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 **Ronit Santwani** of **D15A** semester **VI,** have successfully completed necessary experiments in the **MAD & PWA Lab** under my supervision in **VES Institute of Technology** during the academic year **2024-2025.**

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Project Title: Event Track **Roll No:48**

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

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

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers : Mrs. Kajal Joseph.

Email : kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: Event Track **Roll No:48**

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: Event Track **Roll No:48**

Index

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

Project Title: Event Track **Roll No:48**

MAD & PWA Lab
Journal

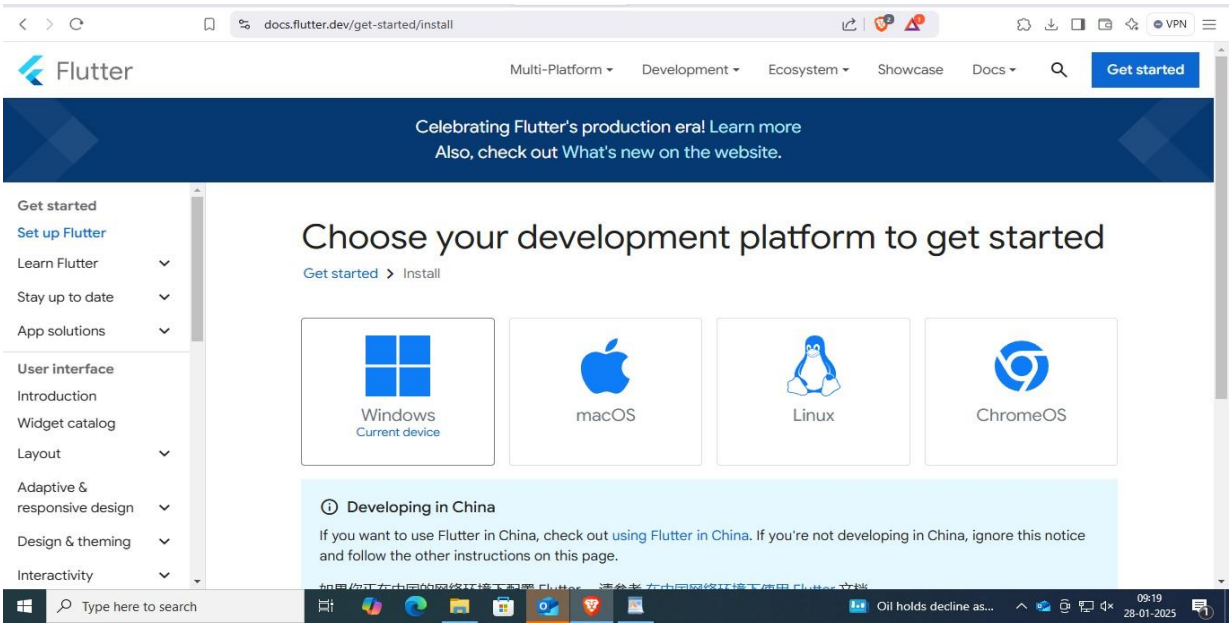
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	48
Name	Ronit Santwani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

EXPERIMENT NO: - 01

Name:- Ronit Santwani **Class:-** D15A **Roll:No: -** 48

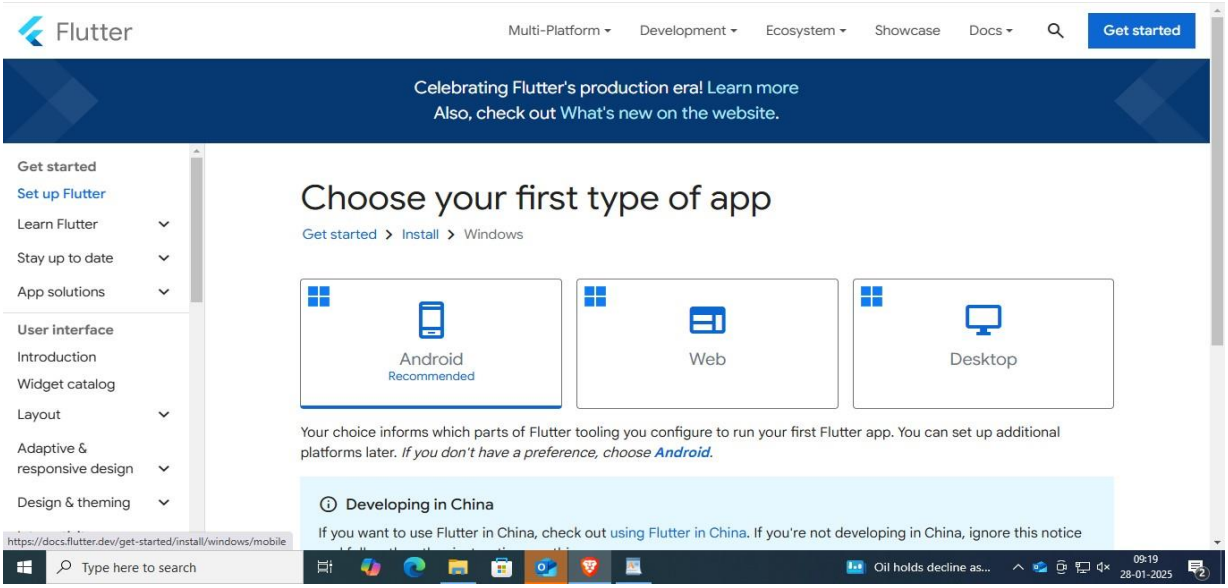
AIM: - Installation and Configuration of Flutter Environment.

Step 1: Go to the official Flutter website: https://docs.flutter.dev/get-started/install

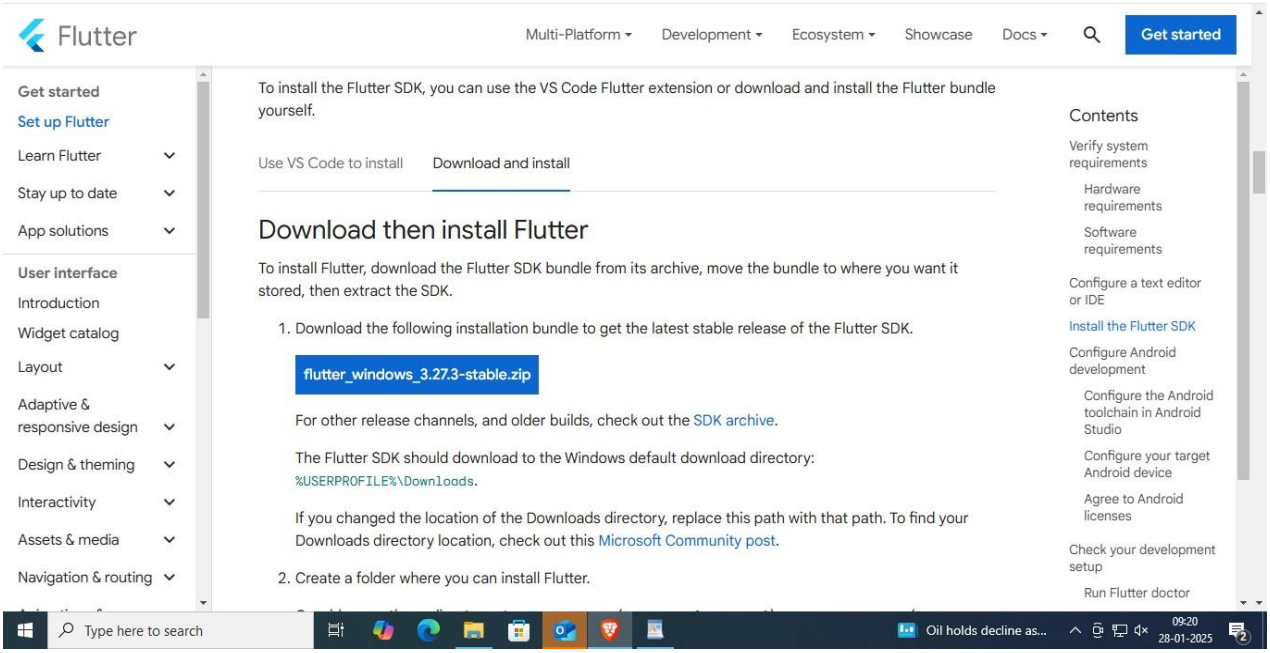


Step 2: To download the latest Flutter SDK, click on the Windows icon > Android

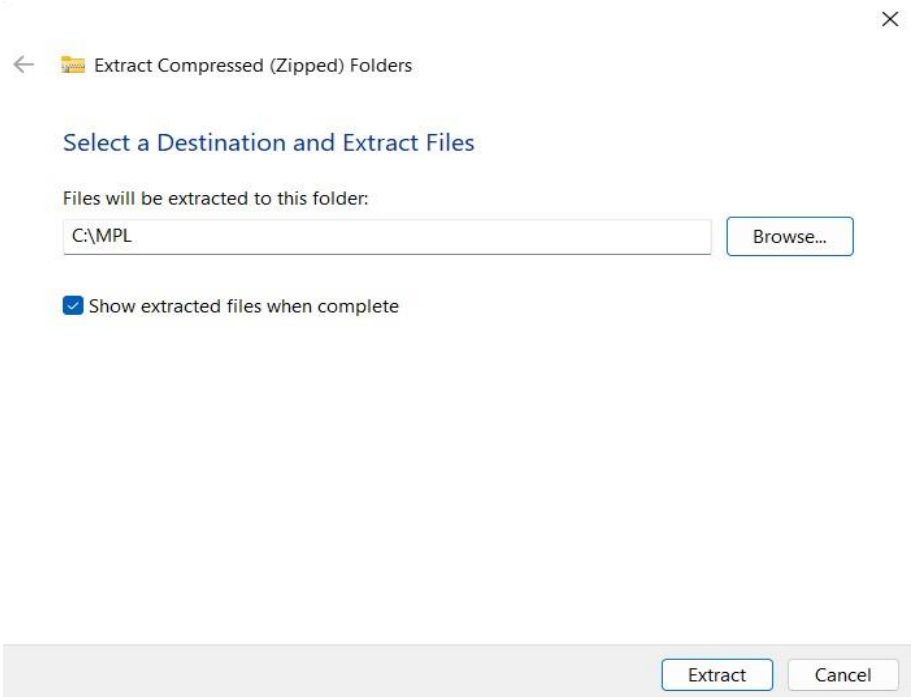
Step



3: For Windows, download the stable release (a .zip file).



Step 4: Extract the ZIP file to a folder (e.g., C:\flutter).

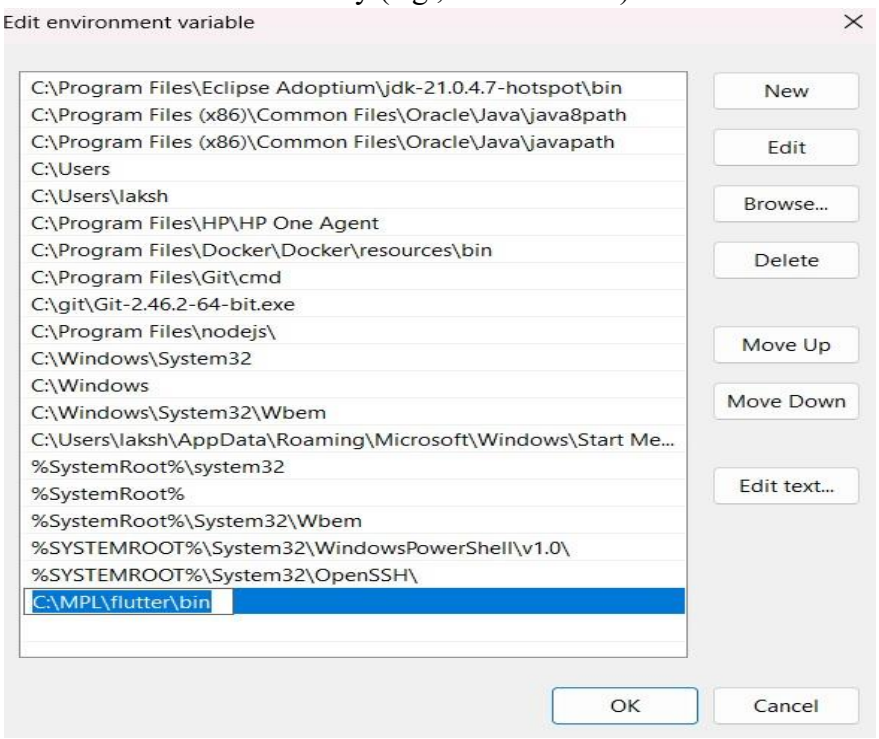


Step 5 :- Add Flutter to System PATH

Right-click on the Start Menu > System > Advanced system settings > Environment Variables.

Under System Variables, find Path and click Edit.

Add the full path to the flutter/bin directory (e.g., C:\flutter\bin).



Step

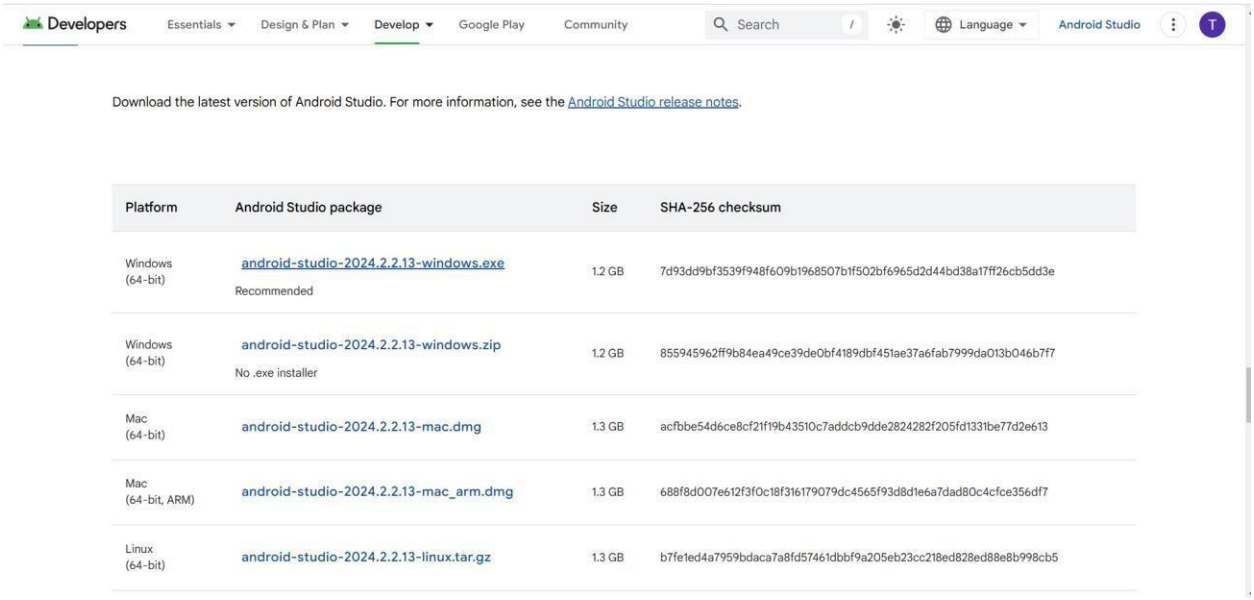
Step 6 : - Now, run the $ flutter command in command prompt.

```
Microsoft Windows [Version 10.0.26100.2894]
(c) Microsoft Corporation. All rights reserved.

C:\Users\laksh>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:
```

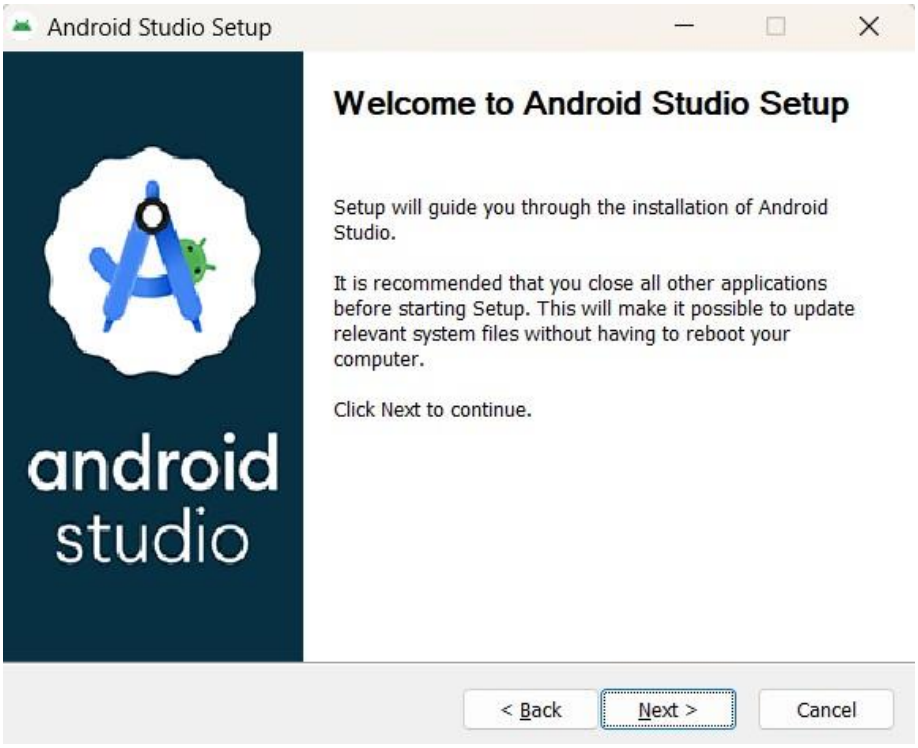
7:- Run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation

```
C:\Users\laksh>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.26100.2894], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[X] Android toolchain - develop for Android devices
    X Unable to locate Android SDK.
      Install Android Studio from: https://developer.android.com/studio/index.html
      On first launch it will assist you in installing the Android SDK components.
      (or visit https://flutter.dev/to/windows-android-setup for detailed instructions).
      If the Android SDK has been installed to a custom location, please use
      `flutter config --android-sdk` to update to that location.

[√] Chrome - develop for the web
[X] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows apps.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[!] Android Studio (not installed)
[√] VS Code (version 1.96.3)
[√] VS Code, 64-bit edition (version 1.92.2)
[√] Connected device (3 available)
[√] Network resources

! Doctor found issues in 3 categories.
```

Step 8 : - Go to Android Studio and download the installer.

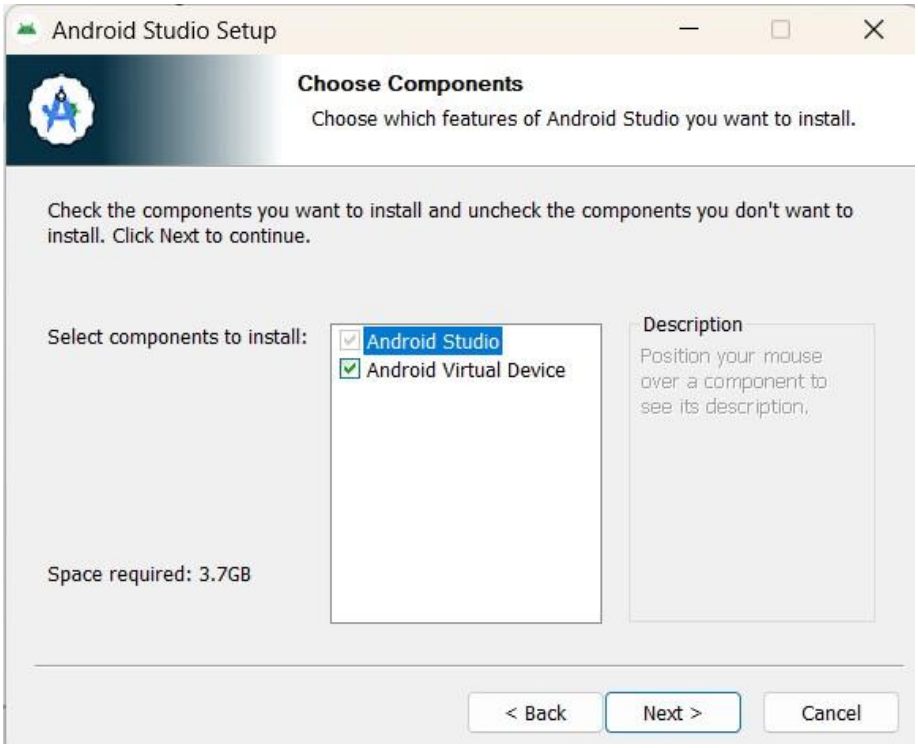


Step 8.1: - When the download is complete, open the .exe file and run it. You will get the following dialog box

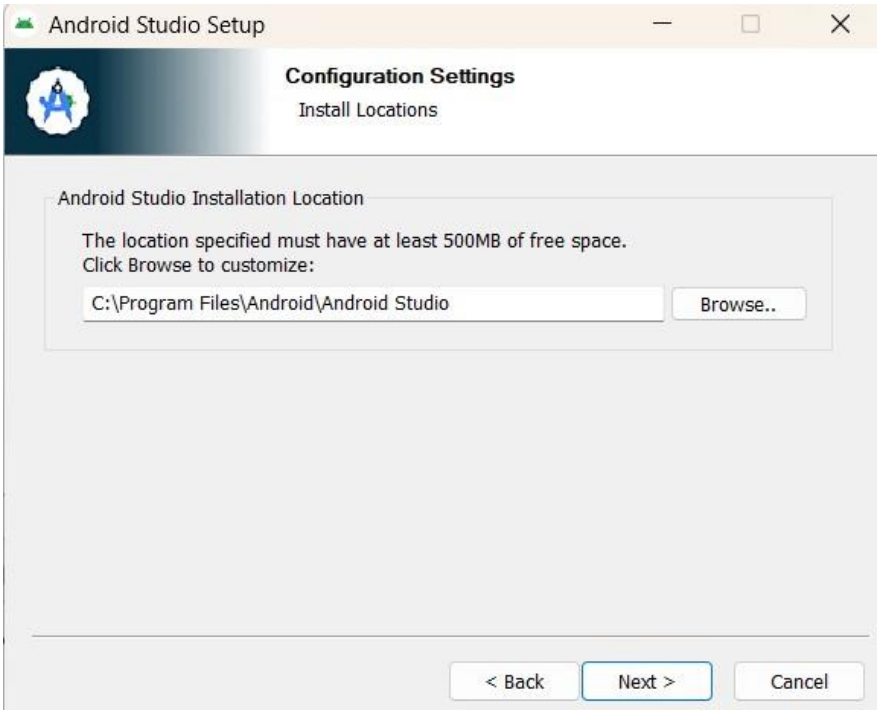


Step 8.2: - Select all the Checkboxes and Click on ‘Next’ Button.

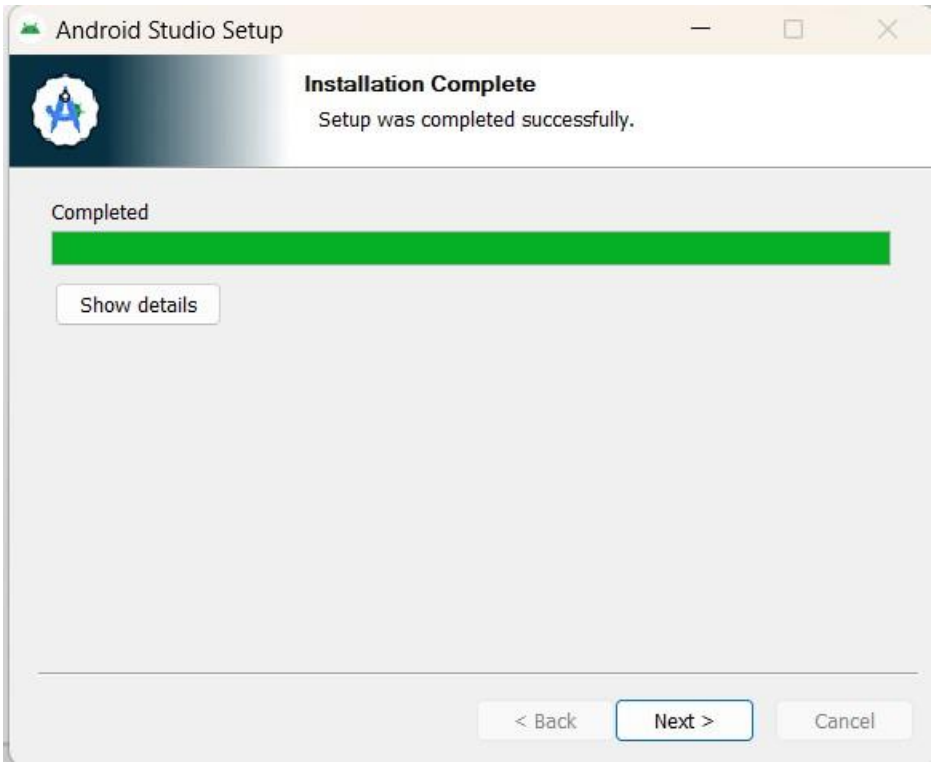
Step

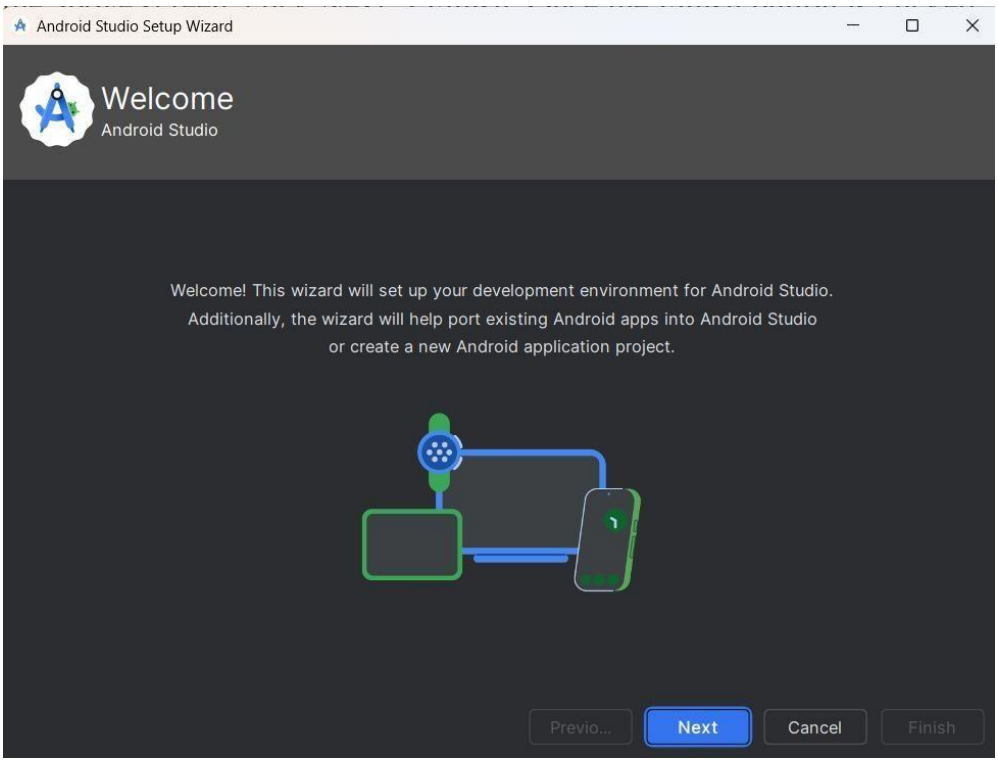


8.3: - Change the destination as per your convenience and click on 'Next' Button.



Step 8.4: - Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.

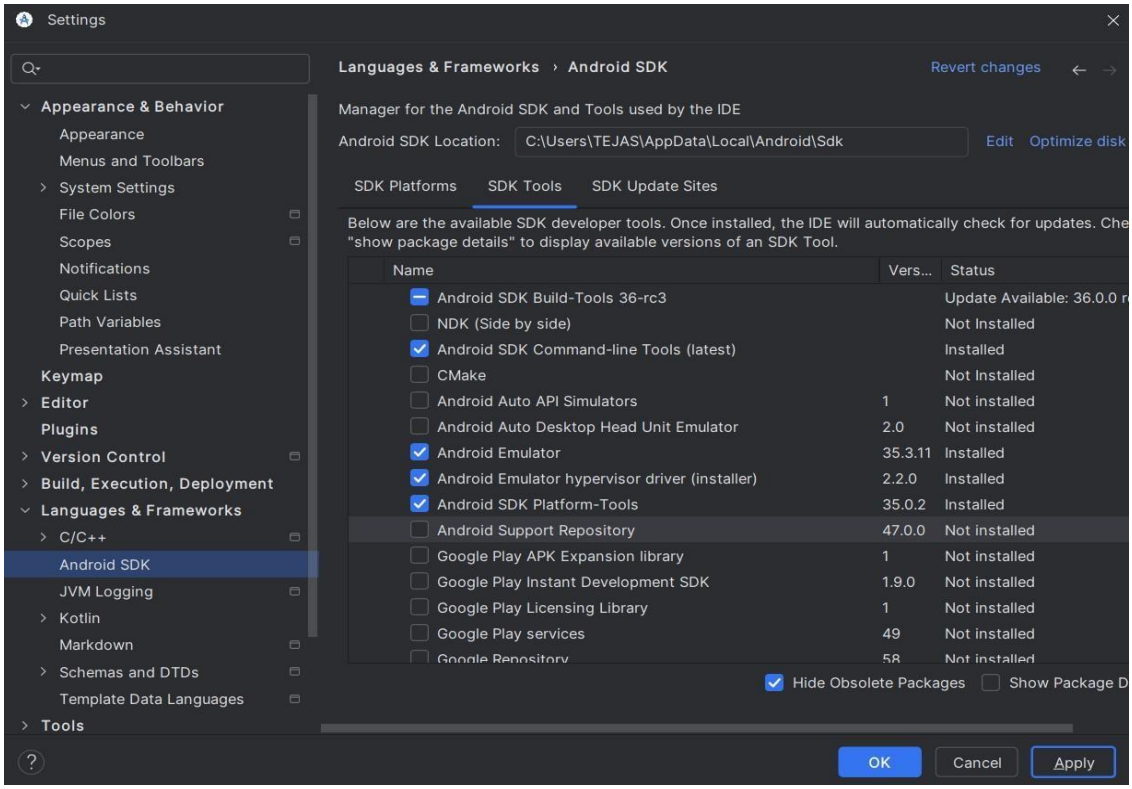




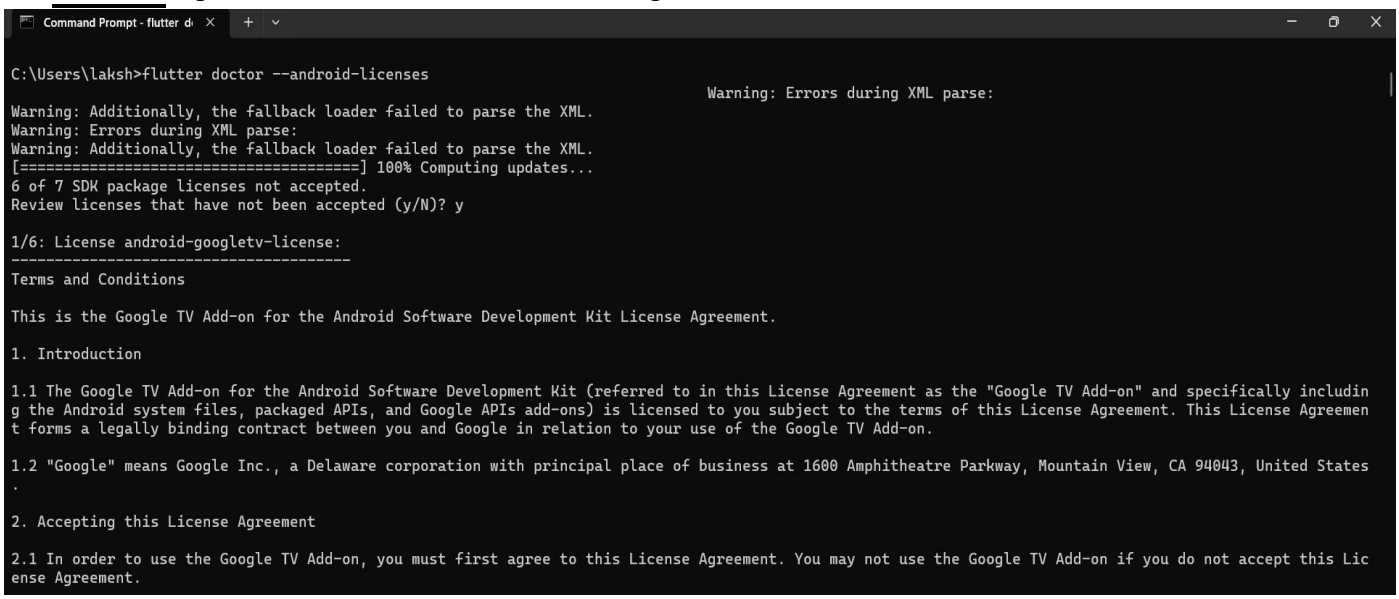
Step 8.5: - Go to Preferences > Appearance & Behavior > System Settings > Android SDK.

Select the SDK Tools tab and check Android SDK Command-line Tools and Install it.

Step



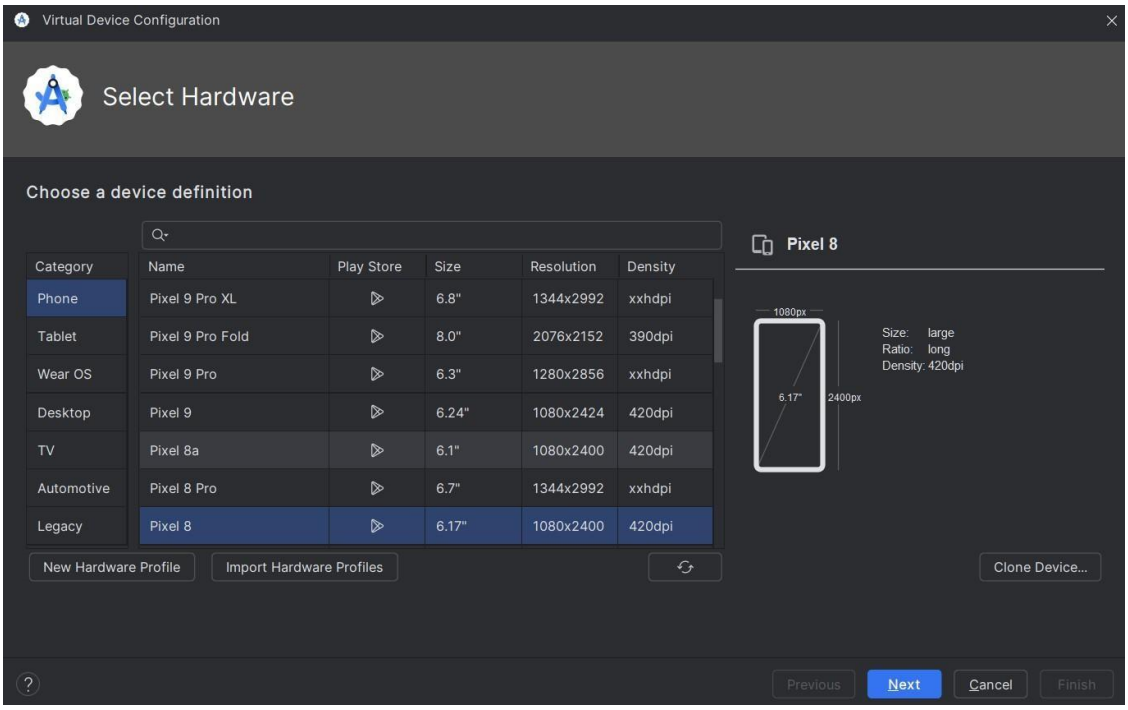
9: - Open a terminal and run the following command




C:\Users\laksh>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.26100.2894], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[√] Android toolchain - develop for Android devices (Android SDK version 35.0.1)
[√] Chrome - develop for the web
[X] Visual Studio - develop Windows apps
X Visual Studio not installed; this is necessary to develop Windows apps.
Download at https://visualstudio.microsoft.com/downloads/.
Please install the "Desktop development with C++" workload, including all of its default components
[√] Android Studio (version 2024.2)
[√] VS Code (version 1.96.3)
[√] VS Code, 64-bit edition (version 1.92.2)
[√] Connected device (3 available)
[√] Network resources
! Doctor found issues in 1 category.

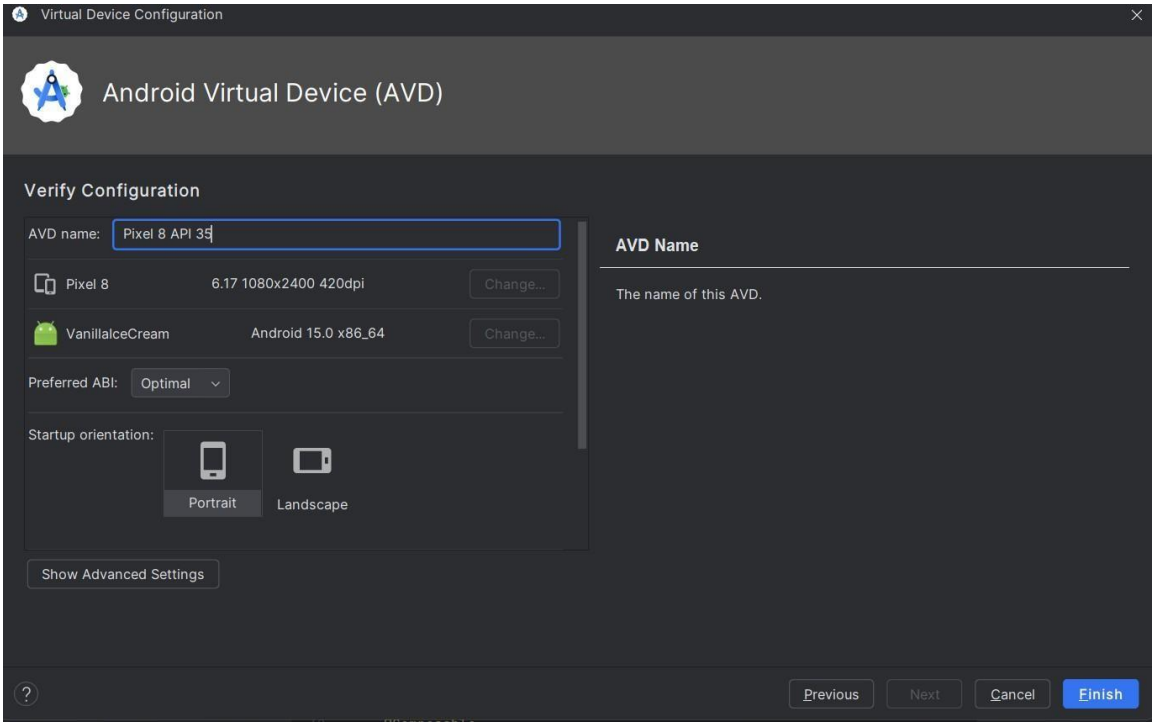
Step

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



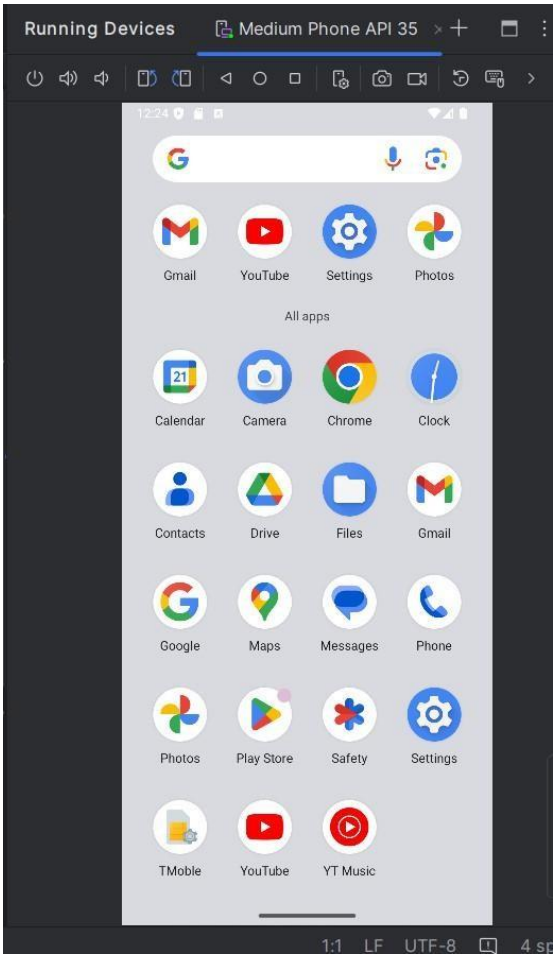
Step 10.1: - Open Android Studio and go to Tools > AVD Manager. Create a new virtual device.

Step

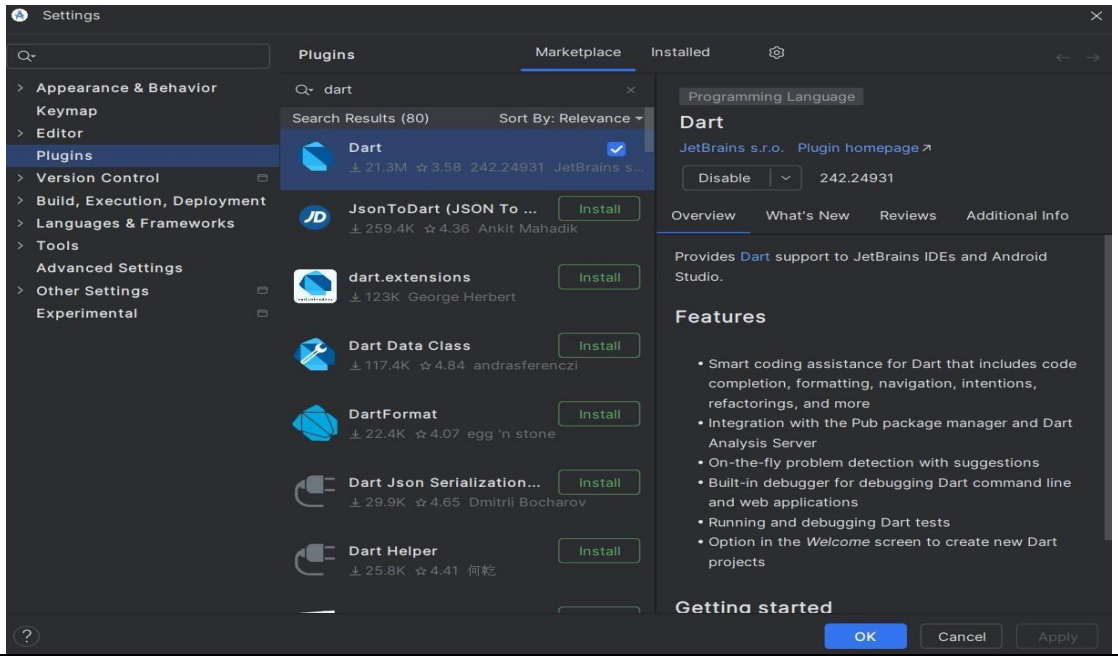


10.2: - Click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen

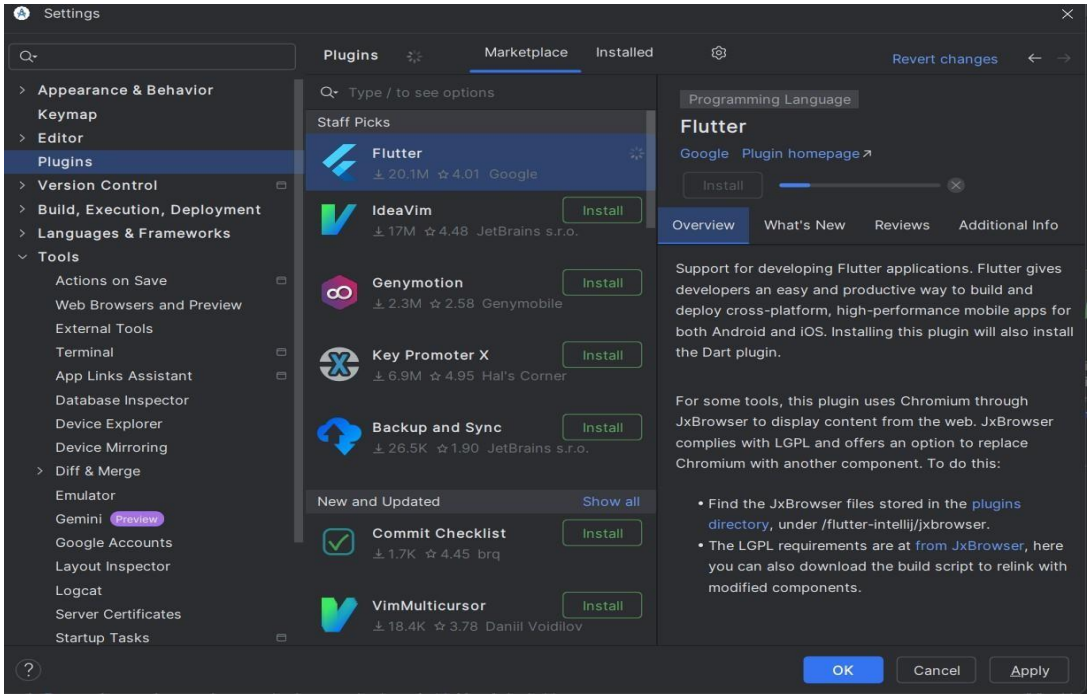
Step



11: - Now, install Flutter and Dart plugin for building Flutter application in Android Studio.



Step



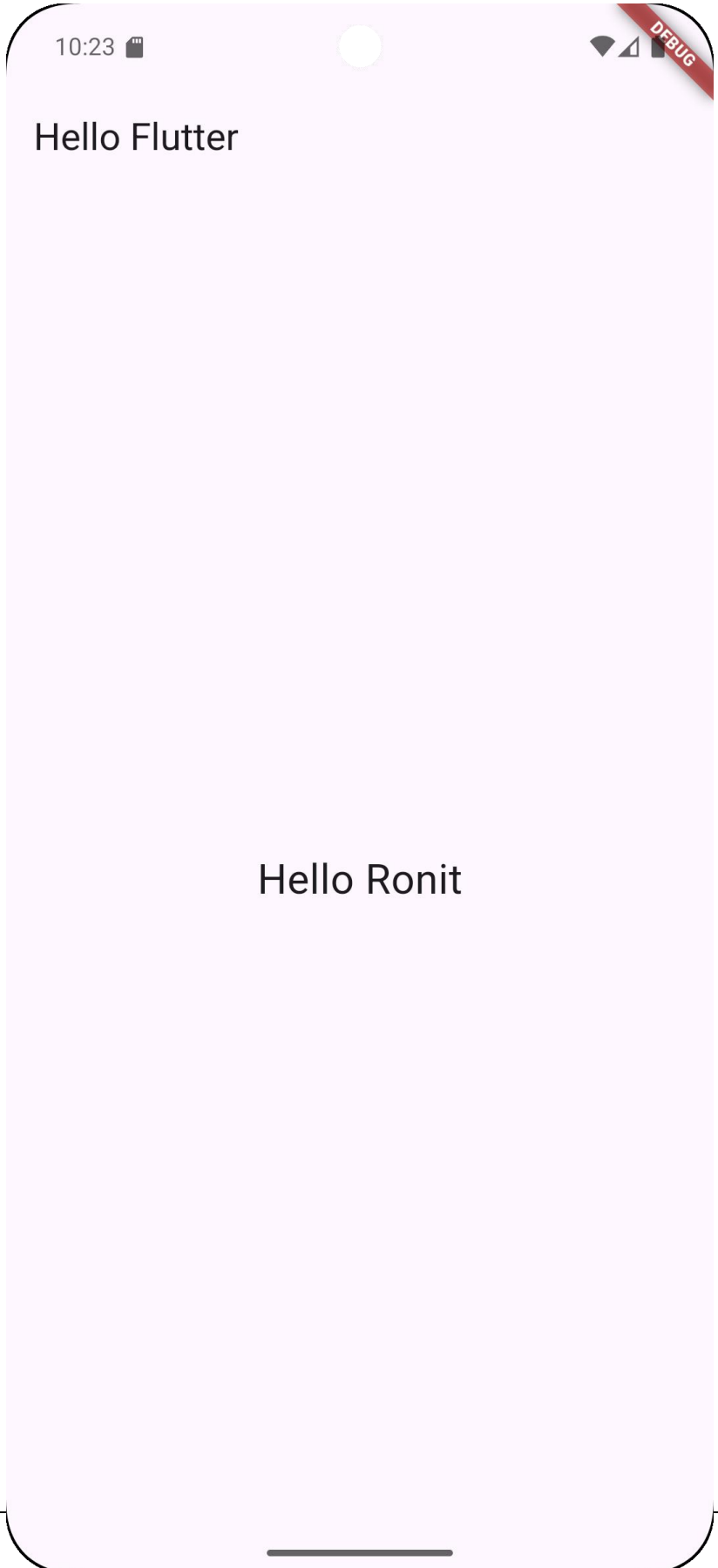
These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself

Step 11.1: - Open the Android Studio and then go to File->Settings->Plugins. Now, search the Flutter plugin. If found, select Flutter plugin and click install

Step 11.2: - Restart the Android Studio

12: - Go to File > New Project > Create Flutter Project, then select the project name and location, and click Next to proceed.

Step



Project Title: **Event Track** **Roll No:48**

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	48
Name	Ronit Santwani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Name: Ronit Santwani
Class:D15A Roll
No:48

MPL LAB-2

Aim: To design flutter UI using common widgets

Code :

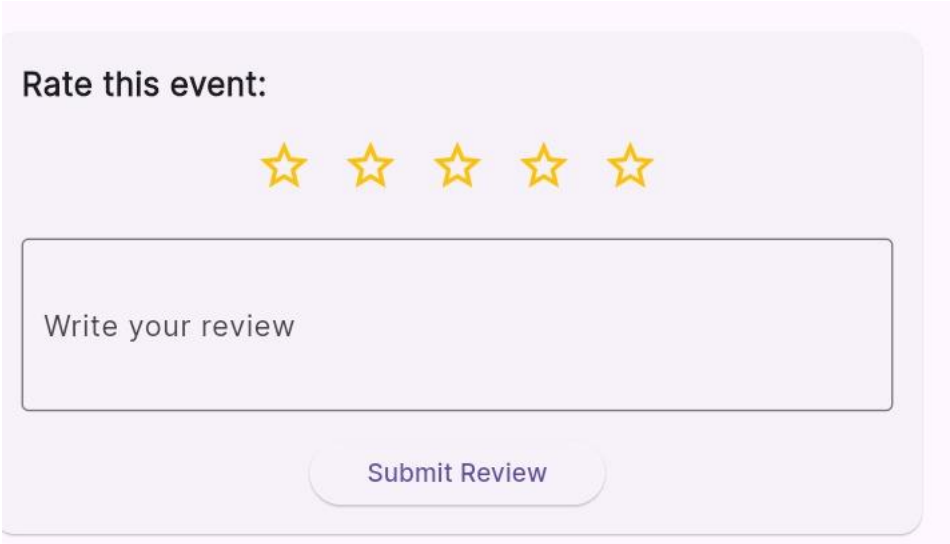
```
import 'package:flutter/material.dart';


class LoginScreen extends StatefulWidget {
  @override
  _LoginScreenState createState() => _LoginScreenState();
}


class _LoginScreenState extends State<LoginScreen> {   final
TextEditingController emailController = TextEditingController();   final
TextEditingController passwordController = TextEditingController();


  @override
  Widget    build(BuildContext    context)    {
return Scaffold(
     backgroundColor: Colors.black, // Dark background
body: Center(         child: Padding(
          padding:   EdgeInsets.symmetric(horizontal:   20),
child: Column(
            mainAxisSize: MainAxisSize.min,
children: [              Text(              "Login",
style: TextStyle(                   fontSize: 32,
                fontWeight: FontWeight.bold,
                color: Colors.limeAccent,
              ),
            ),
```

```
            SizedBox(height:          20),
_buildTextField(                controller:
emailController,                    label:
"Email",                            icon:
Icons.email_outlined,
            ),
            SizedBox(height: 15),
_buildTextField(                controller:
passwordController,                 label:
"Password",               icon:
Icons.lock_outline,
isPassword: true,
            ),
            SizedBox(height:          10),
Align(
              alignment: Alignment.centerRight,
child: TextButton(                  onPressed: ()
{},               child: Text(                 "Forget
Password",
                style: TextStyle(color: Colors.limeAccent),
              ),
            ),
            ),
            SizedBox(height: 10),
ElevatedButton(                 onPressed: ()
{},               style:
ElevatedButton.styleFrom(
backgroundColor: Colors.black,
padding: EdgeInsets.symmetric(vertical:
14, horizontal: 100),               shape:
RoundedRectangleBorder(
borderRadius: BorderRadius.circular(8),
```

```
                ),
),              child:
Text(
"Login",
                style: TextStyle(color: Colors.limeAccent),
              ),
            ),
            SizedBox(height: 20),
TextButton(
onPressed: () {},               child:
Text.rich(                TextSpan(
                  text: "Create a New Account ? ",
style: TextStyle(color: Colors.limeAccent),
children: [                  TextSpan(                  text:
"Sign Up",                    style: TextStyle(
fontWeight: FontWeight.bold,                    color:
Colors.limeAccent,
                    ),
                  ),
                ],
              ),
            ),
          ),
        ],
      ),
    ),
  );
 }

  Widget  _buildTextField({          required
TextEditingController controller,     required
```

```
String label,        required IconData icon,
bool isPassword = false,
 }) {
   return TextField(      controller:
controller,              obscureText:
isPassword,              decoration:
InputDecoration(
       filled:   true,                    fillColor:
Colors.limeAccent,          prefixIcon: Icon(icon,
color: Colors.black),       labelText: label,
       labelStyle:  TextStyle(color:  Colors.black),
border: OutlineInputBorder(         borderRadius:
BorderRadius.circular(12),              borderSide:
BorderSide.none,
       ),
     ),
   );
 }
}
```

Output:



Project Title: Event Track **Roll No:48**

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	48
Name	Ronit Santwani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Name: Ronit Santwani

Class: D15A Roll: 48

Experiment 03: Including Icons, Images, and Fonts in a Flutter App

Introduction

Flutter allows seamless integration of icons, images, and custom fonts to enhance the visual appeal of mobile applications. In this document, we will explore how to include and use these assets effectively.

1. Including Icons in Flutter

Flutter provides built-in icons via the `Icons` class and allows the use of custom icons through the `pubspec.yaml` file.

1.1 Using Built-in Icons

Flutter provides an extensive set of material icons that can be used as follows:

```
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("Flutter Icons")),
body: Center(           child: Icon(
Icons.favorite,
           color: Colors.red,
size: 50.0,
         ),
```

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

1.2 Using Custom Icons

To use custom icons, first, download or generate an icon font using FlutterIcon and add it to the pubspec.yaml file:

```
flutter:
fonts:
- family: CustomIcons       fonts:
- asset: assets/fonts/custom_icons.ttf
```

Use it in your app:

```
Icon(IconData(0xe900, fontFamily: 'CustomIcons'))
```

2. Including Images in Flutter

Flutter supports various ways to include images, such as from the assets folder, network URLs, and memory.

2.1 Adding Image Assets

1. Place images inside the assets/images/ directory.
2. Declare them in pubspec.yaml:

```
flutter:
assets:
  - assets/images/sample.png
```

3. Use them in your app:

```
Image.asset('assets/images/sample.png', width: 200, height: 200)
```

2.2 Using Network Images

Load images directly from the internet:

```
Image.network('https://example.com/sample.jpg')
```

3. Including Custom Fonts in Flutter

Custom fonts can enhance the UI by providing unique typography.

3.1 Adding Custom Fonts

1. Place font files in `assets/fonts/`.
2. Declare them in `pubspec.yaml`:

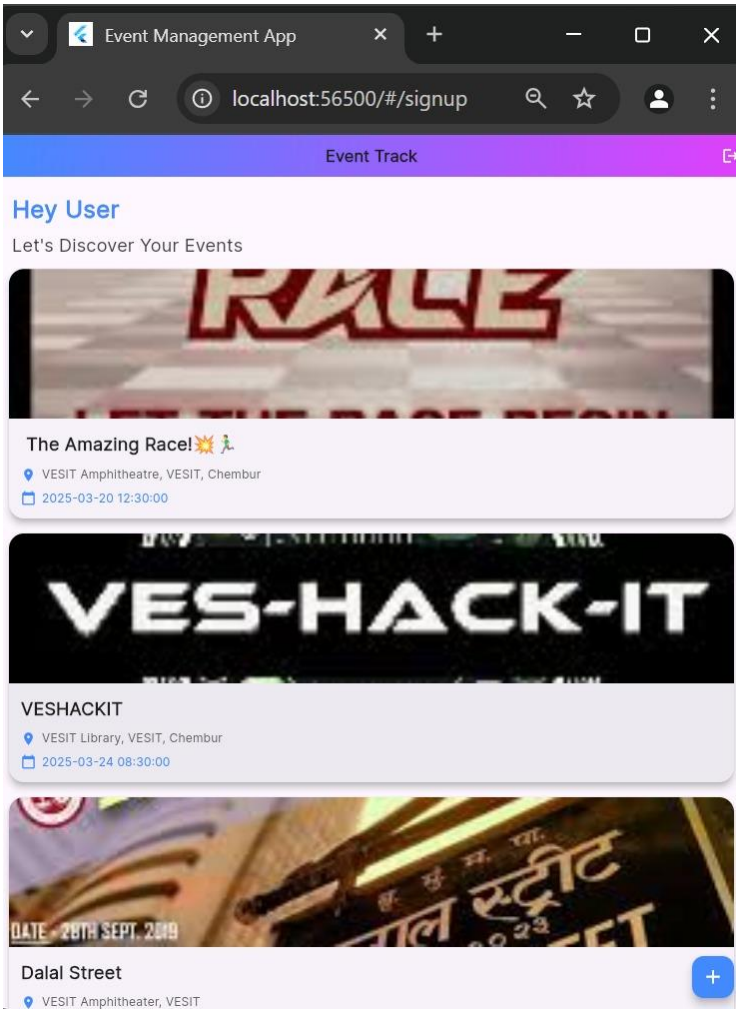
```
flutter:
fonts:
- family: CustomFont       fonts:
- asset: assets/fonts/CustomFont-Regular.ttf
```

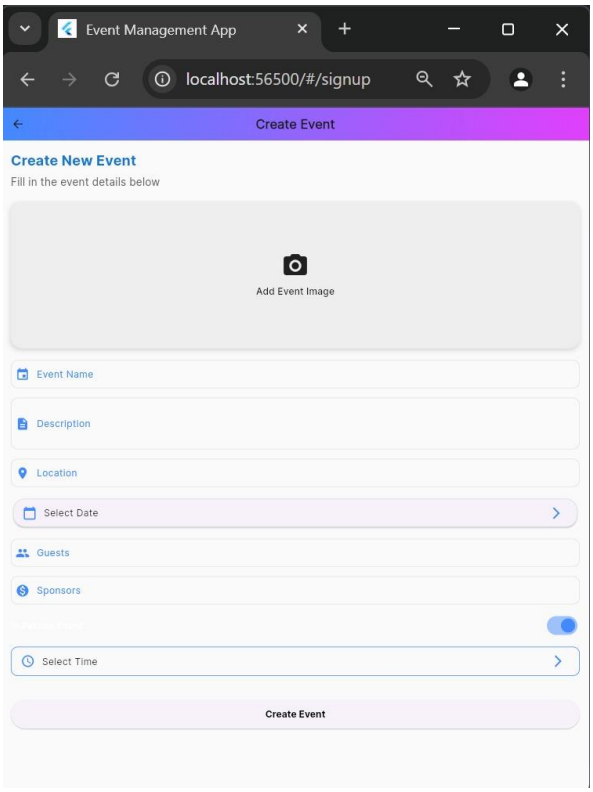
3.2 Using Custom Fonts in the App

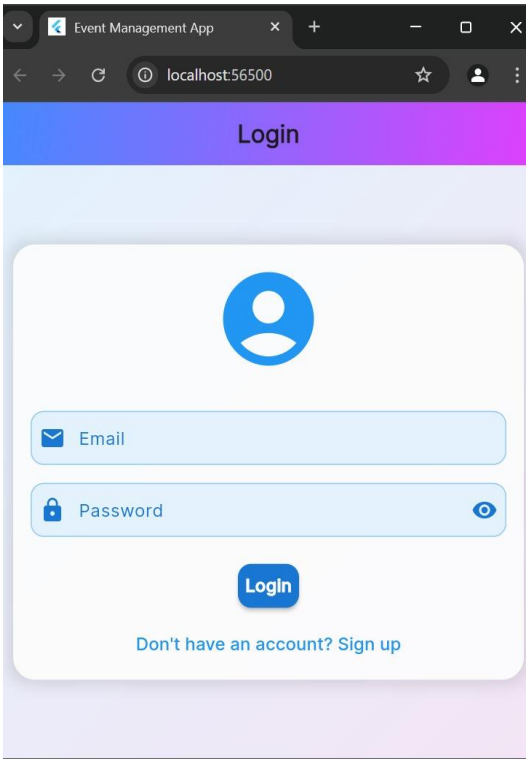
Apply the font to text widgets:

```
Text(
  'Hello, Flutter!',
  style: TextStyle(fontFamily: 'CustomFont', fontSize: 24),
)
```

Output –


Event Management App
localhost:56500/#/signup
Event Track
Hey User
Let's Discover Your Events
The Amazing Race!💥🏃
VESIT Amphitheatre, VESIT, Chembur
2025-03-20 12:30:00
VES-HACK-IT
VESHACKIT
VESIT Library, VESIT, Chembur
2025-03-24 08:30:00
DATE - 26TH SEPT. 2019
Dalal Street
VESIT Amphitheater, VESIT


Event Management App
localhost:56500/#/signup
Create Event
Create New Event
Fill in the event details below
Add Event Image
Event Name
Description
Location
Select Date
Guests
Sponsors
Select Time
Create Event


Event Management App
localhost:56500
Login
Email
Password
Login
Don't have an account? Sign up

Conclusion

This document covered the integration of icons, images, and fonts in a Flutter app. These elements significantly enhance UI design, making the app more engaging and visually appealing.

Project Title: Event Track **Roll No:48**

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	48
Name	Ronit Santwani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Name: Ronit Santwani
Class:D15A
Roll No:48

MPL EXP-4

Aim: To create an interactive Form using form widget

Code :

Form to Create an Event:

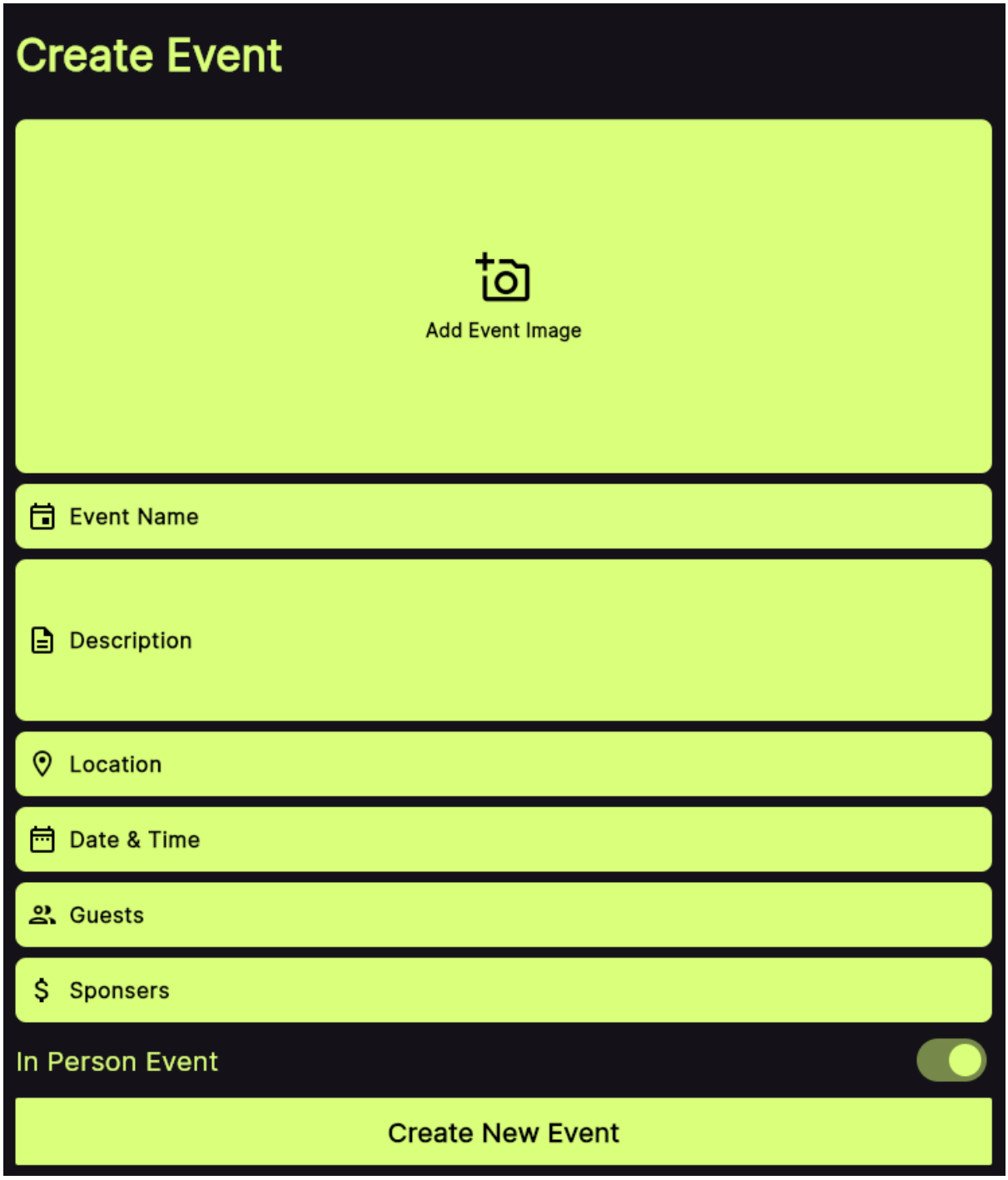
```
import 'package:flutter/material.dart';


class CreateEventScreen extends StatefulWidget {
  @override
  _CreateEventScreenState createState() => _CreateEventScreenState();
}


class _CreateEventScreenState extends State<CreateEventScreen> {
  bool isInPerson = false;


  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      appBar: AppBar(
        title: Text(
          "Create Event",
          style: TextStyle(color: Colors.limeAccent, fontSize: 24),
        ),
        backgroundColor: Colors.black,
        elevation: 0,
      ),
      body: Padding(
        padding: EdgeInsets.all(16.0),
        child: Column(
```

```
children: [
  // Image Upload Section
  Container(
    height: 150,
    width: double.infinity,
    decoration: BoxDecoration(
      color: Colors.limeAccent,
      borderRadius: BorderRadius.circular(10),
    ),
    child: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Icon(Icons.add_a_photo, size: 40, color: Colors.black),
          SizedBox(height: 5),
          Text(
            "Add Event Image",
            style: TextStyle(color: Colors.black),
          ),
        ],
      ),
    ),
  ),
  SizedBox(height: 20),

  // Input Fields
  _buildTextField("Event Name", Icons.event),
  _buildTextField("Description", Icons.description),
  _buildTextField("Location", Icons.location_on),
  _buildTextField("Date & Time", Icons.calendar_today),
  _buildTextField("Guests", Icons.people),
  _buildTextField("Sponsors", Icons.attach_money),
```

```
SizedBox(height: 15),

// In-Person Event Toggle
Container(
  padding: EdgeInsets.symmetric(vertical: 10, horizontal: 10),
  decoration: BoxDecoration(
    color: Colors.black,
    borderRadius: BorderRadius.circular(5),
  ),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [
      Text(
        "In Person Event",
        style: TextStyle(color: Colors.limeAccent, fontSize: 16),
      ),
      Switch(
        value: isInPerson,
        activeColor: Colors.limeAccent,
        onChanged: (value) {
          setState(() {
            isInPerson = value;
          });
        },
      ),
    ],
  ),
),

SizedBox(height: 15),
```

```
          // Create Event Button
          ElevatedButton(
            onPressed: () {},
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.limeAccent,
              padding: EdgeInsets.symmetric(vertical: 14, horizontal: 60),
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(8),
              ),
            ),
            child: Text(
              "Create New Event",
              style: TextStyle(color: Colors.black, fontSize: 16),
            ),
          ),
        ],
      ),
    ),
  );
}

Widget _buildTextField(String label, IconData icon) {
  return Padding(
    padding: EdgeInsets.symmetric(vertical: 5),
    child: TextField(
      decoration: InputDecoration(
        filled: true,
        fillColor: Colors.limeAccent,
        prefixIcon: Icon(icon, color: Colors.black),
        labelText: label,
        labelStyle: TextStyle(color: Colors.black),
        border:
```

Output:


Create Event
Add Event Image
Event Name
Description
Location
Date & Time
Guests
Sponsers
In Person Event
Create New Event

Project Title: Event Track **Roll No:48**

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	48
Name	Ronit Santwani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Name – Ronit Santwani

D15A-48

Experiment 5: Navigation, Routing, and Gestures in Smart Study App

Objective:

To implement **navigation, routing, and gestures** in the Smart Study App using Flutter.

Theory:

Flutter provides a robust navigation system that allows smooth transitions between different screens using **Navigator** and **Routes**. Gesture detection enhances user interaction by recognizing various touch patterns.

1. Navigation & Routing:

- **Navigator.push()**: Moves to a new screen.
- **Navigator.pop()**: Returns to the previous screen.
- **Named Routes**: Defines structured navigation between multiple screens.
- **onGenerateRoute**: Dynamically generates routes.

2. Gesture Detection:

Flutter supports touch gestures like tapping, swiping, and long pressing using **GestureDetector** and **InkWell**.

Implementation:

1. Navigation in Smart Study App

Step 1: Define Routes in main.dart

```
void main() {   runApp(MaterialApp(
   initialRoute: '/',
routes: {
     '/': (context) => HomeScreen(),
     '/subjects': (context) => SubjectsScreen(),
     '/tasks': (context) => TasksScreen(),
   },
  ));
}
```

Step 2: Navigating to a New Screen

```
ElevatedButton(
onPressed: () {
   Navigator.pushNamed(context, '/subjects');
  },
  child: Text("Go to Subjects"),
);
```

Step 3: Returning to Previous Screen

```
ElevatedButton(
onPressed: () {
   Navigator.pop(context);
  },
  child: Text("Back"),
);
```

---

2. Gesture Implementation

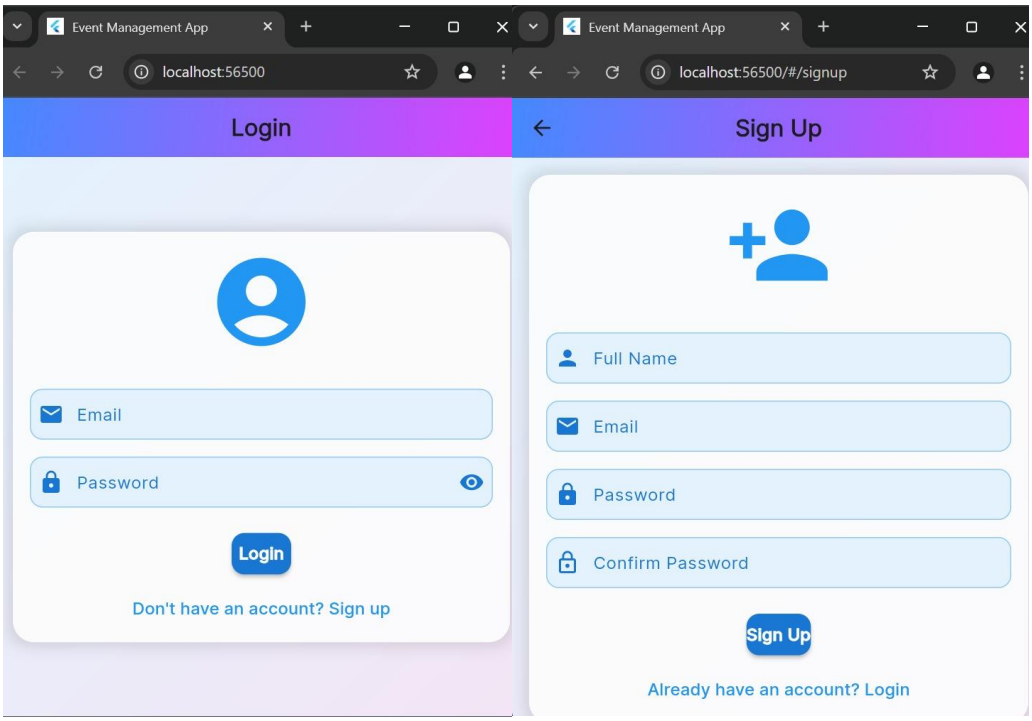
Step 1: Using GestureDetector

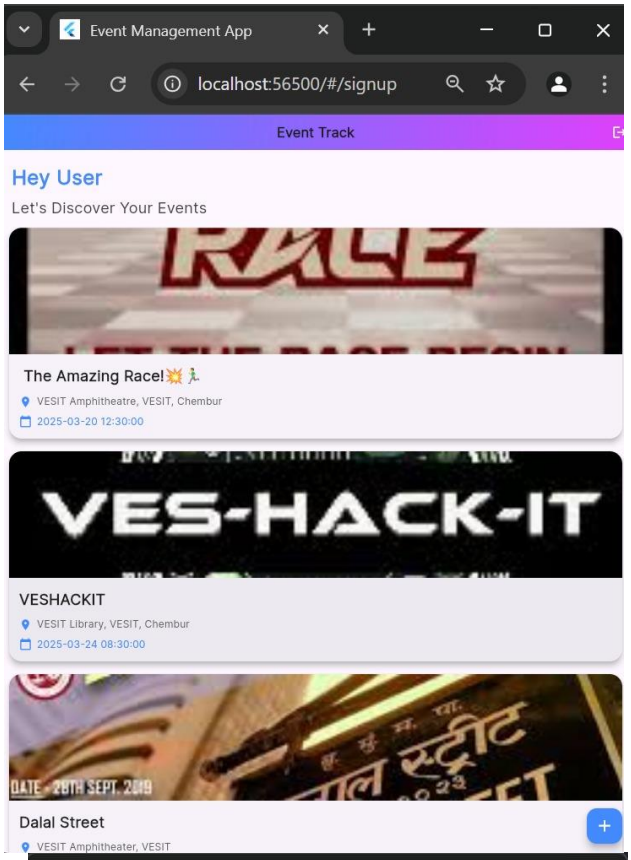
```
GestureDetector(
onTap: () {
   print("Card tapped!");
  },
  child: Card(
child: Padding(
padding:
EdgeInsets.all(16.
```

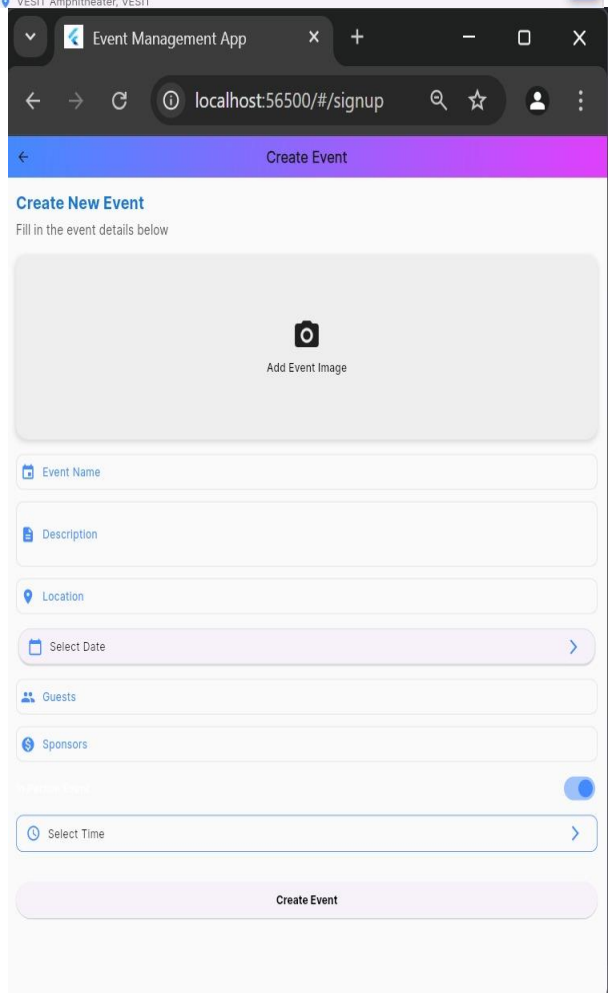
```
0),       child:
Text("Tap Me"),
    ),
  ),
);
```

Step 2: Adding Swipe Gesture

```
GestureDetector(
onHorizontalDragEnd: (details) {
if (details.primaryVelocity! > 0) {
print("Swiped Right");
    } else {
print("Swiped Left");
    }
  },
  child: Container(
color: Colors.blue,
height: 100,     width:
100,
  ),
);
```




Event Management App
localhost:56500/#/signup
Event Track
Hey User
Let's Discover Your Events
The Amazing Race!💥🏃
VESIT Amphitheatre, VESIT, Chembur
2025-03-20 12:30:00
VES-HACK-IT
VESHACKIT
VESIT Library, VESIT, Chembur
2025-03-24 08:30:00
DATE - 28TH SEPT, 2019
Dalal Street
VESIT Amphitheater, VESIT


Event Management App
localhost:56500/#/signup
Create Event
Create New Event
Fill in the event details below
Add Event Image
Event Name
Description
Location
Select Date
Guests
Sponsors
Select Time
Create Event

Conclusion:

In this experiment, we successfully implemented **navigation, routing, and gesture detection** in the **Smart Study App**. Users can navigate between screens and interact with UI elements using tap and swipe gestures.

Key Learnings:

1. Implemented **Navigator** for screen transitions.
2. Used **named routes** for structured navigation.
3. Applied **GestureDetector** for touch-based interactions.

This enhances the **Smart Study App** by improving user experience and accessibility.

Project Title: Event Track **Roll No:48**

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	48
Name	Ronit Santwani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

Name: Ronit Santwani
Class: D15A
Roll No: 48

Setting Up Firebase with Flutter for iOS and Android Apps

This document provides a detailed step-by-step guide on how to integrate Firebase with a Flutter project for both iOS and Android platforms. Firebase offers a suite of tools for app development, including analytics, authentication, cloud storage, and more. By following this guide, you will be able to set up Firebase in your Flutter app and start using its features.

Prerequisites
Before starting, ensure you have the following:

Flutter SDK installed on your machine.

Android Studio or Xcode for Android and iOS development, respectively.

A Firebase account (create one at firebase.google.com).

A Flutter project created (flutter create project_name).

Step 1: Create a Firebase Project Go
to the Firebase Console.

Click Add Project.

Enter a project name and follow the prompts to create the project.

Once the project is created, you will be redirected to the Firebase project dashboard.

Step 2: Add Firebase to Your Flutter Project
For Android
In the Firebase Console, click the Android icon to add an Android app to your Firebase project.

Enter your app's details:

Android package name: Find this in your android/app/build.gradle file under applicationId.

App nickname (optional): Add a nickname for your app.

Debug signing certificate SHA-1 (optional): If you need Firebase Authentication or Dynamic Links, add your SHA-1 key.

Click Register App.

Download the google-services.json file and place it in the android/app directory of your Flutter project.

Add the following dependencies to your android/build.gradle file:

buildscript { dependencies { classpath 'com.google.gms:google-services:4.3.15' // Use the latest version
} }
Add the following to the bottom of your android/app/build.gradle file:

apply plugin: 'com.google.gms.google-services'
For iOS
In the Firebase Console, click the iOS icon to add an iOS app to your Firebase project.

Enter your app's details:

iOS bundle ID: Find this in your Xcode project under Bundle Identifier.

App nickname (optional): Add a nickname for your app.

Click Register App.

Download the GoogleService-Info.plist file.

Open your Flutter project in Xcode.

Drag and drop the GoogleService-Info.plist file into the Runner directory in Xcode.

Ensure the file is added to the Runner target.

Add the following to your ios/Podfile:

platform :ios, '11.0' # or higher
Run pod install in the ios directory to install Firebase dependencies.

Step 3: Add Firebase Dependencies to Flutter
Open your pubspec.yaml file in your Flutter project.

Add the following dependencies under dependencies:

dependencies:
flutter:
sdk: flutter firebase_core: latest_version # Required for Firebase integration firebase_analytics: latest_version # Optional: For analytics firebase_auth: latest_version # Optional: For authentication cloud_firestore: latest_version # Optional: For Firestore database firebase_storage: latest_version # Optional: For cloud storage Run flutter pub get to install the dependencies.

Step 4: Initialize Firebase in Your Flutter App Open your lib/main.dart file.

Import the Firebase Core package:

import 'package:firebase_core/firebase_core.dart'; Initialize Firebase in the main function:

dart Copy
void main() async {
WidgetsFlutterBinding.ensureInitialized();
await Firebase.initializeApp();
runApp(MyApp());
}

Step 5: Test Firebase Integration
Run your app on an Android or iOS emulator/device.

Check the Firebase Console to ensure your app is connected and sending data (e.g., analytics events).

Step 6: Use Firebase Services
Now that Firebase is set up, you can start using its services in your Flutter app. For example:

Firebase Authentication: Add user authentication using email/password, Google Sign-In, etc.

Firestore: Store and retrieve data from a NoSQL database.

Firebase Storage: Upload and download files.

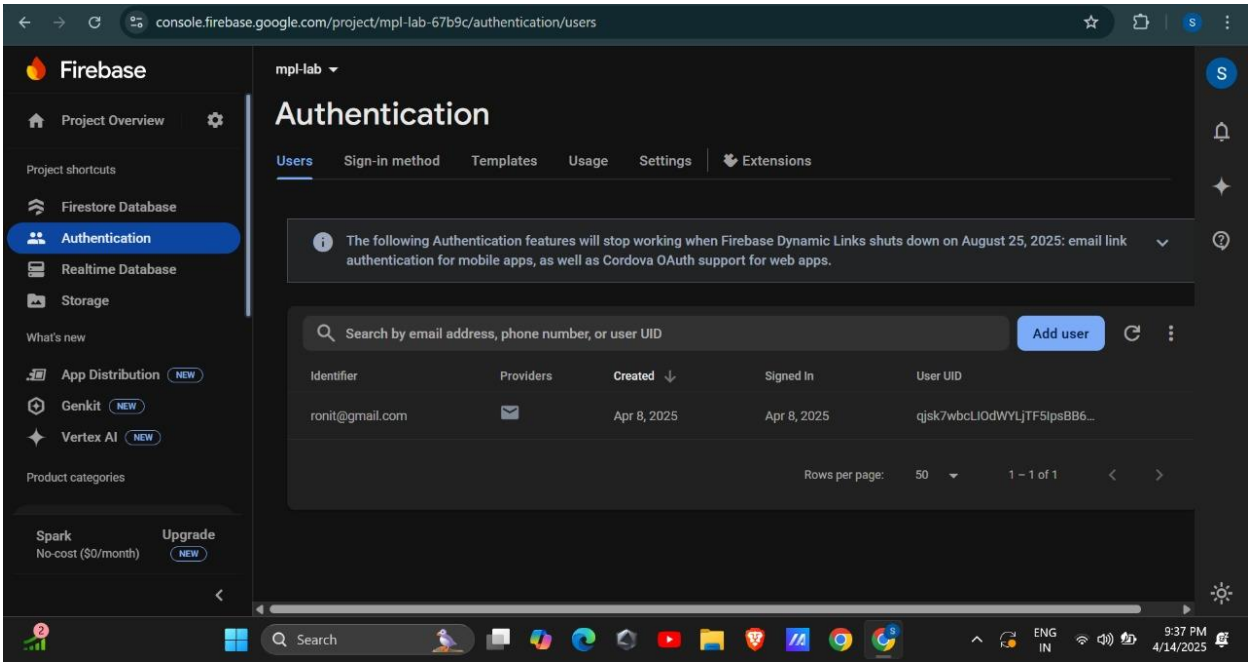
Firebase Analytics: Track user behavior and app usage.

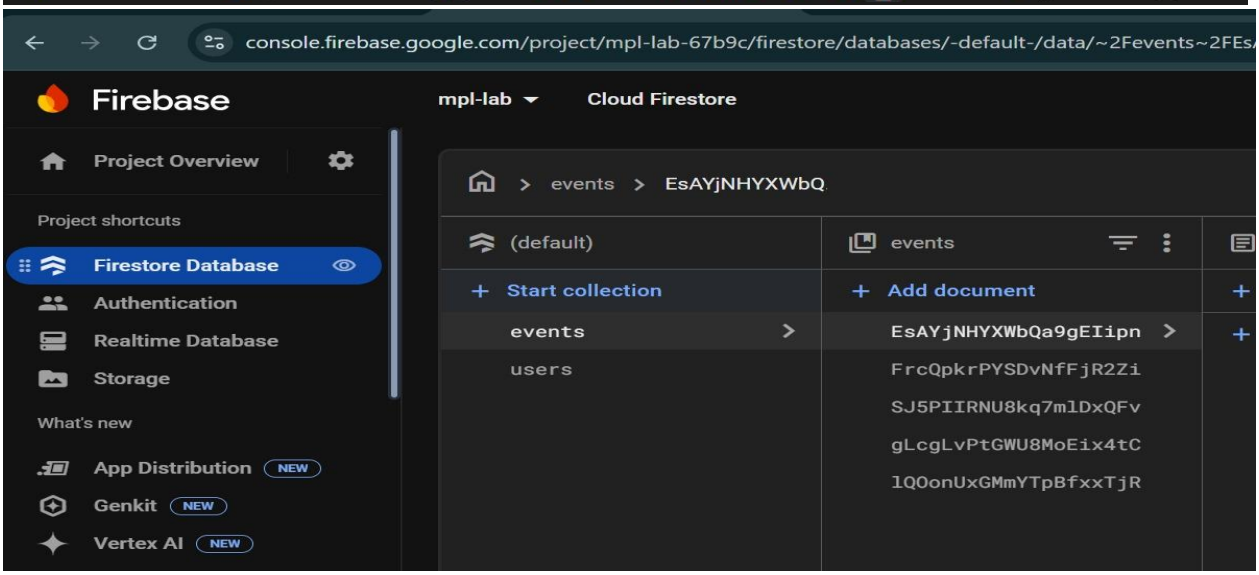
Troubleshooting

Android: If you encounter issues with the google-services.json file, ensure it is placed in the correct directory (android/app).

iOS: If the app crashes on launch, ensure the GoogleService-Info.plist file is added to the Xcode project and the Runner target.

Dependencies: Always use the latest versions of Firebase plugins and ensure there are no version conflicts.





Conclusion

You have successfully set up Firebase in your Flutter app for both iOS and Android platforms. You can now leverage Firebase's powerful features to enhance your app's functionality. For more details, refer to the official Firebase Flutter documentation.Replace latest_version with the actual version numbers of the Firebase plugins you are using. You can find the latest versions on pub.dev.

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.	48
Name	Ronit Santwani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

Experiment No. 7

Title: 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 nativedevice 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 endusers 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:

- 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

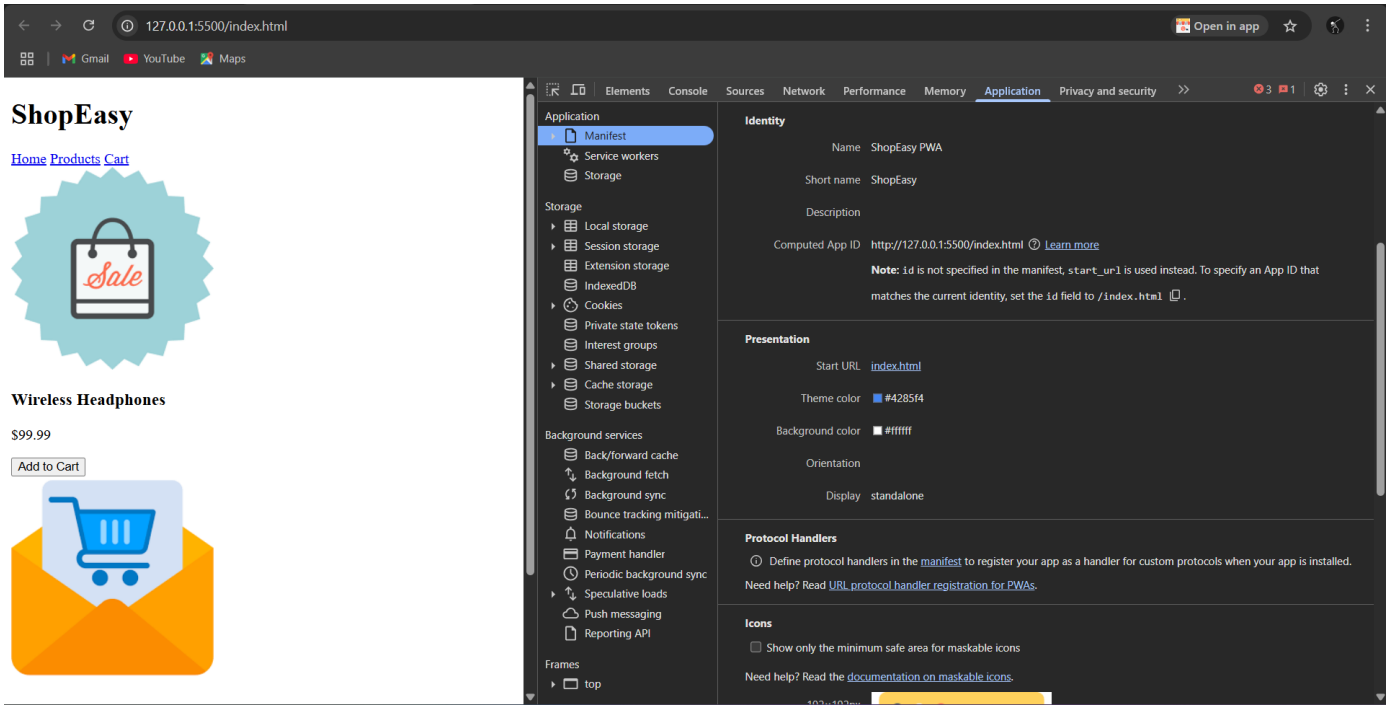
- Lack of presence on the stores (there is no possibility to acquire traffic from that channel)
- There is no “body” of control (like the stores) and an approval process
- Limited access to some hardware components of the devices
- Little flexibility regarding “special” content for users (eg loyalty programs, loyalty, etc.)
  Code:

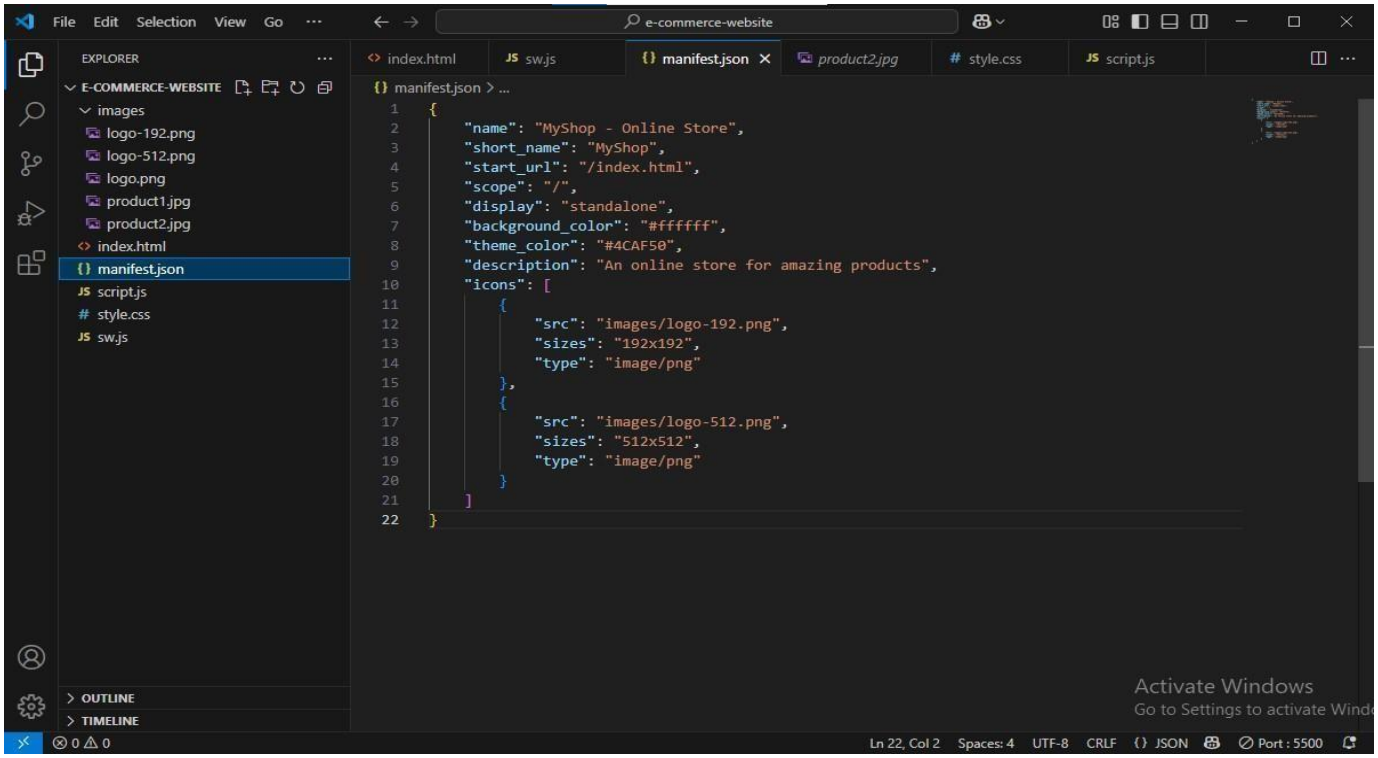
manifest.json:

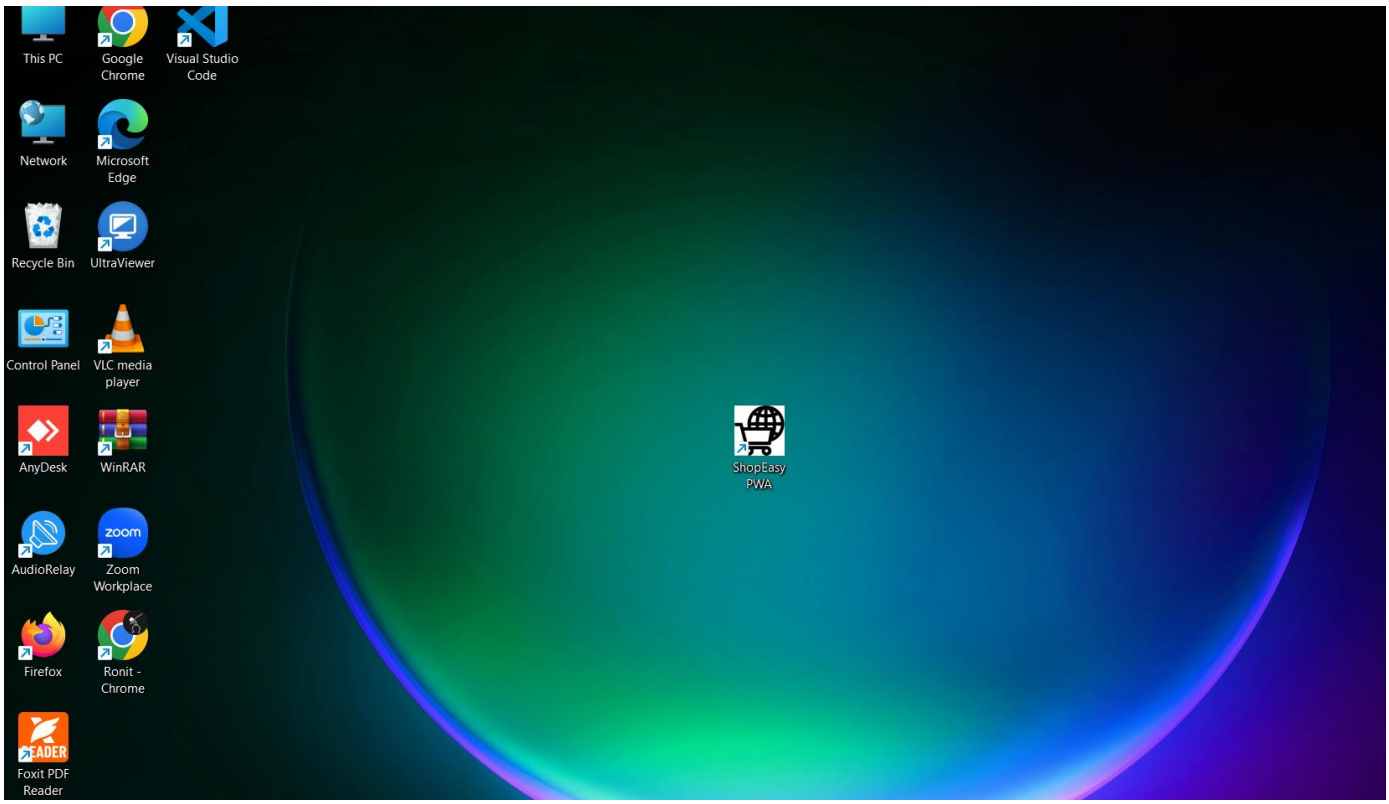
```
{
    "id": "/",
    "short_name": "My Bakery",
    "name": "My Bakery",
    "description": "My Bakery Website",
    "start_url": "/index.html",
    "display": "standalone",
    "background_color": "#ffffff",
    "theme_color": "#000000",
    "orientation": "portrait",
    "icons": [
     {
       "src": "/icons/icon-192-1.png",
       "sizes": "192x192",
       "type": "image/png",
       "purpose": "any"
     },
     {
       "src": "/icons/icon-512.png",
       "sizes": "512x512",
       "type": "image/png",
       "purpose": "maskable"
     }
```

```
  ]
}
```

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







Conclusion:

Hence, we learnt how to write a metadata of our 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 Ecommerce PWA
Roll No.	48
Name	Ronit Santwani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Experiment No. 8

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

Theory:

Service Worker

Service Worker is a script that works on browser background without user interaction independently. Also, It resembles a proxy that works on the user side. With this script, you can

track network traffic of the page, manage push notifications and develop “offline first” web applications with Cache API.

Things to note about Service Worker:

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

What can we do with Service Workers?

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

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

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

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

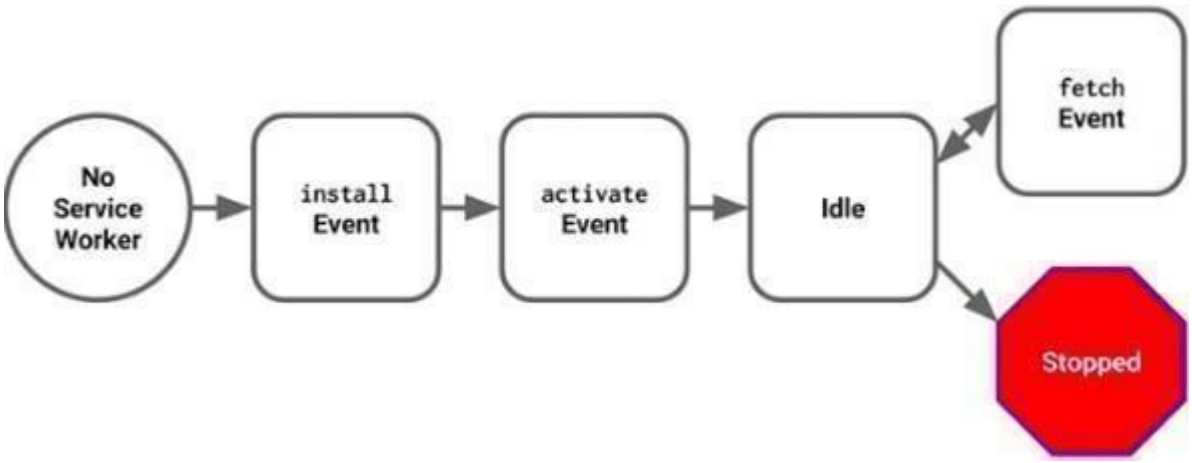
What can’t we do with Service Workers?

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

- You can’t work it on **80 Port**

Service Worker just can work on HTTPS protocol. But you can work on localhost during development.

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:

```
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. 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

```
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.

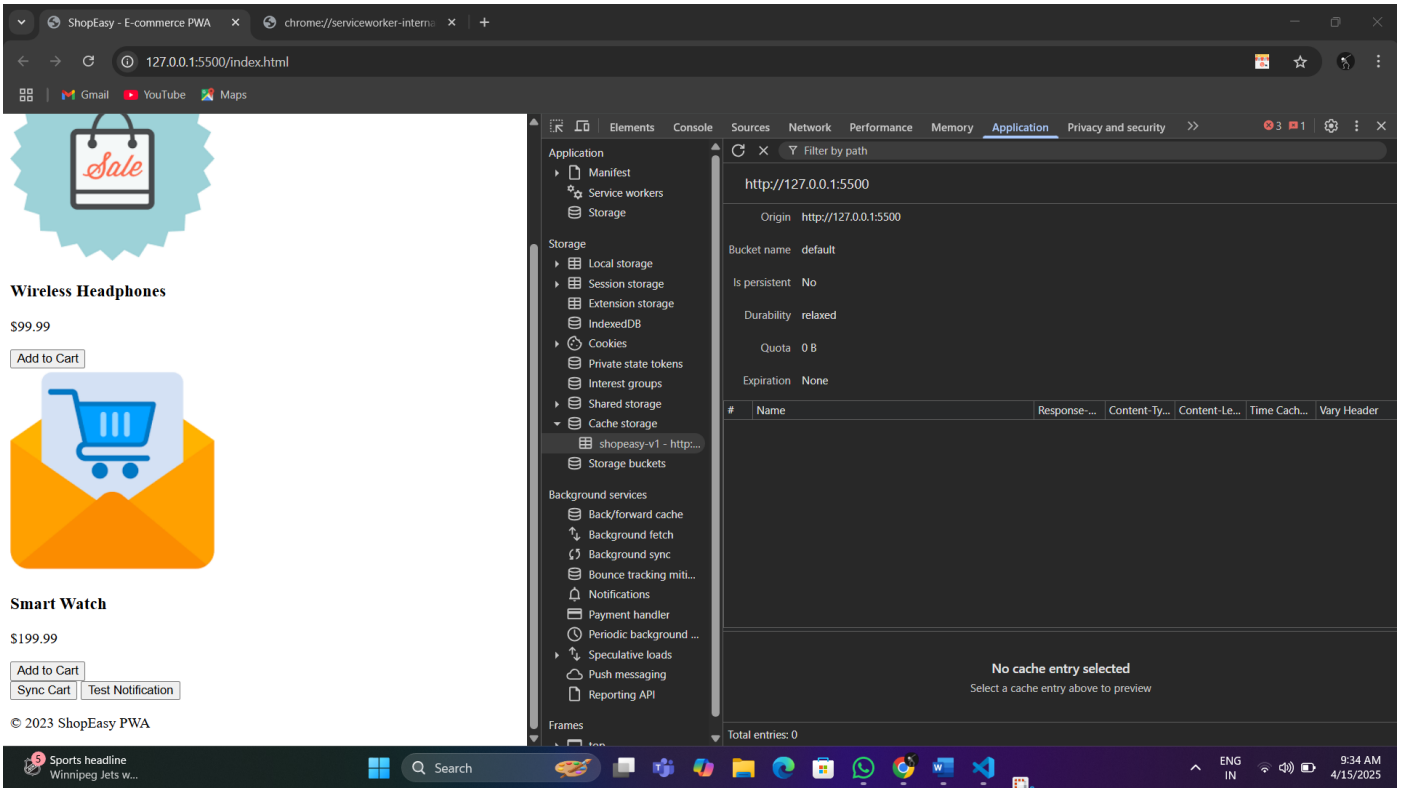
Code:

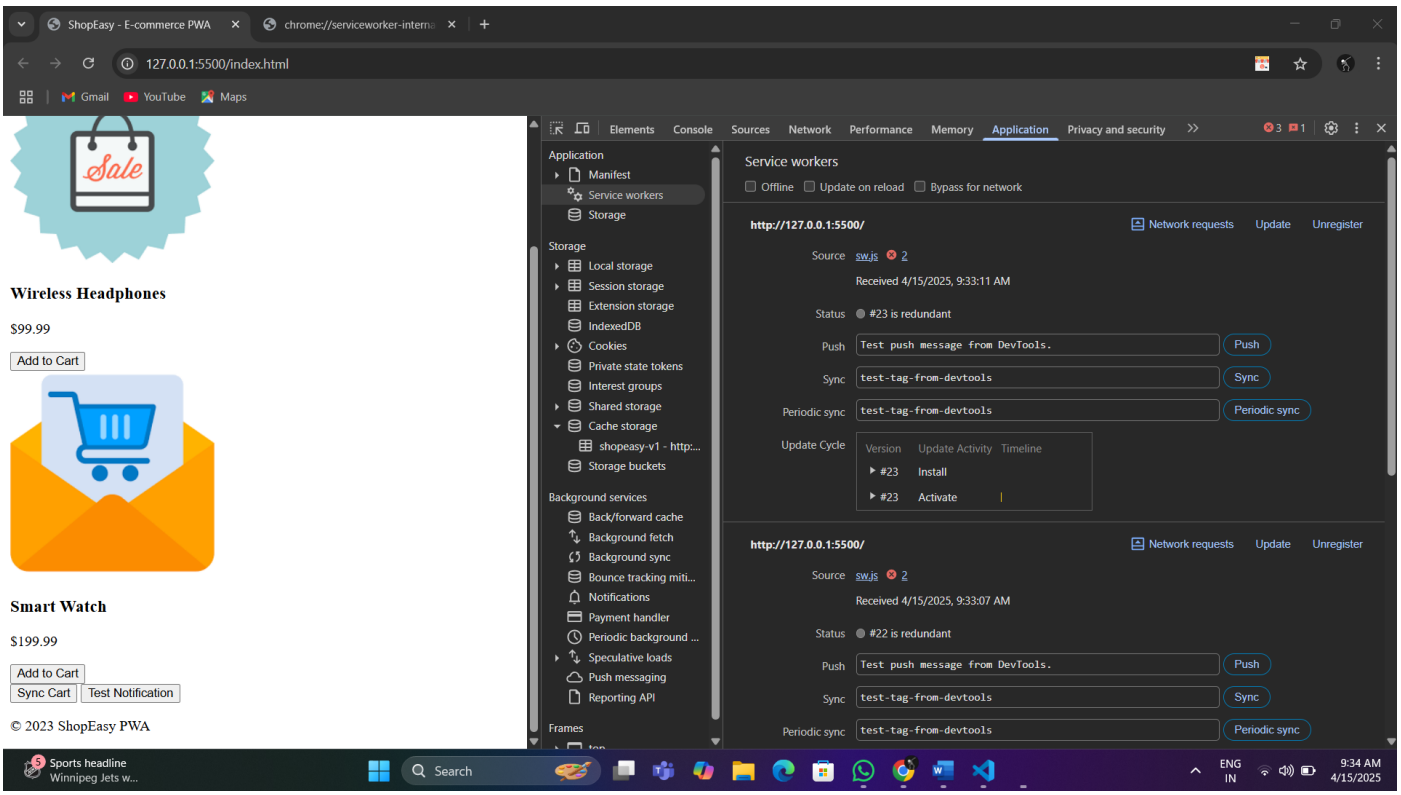
```
<script> if ('serviceWorker' in navigator) { window.addEventListener('load', () => {
    navigator.serviceWorker.register('/service-worker.js')
      .then((registration) => { console.log('Service Worker registered with scope: ', registration.scope);
      })
      .catch((error) => { console.log('Service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

service-worker.js

```
const CACHE_NAME = 'my-pwa-cache-v1';
const urlsToCache = [
 '/',
 '/index.html',
 '/styles.css',
 '/icons/icon-192-1.png',
 '/icons/icon-512.png',
 '/assets/brownie.png',
 '/assets/cakes.png',
 '/assets/cookies.png',
 '/assets/donuts.png',
 '/assets/logo.png',
 '/assets/muffins.png',
 '/assets/pastry.png',
];

// Install service worker
```

```
self.addEventListener('install', (event) => { event.waitUntil(
   caches.open(CACHE_NAME).then((cache) => {
     console.log('Opened cache'); return
     cache.addAll(urlsToCache);
     })
 );  });

// Activate service worker self.addEventListener('activate',
(event) => {  const cacheWhitelist = [CACHE_NAME];
event.waitUntil(  caches.keys().then((cacheNames) => {
return Promise.all( cacheNames.map((cacheName)
     => { if
     (!cacheWhitelist.includes(cacheName)) {  return caches.delete(cacheName);
        }
      })
     );
   })
 );  });

// Fetch content from the cache or network self.addEventListener('fetch',
(event) => {  event.respondWith(
caches.match(event.request).then((cachedResponse) => {   return
     cachedResponse || fetch(event.request);  })
 );
});
```


ShopEasy - E-commerce PWA
chrome://serviceworker-intern
127.0.0.1:5500/index.html
Gmail
YouTube
Maps
Sale
Wireless Headphones
$99.99
Add to Cart
Smart Watch
$199.99
Add to Cart
Sync Cart
Test Notification
© 2023 ShopEasy PWA
Elements
Console
Sources
Network
Performance
Memory
Application
Privacy and security
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
shopeasy-v1 - http:...
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking miti...
Notifications
Payment handler
Periodic background ...
Speculative loads
Push messaging
Reporting API
Frames
Filter by path
http://127.0.0.1:5500
Origin http://127.0.0.1:5500
Bucket name default
Is persistent No
Durability relaxed
Quota 0 B
Expiration None
Name Response-... Content-Ty... Content-Le... Time Cach... Vary Header
No cache entry selected
Select a cache entry above to preview
Total entries: 0
Sports headline
Winnipeg Jets w...
Search
ENG
IN
9:34 AM
4/15/2025


ShopEasy - E-commerce PWA
chrome://serviceworker-intern
127.0.0.1:5500/index.html
Gmail
YouTube
Maps
Sale
Wireless Headphones
$99.99
Add to Cart
Smart Watch
$199.99
Add to Cart
Sync Cart
Test Notification
© 2023 ShopEasy PWA
Elements
Console
Sources
Network
Performance
Memory
Application
Privacy and security
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
shopeasy-v1 - http:...
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking miti...
Notifications
Payment handler
Periodic background ...
Speculative loads
Push messaging
Reporting API
Frames
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Network requests
Update
Unregister
Source sw.js 2
Received 4/15/2025, 9:33:11 AM
Status #23 is redundant
Push Test push message from DevTools. Push
Sync test-tag-from-devtools Sync
Periodic sync test-tag-from-devtools Periodic sync
Update Cycle
Version Update Activity Timeline
#23 Install
#23 Activate
http://127.0.0.1:5500/
Network requests
Update
Unregister
Source sw.js 2
Received 4/15/2025, 9:33:07 AM
Status #22 is redundant
Push Test push message from DevTools. Push
Sync test-tag-from-devtools Sync
Periodic sync test-tag-from-devtools Periodic sync
Sports headline
Winnipeg Jets w...
Search
ENG
IN
9:34 AM
4/15/2025

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.	48
Name	Ronit Santwani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

EXPERIMENT NO. 9

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

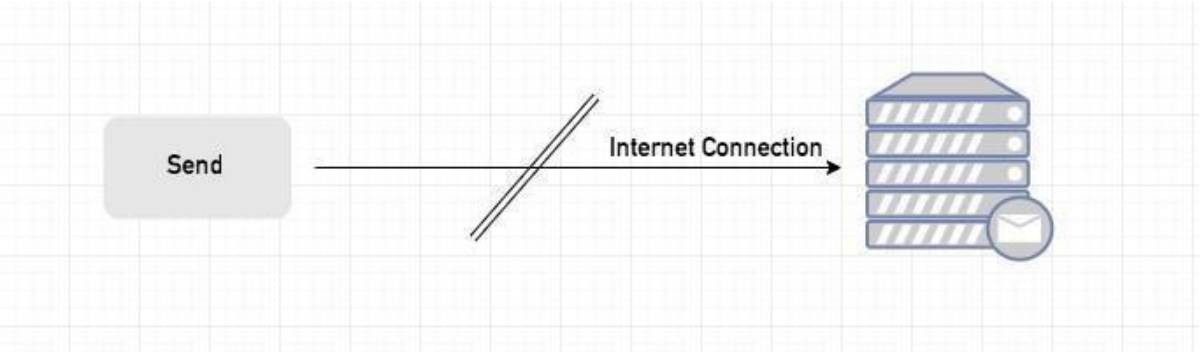
Of course, you can use many different methods but you can find in the following example a “cache first” and “network first” approach. In this example, if the request’s and current location’s origin are the same (Static content is requested.), this is called “cacheFirst” but if you request a targeted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

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

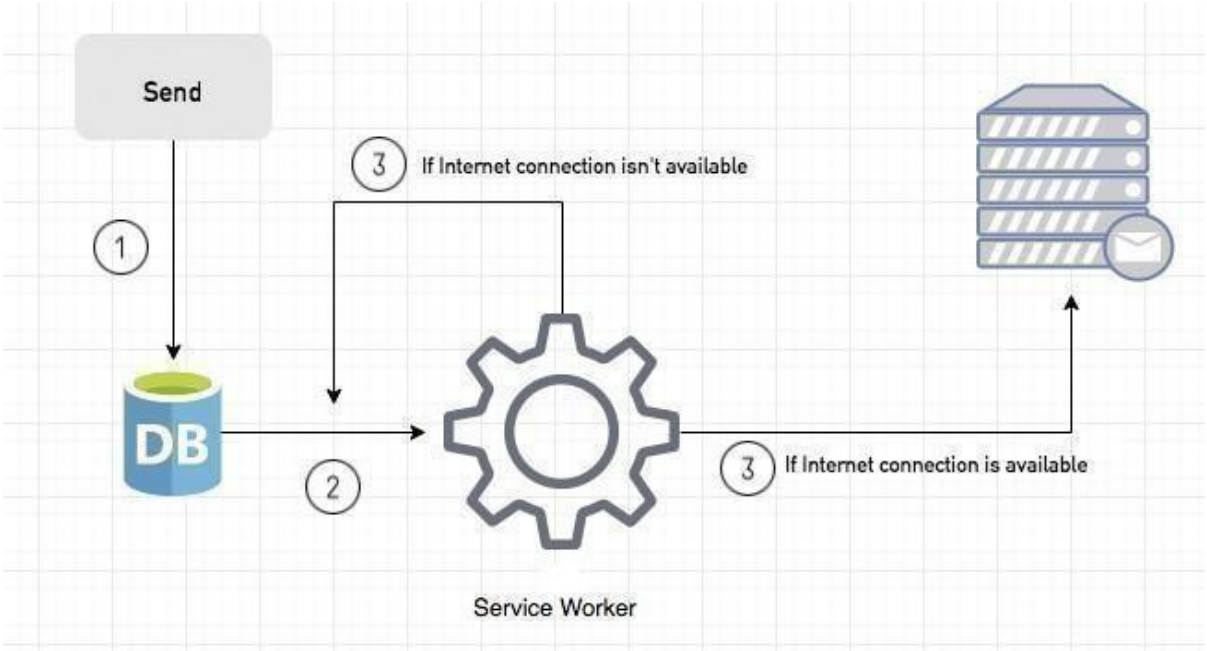
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

Here is a job for the Background Sync.

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



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



1. When we click the "send" button, email content will be saved to IndexedDB.
2. Background Sync registration.
3. **If the Internet connection is available**, all email content will be read and sent to Mail Server.
   If the Internet connection is unavailable, the service worker waits until the connection is available even though the window is closed. When it is available, email content will be sent to Mail Server. You can see the working process within the following code block.

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]");
    }
});
```

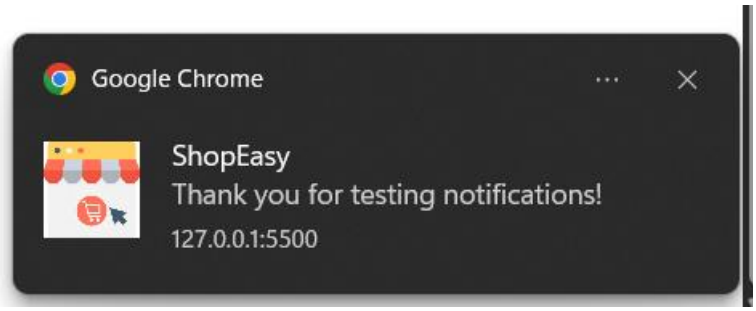
Push Event

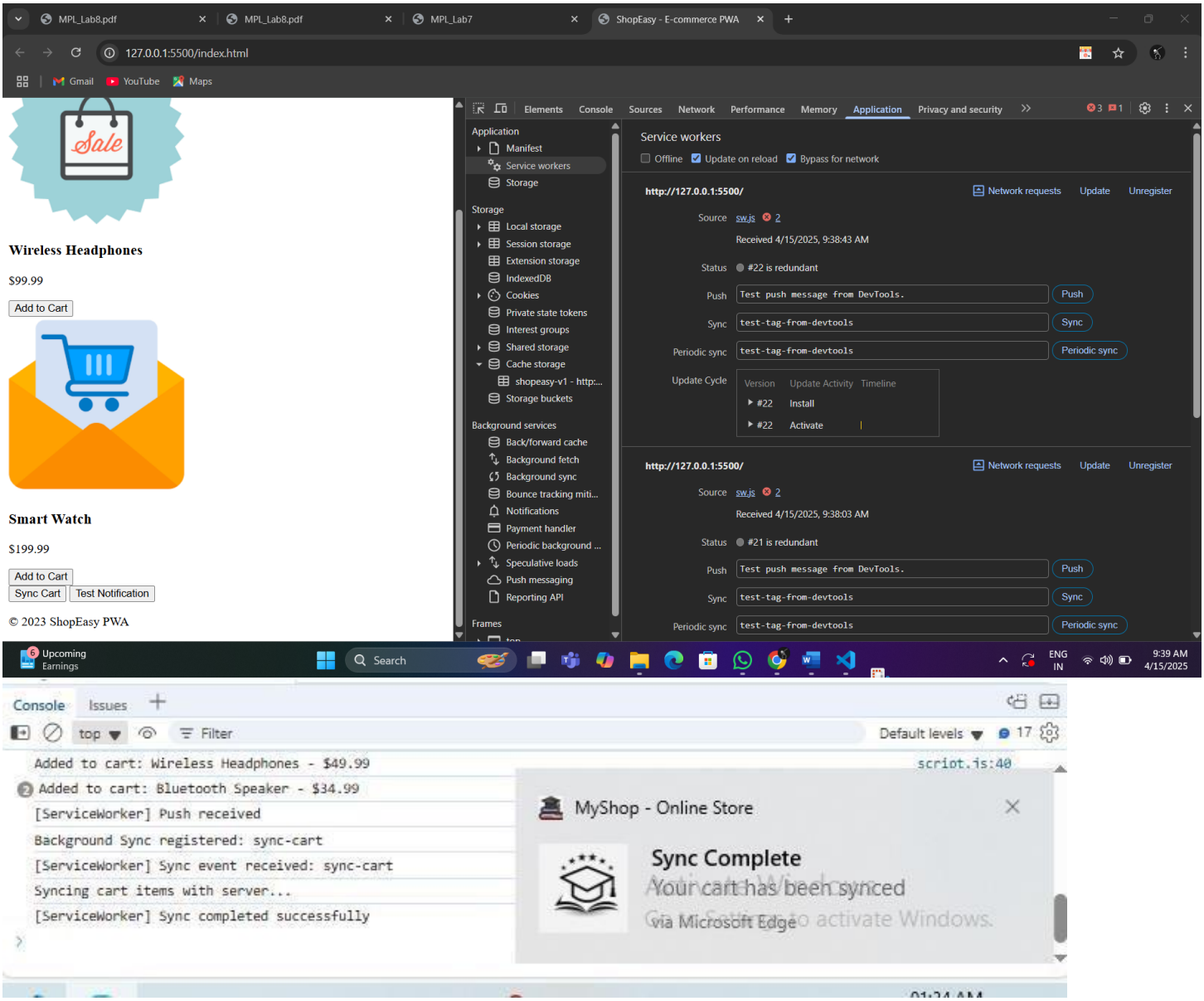
This is the event that handles push notifications that are received from the server. You can apply any method with received data.

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” proper




MPL_Lab8.pdf
MPL_Lab8.pdf
MPL_Lab7
ShopEasy - E-commerce PWA
127.0.0.1:5500/index.html
Gmail
YouTube
Maps
Sale
Wireless Headphones
$99.99
Add to Cart
Smart Watch
$199.99
Add to Cart
Sync Cart
Test Notification
© 2023 ShopEasy PWA
Elements
Console
Sources
Network
Performance
Memory
Application
Privacy and security
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
shopeasy-v1 - http:...
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking miti...
Notifications
Payment handler
Periodic background ...
Speculative loads
Push messaging
Reporting API
Frames
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Network requests
Update
Unregister
Source sw.js 2
Received 4/15/2025, 9:38:43 AM
Status #22 is redundant
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#22 Install
#22 Activate
http://127.0.0.1:5500/
Network requests
Update
Unregister
Source sw.js 2
Received 4/15/2025, 9:38:03 AM
Status #21 is redundant
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Upcoming
Earnings
Search
ENG
IN
9:39 AM
4/15/2025

Console
Issues
top
Filter
Default levels
17
Added to cart: Wireless Headphones - $49.99
script.js:40
Added to cart: Bluetooth Speaker - $34.99
[ServiceWorker] Push received
Background Sync registered: sync-cart
[ServiceWorker] Sync event received: sync-cart
Syncing cart items with server...
[ServiceWorker] Sync completed successfully
MyShop - Online Store
Sync Complete
Your cart has been synced
via Microsoft Edge

MAD & PWA Lab Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	48
Name	Ronit Santwani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Experiment No. 10

Aim:

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

Theory:

GitHub Pages

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

GitHub Pages provides the following key features:

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

Reasons for favoring this over Firebase:

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

GitHub Pages is used by Lyft, CircleCI, and HubSpot.

GitHub Pages is listed in 775 company stacks and 4401 developer stacks.

Pros

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

Cons

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

Firebase

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

Some of the features offered by Firebase are:

1. Add the Firebase library to your app and get access to a shared data structure. Any changes made to that data are automatically synchronized with the Firebase cloud and with other clients within milliseconds.
2. Firebase apps can be written entirely with client-side code, update in real-time out-ofthebox, 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.

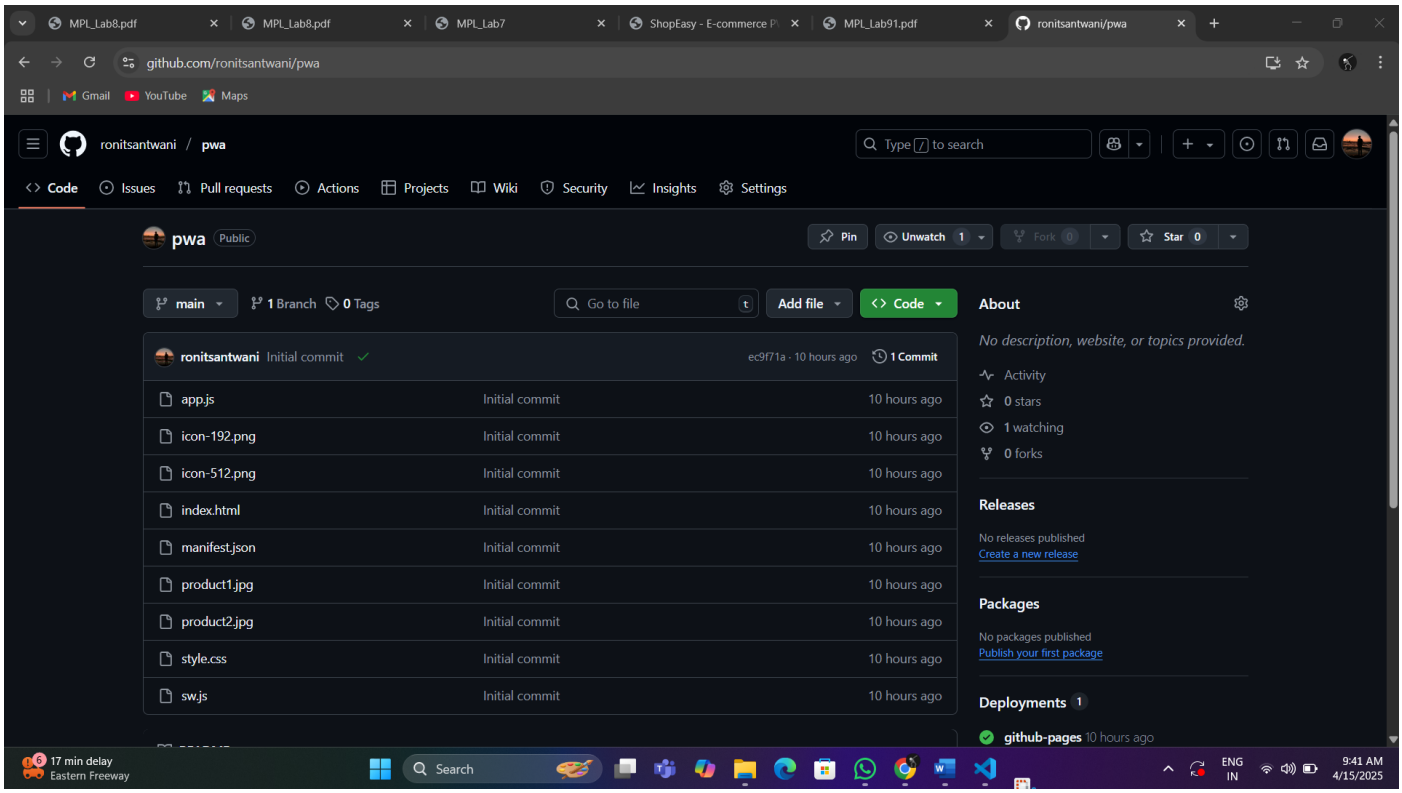
5. Support for HTTPS. A free certificate will be provisioned for your custom domain within 24 hours.

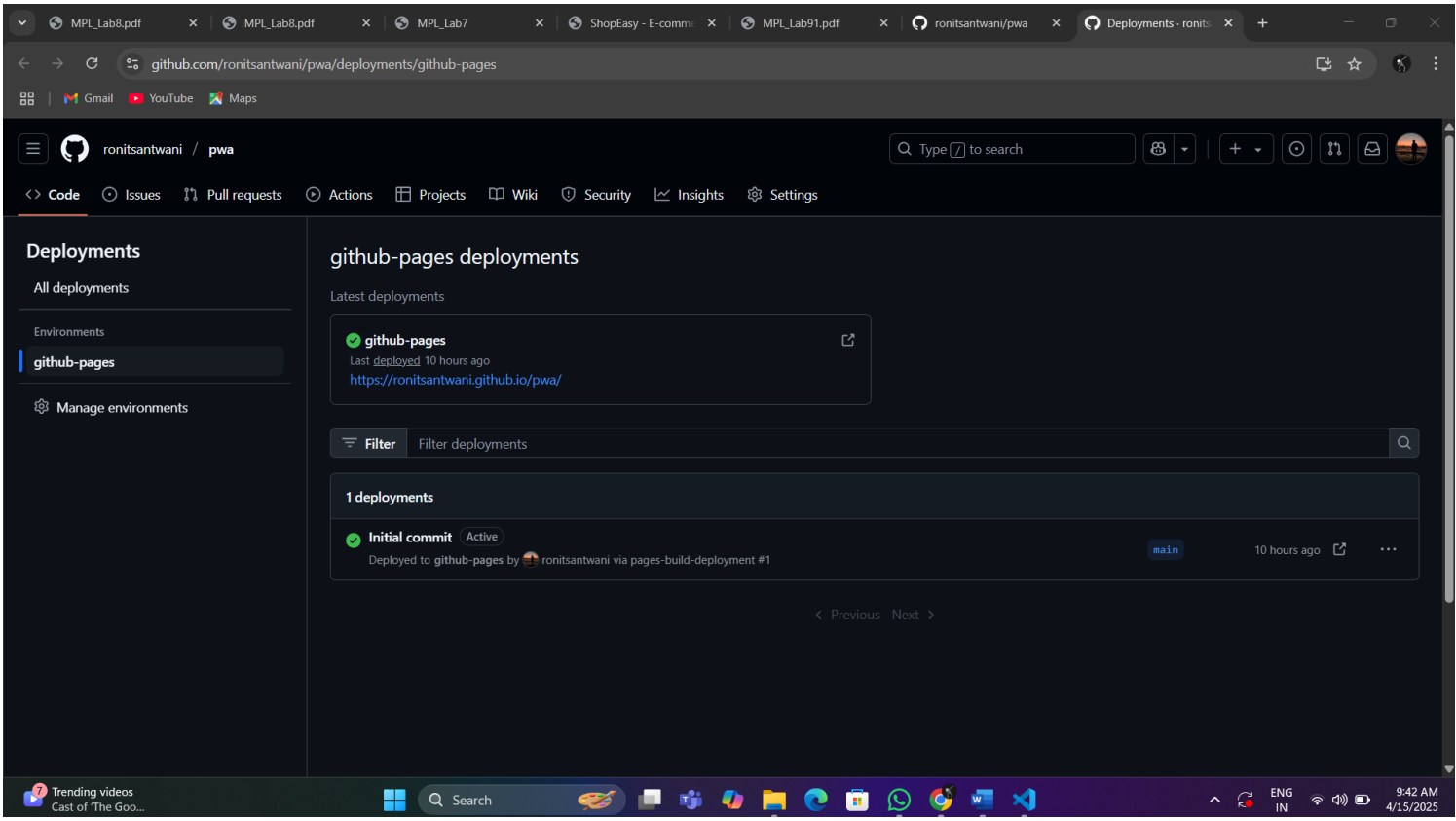
Cons

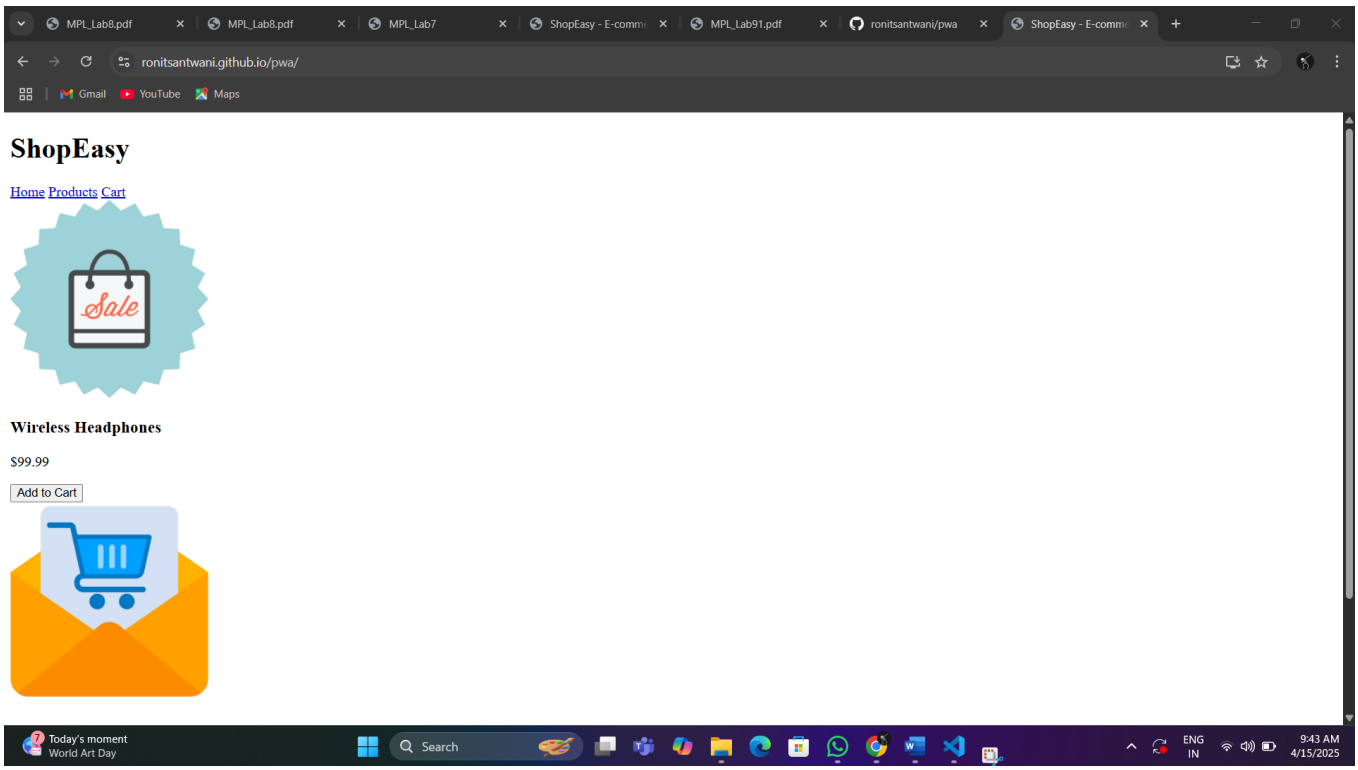
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator.

Link to our GitHub repository: https://github.com/ronitsantwani/pwa.git

Github Screenshot:







Deployed Link: https://ronitsantwani.github.io/pwa/

MAD & PWA Lab Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	48
Name	Ronit Santwani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

Experiment No. 11

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

Theory :

Google Lighthouse :

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

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

Key Features and Audit Metrics

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

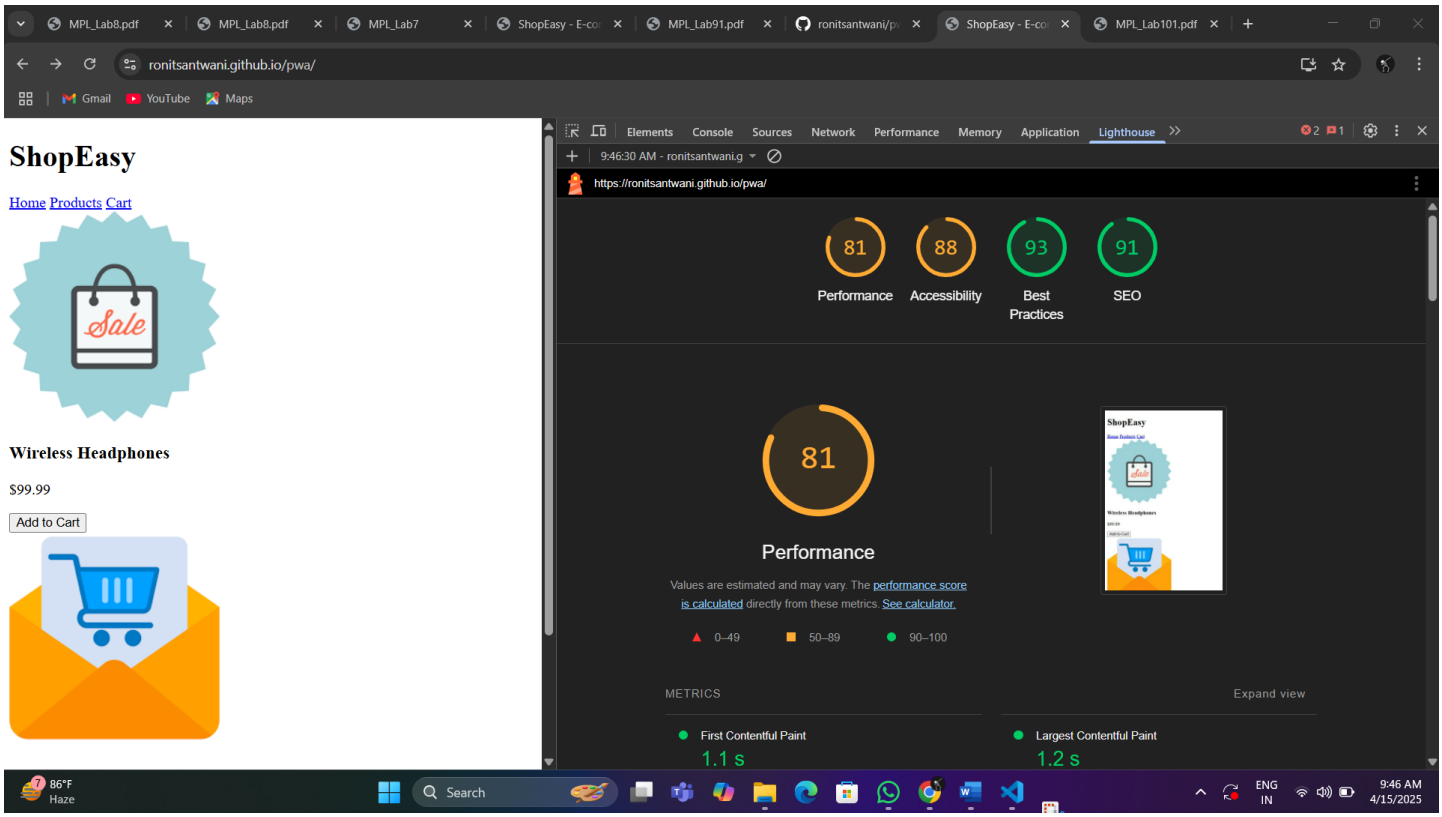
1. **Performance:** This score is an aggregation of how the page fared in aspects such as (but not limited to) loading speed, time taken for loading for basic frame(s), displaying meaningful content to the user, etc. To a layman, this score is indicative of how decently the site performs, with a score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

in script-disabled environments, etc.

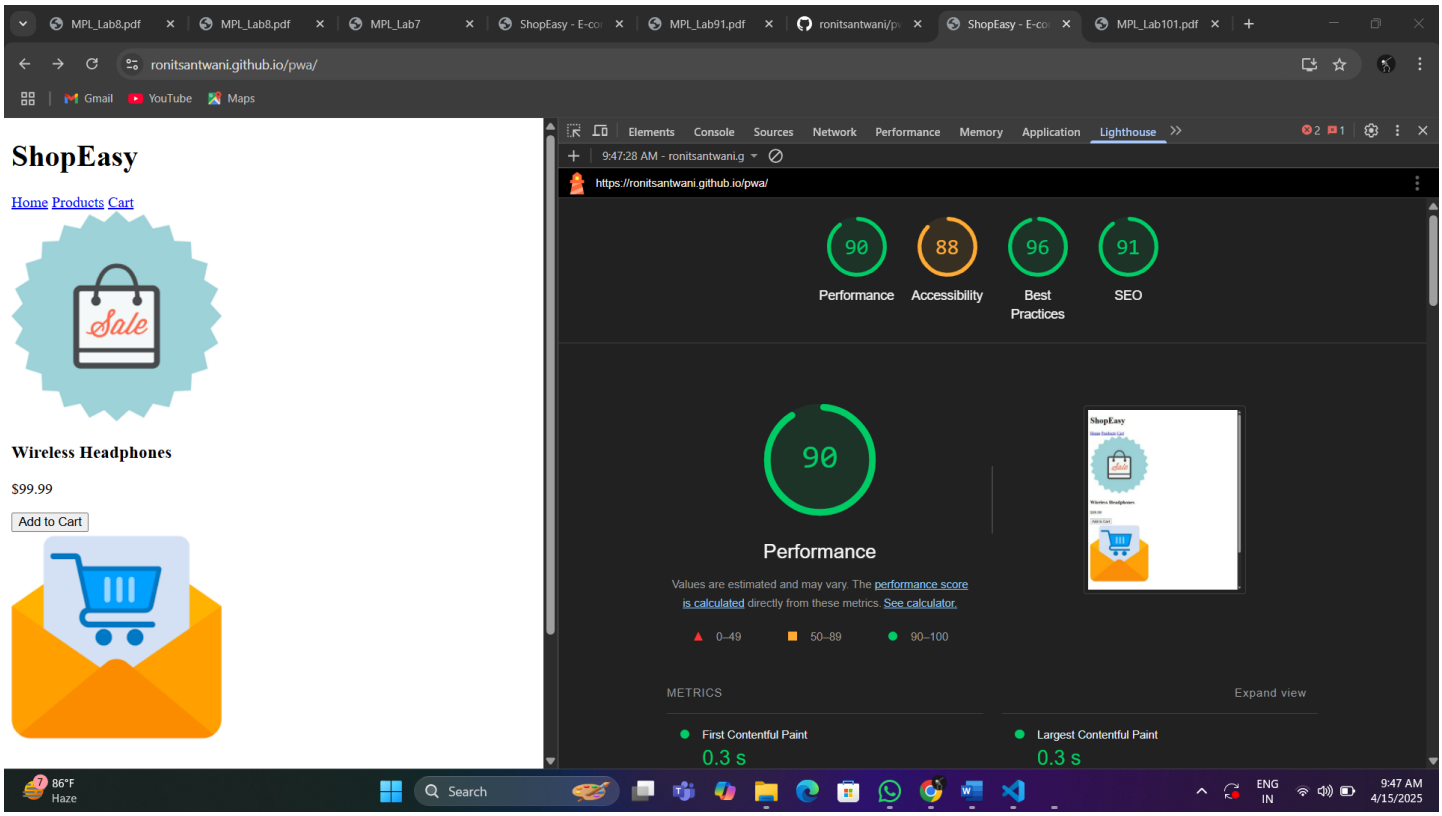
3. **Accessibility:** As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics though, Accessibility metrics score on a pass/fail basis i.e. if all possible elements of the page are not screen-reader friendly (HTML5 introduced features that would make pages easy to interpret for screen readers used by visually challenged people like tag names, tags such as <section>, <article>, etc.), you get a 0 on that score.
   The aggregate of these scores is your Accessibility metric score.
4. **Best Practices:** As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS
   Avoiding the use of deprecated code elements like tags, directives, libraries, etc.
   Password input with paste-into disabled
   Geo-Location and cookie usage alerts on load, etc.

Screenshots: Mobile:

initial


MPL_Lab8.pdf
MPL_Lab8.pdf
MPL_Lab7
ShopEasy - E-co
MPL_Lab91.pdf
ronitsantwani/p
ShopEasy - E-co
MPL_Lab101.pdf
ronitsantwani.github.io/pwa/
Gmail
YouTube
Maps
ShopEasy
Home Products Cart
Sale
Wireless Headphones
$99.99
Add to Cart
Elements
Console
Sources
Network
Performance
Memory
Application
Lighthouse
9:46:30 AM - ronitsantwani.g
https://ronitsantwani.github.io/pwa/
81
Performance
88
Accessibility
93
Best Practices
91
SEO
81
Performance
Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.
0–49
50–89
90–100
METRICS
Expand view
First Contentful Paint
1.1 s
Largest Contentful Paint
1.2 s
86°F
Haze
Search
ENG
IN
9:46 AM
4/15/2025

Desktop


MPL_Lab8.pdf
MPL_Lab8.pdf
MPL_Lab7
ShopEasy - E-co
MPL_Lab91.pdf
ronitsantwani/p
ShopEasy - E-co
MPL_Lab101.pdf
ronitsantwani.github.io/pwa/
Gmail
YouTube
Maps
ShopEasy
Home Products Cart
Sale
Wireless Headphones
$99.99
Add to Cart
Elements
Console
Sources
Network
Performance
Memory
Application
Lighthouse
9:47:28 AM - ronitsantwani.g
https://ronitsantwani.github.io/pwa/
90
Performance
88
Accessibility
96
Best Practices
91
SEO
90
Performance
Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.
0–49
50–89
90–100
METRICS
Expand view
First Contentful Paint
0.3 s
Largest Contentful Paint
0.3 s
86°F
Haze
Search
ENG
IN
9:47 AM
4/15/2025

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