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

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

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

Project Title: **Roll No.**

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

Faculty Incharge : Mrs. Kajal Joseph.

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

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

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: **Roll No.**

Lab Objectives:

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

Lab Outcomes:

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

Project Title: **Roll No.**

Index

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

Project Title: **Roll No.**

MAD & PWA Lab

Journal

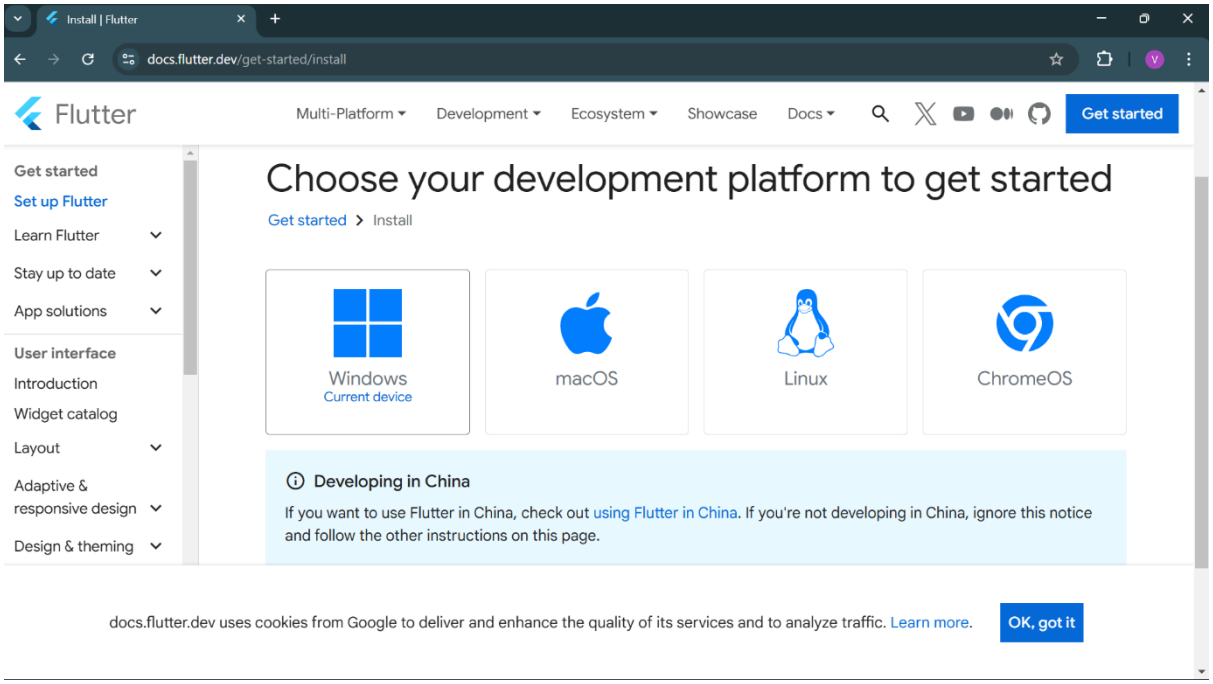
Experiment No.	Assignment-1
Assignment 1 Questions	1. Flutter Overview:Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community. 2. Widget Tree and Composition: Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree. 3. State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable. 4. Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.
Roll No.	17
Name	Omkar Gholap
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

EXPERIMENT NO: - 01

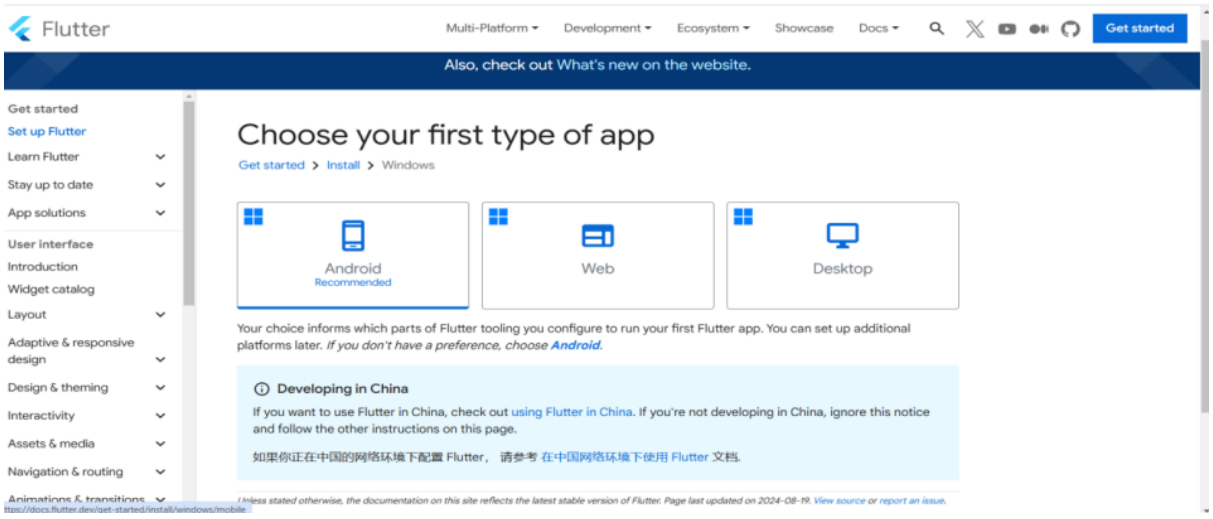
Name:- Omkar Gholap Class:- D15A Roll:No: - 17

AIM: - Installation and Configuration of Flutter Environment.

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

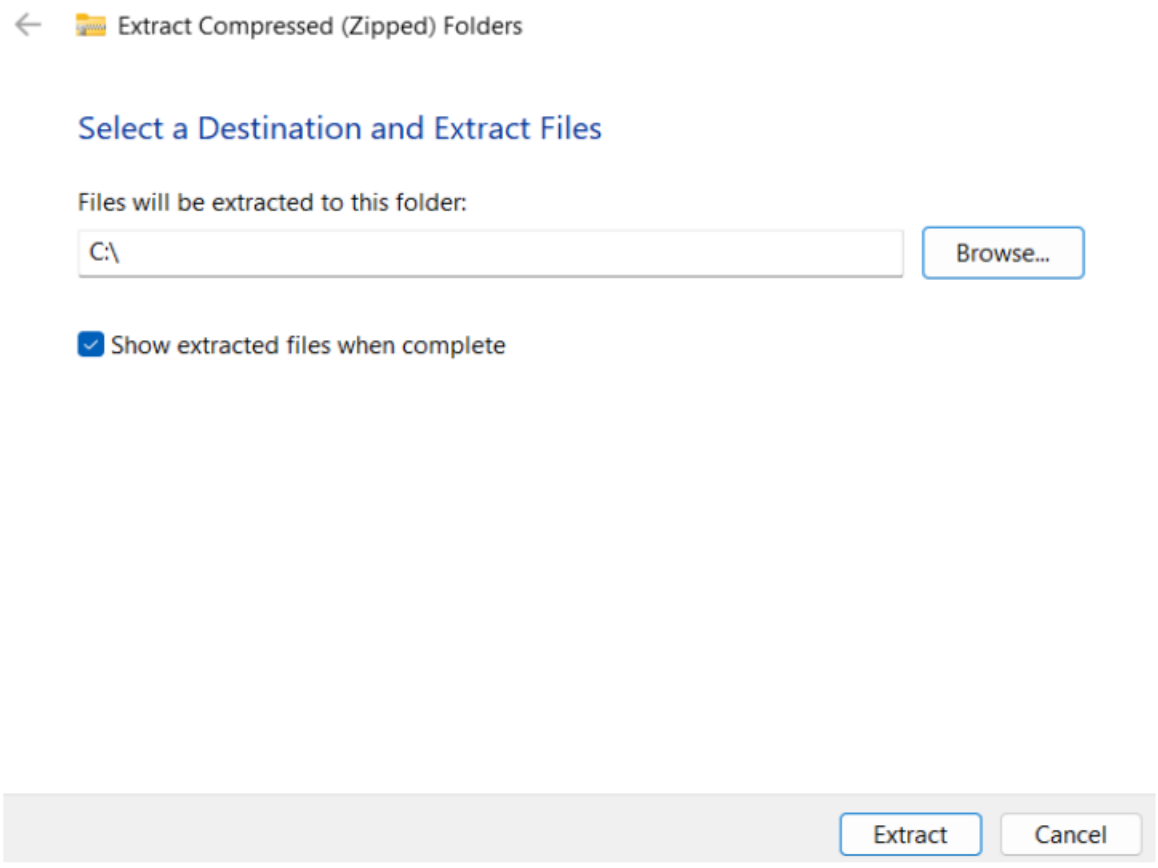


Step 2: To download the latest Flutter SDK, click on the Windows icon > Android

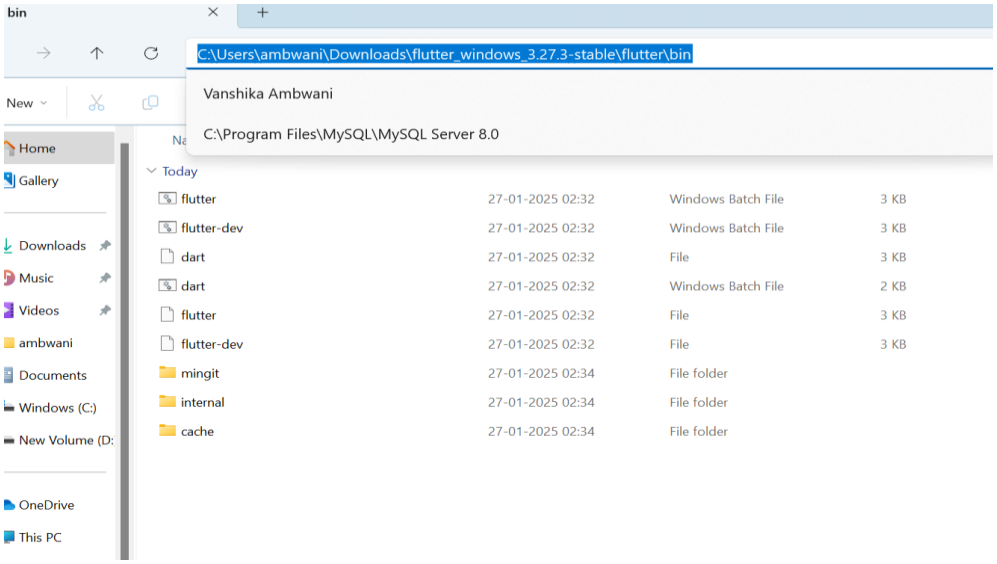


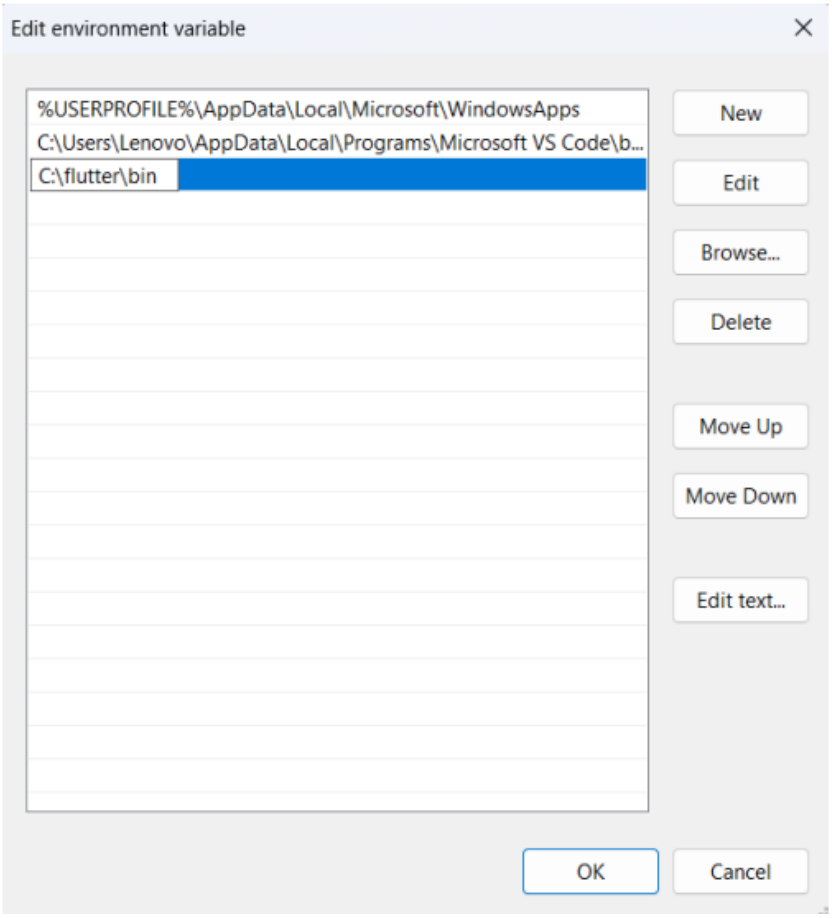
Step 3: For Windows, download the stable release (a .zip file).

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).





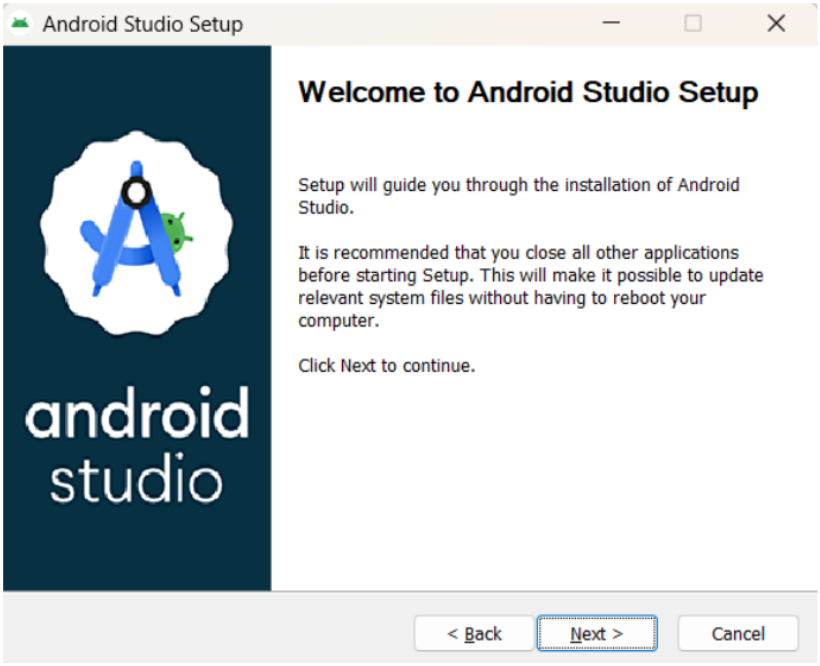
Step 6 : - Now, run the $ flutter command in command prompt

```
  install           Install a Flutter app on an attached device.
  logs              Show log output for running Flutter apps.
  screenshot        Take a screenshot from a connected device.
  symbolize         Symbolize a stack trace from an AOT-compiled Flutter app.

Run "flutter help <command>" for more information about a command.
Run "flutter help -v" for verbose help output, including less commonly used options.

                Welcome to Flutter! - https://flutter.dev

The Flutter tool uses Google Analytics to anonymously report feature usage
statistics and basic crash reports. This data is used to help improve
Flutter tools over time.

Flutter tool analytics are not sent on the very first run. To disable
reporting, type 'flutter config --no-analytics'. To display the current
setting, type 'flutter config'. If you opt out of analytics, an opt-out
event will be sent, and then no further information will be sent by the
Flutter tool.

By downloading the Flutter SDK, you agree to the Google Terms of Service.
The Google Privacy Policy describes how data is handled in this service.

Moreover, Flutter includes the Dart SDK, which may send usage metrics and
crash reports to Google.

Read about data we send with crash reports:
https://flutter.dev/to/crash-reporting

See Google's privacy policy:
```

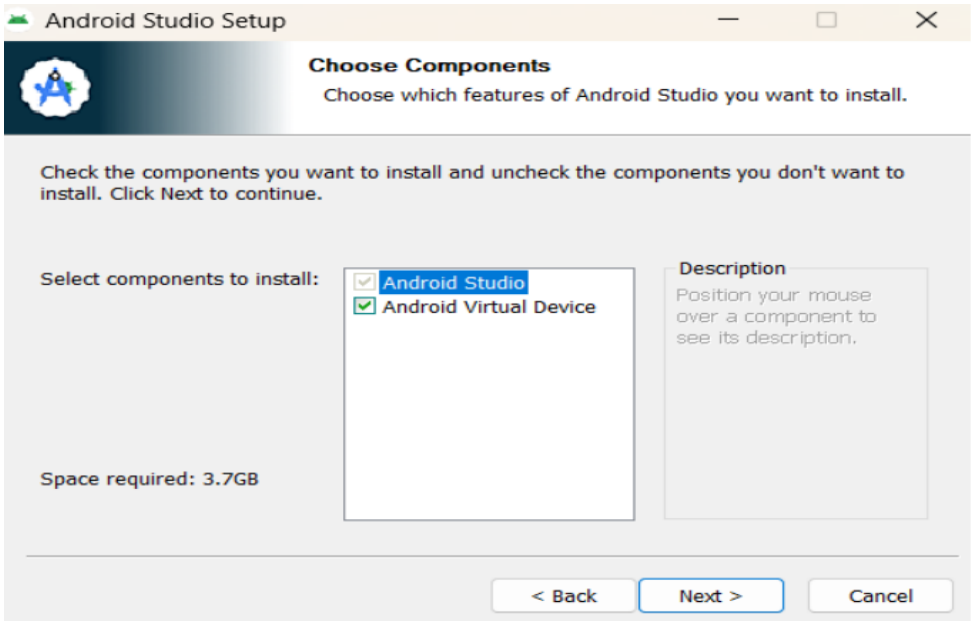
Step 7:- Run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation

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

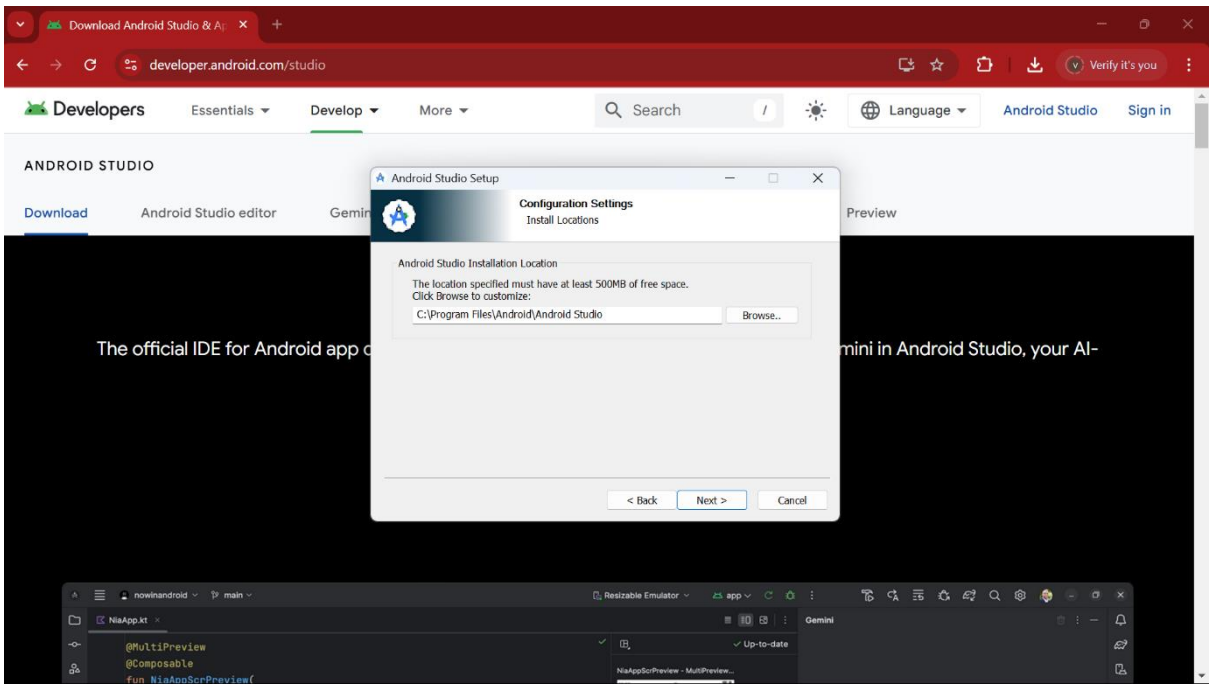
Step 8.1: - When the download is complete, open the .exe file and run it. You will get the following dialog box



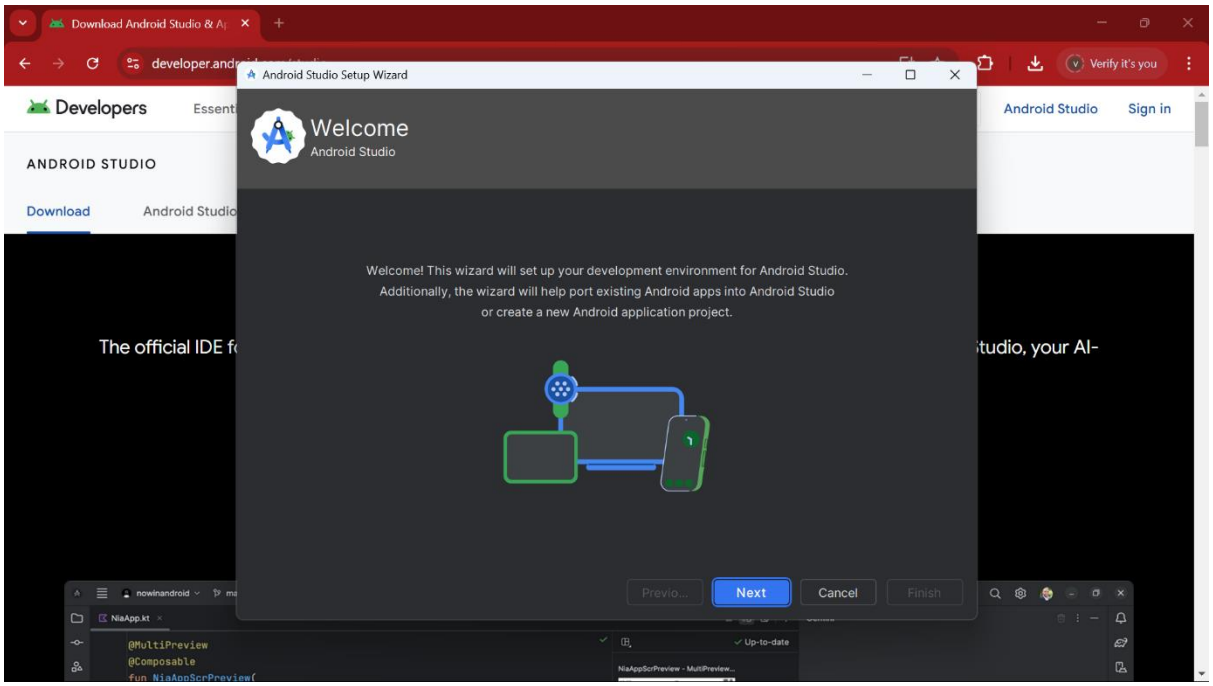
Step 8.2: - Select all the Checkboxes and Click on ‘Next’ Button.

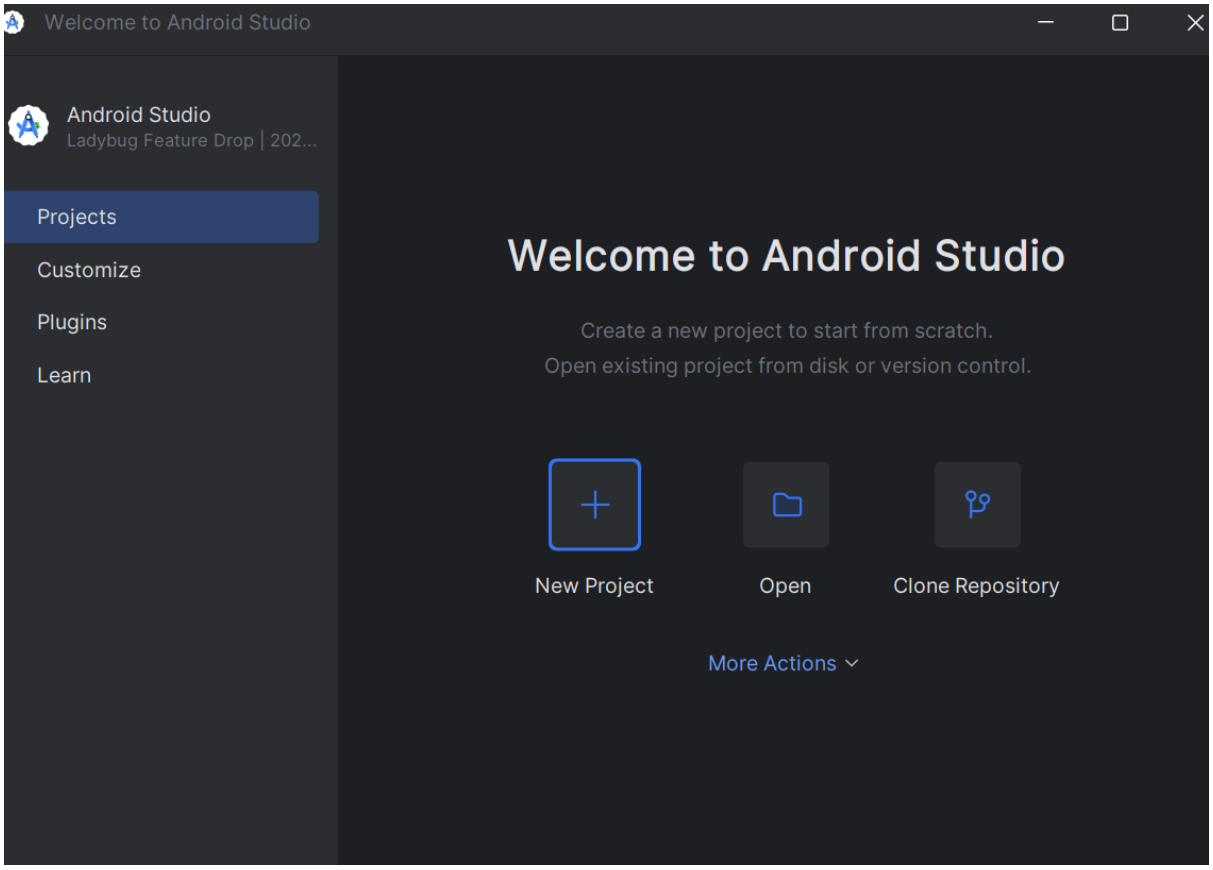


Step 8.3: - Change the destination as per your convenience and click on ‘Next’ Button.



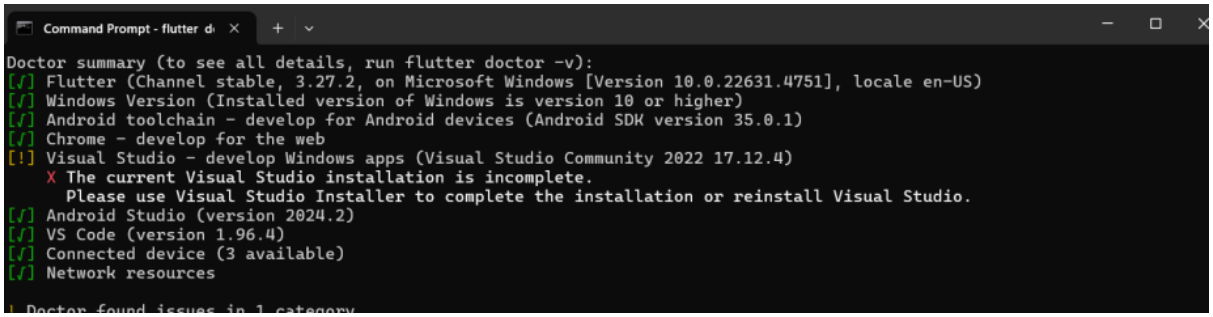
Step 8.4: - Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.

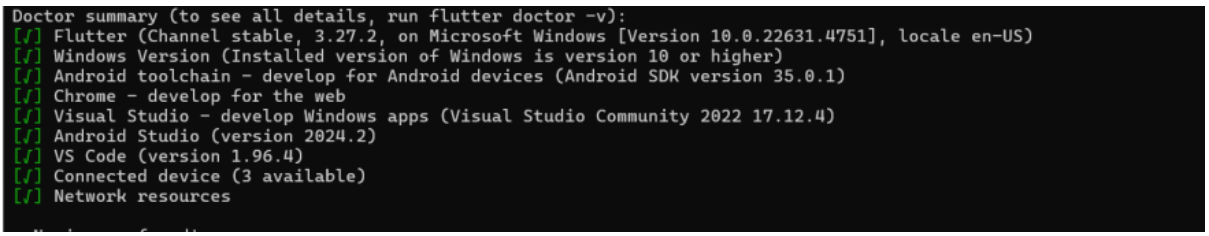




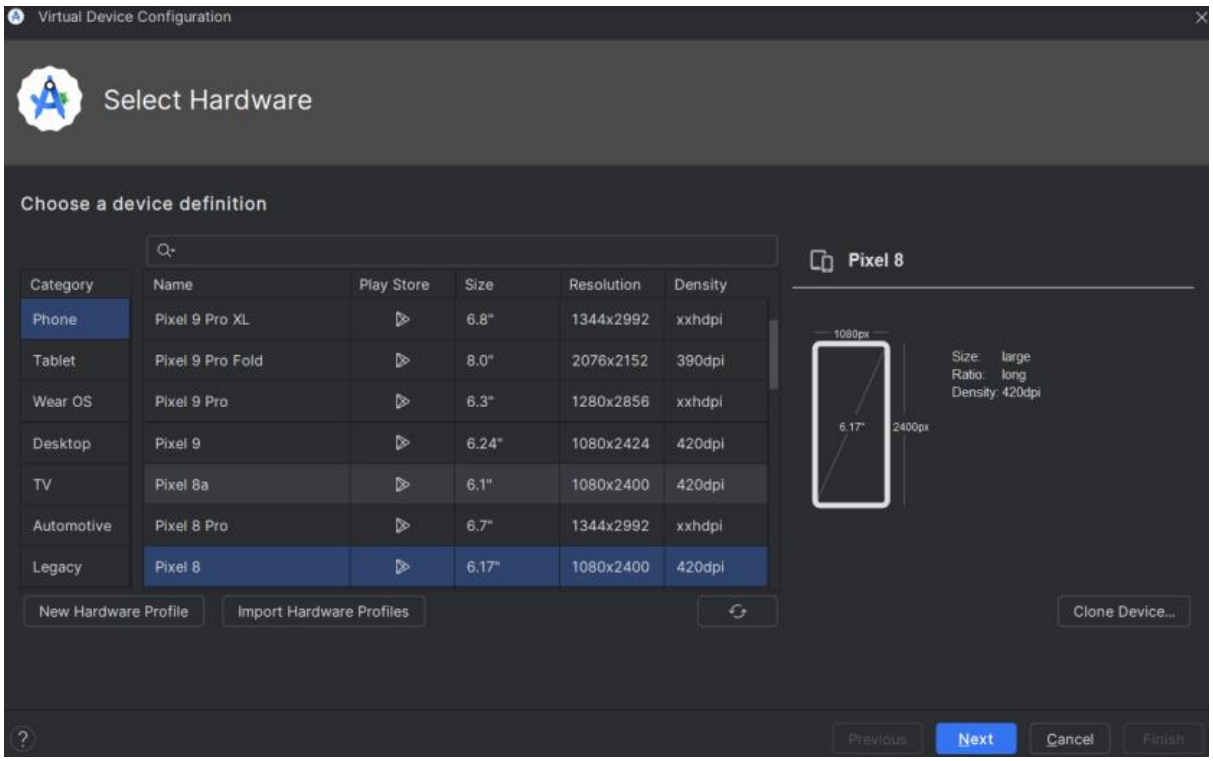
Step 8.5: - Go to Preferences > Appearance & Behavior > System Settings > Android SDK. Select the SDK Tools tab and check Android SDK Command-line Tools and Install it.

Step 9: - Open a terminal and run the following command

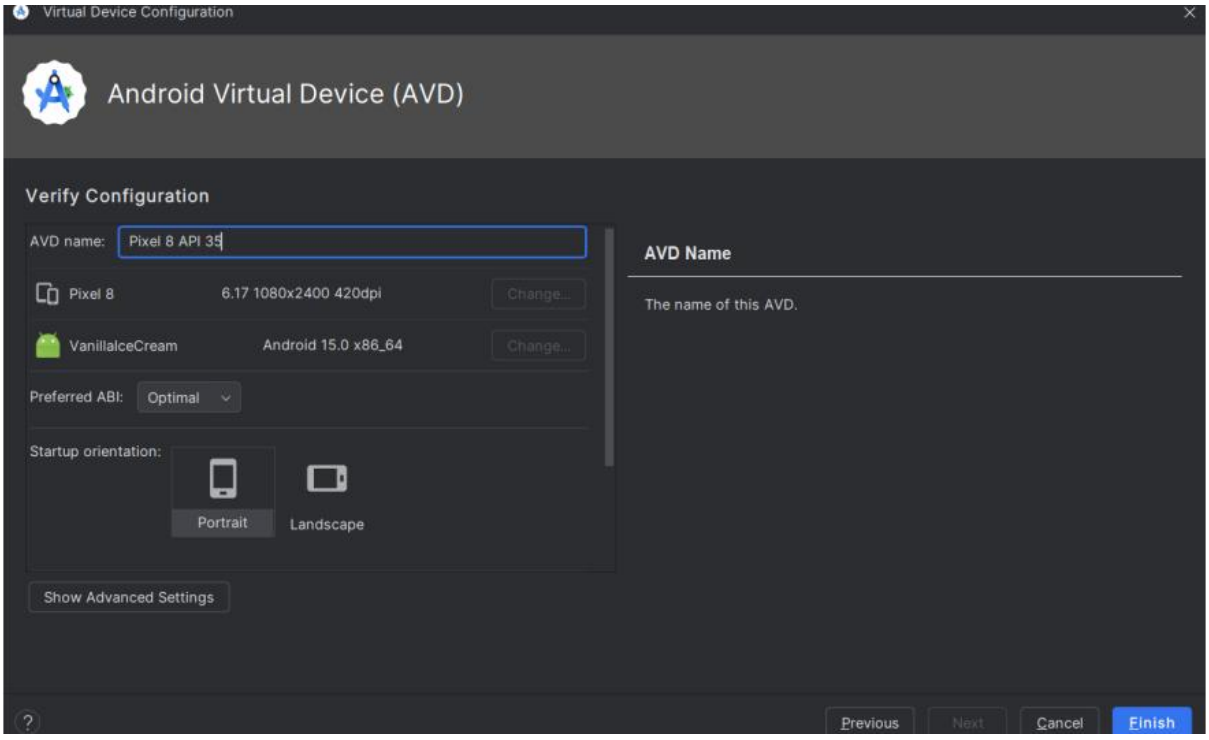




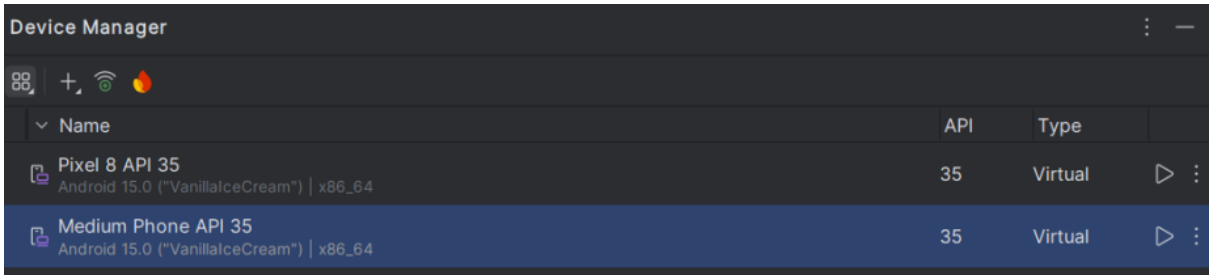
Step 10: - Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application

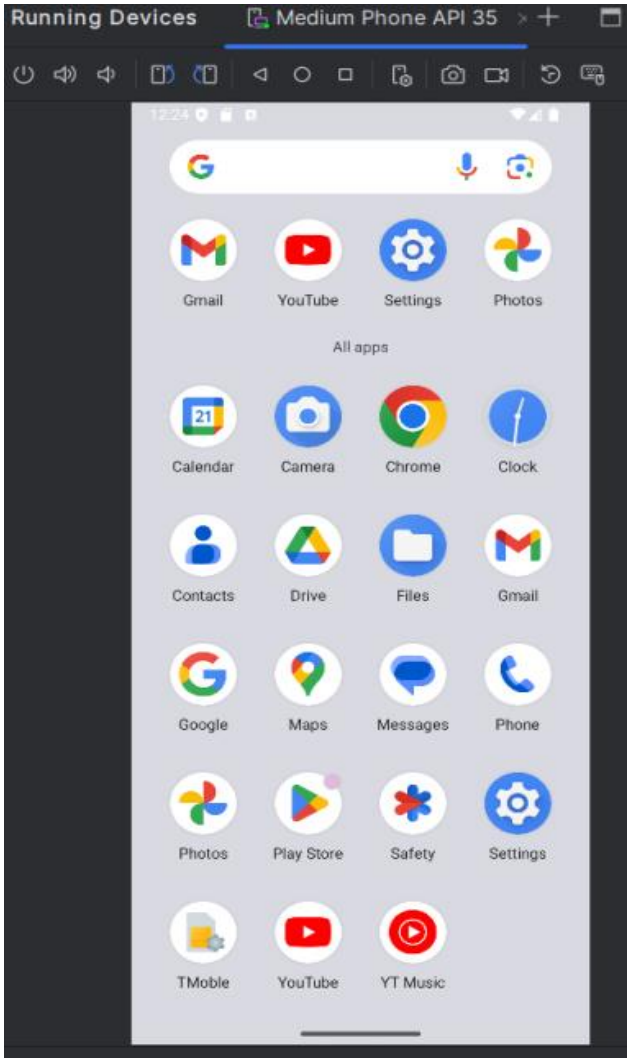


Step 10.1: - Open Android Studio and go to Tools > AVD Manager. Create a new virtual device.



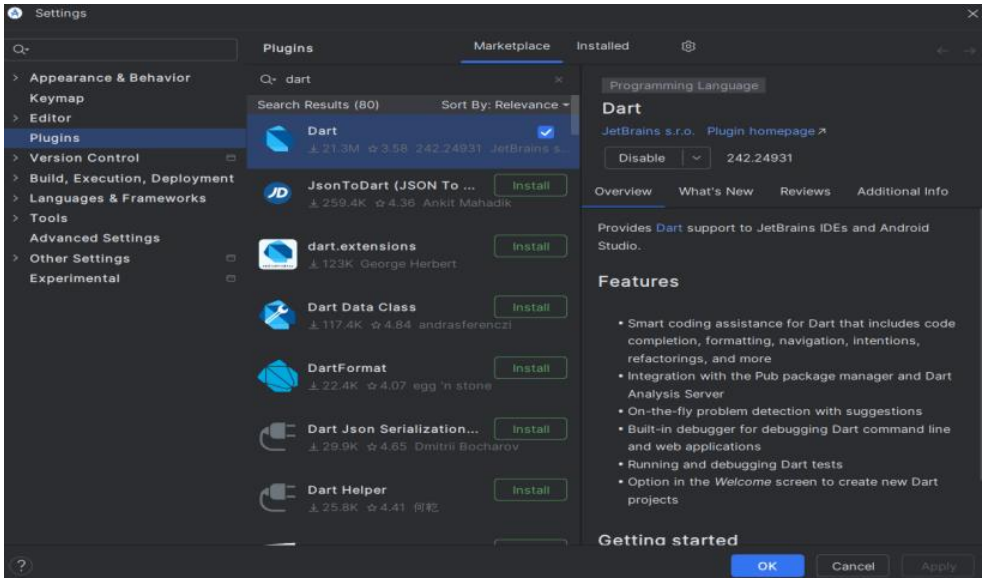
Step 10.2: - Click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen





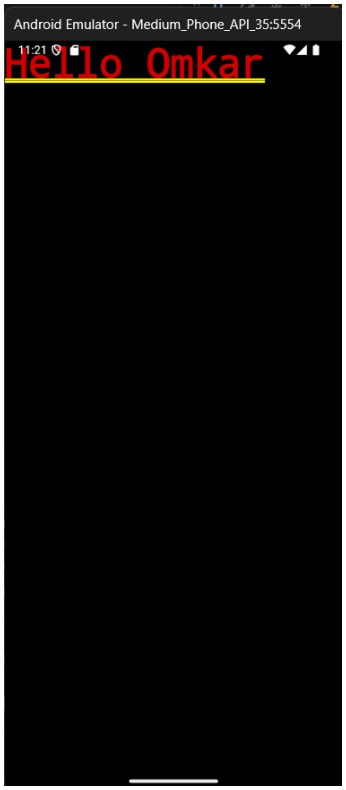
Step 11: - Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself

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

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



Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	17
Name	Omkar Gholap
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO :- 02

AIM: - To design Flutter UI by including common widgets.

Theory: -

Each element on the screen of the Flutter app is a widget. The view of the screen completely depends upon the choice and sequence of the widgets used to build the apps. And the structure of the code of apps is a tree of widgets.
When you made any alteration in the code, the widget rebuilds its description by calculating the difference of previous and current widget to determine the minimal changes for rendering in UI of the app.Widgets are nested with each other to build the app. It means the root of your app is itself a widget, and all the way down is a widget also. For example, a widget can display something, can define design, can handle interaction, etc.
The single child layout widget is a type of widget, which can have only one child widget inside the parent layout widget. These widgets can also contain special layout functionality. Flutter provides us with many single child widgets to make the app UI attractive. If we use these widgets appropriately, it can save us time and make the app code more readable.
The multiple child widgets are a type of widget, which contains more than one child widget, and the layout of these widgets are unique. For example, Row widget laying out of its child widget in a horizontal direction, and Column widget laying out of its child widget in a vertical direction. If we combine the Row and Column widget, then it can build any level of the complex widget.

Type of Widgets

- StatefulWidget
    - A StatefulWidget has state information. It contains mainly two classes: the state object and the widget. It is dynamic because it can change the inner data during the widget lifetime. This widget does not have a build() method. It has a createState() method, which returns a class that extends the Flutters State Class. The examples of the StatefulWidget are Checkbox, Radio, Slider, InkWell, Form, and TextField.
- StatelessWidget
    - The StatelessWidget does not have any state information. It remains static throughout its lifecycle. The examples of the StatelessWidget are Text, Row, Column, Container, etc.

Some of the commonly used widgets

Container – A box widget used for styling with padding, margins, colors, borders, and constraints. It helps in layout structuring and positioning.
Row & Column – Used to arrange widgets in horizontal (Row) or vertical (Column) orientation. They manage spacing, alignment, and distribution of child widgets.
Stack – Overlaps widgets on top of each other, useful for creating layered UIs like banners, tooltips, or floating elements.
Text – Displays text on the screen with customizable font size, color, alignment, and styling options
Image – Loads and displays images from assets, network, or memory with scaling, fit, properties.
Scaffold – Provides a basic layout structure with an app bar, body, floating action button, and bottom navigation.
ListView – A scrollable list widget that efficiently renders large amounts of dynamic content.
Supports both vertical and horizontal scrolling.

GridView – Displays widgets in a grid format, useful for galleries, product listings, or dashboards. It supports dynamic column adjustments.
SizedBox – Used to create space between widgets or define fixed width and height for layout adjustments.
ElevatedButton – A button with elevation that provides a raised effect, customizable with color, shape, and click actions.
TextField – A user input field that supports text entry, keyboard configurations, validation.
AppBar – A top navigation bar that includes a title, actions, and menu icons, commonly used in Scaffold.
BottomNavigationBar – A bar at the bottom of the screen used for navigation between different app sections with icons and labels.
Drawer – A side navigation panel that slides out from the left, typically used for app menus and quick navigation.
Card – A material design component that displays content inside a box with elevation.

Code:

home_screen.dart

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

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

  @override
  State<HomeScreen> createState() =>
_HomeScreenState();
}

class _HomeScreenState extends
State<HomeScreen> {   int
_currentCarouselIndex = 0;   final
PageController _pageController =
PageController();

  final List<Map<String, dynamic>>
carouselData = [
    {
      'icon': Icons.fitness_center,
      'title': 'Featured Workout',
      'description': 'Try our new HIIT routine for
maximum burn!',
      'color': Colors.blueAccent,
    },
    {
      'icon': Icons.local_dining,
      'title': 'Nutrition Tip',
      'description': 'Include more protein in your
meals for muscle gain.',
      'color': Colors.green,
    },
    {
      'icon': Icons.self_improvement,
      'title': 'Mindfulness',
      'description': 'Take a moment to meditate
and destress.',
      'color': Colors.purple,
    },
  ];

  @override   void dispose()
{
_pageController.dispose();
super.dispose();
  }

  @override
  Widget build(BuildContext context)
{     return Scaffold(       appBar:
AppBar(         title: const
Text("Home"),
backgroundColor: Colors.black,
centerTitle: true,
      ),
      backgroundColor: Colors.black,
body: SingleChildScrollView(         child:
Column(           children: [
            // Carousel Section with Custom
Widgets
            SizedBox(               height: 200,
child: PageView.builder(
controller: _pageController,
itemCount: carouselData.length,
```

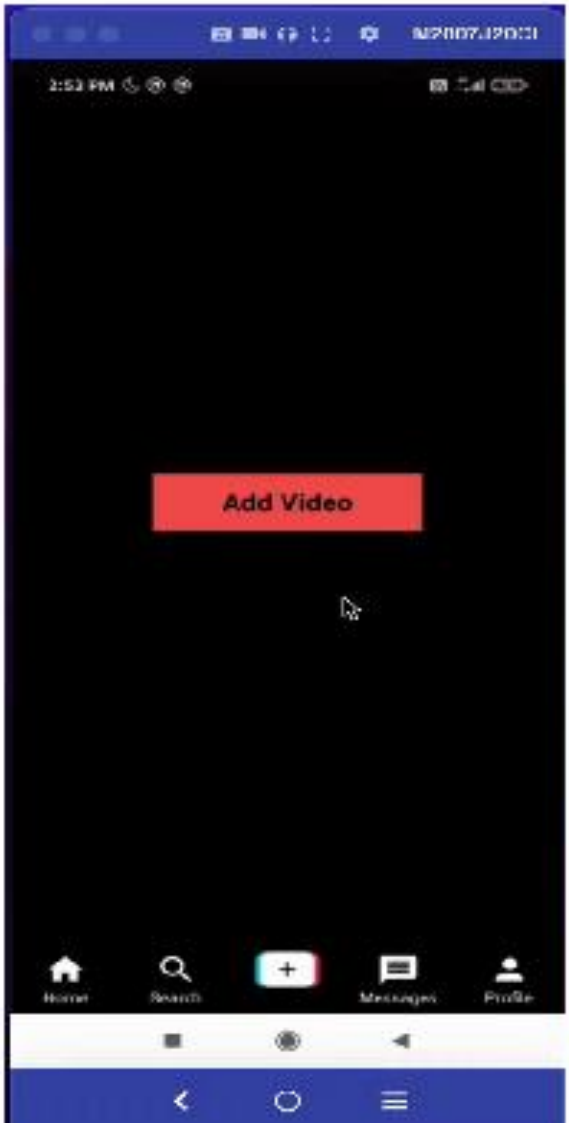
```
onPageChanged: (index) {
setState(() {
                  _currentCarouselIndex = index;
                });
              },
              itemBuilder: (context, index) {
final item = carouselData[index];
return Container(                    margin:
const
EdgeInsets.symmetric(horizontal: 10, vertical:
10),                  decoration:
BoxDecoration(
borderRadius:
BorderRadius.circular(12),
color:
item['color'].withOpacity(0.2),
                ),
padding: const
EdgeInsets.all(16),
                child: Row(
children: [                    Icon(
item['icon'],                      size:
48,                    color:
item['color'],
                  ),
                  const SizedBox(width:
16),                  Expanded(
child: Column(
mainAxisAlignment:
MainAxisAlignment.center,
                    crossAxisAlignment:
CrossAxisAlignment.start,
                    children: [
Text(                    item['title'],
style: const TextStyle(
color: Colors.white,
fontSize: 20,

fontWeight:
FontWeight.bold,
                      ),
                    ),
                    const SizedBox(height:
8),                      Text(
item['description'],                       style:
const TextStyle(                      color:
Colors.white70,
fontSize: 16,
                      ),
                    ),
                  ],
                ),
              );
            },
          ),
        ),
        // Carousel Indicator
Dots          Row(
mainAxisAlignment:
MainAxisAlignment.center,
children:
List.generate(carouselData.length, (index)
{              return AnimatedContainer(
duration: const Duration(milliseconds:
300),                margin: const
EdgeInsets.symmetric(horizontal: 4),
                height: 8,                width:
_currentCarouselIndex == index ? 20 : 8,
decoration: BoxDecoration(
color: _currentCarouselIndex == index ?
Colors.blue : Colors.white54,
borderRadius:
BorderRadius.circular(4),
                ),
              );
            }),
        ),
        const SizedBox(height: 20),

        // Trackers
Section          Padding(
padding: const
EdgeInsets.symmetric(horizontal: 16),


            child: Row(
children: const [
Text(
                  "Trackers",
style: TextStyle(                    color:
Colors.white,                  fontSize: 20,
fontWeight: FontWeight.bold,
                ),
              ),
            ],
          ),
        ),
```

```
          const SizedBox(height:
10),           SizedBox(
height: 150,              child:
ListView(
              scrollDirection: Axis.horizontal,
padding: const
EdgeInsets.symmetric(horizontal: 16),
              children: [
                _buildTrackerCard(
icon: Icons.favorite,
title: "Heart Rate",
value: "72 bpm",
                ),
                const SizedBox(width: 10),
                _buildTrackerCard(
icon: Icons.directions_run,
title: "Steps",                  value:
"5,432",
                ),
                const SizedBox(width: 10),
                _buildTrackerCard(
icon: Icons.local_fire_department,
title: "Calories",                  value: "1,234
cal",
                ),
              ],
            ),
          ),
          const SizedBox(height: 20),

          // Goals Section
Padding(
            padding: const
EdgeInsets.symmetric(horizontal: 16),
            child: Row(
children: const [
Text(
                  "Your Goals",
style: TextStyle(
color: Colors.white,
fontSize: 20,
fontWeight: FontWeight.bold,
                  ),
                ),
              ],
            ),
          ),
          const SizedBox(height: 10),
Padding(
            padding: const
EdgeInsets.symmetric(horizontal: 16),
            child: Row(
children: [
Expanded(
                child: _buildGoalCard(
title: "Remaining Calories",
value: "850 cal",                  icon:
Icons.local_fire_department,
                  color: Colors.orange,
                ),
              ),
              const SizedBox(width:
10),                Expanded(
child: _buildGoalCard(
title: "Water Intake",
value: "5/8 glasses",
icon: Icons.water,                  color:
Colors.lightBlue,
                ),
              ),
            ],
          ),
        ),
        const SizedBox(height: 20),

        // Graph Section
Padding(
          padding: const
EdgeInsets.symmetric(horizontal: 16),
          child: Row(
children: const [
            Text(
              "Progress Graph",
style: TextStyle(                  color:
Colors.white,                fontSize: 20,
fontWeight: FontWeight.bold,
              ),
            ),
          ],
        ),
        const SizedBox(height:
10),          _buildGraphWidget(),
const SizedBox(height: 30),
      ],
    ),
  );
 }
```

```
  // Tracker Card   Widget
_buildTrackerCard({
required IconData icon,
required String title,
required String value,
  }) {
    return Container(       width: 140,
padding: const EdgeInsets.all(16),
decoration: BoxDecoration(         color:
Colors.grey[900],         borderRadius:
BorderRadius.circular(12),
      ),
      child: Column(
mainAxisAlignment:
MainAxisAlignment.center,
        children: [
          Icon(icon, size: 40, color:
Colors.blue),           const SizedBox(height:
10),          Text(             title,
            style: const TextStyle(
color: Colors.white70,
fontSize: 16,
            ),
          ),
          const SizedBox(height: 5),
Text(             value,             style:
const TextStyle(               color:
Colors.white,               fontSize: 18,
fontWeight: FontWeight.bold,
        borderRadius: BorderRadius.circular(12),
      ),
      child: const Center(
child: Text(
          "Graph \n(Progress Over Time)",
textAlign: TextAlign.center,           style:
TextStyle(



            ),
          )
        ],
      ),
    );
  }

  // Goal Card   Widget
_buildGoalCard({
required String title,
required String value,
required IconData icon,
required Color color,
  }) {
    return Container(       padding: const
EdgeInsets.all(16),       decoration:
BoxDecoration(         color: Colors.grey[900],
borderRadius: BorderRadius.circular(12),
      ),
      child: Column(
children: [
          Icon(icon, size: 40, color:
color),           const SizedBox(height:
10),          Text(             title,
            textAlign: TextAlign.center,
style: const TextStyle(
color: Colors.white70,
fontSize: 14,
            ),
          ),
          const SizedBox(height: 5),
Text(             value,             style:
const TextStyle(               color:
Colors.white,               fontSize: 16,
fontWeight: FontWeight.bold,
            ),
          )
        ],
      ),

            color: Colors.white70,
fontSize: 16,
          ),
        ),
      ),
    );
  }
}

    );
  }

  // Placeholder for Graph
Widget   Widget
_buildGraphWidget() {     return
Container(       margin: const
EdgeInsets.symmetric(horizontal:
16),       height: 200,       decoration:
BoxDecoration(         color:
Colors.grey[900], main_screen.dart
```

```
import
'package:flutter/material.dart';
import 'home_screen.dart'; import
'log_food_screen.dart'; import
'settings_screen.dart';

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

  @override
  State<MainScreen> createState() =>
_MainScreenState();
}

class _MainScreenState
extends State<MainScreen> {
int _currentIndex = 0;

  // List of pages corresponding to
each bottom nav item.   final
List<Widget> _pages = const [
    HomeScreen(),
    LogFoodScreen(),
    SettingsScreen(),
  ];

  @override
  Widget build(BuildContext context) {
return Scaffold(       backgroundColor:
Colors.black, // Ensures a black background
for the screen       body:
_pages[_currentIndex], // Display selected
page
      bottomNavigationBar:
BottomNavigationBar(
        backgroundColor: Colors.grey[850],
// Ensure a distinct background color
currentIndex: _currentIndex,
unselectedItemColor: Colors.white70,
selectedItemColor: Colors.blue,
showUnselectedLabels: true,         items:
const [
          BottomNavigationBarItem(
icon: Icon(Icons.home),
label: "Home",
          ),
          BottomNavigationBarItem(
icon: Icon(Icons.fastfood),
label: "Log Food",
          ),
          BottomNavigationBarItem(
icon: Icon(Icons.settings),
label: "Settings",
          ),
        ],
        onTap: (index) {
setState(() {
            _currentIndex = index; // Update the
current selected index
          });
        },
      ),
    );
  }
}
```

Screenshots:


Add Video
Home
Search
Messages
Profile


2:54
2
0
0
rivaanranawat
No Song, Only Inpl
IDK
Home
Search
Messages
Profile

Project Title: **Roll No.**

MAD & PWA Lab

Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	17
Name	Omkar Gholap
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Experiment No. 3

AIM: To include icons, images, fonts in Flutter app

THEORY:

Flutter provides built-in support for adding icons, images, and custom fonts to enhance the visual appeal of applications. These assets help in creating a more engaging and personalized user interface.

1. Adding Icons in Flutter

Icons are commonly used to represent actions, categories, or UI elements. Flutter offers both built-in icons and custom icon support.

1.1. Built-in Material Icons

Flutter provides a rich set of **Material Icons**, which can be used directly in the UI. These icons follow Google's Material Design guidelines and do not require additional setup.

1.2. Custom Icons

For unique icons, custom icon packs can be added using third-party libraries like **FontAwesome** or by importing **SVG** or **PNG** icons as images.

1.3. Vector Icons (SVG)

Vector icons are scalable and provide better quality across different screen sizes. The **flutter_svg** package is commonly used for adding SVG icons.

---

2. Adding Images in Flutter

Images enhance the visual representation of an app. Flutter supports various types of images:

2.1. Asset Images (Local Images)

Local images are stored within the app's **assets** folder. These images must be declared in the `pubspec.yaml` file before use.

2.2. Network Images

Flutter allows loading images from the internet using their URLs. This is useful for displaying dynamic content fetched from APIs.

2.3. Memory and File Images

Flutter can also load images directly from device storage or memory, making it useful for user-uploaded images.

---

3. Adding Custom Fonts in Flutter

Custom fonts improve typography and branding in the app.

3.1. Using System Fonts

Flutter supports default system fonts available on the device, such as Roboto and San Francisco.

3.2. Adding Custom Fonts

To include a custom font, the font file (e.g., **.ttf, .otf**) must be added to the **assets/fonts** directory and declared in `pubspec.yaml`.

3.3. Applying Fonts

Once declared, the custom font can be applied globally or to specific text widgets for a unique look.

CODE & OUTPUT:

PROFILE PAGE:

```dart
import 'package:fancy_shimmer_image/fancy_shimmer_image.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_iconly/flutter_iconly.dart';
import 'package:grocery_app/widgets/heart_btn.dart';
import 'package:provider/provider.dart';

import '../providers/cart_provider.dart';
import '../providers/products_provider.dart';
import '../providers/viewed_prod_provider.dart';
import '../providers/wishlist_provider.dart';
import '../services/utils.dart';
import '../widgets/text_widget.dart';

class ProductDetails extends StatefulWidget {
 static const routeName = '/ProductDetails';

 const ProductDetails({Key? key}) : super(key: key);

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

class _ProductDetailsState extends State<ProductDetails> {
 final _quantityTextController = TextEditingController(text: '1');

 @override
 void dispose() {
   // Clean up the controller when the widget is disposed.
   _quantityTextController.dispose();
   super.dispose();
 }

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

```dart
Size size = Utils(context).getScreenSize;
final Color color = Utils(context).color;
final productProvider = Provider.of<ProductsProvider>(context);
final cartProvider = Provider.of<CartProvider>(context);
final wishlistProvider = Provider.of<WishlistProvider>(context);
final productId = ModalRoute.of(context)!.settings.arguments as String;
final getCurrProduct = productProvider.findProdById(productId);
double usedPrice = getCurrProduct.isOnSale
    ? getCurrProduct.salePrice
    : getCurrProduct.price;
double totalPrice = usedPrice * int.parse(_quantityTextController.text);
bool? _isInCart = cartProvider.getCartItems.containsKey(getCurrProduct.id);

bool? _isInWishlist =
    wishlistProvider.getWishlistItems.containsKey(getCurrProduct.id);

final viewedProdProvider = Provider.of<ViewedProdProvider>(context);
return WillPopScope(
  onWillPop: () async {
    viewedProdProvider.addProductToHistory(productId: productId);
    return true;
  },
  child: Scaffold(
    appBar: AppBar(
        leading: InkWell(
          borderRadius: BorderRadius.circular(12),
          onTap: () =>
              Navigator.canPop(context) ? Navigator.pop(context) : null,
          child: Icon(
            IconlyLight.arrowLeft2,
            color: color,
            size: 24,
          ),
        ),
        elevation: 0,
        backgroundColor: Theme.of(context).scaffoldBackgroundColor),
    body: Column(children: [
      Flexible(
        flex: 2,
        child: FancyShimmerImage(
          imageUrl: getCurrProduct.imageUrl,
          boxFit: BoxFit.scaleDown,
          width: size.width,
          // height: screenHeight * .4,
        ),
      ),
      Flexible(
        flex: 3,
        child: Container(
```

```
          decoration: BoxDecoration(
            color: Theme.of(context).cardColor,
            borderRadius: const BorderRadius.only(
              topLeft: Radius.circular(40),
              topRight: Radius.circular(40),
            ),
          ),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Padding(
                padding:
                    const EdgeInsets.only(top: 20, left: 30, right: 30),
                child: Row(
                  mainAxisAlignment: MainAxisAlignment.spaceBetween,
                  children: [
                    Flexible(
                      child: TextWidget(
                        text: getCurrProduct.title,
                        color: color,
                        textSize: 25,
                        isTitle: true,
                      ),
                    ),
                    HeartBTN(
                      productId: getCurrProduct.id,
                      isInWishlist: _isInWishlist,
                    )
                  ],
                ),
              ),
              Padding(
                padding:
                    const EdgeInsets.only(top: 20, left: 30, right: 30),
                child: Row(
                  mainAxisAlignment: MainAxisAlignment.start,
                  crossAxisAlignment: CrossAxisAlignment.center,
                  children: [
                    TextWidget(
                      text: '\₹${usedPrice.toStringAsFixed(2)}',
                      color: Colors.green,
                      textSize: 22,
                      isTitle: true,
                    ),
                    TextWidget(
                      text: getCurrProduct.isPiece ? '/Piece' : '/Kg',
                      color: color,
                      textSize: 12,
                      isTitle: false,
```

```
            ),
            const SizedBox(
              width: 10,
            ),
            Visibility(
              visible: getCurrProduct.isOnSale ? true : false,
              child: Text(
                '\₹${getCurrProduct.price.toStringAsFixed(2)}',
                style: TextStyle(
                    fontSize: 15,
                    color: color,
                    decoration: TextDecoration.lineThrough),
              ),
            ),
            const Spacer(),
            Container(
              padding: const EdgeInsets.symmetric(
                  vertical: 4, horizontal: 8),
              decoration: BoxDecoration(
                  color: const Color.fromRGBO(63, 200, 101, 1),
                  borderRadius: BorderRadius.circular(5)),
              child: TextWidget(
                text: 'Free delivery',
                color: Colors.white,
                textSize: 20,
                isTitle: true,
              ),
            ),
          ],
        ),
      ),
      const SizedBox(
        height: 30,
      ),
      Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          quantityControl(
            fct: () {
              if (_quantityTextController.text == '1') {
                return;
              } else {
                setState(() {
                  _quantityTextController.text =
                      (int.parse(_quantityTextController.text) - 1)
                          .toString();
                });
              }
            },
```

```
                  icon: CupertinoIcons.minus,
                  color: Colors.red,
                ),
                const SizedBox(
                  width: 5,
                ),
                Flexible(
                  flex: 1,
                  child: TextField(
                    controller: _quantityTextController,
                    key: const ValueKey('quantity'),
                    keyboardType: TextInputType.number,
                    maxLines: 1,
                    decoration: const InputDecoration(
                      border: UnderlineInputBorder(),
                    ),
                    textAlign: TextAlign.center,
                    cursorColor: Colors.green,
                    enabled: true,
                    inputFormatters: [
                      FilteringTextInputFormatter.allow(RegExp('[0-9]')),
                    ],
                    onChanged: (value) {
                      setState(() {
                        if (value.isEmpty) {
                          _quantityTextController.text = '1';
                        } else {}
                      });
                    },
                  ),
                ),
                const SizedBox(
                  width: 5,
                ),
                quantityControl(
                  fct: () {
                    setState(() {
                      _quantityTextController.text =
                          (int.parse(_quantityTextController.text) + 1)
                              .toString();
                    });
                  },
                  icon: CupertinoIcons.plus,
                  color: Colors.green,
                ),
              ],
            ),
            const Spacer(),
            Container(
```

```
                    width: double.infinity,
                    padding: const EdgeInsets.symmetric(
                        vertical: 20, horizontal: 30),
                    decoration: BoxDecoration(
                      color: Theme.of(context).colorScheme.secondary,
                      borderRadius: const BorderRadius.only(
                        topLeft: Radius.circular(20),
                        topRight: Radius.circular(20),
                      ),
                    ),
                    child: Row(
                      mainAxisAlignment: MainAxisAlignment.spaceBetween,
                      children: [
                        Flexible(
                          child: Column(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                              TextWidget(
                                text: 'Total',
                                color: Colors.red.shade300,
                                textSize: 20,
                                isTitle: true,
                              ),
                              const SizedBox(
                                height: 5,
                              ),
                              FittedBox(
                                child: Row(
                                  children: [
                                    TextWidget(
                                      text:

'\₹${totalPrice.toStringAsFixed(2)}/',
                                      color: color,
                                      textSize: 20,
                                      isTitle: true,
                                    ),
                                    TextWidget(
                                      text:
'₹${_quantityTextController.text}Kg',
                                      color: color,
                                      textSize: 16,
                                      isTitle: false,
                                    ),
                                  ],
                                ),
                              ),
                            ],
                          ),
```

```
                    ),
                    const SizedBox(
                      width: 8,
                    ),
                    Flexible(
                      child: Material(
                        color: Colors.green,
                        borderRadius: BorderRadius.circular(10),
                        child: InkWell(
                          onTap: _isInCart
                              ? null
                              : () {
                                  // if (_isInCart) {
                                  //   return;
                                  // }
                                  cartProvider.addProductsToCart(
                                      productId: getCurrProduct.id,
                                      quantity: int.parse(
                                          _quantityTextController.text));
                                },
                          borderRadius: BorderRadius.circular(10),
                          child: Padding(
                              padding: const EdgeInsets.all(12.0),
                              child: TextWidget(
                                  text:
                                      _isInCart ? 'In cart' : 'Add to
cart',
                                  color: Colors.white,
                                  textSize: 18)),
                        ),
                      ),
                    ),
                  ],
                ),
              ),
            ],
          ),
        )
      ]),
    ),
  );
}

Widget quantityControl(
    {required Function fct, required IconData icon, required Color color}) {
  return Flexible(
    flex: 2,
    child: Material(
```

```
      borderRadius: BorderRadius.circular(12),
      color: color,
      child: InkWell(
          borderRadius: BorderRadius.circular(12),
          onTap: () {
            fct();
          },
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Icon(
              icon,
              color: Colors.white,
              size: 25,
            ),
          )),
    ),
  );
 }
}
```

HOME PAGE:

```
import 'package:flutter/cupertino.dart';
import 'package:grocery_app/models/wishlist_model.dart';

class WishlistProvider with ChangeNotifier {
 Map<String, WishlistModel> _wishlistItems = {};

 Map<String, WishlistModel> get getWishlistItems {
   return _wishlistItems;
 }

 void addRemoveProductToWishlist({required String productId}) {
   if (_wishlistItems.containsKey(productId)) {
     removeOneItem(productId);
   } else {
     _wishlistItems.putIfAbsent(
         productId,
         () => WishlistModel(
             id: DateTime.now().toString(), productId: productId));
   }
   notifyListeners();
 }

 void removeOneItem(String productId) {
   _wishlistItems.remove(productId);
   notifyListeners();
 }

 void clearWishlist() {
   _wishlistItems.clear();
   notifyListeners();
 }
}
```

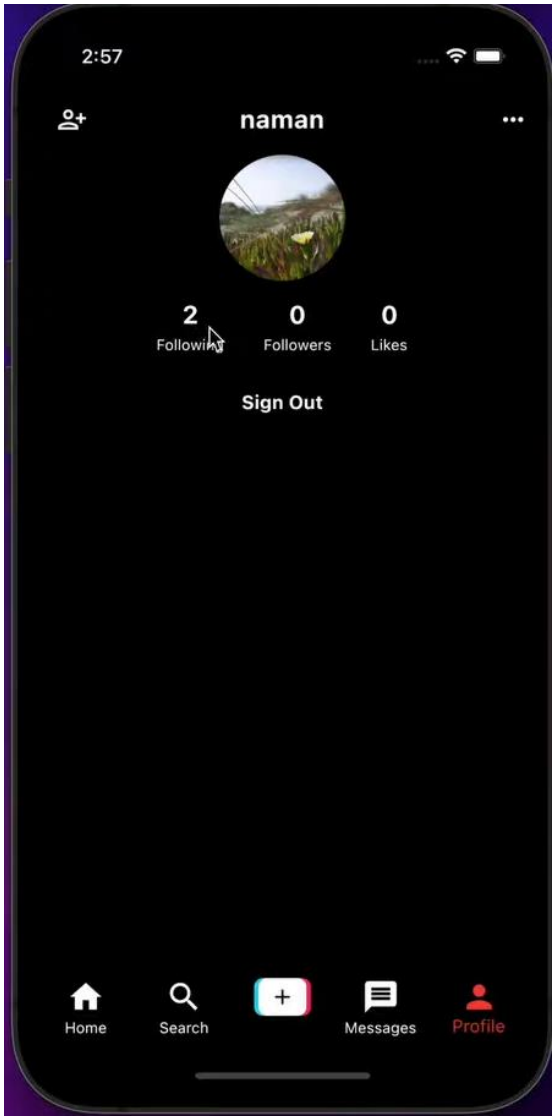
SEARCH PAGE:

```dart
import 'package:flutter/material.dart';
import 'package:grocery_app/services/utils.dart';
import 'package:grocery_app/widgets/categories_widget.dart';
import 'package:grocery_app/widgets/text_widget.dart';

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

 List<Color> gridColors = [
   const Color(0xff53B175),
   const Color(0xffF8A44C),
   const Color(0xffF7A593),
   const Color(0xffD3B0E0),
   const Color(0xffFDE598),
   const Color(0xffB7DFF5),
 ];

List<Map<String, dynamic>> catInfo = [
   {
     'imgPath': 'assets/images/cat/fruits.png',
     'catText': 'Fruits',
   },
   {
     'imgPath': 'assets/images/cat/veg.png',
     'catText': 'Vegetables',
   },
   {
     'imgPath': 'assets/images/cat/Spinach.png',
     'catText': 'Herbs',
   },
   {
     'imgPath': 'assets/images/cat/nuts.png',
     'catText': 'Nuts',
   },
   {
     'imgPath': 'assets/images/cat/spices.png',
     'catText': 'Spices',
```

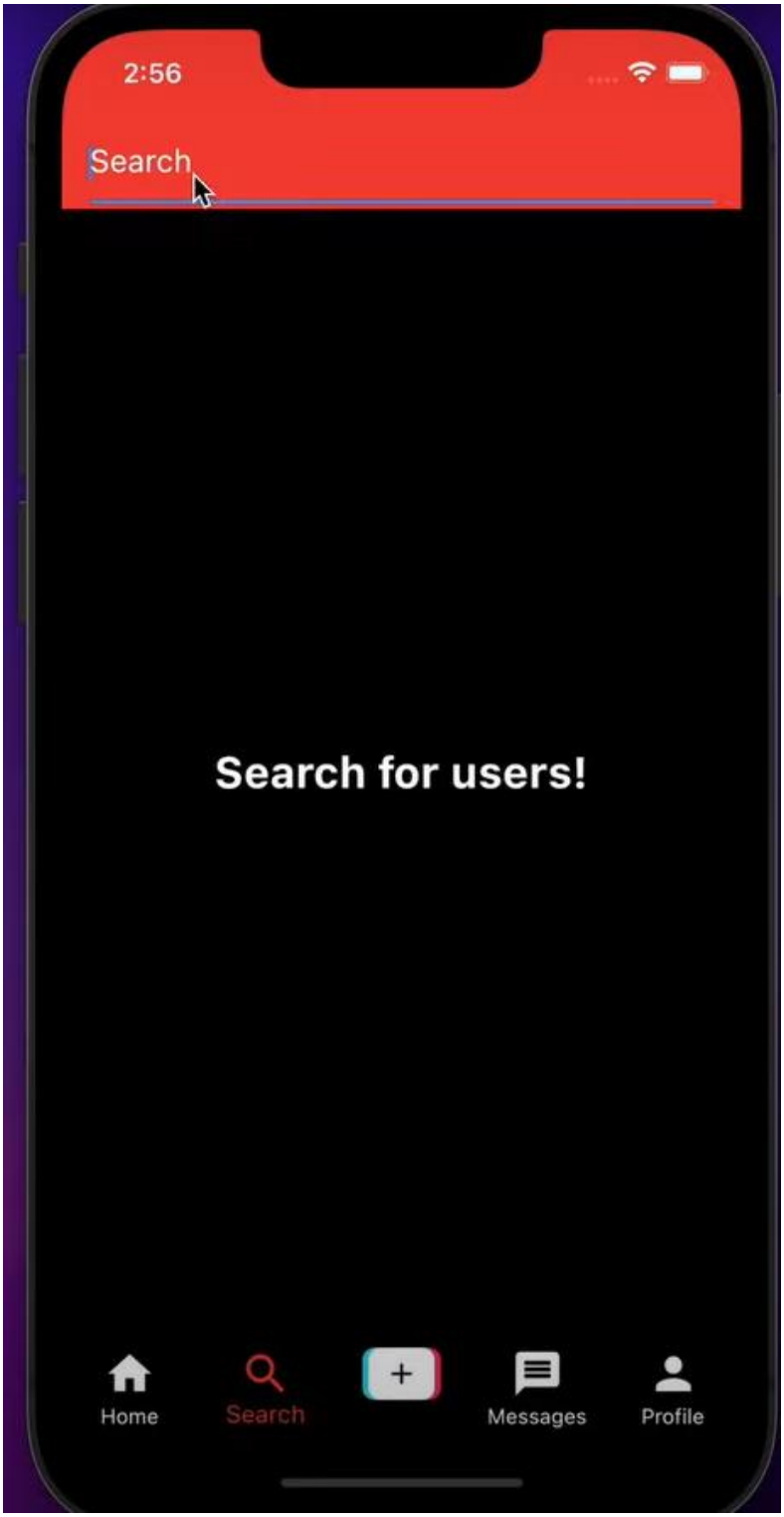
```
  },
   {
    'imgPath': 'assets/images/cat/grains.png',
    'catText': 'Grains',
  },
];
@override
Widget build(BuildContext context) {

  final utils = Utils(context);
  Color color = utils.color;
  return Scaffold(
      appBar: AppBar(
        elevation: 0,
        backgroundColor: Theme.of(context).scaffoldBackgroundColor,
        title: TextWidget(
          text: 'Categories',
          color: color,
          textSize: 24,
          isTitle: true,
        ),
      ),
      body: Padding(
        padding: const EdgeInsets.all(8.0),
        child: GridView.count(
          crossAxisCount: 2,
          childAspectRatio: 240 / 250,
          crossAxisSpacing: 10, // Vertical spacing
          mainAxisSpacing: 10, // Horizontal spacing
          children: List.generate(6, (index) {
            return CategoriesWidget(
              catText: catInfo[index]['catText'],
              imgPath: catInfo[index]['imgPath'],
              passedColor: gridColors[index],
            );
          }),
        ),
      ));
}
}
```

Images:

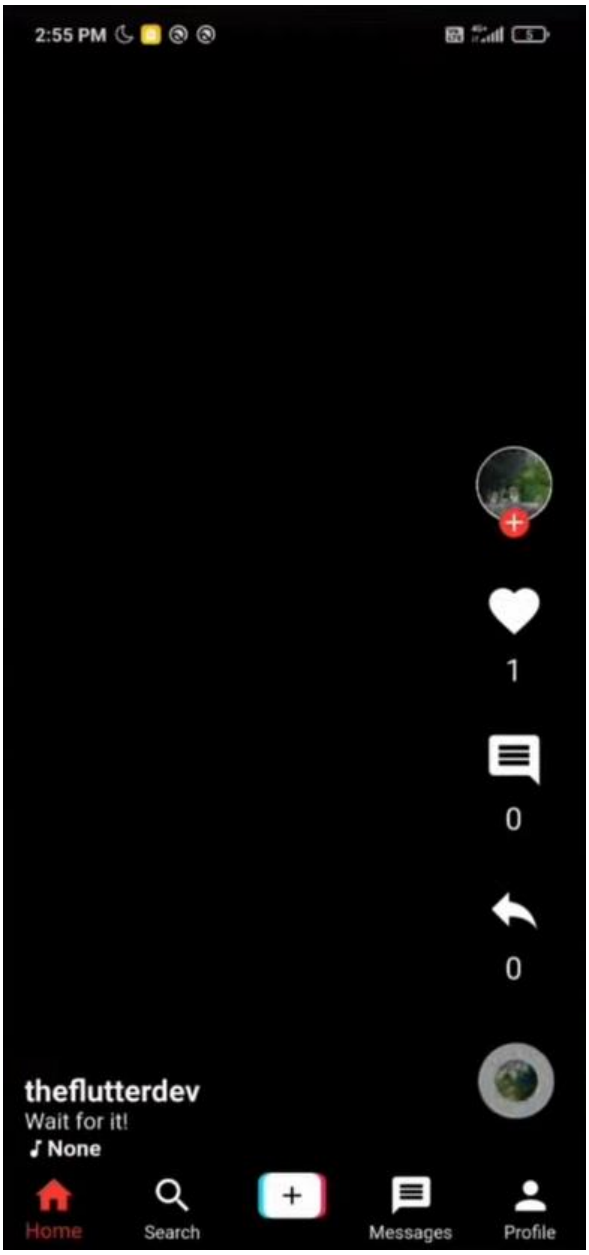
1. PROFILE SCREEN



2. SEARCH SCREEN



3. HOME SCREEN



Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	17
Name	Omkar Gholap
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Exp 4

AIM:- To create an interactive Form using form widget

Code:-

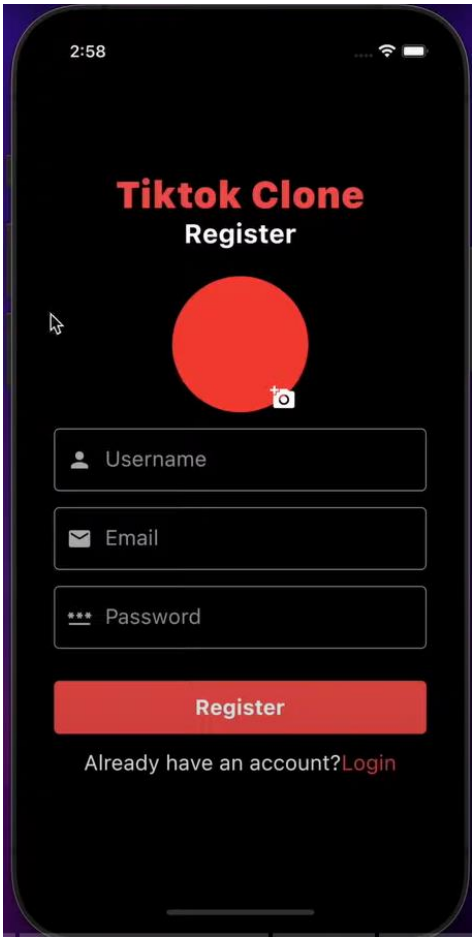
signup_screen.dart

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

class SignupScreen extends StatefulWidget {
  @override
  _SignupScreenState createState() => _SignupScreenState();
}

class _SignupScreenState extends State<SignupScreen> {
  final _formKey = GlobalKey<FormState>();
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();
  final _confirmPasswordController = TextEditingController();
  String _errorMessage = '';

  void _signup() {
    if (_formKey.currentState!.validate()) {
      if (_passwordController.text != _confirmPasswordController.text) {
        setState(() {
          _errorMessage = 'Passwords do not match';
        });
        return;
      }

      // Simulate successful signup
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => ExpensesScreen()),
      );
    }
  }

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

```
    return Scaffold(
      appBar: AppBar(title: Text('Sign Up', style: TextStyle(fontWeight: FontWeight.bold))),
      body: Padding(
        padding: EdgeInsets.all(16.0),
        child: Form(
          key: _formKey,
          child: Column(
            children: [
              TextFormField(
                controller: _emailController,
                decoration: InputDecoration(
                  labelText: 'Email',
                  prefixIcon: Icon(Icons.email),
                ),
                validator: (value) => value!.isEmpty ? 'Enter email' : null,
              ),
              SizedBox(height: 16),
              TextFormField(
                controller: _passwordController,
                obscureText: true,
                decoration: InputDecoration(
                  labelText: 'Password',
                  prefixIcon: Icon(Icons.lock),
                ),
                validator: (value) => value!.length < 6 ? 'Min 6 chars' : null,
              ),
              SizedBox(height: 16),
              TextFormField(
                controller: _confirmPasswordController,
                obscureText: true,
                decoration: InputDecoration(
                  labelText: 'Confirm Password',
                  prefixIcon: Icon(Icons.lock_outline),
                ),
                validator: (value) => value!.isEmpty ? 'Confirm password' : null,
              ),
              SizedBox(height: 16),
              if (_errorMessage.isNotEmpty)
                Text(_errorMessage, style: TextStyle(color: Colors.red, fontWeight:
FontWeight.bold)),
              SizedBox(height: 16),
              ElevatedButton(
                onPressed: _signup,
                child: Text('Sign Up', style: TextStyle(fontWeight: FontWeight.bold)),
```

```
  ),
  SizedBox(height: 16),
  Row(
    children: [
      Expanded(child: Divider(thickness: 1)),
      Padding(
        padding: EdgeInsets.symmetric(horizontal: 8.0),
        child: Text('OR', style: TextStyle(fontWeight: FontWeight.bold)),
      ),
      Expanded(child: Divider(thickness: 1)),
    ],
  ),
  SizedBox(height: 16),
  ElevatedButton.icon(
    onPressed: () {},
    icon: Icon(Icons.phone),
    label: Text('Sign up with Phone'),
  ),
  SizedBox(height: 16),
  ElevatedButton.icon(
    onPressed: () {},
    icon: Icon(Icons.g_mobiledata),
    label: Text('Sign up with Google'),
  ),
  SizedBox(height: 16),
  ElevatedButton.icon(
    onPressed: () {},
    icon: Icon(Icons.apple),
    label: Text('Sign up with Apple'),
  ),
  SizedBox(height: 16),
  GestureDetector(
    onTap: () {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => LoginScreen()),
      );
    },
    child: Text(
      'Already registered? Login',
      style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold, color: Colors.blue),
    ),
  ),
],
```

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



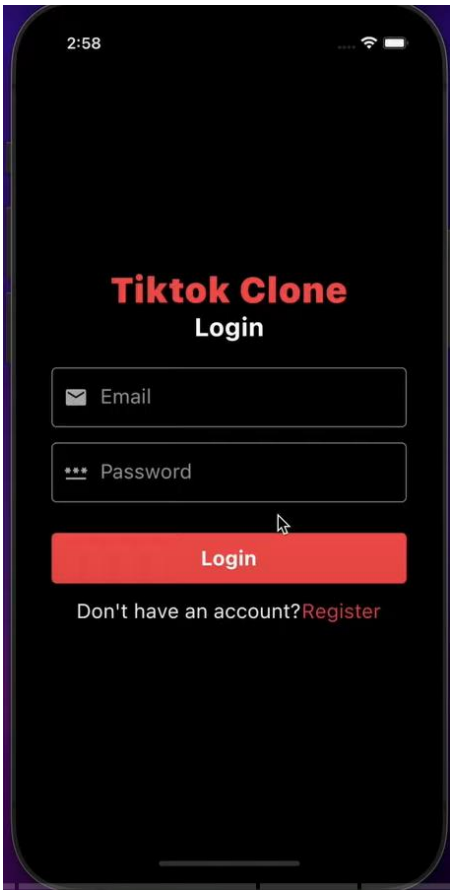
login_screen:-

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

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

```
class _LoginScreenState extends State<LoginScreen> {
  final _formKey = GlobalKey<FormState>();
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();
  String _errorMessage = '';

  void _login() {
    if (_formKey.currentState!.validate()) {
      // Simulate successful login
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => ExpensesScreen()),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Login', style: TextStyle(fontWeight: FontWeight.bold))),
      body: Padding(
        padding: EdgeInsets.all(16.0),
        child: Form(
          key: _formKey,
          child: Column(
            children: [
              TextFormField(
                controller: _emailController,
                decoration: InputDecoration(
                  labelText: 'Email',
                  prefixIcon: Icon(Icons.email),
                ),
                validator: (value) => value!.isEmpty ? 'Enter email' : null,
              ),
              SizedBox(height: 16),
              TextFormField(
                controller: _passwordController,
                obscureText: true,
                decoration: InputDecoration(
                  labelText: 'Password',
                  prefixIcon: Icon(Icons.lock),
                ),
                validator: (value) => value!.length < 6 ? 'Min 6 chars' : null,
              ),
```

```
            SizedBox(height: 16),
            if (_errorMessage.isNotEmpty)
              Text(_errorMessage, style: TextStyle(color: Colors.red, fontWeight:
FontWeight.bold)),
            SizedBox(height: 16),
            ElevatedButton(
              onPressed: _login,
              child: Text('Login', style: TextStyle(fontWeight: FontWeight.bold)),
            ),
            SizedBox(height: 16),
            Row(
              children: [
                Expanded(child: Divider(thickness: 1)),
                Padding(
                  padding: EdgeInsets.symmetric(horizontal: 8.0),
                  child: Text('OR', style: TextStyle(fontWeight: FontWeight.bold)),
                ),
                Expanded(child: Divider(thickness: 1)),
              ],
            ),
            SizedBox(height: 16),
            ElevatedButton.icon(
              onPressed: () {},
              icon: Icon(Icons.phone),
              label: Text('Login with Phone'),
            ),
            SizedBox(height: 16),
            ElevatedButton.icon(
              onPressed: () {},
              icon: Icon(Icons.g_mobiledata),
              label: Text('Login with Google'),
            ),
            SizedBox(height: 16),
            ElevatedButton.icon(
              onPressed: () {},
              icon: Icon(Icons.apple),
              label: Text('Login with Apple'),
            ),
            SizedBox(height: 16),
            GestureDetector(
              onTap: () {
                Navigator.pushReplacement(
                  context,
                  MaterialPageRoute(builder: (context) => SignupScreen()),
                );
```

```
            },
            child: Text(
              'Don’t have an account? Sign Up',
              style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold, color: Colors.blue),
            ),
          ),
        ],
      ),
    ),
  ),
 );
 }
}
```



Project Title: **Roll No.**

MAD & PWA Lab

Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	17
Name	Omkar Gholap
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: - 05

Name:- Omkar Gholap **Class:-** D15A **Roll:No: -** 17

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

Theory: -

In Flutter, the screens and pages are known as routes, and these routes are just a widget. In Android, a route is similar to an Activity.

In any mobile app, navigating to different pages defines the workflow of the application, and the way to handle the navigation is known as routing. Flutter provides a basic routing class MaterialPageRoute and two methods Navigator.push() and Navigator.pop() that shows how to navigate between two routes. The following steps are required to start navigation in your application.

Gestures enable the app to respond to user interactions, making the application more dynamic and responsive.

Navigation and Routing in Flutter

Navigation is the process of moving between different screens or pages in an app. Flutter provides a simple and effective way to handle this through the use of the Navigator widget and routes.

1. Using Navigator Widget

The Navigator widget manages a stack of routes, allowing for pushing and popping routes on the stack.

- **Pushing a Route**: To navigate to a new screen, use Navigator.push().
- **Popping a Route**: To go back to the previous screen, use Navigator.pop().

```
ElevatedButton(
onPressed: () {
```

```
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => SecondScreen()),
    );},
);
```

2. Named Routes

Flutter also allows the use of named routes to navigate, which can make the routing process cleaner, especially in larger applications.

```
MaterialApp(
  initialRoute:            '/',
routes: {
    '/': (context) => HomeScreen(),
    '/second': (context) => SecondScreen(),
  },
);
```

Navigate to the route using Navigator.pushNamed()

```
Navigator.pushNamed(context, '/second');
```

Handling Gestures in Flutter

Gestures refer to user interactions with the app, such as taps, swipes, pinches, and drags. Flutter provides several widgets and gesture detectors to handle these interactions.

Tap Gestures

The most common gesture is the tap, which can be handled using the GestureDetector widget or specific buttons like InkWell or ElevatedButton.

Long Press Gesture

For long press gestures, Flutter provides the onLongPress callback in GestureDetector or InkWell.

Swipe and Drag Gestures

Flutter also provides swipe and drag gesture handling. The onHorizontalDragUpdate and onVerticalDragUpdate callbacks are used for dragging gestures.

Code: -

Home_page.dart main.dart

```
import 'package:flutter/material.dart';
import 'pages/login_page.dart'; import
'pages/register_page.dart'; import
'pages/otp_verification_page.dart'; import
'pages/myaccountpage.dart'; import
'pages/home_page.dart';


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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
return MaterialApp(
debugShowCheckedModeBanner: false,
     title: 'AgriApp',       theme: ThemeData(
primarySwatch: Colors.green,         colorScheme:
ColorScheme.fromSeed(seedColor:
Color(0xFF6A9A5B)),
      ),
      initialRoute: '/login', // Set initial page
routes: {
        '/login': (context) => LoginPage(),
        '/register': (context) => RegistrationPage(),
        '/otp': (context) => OtpVerificationPage(),
        '/myaccount': (context) => MyAccountPage(),
        '/home': (context) => HomePage(),
      },
    );
  } }
```

Login_page.dart

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

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

class _LoginPageState extends
State<LoginPage> {
  final _formKey = GlobalKey<FormState>();

  @override
  Widget build(BuildContext context) {
return Scaffold(       backgroundColor:
Colors.white,       body: SafeArea(
child: SingleChildScrollView(           child:
Padding(            padding: const
EdgeInsets.all(16.0),           child:
Column(             crossAxisAlignment:
CrossAxisAlignment.stretch,
            children: [
SizedBox(height: 50),
              // Logo                Center(
child: Image.asset(
'assets/images/logo.png',
height: 150,
                ),
              ),
              SizedBox(height: 16),
              // Title
              Text(
                'AgriApp: The mix of Agriculture &
Smart, Scientific, Sustainable, Modern
Technology Methods for Precision Farming.',
textAlign: TextAlign.center,                style:
TextStyle(
                  fontSize: 15,
color: Colors.black,
                ),
              ),
              SizedBox(height: 32),
              // Form                Form(
key: _formKey,                child:
Column(                   children: [
_buildTextField(                    label:
'Email',                  hint: 'Enter
your email',                    icon:
Icons.email,                    validator:
(value) {                   if (value ==
null ||
value.isEmpty) {
                    return 'Email is required';
                  } else if
(!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(
value)) {
                    return 'Enter a valid email
address';
                  }
return null;
                },
              ),
              SizedBox(height: 16),
_buildTextField(                    label:
'Password',                   hint: 'Enter
your password',                    icon:
Icons.lock,                   isPassword:
true,                 validator: (value) {
if (value == null ||
value.isEmpty) {
                    return 'Password is required';
} else if (value.length < 6) {
return 'Password must be at least 6 characters';
                  }
return null;
},
                ),
              ],
            ),
          ),
          SizedBox(height: 32),
          // Login Button
ElevatedButton(
onPressed: () {
if
(_formKey.currentState!.validate()) {

Navigator.pushReplacementNamed(context,
'/home');
              }
            },
```

```
                style: ElevatedButton.styleFrom(
backgroundColor: Colors.green,
minimumSize: Size(double.infinity,
50),                                    shape:
RoundedRectangleBorder(
borderRadius:
BorderRadius.circular(8),
                ),              ),              child:
Text(               'Login',              style:
TextStyle(fontSize: 18, color:
Colors.white),
                ),
              ),
              SizedBox(height: 16),
              // Registration link
Center(               child:
TextButton(
onPressed: () {
                  Navigator.pushNamed(context,
'/register'); // Add registration route later
                },
child: Text(
                  'Don't have an account? Register',
style: TextStyle(color:
Colors.green), // Green color applied
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
 }

  Widget _buildTextField({     required String
label,     required String hint,     required
IconData icon,     bool isPassword = false,
required String? Function(String?) validator,
  }) {     return TextFormField(       obscureText:
isPassword,       decoration: InputDecoration(
labelText: label,         hintText: hint,
prefixIcon: Icon(icon, color: Colors.green),
border: OutlineInputBorder(
borderRadius: BorderRadius.circular(8),
      ),
      focusedBorder: OutlineInputBorder(
borderSide: BorderSide(color: Colors.green), //
Green color applied          borderRadius:
BorderRadius.circular(8),
      ),
    ),
    validator: validator,
  );
 }
}
```

add_video.dart

```
import 'package:flutter/material.dart'; import
'package:http/http.dart' as http; import
'dart:convert';

class PillMate extends StatefulWidget {
@override
  _PillMateState createState() =>
_PillMateState();
}
```

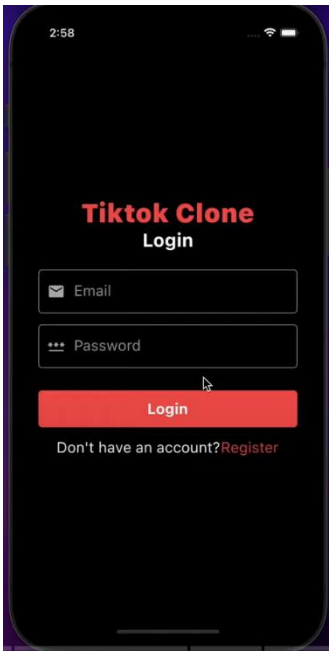
```
class _PillMateState extends State<PillMate> {
final _formKey = GlobalKey<FormState>();
final TextEditingController _cityController =
TextEditingController();

  String? _city;
  String? _address;
  String? _updatedAt;
  String? _status;
  String? _temp;
  String? _tempMin;
  String? _tempMax;
  String? _windSpeed;
  String? _pressure;
  String? _humidity;
  String? _sunrise;   String? _sunset;   bool
_isLoading = false;   bool _isError = false;
bool _isDataFetched = false; // Add this flag to
control the visibility of weather details

  Future<void> _fetchWeather() async {
setState(() {       _isLoading = true;
      _isError = false;
      _isDataFetched = false; // Reset this flag
when fetching new data
    });

    final response = await http.get(Uri.parse(

'https://api.openweathermap.org/data/2.5/weathe
r?q=$_city&units=metric&appid=73cbebdd032
2 acd49bda6ede059b2b18'));

    if (response.statusCode == 200) {
final data = jsonDecode(response.body);

      setState(() {         _address =
'${data['name']},
${data['sys']['country']}';
_updatedAt = 'Updated At:
${DateTime.fromMillisecondsSinceEpoch(data[
' dt'] * 1000).toString()}';         _status =
data['weather'][0]['description'].toUpperCase();
_temp = '${data['main']['temp']}°C';
_tempMin = 'Min Temp:
${data['main']['temp_min']}°C';         _tempMax
= 'Max Temp: ${data['main']['temp_max']}°C';
_pressure = 'Pressure:
${data['main']['pressure']} hPa';         _humidity
= 'Humidity:
${data['main']['humidity']}%';
_windSpeed = 'Wind Speed:
${data['wind']['speed']} m/s';
        _sunrise =
DateTime.fromMillisecondsSinceEpoch(data['sy
s']['sunrise'] * 1000).toString();
        _sunset =
DateTime.fromMillisecondsSinceEpoch(data['sy
s']['sunset'] * 1000).toString();
        _isLoading = false;
        _isDataFetched = true; // Set the flag to
true after data is fetched
      });     } else {
setState(() {
_isLoading = false;
        _isError = true;
        _isDataFetched = false; // Reset the flag if
there is an error
      });
    }
  }

  @override
  Widget build(BuildContext context) {
return Scaffold(       appBar: AppBar(
title: Text('Weather Forecasting'),
//backgroundColor: Colors.white,
      ),
      //backgroundColor: Colors.white,
body: SafeArea(         child:
SingleChildScrollView(           child:
Padding(             padding: const
EdgeInsets.all(16.0),             child:
Column(               crossAxisAlignment:
CrossAxisAlignment.stretch,
              children: [
SizedBox(height: 50),
                // Logo                 Center(
child: Image.asset(
'assets/images/logo.png', // Replace with your
logo
                  height: 150,
                ),
```

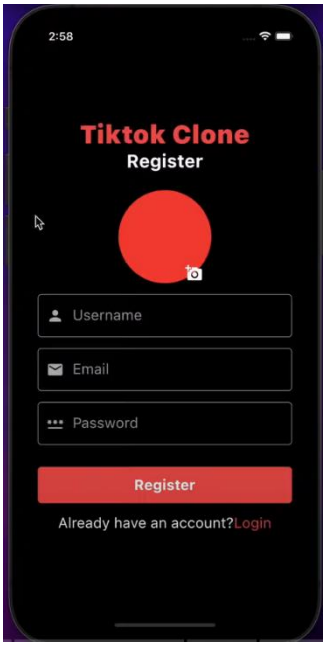
```
            ),
            SizedBox(height: 16),
            // Title
            Text(
              'Weather Forecasting',
textAlign: TextAlign.center,
style: TextStyle(                fontSize:
18,                color: Colors.black,
              ),
            ),
            SizedBox(height: 32),
            // Form
Form(                key:
_formKey,                child:
Column(
children: [
TextFormField(
controller: _cityController,
decoration: InputDecoration(
labelText: 'City',
hintText: 'Enter the city
name',
prefixIcon:
Icon(Icons.location_city, color: Colors.green),
border: OutlineInputBorder(
borderRadius:
BorderRadius.circular(8),
                    ),
focusedBorder:
OutlineInputBorder(
                      borderSide: BorderSide(color:
Colors.green),
                      borderRadius:
BorderRadius.circular(8),
                    ),
), validator:
(value) {                  if
(value == null ||
value.isEmpty) {
                    return 'City is required';
                  }
return null;
                },
              ),
              SizedBox(height:
16),                ElevatedButton(
onPressed: () {                  if
(_formKey.currentState!.validate()) {
                    setState(() {
                      _city = _cityController.text;
                    });
                    _fetchWeather();
                  }
                },
                style: ElevatedButton.styleFrom(
backgroundColor: Colors.green,
minimumSize: Size(double.infinity, 50),
shape:
RoundedRectangleBorder(
                    borderRadius:
BorderRadius.circular(8),
                  ),
                ),
```

```
                    child: Text(
'Get Weather',                        style:
TextStyle(fontSize: 18, color: Colors.white),
                    ),
                  ),
                ],
              ),
            ),
            SizedBox(height: 32),
            _isLoading
                ? CircularProgressIndicator()
                : _isError
                ? Text(
              'Error fetching weather data.',
style: TextStyle(color: Colors.red),
textAlign: TextAlign.center,
            )
                : _isDataFetched
                ? Column(
crossAxisAlignment:
CrossAxisAlignment.start,
              children: [
                Text(
                  'Location: $_address',
style: TextStyle(fontSize: 16, fontWeight:
FontWeight.bold),
                ),
                SizedBox(height: 8),
                Text('$_updatedAt'),
                SizedBox(height: 16),
                Text('Status: $_status'),
                SizedBox(height: 8),
                Text('Temperature: $_temp'),
                SizedBox(height: 8),
Text('$_tempMin'),
                SizedBox(height: 8),
Text('$_tempMax'),
                SizedBox(height: 8),
                Text('$_windSpeed'),
                SizedBox(height: 8),
Text('$_pressure'),
                SizedBox(height: 8),
Text('$_humidity'),
                SizedBox(height: 8),
                Text('Sunrise: $_sunrise'),
                SizedBox(height: 8),
                Text('Sunset: $_sunset'),
              ],
            )
                : Container(),
          ],
        ),
      ),
    ),
  );
 }
}
```

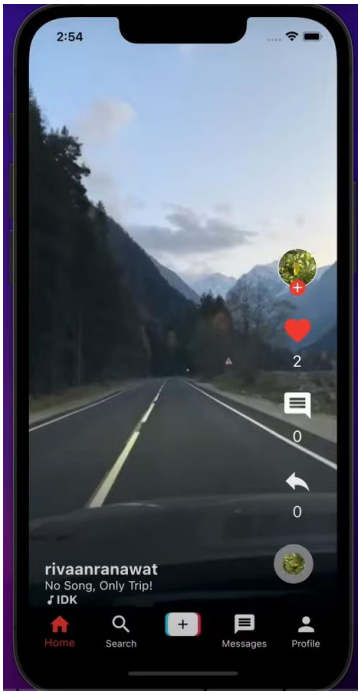
OUTPUT: -

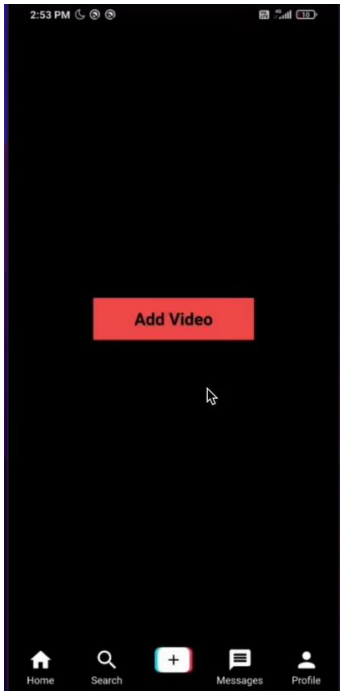
After clicking on Already have an account? it navigates to the registration page.



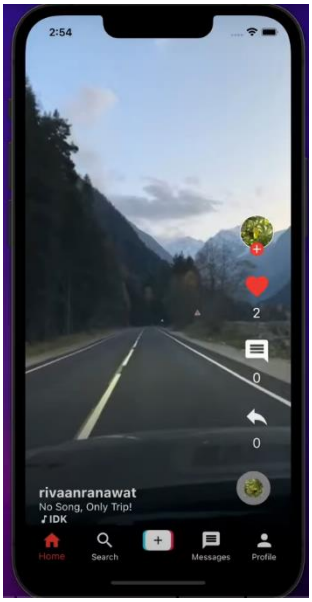


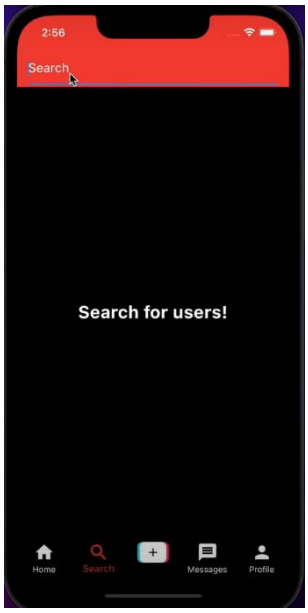
In home page, after clicking on "+" icon it navigates to the add video page.



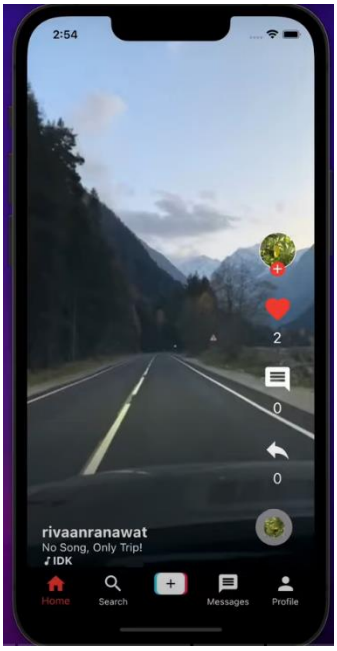


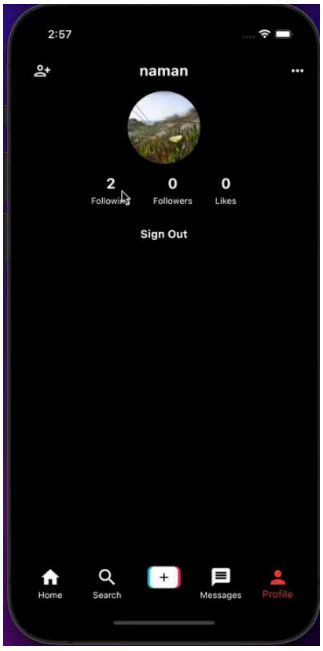
In home page, after clicking on "🔍" icon it navigates to the search video page.





In home page, after clicking on "👩‍♂️" icon it navigates to the profile page.





Project Title: **Roll No.**

MAD & PWA Lab

Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	17
Name	Omkar Gholap
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:	

MPL Experiment 6

Name: Omkar Gholap **Class:** D15A **Roll no:**36

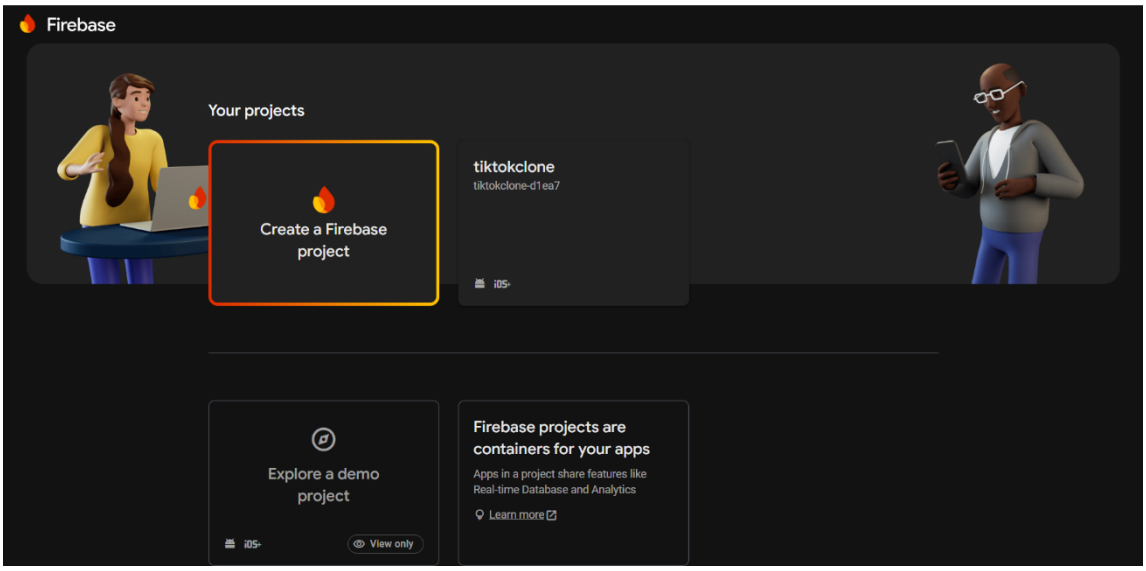
Aim: How To Set Up Firebase with Flutter for iOS and Android Apps

Theory: -

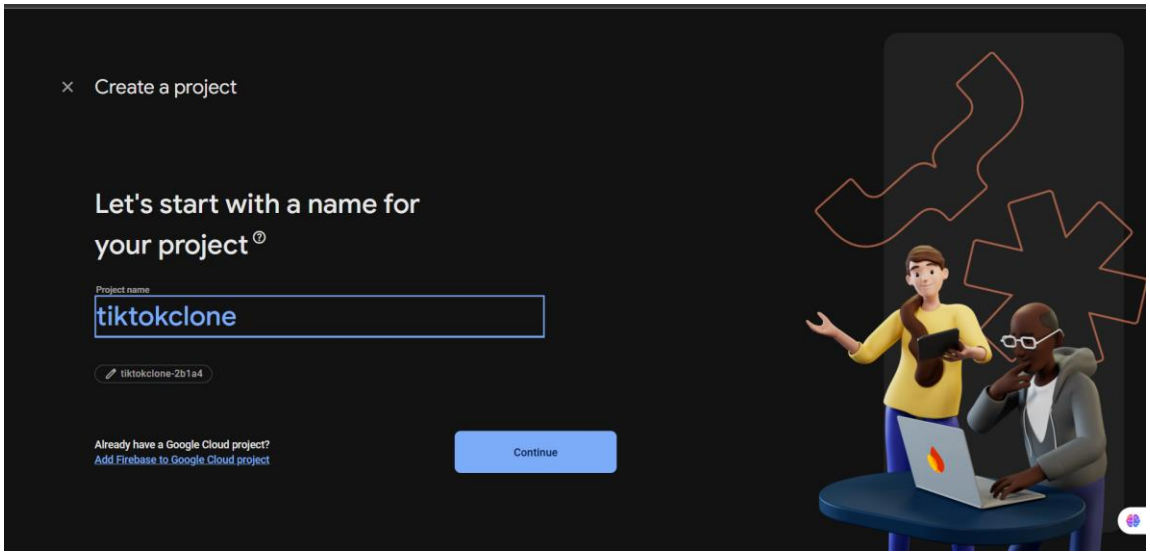
Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. Firebase, a Backend-as-a-Service (BaaS) platform, provides real-time database, authentication, and cloud storage services, making it a powerful backend solution for Flutter applications.
By integrating Firebase with Flutter, developers can store and retrieve data in real time, authenticate users, and manage cloud-based data efficiently. This is particularly useful for applications requiring dynamic content updates and user interactions.

Steps to Connect Flutter UI with Firebase Database:

Step 1:
1.1) Go to Firebase Console and Create a Firebase Project

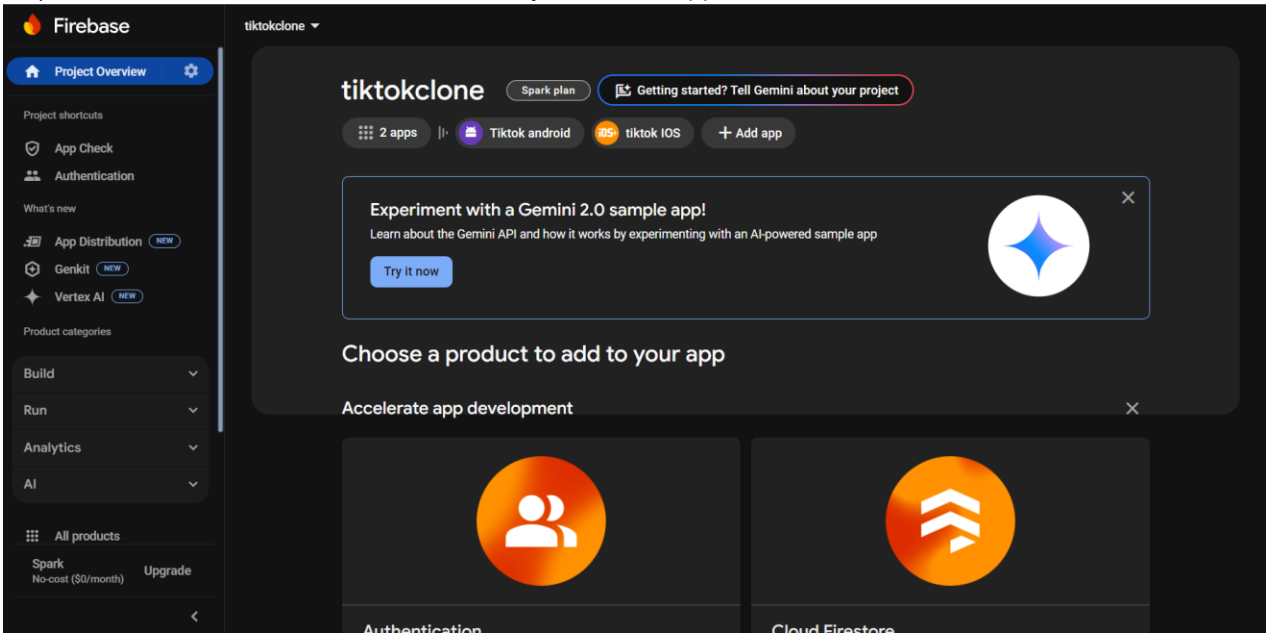


1.2) Click on Create a Project and give it a suitable name.



Step 2:- Add Firebase to Your Flutter App

2.1) Click on Android/iOS/Web based on your Flutter application

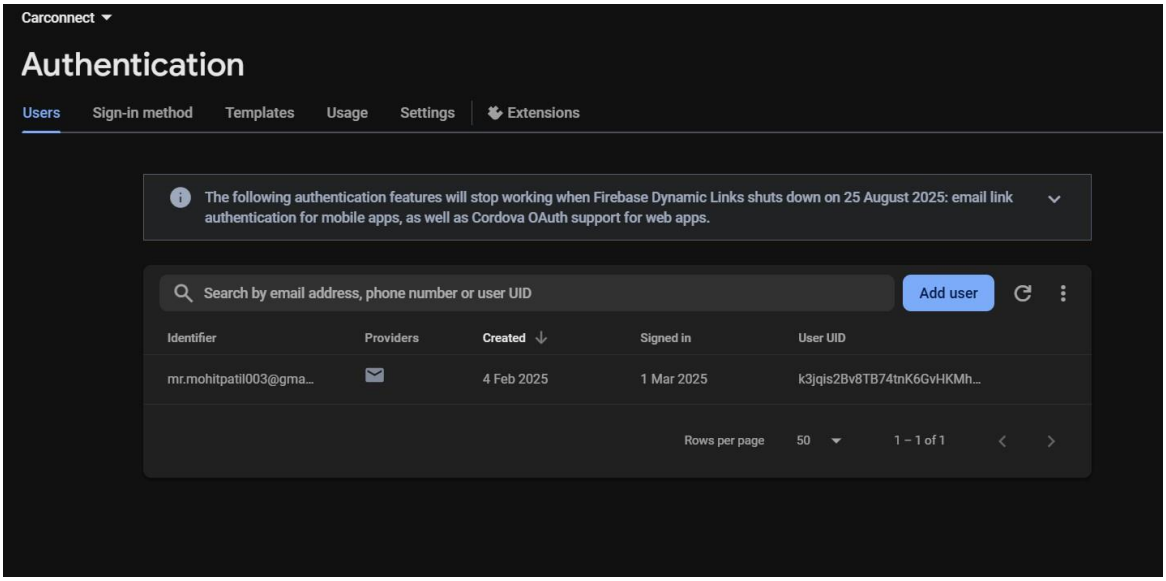


Step 3: - Add Firebase Authentication to Your App

3.1) Add Firebase Authentication Dependencies

```yaml
dependencies:
  flutter:
    sdk: flutter
  firebase_core: ^3.11.0
  firebase_auth: ^5.4.2  # For authentication
  cloud_firestore: ^5.6.3  # For Firestore, if you need it
  firebase_messaging: ^15.2.2
  http: ^0.13.3
  image_picker: ^1.0.4
  tflite_flutter: ^0.11.0
  image: ^3.2.0
  url_launcher: ^6.1.14
```

3.2) Enable Authentication in Firebase Console Go to Firebase Console → Authentication. Click on Sign-in method and enable Email/Password (or any other method like Google). Click Save



3.3)

Implement Authentication in Flutter Modify main.dart

import 'package:firebase_core/firebase_core.dart'; import 'package:firebase_auth/firebase_auth.dart';

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

Step 4: -Configure Firebase Realtime Database

1. Go to Firebase Console → Realtime Database.
2. Click Create Database → Choose location → Set rules (for development, set read/write to true).
3. Click Publish.

CODE:

Login page.dart

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

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

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

class _LoginPageState extends State<LoginPage> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  final FirebaseAuth _auth = FirebaseAuth.instance;

  // Use the renderButton method for Google Sign-In on web
```

```
void _signInWithGoogle() async {
  final GoogleSignIn googleSignIn = GoogleSignIn();

  try {
    GoogleSignInAccount? googleUser = await googleSignIn.signIn();
    if (googleUser == null) return;

    final GoogleSignInAuthentication googleAuth =
        await googleUser.authentication;

    final AuthCredential credential = GoogleAuthProvider.credential(
      accessToken: googleAuth.accessToken,
      idToken: googleAuth.idToken,
    );

    await _auth.signInWithCredential(credential);
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Google Sign-In Successful!')),
    );
  } catch (e) {
    print("Google Sign-In Error: $e");
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Google Sign-In Failed: $e')),
    );
  }
}

void _signInWithEmailAndPassword() async {
  if (_formKey.currentState!.validate()) {
    try {
      await _auth.signInWithEmailAndPassword(
        email: _emailController.text.trim(),
        password: _passwordController.text.trim(),
      );
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Login Successful!')),
      );

      // Redirect to Home Page after login
      Navigator.pushReplacementNamed(context, '/home');
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Login Failed: $e')),
```

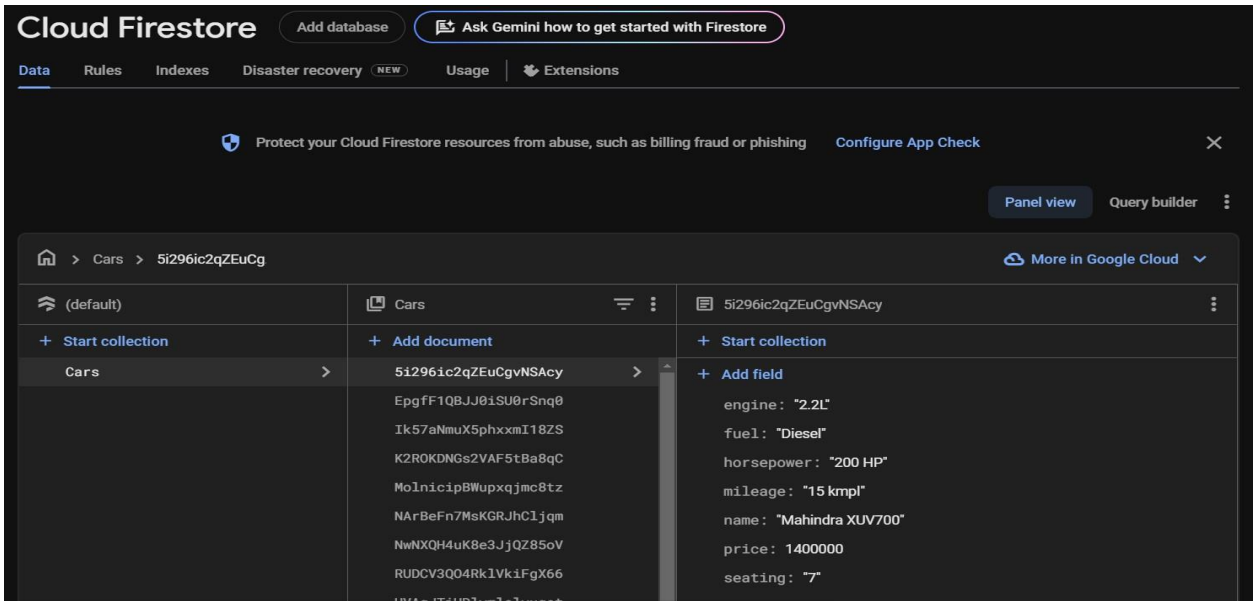
```
      );
    }
  }
}

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

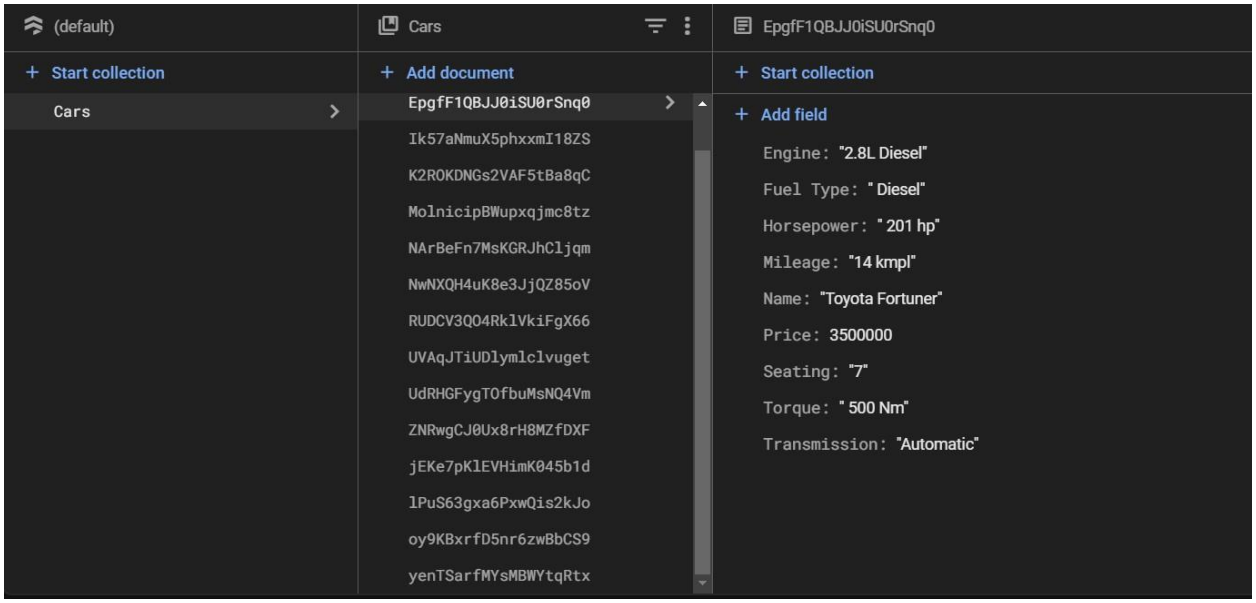
```
    body: Container(
      color: Colors.black,
      child: Center(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Card(
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(20),
            ),
            elevation: 10,
            color: Colors.white,
            child: Padding(
              padding: const EdgeInsets.all(30.0),
              child: Form(
                key: _formKey,
                child: Column(
                  mainAxisSize: MainAxisSize.min,
                  crossAxisAlignment: CrossAxisAlignment.stretch,
                  children: [
                    Column(
                      children: [
                        Image.asset(
                          'assets/c1.png', // Ensure image is in assets folder
                          height: 150,
                        ),
                        const SizedBox(height: 10),
                        Text(
                          'CarConnect',
                          style: GoogleFonts.alegreyaSans(
                            fontSize: 40,
                            fontWeight: FontWeight.bold,
                            color: Colors.black,
                          ),
                        ),
                      ],
```

```
),
const SizedBox(height: 20),
TextFormField(
  controller: _emailController,
  decoration: InputDecoration(
    labelText: 'Email',
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
    ),
    prefixIcon:
        const Icon(Icons.email, color: Colors.black),
  ),
```

```
  validator: (value) {
    if (value == null ||
        value.isEmpty ||
        !value.contains('@')) {
      return 'Enter a valid email';
    }
    return null;
  },
),
const SizedBox(height: 15),
TextFormField(
  controller: _passwordController,
  obscureText: true,
  decoration: InputDecoration(
    labelText: 'Password',
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
    ),
    prefixIcon:
        const Icon(Icons.lock, color: Colors.black),
  ),
  validator: (value) {
    if (value == null || value.length < 6) {
      return 'Password must be at least 6 characters';
    }
    return null;
  },
),
const SizedBox(height: 20),
ElevatedButton(
  style: ElevatedButton.styleFrom(
```

```
                      padding: const EdgeInsets.symmetric(vertical: 14),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(12),
                      ),
                      backgroundColor: Colors.black,
                    ),
                    onPressed: _signInWithEmailAndPassword,
                    child: const Text(
                      'Login',
                      style: TextStyle(fontSize: 20, color: Colors.white),
                    ),
                  ),
                  const SizedBox(height: 15),
                  ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.white,
                      padding: const EdgeInsets.symmetric(vertical: 14),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(12),
                        side: const BorderSide(color: Colors.black),
                      ),
                    ),
                    onPressed: _signInWithGoogle,
                    child: const Text(
                      'Sign in with Google',
                      style: TextStyle(fontSize: 20, color: Colors.black),
                    ),
                  ),
                ],
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```


Cloud Firestore
Add database
Ask Gemini how to get started with Firestore
Data
Rules
Indexes
Disaster recovery
NEW
Usage
Extensions
Protect your Cloud Firestore resources from abuse, such as billing fraud or phishing
Configure App Check
Panel view
Query builder
Cars > 5i296ic2qZEuCg
More in Google Cloud
(default)
Cars
5i296ic2qZEuCgvNSAcy
Start collection
Add document
Start collection
Cars
5i296ic2qZEuCgvNSAcy
Add field
EpgfF1QBJJ0iSU0rSnq0
Ik57aNmuX5phxxmI18ZS
K2ROKDNGs2VAF5tBa8qC
MolnicipBWupxqjmc8tz
NArBeFn7MsKGRJhCljqm
NwNXQH4uK8e3JjQZ85oV
RUDCV3QO4RklVkiFgX66
engine: "2.2L"
fuel: "Diesel"
horsepower: "200 HP"
mileage: "15 kmpl"
name: "Mahindra XUV700"
price: 1400000
seating: "7"


(default)
Cars
EpgfF1QBJJ0iSU0rSnq0
Start collection
Add document
Start collection
Cars
EpgfF1QBJJ0iSU0rSnq0
Add field
Ik57aNmuX5phxxmI18ZS
K2ROKDNGs2VAF5tBa8qC
MolnicipBWupxqjmc8tz
NArBeFn7MsKGRJhCljqm
NwNXQH4uK8e3JjQZ85oV
RUDCV3QO4RklVkiFgX66
UVAqJTiUDlymlclvuget
UdRHGFygTOfbuMsNQ4Vm
ZNRwgCJ0Ux8rH8MZfDXF
jEKe7pKlEVHimK045b1d
lPuS63gxa6PxwQis2kJo
oy9KBxrfD5nr6zwBbCS9
yenTSarfMYsMBWYtqRtx
Engine: "2.8L Diesel"
Fuel Type: " Diesel"
Horsepower: " 201 hp"
Mileage: "14 kmpl"
Name: "Toyota Fortuner"
Price: 3500000
Seating: "7"
Torque: " 500 Nm"
Transmission: "Automatic"

Project Title: **Roll No.**

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.	17
Name	Omkar Gholap
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

EXPERIMENT 7

Aim-To write meta data of your Ecommerce PWA in a Web app manifest file to enable add to home screen 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: -

//Manifest.json:

```
{
```

```
"short_name": "Movie Rec",
"name": "Movie Recommendation
App", "icons": [
 {
  "src": "favicon.ico",
  "sizes": "64x64 32x32 24x24 16x16",
  "type": "image/x-icon"
 },
 {
  "src": "logo192.png",
  "type": "image/png",
  "sizes": "192x192",
  "purpose": "any
  maskable"
 },
 {
  "src": "logo512.png",
  "type":
  "image/png",
  "sizes": "512x512"
 }
], "start_url":
".", "display":
"standalone",
"theme_color": "#000000",
"background_color": "#ffffff",
```

```
 “description”: “Find your next favorite
 movie”, “orientation”:
 “portrait-primary”,
 “categories”: [“entertainment”, “movies”]
}
```

//Serviceworker.js

```
// Import workbox from CDN (you can also use the
workbox-webpack-plugin)
importScripts('https://storage.google
apis.com/workbox-
cdn/releases/6.4.1/workbox-sw.js');
// Precache and route setup
workbox.routing.registerRoute(
 ({request}) =>
 request.destination === 'image', new
 workbox.strategies.CacheFirst({ cacheName:
 'images',
  plugins: [
   new
    workbox.expiration.ExpirationPlugi
    n({ maxEntries: 60,
    maxAgeSeconds: 30 * 24 * 60 * 60, // 30 Days
   }),
  ],
 })
);

// Cache CSS and JavaScript
Files
workbox.routing.registerRoute(
 ({request}) => request.destination === 'script' ||
```

```
 request.destination === 'style', new
 workbox.strategies.StaleWhileRevalidate({ cacheName:
 'static-resources',

 })

);
```

//index.html

```
<!DOCTYPE html>

<html lang="en">

 <head>

  <meta charset="utf-8" />

  <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />

  <meta name="viewport" content="width=device-width,
  initial-scale=1" />

  <meta name="theme-color" content="#000000" />

  <meta
   name="descripti
   on"
   content="Movie recommendations app - find your next
   favorite film"

  />

  <link rel="apple-touch-icon"
  href="%PUBLIC_URL%/logo192.png" />

  <link rel="manifest"
  href="%PUBLIC_URL%/manifest.json" />

  <title>Movie Recommendations</title>

 </head>

 <body>

  <noscript>You need to enable JavaScript to run this
  app.</noscript>
  <div id="root"></div>
```

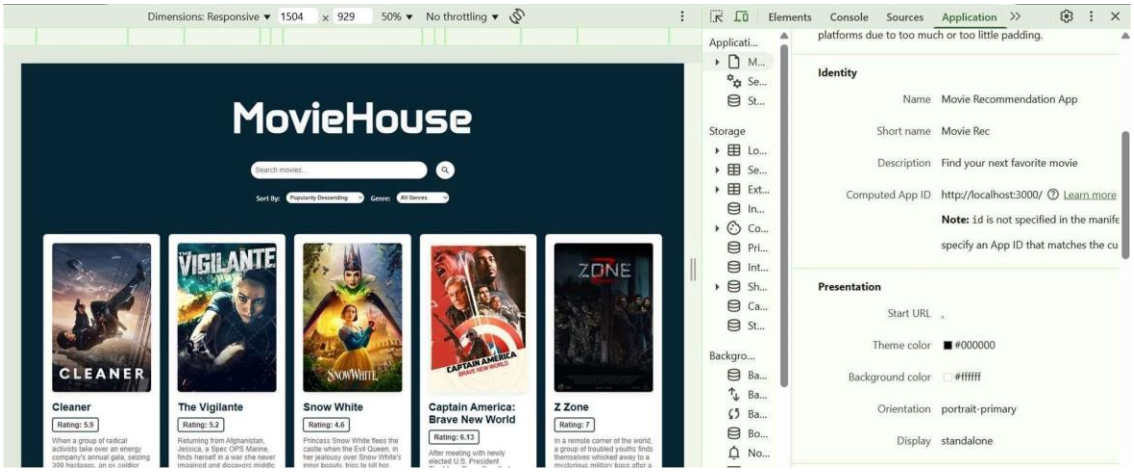
```
 </body>
</html>
```

Output

//Open folder in VS Code and run the website

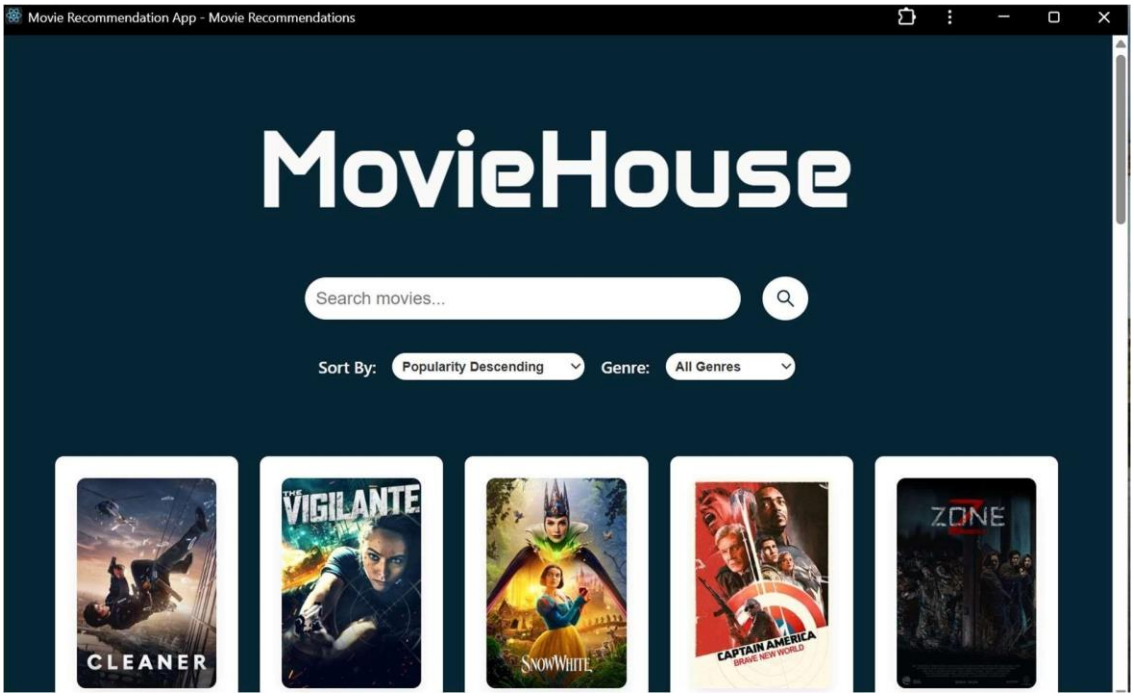
```
HOME@LAPTOP-9JIMM8I3 MINGW64 /e/New folder/Movie-Recommendation-App (main)
$ npm start

Compiled successfully!

You can now view movie-recommendation-app in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.162.1:3000
```

//open developers tools -> Applications



// Install the app by following this route Click on three dots on top right corner of the browser from app option ->Cast,share,save->Install the app

// This is the app



Project Title: **Roll No.**

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.	17
Name	Omkar Gholap
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?

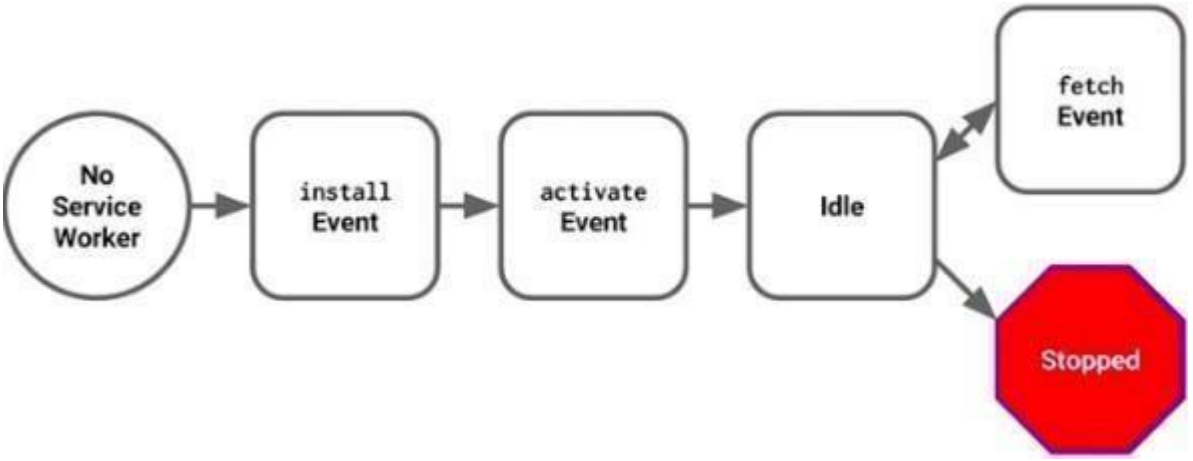
You can't access the **Window**

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:

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-

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta name="description" content="Movie recommendations app - find your next favorite film" />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>Movie Recommendations</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

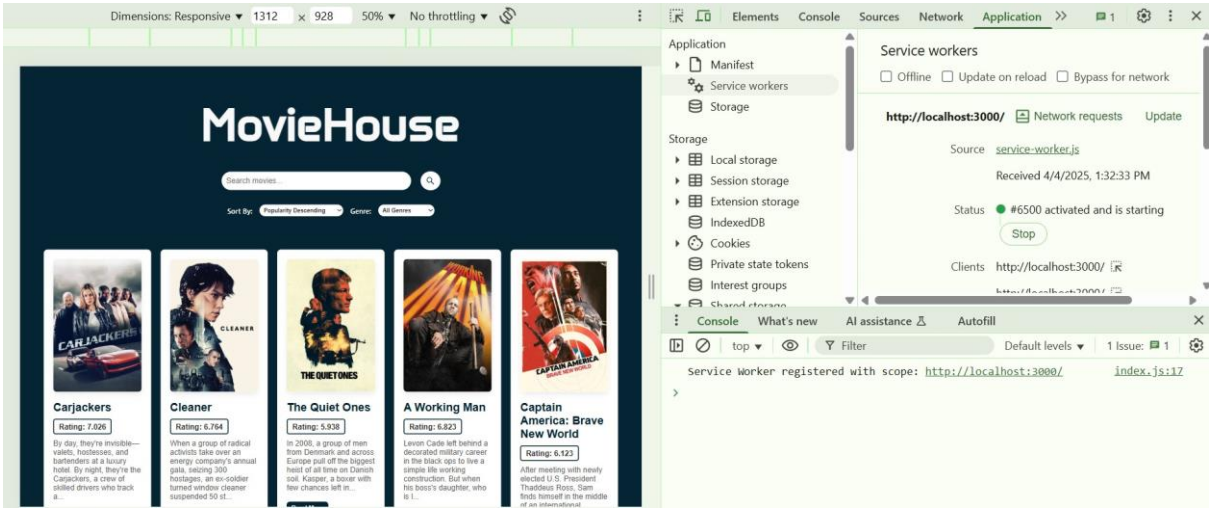
service-worker.js

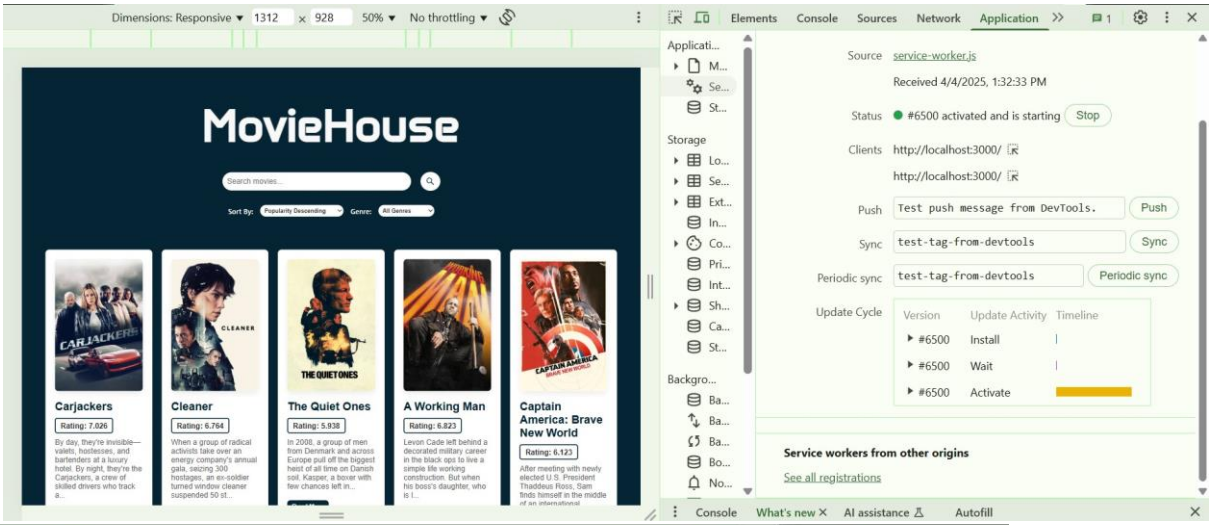
```
importScripts('https://storage.googleapis.com/workbox-cdn/releases/6.4.1/workbox-sw.js');

if (workbox) {
  console.log('□ Workbox loaded');
```

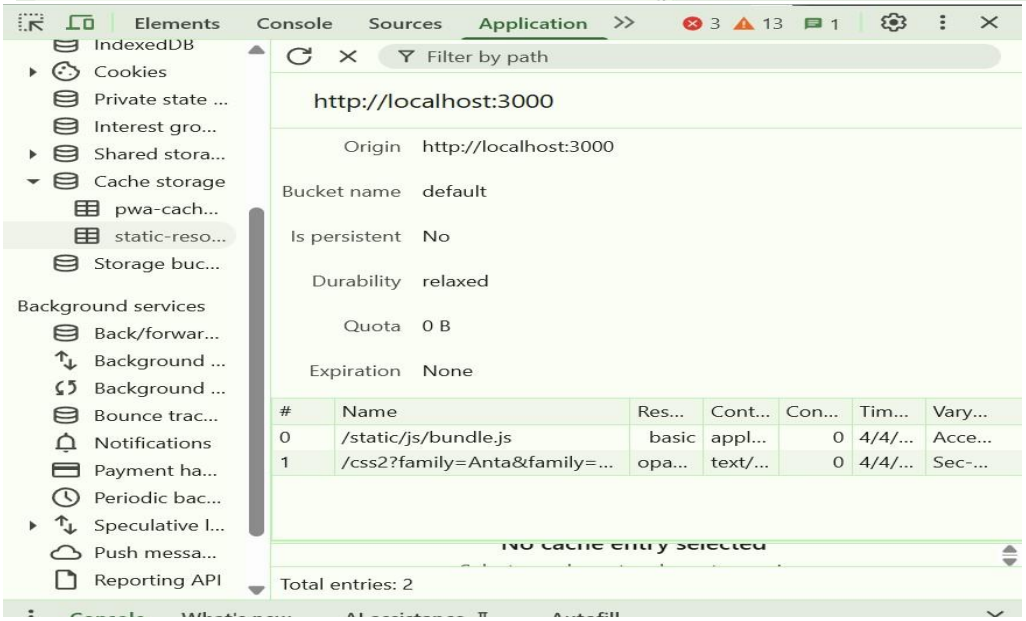
```
const CACHE_NAME = 'pwa-cache-v1';
const PRECACHE_ASSETS = [
  '/',
  '/index.html',
  '/favicon.ico',
  '/logo192.png',
  '/logo512.png',
  '/manifest.json',
  '/static/css/main.css',
  '/static/js/main.js'
];

// Precache assets manually
workbox.precaching.precacheAndRoute(PRECACHE_ASSETS);

// Cache images
workbox.routing.registerRoute(
  ({ request }) => request.destination === 'image',
new workbox.strategies.CacheFirst({
cacheName: 'images-cache', plugins: [ new
workbox.expiration.ExpirationPlugin({
maxEntries: 60, maxAgeSeconds: 30 * 24 * 60
* 60, // 30 Days
    }),
  ],
  })
);

// Cache CSS & JS
workbox.routing.registerRoute(
  ({ request }) => request.destination === 'script' || request.destination ===
'style',
  new workbox.strategies.StaleWhileRevalidate({
cacheName: 'static-resources',
  })
);

// Debug Fetch Requests self.addEventListener('fetch',
(event) => { console.log('[Service Worker] Fetching:',
event.request.url); event.respondWith(
caches.match(event.request).then((cachedResponse) => {
return cachedResponse || fetch(event.request);
  })
  );
```

```
  });

  // Install Event   self.addEventListener('install',
(event) => {     console.log('[Service Worker]
Installing...');     event.waitUntil(
caches.open(CACHE_NAME).then((cache) => {
console.log('[Service Worker] Caching assets');
return cache.addAll(PRECACHE_ASSETS);
      })
    );
    self.skipWaiting();
  });

  // Activate Event   self.addEventListener('activate', (event) => {
console.log('[Service Worker] Activating...');     event.waitUntil(
caches.keys().then((cacheNames) => {         return Promise.all(
cacheNames.map((cache) => {             if (cache !==
CACHE_NAME) {               console.log('[Service Worker]
Deleting old cache:', cache);               return caches.delete(cache);
            }
          })
        );
      })
    );
    self.clients.claim();
  });

} else {
  console.log(' Workbox failed to load');
}
```

OUTPUT


MovieHouse
Carjackers
Cleaner
The Quiet Ones
A Working Man
Captain America: Brave New World
Service workers
Offline
Update on reload
Bypass for network
http://localhost:3000/
Network requests
Update
Source service-worker.js
Received 4/4/2025, 1:32:33 PM
Status #6500 activated and is starting
Stop
Clients http://localhost:3000/
Service Worker registered with scope: http://localhost:3000/ index.js:17


MovieHouse
Source service-worker.js
Received 4/4/2025, 1:32:33 PM
Status #6500 activated and is starting
Stop
Clients http://localhost:3000/
http://localhost:3000/
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
#6500 Install
#6500 Wait
#6500 Activate
Service workers from other origins
See all registrations


Elements
Console
Sources
Application
Filter by path
http://localhost:3000
Origin http://localhost:3000
Bucket name default
Is persistent No
Durability relaxed
Quota 0 B
Expiration None
Name Res... Cont... Con... Tim... Vary...
0 /static/js/bundle.js basic appl... 0 4/4/... Acce...
1 /css2?family=Anta&family=... opa... text/... 0 4/4/... Sec-...
Total entries: 2

Project Title: **Roll No.**

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.	17
Name	Omkar Gholap
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

EXPERIMENT 9

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

Theory:

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

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

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

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

- **CacheFirst** - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.
- **NetworkFirst** - In this function, firstly we can try getting an updated response from the network, if this process completed successfully, the new

response will be cached and returned. But if this process fails, we check whether the request has been cached before or not. If a cache exists, it is returned to the page, but if not, this is up to you. You can return dummy content or information messages to the page.

Sync Event

Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are writing e-mail content and we didn't realize it. When completing the writing, we click the send button. Here is a job for the Background Sync.
The following view shows the classical process of sending email to us. If the Internet Connection is broken, we can't send any content to Mail Server.

Push Event

This is the event that handles push notifications that are received from the server.You can apply any method with received data.
We can check in the following example.
"Notification.requestPermission();" is the necessary line to show notification to the user. If you don't want to show any notification, you don't need this line. In the following code block is in sw.js file. You can handle push notifications with this event. In this example, I kept it simple. We send an object that has "method" and "message" properties. If the method value is "pushMessage", we open the information notification with the "message" property.

CODE-

Service-worker.js //

Import Workbox

```
importScripts('https://storage.googleapis.com/workbox-cdn/releases/6.4.1/workbox-sw.js');


if (workbox) {   console.log(' Workbox Loaded

Successfully');
```

```
  // Precache assets
workbox.precaching.precacheAndRoute(self.__WB_MANIFEST || []);


  //  Cache Images (Cache-First Strategy)
workbox.routing.registerRoute(
   ({ request }) => request.destination === 'image',
new workbox.strategies.CacheFirst({
cacheName: 'images-cache',       plugins: [         new
workbox.expiration.ExpirationPlugin({
maxEntries: 60,           maxAgeSeconds: 30 * 24 * 60
* 60, // 30 Days
       }),
     ],
   })
  );


  //  Cache CSS & JS (Stale-While-Revalidate Strategy)
workbox.routing.registerRoute(
   ({ request }) => request.destination === 'script' || request.destination ===
'style',
   new workbox.strategies.StaleWhileRevalidate({
cacheName: 'static-resources',
   })
  );
```

```
  // Cache API Responses (Network-First Strategy)
workbox.routing.registerRoute(
   ({ url }) => url.origin.includes('api.themoviedb.org'),
new workbox.strategies.NetworkFirst({
cacheName: 'api-cache',       plugins: [         new
workbox.expiration.ExpirationPlugin({
maxEntries: 50,           maxAgeSeconds: 5 * 60, // 5
minutes
       }),
     ],
   })
  );

}

//  Fetch Event Logging (Works outside Workbox)
self.addEventListener('fetch', (event) => {     console.log(`
Fetch event detected: ${event.request.url}`);

   if (event.request.url.includes('api.themoviedb.org')) {
console.log(`Intercepting API Request: ${event.request.url}`);

     event.respondWith(
caches.match(event.request).then((cachedResponse) => {          return
cachedResponse || fetch(event.request).then((response) => {
console.log(` API Response Fetched: ${event.request.url}`);
```

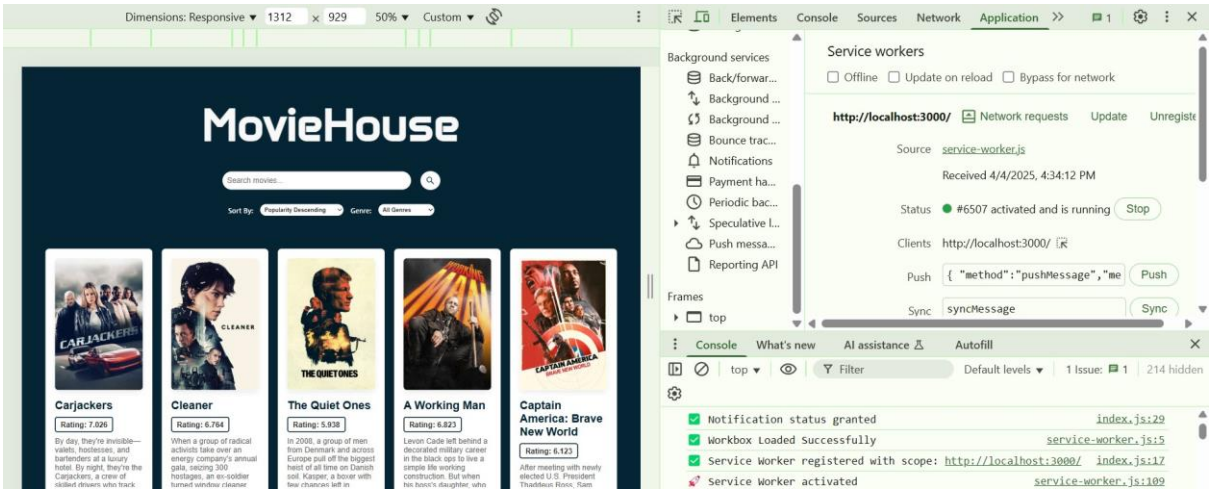
```
return response;          }).catch((err) => {          console.error(` API
Fetch Failed: ${event.request.url}`, err);          return new
Response('API fetch failed', { status: 500 });
        });
      })
    );
  }
 });

// Background Sync Event (Syncing Watchlist)
self.addEventListener('sync', (event) => {  if
(event.tag === 'sync-watchlist') {    console.log('
Sync event triggered: sync-watchlist');
event.waitUntil(      syncWatchlist().then(() => {
console.log(' Sync successful');
    }).catch((err) => {
console.error(' Sync failed:', err);
    })
  );
 }
});

// Example Function to Sync Watchlist async
function syncWatchlist() {  console.log('
Syncing watchlist data...');  return
fetch('/sync-watchlist', { method: 'POST' })
```

```
    .then(() => console.log(' Sync request sent successfully!'))
    .catch(() => console.log(' Sync request failed, retrying later.'));
}

//  Push Notification Event self.addEventListener('push',
(event) => {    console.log('Push notification received');

  const notificationData = event.data ? event.data.text() ;   console.log(`
Push payload: ${notificationData}`);

  const options = {
body: notificationData,
icon: '/logo192.png',
badge: '/logo192.png',
  };

  event.waitUntil(     self.registration.showNotification('
Movie House', options)
      .then(() => console.log(' Push notification sent successfully'))
      .catch((err) => console.error(' Push notification failed:', err))
  );
});

//  Activate event - Cleanup old caches self.addEventListener('activate',
(event) => {    console.log(' Service Worker activated');   event.waitUntil(
```

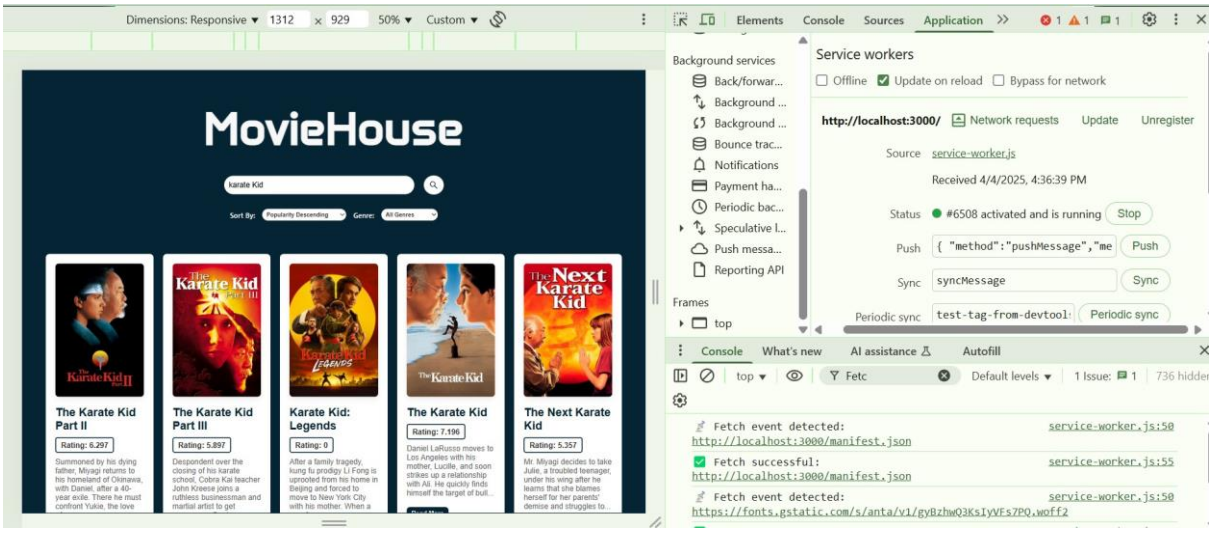
```
caches.keys().then((cacheNames) =>       Promise.all(
cacheNames.map((cache) => {           if (!['images-cache', 'static-
resources', 'api-cache'].includes(cache)) {           console.log('🗑 Deleting
old cache:', cache);           return caches.delete(cache);
        }
      })
    )
  )
 );
 self.clients.claim();
});
```

OUTPUT

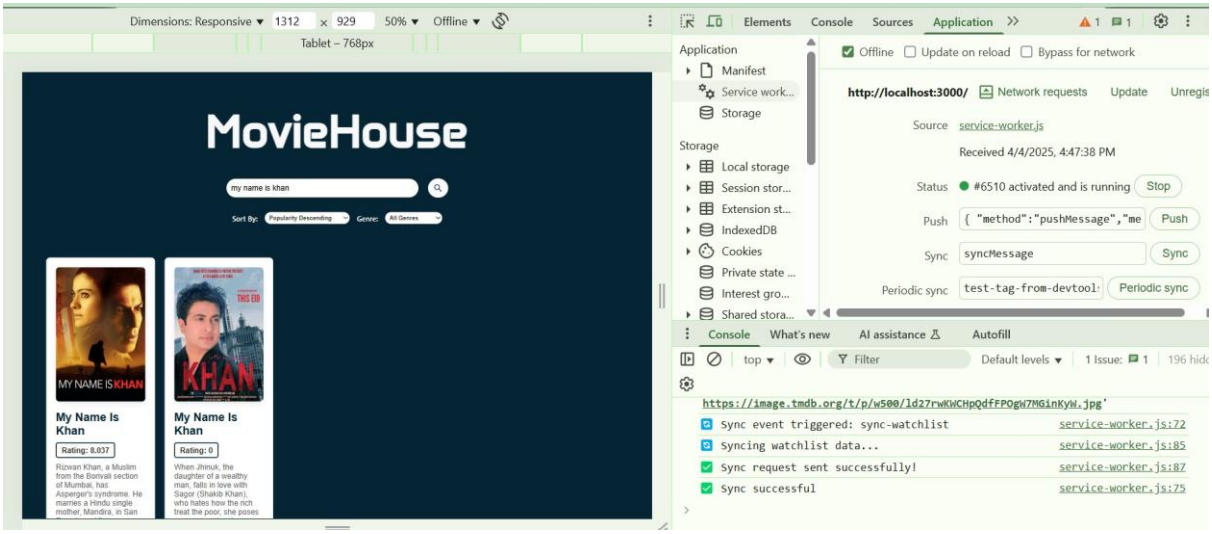
Service worker activated



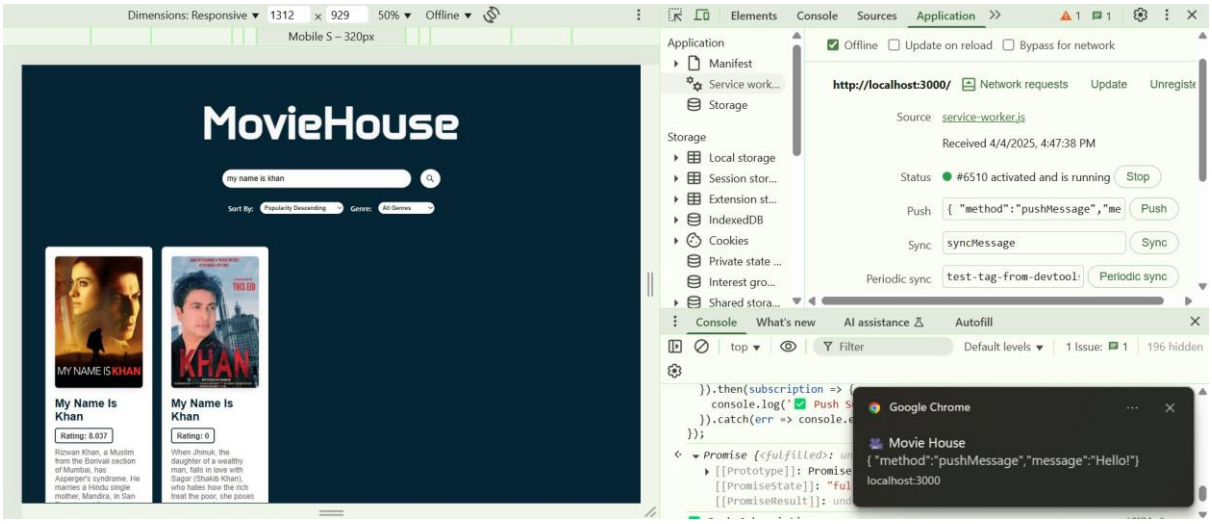
Fetch Event



Sync Event


Dimensions: Responsive 1312 x 929 50% Offline
Tablet – 768px
MovieHouse
my name is khan
Sort By: Popularity Descending Genre: All Genres
My Name Is Khan
Rating: 8.037
My Name Is Khan
Rating: 0
Elements Console Sources Application
Application
Manifest
Service work...
Storage
Storage
Local storage
Session stor...
Extension st...
IndexedDB
Cookies
Private state ...
Interest gro...
Shared stora...
Offline Update on reload Bypass for network
http://localhost:3000/ Network requests Update
Source service-worker.js
Received 4/4/2025, 4:47:38 PM
Status #6510 activated and is running Stop
Push { "method":"pushMessage","me Push
Sync syncMessage Sync
Periodic sync test-tag-from-devtool Periodic sync
Console What's new AI assistance Autofill
Filter Default levels 1 Issue: 1
https://image.tmdb.org/t/p/w500/ld27rWKWCHpQdfFPOgW7MGinKyW.jpg'
Sync event triggered: sync-watchlist service-worker.js:72
Syncing watchlist data... service-worker.js:85
Sync request sent successfully! service-worker.js:87
Sync successful service-worker.js:75

Push Event


Dimensions: Responsive 1312 x 929 50% Offline
Mobile S – 320px
MovieHouse
my name is khan
Sort By: Popularity Descending Genre: All Genres
My Name Is Khan
Rating: 8.037
My Name Is Khan
Rating: 0
Elements Console Sources Application
Application
Manifest
Service work...
Storage
Storage
Local storage
Session stor...
Extension st...
IndexedDB
Cookies
Private state ...
Interest gro...
Shared stora...
Offline Update on reload Bypass for network
http://localhost:3000/ Network requests Update
Source service-worker.js
Received 4/4/2025, 4:47:38 PM
Status #6510 activated and is running Stop
Push { "method":"pushMessage","me Push
Sync syncMessage Sync
Periodic sync test-tag-from-devtool Periodic sync
Console What's new AI assistance Autofill
Filter Default levels 1 Issue: 1
Google Chrome
Movie House
{ "method":"pushMessage","message":"Hello!"}
localhost:3000

Project Title: **Roll No.**

MAD & PWA Lab

Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	17
Name	Omkar Gholap
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: Static Website Hosting Made Simple

GitHub Pages is a free hosting service that allows users to publish public webpages directly from a GitHub repository. It is particularly useful for static websites, project documentation, and blogs.

Key Features

- Jekyll Integration: Built-in support for Jekyll enables easy blogging.
- Custom Domains: Allows users to configure their own URLs.
- Automatic Page Deployment: Simply push your changes to the repository, and the updates go live.

Why Choose GitHub Pages?

- Completely Free: No hosting charges.
- Seamless GitHub Integration: Works directly with your repositories.
- Quick Setup: Just create a repository, push your files, and your site is live.

Who Uses GitHub Pages?

Companies like Lyft, CircleCI, and HubSpot use GitHub Pages for their documentation and static sites. It is widely adopted, appearing in 775 company stacks and 4,401 developer stacks.

Pros & Cons

Pros

- Familiar interface for GitHub users.
- Simple deployment via the gh-pages branch.

- Supports custom domains with easy DNS configuration.

Cons

- Repositories need to be public unless you have a paid plan.
- Limited HTTPS support for custom domains (expected to improve).
- Jekyll plugins have limited support.

Firebase: A Full-Featured Real-Time Backend

Firebase is a cloud-based real-time application platform developed by Google. It enables developers to build dynamic, collaborative applications with ease.

Key Features

- Real-Time Database: Automatically syncs data across all connected clients.
- Cloud-Based Storage: JSON-based storage accessible via REST APIs.
- Scalable Infrastructure: Works well with existing services and scales automatically.
- Authentication & Cloud Messaging: Secure login and push notifications.

Why Choose Firebase?

- Instant Backend Setup: No need to build a separate backend.
- Fast & Responsive: Real-time data synchronization.
- Built-in HTTPS: Free SSL certificates for custom domains.

Who Uses Firebase?

Companies like Instacart, 9GAG, and Twitch rely on Firebase for their backend needs. Firebase is widely adopted, appearing in 1,215 company stacks and 4,651 developer stacks.

Pros & Cons

Pros

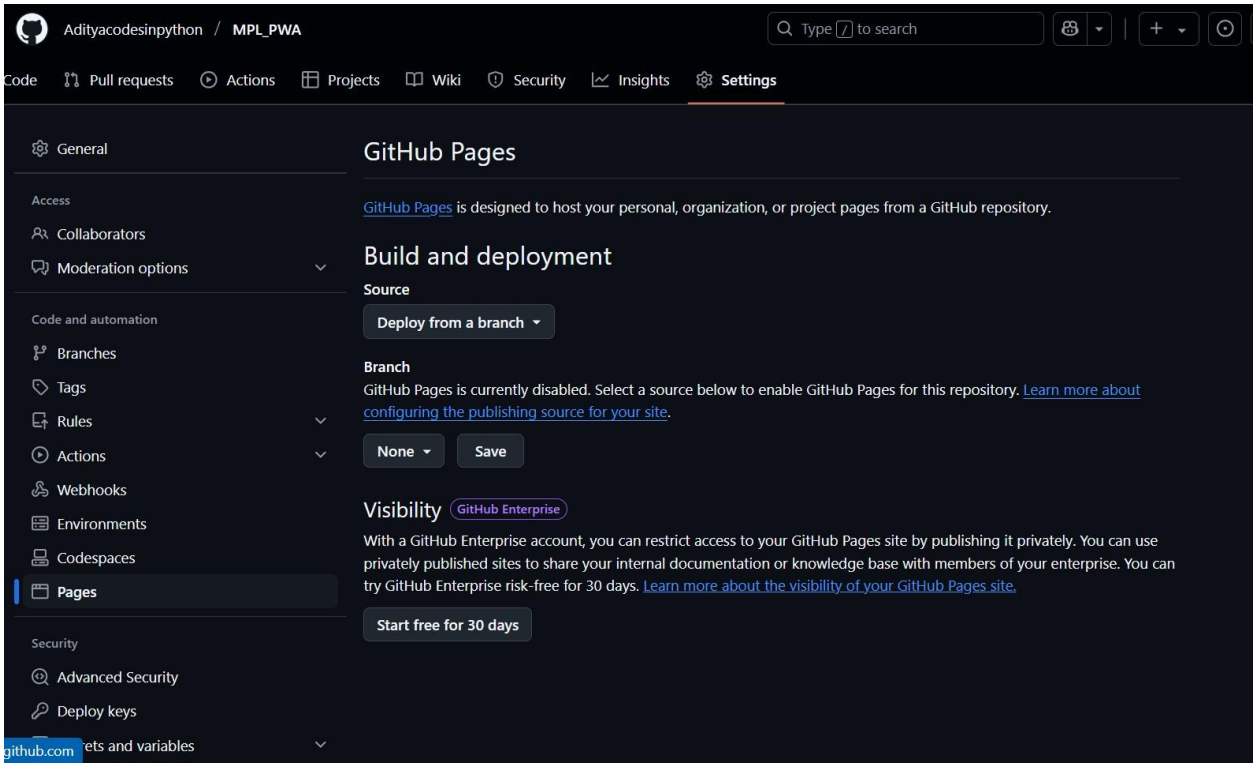
- Hosted by Google, ensuring reliability and security.
- Comes with authentication, messaging, and real-time database services.
- Free HTTPS support for all custom domains.

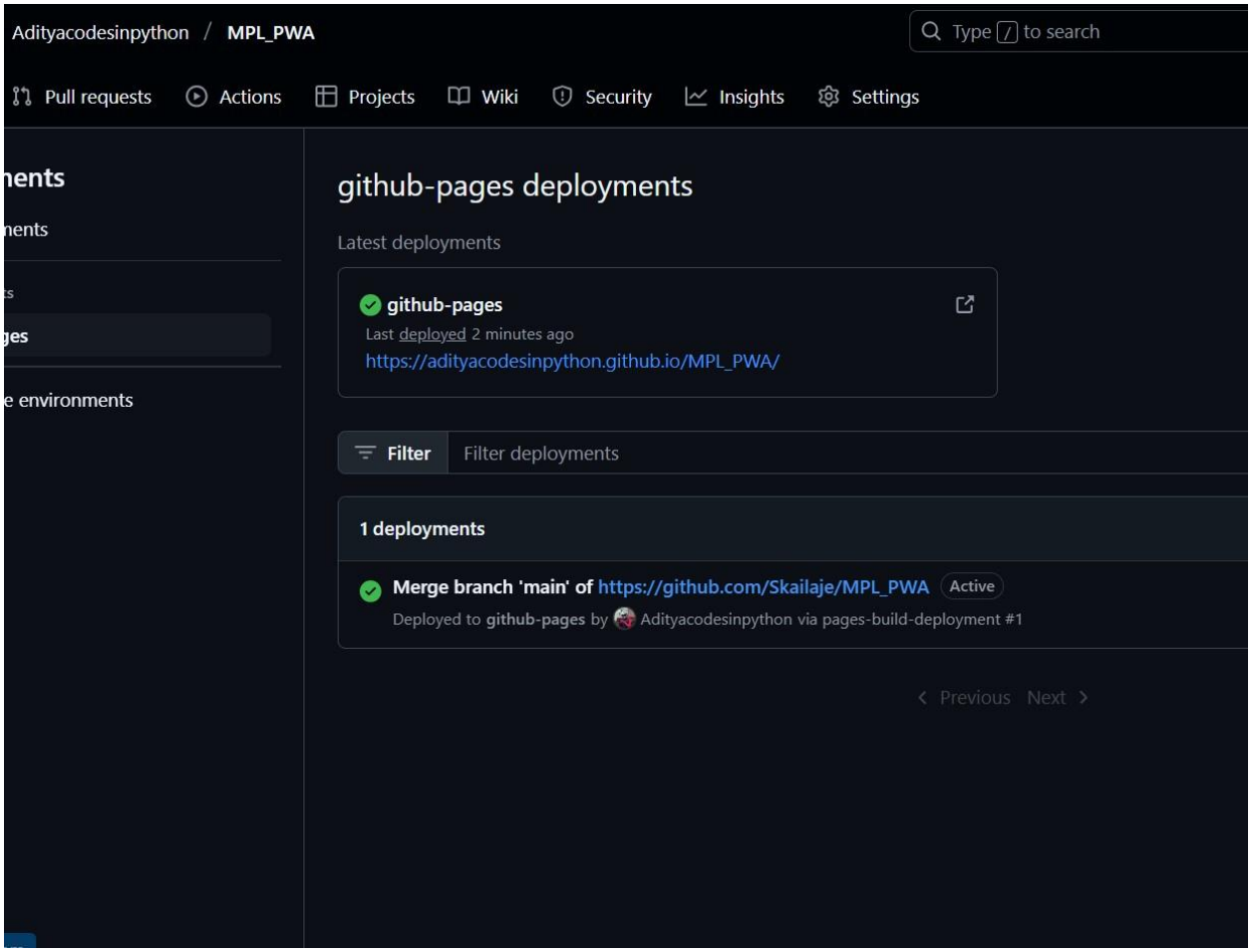
Cons

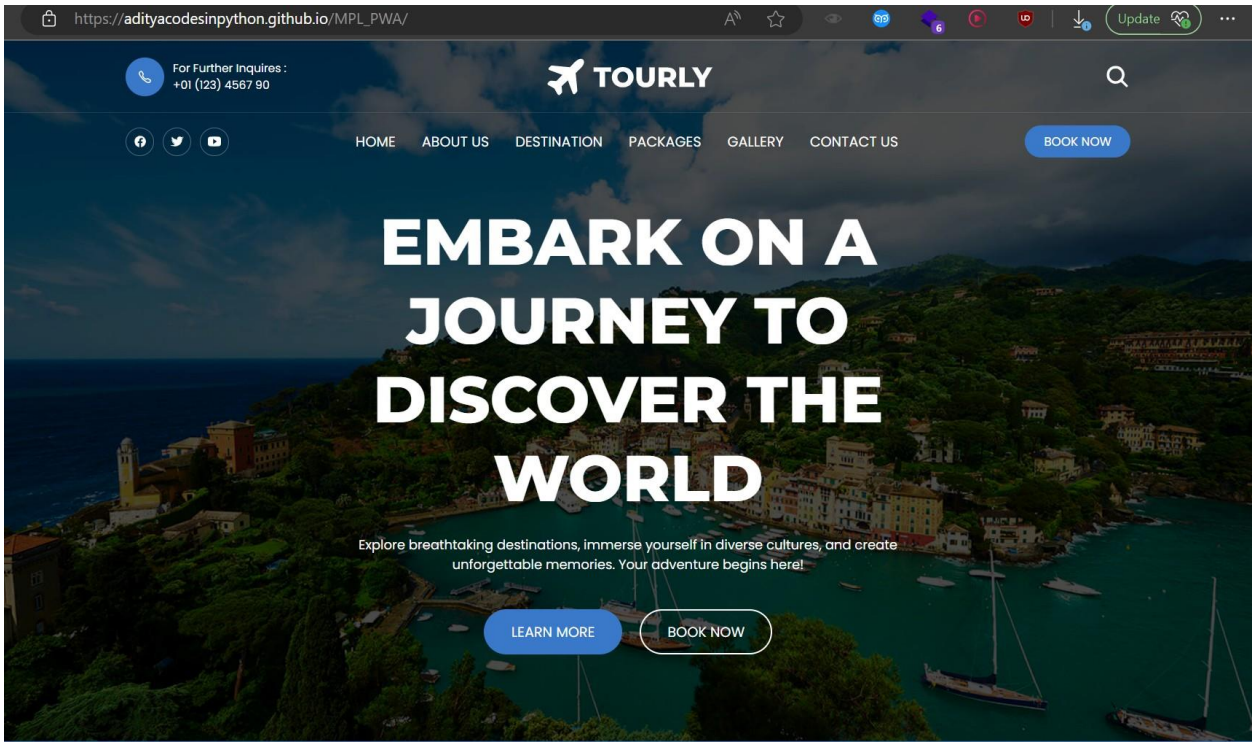
- Limited Free Plan: 10 GB of data transfer per month (can be mitigated with a CDN).
- Command-Line Deployment: No GUI for hosting.
- No Built-in Static Site Generator Support: Unlike GitHub Pages, Firebase doesn't natively support static site generators like Jekyll.

hosted link: [Tourly - Travel agancy]

Github Screenshot:




Adityacodesinpython / MPL_PWA
Type / to search
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
github-pages deployments
Latest deployments
github-pages
Last deployed 2 minutes ago
https://adityacodesinpython.github.io/MPL_PWA/
Filter
Filter deployments
1 deployments
Merge branch 'main' of https://github.com/Skailaje/MPL_PWA
Active
Deployed to github-pages by Adityacodesinpython via pages-build-deployment #1
Previous
Next


https://adityacodesinpython.github.io/MPL_PWA/
Update
For Further Inquires :
+01 (123) 4567 90
TOURLY
HOME
ABOUT US
DESTINATION
PACKAGES
GALLERY
CONTACT US
BOOK NOW
EMBARK ON A JOURNEY TO DISCOVER THE WORLD
Explore breathtaking destinations, immerse yourself in diverse cultures, and create unforgettable memories. Your adventure begins here!
LEARN MORE
BOOK NOW

Project Title: **Roll No.**

MAD & PWA Lab

Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	17
Name	Omkar Gholap
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 is an open-source tool that audits web applications based on multiple key parameters, including performance, accessibility, Progressive Web App (PWA) implementation, and best practices. It provides a detailed, automated report that helps developers optimize their websites efficiently. Unlike traditional manual audits, which can take days or even weeks, Lighthouse generates insights within minutes.

One of the key advantages of Lighthouse is its ease of use—no complex setup is required. Simply run it on a webpage or provide a URL, and it will generate an extensive performance report.

Key Features and Audit Metrics

Lighthouse can audit both desktop and mobile versions of a webpage. The core evaluation criteria include:

1. Performance

This metric measures how efficiently a webpage loads and displays content. Key factors influencing the performance score include:

- Page load speed – How quickly the page becomes visible to the user.
- First Contentful Paint (FCP) – The time taken for the first piece of content to appear.
- Largest Contentful Paint (LCP) – The time taken for the main content to fully load.
- Cumulative Layout Shift (CLS) – Measures how visually stable a page is (i.e., avoiding unexpected shifts in content).
- Time to Interactive (TTI) – The time it takes for the page to become fully functional. Lighthouse assigns a score from 0 to 100 based on percentile rankings, where:
- 100 → Top 2% of websites (98th percentile)
- 50 → Around the 75th percentile
- Lower scores → Indicate areas that need optimization

2. Progressive Web App (PWA) Score (Mobile)

With the rise of PWAs, modern web applications aim to provide a native app-like experience. Lighthouse evaluates the PWA implementation based on Google's Baseline PWA Checklist, which includes:

- Service Worker implementation – Ensuring offline support and background synchronization.
- App Manifest compliance – Providing metadata for better mobile integration.
- Viewport configuration – Optimizing mobile responsiveness.
- Performance in script-disabled environments – Ensuring the page functions even when JavaScript is disabled.

A high PWA score indicates that the application meets essential PWA criteria and provides an app-like user experience.

3. Accessibility
Accessibility ensures that web applications are usable by individuals with disabilities. Lighthouse audits a webpage based on:

- ARIA attributes – Enhancing accessibility through attributes like aria-required.
- Text alternatives for media – Ensuring audio and visual content is accessible.
- Semantic HTML – Proper use of <section>, <article>, <button>, and other elements that improve screen-reader compatibility.

Unlike other metrics, accessibility checks follow a pass/fail approach—if a necessary feature is missing, it significantly impacts the score. A higher accessibility score ensures inclusivity for users with visual or cognitive impairments. 4. Best Practices
This metric evaluates whether the website follows modern web development best practices, including:

- Use of HTTPS – Ensuring secure data transmission.
- Avoiding deprecated code – Preventing the use of outdated elements, directives, and libraries.
- Secure password inputs – Disabling paste-into fields to mitigate credential theft risks.
- User security alerts – Prompting users about geo-location access and cookie usage on load. A high score indicates that the website follows industry standards, improving security, usability, and maintainability.

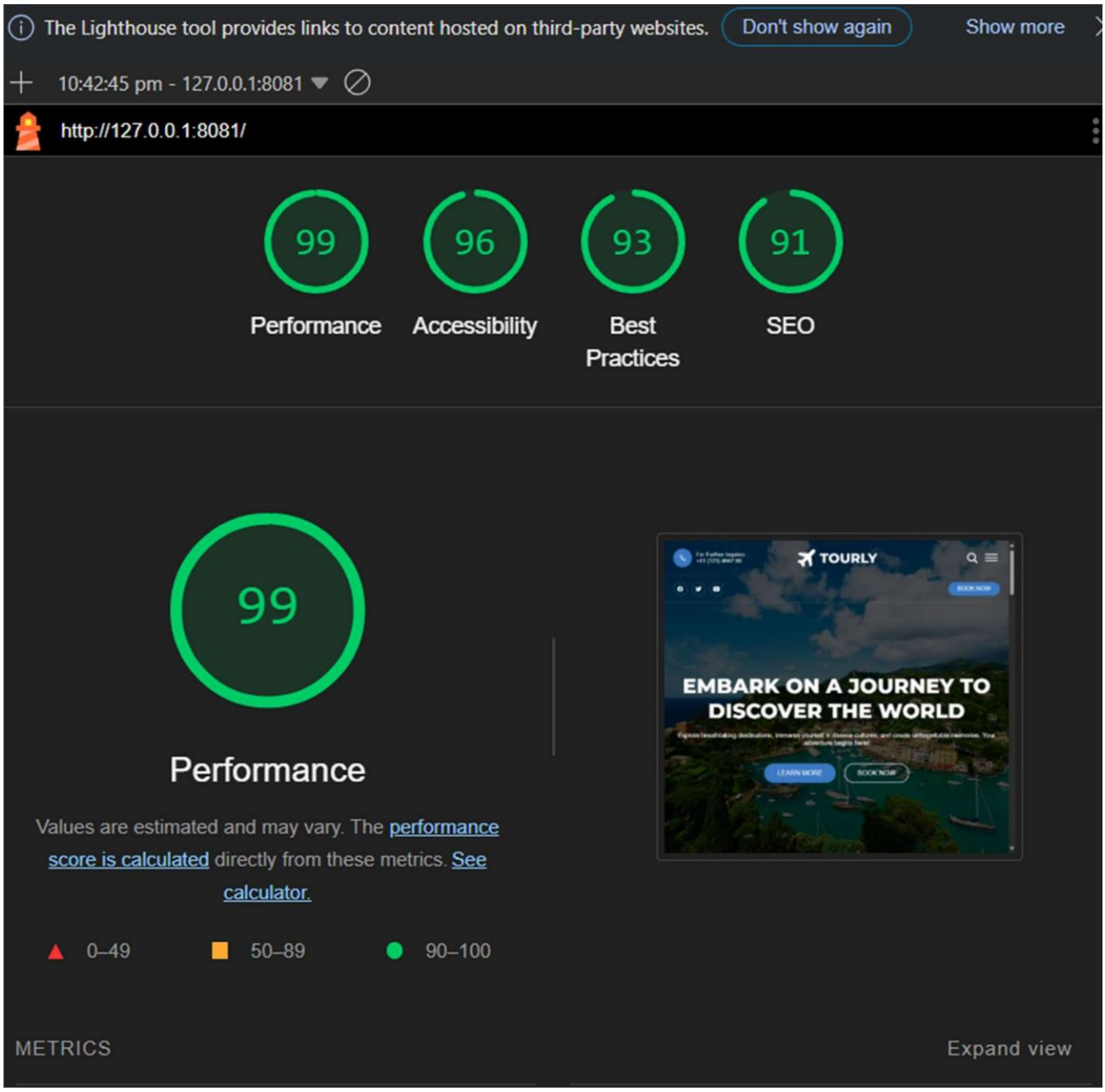
Manifest.json

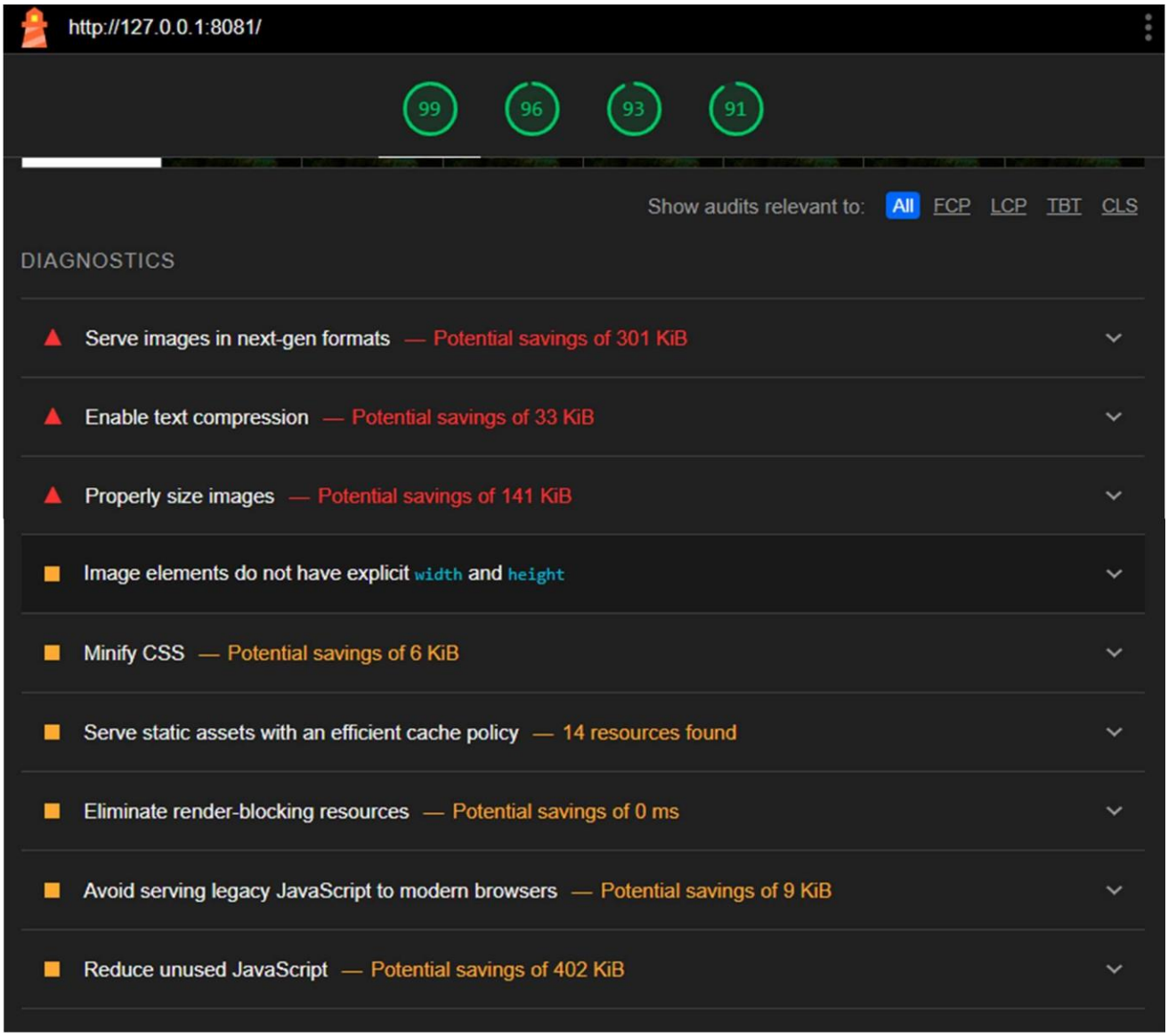
```
{
  "id": "/",
  "name": "TOURLY - Travel Ecommerce",
  "short_name": "TLY",
"description": "A travel ecommerce platform for
   booking tours and experiences.", "start_url":
"/index.html",
  "scope": "/",
  "display": "standalone",
  "background_color": "#FFFFFF",
  "theme_color": "#000000",
  "orientation": "portrait",
  "lang": "en",
  "categories": ["travel", "shopping",
"ecommerce"],
  "icons": [
    {
      "src": "assets/images/logo_tourly_192.png",
      "sizes": "192x192",
      "type": "image/png", "purpose":
      "any maskable"

},
  { "src":
    "assets/images/logo_tourly_512.png",
    "sizes": "512x512",
    "type": "image/png", "purpose":
    "any
    maskable"
  },
  {

 "src": "assets/images/logo_tourly_192.webp",
    "sizes": "192x192",
    "type": "image/webp", "purpose":
    "any
    maskable"
  },
  { "src":
"assets/images/logo_tourly_512.webp",
    "sizes": "512x512",
    "type": "image/webp", "purpose":
    "any
    maskable"
  }
```

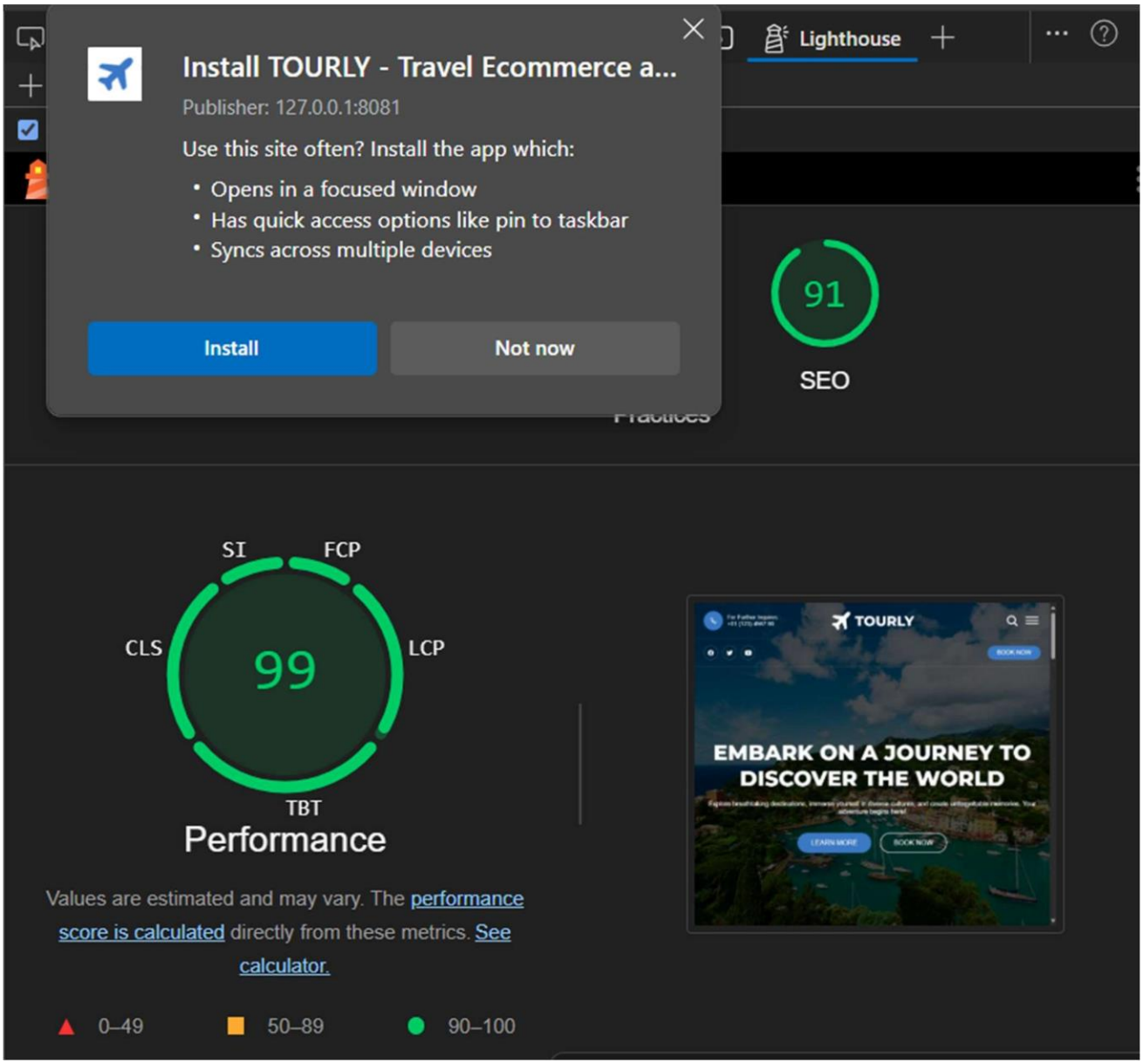
```
],
"screenshots": [
  {
```

```
  "src": "assets/screenshots/homepage.png",
     "sizes": "1080x1920",
     "type": "image/png"
   },
   {
     "src": "assets/screenshots/booking.png",
     "sizes": "1080x1920",
     "type": "image/png"
   }
  ],
  "shortcuts": [
   {


  "name": "Book a Tour",
     "short_name": "Book",
     "description": "Go directly to tour booking",
     "url": "/book.html",
     "icons": [
       {
         "src": "images/shortcut-icon.png",
         "sizes": "96x96",
         "type": "image/png"
       }
     ]
   },
   {
     "name": "View Popular Tours",
     "short_name": "Popular",
     "description": "Explore popular tours",
     "url": "/popular.html",
     "icons": [
       {
         "src": "images/popular-shortcut-
icon.png",
         "sizes": "96x96",
         "type": "image/png"
       }
     ]

   }
  ],
   "iarc_rating_id":
  "e10c6b5e-3b3c-4d3a-9f3b- 1e5f3e2b5f3c",
   "dir": "ltr",
   "related_applicatio
   ns": [
     {
       "platform":
       "play", "url":
  "https://play.google.com/store/apps/details?id=
  c om.tourly.app",
       "id": "com.tourly.app"
     },
     {
     "platform":
     "itunes", "url":
"https://apps.apple.com/app/id1234567890"
     }
   ],
   "prefer_related_applications": false }
```


TOURLY
BOOK NOW
EMBARK ON A JOURNEY TO DISCOVER THE WORLD
Explore breathtaking destinations, immerse yourself in diverse cultures, and create unforgettable memories. Your adventure begins here!
LEARN MORE
BOOK NOW
Search Destination*
Enter Destination
Pax Number*
No.of People


The Lighthouse tool provides links to content hosted on third-party websites.
Don't show again
Show more
10:42:45 pm - 127.0.0.1:8081
http://127.0.0.1:8081/
99
Performance
96
Accessibility
93
Best Practices
91
SEO
99
Performance
Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.
0–49
50–89
90–100
TOURLY
EMBARK ON A JOURNEY TO DISCOVER THE WORLD
LEARN MORE
BOOK NOW
METRICS
Expand view


http://127.0.0.1:8081/
99
96
93
91
Show audits relevant to: All FCP LCP TBT CLS
DIAGNOSTICS
Serve images in next-gen formats — Potential savings of 301 KiB
Enable text compression — Potential savings of 33 KiB
Properly size images — Potential savings of 141 KiB
Image elements do not have explicit width and height
Minify CSS — Potential savings of 6 KiB
Serve static assets with an efficient cache policy — 14 resources found
Eliminate render-blocking resources — Potential savings of 0 ms
Avoid serving legacy JavaScript to modern browsers — Potential savings of 9 KiB
Reduce unused JavaScript — Potential savings of 402 KiB


Lighthouse
Install TOURLY - Travel Ecommerce a...
Publisher: 127.0.0.1:8081
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
91
SEO
SI
FCP
CLS
99
LCP
TBT
Performance
Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.
0–49
50–89
90–100
TOURLY
EMBARK ON A JOURNEY TO DISCOVER THE WORLD

Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	Assignment-2
Assignment 2 Questions	1. Define Progressive Web App (PWA) and explain its significance in modern web development.Discuss the key characteristics that differentiate PWAs from traditional mobile apps 2. Define responsive web design and explain its importance in the context of Progressive Web Apps.Compare and contrast responsive, fluid, and adaptive web design approaches. 3. Describe the lifecycle of Service Workers, including registration, installation, and activation phases. 4. Explain the use of IndexedDB in the Service Worker for data storage.
Roll No.	17
Name	Omkar Gholap
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4:Understand various PWA frameworks and their requirements LO5: Design and Develop a responsive User Interface by applying PWA Design techniques LO6:Develop and Analyze PWA Features and deploy it over app hosting solutions
Grade:	

Project Title: **Roll No.**

MAD & PWA Lab

Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	17
Name	Omkar Gholap
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	