  decoration: const InputDecoration(
    labelText: 'Email',
    border: OutlineInputBorder(),
    prefixIcon: Icon(Icons.email),
  ),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your email';
    }
    if (!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(value)) {
      return 'Enter a valid email';
    }
    return null;
  },
),
const SizedBox(height: 15),

// Phone Number Field
TextFormField(
  controller: _phoneController,
```

```
  keyboardType: TextInputType.phone,
  decoration: const InputDecoration(
    labelText: 'Phone Number',
    border: OutlineInputBorder(),
    prefixIcon: Icon(Icons.phone),
  ),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your phone number';
    }
    if (value.length != 10) {
      return 'Phone number must be 10 digits';
    }
    return null;
  },
),
const SizedBox(height: 15),

// Password Field
TextFormField(
  controller: _passwordController,
  obscureText: true,
  decoration: const InputDecoration(
    labelText: 'Password',
    border: OutlineInputBorder(),
    prefixIcon: Icon(Icons.lock),
  ),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter a password';
    }
    if (value.length < 6) {
      return 'Password must be at least 6 characters long';
    }
    return null;
  },
),
const SizedBox(height: 15),

// Address Field
TextFormField(
  controller: _addressController,
  decoration: const InputDecoration(
    labelText: 'Address',
```

```
    border: OutlineInputBorder(),
    prefixIcon: Icon(Icons.location_on),
   ),
   validator: (value) {
    if (value == null || value.isEmpty) {
     return 'Please enter your address';
    }
    return null;
   },
  ),
  const SizedBox(height: 25),

  // Register Button
  SizedBox(
   width: double.infinity, // Makes the button take the full width of the parent
   child: ElevatedButton(
    onPressed: () {
     if (_formKey.currentState!.validate()) {
      ScaffoldMessenger.of(context).showSnackBar(
       const SnackBar(content: Text('Registration Successful!')),
      );
      Navigator.pop(context);
     }
    },
    style: ElevatedButton.styleFrom(
     backgroundColor: Colors.green,
     padding: const EdgeInsets.symmetric(vertical: 16),
    ),
    child: const Text('REGISTER', style: TextStyle(color: Colors.white)),
   ),
  ),

  const SizedBox(height: 15),
  // Already Registered? Login Text
  TextButton(
   onPressed: () {
    // Navigate to login screen
    Navigator.pop(context); // Replace with correct navigation if needed
   },
   child: const Text(
    'Already registered? Login',
    style: TextStyle(fontSize: 16, color: Colors.green),
   ),
  ),
```

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


DEBUG
bigbasket
Food is on it's way
Login
FACEBOOK
GOOGLE
or
Email
Password
Remember me
TERMS & CONDITIONS
LOGIN
New to the platform?
REGISTER HERE



Home.dart

```
import 'package:flutter/material.dart';
import 'dishes.dart';
import 'icecreamprod.dart';
class HomeScreen extends StatelessWidget {
  const HomeScreen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
      backgroundColor: Colors.white,
      appBar: AppBar(
        backgroundColor: Color(0xFF1E8040),
        elevation: 0,
        title: Column(
          mainAxisSize: MainAxisSize.min,  // Prevents Column from expanding beyond AppBar
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Flexible(  // Prevents overflow
              child: Image.network(

'https://hebbkx1anhila5yf.public.blob.vercel-storage.com/WhatsApp%20Image%202025-02-03%20at
%207.18.38%20PM%20(1)-5NeEZVgjFEJKRwg1Bk7oHBqBwMGcdv.jpeg',
                height: 40,
                fit: BoxFit.contain,  // Ensures image does not expand unexpectedly
              ),
            ),
            Row(
              children: [
                Icon(Icons.location_on, color: Colors.white, size: 20),
                Expanded( // Ensures text doesn't overflow
                  child: Text(
                    'Deliver to Selected Location',
                    style: TextStyle(fontSize: 14, color: Colors.white),
                    overflow: TextOverflow.ellipsis,
                  ),
                ),
                Icon(Icons.keyboard_arrow_down, color: Colors.white),
              ],
            ),
            Text(
              'Mahadeo Wadi, Mumbai - 400071',
              style: TextStyle(fontSize: 12, color: Colors.white70),
              overflow: TextOverflow.ellipsis,
            ),
          ],
        ),

        actions: [
          Container(
            padding: EdgeInsets.all(8),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text(
                  'Delivers in',
```

```
            style: TextStyle(fontSize: 12, color: Colors.white70),
          ),
          Container(
            padding: EdgeInsets.symmetric(horizontal: 8, vertical: 4),
            decoration: BoxDecoration(
              color: Colors.white,
              borderRadius: BorderRadius.circular(4),
            ),
            child: Row(
              children: [
                Icon(Icons.flash_on, size: 16, color: Colors.black),
                Text(
                  '6 mins',
                  style: TextStyle(color: Colors.black, fontSize: 12),
                ),
              ],
            ),
          ),
        ],
      ),
    ),
    IconButton(
      icon: Icon(Icons.person_outline),
      onPressed: () {},
    ),
  ],
),
body: SingleChildScrollView(
  child: Column(
    children: [
      // Search Bar
      Padding(
        padding: EdgeInsets.all(16),
        child: TextField(
          decoration: InputDecoration(
            hintText: 'Search 20000+ products',
            prefixIcon: Icon(Icons.search),
            suffixIcon: Icon(Icons.mic),
            border: OutlineInputBorder(
              borderRadius: BorderRadius.circular(8),
            ),
            filled: true,
            fillColor: Colors.white,
          ),
        ),
      ),
```

```
// Sale Banner
Container(
  padding: EdgeInsets.all(16),
  color: Color(0xFFFF6B4A),
  child: Column(
    children: [
      Text(
        'BIG INDIAN',
        style: TextStyle(
          color: Colors.white,
          fontSize: 24,
          fontWeight: FontWeight.bold,
        ),
      ),
      Text(
        'GROCERY SALE',
        style: TextStyle(
          color: Colors.white,
          fontSize: 32,
          fontWeight: FontWeight.bold,
        ),
      ),
      Text(
        '1st to 9th Feb',
        style: TextStyle(color: Colors.white),
      ),
    ],
  ),
),

// Categories Grid
GridView.count(
  shrinkWrap: true,
  physics: NeverScrollableScrollPhysics(),
  crossAxisCount: 2,
  padding: EdgeInsets.all(16),
  mainAxisSpacing: 16,
  crossAxisSpacing: 16,
  children: [
    _buildCategoryCard(
      'Icecream',
      'assets/sunday.jpg',
      context, // Pass context
    ),
    _buildCategoryCard(
```

```
          'Cleaning &\nhousehold',
          'assets/household.png',
          context,
        ),
        _buildCategoryCard(
          'Lays',
          'assets/lays.jpg',
          context,
        ),
        _buildCategoryCard(
          'Beauty &\nhygiene',
          'assets/cosmetics.png',
          context,
        ),
        _buildCategoryCard(
          'Kitchen, garden\n& pets',
          'assets/pet.png',
          context,
        ),
        _buildCategoryCard(
          'Electronic &\napparel',
          'assets/electronics.png',
          context,
        ),
      ],
    ),


    // Bottom Navigation Bar
    Container(
      padding: EdgeInsets.symmetric(vertical: 8),
      decoration: BoxDecoration(
        border: Border(top: BorderSide(color: Colors.grey.shade300)),
      ),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceAround,
        children: [
          _buildNavItem(Icons.home, 'Home', true, context),
          _buildNavItem(Icons.grid_view, 'Dishes', false, context),
          _buildNavItem(Icons.refresh, 'Re-Order', false, context),
        ],
      ),

    ),
  ],
),
```

```
    ),
  );
}

Widget _buildCategoryCard(String title, String imageAsset, BuildContext context) {
  return GestureDetector(
    onTap: () {
      if (title == 'Icecream') {
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => IceCreamShop()),
        );
      }
    },
    child: Container(
      decoration: BoxDecoration(
        color: Color(0xFFFFF3E0),
        borderRadius: BorderRadius.circular(8),
      ),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Image.asset(imageAsset, height: 80),
          SizedBox(height: 8),
          Text(
            title,
            textAlign: TextAlign.center,
            style: TextStyle(
              fontSize: 16,
              fontWeight: FontWeight.bold,
            ),
          ),
        ],
      ),
    ),
  );
}


Widget _buildNavItem(IconData icon, String label, bool isSelected, BuildContext context) {
  return GestureDetector(
    onTap: () {
      if (label == 'Dishes') {
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => HomePage()),
```

```
        );
      }
    },
    child: Column(
      mainAxisSize: MainAxisSize.min,
      children: [
        Icon(icon, color: isSelected ? Colors.green[800] : Colors.grey),
        Text(label, style: TextStyle(color: isSelected ? Colors.green[800] : Colors.grey)),
      ],
    ),
  );
 }

}
```

Deliver to Selected Loc...

Delivers in

6 mins

BOTTOM OVERFLOWED BY 2.0 PIXELS

DEBUG

GROCERY SALE

1st to 9th Feb

Icecream

Cleaning & household

Lays

Beauty & hygiene

Kitchen, garden & pets

Electronic & apparel

Home

Dishes

Re-Order

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	27
Name	Rujuta Medhi
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:	

Project Title: BigBasket RollNo.27

MAD & PWA Lab Journal

Experiment No.	06
Experiment Title.	To connect flutter UI with firebase database.
Roll No.	27
Name	Rujuta Medhi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

AIM : To connect flutter UI with firebase database.

THEORY:

Introduction to Firebase and Flutter Integration

Firebase is a comprehensive platform developed by Google, designed to help developers build high-quality applications for both mobile and web. It provides essential services such as real-time databases, authentication, cloud storage, hosting, and much more. One of the most widely used Firebase services is the Firebase Realtime Database, which is a NoSQL cloud database that allows data to be stored and synced in real-time across all connected devices. Flutter, on the other hand, is an open-source UI software development kit created by Google, which allows developers to build natively compiled applications for mobile, web, and desktop from a single codebase. Its rich set of pre-designed widgets and powerful tools makes Flutter an attractive option for developing visually appealing and performant applications. Integrating Firebase with Flutter allows developers to leverage the full potential of Firebase services in their applications. By using Firebase's Realtime Database, Flutter apps can achieve features such as real-time data synchronization, secure authentication, and cloud-based storage. This combination enables developers to create powerful, scalable, and feature-rich mobile and web applications.

Setting Up Firebase in Flutter :

To connect a Flutter app with Firebase, the following steps are typically followed:

1. Creating a Firebase Project: To start using Firebase with Flutter, the first step is to create a Firebase project in the Firebase Console. Once the project is created, developers can associate their Flutter app with the Firebase project by following the platform-specific instructions for Android or iOS. This usually involves configuring API keys, downloading configuration files, and adding them to the Flutter project.

2. Integrating Firebase SDK in Flutter: After the Firebase project is set up, developers need to integrate Firebase's SDK into the Flutter app. This involves adding the necessary dependencies to the Flutter project's pubspec.yaml file. For Firebase's Realtime Database, the package firebase_database is used. Additionally, Firebase's core SDK (firebase_core) must also be included to initialize Firebase services.

3. Initializing Firebase: Before any Firebase functionality can be used, it is essential to initialize Firebase in the Flutter app. This is done by calling Firebase.initializeApp() in the main entry point of the app (usually in the main.dart file). Firebase needs to be initialized before interacting with any Firebase services, such as the Realtime Database, Cloud Firestore, or Authentication. Connecting Firebase to a Flutter app enables developers to create robust, scalable, and real-time applications with ease. Firebase's Realtime Database offers a powerful, cloud-based solution for managing data in realtime, while Firebase Authentication ensures secure access control. By integrating Firebase with Flutter, developers can take advantage of

real-time data synchronization, offline support, and a wide range of other Firebase features, allowing them to build feature-rich apps that meet modern user expectations.

PROGRAM:

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'firebase_options.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'home.dart';
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  try {
    await Firebase.initializeApp(
      options: DefaultFirebaseOptions.currentPlatform,
    );
    print("✅ Firebase Connection Successful");
  } catch (e) {
    print("❌ Firebase Initialization Failed: $e");
  }
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Login Screen',
      theme: ThemeData(primarySwatch: Colors.red),
      home: const LoginScreen(),
    );
  }
}



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

  @override
```

```
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  final FirebaseAuth _auth = FirebaseAuth.instance;
  bool _isLoading = false;

  void _loginUser() async {
    setState(() {
      _isLoading = true;
    });

    try {
      UserCredential userCredential = await _auth.signInWithEmailAndPassword(
        email: _emailController.text.trim(),
        password: _passwordController.text.trim(),
      );

      User? user = userCredential.user;

      if (user != null) {
        // Check if user exists in Firestore
        DocumentSnapshot userDoc = await FirebaseFirestore.instance
            .collection("users")
            .doc(user.uid)
            .get();

        if (userDoc.exists) {
          // Navigate to HomeScreen after successful login
          Navigator.pushReplacement(
            context,
            MaterialPageRoute(builder: (context) => const HomeScreen()), // Ensure HomeScreen
is correct
          );
        } else {
          // User not found in Firestore (unlikely), show error
          ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(content: Text("User record not found.")),
          );
        }
      }
    } on FirebaseAuthException catch (e) {
```

```
    String errorMessage = "Login failed. Please try again.";

    if (e.code == 'user-not-found') {
      errorMessage = "No user found for this email.";
    } else if (e.code == 'wrong-password') {
      errorMessage = "Wrong password provided.";
    }

    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text(errorMessage)),
    );
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text("An error occurred: $e")),
    );
  } finally {
    setState(() {
      _isLoading = false;
    });
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: SingleChildScrollView(
      child: Column(
        children: [
          Container(
            height: 250,
            decoration: const BoxDecoration(
              image: DecorationImage(
                image: AssetImage('assets/loginnn.png'),
                fit: BoxFit.cover,
              ),
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(24.0),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.stretch,
              children: [
                const Text('Login',
                    style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold)),
```

```
const SizedBox(height: 24),
Row(
  children: [
    Expanded(
      child: ElevatedButton.icon(
        onPressed: () {},
        icon: const Icon(Icons.facebook, color: Colors.white),
        label: const Text('FACEBOOK',
            style: TextStyle(color: Colors.white)),
        style: ElevatedButton.styleFrom(
          backgroundColor: const Color(0xFF3B5998),
        ),
      ),
    ),
    const SizedBox(width: 16),
    Expanded(
      child: OutlinedButton.icon(
        onPressed: () {},
        icon: const Icon(Icons.g_mobiledata, color: Colors.black87),
        label: const Text('GOOGLE',
            style: TextStyle(color: Colors.black87)),
      ),
    ),
  ],
),
const SizedBox(height: 24),
TextField(
  controller: _emailController,
  decoration: const InputDecoration(
    hintText: 'Email',
    border: OutlineInputBorder(),
  ),
),
const SizedBox(height: 16),
TextField(
  controller: _passwordController,
  obscureText: true,
  decoration: const InputDecoration(
    hintText: 'Password',
    border: OutlineInputBorder(),
  ),
),
const SizedBox(height: 24),
ElevatedButton(
```

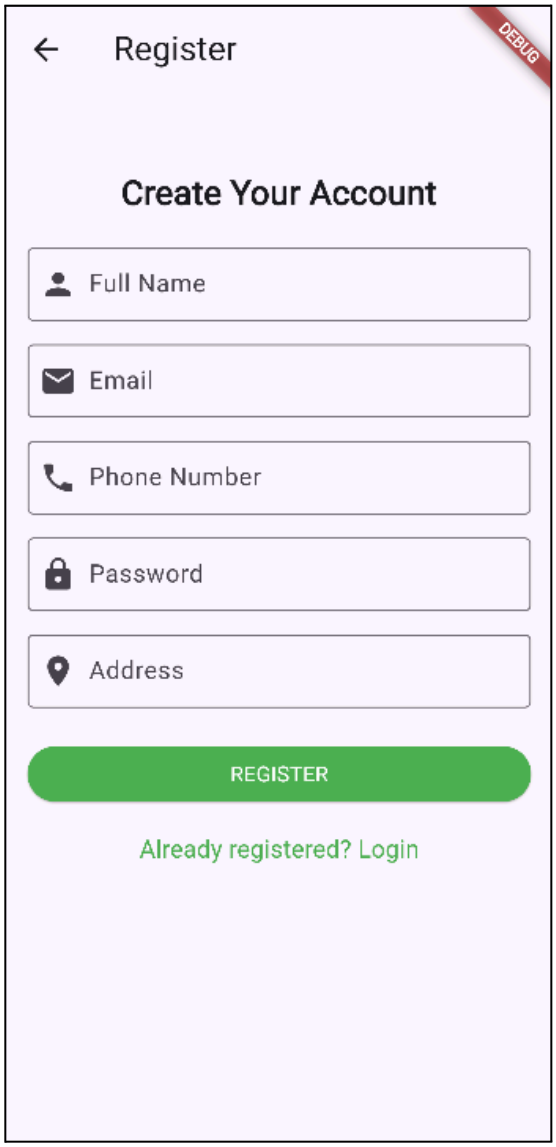
```
                onPressed: _isLoading ? null : _loginUser,
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.green,
                ),
                child: _isLoading
                    ? const CircularProgressIndicator(color: Colors.white)
                    : const Text('LOGIN', style: TextStyle(color: Colors.white)),
              ),
              const SizedBox(height: 24),
              TextButton(
                onPressed: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(builder: (context) => const RegisterScreen()),
                  );
                },
                child: const Text('REGISTER HERE', style: TextStyle(color: Colors.green)),
              ),
            ],
          ),
        ),
      ),
    ],
  ),
 ),
 );
 }
}
```

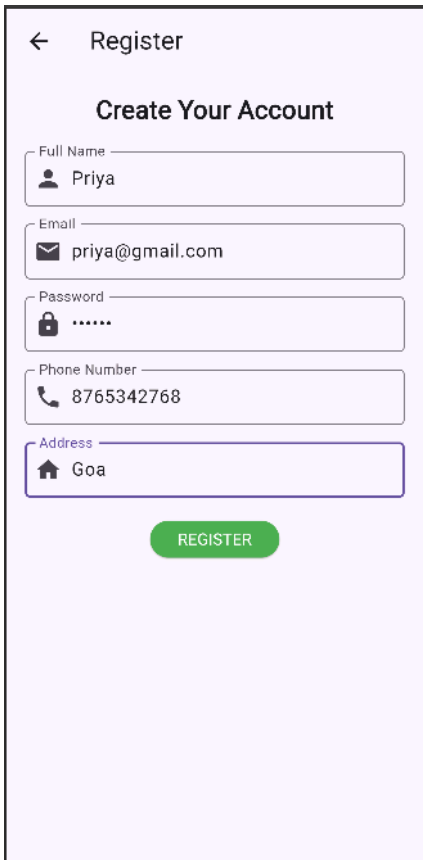
```
class RegisterScreen extends StatefulWidget {
 const RegisterScreen({super.key});

 @override
 _RegisterScreenState createState() => _RegisterScreenState();
}

class _RegisterScreenState extends State<RegisterScreen> {
 final _formKey = GlobalKey<FormState>();
 final TextEditingController _nameController = TextEditingController();
 final TextEditingController _emailController = TextEditingController();
 final TextEditingController _phoneController = TextEditingController();
 final TextEditingController _passwordController = TextEditingController();
```

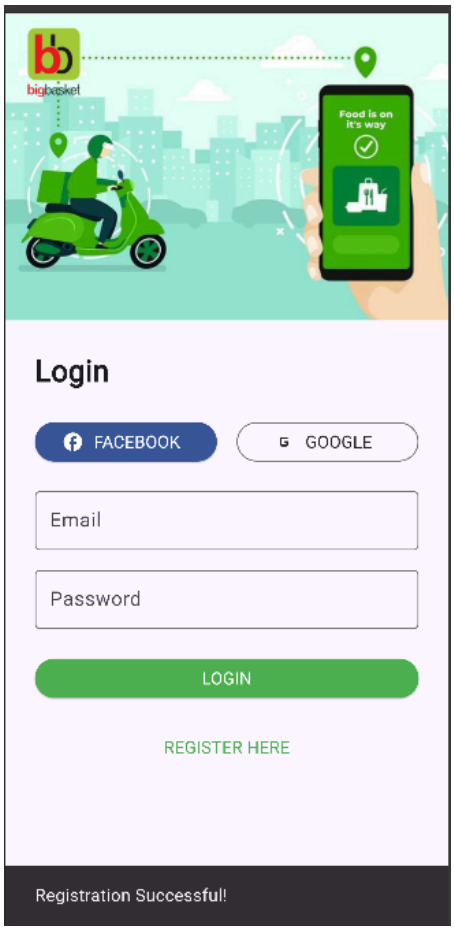
```
  final TextEditingController _addressController = TextEditingController();

  // Function to register user
  void registerUser() async {
    if (_formKey.currentState!.validate()) {
      try {
        // Register user in Firebase Authentication
        UserCredential userCredential = await
FirebaseAuth.instance.createUserWithEmailAndPassword(
          email: _emailController.text.trim(),
          password: _passwordController.text.trim(),
        );

        // Get the user ID
        String userId = userCredential.user!.uid;

        // Store user details in Firestore
        await FirebaseFirestore.instance.collection("users").doc(userId).set({
          "name": _nameController.text.trim(),
          "email": _emailController.text.trim(),
          "phone": _phoneController.text.trim(),
          "address": _addressController.text.trim(),
          "timestamp": FieldValue.serverTimestamp(),
        });

        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(content: Text('Registration Successful!')),
        );

        Navigator.pop(context); // Go back to Login Screen
      } catch (e) {
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text('Registration Failed: $e')),
        );
      }
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Register')),
      body: SingleChildScrollView(
        child: Padding(
```

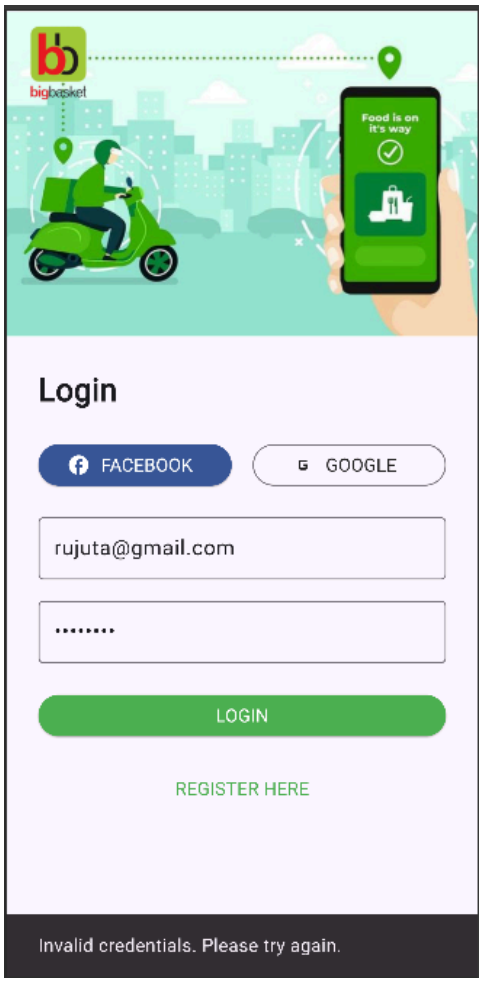
```
padding: const EdgeInsets.all(16.0),
child: Form(
  key: _formKey,
  child: Column(
    children: [
      const Text('Create Your Account',
          style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold)),
      const SizedBox(height: 20),
      TextFormField(
        controller: _nameController,
        decoration: const InputDecoration(
          labelText: 'Full Name',
          border: OutlineInputBorder(),
          prefixIcon: Icon(Icons.person),
        ),
        validator: (value) =>
        value!.isEmpty ? 'Please enter your name' : null,
      ),
      const SizedBox(height: 15),
      TextFormField(
        controller: _emailController,
        decoration: const InputDecoration(
          labelText: 'Email',
          border: OutlineInputBorder(),
          prefixIcon: Icon(Icons.email),
        ),
        validator: (value) =>
        !RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(value!)
            ? 'Enter a valid email'
            : null,
      ),
      const SizedBox(height: 15),
      TextFormField(
        controller: _passwordController,
        obscureText: true,
        decoration: const InputDecoration(
          labelText: 'Password',
          border: OutlineInputBorder(),
          prefixIcon: Icon(Icons.lock),
        ),
        validator: (value) =>
        value!.length < 6 ? 'Password must be at least 6 characters' : null,
      ),
      const SizedBox(height: 15),
```

```
                TextFormField(
                  controller: _phoneController,
                  decoration: const InputDecoration(
                    labelText: 'Phone Number',
                    border: OutlineInputBorder(),
                    prefixIcon: Icon(Icons.phone),
                  ),
                  validator: (value) =>
                  value!.isEmpty ? 'Please enter your phone number' : null,
                ),
                const SizedBox(height: 15),
                TextFormField(
                  controller: _addressController,
                  decoration: const InputDecoration(
                    labelText: 'Address',
                    border: OutlineInputBorder(),
                    prefixIcon: Icon(Icons.home),
                  ),
                  validator: (value) =>
                  value!.isEmpty ? 'Please enter your address' : null,
                ),
                const SizedBox(height: 20),
                ElevatedButton(
                  onPressed: registerUser, // Call registerUser function
                  style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
                  child: const Text('REGISTER', style: TextStyle(color: Colors.white)),
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
}
```


Register
DEBUG
Create Your Account
Full Name
Email
Phone Number
Password
Address
REGISTER
Already registered? Login


Register
Create Your Account
Full Name
Priya
Email
priya@gmail.com
Password
Phone Number
8765342768
Address
Goa
REGISTER


bigbasket
Food is on it's way
Login
FACEBOOK
GOOGLE
Email
Password
LOGIN
REGISTER HERE
Registration Successful!


bigbasket
Food is on it's way
Login
FACEBOOK
GOOGLE
rujuta@gmail.com
LOGIN
REGISTER HERE
Invalid credentials. Please try again.

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.	27
Name	Rujuta Medhi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

Rujuta Medhi **D15A-27**

Experiment No. 7

To write meta data of your Ecommerce PWA

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

Theory:-

Regular Web App

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

Progressive Web App

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

Difference between PWAs vs. Regular Web Apps:

A Progressive Web is different and better than a Regular Web app with features like:

1. Native Experience

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

2. Ease of Access

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

3. Faster Services

PWAs can cache the data and serve the user with text stylesheets, images, and other web content even before the page loads completely. This lowers the waiting time for the end-users and helps

the brands improve the user engagement and retention rate, which eventually adds value to their business.

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

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

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

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

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

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

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

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

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

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

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

Linkable — Easily shared via URL without complex installations.

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

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

Code:-

```
var staticCacheName = "eyojana1";

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

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

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

manifest.json:-

```
{
 "short_name": "React App",
 "name": "Create React App Sample",
 "icons": [
   {
    "src": "favicon.ico",
    "sizes": "64x64",
    "type": "image/x-icon"
   },
   {
    "src": "logo1.png",
    "type": "image/png",
    "sizes": "192x192"
   },
   {
    "src": "logo2.png",
    "type": "image/png",
    "sizes": "512x512"
   }
 ],
 "start_url": ".",
```

```
 "display": "standalone",
 "theme_color": "#000000",
 "background_color": "#ffffff"
}
```

Add the link tag to link to the manifest.json file

```
<html>
   <head>
      <link rel="manifest" href="manifest.json">
      <script src="myscript.js"></script>
      <title>Eyojana</title>
      <style>
```

Output:-

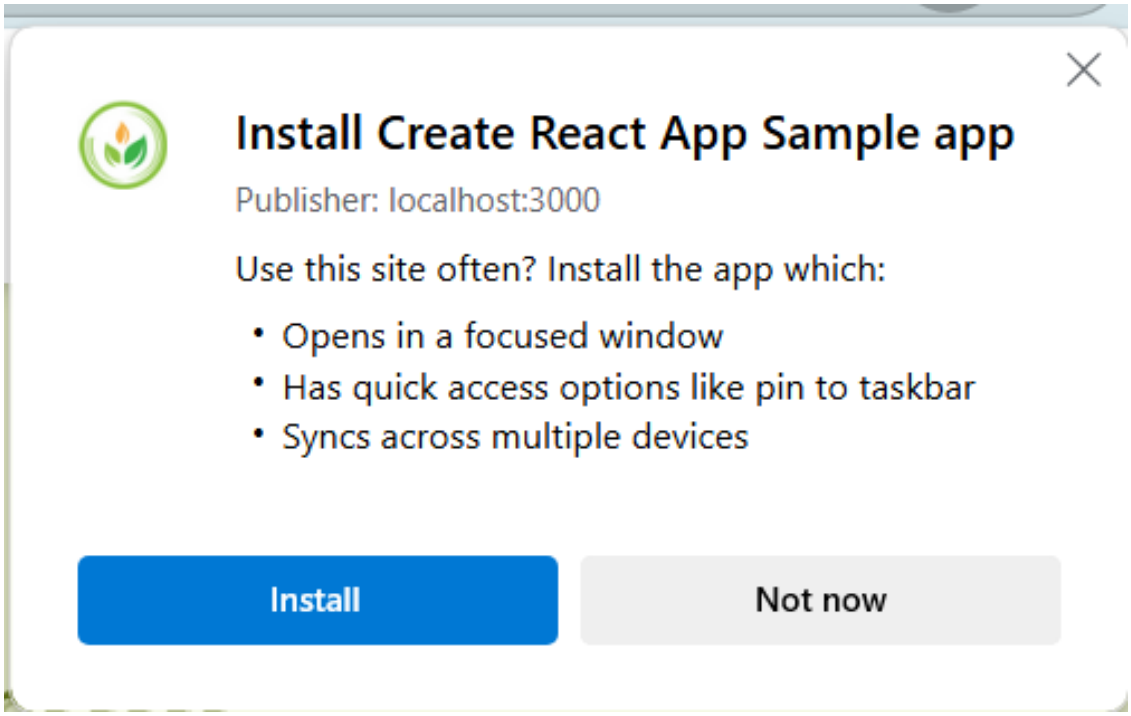
Install nodejs

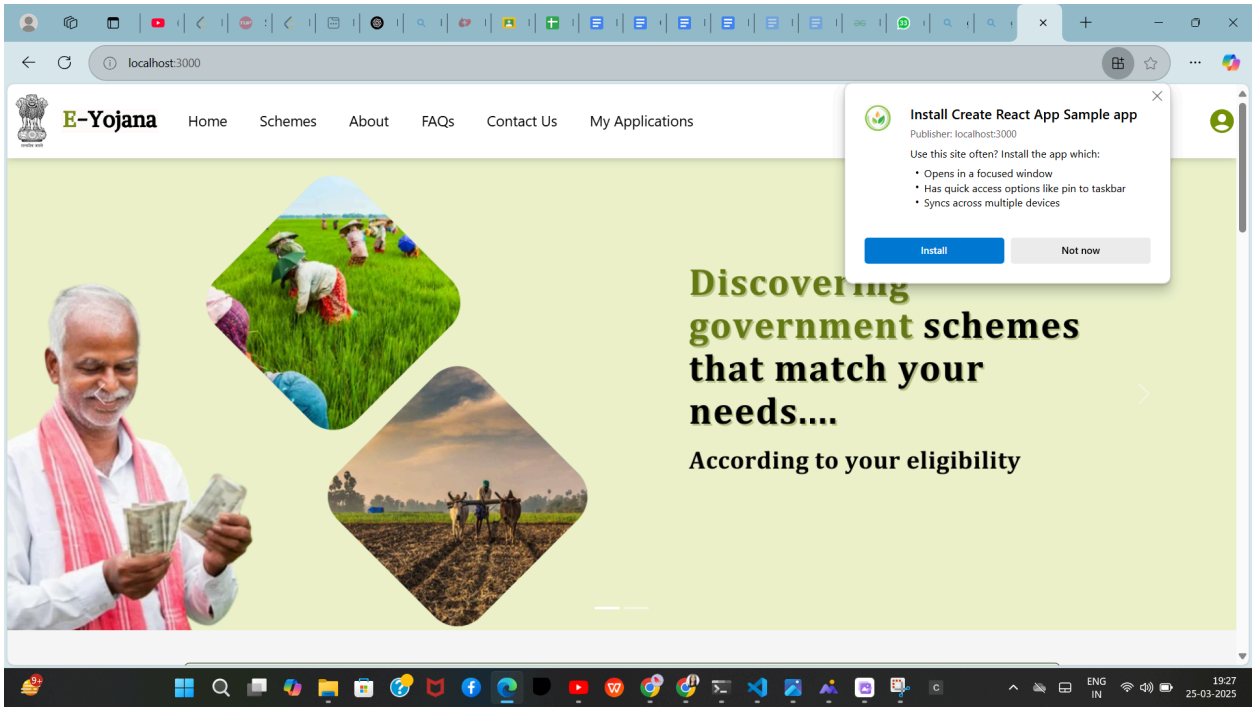
https://nodejs.org/en/download/

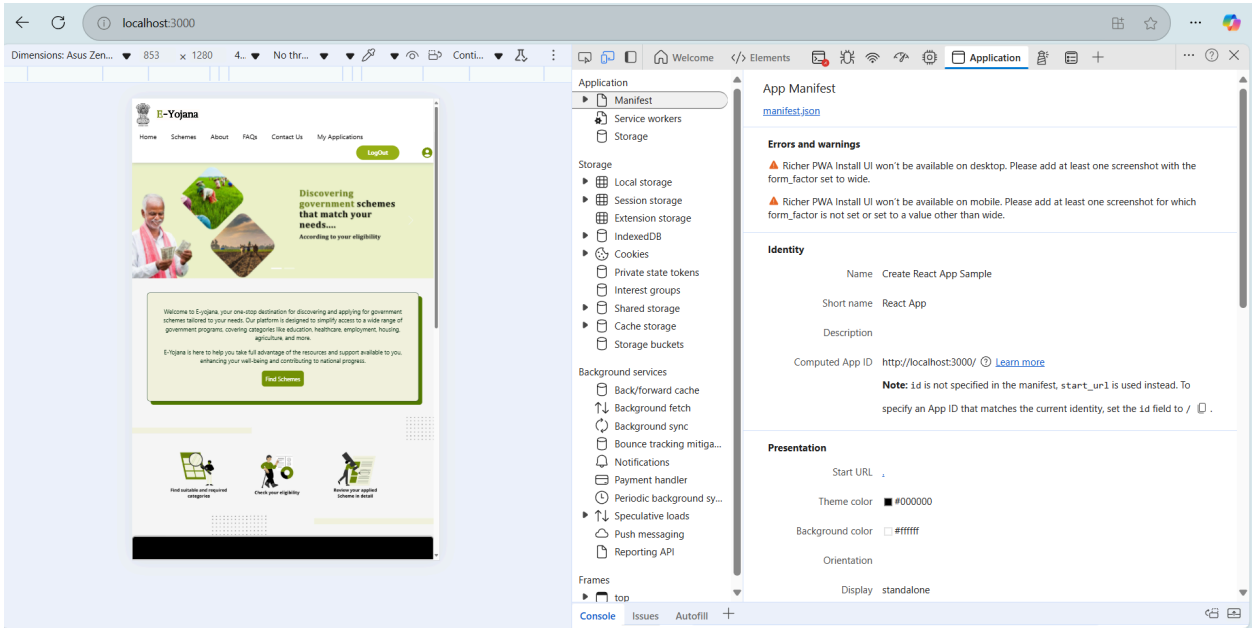
In cmd

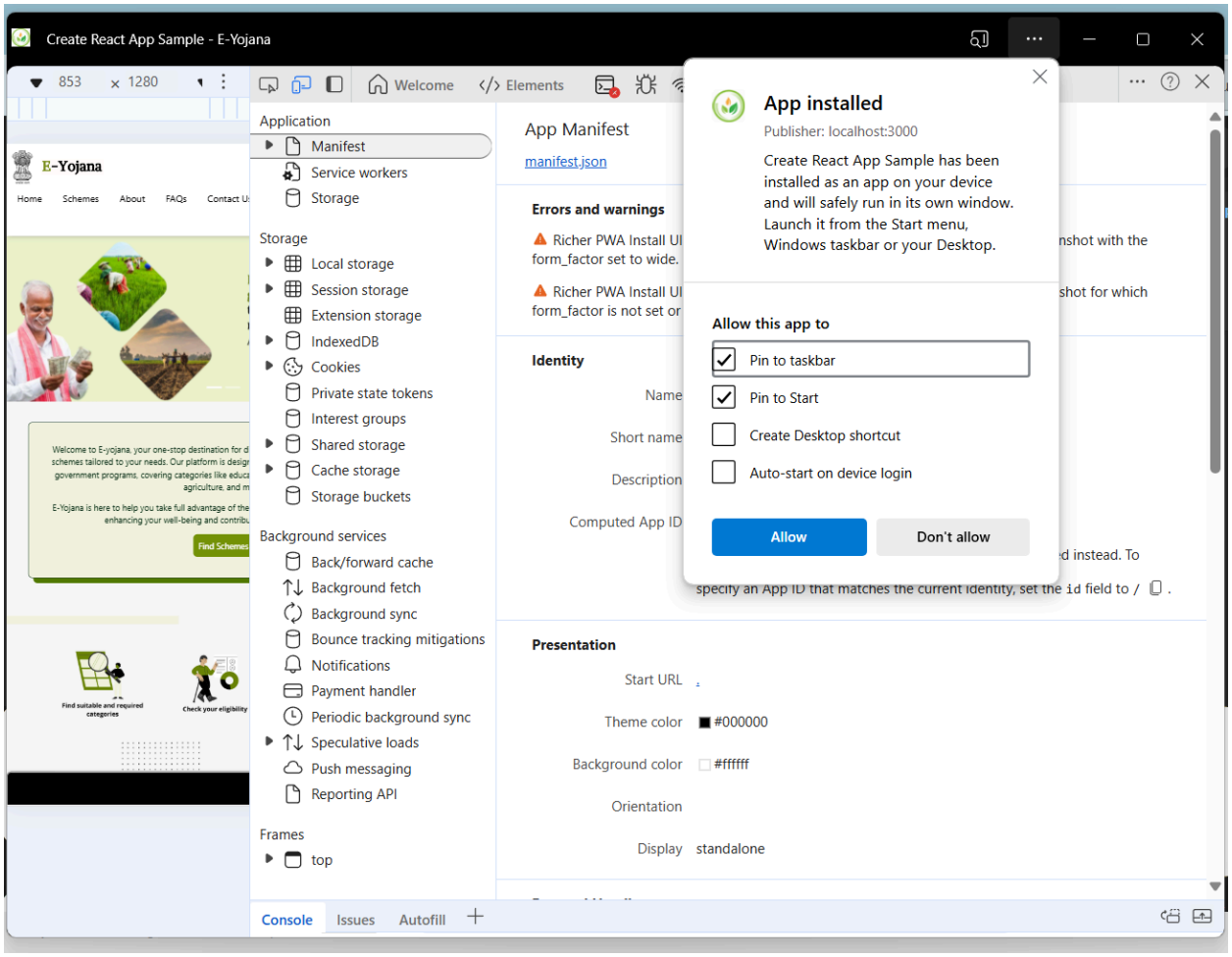
npm -v npm install -g browser-sync

npm install -g live-server


Install Create React App Sample app
Publisher: localhost:3000
Use this site often? Install the app which:
• Opens in a focused window
• Has quick access options like pin to taskbar
• Syncs across multiple devices
Install
Not now


localhost:3000
E-Yojana
Home
Schemes
About
FAQs
Contact Us
My Applications
Install Create React App Sample app
Publisher: localhost:3000
Use this site often? Install the app which:
• Opens in a focused window
• Has quick access options like pin to taskbar
• Syncs across multiple devices
Install
Not now
government schemes that match your needs....
According to your eligibility
ENG
IN
19:27
25-03-2025


localhost:3000
E-Yojana
Home
Schemes
About
FAQs
Contact Us
My Applications
LogOut
Discovering government schemes that match your needs....
According to your eligibility
Welcome to E-yojana, your one-stop destination for discovering and applying for government schemes tailored to your needs. Our platform is designed to simplify access to a wide range of government programs, covering categories like education, healthcare, employment, housing, agriculture, and more.
E-Yojana is here to help you take full advantage of the resources and support available to you, enhancing your well-being and contributing to national progress.
Find Schemes
Welcome
Elements
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitiga...
Notifications
Payment handler
Periodic background sy...
Speculative loads
Push messaging
Reporting API
Frames
top
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.
Identity
Name
Create React App Sample
Short name
React App
Description
Computed App ID
http://localhost:3000/
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 /
Presentation
Start URL
Theme color
#000000
Background color
#ffffff
Orientation
Display
standalone
Console
Issues
Autofill



Conclusion:-

Hence, we learnt how to write a metadata of our E-commerce website PWA in a Web App Manifest File to enable add to homescreen feature.

MAD & PWA Lab
Journal

Experiment No.	08
Experiment Title.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA
Roll No.	27
Name	Rujuta Medhi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Rujuta Medhi
D15A-27

Experiment: 08

MAD and PWA ab

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

What can we do with Service Workers?

- You can dominate **Network Traffic**

  You can manage all network traffic of the page and do any manipulations. For example, when the page requests a CSS file, you can send plain text as a response or when the page requests an HTML file, you can send a png file as a response. You can also send a true response too.

- You can **Cache**

  You can cache any request/response pair with Service Worker and Cache API and you can access these offline content anytime.

- You can manage **Push Notifications**

  You can manage push notifications with Service Worker and show any information message to the user.

- You can **Continue**

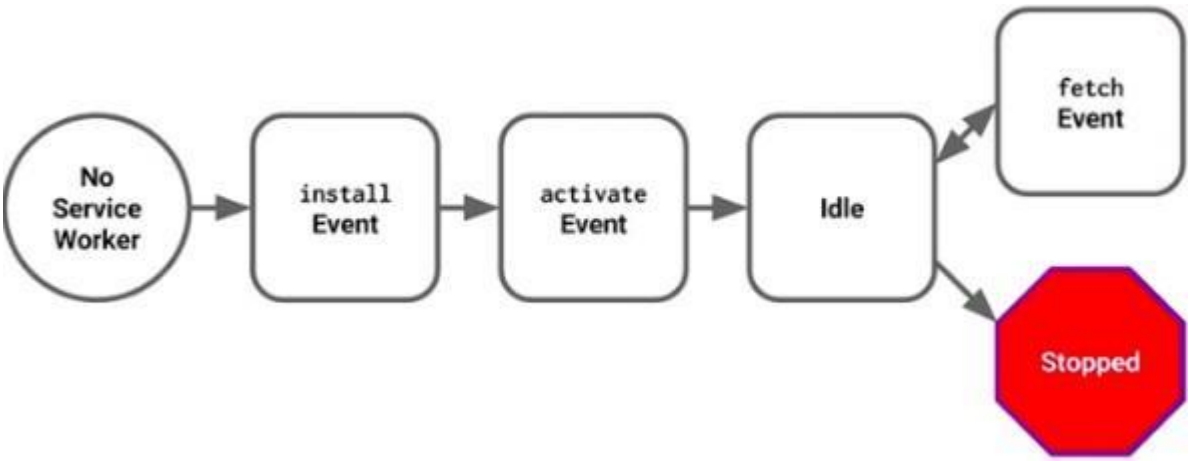
  Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

What can’t we do with Service Workers?

- You can’t access the **Window**
  You can’t access the window, therefore, You can’t manipulate DOM elements. But, you can communicate to the window through post Message and manage processes that you want.

- You can’t work it on **80 Port**
  Service Worker just can work on HTTPS protocol. But you can work on localhost during development.

Service Worker Cycle



A service worker goes through three steps in its life cycle:

- Registration
- Installation
- Activation

Registration

To install a service worker, you need to register it in your main JavaScript code. Registration tells the browser where your service worker is located, and to start installing it in the background. Let's look at an example:

main.js

```
if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('/service-worker.js')
  .then(function(registration) {
    console.log('Registration successful, scope is:', registration.scope);
  })
  .catch(function(error) {
    console.log('Service worker registration failed, error:', error);
  });
}
```

This code starts by checking for browser support by examining **navigator.serviceWorker**. The service worker is then registered with navigator.serviceWorker.register, which returns a promise that resolves when the service worker has been successfully registered. The scope of the service worker is then logged with registration.scope. If the service worker is already installed, navigator.serviceWorker.register returns the registration object of the currently active service worker.

The scope of the service worker determines which files the service worker controls, in other words, from which path the service worker will intercept requests. The default scope is the location of the service worker file, and extends to all directories below. So if service-worker.js is located in the root directory, the service worker will control requests from all files at this domain.

You can also set an arbitrary scope by passing in an additional parameter when registering. For example:
main.js

```
navigator.serviceWorker.register('/service-worker.js', {
  scope: '/app/'
});
```

In this case we are setting the scope of the service worker to /app/, which means the service worker will control requests from pages like /app/, /app/lower/ and /app/lower/lower, but not from pages like /app or /, which are higher.

If you want the service worker to control higher pages e.g. /app (without the trailing slash) you can indeed change the scope option, but you'll also need to set the Service-Worker-Allowed HTTP Header in your server config for the request serving the service worker script.

main.js

```
navigator.serviceWorker.register('/app/service-worker.js', {
  scope: '/app'
});
```

Installation

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

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

service-worker.js

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

Activation

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

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

service-worker.js

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

Once activated, the service worker controls all pages that load within its scope, and starts listening for events from those pages. However, pages in your app that were loaded before the service worker activation will not be under service worker control. The new service worker will only take over when you close and reopen your app, or if the service worker calls **clients.claim()**. Until then, requests from this page will not be intercepted by the new service worker. This is intentional as a way to ensure consistency in your site.

Code

sw.js

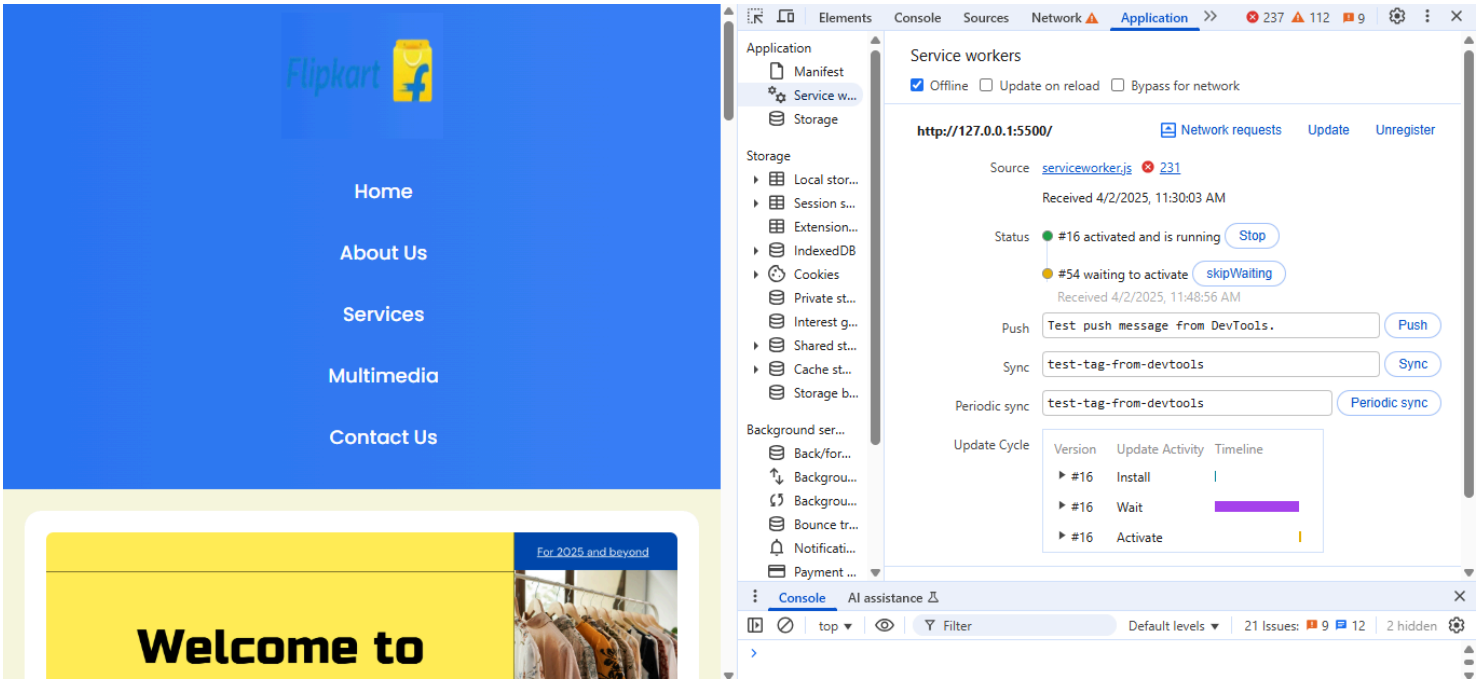
```
const CACHE_NAME = "ecommerce-pwa-cache-v1";
const urlsToCache = [
   "/",            // Root path
   "/index.html",   // Main page
   "/style.css",   // CSS file
   "/index.js",       // Main JavaScript file
   "/flipkart.png",
   "flipkart4.png",
   "/electronics.png",
   "/fashion.png",
   "/home.png",
   "/books.png"
];

// Install event: Cache static assets
self.addEventListener('install', (event) => {
   event.waitUntil(
      caches.open(CACHE_NAME)
         .then(cache => {
            console.log("Opened cache");
            return Promise.all(
               urlsToCache.map(url =>
                  fetch(url) // Try to fetch the file first
                     .then(response => {
                        if (!response.ok) throw new Error(`Failed to fetch ${url}`);
                        return cache.put(url, response);
                     })
                     .catch(err => console.warn(`Skipping ${url}:`, err))
               )
            );
         })
         .catch(err => console.log("Cache install failed:", err))
   );
});
```

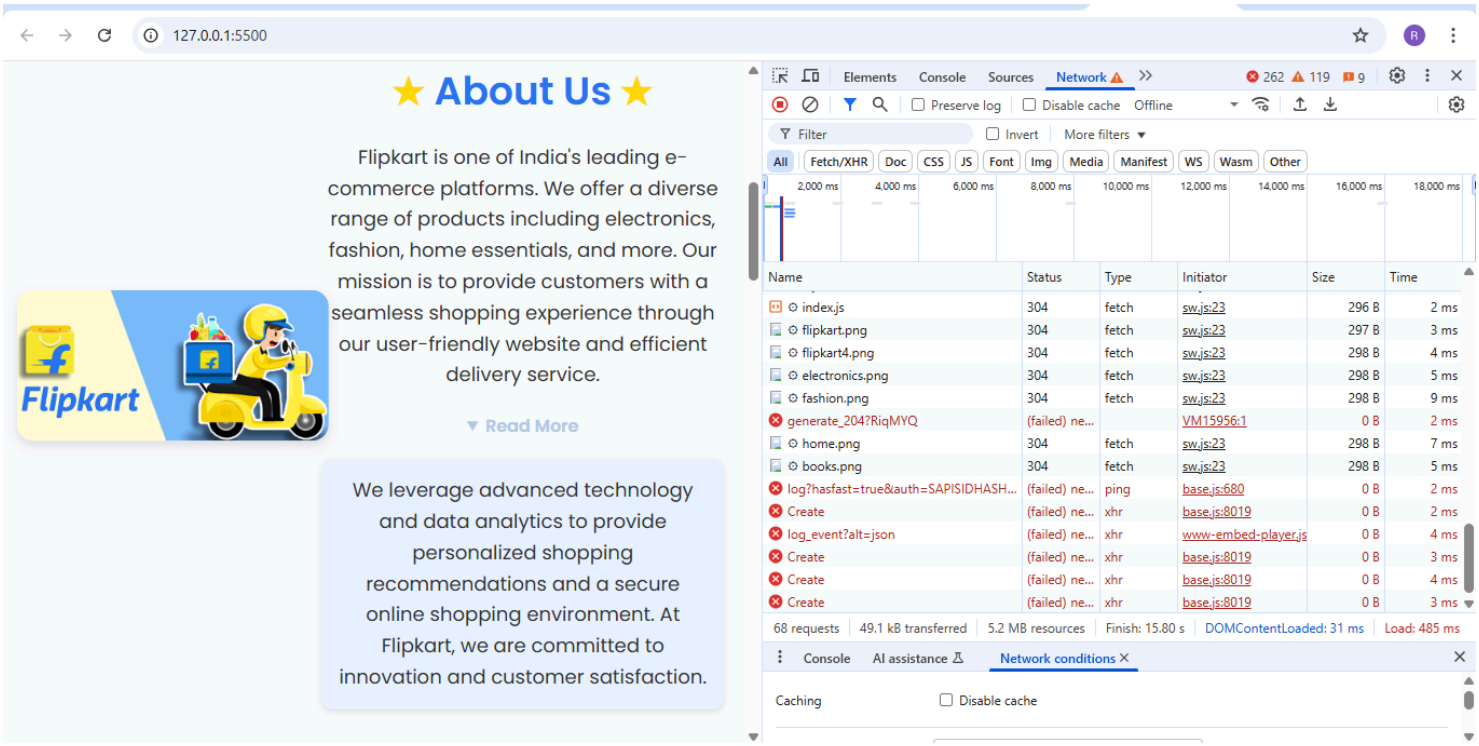
```
// Activate event: Cleanup old caches
self.addEventListener('activate', (event) => {
    event.waitUntil(
        caches.keys().then(cacheNames => {
            return Promise.all(
                cacheNames.map(cache => {
                    if (cache !== CACHE_NAME) {
                        console.log("Deleting old cache:", cache);
                        return caches.delete(cache);
                    }
                })
            );
        })
    );
});

// Fetch event: Serve from cache or fetch new data
self.addEventListener('fetch', (event) => {
    event.respondWith(
        caches.match(event.request)
            .then(response => {
                return response || fetch(event.request);
            })
            .catch(() => {
                if (event.request.destination === 'document') {
                    return caches.match('/index.html');
                }
            })
    );
});
```

index.js

```
if ('serviceWorker' in navigator) {
    window.addEventListener('load', () => {
        navigator.serviceWorker.register('/sw.js')
            .then(reg => console.log('Service Worker registered!', reg))
            .catch(err => console.log('Service Worker registration failed:', err));
    });
```

Output


Flipkart
Home
About Us
Services
Multimedia
Contact Us
For 2025 and beyond
Welcome to
Elements
Console
Sources
Network
Application
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Network requests
Update
Unregister
Source
serviceworker.js
231
Received 4/2/2025, 11:30:03 AM
Status
#16 activated and is running
Stop
#54 waiting to activate
skipWaiting
Received 4/2/2025, 11:48:56 AM
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#16 Install
#16 Wait
#16 Activate
Console
AI assistance


127.0.0.1:5500
★ About Us ★
Flipkart is one of India's leading e-commerce platforms. We offer a diverse range of products including electronics, fashion, home essentials, and more. Our mission is to provide customers with a seamless shopping experience through our user-friendly website and efficient delivery service.
▼ Read More
We leverage advanced technology and data analytics to provide personalized shopping recommendations and a secure online shopping environment. At Flipkart, we are committed to innovation and customer satisfaction.
Elements
Console
Sources
Network
Preserve log
Disable cache
Offline
Filter
Invert
More filters
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
WS
Wasm
Other
Name
Status
Type
Initiator
Size
Time
68 requests
49.1 kB transferred
5.2 MB resources
Finish: 15.80 s
DOMContentLoaded: 31 ms
Load: 485 ms
Console
AI assistance
Network conditions
Caching
Disable cache

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.	27
Name	Rujuta Medhi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

EXPERIMENT NO. 9
MAD and PWA Lab

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

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

Things to note about Service Worker:

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

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

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

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

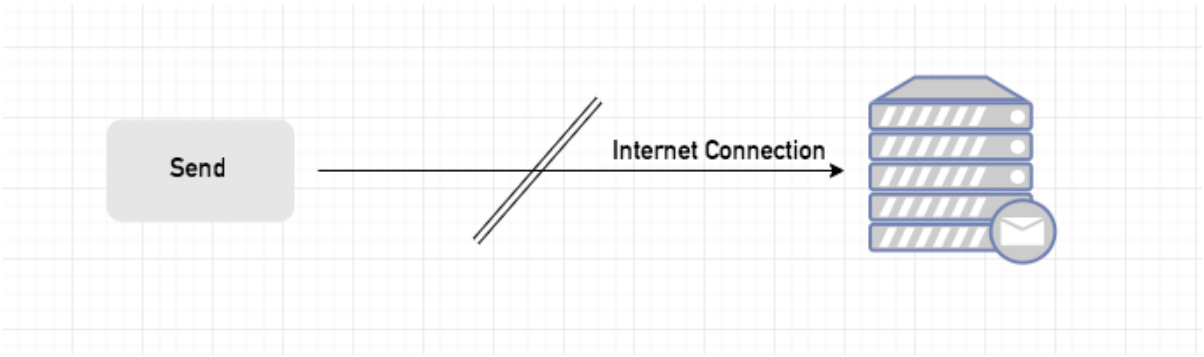
```
self.addEventListener("fetch", function (event) {
    const req = event.request;
    const url = new URL(req.url);

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

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

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

Sync Event

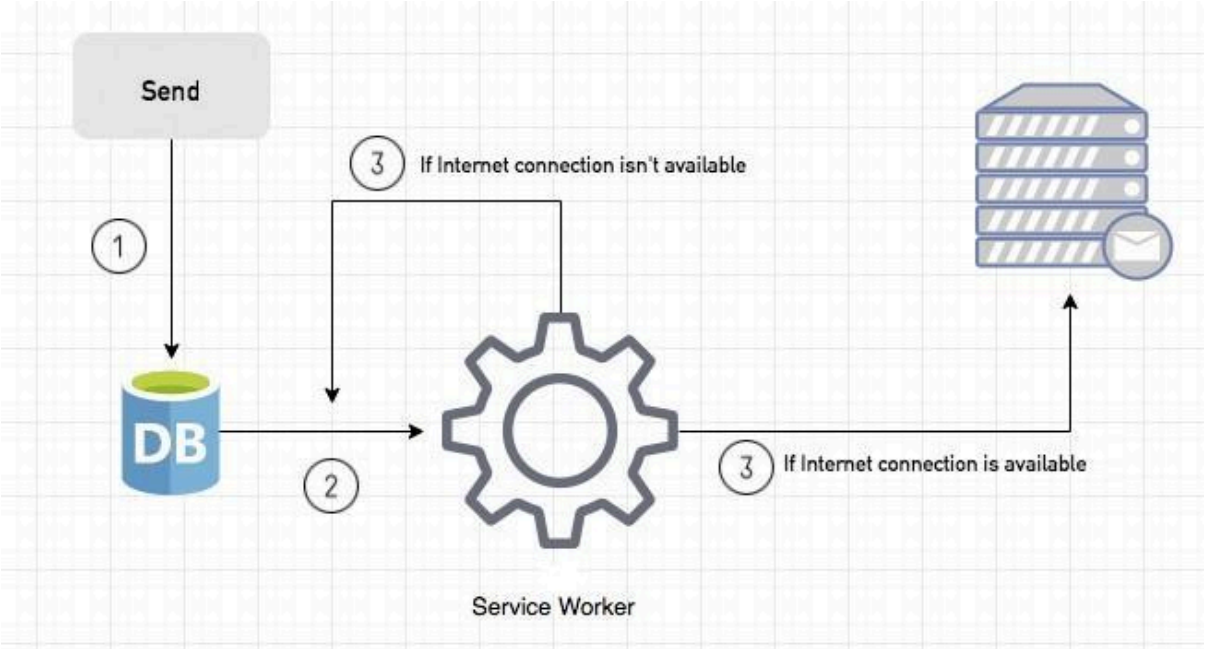
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

Here is a job for the Background Sync.

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



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



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

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

Event Listener for Background Sync Registration

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

Event Listener for sw.js

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

Push Event

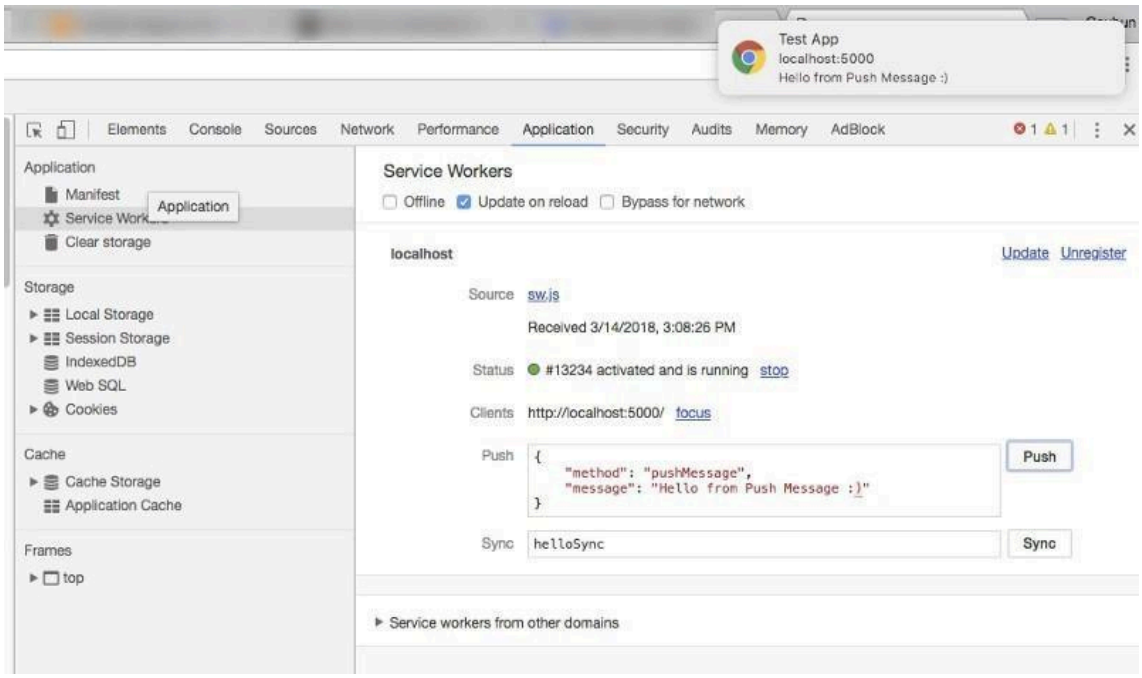
This is the event that handles push notifications that are received from the server. You can apply any method with received data.

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

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

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

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



Code:
sw.js

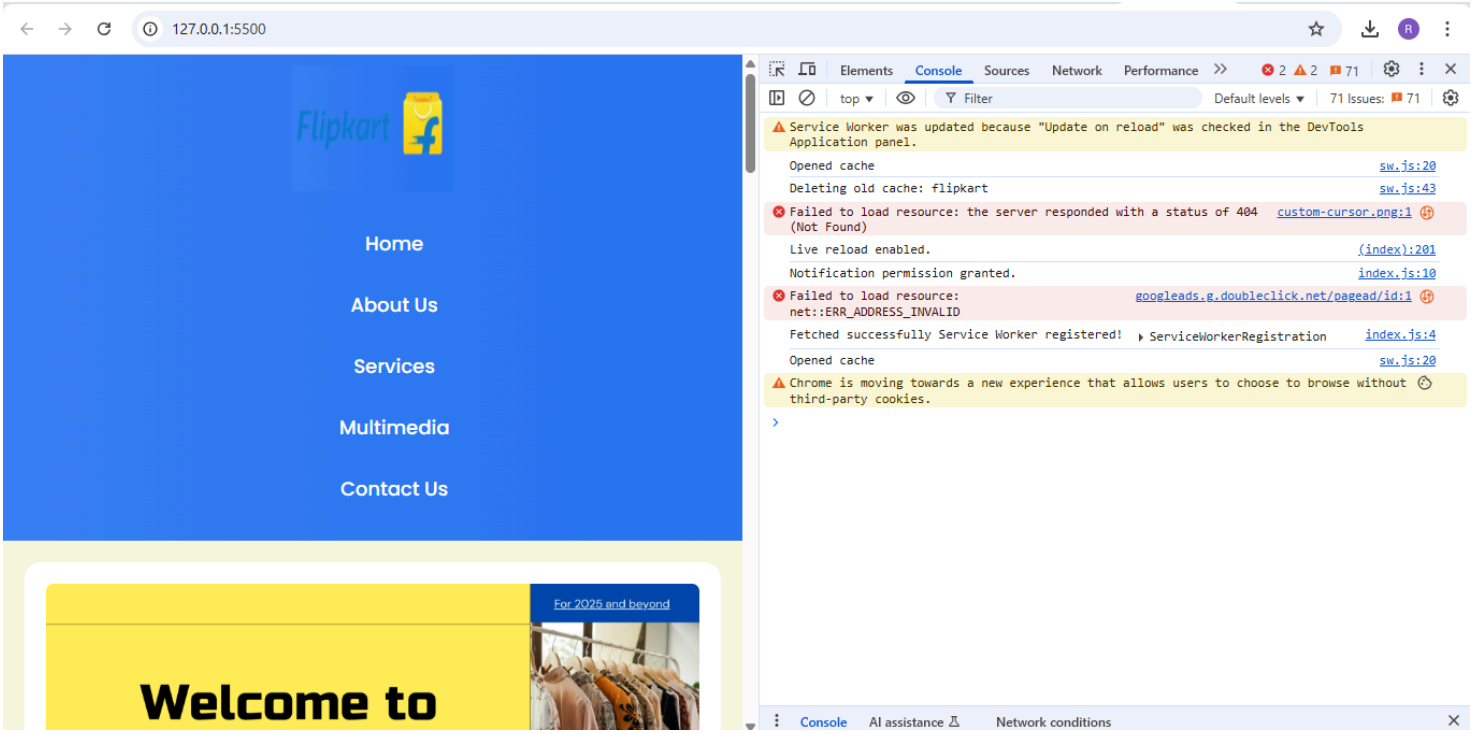
```
self.addEventListener("install", function (event) {
    event.waitUntil(preLoad());
});

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

self.addEventListener('sync', event => { if
    (event.tag === 'syncMessage') {
```

```
        console.log("Sync successful!")
    }
});

self.addEventListener('push', function   (event)   {
    if (event && event.data) {
        var data = event.data.json();
        if (data.method == "pushMessage") {
            console.log("Push notification sent");
            event.waitUntil(self.registration.showNotification("Omkar Sweets Corner", { body:
                data.message
            }))
        }
    }
})


var filesToCache = [
    '/',
    '/menu',
    '/contactUs',
    '/offline.html',
];

var preLoad = function () {
    return   caches.open("offline").then(function   (cache)   {
        // caching index and important routes
        return cache.addAll(filesToCache);
    });
};

var checkResponse =   function   (request)   { return
    new Promise(function (fulfill, reject) {
        fetch(request).then(function    (response)    {
            if (response.status !== 404) {
                fulfill(response);
            } else {
                reject();
```

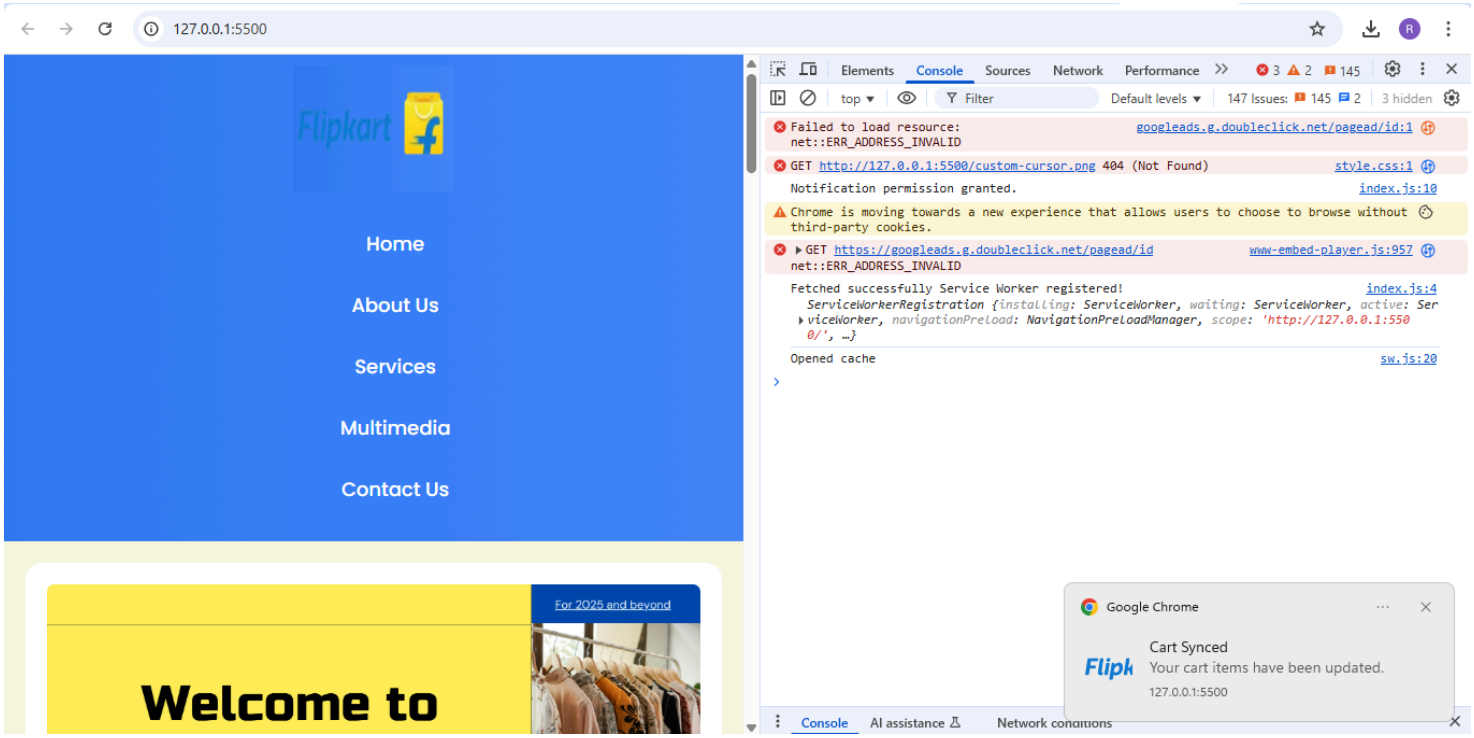
```
}
}, reject);
});
};

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

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

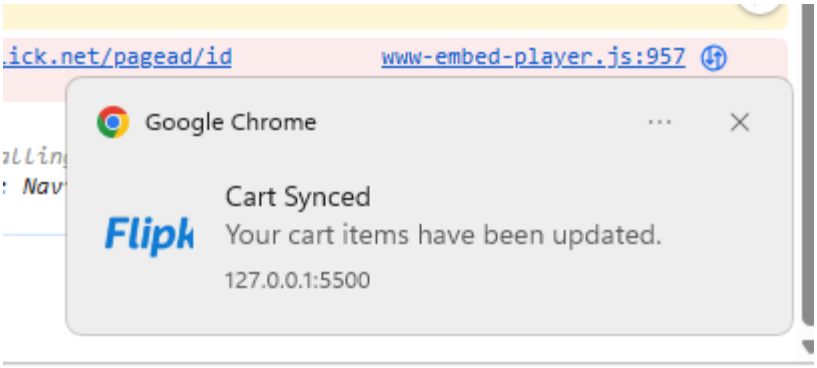
Output:

Fetch event

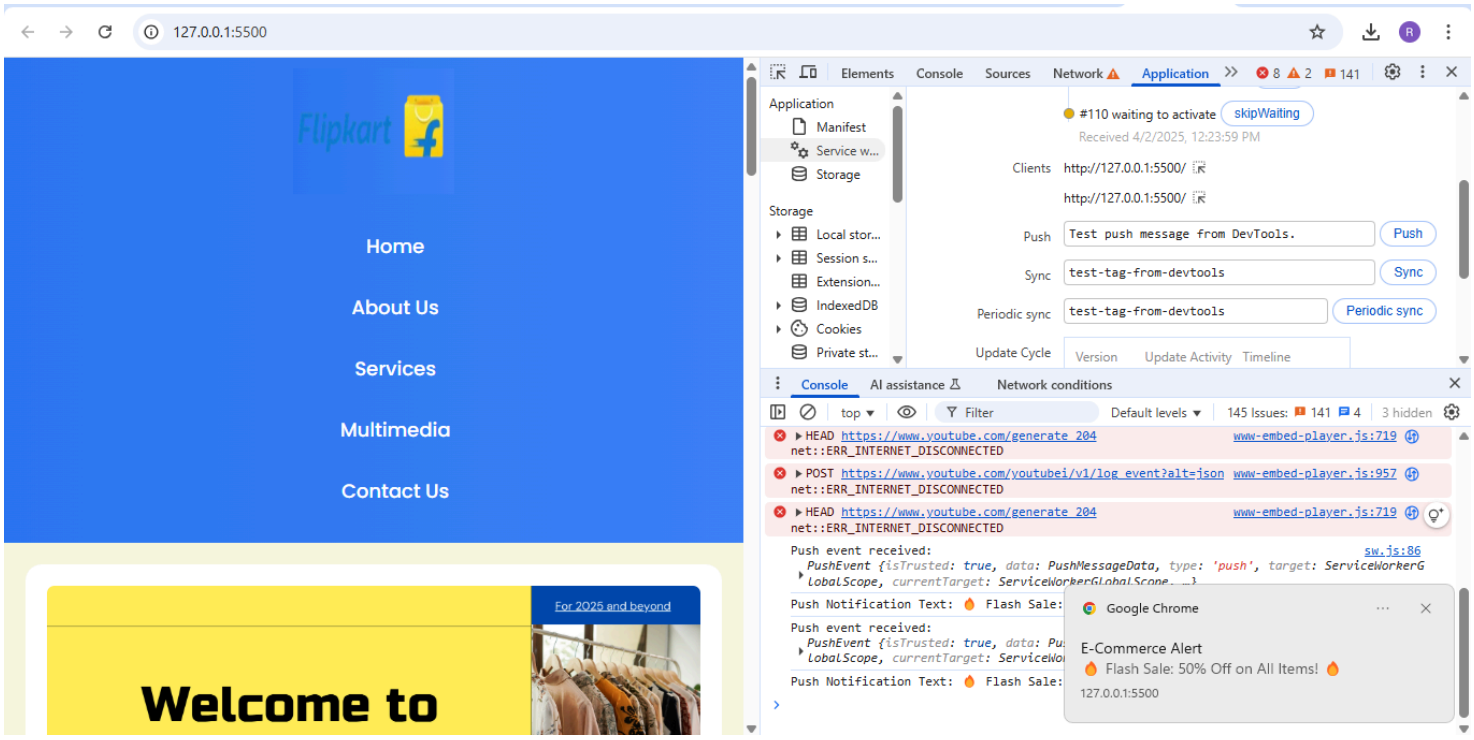

127.0.0.1:5500
Flipkart
Home
About Us
Services
Multimedia
Contact Us
For 2025 and beyond
Welcome to
Elements
Console
Sources
Network
Performance
Service Worker was updated because "Update on reload" was checked in the DevTools Application panel.
Opened cache
sw.js:20
Deleting old cache: flipkart
sw.js:43
Failed to load resource: the server responded with a status of 404 (Not Found)
custom-cursor.png:1
Live reload enabled.
(index):201
Notification permission granted.
index.js:10
Failed to load resource: net::ERR_ADDRESS_INVALID
googleads.g.doubleclick.net/pagead/id:1
Fetched successfully Service Worker registered!
ServiceWorkerRegistration
index.js:4
Chrome is moving towards a new experience that allows users to choose to browse without third-party cookies.
Console
AI assistance
Network conditions

Sync event


127.0.0.1:5500
Flipkart
Home
About Us
Services
Multimedia
Contact Us
For 2025 and beyond
Welcome to
Elements
Console
Sources
Network
Performance
Failed to load resource: net::ERR_ADDRESS_INVALID
googleads.g.doubleclick.net/pagead/id:1
GET http://127.0.0.1:5500/custom-cursor.png 404 (Not Found)
style.css:1
Notification permission granted.
index.js:10
Chrome is moving towards a new experience that allows users to choose to browse without third-party cookies.
GET https://googleads.g.doubleclick.net/pagead/id net::ERR_ADDRESS_INVALID
www-embed-player.js:957
Fetched successfully Service Worker registered!
index.js:4
Opened cache
sw.js:20
Google Chrome
Cart Synced
Your cart items have been updated.
127.0.0.1:5500
Console
AI assistance
Network conditions


ick.net/pagead/id
www-embed-player.js:957
Google Chrome
Cart Synced
Your cart items have been updated.
127.0.0.1:5500

Push event


127.0.0.1:5500
Flipkart
Home
About Us
Services
Multimedia
Contact Us
For 2025 and beyond
Welcome to
Elements
Console
Sources
Network
Application
Application
Manifest
Service w...
Storage
Storage
Local stor...
Session s...
Extension...
IndexedDB
Cookies
Private st...
#110 waiting to activate
skipWaiting
Received 4/2/2025, 12:23:59 PM
Clients
http://127.0.0.1:5500/
http://127.0.0.1:5500/
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
Console
AI assistance
Network conditions
top
Filter
Default levels
145 Issues: 141 4
3 hidden
HEAD https://www.youtube.com/generate_204 www-embed-player.js:719
net::ERR_INTERNET_DISCONNECTED
POST https://www.youtube.com/youtubei/v1/log_event?alt=json www-embed-player.js:957
net::ERR_INTERNET_DISCONNECTED
HEAD https://www.youtube.com/generate_204 www-embed-player.js:719
net::ERR_INTERNET_DISCONNECTED
Push event received:
sw.js:86
PushEvent {isTrusted: true, data: PushMessageData, type: 'push', target: ServiceWorkerG
lobalScope, currentTarget: ServiceWorkerGlobalScope, ...}
Push Notification Text: Flash Sale:
Push event received:
PushEvent {isTrusted: true, data: Pu
lobalScope, currentTarget: ServiceWo
Push Notification Text: Flash Sale:
Google Chrome
E-Commerce Alert
Flash Sale: 50% Off on All Items!
127.0.0.1:5500

MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	27
Name	Rujuta Medhi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Rujuta Medhi
D15A-27

Experiment No.10
MAD & PWA Lab

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

Theory:

GitHub Pages

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

GitHub Pages provides the following key features:

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

Reasons for favoring this over Firebase:

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

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

Pros

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

Cons

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

Firebase

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

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

Pros

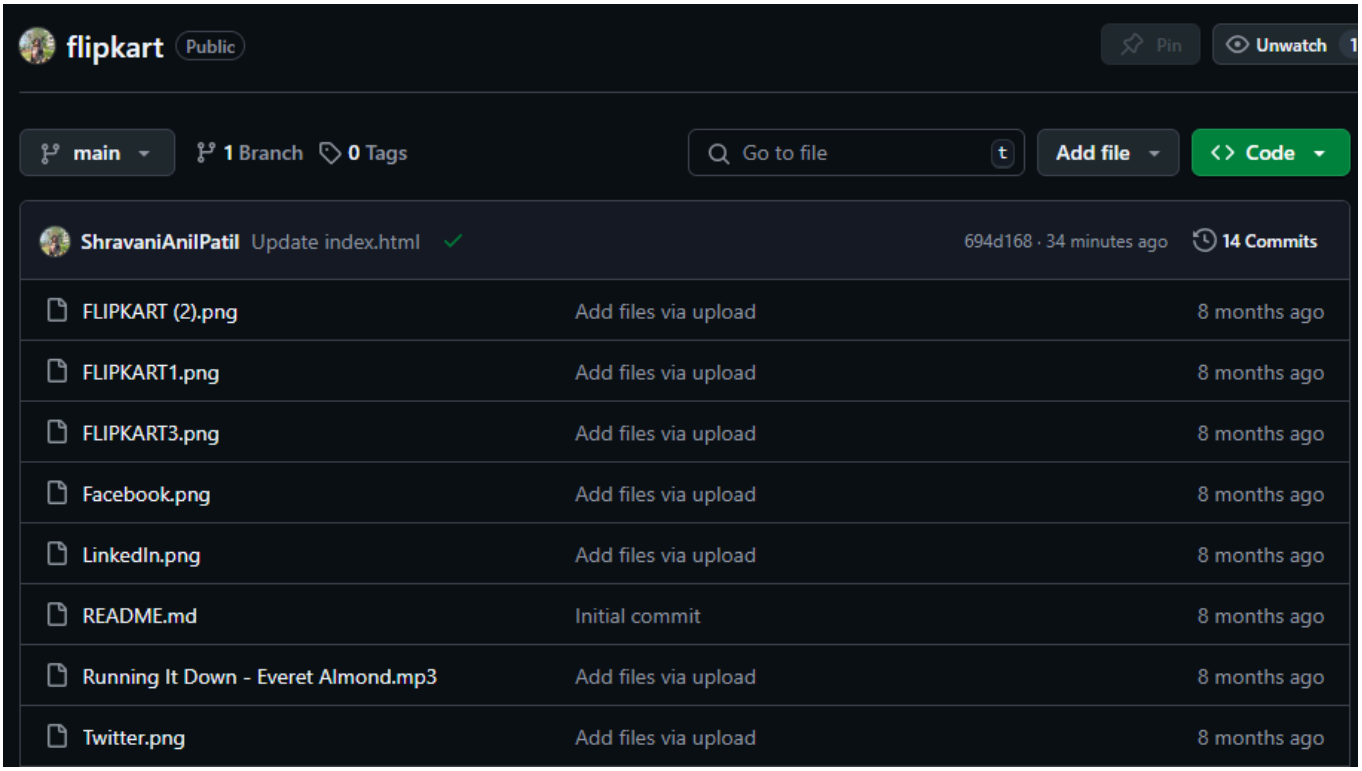
1. Hosted by Google. Enough said.
2. Authentication, Cloud Messaging, and a whole lot of other handy services will be available to you.
3. A real-time database will be available to you, which can store 1 GB of data.
4. You'll also have access to a blob store, which can store another 1 GB of data.
5.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

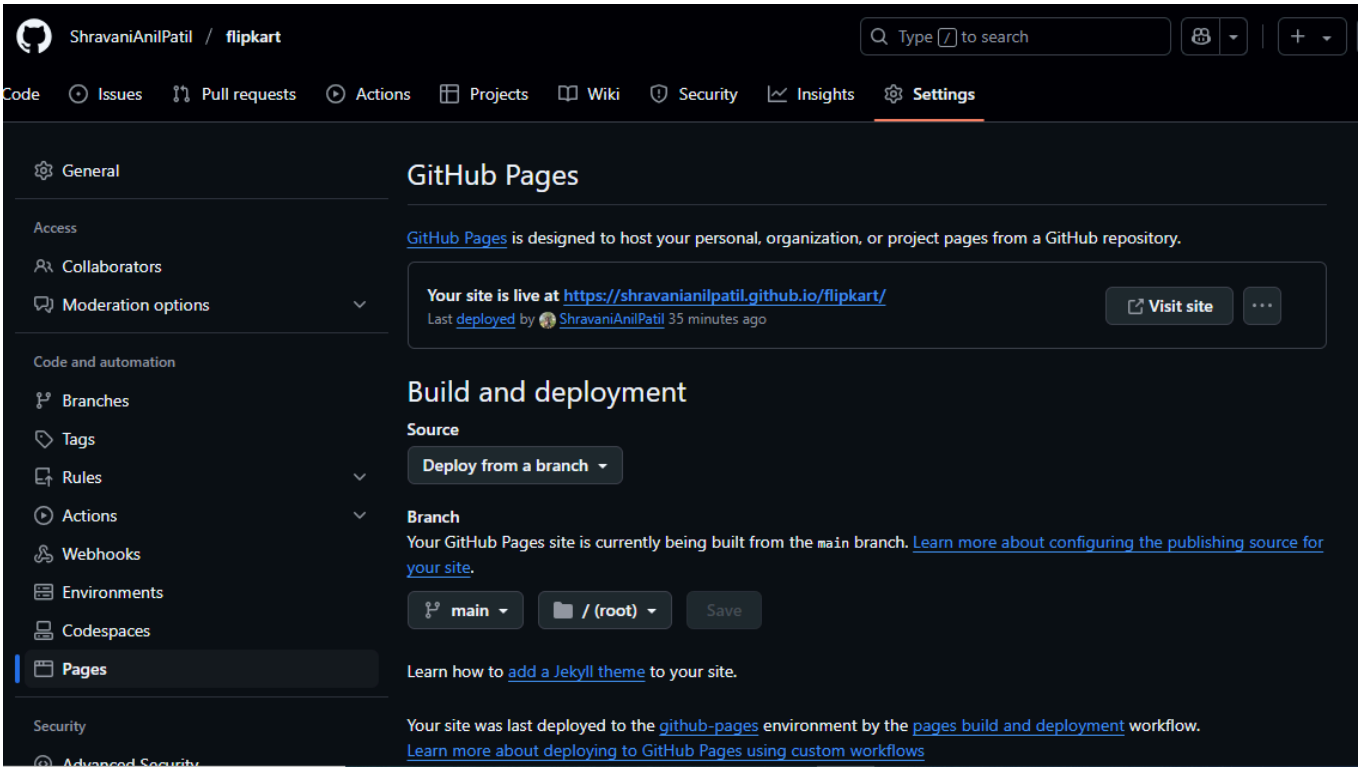
Cons

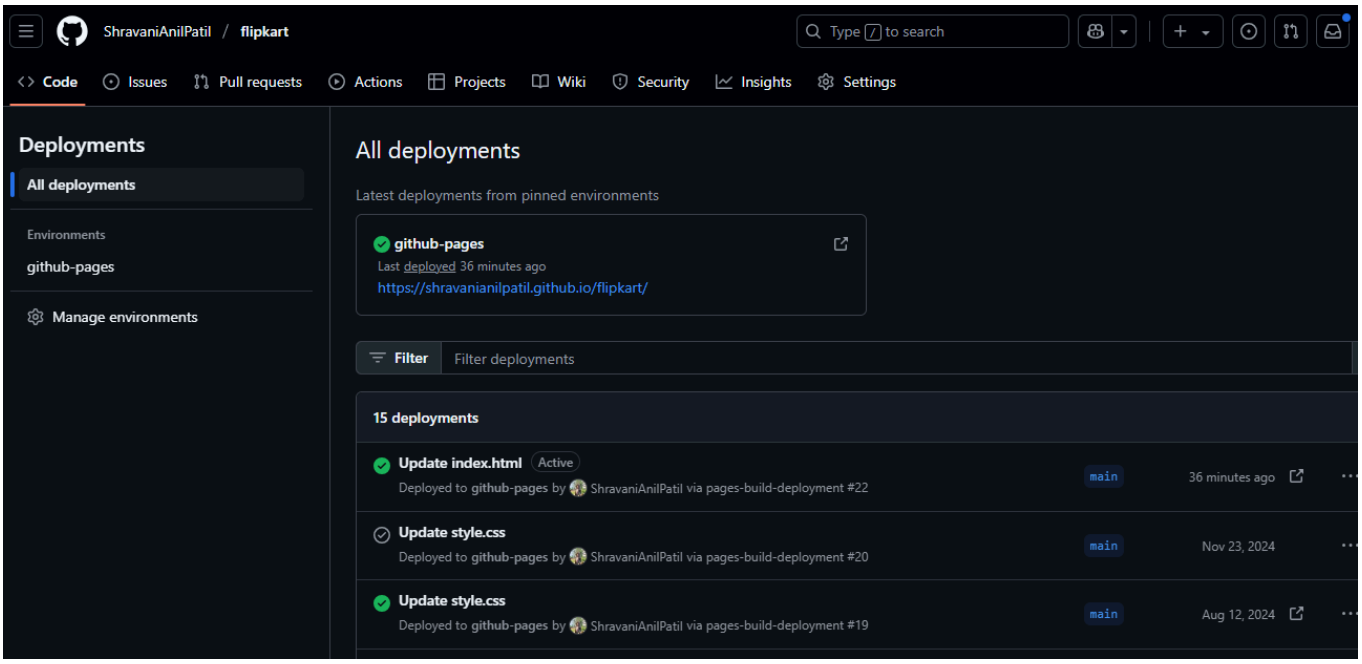
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator.

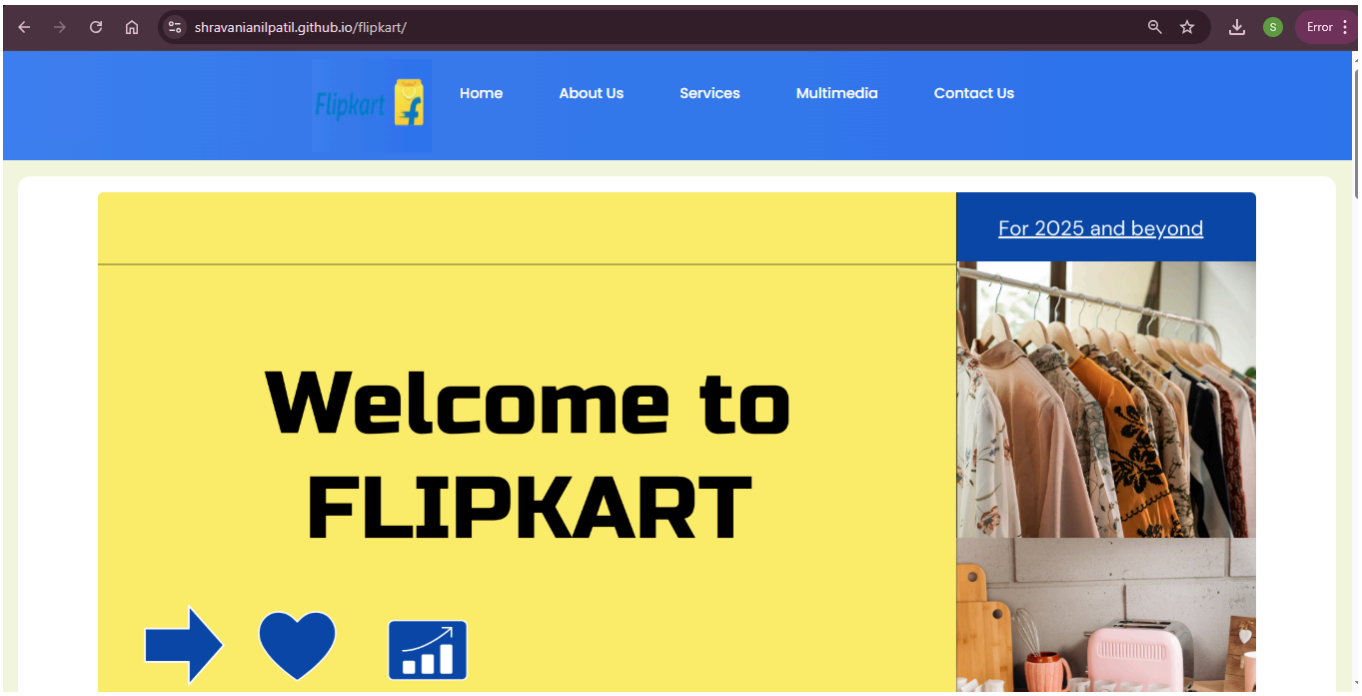
Link to our GitHub repository: https://github.com/ShravaniAnilPatil/flipkart

Github Screenshot:









Deployed Link: https://shravanianilpatil.github.io/flipkart/