

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 Shravani Anil Patil of D15A/D15B 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.

| Project Title: | Roll No. |
|--|-----------------|
| 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 | | | |

MAD & PWA Lab Journal

| | |
|-------------------|---|
| Experiment No. | 01 |
| Experiment Title. | To install and configure the Flutter Environment |
| Roll No. | 37 |
| Name | Shravani Anil Patil |
| 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

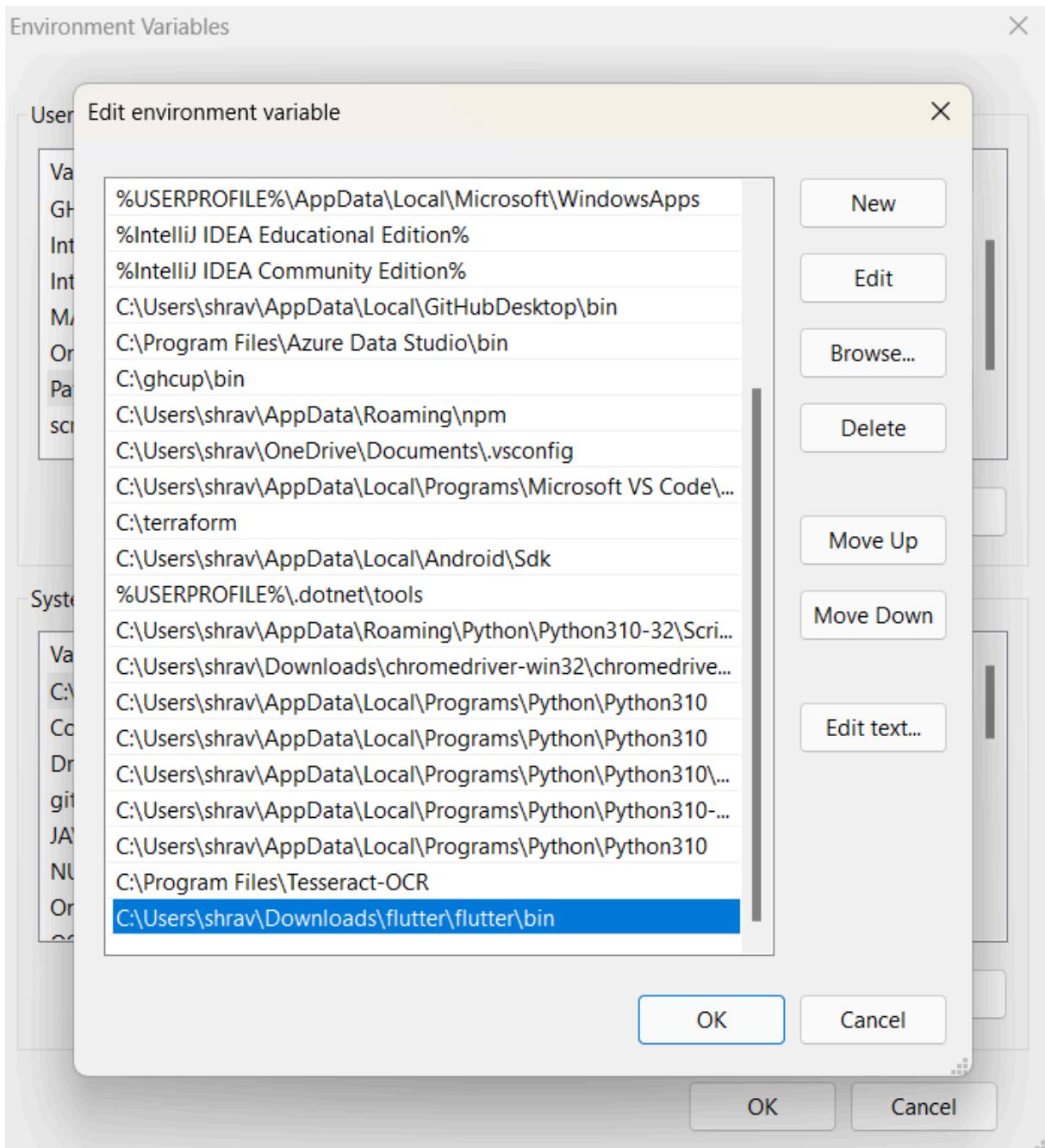
The image contains two screenshots of the Flutter website's setup page. Both screenshots show a sidebar with navigation links like Get started, Set up Flutter, Learn Flutter, Stay up-to-date, App solutions, User interface, Introduction, Widget catalog, Layout, Adaptive & responsive design, Design & theming, Interactivity, Assets & media, and Navigation & routing. The main content area is titled "Choose your development platform to get started". In the first screenshot, the Windows icon is selected, and a note about developing in China is visible. In the second screenshot, the Android icon is selected, and a note about developing in China is also visible. Both screenshots include a note at the bottom about the latest stable version of Flutter.

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

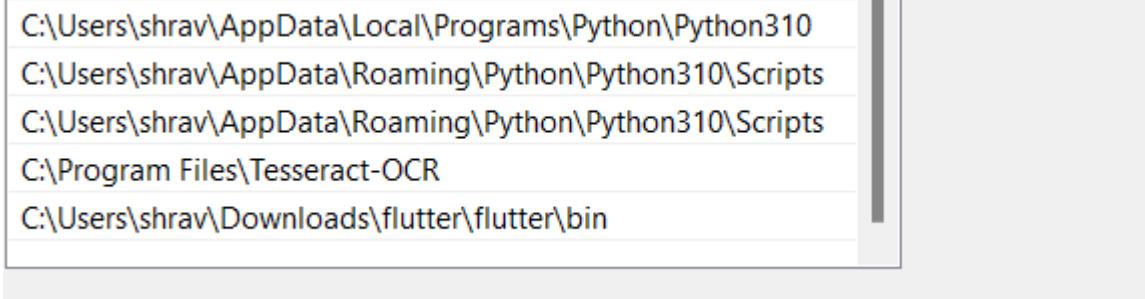
Step 3: Add Flutter to PATH

- 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.

Environment Variables



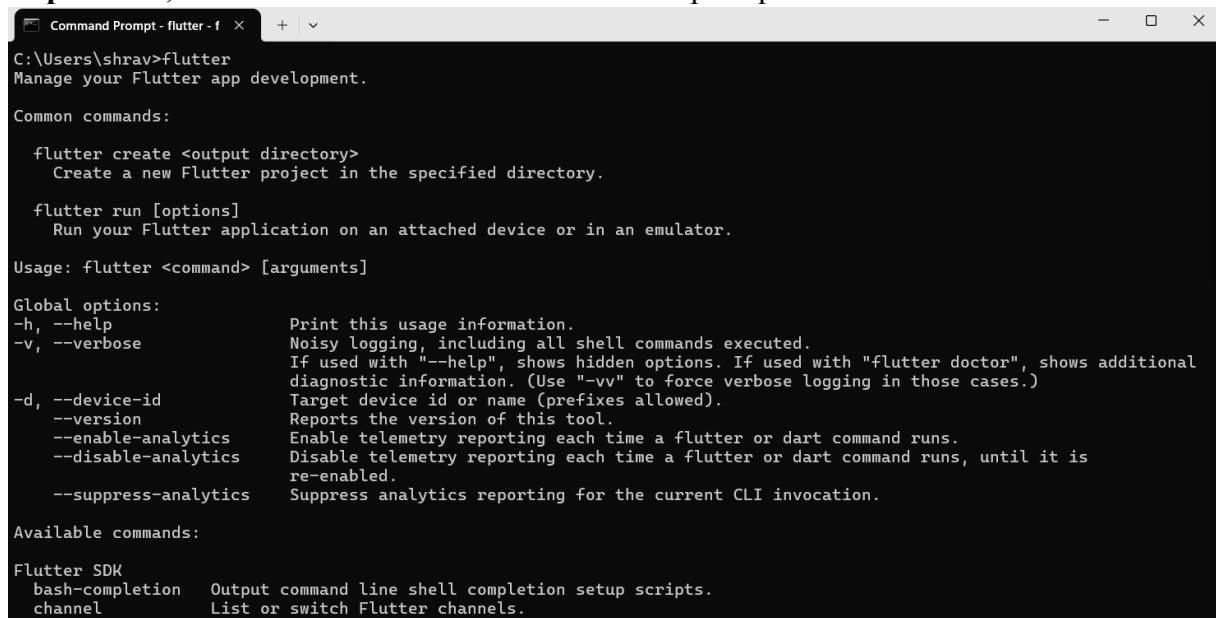
Now, select path -> click on edit. The following screen appears



C:\Users\shraw\AppData\Local\Programs\Python\Python310
C:\Users\shraw\AppData\Roaming\Python\Python310\Scripts
C:\Users\shraw\AppData\Roaming\Python\Python310\Scripts
C:\Program Files\Tesseract-OCR
C:\Users\shraw\Downloads\flutter\flutter\bin

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.



```
Command Prompt - flutter - f
C:\Users\shraw>flutter
Manage your Flutter app development.

Common commands:

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

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

Usage: flutter <command> [arguments]

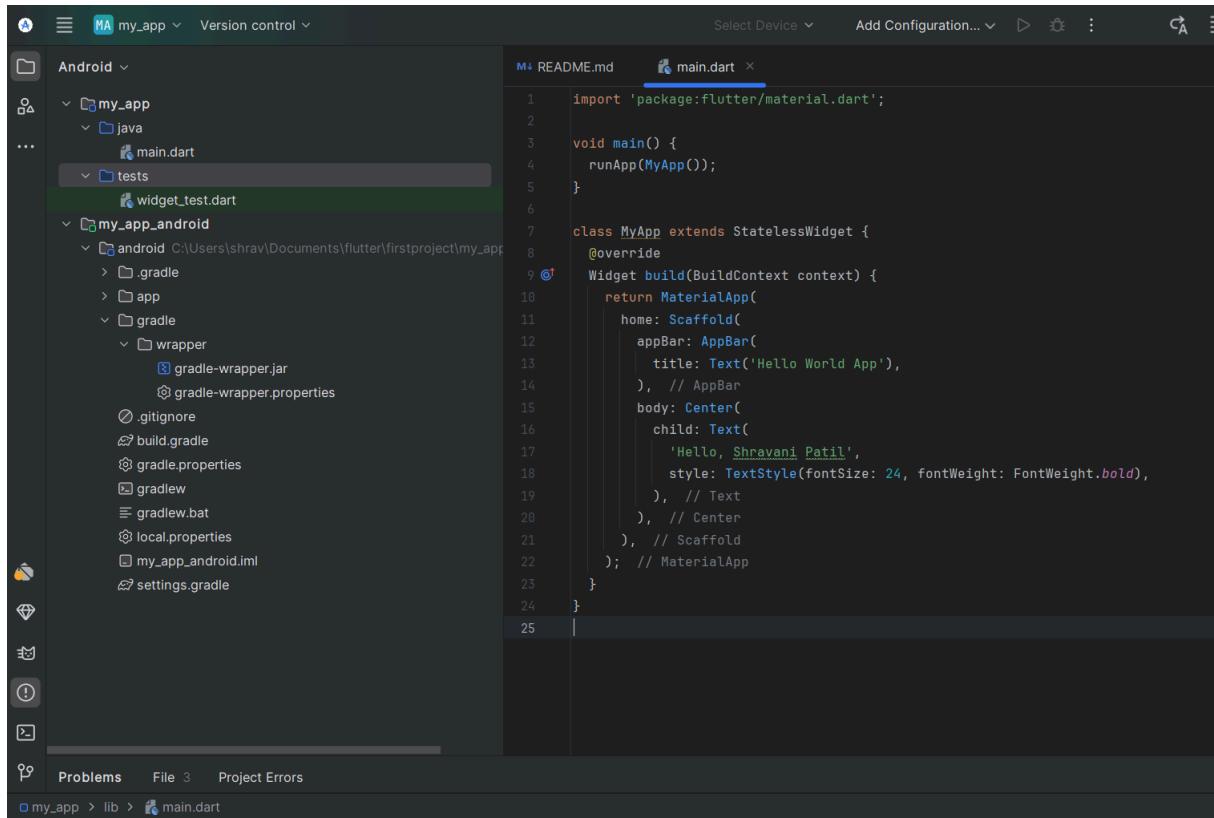
Global options:
-h, --help                  Print this usage information.
-v, --verbose                Noisy logging, including all shell commands executed.
                             If used with "--help", shows hidden options. If used with "flutter doctor", shows additional
                     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

Available commands:

Flutter SDK
  bash-completion  Output command line shell completion setup scripts.
  channel          List or switch Flutter channels.
```

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\shraw>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.22631.4751], 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 17.12.3)
[✓] Android Studio (version 2024.1)
[✓] VS Code (version 1.96.4)
[✓] Connected device (3 available)
[✓] Network resources

• No issues found!

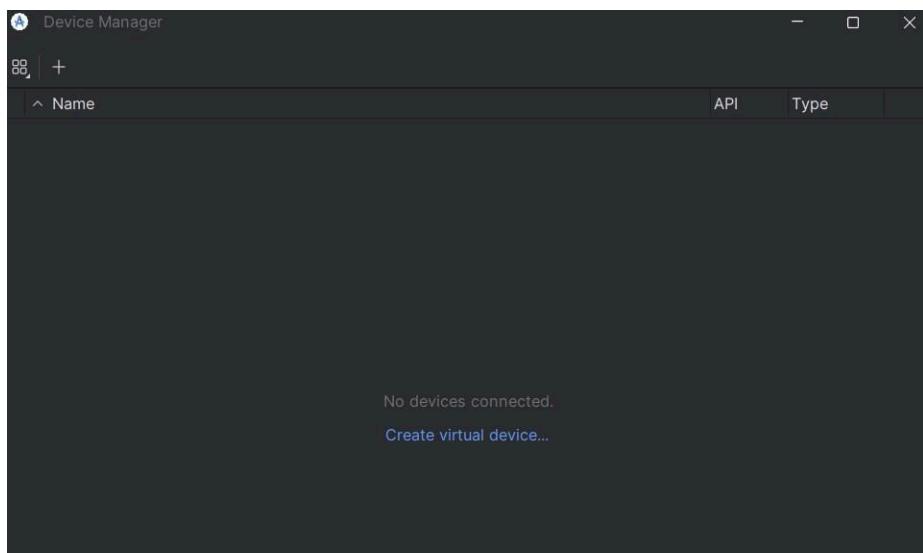
C:\Users\shraw>
```

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 appears.

A screenshot of the System Image selection interface. The title bar says "System Image". On the left, there is a circular icon with a stylized "A" and a gear. Below it, the text "Select a system image" is displayed. There are three tabs at the top: "Recommended", "x86 Images" (which is selected), and "Other Images". A table below lists various system images with columns for "Release Name", "API", "ABI", "xABI", and "Target".

| Release Name | API | ABI | xABI | Target |
|-------------------|-----------|---------------|------|-----------------------------------|
| Baklava ↴ | Baklava | x86_64 | | Android API Baklava (Google Play) |
| Baklava ↴ | Baklava | x86_64 | | Android API Baklava (16 KB) |
| VanillaIceCream ↴ | 35 | x86_64 | | Android 15.0 (Google Play) |
| VanillaIceCream ↴ | 35 | x86_64 | | Android 15.0 (16 KB Page) |
| UpsideDownCake ↴ | 34 | x86_64 | | Android 14.0 (Google Play) |
| Tiramisu ↴ | 33 | x86_64 | | Android 13.0 (Google Play) |
| Sv2 | 32 | x86_64 | | Android 12L (Google Play) |
| S ↴ | 31 | x86_64 | | Android 12.0 (Google Play) |
| R ↴ | 30 | x86 | | Android 11.0 (Google Play) |
| Q ↴ | 29 | x86 | | Android 10.0 (Google Play) |
| Pie ↴ | 28 | x86 | | Android 9.0 (Google Play) |

The "Sv2" row is highlighted with a blue background. To the right of the table, there is a detailed view for the "Sv2" entry:

Sv2

API Level: **32**

Type: **Google Play**

Android: **12L**

Google Inc.

System Image: **x86_64**

We recommend these Google Play images because this device is compatible with Google Play.

Questions on API level?
See the [API level distribution chart](#).

At the bottom of the interface, there is a toolbar with icons for refresh, search, and more, followed by a table showing the selected "Pixel 9 Pro API 32" entry with columns for "Name", "API", "Type", and "More".

| Name | API | Type | More |
|--------------------|-----|---------|------|
| Pixel 9 Pro API 32 | 32 | Virtual | ⋮ |

Details for the selected entry:

Pixel 9 Pro API 32

Android 12L ("Sv2") | x86_64

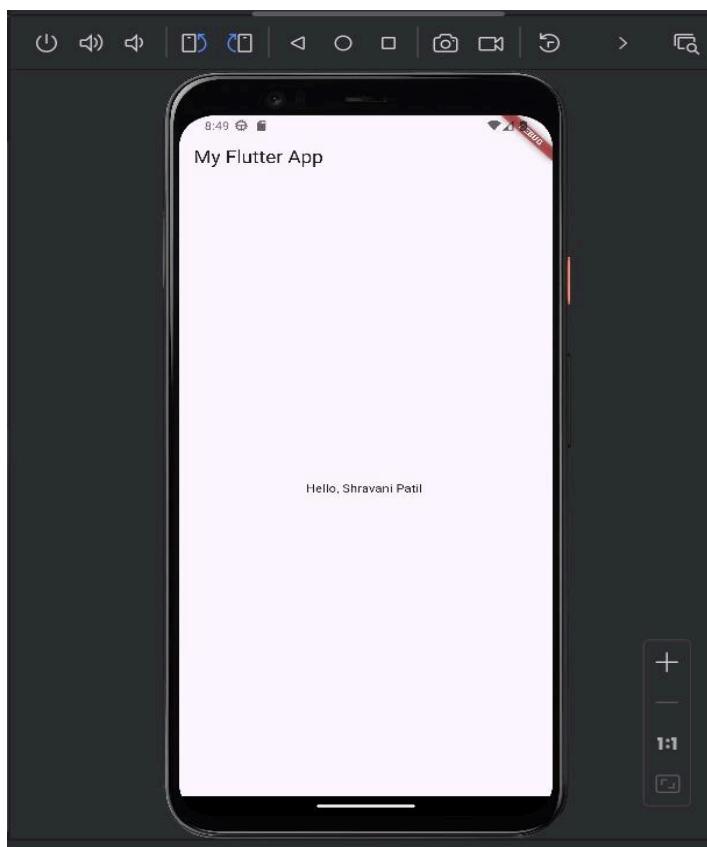
Virtual device created.

Creating a hello world program in flutter:

```
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('Hello World App'),
        ),
        body: Center(
          child: Text(
            'Hello, Shravani Patil',
            style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
          ),
        ),
      );
  }
}
```



MAD & PWA Lab

Journal

| | |
|-------------------|--|
| Experiment No. | 02 |
| Experiment Title. | To design Flutter UI by including common widgets. |
| Roll No. | 37 |
| Name | Shravani Anil Patil |
| Class | D15A/D15B |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO2: Design and Develop interactive Flutter App by using <u>widgets, layouts, gestures and animation</u> |
| Grade: | |

EXPERIMENT NO:02

Aim: To design Flutter UI by including common widgets.

Theory:**Flutter Widgets Overview**

Each element on the screen of a Flutter app is a **widget**. The UI of the app is built by combining and nesting these widgets, forming a widget tree. The structure and behavior of an app depend on the choice and sequence of widgets used.

When any modification is made to the code, Flutter's widget system rebuilds the widget tree by computing the difference between the previous and current widget states. This ensures that only the necessary parts of the UI are updated efficiently, improving performance and responsiveness.

Types of Widgets**1.Single-Child Layout Widgets**

Single-child layout widgets can contain only one child widget inside a parent layout. These widgets often include specific layout functionalities, which help in structuring the UI efficiently. Flutter provides a variety of single-child widgets that enhance readability and maintainability of the code.

2.Multiple-Child Layout Widgets

Multiple-child layout widgets allow more than one child inside a parent widget. Their layouts vary based on how they arrange children. Examples include:

- **Row** – Arranges child widgets horizontally.
- **Column** – Arranges child widgets vertically.
- **Stack** – Overlaps child widgets, useful for layered UI designs.

By combining **Row** and **Column**, complex UI structures can be created efficiently.

3.StatefulWidget

A **StatefulWidget** maintains state information that can change over the widget's lifecycle. It consists of two primary classes:

1. **The widget class**
2. **The state class**

Instead of a `build()` method, a **StatefulWidget** has a `createState()` method that returns a class extending Flutter's **State** class. Common examples of **StatefulWidget**s include:

- **Checkbox**
- **Radio**
- **Slider**
- **InkWell**
- **Form**
- **TextField**

4.StatelessWidget

A **StatelessWidget** does not hold any state and remains static throughout its lifecycle. It only rebuilds when its parent widget triggers an update. Examples include:

- **Text**
- **Row**
- **Column**
- **Container**

Commonly Used Widgets in Flutter

- **Container** – A box widget used for layout styling with padding, margins, borders, colors, and size constraints.
- **Row & Column** – Arranges widgets horizontally (**Row**) or vertically (**Column**) with spacing and alignment control.
- **Stack** – Allows widgets to overlap, useful for banners, floating elements, or layered designs.
- **Text** – Displays customizable text with various font styles, colors, and alignments.
- **Image** – Loads and displays images from assets, network, or memory with scaling and fit options.
- **Scaffold** – Provides a structural layout with an app bar, body, floating action button, and bottom navigation.

- **ListView** – A scrollable list that efficiently handles large dynamic content, supporting both vertical and horizontal scrolling.
- **GridView** – Displays widgets in a grid format, ideal for galleries, product lists, and dashboards.
- **SizedBox** – Creates space between widgets or defines fixed dimensions.
- **ElevatedButton** – A button with an elevation effect, customizable in color, shape, and actions.
- **TextField** – A user input field for text entry, supporting validation and keyboard configurations.
- **AppBar** – A top navigation bar with a title, actions, and menu icons.
- **BottomNavigationBar** – A bottom navigation bar with icons and labels for switching between app sections.
- **Drawer** – A slide-out navigation panel for quick access to menus and settings.
- **Card** – A Material Design component that organizes content inside an elevated box.

Flutter's extensive widget system enables developers to create visually appealing, efficient, and responsive applications by structuring UI elements in a modular way.

Code:

Main.dart:

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: SplashScreen(),
    );
  }
}

class SplashScreen
extends StatefulWidget {
  @override
  _SplashScreenState createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen> {
  @override
  void initState() {
    super.initState();
    Future.delayed(Duration(
      seconds: 7), () {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder:
          (context) => LoginSignupPage()),
      );
    });
  }
}
```

```
    });
}

@Override
Widget build(BuildContext context) {
    return Scaffold(
        backgroundColor: Colors.red,
        body: Center(
            child: Image.asset(
                'assets/logo.png',
                width: 150,
                height: 150,
            ),
        ),
    );
}

class LoginSignupPage extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            backgroundColor: Colors.white,
            appBar: AppBar(
                backgroundColor: Colors.white,
                elevation: 0,
                actions: [
                    TextButton(
                        onPressed: () {},
                        child: Text(
                            "SKIP",
                            style:
                                TextStyle(color: Colors.black, fontSize: 16),
                        ),
                    ),
                ],
            ),
            body: Padding(
                padding:
                    const EdgeInsets.symmetric(horizontal: 20),
            ),
        );
    }
}
```

```
child: Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Image.asset('assets/logo.png', width: 120, height: 120),
    SizedBox(height: 10),
    Text(
      "Enjoy faster show booking through our recommendations tailored for you.",
      textAlign: TextAlign.center,
      style: TextStyle(fontSize: 14, color: Colors.black54),
    ),
    SizedBox(height: 30),

// Google Sign-in Button
ElevatedButton.icon(
  onPressed: () {
    // Navigate to HomePage on Google Sign-in click
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => LoginPage()),
    );
  },
  icon: Image.asset('assets/Google.png', width: 24, height: 24),
  label: Text("Login"),
  style: ElevatedButton.styleFrom(
    backgroundColor: Colors.white,
    foregroundColor: Colors.black,
    minimumSize: Size(double.infinity, 50),
    side: BorderSide(color: Colors.black26),
  ),
),
SizedBox(height: 15),

// Email Sign-in Button
ElevatedButton.icon(
  onPressed: () {},
  icon: Icon(Icons.email, color: Colors.black),
  label: Text("Sign Up"),
  style: ElevatedButton.styleFrom(
    backgroundColor: Colors.white,
    foregroundColor: Colors.black,
```

```
        minimumSize: Size(double.infinity, 50),
        side: BorderSide(color: Colors.black26),
    ),
),
SizedBox(height: 20),  
  
// Mobile Number Input Field
Container(
    padding: EdgeInsets.symmetric(horizontal: 10),
    decoration: BoxDecoration(
        border: Border.all(color: Colors.black26),
        borderRadius: BorderRadius.circular(8),
    ),
    child: Row(
        children: [
            Image.asset(
                'assets/flag.png', // Use an Indian flag asset
                width: 24,
                height: 24,
            ),
            SizedBox(width: 10),
            Text("+91", style: TextStyle(fontSize: 16, color: Colors.black)),
            SizedBox(width: 10),
            Expanded(
                child: TextField(
                    keyboardType: TextInputType.phone,
                    decoration: InputDecoration(
                        hintText: "Enter mobile number",
                        border: InputBorder.none,
                    ),
                ),
            ),
        ],
    ),
),
Spacer(),  
  
// Terms & Conditions
Text.rich(  
    
```

```

TextSpan(
    text: "I agree to the ",
    style: TextStyle(fontSize: 12, color: Colors.black54),
    children: [
        TextSpan(
            text: "Terms & Conditions",
            style: TextStyle(color: Colors.blue, decoration: TextDecoration.underline),
        ),
        TextSpan(text: " and "),
        TextSpan(
            text: "Privacy Policy",
            style: TextStyle(color: Colors.blue, decoration: TextDecoration.underline),
        ),
    ],
),
),
),
),
textAlign: TextAlign.center,
),
SizedBox(height: 20),
],
),
),
),
);
}
}

```

Login.dart:

```

import 'package:flutter/material.dart';
import 'home.dart'; // Import the HomePage file

class LoginPage extends StatelessWidget {
    final TextEditingController emailController = TextEditingController();
    final TextEditingController passwordController = TextEditingController();

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            backgroundColor: Colors.white,
            appBar: AppBar(
                backgroundColor: Colors.white,
                elevation: 0,

```

```
leading: IconButton(  
    icon: Icon(Icons.arrow_back, color: Colors.black),  
    onPressed: () {  
        Navigator.pop(context);  
    },  
,  
,  
body: Padding(  
    padding: const EdgeInsets.symmetric(horizontal: 20),  
    child: Column(  
        mainAxisAlignment: MainAxisAlignment.center,  
        children: [  
            Image.asset('assets/logo.png', width: 120, height: 120),  
            SizedBox(height: 20),  
            Text(  
                "Welcome Back!",  
                style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold),  
            ),  
            SizedBox(height: 10),  
            Text(  
                "Login to continue",  
                style: TextStyle(fontSize: 14, color: Colors.black54),  
            ),  
            SizedBox(height: 30),  
            TextField(  
                controller: emailController,  
                decoration: InputDecoration(  
                    labelText: "Email",  
                    border: OutlineInputBorder(),  
                ),  
            ),  
            SizedBox(height: 15),  
            TextField(  
                controller: passwordController,  
                obscureText: true,  
                decoration: InputDecoration(  
                    labelText: "Password",  
                    border: OutlineInputBorder(),  
                ),  
            ),  
        ],  
    ),  
),
```

```
SizedBox(height: 20),  
ElevatedButton(  
    onPressed: () {  
        String email = emailController.text;  
        String password = passwordController.text;  
  
        if (email.isNotEmpty && password.isNotEmpty) {  
            Navigator.pushReplacement(  
                context,  
                MaterialPageRoute(builder: (context) => HomePage()),  
            );  
        } else {  
            ScaffoldMessenger.of(context).showSnackBar(  
                SnackBar(content: Text("Please enter email and password")),  
            );  
        }  
    },  
    child: Text("Login"),  
    style: ElevatedButton.styleFrom(  
        backgroundColor: Colors.red,  
        foregroundColor: Colors.white,  
        minimumSize: Size(double.infinity, 50),  
    ),  
,  
    SizedBox(height: 15),  
    TextButton(  
        onPressed: () {},  
        child: Text(  
            "Forgot Password?",  
            style: TextStyle(color: Colors.red),  
        ),  
    ),  
    SizedBox(height: 20),  
    Text("Or login with"),  
    SizedBox(height: 10),  
    ElevatedButton.icon(  
        onPressed: () {},  
        icon: Image.asset('assets/Google.png', width: 24, height: 24),  
        label: Text("Login with Google"),  
        style: ElevatedButton.styleFrom(  
    ),
```

```
        backgroundColor: Colors.white,
        foregroundColor: Colors.black,
        minimumSize: Size(double.infinity, 50),
        side: BorderSide(color: Colors.black26),
    ),
),
],
),
),
),
);
}
}
```

Home.dart:

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

class HomePage extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: Text('Welcome Guest!'),
                actions: [
                    IconButton(icon: Icon(Icons.search), onPressed: () {}),
                    IconButton(icon: Icon(Icons.qr_code), onPressed: () {}),
                ],
            ),
            body: SingleChildScrollView(
                child: Column(
                    mainAxisAlignment: MainAxisAlignment.start,
                    children: [
                        // Location Bar
                        Padding(
                            padding: const EdgeInsets.symmetric(horizontal: 15, vertical: 5),
                            child: Text("Navi Mumbai", style: TextStyle(color: Colors.red)),
                        ),
                        // Location Enable Bar
                        Container(

```

```
width: double.infinity,
color: Colors.blue,
padding: EdgeInsets.all(10),
child: Text(
  "Enable location to discover nearby events, movies, and more.",
  style: TextStyle(color: Colors.white),
  textAlign: TextAlign.center,
),
),
),

SizedBox(height: 10),

// Categories (Movies, Music Shows, etc.)
SingleChildScrollView(
  scrollDirection: Axis.horizontal,
  padding: EdgeInsets.symmetric(horizontal: 10),
  child: Row(
    children: [
      _categoryIcon(Icons.movie, "Movies"),
      _categoryIcon(Icons.music_note, "Music Shows"),
      _categoryIcon(Icons.theater_comedy, "Comedy Shows"),
      _categoryIcon(Icons.event, "Plays"),
    ],
),
),
),

SizedBox(height: 20),
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 10),
  child: Row(
    children: [
      Expanded(child: Image.asset('assets/event1.jpg', fit: BoxFit.cover)),
      SizedBox(width: 10),
      Expanded(child: Image.asset('assets/event2.jpg', fit: BoxFit.cover)),
    ],
),
),
),

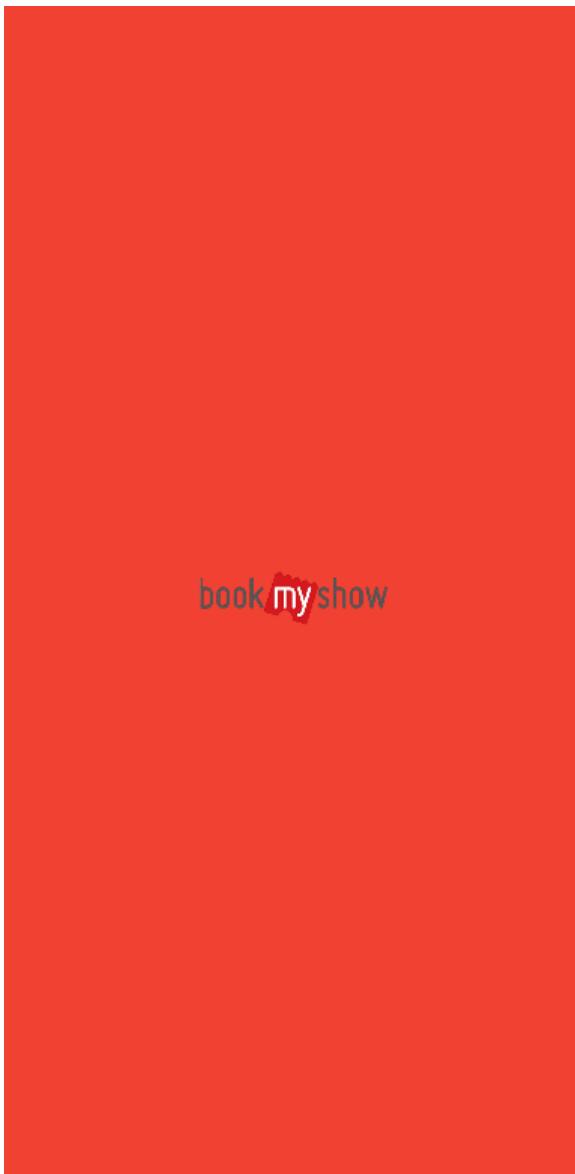
// Recommended Movies Section
```

```
Padding(  
  padding: const EdgeInsets.symmetric(horizontal: 15),  
  child: Row(  
    mainAxisAlignment: MainAxisAlignment.spaceBetween,  
    children: [  
      Text("Recommended Movies", style: TextStyle(fontSize: 18, fontWeight:  
FontWeight.bold)),  
      Text("See All >", style: TextStyle(color: Colors.red)),  
    ],  
  ),  
),  
),  
  
SizedBox(height: 10),  
SingleChildScrollView(  
  scrollDirection: Axis.horizontal,  
  padding: EdgeInsets.symmetric(horizontal: 10),  
  child: Row(  
    children: [  
      _movieCard('assets/movie1.jpg', 'PROMOTED'),  
      _movieCard('assets/movie2.jpg', ''),  
      _movieCard('assets/movie3.jpg', ''),  
    ],  
  ),  
),  
],  
),  
),  
),  
),  
bottomNavigationBar: BottomNavigationBar(  
  selectedItemColor: Colors.red,  
  unselectedItemColor: Colors.black54,  
  items: [  
    BottomNavigationBarItem(icon: Icon(Icons.home), label: "Home"),  
    BottomNavigationBarItem(icon: Icon(Icons.movie), label: "Movies"),  
    BottomNavigationBarItem(icon: Icon(Icons.event), label: "Live Events"),  
    BottomNavigationBarItem(icon: Icon(Icons.person), label: "Profile"),  
  ],  
),  
);  
}
```

```
Widget _categoryIcon(IconData icon, String label) {
  return Column(
    children: [
      Icon(icon, size: 40),
      SizedBox(height: 5),
      Text(label, style: TextStyle(fontSize: 14)),
    ],
  );
}

Widget _movieCard(String imagePath, String tag) {
  return Padding(
    padding: const EdgeInsets.only(right: 10),
    child: Column(
      children: [
        Stack(
          children: [
            Image.asset(imagePath, width: 100, height: 150),
            if (tag.isNotEmpty)
              Positioned(
                top: 5,
                left: 5,
                child: Container(
                  color: Colors.red,
                  padding: EdgeInsets.symmetric(horizontal: 5, vertical: 2),
                  child: Text(tag, style: TextStyle(color: Colors.white, fontSize: 10)),
                ),
              ),
            ],
          ),
        ],
      ),
    );
}
```

OUTPUT:



SKIP



Enjoy faster show booking through our
recommendations tailored for you.

Login

Sign Up

+91 Enter mobile number

I agree to the [Terms & Conditions](#) and [Privacy Policy](#)



Welcome Back!

Login to continue

Email

Password

Login

[Forgot Password?](#)

Or login with

Login with Google



Welcome Back!

Login to continue

Email

shravani@gmail.com

Password

...

Login

[Forgot Password?](#)

Or login with

Login with Google



Welcome Guest!



Navi Mumbai

Enable location to discover nearby events, movies,
and more.



Movies Music Shows Comedy Shows Plays



Recommended Movies

[See All >](#)



Home



MAD & PWA Lab

Journal

| | |
|-------------------|---|
| Experiment No. | 03 |
| Experiment Title. | To include icons, images, fonts in Flutter app |
| Roll No. | 37 |
| Name | Shravani Anil Patil |
| Class | D15A/D15B |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation |
| Grade: | |

Experiment:03

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:

main.dart

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

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

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: SplashScreen(),
    );
  }
}
```

```
class SplashScreen extends StatefulWidget {
  @override
  _SplashScreenState createState() => _SplashScreenState();
}
```

```
class _SplashScreenState extends State<SplashScreen> {
  @override
  void initState() {
    super.initState();
    Future.delayed(Duration(seconds: 7), () {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => LoginSignupPage()),
      );
    });
  }
}
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.red,
    body: Center(
      child: Image.asset(
        'assets/logo.png',
        width: 150,
        height: 150,
      ),
    ),
  );
}
```

```
class LoginSignupPage extends StatelessWidget {
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.white,
    appBar: AppBar(
      backgroundColor: Colors.white,
      elevation: 0,
      actions: [
        TextButton(
          onPressed: () {},
          child: Text(
            "SKIP",
            style: TextStyle(color: Colors.black, fontSize: 16),
          ),
        ),
      ],
    ),
    body: Padding(
      padding: const EdgeInsets.symmetric(horizontal: 20),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Image.asset('assets/logo.png', width: 120, height: 120),
          SizedBox(height: 10),
          Text(
            "Enjoy faster show booking through our recommendations tailored for you.",
            textAlign: TextAlign.center,
            style: TextStyle(fontSize: 14, color: Colors.black54),
          ),
          SizedBox(height: 30),
        ],
      ),
    )));
}

// Google Sign-in Button
ElevatedButton.icon(
  onPressed: () {
    // Navigate to HomePage on Google Sign-in click
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => LoginPage()),
    );
  },
  icon: Image.asset('assets/Google.png', width: 24, height: 24),
  label: Text("Login"),
  style: ElevatedButton.styleFrom(
    backgroundColor: Colors.white,
    foregroundColor: Colors.black,
    minimumSize: Size(double.infinity, 50),
    side: BorderSide(color: Colors.black26),
  ),
),
SizedBox(height: 15),

// Email Sign-in Button
```

```
ElevatedButton.icon(  
    onPressed: () {},  
    icon: Icon(Icons.email, color: Colors.black),  
    label: Text("Sign Up"),  
    style: ElevatedButton.styleFrom(  
        backgroundColor: Colors.white,  
        foregroundColor: Colors.black,  
        minimumSize: Size(double.infinity, 50),  
        side: BorderSide(color: Colors.black26),  
    ),  
,  
SizedBox(height: 20),  
  
// Mobile Number Input Field  
Container(  
    padding: EdgeInsets.symmetric(horizontal: 10),  
    decoration: BoxDecoration(  
        border: Border.all(color: Colors.black26),  
        borderRadius: BorderRadius.circular(8),  
    ),  
    child: Row(  
        children: [  
            Image.asset(  
                'assets/flag.png', // Use an Indian flag asset  
                width: 24,  
                height: 24,  
            ),  
            SizedBox(width: 10),  
            Text("+91", style: TextStyle(fontSize: 16, color: Colors.black)),  
            SizedBox(width: 10),  
            Expanded(  
                child: TextField(  
                    keyboardType: TextInputType.phone,  
                    decoration: InputDecoration(  
                        hintText: "Enter mobile number",  
                        border: InputBorder.none,  
                    ),  
                ),  
            ),  
        ],  
    ),  
,  
Spacer(),  
  
// Terms & Conditions  
Text.rich(  
    TextSpan(  
        text: "I agree to the ",  
        style: TextStyle(fontSize: 12, color: Colors.black54),  
        children: [  
            TextSpan(  
                text: "Privacy Policy",  
                style: TextStyle(fontSize: 12, color: Colors.blue),  
                recognizer: TapGestureRecognizer(),  
            ),  
        ],  
    ),  
),  
Text.rich(  
    TextSpan(  
        text: "I agree to the ",  
        style: TextStyle(fontSize: 12, color: Colors.black54),  
        children: [  
            TextSpan(  
                text: "Privacy Policy",  
                style: TextStyle(fontSize: 12, color: Colors.blue),  
                recognizer: TapGestureRecognizer(),  
            ),  
        ],  
    ),  
),
```

```
        text: "Terms & Conditions",
        style: TextStyle(color: Colors.blue, decoration: TextDecoration.underline),
    ),
    TextSpan(text: " and "),
    TextSpan(
        text: "Privacy Policy",
        style: TextStyle(color: Colors.blue, decoration: TextDecoration.underline),
    ),
],
),
),
);
},
);
}
}
```

SKIP



Enjoy faster show booking through our
recommendations tailored for you.

Login

Sign Up

+91 Enter mobile number

I agree to the [Terms & Conditions](#) and [Privacy Policy](#)

Login.dart

```
import 'package:flutter/material.dart';
import 'home.dart'; // Import the HomePage file

class LoginPage extends StatelessWidget {
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        backgroundColor: Colors.white,
```

```
elevation: 0,
leading: IconButton(
  icon: Icon(Icons.arrow_back, color: Colors.black),
  onPressed: () {
    Navigator.pop(context);
  },
),
),
body: Padding(
  padding: const EdgeInsets.symmetric(horizontal: 20),
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Image.asset('assets/logo.png', width: 120, height: 120),
      SizedBox(height: 20),
      Text(
        "Welcome Back!",
        style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold),
      ),
      SizedBox(height: 10),
      Text(
        "Login to continue",
        style: TextStyle(fontSize: 14, color: Colors.black54),
      ),
      SizedBox(height: 30),
      TextField(
        controller: emailController,
        decoration: InputDecoration(
          labelText: "Email",
          border: OutlineInputBorder(),
        ),
      ),
      SizedBox(height: 15),
      TextField(
        controller: passwordController,
        obscureText: true,
        decoration: InputDecoration(
          labelText: "Password",
          border: OutlineInputBorder(),
        ),
      ),
      SizedBox(height: 20),
      ElevatedButton(
        onPressed: () {
          String email = emailController.text;
          String password = passwordController.text;

          if (email.isNotEmpty && password.isNotEmpty) {
            Navigator.pushReplacement(
              context,
              MaterialPageRoute(builder: (context) => HomePage()),
            );
          }
        },
      ),
    ],
  ),
);
```

```
        } else {
            ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(content: Text("Please enter email and password")),
            );
        },
    child: Text("Login"),
    style: ElevatedButton.styleFrom(
        backgroundColor: Colors.red,
        foregroundColor: Colors.white,
        minimumSize: Size(double.infinity, 50),
    ),
),
SizedBox(height: 15),
TextButton(
    onPressed: () {},
    child: Text(
        "Forgot Password?",
        style: TextStyle(color: Colors.red),
    ),
),
SizedBox(height: 20),
Text("Or login with"),
SizedBox(height: 10),
ElevatedButton.icon(
    onPressed: () {},
    icon: Image.asset('assets/Google.png', width: 24, height: 24),
    label: Text("Login with Google"),
    style: ElevatedButton.styleFrom(
        backgroundColor: Colors.white,
        foregroundColor: Colors.black,
        minimumSize: Size(double.infinity, 50),
        side: BorderSide(color: Colors.black26),
    ),
),
],
),
);
}
```



Welcome Back!

Login to continue

Email

Password

Login

[Forgot Password?](#)

Or login with

Login with Google

Home.dart

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

class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
```

```
return Scaffold(  
    appBar: AppBar(  
        title: Text('Welcome Guest!'),  
        actions: [  
            IconButton(icon: Icon(Icons.search), onPressed: () {}),  
            IconButton(icon: Icon(Icons.qr_code), onPressed: () {}),  
        ],  
    ),  
    body: SingleChildScrollView(  
        child: Column(  
            crossAxisAlignment: CrossAxisAlignment.start,  
            children: [  
                // Location Bar  
                Padding(  
                    padding: const EdgeInsets.symmetric(horizontal: 15, vertical: 5),  
                    child: Text("Navi Mumbai", style: TextStyle(color: Colors.red)),  
                ),  
  
                // Location Enable Bar  
                Container(  
                    width: double.infinity,  
                    color: Colors.blue,  
                    padding: EdgeInsets.all(10),  
                    child: Text(  
                        "Enable location to discover nearby events, movies, and more.",  
                        style: TextStyle(color: Colors.white),  
                        textAlign: TextAlign.center,  
                    ),  
                ),  
  
                SizedBox(height: 10),  
  
                // Categories (Movies, Music Shows, etc.)  
                SingleChildScrollView(  
                    scrollDirection: Axis.horizontal,  
                    padding: EdgeInsets.symmetric(horizontal: 10),  
                    child: Row(  
                        children: [  
                            _categoryIcon(Icons.movie, "Movies"),  
                            _categoryIcon(Icons.music_note, "Music Shows"),  
                            _categoryIcon(Icons.theater_comedy, "Comedy Shows"),  
                            _categoryIcon(Icons.event, "Plays"),  
                        ],  
                    ),  
                ),  
            ],  
        ),  
    ),
```

```
SizedBox(height: 20),  
Padding(  
padding: const EdgeInsets.symmetric(horizontal: 10),  
child: Row(  
children: [  
Expanded(child: Image.asset('assets/event1.jpg', fit: BoxFit.cover)),  
SizedBox(width: 10),  
Expanded(child: Image.asset('assets/event2.jpg', fit: BoxFit.cover)),  
],  
),  
),
```

```
// Recommended Movies Section  
Padding(  
padding: const EdgeInsets.symmetric(horizontal: 15),  
child: Row(  
mainAxisAlignment: MainAxisAlignment.spaceBetween,  
children: [  
Text("Recommended Movies", style: TextStyle(fontSize: 18, fontWeight:  
FontWeight.bold)),  
Text("See All >", style: TextStyle(color: Colors.red)),  
],  
),  
),
```

```
SizedBox(height: 10),  
SingleChildScrollView(  
scrollDirection: Axis.horizontal,  
padding: EdgeInsets.symmetric(horizontal: 10),  
child: Row(  
children: [  
_movieCard('assets/movie1.jpg', 'PROMOTED', context),  
_movieCard('assets/movie2.jpg', "", context),  
_movieCard('assets/movie3.jpg', "", context),  
],  
),  
),
```

```
],  
),  
),  
bottomNavigationBar: BottomNavigationBar(  
selectedItemColor: Colors.red,  
unselectedItemColor: Colors.black54,
```

```
        items: [
            BottomNavigationBarItem(icon: Icon(Icons.home), label: "Home"),
            BottomNavigationBarItem(icon: Icon(Icons.movie), label: "Movies"),
            BottomNavigationBarItem(icon: Icon(Icons.event), label: "Live Events"),
            BottomNavigationBarItem(icon: Icon(Icons.person), label: "Profile"),
        ],
    ),
);
}

Widget _categoryIcon(IconData icon, String label) {
    return Column(
        children: [
            Icon(icon, size: 40),
            SizedBox(height: 5),
            Text(label, style: TextStyle(fontSize: 14)),
        ],
    );
}

Widget _movieCard(String imagePath, String tag, BuildContext context) {
    return GestureDetector(
        onTap: () {
            Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => DetailsPage()),
            );
        },
        child: Padding(
            padding: const EdgeInsets.only(right: 10),
            child: Column(
                children: [
                    Stack(
                        children: [
                            Image.asset(imagePath, width: 100, height: 150),
                            if (tag.isNotEmpty)
                                Positioned(
                                    top: 5,
                                    left: 5,
                                    child: Container(
                                        color: Colors.red,
                                        padding: EdgeInsets.symmetric(horizontal: 5, vertical: 2),
                                        child: Text(tag, style: TextStyle(color: Colors.white, fontSize: 10)),
                                    ),
                            ),
                        ],
                    ),
                ],
            ),
        ),
    );
}
```

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

[Welcome Guest!](#)

[Navi Mumbai](#)

Enable location to discover nearby events, movies, and more.

Movies Music Shows Comedy Shows Plays

Recommended Movies [See All >](#)

PROMOTED

[Home](#) [Movies](#) [Shows](#) [Profile](#)

details.dart

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'booking_page.dart'; // Import the BookingPage

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Movie Details',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: DetailsPage(),
    );
  }
}

class DetailsPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        title: Text("Emergency"),
        leading: IconButton(
          icon: Icon(Icons.arrow_back),
          onPressed: () => Navigator.pop(context),
        ),
        actions: [
          Icon(Icons.share),
          SizedBox(width: 10),
          Icon(Icons.more_vert),
          SizedBox(width: 10),
        ],
        elevation: 0,
      ),
      body: SingleChildScrollView(
        child: Padding(
          padding: EdgeInsets.all(12.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Movie Banner
              Stack(
                children: [

```

```

ClipRRect(
  borderRadius: BorderRadius.circular(10),
  child: Image.asset(
    "assets/banner1.jpg",
    height: 180,
    width: double.infinity,
    fit: BoxFit.cover,
  ),
),
),
Positioned(
  top: 70,
  left: 140,
  child: Column(
    children: [
      Icon(Icons.play_circle_fill, color: Colors.white, size: 60),
      SizedBox(height: 5),
      Text(
        "Trailers (4)",
        style: GoogleFonts.roboto(color: Colors.white, fontSize: 14),
      ),
    ],
),
),
],
),
],
),
),
SizedBox(height: 10),


// Rating Section
Row(
  children: [
    Icon(Icons.star, color: Colors.redAccent, size: 24),
    SizedBox(width: 5),
    Text(
      "8.2/10",
      style: GoogleFonts.roboto(fontSize: 18, fontWeight: FontWeight.bold),
    ),
    Text("(7.8K Votes)", style: GoogleFonts.roboto(fontSize: 14, color: Colors.grey)),
    Spacer(),
    ElevatedButton(
      onPressed: () {},
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.pink.shade100,
        shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(10)),
      ),
      child: Text("Rate now", style: TextStyle(color: Colors.redAccent)),
    ),
  ],
),
),
SizedBox(height: 10),


// Movie Info
Row(

```

```

children: [
  Container(
    padding: EdgeInsets.symmetric(horizontal: 8, vertical: 4),
    decoration: BoxDecoration(
      color: Colors.grey.shade300,
      borderRadius: BorderRadius.circular(5),
    ),
    child: Text("2D", style: GoogleFonts.roboto(fontWeight: FontWeight.bold)),
  ),
  SizedBox(width: 8),
  Text("HINDI", style: GoogleFonts.roboto(fontSize: 16, fontWeight: FontWeight.bold)),
],
),
SizedBox(height: 5),
Text(
  "2h 27m • Drama, Historical • UA • 17 Jan, 2025",
  style: GoogleFonts.roboto(fontSize: 14, color: Colors.grey.shade600),
),
SizedBox(height: 8),
Text(
  "Emergency is based on true events that unfolded in 1975. The film chronicles incidents that took pla...",
  style: GoogleFonts.roboto(fontSize: 14, color: Colors.black),
),
SizedBox(height: 8),
// Trending Section
Container(
  padding: EdgeInsets.all(10),
  decoration: BoxDecoration(
    color: Colors.blue.shade50,
    borderRadius: BorderRadius.circular(8),
  ),
  child: Row(
    children: [
      Icon(Icons.trending_up, color: Colors.blue),
      SizedBox(width: 8),
      Text("Trending", style: GoogleFonts.roboto(fontSize: 14, fontWeight: FontWeight.bold,
color: Colors.blue)),
      Spacer(),
      Text("18.28K tickets booked in last 24 hours", style: GoogleFonts.roboto(fontSize: 14)),
    ],
  ),
),
SizedBox(height: 15),
// Ads Banner
ClipRRect(
  borderRadius: BorderRadius.circular(8),
  child: Image.asset(
    "assets/prime.jpg",
    height: 100,
  )
)

```

```
width: double.infinity,
fit: BoxFit.cover,
),
),
SizedBox(height: 15),

// Book Tickets Button
Center(
  child: ElevatedButton(
    onPressed: () {
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => BookingPage()),
      );
    },
    style: ElevatedButton.styleFrom(
      backgroundColor: Colors.pinkAccent,
      shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(8)),
      padding: EdgeInsets.symmetric(vertical: 12, horizontal: 100),
    ),
    child: Text("Book tickets", style: TextStyle(fontSize: 16, color: Colors.white)),
  ),
),
],
),
),
),
);
}
}
```



Emergency



★ 8.2/10 (7.8K Votes)

Rate now

2D HINDI

2h 27m • Drama, Historical • UA • 17 Jan, 2025

Emergency is based on true events that unfolded in 1975. The film chronicles incidents that took place during the Emergency period in India.

↗ Trending 18.28K tickets booked in last 24 hours



Book tickets

MAD & PWA Lab

Journal

| | |
|-------------------|---|
| Experiment No. | 04 |
| Experiment Title. | To create an interactive Form using form widget |
| Roll No. | 37 |
| Name | Shravani Anil Patil |
| Class | D15A/D15B |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation |
| Grade: | |

EXPERIMENT NO:04

Aim: To create an interactive Form using form widget.

Theory:**Understanding Forms in Flutter:**

A **Form** in Flutter is a structured container used to collect user input through various fields like **text fields, dropdowns, checkboxes, and buttons**. Forms play a crucial role in applications that require user data entry, such as **login pages, registration forms, and feedback submissions**.

Flutter provides the **Form** widget, which works alongside **TextField** and other input elements to handle:

- **Validation** – Ensuring data accuracy before submission.
- **State management** – Storing and retrieving user input.
- **Error handling** – Displaying validation errors when needed.

By implementing form validation techniques, developers can improve data reliability and enhance the overall user experience.

Creating a Form in Flutter

When creating a form in Flutter, several essential components come into play:

1. The Form Widget

- The **Form** widget acts as a **container** for grouping multiple form fields and managing their validation.

2. GlobalKey for Identification

- A **GlobalKey<FormState>** is required to uniquely identify the form and enable operations like validation and data retrieval.

3. TextFormField for User Input

- The **TextFormField** widget allows users to enter data such as names, phone numbers, and email addresses.
- It renders a material design text field and provides validation error messages when necessary.

4. Customizing Input Fields with InputDecoration

- The **InputDecoration** property enhances usability by customizing:
 - Labels
 - Icons
 - Borders
 - Hint text

5. Implementing Form Validation

- The **validator** property ensures user input meets specific criteria before submission.
- Different input types require appropriate keyboard settings, such as:
 - **TextInputType.number** for numeric fields.
 - **TextInputType.emailAddress** for email fields.

6. Managing State Efficiently

- Proper **state management** is required to store and process user input accurately.

7. Adding a Submit Button

- A **submit button** is essential to trigger form validation and process the collected data.

Key Properties of the Form Widget

1. **key** – A **GlobalKey** that uniquely identifies the form, enabling operations like validation, resetting, and saving.
2. **child** – Contains the form fields, typically wrapped in a **Column** or **ListView** for structured layout.
3. **autovalidateMode** – Controls when the form fields should be automatically validated.

Important Methods of the Form Widget

1. **validate()**
 - Checks if all form fields are valid.
 - Returns **true** if valid, otherwise **false**.
 - Useful for ensuring form correctness before submission.
2. **save()**
 - Saves the current values of all form fields.
 - Calls the **onSaved** callback for each field.
 - Typically used after successful validation.
3. **reset()**
 - Resets the form to its **initial state**, clearing user-entered data.
4. **currentState**
 - Returns the current **FormState** associated with the form, allowing interaction with form elements.
 -

Login.dart:

```
import 'package:flutter/material.dart';
import 'home.dart'; // Import the HomePage file
```

```
class LoginPage extends StatelessWidget {
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();
```

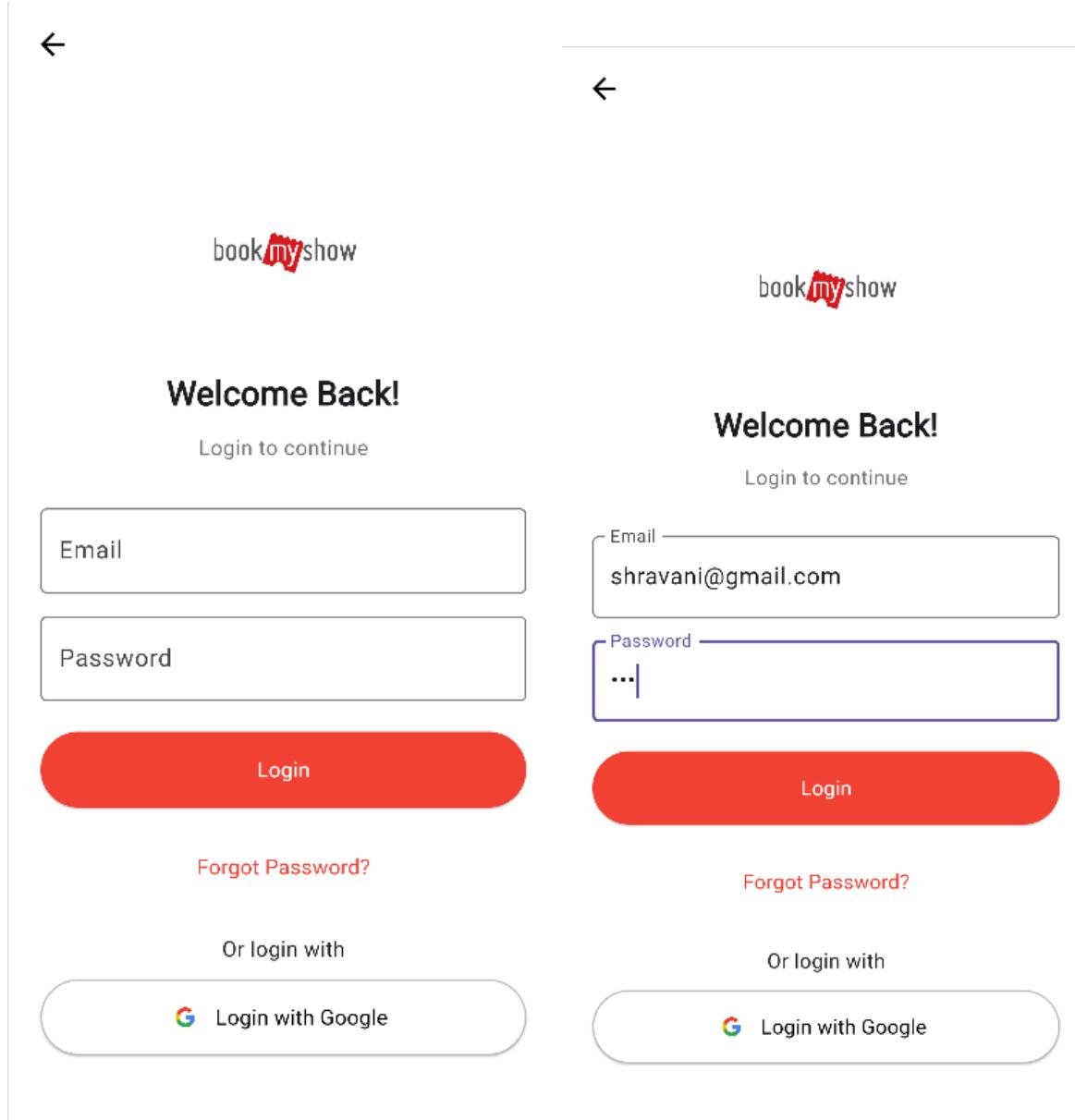
```
@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.white,
    appBar: AppBar(
      backgroundColor: Colors.white,
      elevation: 0,
      leading: IconButton(
```

```
icon: Icon(Icons.arrow_back, color: Colors.black),
onPressed: () {
  Navigator.pop(context);
},
),
),
body: Padding(
  padding: const EdgeInsets.symmetric(horizontal: 20),
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Image.asset('assets/logo.png', width: 120, height: 120),
      SizedBox(height: 20),
      Text(
        "Welcome Back!",
        style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold),
      ),
      SizedBox(height: 10),
      Text(
        "Login to continue",
        style: TextStyle(fontSize: 14, color: Colors.black54),
      ),
      SizedBox(height: 30),
      TextField(
        controller: emailController,
        decoration: InputDecoration(
          labelText: "Email",
          border: OutlineInputBorder(),
        ),
      ),
      SizedBox(height: 15),
      TextField(
        controller: passwordController,
        obscureText: true,
        decoration: InputDecoration(
          labelText: "Password",
          border: OutlineInputBorder(),
        ),
      ),
      SizedBox(height: 20),
```

```
ElevatedButton(  
    onPressed: () {  
        String email = emailController.text;  
        String password = passwordController.text;  
  
        if (email.isNotEmpty && password.isNotEmpty) {  
            Navigator.pushReplacement(  
                context,  
                MaterialPageRoute(builder: (context) => HomePage()),  
            );  
        } else {  
            ScaffoldMessenger.of(context).showSnackBar(  
                SnackBar(content: Text("Please enter email and password")),  
            );  
        }  
    },  
    child: Text("Login"),  
    style: ElevatedButton.styleFrom(  
        backgroundColor: Colors.red,  
        foregroundColor: Colors.white,  
        minimumSize: Size(double.infinity, 50),  
    ),  
,  
    SizedBox(height: 15),  
    TextButton(  
        onPressed: () {},  
        child: Text(  
            "Forgot Password?",  
            style: TextStyle(color: Colors.red),  
        ),  
    ),  
    SizedBox(height: 20),  
    Text("Or login with"),  
    SizedBox(height: 10),  
    ElevatedButton.icon(  
        onPressed: () {},  
        icon: Image.asset('assets/Google.png', width: 24, height: 24),  
        label: Text("Login with Google"),  
        style: ElevatedButton.styleFrom(  
            backgroundColor: Colors.white,
```

```
foregroundColor: Colors.black,  
minimumSize: Size(double.infinity, 50),  
side: BorderSide(color: Colors.black26),  
,  
,  
],
```

OUTPUT:



MAD & PWA Lab

Journal

| | |
|-------------------|---|
| Experiment No. | 05 |
| Experiment Title. | To apply navigation, routing and gestures in Flutter App |
| Roll No. | 37 |
| Name | Shravani Anil Patil |
| Class | D15A/D15B |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation |
| Grade: | |

EXPERIMENT NO:05

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

Theory:**Navigation, Routing, and Gestures in Flutter App Development**

Flutter provides a powerful and flexible way to manage navigation, routing, and user gestures in mobile applications. Understanding these concepts is crucial for creating smooth, user-friendly, and efficient applications. Below is a detailed explanation of each aspect.

Navigation in Flutter

Navigation in Flutter refers to moving from one screen (or page) to another within the application. Flutter uses the concept of a **Navigator** and **Routes** to handle navigation.

Navigator Class

Flutter provides various methods for navigation:

The Navigator class manages a stack of routes (screens) and allows transitioning between them. Flutter uses a stack-based approach where each new screen is pushed onto the stack, and when you go back, it is popped from the stack.

Basic Navigation Methods

- **push()**: Adds a new route to the navigation stack.
- **pop()**: Removes the current route and returns to the previous screen.
- **pushReplacement()**: Replaces the current route with a new route.
- **pushAndRemoveUntil()**: Pushes a new route and removes all previous routes until a condition is met.

Code:

Login.dart:

```
import 'package:flutter/material.dart';
import 'home.dart'; // Import the HomePage file

class LoginPage extends StatelessWidget {
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        backgroundColor: Colors.white,
        elevation: 0,
        leading: IconButton(
          icon: Icon(Icons.arrow_back, color: Colors.black),
          onPressed: () {
            Navigator.pop(context);
          },
        ),
      ),
      body: Padding(
        padding: const EdgeInsets.symmetric(horizontal: 20),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Image.asset('assets/logo.png', width: 120, height: 120),
            SizedBox(height: 20),
            Text(
              "Welcome Back!",
              style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold),
            ),
            SizedBox(height: 10),
            Text(
              "Login to continue",
              style: TextStyle(fontSize: 14, color: Colors.black54),
            ),
            SizedBox(height: 30),
          ],
        ),
      ),
    );
  }
}
```

```
TextField(  
    controller: emailController,  
    decoration: InputDecoration(  
        labelText: "Email",  
        border: OutlineInputBorder(),  
    ),  
,  
SizedBox(height: 15),  
TextField(  
    controller: passwordController,  
    obscureText: true,  
    decoration: InputDecoration(  
        labelText: "Password",  
        border: OutlineInputBorder(),  
    ),  
,  
SizedBox(height: 20),  
ElevatedButton(  
    onPressed: () {  
        String email = emailController.text;  
        String password = passwordController.text;  
  
        if (email.isNotEmpty && password.isNotEmpty) {  
            Navigator.pushReplacement(  
                context,  
                MaterialPageRoute(builder: (context) => HomePage()),  
            );  
        } else {  
            ScaffoldMessenger.of(context).showSnackBar(  
                SnackBar(content: Text("Please enter email and password")),  
            );  
        }  
    },  
    child: Text("Login"),  
    style: ElevatedButton.styleFrom(  
        backgroundColor: Colors.red,  
        foregroundColor: Colors.white,  
        minimumSize: Size(double.infinity, 50),  
    ),  
,
```

```

SizedBox(height: 15),
TextButton(
  onPressed: () {},
  child: Text(
    "Forgot Password?",
    style: TextStyle(color: Colors.red),
  ),
),
SizedBox(height: 20),
Text("Or login with"),
SizedBox(height: 10),
ElevatedButton.icon(
  onPressed: () {},
  icon: Image.asset('assets/Google.png', width: 24, height: 24),
  label: Text("Login with Google"),
  style: ElevatedButton.styleFrom(
    backgroundColor: Colors.white,
    foregroundColor: Colors.black,
    minimumSize: Size(double.infinity, 50),
    side: BorderSide(color: Colors.black26),
  ),
),
],
),
),
),
);
}
}
}

```

After clicking on login button , it will navigate to BookMyShow Home Page

Home.dart:

```

import 'package:flutter/material.dart';

class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Welcome Guest!'),
      ),
    );
  }
}

```

```
actions: [
    IconButton(icon: Icon(Icons.search), onPressed: () {}),
    IconButton(icon: Icon(Icons.qr_code), onPressed: () {}),
],
),
body: SingleChildScrollView(
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
// Location Bar
Padding(
padding: const EdgeInsets.symmetric(horizontal: 15, vertical: 5),
child: Text("Navi Mumbai", style: TextStyle(color: Colors.red)),
),

// Location Enable Bar
Container(
width: double.infinity,
color: Colors.blue,
padding: EdgeInsets.all(10),
child: Text(
"Enable location to discover nearby events, movies, and more.",
style: TextStyle(color: Colors.white),
textAlign: TextAlign.center,
),
),
),

SizedBox(height: 10),

// Categories (Movies, Music Shows, etc.)
SingleChildScrollView(
scrollDirection: Axis.horizontal,
padding: EdgeInsets.symmetric(horizontal: 10),
child: Row(
children: [
_categoryIcon(Icons.movie, "Movies"),
_categoryIcon(Icons.music_note, "Music Shows"),
_categoryIcon(Icons.theater_comedy, "Comedy Shows"),
_categoryIcon(Icons.event, "Plays"),
],
),
```

```
),
),

SizedBox(height: 20),
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 10),
  child: Row(
    children: [
      Expanded(child: Image.asset('assets/event1.jpg', fit: BoxFit.cover)),
      SizedBox(width: 10),
      Expanded(child: Image.asset('assets/event2.jpg', fit: BoxFit.cover)),
    ],
  ),
),

// Recommended Movies Section
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 15),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [
      Text("Recommended Movies", style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold)),
      Text("See All >", style: TextStyle(color: Colors.red)),
    ],
  ),
),

SizedBox(height: 10),
SingleChildScrollView(
  scrollDirection: Axis.horizontal,
  padding: EdgeInsets.symmetric(horizontal: 10),
  child: Row(
    children: [
      _movieCard('assets/movie1.jpg', 'PROMOTED'),
      _movieCard('assets/movie2.jpg', ''),
      _movieCard('assets/movie3.jpg', ''),
    ],
  ),
),
```

```
        ),
    ],
),
),
bottomNavigationBar: BottomNavigationBar(
    selectedItemColor: Colors.red,
    unselectedItemColor: Colors.black54,
    items: [
        BottomNavigationBarItem(icon: Icon(Icons.home), label: "Home"),
        BottomNavigationBarItem(icon: Icon(Icons.movie), label: "Movies"),
        BottomNavigationBarItem(icon: Icon(Icons.event), label: "Live Events"),
        BottomNavigationBarItem(icon: Icon(Icons.person), label: "Profile"),
    ],
),
);
}
```

```
Widget _categoryIcon(IconData icon, String label) {
    return Column(
        children: [
            Icon(icon, size: 40),
            SizedBox(height: 5),
            Text(label, style: TextStyle(fontSize: 14)),
        ],
    );
}

Widget _movieCard(String imagePath, String tag) {
    return Padding(
        padding: const EdgeInsets.only(right: 10),
        child: Column(
            children: [
                Stack(
                    children: [
                        Image.asset(imagePath, width: 100, height: 150),
                        if (tag.isNotEmpty)
                            Positioned(
                                top: 5,
                                left: 5,
                                child: Container(
                                    color: Colors.red,
                                    padding: EdgeInsets.symmetric(horizontal: 5, vertical: 2),
                                    child: Text(tag, style: TextStyle(color: Colors.white, fontSize: 10)),
                                ),
                            )
                    ],
                ),
            ],
        ),
    );
}
```

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

OUTPUT:

bookmyshow

bookmyshow

Welcome Back!

Login to continue

Email

Password

Login

Forgot Password?

Or login with

Login with Google

Welcome Back!

Login to continue

Email — shravani@gmail.com

Password — ...

Login

Forgot Password?

Or login with

Login with Google

Click on Login->Home

← Welcome Guest! ⌂

Navi Mumbai

Enable location to discover nearby events, movies, and more.

Movies Music Shows Comedy Shows Plays

Recommended Movies [See All >](#)

PROMOTED

Home

MAD & PWA Lab

Journal

| | |
|-------------------|--|
| Experiment No. | 06 |
| Experiment Title. | To Connect Flutter UI with fireBase database |
| Roll No. | 37 |
| Name | Shravani Anil Patil |
| Class | D15A/D15B |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS |
| Grade: | |

Experiment:06

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.

- × Create a project

Let's start with a name for your project[?]

Project name

BookMyShow

 bookmyshow-ddab2

 ves.ac.in

 You're 3 projects away from the project limit. Consider adding Firebase to an existing project or request an increased limit.

[Request an increase](#) 

Already have a Google Cloud project?
[Add Firebase to Google Cloud project](#)

[Continue](#)



BookMyShow

 Your Firebase project is ready

[Continue](#)

× Add Firebase to your Android app

1 Register app

Android package name [?](#)

com.example.my_app

App nickname (optional) [?](#)

BookMyshow

Debug signing certificate SHA-1 (optional) [?](#)

 Required for Dynamic Links, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

Register app

2 Download and then add config file

3 Add Firebase SDK



Register app

Android package name: com.example.my_app, App nickname: BookMyshow

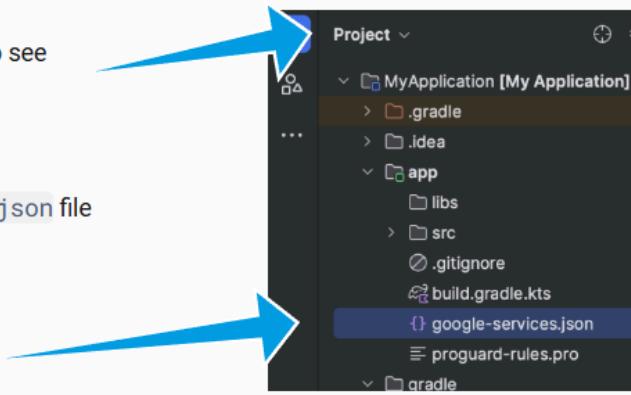
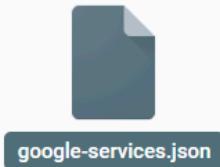
2 Download and then add config file

Instructions for Android Studio below | [Unity](#) [C++](#)

[Download google-services.json](#)

Switch to the **Project** view in Android Studio to see your project root directory.

Move your downloaded `google-services.json` file into your module (app-level) root directory.



[Next](#)

× Add Firebase to your Android app

Register app

Android package name: com.example.my_app, App nickname: BookMyshow

Download and then add config file

Add Firebase SDK

4 Next steps

You're all set!

Make sure to check out the [documentation](#) to learn how to get started with each Firebase product that you want to use in your app.

You can also explore [sample Firebase apps](#).

Or, continue to the console to explore Firebase.

Previous

Continue to console

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. | 37 |
| Name | Shravani Anil Patil |
| Class | D15A/D15B |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO4: Understand various PWA frameworks and their requirements |
| Grade: | |

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:-

```
1  var staticCacheName = "eyojana1";
2
3  self.addEventListener("install", function (e) {
4    e.waitUntil(
5      caches.open(staticCacheName).then(function (cache) {
6        return cache.addAll(["/"]);
7      })
8    );
9  });
10
11 self.addEventListener("fetch", function (event) {
12   console.log(event.request.url);
13
14 event.respondWith(
15   caches.match(event.request).then(function (response) {
16     return response || fetch(event.request);
17   })
18 );
19});
```

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:-

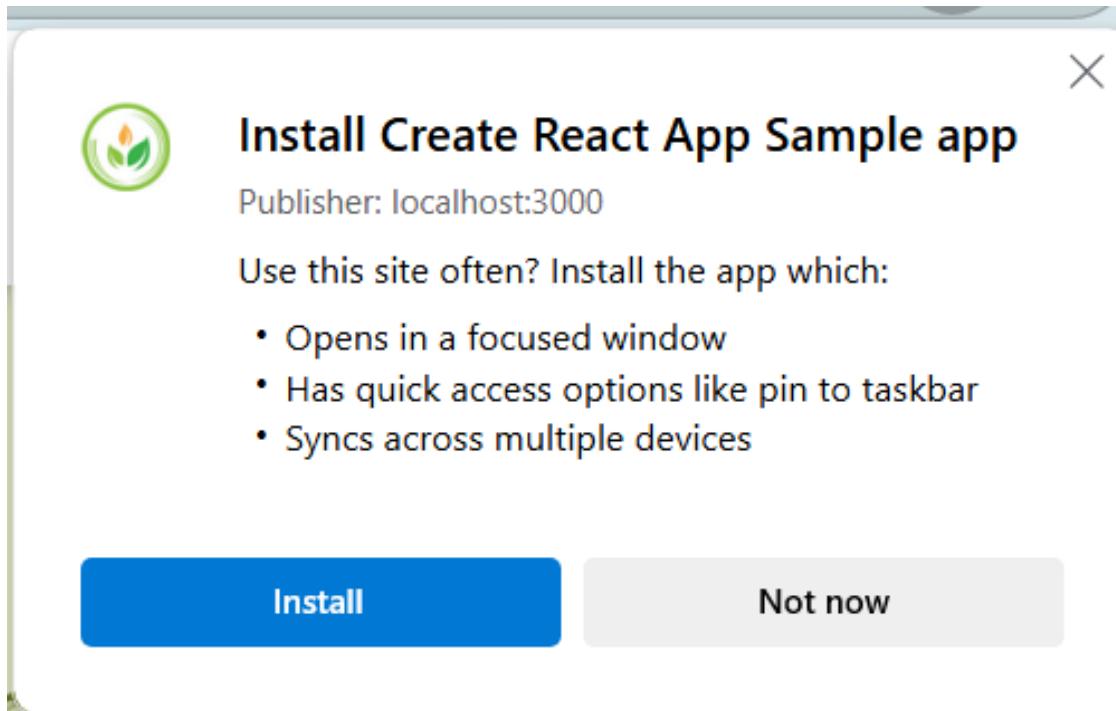
Install nodejs

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

In cmd

npm -v
npm install -g browser-sync

npm install -g live-server



E-Yojana

Home Schemes About FAQs Contact Us My Applications

Discovering government schemes that match your needs....

According to your eligibility

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

The screenshot shows a web browser window with the URL `localhost:3000`. The main content area displays the homepage of the E-Yojana platform, which is a PWA. The page features a banner with Prime Minister Narendra Modi, the text "Discovering government schemes that match your needs.... According to your eligibility", and three call-to-action buttons: "Find suitable and required schemes", "Check your eligibility", and "Review your applied Schemes in detail". The browser's address bar shows "Dimensions: Asus Zen... 853 x 1280 4. No thr..." and the title bar includes "localhost:3000", "Welcome", and the "Application" tab.

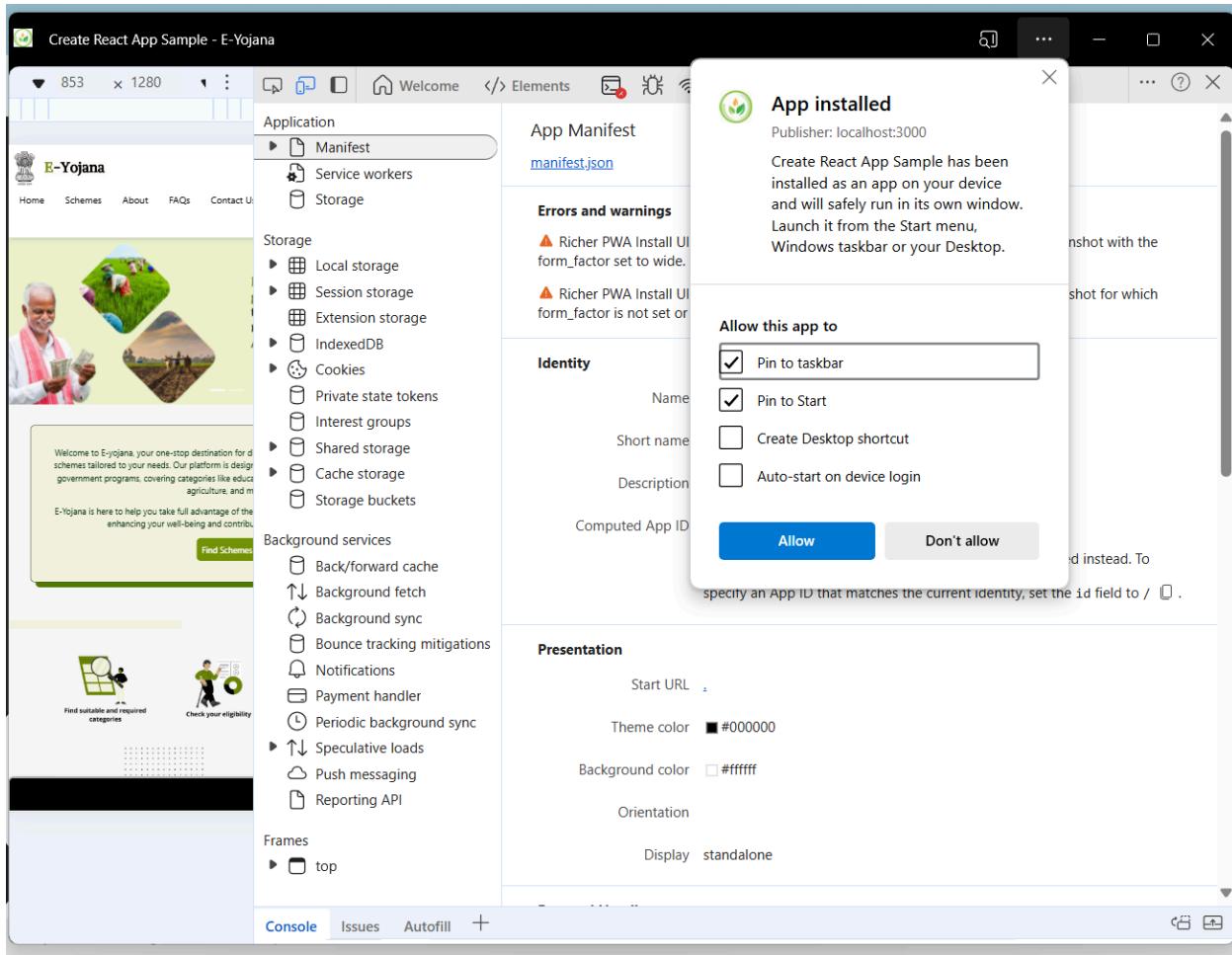
The right side of the screen is filled with the Chrome DevTools Application panel. It shows the "App Manifest" section with the file path `manifest.json`. The panel lists several errors and warnings:

- 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.

The "Identity" section contains fields for Name (set to "Create React App Sample"), Short name ("React App"), Description, and Computed App ID (`http://localhost:3000/`). A note states that `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 `/`.

The "Presentation" section includes fields for Start URL, Theme color (#000000), Background color (#ffffff), Orientation (set to "standalone"), and Display (set to "standalone").

At the bottom of the panel, there are tabs for "Console", "Issues", and "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. | 37 |
| Name | Shravani Anil Patil |
| Class | D15A/D15B |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO5: Design and Develop a responsive User Interface by applying PWA Design techniques |
| Grade: | |

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?

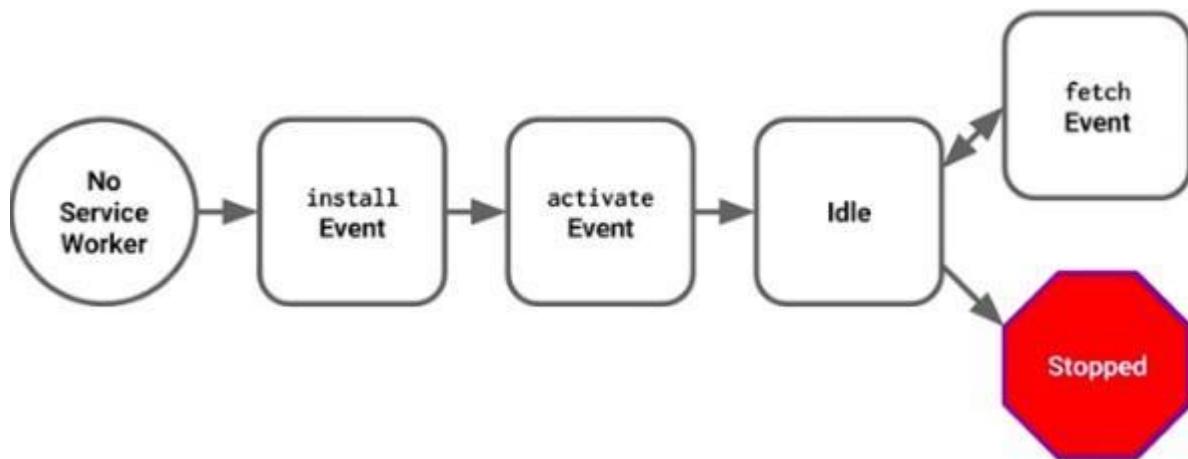
- You can't access the **Window**

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

The screenshot shows the Flipkart homepage with a blue header containing the logo and navigation links: Home, About Us, Services, Multimedia, and Contact Us. Below the header is a yellow banner with the text "Welcome to" and "For 2025 and beyond". To the right of the banner is a small image of a clothing rack.

On the right side of the screen, the Microsoft Edge developer tools are open. The "Application" tab is selected, showing the service worker status. It indicates that service worker #16 is activated and running, with a push message received at 11:30:03 AM. There is also a sync entry for "test-tag-from-devtools". The "Update Cycle" section shows the install, wait, and activate phases for this service worker.

Below the Application tab is the Network tab, which displays a list of network requests. The table includes columns for Name, Status, Type, Initiator, Size, and Time. Requests listed include index.js, flipkart.png, flipkart4.png, electronics.png, fashion.png, generate_204?RiqMYQ, home.png, books.png, log?hasfast=true&auth=SAPISIDHASH..., Create, log_event?alt=json, Create, Create, and Create. Most requests are 304 status code fetches from sw.js:23, except for the ping request which is a 0 B ping from base.js:680.

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. | 37 |
| Name | Shravani Anil Patil |
| Class | D15A/D15B |
| 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");
  }
}

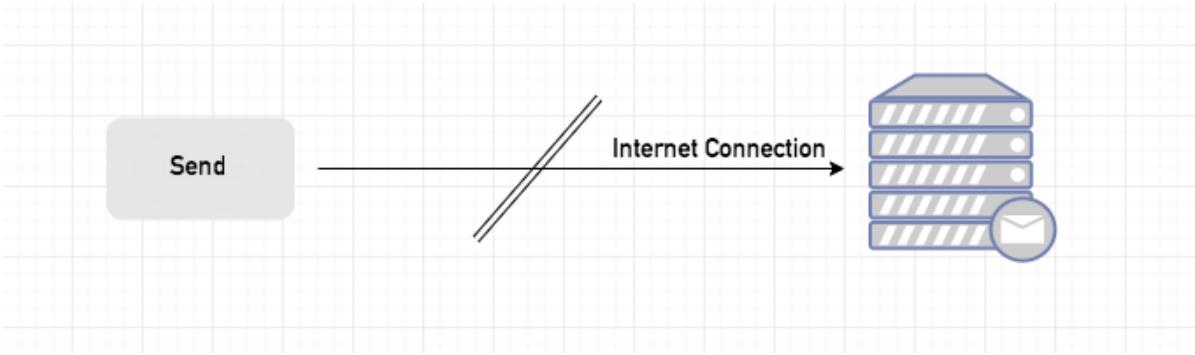
```

Sync Event

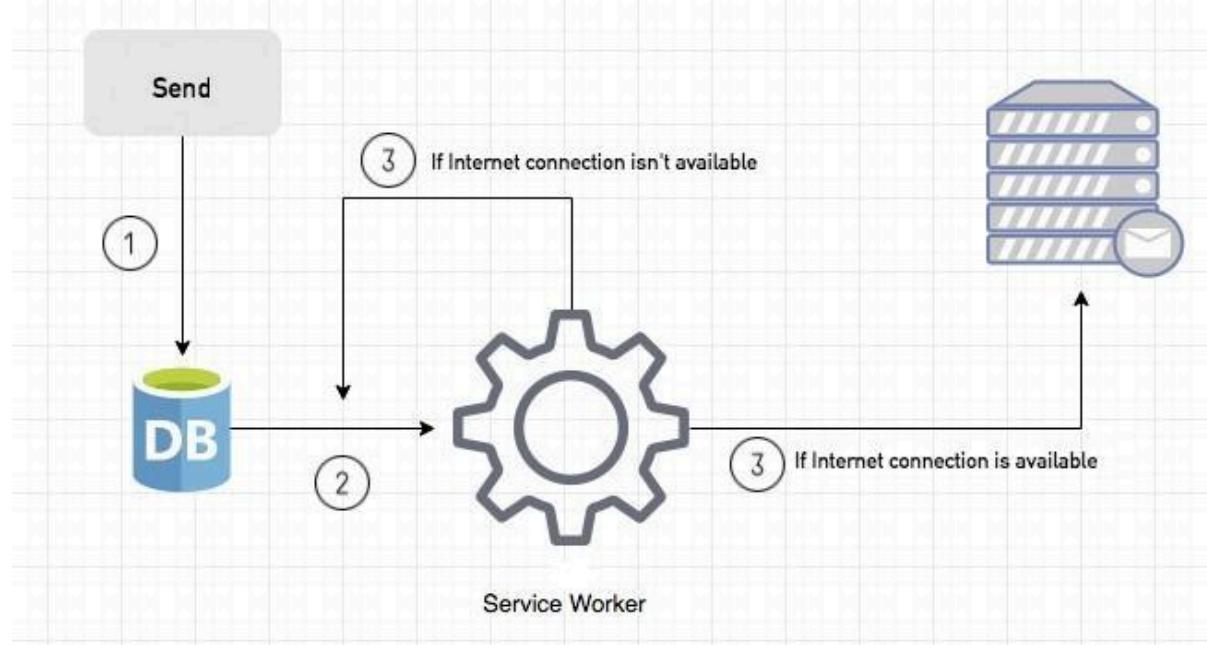
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are 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.
2. Background Sync registration.
3. **If the Internet connection is available**, all email content will be read and sent to Mail Server.
If the Internet connection is unavailable, the service worker waits until the connection is available even though the window is closed. When it is available, email content will be sent to Mail Server.

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

Event Listener for Background Sync Registration

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

Event Listener for sw.js

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

Push Event

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

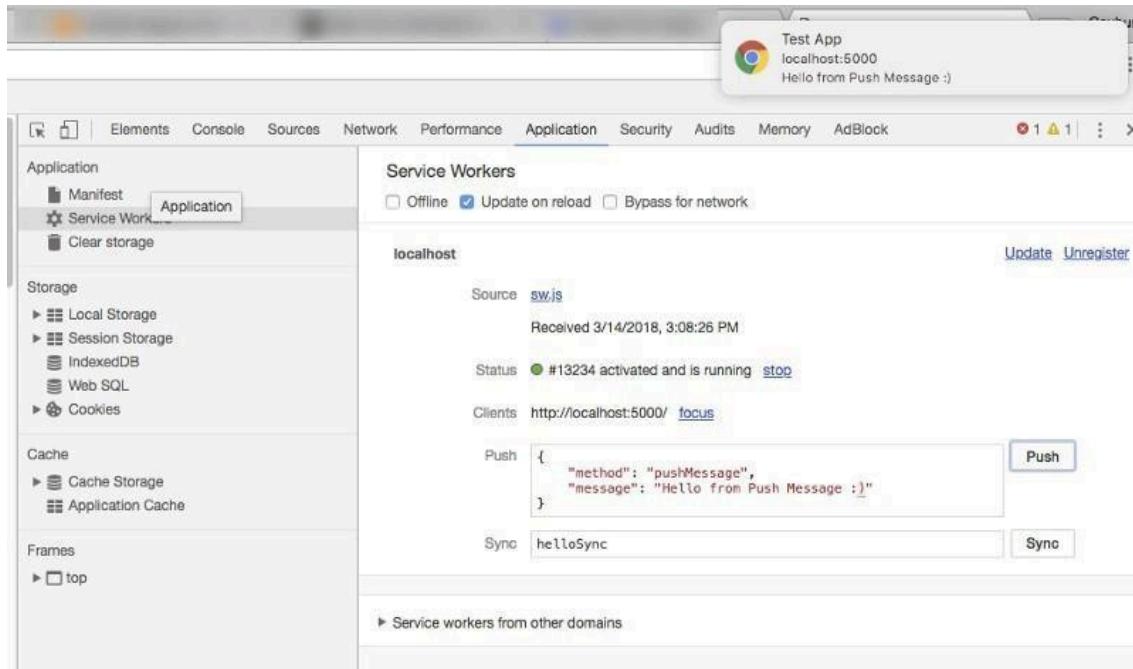
We can check in the following example.

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

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

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

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



Code

:
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

Welcome to

For 2025 and beyond

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 custom-cursor.png:1 (Not Found)

Live reload enabled. (index):201

Notification permission granted. index.js:10

Failed to load resource: net::ERR_ADDRESS_INVALID googleleads.g.doubleclick.net/pagead/id:1

Fetched successfully Service Worker registered! > ServiceWorkerRegistration index.js:4

Opened cache sw.js:20

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

Welcome to

For 2025 and beyond

Failed to load resource: net::ERR_ADDRESS_INVALID googleleads.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.

Chrome is moving towards a new experience that allows users to choose to browse without third-party cookies.

GET https://googleleads.g.doubleclick.net/pagead/id net::ERR_ADDRESS_INVALID www-embed-player.js:957

Fetched successfully Service Worker registered! ServiceWorkerRegistration {installing: ServiceWorker, active: ServiceWorker, navigationPreload: NavigationPreloadManager, scope: 'http://127.0.0.1:5500/'}

Opened cache index.js:4

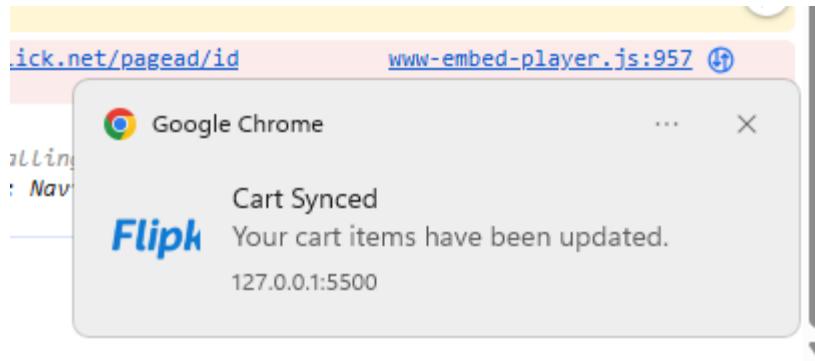
Google Chrome

Cart Synced

Flip Your cart items have been updated.

127.0.0.1:5500

Console AI assistance Network conditions



Push event

A screenshot of a browser developer tools window, specifically the Application tab. The left sidebar shows storage components: Manifest, Service worker (selected), Storage, Local storage, Session storage, Extension storage, IndexedDB, Cookies, and Private storage. The right panel displays push notifications. One notification is active: '#110 waiting to activate skipWaiting' received on 4/2/2025 at 12:23:59 PM. It has a 'Push' button. Other notifications include 'test-tag-from-devtools' with a 'Sync' button, and 'test-tag-from-devtools' with a 'Periodic sync' button. The console tab shows network activity, including errors for HEAD and POST requests to YouTube due to 'ERR_INTERNET_DISCONNECTED'. The bottom right shows a preview of a service worker notification: 'E-Commerce Alert' with the message 'Flash Sale: 50% Off on All Items!'.

Push event received:
PushEvent {isTrusted: true, data: PushMessageData, type: 'push', target: ServiceWorkerGlobalScope...}
Push Notification Text: 🔥 Flash Sale:
Push event received:
PushEvent {isTrusted: true, data: PushMessageData, type: 'push', target: ServiceWorkerGlobalScope...}
Push Notification Text: 🔥 Flash Sale:
>

E-Commerce Alert
🔥 Flash Sale: 50% Off on All Items! 🔥
127.0.0.1:5500

The screenshot shows a web browser window with the Omkar Sweets Corner website loaded. The website features a large yellow 'ॐ कार' logo, the text 'Omkar Sweets Corner', and a banner with images of sweets. Below the banner, there is a message in Marathi: 'आम्ही नाती, परंपरा आणि गोडवा जपतो'. A footer section includes links for 'Home', 'Menu', 'Offers', and 'Contact', along with social media icons for WhatsApp and GetButton.

In the background, the Google Chrome DevTools Application tab is open, specifically the Service Workers panel. It shows a service worker named 'sw.js' (version #433) is activated and running. A 'Push' message has been sent with the payload: { "method": "pushMessage", "message": "Hello!" }. The status bar indicates the message was received on 26/03/2022, 20:51:31. The DevTools also show a 'Sync' message being processed.

The DevTools Console tab displays the following log entries:

```
44 Fetch successful!
Notification permission status: granted
② Fetch successful!
Sync successful!
Push notification sent
```

A separate browser window for Google Chrome shows the website's response: 'Omkar Sweets Corner', 'Hello!', and the URL '127.0.0.1:8000'.

MAD & PWA Lab

Journal

| | |
|-------------------|---|
| Experiment No. | 10 |
| Experiment Title. | To study and implement deployment of Ecommerce PWA to GitHub Pages. |
| Roll No. | 37 |
| Name | Shravani Anil Patil |
| Class | D15A/D15B |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO5: Design and Develop a responsive User Interface by applying PWA Design techniques |
| Grade: | |

Shravani Anil Patil
D15A-37

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:

The screenshot shows two main sections of a GitHub repository.

Repository Overview:

- Repository name: flipkart
- Status: Public
- Branches: main (selected), 1 Branch
- Tags: 0 Tags
- Commits: 14 Commits (by ShravaniAnilPatil, 34 minutes ago)
- Files listed in the commit history:

 - FLIPKART (2).png (Add files via upload, 8 months ago)
 - FLIPKART1.png (Add files via upload, 8 months ago)
 - FLIPKART3.png (Add files via upload, 8 months ago)
 - Facebook.png (Add files via upload, 8 months ago)
 - LinkedIn.png (Add files via upload, 8 months ago)
 - README.md (Initial commit, 8 months ago)
 - Running It Down - Everet Almond.mp3 (Add files via upload, 8 months ago)
 - Twitter.png (Add files via upload, 8 months ago)

Repository Settings:

- Owner: ShravaniAnilPatil / flipkart
- Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, Settings (Settings is selected)
- GitHub Pages:** Your site is live at <https://shravaniapnilpatil.github.io/flipkart/> (Last deployed by ShravaniAnilPatil 35 minutes ago)
- Build and deployment:** Deploy from a branch (main), Branch (main), Save
- Learn how to add a Jekyll theme to your site.
- Your site was last deployed to the [github-pages](#) environment by the [pages build and deployment](#) workflow. Learn more about deploying to GitHub Pages using custom workflows.

Screenshot of the GitHub Deployments page for the repository ShrawaniAnilPatil / flipkart.

The sidebar shows:

- Deployments**
- All deployments** (selected)
- Environments
- github-pages
- [Manage environments](#)

The main area shows "All deployments" with the heading "Latest deployments from pinned environments". It lists one deployment for "github-pages":

- github-pages** (Active)
Last deployed 36 minutes ago
<https://shrawanianilpatil.github.io/flipkart/>

Below this, there is a section for "15 deployments" with three entries:

- Update index.html** (Active)
Deployed to github-pages by ShrawaniAnilPatil via pages-build-deployment #22
main 36 minutes ago
- Update style.css**
Deployed to github-pages by ShrawaniAnilPatil via pages-build-deployment #20
main Nov 23, 2024
- Update style.css**
Deployed to github-pages by ShrawaniAnilPatil via pages-build-deployment #19
main Aug 12, 2024

Screenshot of the deployed website at <https://shrawanianilpatil.github.io/flipkart/>.

The page has a blue header with the Flipkart logo and navigation links: Home, About Us, Services, Multimedia, Contact Us.

The main content area features a large yellow background with the text "Welcome to FLIPKART" in bold black letters. Below the text are three icons: a blue arrow pointing right, a blue heart, and a blue chart with an upward trend.

To the right of the main content, there is a sidebar with the heading "For 2025 and beyond" and a small image showing a rack of colorful clothing items.

Deployed Link: <https://shrawanianilpatil.github.io/flipkart/>

MAD & PWA Lab

Journal

| | |
|-------------------|---|
| Experiment No. | 11 |
| Experiment Title. | To use google Lighthouse PWA Analysis Tool to test the PWA functioning. |
| Roll No. | 37 |
| Name | Shravani Anil Patil |
| Class | D15A/D15B |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO6: Develop and Analyze PWA Features and deploy it over app hosting solution |
| Grade: | |

Experiment 11

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

Theory :

What is Google Lighthouse?

Google Lighthouse is an open-source automated tool designed to improve the quality, performance, accessibility, SEO, and best practices of web applications. It is commonly used by developers and website owners to audit their websites and identify areas for optimization.

Key Features of Google Lighthouse

Lighthouse provides a detailed report in five key categories:

1. Performance:
Measures page speed, load time, and responsiveness.
Includes metrics like First Contentful Paint (FCP), Largest Contentful Paint (LCP), Cumulative Layout Shift (CLS), and Time to Interactive (TTI).
2. Accessibility:
Evaluates how easily users (especially those with disabilities) can interact with a webpage.
Checks for proper contrast ratios, alt text, ARIA attributes, keyboard navigability, and screen reader compatibility.
3. Best Practices:
Ensures the website follows modern web development standards.
Checks for HTTPS security, proper image formats, JavaScript errors, and vulnerabilities.
4. SEO (Search Engine Optimization):
Analyzes on-page SEO elements to improve search engine ranking.
Checks for meta descriptions, mobile-friendliness, structured data, and canonical tags.
5. Progressive Web App (PWA):
Evaluates if a website meets PWA standards, allowing it to function like a native app.
Checks for service workers, offline capabilities, and fast load times.

How Lighthouse Scores Are Calculated

Lighthouse assigns scores between 0 and 100 based on various metrics.

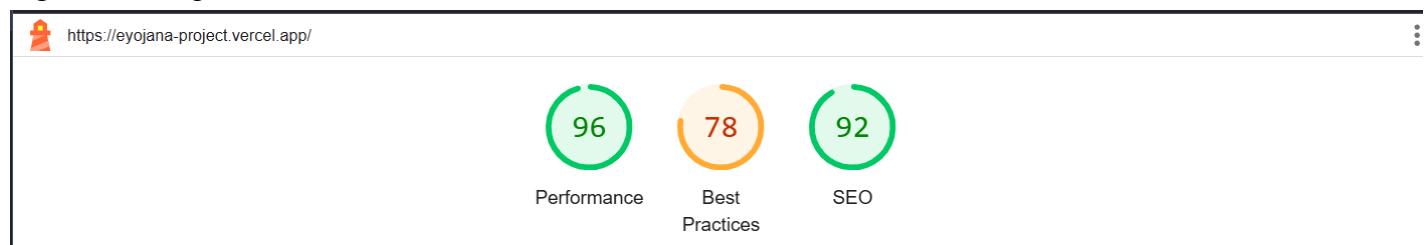
- 90-100: Excellent
- 50-89: Needs improvement
- 0-49: Poor

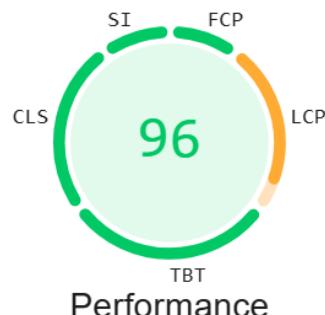
Each score is based on real-world performance data and lab tests.

Deployed website:

The screenshot shows the deployed version of the E-Yojana website. The top navigation bar includes links for Home, Schemes, About, FAQs, Contact Us, My Applications, Login, and a user profile icon. The main hero section features a large image of a smiling man in a white shirt and red dhoti holding Indian rupee notes. Overlaid on this are two diamond-shaped images: one of people working in a green field and another of a plowed field with a bullock cart. To the right, the text "Discovering government schemes that match your needs.... According to your eligibility" is displayed. Below this, a circular performance report card is shown with a large green "96" in the center. The card is divided into segments labeled SI, FCP, LCP, CLS, and TBT, each with a corresponding color (green, orange, yellow, blue, and light blue). To the right of the card, a legend indicates scores: ▲ 0–49 (red), ■ 50–89 (orange), and ● 90–100 (green). The bottom of the page shows a sidebar with a mobile view of the site and navigation links for Home, Schemes, About, FAQs, Contact Us, My Applications, and a "Find Schemes" button.

Lighthouse Report:





Values are estimated and may vary. The [performance score is calculated](#) directly from these metrics. [See calculator.](#)

▲ 0–49 ■ 50–89 ● 90–100

METRICS

● First Contentful Paint

0.6 s

■ Largest Contentful Paint

1.3 s

● Total Blocking Time

0 ms

● Cumulative Layout Shift

0.007

● Speed Index

0.6 s

DIAGNOSTICS

▲ Serve images in next-gen formats — Potential savings of 1,113 KiB

▼

▲ Properly size images — Potential savings of 916 KiB

▼

▲ Preload Largest Contentful Paint image — Potential savings of 40 ms

▼

▲ Reduce unused JavaScript — Potential savings of 267 KiB

▼

▲ Largest Contentful Paint element — 1,320 ms

▼

▲ Reduce unused CSS — Potential savings of 33 KiB

▼

▲ Eliminate render-blocking resources — Potential savings of 250 ms

▼

■ Image elements do not have explicit `width` and `height`

▼

■ Minify JavaScript — Potential savings of 25 KiB

▼

Conclusion: Thus we successfully used google Lighthouse PWA Analysis Tool for testing the PWA functioning.