

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

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Name of the Course : MAD & PWA Lab

Course Code : ITL604

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

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers : Mrs. Kajal Joseph.

Email : kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

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:

On Completion of the course the learner/student should be able to:		
Sr. No.	Lab Outcomes	Cognitive levels of attainment as per Bloom's Taxonomy
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.	Assignment-1
Assignment 1 Questions	<p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p>
Roll No.	55
Name	Prajyot Shinde
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	<p>LO1: Understand cross platform mobile application development using Flutter framework</p> <p>LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation</p> <p>LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS</p>
Grade:	

MAD & PWA Lab

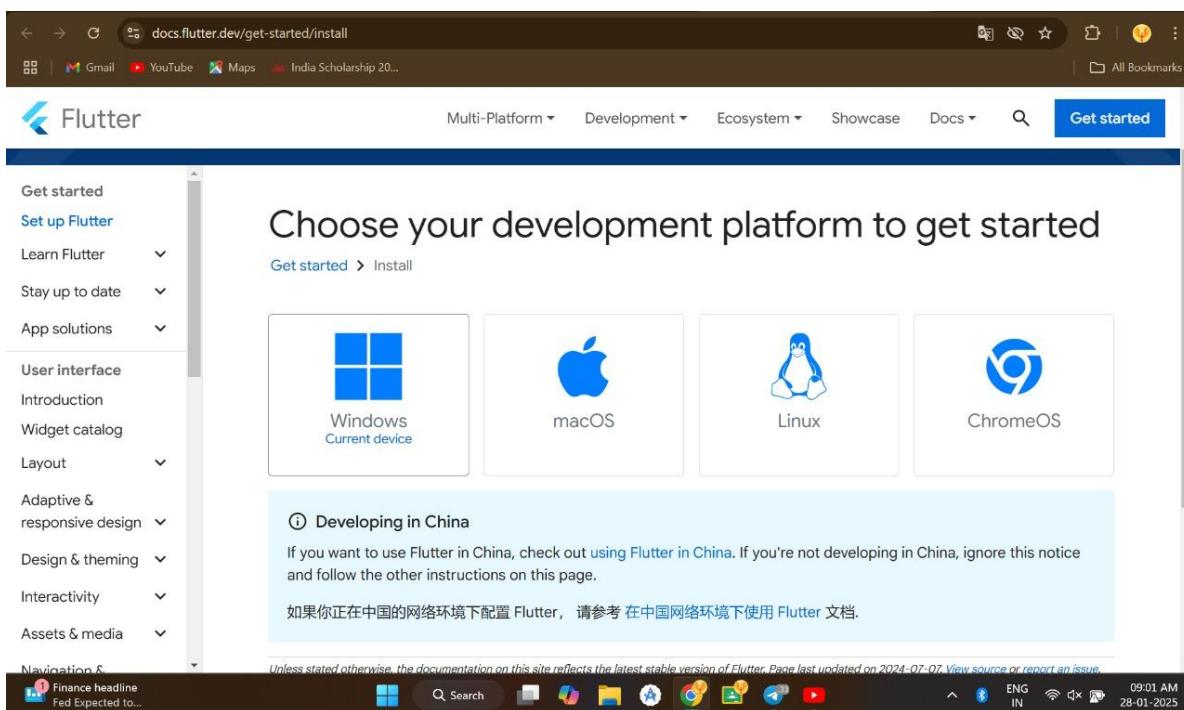
Journal

Experiment No.	Assignment-2
Assignment 2 Questions	<ol style="list-style-type: none"> 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.	55
Name	Prajyot Shinde
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:	

MAD & PWA Lab

Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	55
Name	Prajyot Shinde
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15

EXPERIMENT NO: - 01**Name:-** Prajyot Shinde**Class:-** D15A**Roll:No:** - 55**AIM:** - Installation and Configuration of Flutter Environment.**Step 1:** Go to the official Flutter website: <https://docs.flutter.dev/get-started/install>

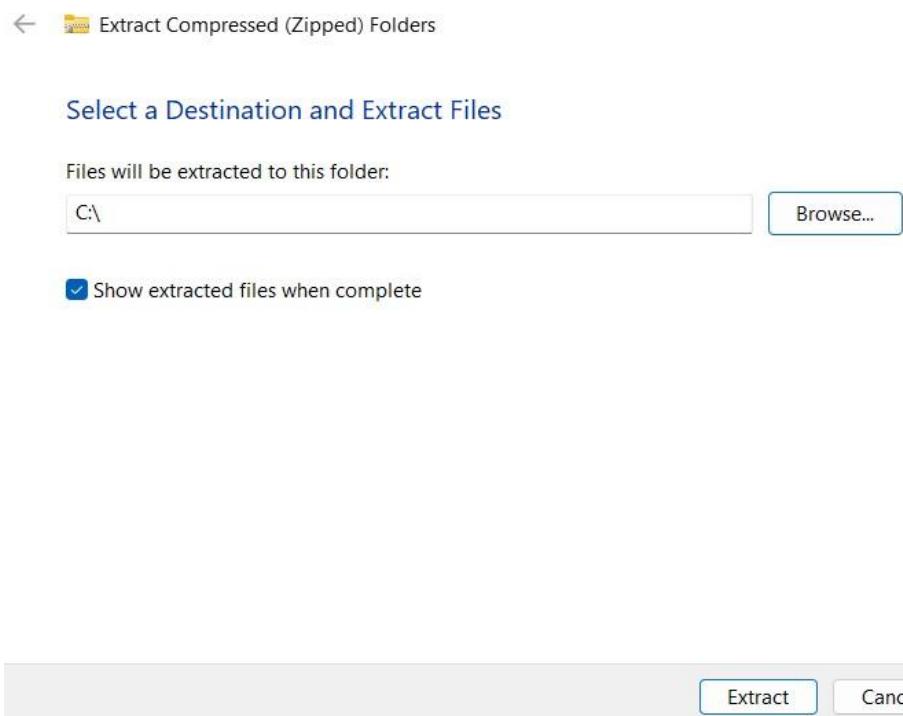
Step 2: To download the latest Flutter SDK, click on the Windows icon > Android

The screenshot shows the Flutter documentation website at docs.flutter.dev/get-started/install/windows. The main heading is "Choose your first type of app". Below it, there are three options: "Android Recommended" (selected), "Web", and "Desktop". A note below says: "Your choice informs which parts of Flutter tooling you configure to run your first Flutter app. You can set up additional platforms later. If you don't have a preference, choose [Android](#)". A callout box for "Developing in China" says: "If you want to use Flutter in China, check out using [Flutter in China](#). If you're not developing in China, ignore this notice and follow the other instructions on this page." Another note in Chinese says: "如果你正在中国的网络环境下配置 Flutter, 请参考 [在中国网络环境下使用 Flutter 文档](#)." The left sidebar has a "Get started" section with various links like "Set up Flutter", "Learn Flutter", etc.

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

The screenshot shows the Flutter documentation website at docs.flutter.dev/get-started/install/windows/mobile. The main heading is "Download then install Flutter". It says: "To install Flutter, download the Flutter SDK bundle from its archive, move the bundle to where you want it stored, then extract the SDK." Step 1: "Download the following installation bundle to get the latest stable release of the Flutter SDK." A blue button labeled "flutter_windows_3.27.3-stable.zip" is highlighted. Step 2: "Create a folder where you can install Flutter." It says: "Consider creating a directory at %USERPROFILE% (C:\Users\{username}) or %LOCALAPPDATA% (C:\Users\{username}\AppData\Local)". On the right, there's a "Contents" sidebar with links like "Verify system requirements", "Hardware requirements", "Software requirements", etc. The left sidebar has a "Get started" section with various links like "Set up Flutter", "Learn Flutter", etc.

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

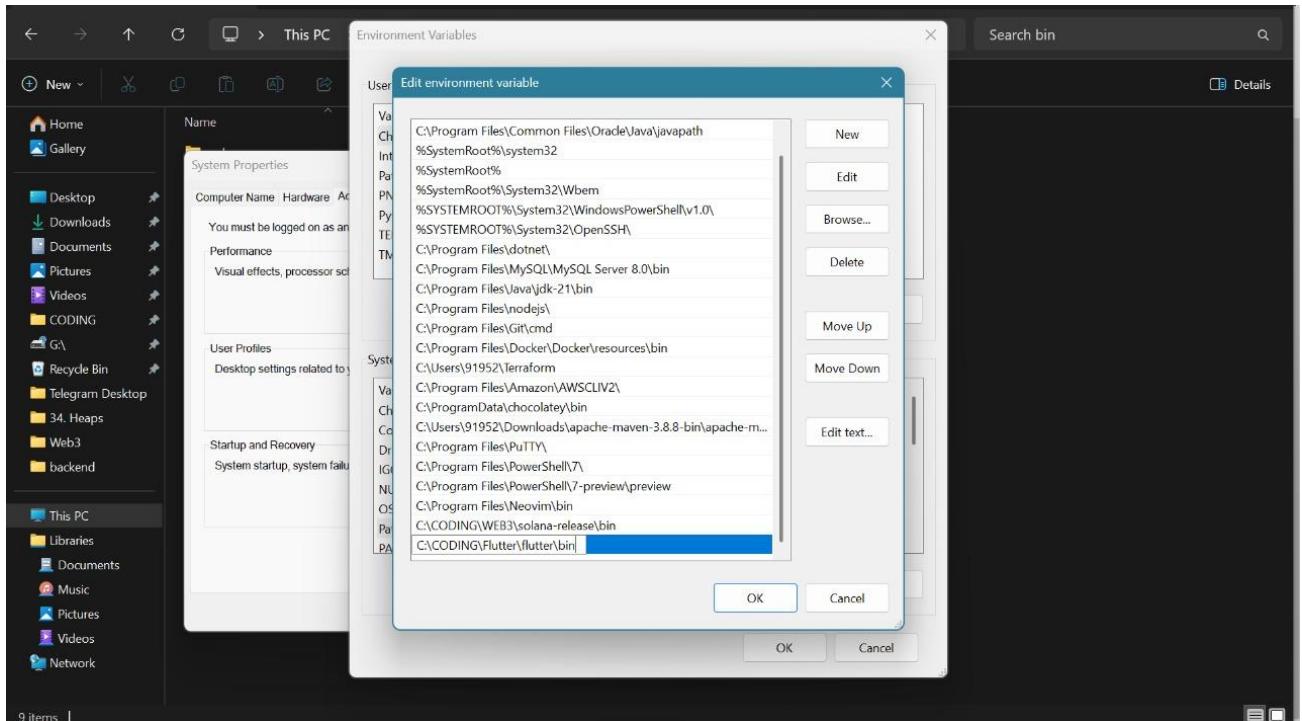


Step 5 :- Add Flutter to System PATH

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.

```

PowerShell 7.4.6

A new PowerShell stable release is available: v7.5.0
Upgrade now, or check out the release page at:
https://aka.ms/PowerShell-Release?tag=v7.5.0

PS C:\Users\91952> flutter
Manage your Flutter app development.

Common commands:

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

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

Usage: flutter <command> [arguments]

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

Available commands:

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

  News for you
  China's DeepSee...
  Search
  File Explorer
  Task View
  Edge
  Google Chrome
  File Manager
  Notepad
  Paint
  Snipping Tool
  Calculator
  Mail
  YouTube
  WhatsApp
  Microsoft Store
  Power Shell
  09:16 AM
  ENG IN
  28-01-2025

```

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

```

PS C:\Users\91952> flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.26100.2894], locale en-IN)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[!] Android toolchain - develop for Android devices (Android SDK version 35.0.1)
    X cmdline-tools component is missing
        Run 'path/to/sdkmanager --install "cmdline-tools;latest"'
        See https://developer.android.com/studio/command-line for more details.
    X Android license status unknown.
        Run 'flutter doctor --android-licenses' to accept the SDK licenses.
        See https://flutter.dev/to/windows-android-setup for more details.
[✓] Chrome - develop for the web
[!] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.11.1)
    X Visual Studio is missing necessary components. Please re-run the Visual Studio installer for the "Desktop development with C++" workload, and include these components:
        MSVC v142 - VS 2019 C++ x64/x86 build tools
            - If there are multiple build tool versions available, install the latest
        C++ CMake tools for Windows
        Windows 10 SDK
[✓] Android Studio (version 2024.2)
[✓] IntelliJ IDEA Community Edition (version 2023.2)
[✓] VS Code (version 1.96.4)
[✓] Connected device (3 available)
[✓] Network resources

! Doctor found issues in 2 categories.
PS C:\Users\91952>

```



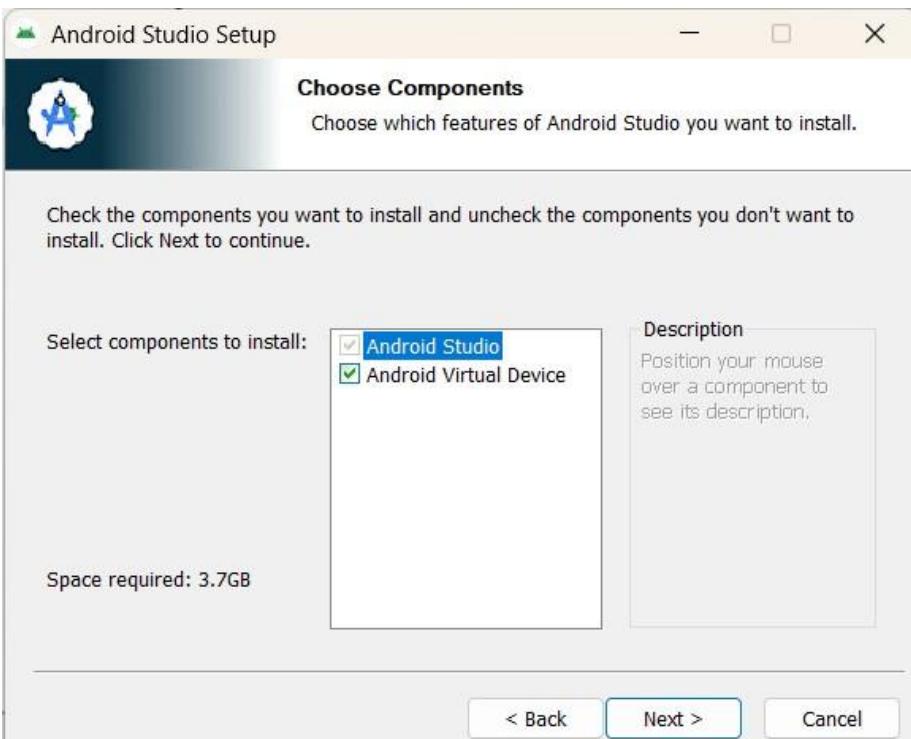
Step 8 : - Go to Android Studio and download the installer.

Platform	Android Studio package	Size	SHA-256 checksum
Windows (64-bit)	android-studio-2024.2.2.13-windows.exe Recommended	1.2 GB	7d93dd9bf3539f948f609b1968507bf502bf6965d2d44bd38a1ff26cb5dd3e
Windows (64-bit)	android-studio-2024.2.2.13-windows.zip No .exe installer	1.2 GB	855945962ff9b84ea49ce39de0bf4189dbf45iae37a6fab7999da013b046b7f7
Mac (64-bit)	android-studio-2024.2.2.13-mac.dmg	1.3 GB	acfbbbe54d6ce8cf21f19b43510c7addcb9dde2824282f205fd1331be77d2e613
Mac (64-bit, ARM)	android-studio-2024.2.2.13-mac_arm.dmg	1.3 GB	688f8d007e612f3f0c18f316179079dc4565f93d8d1e6a7dad80c4fce356df7
Linux (64-bit)	android-studio-2024.2.2.13-linux.tar.gz	1.3 GB	b7fe1ed4a7959bdaca7a8fd57461dbbf9a205eb23cc218ed828ed88e8b998cb5

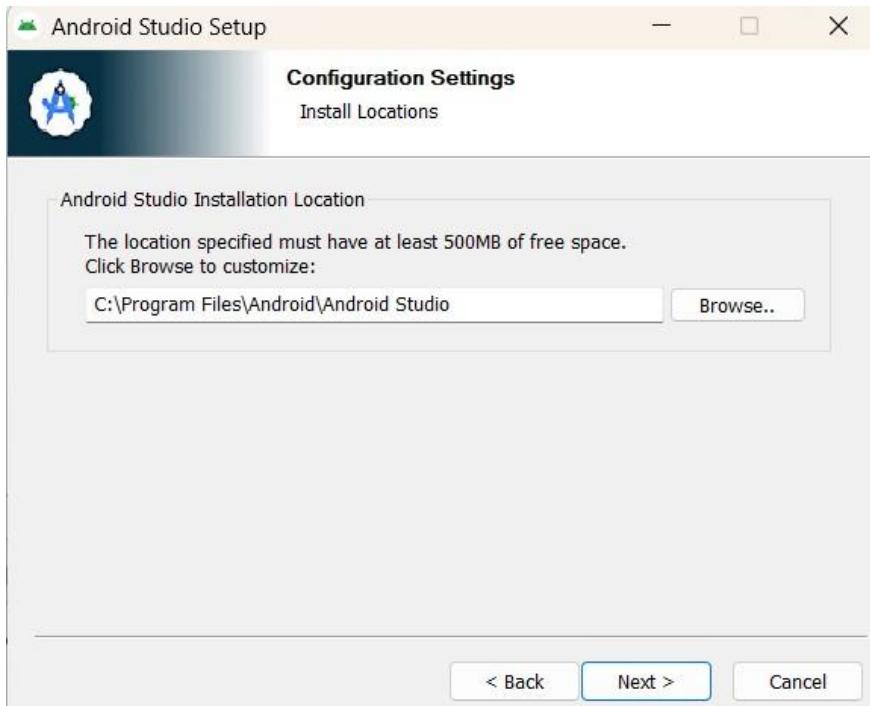
Step 8.1: - When the download is complete, open the .exe file and run it. You will get the following dialog box



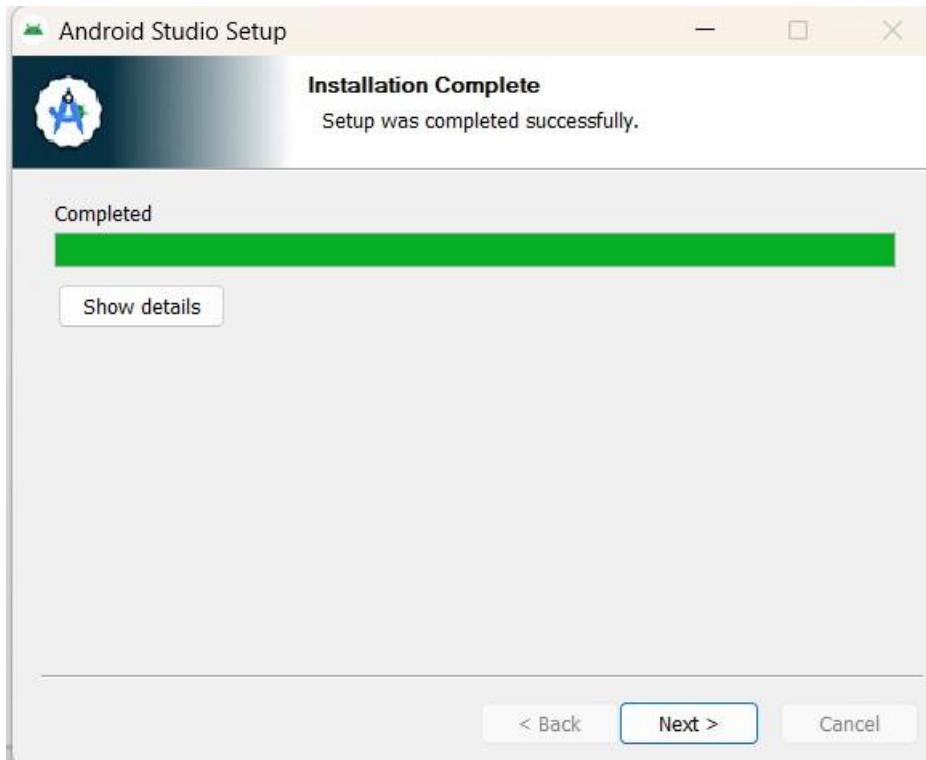
Step 8.2: - Select all the Checkboxes and Click on ‘Next’ Button.

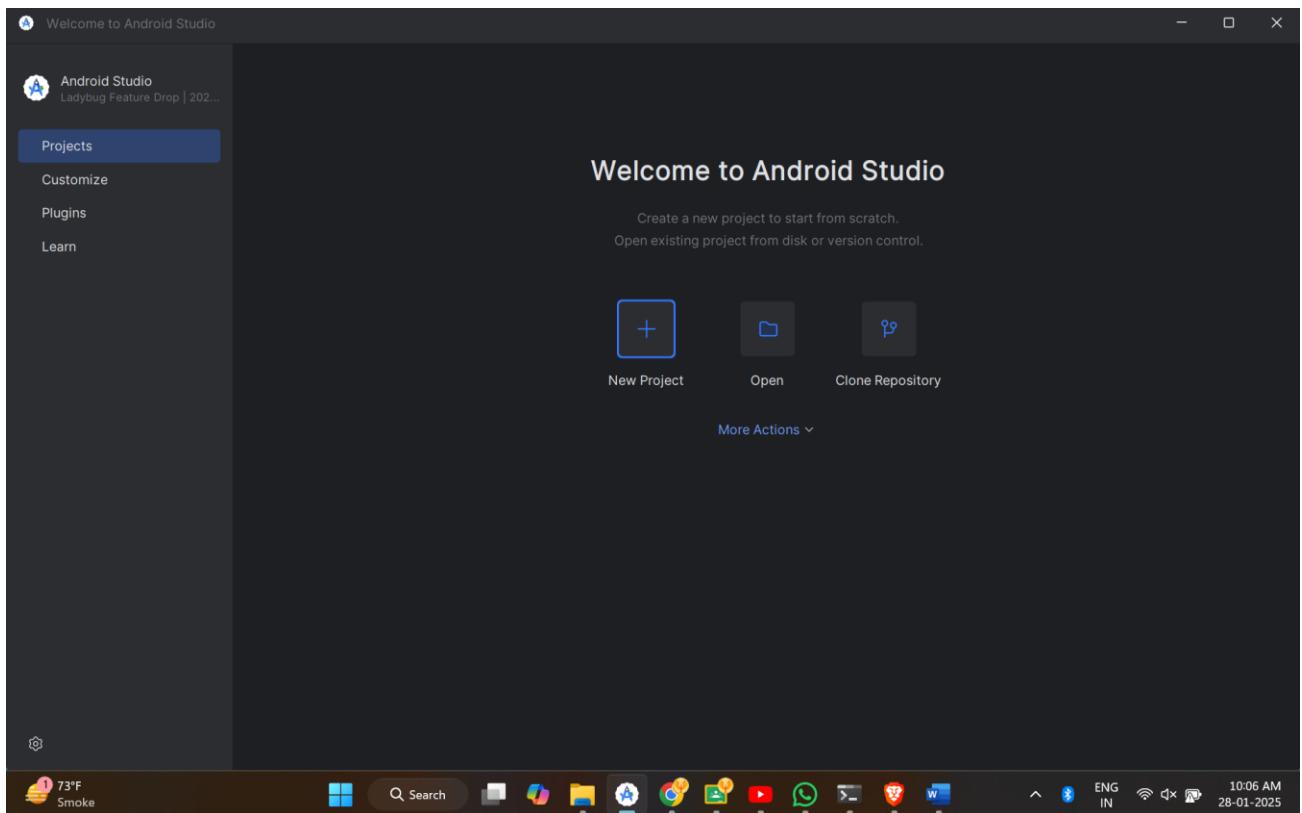


Step 8.3: - Change the destination as per your convenience and click on ‘Next’ Button.



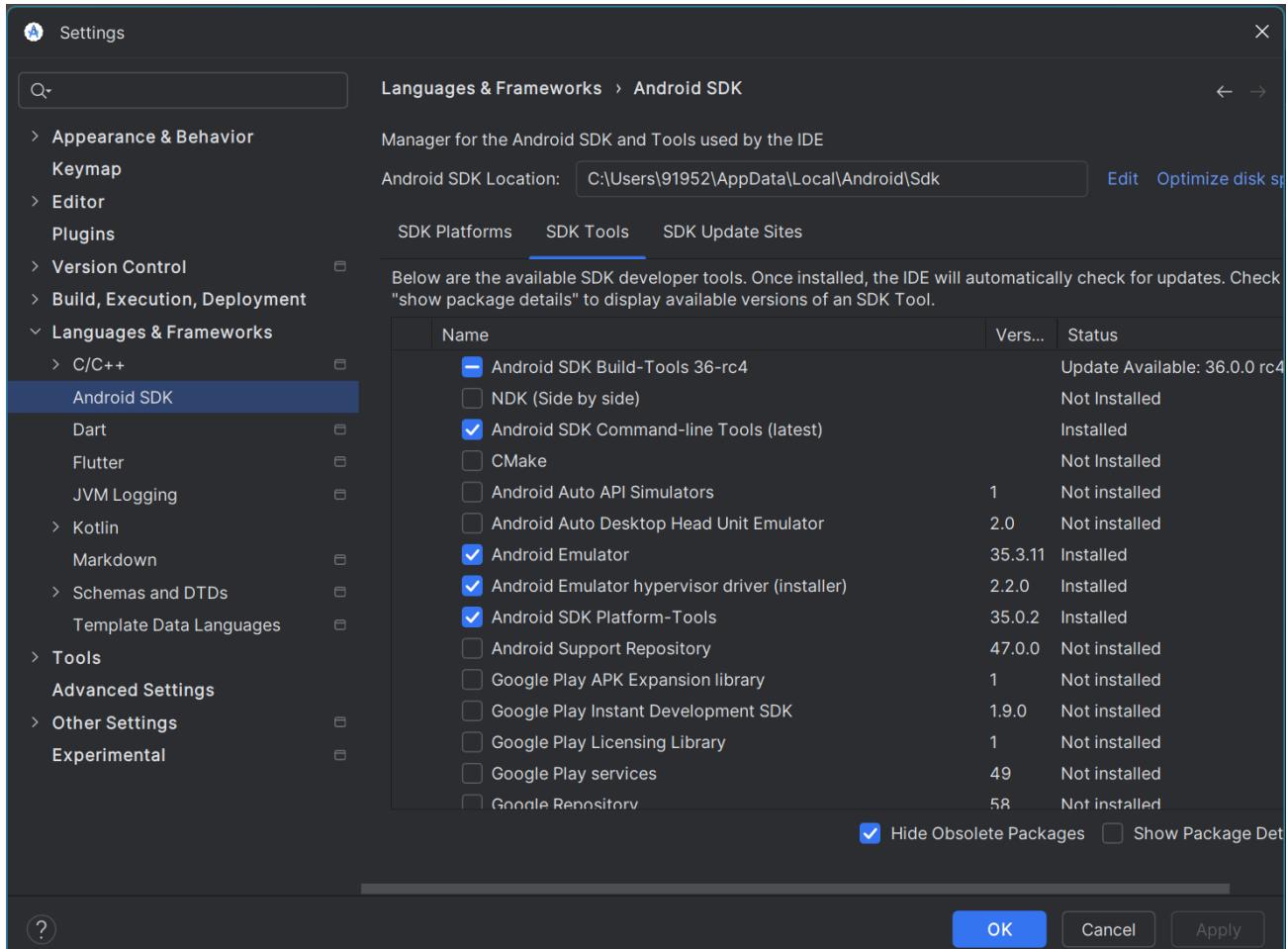
Step 8.4: - Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.





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



```

PowerShell 7 (x64) + X - X

! Doctor found issues in 2 categories.
PS C:\Users\91952> flutter doctor --android-licenses
Warning: Errors during XML parse:
Warning: Additionally, the fallback loader failed to parse the XML.ry...
Warning: Errors during XML parse:           ] 62% Fetch remote repository...
                                                Warning: Additionally, the fallback loader failed to
parse the XML.
[=====] 100% Computing updates...
6 of 7 SDK package licenses not accepted.
Review licenses that have not been accepted (y/N)? y

1/6: License android-googletv-license:
-----
Terms and Conditions

This is the Google TV Add-on for the Android Software Development Kit License Agreement.

1. Introduction

1.1 The Google TV Add-on for the Android Software Development Kit (referred to in this License Agreement as the "Google TV Add-on" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of this License Agreement. This License Agreement forms a legally binding contract between you and Google in relation to your use of the Google TV Add-on.

1.2 "Google" means Google Inc., a Delaware corporation with principal place of business at 1600 Amphitheatre Parkway, Mountain View, CA 94043, United States.

2. Accepting this License Agreement

2.1 In order to use the Google TV Add-on, you must first agree to this License Agreement. You may not use the Google TV Add-on if you do not accept this License Agreement.

2.2 You can accept this License Agreement by:

(A) clicking to accept or agree to this License Agreement, where this option is made available to you; or

```



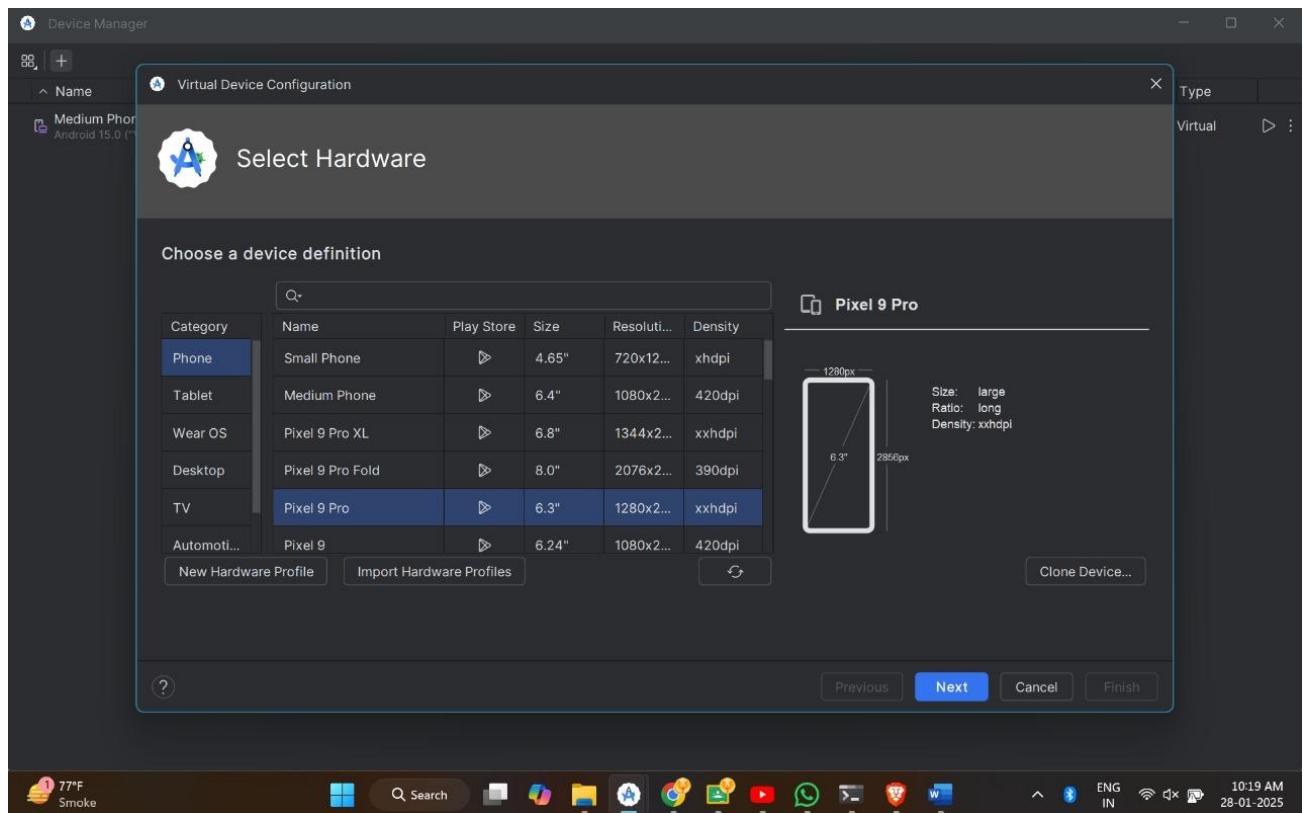
```

PS C:\Users\91952> flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.26100.2894], locale en-IN)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 35.0.1)
[✓] Chrome - develop for the web
[!] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.11.1)
  X Visual Studio is missing necessary components. Please re-run the Visual Studio installer for the "Desktop development with C++"
  workload, and include these components:
    MSVC v142 - VS 2019 C++ x64/x86 build tools
      - If there are multiple build tool versions available, install the latest
        C++ CMake tools for Windows
        Windows 10 SDK
[✓] Android Studio (version 2024.2)
[✓] IntelliJ IDEA Community Edition (version 2023.2)
[✓] VS Code (version 1.96.4)
[✓] Connected device (3 available)
[✓] Network resources

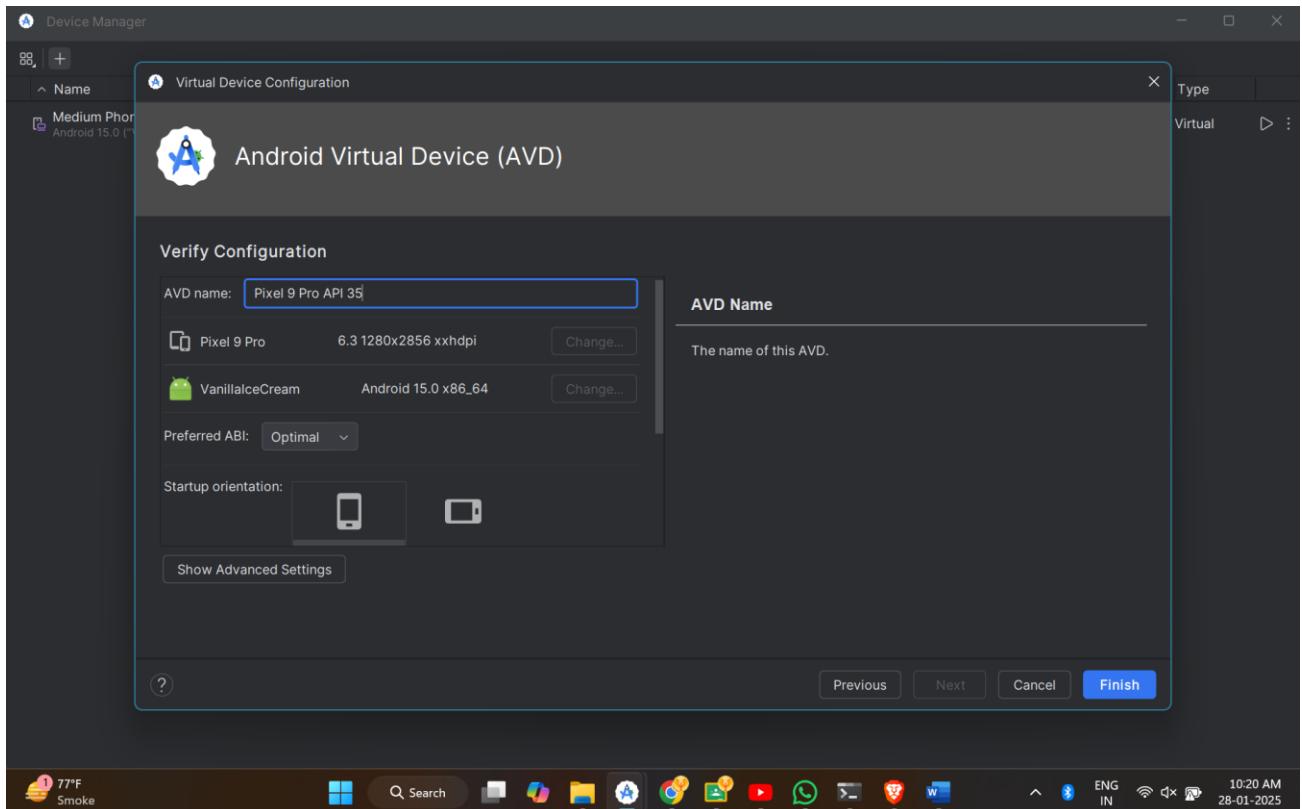
! Doctor found issues in 1 category.
PS C:\Users\91952> |

```

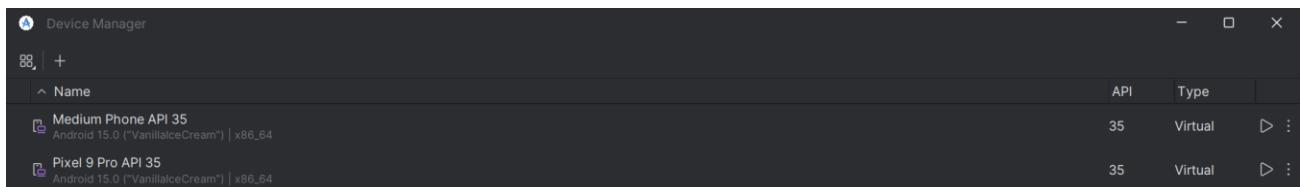
Step 10: - Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application

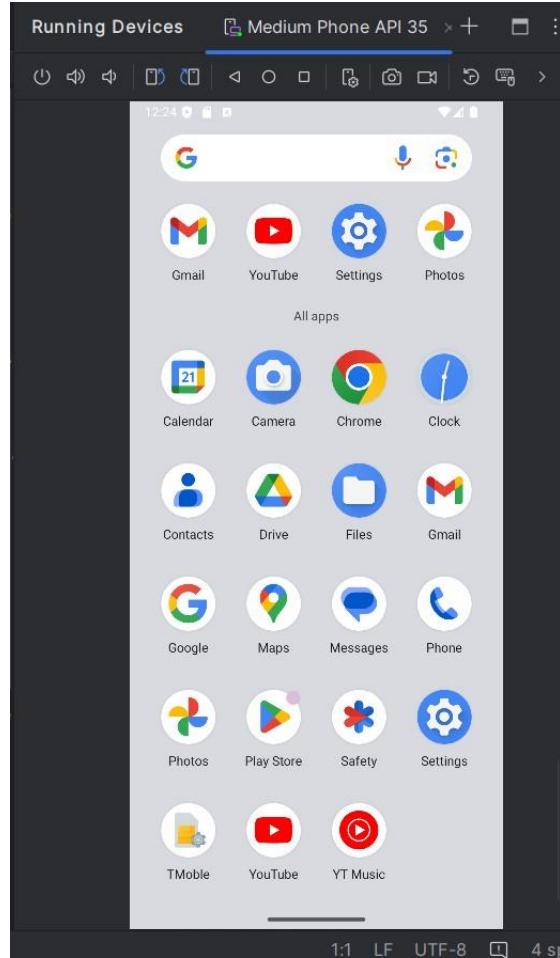


Step 10.1: - Open Android Studio and go to Tools > AVD Manager. Create a new virtual device.



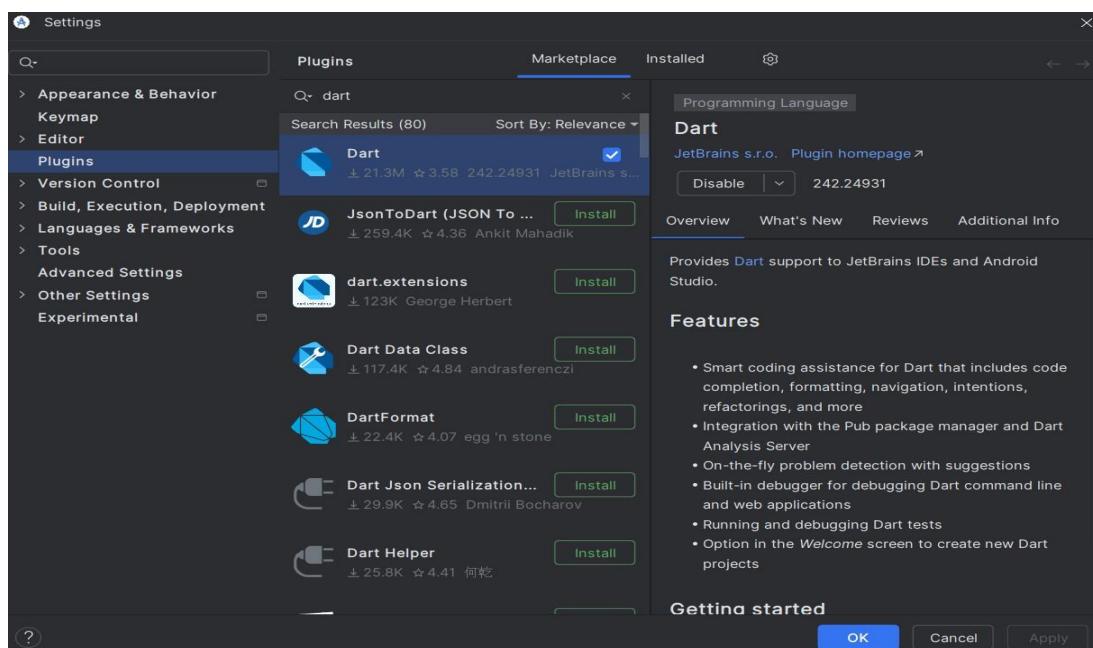
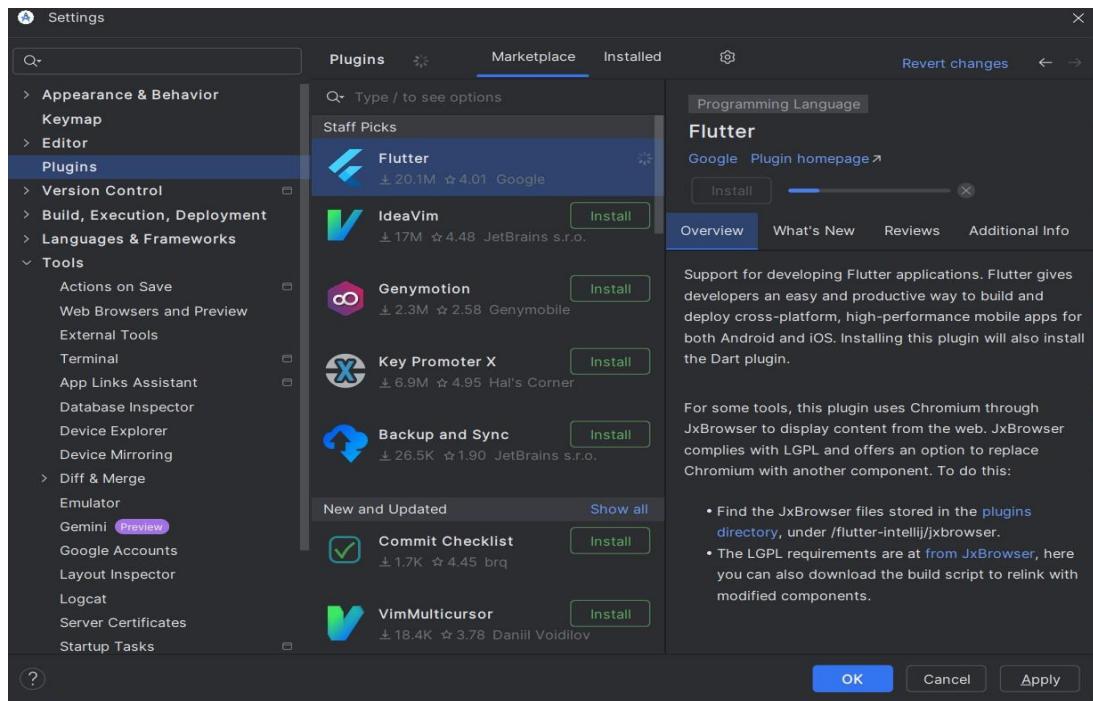
Step 10.2: - Click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen





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.

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Professional UI',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        textTheme: const TextTheme(
          headlineMedium: TextStyle(fontWeight: FontWeight.bold, color: Colors.deepPurple,
          fontSize: 24),
          bodyLarge: TextStyle(color: Colors.black87, fontSize: 18),
        ),
        useMaterial3: true,
      ),
      home: const MyHomePage(title: 'Welcome Prajyot'),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.deepPurple,
        title: Text(widget.title, style: const TextStyle(color: Colors.white)),
        centerTitle: true,
    
```

```
),
body: Center(
  child: Container(
    padding: const EdgeInsets.all(16.0),
    decoration: BoxDecoration(
      borderRadius: BorderRadius.circular(12.0),
      color: Colors.deepPurple.shade50,
      boxShadow: [
        BoxShadow(
          color: Colors.deepPurple.withOpacity(0.2),
          blurRadius: 10,
          spreadRadius: 1,
          offset: const Offset(0, 5),
        )
      ],
    ),
    child: const Text(
      'Hello Prajyot',
      style: TextStyle(
        fontSize: 28,
        fontWeight: FontWeight.bold,
        color: Colors.deepPurple
      ),
    ),
  ),
),
),
),
),
);
}
```

The screenshot shows a development environment for a Flutter application. The code editor displays `main.dart` with the following content:

```
45 child: Container(
46   padding: const EdgeInsets.all(16.0),
47   decoration: BoxDecoration(
48     borderRadius: BorderRadius.circular(12.0),
49     color: Colors.deepPurple.shade50,
50     boxShadow: [
51       BoxShadow(
52         color: Colors.deepPurple.withOpacity(0.2),
53         blurRadius: 10,
54         spreadRadius: 1,
55         offset: const Offset(0, 5),
56       ) // BoxShadow
57     ],
58   ), // BoxDecoration
59   child: const Text(
60     'Hello Prajyot',
61     style: TextStyle(
62       fontSize: 28,
63       fontWeight: FontWeight.bold,
64       color: Colors.deepPurple
65     ), // TextStyle
66   ), // Text
67 ), // Container
68 ), // Center
69 ); // Scaffold
70 }
71 }
```

The running device in the top right corner displays the text "Hello Prajyot" in a purple box.

MAD & PWA Lab

Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	55
Name	Prajyot Shinde
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

EXPERIMENT NO: - 02

Name:- Prajyot Shinde

Class:- D15A

Roll:No: - 55

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 many single child widgets to make the app UI attractive. If we use these widgets appropriately, it can save our time and makes 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 Widgetss

➤ **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 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';                                // Banner Slider at the top
import './widgets/service_category.dart';
import './widgets/banner_slider.dart';
const BannerSlider(),


import 'package:flutter/material.dart';
import './widgets/service_category.dart';
import './widgets/banner_slider.dart';
class HomeScreen extends StatefulWidget {
  const HomeScreen({super.key});
  @override
  State<HomeScreen> createState() =>
      _HomeScreenState();
}
class _HomeScreenState extends State<HomeScreen> {
  final TextEditingController _searchController =
  TextEditingController();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Urban Company'),
        backgroundColor: Colors.transparent,
        elevation: 0,
      ),
      body: LayoutBuilder(
        builder: (context, constraints) {
          return SingleChildScrollView(
            child: ConstrainedBox(
              constraints: BoxConstraints(minHeight:
constraints.maxHeight),
              child: Column(
                mainAxisAlignment:
CrossAxisAlignment.start,
                children: [

```

- // Search bar
- Padding(
- padding: const EdgeInsets.all(16.0),
- child: TextField(
- controller: _searchController,
- decoration: InputDecoration(
- hintText: 'Search for services...',
- prefixIcon: const Icon(Icons.search),
- suffixIcon: IconButton(
- icon: const Icon(Icons.clear),
- onPressed: () {
- _searchController.clear();
- },
-),
- border: OutlineInputBorder(
- borderRadius: BorderRadius.circular(12),
-),
-),
-),
-),
- // Services heading
- const Padding(
- padding: EdgeInsets.all(16.0),
- child: Text(
- 'Services',
- style: TextStyle(
- fontSize: 20,
- fontWeight: FontWeight.bold,
-),

```

        ),
        Navigator.pushNamed(
        ),
        context,
        '/service-details',
        arguments: 'MSaloon',
    );
}

// Grid of service categories
Padding(
    padding: const EdgeInsets.symmetric(horizontal: 16),
    child: GridView.count(
        shrinkWrap: true,
        physics: const NeverScrollableScrollPhysics(),
        crossAxisCount: 2,
        padding: const EdgeInsets.all(8),
        mainAxisSpacing: 16,
        crossAxisSpacing: 16,
        children: [
            ServiceCategory(
                title: "Women's Salon",
                image: 'assets/images/categories/womens_salon.png',
                onTapCallback: () {
                    Navigator.pushNamed(
                        context,
                        '/service-details',
                        arguments: 'WSaloon',
                    );
                },
            ),
            ServiceCategory(
                title: "Men's Salon",
                image: 'assets/images/categories/mens_salon.png',
                onTapCallback: () {
                    Navigator.pushNamed(
                        context,
                        '/service-details',
                        arguments: 'MSaloon',
                    );
                },
            ),
            ServiceCategory(
                title: 'AC & Appliance Repair',
                image: 'assets/images/categories/ac_appliance.png',
                onTapCallback: () {
                    Navigator.pushNamed(
                        context,
                        '/service-details',
                        arguments: 'Repair',
                    );
                },
            ),
            ServiceCategory(
                title: 'Cleaning',
                image: 'assets/images/categories/cleaning.png',
                onTapCallback: () {
                    Navigator.pushNamed(
                        context,
                        '/service-details',
                        arguments: 'Cleaning',
                    );
                },
            ),
        ],
    );
}

```



```

width:                      required this.image,
MediaQuery.of(context).size.width, // Full      this.onTapCallback,
width of the device           }) : super(key: key);

margin:        const
EdgeInsets.symmetric(horizontal: 5.0),          @override
decoration: BoxDecoration(Widget build(BuildContext context) {
borderRadius: BorderRadius.circular(10),          return GestureDetector(
image: DecorationImage(            onTap: onTapCallback,
image: AssetImage(imagePath),            child: ClipRRect(
fit: BoxFit                    borderRadius: BorderRadius.circular(12),
.cover, // This will ensure the image      child: Container(
covers the entire container            color: Colors.grey[200],
),                                           child: Column(
),                                         mainAxisAlignment:
),                                           MainAxisAlignment.center,
);                                           children: [
},                                           Expanded(
),                                           child: Image.asset(
);                                           image,
);                                           fit: BoxFit.cover,
},                                           ),
),                                           const SizedBox(height: 8),
);                                           Padding(
},                                           padding: const EdgeInsets.all(8.0),
);                                           child: Text(
},                                           title,
);                                           textAlign: TextAlign.center,
);                                           style: const TextStyle(
},                                           fontSize: 14,
);                                           fontWeight: FontWeight.bold,
);
}

```

Service_category.dart:

```

import 'package:flutter/material.dart';

class ServiceCategory extends StatelessWidget {
final String title;
final String image;
final VoidCallback? onTapCallback;

const ServiceCategory({
Key? key,
required this.title,

```

```

        overflow: TextOverflow.ellipsis,
),
),
],
),
),
),
),
),
),
);
}
}
},
{
'title': 'Women Salon & Beauty',
'category': 'WSaloon',
'duration': '60 mins',
'price': '₹799',
'rating': '4.9',
'reviews': '2.1K',
'image':
'assets/images/services/msaloon_banner.jpeg',
),
{
'title': 'AC Repair & Maintenance',
'category': 'Repair',
'duration': '90 mins',
'price': '₹999',
'rating': '4.7',
'reviews': '3.5K',
'image':
'assets/images/services/wsaloon_banner.jpeg',
),
{
'title': 'House Cleaning Services',
'category': 'Cleaning',
'duration': '120 mins',
'price': '₹1499',
'rating': '4.6',
'reviews': '2.8K',
'image':
'assets/images/services/cleaning_banner.jpeg',
),
{

```

Service_details_screen:

```

import 'package:flutter/material.dart';
import '../screens/widgets/service_card.dart';

class ServiceDetailsScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final selectedCategory =
      ModalRoute.of(context)!.settings.arguments as
    String;
    final List<Map<String, String>> services = [
      {
        'title': 'Haircut + Beard grooming',
        'category': 'MSaloon',
        'duration': '45 mins',
        'price': '₹499',
        'rating': '4.8',
        'reviews': '1.2K',
        'image':
        'assets/images/services/repair_banner.jpeg',
      },
      {
        'title': 'House Cleaning Services',
        'category': 'Cleaning',
        'duration': '120 mins',
        'price': '₹1499',
        'rating': '4.6',
        'reviews': '2.8K',
        'image':
        'assets/images/services/cleaning_banner.jpeg',
      },
    ],
  }
}
```

```

{
  'title': 'Relaxing Body Massage',
  'category': 'Massage',
  'duration': '60 mins',
  'price': '₹1299',
  'rating': '4.9',
  'reviews': '1.8K',
  'image': 'assets/images/services/massage.jpeg',
},
];
final filteredServices = services
  .where((service) => service['category'] ==
selectedCategory)
  .toList();

return Scaffold(
  appBar: AppBar(
    title: Text('$selectedCategory Services'),
  ),
  body: SingleChildScrollView(
    child: Column(
      children: [
        Image.asset(
          'assets/images/services/${selectedCategory.t
oLowerCase()}_banner.jpeg',
          width: double.infinity,
          height: 200,
          fit: BoxFit.cover,
        ),
        Padding(
          padding: const EdgeInsets.all(16.0),
        ),
      ],
    ),
  ),
);
}

child: Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
    const Text(
      'Available Services',
      style: TextStyle(
        fontSize: 18,
        fontWeight: FontWeight.bold,
      ),
    ),
    const SizedBox(height: 16),
    Column(
      children: filteredServices.map((service) {
        return ServiceCard(
          title: service['title']!,
          duration: service['duration']!,
          price: service['price']!,
          rating: double.parse(service['rating']!),
          reviews: service['reviews']!,
          image: service['image']!,
        );
      }).toList(),
    ),
  ],
);
}

```

```

}

}

Service_card:

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

class ServiceCard extends StatelessWidget {
  final String title;
  final String duration;
  final String price;
  final double rating;
  final String reviews;
  final String image;

  const ServiceCard({
    required this.title,
    required this.duration,
    required this.price,
    required this.rating,
    required this.reviews,
    required this.image,
  });

  @override
  Widget build(BuildContext context) {
    return Card(
      margin: const EdgeInsets.symmetric(vertical: 8.0),
      shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(12)),
      child: Padding(
        padding: const EdgeInsets.all(12.0),
        child: Row(
          children: [
            ClipRRect(
              borderRadius: BorderRadius.circular(8),
              child: Image.asset(
                image,
                width: 80,
                height: 80,
                fit: BoxFit.cover,
              ),
            ),
            const SizedBox(width: 12),
            Expanded(
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    title,
                    style: const TextStyle(
                      fontSize: 16,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                  const SizedBox(height: 8),
                  Text(duration),
                  const SizedBox(height: 8),
                  Text(
                    price,
                    style: const TextStyle(
                      fontSize: 14,
                    ),
                  ),
                ],
              ),
            ),
          ],
        ),
      ),
    );
  }
}

```

```
fontWeight: FontWeight.bold,  
color: Colors.green,  
),  
,  
const SizedBox(height: 8),  
Row(  
children: [  
const Icon(Icons.star, size: 16, color:  
Colors.orange),  
Text('$rating ($reviews reviews)'),  
],  
,  
[  
,  
],  
),  
ElevatedButton(  
onPressed: () {  
Navigator.push(  
context,  
MaterialPageRoute(builder: (context) =>  
SummaryPage()),  
);  
},  
style: ElevatedButton.styleFrom(  
backgroundColor: Colors.purple,  
shape: RoundedRectangleBorder(  
borderRadius: BorderRadius.circular(8),  
),  
child: const Text('Add'),  
),  
],  
),  
});  
}  
};
```

OUTPUT: -

1:02 • LTE 5G+ 100%

Urban Company

Wedding special

Elevate your wedding glow

Salon for women

Book now

• • •

✨ Handyman Services

Quick fixes for your home!

Search for services...

All Services

Women's Salon  ★ 4.5

Men's Salon  ★ 4.5

Home Bookings Get Help Profile

1:05 • LTE 5G+ 100%

MSaloon Services



Available Services

Haircut + Beard grooming
45 mins ₹499 ★4.8 (1.2K reviews)

Add

The screenshot shows a mobile application interface for managing bookings and plans. At the top, there is a status bar with icons for signal strength, battery level (100%), and connectivity (LTE, 5G+). Below the status bar, the title "Bookings & Plans" is displayed with a back arrow icon. The interface is divided into two tabs: "UPCOMING" (selected) and "HISTORY".

UPCOMING Bookings:

- Picture Hanging**
Date: 2025-02-26 09:38:00.000 >
Price: ₹200
- Picture Hanging**
Date: 2025-02-27 05:51:00.000 >
Price: ₹200

HISTORY Bookings:

- Furniture Assembly**
Date: 2025-02-26 01:10:00.000 >
Price: ₹300
- Furniture Assembly**
Date: 2025-02-26 04:32:00.000 >
Price: ₹300

MAD & PWA Lab

Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	55
Name	Prajyot Shinde
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

EXPERIMENT NO: - 03

Name:- Prajyot Shinde

Class:- D15A

Roll:No: - 55

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:**Booking_screen.dart**

```

import 'package:flutter/material.dart';

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

  @override
  Widget build(BuildContext context) {
    return DefaultTabController(
      length: 2,
      child: Scaffold(
        appBar: AppBar(
          title: const Text('Bookings & plans'),
          bottom: const TabBar(
            tabs: [
              Tab(text: 'UPCOMING'),
              Tab(text: 'HISTORY'),
            ],
            labelColor: Colors.purple,
            unselectedLabelColor: Colors.grey,
          ),
        ),
        body: TabBarView(
          children: [
            _buildUpcomingBookings(),
            _buildBookingHistory(),
          ],
        ),
      ),
    );
  }
}

```

),

);

}

Widget _buildUpcomingBookings() {

return ListView.builder(

itemCount: 2,

itemBuilder: (context, index) {

return Card(

margin: const EdgeInsets.all(8),

child: ListTile(

title: Text('Booking \${index + 1}'),

subtitle: const Text('Upcoming service details'),

trailing: const Text('₹499'),

),

);

},

);

}

Widget _buildBookingHistory() {

return ListView.builder(

itemCount: 5,

itemBuilder: (context, index) {

return Card(

margin: const EdgeInsets.all(8),

child: ListTile(

title: Text('Past Booking \${index + 1}'),


```

  children: [
    ClipRRect(
      borderRadius: BorderRadius.circular(8),
      child: Image.asset(
        'assets/images/services/men_salon.jpe
g',// Replace with actual image path
        width: 80,
        height: 80,
        fit: BoxFit.cover,
      ),
    ),
    const SizedBox(width: 12),
    Expanded(
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: const [
          Text(
            'Service Title',
            style: TextStyle(
              fontSize: 16,
              fontWeight: FontWeight.bold,
            ),
          ),
          SizedBox(height: 8),
          Text('Duration: 30 mins'),
          SizedBox(height: 8),
          Text(
            '₹500',
            style: TextStyle(
              fontSize: 14,
              fontWeight: FontWeight.bold,
              color: Colors.green,
            ),
          ),
        ],
      ),
    ),
  ],
  const Spacer(),
  ElevatedButton(
    onPressed: () {
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) =>
          BookingSuccessPage()),
      );
    },
    style: ElevatedButton.styleFrom(
      backgroundColor: Colors.green,
      minimumSize: const Size(double.infinity,
      50),
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(8),
      ),
    ),
    child: const Text(
      'Proceed to Payment',
    ),
  ),
);
}

```

```

      style: TextStyle(fontSize: 18),
    ),
  ],
),
),
),
),
);
}

}

```

Booking_success.dart:

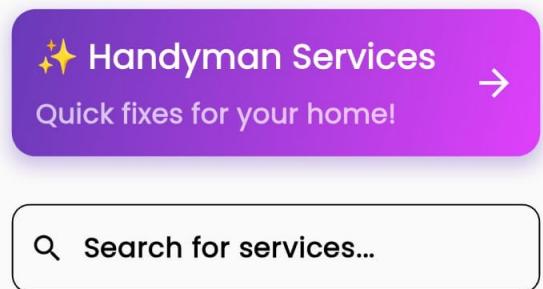
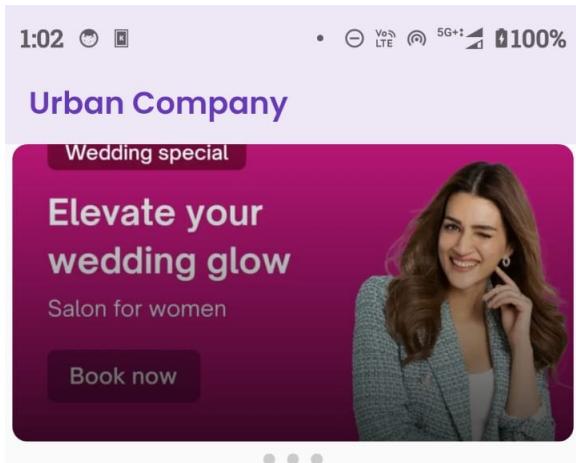
```

      children: [
const Icon(
Icons.check_circle,
color: Colors.green,
size: 100,
),
const SizedBox(height: 16),
const Text(
'Thank you for your booking!',
style: TextStyle(
fontSize: 20,
fontWeight: FontWeight.bold,
),
),
),
const SizedBox(height: 8),
const Text(
'Your payment of ₹500 was successful.',
style: TextStyle(fontSize: 16),
),
),
const SizedBox(height: 16),
ElevatedButton(
 onPressed: () {
Navigator.popUntil(context, (route) =>
route.isFirst);
},
style: ElevatedButton.styleFrom(
backgroundColor: Colors.purple,
minimumSize: const Size(double.infinity,
50),
child: Column(
mainAxisAlignment:
MainAxisAlignment.center,

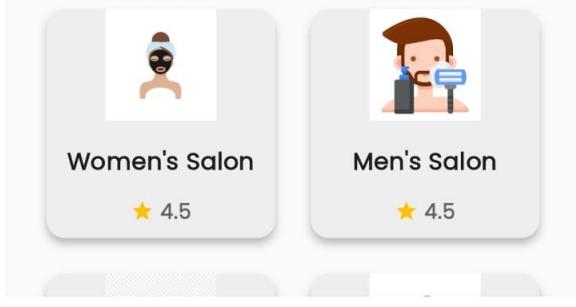
```

```
    ),  
    ),  
    child: const Text('Go to Home'),  
    ),  
    ],  
    ),  
    ),  
    ),  
    ),  
    );  
}  
}
```

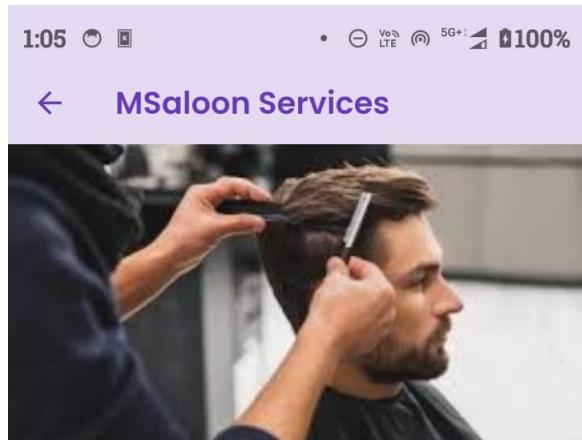
OUTPUT: -



All Services

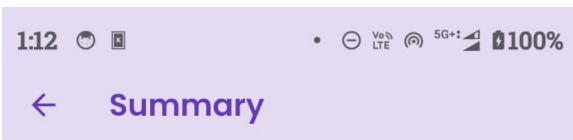


Home Bookings Get Help Profile



Available Services





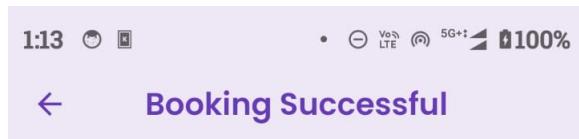
Your Selected Services



House Cleaning Services

Date: 2025-02-25
01:12:00.000

₹1499



**Thank you for your
booking!**

Your payment will be collected at the
time of service.

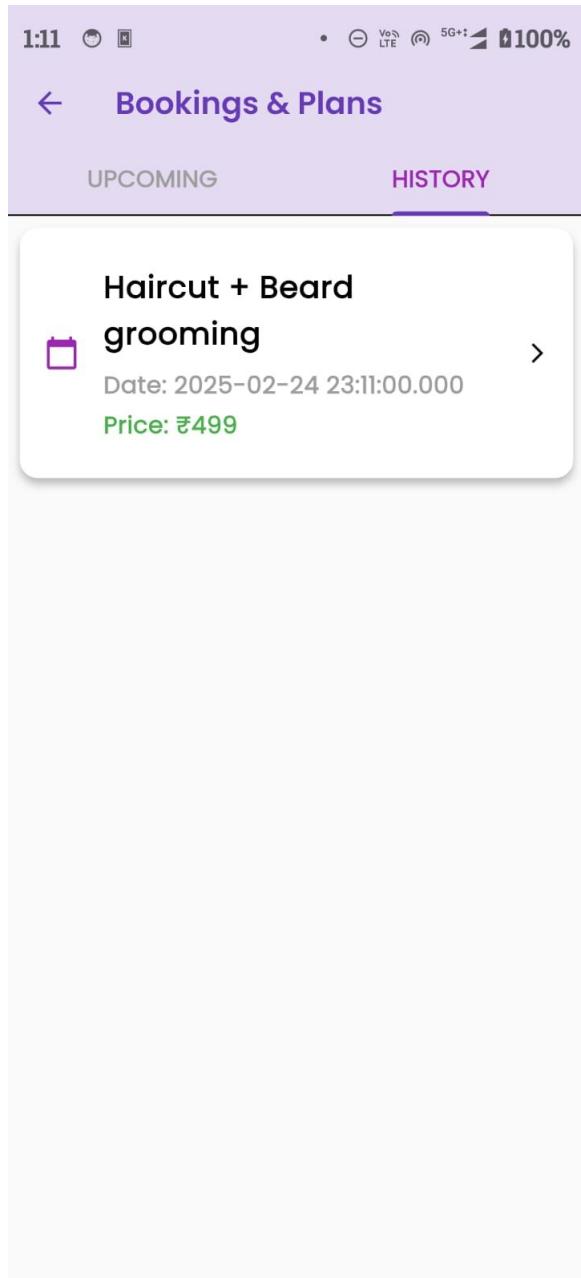
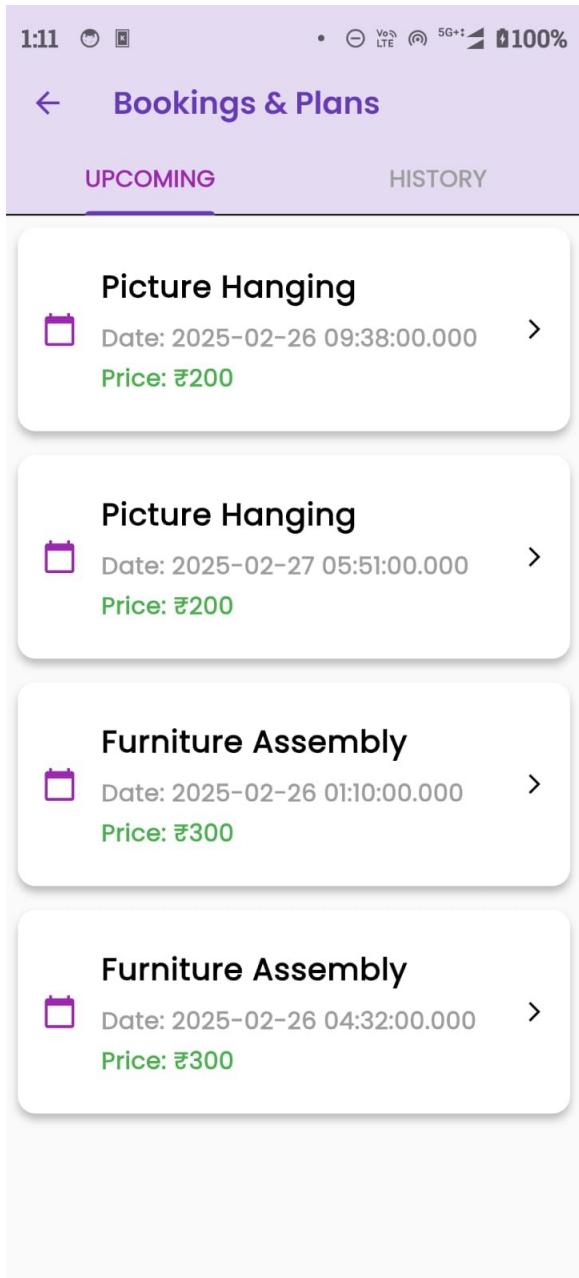
Payment Method

Cash on Delivery (COD)

UPI/Payment Options Coming Soon!

Go to Home

Proceed to Payment



MAD & PWA Lab

Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	55
Name	Prajyot Shinde
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

EXPERIMENT NO: - 04

Name:- Prajyot Shinde

Class:- D15A

Roll:No: - 55

AIM: - To create an interactive Form using 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 **TextField** 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 **TextField** 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

- 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.
- A **GlobalKey<FormState>** is required to uniquely identify the form and enable validation or data retrieval from the form fields.
- The **TextFormField** **widget** is used to provide input fields where users can enter data such as names, phone numbers, or email addresses.
- To enhance the appearance and usability of input fields, **InputDecoration** is used, allowing customization of labels, icons, borders, and hint text.
- Validation plays a crucial role in forms, and the **validator** **property** within **TextFormField** ensures user input meets specific criteria before submission.
- Different types of input require appropriate **keyboard types**, such as **TextInputType.number** for numeric fields or **TextInputType.emailAddress** for email fields.

- Proper **state management** is needed to store and retrieve user input, ensuring the form data is processed correctly.
- 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 Page**

```

import 'package:flutter/material.dart';

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

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

class _LoginScreenState extends State<LoginScreen> {
  final _formKey = GlobalKey<FormState>();
  final _phoneController =
    TextEditingController(text: '9529567991'); //  

  Dummy number

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Form(
            key: _formKey,
            child: Column(
              crossAxisAlignment:
CrossAxisAlignment.stretch,
              children: [
                const SizedBox(height: 48),
                const Text(
                  'Enter your mobile number',
                  style: TextStyle(
                    fontSize: 24,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                const SizedBox(height: 24),
                TextFormField(
                  controller: _phoneController,
                  keyboardType: TextInputType.phone,
                  decoration: InputDecoration(
                    prefixText: '+91 ',
                    hintText: 'Phone number',
                    border: OutlineInputBorder(
                      borderRadius:
BorderRadius.circular(8),
                    ),
                  ),
                  validator: (value) {
                    if (value == null || value.length != 10) {
                      return 'Please enter a valid phone
number';
                    }
                    return null;
                  },
                ),
                const SizedBox(height: 24),
                ElevatedButton(
                  onPressed: () {
                    if (_formKey.currentState!.validate()) {
                      Navigator.pushReplacementNamed(con
text, '/otp');
                    }
                  },
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
}

```

```

        },
        style: ElevatedButton.styleFrom(
          padding: const
          EdgeInsets.symmetric(vertical: 16),
        ),
        child: const Text('Continue'),
      ),
    ],
  ),
),
),
),
),
),
),
),
),
),
);
}
}

```

Otp verification:

```

import 'package:flutter/material.dart';

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

  @override
  State<OtpVerificationScreen> createState() =>
    _OtpVerificationScreenState();
}

class _OtpVerificationScreenState extends State<OtpVerificationScreen> {
  final List<String> _otpValues = List.filled(4, "");
}

```

```

void _validateOtp() {
  final otp = _otpValues.join();
  if (otp == "9096") {
    Navigator.pushReplacementNamed(context,
    '/home');
  } else {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Invalid OTP')),
    );
  }
}

@Override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Please login to continue'),
      elevation: 0,
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment:
        CrossAxisAlignment.stretch,
        children: [
          const SizedBox(height: 48),
          const Text(
            'Enter verification code',
            style: TextStyle(fontSize: 24, fontWeight:
            FontWeight.bold),
            textAlign: TextAlign.center,
          ),
        ],
      ),
    ),
  );
}

```



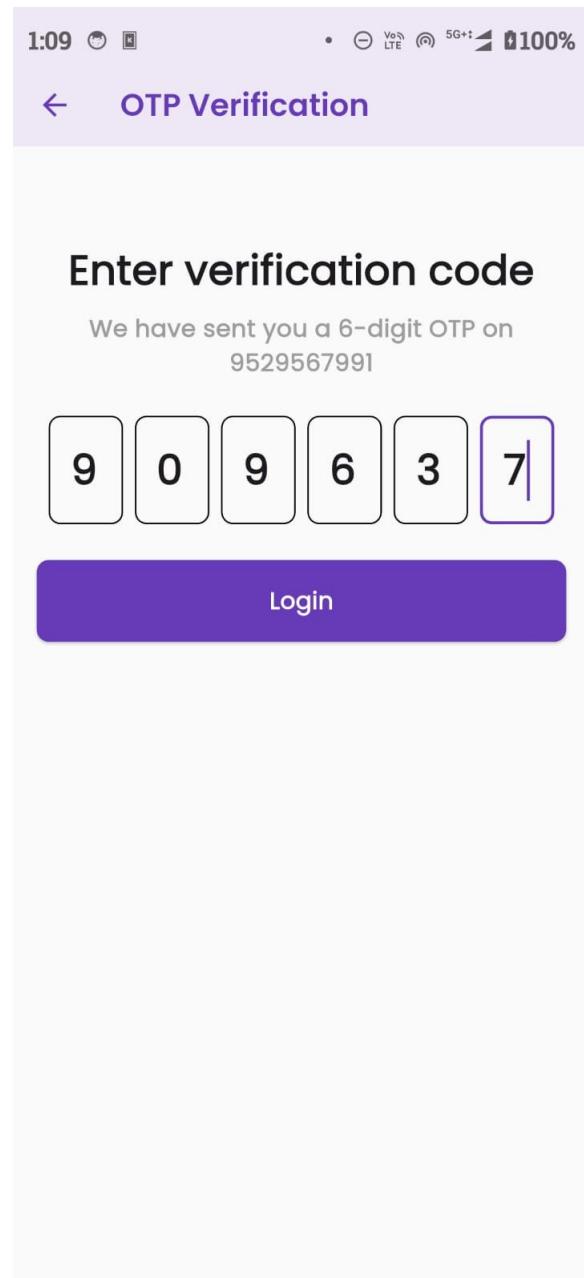
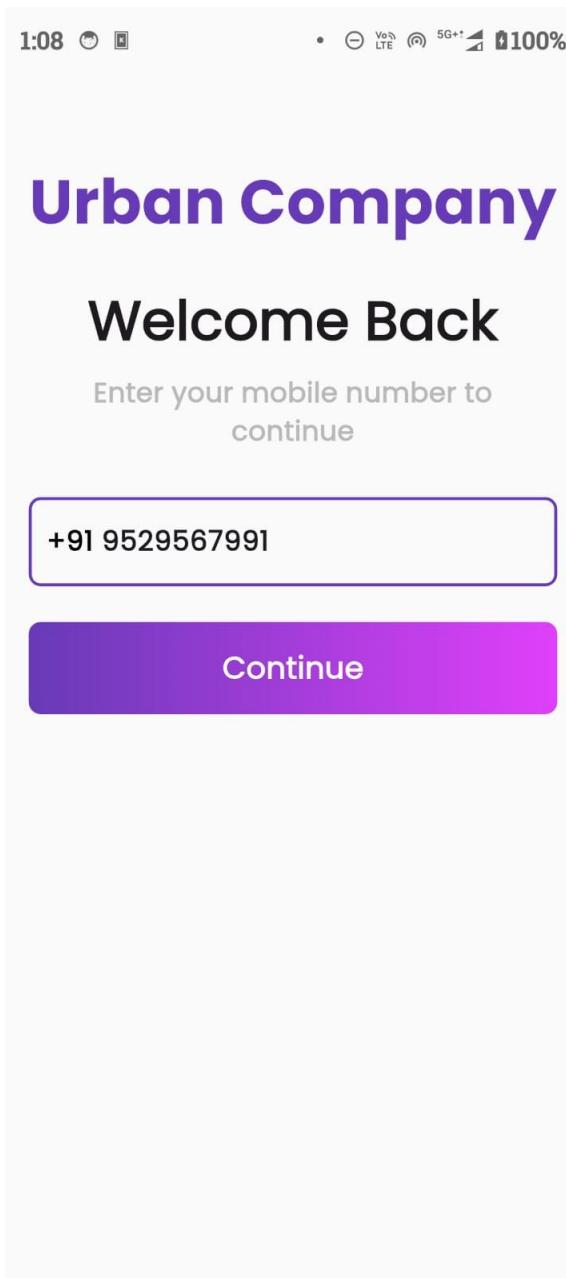
```

Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('My Profile'),
    ),
    body: SingleChildScrollView(
      padding: const EdgeInsets.all(16),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          const CircleAvatar(
            radius: 40,
            child: Icon(Icons.person, size: 40),
          ),
          const SizedBox(height: 16),
          const Text(
            'Personal Information',
            style: TextStyle(
              fontSize: 18,
              fontWeight: FontWeight.bold,
            ),
          ),
          const SizedBox(height: 16),
          TextFormField(
            decoration: const InputDecoration(
              labelText: 'Name',
              border: OutlineInputBorder(),
            ),
            ),
          const SizedBox(height: 16),
          TextFormField(
            decoration: const InputDecoration(
              labelText: 'Email',
              border: OutlineInputBorder(),
            ),
            ),
          const SizedBox(height: 16),
          TextFormField(
            decoration: const InputDecoration(
              labelText: 'Phone',
              border: OutlineInputBorder(),
            ),
            ),
          const SizedBox(height: 24),
          ElevatedButton(
            onPressed: () {},
            style: ElevatedButton.styleFrom(
              minimumSize: const Size.fromHeight(50),
            ),
            child: const Text('Save Changes'),
          ),
          const SizedBox(height: 16),
          TextButton(
            onPressed: () {},
            style: TextButton.styleFrom(
              minimumSize: const Size.fromHeight(50),
              foregroundColor: Colors.red,
            ),
            child: const Text('Logout'),
          ),
        ],
      ),
    ),
  );
}

```

```
 );  
 }  
 }
```

OUTPUT:



1:26 ☺ 🔍 • ⚡ LTE 5G+ 100%

Create Your Account

Please provide the following details:

Full Name

Email Address

Address

Sign Up

1:09 ☺ 🔍 • ⚡ LTE 5G+ 100%

My Profile



Phone: +919529567991

Full Name —

👤 Test @Me

Email —

✉️ test@gmail.com

Address —

From Hell to Heaven



Save Profile

MAD & PWA Lab

Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	55
Name	Prajyot Shinde
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

EXPERIMENT NO: - 05

Name:- Prajyot Shinde

Class:- D15A

Roll:No: - 55

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(),  
            ),  
        );  
    },  
    child: Text("Go to Next Screen"),  
);
```

```

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.

After clicking on Don't have an account? it navigates to the registration page.

Code: -**Main.dart**

```

import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import '/screens/home_screen.dart';
import '/screens/auth/login_screen.dart';
import '/screens/bookings/bookings_screen.dart';
import '/screens/profile/profile_screen.dart';
import '/screens/service_details_screen.dart';
import '/screens/help/help_screen.dart';
import '/screens/auth/otp_verify.dart';
import '/screens/auth/signup_screen.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Urban Company',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        scaffoldBackgroundColor: Colors.white,
        fontFamily: 'Poppins',
        appBarTheme: const AppBarTheme(
          backgroundColor: Colors.white,
          elevation: 0,
          iconTheme: IconThemeData(color: Colors.black),
          titleTextStyle: TextStyle(
            color: Colors.black,
            fontSize: 18,
            fontWeight: FontWeight.w600,
            fontFamily: 'Poppins',
          ),
        ),
        initialRoute: '/',
        routes: {
          '/': (context) => const LoginScreen(),
          '/signup': (context) => const SignupScreen(),
          '/otp': (context) {
            final args =
            ModalRoute.of(context)?.settings.arguments;
            if (args is Map<String, String>) {
              return OtpVerificationScreen(
                phoneNumber: args['phoneNumber'] ?? '',
                verificationId: args['verificationId'] ?? '',
              );
            } else {
              return const Scaffold(
                body: Center(child: Text('Invalid arguments for
OTP screen')),
              );
            }
          },
        },
      ),
    );
  }
}

```

```

'/home': (context) => const HomeScreen(),
'/bookings': (context) => const
BookingsScreen(),
'/help': (context) => const HelpScreen(),
'/profile': (context) => const ProfileScreen(),
'/service-details': (context) =>
ServiceDetailsScreen(),
},
);
}
}

```

Handyman_booking_screen.dart:

```

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

class HandymanBookingScreen extends
StatefulWidget {
final String serviceTitle;
final String price;

const HandymanBookingScreen({
Key? key,
required this.serviceTitle,
required this.price,
}) : super(key: key);

@Override

```

```

State<HandymanBookingScreenState> createState() =>
_HandymanBookingScreenState();
}

class _HandymanBookingScreenState extends
State<HandymanBookingScreenState> {
DateTime? _selectedDate;
TimeOfDay? _selectedTime;

@Override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: const Text(
'Book Handyman',
),
centerTitle: true,
elevation: 0,
),
body: SingleChildScrollView(
padding: const
EdgeInsets.symmetric(horizontal: 16.0, vertical:
24.0), // Adjusted padding
child: Column(
crossAxisAlignment:
CrossAxisAlignment.start,
children: [
// Service Info Card
_buildServiceInfoCard(),
const SizedBox(height: 32),
// Date & Time Picker
_buildDatePicker(context),
const SizedBox(height: 32),

```

```

// Confirm Booking Button
    _buildConfirmButton(context),
],
),
),
);
}

Widget _buildServiceInfoCard() {
return Card(
elevation: 8,
shape: RoundedRectangleBorder(
borderRadius: BorderRadius.circular(16),
),
child: Container(
padding: const EdgeInsets.all(24.0),
decoration: BoxDecoration(
gradient: const LinearGradient(
colors: [Colors.deepPurple,
Colors.purpleAccent],
begin: Alignment.topLeft,
end: Alignment.bottomRight,
),
borderRadius: BorderRadius.circular(16),
),
child: Column(
crossAxisAlignment:
CrossAxisAlignment.start,
children: [
Text(
widget.serviceTitle,
style: const TextStyle(
fontSize: 28,
fontWeight: FontWeight.bold,
color: Colors.white,
),
),
const SizedBox(height: 8),
Text(
widget.price,
style: const TextStyle(
fontSize: 22,
color: Colors.white70,
fontWeight: FontWeight.w600,
),
),
],
),
),
),
);
}

Widget _buildDatePicker(BuildContext
context) {
return Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
const Text(
'Select Date & Time',
style: TextStyle(
fontSize: 22,
fontWeight: FontWeight.bold,
color: Colors.deepPurple,
),
),
),
]
);
}

```

```

),
const SizedBox(height: 16),
Row(
  children: [
    Expanded(
      child: _buildPickerField(
        label: _selectedDate == null
          ? 'Select Date'
          :
          '${_selectedDate!.day}/${_selectedDate!.month}/${_selectedDate!.year}',
        icon: Icons.calendar_today,
        onTap: () async {
          final date = await showDatePicker(
            context: context,
            initialDate: DateTime.now(),
            firstDate: DateTime.now(),
            lastDate: DateTime.now().add(const Duration(days: 365)),
          );
          if (date != null) {
            setState(() => _selectedDate = date);
          }
        },
      ),
    ),
    const SizedBox(width: 16),
    Expanded(
      child: _buildPickerField(
        label: _selectedTime == null
          ? 'Select Time'
          :
          _selectedTime!.format(context),
        icon: Icons.access_time,
        onTap: () async {
          final time = await showTimePicker(
            context: context,
            initialValue: TimeOfDay.now(),
          );
          if (time != null) {
            setState(() => _selectedTime = time);
          }
        },
      ),
    ),
  ],
),
Widget _buildPickerField({
  required String label,
  required IconData icon,
  required VoidCallback onTap,
}) {
  return GestureDetector(
    onTap: onTap,
    child: Container(
      padding: const EdgeInsets.symmetric(vertical: 18, horizontal: 16),
      decoration: BoxDecoration(
        color: Colors.white,
        borderRadius: BorderRadius.circular(12),
        boxShadow: [
          BoxShadow(

```

```

color: Colors.black.withOpacity(0.1),
blurRadius: 10,
spreadRadius: 2,
offset: const Offset(0, 5),
),
],
),
child: Row(
children: [
Icon(icon, color: Colors.deepPurple),
const SizedBox(width: 12),
Expanded( // Added Expanded to prevent
overflow
),
child: Text(
label,
style: const TextStyle(
fontSize: 16,
fontWeight: FontWeight.w500,
),
),
),
),
],
),
),
);
}

Widget _buildConfirmButton(BuildContext context)
{
return SizedBox(
width: double.infinity,
child: ElevatedButton(
onPressed: () => _confirmBooking(context),
style: ElevatedButton.styleFrom(
backgroundColor: Colors.deepPurple,
padding: const EdgeInsets.symmetric(vertical:
20),
shape: RoundedRectangleBorder(
borderRadius: BorderRadius.circular(12),
),
elevation: 4,
),
child: const Text(
'Confirm Booking',
style: TextStyle(
fontSize: 20,
fontWeight: FontWeight.bold,
color: Colors.white,
),
),
),
),
);
}

Future<void> _confirmBooking(BuildContext
context) async {
if (_selectedDate == null || _selectedTime == null)
{
ScaffoldMessenger.of(context).showSnackBar(
const SnackBar(content: Text('Please select date
and time.')),
);
return;
}
}

```

```

final user = FirebaseAuth.instance.currentUser;
        ],
        ),
    );
}

final timestamp = DateTime(
    _selectedDate!.year,
    _selectedDate!.month,
    _selectedDate!.day,
    _selectedTime!.hour,
    _selectedTime!.minute,
);

```

```

await
FirebaseFirestore.instance.collection('bookings').add(
{
    'date': timestamp,
    'price': widget.price,
    'serviceTitle': widget.serviceTitle,
    'status': 'Upcoming',
    'userId': user.uid,
});

```

```

showDialog(
    context: context,
    builder: (context) => AlertDialog(
        title: const Text('Booking Confirmed'),
        content: Text('Your booking for
        "${widget.serviceTitle}" has been confirmed.'),
        actions: [
            TextButton(
                onPressed: () => Navigator.popUntil(context,
                    (route) => route.isFirst),
                child: const Text('OK'),
            ),
        ],
    ),
);

```

Booking_success.dart:

```

import 'package:flutter/material.dart';

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

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: const Text('Booking Successful'),
                centerTitle: true,
                elevation: 0,
                // backgroundColor: Colors.white24,
                // iconTheme: const IconThemeData(color:
                Colors.black),
            ),
            body: Center(
                child: Padding(
                    padding: const EdgeInsets.all(16.0),
                    child: Column(
                        mainAxisAlignment:
MainAxisAlignment.center,

```

```

children: [
    // Success Icon
    const Icon(
        Icons.check_circle,
        color: Colors.green,
        size: 100,
    ),
    const SizedBox(height: 24),
    // Success Message
    const Text(
        'Thank you for your booking!',
        style: TextStyle(
            fontSize: 24,
            fontWeight: FontWeight.bold,
        ),
        textAlign: TextAlign.center,
    ),
    const SizedBox(height: 16),
    // Payment Details
    const Text(
        'Your payment will be collected at the time
        of service.',
        style: TextStyle(
            fontSize: 16,
            color: Colors.grey,
        ),
        textAlign: TextAlign.center,
    ),
    const SizedBox(height: 24),
    // Payment Method
    Card(
        elevation: 4,
        shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(12),
        ),
        child: Padding(
            padding: const EdgeInsets.all(16.0),
            child: Column(
                children: [
                    const Text(
                        'Payment Method',
                        style: TextStyle(
                            fontSize: 18,
                            fontWeight: FontWeight.bold,
                        ),
                    ),
                    const SizedBox(height: 16),
                    Row(
                        mainAxisAlignment: MainAxisAlignment.center,
                        children: const [
                            Icon(Icons.money, color:
                                Colors.green),
                            SizedBox(width: 8),
                            Text(
                                'Cash on Delivery (COD)',
                                style: TextStyle(
                                    fontSize: 16,
                                    fontWeight: FontWeight.w500,
                                ),
                            ),
                        ],
                    ),
                ],
            ),
        ),
    ),
]

```


Output:

1:02 • LTE 5G+ 100%

Urban Company

Wedding special

Elevate your wedding glow

Salon for women

Book now

• • •

Handyman Services →

Quick fixes for your home!

Search for services...

All Services

Women's Salon ★ 4.5

Men's Salon ★ 4.5

Home Bookings Get Help Profile

1:10 • LTE 5G+ 100%

← Book Handyman

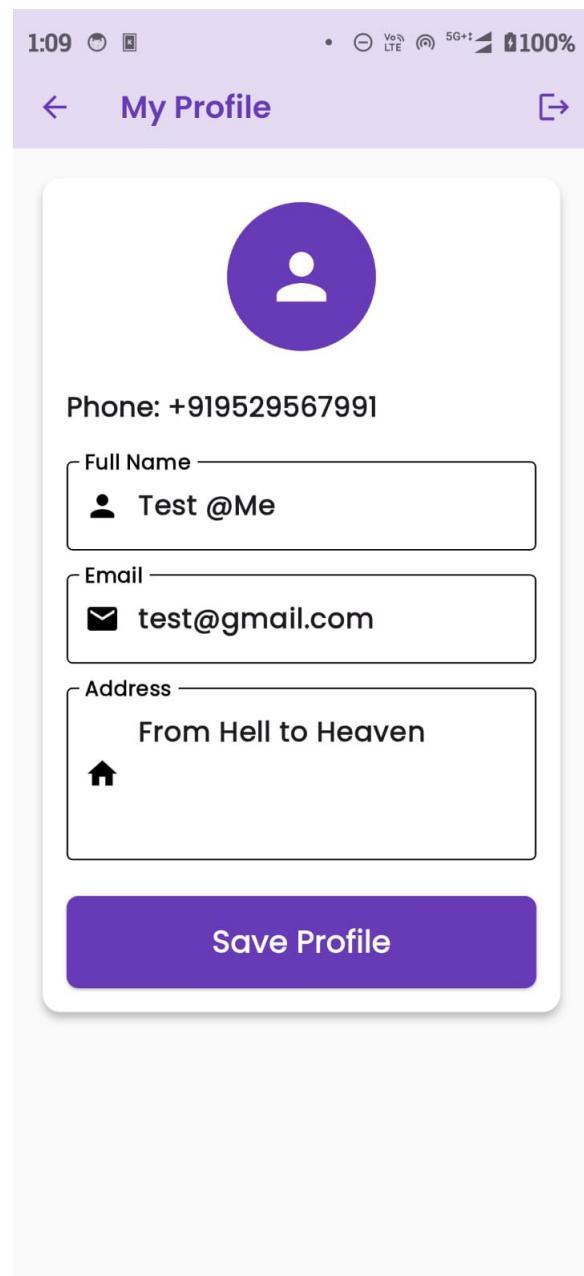
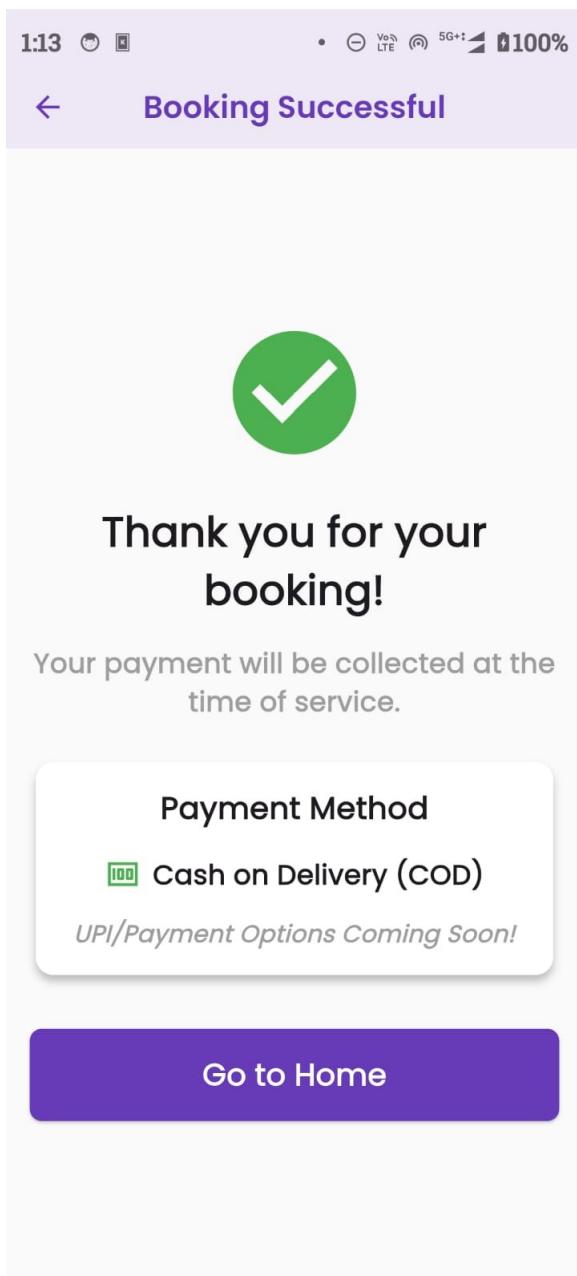
Furniture Assembly

₹300

Select Date & Time

26/2/2025 1:10 AM

Confirm Booking



MAD & PWA Lab

Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	55
Name	Prajyot Shinde
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:	14

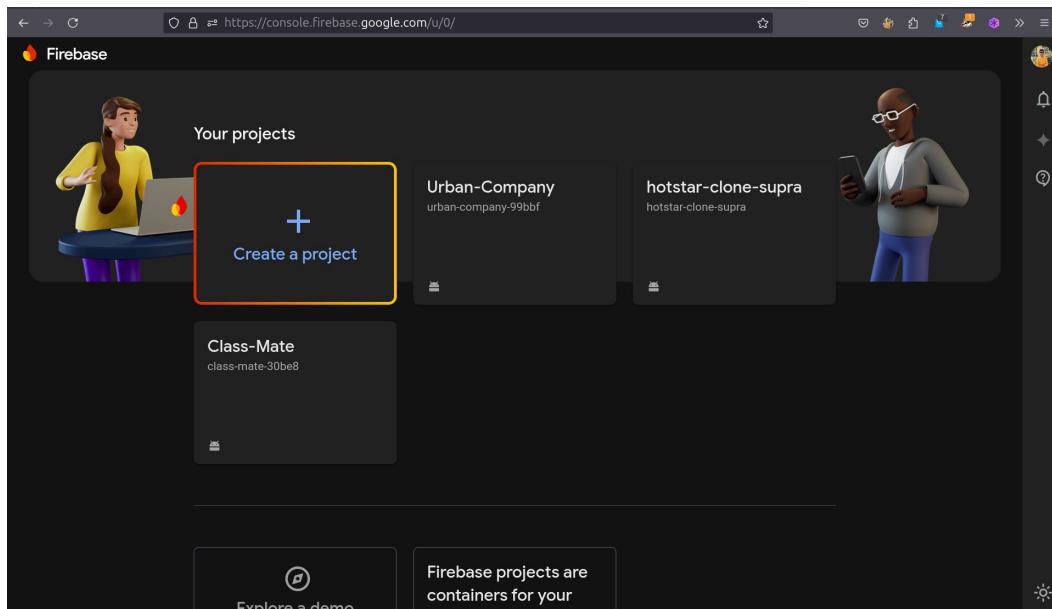
EXPERIMENT NO: - 06**Name:-** Prajyot Shinde**Class:-** D15A**Roll:No:** - 55**AIM:** - To connect Flutter UI with Firebase database.**Theory:** -

Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. Firebase, a Backend-as-a-Service (BaaS) platform, provides real-time database, authentication, and cloud storage services, making it a powerful backend solution for Flutter applications.

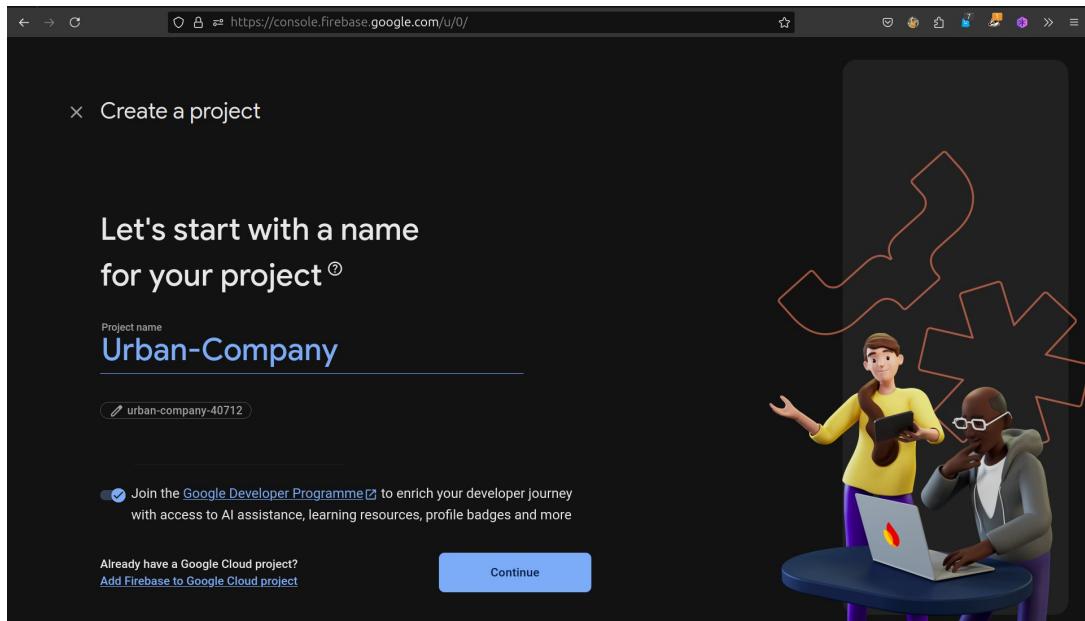
By integrating Firebase with Flutter, developers can store and retrieve data in real time, authenticate users, and manage cloud-based data efficiently. This is particularly useful for applications requiring dynamic content updates and user interactions.

➤ **Steps to Connect Flutter UI with Firebase Database****Step 1:**

- 1.1) Go to Firebase Console and Create a Firebase Project



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



- 1.3) Enable Google Analytics (optional) & Click continue and complete the setup

Project Overview

Urban-Company

Join now Dismiss

Share your feedback with Firebase

Participate now

Build

Firestore

Reads (current)	Writes (current)
84 -63.6%	0 -100%

Feb 24 Feb 25 Feb 26 Feb 27 Feb 28 Mar 1 Mar 2

Step 2:- Add Firebase to Your Flutter App

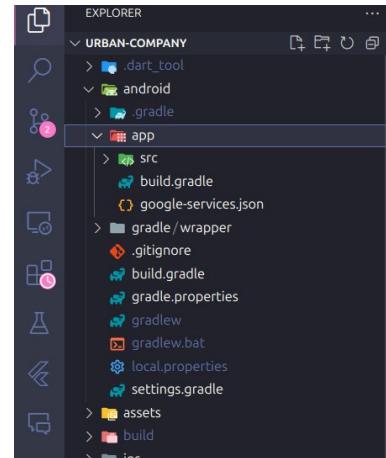
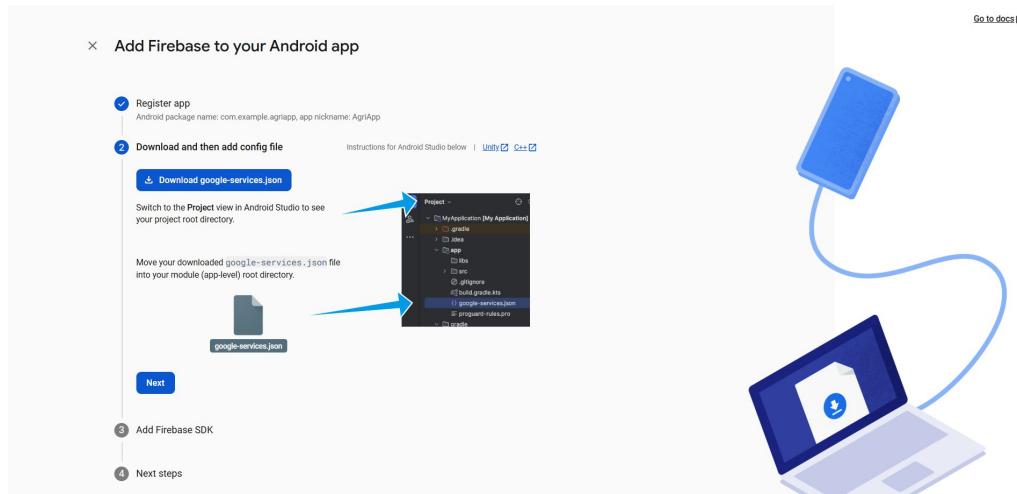
- 2.1) Click on Android/iOS/Web based on your Flutter application

The screenshot shows the Firebase Project Overview page for a project named 'Urban-Company'. On the left, there's a sidebar with options like Generative AI, Authentication, Firestore Database, Build, Run, and Analytics. The main area displays real-time metrics for Firestore: Reads (current: 84, -63.6%) and Writes (current: 0, -100%). Below the metrics are two line graphs showing data trends from Feb 24 to Mar 2. A modal window titled 'Share your feedback with Firebase' is overlaid on the page, asking users to participate in a Firebase research study.

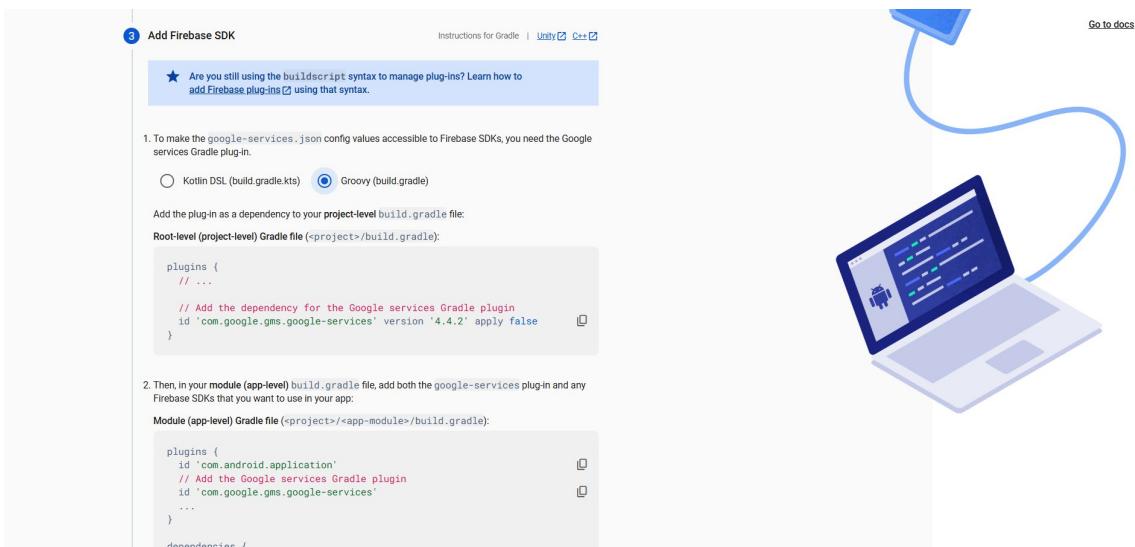
- 2.2) Register your app with a unique package name (found in android/app/build.gradle for Android).

The screenshot shows the 'Add Firebase to your Android app' registration step. It includes fields for 'Android package name' (set to 'com.company.urban') and 'App nickname(optional)' (set to 'UrbanCompany'). There's also a field for 'Debug signing certificate SHA-1 (optional)' with a placeholder value. A note below the SHA-1 field states: '(Required for Dynamic Links and Google Sign-In or phone number support in Auth. Edit SHA-1s in settings.)'. At the bottom is a 'Register app' button. To the right of the form is an illustration of a smartphone connected to a laptop by a cable, symbolizing the integration process.

- 2.3) Download the google-services.json (for Android) & place the JSON file inside android/app/ directory.



- 2.4) Add Firebase SDK dependencies to android/build.gradle



```

File Edit Selection View Go Run Terminal Help ← → ⌘ Urban-Company ⌘ v
android > app > build.gradle > android
You, 2 weeks ago | 1 author (You)
1  plugins {
2    id "com.android.application"
3    id "kotlin-android"
4    id "dev.flutter.flutter-gradle-plugin"
5  }
6
7  android {
8    namespace = "com.example.urban_company_app"
9    compileSdk = flutter.compileSdkVersion
10   ndkVersion = flutter.ndkVersion
11
12   defaultConfig {
13     applicationId = "com.example.urban_company_app"
14     minSdk = flutter.minSdkVersion
15     targetSdk = flutter.targetSdkVersion
16     versionCode = flutter.versionCode
17     versionName = flutter.versionName
18   }
19
20   compileOptions {
21     sourceCompatibility = JavaVersion.VERSION_1_8
22     targetCompatibility = JavaVersion.VERSION_1_8
23   }
24
25   kotlinOptions {
26     jvmTarget = "1.8"
27   }
28
29   buildTypes {
30     release {
31       signingConfig = signingConfigs.debug
32     }
33   }
}

```

Step 3: - Add Firebase Authentication to Your App

3.1) Add Firebase Authentication Dependencies

```

service_card.dart BaseScreen.dart profile_screen.dart pubspec.yaml
pubspec.yaml
You, last week | 1 author (You)
1 name: urban_company_app
2 description: A Flutter app for Urban Company services
3
4 environment:
5   sdk: ">=3.0.0 <4.0.0"
6
7 dependencies:
8   flutter:
9     sdk: flutter
10    google_fonts: ^4.0.0
11    animated_text_kit: ^4.2.2 You, last week + Add
12    firebase_core: ^2.15.0
13    firebase_auth: ^4.16.0
14    cloud_firestore: ^4.17.5
15    firebase_database: ^10.2.0
16    cupertino_icons: ^1.0.6
17    provider: ^6.1.1
18    cached_network_image: ^3.3.0
19    flutter_rating_bar: ^4.0.1
20    intl: ^0.18.1
21    carousel_slider: ^5.0.0
22
23 dev_dependencies:
24   flutter_test:
25     sdk: flutter
26   flutter_lints: ^2.0.0
27
28 flutter:
29   uses-material-design: true
30   assets:
31     # Banners
32     - assets/images/banners/banner2.png

```

3.2) Enable Authentication in Firebase Console

Go to **Firebase Console → Authentication.**

Click on **Sign-in method** and enable **Email/Password** (or any other method like Google).

Click Save

Identifier	Providers	Created	Signed in	User UID
+917506509308		23 Feb 2025	25 Feb 2025	TLuSiH0FyFN1SrWg3aEa...
+917506509307		11 Feb 2025	25 Feb 2025	OIUVRnfPKHRYXHBYTnI...
+919529567991		11 Feb 2025	25 Feb 2025	xoKbxR8PGfdiUoFTm37s...

3.3) Implement Authentication in Flutter Modify main.dart

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

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

Step 4: -Configure Firebase Realtime Database

- 4.1) Go to Firebase Console → Realtime Database.
- 4.2) Click **Create Database** → Choose location → Set rules (for development, set read/write to true).
- 4.3) Click **Publish**.

Code:-**Signup_page.dart**

```

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

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

  @override
  State<SignupScreen> createState() =>
      _SignupScreenState();
}

class _SignupScreenState extends State<SignupScreen> {
  final _formKey = GlobalKey<FormState>();
  final _nameController = TextEditingController();
  final _emailController = TextEditingController();
  final _addressController = TextEditingController();
  final _authService = AuthService();
  bool _isLoading = false;

  Future<void> _signup() async {
    if (_formKey.currentState!.validate()) {
      setState(() => _isLoading = true);
      try {
        await _authService.signupUser(
          name: _nameController.text,
          email: _emailController.text,
          address: _addressController.text,
        );
        Navigator.pushReplacementNamed(context,
          '/home');
      } catch (e) {
        print("Error in signup: $e");
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text('Error signing up: $e')),
        );
      } finally {
        setState(() => _isLoading = false);
      }
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Create Your Account'),
        centerTitle: true,
        elevation: 0,
      ),
    );
  }
}

```

```

body: SafeArea(
  child: Padding(
    padding: const EdgeInsets.all(16.0),
    child: SingleChildScrollView(
      child: Form(
        key: _formKey,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            const SizedBox(height: 24),
            Text(
              'Please provide the following details:',
              style: TextStyle(
                fontSize: 16,
                color: Colors.grey[700],
              ),
            ),
            const SizedBox(height: 24),
            _buildTextField(
              controller: _nameController,
              labelText: 'Full Name',
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Please enter your name';
                }
                return null;
              },
            ),
            const SizedBox(height: 16),
            _buildTextField(
              controller: _emailController,
              labelText: 'Email Address',
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Please enter your email';
                }
                if (!value.contains('@')) {
                  return 'Please enter a valid email';
                }
                return null;
              },
            ),
            const SizedBox(height: 16),
            _buildTextField(
              controller: _addressController,
              labelText: 'Address',
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Please enter your address';
                }
                return null;
              },
            ),
            const SizedBox(height: 24),
            SizedBox(
              width: double.infinity,
              child: ElevatedButton(
                onPressed: _isLoading ? null : _signup,
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.deepPurple,
                  padding: const EdgeInsets.symmetric(vertical: 16),
                  shape: RoundedRectangleBorder(

```

```

        borderRadius:
BorderRadius.circular(8),
),
),
),
child: _isLoading
? const CircularProgressIndicator(
    color: Colors.white,
)
: const Text(
    'Sign Up',
    style: TextStyle(
        fontSize: 18,
        fontWeight: FontWeight.bold,
    ),
),
),
),
),
],
),
),
),
),
),
),
);
}

// Custom TextField widget for cleaner code
Widget _buildTextField({
    required TextEditingController controller,
    required String labelText,
    required String? Function(String?) validator,
}) {
}
return TextFormField(
    controller: controller,
    decoration: InputDecoration(
        labelText: labelText,
        border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(8),
        ),
        contentPadding: const EdgeInsets.symmetric(vertical: 16, horizontal: 12),
        filled: true,
        fillColor: Colors.grey[100],
    ),
    validator: validator,
);
}
}

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

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

```

login_page.dart

```

import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:animated_text_kit/animated_text_kit.dart';
import 'auth_service.dart';

```

```
class LoginScreen extends StatefulWidget {
```

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

```
@override
```

```
State<LoginScreen> createState() =>
_LoginScreenState();
```

```
}
```

```

        onError: (error) {
            setState(() => _isLoading = false);
            ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(content: Text(error)),
            );
        },
    );
}

@Override
void dispose() {
    _phoneController.dispose();
    super.dispose();
}

Future<void> _sendOtp() async {
    if (_formKey.currentState!.validate()) {
        setState(() => _isLoading = true);
        final phoneNumber = _phoneController.text;

        await _authService.verifyPhone(
            phoneNumber: phoneNumber,
            onCodeSent: (verificationId) {
                setState(() => _isLoading = false);
                Navigator.pushNamed(
                    context,
                    '/otp',
                    arguments: {
                        'phoneNumber': phoneNumber,
                        'verificationId': verificationId,
                    },
                );
            },
        );
    }
}

onError: (error) {
    setState(() => _isLoading = false);
    ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text(error)),
    );
},
);

}

@Override
Widget build(BuildContext context) {
    return Scaffold(
        body: SafeArea(
            child: Padding(
                padding: const EdgeInsets.all(16.0),
                child: Form(
                    key: _formKey,
                    child: Column(
                        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                        crossAxisAlignment: CrossAxisAlignment.stretch,
                        children: [
                            const SizedBox(height: 48),
                            Center(
                                child: AnimatedTextKit(
                                    animatedTexts: [
                                        TyperAnimatedText(
                                            'Urban Company',
                                            textStyle: GoogleFonts.poppins(
                                                fontSize: 36.0,
                                                fontWeight: FontWeight.bold,
                                                color: Colors.deepPurple,
                                            ),
                                        ),
                                    ],
                                ),
                            ),
                        ],
                    ),
                ),
            ),
        ),
    );
}
}

```

```

),
speed: const Duration(milliseconds:
100),
),
],
repeatForever: true,
pause: const Duration(seconds: 2),
),
),
const SizedBox(height: 24),
const Text(
'Welcome Back',
style: TextStyle(
fontSize: 32,
fontWeight: FontWeight.bold,
),
textAlign: TextAlign.center,
),
const SizedBox(height: 8),
const Text(
'Enter your mobile number to continue',
style: TextStyle(
fontSize: 16,
color: Colors.black26,
),
textAlign: TextAlign.center,
),
const SizedBox(height: 32),
TextField(
controller: _phoneController,
keyboardType: TextInputType.phone,
decoration: InputDecoration(
prefixText: '+91 ',
hintText: 'Phone number',
border: OutlineInputBorder(
borderRadius: BorderRadius.circular(8),
),
),
validator: (value) {
final trimmedValue = value?.trim() ?? "";
if (trimmedValue.isEmpty) {
return 'Please enter your phone
number';
}
if (trimmedValue.length != 10) {
return 'Phone number must be 10
digits';
}
if (!RegExp(r'^\d{10}$').hasMatch(trimmedValue)) {
return 'Please enter a valid phone
number';
}
return null;
},
),
const SizedBox(height: 24),
Container(
decoration: BoxDecoration(
gradient: const LinearGradient(
colors: [Colors.deepPurple,
Colors.purpleAccent],
),
borderRadius: BorderRadius.circular(8),
),
),
)
);

```

```

        child: ElevatedButton(
            onPressed: _isLoading ? null : _sendOtp,
            style: ElevatedButton.styleFrom(
                backgroundColor: Colors.transparent,
                foregroundColor: Colors.white,
                minimumSize: const
                Size(double.infinity, 50),
                elevation: 0,
            ),
            child: _isLoading
                ? const SizedBox(
                    height: 20,
                    width: 20,
                    child: CircularProgressIndicator(
                        strokeWidth: 2,
                        color: Colors.white,
                    ),
                )
                : const Text('Continue',
                    style: TextStyle(fontSize: 18)),
            ),
        ],
    ),
),
),
),
),
),
),
),
);
}
}

```

Profile_screen.dart

```

import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import '../auth/auth_service.dart';
import 'package:intl/intl.dart';

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

    @override
    State<ProfileScreen> createState() =>
        _ProfileScreenState();
}

class _ProfileScreenState extends State<ProfileScreen> {
    final _formKey = GlobalKey<FormState>();
    final _nameController = TextEditingController();
    final _emailController = TextEditingController();

```

```

final _addressController = TextEditingController();
final _phoneController = TextEditingController();
bool _isLoading = true;

final _authService = AuthService();

@Override
void initState() {
    super.initState();
    _loadProfile();
}

String _formatDate(String timestamp) {
    try {
        final date = DateTime.parse(timestamp);
        return DateFormat('yyyy-MM-dd')
            .format(date); // Formats date as YYYY-MM-
DD
    } catch (e) {
        return 'Invalid date';
    }
}

Future<void> _loadProfile() async {
    final user = FirebaseAuth.instance.currentUser;
    if (user == null) {
        setState(() {
            _isLoading = false;
        });
        // Redirect to login screen if the user is not
        authenticated
    } else {
        Navigator.pushReplacementNamed(context,
        '/login');
        return;
    }

    try {
        // Normalize phone number
        String userPhone = user.phoneNumber ?? "";
        String normalizedPhone =
        _normalizePhoneNumber(userPhone);

        // Fetch profile data
        final profileData =
        await
        _authService.getUserProfileByPhoneNumber(normalizedPhone);

        if (profileData != null) {
            setState(() {
                _nameController.text = profileData['name'] ??
                "";
                _emailController.text = profileData['email'] ??
                "";
                _addressController.text =
                profileData['address'] ??
                "";
                _phoneController.text = profileData['phone'] ??
                "";
                _isLoading = false;
            });
        } else {
            ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(content: Text('No profile
            found')),
            );
            setState(() {
        }
    }
}

```

```

    _isLoading = false;
  });
}

} catch (e) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(content: Text('Error loading profile: $e')),
);
 setState(() {
  _isLoading = false;
});
}
}

void _cancelSubscription(BuildContext context,
String subscriptionId) async {
  await FirebaseFirestore.instance
    .collection('user_subscriptions')
    .doc(subscriptionId)
    .update({'status': 'cancelled'});
}

ScaffoldMessenger.of(context).showSnackBar(
  const SnackBar(content: Text('Subscription cancelled.')),
);
setState(() {}); // Refresh the UI
}

String _normalizePhoneNumber(String
phoneNumber) {
  String normalizedPhone =
  phoneNumber.replaceAll(RegExp(r'[^\d]'), '');
  if (normalizedPhone.length > 10) {
    normalizedPhone =
    normalizedPhone.substring(normalizedPhone.length -
    10);
  }
  return normalizedPhone;
}

}

@override
Widget build(BuildContext context) {
  if (_isLoading) {
    return const Scaffold(
      body: Center(child:
CircularProgressIndicator()),
);
  }
  return Scaffold(
    appBar: AppBar(
      title: const Text('My Profile'),
      actions: [
        IconButton(
          icon: const Icon(Icons.logout),
          onPressed: _logout,
        ),
      ],
    ),
    body: SingleChildScrollView(
      padding: const EdgeInsets.all(16),
      child: Column(
        children: [
          Card(
            elevation: 4,
            shape: RoundedRectangleBorder(

```

```

borderRadius: BorderRadius.circular(12),
),
child: Padding(
padding: const EdgeInsets.all(16),
child: Form(
key: _formKey,
child: Column(
crossAxisAlignment:
CrossAxisAlignment.start,
children: [
const Center(
child: CircleAvatar(
radius: 50,
backgroundColor:
Colors.deepPurple,
child: Icon(
Icons.person,
size: 50,
color: Colors.white,
),
),
),
),
const SizedBox(height: 24),
Text(
'Phone:
${FirebaseAuth.instance.currentUser?.phoneNumber
?? _phoneController.text}',
style: const TextStyle(
fontSize: 16,
fontWeight: FontWeight.bold,
),
textAlign: TextAlign.center,
),
),
const SizedBox(height: 24),
TextFormField(
controller: _nameController,
decoration: const InputDecoration(
labelText: 'Full Name',
border: OutlineInputBorder(),
prefixIcon: Icon(Icons.person),
),
validator: (value) {
if (value == null || value.isEmpty) {
return 'Please enter your name';
}
return null;
}),
const SizedBox(height: 16),
TextFormField(
controller: _emailController,
decoration: const InputDecoration(
labelText: 'Email',
border: OutlineInputBorder(),
prefixIcon: Icon(Icons.email),
),
validator: (value) {
if (value == null || value.isEmpty) {
return 'Please enter your email';
}
if (!value.contains('@')) {
return 'Please enter a valid email';
}
return null;
}),
),

```

```

),
const SizedBox(height: 16),
TextField(
  controller: _addressController,
  decoration: const InputDecoration(
    labelText: 'Address',
    border: OutlineInputBorder(),
    prefixIcon: Icon(Icons.home),
  ),
  maxLines: 3,
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your address';
    }
    return null;
  },
),
const SizedBox(height: 24),
SizedBox(
  width: double.infinity,
  child: ElevatedButton(
    onPressed: _isLoading ? null : _saveProfile,
    style: ElevatedButton.styleFrom(
      backgroundColor:
        Colors.deepPurple,
      padding: const
        EdgeInsets.symmetric(vertical: 16),
      shape: RoundedRectangleBorder(
        borderRadius:
          BorderRadius.circular(8),
      ),
    ),
),
child: _isLoading
  ? const CircularProgressIndicator(
    color: Colors.white,
  )
  : const Text(
    'Save Profile',
    style: TextStyle(
      fontSize: 18,
      fontWeight: FontWeight.bold,
    ),
  ),
),
const SizedBox(height: 24),
ExpansionTile(
  title: const Text(
    'My Subscriptions',
    style: TextStyle(
      fontSize: 18,
      fontWeight: FontWeight.bold,
    ),
  ),
  initiallyExpanded: true,
  children: [
    FutureBuilder<QuerySnapshot>(
      future: FirebaseFirestore.instance
        .collection('user_subscriptions')

```

```

    .where('userId',
           isEqualTo:
             FirebaseAuth.instance.currentUser!.uid)
    .get(),
    builder: (context, snapshot) {
      if (snapshot.connectionState ==
          ConnectionState.waiting) {
        return const Center(child:
          CircularProgressIndicator());
      } else if (snapshot.hasError) {
        return Center(child: Text('Error:
          ${snapshot.error}'));
      } else if (!snapshot.hasData ||
          snapshot.data!.docs.isEmpty) {
        return const Center(
          child: Text('No active
          subscriptions.'));
      } else {
        final subscriptions =
        snapshot.data!.docs;
        return ListView.builder(
          shrinkWrap: true,
          physics: const
            NeverScrollableScrollPhysics(),
          itemCount: subscriptions.length,
          itemBuilder: (context, index) {
            final subscription =
            subscriptions[index].data()
              as Map<String, dynamic>;
            return Card(
              elevation: 4,
              margin: const
                EdgeInsets.symmetric(vertical: 8),
              shape: RoundedRectangleBorder(
                borderRadius:
                  BorderRadius.circular(12),
                ),
                child: Padding(
                  padding: const EdgeInsets.all(16),
                  child: Column(
                    crossAxisAlignment:
                      CrossAxisAlignment.start,
                    children: [
                      Row(
                        children: [
                          const Icon(Icons.assignment,
                            color: Colors.deepPurple),
                          const SizedBox(width: 8),
                          Expanded(
                            child: Text(
                              subscription['serviceDetails']
                                ['title'],
                            style: const TextStyle(
                              fontSize: 18,
                              fontWeight:
                                FontWeight.bold,
                            ),
                            overflow:
                              TextOverflow.ellipsis,
                          )));
                        ],
                      ),
                      const SizedBox(height: 8),
                      Row(
                        children: [
                          const
                            Icon(Icons.calendar_today,
                            ),
                          ],
                        ),
                      ],
                    ],
                  ),
                ),
              );
            }
          },
        ),
      );
    }
  ),
);

```

```

        size: 16, color:
Colors.blue),
        const SizedBox(width: 8),
        Text(
            'Next Delivery:
${_formatDate(subscription['nextDelivery'])}',
            style: const TextStyle(
                fontSize: 14,
                color: Colors.blue,
            ),
        ),
    ],
),
const SizedBox(height: 8),
Row(
    children: [
        const Icon(Icons.star,
            size: 16, color:
Colors.orange),
        const SizedBox(width: 8),
        Text(
            'Status:
${subscription['status']}',
            style: TextStyle(
                fontSize: 14,
                color:
                    subscription['status'] ==
'active'
                    ? Colors.green
                    : Colors.red,
            ),
        ),
    ],
),
        const SizedBox(height: 16),
        Align(
            alignment: Alignment.centerRight,
            child: ElevatedButton(
                onPressed: () {
                    _cancelSubscription(
                        context,
                        subscriptions[index].id);
                },
                style: ElevatedButton.styleFrom(
                    backgroundColor:
Colors.red,
                    padding: const EdgeInsets.symmetric(
                        horizontal: 16, vertical:
8),
                    shape: RoundedRectangleBorder(
                        borderRadius:
BorderRadius.circular(8),
                    ),
                ),
                child: const Text(
                    'Cancel',
                    style: TextStyle(
                        fontSize: 14, color:
Colors.white),
                ),
            ),
        ],
),
),

```

```

        ),
        email: _emailController.text,
        );
        address: _addressController.text,
        );
        if (mounted) {
        ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Profile updated
        successfully')),
        );
        }
        } catch (e) {
        if (mounted) {
        ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Error updating
        profile: $e')),
        );
        }
        Future<void> _saveProfile() async {
        if (_formKey.currentState!.validate()) {
        setState(() {
        _isLoading = true;
        });
        } finally {
        if (mounted) {
        setState(() {
        _isLoading = false;
        });
        }
        try {
        final user = FirebaseAuth.instance.currentUser;
        if (user == null) {
        throw Exception('User not logged in');
        }
        }
        Future<void> _logout() async {
        try {
        await FirebaseAuth.instance.signOut();
        if (mounted) {
        Navigator.pushReplacementNamed(context,
        '/login');
        }
        }
        
```

```
        }

    } catch (e) {

        if (mounted) {

            ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(content: Text('Error signing out:
$e')),

        );
    }
}
}
```

Main.dart

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import '/screens/home_screen.dart';
import '/screens/auth/login_screen.dart';
import '/screens/bookings/bookings_screen.dart';
import '/screens/profile/profile_screen.dart';
import '/screens/service_details_screen.dart';
import '/screens/help/help_screen.dart';
import '/screens/auth/otp_verify.dart';
import '/screens/auth/signup_screen.dart';
import './screens/handyman_service_screen.dart';

void main() async {
```

```
WidgetsFlutterBinding.ensureInitialized();

await Firebase.initializeApp();

runApp(const MyApp());

}

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

@override
Widget build(BuildContext context) {
return MaterialApp(
debugShowCheckedModeBanner: false,
title: 'Urban Company',
theme: ThemeData(
colorScheme: ColorScheme.light(
primary: Colors.deepPurple,
secondary: Colors.orangeAccent,
),
scaffoldBackgroundColor: Colors.grey[50],
fontFamily: 'Poppins',
textTheme: GoogleFonts.poppinsTextTheme(),
appBarTheme: AppBarTheme(
backgroundColor: Colors.deepPurple[50],
elevation: 2,
iconTheme: const IconThemeData(color: Colors.deepPurple),
titleTextStyle: GoogleFonts.poppins(
color: Colors.deepPurple,
fontSize: 18,
fontWeight: FontWeight.w600,
),
),
),
```

```

elevatedButtonTheme:
ElevatedButtonThemeData(
    style: ElevatedButton.styleFrom(
        backgroundColor: Colors.deepPurple,
        foregroundColor: Colors.white,
        padding: const EdgeInsets.symmetric(vertical:
16, horizontal: 24),
        shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(8),
        ),
    ),
),
),
),
),
),

home: AuthWrapper(),
routes: {
    '/signup': (context) => const SignupScreen(),
    '/otp': (context) {
        final args =
        ModalRoute.of(context)?.settings.arguments;

        if (args is Map<String, String>) {
            return OtpVerificationScreen(
                phoneNumber: args['phoneNumber'] ?? '',
                verificationId: args['verificationId'] ?? '',
            );
        } else {
            return const Scaffold(
                body: Center(child: Text('Invalid arguments
for OTP screen')),
            );
        }
    },
    '/login': (context) => const LoginScreen(),
},
'/home': (context) => const HomeScreen(),
'/bookings': (context) => const
BookingsScreen(),
'/help': (context) => const HelpScreen(),
'/profile': (context) => const ProfileScreen(),
'/service-details': (context) =>
ServiceDetailsScreen(),
'/handyman-services': (context) =>
HandymanServiceScreen(),
},
);
}
}

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

@override
Widget build(BuildContext context) {
return StreamBuilder<User?>(
stream:
FirebaseAuth.instance.authStateChanges(),
builder: (context, authSnapshot) {
if (authSnapshot.connectionState ==
ConnectionState.waiting) {
return const Scaffold(
body: Center(child:
CircularProgressIndicator()),
);
} else if (authSnapshot.hasData) {
return FutureBuilder<Map<String,
dynamic>?>(
future:
_checkUserProfile(authSnapshot.data!),
builder: (context, profileSnapshot) {
}
)
}
}
}

```

```
        if(profileSnapshot.connectionState ==
ConnectionState.waiting) {

            return const Scaffold(
                body: Center(child:
CircularProgressIndicator()),
            );
        } else if(profileSnapshot.hasError
|| !profileSnapshot.hasData) {

WidgetsBinding.instance.addPostFrameCallback(_)
{
    Navigator.pushReplacementNamed(context, '/login');

    });
}

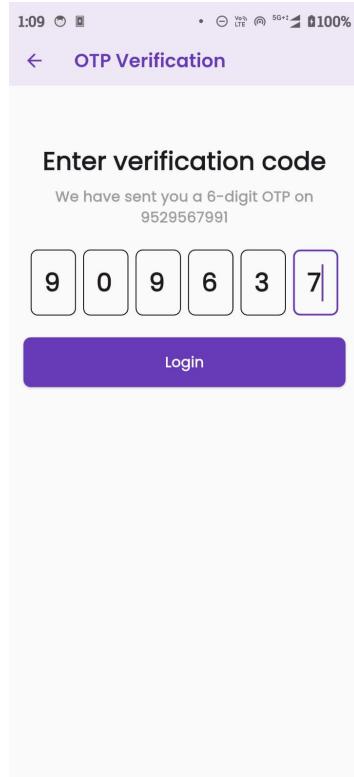
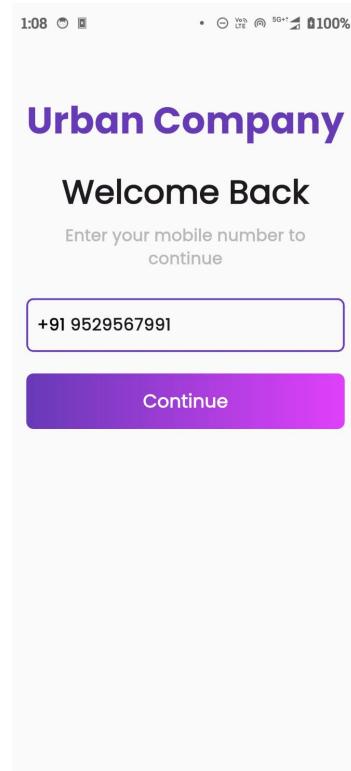
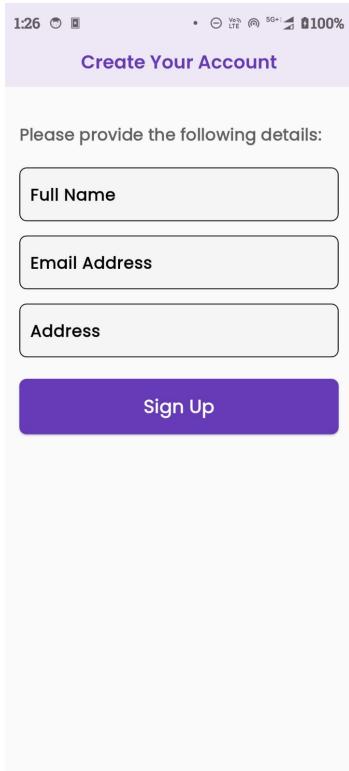
return const Scaffold(
    body: Center(child: Text('Redirecting to
login...')),
);

} else {
    return const HomeScreen();
}
},
);
}

} else {
    return const LoginScreen();
}
},
);
}
}
```

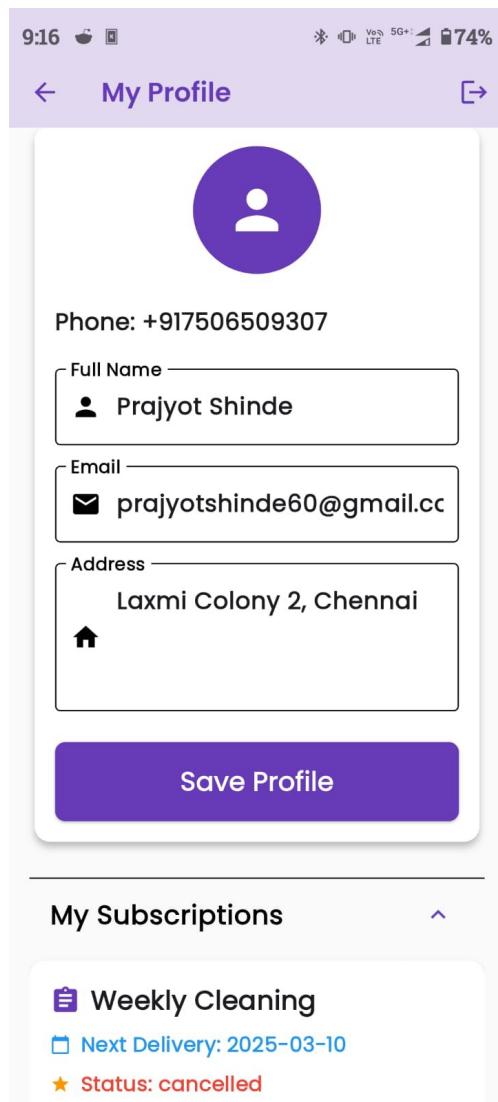
```
Future<Map<String, dynamic>?>
_checkUserProfile(User user) async {
    try {
        final doc = await FirebaseFirestore.instance
            .collection('users')
            .doc(user.uid)
            .get();

        if(doc.exists) {
            return doc.data();
        } else {
            return null;
        }
    } catch (e) {
        print("Error checking user profile: $e");
        return null;
    }
}
```



After Registering, the user details is saved in the database.

User details get fetched from the database in the profile page.



The screenshot shows the Firebase Cloud Firestore interface. On the left, the navigation sidebar includes sections for Project Overview, Generative AI, Build with Gemini, Genkit (NEW), Project shortcuts, Authentication, and the selected Firestore Database. The main area displays the 'Cloud Firestore' dashboard for the 'Urban-Company' project. The top navigation bar has tabs for Data, Rules, Indexes, Disaster recovery (NEW), Usage, and Extensions. A banner at the top right encourages users to 'Protect your Cloud Firestore resources from abuse, such as billing fraud or phishing' and provides a link to 'Configure App Check'. Below this, there's a 'Panel view' button and a 'Query builder' section. The main content area shows a hierarchical view of collections: (default) > users > OIUVRnfPKHRYXHBYTnI7ThlYJ123. The 'users' collection contains a single document with the ID 'OIUVRnfPKHRYXHBYTnI7ThlYJ123'. The document details are as follows:

address: "Laxmi Colony 2, Chennai"
createdAt: 24 February 2025 at 22:43:09 UTC+5:30
email: "prajyotshinde60@gmail.com"
name: "Prajyot Shinde"
phone: "+91 7506509307"
uid: "OIUVRnfPKHRYXHBYTnI7ThlYJ123"

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.	55
Name	Prajyot Shinde
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

Experiment No. 7

Title: To write meta data of your Ecommerce PWA

Aim: To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.

Theory:

Regular Web App

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

Progressive Web App

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

Difference between PWAs vs. Regular Web Apps:

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

1. Native Experience

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

2. Ease of Access

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

3. Faster Services

PWAs can cache the data and serve the user with text stylesheets, images, and other web content even before the page loads completely. This lowers the waiting time for the end-users

and helps the brands improve the user engagement and retention rate, which eventually adds value to their business.

4. Engaging Approach

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

5. Updated Real-Time Data Access

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

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

6. Discoverable

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

7. Lower Development Cost

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

Pros and cons of the Progressive Web App

The main features are:

Progressive — They work for every user, regardless of the browser chosen because they are built at the base with progressive improvement principles.

Responsive — They adapt to the various screen sizes: desktop, mobile, tablet, or dimensions that can later become available.

App-like — They behave with the user as if they were native apps, in terms of interaction and navigation.

Updated — Information is always up-to-date thanks to the data update process offered by service workers.

Secure — Exposed over HTTPS protocol to prevent the connection from displaying information or altering the contents.

Searchable — They are identified as “applications” and are indexed by search engines.

Reactivable — Make it easy to reactivate the application thanks to capabilities such as web notifications.

Installable — They allow the user to “save” the apps that he considers most useful with the corresponding icon on the screen of his mobile terminal (home screen) without having to face all the steps and problems related to the use of the app store.

Linkable — Easily shared via URL without complex installations.

Offline — Once more it is about putting the user before everything, avoiding the usual error message in case of weak or no connection. The PWA are based on two particularities: first of all the ‘skeleton’ of the app, which recalls the page structure, even if its contents do not respond and its elements include the header, the page layout, as well as an illustration that signals that the page is loading.

Weaknesses:

- IOS support from version 11.3 onwards
- Greater use of the device battery
- Not all devices support the full range of PWA features (same speech for iOS and Android operating systems)
- It is not possible to establish a strong re-engagement for iOS users (URL scheme, standard web notifications)
- Support for offline execution is however limited
- Lack of presence on the stores (there is no possibility to acquire traffic from that channel)
- There is no “body” of control (like the stores) and an approval process
- Limited access to some hardware components of the devices
- Little flexibility regarding “special” content for users (eg loyalty programs, loyalty, etc.)

Code:

manifest.json:

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

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

manifest.json

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

Vite + React - Chromium

Overview

- Total Sales \$12,34!
- New Users 1,234
- Total Products 567
- Conversion Rate 12.5%
- Sales Overview 8000

Manifest

Identity

- Name E-Commerce Dashboard
- Short name E-ComDash
- Description A progressive web app for e-commerce administration
- Computed App ID https://supra-dashboard.netlify.app/ [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 #4f46e5
- 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](#).

Console Network

Default levels No Issues

[SW] Mock sync completed

[SW] Sync event: test-tag-from-devtools

[SW] Push event received

[SW] Showing notification: {title: 'New Update', body: 'Test push message from DevTools.', url: '/'}

[SW] Sync event: test-tag-from-devt

[SW] JS:188

[SW] JS:113

[SW] JS:145

[SW] JS:188



Conclusion:

Hence, we learnt how to write a metadata of our website PWA in a Web App Manifest File to enable add to homescreen feature.

MAD & PWA Lab

Journal

Experiment No.	08
Experiment Title.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA
Roll No.	55
Name	Prajyot Shinde
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Experiment No. 8

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

What can we do with Service Workers?

- You can dominate **Network Traffic**

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

- You can **Cache**

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

- You can manage **Push Notifications**

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

- You can **Continue**

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

What can't we do with Service Workers?

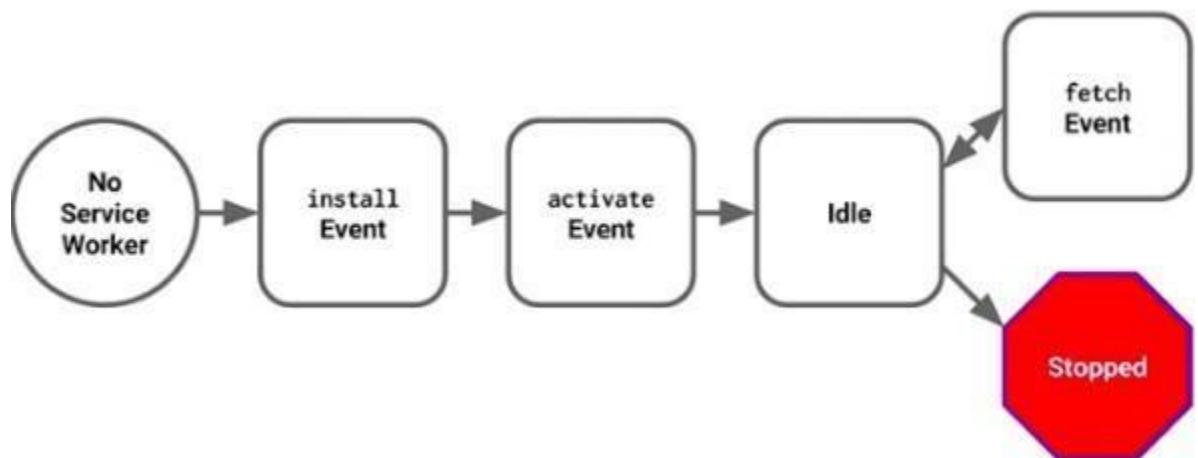
- You can't access the **Window**

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

- You can't work it on **80 Port**

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

Service Worker Cycle



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

- Registration
- Installation
- Activation

Registration

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

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

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

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

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

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

In this case we are setting the scope of the service worker to `/app/`, which means the service worker will control requests from pages like `/app/`, `/app/lower/` and `/app/lower/lower`, but not from pages like `/app` or `/`, which are higher.

If you want the service worker to control higher pages e.g. `/app` (without the trailing slash) you can indeed change the scope option, but you'll also need to set the Service-Worker-Allowed HTTP Header in your server config for the request serving the service worker script.

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

Installation

Once the browser registers a service worker, installation can be attempted. This occurs if the service worker is considered to be new by the browser, either because the site currently doesn't have a registered service worker, or because there is a byte difference between the new service worker and the previously installed one.

A service worker installation triggers an install event in the installing service worker. We can include an install event listener in the service worker to perform some task when the service worker installs. For instance, during the install, service workers can precache parts of a web app so that it loads instantly the next time a user opens it (see caching the application shell). So, after that first load, you're going to benefit from instant repeat loads and your time to interactivity is going to be even better in those cases. An example of an installation event listener looks like this:

service-worker.js

```
self.addEventListener('install', function(event) {  
    // Perform some task  
});
```

Activation

Once a service worker has successfully installed, it transitions into the activation stage. If there are any open pages controlled by the previous service worker, the new service worker enters a waiting state. The new service worker only activates when there are no longer any pages loaded that are still using the old service worker. This ensures that only one version of the service worker is running at any given time.

When the new service worker activates, an activate event is triggered in the activating service worker. This event listener is a good place to clean up outdated caches (see the Offline Cookbook for an example).

service-worker.js

```
self.addEventListener('activate', function(event) {  
    // Perform some task  
});
```

Once activated, the service worker controls all pages that load within its scope, and starts listening for events from those pages. However, pages in your app that were loaded before the service worker activation will not be under service worker control. The new service worker will only take over when you close and reopen your app, or if the service worker calls `clients.claim()`. Until then, requests from this page will not be intercepted by the new service worker. This is intentional as a way to ensure consistency in your site.

Code:

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

service-worker.js

```
const CACHE_NAME = 'my-pwa-cache-v1';
const urlsToCache = [
  '/',
  '/index.html',
  '/styles.css',
  '/icons/icon-192-1.png',
  '/icons/icon-512.png',
  '/assets/brownie.png',
  '/assets/cakes.png',
  '/assets/cookies.png',
  '/assets/donuts.png',
  '/assets/logo.png',
  '/assets/muffins.png',
  '/assets/pastry.png',
];
// Install service worker
self.addEventListener('install', (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => {
      console.log('Opened cache');
      return cache.addAll(urlsToCache);
    })
  );
});
```

```
        })
    );
});

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

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

79% ↑ 25 KB/s Mar 31 10:35 PM Vite + React - Chromium en₂ 92%

@criccoder Vite + React Vite + React Vite + React Settings - Site setting chrome://serviceworker

supra-dashboard.netlify.app Overview

Total Sales \$12,34!

New Users 1,234

Total Products 567

Conversion Rate 12.5%

Sales Overview 8000

Service workers

Manifest

Storage

Local storage

Session storage

Extension storage

IndexedDB

Cookies

Shared storage

Cache storage

Storage buckets

Background services

Back/forward cache

Background fetch

Background sync

Bounce tracking mitigation

Notifications

Payment handler

Periodic background ...

Source: sw.js Received 3/31/2025, 9:48:54 PM Status: #68 activated and is running Stop Push Test push message from DevTools. Sync test-tag-from-devt Periodic sync test-tag-from-devtools Update Cycle Version Update Activity Timeline #68 Install #68 Wait #68 Activate

Service workers from other origins See all registrations

Console Network

[SW] MOCK sync completed [SW] Sync event: test-tag-from-devtools [SW] Push event received [SW] Showing notification: {title: 'New Update', body: 'Test push message from DevTools.', url: ''} [SW] Sync event: test-tag-from-devt

81% ↓ 1 KB/s Mar 31 10:35 PM Vite + React - Chromium en₂ 93%

@criccoder Vite + React Vite + React Vite + React Settings - Site setting chrome://serviceworker

supra-dashboard.netlify.app Overview

Total Sales \$12,34!

New Users 1,234

Total Products 567

Conversion Rate 12.5%

Sales Overview 8000

Application

Manifest

Service workers

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

Bucket name: default

is persistent: No

Durability: relaxed

Quota: 0 B

Expiration: None

#	Name	Response-Type	Content-Type	Content-Length	Time Cached	Vary Header
0	/	basic	text/html	825	3/31/2025, 9...	
1	/admin.png	basic	image/png	9,287	3/31/2025, 9...	
2	/assets/index-Nd1ma4d3js	basic	application/javascript	195,705	3/31/2025, 9...	Accept-Encod...
3	/assets/index-bf150Ph.css	basic	text/css	3,166	3/31/2025, 9...	Accept-Encod...
4	/index.html	basic	text/html	825	3/31/2025, 9...	
5	/manifest.json	basic	application/json	360	3/31/2025, 9...	
6	/offline.html	basic	text/html	531	3/31/2025, 9...	
7	/sw.js	basic	application/javascript	0	3/31/2025, 9...	Accept-Encod...
8	/vite.svg	basic	image/svg+xml	715	3/31/2025, 9...	Accept-Encod...

Console Network

[SW] MOCK sync completed [SW] Sync event: test-tag-from-devtools [SW] Push event received [SW] Showing notification: {title: 'New Update', body: 'Test push message from DevTools.', url: ''} [SW] Sync event: test-tag-from-devt

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.	55
Name	Prajyot Shinde
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

EXPERIMENT NO. 9

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

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

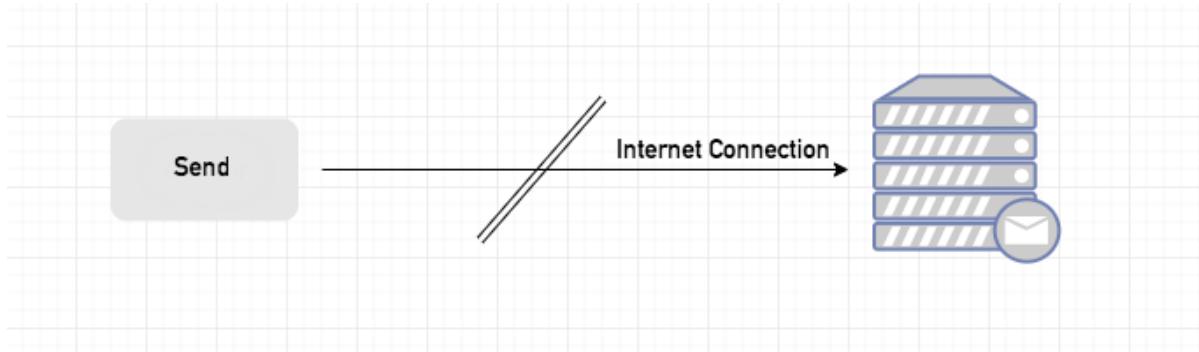
- **CacheFirst** - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.
- **NetworkFirst** - In this function, firstly we can try getting an updated response from the network, if this process completed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

Sync Event

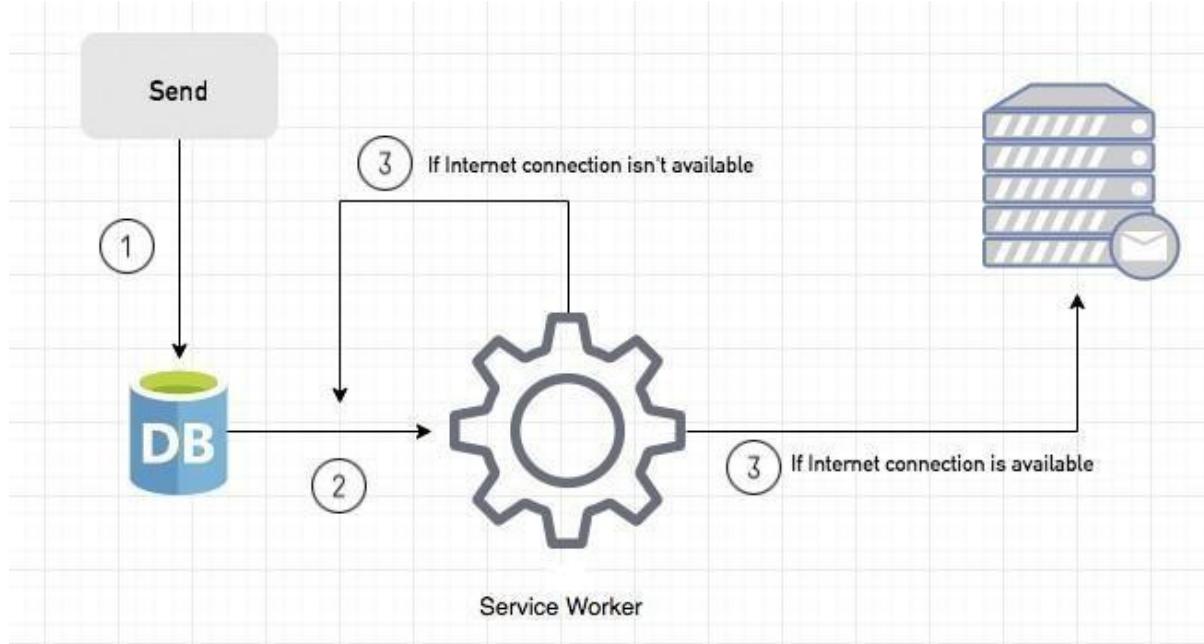
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

Here is a job for the Background Sync.

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



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



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

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

Event Listener for Background Sync Registration

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

Event Listener for sw.js

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

Push Event

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

We can check in the following example.

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

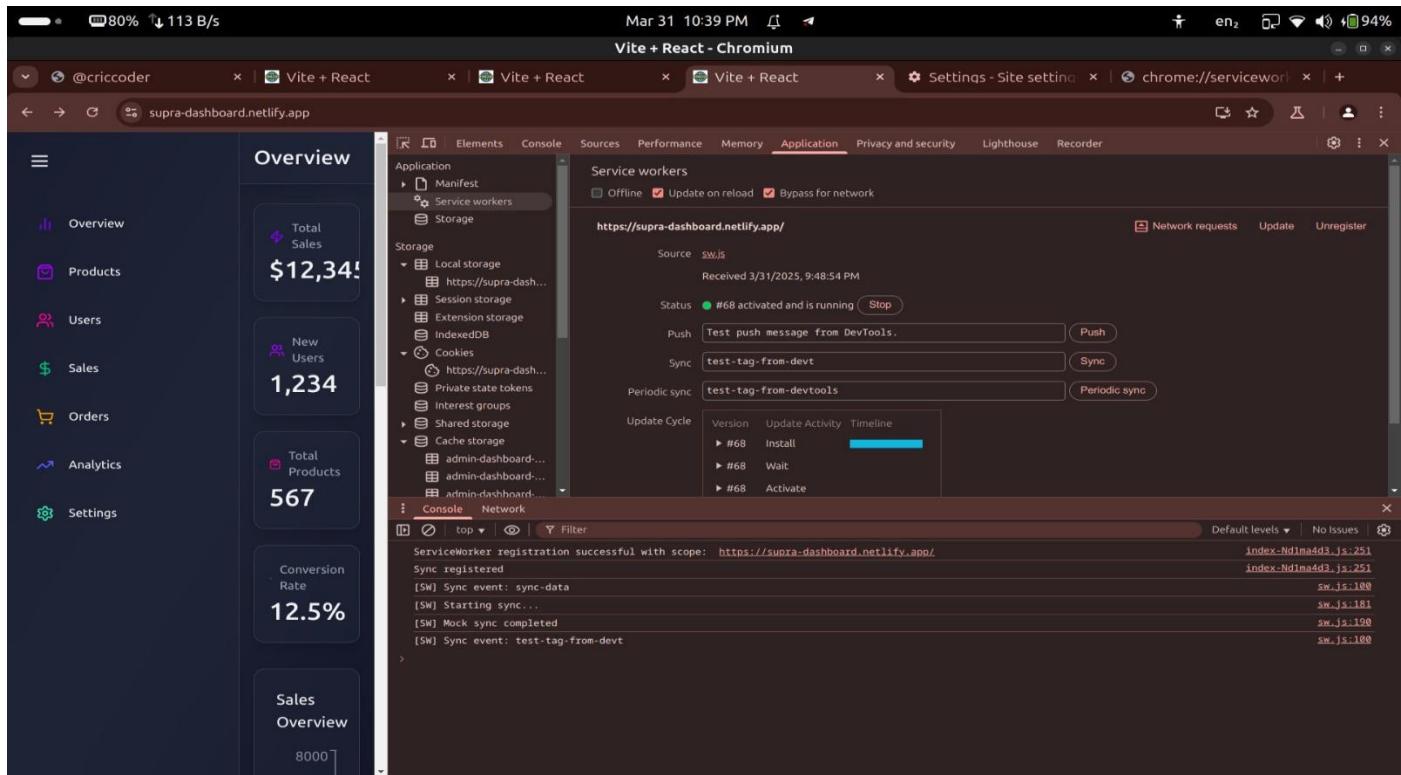
In the following code block is in sw.js file. You can handle push notifications with this event. In this example, I kept it simple. We send an object that has “method” and “message” properties. If the method value is “pushMessage”, we open the information notification with the “message” proper

The screenshot shows a browser window with multiple tabs. The active tab is titled "Chromium Web Browser Just now" and displays the URL "supra-dashboard.netlify.app". A modal window from DevTools is open, specifically the "Service workers" section under the "Application" tab. The modal shows a list of registered service workers, including one named "#68 activated and is running". It also displays configuration details like Push, Sync, and Periodic sync events, along with an Update Cycle timeline showing the "Install", "Wait", and "Activate" steps. Below the main modal, there's a "Console" tab showing log entries related to service workers.

127.0.0.1:5500

⚠ Your connection to this site is not secure
You should not enter any sensitive information
on this site (for example, passwords or credit
cards), because it could be stolen by attackers.
[Learn more](#)

This screenshot shows a "Site settings" dialog box. At the top, it displays a warning about the connection being unsecured. Below this, there are three main sections: "Notifications" (with a purple toggle switch), "Cookies and site data" (with a circular arrow icon), and "Site settings" (with a gear icon). Each section has a "Reset permission" button. The "Notifications" section is currently active.



```

[...]
[✓] Service Worker is ready: main.aa7436b2a0c83ad...d.hot-update.js:469
  ServiceWorkerRegistration {installing: null, waiting: null, active: ServiceWorker, navigationPreLoad: NavigationPreLoadManager, scope: 'http://localhost:3000/'}

[✓] Notification sent successfully main.aa7436b2a0c83ad...d.hot-update.js:489
[✓] Background Sync registered main.aa7436b2a0c83ad...d.hot-update.js:495
[SW] Network response for https://supra-dashboard.netlify.app/manifest.json sw.js:251
Sync registered index-Nd1ma4d3.js:251
ServiceWorker registration successful with scope: index-Nd1ma4d3.js:251
https://supra-dashboard.netlify.app/

② [SW] Network response for https://supra-dashboard.netlify.app/admin.png sw.js:65
[SW] Push event received sw.js:113
[SW] Showing notification: sw.js:145
  {title: 'New Update', body: 'Test push message from DevTools.', url: '/'}
>

```

MAD & PWA Lab

Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	55
Name	Prajyot Shinde
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Experiment No. 10

Aim:

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

Theory:

GitHub Pages

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

GitHub Pages provides the following key features:

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

Reasons for favoring this over Firebase:

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

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

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

Pros

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

Cons

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

Firebase

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

Some of the features offered by Firebase are:

1. Add the Firebase library to your app and get access to a shared data structure. Any changes made to that data are automatically synchronized with the Firebase cloud and with other clients within milliseconds.
2. Firebase apps can be written entirely with client-side code, update in real-time out-of-the-box, interoperate well with existing services, scale automatically, and provide strong data security.
3. Data Accessibility- Data is stored as JSON in Firebase. Every piece of data has its own URL which can be used in Firebase's client libraries and as a REST endpoint. These URLs can also be entered into a browser to view the data and watch it update in real-time.

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

Instacart, 9GAG, and Twitch are some of the popular companies that use Firebase. Firebase has a broader approval, being mentioned in 1215 company stacks & 4651 developer 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/prajyots60/AdminDashboard>

Github Screenshot:

The screenshot shows the GitHub repository page for 'AdminDashboard' owned by 'prajyots60'. The repository is public and has 1 branch and 0 tags. The main file listed is 'main'. The repository was last updated 1 hour ago. The commit history shows several commits from 'prajyots60' refactoring push notification handling in service workers and enhancing functionality. The repository has 0 stars, 1 watcher, and 0 forks. It includes sections for Activity, Releases, Packages, and Languages, with JavaScript being the primary language at 96.2%.

The screenshot shows the GitHub Pages settings page for the 'AdminDashboard' repository. The 'Pages' tab is selected. The 'General' section is open, showing options for Access, Collaborators, and Moderation options. The 'Code and automation' section is also visible. The 'Build and deployment' section shows that GitHub Pages is currently disabled and provides instructions to enable it by selecting a source branch. The 'Source' dropdown is set to 'Deploy from a branch'. The 'Branch' dropdown is set to 'None'. The 'Save' button is present. The 'Visibility' section is shown for GitHub Enterprise accounts, indicating that privately published sites can be shared with enterprise members. A 'Start free for 30 days' button is available for GitHub Enterprise users.

80% 328 B/s Mar 31 11:25 PM en₂ 100%

DeepSeek Generative Download WhatsApp Network Prob WorkFlo Vite + React Vite + React Site overview Vite + React + - ⌂

github.com/prajyots60/AdminDashboard/actions

prajyots60 / AdminDashboard Actions Code Issues Pull requests Actions Projects Wiki Security Insights Settings

All workflows New workflow Filter workflow runs

Help us improve GitHub Actions Give feedback

1 workflow run Event Status Branch Actor

pages build and deployment pages-build-deployment #1: by prajyots60 main 5 minutes ago 38s ...

This screenshot shows the GitHub Actions dashboard for a repository named 'AdminDashboard'. On the left, there's a sidebar with sections like 'Management' (Caches, Deployments, Attestations, Runners, Usage metrics, Performance metrics) and 'Actions' (All workflows, New workflow). The main area is titled 'All workflows' and shows a single workflow run for 'pages build and deployment'. The run was triggered by 'prajyots60' on the 'main' branch, completed 5 minutes ago, and took 38 seconds. There's also a feedback card asking for improvements to GitHub Actions.

80% 329 KB/s Mar 31 11:21 PM en₂ 100%

DeepSeek - Iri Generative A Download Th WhatsApp Network Prob Vite + React Site overview Vite + React + - ⌂

supra-dashboard.netlify.app

☰ Overview Products Users Sales Orders Analytics Settings

Overview

Total Sales: \$12,345

New Users: 1,234

Total Products: 567

Conversion Rate: 12.5%

Sales Overview

A line chart titled 'Sales Overview' showing monthly sales from July to January. The y-axis represents sales in thousands, ranging from 0 to 8,000. The x-axis shows months from July to January. The data points are: July (~4,200), August (~3,800), September (~5,000), October (~4,500), November (~5,200), December (~7,500), and January (~6,000).

Category Distribution

A pie chart titled 'Category Distribution' showing the percentage of sales by category. The categories and their percentages are: Electronics (31%), Clothing (22%), Home & Garden (19%), Sports & Outdoors (14%), and Books (14%).

This screenshot shows a Netlify-deployed dashboard for 'supra-dashboard.netlify.app'. It features a sidebar with links for Overview, Products, Users, Sales, Orders, Analytics, and Settings. The main area has four cards: 'Total Sales' (\$12,345), 'New Users' (1,234), 'Total Products' (567), and 'Conversion Rate' (12.5%). Below these are two larger sections: 'Sales Overview' (a line chart from July to January) and 'Category Distribution' (a pie chart showing product categories and their proportions).

Deployed Link: <https://github.com/prajyots60/AdminDashboard>

MAD & PWA Lab

Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	55
Name	Prajyot Shinde
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

Experiment No. 11

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

Theory :

Google Lighthouse :

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

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

Key Features and Audit Metrics

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

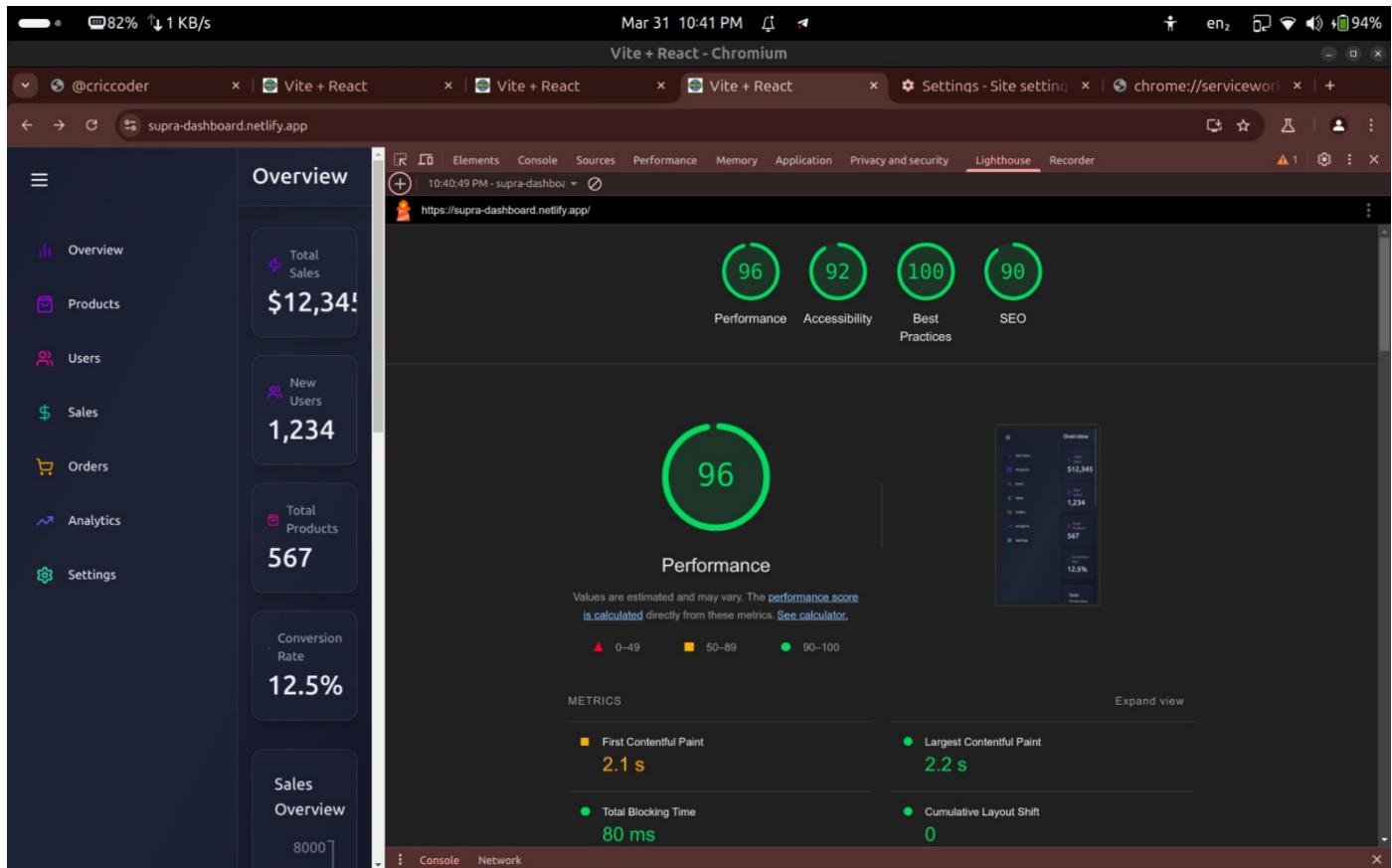
1. **Performance:** This score is an aggregation of how the page fared in aspects such as (but not limited to) loading speed, time taken for loading for basic frame(s), displaying meaningful content to the user, etc. To a layman, this score is indicative of how decently the site performs, with a score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

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

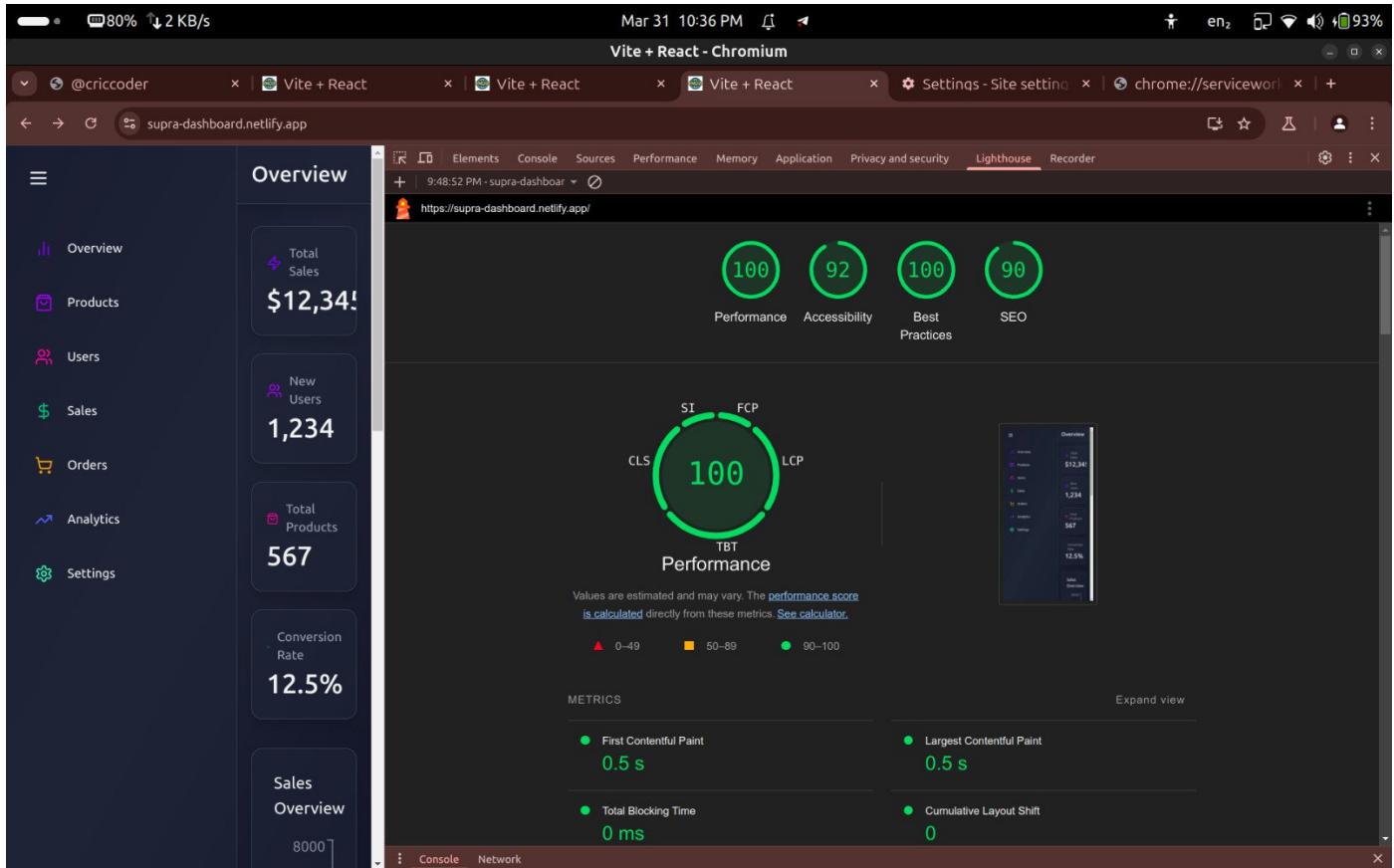
4. **Best Practices:** As any developer would know, there are a number of practices that have been deemed ‘best’ based on empirical data. This metric is an aggregation of many such points, including but not limited to:
Use of HTTPS
Avoiding the use of deprecated code elements like tags, directives, libraries,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

Screenshots:

Mobile: initial



Desktop



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