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 **Miss Anmol Vaswani** of **D15A** semester **VI,** have successfully completed necessary experiments in the **MAD & PWA Lab** under my supervision in **VES Institute of Technology** during the academic year **2023-2024.**

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Project Title: Unacademy Clone/PWA- Counter App **Roll No. 67**

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

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

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers : Mrs. Kajal Jewani.

Email : kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: Unacademy Clone/PWA- Counter App **Roll No. 67**

Lab Objectives:

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

Lab Outcomes:

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

Index

Sr. No	Experiment Title	LO	DOP	DOS	Grade
1.	To install and configure the Flutter Environment	LO1	19/01	02/02	15
2.	To design Flutter UI by including common widgets.	LO2	26/01	02/02	15
3.	To include icons, images, fonts in Flutter app	LO2	02/02	09/02	13
4.	To create an interactive Form using form widget	LO2	09/02	16/02	13
5.	To apply navigation, routing and gestures in Flutter App	LO2	16/02	23/02	12
6.	To Connect Flutter UI with fireBase database	LO3	23/02	08/03	12
7.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4	08/03	22/03	15
8.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA	LO5	15/03	22/03	15
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5	22/03	01/04	15
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5	29/03	01/04	15
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6	29/03	01/04	15
12.	Assignment-1	LO1,LO2 ,LO3	28/01	05/02	5
13.	Assignment-2	LO4,LO5 ,LO6	14/03	21/03	4

MAD & PWA Lab

Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	67
Name	Anmol Vaswani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15

Name: - Anmol Vaswani
Div: - D15A
Roll No.: -67
Batch: - C

Experiment-1

Aim: - To install and configure the Flutter Environment.

Theory: -

Flutter:
Flutter is an open-source UI software development toolkit created by Google for building natively compiled applications for mobile, web, and desktop from a single codebase.

Key Features of Flutter:
Single Codebase: Flutter allows developers to write code once and deploy it on multiple platforms like Android, iOS, web, and desktop. Hot Reload: One of the most significant features of Flutter is its hot reload capability. Changes to the code can be instantly reflected in the running app, making the development process faster and more interactive.

Widget-Based Framework:
Flutter is based on a reactive widget framework. Widgets are the basic building blocks of the user interface, and they are used to create complex UIs. Rich Set of Widgets: Flutter provides a comprehensive set of customizable widgets, including material design and Cupertino-style widgets, to create visually appealing and platform-specific user interfaces.

Native Performance:
Flutter compiles to native code, resulting in high performance and smooth animations. It does not rely on a bridge to communicate with the native modules, enhancing the app's speed.

Dart Programming Language:
Flutter uses Dart as its programming language. Dart is an object-oriented, garbage-collected language that is easy to learn and provides a good developerexperience.

Community and Ecosystem:
Flutter has a growing and active community that contributes to its ecosystem. There is a wide range of packages and plugins available on pub.dev, Flutter's package repository.

Flutter is a cutting-edge open-source framework developed by Google for crafting high-quality native interfaces on various platforms using a single codebase. It empowers developers to build visually appealing and responsive applications with ease. The framework is known for its expressive and flexible UI, enabling seamless development

across iOS, Android, and web platforms. Flutter's hot-reload feature allows developers to instantly view changes, streamlining the development process. Its robust widget-based architecture facilitates the creation of beautiful and performant apps, making it a preferred choice for modern app development.

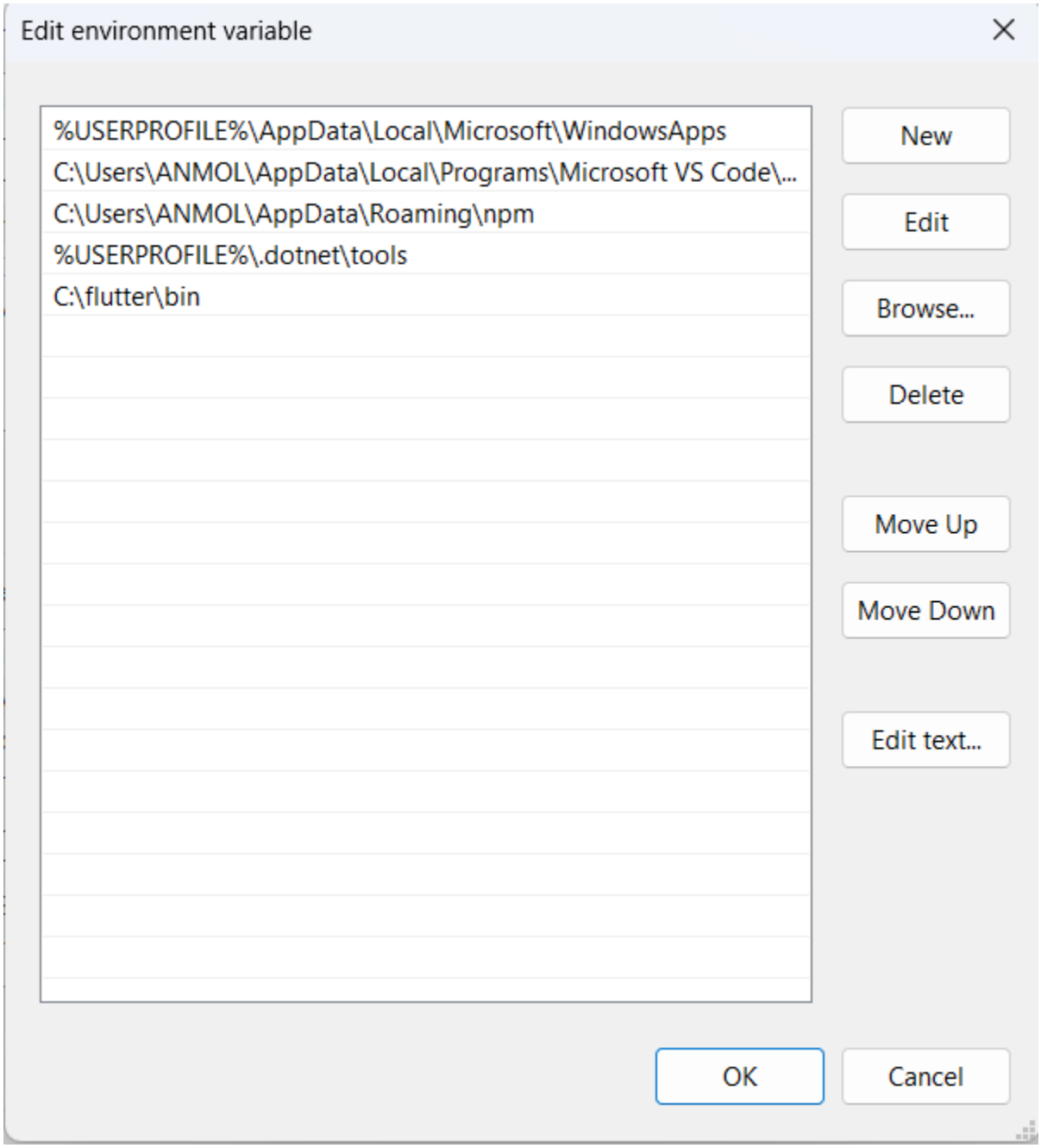
Android Studio, an advanced integrated development environment (IDE) created byGoogle, stands as the primary choice for Android app development. Boasting a user-friendly interface, powerful coding tools, and seamless integration with the Android platform, it provides developers with a robust environment for crafting high-performance applications. With features like intelligent code completion, real-time error checking, and an intuitive layout editor, Android Studio enhances productivity and accelerates the app development lifecycle. Its support for various Android devices, extensive testing capabilities, and built-in emulators contribute to creating reliable and efficient Android applications.

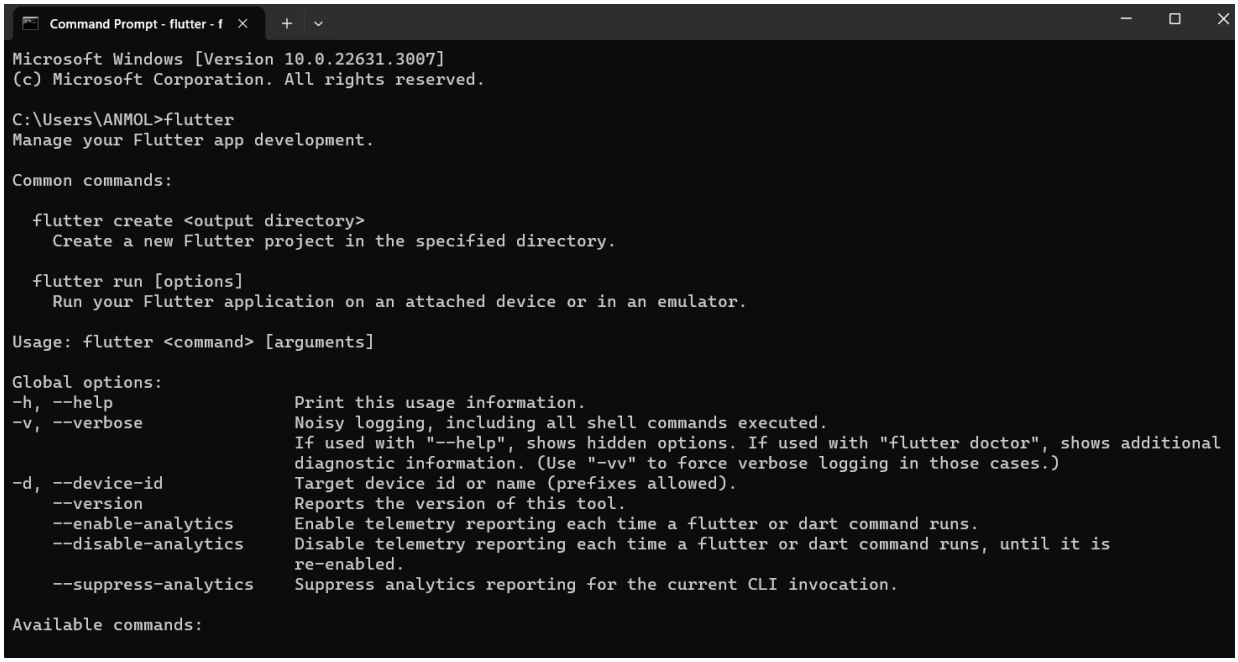
Flutter installed successfully: -

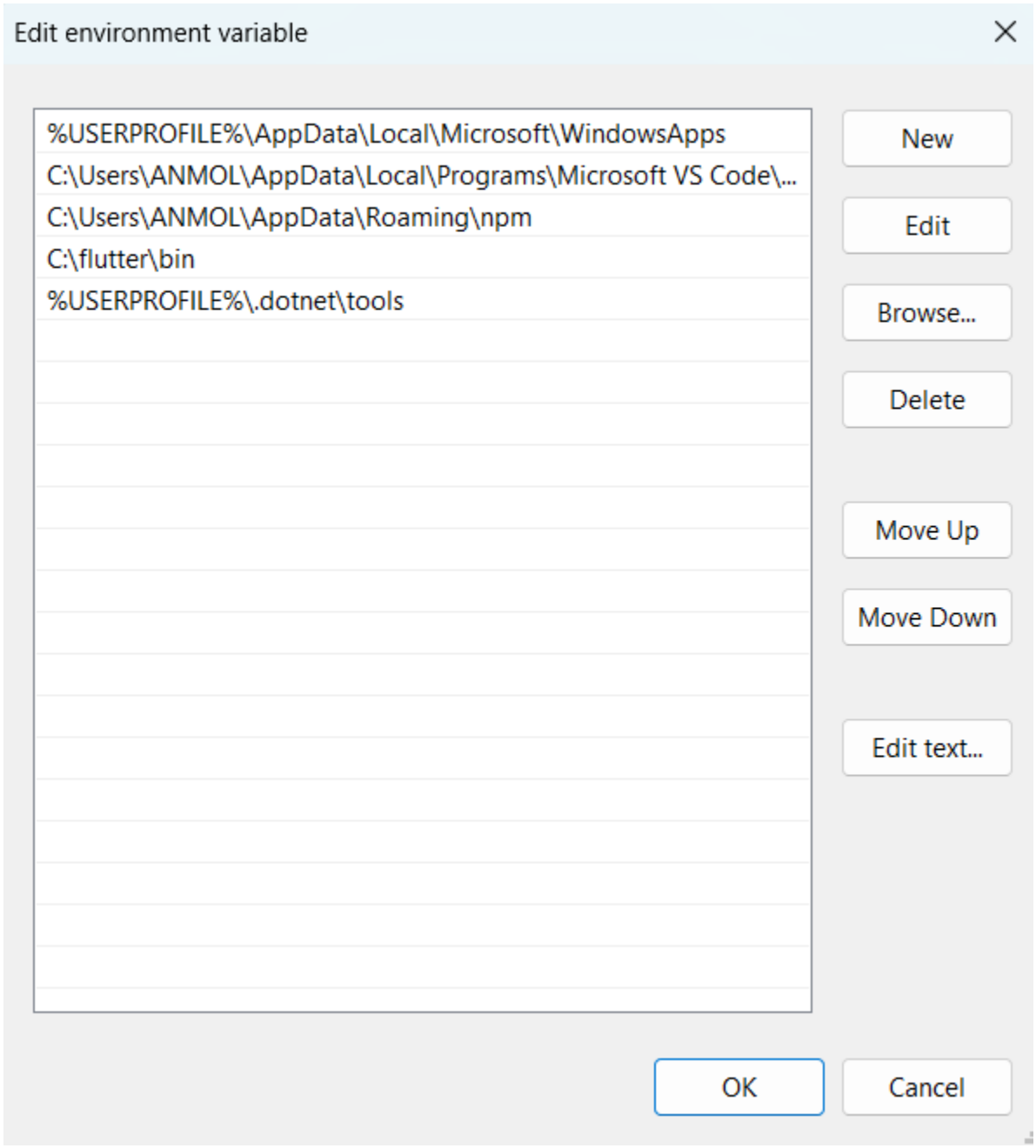
```
PS C:\Users\Student> flutter
Manage your Flutter app development.

Common commands:

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

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

Usage: flutter <command> [arguments]

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

Available commands:

Flutter SDK
```

Flutter doctor

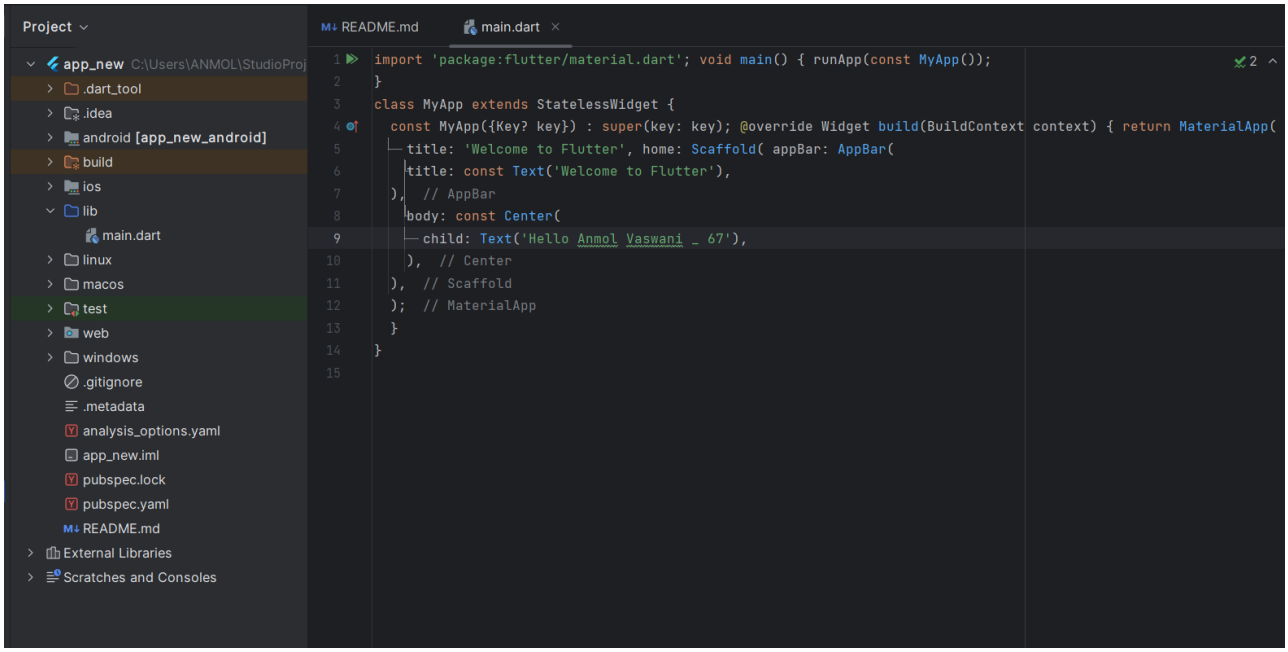
```
Microsoft Windows [Version 10.0.22000.2652]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Student>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.16.7, on Microsoft Windows [Version 10.0.22000.2652], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[!] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
    X cmdline-tools component is missing
      Run `path/to/sdkmanager --install "cmdline-tools;latest"`
      See https://developer.android.com/studio/command-line for more details.
    X Android license status unknown.
      Run `flutter doctor --android-licenses` to accept the SDK licenses.
      See https://flutter.dev/docs/get-started/install/windows#android-setup for more details.
[√] Chrome - develop for the web
[X] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows apps.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[√] Android Studio (version 2023.1)
[√] VS Code (version 1.85.1)
[√] Connected device (3 available)
[!] Network resources
    X A cryptographic error occurred while checking "https://pub.dev/": Connection terminated during handshake
      You may be experiencing a man-in-the-middle attack, your network may be compromised, or you may have malware
      installed on your computer.
    X A cryptographic error occurred while checking "https://maven.google.com/": Connection terminated during handshake
      You may be experiencing a man-in-the-middle attack, your network may be compromised, or you may have malware
      installed on your computer.

! Doctor found issues in 3 categories.
```


Edit environment variable
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Users\ANMOL\AppData\Local\Programs\Microsoft VS Code\...
C:\Users\ANMOL\AppData\Roaming\npm
%USERPROFILE%\.dotnet\tools
C:\flutter\bin
New
Edit
Browse...
Delete
Move Up
Move Down
Edit text...
OK
Cancel

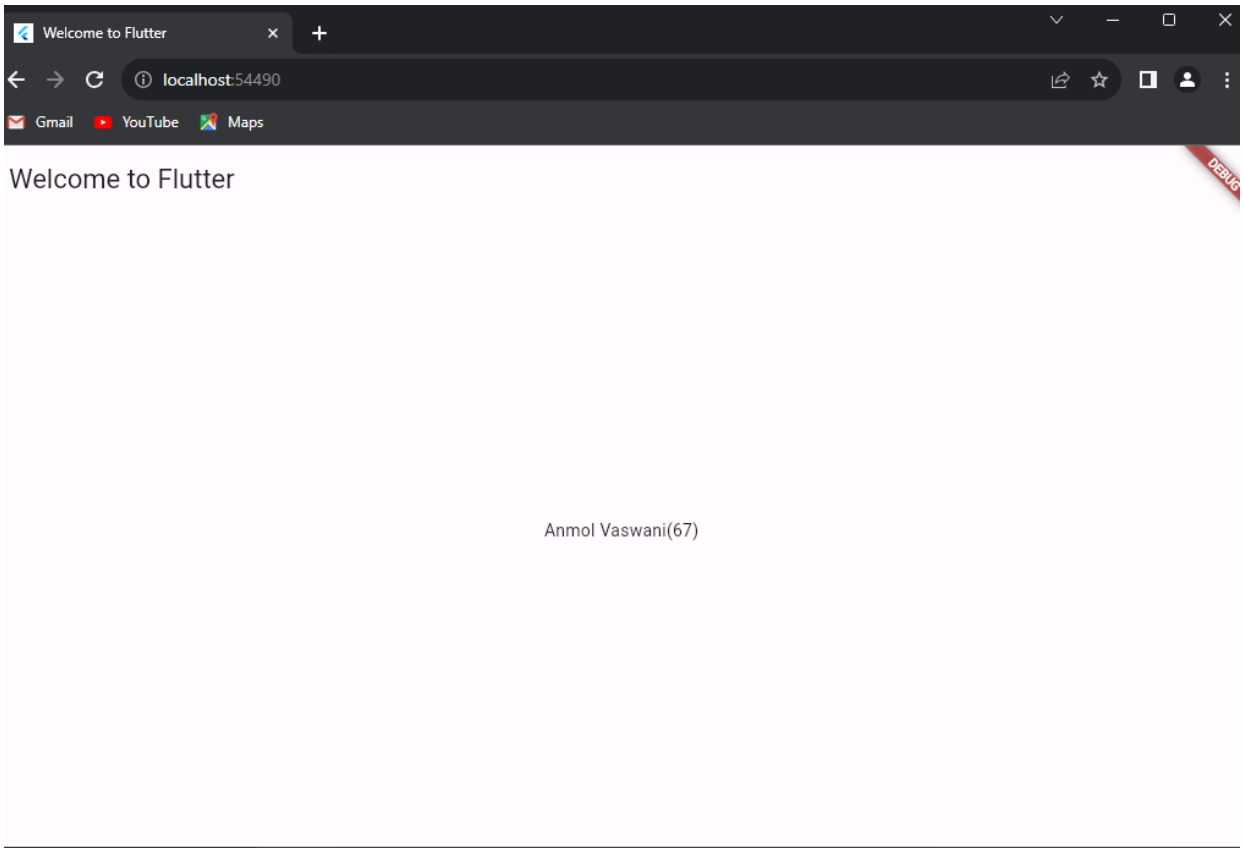

Command Prompt - flutter - f
Microsoft Windows [Version 10.0.22631.3007]
(c) Microsoft Corporation. All rights reserved.
C:\Users\ANMOL>flutter
Manage your Flutter app development.
Common commands:
flutter create <output directory>
Create a new Flutter project in the specified directory.
flutter run [options]
Run your Flutter application on an attached device or in an emulator.
Usage: flutter <command> [arguments]
Global options:
-h, --help Print this usage information.
-v, --verbose Noisy logging, including all shell commands executed.
If used with "--help", shows hidden options. If used with "flutter doctor", shows additional diagnostic information. (Use "-vv" to force verbose logging in those cases.)
-d, --device-id Target device id or name (prefixes allowed).
--version Reports the version of this tool.
--enable-analytics Enable telemetry reporting each time a flutter or dart command runs.
--disable-analytics Disable telemetry reporting each time a flutter or dart command runs, until it is re-enabled.
--suppress-analytics Suppress analytics reporting for the current CLI invocation.
Available commands:

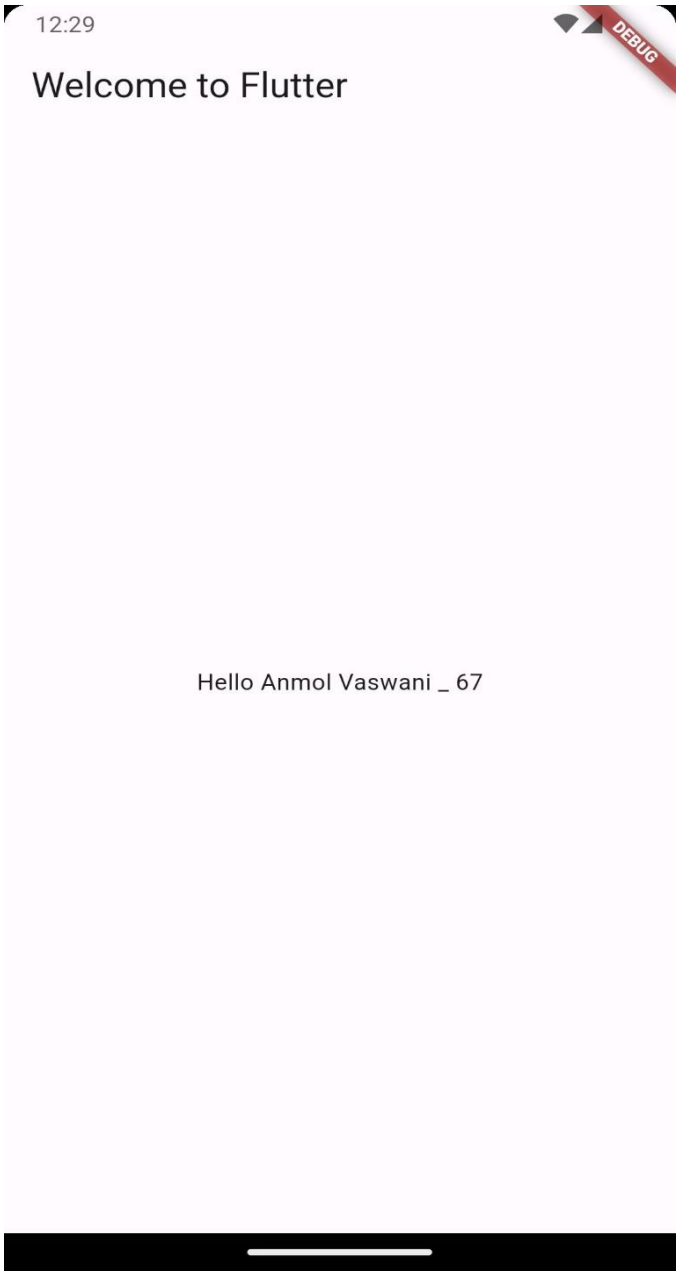

Edit environment variable
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Users\ANMOL\AppData\Local\Programs\Microsoft VS Code\...
C:\Users\ANMOL\AppData\Roaming\npm
C:\flutter\bin
%USERPROFILE%\.dotnet\tools
New
Edit
Browse...
Delete
Move Up
Move Down
Edit text...
OK
Cancel

Code: -


Project
app_new C:\Users\ANMOL\StudioProj
.dart_tool
.idea
android [app_new_android]
build
ios
lib
main.dart
linux
macos
test
web
windows
.gitignore
.metadata
analysis_options.yaml
app_new.iml
pubspec.lock
pubspec.yaml
README.md
External Libraries
Scratches and Consoles
README.md
main.dart
import 'package:flutter/material.dart'; void main() { runApp(const MyApp());
}
class MyApp extends StatelessWidget {
const MyApp({Key? key}) : super(key: key); @override Widget build(BuildContext context) { return MaterialApp(
title: 'Welcome to Flutter', home: Scaffold(appBar: AppBar(
title: const Text('Welcome to Flutter'),
), // AppBar
body: const Center(
child: Text('Hello Anmol Vaswani _ 67'),
), // Center
), // Scaffold
); // MaterialApp
}
}

Output: -


Welcome to Flutter
localhost:54490
Gmail
YouTube
Maps
Welcome to Flutter
DEBUG
Anmol Vaswani(67)


12:29
DEBUG
Welcome to Flutter
Hello Anmol Vaswani _ 67

MAD & PWA Lab

Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	67
Name	Anmol Vaswani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Name: - Anmol Vaswani
Div: - D15A
Roll No.: -67

Experiment 2

Aim: To design Flutter UI by including common widgets.

Theory:

Widgets: In Flutter, each element on a screen of the app is represented as a widget. The view of the screen is constructed by arranging these widgets in a tree-like structure. Widgets serve as the building blocks of the app's UI, and the layout, appearance, and functionality of the app depend on the choice and arrangement of these widgets.
Flutter Container: Container is a versatile widget in Flutter that allows developers to customize its appearance using properties such as color, padding, margin, width, height, and more. It acts as a basic building block for creating layouts and can contain other widgets.

Flutter Scaffold: Scaffold is a layout structure provided by Flutter that serves as a framework for implementing the basic material design visual layout structure of the app. It provides a scaffold upon which other widgets can be built, including app bars, drawers, bottom navigation bars, and more.

Flutter Text: Text widget is used to display a string of text with a single style. It allows developers to customize the text's appearance, including its font, size, color, alignment, and more.

Flutter Row & Column: Row and Column are layout widgets used for arranging child widgets horizontally (Row) or vertically (Column). These widgets allow developers to create flexible and responsive layouts by arranging widgets in rows or columns.

Flutter Icons: Icons widget is used to display Material icons in the app. Material icons are a collection of system icons provided by Flutter that represent common actions, items, and concepts. The Icons widget allows developers to easily incorporate these icons into their app's UI by specifying the desired icon's name.

Flutter Buttons: Buttons are interactive widgets that trigger actions when tapped by the user. Flutter provides various button widgets like Raised Button, Flat Button, IconButton, FloatingActionButton, etc. These widgets can be customized with different colors, text, icons, and onPressed handlers to define their behavior.

The Code of Centres.dart:

```
import 'package:flutter/material.dart';
import 'package:unacademy_app/BookVisit.dart';
import 'BookVisit.dart'; // Importing the BookVisit.dart file
import 'DownloadBrochure.dart';
class Centres extends StatefulWidget {
  const Centres({super.key});

  @override
  _CentresState createState() => _CentresState();
}
class _CentresState extends State<Centres> {

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

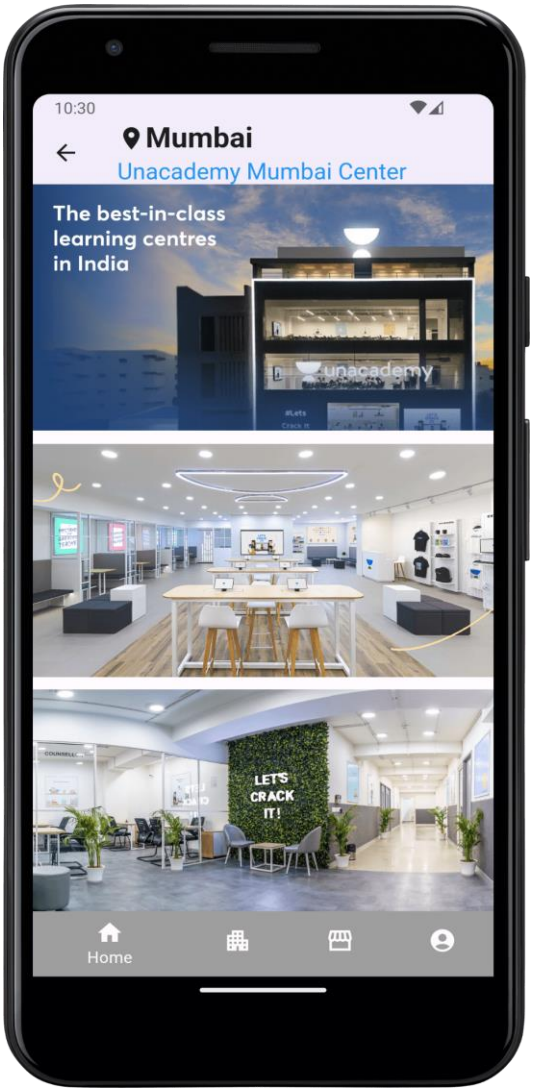
```
appBar: AppBar(
  title: const SingleChildScrollView(
    scrollDirection: Axis.horizontal,
    child: Row(
      children: [
        Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Row(
              children: [
                Icon(Icons.location_on),
                Text(
                  'Mumbai',
                  style: TextStyle(fontWeight: FontWeight.bold, fontSize: 25),
                ),
              ],
            ),
            SizedBox(height: 0.5),
            Text(
              'Unacademy Mumbai Center',
              style: TextStyle(color: Colors.blue, fontSize: 20),
            ),
          ],
        ),
      ],
    ),
  ),
),
body: SingleChildScrollView(
  child: Column(
    children: [
      SizedBox(
        width: 800, // Adjust width as needed
        height: 225, // Adjust height as needed
        child: Image.asset('assets/images/building.png'),
      ),
      SizedBox(
        width: 500, // Adjust width as needed
        height: 225, // Adjust height as needed
        child: Image.asset('assets/images/Office.jpg'),
      ),
      SizedBox(
        width: 800, // Adjust width as needed
        height: 225, // Adjust height as needed
        child: Image.asset('assets/images/closer.jpeg'),
      ),
      const SizedBox(height: 15),
      const Text(
        'Why Unacademy Centre?',
        style: TextStyle(fontSize: 29, fontWeight: FontWeight.bold),
      ),
      const SizedBox(height: 2),
      const Text(

        'Get an all-encompassing learning package ',

      textAlign: TextAlign.center,
```

```
),
const SizedBox(height: 2),
const Text(
  'that includes the best study material, ',
  textAlign: TextAlign.center,
),
const SizedBox(height: 2),
const Text(
  'library resources, advanced technology,',
  textAlign: TextAlign.center,
),
const SizedBox(height: 2),
const Text(
  'double-clearing sessions,1:1 mentorship and more',
  textAlign: TextAlign.center,
),
const SizedBox(height: 5),
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    ElevatedButton(
      onPressed: () {
        Navigator.push( // Navigating to BookVisit page
          context,
          MaterialPageRoute(builder: (context) => const BookVisit()),
        );
      },
      child: Text('Book a Visit'),
      style: ElevatedButton.styleFrom(
        primary: Colors.blue,
        onPrimary: Colors.white,
      ),
    ),
    const SizedBox(width: 20),
    ElevatedButton(
      onPressed: () {
        Navigator.push( // Navigating to BookVisit page
          context,
          MaterialPageRoute(builder: (context) =>  DownloadBrochure()),
        );// Add functionality for downloading brochure
      },
      style: ElevatedButton.styleFrom(
        primary: Colors.white,
        onPrimary: Colors.black,
        side: const BorderSide(color: Colors.black),
      ),
      child: const Text('See Brochure'),
    ),
  ],
),
SizedBox(
  width: 1000, // Adjust width as needed
  height: 650, // Adjust height as needed

  child: Image.asset('assets/images/service.jpeg'),
),
SizedBox(
```

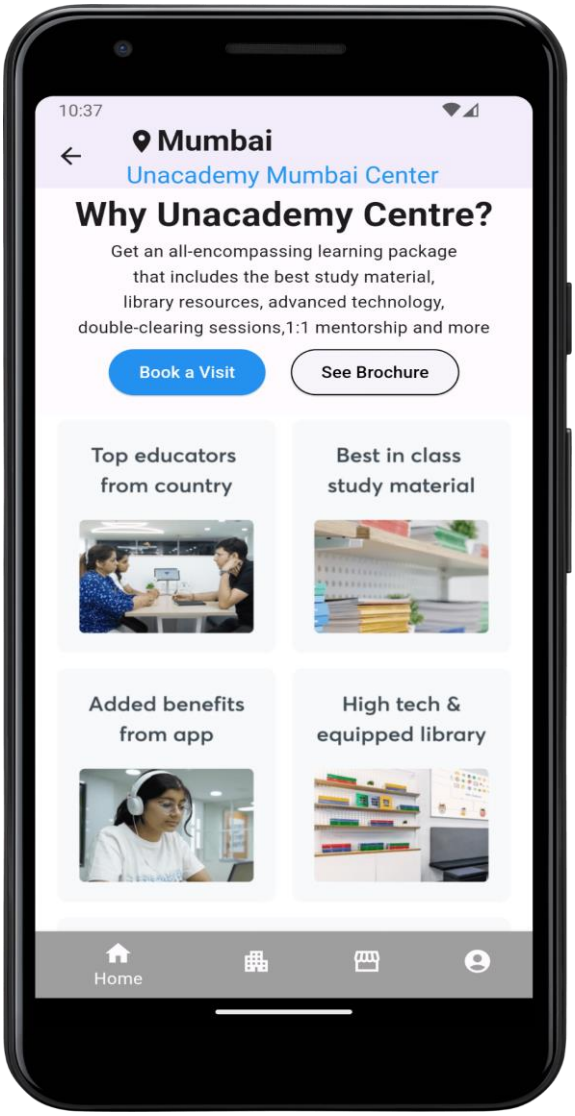
```
          width: 1000, // Adjust width as needed
            height: 1330, // Adjust height as needed
            child: Image.asset('assets/images/rankers.jpeg'),
          ),
          SizedBox(
            width: 3000, // Adjust width as needed
            height: 350, // Adjust height as needed
            child: Image.asset('assets/images/mumbai.jpeg'),
          ),
        ],
      ),
    ),
    bottomNavigationBar: BottomNavigationBar(
      items: const [
        BottomNavigationBarItem(
          icon: Icon(Icons.home),
          label: 'Home',
          backgroundColor: Colors.grey,
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.apartment),
          label: 'Centres',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.storefront_sharp),
          label: 'Store',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.account_circle),
          label: 'Me',
        ),
      ],
    ),
  );
 }
}
```

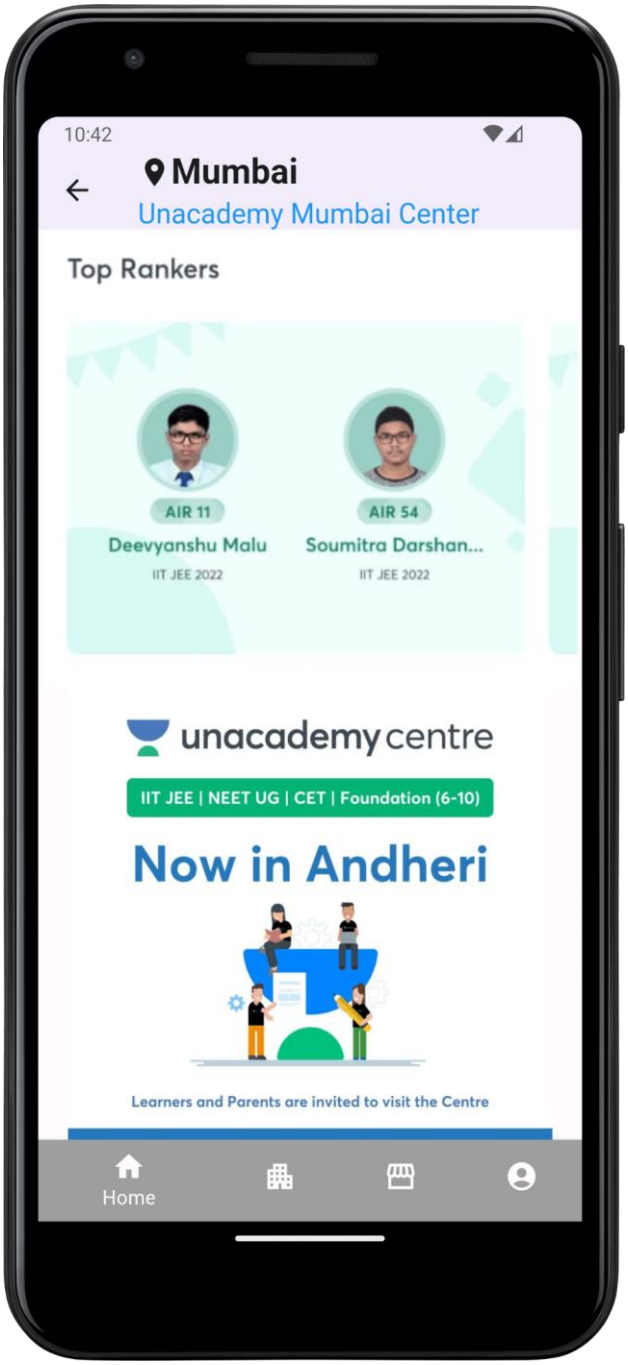
Output: -

Project Title: Unacademy Clone/PWA- Counter App **Roll No. 67**



Project Title: Unacademy Clone/PWA- Counter App **Roll No. 67**





Conclusion: - Through the implementation of various widgets such as Icons, fonts, Bottom Navigation Bar, and images in the Unacademy app project, a comprehensive understanding of fundamental components in mobile app development was achieved.

MAD & PWA Lab

Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	67
Name	Anmol Vaswani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	13

Name: - Anmol Vaswani
Div: - D15A
Roll No.: - 67

Experiment 3

AIM: To design Flutter UI by including common widgets (using Images).

THEORY:

Widgets as Building Blocks:

Flutter apps are built using widgets, which are reusable components that represent UI elements. Common widgets like Text, Image, Container, Row, Column, ListView, etc., provide building blocks for your UI.

Widgets can be nested within each other to create complex layouts and hierarchies.

Image Handling:

Flutter provides various ways to display images in your UI:

Asset Images: Images stored within your app's assets folder. Use AssetImage widget.

Network Images: Images loaded from URLs. Use NetworkImage widget.

Memory Images: Images loaded from memory buffers. Use MemoryImage widget.

File Images: Images loaded from local files. Use FileImage widget.

Each widget offers customization options like scaling, fitting, and alignment.

Layout and Composition:

Flutter uses a flexible layout system based on widgets.

Widgets like Container can define padding, margins, and background colors.

Layout widgets like Row and Column arrange children horizontally or vertically.

Stack widget allows layering widgets with positioning control.

Use padding, margins, and alignment properties to fine-tune the visual hierarchy.

Best Practices:

Choose the appropriate image widget based on the source and loading strategy. Optimize image sizes and compression to improve performance.

Use placeholder widgets while images are loading to enhance user experience. Consider using image caches to avoid redundant downloads.

Leverage Flutter's built-in animation features for smooth transitions and effects.

Follow accessibility guidelines to ensure your UI is usable for everyone.

Additional Considerations:

Explore advanced image manipulation packages like image and cached_network_image.

Experiment with different layout widgets and techniques to achieve desired UI structures.

Test your UI on different devices and screen sizes for responsiveness.

Consider incorporating image gestures like tapping, zooming, and panning.

The Code is of home_page. dart:

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

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

  @override
  Widget build(BuildContext context) { return
    Scaffold(
      appBar: AppBar(
        title: const Text('IIT JEE'),
        actions: [
          TextButton(
            onPressed: () {
              // Implement action for booking a call
            },
            child: const Text('Book a Call'),
          ),
          IconButton(
            icon: const Icon(Icons.phone),
            onPressed: () {
              // Implement action for booking a call
            },
          ),
        ],
      ),
      body: ListView(
        children: [
        Stack(
            children: [
              Column(
                crossAxisAlignment: CrossAxisAlignment.stretch,
                children: [
                  Container(
                    alignment: Alignment.center, padding:
                    const EdgeInsets.all(0.11),
```

```
  child: const Row(
   mainAxisAlignment: MainAxisAlignment.spaceBetween,
   children: [
    Text(
     ' Your Selected Goal', // Replace with the selected goal style:
     TextStyle(
      fontWeight: FontWeight.bold,
     ),
    ),
   ],
  ),
 ),
 const Padding(
  padding: EdgeInsets.all(0.011),
  child: TextField(
   decoration: InputDecoration(
    hintText: 'Search in the selected goal...', border:
    OutlineInputBorder(),
    prefixIcon: Icon(Icons.search), // Added search icon
   ),
  ),
 ),
 SizedBox(
  width: 400, // Adjust the width as needed height:
  180, // Adjust the height as needed
  child: Image.asset("assets/images/unacademybackground.jpg"), // Replace with your
image
 ),

 const SizedBox(height: 18.0),
 Container(
  padding: const EdgeInsets.all(18.0),
  color: Colors.yellow[100], // Light yellow background color child:
  Row(
   mainAxisAlignment: MainAxisAlignment.spaceBetween,
   children: [
    const Column(
```

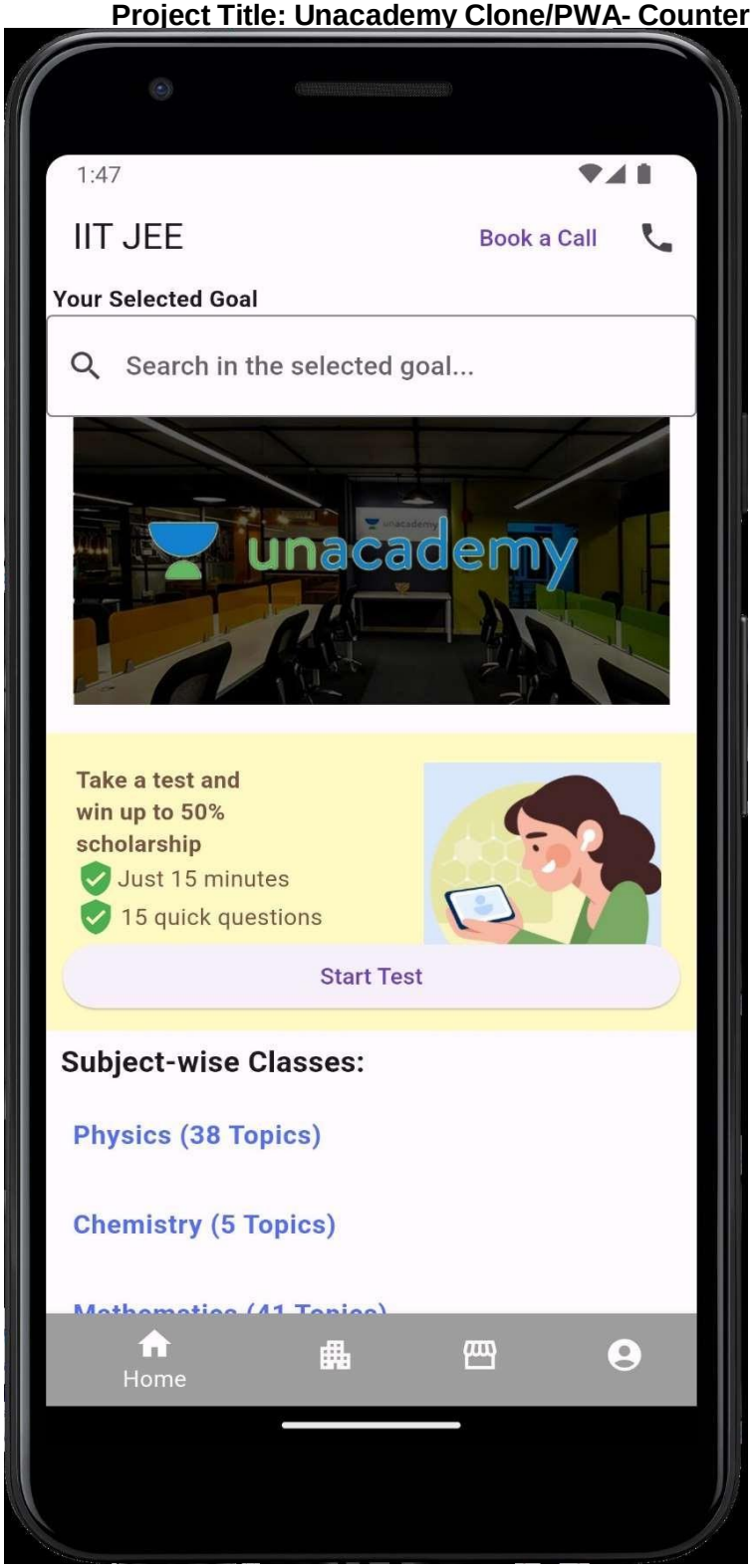
```
crossAxisAlignment: CrossAxisAlignment.start,
children: [
  Text(
    'Take a test and', // First line
    style: TextStyle(color: Colors.brown, fontWeight: FontWeight.bold),
  ),
  Text(
    'win up to 50% ', // Second line
    style: TextStyle(color: Colors.brown, fontWeight: FontWeight.bold),
  ),
  Text(
    'scholarship', // Third line
    style: TextStyle(color: Colors.brown, fontWeight: FontWeight.bold),
  ),
  Row(
    children: [
      Icon(Icons.verified_user, color: Colors.green), // Verification icon
      SizedBox(width: 1.0),
      Text(
        'Just 15 minutes', // Text next to verification icon style:
        TextStyle(color: Colors.brown),
      ),
    ],
  ),
  Row(
    children: [
      Icon(Icons.verified_user, color: Colors.green), // Verification icon
      SizedBox(width: 3.0),
      Text(
        '15 quick questions', // Text next to verification icon style:
        TextStyle(color: Colors.brown),
      ),
    ],
  ),
  SizedBox(height: 40),
 ],
),
```

```
                SizedBox(
                  width: 150, // Adjust the width as needed height:
                  150, // Adjust the height as needed
                  child: Image.asset("assets/images/student.png"), // Replace with your
image
                ),
              ],
            ),
          ),
        ],
      ),
      Positioned(
        bottom: 10.0,
        left: 10.0,
        right: 10.0,
        child: ElevatedButton(
          onPressed: () {
            // Implement action for starting the test
          },
          child: const Text('Start Test'),
        ),
      ),
    ],
  ),
  const SizedBox(height: 3.0),
  const Padding(
    padding: EdgeInsets.all(4.0),
    child: Text(
      'Subject-wise Classes:',
      style: TextStyle(
      fontSize: 18.0,
        fontWeight: FontWeight.bold,
      ),
    ),
  ),
  const SizedBox(height: 1.0),
  ListTile(
```

```
        title: const Text( 'Physics
          (38 Topics)',
          style: TextStyle(color: Colors.indigoAccent, fontWeight: FontWeight.bold),
        ),
        onTap: () {
          // Implement action for opening Physics lectures
        },
      ),
      ListTile(
        title: const Text(
          'Chemistry (5 Topics)',
          style: TextStyle(color: Colors.indigoAccent, fontWeight: FontWeight.bold),
        ),
        onTap: () {
          // Implement action for opening Chemistry lectures
        },
      ),
      ListTile(
        title: const Text( 'Mathematics
          (41 Topics)',
          style: TextStyle(color: Colors.indigoAccent, fontWeight: FontWeight.bold),
        ),
        onTap: () {
          // Implement action for opening Mathematics lectures
        },
      ),
    ],
  ),
  bottomNavigationBar: BottomNavigationBar(
    items: const [
      BottomNavigationBarItem(
        icon: Icon(Icons.home), label:
        'Home',
        backgroundColor: Colors.grey,
      ),
      BottomNavigationBarItem(
        icon: Icon(Icons.apartment),
```

```
          label: 'Centres',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.storefront_sharp),//arrow_forward_ios
          label: 'Store',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.account_circle),
          label: 'Me',
        ),
      ],
      onTap: (int index) {
        switch (index) { case
        1:
          Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => Centres()), // Navigate to Centres page
          );
          break;
        // Add cases for other tabs if needed
        }
      },
    ),
  );
 }
}
```

OUTPUT:



CONCLUSION: Thus understood the use of icons, images and font widgets in flutter. Implemented icons, images and fonts in my flutter application.

MAD & PWA Lab

Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	67
Name	Anmol Vaswani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	13

Name: - Anmol Vaswani
Div: -D15A
Roll No.: - 67

Experiment-4

Aim: To create an interactive Form using a form widget.

Theory: -

Form Widget:

- In Flutter, a Form widget is used to group multiple form fields together.
- It provides functionality for managing form state, validation, and submission.
- The Form widget keeps track of the form's state using a Form State object.

Form Fields:

- Form fields represent input elements within the form, such as text fields, checkboxes, radiobuttons, dropdowns, etc.
- Each form field typically has properties for input validation and handling user input changes.

Form State Management:

- The Form State object is responsible for managing the state of a form.
- It provides methods for validating the form, saving form data, resetting the form, and more.

User Interaction:

- User interaction with the form involves entering data into form fields, interacting with input elements, and submitting the form.

The Code is of login_page.dart:

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

import 'package:unacademy_app/home_page.dart';

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

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

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

  final List<String> countries = [
  'AF Afghanistan - +93',
    'AM Armenia - +374',
    'AZ Azerbaijan -+994',
    'BH Bahrain -+973',
    'BD Bangladesh - +880',
```

'BT Bhutan - +975',
'IO British Indian Ocean Territory - +246',
'BN Brunei - +673',
'KH Cambodia - +855',
'CN China - +86',
'CY Cyprus -+357',
'EG Egypt - +20',
'GE Georgia - +995',
'HK Hong Kong -+852',
'IN India - +91',
'ID Indonesia - +62',
'IR Iran - +98',
'IQ Iraq - +964',
'IL Israel -+972',
'JP Japan - +81',
'JO Jordan - +962',
'KZ Kazakhstan - +7',
'KP North Korea -+850',
'KR South Korea - +82',
'KW Kuwait - +965',
'KG Kyrgyzstan -+996',
'LA Laos - +856',
'LB Lebanon - +961',
'MO Macau - +853',
'MY Malaysia - +60',
'MV Maldives -+960',
'MN Mongolia +976',
'MM Myanmar - +95',
'NP Nepal - +977',
'OM Oman - +968',
'PK Pakistan - +92',
'PS State of Palestine - +970',
'PH Philippines - +63',
'QA Qatar - +974',
'RU Russia - +7',
'SA Saudi Arabia -+966',
'SG Singapore - +65',
'LK Sri Lanka -+94',
'SY Syria - +963',
'TW Taiwan -+886',
'TJ Tajikistan -+992',
'TH Thailand- +66',

```
  'TL Timor-Leste -+670',
  'TR Turkey - +90',
  'TM Turkmenistan - +993',
  'AE United Arab Emirates -+971',
  'UZ Uzbekistan - +998',
  'VN Vietnam -+84',
  'YE Yemen - +967',
  'AB Abkhazia - +7 840',
  'CX Christmas Island - +61',
  'CC Cocos (Keeling) Islands -
  +61','AZ Nagorno-Karabakh -
  +374',
  'CY Northern Cyprus - +90 392',
  'GE South Ossetia - +995 34',
 ];

 @override
 Widget build(BuildContext context)
  {return Scaffold(
   body: Padding(
    padding: const EdgeInsets.all(16.0),
    child: Form(
     key:
     _formKey,
     child: Stack(
     children: [
        Column(
         crossAxisAlignment:
         CrossAxisAlignment.start,children: [
          // Upper Logo and Heading
          Column(
           crossAxisAlignment:
           CrossAxisAlignment.start,children: [
            Image.asset('assets/images/your_logo.png', height: 60, width: 60),
            const SizedBox(height: 16),
            RichText(
             text: TextSpan(
              style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold, color:
              Colors.black),children: [
               TextSpan(text: 'Welcome to\n', style: TextStyle(fontSize: 25,
fontWeight:FontWeight.bold, color: Colors.black)),
               TextSpan(text: 'Unacademy\n', style: TextStyle(fontSize: 25,
fontWeight:FontWeight.bold, color: Colors.black)),
               TextSpan(text: 'Over', style: TextStyle(fontSize: 15, color:
               Colors.grey)),TextSpan(text: '10 crore', style: TextStyle(fontSize: 15,
               fontWeight:
FontWeight.bold, color: Colors.green)), // Highlight "10 crore" in green
```

```
                    TextSpan(text: ' learners trust\n', style: TextStyle(fontSize: 15,
color:Colors.grey)), // Grey color for "learners trust"
                    TextSpan(text: 'us for online and offline\n', style: TextStyle(fontSize: 15,
color:Colors.grey)), // Grey color for "us for online and offline"
                    TextSpan(text: 'coaching', style: TextStyle(fontSize: 15, color: Colors.grey)),
//Grey color for "coaching"
                  ],
                ),
              ),
            ],
          ),

          const SizedBox(height: 16),

          // Country Selection
          DropdownButton<String>(
          value: _selectedCountry,
            onChanged: (String? newValue)
              {setState(() {
                _selectedCountry = newValue;
              });
            },
            items: countries.map<DropdownMenuItem<String>>((String value)
              {return DropdownMenuItem<String>(
                value: value,
                child:
                Text(value),
              );
            }).toList(),
            selectedItemBuilder: (BuildContext context) {
              return countries.map<Widget>((String item)
              {final String flag = item.split(' ')[0];
                final String countryName = item.substring(item.indexOf(' ') +
                1);return Row(
                  children: [
                    Text(flag)
                    ,
                    const SizedBox(width:
                    8),Text(countryName),
                  ],
                );
              }).toList();
            },
            hint: Text('Select your country'),
          ),

          const SizedBox(height: 16),
```

```
// Mobile Number
InputTextFormField(
  keyboardType: TextInputType.phone,
  decoration: InputDecoration(
    labelText: 'Enter your mobile number',
  ),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your mobile number';
    }
    if (value.length != 10) {
      return 'Mobile number must be exactly 10 digits';
    }
    if (!RegExp(r'^[0-9]+$').hasMatch(value))
      {return 'Please enter only numeric
      digits';
    }
    return null;
  },
  onSaved: (newValue) {
    _phoneNumber = newValue;
  },
),

const SizedBox(height: 16),

// Continue Button at the bottom
Expanded(
  child: Align(
    alignment:
    Alignment.bottomCenter,child:
    ElevatedButton(
      onPressed: () {
        if (_selectedCountry == null) {
          ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
              content: Text('Please select your country'),
              backgroundColor: Colors.red,
            ),
          );
        } else if (_formKey.currentState!.validate()) {
          _formKey.currentState!.save();
          // Handle continue button press
          // Navigate to the next screen or perform necessary actions
          Navigator.push(
            context,
            MaterialPageRoute(
```

```
                        builder: (context) =>
                          VerifyOTPPage(phoneNumber:
                          _phoneNumber ?? ",email: ",
                        ),
                      ),
                    );
                  }
                },
                child: Text('Continue'),
              ),
            ),
          )
        ],
      ),
      Positioned(
        top: 0,
        right: 0,
        child: InkWell(
          onTap: () {
          Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => LoginWithEmailPage()),
            );
          },
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Text(
              'LOGIN WITH EMAIL>',
              style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
            ),
          ),
        ),
      ),
    ],
  ),
 ),
),
),
);
 }
}

class LoginWithEmailPage extends StatefulWidget
 {@override
 _LoginWithEmailPageState createState() => _LoginWithEmailPageState();
}
class _LoginWithEmailPageState extends State<LoginWithEmailPage> {
 final TextEditingController _emailController = TextEditingController();
 final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
 bool _emailValidated = true;
```

```
 @override
Widget build(BuildContext context)
  {return Scaffold(
    appBar: AppBar(
      title: Text('Your email address'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Form(
        key:
        _formKey,
        child: Column(
          crossAxisAlignment:
          CrossAxisAlignment.start,children: [
            Text(
              'We\'ll send an OTP for
              verification',style:
              TextStyle(fontSize: 16),
            ),
            SizedBox(height: 16),
            TextFormField(
              controller: _emailController,
              keyboardType: TextInputType.emailAddress,
              decoration: InputDecoration(
                labelText: 'Enter your email address',
                errorText: _emailValidated ? null : 'This email address is invalid',
              ),
              validator: (value) {
                if (value == null || value.isEmpty)
                  {return 'This field is required';
                }
                if (!value.contains('@gmail.com'))
                  {setState(() {
                    _emailValidated = false;
                  });
                  return null;
                } else {
                  setState(() {
                    _emailValidated = true;
                  });
                  return null;
                }
              },
            ),
```

```
            SizedBox(height: 10),
            Row(
              mainAxisAlignment:
              MainAxisAlignment.center,children: [
                TextButton(
                  onPressed: () {
                    Navigator.pop(context);
                  },
                  child: Text('LOGIN WITH MOBILE NUMBER >'),
                ),
              ],
            ),
            SizedBox(height: 16),
            Expanded(
              child: Align(
                alignment:
                Alignment.bottomCenter,child:
                ElevatedButton(
                  onPressed: () {
                    if (_formKey.currentState!.validate()) {
                      _formKey.currentState!.save();
                      // Handle continue button press
                      // Navigate to the next screen or perform necessary actions
                      Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) =>
                            VerifyOTPPage(phoneNumber: '',
                            email: _emailController.text,
                          ),
                        ),
                      );
                    }
                  },
                  child: Text('Continue'),
                ),
              ),
            )
          ],
        ),
      ),
    ),
  );
 }
}

class VerifyOTPPage extends StatefulWidget {
  final String phoneNumber;
  final String email;
```

```
  VerifyOTPPage({required this.phoneNumber, required this.email});

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

class _VerifyOTPPageState extends State<VerifyOTPPage> {
  TextEditingController _otpController =
  TextEditingController();bool _isResendingOTP = true;
  int _resendTimeout = 50;

  @override
  void initState() {
    super.initState();
    // Start countdown for resend OTP
    startResendTimer();
  }
  void startResendTimer() {
    const oneSec = const Duration(seconds: 1);
    Timer.periodic(oneSec, (Timer timer) {
    setState(() {
        if (_resendTimeout < 1) {
          _isResendingOTP = true;
          timer.cancel();
        } else {
          _resendTimeout--;
        }
     });
    });
  }
  void resendOTP() {
    // Implement OTP resend logic here
    // You can call your API or perform any necessary actions
    setState(() {
      _isResendingOTP = true;
      _resendTimeout = 50; // Reset resend timeout
    });
    startResendTimer(); // Start countdown again
  }
  @override
  Widget build(BuildContext context)
    {return Scaffold(
      appBar: AppBar(
        title: Text('Verify OTP'),
```

```
    body: Padding(
     padding: const EdgeInsets.all(16.0),
     child: Column(
       crossAxisAlignment:
       CrossAxisAlignment.start,children: [
         Text(
           'We\'ve sent an OTP to ${widget.phoneNumber.isNotEmpty ? widget.phoneNumber
:widget.email}',
           style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
         ),
         SizedBox(height: 16),
         Row(
           mainAxisAlignment:
           MainAxisAlignment.spaceAround,children:
           List.generate(
             6,
                (index) =>
               SizedBox(width: 40,
               child:
                 TextFormField(
                 maxLength: 1,
                 keyboardType: TextInputType.number,
                 textAlign: TextAlign.center,
                 decoration: InputDecoration(
                   counterText: "",
                   enabledBorder:
                     OutlineInputBorder(borderSide:
                     BorderSide(width: 1),
                   ),
                   focusedBorder:
                     OutlineInputBorder(borderSide:
                     BorderSide(width: 1),
                   ),
                 ),
                 onChanged: (value) {
                   // Automatically move to the next box when a digit is
                   enteredif (value.isNotEmpty) {
                     if (index < 5) {
                       FocusScope.of(context).nextFocus();
                     } else {
                       FocusScope.of(context).unfocus();
                     }
                   }
                 },
               ),
             ),
           ),
         ),
```

```
        SizedBox(height: 16),
        Row(
          mainAxisAlignment:
          MainAxisAlignment.center,children: [
            Text(
              _isResendingOTP ? 'Resend OTP in $_resendTimeout s' : '',
              style: TextStyle(color: Colors.blue),
            ),
            _isResendingOTP
                ? SizedBox()
                : TextButton(
              onPressed: () {
               resendOTP();
              },
              child: Text('Resend OTP'),
            ),
          ],
        ),
        SizedBox(height: 12),
        InkWell(
          onTap: () {
            // Open email client to write to support
            // You can use a package like url_launcher to achieve this
          },
          child: Center(
            child: Text(
              'Having trouble? Write to us at help@unacademy.com',
              style: TextStyle(fontWeight: FontWeight.bold, color: Colors.blue),
            ),
          ),
        ),
        SizedBox(height: 16),
        Expanded(
          child: Align(
            alignment:
            Alignment.bottomCenter,child:
            ElevatedButton(
                onPressed: () {
                  Navigator.push( // Navigating to BookVisit
                    pagecontext,
                    MaterialPageRoute(builder: (context) => const HomePage()),
                  );
                },
                child: Text('Verify'),
            ),
          ),
        ),

      ],
    ),
  ),
```

```
    );
  }
}
```

The Code is of VerifyOTPPage.dart:

```
import 'dart:async'; // Add this import statement

import 'package:flutter/material.dart';
import 'package:unacademy_app/home_page.dart';

class VerifyOTPPage extends StatefulWidget {
  final String phoneNumber;
  final String email;

  VerifyOTPPage({required this.phoneNumber, required this.email});

  @override
  _VerifyOTPPageState createState() => _VerifyOTPPageState();
}
class _VerifyOTPPageState extends State<VerifyOTPPage> {
  TextEditingController _otpController =
  TextEditingController();bool _isResendingOTP = false;
  int _resendTimeout = 50;
  bool _isPhoneVerification = false;

  @override
    void initState() { super.initState();
    // Start countdown for resend OTP
    startResendTimer();
  }

  void startResendTimer() {
    const oneSec = const Duration(seconds: 1);
    Timer.periodic(oneSec, (Timer timer) {
    setState(() {
        if (_resendTimeout < 1) {
          _isResendingOTP = false;
          timer.cancel();
        } else {
          _resendTimeout--;
        }
      });
    });
  }

  void resendOTP() {
    // Implement OTP resend logic here
    // You can call your API or perform any necessary actions
    setState(() {
      _isResendingOTP = true;
      _resendTimeout = 50; // Reset resend timeout
```

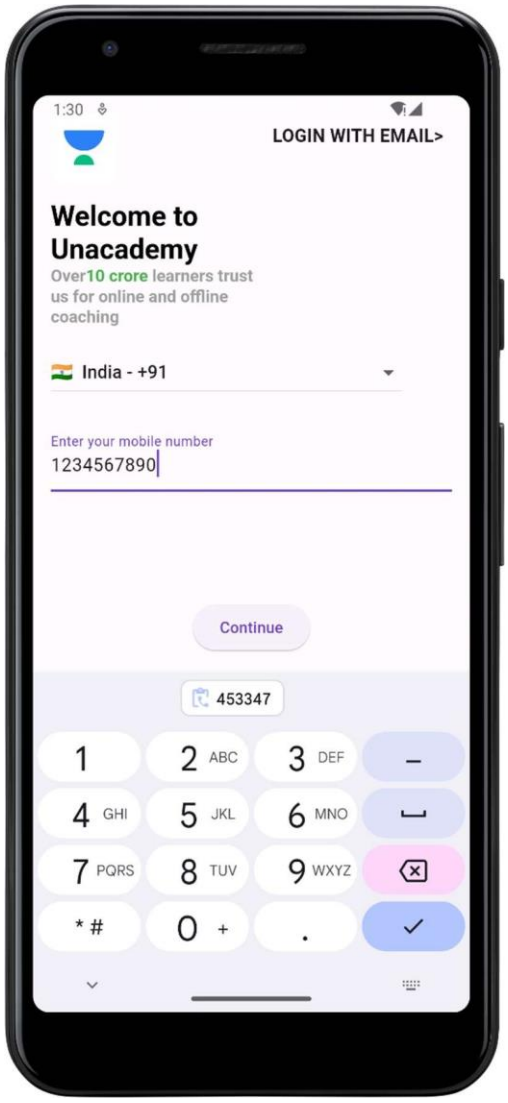
```
  });
  startResendTimer(); // Start countdown again
}
void verifyOTP() {
  // Implement OTP verification logic here
  // Check if the entered OTP is correct or not
  // For demonstration, let's assume the correct OTP is '453347'
  String correctOTP = '453347';
  String enteredOTP =
  _otpController.text;if (enteredOTP ==
  correctOTP) {
    // OTP verified successfully
    ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
        content: Text('OTP verified successfully. You are now a verified
        user.'),backgroundColor: Colors.green,
      ),
    );
    // Navigate to HomePage
    print("Navigating to
    HomePage");
    Navigator.pushAndRemoveUntil
    (context,
      MaterialPageRoute(builder: (context) => HomePage()),
          (route) => false, // Remove all routes until the new route is pushed
    );
  } else {
    // Incorrect OTP
    ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
        content: Text('Incorrect OTP. Please try again.'),
        backgroundColor: Colors.red,
      ),
    );
  }
}


@override
Widget build(BuildContext context)
  {return Scaffold(
    appBar: AppBar(
      title: Text('Verify OTP'),
    ),
```

```
body: Padding(
  padding: const EdgeInsets.all(16.0),
  child: Column(
    crossAxisAlignment:
    CrossAxisAlignment.start,children: [
      Text(
        _isPhoneVerification
            ? 'We\'ve sent an OTP to your phone number ${widget.phoneNumber}'
            : 'We\'ve sent an OTP to your email ${widget.email}',
        style: TextStyle(fontSize: 16, fontWeight:
        FontWeight.bold),
      ),
      SizedBox(height: 16),
      Row(
        mainAxisAlignment:
        MainAxisAlignment.spaceAround,children:
        List.generate(
          6,
              (index) =>
            SizedBox(width: 40,
            child: TextFormField(
              controller:
              _otpController,
              maxLength: 1,
              keyboardType: TextInputType.number,
              textAlign: TextAlign.center,
              decoration: InputDecoration(
                counterText: "",
                enabledBorder:
                  OutlineInputBorder(borderSide:
                  BorderSide(width: 1),
                ),
                focusedBorder:
                  OutlineInputBorder(borderSide:
                  BorderSide(width: 1),
                ),
              ),
              onChanged: (value) {
                // Automatically move to the next box when a digit is
                enteredif (value.isNotEmpty) {
                  if (index < 5) {
                    FocusScope.of(context).nextFocus();
                  } else {
                    FocusScope.of(context).unfocus();
                  }
                }
              },
            ),
          ),
        ),
```

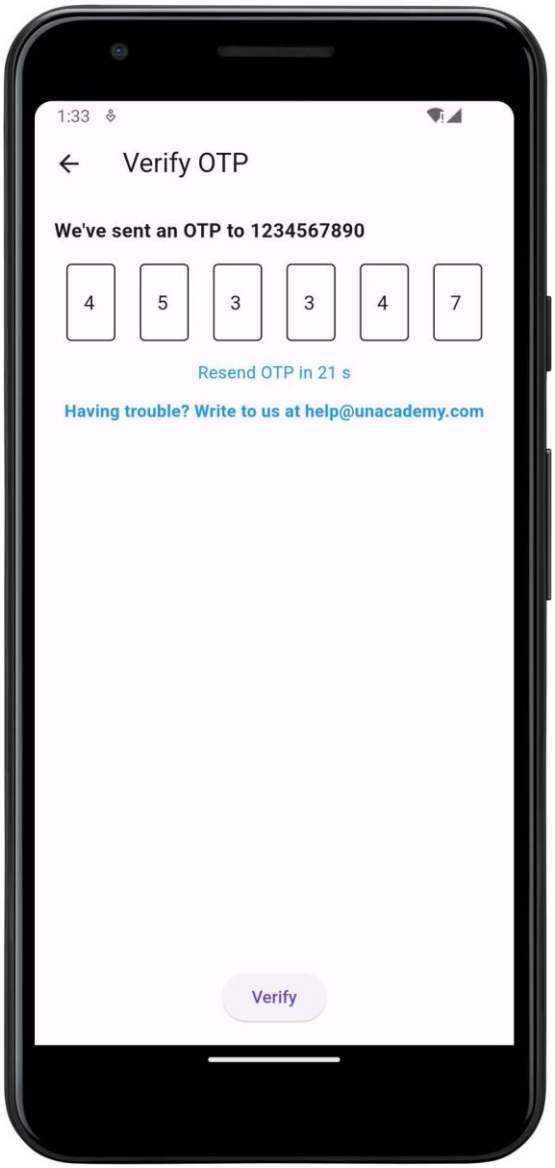
```
          ),
          SizedBox(height: 16),
          Row(
            mainAxisAlignment:
            MainAxisAlignment.center,children: [
              Text(
                _isResendingOTP ? 'Resend OTP in $_resendTimeout s' : 'Resend OTP',
                style: TextStyle(color: Colors.blue),
              ),
              _isResendingOTP
                  ? SizedBox()
                  : TextButton(
                onPressed: () {
                 resendOTP();
                },
                child: Text('Resend OTP'),
              ),
            ],
          ),
          SizedBox(height: 16),
          InkWell(
            onTap: () {
              // Open email client to write to support
              // You can use a package like url_launcher to achieve this
            },
            child: Center(
              child: Text(
                'Having trouble? Write to us at help@unacademy.com',
                style: TextStyle(fontWeight: FontWeight.bold, color: Colors.blue),
              ),
            ),
          ),
          SizedBox(height: 16),
          Expanded(
            child: Align(
              alignment:
              Alignment.bottomCenter,child:
              ElevatedButton(
                onPressed: () {
                  verifyOTP(); // Call verifyOTP function when button is pressed
                },
                child: Text('Verify'),
              ),
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```

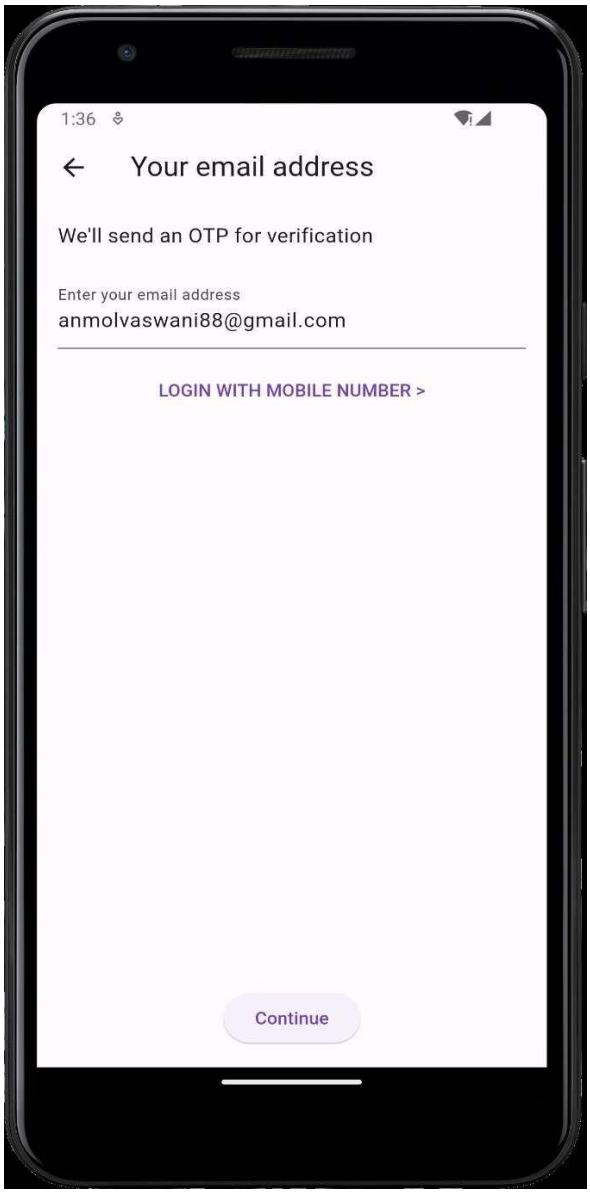
Project Title: Unacademy Clone/PWA- Counter App **Roll No. 67**

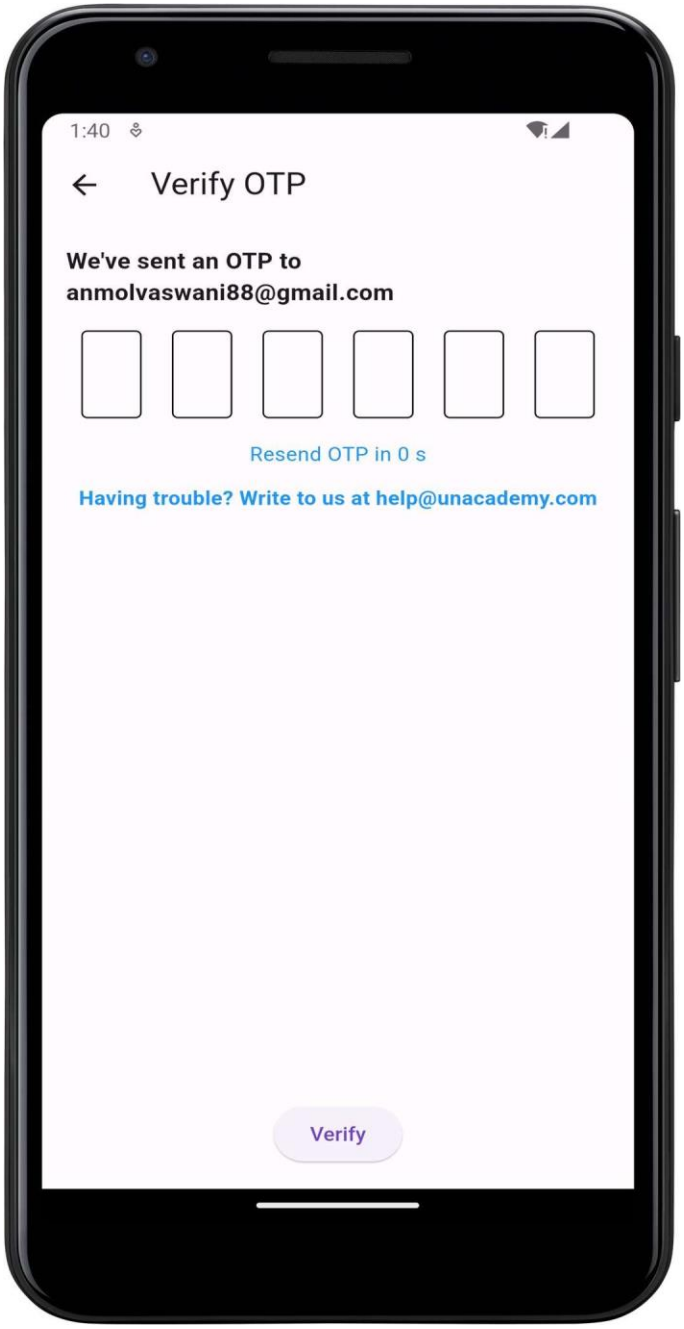
OUTPUT: -



Project Title: Unacademy Clone/PWA- Counter App **Roll No. 67**




1:36
Your email address
We'll send an OTP for verification
Enter your email address
anmolvaswani88@gmail.com
LOGIN WITH MOBILE NUMBER >
Continue



Conclusion: - We have successfully implemented the form in Flutter UI using widgets.

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	67
Name	Anmol Vaswani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	12

Name: - Anmol Vaswani
Div: - D15A
Roll No.: - 67

Experiment 5

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

Theory: - In Flutter, navigation, routing, and gestures play vital roles in creating engaging and intuitive user experiences.

Navigation and Routing: Navigation refers to moving between different screens or pages within an app. Routing is the mechanism that defines the paths and transitions between these screens. In Flutter, navigation and routing are managed using a Navigator widget.

- Navigator Widget: Each Flutter app has a Navigator widget that manages a stack of Route objects. These Route objects represent individual screens or pages in the app.

- Navigation Stack: The Navigator maintains a stack of routes. Pushing a new route onto the stack navigates to a new screen, and popping a route removes the topmost route from the stack, navigating back to the previous screen.

- Routes: Routes are represented by instances of PageRoute or its subclasses like MaterialPageRoute. They define the content and behavior of each screen.

- Routing Methods: Flutter provides methods like Navigator.push, Navigator.pop, Navigator.pushNamed, and Navigator.popUntil to navigate between routes. Gestures: Gestures enable users to interact with the app using touch inputs like tapping, dragging, swiping, etc. Flutter provides a rich set of gesture recognizers that make it easy to handle various touch interactions.

- Gesture Detector: The GestureDetector widget is the primary way to add gesture recognition to widgets in Flutter. It listens for various touch events and provides callbacks for handling those events.

- Handling Gestures: When a gesture is detected, the corresponding callback (e.g., onTap, onLongPress, onPanUpdate, etc.) is invoked, allowing you to respond to user interactions accordingly.

- Gesture Widgets: Some widgets like ListView, GestureDetector, InkWell,

Draggable, Dismissible, etc., come with built-in gesture handling capabilities

The Code of Centres.dart:

```
import 'package:flutter/material.dart';
import  'package:unacademy_app/BookVisit.dart';
import 'BookVisit.dart'; // Importing the BookVisit.dart file
import 'DownloadBrochure.dart';
class Centres extends StatefulWidget {
  const Centres({super.key});

  @override
  _CentresState createState() => _CentresState();
}
class _CentresState extends State<Centres> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title:  const  SingleChildScrollView(
          scrollDirection: Axis.horizontal,
          child: Row(
            children: [
              Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Row(
                    children: [
                      Icon(Icons.location_on),
                      Text(
                        'Mumbai',
                        style: TextStyle(fontWeight: FontWeight.bold, fontSize: 25),
                      ),
                    ],
                  ),
                  SizedBox(height: 0.5),
                  Text(
                    'Unacademy Mumbai Center',
                    style: TextStyle(color: Colors.blue, fontSize: 20),
                  ),
                ],
              ),
            ],
          ),
        ),
      ),
      body: SingleChildScrollView(
        child: Column(
          children: [
            SizedBox(
              width: 800, // Adjust width as needed
              height: 225, // Adjust height as needed
              child:  Image.asset('assets/images/building.png'),
            ),
            SizedBox(
              width: 500, // Adjust width as needed
              height: 225, // Adjust height as needed
              child:  Image.asset('assets/images/Office.jpg'),
            ),
```

```
SizedBox(
  width: 800, // Adjust width as needed
  height: 225, // Adjust height as needed
  child: Image.asset('assets/images/closer.jpeg'),
),
const SizedBox(height: 15),
const Text(
  'Why Unacademy Centre?',
  style: TextStyle(fontSize: 29, fontWeight: FontWeight.bold),
),
const SizedBox(height: 2),
const Text(
  'Get an all-encompassing learning package ',
  textAlign: TextAlign.center,
),
const SizedBox(height: 2),
const Text(
  'that includes the best study material, ',
  textAlign: TextAlign.center,
),
const SizedBox(height: 2),
const Text(
  'library resources, advanced technology,',
  textAlign: TextAlign.center,
),
const SizedBox(height: 2),
const Text(
  'double-clearing sessions,1:1 mentorship and more',
  textAlign: TextAlign.center,
),
const SizedBox(height: 5),
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    ElevatedButton(
      onPressed: () {
        Navigator.push( // Navigating to BookVisit page
          context,
          MaterialPageRoute(builder: (context) => const BookVisit()),
        );
      },
      child: Text('Book a Visit'),
      style: ElevatedButton.styleFrom(
        primary: Colors.blue,
        onPrimary: Colors.white,
      ),
    ),
    const SizedBox(width: 20),
    ElevatedButton(
    onPressed: () {
        Navigator.push( // Navigating to BookVisit page
          context,
          MaterialPageRoute(builder: (context) =>  DownloadBrochure()),
        );// Add functionality for downloading brochure
      },
      style: ElevatedButton.styleFrom(
        primary: Colors.white,
        onPrimary: Colors.black,
        side: const BorderSide(color: Colors.black),
      ),
```

```
                child: const Text('See Brochure'),
              ),
            ],
          ),
          SizedBox(
            width: 1000, // Adjust width as needed
            height: 650, // Adjust height as needed
            child:  Image.asset('assets/images/service.jpeg'),
          ),
          SizedBox(
            width: 1000, // Adjust width as needed
            height: 1330, // Adjust height as needed
            child:  Image.asset('assets/images/rankers.jpeg'),
          ),
          SizedBox(
            width: 3000, // Adjust width as needed
            height: 350, // Adjust height as needed
            child:  Image.asset('assets/images/mumbai.jpeg'),
          ),
        ],
      ),
    ),
    bottomNavigationBar: BottomNavigationBar(
      items: const [
        BottomNavigationBarItem(
          icon: Icon(Icons.home),
          label: 'Home',
          backgroundColor: Colors.grey,
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.apartment),
          label: 'Centres',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.storefront_sharp),
          label: 'Store',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.account_circle),
          label: 'Me',
        ),
      ],
    ),
  );
 }
}
```

The code of BookVisit.dart:

```
import 'package:flutter/material.dart';
class BookVisit extends
StatefulWidget {const
BookVisit({super.key}); @override
 _BookVisitState createState() => _BookVisitState();
}
```

```
class _BookVisitState extends State<BookVisit> {
 TextEditingController nameController =
 TextEditingController();TextEditingController phoneController
 = TextEditingController();bool isSubmitted = false;
 @override
 Widget build(BuildContext
  context) {return Scaffold(
   appBar: AppBar(
    title: const Text('Book a Visit'),
   ),
   body: SingleChildScrollView(
    padding: const
    EdgeInsets.all(20.0),child:
    Column(
     crossAxisAlignment:
     CrossAxisAlignment.center,children: [
      const Text(
'Start your preparation at our Unacademy Centre',
style: TextStyle(fontSize: 20.0, fontWeight: FontWeight.bold),
      ),
      const SizedBox(height: 20.0),
      Image.asset(
'assets/images/building.png', // Replace with your image path
width: 800.0, // Adjust the width as needed
height: 205.0, // Adjust the height as needed
      ),
      const SizedBox(height: 20.0),
      const Text(
'Register with Unacademy',
style: TextStyle(fontSize: 18.0, fontWeight: FontWeight.bold),
      ),
      const SizedBox(height: 10.0),
      const Text(
'Personal details',
style: TextStyle(fontSize: 16.0),
      ),
      const SizedBox(height: 10.0),
      buildTextField('Name',  nameController),
      const SizedBox(height: 10.0),
      buildTextField('Phone No.',
      phoneController),const SizedBox(height:
      20.0), ElevatedButton(
onPressed: ()
{
        if (nameController.text.isNotEmpty && phoneController.text.isNotEmpty) {
         setState(() {
          isSubmitted = true;
         }
        );
```

```
            }
},
child: const Text('Submit'),
          ),
          const SizedBox(height:
          20.0),if (isSubmitted)
const Text(
            'Your response has been
            noted.',style:
            TextStyle(fontSize: 16.0),
),
         ],
        ),
      ),
    );
  }
  Widget buildTextField(String labelText, TextEditingController controller) {
   return TextFormField(
    controller: controller,
    decoration:
    InputDecoration(
    labelText: labelText,
     border: const OutlineInputBorder(),
    ),
   );
  }
 }
```

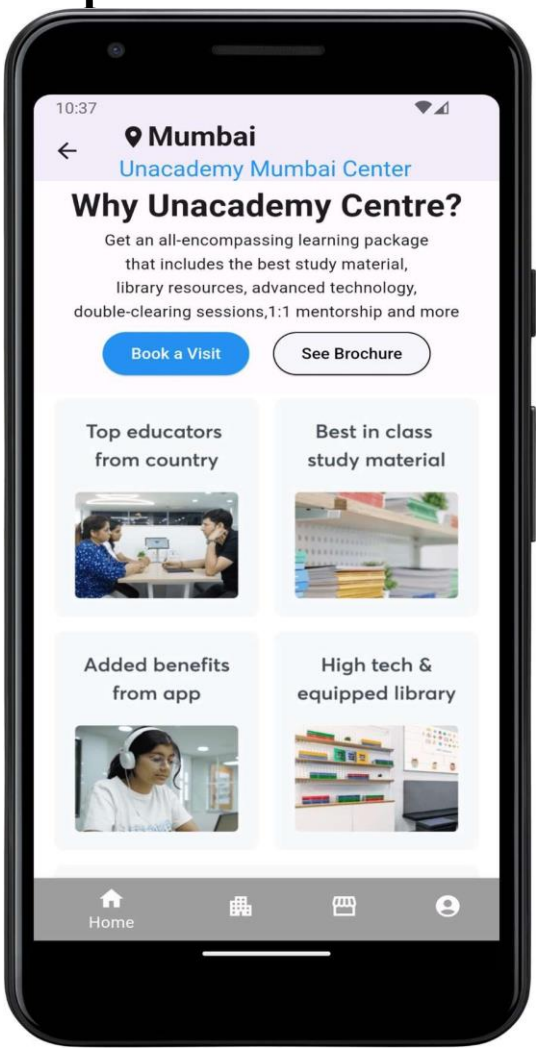
The code of DownloadBrochure.dart:

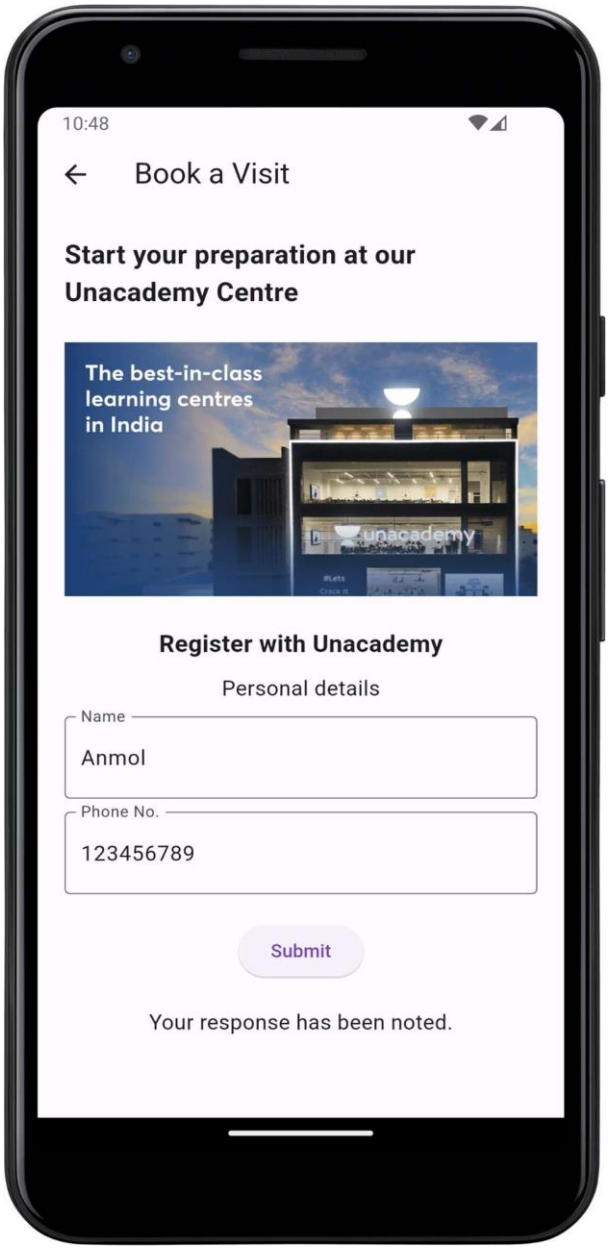
```
import 'package:flutter/material.dart';
class DownloadBrochure extends StatelessWidget
 {@override
  Widget build(BuildContext
   context) {return Scaffold(
    appBar: AppBar(
     title: Text('Unacademy Brochure'),
    ),
    body:
     SingleChildScrollView(
     padding:
     EdgeInsets.all(10.0),child:
     Column(
      crossAxisAlignment:
      CrossAxisAlignment.center,children: [
SizedBox(height: 1.0),
```

```
            Image.asset(
  'assets/images/Page1.png', // Replace with your image pathwidth:
  900.0, // Adjust the width as needed
  height: 540.0, // Adjust the height as needed
            ),
            SizedBox(height: 1.0),
            Image.asset(
  'assets/images/Page2.png', // Replace with your image pathwidth:
  900.0, // Adjust the width as needed
  height: 540.0, // Adjust the height as needed
            ),
            SizedBox(height: 1.0),
            Image.asset(
  'assets/images/Page3.png', // Replace with your image pathwidth:
  900.0, // Adjust the width as needed
  height: 540.0, // Adjust the height as needed
            ),
            SizedBox(height: 1.0),
            Image.asset(
  'assets/images/Page4.png', // Replace with your image pathwidth:
  900.0, // Adjust the width as needed
  height: 540.0, // Adjust the height as needed
            ),
            SizedBox(height: 1.0),
            Image.asset(
  'assets/images/Page5.png', // Replace with your image pathwidth:
  900.0, // Adjust the width as needed
  height: 540.0, // Adjust the height as needed
            ),
            SizedBox(height: 1.0),
            Image.asset(
  'assets/images/Page6.png', // Replace with your image pathwidth:
  900.0, // Adjust the width as needed
  height: 540.0, // Adjust the height as needed
            ),
            SizedBox(height: 1.0),
            Image.asset(
  'assets/images/Page7.png', // Replace with your image pathwidth:
  900.0, // Adjust the width as needed
  height: 540.0, // Adjust the height as needed
            ),
            SizedBox(height: 1.0),
            Image.asset(
  'assets/images/Page8.png', // Replace with your image pathwidth:
  900.0, // Adjust the width as needed
  height: 540.0, // Adjust the height as needed
  ),
```

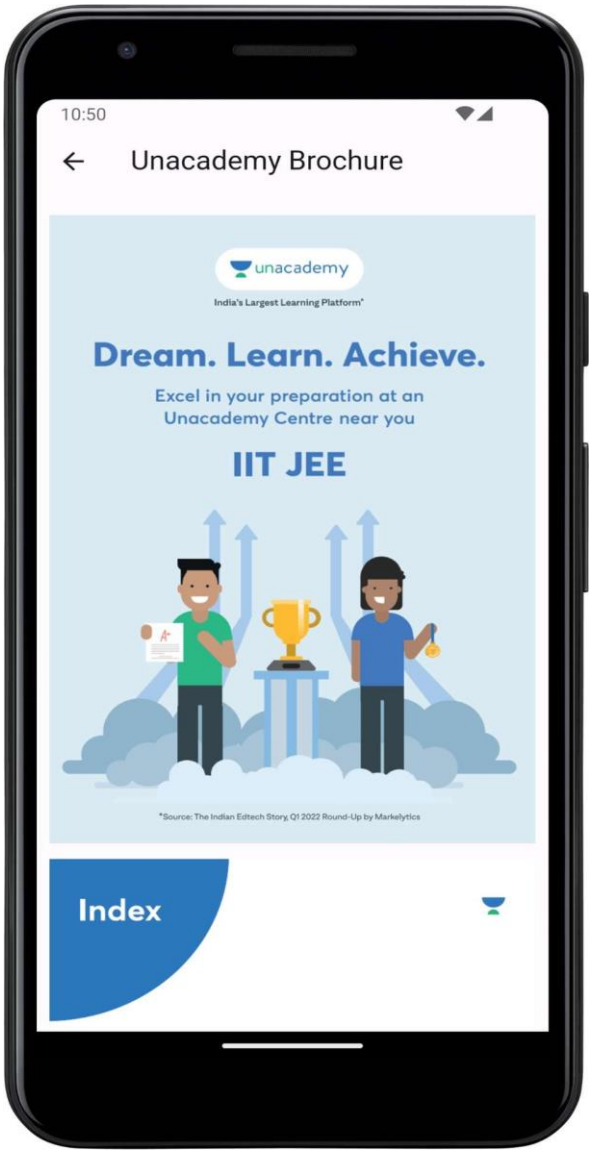
```
            SizedBox(height: 1.0),
            Image.asset(
  'assets/images/Page9.png', // Replace with your image pathwidth:
  900.0, // Adjust the width as needed
  height: 540.0, // Adjust the height as needed
            ),
            SizedBox(height: 1.0),
            Image.asset(
  'assets/images/Page10.png', // Replace with your image pathwidth:
  900.0, // Adjust the width as needed
  height: 540.0, // Adjust the height as needed
            ),
          ],
        ),
      ),
    );
  }
 }
```

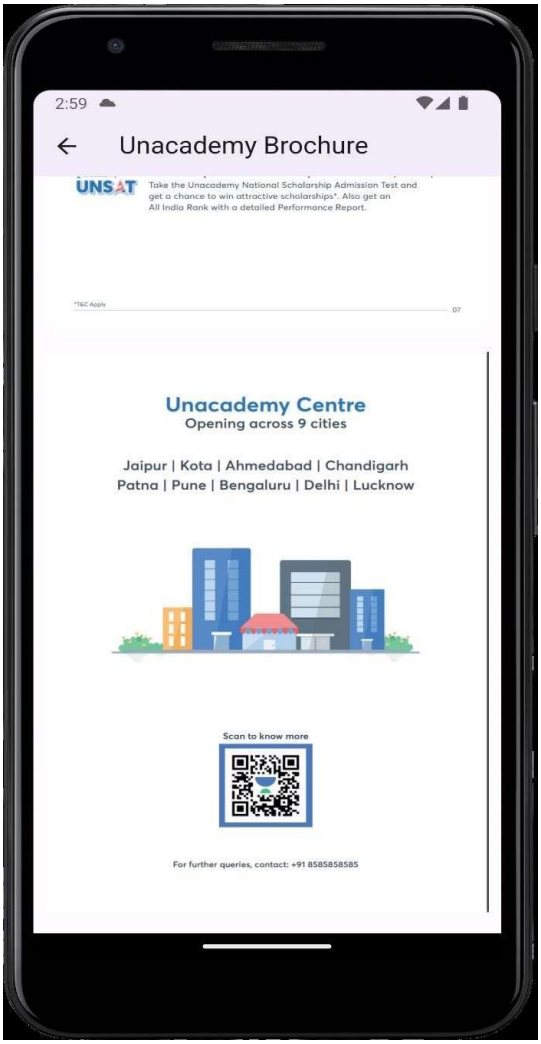
Output:-


10:37
Mumbai
Unacademy Mumbai Center
Why Unacademy Centre?
Get an all-encompassing learning package
that includes the best study material,
library resources, advanced technology,
double-clearing sessions,1:1 mentorship and more
Book a Visit
See Brochure
Top educators
from country
Best in class
study material
Added benefits
from app
High tech &
equipped library
Home


10:48
Book a Visit
Start your preparation at our Unacademy Centre
The best-in-class learning centres in India
unacademy
Register with Unacademy
Personal details
Name
Anmol
Phone No.
123456789
Submit
Your response has been noted.

Project Title: Unacademy Clone/PWA- Counter App **Roll No. 67**





Conclusion: - Thus, we studied navigation, routing and gesturefeatures in Flutter Application.

MAD & PWA Lab

Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	67
Name	Anmol Vaswani
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:	12

Name: - Anmol Vaswani
Div: - D15A
Roll No.: -67

Experiment-6

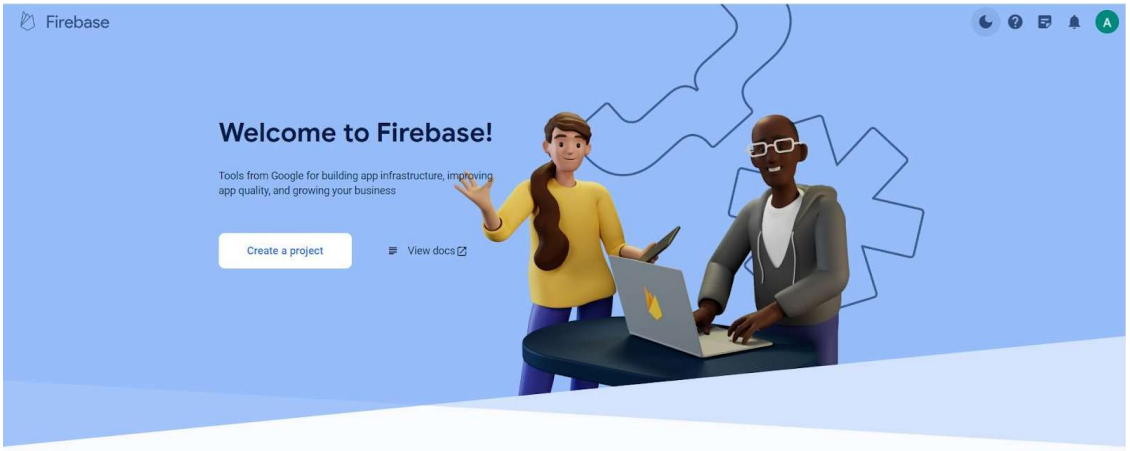
Aim: - To connect Flutter UI with Firebase Database.

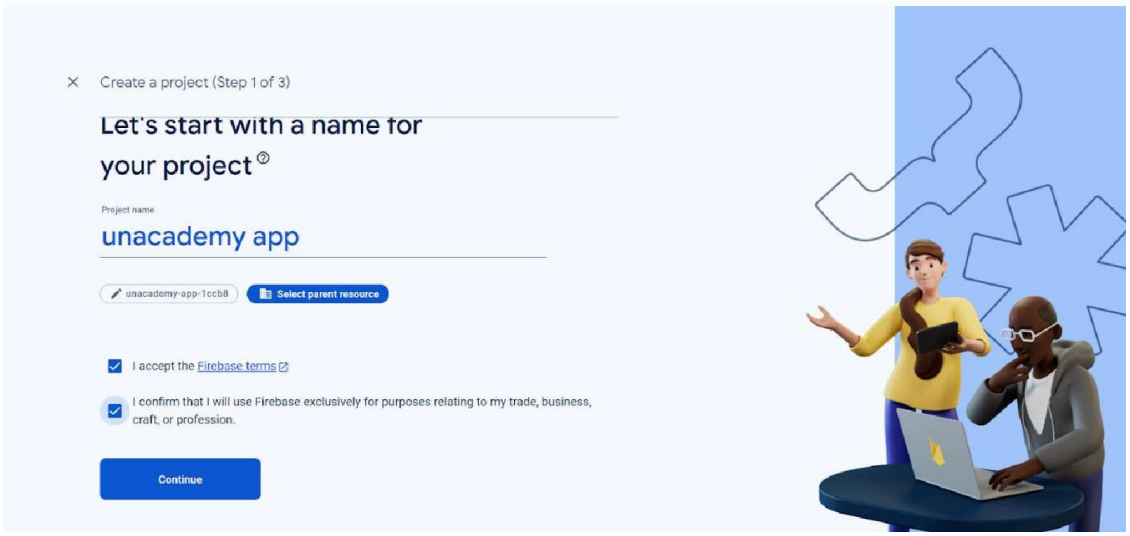
Theory: -

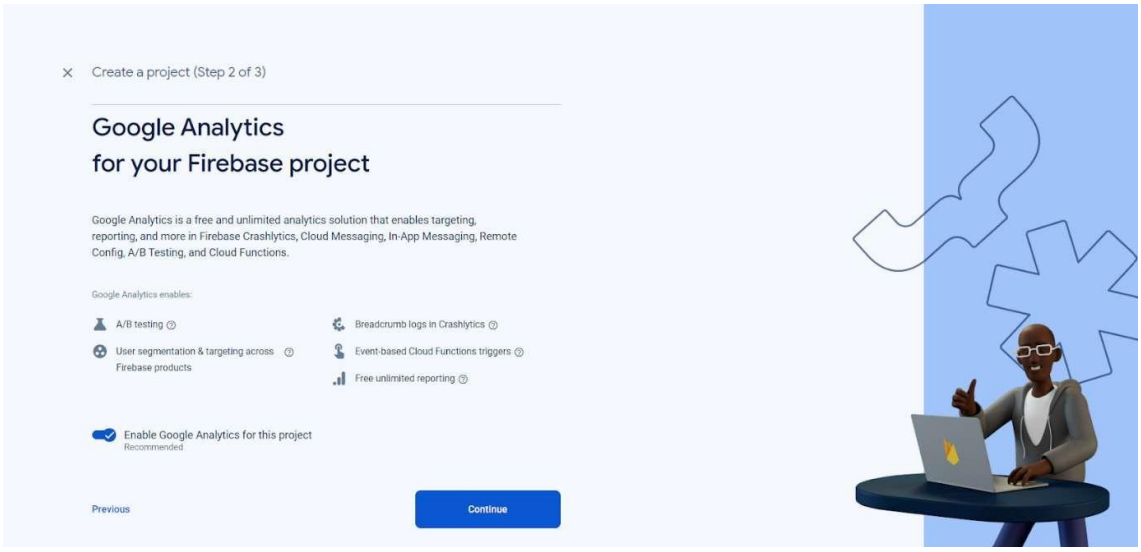
Prerequisites:
To complete this tutorial, you will need:

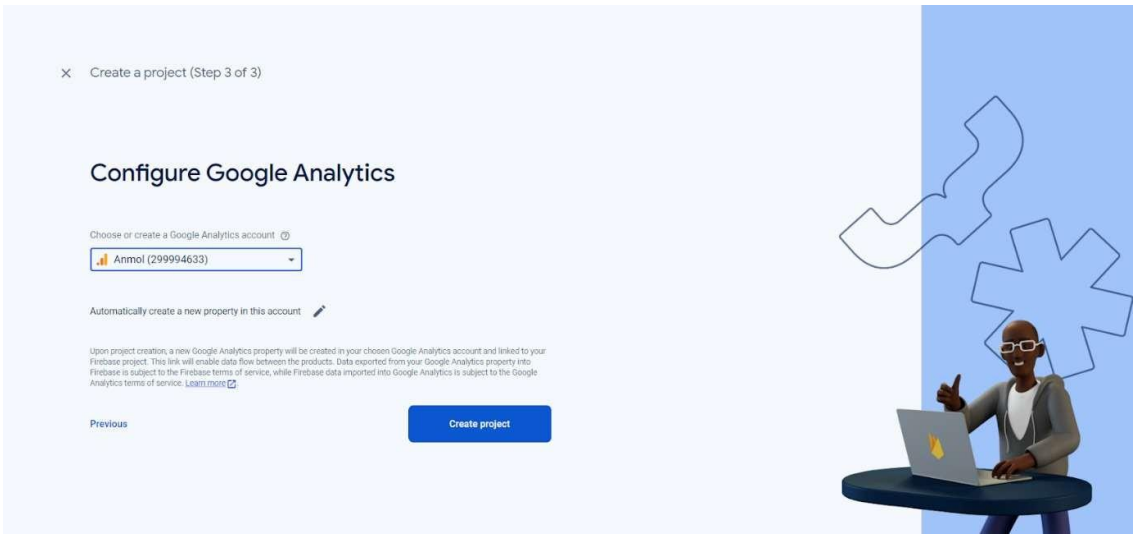
- A Google account to use Firebase.
- Developing for iOS will require Xcode.
- To download and install Flutter.
- To download and install Android Studio and Visual Studio Code.
- It is recommended to install plugins for your code editor:

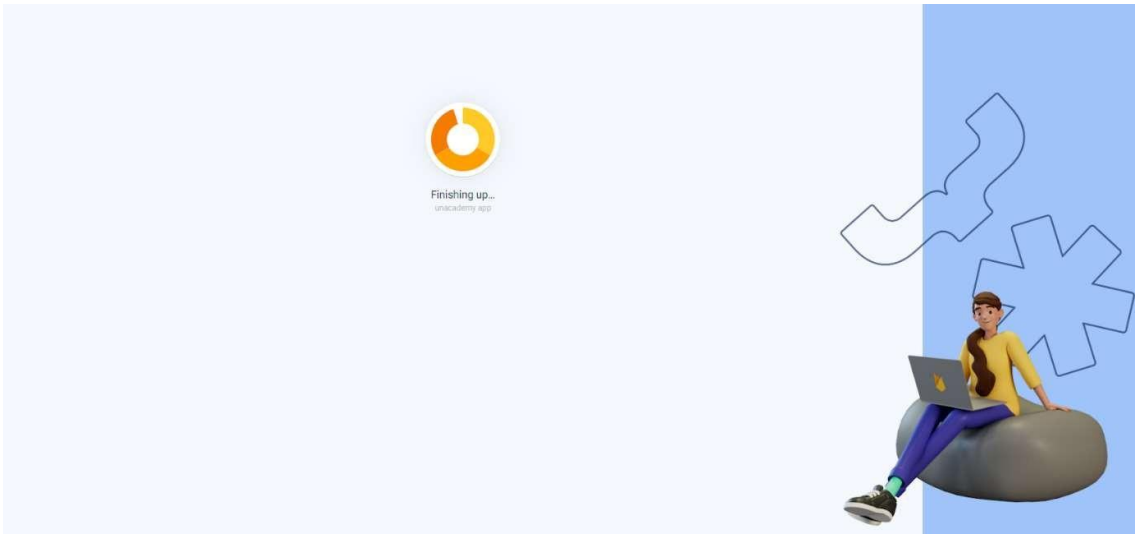
  - Flutter and Dart plugins installed for Android Studio.
  - Flutter extension installed for Visual Studio Code.

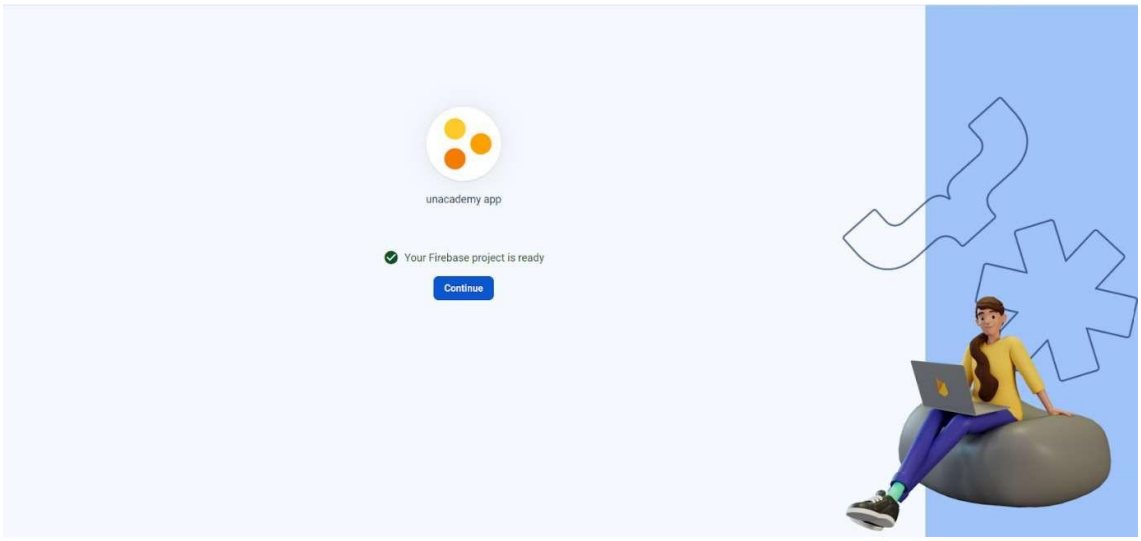


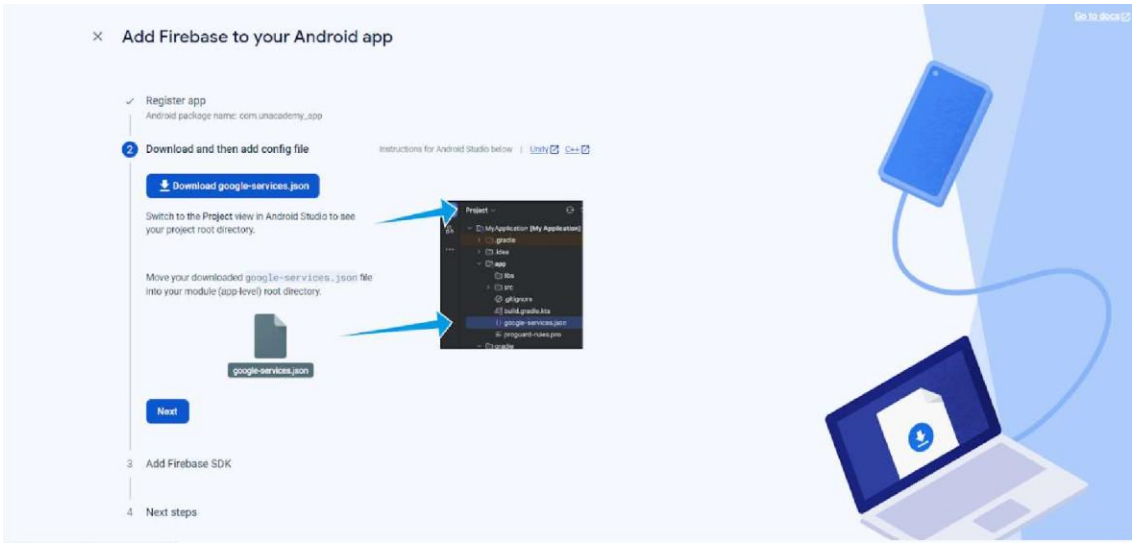


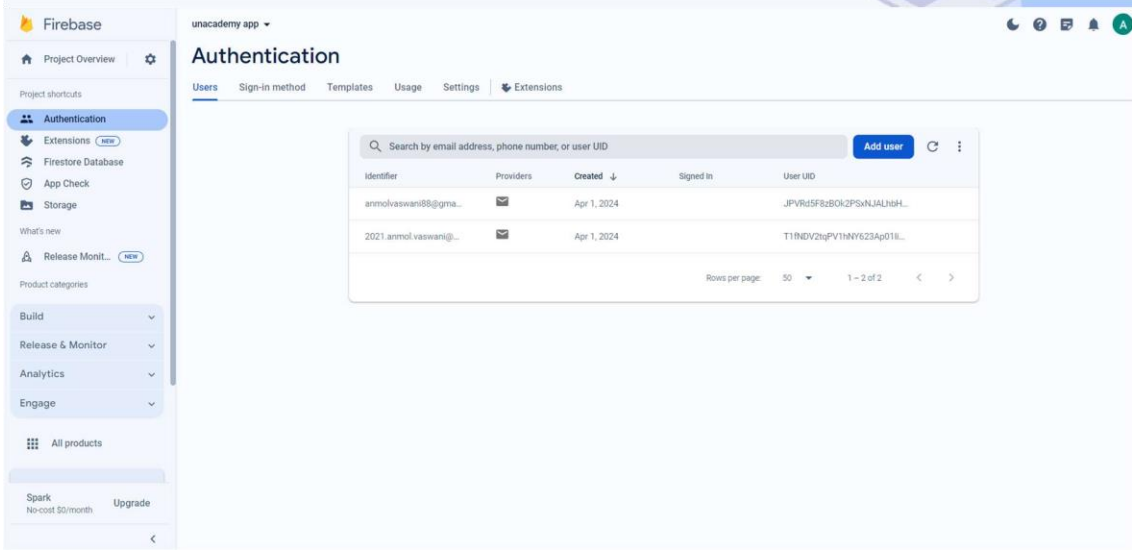

Create a project (Step 2 of 3)
Google Analytics
for your Firebase project
Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, and Cloud Functions.
Google Analytics enables:
A/B testing
User segmentation & targeting across Firebase products
Breadcrumb logs in Crashlytics
Event-based Cloud Functions triggers
Free unlimited reporting
Enable Google Analytics for this project
Recommended
Previous
Continue


Create a project (Step 3 of 3)
Configure Google Analytics
Choose or create a Google Analytics account
Anmol (299994633)
Automatically create a new property in this account
Upon project creation, a new Google Analytics property will be created in your chosen Google Analytics account and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. Learn more
Previous
Create project


Finishing up...
unacademy app







Conclusion: - Thus, we have connected firebase to our flutter UI.

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.	67
Name	Anmol Vaswani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	15

Name: - Anmol Vaswani
Div: - D15A
Roll No.: -67

PWA Experiment - 7

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

Theory -

Regular Web App

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

Progressive Web App

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

Difference between PWAs vs. Regular Web Apps:

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

1. Native Experience

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

2. Ease of Access

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

3. Faster Services

PWAs can cache the data and serve the user with text stylesheets, images, and other web content even before the page loads completely. This lowers the waiting time for the end-users and helps the brands improve the user engagement and retention rate, which eventually adds value to their business.

4. Engaging Approach

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

5. Updated Real-Time Data Access

Another plus point of PWAs is that these apps get updated on their own. They do not demand the end-users to go to the App Store or other such platforms to download the update and wait until installed.

In this app type, the web app developers can push the live update from the server, which reaches the apps residing on the user's devices automatically. Therefore, it is easier for the mobile app developer to provide the best of the updated functionalities and services to the end-users without forcing them to update their app.

6. Discoverable

PWAs reside in web browsers. This implies higher chances of optimizing them as per the Search Engine Optimization (SEO) criteria and improving the Google rankings like that in websites and other web apps.

7. Lower Development Cost

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

<u>Pros and cons of the Progressive Web App</u>

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

```
{
    "name":"PWA Tutorial",
    "short_name":"PWA",
    "start_url":"index.html",
    "display":"standalone",
    "background_color":"#5900b3",
    "theme_color":"black",
    "scope": ".",
    "description":"Counter App",
    "icons":[
    {
```

```
    "src":"images/two_192.png",
    "sizes":"192x192",
    "type":"image/png"
    },
    {
    "src":"images/one_512.png",
    "sizes":"512x512",
    "type":"image/png"
    }
]
}
```

index.html

```
<html>
    <head><link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/normalize/8.0.1/normalize
.css">
        <link rel="stylesheet" href="index.css">
        <link rel="manifest" href="/manifest.json">
    </head>
    <body>
        <h1>People entered:</h1>
        <h2 id="count-el">0</h2>
        <button id="increment-btn"
onclick="increment()">INCREMENT</button>
        <button id="decrement-btn"
onclick="decrement()">DECREMENT</button>
        <button id="save-btn" onclick="save()">SAVE</button>
        <p id="save-el">Previous Entries:</p>
        <script src="index.js"></script>
    </body>
<script>
  window.addEventListener('load', () => {
    registerSW();
  });
  async function registerSW() {
    if ('serviceWorker' in navigator) {
      try {
        await navigator
              .serviceWorker
              .register('serviceworker.js');
      }
      catch (e) {
```

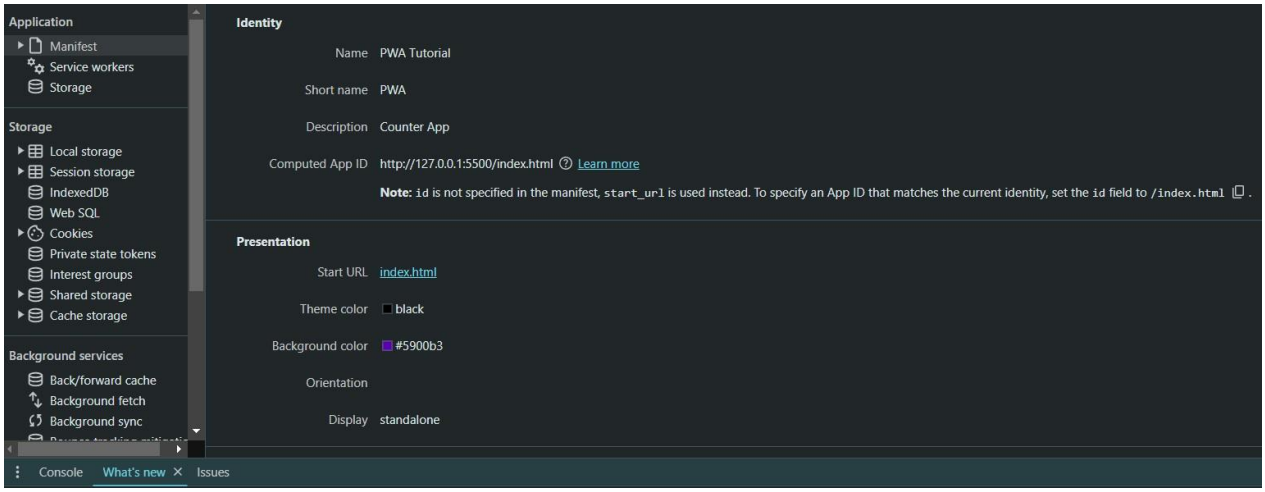
```
        console.log('SW registration failed');
      }
    }
  }
</script>
</html>
```

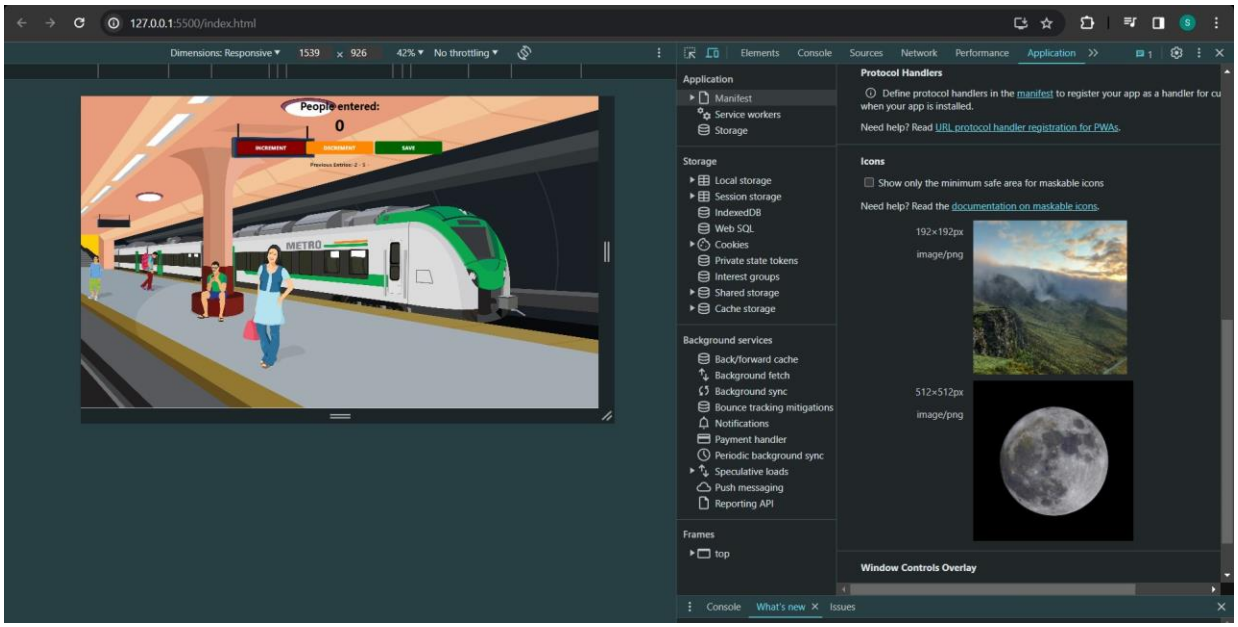
service-worker.js

```
var staticCacheName = "pwa";

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

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

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

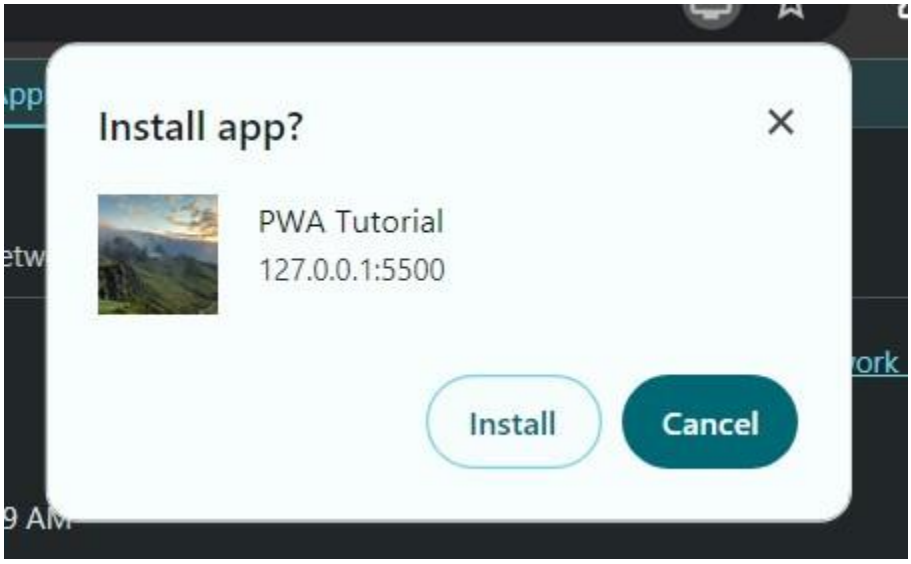
Output-

Right click->Inspect->Manifest

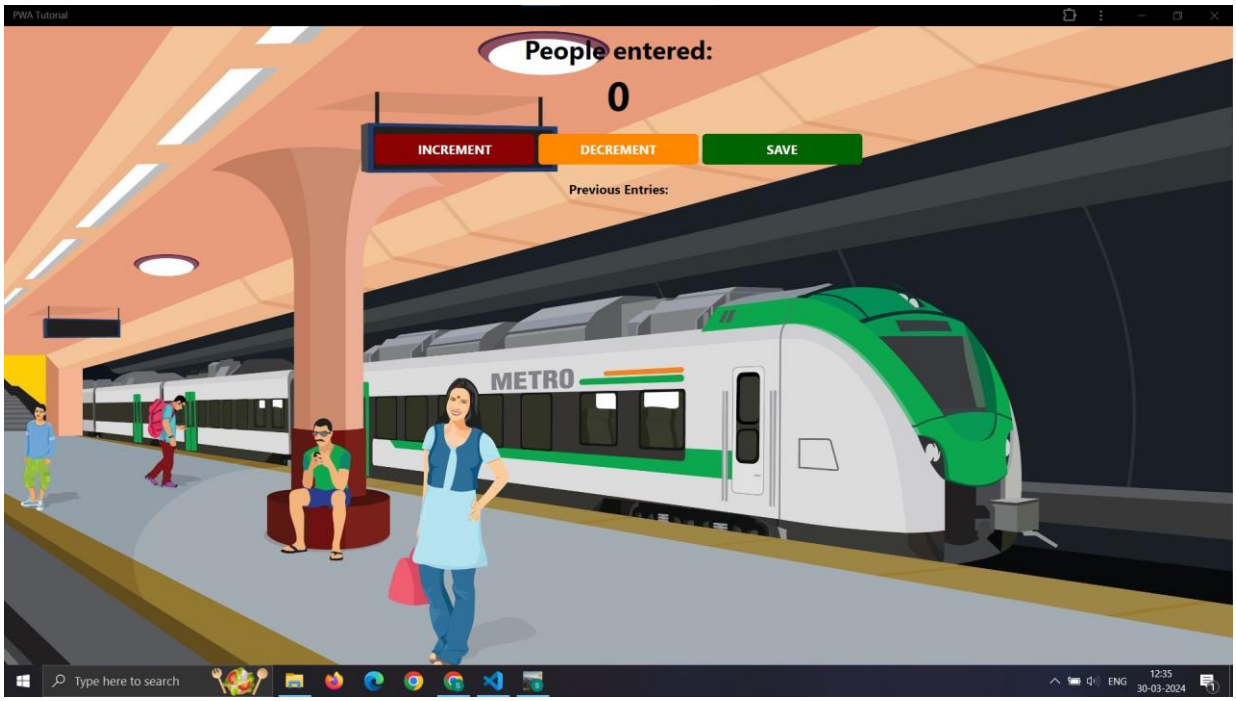




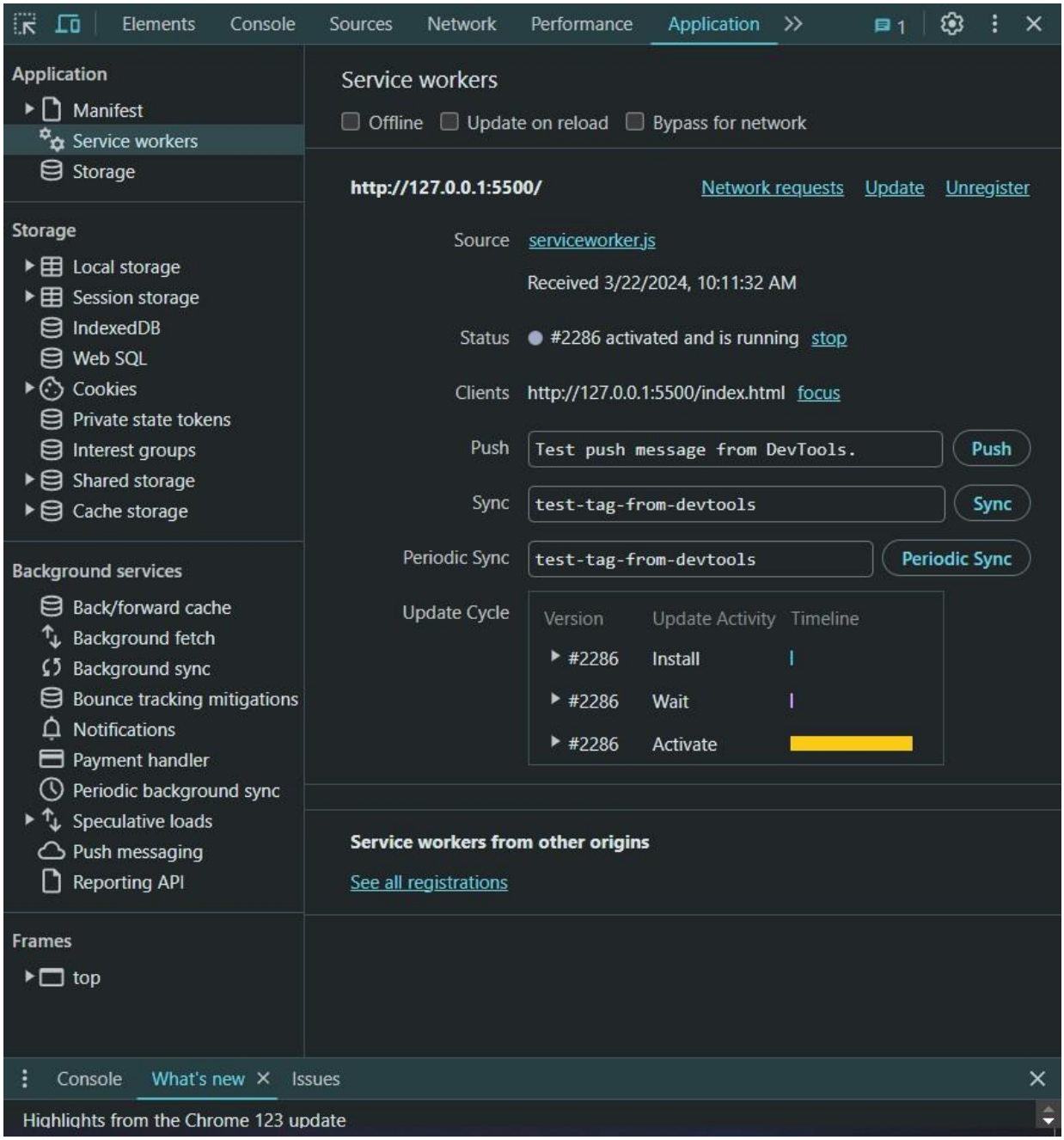
In toolbar section you will find install app. Click on that and click Install

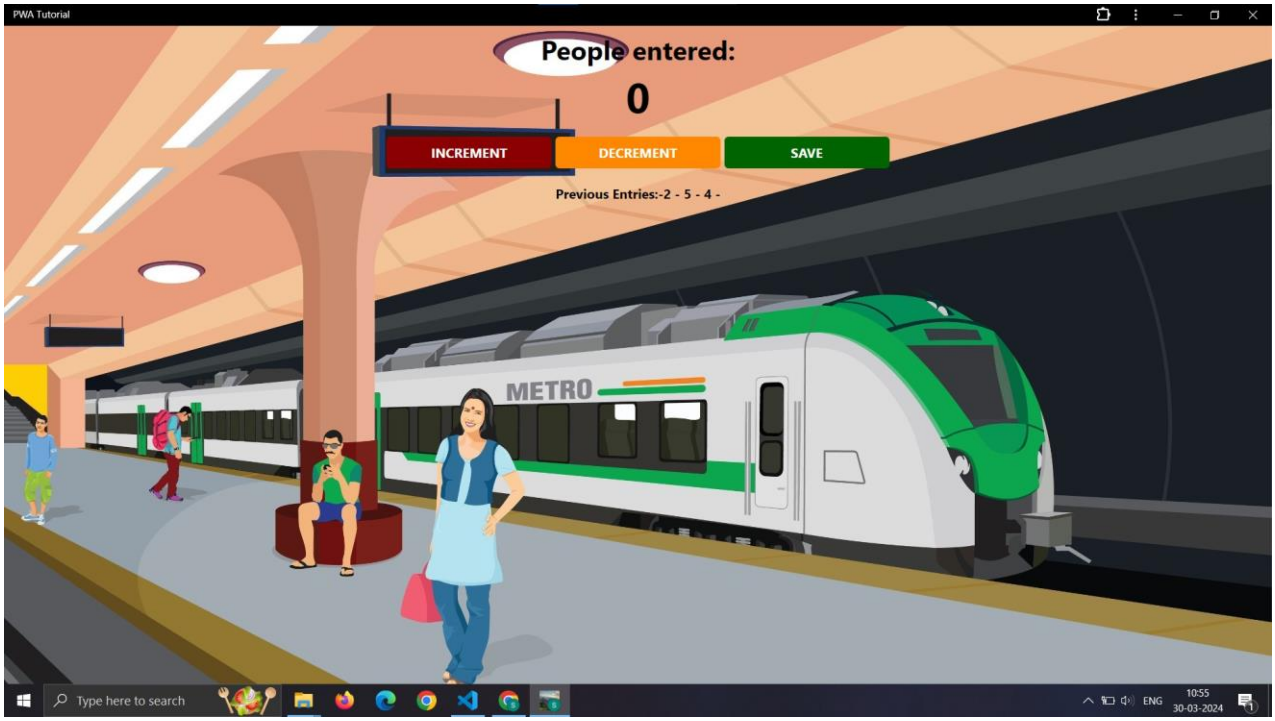


Installed app-



Project Title: Unacademy Clone/PWA- Counter App **Roll No. 67**





Conclusion-

Setting up the metadata in the Web app manifest file allows users to easily add our Ecommerce PWA to their device's home screen.

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.	67
Name	Anmol Vaswani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Name:- Anmol Vaswani
Roll No.:- 67
Div:- D15A

PWA Experiment - 8

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

Theory:

Service Worker

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

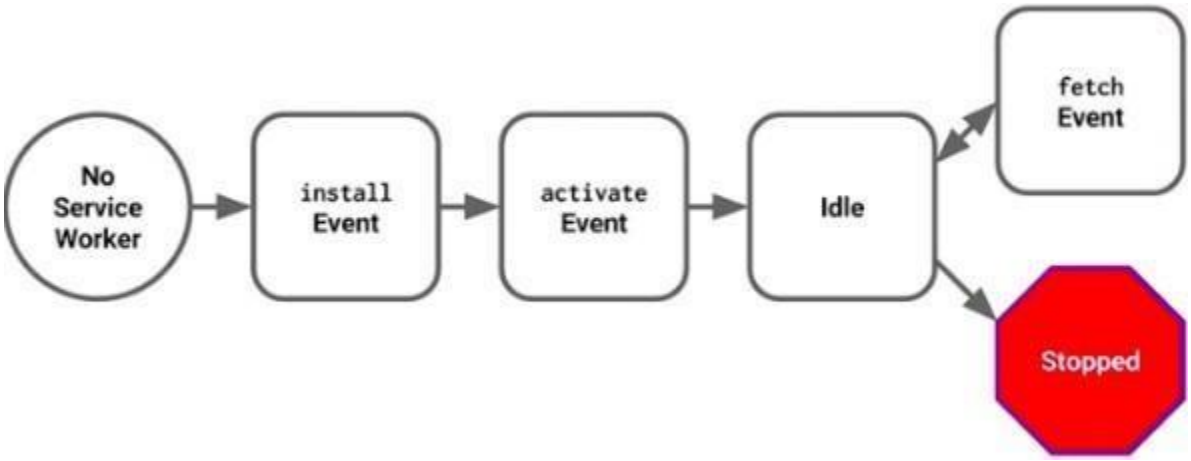
Sync of Service Worker.

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('/servi
ce-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/servi
  ce-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

```
<html>
    <head><link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/normalize/8.0.1/normalize
.css">
```

```
        <link rel="stylesheet" href="index.css">
        <link rel="manifest" href="/manifest.json">
    </head>
    <body>
        <h1>People entered:</h1>
        <h2 id="count-el">0</h2>
        <button id="increment-btn"
onclick="increment()">INCREMENT</button>
        <button id="decrement-btn"
onclick="decrement()">DECREMENT</button>
        <button id="save-btn" onclick="save()">SAVE</button>
        <p id="save-el">Previous Entries:</p>
        <script src="index.js"></script>
    </body>
</html>
```

index.js

```
let countEl = document.getElementById("count-el")
let count=0

function increment(){
    count=count+1
    countEl.innerText=count
}
function decrement(){
    count=count-1
    countEl.innerText=count
}
function save(){
    entry=document.getElementById("save-el").innerText
    document.getElementById("save-el").textContent += count  + " - "
    countEl.innerText=0
    count=0
}
if ('serviceWorker' in navigator) {
    window.addEventListener('load', () => {
        navigator.serviceWorker.register('service-worker.js')
            .then(registration => {
                console.log('Service Worker registered with scope:',
registration.scope);
            })
            .catch(error => {
```

```
                console.error('Service Worker registration failed:',
error);
            });
    });
}
```

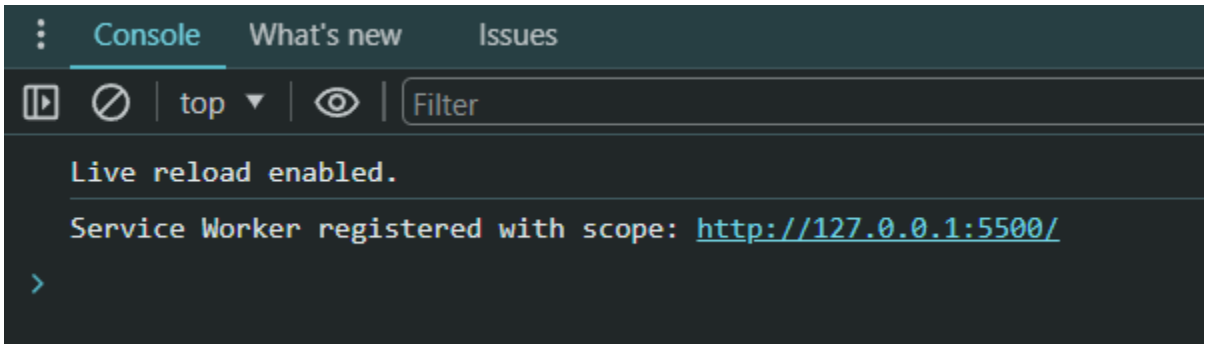
service-worker.js

```
const cacheName = 'ecommerce-pwa-v1';
const assetsToCache = [
    '/',
    '/index.html',
    '/index.css',
    '/index.js',
    '/images/one_512.png',
    '/images/two_.png'
];

self.addEventListener("install", function (event) {
    event.waitUntil(
        caches.open(cacheName).then(function (cache) {
            return cache.addAll(assetsToCache.map(url => new
Request(url, {credentials: 'same-origin'})));
        })
    );
});

self.addEventListener("fetch", function (event) {
    event.respondWith(
        caches.match(event.request).then(function (response) {
            return response || fetch(event.request);
        })
    );
});

self.addEventListener('activate', event => {
    event.waitUntil(
        caches.keys().then(cacheNames => {
            return Promise.all(
                cacheNames.filter(name => {
                    return name !== cacheName;
                }).map(name => {
                    return caches.delete(name);
                })
```

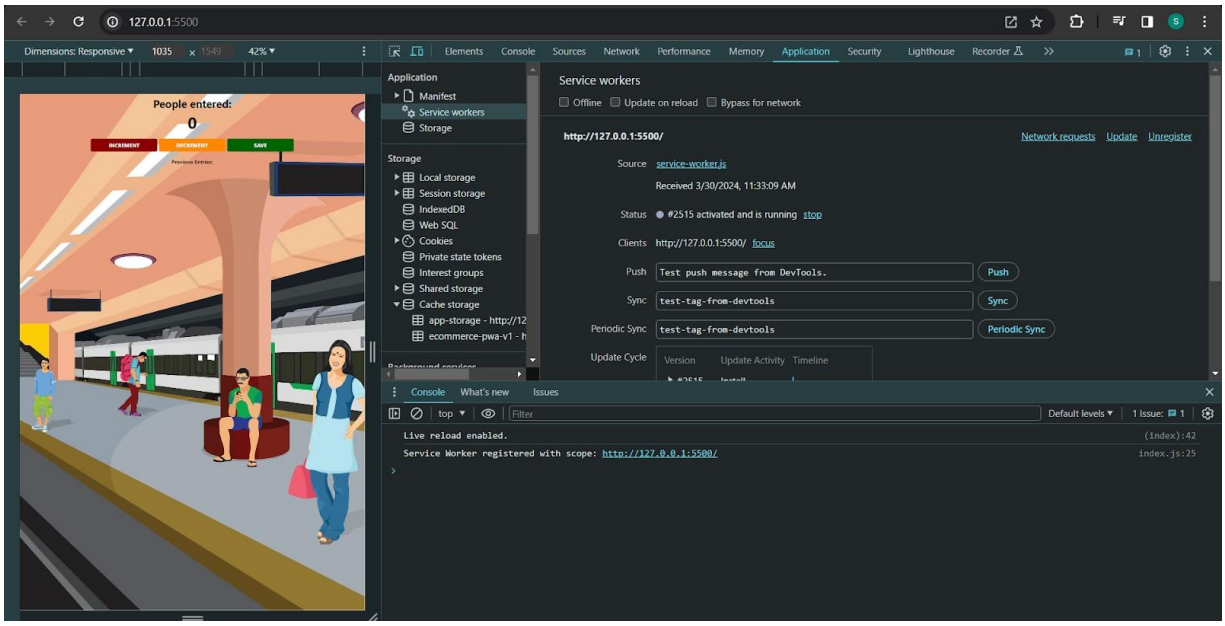
```
                );
            })
        );
});
```

Output-

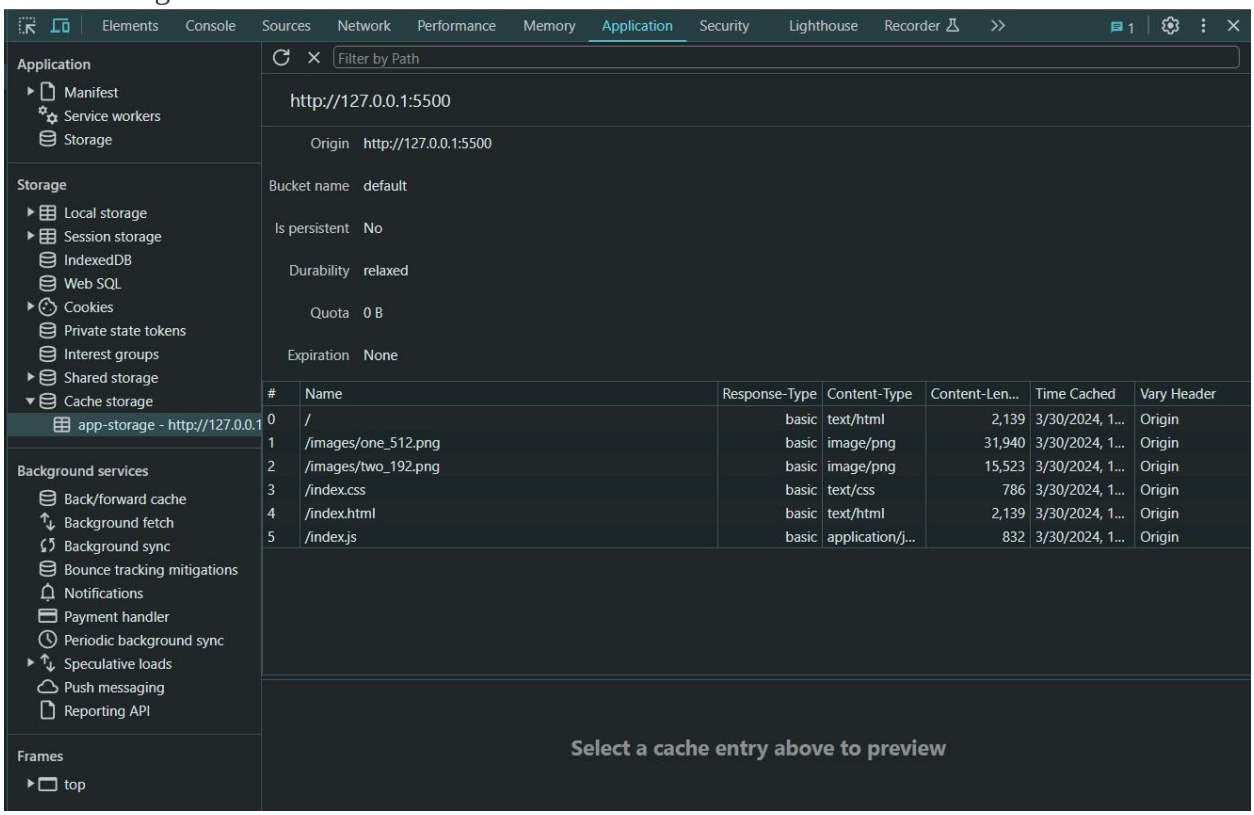
Right click->Inspect->Application->Service Worker



Service worker registered



Cache Storage-



Conclusion-

Through debugging the service worker registration and caching process, the issue of a 404 error was identified and resolved by ensuring correct file paths and server configurations. By addressing these key factors, the service worker was successfully registered and caching operations executed as intended, resulting in improved offline capabilities for the web application.

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.	67
Name	Anmol Vaswani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Name:- Anmol Vaswani
Roll No.:- 67
Div:- D15A

PWA 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.

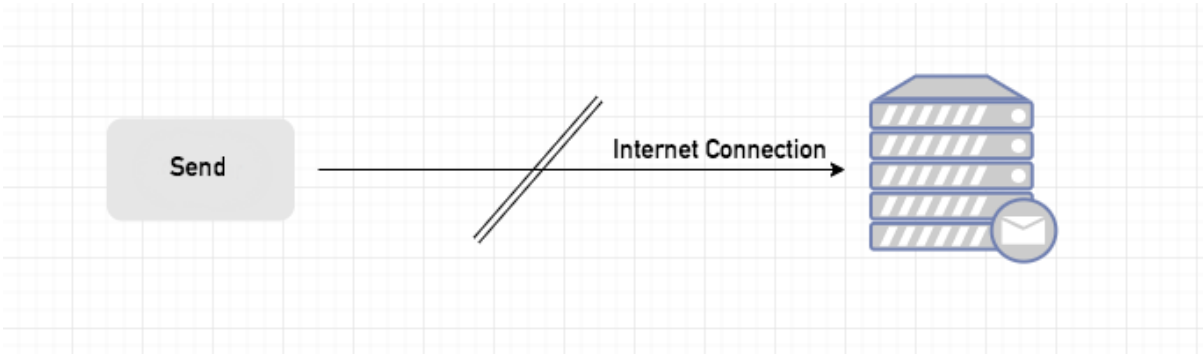
```
self.addEventListener("fetch", function (event) {
    const req = event.request;
    const url = new URL(req.url);

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

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

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

Sync Event

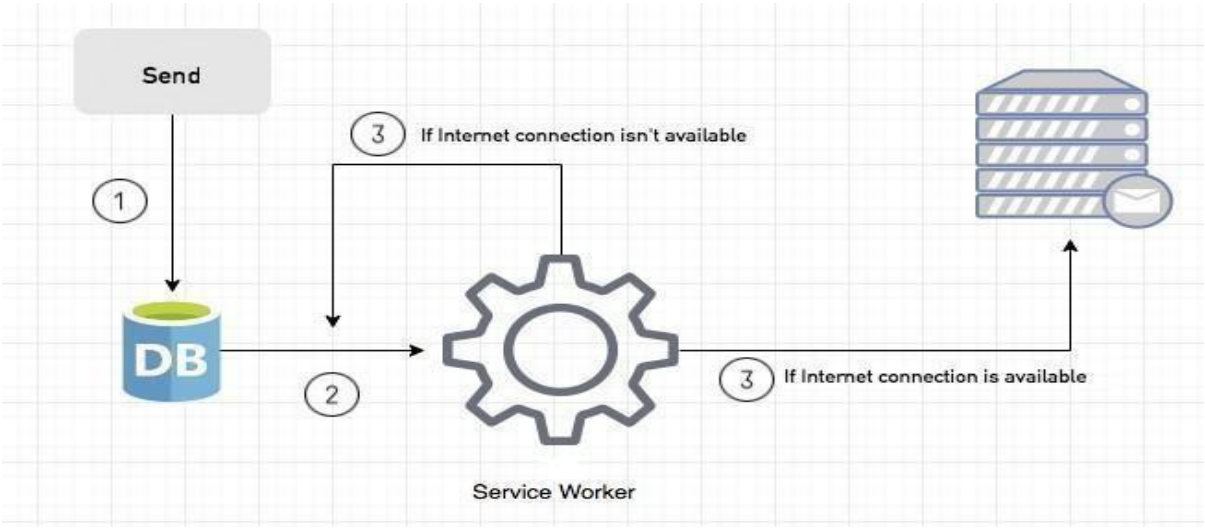
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

Here is a job for the Background Sync.

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



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



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

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

Event Listener for Background Sync Registration

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

Event Listener for sw.js

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

Push Event

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

We can check in the following example.

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

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

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

information notification with the “message” property.

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

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

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

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

Code:

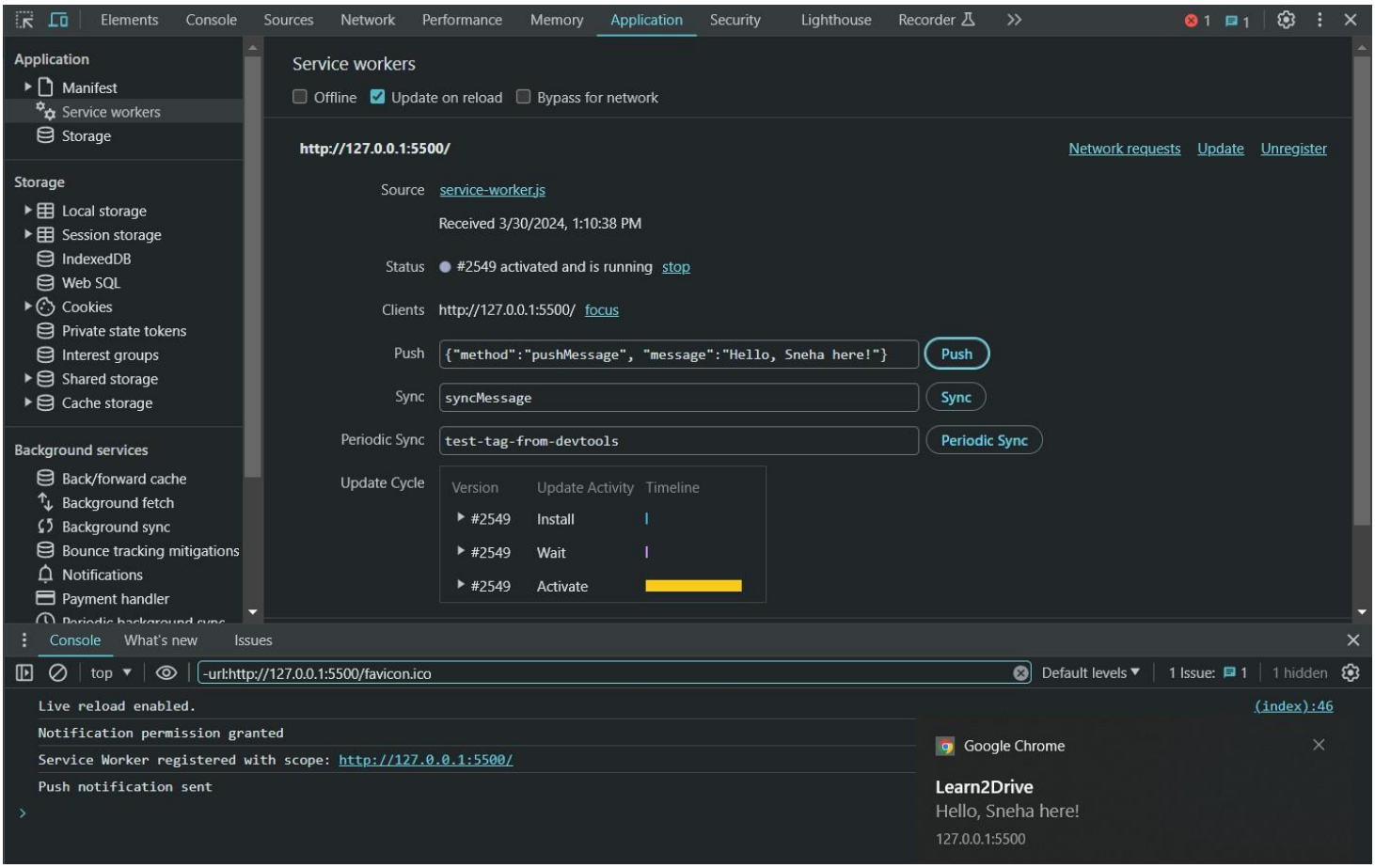
service-worker.js

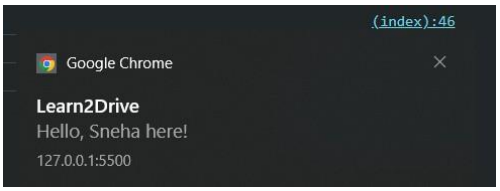
```
self.addEventListener("install", function (event) {
  event.waitUntil(preLoad());
});
self.addEventListener("fetch", function (event) {
  event.respondWith(
    checkResponse(event.request).catch(function () {
      console.log("Fetch from cache successful!");
      return returnFromCache(event.request);
    })
  );
  console.log("Fetch successful!");
  event.waitUntil(addToCache(event.requesst));
});
self.addEventListener("sync", (event) => {
  if (event.tag === "syncMessage") {
    console.log("Sync successful!");
  }
});
self.addEventListener("push", function (event) {
  if (event && event.data) {
    try {
      var data = event.data.json();
      if (data && data.method === "pushMessage") {
        console.log("Push notification sent");
        self.registration.showNotification("Learn2Drive", {
          body: data.message,
        });
      }
    } catch (error) {
      console.error("Error parsing push data:", error);
    }
  }
});
var preLoad = function () {
  return caches.open("offline").then(function (cache) {
    // caching index and important routes
    return cache.addAll(
    '/',
    '/index.html',
    '/index.css',
    '/index.js',
    '/images/one_512.png',
    '/images/two_.png');
  });
```

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

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


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

Output-

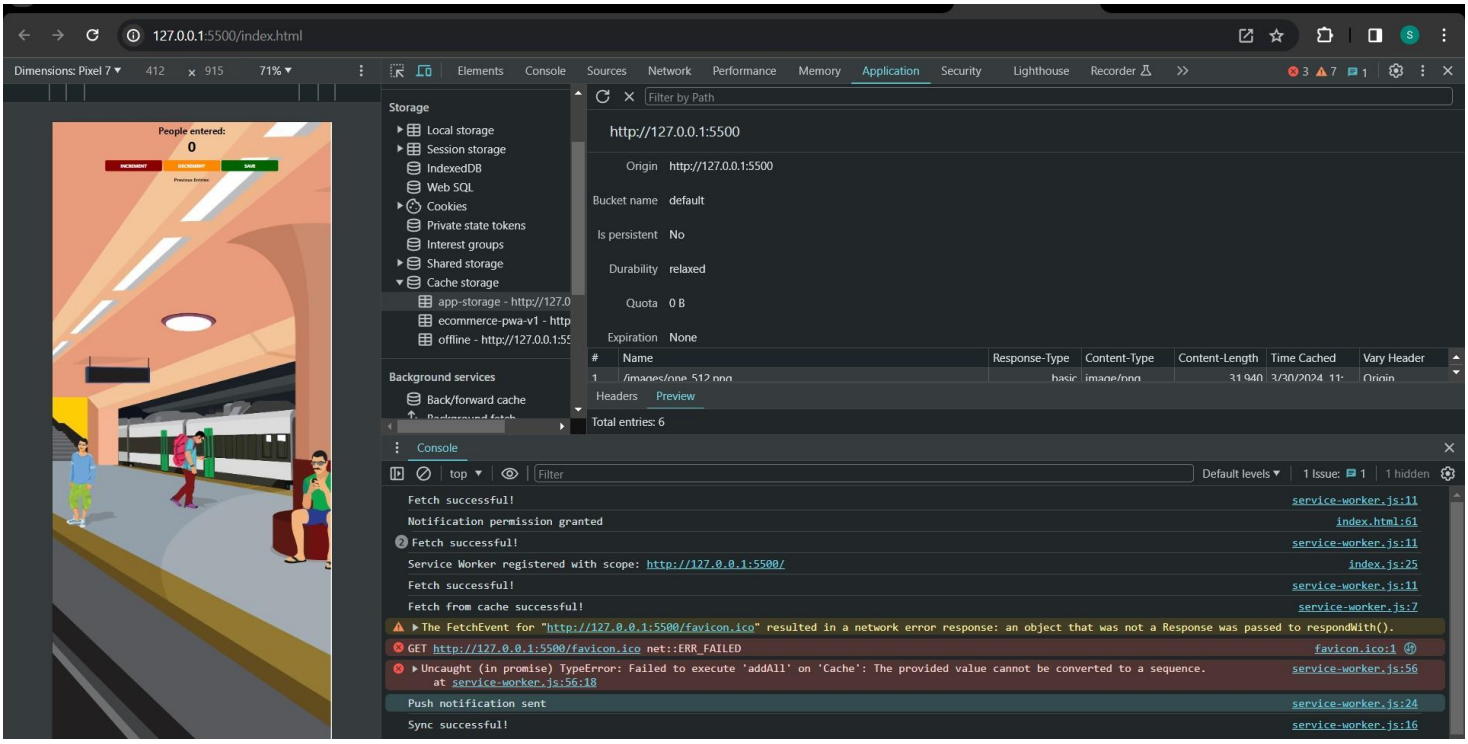
Push-


Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
IndexedDB
Web SQL
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigations
Notifications
Payment handler
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Network requests
Update
Unregister
Source
service-worker.js
Received 3/30/2024, 1:10:38 PM
Status
#2549 activated and is running
stop
Clients
http://127.0.0.1:5500/
focus
Push
{"method":"pushMessage", "message":"Hello, Sneha here!"}
Push
Sync
syncMessage
Sync
Periodic Sync
test-tag-from-devtools
Periodic Sync
Update Cycle
Version
Update Activity
Timeline
#2549
Install
#2549
Wait
#2549
Activate
Console
What's new
Issues
top
-url:http://127.0.0.1:5500/favicon.ico
Default levels
1 Issue:
1 hidden
Live reload enabled.
(index):46
Notification permission granted
Service Worker registered with scope: http://127.0.0.1:5500/
Push notification sent
Google Chrome
Learn2Drive
Hello, Sneha here!
127.0.0.1:5500

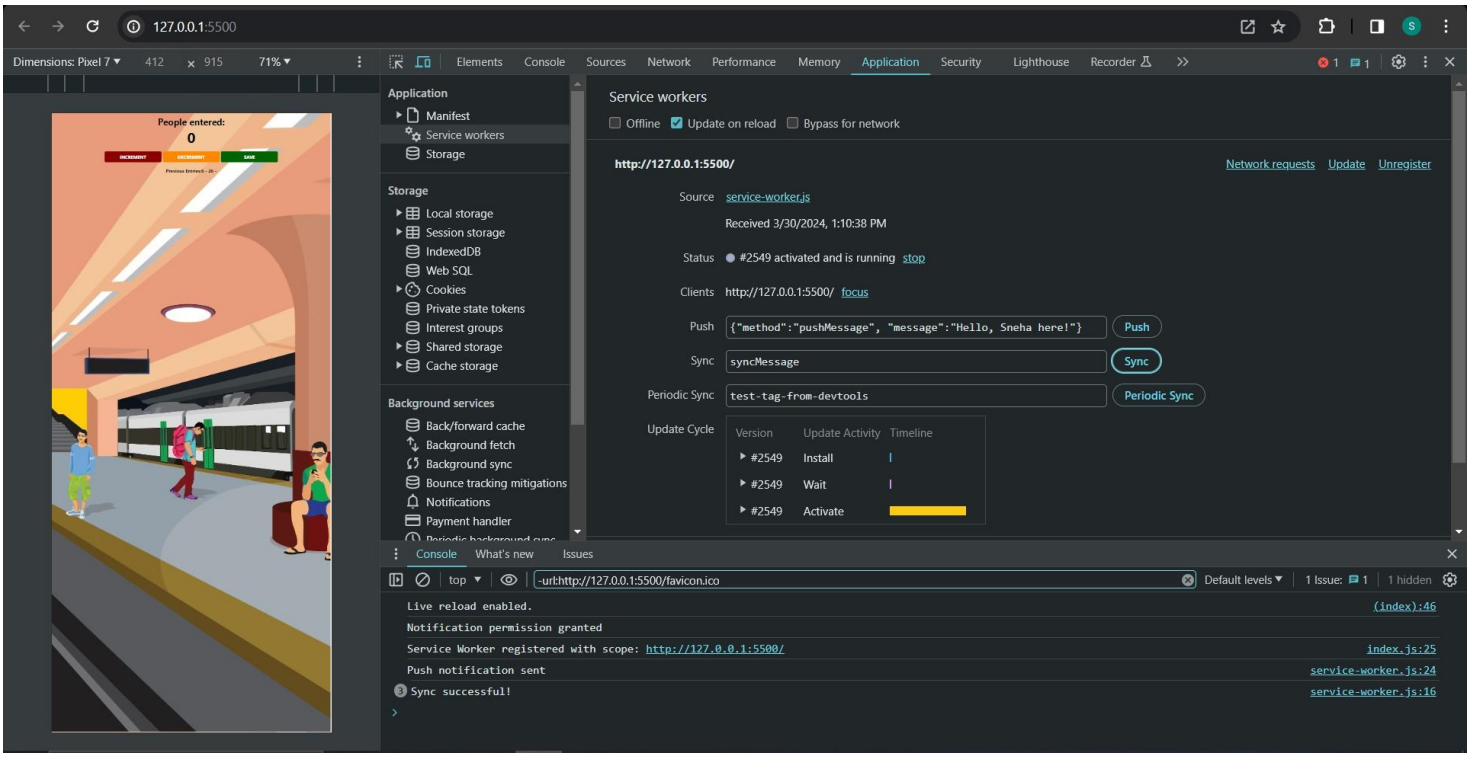

(index):46
Google Chrome
Learn2Drive
Hello, Sneha here!
127.0.0.1:5500

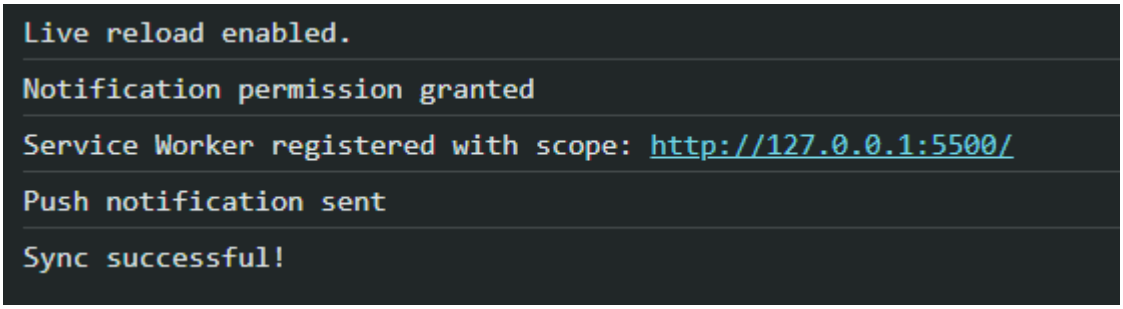
Project Title: Unacademy Clone/PWA- Counter App **Roll No. 67**

Fetch-



Sync





Conclusion-

Implemented fetch, push, sync operations of Service worker for PWA.

MAD & PWA Lab

Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	67
Name	Anmol Vaswani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Name: - Anmol Vaswani
Roll No.: - 67
Div: - D15A

MAD & PWA EXPERIMENT - 10

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

Theory:

GitHub Pages

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

GitHub Pages provides the following key features:

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

Reasons for favoring this over Firebase:

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

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

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

Cons

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

Firebase

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

Some of the features offered by Firebase are:

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

3. Data Accessibility- Data is stored as JSON in Firebase. Every piece of data has its own URL which can be used in Firebase's client libraries and as a REST endpoint. These URLs can also be entered into a browser to view the data and watch it update in real-time.

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

Pros

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

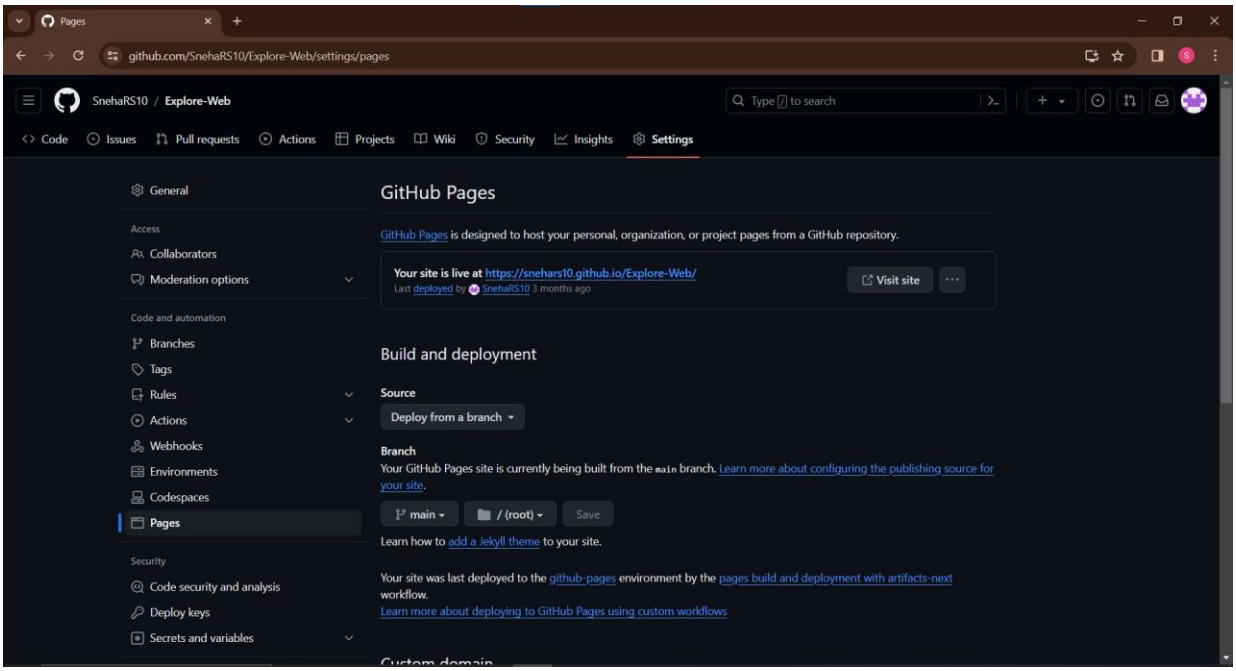
Cons

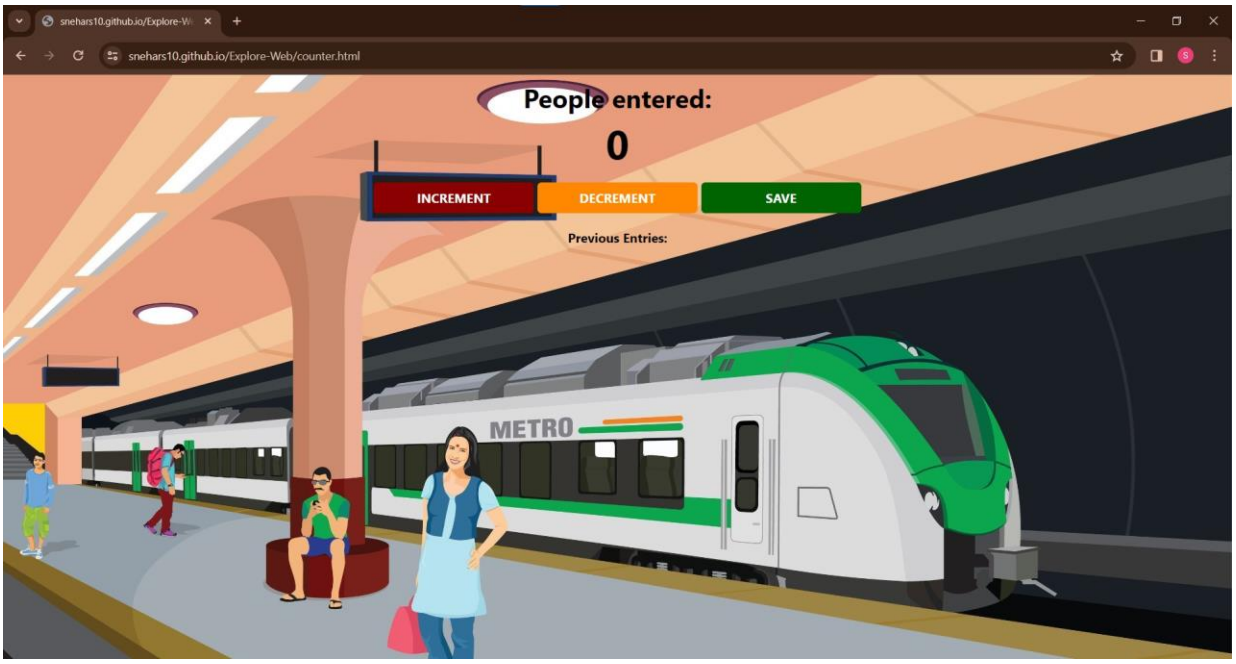
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator.

Link to our GitHub repository:
Pages link - https://snehars10.github.io/Explore-Web/
Repository link - https://github.com/SnehaRS10/Explore-Web

Github Screenshot:

Project Title: Unacademy Clone/PWA- Counter App **Roll No. 67**





MAD & PWA Lab

Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	67
Name	Anmol Vaswani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	15

Project Title: Unacademy Clone/PWA- Counter App **Roll No. 67**

Name: - Anmol Vaswani
Roll No.: - 67
Div: - D15A

MAD & PWA EXPERIMENT - 11

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

Theory :
Reference : https://www.semrush.com/blog/google-lighthouse/

Google Lighthouse :

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

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

Key Features and Audit Metrics

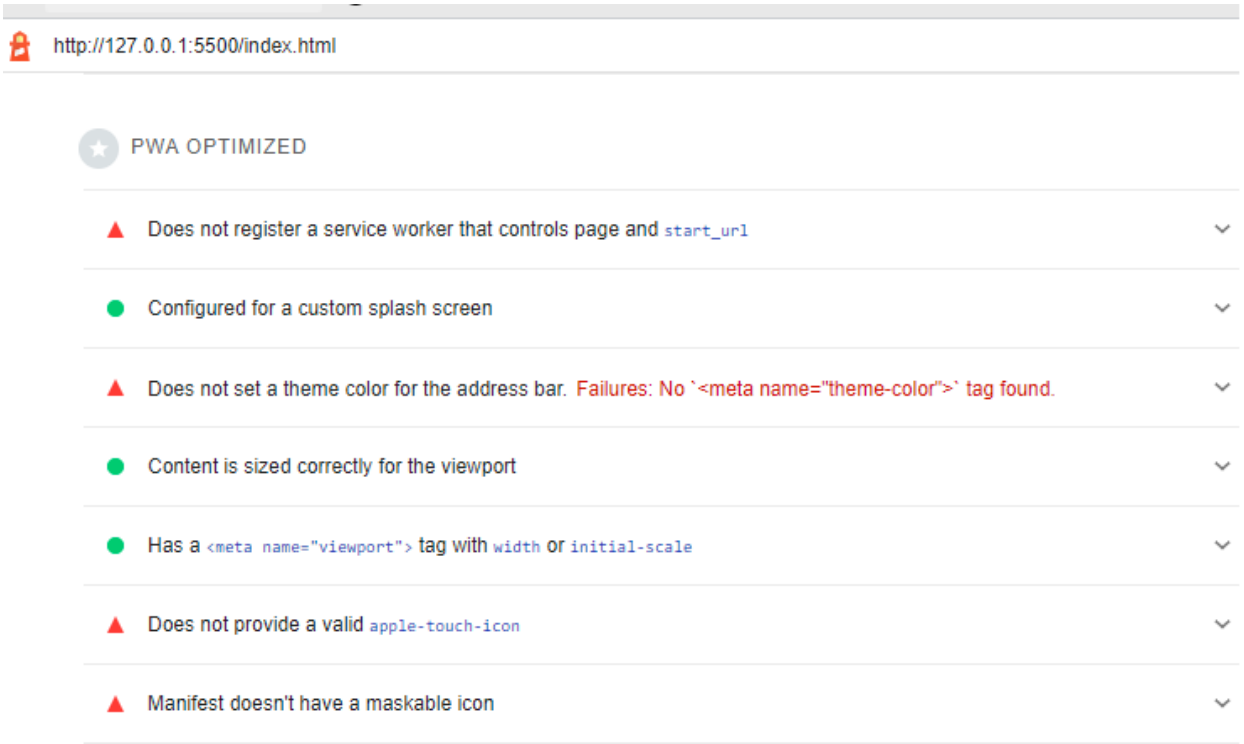
Google Lighthouse has the option of running the Audit for Desktop as well as mobile version of your page(s). The top metrics that will be measured in the Audit are:

1. **Performance:** This score is an aggregation of how the page fared in aspects such as (but not limited to) loading speed, time taken for loading for basic frame(s), displaying meaningful content to the user, etc. To a layman, this score is indicative of how decently the site performs, with a score of 100 meaning that you figure in the 98th percentile, 50 meaning that you figure in the 75th percentile and so on.
2. **PWA Score (Mobile):** Thanks to the rise of Service Workers, app manifests, etc., a lot of modern web applications are moving towards the PWA paradigm, where the objective is to make the application behave as close as possible to native mobile applications. Scoring points are based on the Baseline PWA checklist laid down by Google which includes Service Worker implementation(s), viewport handling, offline functionality, performance in script-disabled environments, etc.
3. **Accessibility:** As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button

names, etc.). Unlike the other metrics though, Accessibility metrics score on a pass/fail basis i.e. if all possible elements of the page are not screen-reader friendly (HTML5 introduced features that would make pages easy to interpret for screen readers used by visually challenged people like tag names, tags such as <section>, <article>, etc.), you get a 0 on that score. The aggregate of these scores is your Accessibility metric score.

4. **Best Practices:** As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS
   Avoiding the use of deprecated code elements like tags, directives, libraries, etc. Password input with paste-into disabled
   Geo-Location and cookie usage alerts on load, etc.

Changes made to the code :



For theme color add a meta tag in index.html-

<meta name="theme-color" content="#4285f4">

For a maskable icon add "purpose": "any maskable" to the icons in manifest.json

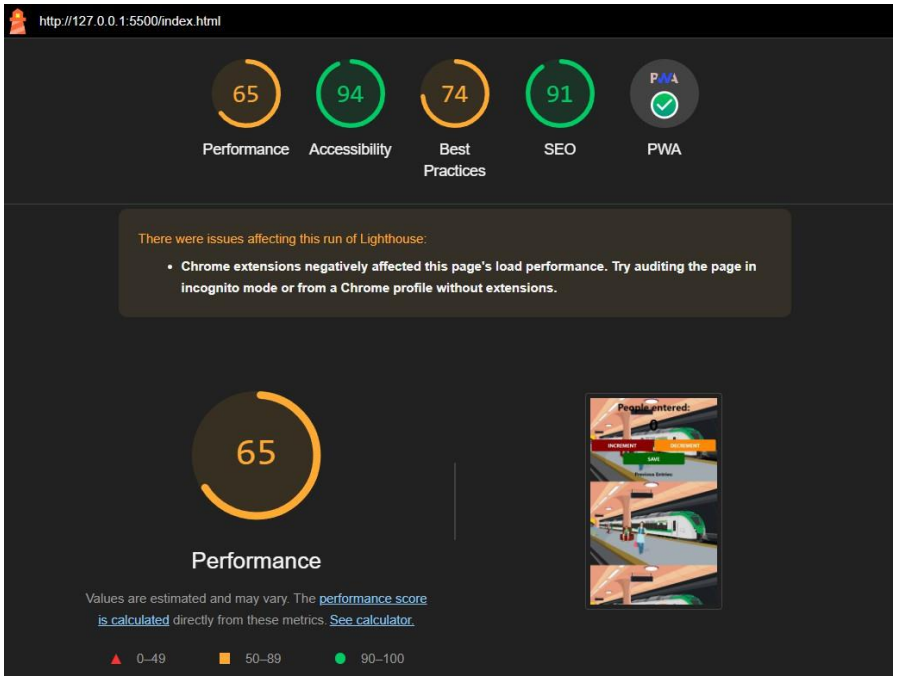
file For apple touch icon add the following meta tag in index.html-

<link rel="apple-touch-icon" href="">

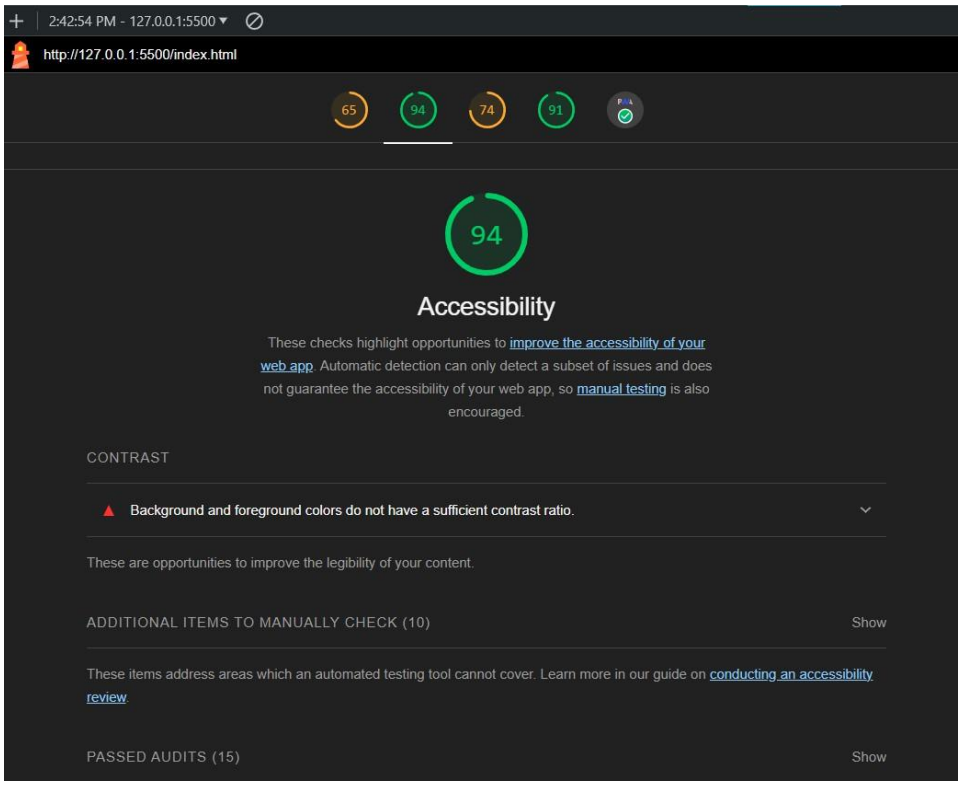
Changes in manifest.json

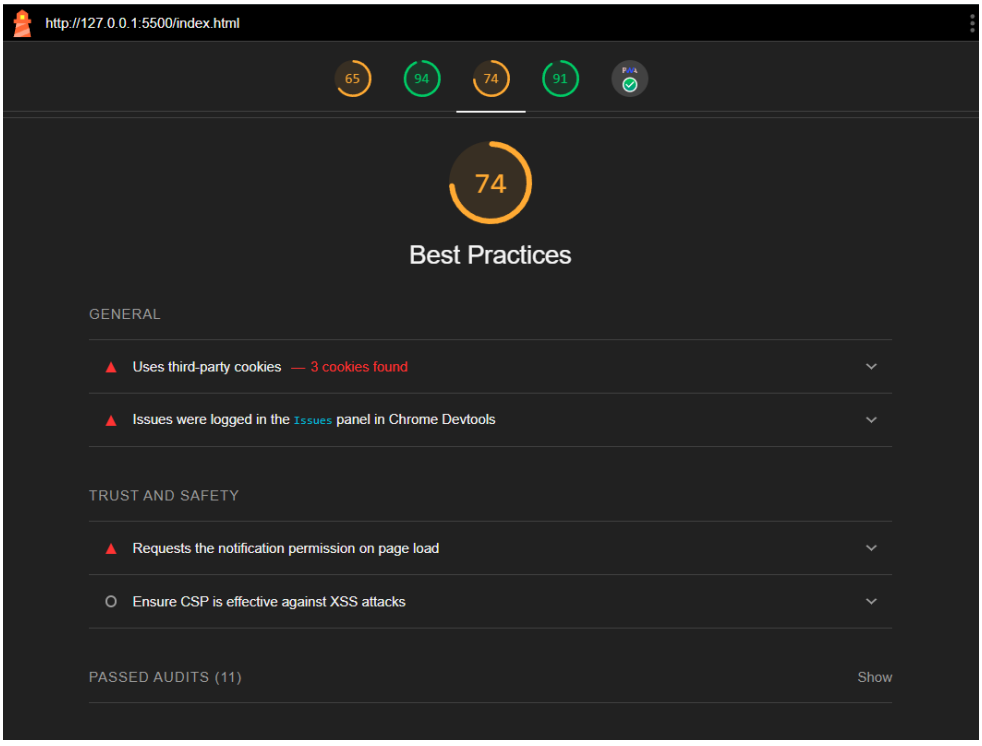
```
{
   "name":"PWA Tutorial",
   "short_name":"PWA",
   "start_url":"index.html",
   "display":"standalone",
   "background_color":"#5900b3",
   "theme_color":"black",
   "scope": ".",
   "description":"Counter App",
   "icons":[
   {
   "src":"images/two_192.png",
   "sizes":"192x192",
   "type":"image/png",
   "purpose":"any maskable"
   },
   {
   "src":"images/one_512.png",
   "sizes":"512x512",
   "type":"image/png",
   "purpose":"any maskable"
   }
]
}
```

Report Generated-

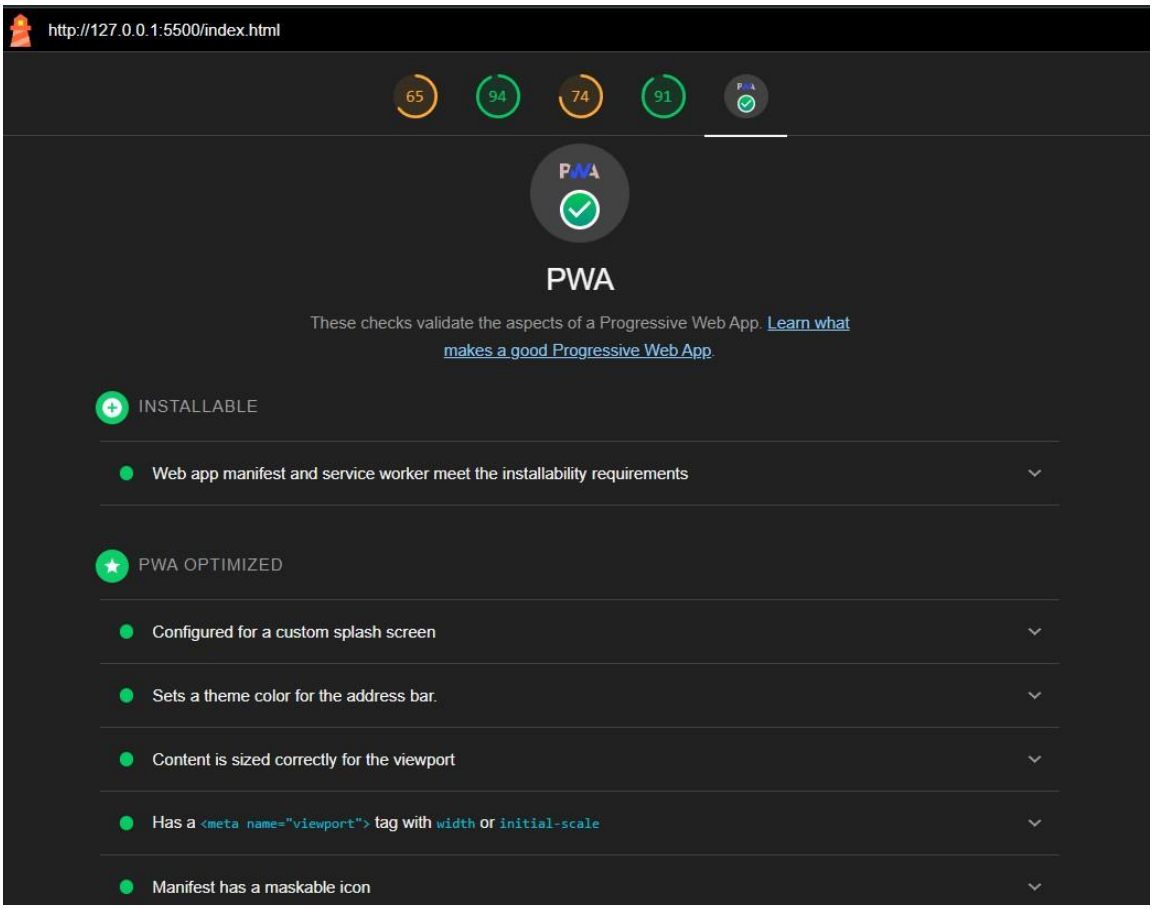


Project Title: Unacademy Clone/PWA- Counter App **Roll No. 67**





Project Title: Unacademy Clone/PWA- Counter App **Roll No. 67**



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

MAD & PWA Lab
Journal

Experiment No.	Assignment-1
Assignment 1 Questions	1. Flutter Overview:Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community. 2. Widget Tree and Composition: Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree. 3. State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable. 4. Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.
Roll No.	67
Name	Anmol Vaswani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	5

11

Name :- Anmol Vaswani
Div :- D15A
Roll No :- 67

Assignment - 1

Q1 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.

→ (1) Key Features of flutter :-

(a) Single Codebase for Multiple Platforms :-
Flutter allows developers to write code and deploy it on both iOS and Android platforms.

(b) Hot Reload :-
This enables developers to instantly see the results of the code changes they make.

(c) Expressive UI :-
Developers have the flexibility to create expressive and flexible UIs.

(d) Integration with other Tools :-
Flutter can easily integrate with other popular development tools and frameworks.

(2) Advantages of flutter :-

(a) Faster Development
Uses Single codebase for multiple platforms.

(b) Consistent UI Across Platforms
Widgets provide a consistent look and feel across different platforms.

(c) Cost - Efficiency
Developing and maintaining single codebase for both iOS and Android reduces development cost and

resources.

3 Differ from Traditional Approach.

(a) Traditional approach uses a hierarchical structure for UI components, whereas flutter uses a widget-based approach.

(b) Flutter compiles to native ARM code, providing performance comparable to native applications.

(c) Hot Reloads allows to see changes made instantly.

Flutter's popularity is driven by increased productivity, a growing community, ~~flexibl~~ flexibility in UI design, cross-platform development capabilities and adoption by major companies.

Q2 Widget Tree and Compositions : Describe the concept of wideg 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 bee.

→ Widget Tree

- The widget tree is a hierarchical structure of widgets that defines the user interface of an application.
- Every visual element, from simple components to complex layouts, is represented by a widget.
- Widget can be categorized in 2 types

(a) Stateless Widget

It is immutable and cannot change over time.

Eg:- images, text.

(b) Stateful Widget

Widget that can change its state over time.

Eg:- buttons, forms.

Widget Composition

- Widget composition in flutter involves combining multiple simple widgets to create more complex and compound widgets.
- This composability is a powerful concept that allows developers to build sophisticated user interfaces by nesting widgets within each other.

Commonly Used Widgets :-

1. Container :- A box model for padding, margin and decoration.
2. Stack :-
   Overlapping widget, allowing them to be layered on top of each other.
3. Column and Row :- layout widgets for arranging children vertically or horizontally.
4. Grid View :- A scrollable grid of widgets.
5. List View :- A scrollable list of widgets.
6. AppBar :- A material design app bar typically at top of screen.
7. Button Widgets :- Interactive buttons for user actions.
8. Text Field :- An input field for users to enter text.

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

→ • State management is crucial in Flutter applications because it involves managing the data that can change over time.
• Flutter is reactive, meaning the UI rebuilds when the underlying data changes.

SetState	Provider	Riverpod.
① Built-in flutter method.	① External package named ('provider').	① External package ('riverpod')
② Limited scalability for large apps.	② Suitable for medium sized apps.	② Designed for large and complex apps
③ May lead to code redundancy.	③ Balances simplicity and readability.	③ Emphasizes readability and clean syntax.
④ local state within a widget	④ Global state within widget bee.	④ Global state with additional features.

⑤ Widely used, well established.	⑤ Widely adopted, well supported	⑤ Gaining popularity growing community.
⑥ Testing can be more challenging.	⑥ Good testability support.	⑥ Enhanced testing experience.

Scenario where each is applicable.

① SetState.
For small to ~~moderatory~~ moderately complex applications. When managing local state within a widget.
Eg:- Simple forms, UI components with local UI-specific state.

② Provider
For medium to large-sized applications. When a centralized state is needed accessible by multiple widgets.
Eg:- Managing user authentication, theme changes or app-wide configuration.

③ Riverpod
For large and complex applications. When testability and maintaining are top priorities.
Eg:- Complex applications with multiple features dynamic UIs.

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

→ Integration:-

① Go to firebase console and create a new project.

② Add firebase SDK by including dependencies in pubspec.yaml.

```
dependencies:
  firebase_core: ^version.
  firebase_auth: ^version.
  cloud_firestore: ^version.
```

③ Run flutter pub get.

④ Initialise firebase by calling 'Firebase.initialiseApp()' in main.

```
import 'package.firebase_core/firebase_core.dart';

void main() async{
widgets FlutterBinding.ensureInitialized();
await Firebase.initializeApp().
runApp(myApp());
}
```

Benefits of Using Firebase as Backend

1. **Real-time Database:-** Firebase offers a real-time NoSQL database
2. **Authentication:-** Provides a secure and easy-to-implement solution for user authentication.
3. **Cloud Firestore:-** Firebase's Cloud Firestore provides a secure and easy-to-implement scalable NoSQL database that allows you to store and sync data in real time.
4. **Hosting:-** Firebase Hosting provides a simple and efficient way to deploy and host web applications.

Data Synchronization

1. **Real-time Database**
   - When data changes on one client, it triggers events that automatically update data on other clients.

2. **Cloud Firestore**
   It notifies clients when data changes, allowing for seamless real-time updates.

3. **Authentication.**
   If user sign in or out on one device, the authentication state is automatically reflected on other devices.

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.	67
Name	Anmol Vaswani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4:Understand various PWA frameworks and their requirements LO5: Design and Develop a responsive User Interface by applying PWA Design techniques LO6:Develop and Analyze PWA Features and deploy it over app hosting solutions
Grade:	4

Anmol Vaswani
D15A
67

Assignment - 2

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.

→ (i) Progressive Web App (PWA)

A progressive web App (PWA) is a type of web application that combines the best features of both web and native mobile apps.

(ii) Significance in Modern Web development:

- PWAs provide an enhanced user experience by offering fast loading times, offline functionality, and smooth interactions.
- They bridge the gap between web and native apps, allowing users to access app-like features directly from their browsers.
- PWAs are platform-agnostic, meaning they work seamlessly across different devices and operating systems.

(iii) Key characteristics differentiating PWA from Traditional Mobile Apps:-

- Offline Capabilities: PWAs can function offline or in low-network conditions, thanks to service workers and caching mechanisms.
- Responsive Design: PWAs adapt to various screen size, making them suitable for both desktop and mobile devices.
- App Shell Architecture: PWAs load essential

resources first, providing a fast initial load time.
- Push Notifications: PWAs can send notifications to users even when the app is not open.
- Installation:- Users can install PWAs on their home screens, similar to native 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.

→ (i) Responsive web design ensures that a website adapts to different screen sizes and orientations.

(ii) Importance in the context of PWAs:-
- PWAs must be responsive to provide a consistent experience across devices.
- Responsive design ensures that PWAs look and function well on both desktops and mobile devices.

(iii)

	Responsive	Fluid	adaptive web design.
Definition	(1) Responsive web design involves creating websites that automatically adjust and respond to different screen sizes by using a mix of flexible grids, layouts, and CSS media queries.	(1) Fluid web design involves creating websites that use proportional units like percentages instead of fixed units like pixels for layout & element sizing.	(1) Adaptive Web design involves creating multiple layouts or templates for different device categories, such as desktops, tablets and smartphones.

Approach	② It uses a single codebase that dynamically changes the layout and content based on the screen size, allowing the website to adapt seamlessly to any device.	② It focuses on creating designs that scale smoothly Smoothly across different screen sizes by allowing elements to strech or shrink based on the viewport width.	② It detects the user's device or screen size and serves a predefined layout optimized for that specific device category.
Example	③ A responsive website rearrange navigation menus, resize images, and adjust text sizes to provide an optimal viewing experience on both desktop computers and mobile devices.	③ In a fluid design, elements like text blocks and images expand or contract smoothly as the browser window is resized, maintaining proper Proportions and alignments.	③ An adaptive website may have seperate layouts tailored for desktops, tablets in landscape orientation, tablets in portrait orientation, and smartphones. When a user visits the site, the appropriate layout is served based on their device.

3 Describe the lifecycle of Service Workers, including registration, installation, and activation phases.

→ The lifcycle of a Service Workers involves three main phases

① Registration,
② Installation,
③ Activation.

① Registration :-

The developers use 'navigator.serviceWorker.register()' method to register the Service Worker. This method takes the path to the Service Worker script as its argument.

The registration phase begins when a JavaScript file containing the Service Worker code is registered in the web page. This registration typically occurs in the main JavaScript file of the web application.

Once registered the browser starts the process of installing the Service Worker in the background.

② Installation :-

During installation, the service worker caches essential resources like HTML, CSS and JS files. The browser downloads the Service Worker script specified during registration. The download script is then executed, and the Service worker initializes by listening for events such as fetch and push notifications.

In the 'install' event handler, developers can perform tasks such as caching static assets or performing other

setup operations. If the installation is successful, the Service Worker moves on to the activation phase.

③ Activation :-

After the installation is completed, the Service Worker enters the activation phase. During activation, the new Service Worker becomes active and takes control of any clients (windows or tabs) under its scope. The old Service worker, if there is one, is then replaced by the new one. This process is known as the "Update" process. Once activated, the Service Worker remains active until it's either replaced by a newer version or unregistered.

4 Explain the use of IndexedDB in the Service Worker for data Storage.

→ IndexedDB is a low-level API that stores large amount of structed data on a user's browser. It uses indexes to enable high-performance Searches of this data. We can use IndexedDB to store data like files, blobs, images, videos, objects, lists, and arrays. IndexedDB is asynchronous, it doesn't stop the user interface from rendering while the data loads. IndexedDB allows to create web applications with rich query abilities regardless of network availability, so your applications can work both online and offline. We can use IndexedDB in a service worker by adding

an event listener to handle message. The maximum limit is based on the browser and the disk space. For example, chrome and chromium-based browsers allow up to 80% disk space. It allows ~~you~~ to categorise your data using object stores. It allows to store large amounts of data.