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

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Project Title: MyFitnessPal (Fitness Tracker) **Roll No. 16**

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

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

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers : Mrs. Kajal Joseph.

Email : kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: MyFitnessPal (Fitness Tracker) **Roll No. 16**

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			
2.	To design Flutter UI by including common widgets.	LO2			
3.	To include icons, images, fonts in Flutter app	LO2			
4.	To create an interactive Form using form widget	LO2			
5.	To apply navigation, routing and gestures in Flutter App	LO2			
6.	To Connect Flutter UI with fireBase database	LO3			
7.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4			
8.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA	LO5			
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5			
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5			
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6			
12.	Assignment-1	LO1,LO2 ,LO3			
13.	Assignment-2	LO4,LO5 ,LO6			

MAD & PWA Lab
Journal

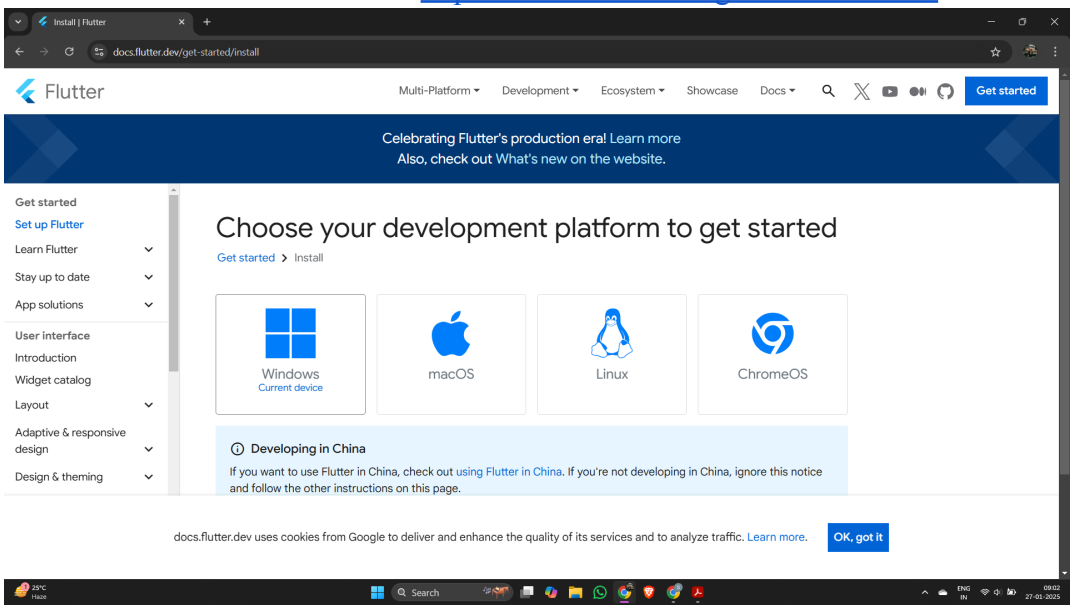
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	16
Name	Shreyash Ganesh Dhekane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

EXPERIMENT NO 01

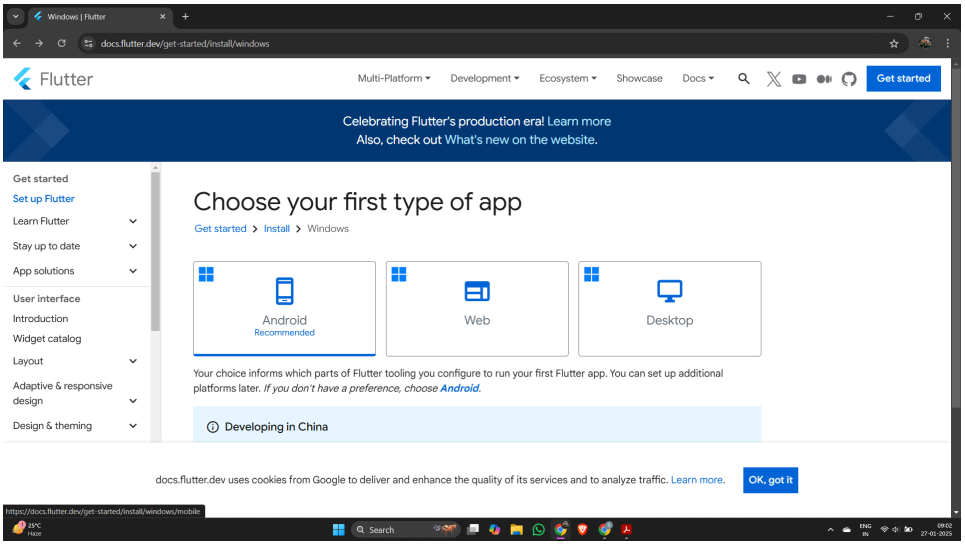
Name:- Shreyash Dhekane **Class:-** D15A **Roll:No:-** 16

AIM:- Installation and Configuration of Flutter Environment.

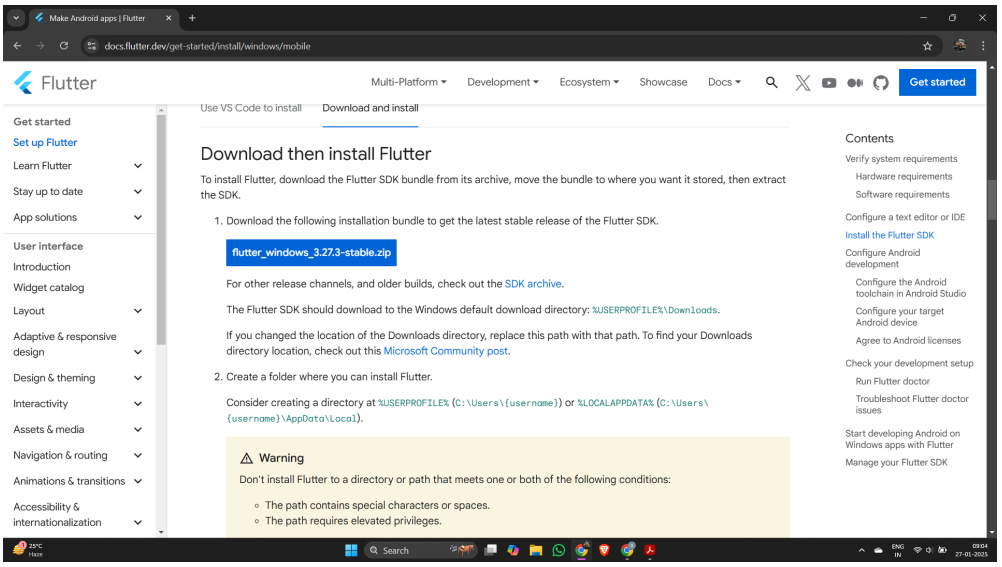
Step 1: Go to the official Flutter website: https://docs.flutter.dev/get-started/install



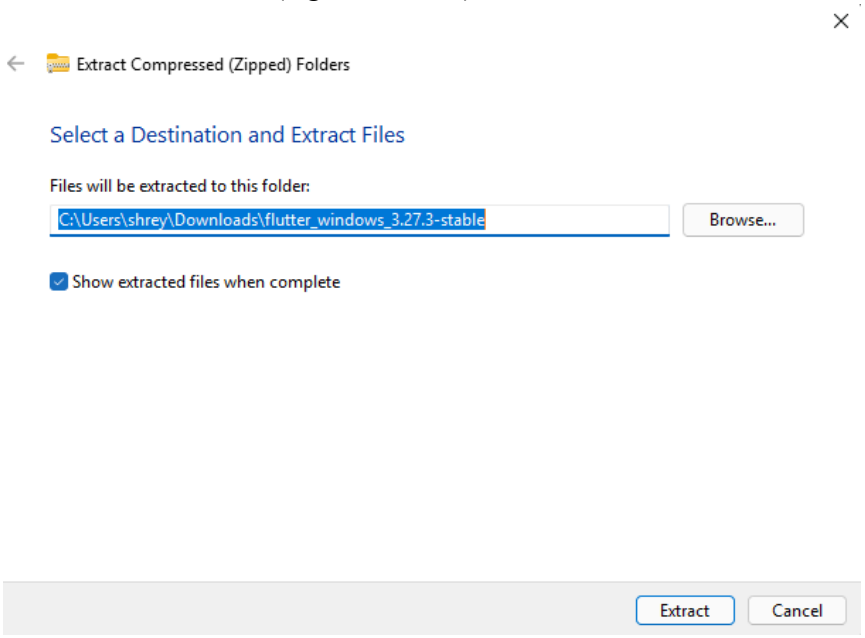
Step 2:To download the latest Flutter SDK, click on the Windows icon > Android



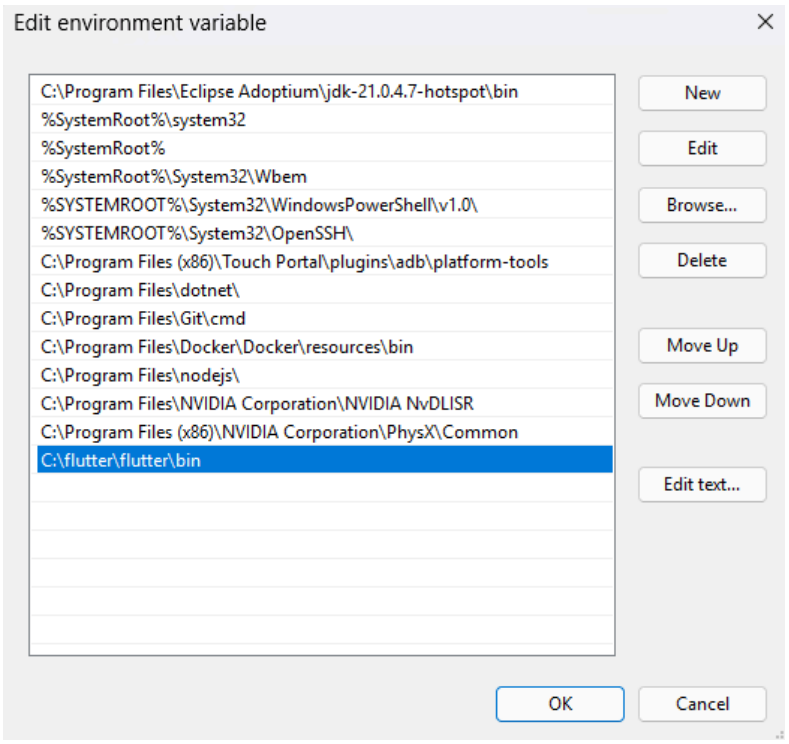
Step 3: For Windows, download the stable release (a .zip file).



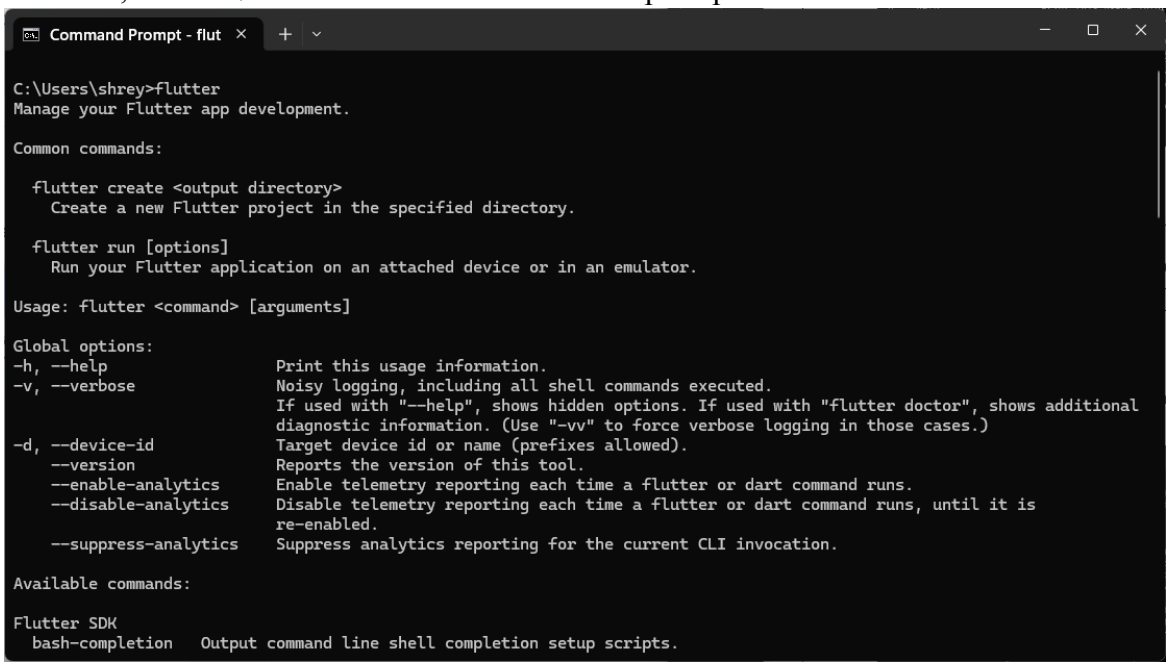
Step 4: Extract the ZIP file to a folder (e.g., C:\flutter).



Step 5 :- Add Flutter to System PATH
Right-click on the Start Menu > System > Advanced system settings > Environment Variables.
Under System Variables, find Path and click Edit.
Add the full path to the flutter/bin directory (e.g., C:\flutter\bin).



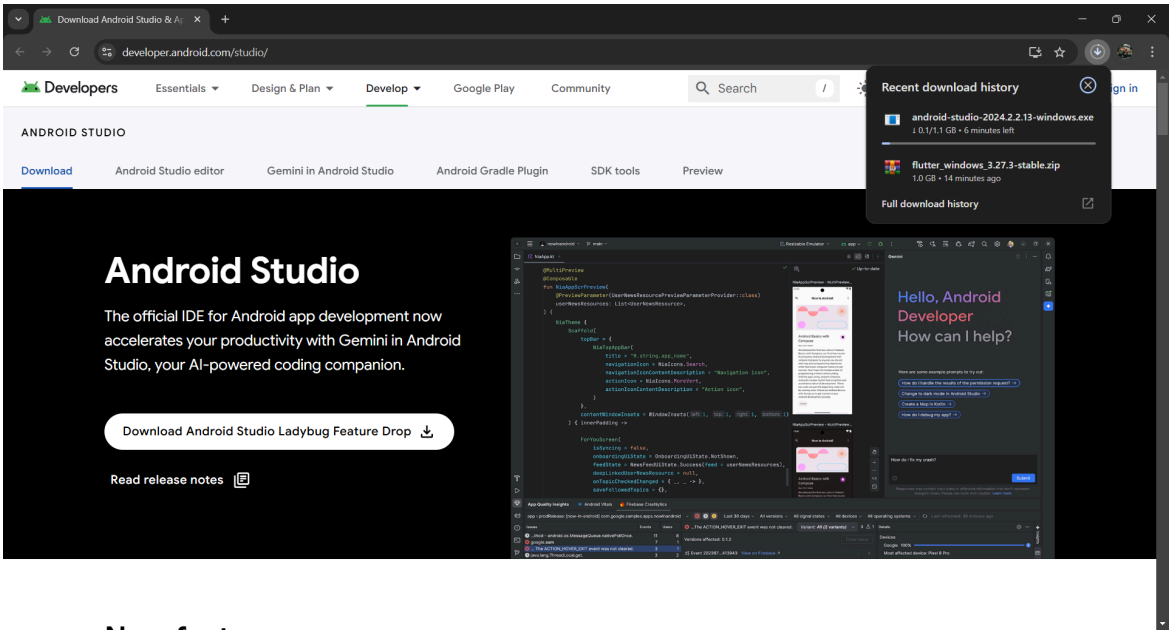
Step 6 : - Now, run the $ flutter command in command prompt.



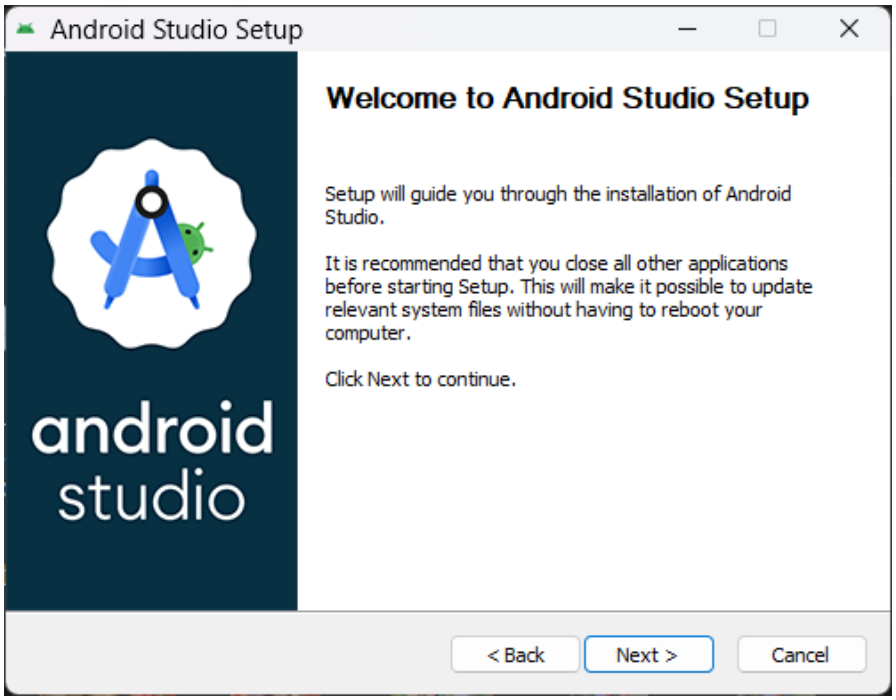
Step 7:- Run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation

```
Command Prompt - flut

C:\Users\shrey>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.26100.2894], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[X] Android toolchain - develop for Android devices
    X cmdline-tools component is missing
      Run `path/to/sdkmanager --install "cmdline-tools;latest"`
      See https://developer.android.com/studio/command-line 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 (not installed)
[√] VS Code (version 1.94.2)
[√] Connected device (3 available)
[√] Network resources

! Doctor found issues in 3 categories.

C:\Users\shrey>
```

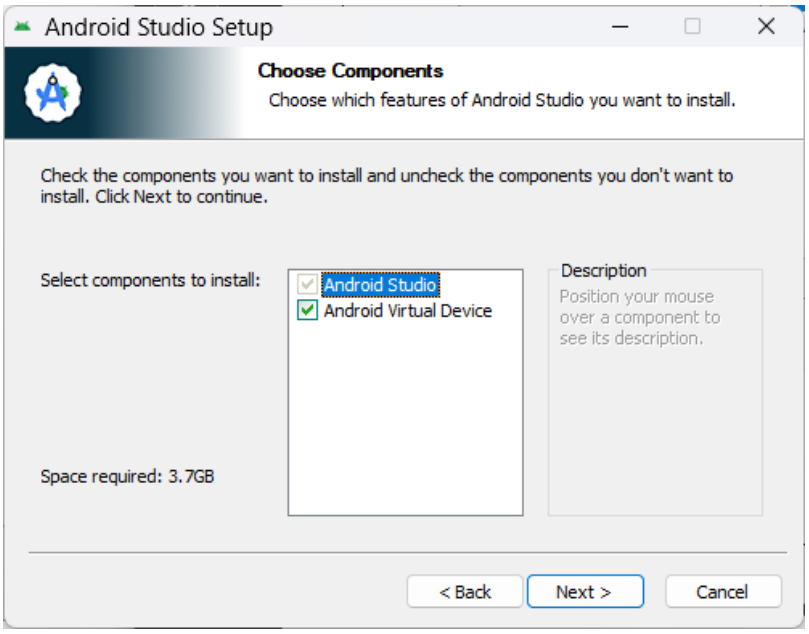
Step 8 : - Go to Android Studio and download the installer.



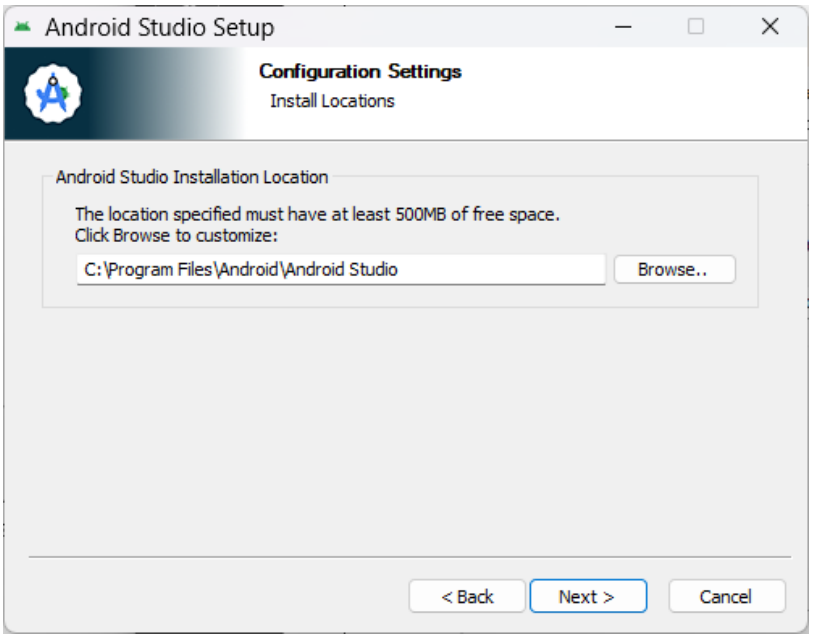
Step 8.1: - When the download is complete, open the .exe file and run it. You will get the following dialog box



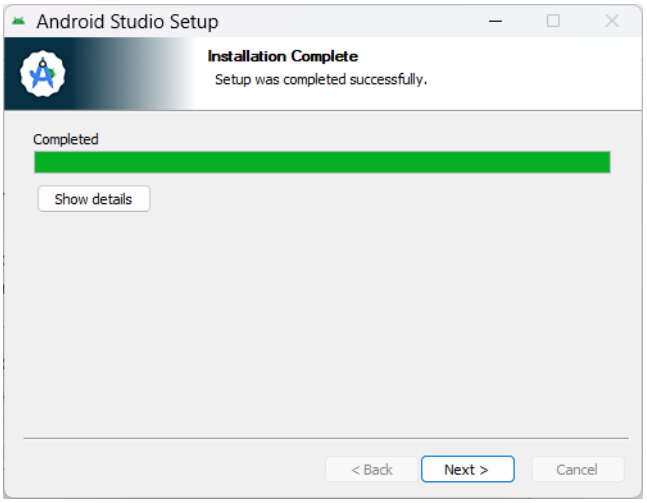
Step 8.2: - Select all the Checkboxes and Click on ‘Next’ Button.

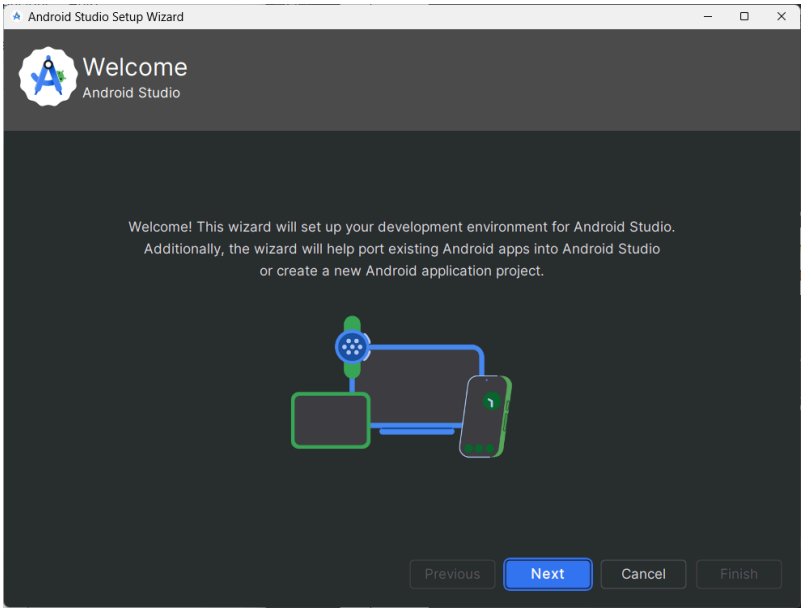


Step 8.3: - Change the destination as per your convenience and click on ‘Next’ Button.

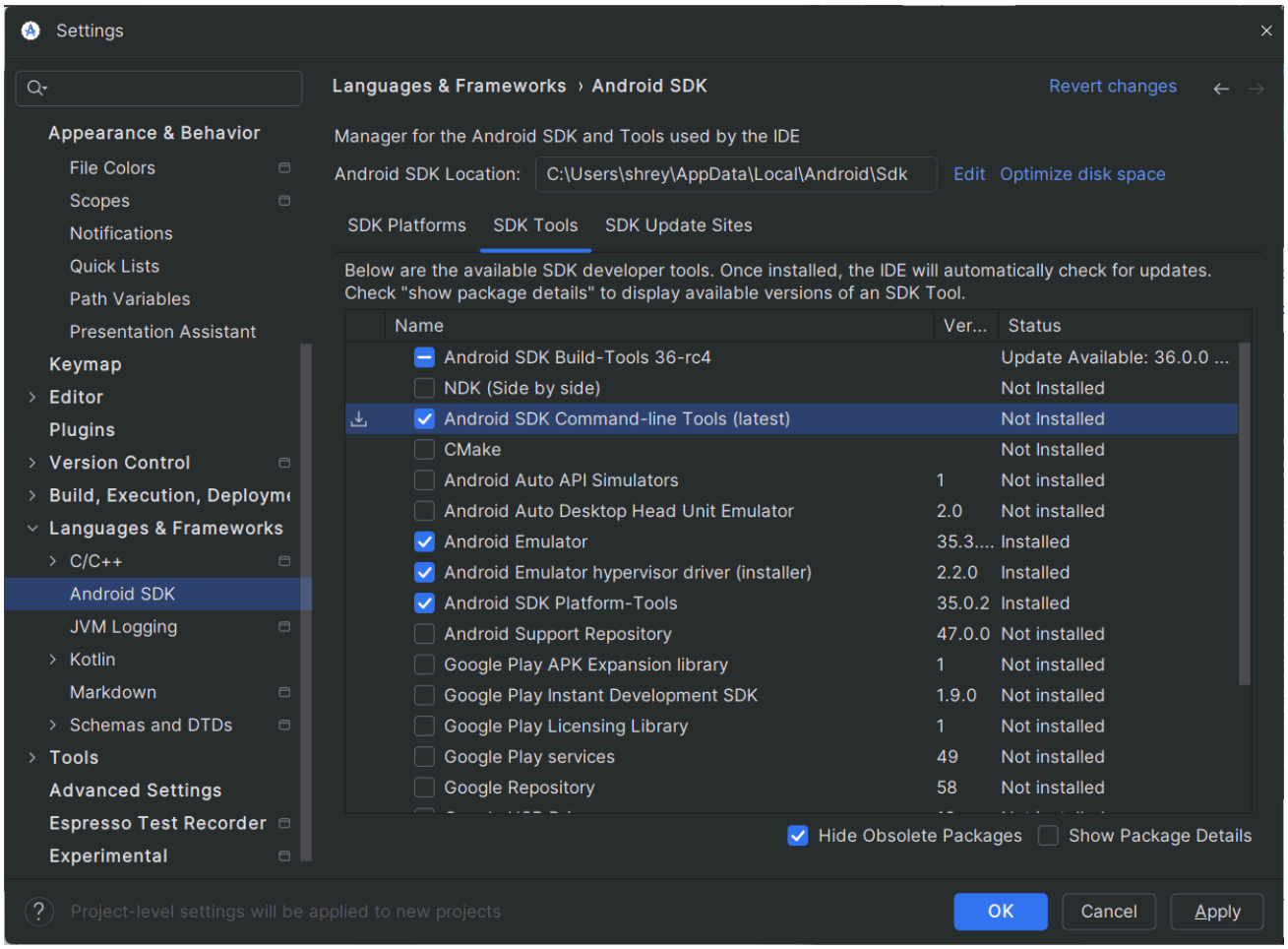


Step 8.4: - Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.

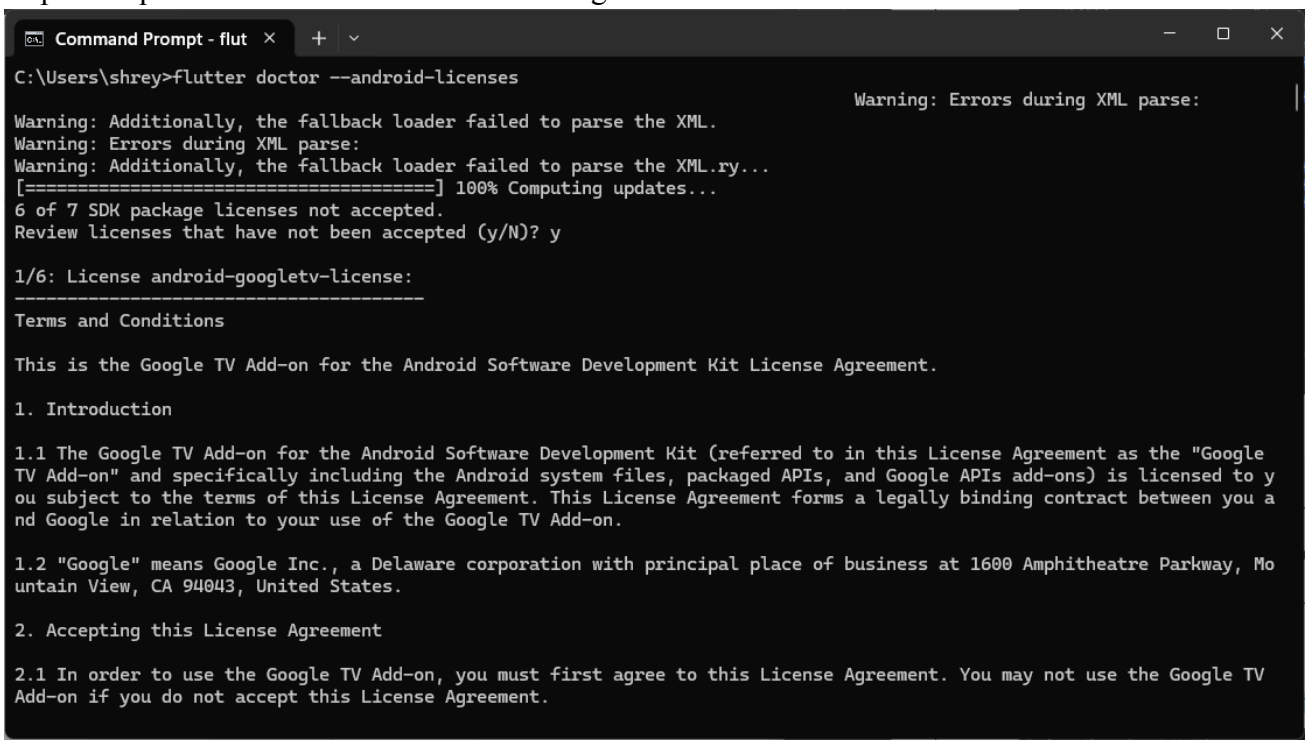


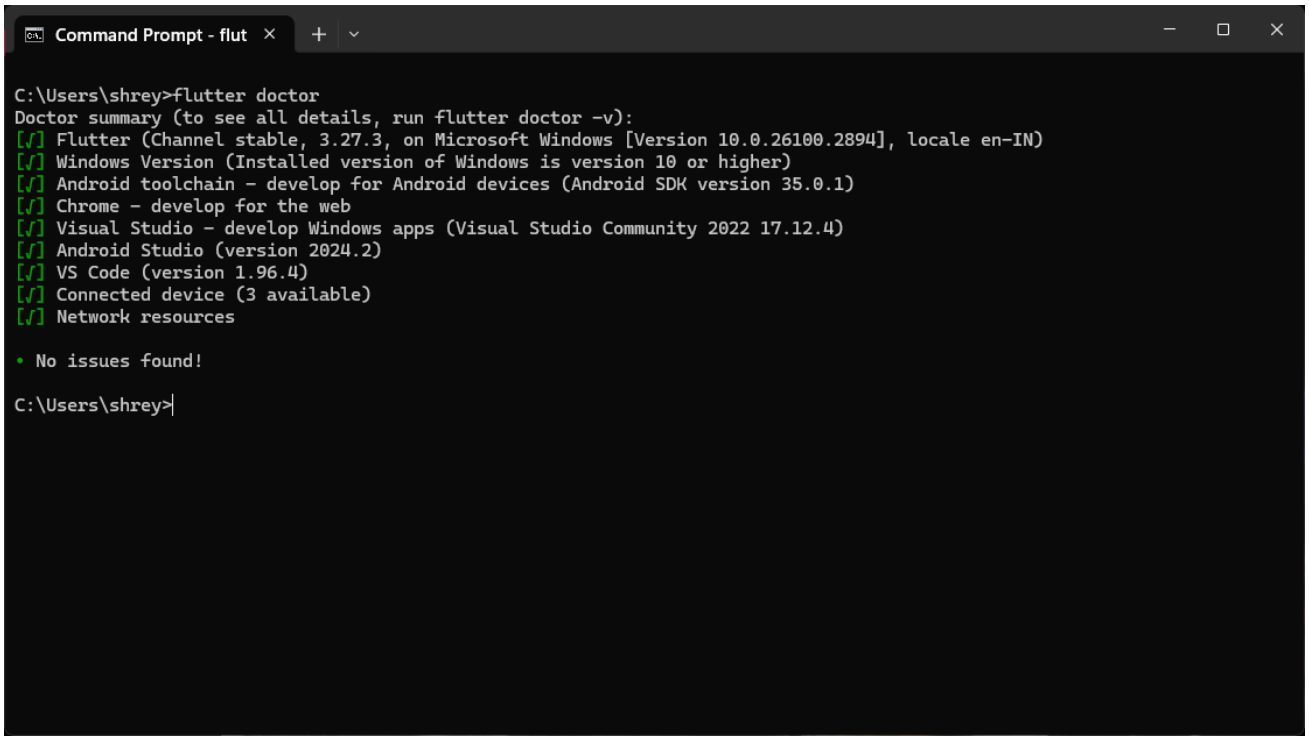


Step 8.5: - Go to Preferences > Appearance & Behavior > System Settings > Android SDK. Select the SDK Tools tab and check Android SDK Command-line Tools and Install it.

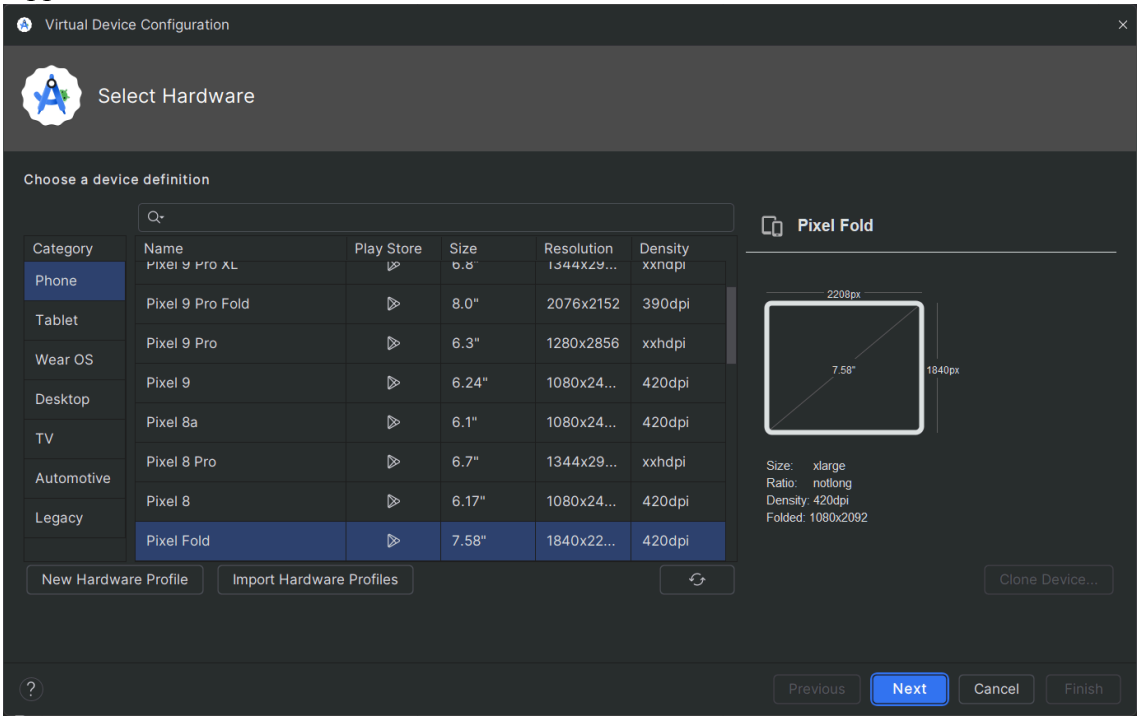


Step 9: - Open a terminal and run the following command

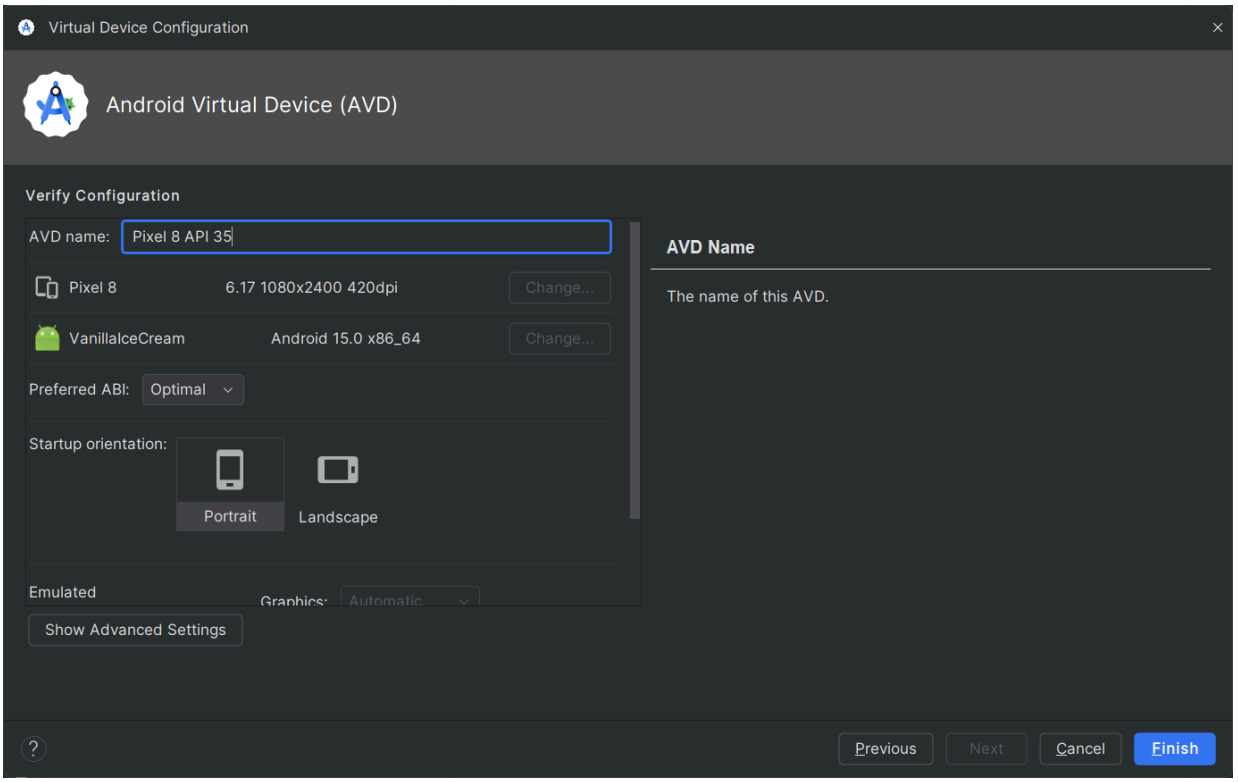




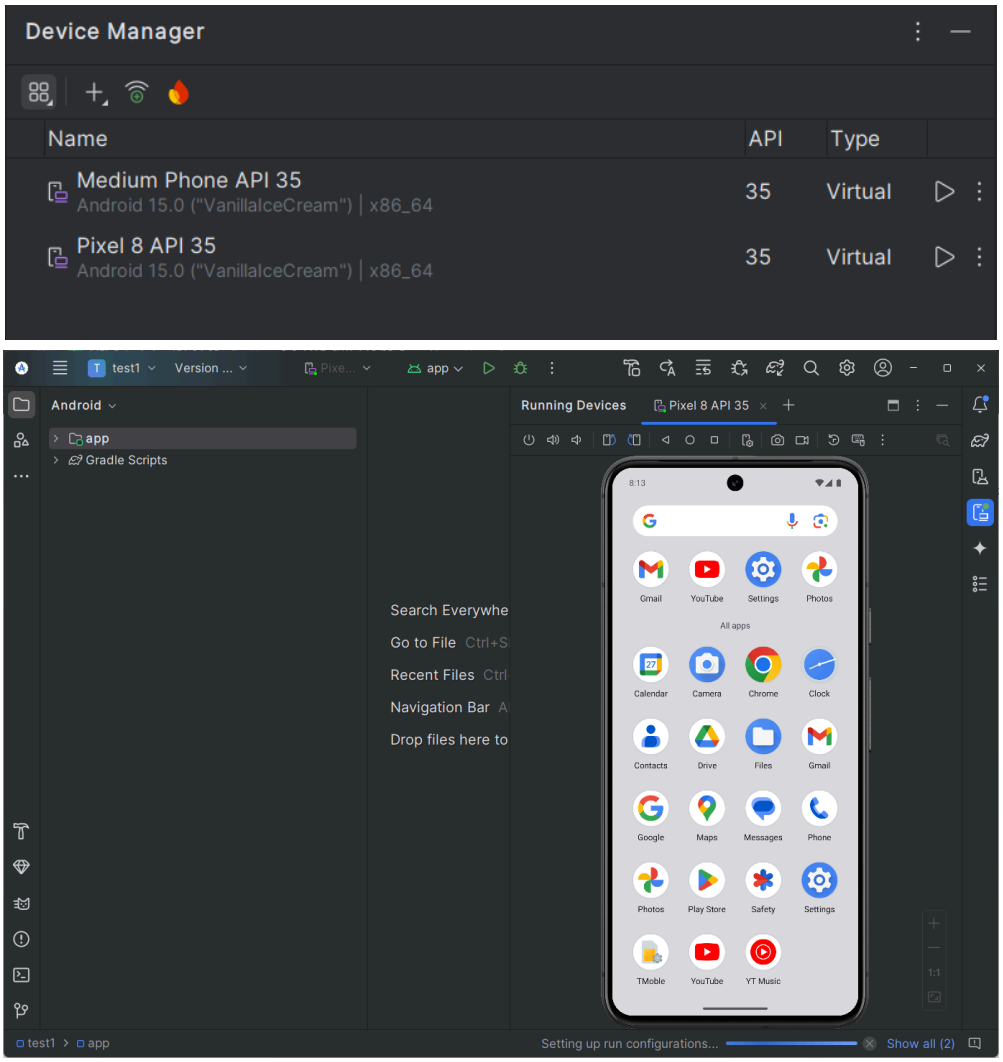
Step 10: - Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application



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

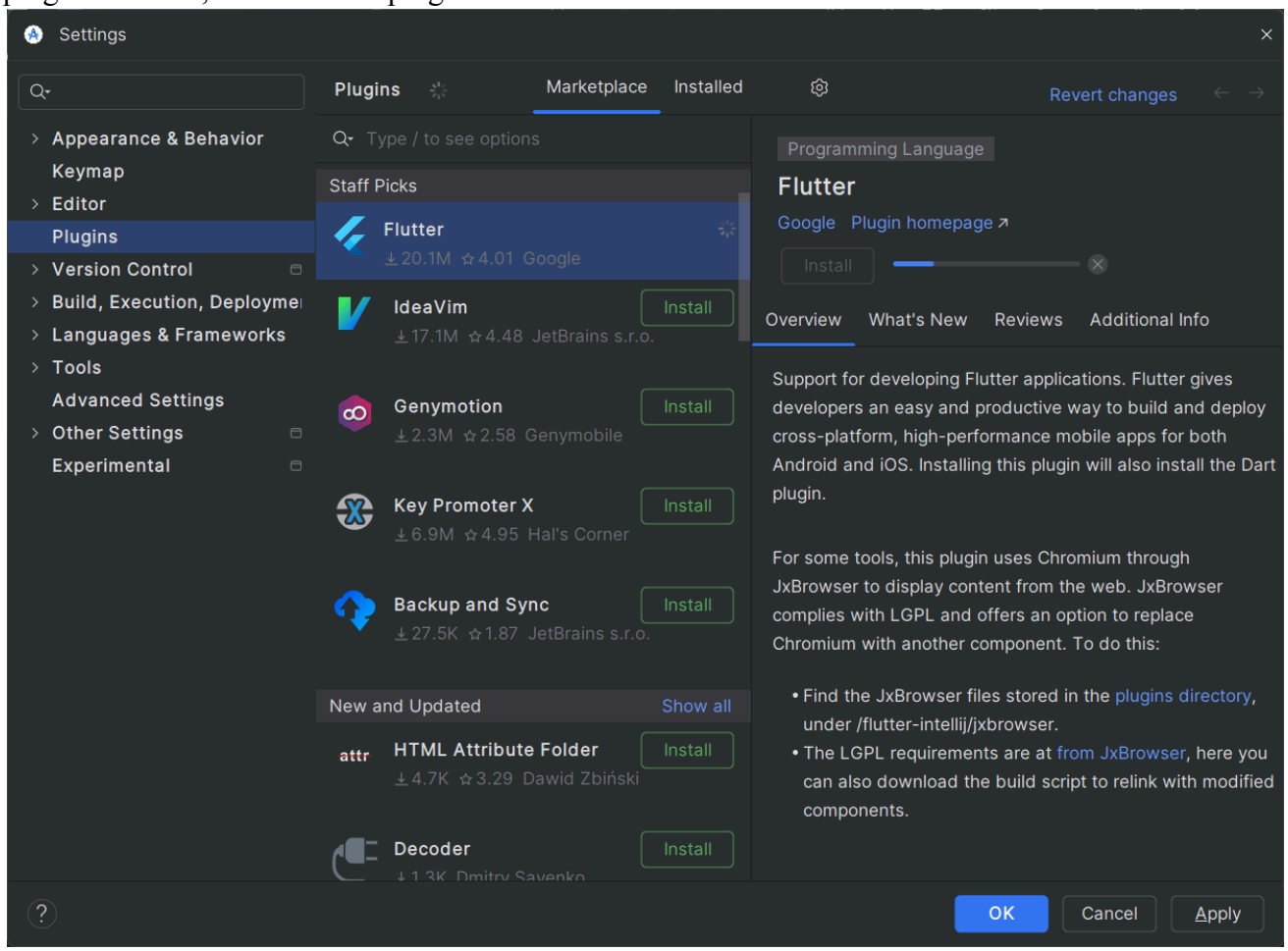


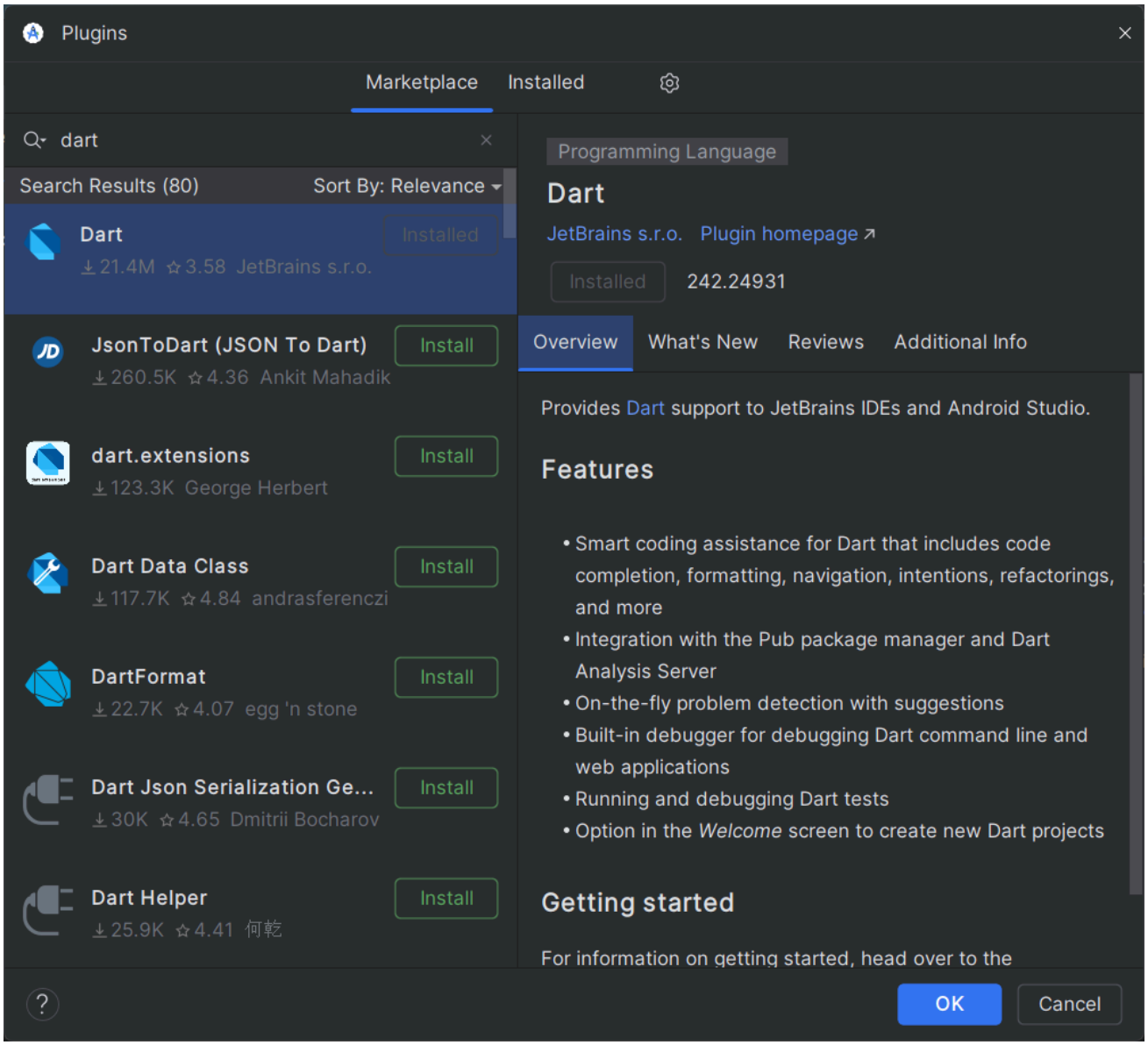
Step 10.2: - Click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen



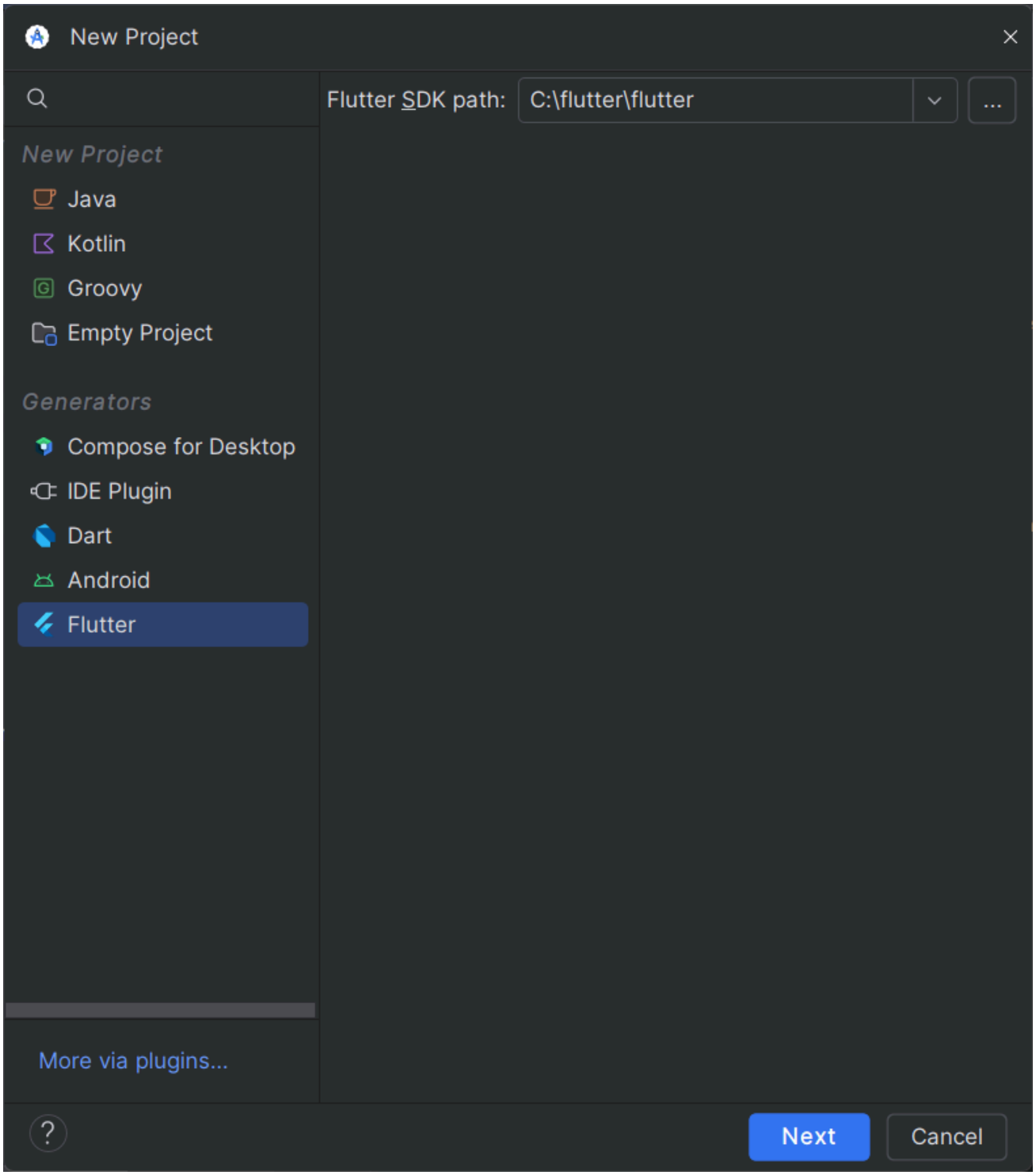
Step 11: - Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself

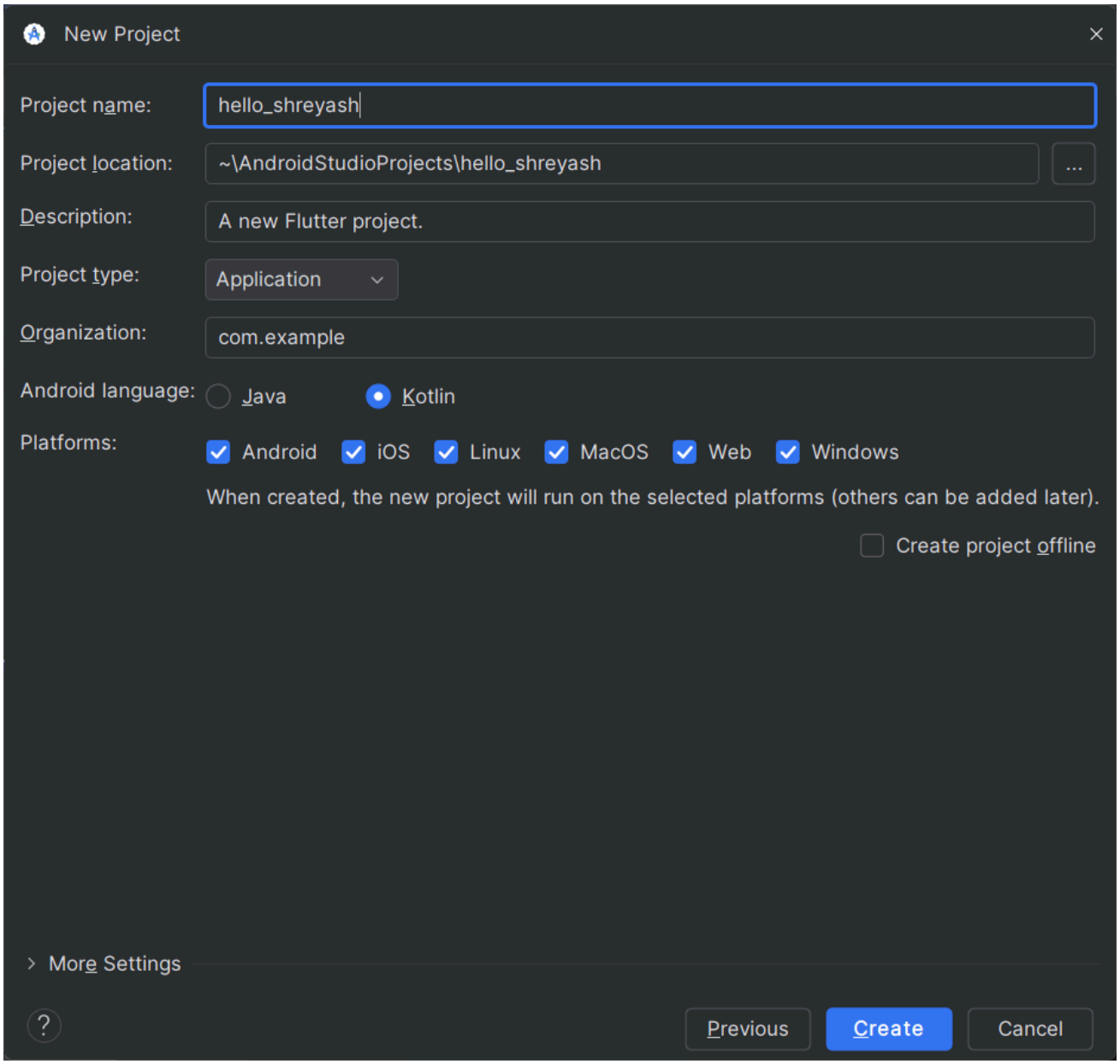
Step 11.1: - Open the Android Studio and then go to File->Settings->Plugins. Now, search the Flutter plugin. If found, select Flutter plugin and click install



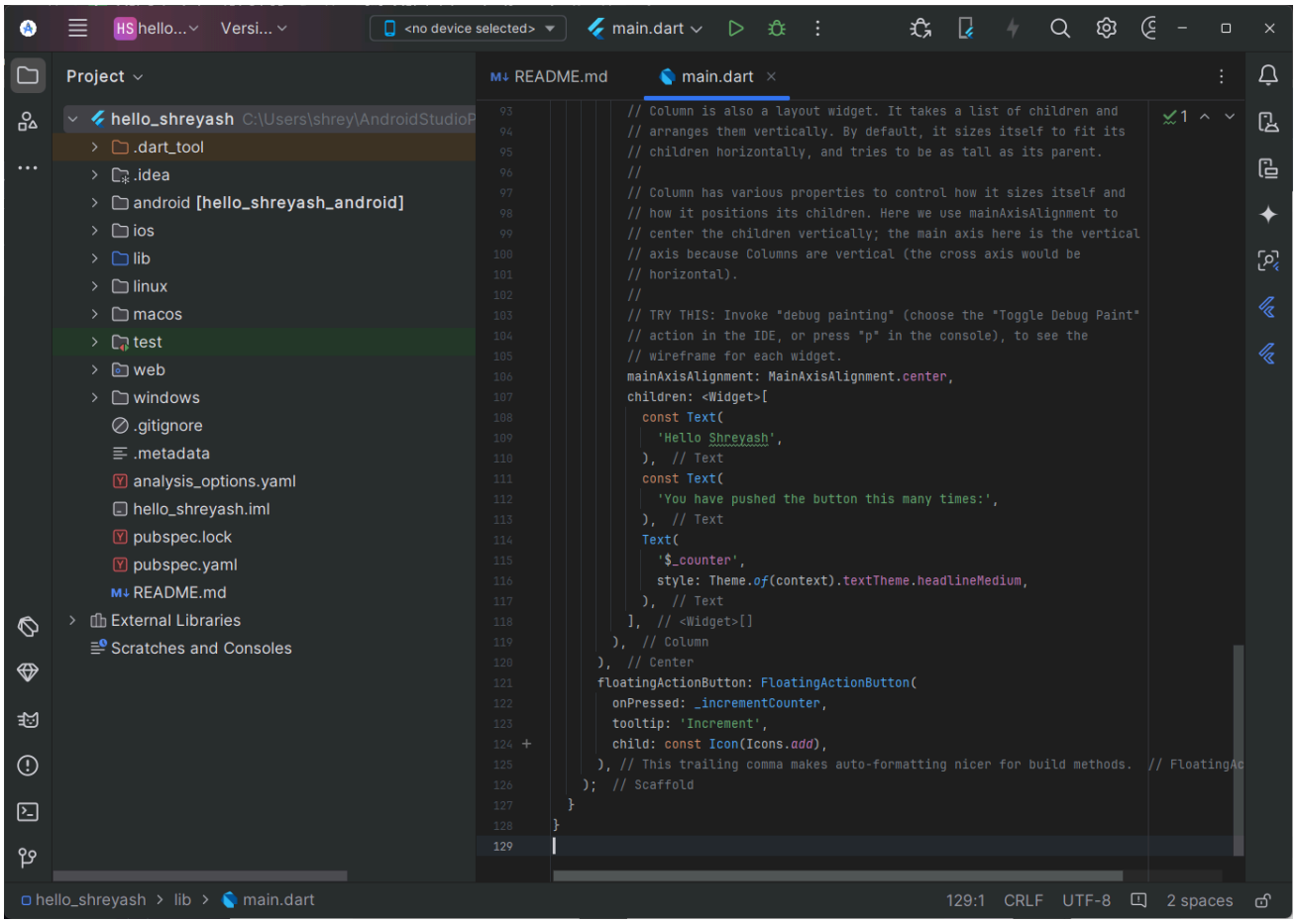


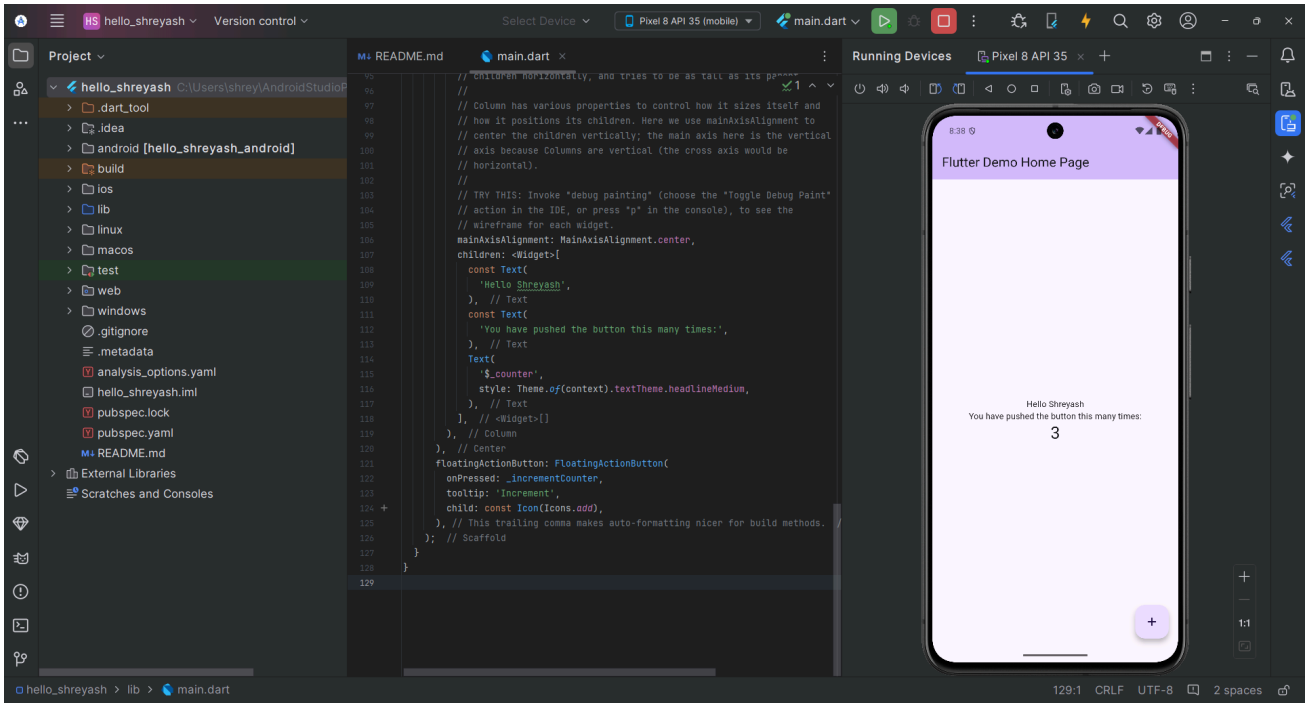
Step 11.2: - Restart the Android Studio
Step 12: - Go to File > New Project > Create Flutter Project, then select the project name and location, and click Next to proceed.

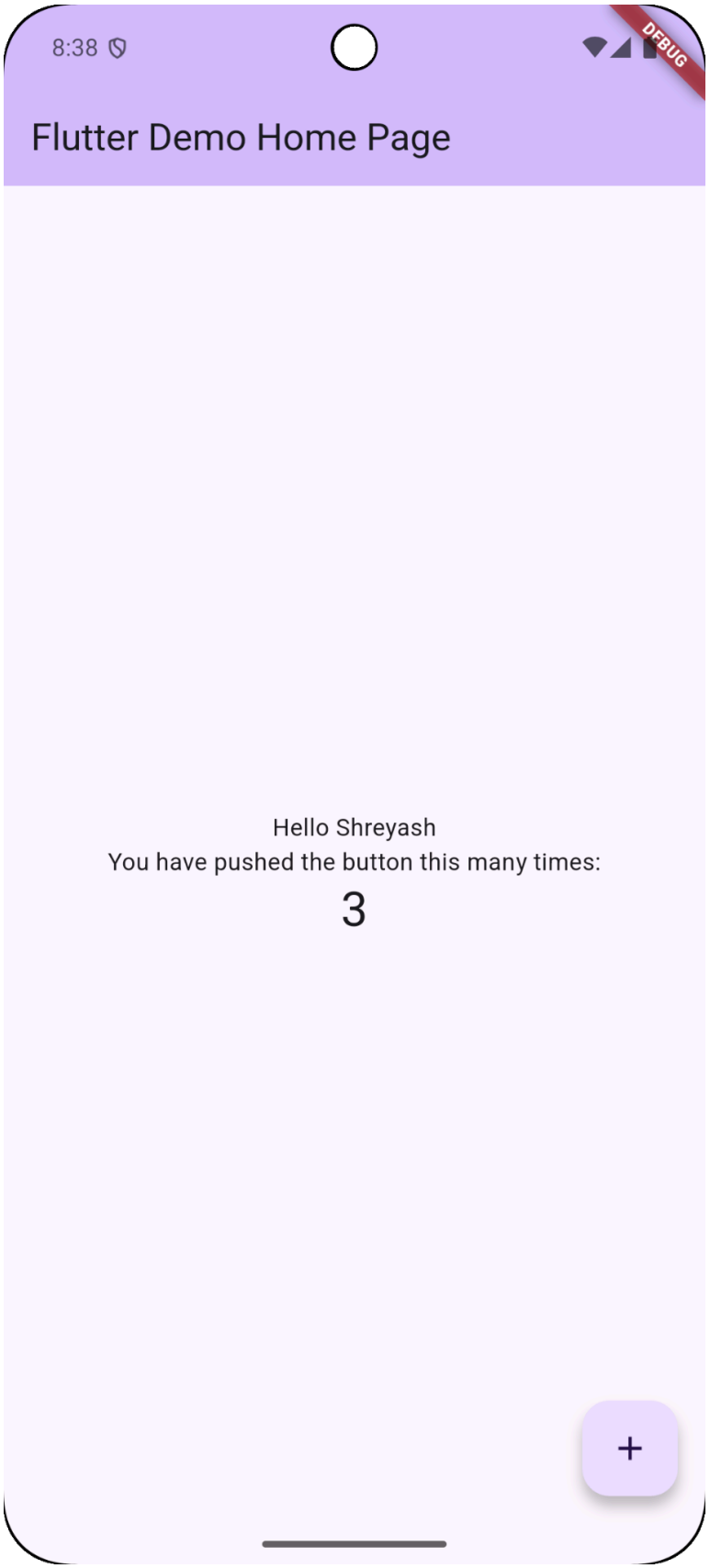

New Project
Flutter SDK path:
C:\flutter\flutter
...
New Project
Java
Kotlin
Groovy
Empty Project
Generators
Compose for Desktop
IDE Plugin
Dart
Android
Flutter
More via plugins...
Next
Cancel


New Project
Project name:
hello_shreyash
Project location:
~\AndroidStudioProjects\hello_shreyash
...
Description:
A new Flutter project.
Project type:
Application
Organization:
com.example
Android language:
Java
Kotlin
Platforms:
Android
iOS
Linux
MacOS
Web
Windows
When created, the new project will run on the selected platforms (others can be added later).
Create project offline
More Settings
Previous
Create
Cancel

Shreyash Dhekane D15A Roll No:16






8:38
DEBUG
Flutter Demo Home Page
Hello Shreyash
You have pushed the button this many times:
3
+

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	16
Name	Shreyash Ganesh Dhekane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO :- 02

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

Theory: -

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

Type of Widgets

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

Some of the commonly used widgets

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

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

Code:

home_screen.dart

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

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

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

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

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

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Home"),
        backgroundColor: Colors.black,
        centerTitle: true,
      ),
      backgroundColor: Colors.black,
      body: SingleChildScrollView(
```

```
      child: Column(
        children: [
          // Carousel Section with Custom
Widgets
          SizedBox(
            height: 200,
            child: PageView.builder(
              controller: _pageController,
              itemCount: carouselData.length,
              onPageChanged: (index) {
                setState(() {
                  _currentCarouselIndex = index;
                });
              },
              itemBuilder: (context, index) {
                final item = carouselData[index];
                return Container(
                  margin: const
EdgeInsets.symmetric(horizontal: 10, vertical:
10),
                  decoration: BoxDecoration(
                    borderRadius:
BorderRadius.circular(12),
                    color:
item['color'].withOpacity(0.2),
                  ),
                  padding: const
EdgeInsets.all(16),
                  child: Row(
                    children: [
                      Icon(
                        item['icon'],
                        size: 48,
                        color: item['color'],
                      ),
                      const SizedBox(width: 16),
                      Expanded(
                        child: Column(
                          mainAxisAlignment:
MainAxisAlignment.center,
                          crossAxisAlignment:
CrossAxisAlignment.start,
                          children: [
                            Text(
                              item['title'],
                              style: const TextStyle(
                                color: Colors.white,
                                fontSize: 20,
                                fontWeight:
FontWeight.bold,
                              ),
                            ),
                            const SizedBox(height: 8),
                            Text(
                              item['description'],
                              style: const TextStyle(
                                color: Colors.white70,
                                fontSize: 16,
                              ),
                            ),
                          ],
                        ),
                      ],
                    ),
                  ],
                ),
              );
            },
          ),
          // Carousel Indicator Dots
          Row(
            mainAxisAlignment:
MainAxisAlignment.center,
            children:
List.generate(carouselData.length, (index) {
              return AnimatedContainer(
                duration: const
Duration(milliseconds: 300),
                margin: const
EdgeInsets.symmetric(horizontal: 4),
                height: 8,
                width: _currentCarouselIndex ==
index ? 20 : 8,
                decoration: BoxDecoration(
                  color: _currentCarouselIndex ==
index ? Colors.blue : Colors.white54,
                  borderRadius:
BorderRadius.circular(4),
                ),
              );
            }),
          ),
          const SizedBox(height: 20),

          // Trackers Section
          Padding(
            padding: const
EdgeInsets.symmetric(horizontal: 16),
            child: Row(
              children: const [
```

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

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

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

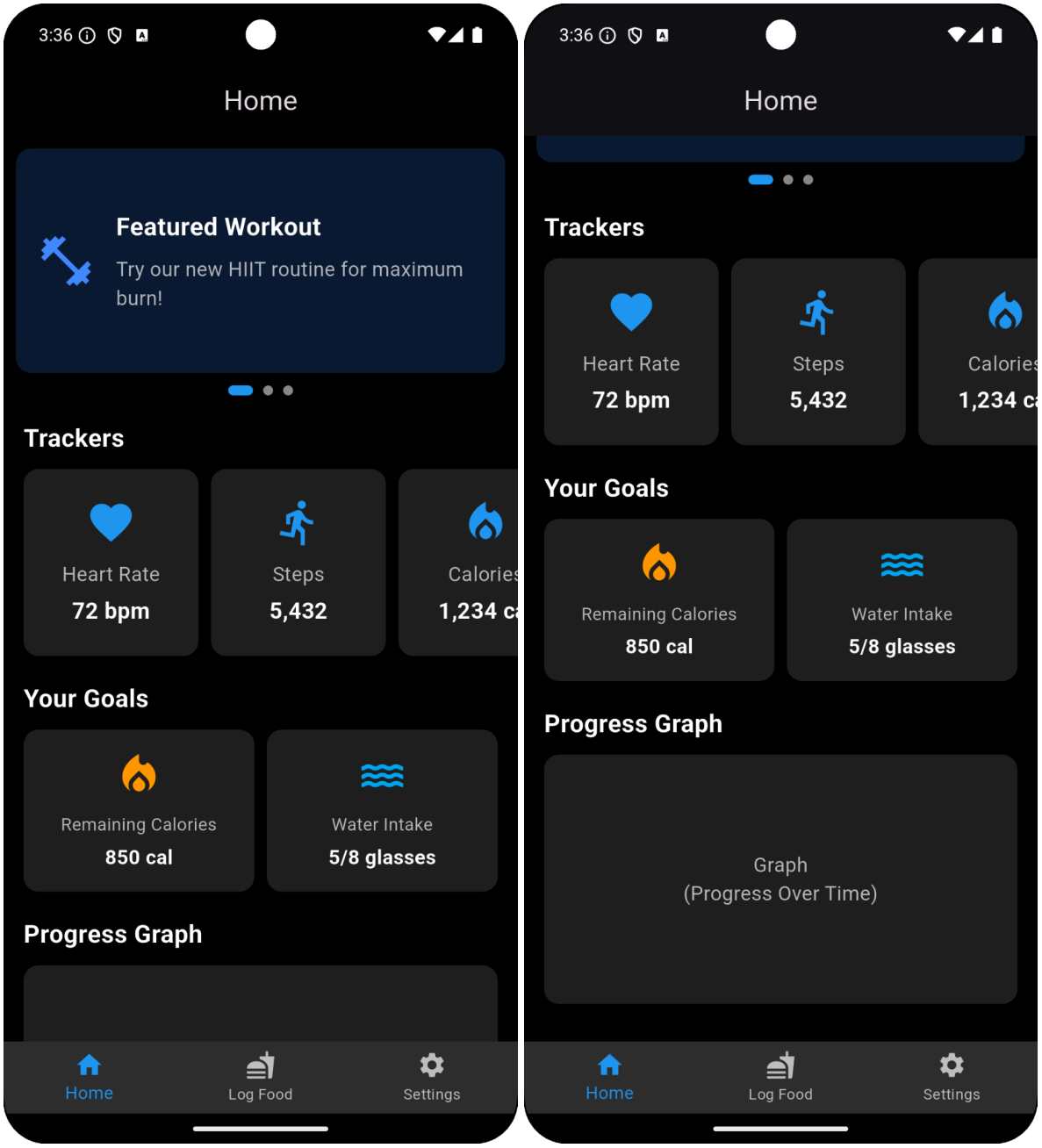
```
                ),
              ],
            ),
          ),
          const SizedBox(height: 10),
          _buildGraphWidget(),
          const SizedBox(height: 30),
        ],
      ),
    ),
  );
}

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

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

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

```
      borderRadius: BorderRadius.circular(12),
     ),
     child: const Center(
      child: Text(
       "Graph \n(Progress Over Time)",
       textAlign: TextAlign.center,
       style: TextStyle(
        color: Colors.white70,
        fontSize: 16,
       ),
      ),
     ),
    );
  }
}
```

main_screen.dart

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

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

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

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

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

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

Screenshots:



MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	16
Name	Shreyash Ganesh Dhekane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO :- 03

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

Theory: -

Flutter is a versatile open-source UI framework , which allows developers to build natively compiled applications for mobile, web, and desktop platforms from a single codebase. One of the key strengths of Flutter is its flexibility in creating highly customizable UIs. This practical focuses on incorporating essential visual elements—icons, images, and custom fonts—into a Flutter application. These elements enhance the visual appeal and usability of the app, providing an engaging experience for users.
A flutter app when built has both assets (resources) and code. Assets are available and deployed during runtime. The asset is a file that can include static data, configuration files, icons, and images. The Flutter app supports many image formats, such as JPEG, WebP, PNG, GIF, animated WebP/GIF, BMP, and WBMP.
Visual elements play a significant role in app development.

- Enhanced User Experience: Images and icons make your app visually appealing and user-friendly.
- Information Conveyance: They convey information quickly and intuitively. A well-chosen icon can replace lengthy text.
- Branding: Custom icons and images reinforce your app's branding, making it memorable.

Adding Icons in Flutter

Flutter provides built-in material design icons through the Icons class. Custom icons can also be added using third-party packages such as flutter_launcher_icons and font_awesome_flutter.

```
Icon(
Icons.home,
size: 40,
);
```

Adding Images in Flutter

Flutter supports images from three sources:

1.Assets (Stored locally in the project)

- Place the image inside the assets/images folder in the project.
  Declare the image in pubspec.yaml

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

- Display the image in the app

```
Image.asset('assets/images/sample.png');
```

2.Network (Fetched from the internet)

- Displaying images from the internet or network is very simple. Flutter provides a built-in method Image.network to work with images from a URL. The Image.network method also allows you to use some optional properties, such as height, width, color, fit, and many more.

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

3.Memory or File (Stored on the device)

Adding Custom Fonts in Flutter

By default, Flutter uses the Roboto font, but custom fonts can be added for a unique UI.

- Download the font and place it in the assets/fonts/ folder.
- Declare the font in pubspec.yaml
- Use the font in the app

```
Text(
'Custom Font Example',
style: TextStyle(fontFamily: 'CustomFont', fontSize: 24),
);
```

Code: -

login_screen.dart

```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'main_screen.dart'; // Import the MainScreen (not HomeScreen)

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

  @override
  State<LoginScreen> createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  // Controllers for input fields
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();

  // Firebase Auth instance
  final FirebaseAuth _auth = FirebaseAuth.instance;

  bool _isLoading = false;

  @override
  void dispose() {
    emailController.dispose();
    passwordController.dispose();
    super.dispose();
  }

  /// Sign in using Firebase Authentication
  Future<void> _login() async {
    setState(() {
      _isLoading = true;
    });

    try {
      // Attempt sign in with email and password
      await _auth.signInWithEmailAndPassword(
        email: emailController.text.trim(),
        password: passwordController.text,
      );

      // On successful login, navigate to the MainScreen with BottomNavigationBar.
      Navigator.pushReplacement(
        context,
```

```
      MaterialPageRoute(builder: (context) =>
const MainScreen()), // Navigate to
MainScreen
      );
    } on FirebaseAuthException catch (e) {
      String message = 'Login failed';
      if (e.code == 'user-not-found') {
        message = 'No user found for that
email.';
      } else if (e.code == 'wrong-password') {
        message = 'Incorrect password
provided.';
      }

ScaffoldMessenger.of(context).showSnackBar
(
        SnackBar(content: Text(message)),
      );
    } catch (e) {

ScaffoldMessenger.of(context).showSnackBar
(
        const SnackBar(content: Text('An error
occurred.')),
      );
    }

    setState(() {
      _isLoading = false;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      appBar: AppBar(
        backgroundColor: Colors.black,
        title: const Text("Log In", style:
TextStyle(color: Colors.white)),
        centerTitle: true,
        actions: [
          // Skip button for direct access (login
later)
          TextButton(
            onPressed: () {
              Navigator.pushReplacement(
                context,
                MaterialPageRoute(builder:
(context) => const MainScreen()), // Navigate
to MainScreen
              );
            },
            child: const Text("Skip", style:
TextStyle(color: Colors.white, fontSize: 16)),
          ),
        ],
        leading: IconButton(
          icon: const Icon(Icons.arrow_back,
color: Colors.white),
          onPressed: () {
            Navigator.pop(context);
          },
        ),
      ),
      body: Padding(
        padding: const
EdgeInsets.symmetric(horizontal: 20),
        child: Column(
          children: [
            const SizedBox(height: 30),
            // Email TextField
            TextField(
              controller: emailController,
              style: const TextStyle(color:
Colors.white),
              decoration: InputDecoration(
                labelText: "Email Address",
                labelStyle: const TextStyle(color:
Colors.white70),
                enabledBorder: OutlineInputBorder(
                  borderSide: const
BorderSide(color: Colors.white38),
                  borderRadius:
BorderRadius.circular(8),
                ),
                focusedBorder: OutlineInputBorder(
                  borderSide: const
BorderSide(color: Colors.blue),
                  borderRadius:
BorderRadius.circular(8),
                ),
              ),
            ),
            const SizedBox(height: 20),
            // Password TextField
            TextField(
```

```
            controller: passwordController,
            obscureText: true,
            style: const TextStyle(color:
Colors.white),
            decoration: InputDecoration(
              labelText: "Password",
              labelStyle: const TextStyle(color:
Colors.white70),
              enabledBorder: OutlineInputBorder(
                borderSide: const
BorderSide(color: Colors.white38),
                borderRadius:
BorderRadius.circular(8),
              ),
              focusedBorder: OutlineInputBorder(
                borderSide: const
BorderSide(color: Colors.blue),
                borderRadius:
BorderRadius.circular(8),
              ),
            ),
          ),
          const SizedBox(height: 20),
          // Login button or a loading indicator
          _isLoading
              ? const CircularProgressIndicator()
              : ElevatedButton(
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.blue,
              minimumSize: const
Size(double.infinity, 50),
            ),
            onPressed: _login,
            child: const Text("Log In", style:
TextStyle(color: Colors.white)),
          ),
          const SizedBox(height: 10),
          TextButton(
            onPressed: () {
              // Implement forgot password logic if
desired.
            },
            child: const Text("Forgot password?",
style: TextStyle(color: Colors.blue)),
          ),
          const SizedBox(height: 20),
          const Row(
            children: [
              Expanded(child: Divider(color:
Colors.white)),
              Padding(
                padding:
EdgeInsets.symmetric(horizontal: 10),
                child: Text("OR", style:
TextStyle(color: Colors.white)),
              ),
              Expanded(child: Divider(color:
Colors.white)),
            ],
          ),
          // Additional social login buttons can
be added here
          const Spacer(),
          // Navigation to registration screen
          TextButton(
            onPressed: () {
              Navigator.pushNamed(context,
'/register');
            },
            child: const Text("Don't have an
account? Register", style: TextStyle(color:
Colors.blue, fontSize: 16)),
          ),
          const SizedBox(height: 20),
        ],
      ),
    ),
  );
 }
}
```

registration_screen.dart

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

class RegistrationScreen extends
StatefulWidget {
```

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

  @override
  State<RegistrationScreen> createState() =>
_RegistrationScreenState();
}

class _RegistrationScreenState extends
State<RegistrationScreen> {
  // Controllers for input fields
  final TextEditingController nameController =
TextEditingController();
  final TextEditingController emailController =
TextEditingController();
  final TextEditingController
passwordController = TextEditingController();
  final TextEditingController
confirmPasswordController =
TextEditingController();

  // Firebase Auth instance
  final FirebaseAuth _auth =
FirebaseAuth.instance;

  bool _isLoading = false;

  @override
  void dispose() {
    nameController.dispose();
    emailController.dispose();
    passwordController.dispose();
    confirmPasswordController.dispose();
    super.dispose();
  }

  /// Register a new user with Firebase
Authentication
  Future<void> _register() async {
    final email = emailController.text.trim();
    final password = passwordController.text;
    final confirmPassword =
confirmPasswordController.text;

    if (password != confirmPassword) {

ScaffoldMessenger.of(context).showSnackBar
(
        const SnackBar(content:
Text("Passwords do not match")),
      );
      return;
    }

    setState(() {
      _isLoading = true;
    });

    try {
      // Create the user
      UserCredential userCredential = await
_auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );

      // Optionally update the display name
      if (userCredential.user != null &&
nameController.text.trim().isNotEmpty) {
        await
userCredential.user!.updateDisplayName(na
meController.text.trim());
      }

      // On successful registration, navigate to
main screen.

Navigator.pushReplacementNamed(context,
'/main');
    } on FirebaseAuthException catch (e) {
      String message = 'Registration failed';
      if (e.code == 'weak-password') {
        message = 'The password provided is
too weak.';
      } else if (e.code == 'email-already-in-use')
{
        message = 'An account already exists for
that email.';
      }

ScaffoldMessenger.of(context).showSnackBar
(
        SnackBar(content: Text(message)),
      );
    } catch (e) {

ScaffoldMessenger.of(context).showSnackBar
(
```

```
      const SnackBar(content: Text("An error
occurred")),
    );
  }

  setState(() {
    _isLoading = false;
  });
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   backgroundColor: Colors.black,
   appBar: AppBar(
    backgroundColor: Colors.black,
    title: const Text("Register", style:
TextStyle(color: Colors.white)),
    centerTitle: true,
    leading: IconButton(
     icon: const Icon(Icons.arrow_back,
color: Colors.white),
     onPressed: () {
      Navigator.pop(context);
     },
    ),
   ),
   body: SingleChildScrollView(
    padding: const
EdgeInsets.symmetric(horizontal: 20, vertical:
20),
    child: Column(
     children: [
      const SizedBox(height: 20),
      // Full Name field (optional)
      TextField(
       controller: nameController,
       style: const TextStyle(color:
Colors.white),
       decoration: InputDecoration(
        labelText: "Full Name",
        labelStyle: const TextStyle(color:
Colors.white70),
        enabledBorder: OutlineInputBorder(
         borderSide: const
BorderSide(color: Colors.white38),
         borderRadius:
BorderRadius.circular(8),
        ),
        focusedBorder: OutlineInputBorder(
         borderSide: const
BorderSide(color: Colors.blue),
         borderRadius:
BorderRadius.circular(8),
        ),
       ),
      ),
      const SizedBox(height: 20),
      // Email Address field
      TextField(
       controller: emailController,
       style: const TextStyle(color:
Colors.white),
       decoration: InputDecoration(
        labelText: "Email Address",
        labelStyle: const TextStyle(color:
Colors.white70),
        enabledBorder: OutlineInputBorder(
         borderSide: const
BorderSide(color: Colors.white38),
         borderRadius:
BorderRadius.circular(8),
        ),
        focusedBorder: OutlineInputBorder(
         borderSide: const
BorderSide(color: Colors.blue),
         borderRadius:
BorderRadius.circular(8),
        ),
       ),
      ),
      const SizedBox(height: 20),
      // Password field
      TextField(
       controller: passwordController,
       obscureText: true,
       style: const TextStyle(color:
Colors.white),
       decoration: InputDecoration(
        labelText: "Password",
        labelStyle: const TextStyle(color:
Colors.white70),
        enabledBorder: OutlineInputBorder(
         borderSide: const
BorderSide(color: Colors.white38),
         borderRadius:
BorderRadius.circular(8),
        ),
```

```
              focusedBorder: OutlineInputBorder(
                borderSide: const
BorderSide(color: Colors.blue),
                borderRadius:
BorderRadius.circular(8),
              ),
            ),
          ),
          const SizedBox(height: 20),
          // Confirm Password field
          TextField(
            controller:
confirmPasswordController,
            obscureText: true,
            style: const TextStyle(color:
Colors.white),
            decoration: InputDecoration(
              labelText: "Confirm Password",
              labelStyle: const TextStyle(color:
Colors.white70),
              enabledBorder: OutlineInputBorder(
                borderSide: const
BorderSide(color: Colors.white38),
                borderRadius:
BorderRadius.circular(8),
              ),
              focusedBorder: OutlineInputBorder(
                borderSide: const
BorderSide(color: Colors.blue),
                borderRadius:
BorderRadius.circular(8),
import 'package:flutter/material.dart';

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

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

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

  final List<Map<String, dynamic>>
carouselData = [
              ),
            ),
          ),
          const SizedBox(height: 20),
          _isLoading
              ? const CircularProgressIndicator()
              : ElevatedButton(
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.blue,
              minimumSize: const
Size(double.infinity, 50),
              ),
              onPressed: _register,
              child: const Text("Register", style:
TextStyle(color: Colors.white)),
            ),
          const SizedBox(height: 10),
          // Optionally add a link to the login
page if already have an account.
        ],
      ),
    ),
  );
 }
}
```

home_screen.dart

```
    {
      'icon': Icons.fitness_center,
      'title': 'Featured Workout',
      'description': 'Try our new HIIT routine for
maximum burn!',
      'color': Colors.blueAccent,
    },
    {
      'icon': Icons.local_dining,
      'title': 'Nutrition Tip',
      'description': 'Include more protein in your
meals for muscle gain.',
      'color': Colors.green,
    },
    {
      'icon': Icons.self_improvement,
      'title': 'Mindfulness',
```

```
    'description': 'Take a moment to meditate and destress.',
    'color': Colors.purple,
  },
 ];

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

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: const Text("Home"),
    backgroundColor: Colors.black,
    centerTitle: true,
   ),
   backgroundColor: Colors.black,
   body: SingleChildScrollView(
    child: Column(
     children: [
      // Carousel Section with Custom Widgets
      SizedBox(
       height: 200,
       child: PageView.builder(
        controller: _pageController,
        itemCount: carouselData.length,
        onPageChanged: (index) {
         setState(() {
          _currentCarouselIndex = index;
         });
        },
        itemBuilder: (context, index) {
         final item = carouselData[index];
         return Container(
          margin: const EdgeInsets.symmetric(horizontal: 10, vertical: 10),
          decoration: BoxDecoration(
           borderRadius: BorderRadius.circular(12),
           color: item['color'].withOpacity(0.2),
          ),
          padding: const EdgeInsets.all(16),
          child: Row(
           children: [
            Icon(
             item['icon'],
             size: 48,
             color: item['color'],
            ),
            const SizedBox(width: 16),
            Expanded(
             child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
               Text(
                item['title'],
                style: const TextStyle(
                 color: Colors.white,
                 fontSize: 20,
                 fontWeight: FontWeight.bold,
                ),
               ),
               const SizedBox(height: 8),
               Text(
                item['description'],
                style: const TextStyle(
                 color: Colors.white70,
                 fontSize: 16,
                ),
               ),
              ],
             ),
            ),
           ],
          ),
         );
        },
       ),
      ),
      // Carousel Indicator Dots
      Row(
       mainAxisAlignment: MainAxisAlignment.center,
       children: List.generate(carouselData.length, (index) {
```

```
            return AnimatedContainer(
              duration: const
Duration(milliseconds: 300),
              margin: const
EdgeInsets.symmetric(horizontal: 4),
              height: 8,
              width: _currentCarouselIndex ==
index ? 20 : 8,
              decoration: BoxDecoration(
                color: _currentCarouselIndex ==
index ? Colors.blue : Colors.white54,
                borderRadius:
BorderRadius.circular(4),
              ),
            );
          }),
        ),
        const SizedBox(height: 20),

        // Trackers Section
        Padding(
          padding: const
EdgeInsets.symmetric(horizontal: 16),
          child: Row(
            children: const [
              Text(
                "Trackers",
                style: TextStyle(
                  color: Colors.white,
                  fontSize: 20,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ],
          ),
        ),
        const SizedBox(height: 10),
        SizedBox(
          height: 150,
          child: ListView(
            scrollDirection: Axis.horizontal,
            padding: const
EdgeInsets.symmetric(horizontal: 16),
            children: [
              _buildTrackerCard(
                icon: Icons.favorite,
                title: "Heart Rate",
                value: "72 bpm",
              ),
              const SizedBox(width: 10),
              _buildTrackerCard(
                icon: Icons.directions_run,
                title: "Steps",
                value: "5,432",
              ),
              const SizedBox(width: 10),
              _buildTrackerCard(
                icon: Icons.local_fire_department,
                title: "Calories",
                value: "1,234 cal",
              ),
            ],
          ),
        ),
        const SizedBox(height: 20),

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

```
                const SizedBox(width: 10),
                Expanded(
                  child: _buildGoalCard(
                    title: "Water Intake",
                    value: "5/8 glasses",
                    icon: Icons.water,
                    color: Colors.lightBlue,
                  ),
                ),
              ],
            ),
          ),
          const SizedBox(height: 20),

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

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

// Goal Card
Widget _buildGoalCard({
  required String title,
  required String value,
  required IconData icon,
  required Color color,
}) {
  return Container(
    padding: const EdgeInsets.all(16),
    decoration: BoxDecoration(
      color: Colors.grey[900],
      borderRadius: BorderRadius.circular(12),
    ),
    child: Column(
      children: [
        Icon(icon, size: 40, color: color),
        const SizedBox(height: 10),
        Text(
```

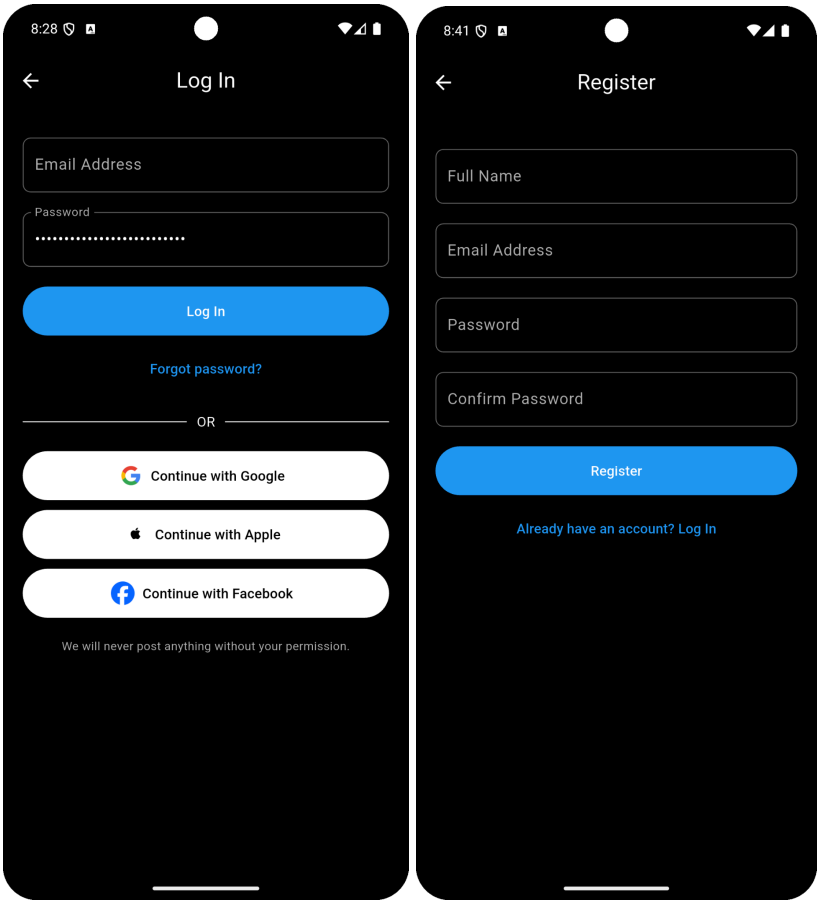
```
          title,
          textAlign: TextAlign.center,
          style: const TextStyle(
            color: Colors.white70,
            fontSize: 14,
          ),
        ),
        const SizedBox(height: 5),
        Text(
          value,
          style: const TextStyle(
            color: Colors.white,
            fontSize: 16,
            fontWeight: FontWeight.bold,
          ),
        )
      ],
    ),
  );
 }

 // Placeholder for Graph Widget
 Widget _buildGraphWidget() {
  return Container(
    margin: const
EdgeInsets.symmetric(horizontal: 16),
    height: 200,
    decoration: BoxDecoration(
      color: Colors.grey[900],
      borderRadius: BorderRadius.circular(12),
    ),
    child: const Center(
      child: Text(
        "Graph \n(Progress Over Time)",
        textAlign: TextAlign.center,
        style: TextStyle(
          color: Colors.white70,
          fontSize: 16,
        ),
      ),
    ),
  );
 }
}
```

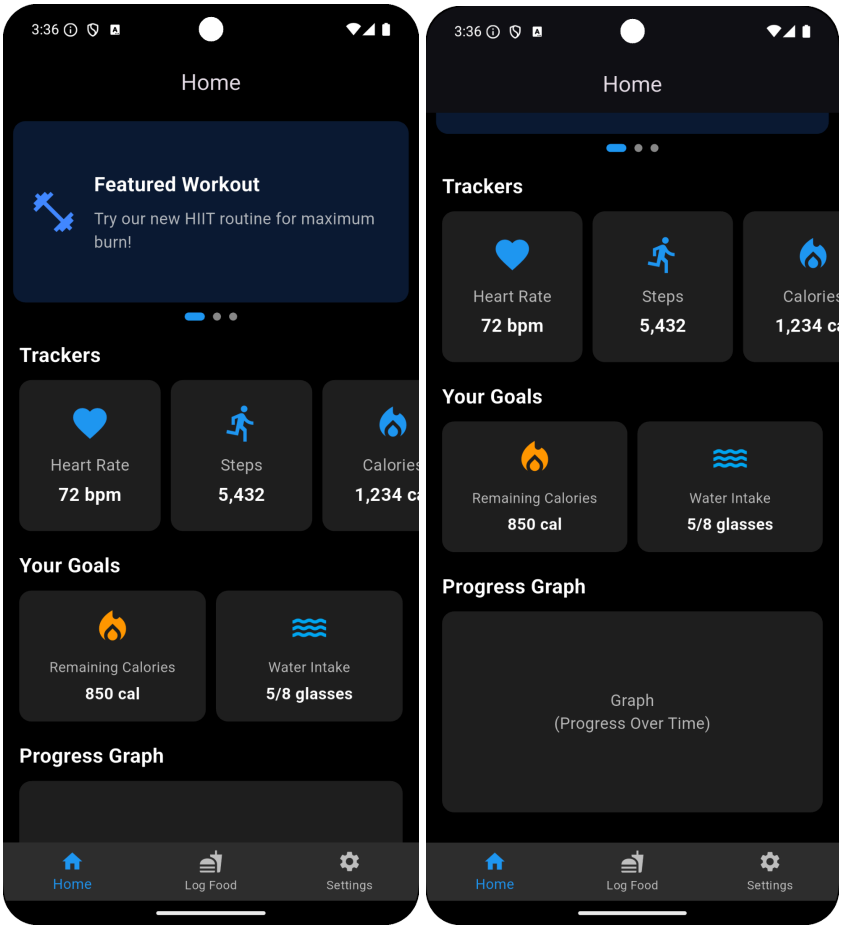
main_screen.dart

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

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

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

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

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

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

```
    icon: Icon(Icons.fastfood),
    label: "Log Food",
   ),
   BottomNavigationBarItem(
    icon: Icon(Icons.settings),
    label: "Settings",
   ),
  ],
  onTap: (index) {
           setState(() {
            _currentIndex = index; // Update the
 current selected index
           });
         },
       ),
     );
   }
  }
```

Screenshots:





MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	16
Name	Shreyash Ganesh Dhekane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO :- 04

AIM: - To create an interactive Form using a form widget.

Theory: -

A form in Flutter is a structured container that collects user input through various fields like text fields, dropdowns, checkboxes, and buttons. It plays a crucial role in applications that require user data entry, such as login pages, registration forms, and feedback submissions. Flutter provides the Form widget, which works alongside TextFormField and other input elements to manage validation, state handling, and error messages efficiently. By using form validation techniques, developers can ensure data accuracy and enhance user experience.
When you create a form, it is necessary to provide the GlobalKey. This key uniquely identifies the form and allows you to do any validation in the form fields. The form widget uses child widget TextFormField to provide the users to enter the text field. This widget renders a material design text field and also allows us to display validation errors when they occur.

Creation of a Form

1. While creating a form in Flutter, the Form widget is essential as it acts as a container for grouping multiple form fields and managing validation.
2. A GlobalKey<FormState> is required to uniquely identify the form and enable validation or data retrieval from the form fields.
3. The TextFormField widget is used to provide input fields where users can enter data such as names, phone numbers, or email addresses.
4. To enhance the appearance and usability of input fields, InputDecoration is used, allowing customization of labels, icons, borders, and hint text.
5. Validation plays a crucial role in forms, and the validator property within TextFormField ensures user input meets specific criteria before submission.
6. Different types of input require appropriate keyboard types, such as TextInputType.number for numeric fields or TextInputType.emailAddress for email fields.
7. Proper state management is needed to store and retrieve user input, ensuring the form data is processed correctly.
8. A submit button is necessary to trigger form validation and submit the collected data for further processing.

Some Properties of Form Widget

- key: A GlobalKey that uniquely identifies the Form. You can use this key to interact with the form, such as validating, resetting, or saving its state.
- child: The child widget that contains the form fields. Typically, this is a Column, ListView, or another widget that allows you to arrange the form fields vertically.
- autovalidateMode: An enum that specifies when the form should automatically validate its fields.
- Some Methods of Form Widget
- validate(): This method is used to trigger the validation of all the form fields within the Form. It returns true if all fields are valid, otherwise false. You can use it to check the overall validity of the form before submitting it.
- save(): This method is used to save the current values of all form fields. It invokes the onSaved callback for each field. Typically, this method is called after validation succeeds.
- reset(): Resets the form to its initial state, clearing any user-entered data.
- currentState: A getter that returns the current FormState associated with the Form.

Code:

login_screen.dart

```
import 'package:flutter/material.dart';
import
'package:firebase_auth/firebase_auth.dart';
import 'main_screen.dart'; // Import the
MainScreen (not HomeScreen)

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

  @override
  State<LoginScreen> createState() =>
_LoginScreenState();
}

class _LoginScreenState extends
State<LoginScreen> {
  // Controllers for input fields
  final TextEditingController emailController =
TextEditingController();
  final TextEditingController
passwordController = TextEditingController();

  // Firebase Auth instance
  final FirebaseAuth _auth =
FirebaseAuth.instance;

  bool _isLoading = false;

  @override
  void dispose() {
    emailController.dispose();
    passwordController.dispose();
    super.dispose();
  }

  /// Sign in using Firebase Authentication
  Future<void> _login() async {
    setState(() {
      _isLoading = true;
    });

    try {
      // Attempt sign in with email and password
      await
_auth.signInWithEmailAndPassword(
        email: emailController.text.trim(),
        password: passwordController.text,
      );

      // On successful login, navigate to the
MainScreen with BottomNavigationBar.
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) =>
const MainScreen()), // Navigate to
MainScreen
      );
    } on FirebaseAuthException catch (e) {
      String message = 'Login failed';
      if (e.code == 'user-not-found') {
        message = 'No user found for that
email.';
      } else if (e.code == 'wrong-password') {
        message = 'Incorrect password
provided.';
      }

ScaffoldMessenger.of(context).showSnackBar
(
        SnackBar(content: Text(message)),
      );
    } catch (e) {

ScaffoldMessenger.of(context).showSnackBar
(
        const SnackBar(content: Text('An error
occurred.')),
      );
    }

    setState(() {
      _isLoading = false;
    });
  }

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

```
      title: const Text("Log In", style:
TextStyle(color: Colors.white)),
      centerTitle: true,
      actions: [
       // Skip button for direct access (login
later)
        TextButton(
         onPressed: () {
          Navigator.pushReplacement(
           context,
           MaterialPageRoute(builder:
(context) => const MainScreen()), // Navigate
to MainScreen
          );
         },
         child: const Text("Skip", style:
TextStyle(color: Colors.white, fontSize: 16)),
        ),
      ],
      leading: IconButton(
       icon: const Icon(Icons.arrow_back,
color: Colors.white),
       onPressed: () {
        Navigator.pop(context);
       },
      ),
     ),
     body: Padding(
      padding: const
EdgeInsets.symmetric(horizontal: 20),
      child: Column(
       children: [
        const SizedBox(height: 30),
        // Email TextField
        TextField(
         controller: emailController,
         style: const TextStyle(color:
Colors.white),
         decoration: InputDecoration(
          labelText: "Email Address",
          labelStyle: const TextStyle(color:
Colors.white70),
          enabledBorder: OutlineInputBorder(
           borderSide: const
BorderSide(color: Colors.white38),
           borderRadius:
BorderRadius.circular(8),
          ),
          focusedBorder: OutlineInputBorder(
           borderSide: const
BorderSide(color: Colors.blue),
           borderRadius:
BorderRadius.circular(8),
          ),
         ),
        ),
        const SizedBox(height: 20),
        // Password TextField
        TextField(
         controller: passwordController,
         obscureText: true,
         style: const TextStyle(color:
Colors.white),
         decoration: InputDecoration(
          labelText: "Password",
          labelStyle: const TextStyle(color:
Colors.white70),
          enabledBorder: OutlineInputBorder(
           borderSide: const
BorderSide(color: Colors.white38),
           borderRadius:
BorderRadius.circular(8),
          ),
          focusedBorder: OutlineInputBorder(
           borderSide: const
BorderSide(color: Colors.blue),
           borderRadius:
BorderRadius.circular(8),
          ),
         ),
        ),
        const SizedBox(height: 20),
        // Login button or a loading indicator
        _isLoading
          ? const CircularProgressIndicator()
          : ElevatedButton(
         style: ElevatedButton.styleFrom(
          backgroundColor: Colors.blue,
          minimumSize: const
Size(double.infinity, 50),
         ),
         onPressed: _login,
         child: const Text("Log In", style:
TextStyle(color: Colors.white)),
        ),
        const SizedBox(height: 10),
        TextButton(
         onPressed: () {
```

```
            // Implement forgot password logic if
desired.
          },
          child: const Text("Forgot password?",
style: TextStyle(color: Colors.blue)),
        ),
        const SizedBox(height: 20),
        const Row(
          children: [
            Expanded(child: Divider(color:
Colors.white)),
            Padding(
              padding:
EdgeInsets.symmetric(horizontal: 10),
              child: Text("OR", style:
TextStyle(color: Colors.white)),
            ),
            Expanded(child: Divider(color:
Colors.white)),
          ],
        ),
        // Additional social login buttons can
be added here
        const Spacer(),
        // Navigation to registration screen
        TextButton(
          onPressed: () {
            Navigator.pushNamed(context,
'/register');
          },
          child: const Text("Don't have an
account? Register", style: TextStyle(color:
Colors.blue, fontSize: 16)),
        ),
        const SizedBox(height: 20),
      ],
    ),
  ),
 );
 }
}
```

registration_screen.dart

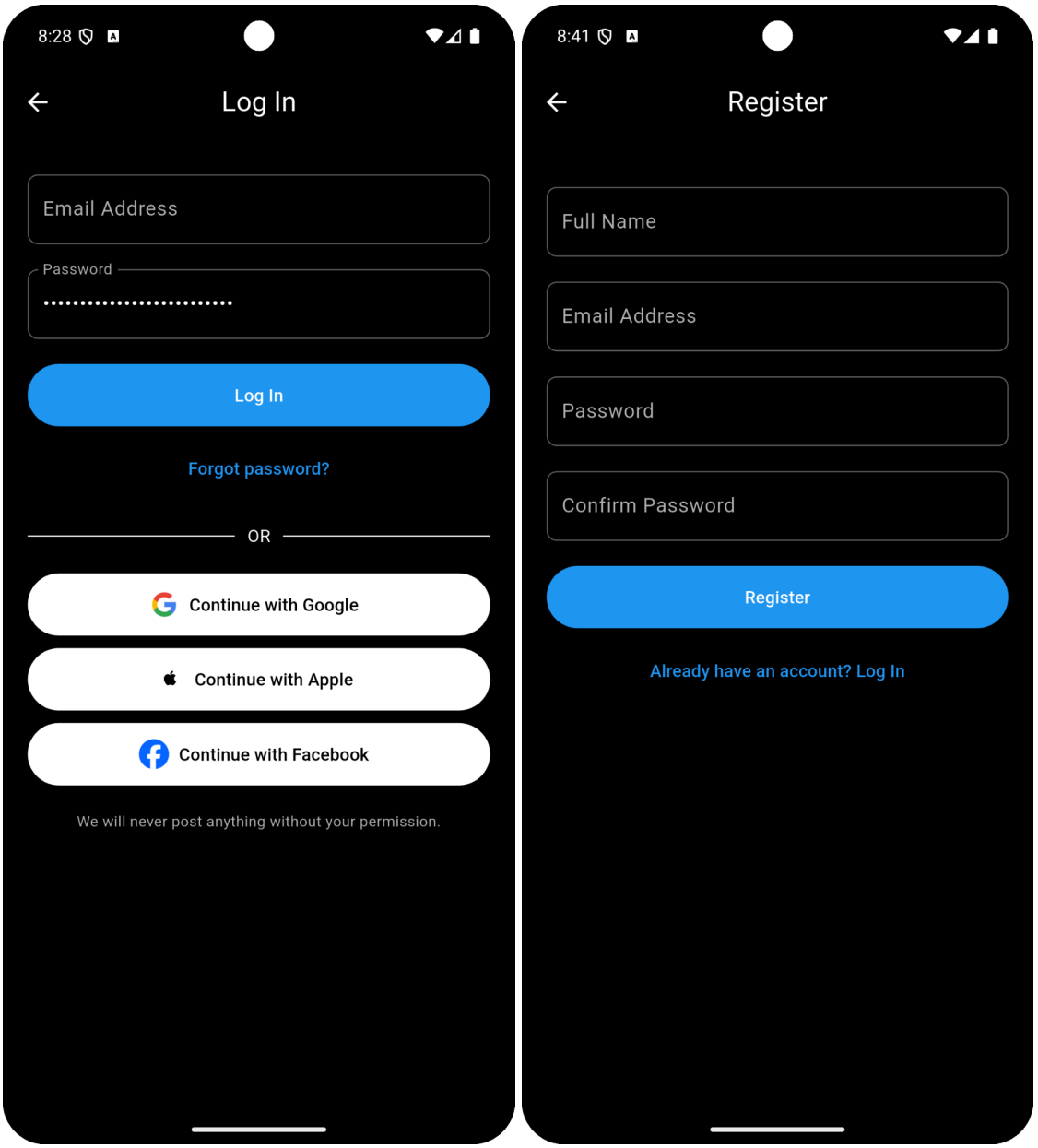
```
import 'package:flutter/material.dart';
import
'package:firebase_auth/firebase_auth.dart';

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

 @override
 State<RegistrationScreen> createState() =>
_RegistrationScreenState();
}

class _RegistrationScreenState extends
State<RegistrationScreen> {
 // Controllers for input fields
 final TextEditingController nameController =
TextEditingController();
 final TextEditingController emailController =
TextEditingController();
 final TextEditingController
passwordController = TextEditingController();
 final TextEditingController
confirmPasswordController =
TextEditingController();

 // Firebase Auth instance
 final FirebaseAuth _auth =
FirebaseAuth.instance;

 bool _isLoading = false;

 @override
 void dispose() {
  nameController.dispose();
  emailController.dispose();
  passwordController.dispose();
  confirmPasswordController.dispose();
  super.dispose();
 }

 /// Register a new user with Firebase
Authentication
 Future<void> _register() async {
  final email = emailController.text.trim();
  final password = passwordController.text;
```

```
    final confirmPassword =
confirmPasswordController.text;

    if (password != confirmPassword) {

ScaffoldMessenger.of(context).showSnackBar
(
        const SnackBar(content:
Text("Passwords do not match")),
      );
      return;
    }

    setState(() {
      _isLoading = true;
    });

    try {
      // Create the user
      UserCredential userCredential = await
_auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );

      // Optionally update the display name
      if (userCredential.user != null &&
nameController.text.trim().isNotEmpty) {
        await
userCredential.user!.updateDisplayName(na
meController.text.trim());
      }

      // On successful registration, navigate to
main screen.

Navigator.pushReplacementNamed(context,
'/main');
    } on FirebaseAuthException catch (e) {
      String message = 'Registration failed';
      if (e.code == 'weak-password') {
        message = 'The password provided is
too weak.';
      } else if (e.code == 'email-already-in-use')
{
        message = 'An account already exists for
that email.';
      }

ScaffoldMessenger.of(context).showSnackBar
(
        SnackBar(content: Text(message)),
      );
    } catch (e) {

ScaffoldMessenger.of(context).showSnackBar
(
        const SnackBar(content: Text("An error
occurred")),
      );
    }

    setState(() {
      _isLoading = false;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      appBar: AppBar(
        backgroundColor: Colors.black,
        title: const Text("Register", style:
TextStyle(color: Colors.white)),
        centerTitle: true,
        leading: IconButton(
          icon: const Icon(Icons.arrow_back,
color: Colors.white),
          onPressed: () {
            Navigator.pop(context);
          },
        ),
      ),
      body: SingleChildScrollView(
        padding: const
EdgeInsets.symmetric(horizontal: 20, vertical:
20),
        child: Column(
          children: [
            const SizedBox(height: 20),
            // Full Name field (optional)
            TextField(
              controller: nameController,
              style: const TextStyle(color:
Colors.white),
              decoration: InputDecoration(
                labelText: "Full Name",
```

```
            labelStyle: const TextStyle(color:
Colors.white70),
            enabledBorder: OutlineInputBorder(
              borderSide: const
BorderSide(color: Colors.white38),
              borderRadius:
BorderRadius.circular(8),
            ),
            focusedBorder: OutlineInputBorder(
              borderSide: const
BorderSide(color: Colors.blue),
              borderRadius:
BorderRadius.circular(8),
            ),
          ),
        ),
        const SizedBox(height: 20),
        // Email Address field
        TextField(
          controller: emailController,
          style: const TextStyle(color:
Colors.white),
          decoration: InputDecoration(
            labelText: "Email Address",
            labelStyle: const TextStyle(color:
Colors.white70),
            enabledBorder: OutlineInputBorder(
              borderSide: const
BorderSide(color: Colors.white38),
              borderRadius:
BorderRadius.circular(8),
            ),
            focusedBorder: OutlineInputBorder(
              borderSide: const
BorderSide(color: Colors.blue),
              borderRadius:
BorderRadius.circular(8),
            ),
          ),
        ),
        const SizedBox(height: 20),
        // Password field
        TextField(
          controller: passwordController,
          obscureText: true,
          style: const TextStyle(color:
Colors.white),
          decoration: InputDecoration(
            labelText: "Password",
            labelStyle: const TextStyle(color:
Colors.white70),
            enabledBorder: OutlineInputBorder(
              borderSide: const
BorderSide(color: Colors.white38),
              borderRadius:
BorderRadius.circular(8),
            ),
            focusedBorder: OutlineInputBorder(
              borderSide: const
BorderSide(color: Colors.blue),
              borderRadius:
BorderRadius.circular(8),
            ),
          ),
        ),
        const SizedBox(height: 20),
        // Confirm Password field
        TextField(
          controller:
confirmPasswordController,
          obscureText: true,
          style: const TextStyle(color:
Colors.white),
          decoration: InputDecoration(
            labelText: "Confirm Password",
            labelStyle: const TextStyle(color:
Colors.white70),
            enabledBorder: OutlineInputBorder(
              borderSide: const
BorderSide(color: Colors.white38),
              borderRadius:
BorderRadius.circular(8),
            ),
            focusedBorder: OutlineInputBorder(
              borderSide: const
BorderSide(color: Colors.blue),
              borderRadius:
BorderRadius.circular(8),
            ),
          ),
        ),
        const SizedBox(height: 20),
        _isLoading
            ? const CircularProgressIndicator()
            : ElevatedButton(
          style: ElevatedButton.styleFrom(
            backgroundColor: Colors.blue,
            minimumSize: const
Size(double.infinity, 50),
```

```
            ),
            onPressed: _register,
            child: const Text("Register", style:
TextStyle(color: Colors.white)),
          ),
          const SizedBox(height: 10),
                // Optionally add a link to the login
page if already have an account.
              ],
            ),
          ),
        );
      }
    }
```

Screenshots:



MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	16
Name	Shreyash Ganesh Dhekane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO :- 05

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

Theory: -

In Flutter, the screens and pages are known as routes, and these routes are just a widget. In Android, a route is similar to an Activity.
In any mobile app, navigating to different pages defines the workflow of the application, and the way to handle the navigation is known as routing. Flutter provides a basic routing class MaterialPageRoute and two methods Navigator.push() and Navigator.pop() that shows how to navigate between two routes. The following steps are required to start navigation in your application. Gestures enable the app to respond to user interactions, making the application more dynamic and responsive.

Navigation and Routing in Flutter

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

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

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

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

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

```
MaterialApp(
initialRoute: '/',
routes: {
'/': (context) => HomeScreen(),
'/second': (context) => SecondScreen(),
},
);
Navigate to the route using Navigator.pushNamed()
Navigator.pushNamed(context, '/second');
```

Handling Gestures in Flutter

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

Tap Gestures

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

Long Press Gesture

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

Swipe and Drag Gestures

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

Code:

main.dart

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

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

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

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

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

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

log_food_screen.dart

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

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

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

class _LogFoodScreenState extends
State<LogFoodScreen> {
  // Dummy list to store custom recipes added
via the custom recipe screen.
  List<Map<String, dynamic>> customRecipes
= [];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Log Food"),
        backgroundColor: Colors.black,
        centerTitle: true,
      ),
      backgroundColor: Colors.black,
      body: SingleChildScrollView(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment:
CrossAxisAlignment.start,
          children: [
            // ---------------------------
            // Existing Log Food UI Content
            // ---------------------------
            const Text(
              "Enter the details of your meal:",
              style: TextStyle(color: Colors.white,
fontSize: 18),
            ),
            const SizedBox(height: 20),
            TextField(
              style: const TextStyle(color:
Colors.white),
              decoration: InputDecoration(
                labelText: "Food Name",
                labelStyle: const TextStyle(color:
Colors.white70),
                enabledBorder: OutlineInputBorder(
                  borderSide: const
BorderSide(color: Colors.white38),
                  borderRadius:
BorderRadius.circular(8),
                ),
                focusedBorder: OutlineInputBorder(
                  borderSide: const
BorderSide(color: Colors.blue),
                  borderRadius:
BorderRadius.circular(8),
                ),
              ),
            ),
            const SizedBox(height: 20),
            TextField(
              style: const TextStyle(color:
Colors.white),
              decoration: InputDecoration(
                labelText: "Calories",
                labelStyle: const TextStyle(color:
Colors.white70),
                enabledBorder: OutlineInputBorder(
                  borderSide: const
BorderSide(color: Colors.white38),
                  borderRadius:
BorderRadius.circular(8),
                ),
                focusedBorder: OutlineInputBorder(
                  borderSide: const
BorderSide(color: Colors.blue),
                  borderRadius:
BorderRadius.circular(8),
                ),
              ),
            ),
            const SizedBox(height: 20),
            ElevatedButton(
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.blue,
                minimumSize: const
Size(double.infinity, 50),
              ),
              onPressed: () {
```

```
            // Add logic to log the food entry.
          },
          child: const Text("Log Food", style:
TextStyle(color: Colors.white)),
        ),
        const SizedBox(height: 30),

        // --------------------------
        // Recipe Catalog Section
        // --------------------------
        const Text(
          "Recipe Catalog",
          style: TextStyle(
            color: Colors.white,
            fontSize: 20,
            fontWeight: FontWeight.bold,
          ),
        ),
        const SizedBox(height: 10),
        customRecipes.isEmpty
            ? const Text(
          "No custom recipes added yet.",
          style: TextStyle(color:
Colors.white70),
        )
            : ListView.builder(
          shrinkWrap: true,
          physics: const
NeverScrollableScrollPhysics(),
          itemCount: customRecipes.length,
          itemBuilder: (context, index) {
            final recipe = customRecipes[index];
            return Card(
              color: Colors.grey[900],
              child: ListTile(
                title: Text(recipe['name'],
                    style: const TextStyle(color:
Colors.white)),
                subtitle: Text(
                  "${recipe['ingredients'].length}
ingredients",
                  style: const TextStyle(color:
Colors.white70),
                ),
              ),
            );
          },
        ),
        const SizedBox(height: 80), // Extra
space for the FAB
      ],
    ),
  ),
  floatingActionButton:
FloatingActionButton(
    backgroundColor: Colors.blue,
    onPressed: () async {
      // Navigate to the
AddCustomRecipeScreen and await the
result.
      final result = await Navigator.push(
        context,
        MaterialPageRoute(builder: (context)
=> const AddCustomRecipeScreen()),
      );
      if (result != null && result is Map<String,
dynamic>) {
        setState(() {
          customRecipes.add(result);
        });
      }
    },
    child: const Icon(Icons.add),
  ),
 );
 }
}
```

add_custom_recipe_screen.dart

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

/// A helper class to hold
TextEditingControllers for each ingredient.
class IngredientField {
 final TextEditingController nameController;
 final TextEditingController caloriesController;
 final TextEditingController proteinController;
 final TextEditingController fibreController;
 final TextEditingController fatController;

 IngredientField({
```

```
    String? name,
    String? calories,
    String? protein,
    String? fibre,
    String? fat,
  })  : nameController =
TextEditingController(text: name),
        caloriesController =
TextEditingController(text: calories),
        proteinController =
TextEditingController(text: protein),
        fibreController =
TextEditingController(text: fibre),
        fatController = TextEditingController(text:
fat);

  void dispose() {
    nameController.dispose();
    caloriesController.dispose();
    proteinController.dispose();
    fibreController.dispose();
    fatController.dispose();
  }
}

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

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

class _AddCustomRecipeScreenState
extends State<AddCustomRecipeScreen> {
  final TextEditingController
recipeNameController =
TextEditingController();
  List<IngredientField> ingredients = [];

  @override
  void initState() {
    super.initState();
    // Start with one ingredient input row.
    ingredients.add(IngredientField());
  }

  @override
  void dispose() {
    recipeNameController.dispose();
    for (var ingredient in ingredients) {
      ingredient.dispose();
    }
    super.dispose();
  }

  void addIngredient() {
    setState(() {
      ingredients.add(IngredientField());
    });
  }

  void removeIngredient(int index) {
    setState(() {
      ingredients[index].dispose();
      ingredients.removeAt(index);
    });
  }

  void submitRecipe() {
    final recipeName =
recipeNameController.text.trim();
    if (recipeName.isEmpty) {

ScaffoldMessenger.of(context).showSnackBar
(
        const SnackBar(content: Text("Please
enter a recipe name")),
      );
      return;
    }
    List<Map<String, String>> ingredientList =
[];
    for (var ingredient in ingredients) {
      final name =
ingredient.nameController.text.trim();
      final calories =
ingredient.caloriesController.text.trim();
      final protein =
ingredient.proteinController.text.trim();
      final fibre =
ingredient.fibreController.text.trim();
      final fat =
ingredient.fatController.text.trim();
      if (name.isEmpty ||
          calories.isEmpty ||
          protein.isEmpty ||
```

```
        fibre.isEmpty ||
        fat.isEmpty) {

ScaffoldMessenger.of(context).showSnackBar
(
        const SnackBar(content: Text("Please
fill out all fields for each ingredient")),
      );
      return;
    }
    ingredientList.add({
      'name': name,
      'calories': calories,
      'protein': protein,
      'fibre': fibre,
      'fat': fat,
    });
  }

  // For UI purposes, simulate submission.
  // In a real app, this data would be sent to a
database for admin verification.
  final recipeData = {
    'name': recipeName,
    'ingredients': ingredientList,
  };

  // Show a success message and pop with
the recipe data.

ScaffoldMessenger.of(context).showSnackBar
(
    const SnackBar(content: Text("Recipe
submitted for verification")),
  );
  Navigator.pop(context, recipeData);
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text("Add Custom Recipe"),
      backgroundColor: Colors.black,
      centerTitle: true,
    ),
    backgroundColor: Colors.black,
    body: SingleChildScrollView(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment:
CrossAxisAlignment.start,
        children: [
          // Recipe Name Field
          TextField(
            controller: recipeNameController,
            style: const TextStyle(color:
Colors.white),
            decoration: InputDecoration(
              labelText: "Recipe Name",
              labelStyle: const TextStyle(color:
Colors.white70),
              enabledBorder: OutlineInputBorder(
                borderSide: const
BorderSide(color: Colors.white38),
                borderRadius:
BorderRadius.circular(8),
              ),
              focusedBorder: OutlineInputBorder(
                borderSide: const
BorderSide(color: Colors.blue),
                borderRadius:
BorderRadius.circular(8),
              ),
            ),
          ),
          const SizedBox(height: 20),
          // Dynamic List of Ingredient Fields
          const Text(
            "Ingredients",
            style: TextStyle(color: Colors.white,
fontSize: 18, fontWeight: FontWeight.bold),
          ),
          const SizedBox(height: 10),
          ListView.builder(
            shrinkWrap: true,
            physics: const
NeverScrollableScrollPhysics(),
            itemCount: ingredients.length,
            itemBuilder: (context, index) {
              return IngredientInputCard(
                ingredientField: ingredients[index],
                onRemove: ingredients.length > 1
? () => removeIngredient(index) : null,
              );
            },
          ),
          const SizedBox(height: 10),
          TextButton.icon(
            onPressed: addIngredient,
```

```
            icon: const Icon(Icons.add, color:
Colors.blue),
            label: const Text("Add Ingredient",
style: TextStyle(color: Colors.blue)),
          ),
          const SizedBox(height: 20),
          ElevatedButton(
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.blue,
              minimumSize: const
Size(double.infinity, 50),
            ),
            onPressed: submitRecipe,
            child: const Text("Submit Recipe",
style: TextStyle(color: Colors.white)),
          ),
        ],
      ),
    ),
  );
 }
}

/// A widget that displays input fields for one
ingredient.
class IngredientInputCard extends
StatelessWidget {
 final IngredientField ingredientField;
 final VoidCallback? onRemove;

 const IngredientInputCard({
  Key? key,
  required this.ingredientField,
  this.onRemove,
 }) : super(key: key);

 @override
 Widget build(BuildContext context) {
  return Card(
   color: Colors.grey[900],
   margin: const
EdgeInsets.symmetric(vertical: 8),
   child: Padding(
    padding: const EdgeInsets.all(8.0),
    child: Column(
     children: [
      // Row for Ingredient Name and
remove button
      Row(
       children: [
        Expanded(
         child: TextField(
          controller:
ingredientField.nameController,
          style: const TextStyle(color:
Colors.white),
          decoration: InputDecoration(
           labelText: "Ingredient Name",
           labelStyle: const TextStyle(color:
Colors.white70),
           enabledBorder:
OutlineInputBorder(
            borderSide: const
BorderSide(color: Colors.white38),
            borderRadius:
BorderRadius.circular(8),
           ),
           focusedBorder:
OutlineInputBorder(
            borderSide: const
BorderSide(color: Colors.blue),
            borderRadius:
BorderRadius.circular(8),
           ),
          ),
         ),
        ),
        if (onRemove != null)
         IconButton(
          icon: const
Icon(Icons.remove_circle, color: Colors.red),
          onPressed: onRemove,
         ),
       ],
      ),
      const SizedBox(height: 10),
      // Row for Calories and Protein
      Row(
       children: [
        Expanded(
         child: TextField(
          controller:
ingredientField.caloriesController,
          style: const TextStyle(color:
Colors.white),
          decoration: InputDecoration(
           labelText: "Calories",
           labelStyle: const TextStyle(color:
Colors.white70),
```

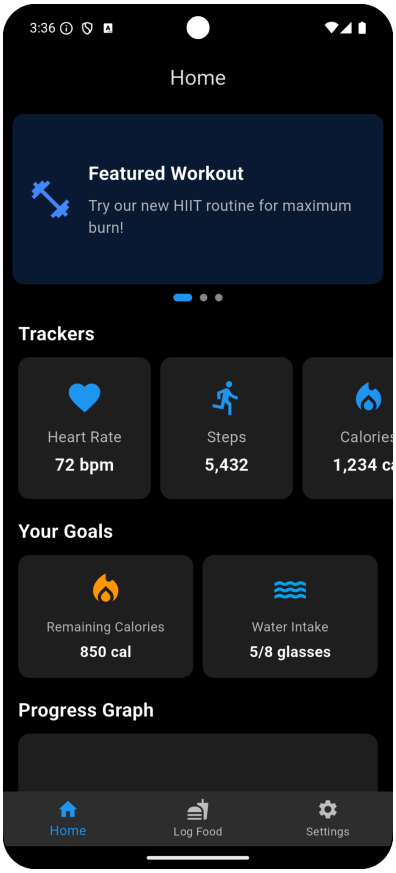
```
                  enabledBorder:
OutlineInputBorder(
                    borderSide: const
BorderSide(color: Colors.white38),
                    borderRadius:
BorderRadius.circular(8),
                  ),
                  focusedBorder:
OutlineInputBorder(
                    borderSide: const
BorderSide(color: Colors.blue),
                    borderRadius:
BorderRadius.circular(8),
                  ),
                ),
                keyboardType:
TextInputType.number,
              ),
            ),
            const SizedBox(width: 8),
            Expanded(
              child: TextField(
                controller:
ingredientField.proteinController,
                style: const TextStyle(color:
Colors.white),
                decoration: InputDecoration(
                  labelText: "Protein",
                  labelStyle: const TextStyle(color:
Colors.white70),
                  enabledBorder:
OutlineInputBorder(
                    borderSide: const
BorderSide(color: Colors.white38),
                    borderRadius:
BorderRadius.circular(8),
                  ),
                  focusedBorder:
OutlineInputBorder(
                    borderSide: const
BorderSide(color: Colors.blue),
                    borderRadius:
BorderRadius.circular(8),
                  ),
                ),
                keyboardType:
TextInputType.number,
              ),
            ),
          ],
        ),
        const SizedBox(height: 10),
        // Row for Fibre and Fat
        Row(
          children: [
            Expanded(
              child: TextField(
                controller:
ingredientField.fibreController,
                style: const TextStyle(color:
Colors.white),
                decoration: InputDecoration(
                  labelText: "Fibre",
                  labelStyle: const TextStyle(color:
Colors.white70),
                  enabledBorder:
OutlineInputBorder(
                    borderSide: const
BorderSide(color: Colors.white38),
                    borderRadius:
BorderRadius.circular(8),
                  ),
                  focusedBorder:
OutlineInputBorder(
                    borderSide: const
BorderSide(color: Colors.blue),
                    borderRadius:
BorderRadius.circular(8),
                  ),
                ),
                keyboardType:
TextInputType.number,
              ),
            ),
            const SizedBox(width: 8),
            Expanded(
              child: TextField(
                controller:
ingredientField.fatController,
                style: const TextStyle(color:
Colors.white),
                decoration: InputDecoration(
                  labelText: "Fat",
                  labelStyle: const TextStyle(color:
Colors.white70),
                  enabledBorder:
OutlineInputBorder(
                    borderSide: const
BorderSide(color: Colors.white38),
```

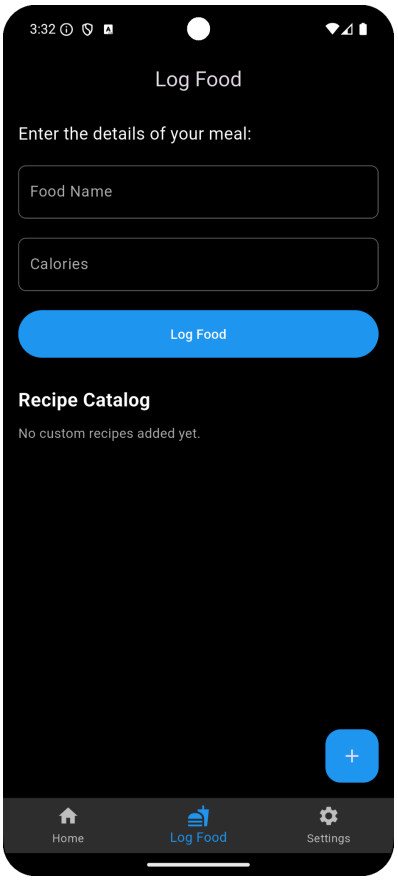
```
                    borderRadius:
BorderRadius.circular(8),
                  ),
                  focusedBorder:
OutlineInputBorder(
                    borderSide: const
BorderSide(color: Colors.blue),
                    borderRadius:
BorderRadius.circular(8),
                  ),
                ),
                    keyboardType:
TextInputType.number,
                  ),
                ),
              ],
            ),
          ],
        ),
      ),
    );
  }
}
```

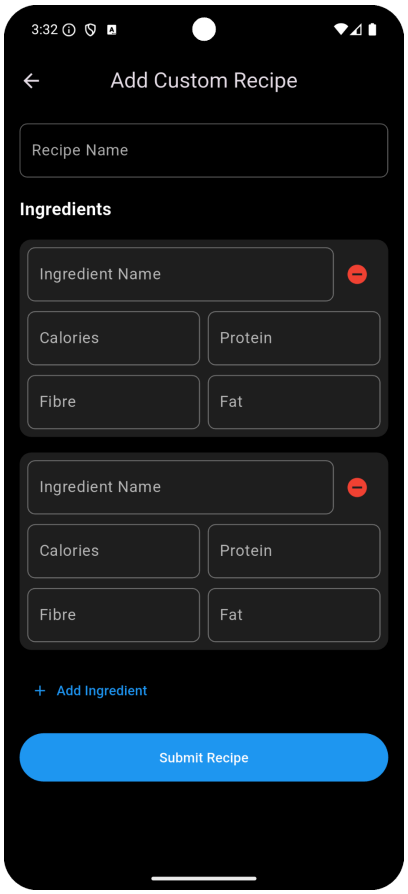
setting_screen.dart

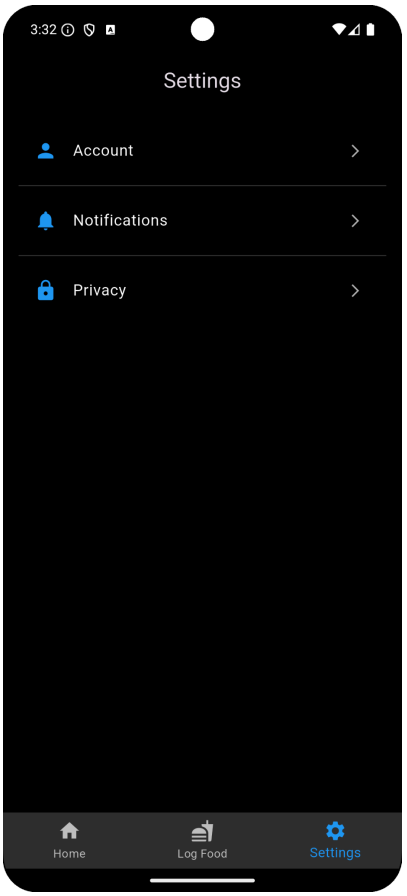
```
import 'package:flutter/material.dart';

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Settings"),
        backgroundColor: Colors.black,
        centerTitle: true,
      ),
      backgroundColor: Colors.black,
      body: ListView(
        padding: const EdgeInsets.all(16.0),
        children: [
          ListTile(
            leading: const Icon(Icons.person,
color: Colors.blue),
            title: const Text("Account", style:
TextStyle(color: Colors.white)),
            trailing: const
Icon(Icons.arrow_forward_ios, color:
Colors.white70, size: 16),
            onTap: () {
              // Navigate to account settings page
            },
          ),
          const Divider(color: Colors.white24),
          ListTile(
            leading: const Icon(Icons.notifications,
color: Colors.blue),
            title: const Text("Notifications", style:
TextStyle(color: Colors.white)),
            trailing: const
Icon(Icons.arrow_forward_ios, color:
Colors.white70, size: 16),
            onTap: () {
              // Navigate to notifications settings
page
            },
          ),
          const Divider(color: Colors.white24),
          ListTile(
            leading: const Icon(Icons.lock, color:
Colors.blue),
            title: const Text("Privacy", style:
TextStyle(color: Colors.white)),
            trailing: const
Icon(Icons.arrow_forward_ios, color:
Colors.white70, size: 16),
            onTap: () {
              // Navigate to privacy settings page
            },
          ),
          // Add additional settings options as
needed...
        ],
      ),
    );
  }
}
```

Screenshots:


3:36
Home
Featured Workout
Try our new HIIT routine for maximum burn!
Trackers
Heart Rate
72 bpm
Steps
5,432
Your Goals
Remaining Calories
850 cal
Water Intake
5/8 glasses
Progress Graph
Home
Log Food
Settings


3:32
Log Food
Enter the details of your meal:
Food Name
Calories
Log Food
Recipe Catalog
No custom recipes added yet.
Home
Log Food
Settings


3:32
Add Custom Recipe
Recipe Name
Ingredients
Ingredient Name
Calories
Protein
Fibre
Fat
Ingredient Name
Calories
Protein
Fibre
Fat
Add Ingredient
Submit Recipe


3:32
Settings
Account
Notifications
Privacy
Home
Log Food
Settings

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	16
Name	Shreyash Ganesh Dhekane
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:	

EXPERIMENT NO :- 06

AIM: - To connect Flutter UI with Firebase Database

Theory: -

Firebase is a cloud-based platform by Google that provides backend services like authentication, real-time databases, and cloud storage. In Flutter, Firebase can be integrated using the Firebase SDK to store and retrieve data dynamically. The Cloud Firestore database enables real-time data synchronization, making it ideal for Flutter applications.

Steps to Connect Flutter UI with Firebase Database

1. Create a Firebase Project

- Go to Firebase Console.
- Click on "Add Project" → Configure settings → Create the project.

2. Add Firebase to Flutter App

- Open your Flutter project.

Run:
flutter pub add firebase_core firebase_firestore

-
- Configure Firebase in `android/app/google-services.json` (for Android) and `ios/Runner/GoogleService-Info.plist` (for iOS).

3. Initialize Firebase in Flutter

Modify `main.dart`:

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

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

4. Connect Firestore Database

- In Firebase Console → Firestore → Create a database.

Create a Firestore instance in Flutter:

```
import 'package:cloud_firestore/cloud_firestore.dart';

FirebaseFirestore firestore = FirebaseFirestore.instance;
```

5. Perform CRUD Operations

Add Data:

```
firestore.collection('users').add({'name': 'John', 'age': 25});
```

Retrieve Data:

```
firestore.collection('users').get().then((snapshot) {

  for (var doc in snapshot.docs) {
    print(doc.data());
  }
});
```

Update Data:

```
firestore.collection('users').doc('docId').update({'age': 26});
```

Delete Data:

```
firestore.collection('users').doc('docId').delete();
```

6. Run the App & Test

Execute `flutter run` and verify Firebase data operations in Firestore.

This setup enables a Flutter UI to interact with Firebase in real-time, ensuring seamless data storage and retrieval.

Code:

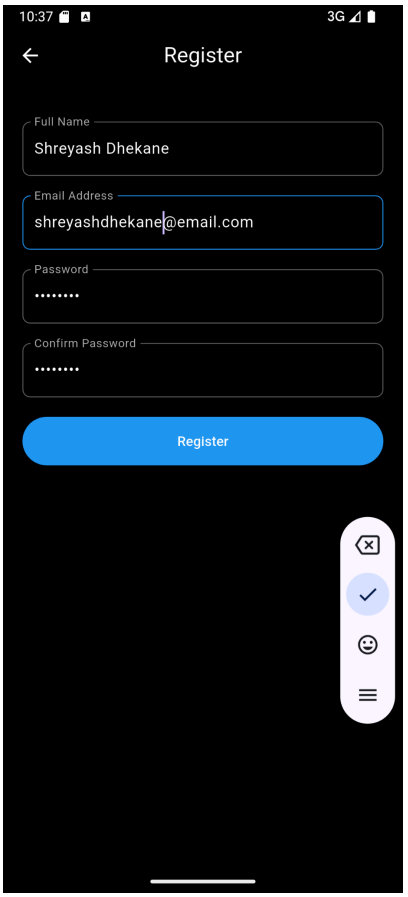
build.gradle

```
buildscript {

    ext.kotlin_version = '2.1.0'  // Update to a newer version

    repositories {

        google()

        mavenCentral()

    }

    dependencies {

        classpath "com.android.tools.build:gradle:8.1.2"

        classpath "com.google.gms:google-services:4.4.2" // ADD THIS LINE

        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"


    }

}


allprojects {

    repositories {

        google()

        mavenCentral()

    }
```

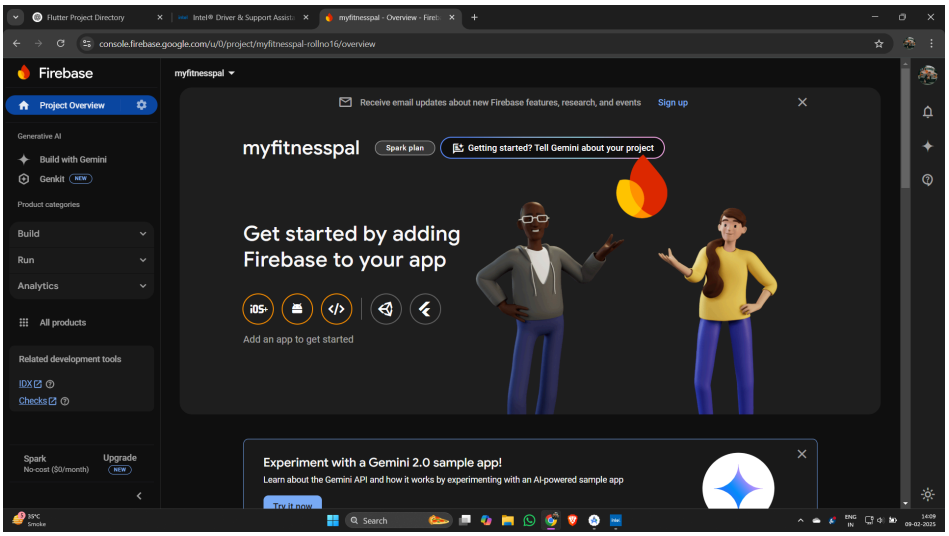
```
}

rootProject.buildDir = "../build"

subprojects {
    project.buildDir =
"${rootProject.buildDir}/${project.name}"
}
```

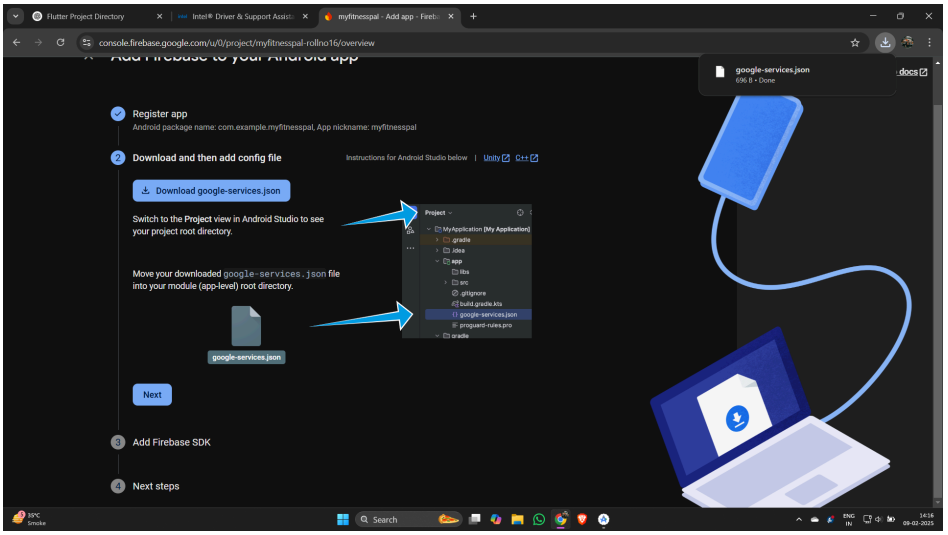
```
subprojects {
    project.evaluationDependsOn(":app")
}

tasks.register("clean", Delete) {
    delete rootProject.buildDir
}
```

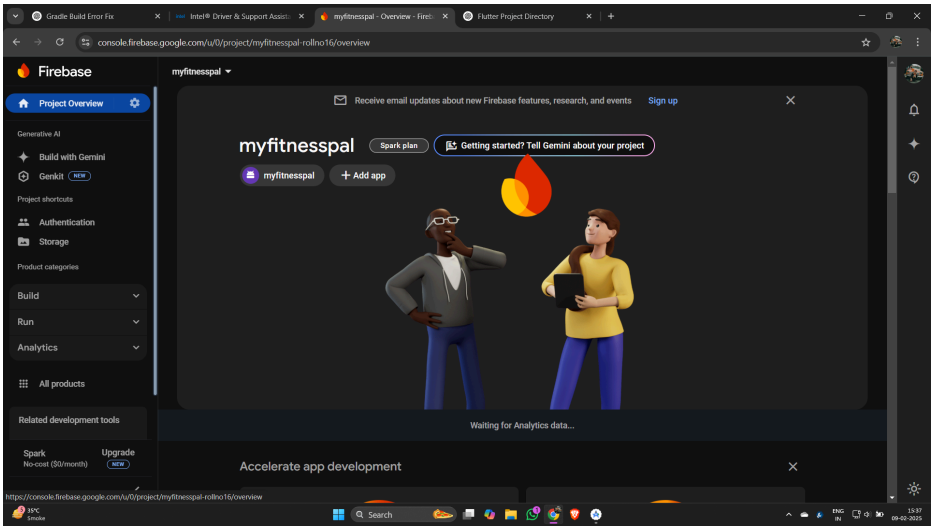
Screenshots:

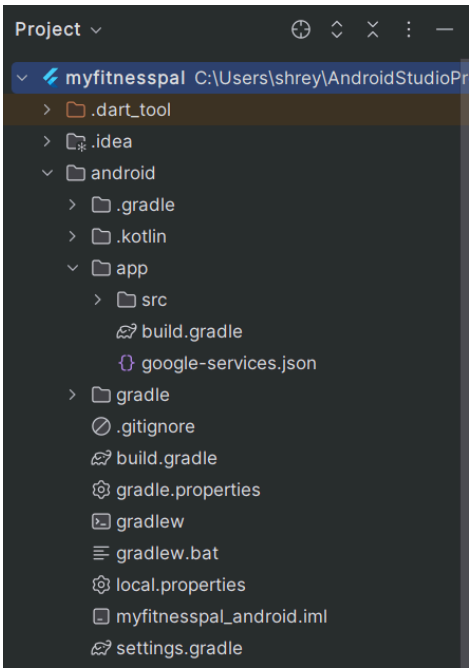


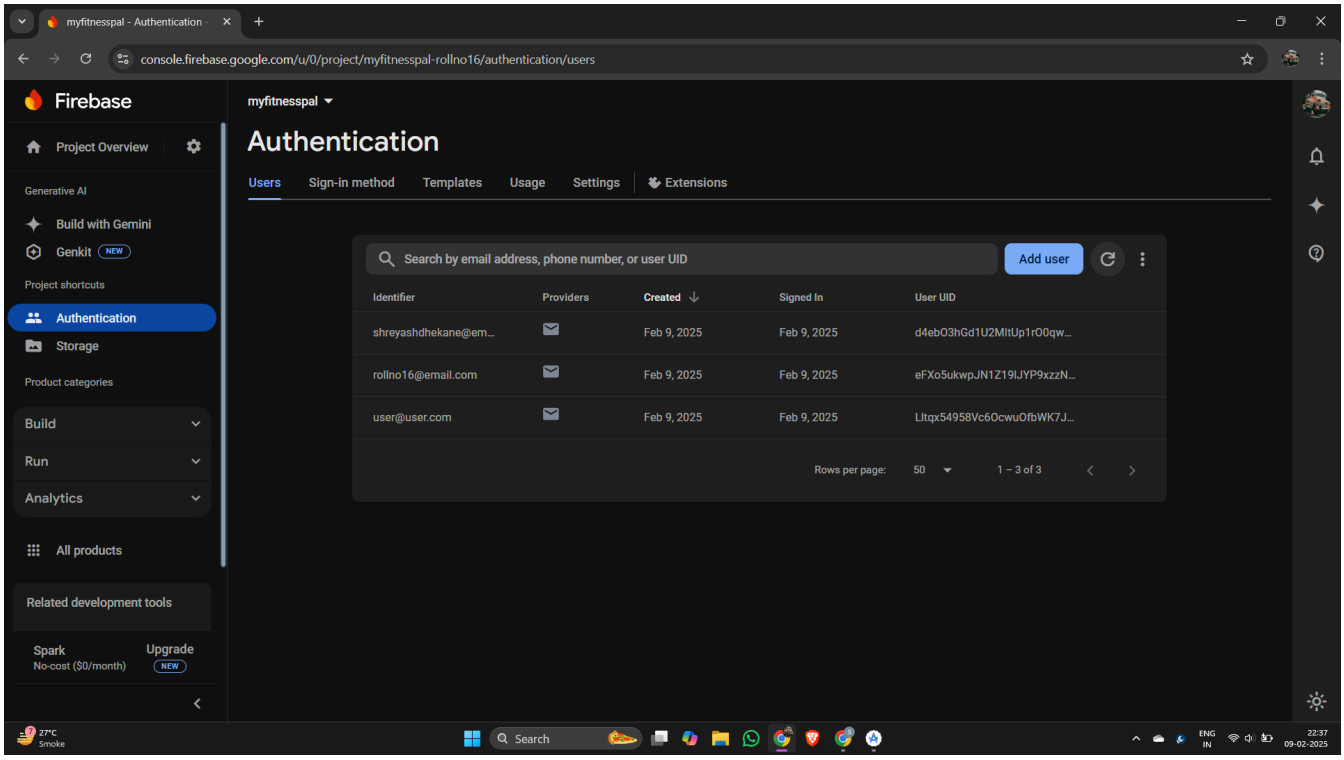
Shreyash G Dhekane D15A Roll No:16








Project
myfitnesspal C:\Users\shrey\AndroidStudioPr
.dart_tool
.idea
android
.gradle
.kotlin
app
src
build.gradle
google-services.json
gradle
.gitignore
build.gradle
gradle.properties
gradlew
gradlew.bat
local.properties
myfitnesspal_android.iml
settings.gradle


myfitnesspal - Authentication
console.firebase.google.com/u/0/project/myfitnesspal-rollno16/authentication/users
Firebase
myfitnesspal
Project Overview
Generative AI
Build with Gemini
Genkit NEW
Project shortcuts
Authentication
Storage
Product categories
Build
Run
Analytics
All products
Related development tools
Spark
No-cost ($0/month)
Upgrade
NEW
Authentication
Users
Sign-in method
Templates
Usage
Settings
Extensions
Search by email address, phone number, or user UID
Add user
Identifier
Providers
Created
Signed In
User UID
shreyashdhekane@em...
Feb 9, 2025
Feb 9, 2025
d4ebO3hGd1U2MltUp1r00qw...
rollno16@email.com
Feb 9, 2025
Feb 9, 2025
eFXo5ukwpJN1Z19IJYP9xzzN...
user@user.com
Feb 9, 2025
Feb 9, 2025
LItqx54958Vc6OcwuOfbWK7J...
Rows per page:
50
1 – 3 of 3
Search
ENG
IN
22:37
09-02-2025

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.	16
Name	Shreyash Ganesh Dhekane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

EXPERIMENT NO. 7

Aim: To write metadata of your E-commerce Progressive Web App (PWA) in a Web App Manifest file to enable the “Add to Home Screen” feature.

Theory :

A **Progressive Web App (PWA)** is a web application that combines the best features of both websites and mobile apps. PWAs are designed to be fast, reliable, and engaging, offering features such as offline access, push notifications, and the ability to be installed on a user's device.

One of the key features of a PWA is the **"Add to Home Screen" (A2HS) functionality**, which allows users to install the web app on their mobile or desktop devices, just like a native app. To enable this feature, a **Web App Manifest** file is required.

PWAs leverage modern web technologies to deliver an app-like experience, eliminating the need for users to download applications from app stores. They provide seamless performance across different devices and platforms, ensuring accessibility and responsiveness. By utilizing service workers for caching, PWAs can function even in low or no-network conditions, making them an excellent choice for enhancing user engagement and retention.

WEB APP MANIFEST

The **Web App Manifest** is a JSON file (manifest.json) that provides important metadata about the PWA, such as its name, icons, colors, and display settings. This file is essential for enabling the **"Add to Home Screen"** feature and ensuring the PWA appears like a native app when installed.

Importance of the Web App Manifest:

1. **Enables App Installation** – Allows users to install the PWA on their home screen or desktop.
2. **Defines App Appearance** – Controls how the app is displayed (standalone, fullscreen, or minimal UI).
3. **Enhances User Experience** – Provides a consistent theme, splash screen, and branding.
4. **Supports Cross-Platform Compatibility** – Works on multiple devices and browsers.

Key Components of a Web App Manifest File (manifest.json)

1. **name and short_name**
   - Defines the full name and a shorter version of the app name used in installation prompts.
2. **start_url**
   - Specifies the URL to open when the app is launched.
3. **display**
   - Defines how the PWA will appear (e.g., standalone, fullscreen, minimal UI, or browser mode).
4. **background_color and theme_color**
   - Controls the splash screen and browser theme when the app is launched.
5. **icons**
   - Provides different icon sizes for various devices.

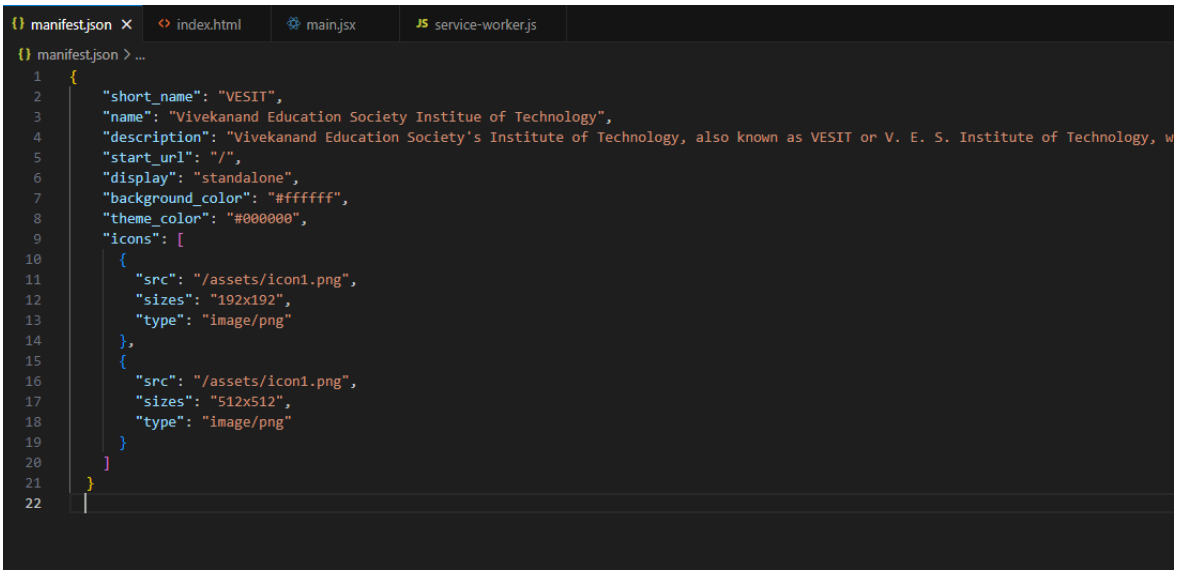
Advantages of Adding a PWA to Home Screen

1. **Fast Access** – Users can launch the app directly from their home screen.
2. **Offline Support** – PWAs can work offline using Service Workers.
3. **Push Notifications** – Engage users with real-time notifications.
4. **Better Performance** – Cached assets make loading faster.
5. **No App Store Required** – Users can install the app without visiting an app store.

Steps to "Add to HomeScreen" feature

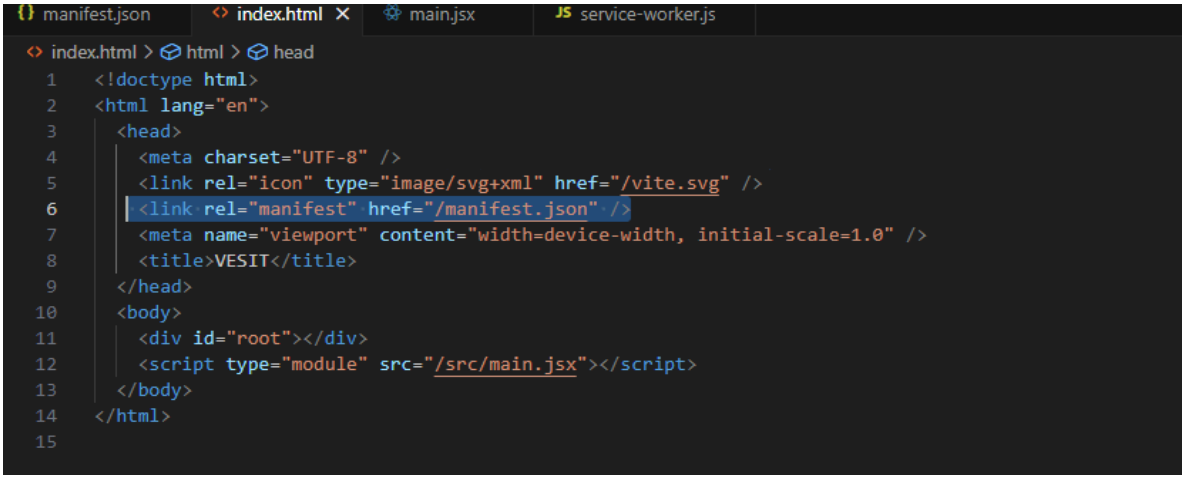
1) Create a Web App Manifest File

Create a manifest.json file in your project's root directory.
Add metadata required for PWA installation



2) Link the Manifest in index.html

Open index.html and add this line inside the <head> tag



3) Register a Service Worker

Create a new file service-worker.js and add the following code to enable caching

var staticCacheName = "pwa-v1"; // Versioned cache name

self.addEventListener("install", function (e) {

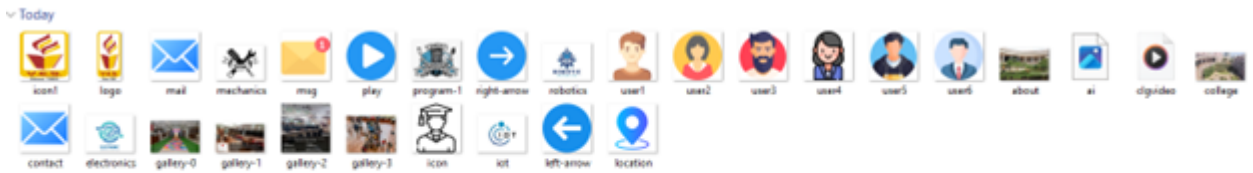
```
  e.waitUntil(
    caches.open(staticCacheName).then(function (cache) {
      return cache.addAll([
        "/",
        "/index.html",
        "/style.css",
        "/script.js",
        "/logo.png",
        "/icon1.png",
        "/offline.html"  // Offline fallback page
      ]);
    })
  );
});

self.addEventListener("activate", function (event) {
    var cacheWhitelist = ["staticCacheName"];

    event.waitUntil(
      caches.keys().then(function (cacheNames) {
        return Promise.all(
          cacheNames.map(function (cacheName) {
            if (cacheWhitelist.indexOf(cacheName) === -1) {
              return caches.delete(cacheName);
            }
          })
        );
      })
    );
  });


self.addEventListener("fetch", function (event) {
  event.respondWith(
    caches.match(event.request).then(function (response) {
      if (response) {
        return response;  // Serve from cache
      }

      return fetch(event.request).catch(function () {
        return caches.match("/offline.html");  // Serve offline fallback
```

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

self.addEventListener('message', (event) => {
   if (event.data.action === 'skipWaiting') {
    self.skipWaiting();
   }
  });
```

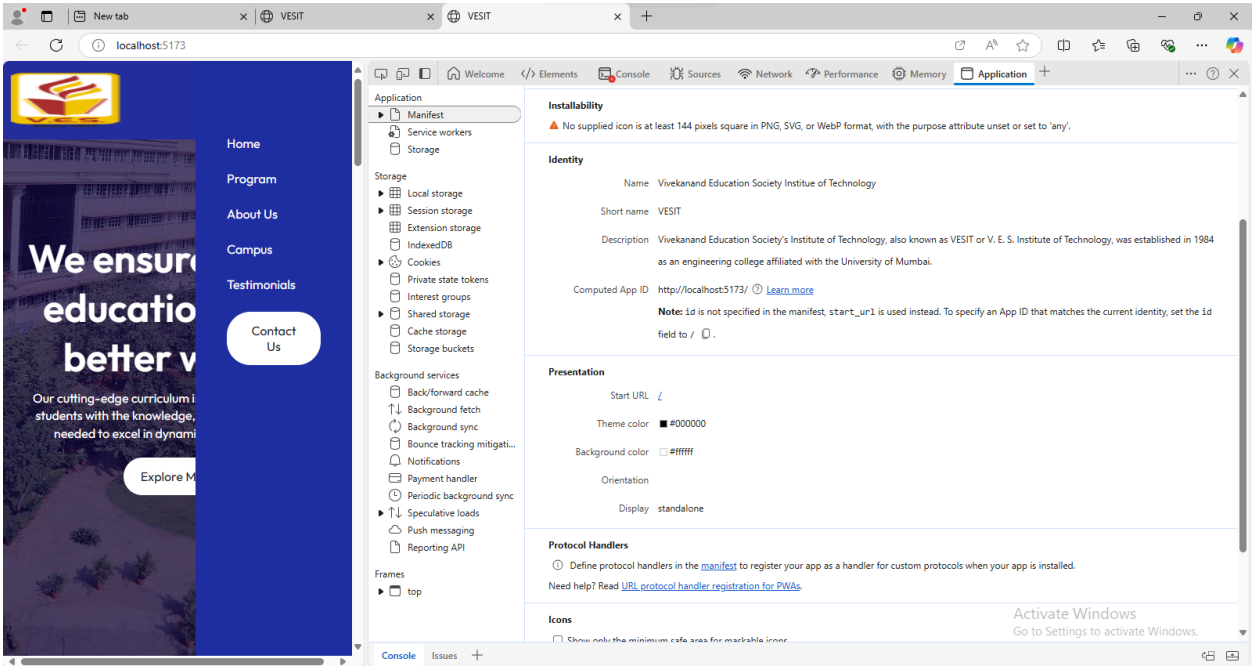
4) <u>Register this service worker in index.html</u>

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

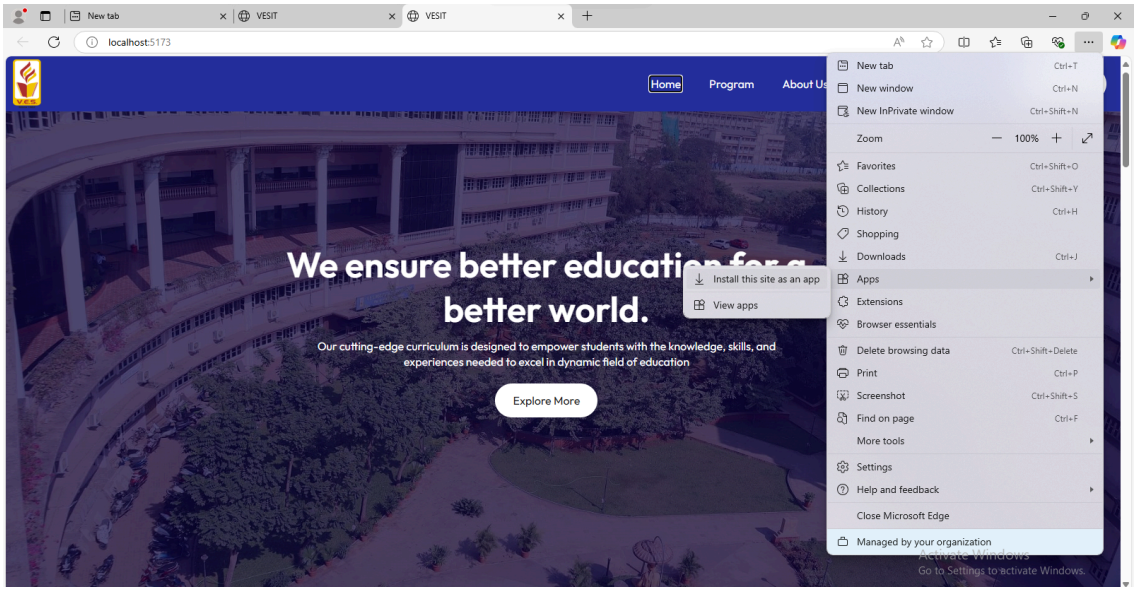
5) <u>Test the PWA Installation</u>



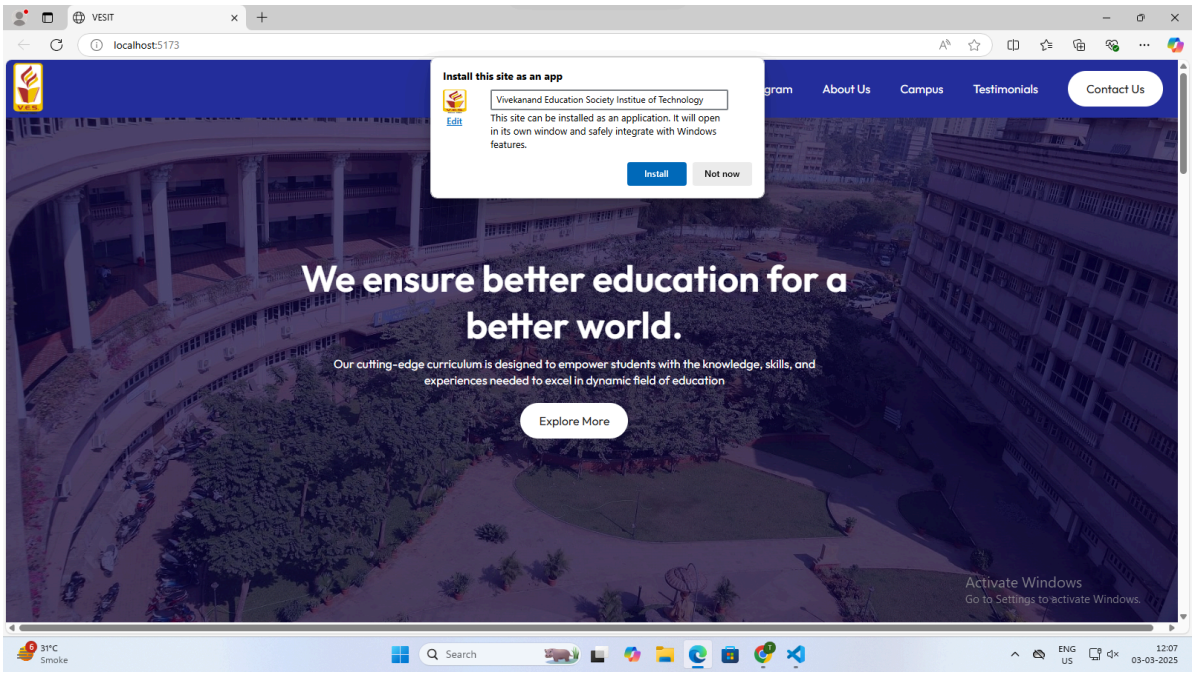
5.1) Open Google Chrome and go to Developer Tools > Application > Manifest to verify your manifest file.


New tab
VESIT
VESIT
localhost:5173
Home
Program
About Us
Campus
Testimonials
Contact Us
We ensur
educatio
better v
Our cutting-edge curriculum i
students with the knowledge,
needed to excel in dynami
Explore M
Welcome
Elements
Console
Sources
Network
Performance
Memory
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigati...
Notifications
Payment handler
Periodic background sync
Speculative loads
Push messaging
Reporting API
Frames
top
Installability
No supplied icon is at least 144 pixels square in PNG, SVG, or WebP format, with the purpose attribute unset or set to 'any'.
Identity
Name Vivekanand Education Society Institue of Technology
Short name VESIT
Description Vivekanand Education Society's Institute of Technology, also known as VESIT or V. E. S. Institute of Technology, was established in 1984 as an engineering college affiliated with the University of Mumbai.
Computed App ID http://localhost:5173/ Learn more
Note: id is not specified in the manifest, start_url is used instead. To specify an App ID that matches the current identity, set the id field to /.
Presentation
Start URL /
Theme color #000000
Background color #ffffff
Orientation
Display standalone
Protocol Handlers
Define protocol handlers in the manifest to register your app as a handler for custom protocols when your app is installed.
Need help? Read URL protocol handler registration for PWAs.
Icons
Activate Windows
Go to Settings to activate Windows.
Console
Issues

5.2) Installing the PWA Using Microsoft Edge's "Add to Home Screen" Feature

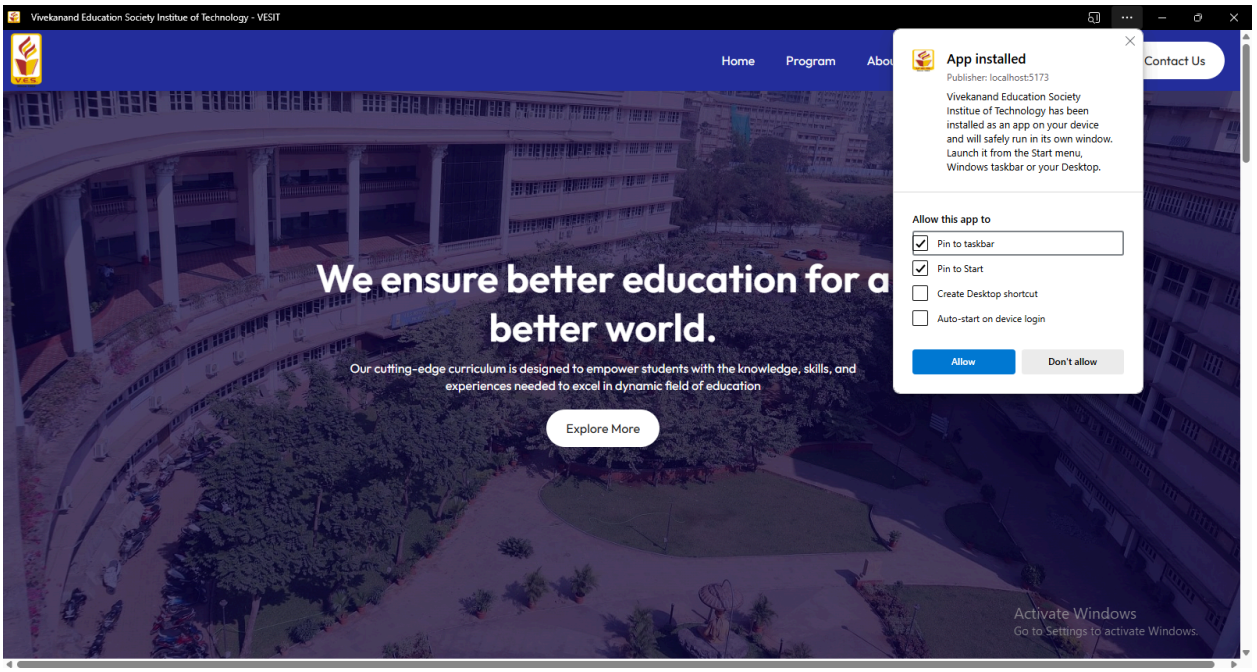


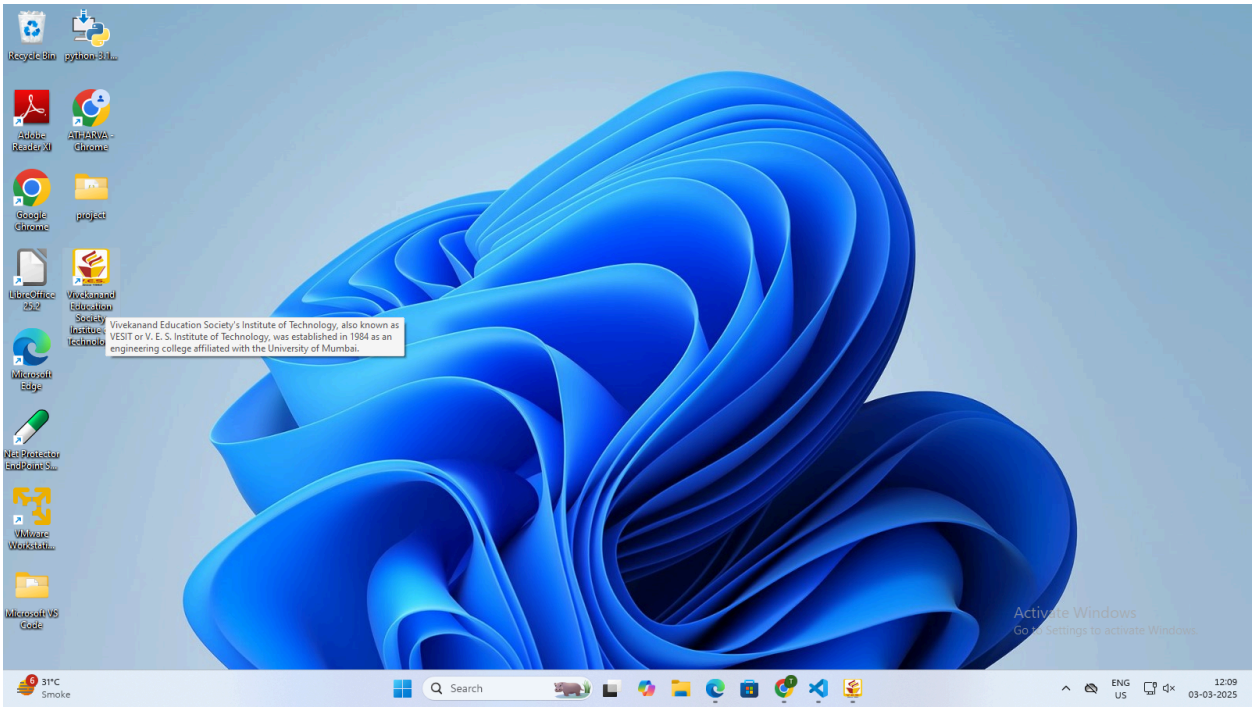
5.3) App icon along with name of the app will appear



5.4) PWA Successfully Installed

Users can choose to **pin the app to the taskbar, Start menu, or create a desktop shortcut** for easy access.





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.	16
Name	Shreyash Ganesh Dhekane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

EXPERIMENT NO: - 08

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

Theory: -

Service Worker

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

Things to note about Service Worker:

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

What can we do with Service Workers?

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

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

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

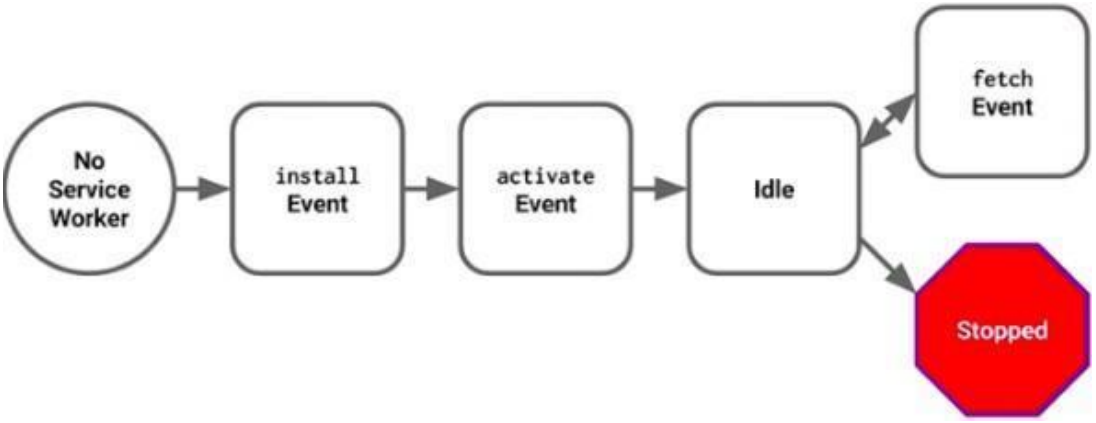
- You can **Continue**
  Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

What can't we do with Service Workers?

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

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

Service Worker Cycle



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

- Registration
- Installation
- Activation

Registration

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

main.js

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

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

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

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

main.js

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

});
```

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: -

main.jsx

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
```

```
import App from './App.jsx'
import './index.css'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
// src/index.js (or src/main.jsx)
if ('serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker
      .register('/service-worker.js')
      .then((registration) => {
        console.log('Service Worker registered with scope:', registration.scope);
      })
      .catch((error) => {
        console.log('Service Worker registration failed:', error);
      });
  });
}
```

service-worker.js

```
var staticCacheName = "pwa-v1"; // Versioned cache name

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

```
        "/",
        "/index.html",
        "/style.css",
        "/script.js",
        "/logo.png",
        "/icon1.png",
        "/offline.html"  // Offline fallback page
      ]);
    })
  );
});

self.addEventListener("activate", function (event) {
    var cacheWhitelist = [staticCacheName];  // Use staticCacheName directly

    event.waitUntil(
      caches.keys().then(function (cacheNames) {
        return Promise.all(
          cacheNames.map(function (cacheName) {
            if (cacheWhitelist.indexOf(cacheName) === -1) {
              return caches.delete(cacheName);  // Delete outdated caches
            }
          })
        );
      })
      .then(function() {
        // After the activation and cache cleaning, take control of the page immediately
        self.clients.claim();  // This ensures the new service worker takes control
      })
    );
});

self.addEventListener("fetch", function (event) {
  event.respondWith(
    caches.match(event.request).then(function (response) {
      if (response) {
        return response;  // Serve from cache
      }

      return fetch(event.request).catch(function () {
        return caches.match("/offline.html");  // Serve offline fallback
      });
    })
  );
});

self.addEventListener('message', (event) => {
```

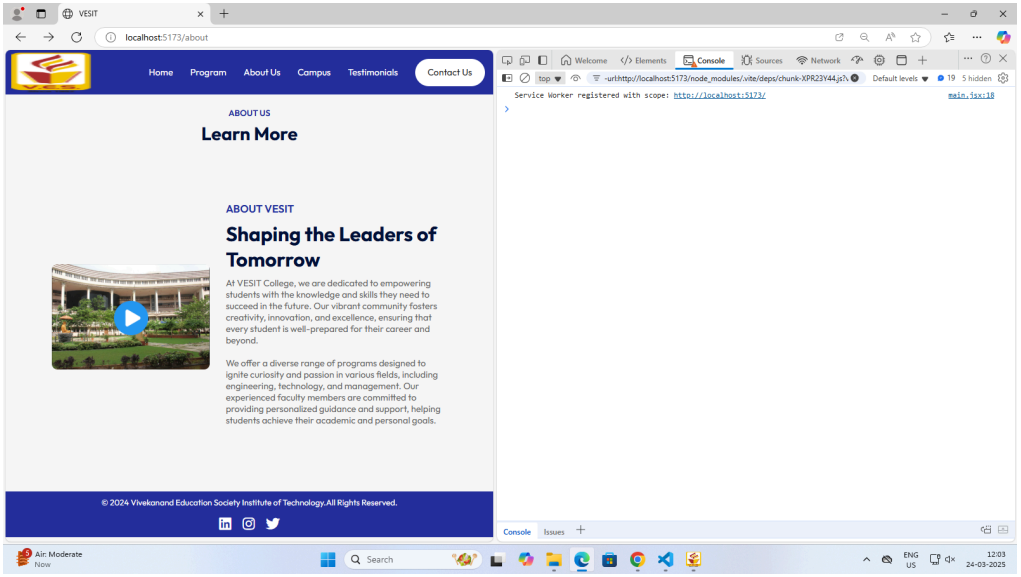
```
  if (event.data.action === 'skipWaiting') {
    self.skipWaiting();
  }
});
```

Index.html

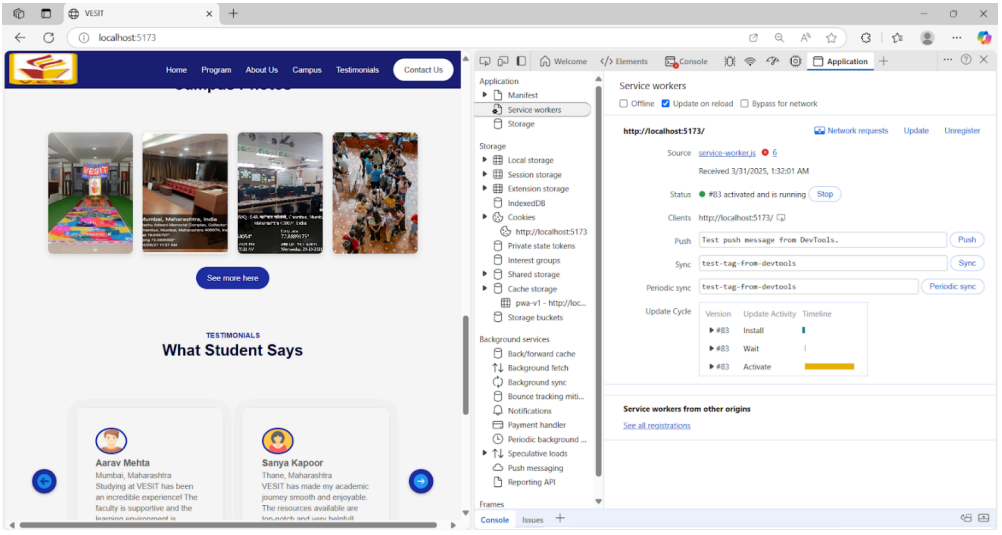
```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <link rel="manifest" href="/manifest.json" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>VESIT</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

OUTPUT: -

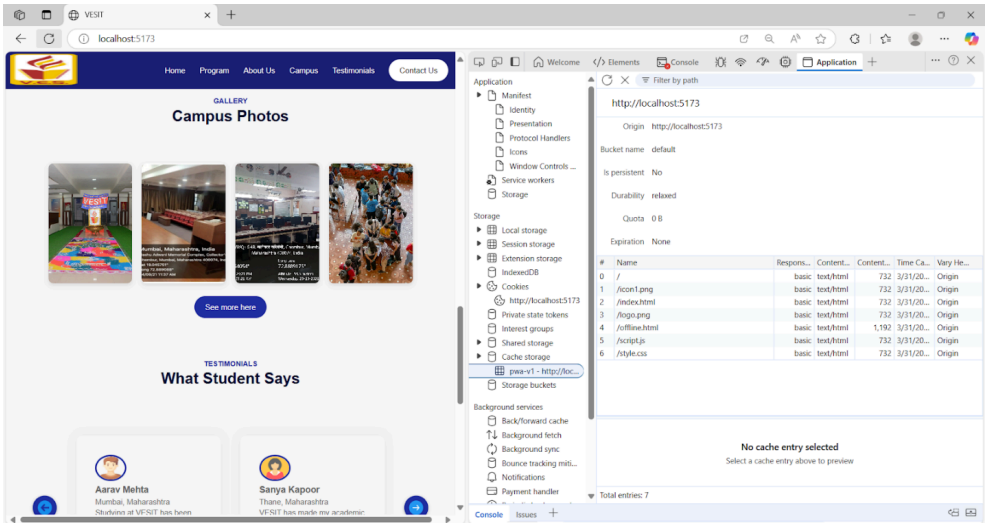
- Service worker successfully registered



- Service Worker Activate status



- Cache storage of the 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.	16
Name	Shreyash Ganesh Dhekane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

EXPERIMENT NO: - 09

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

Theory: -

Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

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

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

before or not. If a cache exists, it is returned to the page, but if not, this is up to you. You can return dummy content or information messages to the page.

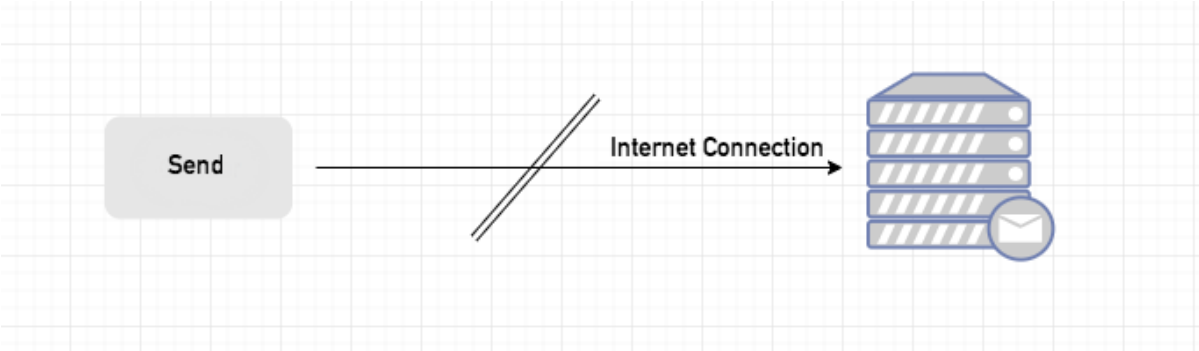
```
self.addEventListener("fetch", function (event) {
    const req = event.request;
    const url = new URL(req.url);

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

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

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

Sync Event

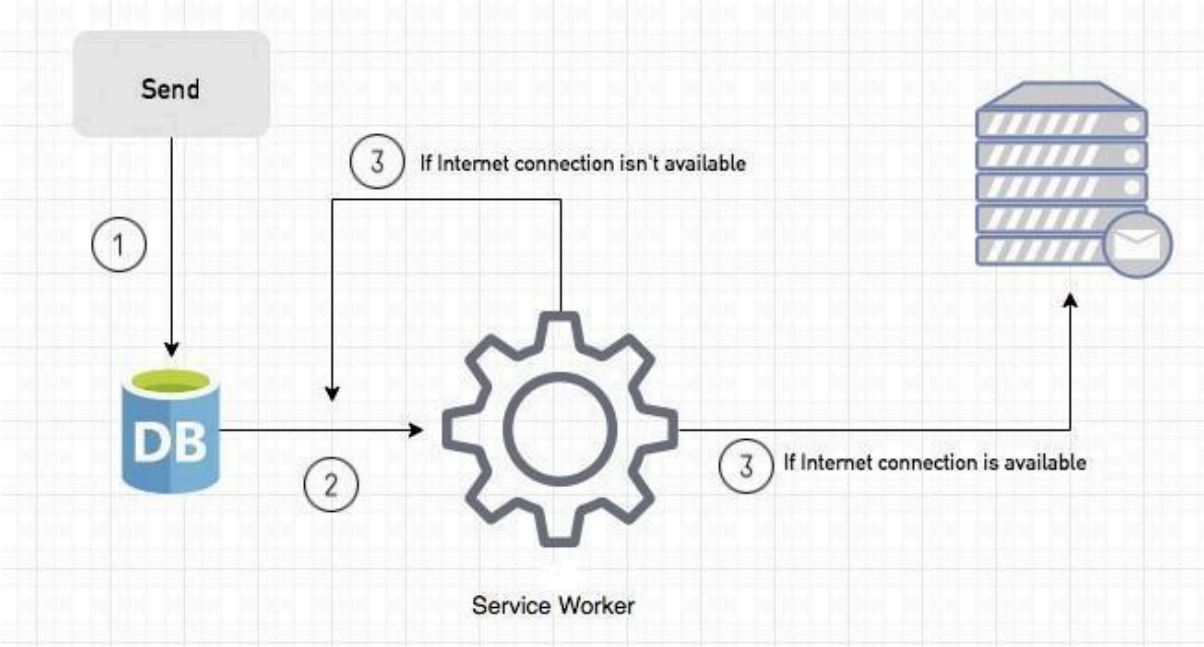
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

Here is a job for the Background Sync.

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



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



1. When we click the “send” button, email content will be saved to IndexedDB.
2. Background Sync registration.
3. **If the Internet connection is available**, all email content will be read and sent to Mail Server.

   If the Internet connection is unavailable, the service worker waits until the connection is available even though the window is closed. When it is available, email content will be sent to Mail Server.

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

Event Listener for Background Sync Registration

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

Event Listener for sw.js

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

Push Event

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

We can check in the following example.

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

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

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

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

Code: -

service-worker.js

```
var staticCacheName = "pwa-v1"; // Versioned cache name

// Install event: Caching static assets
self.addEventListener("install", function (e) {
  console.log("Service Worker: Installing...");
```

```
  e.waitUntil(
    caches.open(staticCacheName).then(function (cache) {
      console.log("Service Worker: Caching files...");
      return cache.addAll([
        "/",
        "/index.html",
        "/styles.css",
        "/app.js",
        "/logo.png",
        "/icon.png",
        "/offline.html" // Offline fallback page
      ]);
    })
  );
});

// Activate event: Cleanup old caches
self.addEventListener("activate", function (event) {
  console.log("Service Worker: Activating...");
  var cacheWhitelist = [staticCacheName];

  event.waitUntil(
    caches.keys().then(function (cacheNames) {
      return Promise.all(
        cacheNames.map(function (cacheName) {
          if (!cacheWhitelist.includes(cacheName)) {
            console.log("Service Worker: Deleting old cache:", cacheName);
            return caches.delete(cacheName);
          }
        })
      );
    }).then(() => {
      console.log("Service Worker: Now controlling all clients.");
      self.clients.claim();
    })
  );
});

// Fetch event: Serve from cache, then network, then offline fallback
self.addEventListener("fetch", function (event) {
  console.log("Fetching:", event.request.url);
  event.respondWith(
    caches.match(event.request).then(function (response) {
      if (response) {
        console.log("Fetch successful (from cache):", event.request.url);
        return response; // Serve from cache
      }
```

```
    return fetch(event.request, { credentials: "include" })
      .then((networkResponse) => {
        console.log("Fetch successful (from network):", event.request.url);
        return networkResponse;
      })
      .catch(() => {
        console.log("Fetch failed, serving offline page.");
        return caches.match("/offline.html");
      });
  })
 );
});

// Push Notification Event
self.addEventListener("push", function (event) {
 console.log("Push event received.");
 const data = event.data ? event.data.json() : { message: "Default notification" };

 const options = {
  body: data.message,
  icon: "/logo.png"
 };

 event.waitUntil(
  self.registration.showNotification("VESIT", options).then(() => {
   console.log("Push Notification displayed successfully.");
  })
 );
});

// Background Sync Event
self.addEventListener("sync", function (event) {
 console.log("Sync event received:", event.tag);

 if (event.tag === "syncMessage") {
  event.waitUntil(
   (async () => {
    console.log("Processing sync event...");

    // Simulate an actual sync operation (like posting data to server)
    try {
     let response = await fetch("/sync-endpoint", { method: "POST" });
     console.log("Sync request sent:", response.status);
    } catch (error) {
     console.error("Sync request failed:", error);
    }
```

```
      // Show notification
      await self.registration.showNotification("VESIT", {
        body: "Sync successful!",
        icon: "/logo.png"
      });

      console.log("Sync event handled successfully.");
    })()
  );
 }
});

// Message event for skipWaiting
self.addEventListener("message", (event) => {
 if (event.data.action === "skipWaiting") {
  console.log("Skipping waiting phase...");
  self.skipWaiting();
 }
});
```

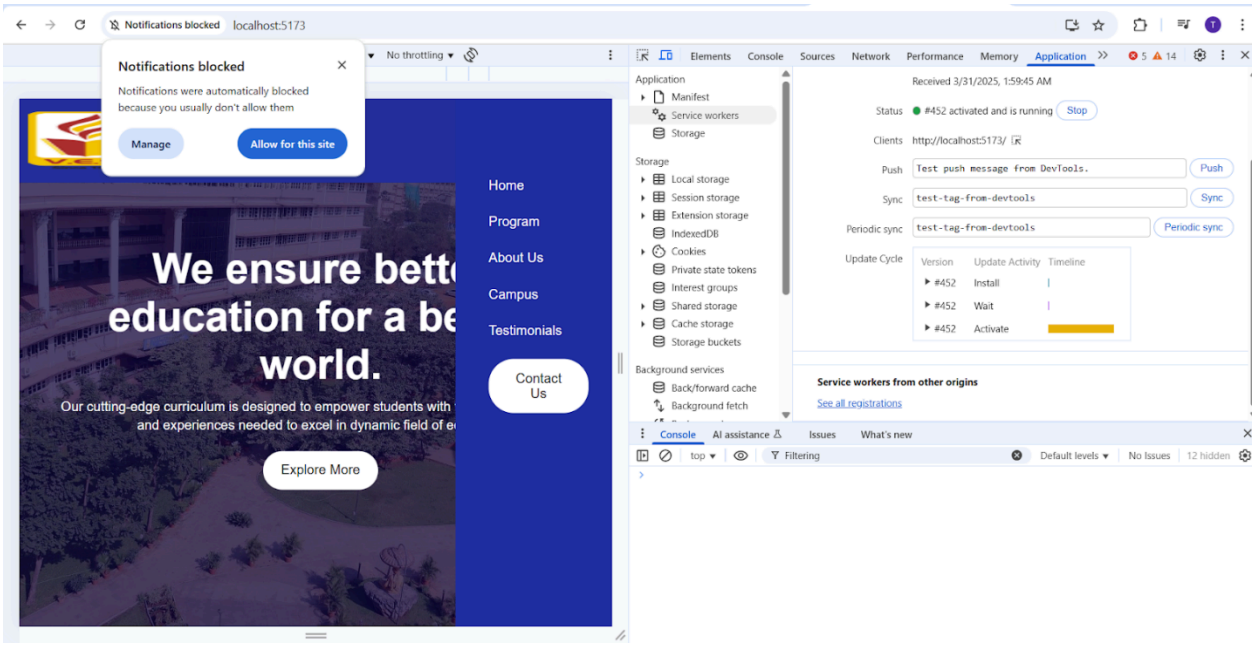
Main.jsx

```
import { StrictMode } from "react";
import { createRoot } from "react-dom/client";
import App from "./App.jsx";
import "./index.css";

createRoot(document.getElementById("root")).render(
 <StrictMode>
  <App />
 </StrictMode>
);

// Register Service Worker
if ("serviceWorker" in navigator) {
 window.addEventListener("load", () => {
  navigator.serviceWorker
   .register("/service-worker.js")
   .then((registration) => {
    console.log("Service Worker registered successfully with scope:", registration.scope);

    // Request Notification Permission
    Notification.requestPermission().then((permission) => {
     if (permission === "granted") {
      console.log("Notification permission granted.");
     } else {
      console.warn("Notification permission denied.");
```

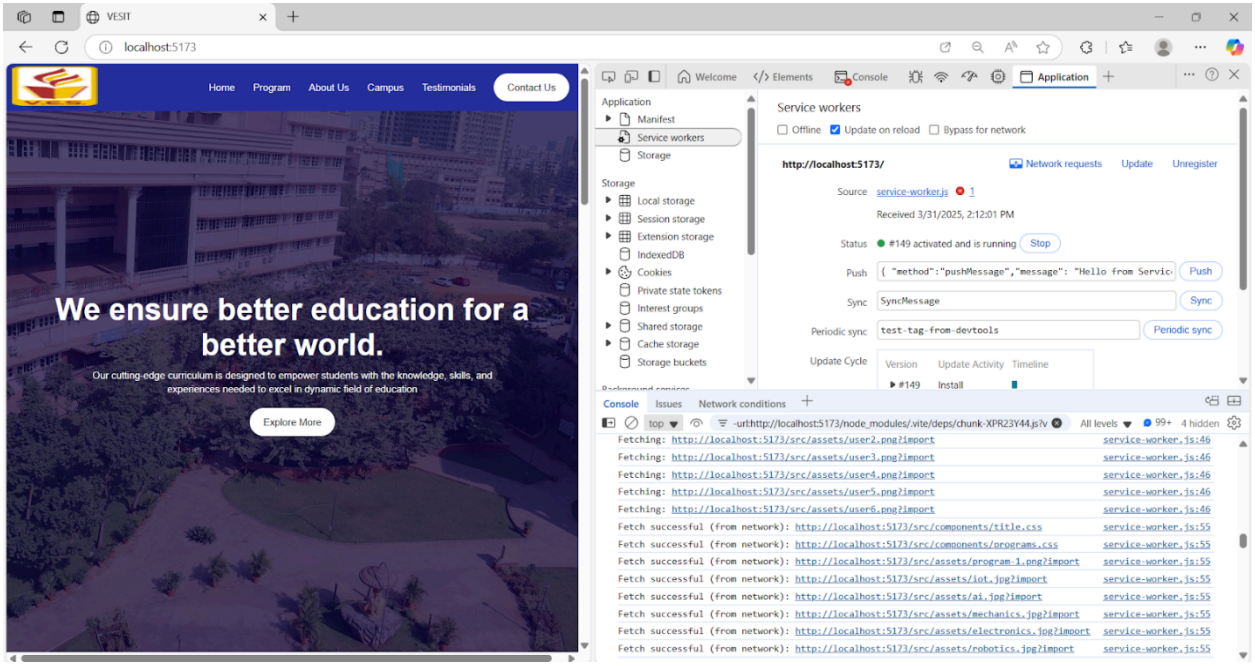
```
      }
    });

    if ('serviceWorker' in navigator && 'SyncManager' in window) {
      navigator.serviceWorker.ready.then(registration => {
          return registration.sync.register('syncMessage')
              .then(() => console.log('Sync event registered'))
              .catch(err => console.error('Sync registration failed', err));
      });
   }


    // Push Notification Setup
    navigator.serviceWorker.ready.then((reg) => {
      reg.pushManager.subscribe({
        userVisibleOnly: true,
        applicationServerKey:
"BD13Iv9g2jOfJ-7JtItR7smZV7rk5DgFWfB43GLh7HgjrATPzJKDS8jCGYFNnaTutJM-5oN885EjYB
0k1CfOG2s" // Replace with actual VAPID key
      }).then((subscription) => {
        console.log("Push Notification Subscription successful:", subscription);
      }).catch((error) => {
        console.error("Push Notification Subscription failed:", error);
      });
    });

    // Background Sync Setup
    navigator.serviceWorker.ready.then((reg) => {
      if ("sync" in reg) {
        reg.sync.register("syncMessage").then(() => {
          console.log("Background sync registered successfully.");
        }).catch((error) => {
          console.error("Background sync registration failed:", error);
        });
      } else {
        console.warn("SyncManager is not supported.");
      }
    });

  })
  .catch((error) => {
    console.error("Service Worker registration failed:", error);
  });
});
}
```

OUTPUT: -

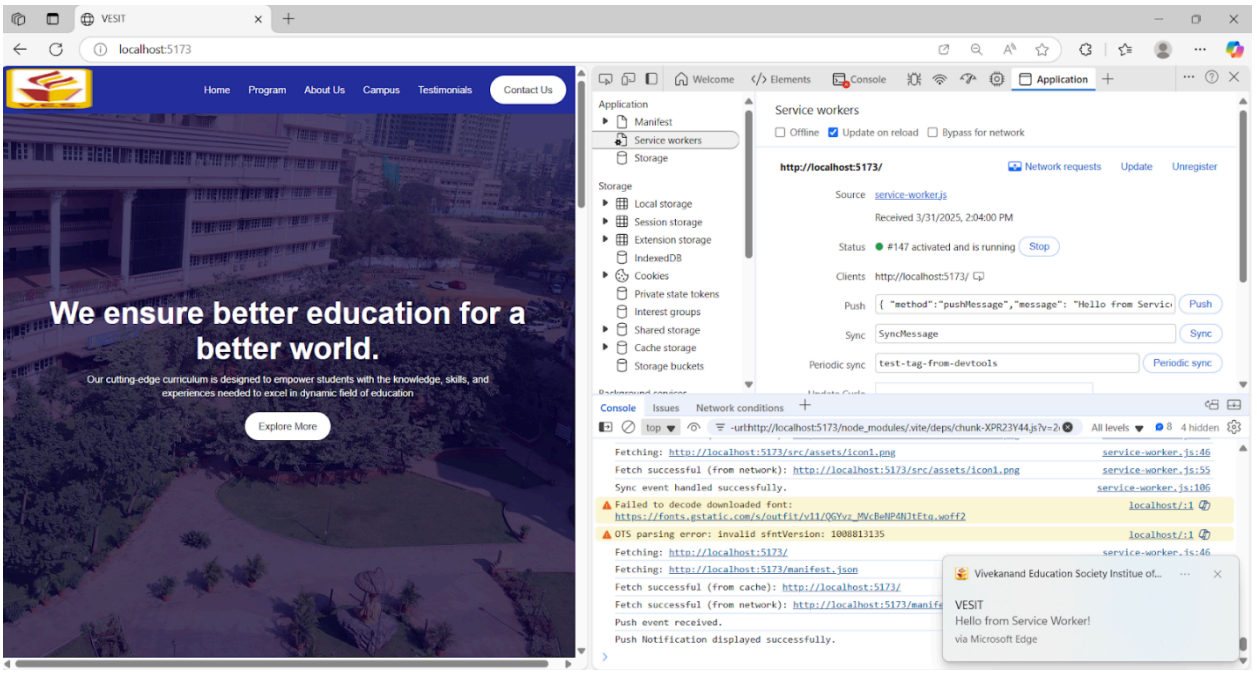
⍰ Notification Permission



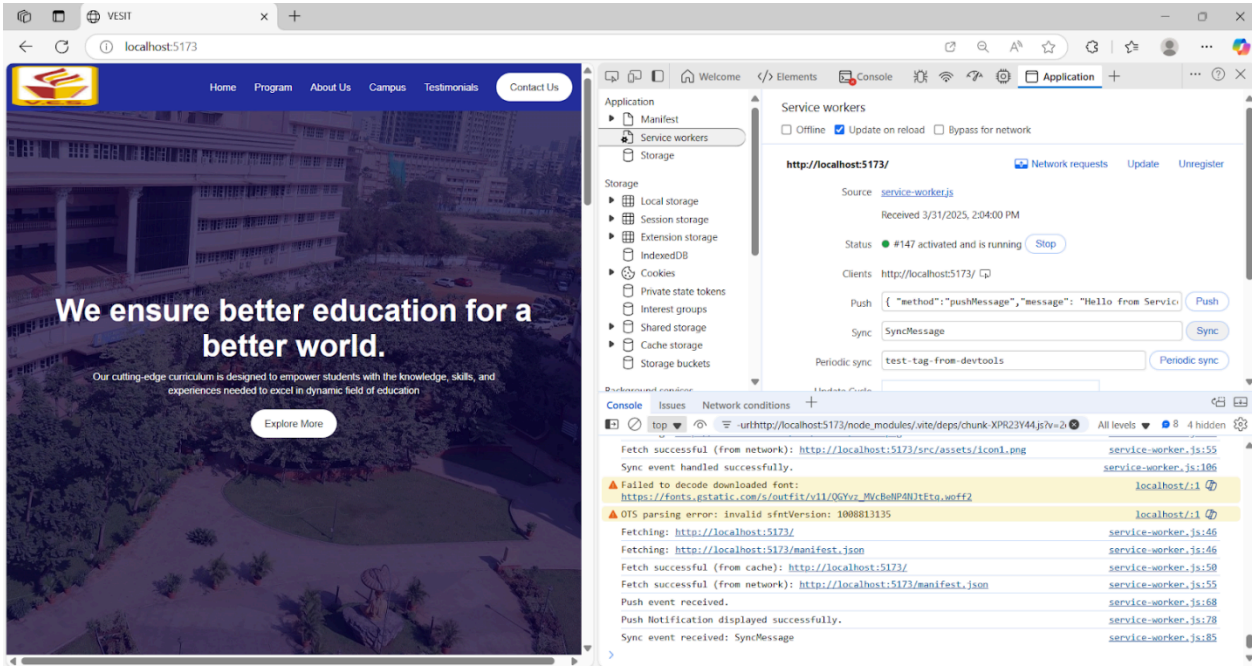
⍰ Fetch Event

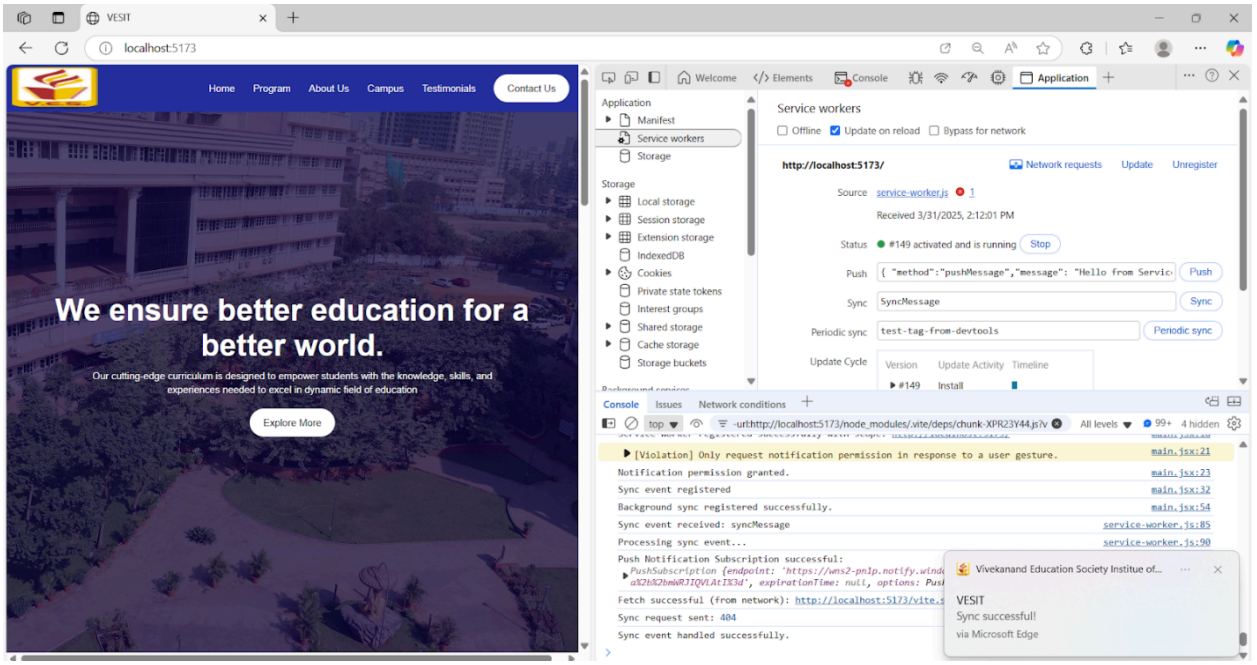


Push Event



Sync Event


VESIT
localhost:5173
Home
Program
About Us
Campus
Testimonials
Contact Us
We ensure better education for a better world.
Our cutting-edge curriculum is designed to empower students with the knowledge, skills, and experiences needed to excel in dynamic field of education
Explore More
Welcome
Elements
Console
Application
Service workers
Offline
Update on reload
Bypass for network
http://localhost:5173/
Network requests
Update
Unregister
Source service-worker.js
Received 3/31/2025, 2:04:00 PM
Status #147 activated and is running
Stop
Clients http://localhost:5173/
Push
Sync
SyncMessage
Periodic sync
test-tag-from-devtools
Console
Issues
Network conditions
Fetch successful (from network): http://localhost:5173/src/assets/icon1.png
Sync event handled successfully.
Failed to decode downloaded font:
OTS parsing error: invalid sfntVersion: 1008813135
Fetching: http://localhost:5173/
Fetching: http://localhost:5173/manifest.json
Fetch successful (from cache): http://localhost:5173/
Fetch successful (from network): http://localhost:5173/manifest.json
Push event received.
Push Notification displayed successfully.
Sync event received: SyncMessage


VESIT
localhost:5173
Home
Program
About Us
Campus
Testimonials
Contact Us
We ensure better education for a better world.
Our cutting-edge curriculum is designed to empower students with the knowledge, skills, and experiences needed to excel in dynamic field of education
Explore More
Service workers
Offline
Update on reload
Bypass for network
http://localhost:5173/
Source service-worker.js
Received 3/31/2025, 2:12:01 PM
Status #149 activated and is running
Update Cycle
Version
Update Activity
Timeline
Install
[Violation] Only request notification permission in response to a user gesture.
Notification permission granted.
Sync event registered
Background sync registered successfully.
Sync event received: syncMessage
Processing sync event...
Push Notification Subscription successful:
Fetch successful (from network): http://localhost:5173/vite.
Sync request sent: 404
Sync event handled successfully.
Vivekanand Education Society Institue of...
VESIT
Sync successful!
via Microsoft Edge

MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	16
Name	Shreyash Ganesh Dhekane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

EXPERIMENT NO: - 10

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

Theory: -

GitHub Pages

GitHub Pages is a free hosting service that allows users to publish public webpages directly from a GitHub repository. It is particularly useful for static websites, project documentation, and blogs.
It seamlessly integrates with Git version control, ensuring automatic updates with every commit. With built-in support for Jekyll, custom domains, and HTTPS (for GitHub subdomains), it provides an easy and efficient way to deploy static sites.

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 , watch it update in real-time.

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

Pros

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

Cons

1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.

3. No in-built support for any static site generator.

Link to our Github Repository: -

https://github.com/tejasgunjal021/college_pwa_website

Link to our Hosted Website : -

https://tejasgunjal021.github.io/college_pwa_website/

Method Followed [gh- pages] : -

We deployed a **React-based PWA** to **GitHub Pages** using the gh-pages package. The process involved:

1. **Setting up GitHub Pages** – Configuring the repository to support deployment.
2. **Adding homepage and scripts in package.json** – Ensuring proper routing support.
3. **Building the React App** – Creating the production-ready files.
4. **Deploying using gh-pages** – Pushing the build folder to the gh-pages branch.
5. **Verifying the Deployment** – Ensuring the website works correctly on GitHub Pages.

Github Screenshots: -

- **Pushing my PWA application into the github repository**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    powershell

PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master> git init
Initialized empty Git repository in C:/Users/TEJAS/Desktop/ip-exp6-master/ip-exp6-master/.git/
PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master> git add .
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'eslint.config.js', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/App.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/About.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Campus.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Contact.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Footer.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Hero.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Navbar.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Programcard.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Programs.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Testimonials.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Title.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/VideoPlayer.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/about.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/campus.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/contact.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/footer.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/hero.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/navbar.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/programs.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/testimonials.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/title.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/videoplayer.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/index.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/main.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'vite.config.js', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master> git commit -m "Initial Commit"
[master (root-commit) 24ee19e] Initial Commit
 63 files changed, 5926 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 README.md
 create mode 100644 eslint.config.js
 create mode 100644 index.html
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    powershell

PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master> git remote add origin https://github.com/tejasgunjal021/college_pwa_website.git
PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master> git branch -M main
PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master> git push -u origin main
Enumerating objects: 67, done.
Counting objects: 100% (67/67), done.
Delta compression using up to 8 threads
Compressing objects: 100% (67/67), done.
Writing objects: 100% (67/67), 80.32 MiB | 3.83 MiB/s, done.
Total 67 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/tejasgunjal021/college_pwa_website.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master>
```

- **Installing gh-pages Package** – Setting up the deployment tool.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    powershell

PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master> npm install gh-pages --save-dev
>>

added 44 packages, and audited 308 packages in 4s

116 packages are looking for funding
  run `npm fund` for details

3 moderate severity vulnerabilities

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master>
```

- **Adding homepage and Deployment scripts in package.json**

```
{
 "name": "college-website",
 "homepage": "https://tejasgunjal021.github.io/college_pwa_website",
```

```
 "private": true,
 "version": "0.0.0",
 "type": "module",
 "scripts": {
  "dev": "vite",
  "build": "vite build",
  "lint": "eslint .",
  "preview": "vite preview",
  "predeploy": "vite build",
  "deploy": "gh-pages -d dist"
 },
 "dependencies": {
  "@fortawesome/fontawesome-free": "^6.6.0",
  "react": "^18.3.1",
  "react-dom": "^18.3.1",
  "react-router-dom": "^6.27.0"
 },
 "devDependencies": {
  "@eslint/js": "^9.11.1",
  "@types/react": "^18.3.10",
  "@types/react-dom": "^18.3.0",
  "@vitejs/plugin-react": "^4.3.2",
  "eslint": "^9.11.1",
  "eslint-plugin-react": "^7.37.0",
  "eslint-plugin-react-hooks": "^5.1.0-rc.0",
  "eslint-plugin-react-refresh": "^0.4.12",
  "gh-pages": "^6.3.0",
  "globals": "^15.9.0",
  "vite": "^5.4.8"
 }
}
```

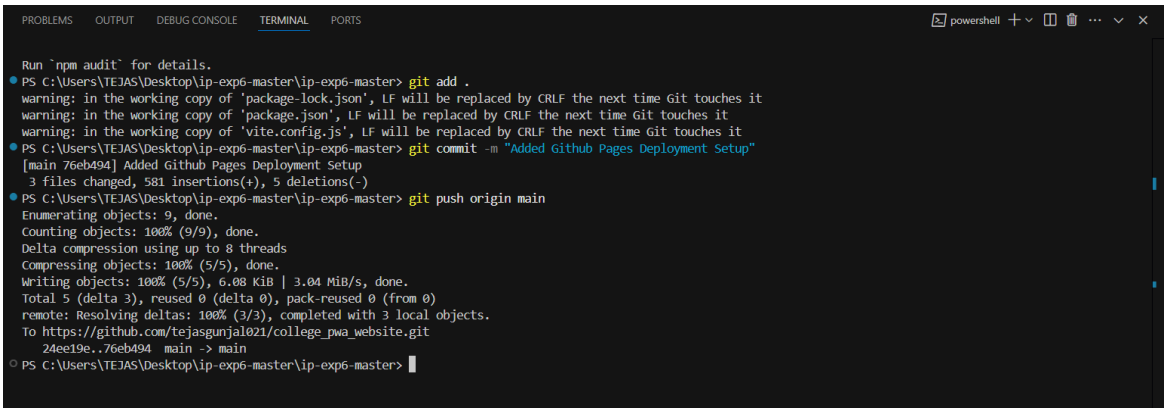
- **Update vite.config.js**

  Since Vite serves the app from / by default, you must configure it to serve from the correct subdirectory.

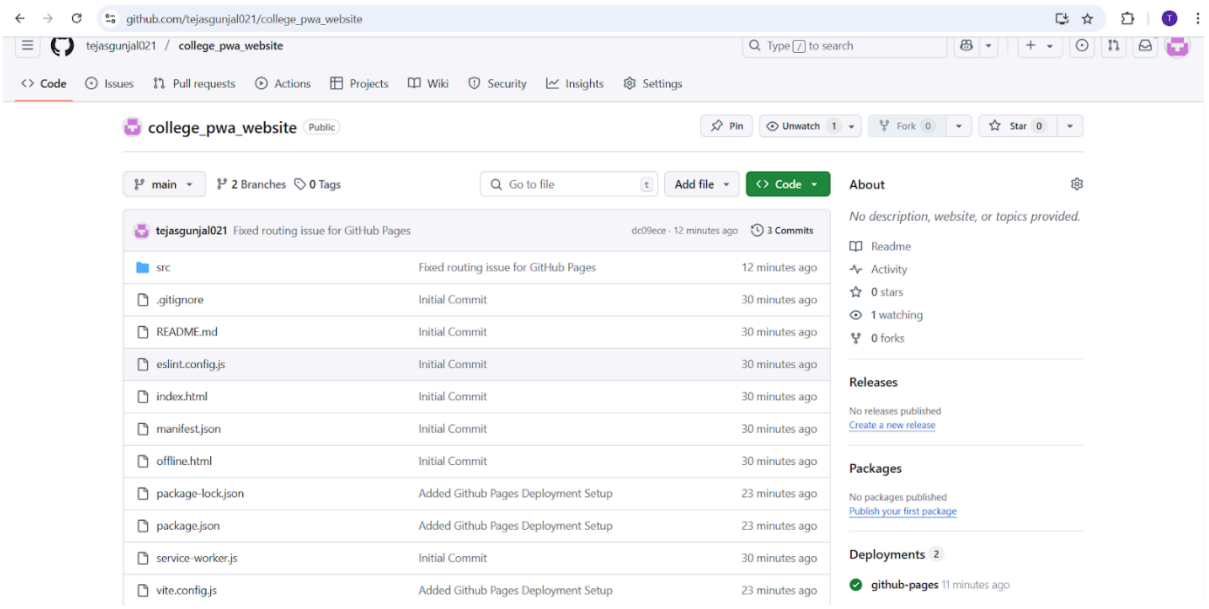
```
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react";
```

```
export default defineConfig({
  plugins: [react()],
  base: "/college_pwa_website/",  // Add this line
});
```

- **Updating the changes in the repository**



- **Code pushed to the repository**

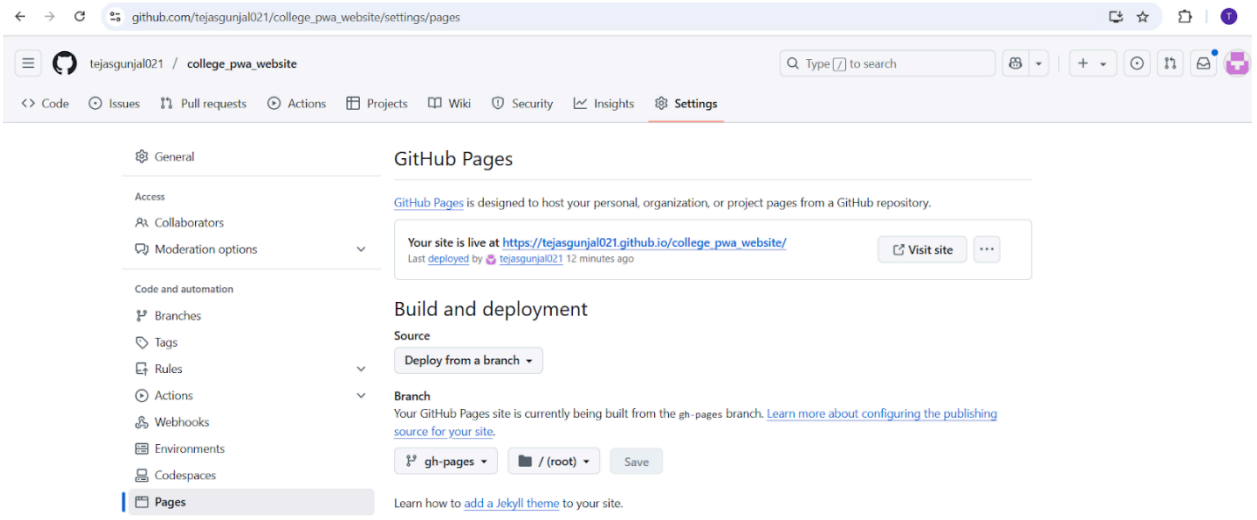


- **Deploy to github pages**

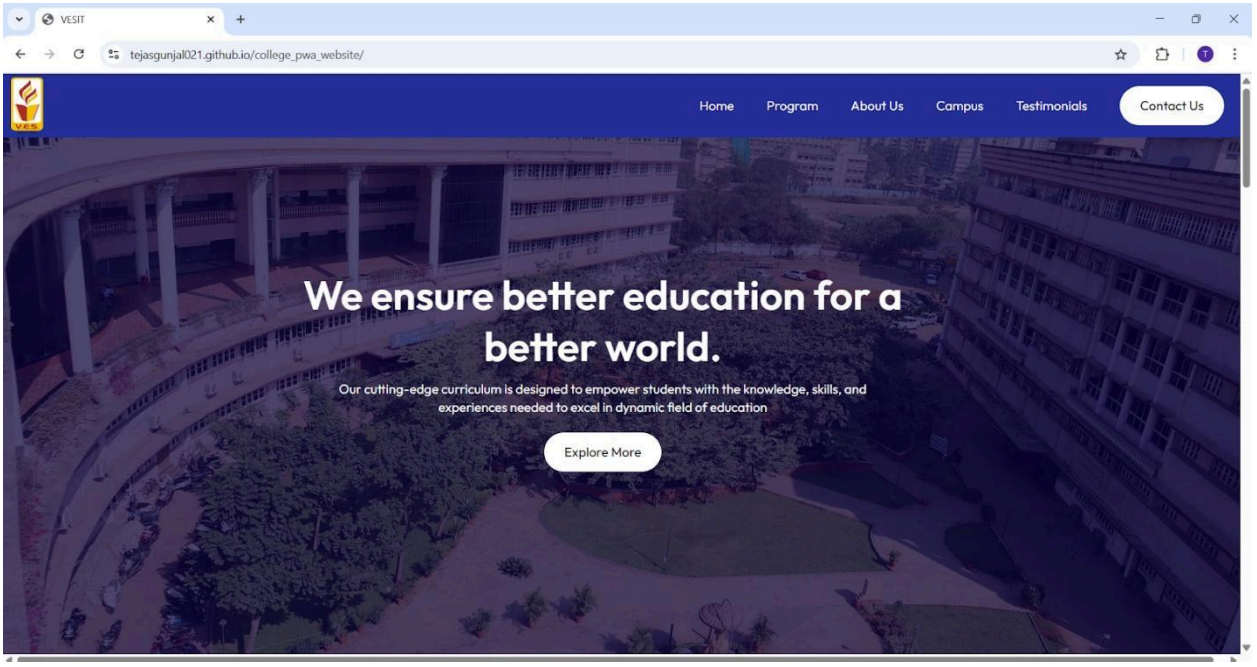
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master> npm run deploy

> college-website@0.0.0 predeploy
> vite build

vite v5.4.14 building for production...
✓ 82 modules transformed.
dist/index.html                          0.58 kB │ gzip:  0.33 kB
dist/assets/manifest-G75MMNiJ.json       0.71 kB │ gzip:  0.35 kB
dist/assets/fa-v4compatibility-C9RhG_FT.woff2      4.80 kB
dist/assets/fa-v4compatibility-CCth-dXg.ttf       10.84 kB
dist/assets/msg-BJ1-p5Un.png              13.63 kB
dist/assets/logo-Bu6wwlcJ.png             14.24 kB
dist/assets/icon-C1OtVbCZ.png             17.12 kB
dist/assets/play-DZJCAgKb.png             17.42 kB
dist/assets/right-arrow-BcmZ5rr2.png      17.59 kB
dist/assets/left-arrow-D-vK3ks6.png       18.44 kB
dist/assets/user1-DjTToPK1.png            24.41 kB
dist/assets/ai-D8sSgONg.jpg               25.19 kB
dist/assets/fa-regular-400-BjRzuEpd.woff2      25.47 kB
dist/assets/user2-Dvg8F2pi.png            27.28 kB
dist/assets/location-CsUE0Atg.png         30.25 kB
dist/assets/user6-g8XIt7bn.png            32.17 kB
dist/assets/user3-CZ6tJ5BY.png            36.06 kB
dist/assets/user4-BtQSOxcZ.png            36.71 kB
dist/assets/user5-CXG3eEZr.png            38.78 kB
dist/assets/iot-By66MvuU.jpg              40.74 kB
dist/assets/mail-Bryx0Fa9.png             42.74 kB
dist/assets/electronics-CxeaB2SO.jpg      44.67 kB
dist/assets/mechanics-7TpPCQUc.jpg        57.48 kB
dist/assets/fa-regular-400-DZaxPHgR.ttf        68.06 kB
dist/assets/fa-brands-400-D_cYUPeE.woff2      118.68 kB
dist/assets/fa-solid-900-CTAAxXor.woff2       158.22 kB
dist/assets/robotics-Bmec4tgd.jpg         180.11 kB
dist/assets/fa-brands-400-D1LuMI3I.ttf         210.79 kB
dist/assets/program-1-DYXM3znK.png        270.08 kB
dist/assets/fa-solid-900-D0aA9rwL.ttf          426.11 kB
dist/assets/gallery-1-BGCASvfo.png        517.44 kB
dist/assets/about-BMKaheDG.png          1,327.94 kB
dist/assets/college-Dx6yeVSG.png        4,579.16 kB
dist/assets/gallery-0-CHbQgws-.png      7,094.24 kB
```

Enable Github Pages

1. Go to your repository on GitHub.
2. Click on **Settings** > **Pages**.
3. In that you will see the github hosted link for your application.



Hosted Website: - https://tejasgunjal021.github.io/college_pwa_website/


VESIT
tejasgunjal021.github.io/college_pwa_website/
Home
Program
About Us
Campus
Testimonials
Contact Us
We ensure better education for a better world.
Our cutting-edge curriculum is designed to empower students with the knowledge, skills, and experiences needed to excel in dynamic field of education
Explore More

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	16
Name	Shreyash Ganesh Dhekane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

EXPERIMENT NO: - 11

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

Theory: -

Google Lighthouse is an open-source tool designed to evaluate and optimize web applications based on multiple key parameters, including performance, accessibility, Progressive Web App (PWA) implementation, and best practices. It provides a detailed automated audit that helps developers improve their websites efficiently. Unlike traditional manual audits that can take days or weeks, Lighthouse generates insights within minutes with minimal setup.

One of Lighthouse's biggest advantages is its ease of use—simply run it on a webpage or provide a URL, and it will generate a comprehensive report. Lighthouse supports audits for both desktop and mobile versions of a website, ensuring an optimal user experience across different devices.

Key Features and Audit Metrics

1. Performance

This metric evaluates how efficiently a webpage loads and becomes interactive. It considers several factors, including:

- Page Load Speed – Measures how quickly content appears to users.
- First Contentful Paint (FCP) – Time taken for the first visible content to load.
- Largest Contentful Paint (LCP) – Measures how long the largest visible content takes to fully render.
- Cumulative Layout Shift (CLS) – Ensures content does not shift unexpectedly, improving user experience.
- Time to Interactive (TTI) – The time taken for the webpage to become fully functional.

Lighthouse assigns a performance score from 0 to 100, where:

- 100 → Top 2% of websites (98th percentile)
- 50 → Around the 75th percentile
- Lower scores → Indicate areas needing optimization

2. Progressive Web App (PWA) Score

With the increasing adoption of PWAs, web applications now strive to deliver an app-like experience. Lighthouse evaluates a website's PWA readiness based on Google's Baseline PWA Checklist, which includes:

- Service Worker Implementation – Enables offline functionality and background synchronization.
- App Manifest Compliance – Provides metadata for mobile app-like integration.
- Viewport Configuration – Ensures responsiveness across different screen sizes.
- Performance in Script-Disabled Environments – Verifies that the page functions properly even if JavaScript is disabled.

A high PWA score indicates that the application meets essential criteria for speed, reliability, and mobile usability.

3. Accessibility

This metric determines how well a website supports users with **visual, cognitive, or physical disabilities**. Lighthouse evaluates accessibility based on:

- **ARIA Attributes** – Enhances screen reader support (e.g., aria-required).
- **Text Alternatives for Media** – Ensures images, audio, and video content are accessible.
- **Semantic HTML Usage** – Encourages proper use of elements like <section>, <article>, and <button> for better screen-reader compatibility.

Unlike other metrics, **accessibility follows a pass/fail approach**—missing key features can **significantly impact the overall score**. Improving accessibility ensures **inclusivity for all users**.

4. Best Practices

Lighthouse also assesses whether the website follows **modern web development best practices**, including:

- **Use of HTTPS** – Ensures secure data transmission.
- **Avoiding Deprecated Code** – Prevents the use of outdated elements, directives, and libraries.
- **Secure Password Inputs** – Disables paste-into fields to reduce security risks.
- **User Security Alerts** – Provides notifications for **geo-location access and cookie usage**.

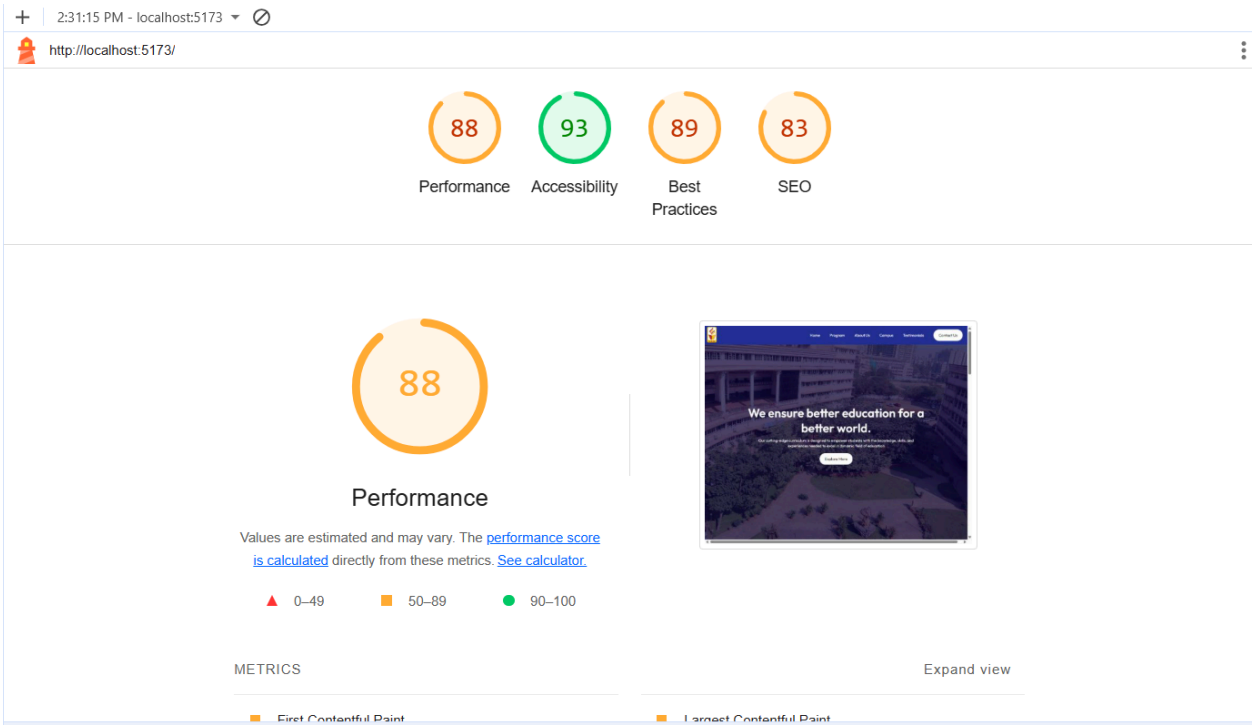
A high **Best Practices score** means the website meets industry standards, leading to better **security, maintainability, and overall performance**.

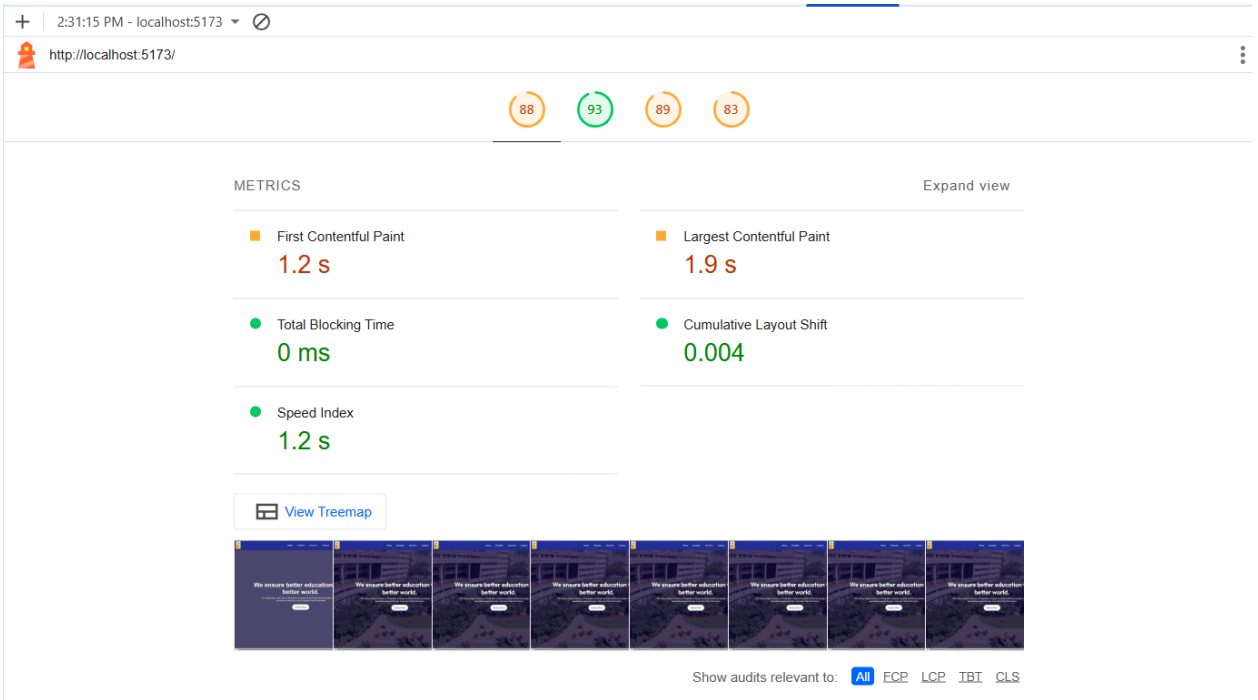
Manifest.json

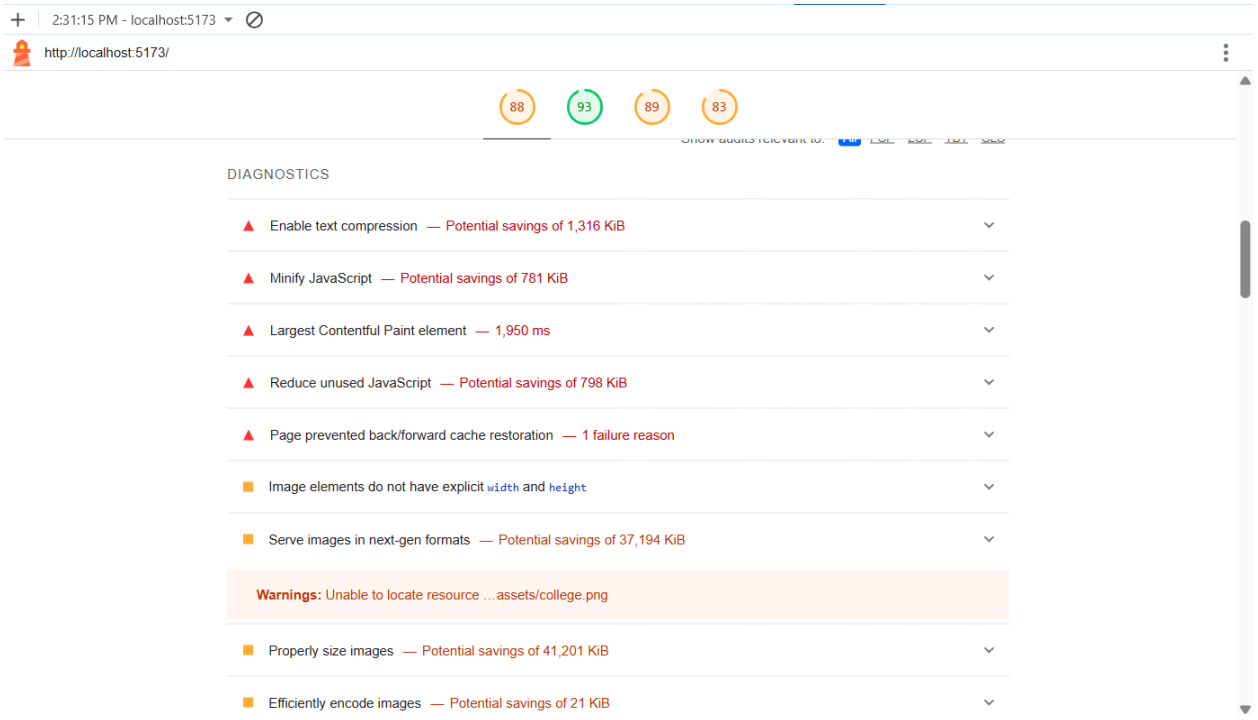

```
{
  "short_name": "VESIT",
  "name": "Vivekanand Education Society Institue of Technology",
   "description": "Vivekanand Education Society's Institute of Technology, also known as VESIT or V. E. S. Institute of Technology, was established in 1984 as an engineering college affiliated with the University of Mumbai.",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#000000",
  "icons": [
    {
      "src": "/src/assets/icon1.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/src/assets/icon1.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```


Output: -

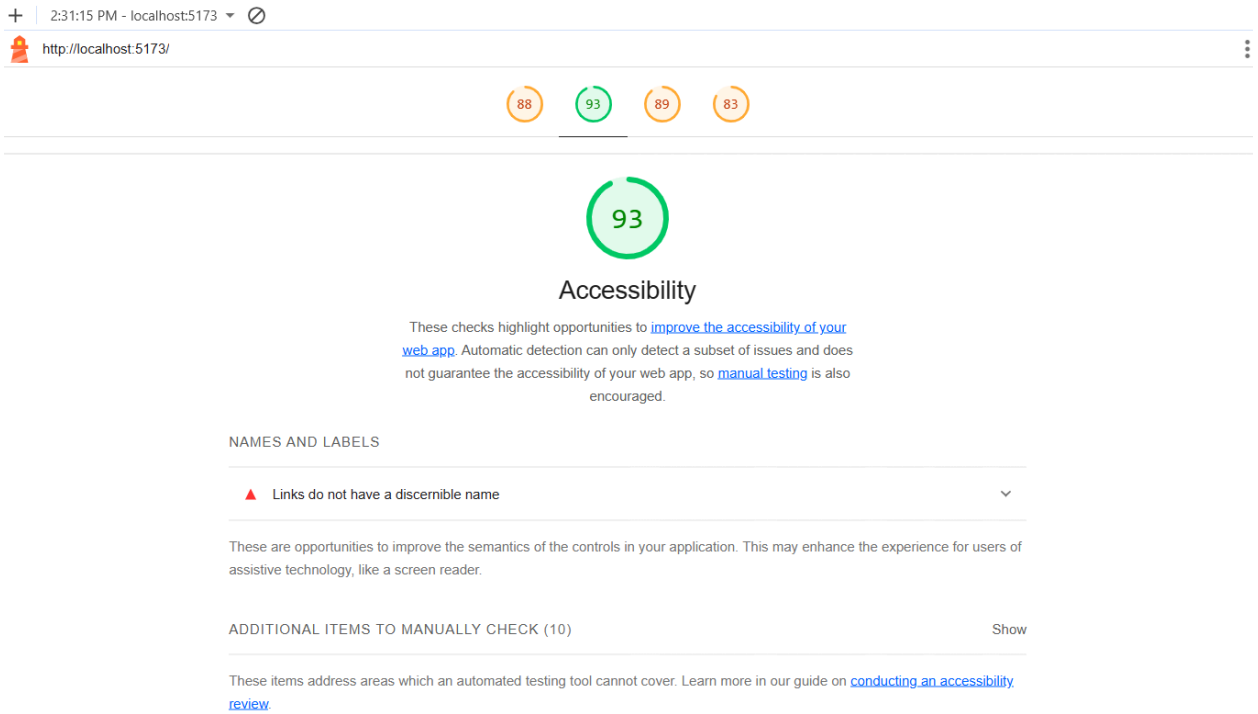
a) Performance



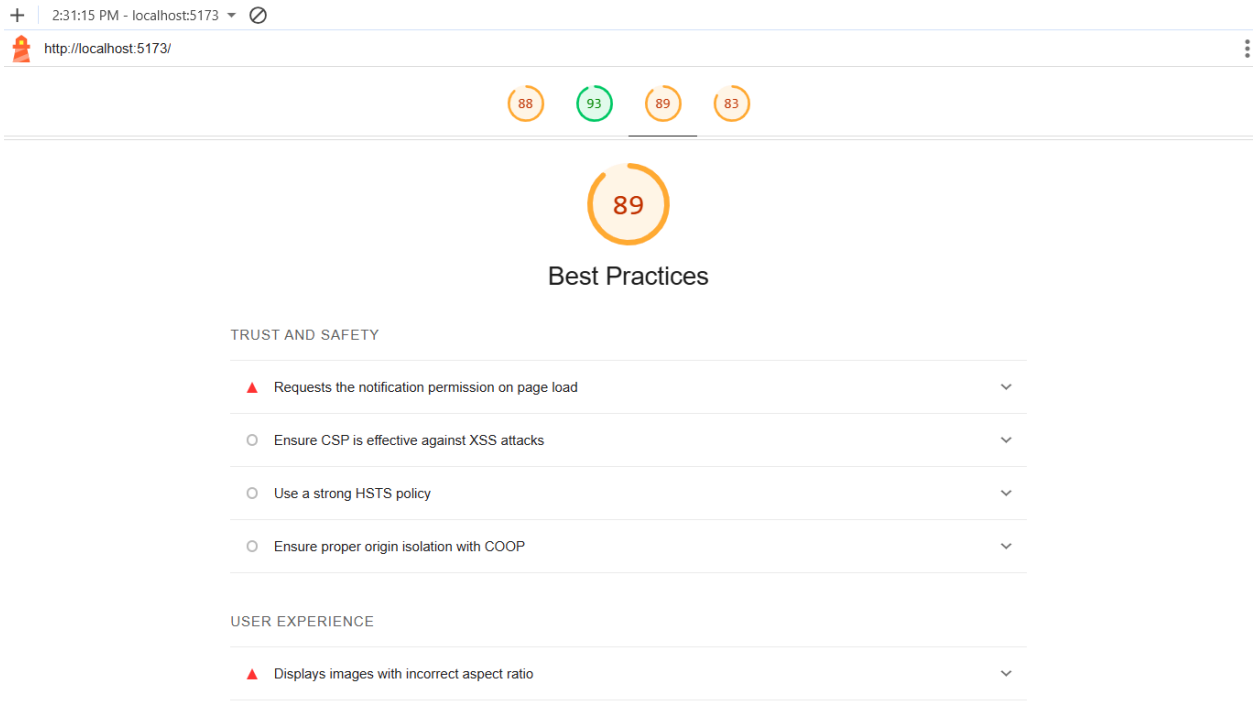




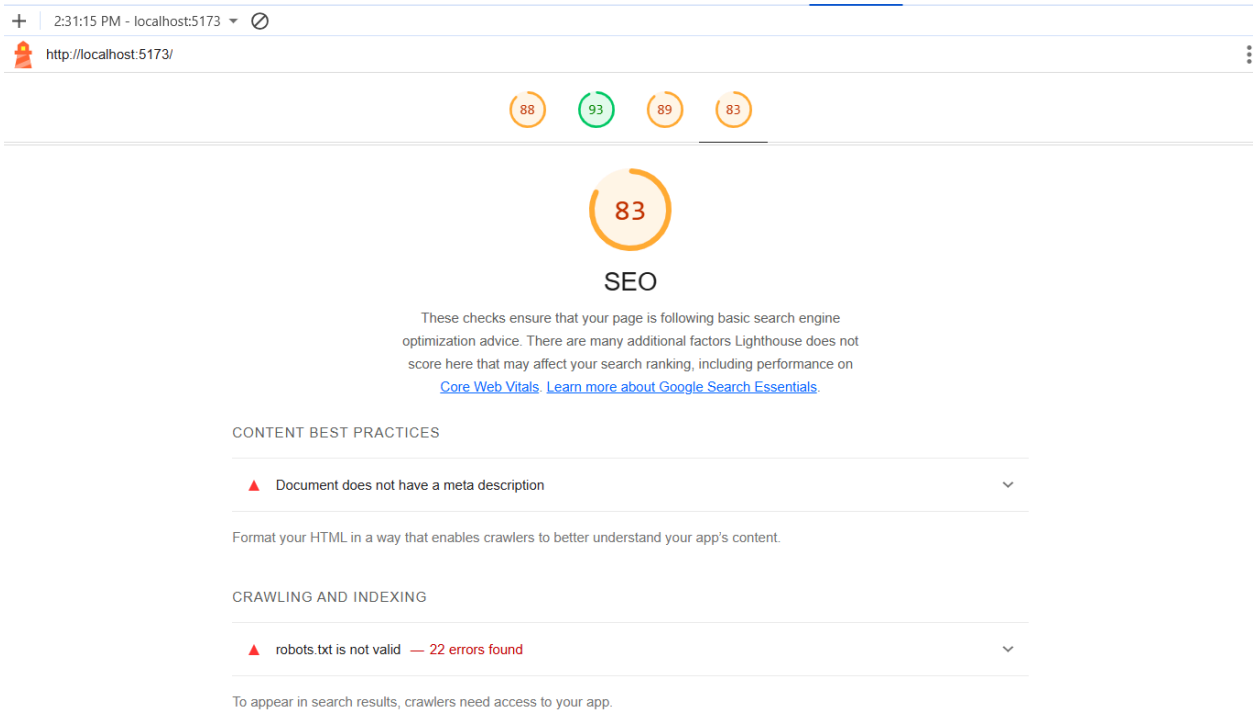
b) Accessibility



c) Best Practices



d) SEO [Search Engine Optimization]



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.	16
Name	Shreyash Ganesh Dhekane
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:	

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.	16
Name	Shreyash Ganesh Dhekane
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:	