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

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Name of the Course : MAD & PWA Lab

Course Code : ITL604

Project Title: Virtual private Network. **Roll No:57**

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

Faculty Incharge : Mrs. Kajal Joseph.

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

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

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: Virtual private Network. **Roll No:57**

Lab Objectives:

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

Lab Outcomes:

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

Project Title: Virtual private Network. **Roll No:57**

Index

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

MAD & PWA Lab
Journal

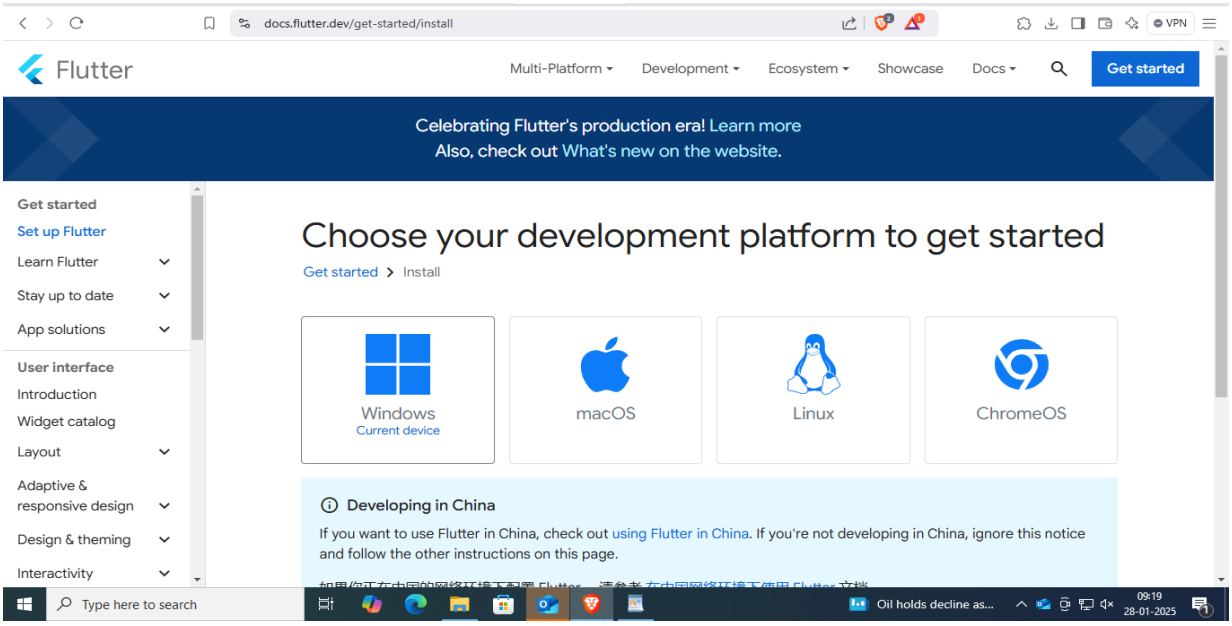
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment.
Roll No.	57
Name	Laksh Sodhai
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework.
Grade:	

EXPERIMENT NO: - 01

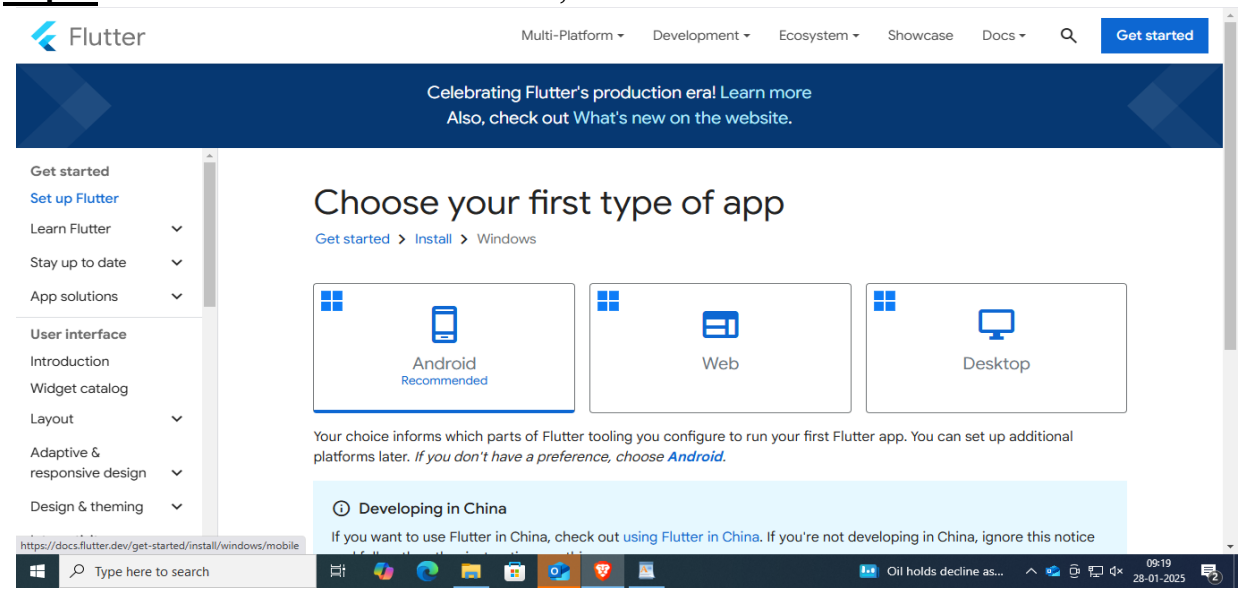
Name:- Laksh Sodhai **Class:-** D15A **Roll:No: -** 57

AIM: - Installation and Configuration of Flutter Environment.

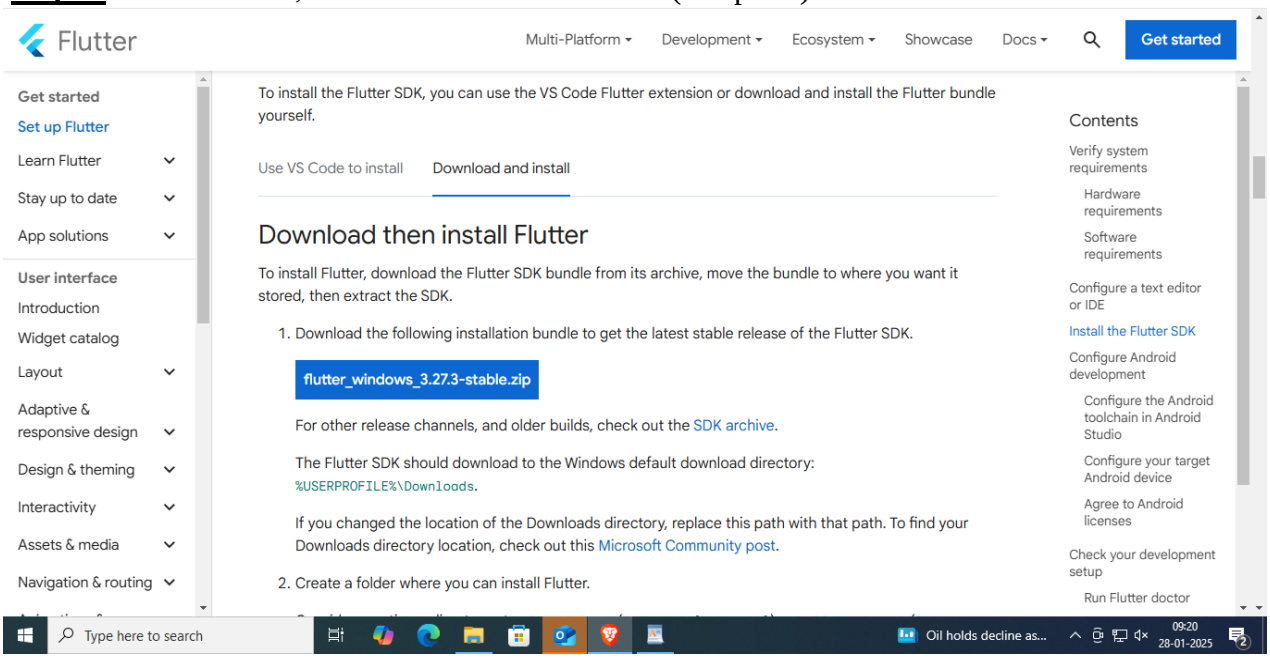
Step 1: Go to the official Flutter website: https://docs.flutter.dev/get-started/install



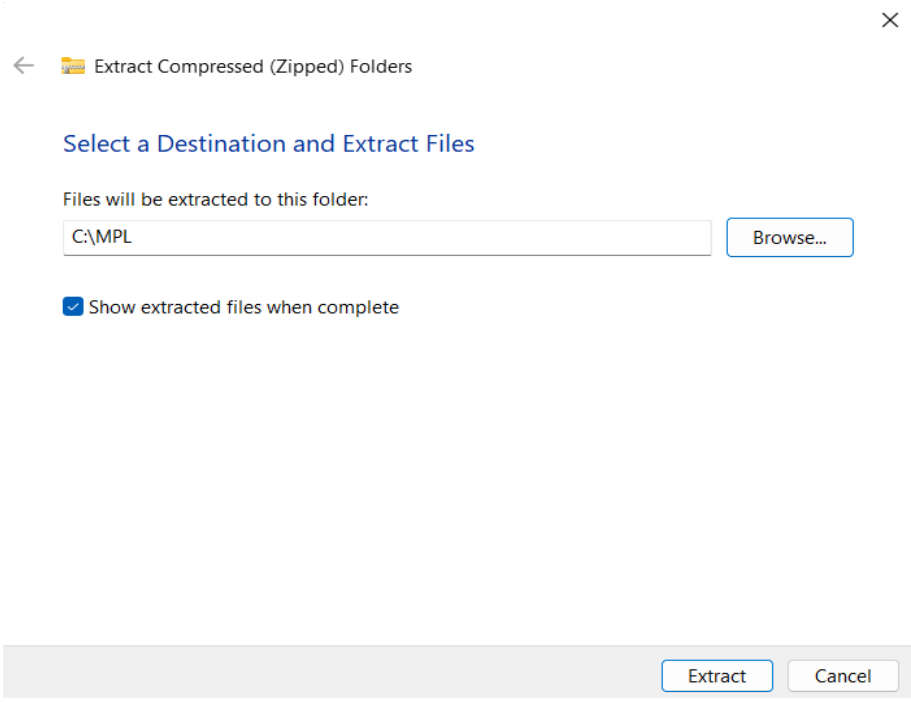
Step 2: To download the latest Flutter SDK, click on the Windows icon > Android



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



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

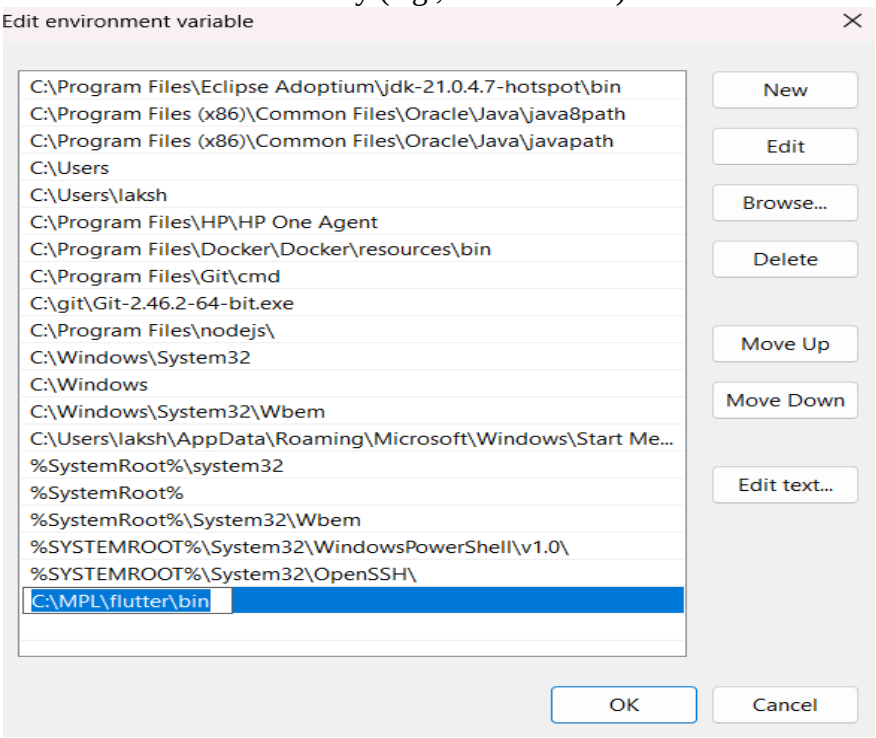


Step 5 :- Add Flutter to System PATH

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

Under System Variables, find Path and click Edit.

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



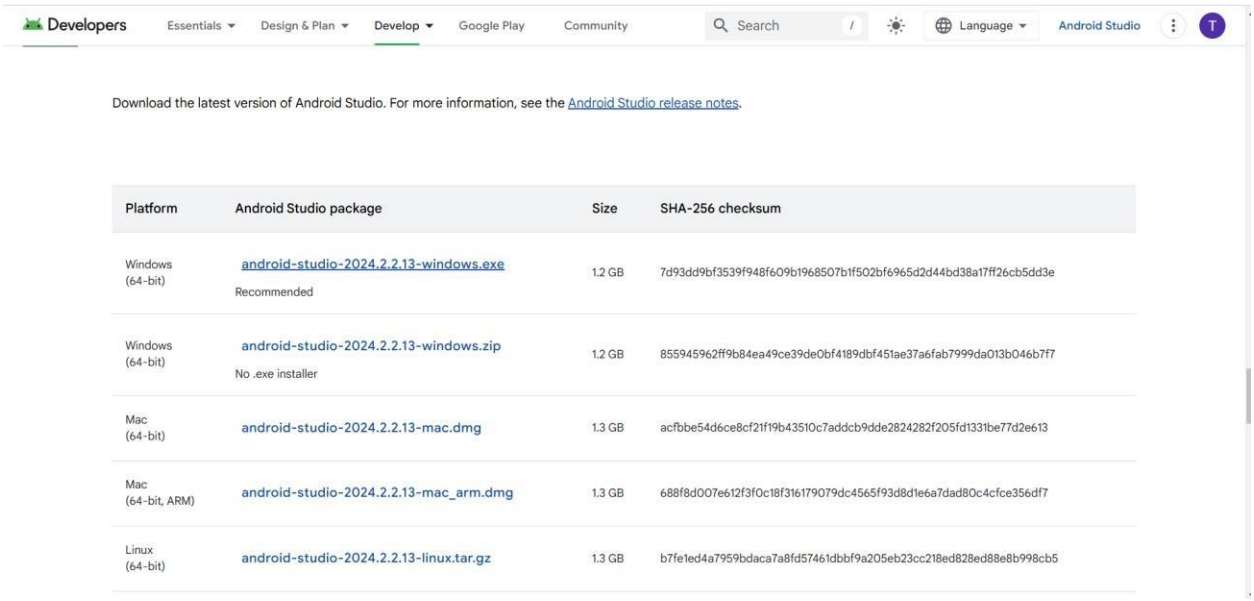
Step 6 : - Now, run the $ flutter command in command prompt.

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

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

Common commands:

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

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

Usage: flutter <command> [arguments]

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

Available commands:
```

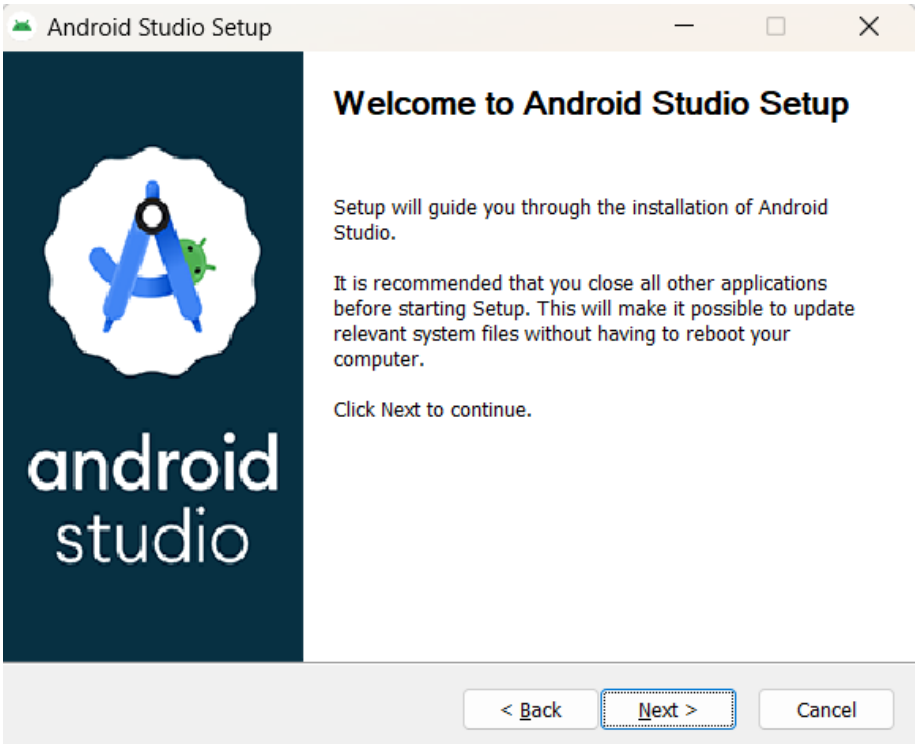
Step 7:- Run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation

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

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

! Doctor found issues in 3 categories.
```

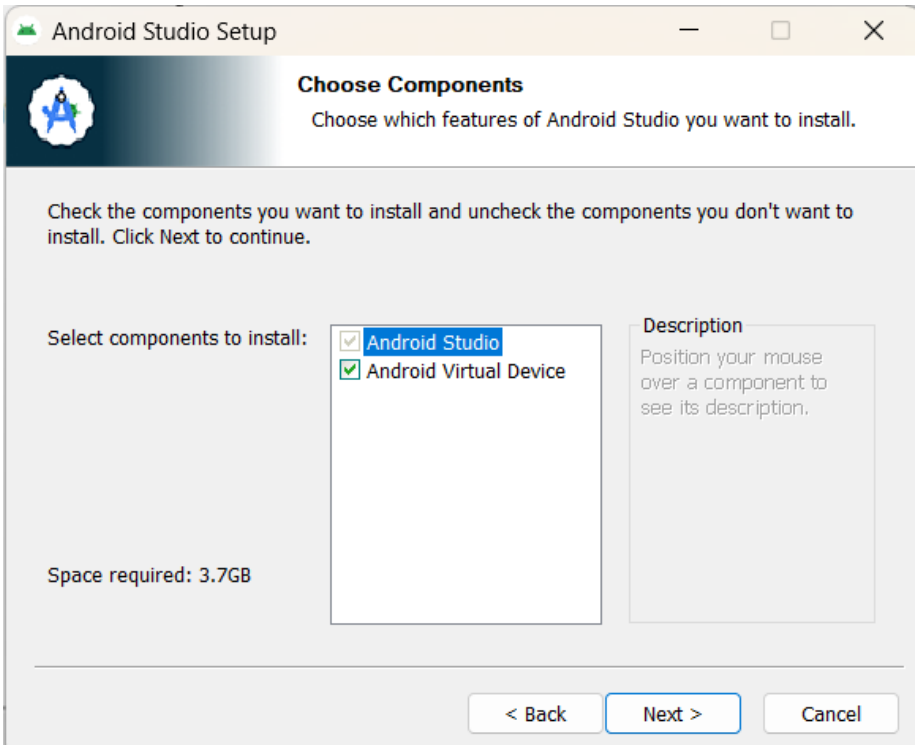
Step 8 : - Go to Android Studio and download the installer.



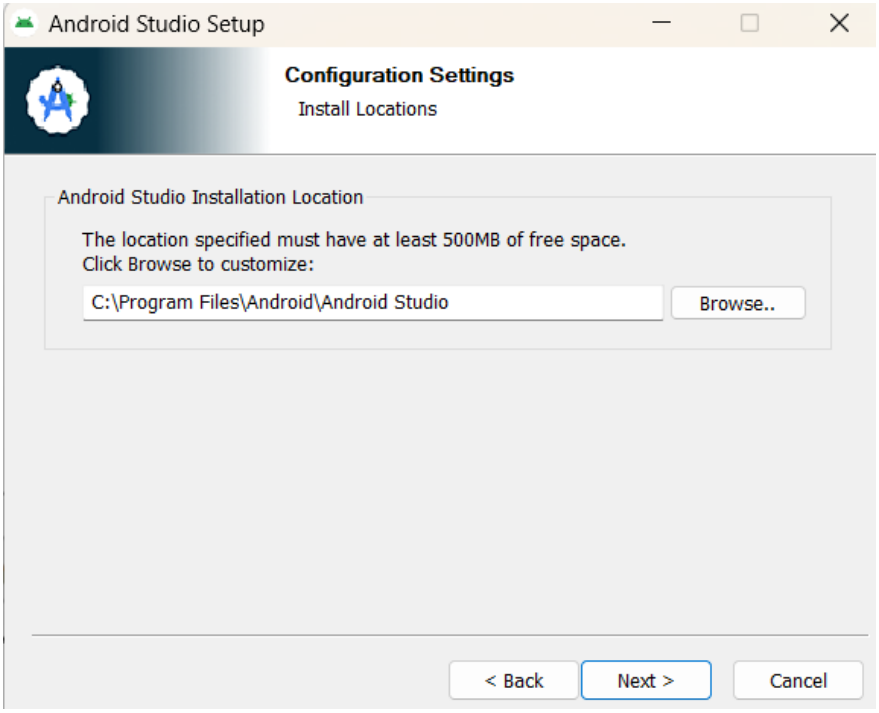
Step 8.1: - When the download is complete, open the .exe file and run it. You will get the following dialog box



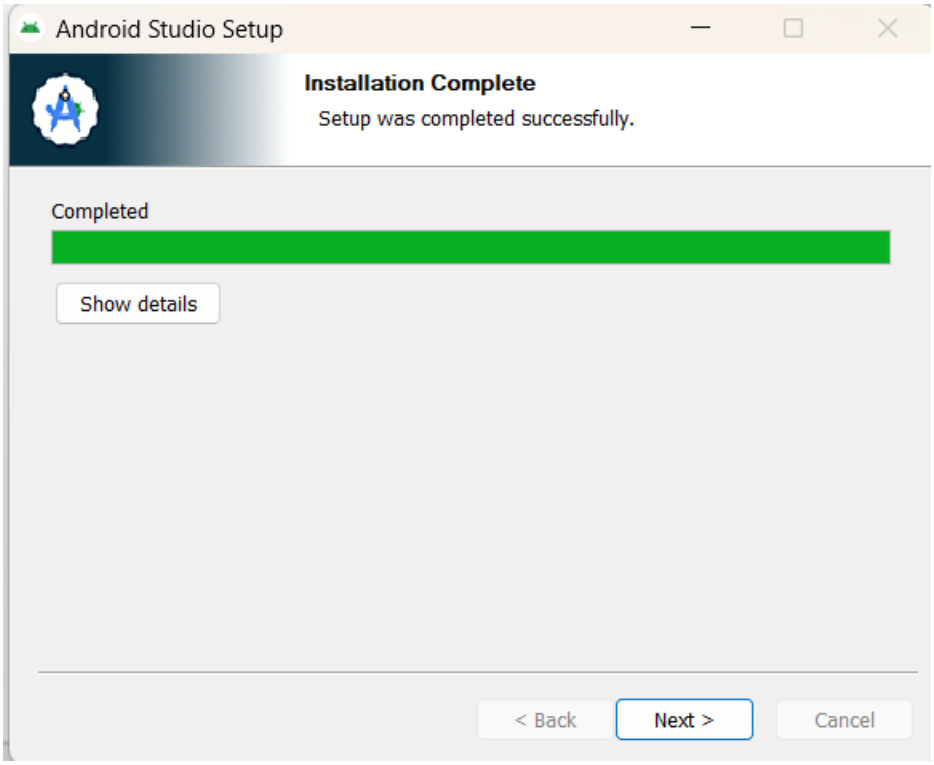
Step 8.2: - Select all the Checkboxes and Click on 'Next' Button.

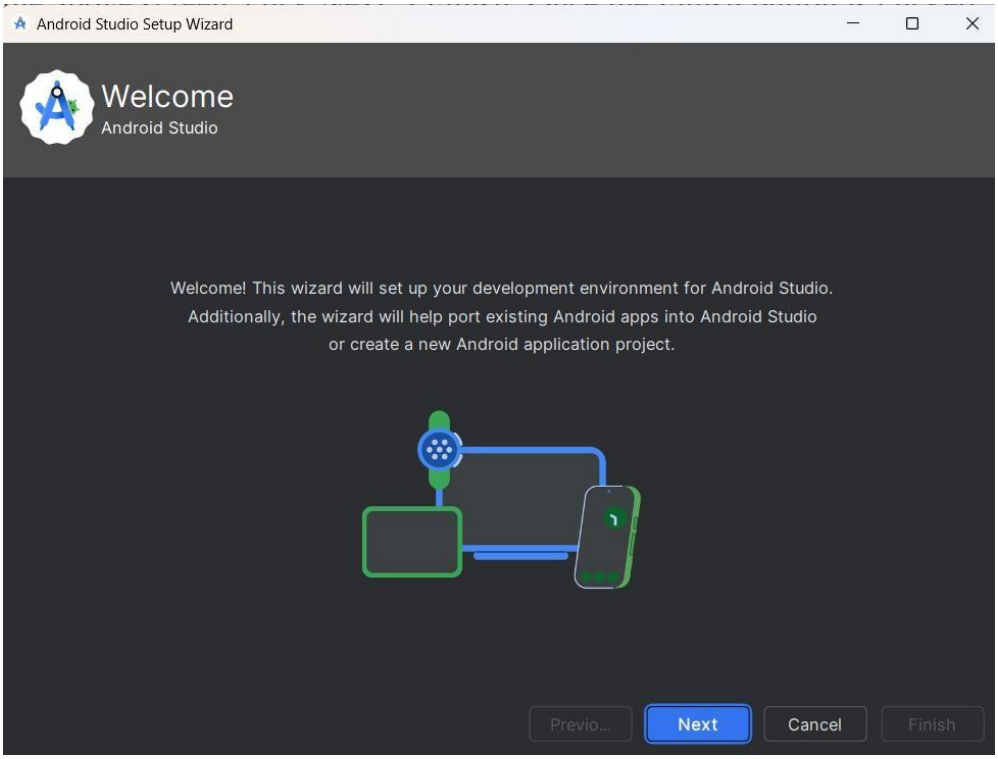


Step 8.3: - Change the destination as per your convenience and click on ‘Next’ Button.



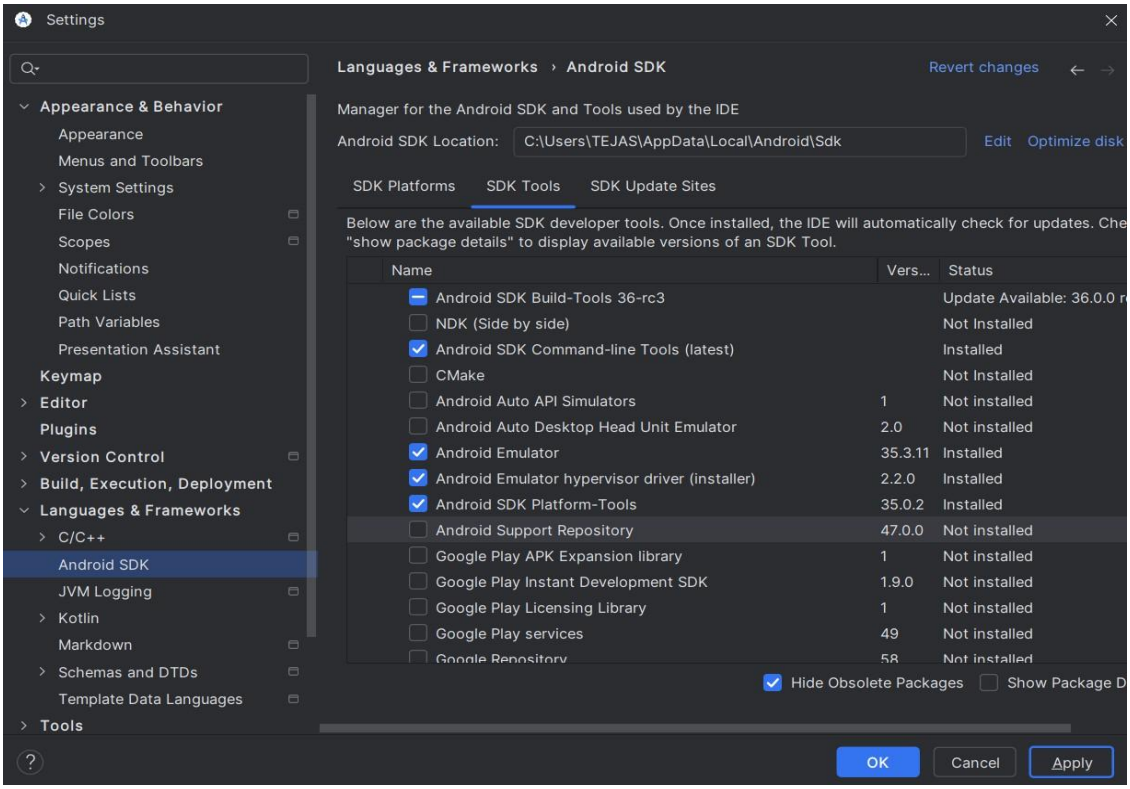
Step 8.4: - Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.



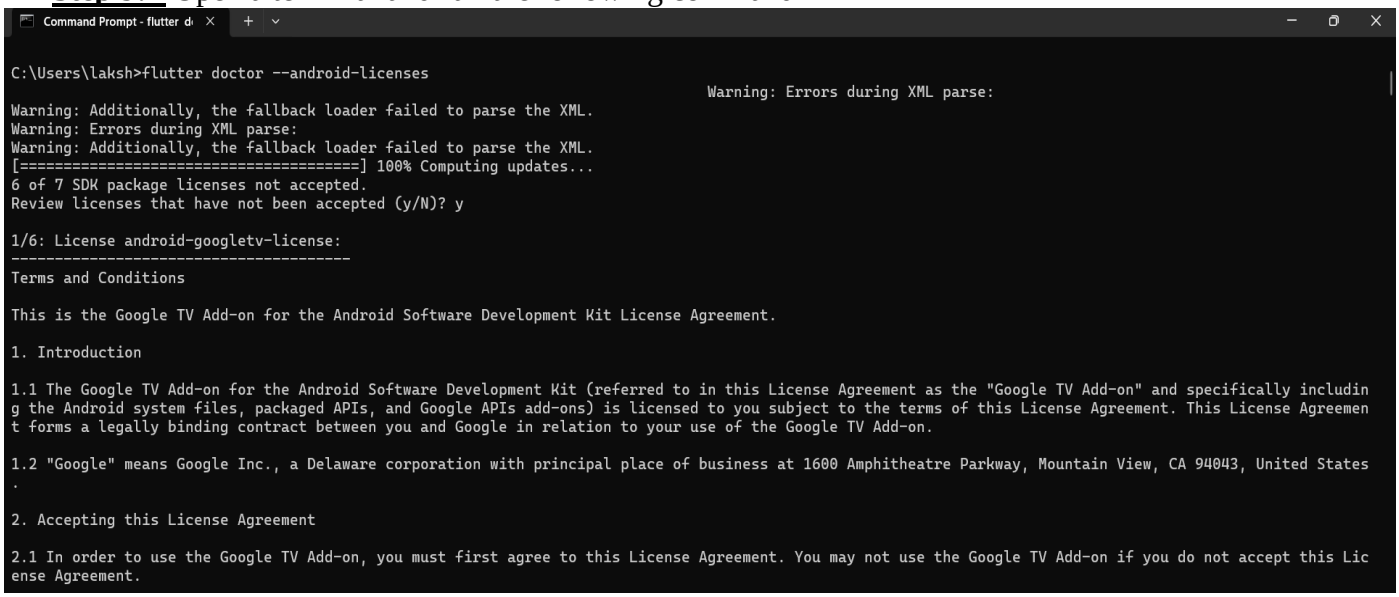


Step 8.5: - Go to Preferences > Appearance & Behavior > System Settings > Android SDK.

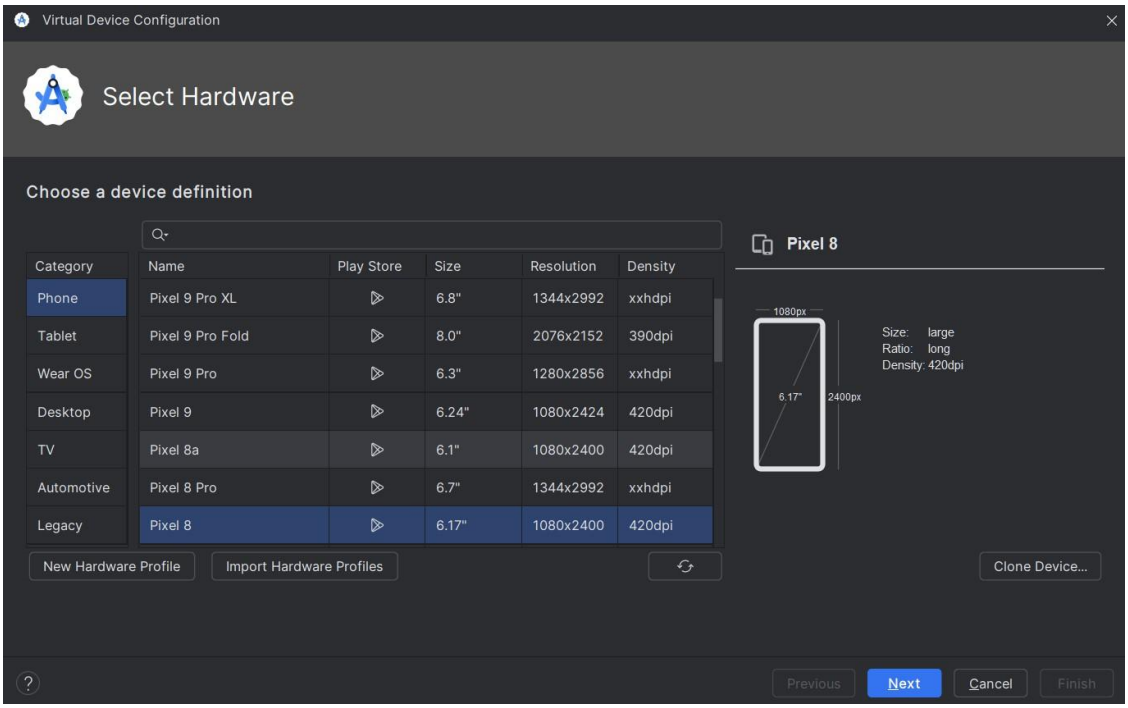
Select the SDK Tools tab and check Android SDK Command-line Tools and Install it.



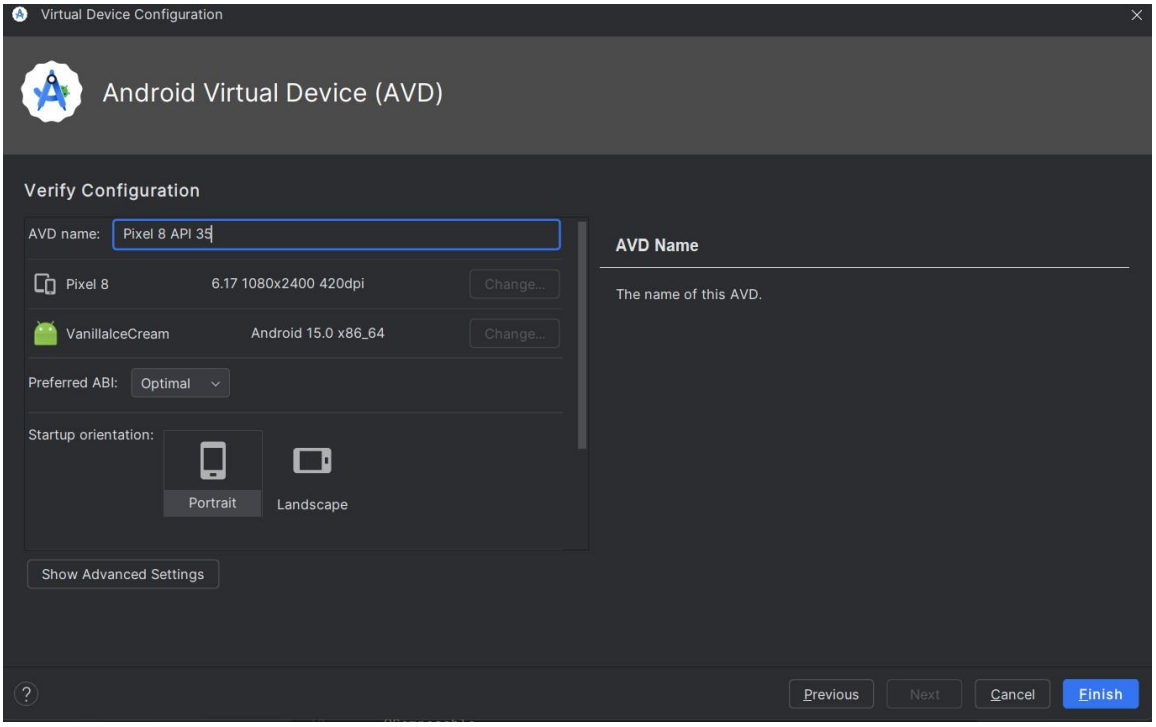
Step 9: - Open a terminal and run the following command



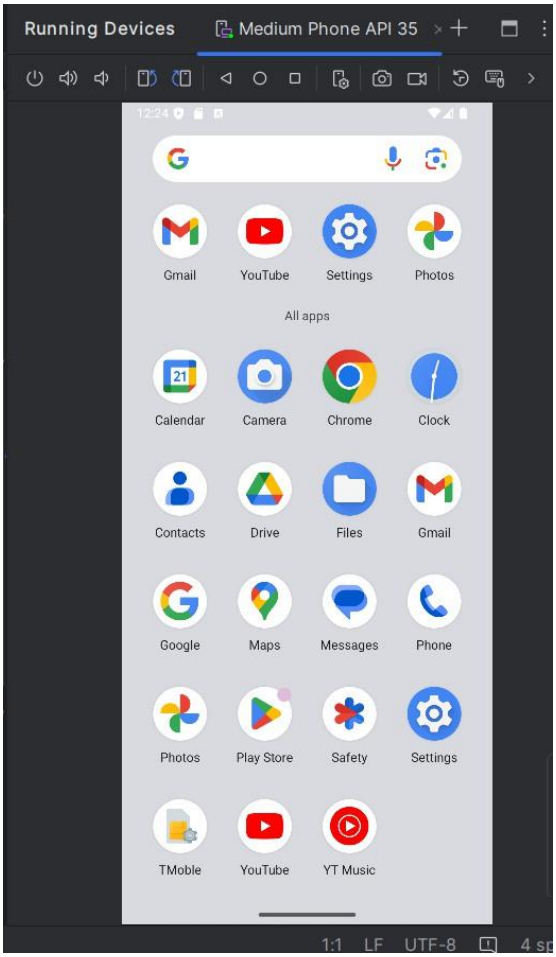
Step 10: - Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application



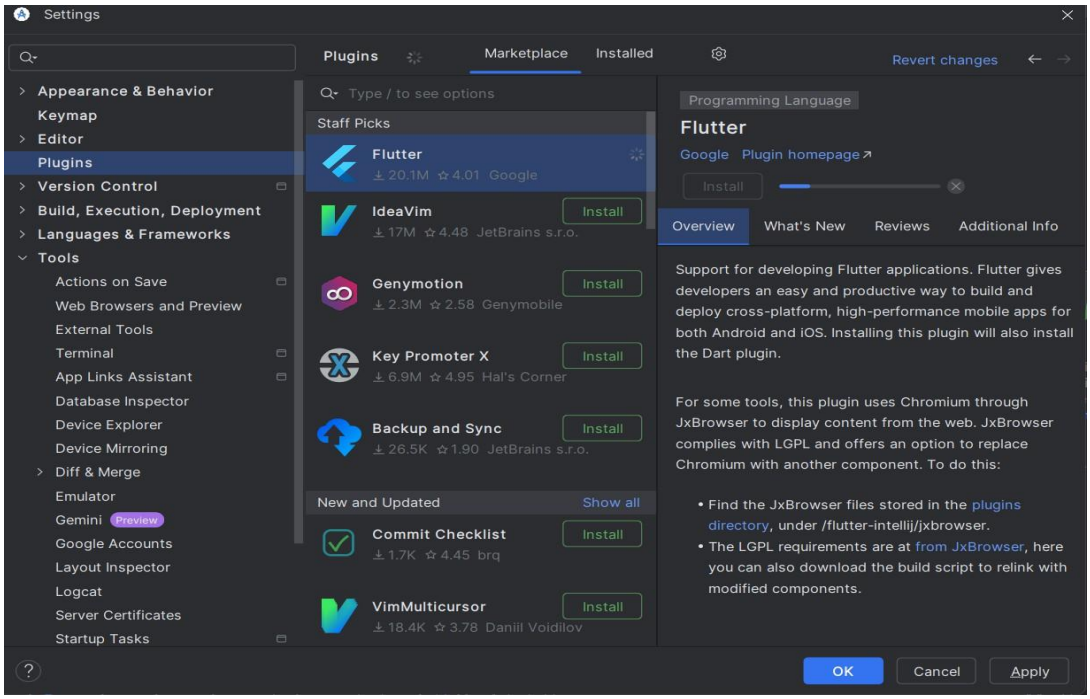
Step 10.1: - Open Android Studio and go to Tools > AVD Manager. Create a new virtual device.



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



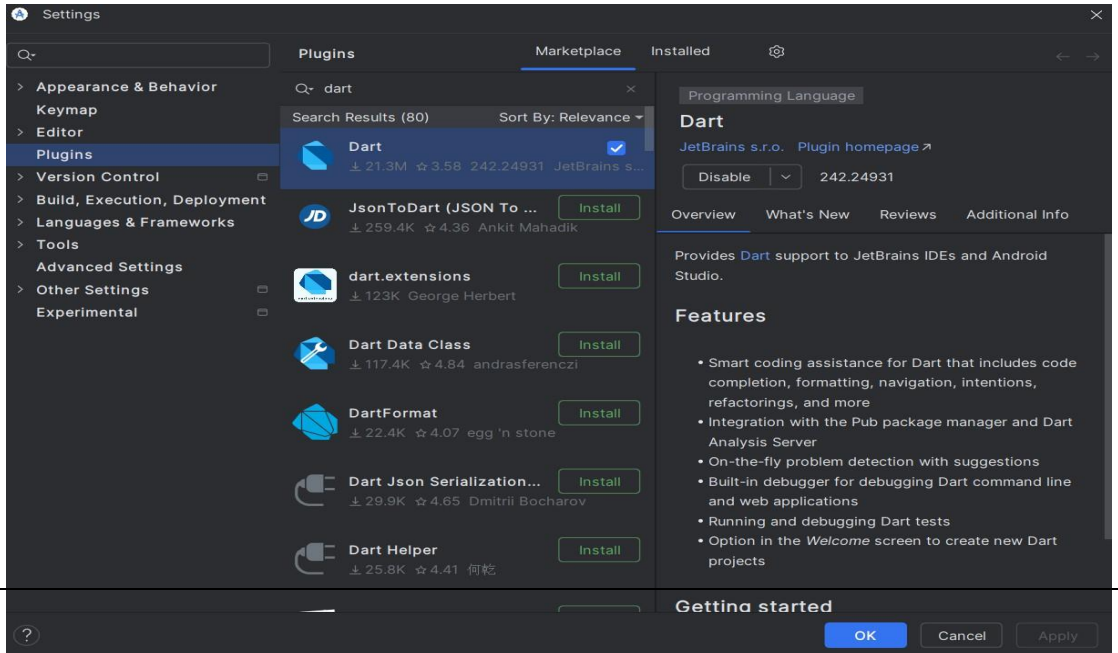
Step 11: - Now, install Flutter and Dart plugin for building Flutter application in Android Studio.



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.

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	57
Name	Laksh Sodhai
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation.
Grade:	

EXPERIMENT NO: - 02

Name:- Laksh Sodhai **Class:-** D15A **Roll:No: -** 57

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';
import 'dart:async';
class HomeScreen extends
StatefulWidget {
  const HomeScreen({super.key});
  @override
  State<HomeScreen> createState()
=> _HomeScreenState();
}
class _HomeScreenState extends
State<HomeScreen> {
  bool isConnected = false;
  String timerText = "00:00:00";
  Timer? _timer;
  int _seconds = 0;
  double downloadSpeed = 0;
  double uploadSpeed = 0;
  int ping = 0;
  void startTimer() {
    _timer = Timer.periodic(const
Duration(seconds: 1), (timer) {
      setState(() {
        _seconds++;
        int hours = _seconds ~/ 3600;
        int minutes = (_seconds % 3600)
~/ 60;
        int seconds = _seconds % 60;
        timerText =
'${hours.toString().padLeft(2,
'0')}:${minutes.toString().padLeft(2,
'0')}:${seconds.toString().padLeft(2,
'0')}';
        downloadSpeed = (300 +
(_seconds % 100)).toDouble();
        uploadSpeed = (150 + (_seconds
% 50)).toDouble();
        ping = 5 + (_seconds % 10);
      });
});
  }
  void stopTimer() {
    _timer?.cancel();
    setState(() {
      _seconds = 0;
      timerText = "00:00:00";
      downloadSpeed = 0;
      uploadSpeed = 0;
      ping = 0;
    });
  }

  void toggleConnection() {
    setState(() {
      isConnected = !isConnected;
      if (isConnected) {
        startTimer();
      } else {
        stopTimer();
      }
    });
  }
```

```
  @override
  void dispose() {
 _timer?.cancel();
   super.dispose();
  }

  @override
  Widget build(BuildContext context)
{
    return Scaffold(
      backgroundColor:
Colors.transparent,
      appBar: _buildAppBar(),
      body: SingleChildScrollView(
        child: Padding(
          padding: const
EdgeInsets.symmetric(horizontal:
20),
          child: Column(
            children: [
              const SizedBox(height: 30),
              _buildConnectionButton(),
              const SizedBox(height: 20),
              _buildConnectionStatus(),
              const SizedBox(height: 20),
              _buildTimer(),
              const SizedBox(height: 30),
              _buildLocationAndPing(),
              const SizedBox(height: 30),
              _buildSpeedMetrics(),
              const SizedBox(height: 30),
              _buildActionButtons(),
            ],
),
          ),
        ),
      ),
    );
  }

  PreferredSizeWidget
_buildAppBar() {
    return AppBar(
      title: ShaderMask(
        shaderCallback: (bounds) =>
const LinearGradient(
          colors: [Colors.white,
Colors.lightBlueAccent],
        ).createShader(bounds),
        child: const Text(
          'SecureShield VPN',
          style: TextStyle(
            fontSize: 24,
            fontWeight: FontWeight.bold,
            color: Colors.white,
          ),
        ),
      ),
      backgroundColor:
Colors.transparent,
      elevation: 0,
      actions: [
        Icon(Icons.notifications, color:
Colors.white.withOpacity(0.7)),
        const SizedBox(width: 16),
        Icon(Icons.info_outline, color:
Colors.white.withOpacity(0.7)),
```

```
const SizedBox(width: 16),
    ],
   );
 }

 Widget _buildConnectionButton() {
  return GestureDetector(
    onTap: toggleConnection,
    child: Container(
      width: 200,
      height: 200,
      decoration: BoxDecoration(
        shape: BoxShape.circle,
        gradient: LinearGradient(
          begin: Alignment.topLeft,
          end: Alignment.bottomRight,
          colors: isConnected
              ? [Colors.green.shade400,
Colors.green.shade700]
              : [Colors.blue.shade400,
Colors.blue.shade700],
        ),
        boxShadow: [
          BoxShadow(
            color: (isConnected ?
Colors.green :
Colors.blue).withOpacity(0.3),
            blurRadius: 20,
            spreadRadius: 5,
          ),
        ],
      ),
      child: Column(
 mainAxisAlignment:
MainAxisAlignment.center,

children: [
          Icon(
            isConnected ?
Icons.power_settings_new :
Icons.power_off,
            size: 50,
            color: Colors.white,
          ),
          const SizedBox(height: 10),
          Text(
            isConnected ? 'Connected' :
'Tap to Connect',
            style: const TextStyle(
              color: Colors.white,
              fontSize: 16,
              fontWeight:
FontWeight.bold,
            ),
          ),
        ],
      ),
    ),
   );
  }

  Widget _buildConnectionStatus() {
   return Container(
     padding: const
EdgeInsets.symmetric(horizontal: 30,
vertical: 12),
```

```
decoration: BoxDecoration(
        gradient: LinearGradient(
          colors: isConnected

? [Colors.green.shade400,
Colors.green.shade700]
              : [Colors.blue.shade400,
Colors.blue.shade700],
        ),
        borderRadius:
BorderRadius.circular(30),
        boxShadow: [
          BoxShadow(
            color: (isConnected ?
Colors.green :
Colors.blue).withOpacity(0.3),
            blurRadius: 10,
            spreadRadius: 2,
          ),
        ],
      ),
      child: Text(
        isConnected ? 'Protected' : 'Not
Protected',
        style: const TextStyle(
          color: Colors.white,
          fontSize: 18,
          fontWeight: FontWeight.bold,
        ),
      ),
    );
  }
```

```
Widget _buildTimer() {
    return Text(
      timerText,
      style: const TextStyle(

fontSize: 48,
        fontWeight: FontWeight.bold,

color: Colors.white,
      ),
    );
  }

  Widget _buildLocationAndPing() {
    return Row(
      mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
      children: [
        _buildMetricCard(
          'United States',
          'US',
          'FREE',
          Colors.blue,
        ),
        _buildMetricCard(
          '$ping ms',
          Icons.speed,
          'PING',
          Colors.orange,
        ),
      ],
    );
  }
```

```
  Widget _buildSpeedMetrics() {
    return Row(
      mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
      children: [
        _buildMetricCard(

'${downloadSpeed.toStringAsFixed(1
)} KB/s',
          Icons.download,
          'DOWNLOAD',
          Colors.green,
  ),
        _buildMetricCard(

'${uploadSpeed.toStringAsFixed(1)}
KB/s',
          Icons.upload,
          'UPLOAD',
          Colors.blue,
        ),
      ],
    );
  }

  Widget _buildMetricCard(String
value, dynamic icon, String label,
Color color) {
    return Container(
      width: 150,
      padding: const EdgeInsets.all(15),
      decoration: BoxDecoration(
        color:
Colors.white.withOpacity(0.1),
        borderRadius:
BorderRadius.circular(20),
        border: Border.all(color:
Colors.white24),
        boxShadow: [
          BoxShadow(

 color: color.withOpacity(0.1),
            blurRadius: 10,

spreadRadius: 2,
          ),
        ],
      ),
      child: Column(
        children: [
          icon is IconData
              ? Icon(icon, color: color,
size: 30)
              : Text(icon, style: const
TextStyle(fontSize: 30)),
          const SizedBox(height: 8),
          Text(
            value,
            style: const TextStyle(
              color: Colors.white,
              fontSize: 16,
              fontWeight:
FontWeight.bold,
            ),
          ),
```

```
          Text(
            label,
            style: TextStyle(
              color:
Colors.white.withOpacity(0.7),
              fontSize: 12,
            ),
          ),
        ],


  ),
    );
  }

  Widget _buildActionButtons() {
    return Column(
      children: [
        _buildGradientButton(
          'Speed Test',
          Icons.speed,
          Colors.purple,
              () =>
Navigator.pushNamed(context,
'/speedtest'),
        ),
        const SizedBox(height: 10),
        _buildGradientButton(
          'Change Location',
          Icons.language,
          Colors.blue,
              () =>
Navigator.pushNamed(context,
'/servers'),
),
      ],
    );
  }

  Widget _buildGradientButton(
      String text,
      IconData icon,
      Color color,
      VoidCallback onTap,


) {
    return Container(
      margin: const
EdgeInsets.only(bottom: 10),
      decoration: BoxDecoration(


 gradient: LinearGradient(
          colors: [color.withOpacity(0.8),
color],
        ),
        borderRadius:
BorderRadius.circular(15),
        boxShadow: [
          BoxShadow(
            color: color.withOpacity(0.3),
            blurRadius: 10,
            spreadRadius: 2,
          ),
        ],
      ),
      child: Material(
        color: Colors.transparent,
```

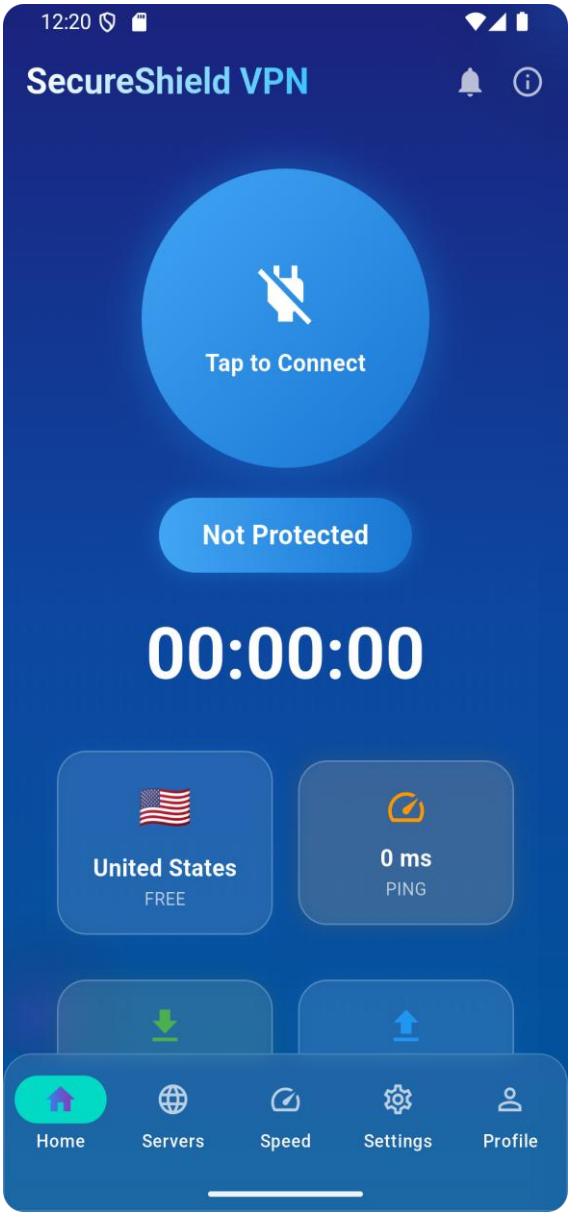
```
      child: InkWell(
        onTap: onTap,
        borderRadius:
BorderRadius.circular(15),
        child: Padding(
          padding: const
EdgeInsets.symmetric(horizontal: 20,
vertical: 15),
          child: Row(
            mainAxisAlignment:
MainAxisAlignment.spaceBetween,

children: [
              Row(
                children: [
                  Icon(icon, color:
Colors.white),
                  const SizedBox(width:


10),
                  Text(
                    text,
                    style: const TextStyle(
                      color: Colors.white,
                      fontSize: 16,
                      fontWeight:
FontWeight.bold,
                    ),
                  ),
                ],
              ),
              const
Icon(Icons.arrow_forward_ios, color:
Colors.white),
                  ],
                ),
              ),
            ),
          ),
        ),
      );
    }
  }
```

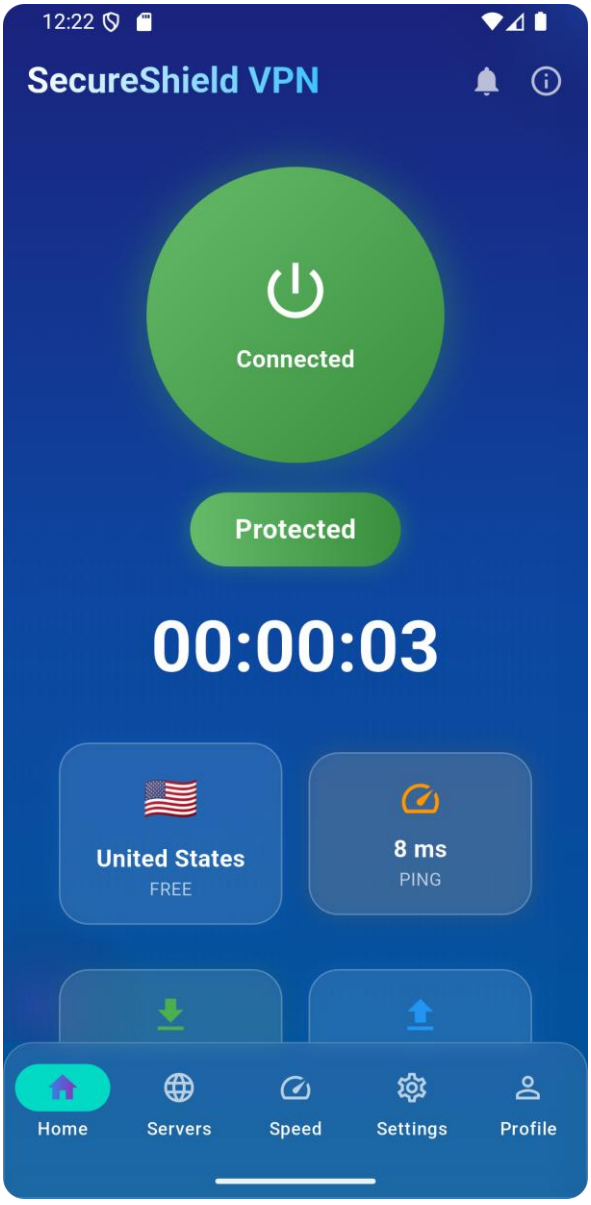
servers_screen.dart

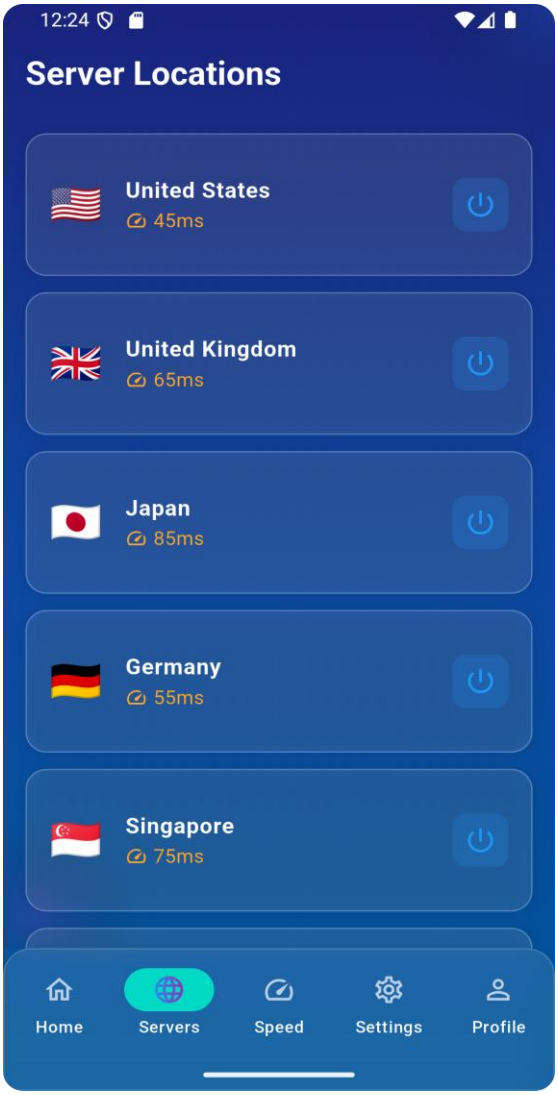
```
import 'package:flutter/material.dart';
import 'dart:math';
import 'dart:async';
class ServersScreen extends StatelessWidget {
  const ServersScreen({super.key});

  @override
  Widget build(BuildContext context) {
    final List<Map<String, dynamic>> servers = [
      {"country": "United States", "flag": "US",
"ping": 45},
      {"country": "United Kingdom", "flag": "GB",
"ping": 65},
      {"country": "Japan", "flag": "JP", "ping":
85},
      {"country": "Germany", "flag": "DE", "ping":
55},
      {"country": "Singapore", "flag": "SG",
"ping": 75},
      {"country": "Canada", "flag": "CA", "ping":
50},
      {"country": "France", "flag": "FR", "ping":
60},
      {"country": "Australia", "flag": "AU", "ping":
90},
      {"country": "Netherlands", "flag": "NL",
"ping": 58},
      {"country": "India", "flag": "IN", "ping":
95},
    ];

    return Scaffold(
      backgroundColor: Colors.transparent,
      appBar: AppBar(
        title: const Text('Server Locations',
          style: TextStyle(fontSize: 24, fontWeight:
FontWeight.bold),
        ),
        backgroundColor: Colors.transparent,
        elevation: 0,
      ),
      body: ListView.builder(
        padding: const EdgeInsets.all(16),
        itemCount: servers.length,
        itemBuilder: (context, index) {
          final server = servers[index];
          return Container(
            margin: const EdgeInsets.only(bottom:
12),
            decoration: BoxDecoration(
              color: Colors.white.withOpacity(0.1),
              borderRadius:
BorderRadius.circular(15),
              border: Border.all(color:
Colors.white24),
              boxShadow: [
                BoxShadow(
                  color: Colors.black.withOpacity(0.1),
                  blurRadius: 10,
                ),
              ],
            ),
            child: ListTile(
              contentPadding: const
EdgeInsets.all(16),
              leading: Text(
                server["flag"].toString(),
                style: const TextStyle(fontSize: 30),
              ),
              title: Text(
                server["country"].toString(),
                style: const TextStyle(
                  color: Colors.white,
                  fontWeight: FontWeight.bold,
                ),
              ),
              subtitle: Row(
                children: [
                  Icon(Icons.speed, size: 16, color:
Colors.orange.shade400),
                  const SizedBox(width: 4),
                  Text(
                    '${server["ping"]}ms',
                    style: TextStyle(color:
Colors.orange.shade400),
                  ),
                ],
              ),
              trailing: Container(
                padding: const EdgeInsets.all(8),
                decoration: BoxDecoration(
                  color: Colors.blue.withOpacity(0.2),
                  borderRadius:
BorderRadius.circular(10),
                ),
                child: const
Icon(Icons.power_settings_new, color:
Colors.blue),
              ),
              onTap: () {

ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(
                  content: Text('Connecting to
```

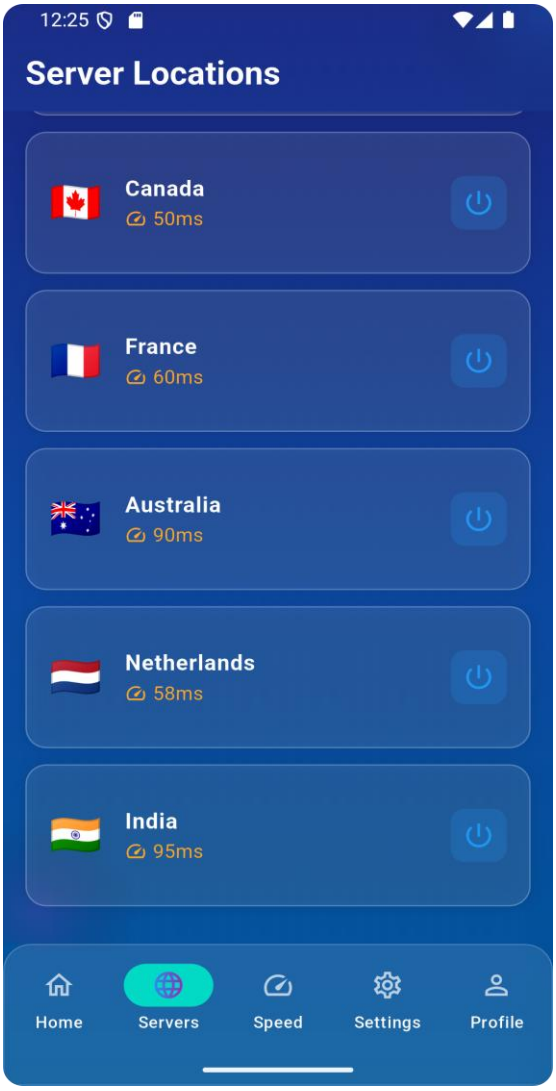
```
                ${server["country"]}...'),
                duration: const Duration(seconds:
1),


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

OUTPUT: -


12:20
SecureShield VPN
Tap to Connect
Not Protected
00:00:00
United States
FREE
0 ms
PING
Home
Servers
Speed
Settings
Profile


12:22
SecureShield VPN
Connected
Protected
00:00:03
United States
FREE
8 ms
PING
Home
Servers
Speed
Settings
Profile


12:24
Server Locations
United States
45ms
United Kingdom
65ms
Japan
85ms
Germany
55ms
Singapore
75ms
Home
Servers
Speed
Settings
Profile


12:25
Server Locations
Canada
50ms
France
60ms
Australia
90ms
Netherlands
58ms
India
95ms
Home
Servers
Speed
Settings
Profile

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app.
Roll No.	57
Name	Laksh Sodhai
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation.
Grade:	

EXPERIMENT NO: - 03

Name:- Laksh Sodhai **Class:-** D15A **Roll:No: -** 57

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: -

Home_screen.dart

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

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

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

class _HomeScreenState extends State<HomeScreen>
{
  bool isConnected = false;
  String timerText = "00:00:00";
  Timer? _timer;
  int _seconds = 0;
  double downloadSpeed = 0;
  double uploadSpeed = 0;
  int ping = 0;

  void startTimer() {
    _timer = Timer.periodic(const Duration(seconds: 1),
(timer) {
      setState(() {
        _seconds++;
        int hours = _seconds ~/ 3600;
        int minutes = (_seconds % 3600) ~/ 60;
        int seconds = _seconds % 60;
        timerText = '${hours.toString().padLeft(2,
'0')}:${minutes.toString().padLeft(2,
'0')}:${seconds.toString().padLeft(2, '0')}';
        downloadSpeed = (300 + (_seconds %
100)).toDouble();
        uploadSpeed = (150 + (_seconds %
50)).toDouble();
        ping = 5 + (_seconds % 10);
      });
    });
  }

  void stopTimer() {
    _timer?.cancel();
    setState(() {
      _seconds = 0;
      timerText = "00:00:00";
      downloadSpeed = 0;
      uploadSpeed = 0;
      ping = 0;
    });
  }

  void toggleConnection() {
    setState(() {
      isConnected = !isConnected;
      if (isConnected) {
        startTimer();
      } else {
        stopTimer();
      }
    });
  }

  @override
  void dispose() {
    _timer?.cancel();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.transparent,
      appBar: _buildAppBar(),
      body: SingleChildScrollView(
        child: Padding(
          padding: const
EdgeInsets.symmetric(horizontal: 20),
          child: Column(
            children: [
              const SizedBox(height: 30),
              _buildConnectionButton(),
              const SizedBox(height: 20),
              _buildConnectionStatus(),
              const SizedBox(height: 20),
              _buildTimer(),
              const SizedBox(height: 30),
              _buildLocationAndPing(),
              const SizedBox(height: 30),
              _buildSpeedMetrics(),
              const SizedBox(height: 30),
              _buildActionButtons(),
            ],
          ),
        ),
      ),
    );
  }

  PreferredSizeWidget _buildAppBar() {
    return AppBar(
      title: ShaderMask(
        shaderCallback: (bounds) => const
LinearGradient(
          colors: [Colors.white, Colors.lightBlueAccent],
        ).createShader(bounds),
```

```
  child: const Text(
        'SecureShield VPN',
        style: TextStyle(
          fontSize: 24,
          fontWeight: FontWeight.bold,
          color: Colors.white,
        ),
      ),
    ),
    backgroundColor: Colors.transparent,
    elevation: 0,
    actions: [
      Icon(Icons.notifications, color:
Colors.white.withOpacity(0.7)),
      const SizedBox(width: 16),
      Icon(Icons.info_outline, color:
Colors.white.withOpacity(0.7)),
      const SizedBox(width: 16),
    ],
  );
 }

 Widget _buildConnectionButton() {
  return GestureDetector(
    onTap: toggleConnection,
    child: Container(
      width: 200,
      height: 200,
      decoration: BoxDecoration(
        shape: BoxShape.circle,
        gradient: LinearGradient(
          begin: Alignment.topLeft,
          end: Alignment.bottomRight,
          colors: isConnected
              ? [Colors.green.shade400,
Colors.green.shade700]
              : [Colors.blue.shade400,
Colors.blue.shade700],
        ),
        boxShadow: [
          BoxShadow(
            color: (isConnected ? Colors.green :
Colors.blue).withOpacity(0.3),
            blurRadius: 20,
            spreadRadius: 5,
          ),
        ],
      ),
      child: Column(
        mainAxisAlignment:
MainAxisAlignment.center,
        children: [
          Icon(
            isConnected ? Icons.power_settings_new :
Icons.power_off,
size: 50,
            color: Colors.white,
          ),
          const SizedBox(height: 10),
          Text(
            isConnected ? 'Connected' : 'Tap to Connect',
            style: const TextStyle(
              color: Colors.white,
              fontSize: 16,
              fontWeight: FontWeight.bold,
            ),
          ),
        ],
      ),
    ),
  );
 }

 Widget _buildConnectionStatus() {
  return Container(
    padding: const EdgeInsets.symmetric(horizontal:
30, vertical: 12),
    decoration: BoxDecoration(
      gradient: LinearGradient(
        colors: isConnected
            ? [Colors.green.shade400,
Colors.green.shade700]
            : [Colors.blue.shade400,
Colors.blue.shade700],
      ),
      borderRadius: BorderRadius.circular(30),
      boxShadow: [
        BoxShadow(
          color: (isConnected ? Colors.green :
Colors.blue).withOpacity(0.3),
          blurRadius: 10,
          spreadRadius: 2,
        ),
      ],
    ),
    child: Text(
      isConnected ? 'Protected' : 'Not Protected',
      style: const TextStyle(
        color: Colors.white,
        fontSize: 18,
        fontWeight: FontWeight.bold,
      ),
    ),
  );
 }

 Widget _buildTimer() {
  return Text(
    timerText,
    style: const TextStyle(
```

```
 fontSize: 48,
       fontWeight: FontWeight.bold,
       color: Colors.white,
     ),
   );
 }

 Widget _buildLocationAndPing() {
   return Row(
     mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
     children: [
       _buildMetricCard(
         'United States',
         'US',
         'FREE',
         Colors.blue,
       ),
       _buildMetricCard(
         '$ping ms',
         Icons.speed,
         'PING',
         Colors.orange,
       ),
     ],
   );
 }

 Widget _buildSpeedMetrics() {
   return Row(
     mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
     children: [
       _buildMetricCard(
         '${downloadSpeed.toStringAsFixed(1)} KB/s',
         Icons.download,
         'DOWNLOAD',
         Colors.green,
       ),
       _buildMetricCard(
         '${uploadSpeed.toStringAsFixed(1)} KB/s',
         Icons.upload,
         'UPLOAD',
         Colors.blue,
       ),
     ],
   );
 }

 Widget _buildMetricCard(String value, dynamic
icon, String label, Color color) {
   return Container(
     width: 150,
     padding: const EdgeInsets.all(15),
     decoration: BoxDecoration(
       color: Colors.white.withOpacity(0.1),
       borderRadius: BorderRadius.circular(20),
       border: Border.all(color: Colors.white24),
       boxShadow: [
         BoxShadow(
           color: color.withOpacity(0.1),
           blurRadius: 10,
           spreadRadius: 2,
         ),
       ],
     ),
     child: Column(
       children: [
         icon is IconData
             ? Icon(icon, color: color, size: 30)
             : Text(icon, style: const TextStyle(fontSize:
30)),
         const SizedBox(height: 8),
         Text(
           value,
           style: const TextStyle(
             color: Colors.white,
             fontSize: 16,
             fontWeight: FontWeight.bold,
           ),
         ),
         Text(
           label,
           style: TextStyle(
             color: Colors.white.withOpacity(0.7),
             fontSize: 12,
           ),
         ),
       ],
     ),
   );
 }

 Widget _buildActionButtons() {
   return Column(
     children: [
       _buildGradientButton(
         'Speed Test',
         Icons.speed,
         Colors.purple,
             () => Navigator.pushNamed(context,
'/speedtest'),
       ),
       const SizedBox(height: 10),
       _buildGradientButton(
         'Change Location',
         Icons.language,
         Colors.blue,
             () => Navigator.pushNamed(context,
'/servers'),
```

```
),
    ],
  );
 }

 Widget _buildGradientButton(
    String text,
    IconData icon,
    Color color,
    VoidCallback onTap,
    ) {
  return Container(
    margin: const EdgeInsets.only(bottom: 10),
    decoration: BoxDecoration(
      gradient: LinearGradient(
        colors: [color.withOpacity(0.8), color],
      ),
      borderRadius: BorderRadius.circular(15),
      boxShadow: [
        BoxShadow(
          color: color.withOpacity(0.3),
          blurRadius: 10,
          spreadRadius: 2,
        ),
      ],
    ),
    child: Material(
      color: Colors.transparent,
      child: InkWell(
        onTap: onTap,
        borderRadius: BorderRadius.circular(15),
```

Servers_screen.dart

```
import 'package:flutter/material.dart';
import 'dart:math';
import 'dart:async';
class ServersScreen extends StatelessWidget {
 const ServersScreen({super.key});

 @override
 Widget build(BuildContext context) {
  final List<Map<String, dynamic>> servers = [
    {"country": "United States", "flag": "US", "ping":
45},
    {"country": "United Kingdom", "flag": "GB",
"ping": 65},
    {"country": "Japan", "flag": "JP", "ping": 85},
    {"country": "Germany", "flag": "DE", "ping":
55},
    {"country": "Singapore", "flag": "SG", "ping":
75},
    {"country": "Canada", "flag": "CA", "ping": 50},
    {"country": "France", "flag": "FR", "ping": 60},
    {"country": "Australia", "flag": "AU", "ping":
90},
    {"country": "Netherlands", "flag": "NL", "ping": 58},
    {"country": "India", "flag": "IN", "ping": 95},
  ];

  return Scaffold(
    backgroundColor: Colors.transparent,
    appBar: AppBar(
      title: const Text('Server Locations',
        style: TextStyle(fontSize: 24, fontWeight:
FontWeight.bold),
      ),
      backgroundColor: Colors.transparent,
      elevation: 0,
    ),
    body: ListView.builder(
      padding: const EdgeInsets.all(16),
      itemCount: servers.length,
      itemBuilder: (context, index) {
        final server = servers[index];
        return Container(
          margin: const EdgeInsets.only(bottom: 12),
          decoration: BoxDecoration(
            color: Colors.white.withOpacity(0.1),
            child: Padding(
              padding: const
EdgeInsets.symmetric(horizontal: 20, vertical: 15),
              child: Row(
                mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                children: [
                  Row(
                    children: [
                      Icon(icon, color: Colors.white),
                      const SizedBox(width: 10),
                      Text(
                        text,
                        style: const TextStyle(
                          color: Colors.white,
                          fontSize: 16,
                          fontWeight: FontWeight.bold,
                        ),
                      ),
                    ],
                  ),
                  const Icon(Icons.arrow_forward_ios, color:
Colors.white),
                ],
              ),
            ),
          ),
        );
      }
```

```
            borderRadius: BorderRadius.circular(15),          },
            border: Border.all(color: Colors.white24),              ),
            boxShadow: [                                          );
              BoxShadow(                                         }
                color: Colors.black.withOpacity(0.1),           }
                blurRadius: 10,
              ),
            ],
          ),
          child: ListTile(
            contentPadding: const EdgeInsets.all(16),
            leading: Text(
              server["flag"].toString(),
              style: const TextStyle(fontSize: 30),
            ),
            title: Text(
              server["country"].toString(),
              style: const TextStyle(
                color: Colors.white,
                fontWeight: FontWeight.bold,
              ),
            ),
            subtitle: Row(
              children: [
                Icon(Icons.speed, size: 16, color:
Colors.orange.shade400),
                const SizedBox(width: 4),
                Text(
                  '${server["ping"]}ms',
                  style: TextStyle(color:
Colors.orange.shade400),
                ),
              ],
            ),
            trailing: Container(
              padding: const EdgeInsets.all(8),
              decoration: BoxDecoration(
                color: Colors.blue.withOpacity(0.2),
                borderRadius: BorderRadius.circular(10),
              ),
              child: const
Icon(Icons.power_settings_new, color: Colors.blue),
            ),
            onTap: () {

ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(
                  content: Text('Connecting to
${server["country"]}...'),
                  duration: const Duration(seconds: 1),
                ),
              );
            },
          ),
        );
```

Vpn_details_screen.dart

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

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: const Color(0xFF1A237E),
      appBar: AppBar(
        backgroundColor: Colors.transparent,
        elevation: 0,
        title: const Text('VPN Details'),
        leading: IconButton(
          icon: const Icon(Icons.arrow_back),
          onPressed: () => Navigator.pop(context),
        ),
      ),
      body: SingleChildScrollView(
        padding: const EdgeInsets.all(16),
        child: Column(
          children: [
            // Connection Details Card
            Container(
              padding: const EdgeInsets.all(16),
              decoration: BoxDecoration(
                color: Colors.white.withOpacity(0.1),
                borderRadius: BorderRadius.circular(12),
              ),
              child: Column(
                crossAxisAlignment:
CrossAxisAlignment.start,
                children: [
                  const Text(
                    'Connection Details',
                    style: TextStyle(
                      color: Colors.white,
                      fontSize: 20,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                  const SizedBox(height: 16),
                  _buildDetailRow('IP Address',
'192.168.1.1'),
                  _buildDetailRow('Protocol', 'UDP'),
                  _buildDetailRow('Port', '1194'),
                  _buildDetailRow('Encryption', 'AES-256-
GCM'),
                ],
              ),
            ),
            const SizedBox(height: 16),

            // Server Stats Card
            Container(
              padding: const EdgeInsets.all(16),
              decoration: BoxDecoration(
                color: Colors.white.withOpacity(0.1),
                borderRadius: BorderRadius.circular(12),
              ),
              child: Column(
                crossAxisAlignment:
CrossAxisAlignment.start,
                children: [
                  const Text(
                    'Server Stats',
                    style: TextStyle(
                      color: Colors.white,
                      fontSize: 20,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                  const SizedBox(height: 16),
                  _buildStatRow(
                    icon: Icons.speed,
                    title: 'Server Load',
                    value: '45%',
                    color: Colors.orange,
                  ),
                  _buildStatRow(
                    icon: Icons.people,
                    title: 'Active Users',
                    value: '1,234',
                    color: Colors.blue,
                  ),
                  _buildStatRow(
                    icon: Icons.network_check,
                    title: 'Uptime',
                    value: '99.9%',
                    color: Colors.green,
                  ),
                ],
              ),
            ),
            const SizedBox(height: 16),

            // Network Info Card
            Container(
              padding: const EdgeInsets.all(16),
              decoration: BoxDecoration(
                color: Colors.white.withOpacity(0.1),
                borderRadius: BorderRadius.circular(12),
              ),
              child: Column(
                crossAxisAlignment:
CrossAxisAlignment.start,
                children: [
                  const Text(
```

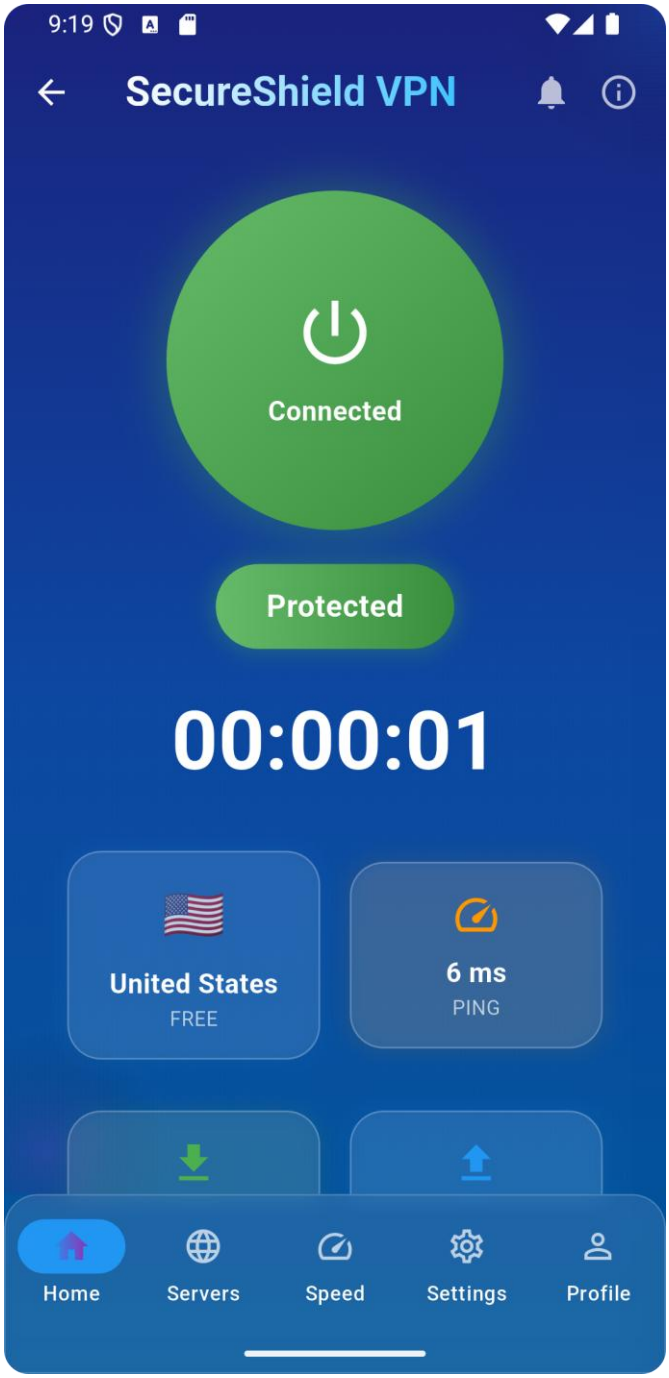
```
'Network Information',
                style: TextStyle(
                  color: Colors.white,
                  fontSize: 20,
                  fontWeight: FontWeight.bold,
                ),
              ),
              const SizedBox(height: 16),
              _buildNetworkInfo(
                'Download Speed',
                '120 Mbps',
                Icons.download,
                Colors.green,
              ),
              _buildNetworkInfo(
                'Upload Speed',
                '85 Mbps',
                Icons.upload,
                Colors.blue,
              ),
              _buildNetworkInfo(
                'Latency',
                '45 ms',
                Icons.timer,
                Colors.orange,
              ),
            ],
          ),
        ),
        const SizedBox(height: 16),

        // Usage Stats Card
        Container(
          padding: const EdgeInsets.all(16),
          decoration: BoxDecoration(
            color: Colors.white.withOpacity(0.1),
            borderRadius: BorderRadius.circular(12),
          ),
          child: Column(
            crossAxisAlignment:
CrossAxisAlignment.start,
            children: [
              const Text(
                'Usage Statistics',
                style: TextStyle(
                  color: Colors.white,
                  fontSize: 20,
                  fontWeight: FontWeight.bold,
                ),
              ),
              const SizedBox(height: 16),
              _buildUsageStats('Data Used', '1.5 GB'),
              _buildUsageStats('Time Connected', '2h
30m'),
              _buildUsageStats('Sessions Today', '3'),
            ],
          ),
        ),
      ],
    ),
  );
}

Widget _buildDetailRow(String label, String value) {
  return Padding(
    padding: const EdgeInsets.symmetric(vertical: 8),
    child: Row(
      mainAxisAlignment:
MainAxisAlignment.spaceBetween,
      children: [
        Text(
          label,
          style: const TextStyle(color: Colors.white70),
        ),
        Text(
          value,
          style: const TextStyle(color: Colors.white),
        ),
      ],
    ),
  );
}

Widget _buildStatRow({
  required IconData icon,
  required String title,
  required String value,
  required Color color,
}) {
  return Padding(
    padding: const EdgeInsets.symmetric(vertical: 8),
    child: Row(
      children: [
        Container(
          padding: const EdgeInsets.all(8),
          decoration: BoxDecoration(
            color: color.withOpacity(0.2),
            borderRadius: BorderRadius.circular(8),
          ),
          child: Icon(icon, color: color, size: 20),
        ),
        const SizedBox(width: 12),
        Text(
          title,
          style: const TextStyle(color: Colors.white70),
        ),
        const Spacer(),
        Text(
```

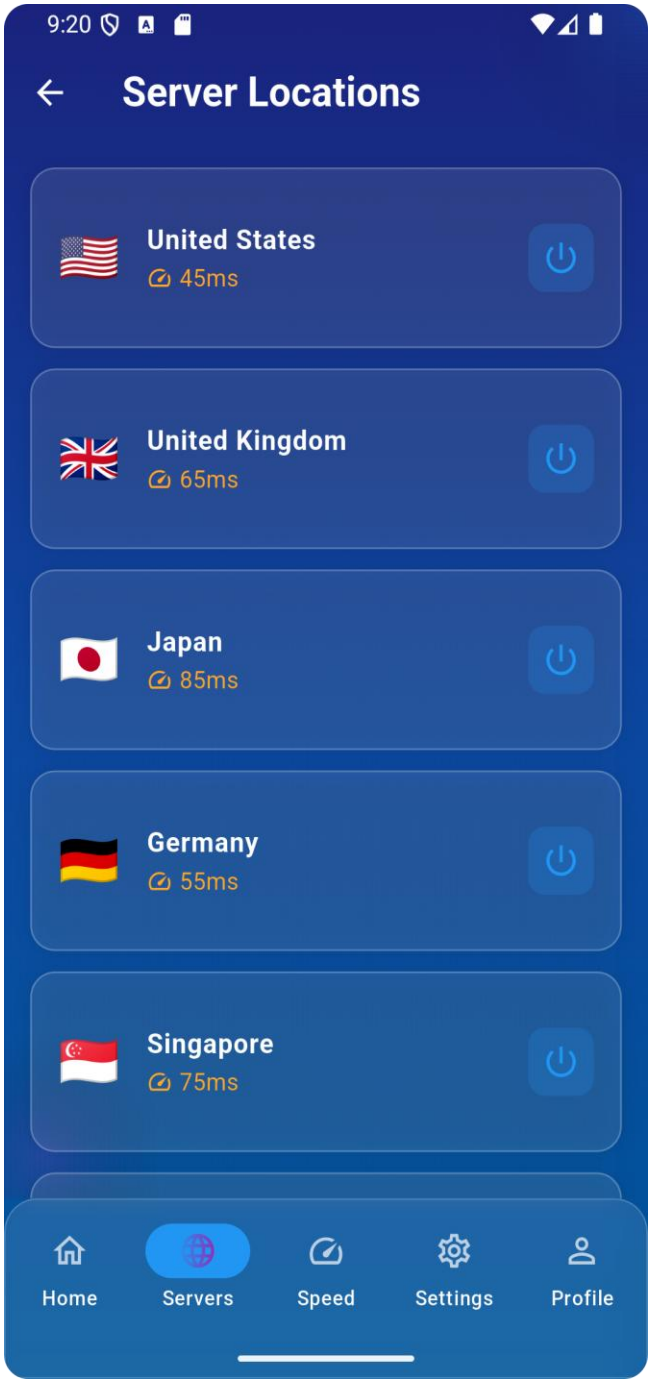
```
 value,
          style: const TextStyle(

 color: Colors.white,
            fontWeight: FontWeight.bold,
          ),
        ),
      ],
    ),
  );
 }

  Widget _buildNetworkInfo(String title, String value,
IconData icon, Color color) {
   return Padding(
     padding: const EdgeInsets.symmetric(vertical: 8),
     child: Row(
       children: [
         Icon(icon, color: color),
         const SizedBox(width: 12),
         Column(
           crossAxisAlignment:
CrossAxisAlignment.start,
           children: [
             Text(
               title,
               style: const TextStyle(color:
Colors.white70),
             ),
             Text(
               value,
               style: const TextStyle(
                 color: Colors.white,
                 fontWeight: FontWeight.bold,
               ),
             ),
           ],
         ),
       ],
     ),
   );
 }

  Widget _buildUsageStats(String label, String value) {
   return Padding(
     padding: const EdgeInsets.symmetric(vertical: 8),
     child: Row(
       mainAxisAlignment:
MainAxisAlignment.spaceBetween,
       children: [
         Text(
           label,
           style: const TextStyle(color: Colors.white70),
         ),
         Text(
```

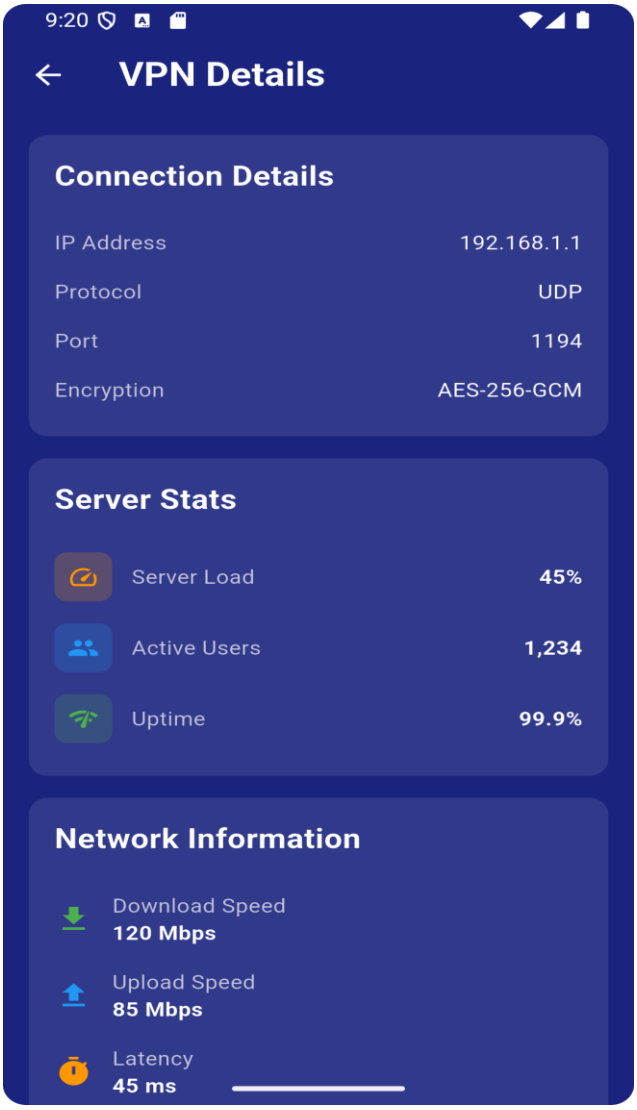
```
 value,
          style: const TextStyle(
            color: Colors.white,
            fontWeight: FontWeight.bold,
          ),

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

OUTPUT: -

v


9:19
SecureShield VPN
Connected
Protected
00:00:01
United States
FREE
6 ms
PING
Home
Servers
Speed
Settings
Profile


9:20
Server Locations
United States
45ms
United Kingdom
65ms
Japan
85ms
Germany
55ms
Singapore
75ms
Home
Servers
Speed
Settings
Profile


9:20
VPN Details
Connection Details
IP Address
192.168.1.1
Protocol
UDP
Port
1194
Encryption
AES-256-GCM
Server Stats
Server Load
45%
Active Users
1,234
Uptime
99.9%
Network Information
Download Speed
120 Mbps
Upload Speed
85 Mbps
Latency
45 ms

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	57
Name	Laksh Sodhai
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation.
Grade:	

`

EXPERIMENT NO: - 04

Name:- Laksh Sodhai **Class:-** D15A **Roll:No: -** 57

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 TextFormField and other input elements to manage validation, state handling, and error messages efficiently. By using form validation techniques, developers can ensure data accuracy and enhance user experience.

When you create a form, it is necessary to provide the GlobalKey. This key uniquely identifies the form and allows you to do any validation in the form fields. The form widget uses child widget TextFormField to provide the users to enter the text field. This widget renders a material design text field and also allows us to display validation errors when they occur.

Creation of a Form

- 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_screen.dart

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

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

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

class _LoginScreenState extends State<LoginScreen> {
  final _formKey = GlobalKey<FormState>();
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();
  bool _isPasswordVisible = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        decoration: const BoxDecoration(
          gradient: LinearGradient(
          begin: Alignment.topCenter,
          end: Alignment.bottomCenter,
          colors: [
            Color(0xFF1A237E),
            Color(0xFF0D47A1),
            Color(0xFF01579B),
          ],
          ),
        ),
        child: Stack(
          children: [
          Positioned(
            top: -100,
            right: -100,
            child: Container(
            width: 200,
              height: 200,
              decoration: BoxDecoration(
                shape: BoxShape.circle,
                color: Colors.white.withOpacity(0.1),
              ),
            ),
          ),
          Positioned(
            bottom: -50,
            left: -50,
            child: Container(
              width: 150,
              height: 150,
              decoration: BoxDecoration(
                shape: BoxShape.circle,
                color: Colors.white.withOpacity(0.1),
              ),
            ),
          ),
          SingleChildScrollView(
            child: Padding(
              padding: const EdgeInsets.all(20.0),
              child: Form(
                key: _formKey,
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: [
                    const SizedBox(height: 80),
                    Container(
                      padding: const EdgeInsets.all(20),
                      decoration: BoxDecoration(
                        borderRadius: BorderRadius.circular(30),
                        boxShadow: [
                          BoxShadow(
                            color: Colors.blue.withOpacity(0.2),
                            blurRadius: 20,
                            spreadRadius: 5,
                          ),
                        ],
                      ),
                      child: Column(
                        children: [
                        Container(
                          padding: const EdgeInsets.all(16),
                          decoration: BoxDecoration(
                            color: Colors.white,
                            shape: BoxShape.circle,
                            boxShadow: [
                            BoxShadow(
                              color: Colors.blue.withOpacity(0.5),
                              blurRadius: 20,
```

```
                    spreadRadius: 5,
                  ),
                ],
              ),
              child: const Icon(
                Icons.shield,
                size: 80,
                color: Colors.blue,
              ),
            ),
            const SizedBox(height: 20),
            ShaderMask(
              shaderCallback: (bounds) =>
const LinearGradient(
                colors: [Colors.white,
Colors.lightBlueAccent],
              ).createShader(bounds),
              child: const Text(
              'SecureShield VPN',
              style: TextStyle(
                fontSize: 32,
                fontWeight:
FontWeight.bold,
                color: Colors.white,
              ),
            ),
          ),
          const Text(
            'Secure • Fast • Reliable',
            style: TextStyle(
              color: Colors.white70,
              fontSize: 16,
            ),
          ),
        ],
      ),
    ),
    const SizedBox(height: 40),
    Container(
      decoration: BoxDecoration(
        color: Colors.white10,
        borderRadius:
BorderRadius.circular(15),
        border: Border.all(color:
Colors.white24),
      ),
      child: TextFormField(
        controller: _emailController,
        style: const TextStyle(color:
Colors.white),
        decoration: InputDecoration(
          labelText: 'Email',
          labelStyle: const
TextStyle(color: Colors.white70),
          border: OutlineInputBorder(
            borderRadius:
BorderRadius.circular(15),
            borderSide: BorderSide.none,
          ),
          prefixIcon: const
Icon(Icons.email, color: Colors.white70),
        ),
        validator: (value) {
          if (value?.isEmpty ?? true) {
            return 'Please enter email';
          }
          return null;
        },
      ),
    ),
    const SizedBox(height: 20),
    Container(
      decoration: BoxDecoration(
        color: Colors.white10,
        borderRadius:
BorderRadius.circular(15),
        border: Border.all(color:
Colors.white24),
      ),
      child: TextFormField(
        controller: _passwordController,
        style: const TextStyle(color:
Colors.white),
        obscureText:
!_isPasswordVisible,
        decoration: InputDecoration(
          labelText: 'Password',
          labelStyle: const
TextStyle(color: Colors.white70),
          border: OutlineInputBorder(
            borderRadius:
BorderRadius.circular(15),
            borderSide: BorderSide.none,
          ),
          prefixIcon: const
Icon(Icons.lock, color: Colors.white70),
          suffixIcon: IconButton(
            icon: Icon(
              _isPasswordVisible
                ? Icons.visibility
                : Icons.visibility_off,
              color: Colors.white70,
            ),
            onPressed: () {
```

```
                    setState(() {
                      _isPasswordVisible = !_isPasswordVisible;
                    });
                  },
                ),
              ),
              validator: (value) {
                if (value?.isEmpty ?? true) {
                  return 'Please enter password';
                }
                return null;
              },
            ),
          ),
          const SizedBox(height: 10),
          Align(
            alignment: Alignment.centerRight,
            child: TextButton(
              onPressed: () {
                // Add forgot password functionality
              },
              child: const Text(
                'Forgot Password?',
                style: TextStyle(color: Colors.white70),
              ),
            ),
          ),
          const SizedBox(height: 20),
          Container(
            decoration: BoxDecoration(
              gradient: const LinearGradient(
                colors: [Colors.blue, Colors.lightBlue],
              ),
              borderRadius: BorderRadius.circular(15),
              boxShadow: [
                BoxShadow(
                  color: Colors.blue.withOpacity(0.3),
                  blurRadius: 10,
                  spreadRadius: 2,
                ),
              ],
            ),
            child: ElevatedButton(
              onPressed: () {
                if (_formKey.currentState?.validate() ?? false) {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) => OTPVerificationScreen(
                        email: _emailController.text,
                      ),
                    ),
                  );
                }
              },
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.transparent,
                shadowColor: Colors.transparent,
                minimumSize: const Size.fromHeight(55),
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(15),
                ),
              ),
              child: const Text(
                'Login',
                style: TextStyle(
                  fontSize: 18,
                  fontWeight: FontWeight.bold,
                  color: Colors.white,
                ),
              ),
            ),
          ),
          const SizedBox(height: 20),
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Container(
                height: 1,
                width: 50,
                color: Colors.white24,
              ),
              const Padding(
                padding: EdgeInsets.symmetric(horizontal: 10),
                child: Text(
                  'Or continue with',
                  style: TextStyle(color:
```

```
Colors.white70),
                    ),
                  ),
                  Container(
                    height: 1,
                    width: 50,
                    color: Colors.white24,
                  ),
                ],
              ),
              const SizedBox(height: 20),
              Row(
                                mainAxisAlignment:
MainAxisAlignment.center,
                children: [
                  _buildSocialButton(
                    icon: Icons.phone,
                    onPressed: () {
                                // Add phone login
functionality
                    },
                  ),
                  const SizedBox(width: 20),
                  _buildSocialButton(
                    icon: Icons.g_mobiledata,
                    onPressed: () {
                                // Add Google login
functionality
                    },
                  ),
                ],
              ),
              const SizedBox(height: 20),
              Row(
                                mainAxisAlignment:
MainAxisAlignment.center,
                children: [
                  const Text(
                    "Don't have an account? ",
                              style: TextStyle(color:
Colors.white70),
                  ),
                  TextButton(
                    onPressed: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) => const
RegisterScreen(),
                      ),
                    );
                    },
                                child: const Text(
                    'Sign Up',
                    style: TextStyle(
                      color: Colors.white,
                                        fontWeight:
FontWeight.bold,
                      ),
                    ),
                  ),
                ],
              ),
            ],
          ),
        ),
      ),
    );
  }

  Widget _buildSocialButton({
    required IconData icon,
    required VoidCallback onPressed,
  }) {
    return Container(
      decoration: BoxDecoration(
        color: Colors.white10,
        borderRadius: BorderRadius.circular(15),
        border: Border.all(color: Colors.white24),
      ),
      child: IconButton(
        icon: Icon(icon, color: Colors.white),
        onPressed: onPressed,
      ),
    );
  }
}
```

register_screen.dart

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

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

@override
State<RegisterScreen> createState() =>
_RegisterScreenState();
}

class _RegisterScreenState extends
State<RegisterScreen> {
final _formKey = GlobalKey<FormState>();
final _nameController = TextEditingController();
final _emailController = TextEditingController();
final _passwordController = TextEditingController();
bool _isPasswordVisible = false;

@override
Widget build(BuildContext context) {
return Scaffold(
body: Container(
decoration: const BoxDecoration(
gradient: LinearGradient(
begin: Alignment.topCenter,
end: Alignment.bottomCenter,
colors: [
Color(0xFF1A237E),
Color(0xFF0D47A1),
Color(0xFF01579B),
],
),
),
child: SafeArea(
child: SingleChildScrollView(
child: Padding(
padding: const EdgeInsets.all(20.0),
child: Form(
key: _formKey,
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
IconButton(
icon: const Icon(Icons.arrow_back_ios, color:
Colors.white),
onPressed: () => Navigator.pop(context),
),
const SizedBox(height: 20),
const Text(
'Create Account',
style:  TextStyle(
fontSize: 32,
fontWeight: FontWeight.bold,
color: Colors.white,
),
),
const SizedBox(height: 10),
Text(
'Join SecureShield VPN',
style: TextStyle(
color: Colors.white.withOpacity(0.7),
fontSize: 16,
),
),
const SizedBox(height: 40),
// Name Field
Container(
decoration: BoxDecoration(
color: Colors.white10,
borderRadius: BorderRadius.circular(15),
border: Border.all(color: Colors.white24),
),
child: TextFormField(
controller: _nameController,
style: const TextStyle(color: Colors.white),
decoration: InputDecoration(
labelText: 'Full Name',
labelStyle: const TextStyle(color: Colors.white70),
border: OutlineInputBorder(
borderRadius: BorderRadius.circular(15),
borderSide: BorderSide.none,
),
prefixIcon: const Icon(Icons.person, color:
Colors.white70),
),
validator: (value) {
if (value?.isEmpty ?? true) {
return 'Please enter your name';
}
return null;
},
),
),
const SizedBox(height: 20),
// Email Field
Container(
decoration: BoxDecoration(
color: Colors.white10,
borderRadius: BorderRadius.circular(15),
border: Border.all(color: Colors.white24),
),
child: TextFormField(
controller: _emailController,
style: const TextStyle(color: Colors.white),
decoration: InputDecoration(
```

```
labelText: 'Email',
labelStyle: const TextStyle(color: Colors.white70),
border: OutlineInputBorder(
borderRadius: BorderRadius.circular(15),
borderSide: BorderSide.none,
),
prefixIcon: const Icon(Icons.email, color:
Colors.white70),
),
validator: (value) {
if (value?.isEmpty ?? true) {
return 'Please enter email';
}
if (!value!.contains('@')) {
return 'Please enter valid email';
}
return null;
},
),
),
const SizedBox(height: 20),
// Password Field
Container(
decoration: BoxDecoration(
color: Colors.white10,
borderRadius: BorderRadius.circular(15),
border: Border.all(color: Colors.white24),
),
child: TextFormField(
controller: _passwordController,
style: const TextStyle(color: Colors.white),
obscureText: !_isPasswordVisible,
decoration: InputDecoration(
labelText: 'Password',
labelStyle: const TextStyle(color: Colors.white70),
border: OutlineInputBorder(
borderRadius: BorderRadius.circular(15),
borderSide: BorderSide.none,
),
prefixIcon: const Icon(Icons.lock, color:
Colors.white70),
suffixIcon: IconButton(
icon: Icon(
_isPasswordVisible
? Icons.visibility
    : Icons.visibility_off,
color: Colors.white70,
),
onPressed: () {
setState(() {
_isPasswordVisible = !_isPasswordVisible;
});
},
),
),
validator: (value) {
if (value?.isEmpty ?? true) {
return 'Please enter password';
}
if (value!.length < 6) {
return 'Password must be at least 6 characters';
}
return null;
},
),
),
const SizedBox(height: 40),
// Register Button
Container(
width: double.infinity,
decoration: BoxDecoration(
gradient: const LinearGradient(
colors: [Colors.blue, Colors.lightBlue],
),
borderRadius: BorderRadius.circular(15),
boxShadow: [
BoxShadow(
color: Colors.blue.withOpacity(0.3),
blurRadius: 10,
spreadRadius: 2,
),
],
),
child: ElevatedButton(
onPressed: () {
if (_formKey.currentState?.validate() ?? false) {
Navigator.push(
context,
MaterialPageRoute(
builder: (context) => OTPVerificationScreen(
email: _emailController.text,
),
),
);
}
},
style: ElevatedButton.styleFrom(
backgroundColor: Colors.transparent,
shadowColor: Colors.transparent,
padding: const EdgeInsets.symmetric(vertical: 15),
shape: RoundedRectangleBorder(
borderRadius: BorderRadius.circular(15),
),
),
child: const Text(
'Create Account',
style: TextStyle(
fontSize: 18,
fontWeight: FontWeight.bold,
),
),
),
```

```
),
const SizedBox(height: 20),
// Login Option
Row(
mainAxisAlignment: MainAxisAlignment.center,
children: [
const Text(
'Already have an account? ',
style: TextStyle(color: Colors.white70),
),
TextButton(
onPressed: () => Navigator.pop(context),
child: const Text(
'Login',
style: TextStyle(
color: Colors.white,
fontWeight: FontWeight.bold,
),
),
),
],
),
],
),
),
),
),
),
),
);
}
}
```

otp_verification_screen.dart

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

class OTPVerificationScreen extends StatefulWidget {
  final String? email;
  final String? phoneNumber;

  const OTPVerificationScreen({
    super.key,
    this.email,
    this.phoneNumber,
  }) : assert(email != null || phoneNumber != null);

  @override
  State<OTPVerificationScreen> createState() =>
_OTPVerificationScreenState();
}

class _OTPVerificationScreenState extends
State<OTPVerificationScreen> {
  final List<TextEditingController> _controllers =
List.generate(
    4,
        (index) => TextEditingController(),
  );
  final List<FocusNode> _focusNodes = List.generate(
    4,
        (index) => FocusNode(),
  );

  Timer? _timer;
  int _timeLeft = 30;

  @override
  void initState() {
    super.initState();
    startTimer();
  }

  void startTimer() {
    _timer = Timer.periodic(const Duration(seconds: 1),
(timer) {
      if (_timeLeft > 0) {
        setState(() {
          _timeLeft--;
        });
      } else {
        _timer?.cancel();
      }
    });
  }
```

```
  @override
  void dispose() {
    _timer?.cancel();
    for (var controller in _controllers) {
      controller.dispose();
    }
    for (var node in _focusNodes) {
      node.dispose();
    }
    super.dispose();
  }

  void verifyOTP() {
    String otp = _controllers.map((e) => e.text).join();
    if (otp.length == 4) {
      // Here you would typically verify the OTP with your backend
      Navigator.pushReplacementNamed(context, '/main');
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        decoration: const BoxDecoration(
          gradient: LinearGradient(
            begin: Alignment.topCenter,
            end: Alignment.bottomCenter,
            colors: [
              Color(0xFF1A237E),
              Color(0xFF0D47A1),
              Color(0xFF01579B),
            ],
          ),
        ),
        child: SafeArea(
          child: Padding(
            padding: const EdgeInsets.all(20.0),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                IconButton(
                  icon: const Icon(Icons.arrow_back_ios, color: Colors.white),
                  onPressed: () => Navigator.pop(context),
                ),
                const SizedBox(height: 30),
                const Text(
                  'Verification Code',
```

```
            style: TextStyle(
              fontSize: 32,
              fontWeight: FontWeight.bold,
              color: Colors.white,
            ),
          ),
          const SizedBox(height: 10),
          Text(
            'We have sent the verification code
to\n${widget.email ?? widget.phoneNumber}',
            style: TextStyle(
              color: Colors.white.withOpacity(0.7),
              fontSize: 16,
            ),
          ),
          const SizedBox(height: 50),
          Row(
            mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
            children: List.generate(
              4,
                (index) => SizedBox(
                width: 60,
                child: Container(
                  decoration: BoxDecoration(
                    color: Colors.white.withOpacity(0.1),
                    borderRadius:
BorderRadius.circular(15),
                    border: Border.all(color:
Colors.white24),
                  ),
                  child: TextField(
                    controller: _controllers[index],
                    focusNode: _focusNodes[index],
                    textAlign: TextAlign.center,
                    style: const TextStyle(
                      color: Colors.white,
                      fontSize: 24,
                      fontWeight: FontWeight.bold,
                    ),
                    keyboardType: TextInputType.number,
                    maxLength: 1,
                    decoration: InputDecoration(
                      counterText: '',
                      filled: true,
                      fillColor: Colors.transparent,
                      border: OutlineInputBorder(
                      borderRadius:
BorderRadius.circular(15),
                        borderSide: BorderSide.none,
                      ),
                    ),
```

```
                    onChanged: (value) {
                      if (value.isNotEmpty && index < 3) {
                        _focusNodes[index +
1].requestFocus();
                      }
                      if (value.isEmpty && index > 0) {
                        _focusNodes[index -
1].requestFocus();
                      }
                      if (index == 3 && value.isNotEmpty)
{
                        verifyOTP();
                      }
                    },
                  ),
                ),
              ),
            ),
          ),
          const SizedBox(height: 40),
          Center(
            child: Text(
              'Time remaining: $_timeLeft seconds',
              style: const TextStyle(
                color: Colors.white70,
                fontSize: 16,
              ),
            ),
          ),
          const SizedBox(height: 20),
          if (_timeLeft == 0)
            Center(
              child: TextButton(
                onPressed: () {
                  setState(() {
                    _timeLeft = 30;
                  });
                  startTimer();
                  // Here you would typically resend the
OTP
                },
                child: const Text(
                  'Resend Code',
                  style: TextStyle(
                    color: Colors.white,
                    fontSize: 16,
                  ),
                ),
              ),
            ),
          const Spacer(),
          Container(
```

```
          width: double.infinity,
          decoration: BoxDecoration(
          gradient: const LinearGradient(
            colors: [Colors.blue, Colors.lightBlue],
           ),
           borderRadius: BorderRadius.circular(15),
           boxShadow: [
            BoxShadow(
              color: Colors.blue.withOpacity(0.3),
              blurRadius: 10,
              spreadRadius: 2,
            ),
           ],
          ),
          child: ElevatedButton(
           onPressed: verifyOTP,
           style: ElevatedButton.styleFrom(
            backgroundColor: Colors.transparent,
            shadowColor: Colors.transparent,
            padding: const
EdgeInsets.symmetric(vertical:  15),
            shape: RoundedRectangleBorder(
             borderRadius: BorderRadius.circular(15),
            ),
           ),
           child: const Text(
            'Verify',
            style: TextStyle(
             fontSize: 18,
             fontWeight: FontWeight.bold,
            ),
           ),
          ),
         ),
        ),
       ),
      );
     }
    }
```

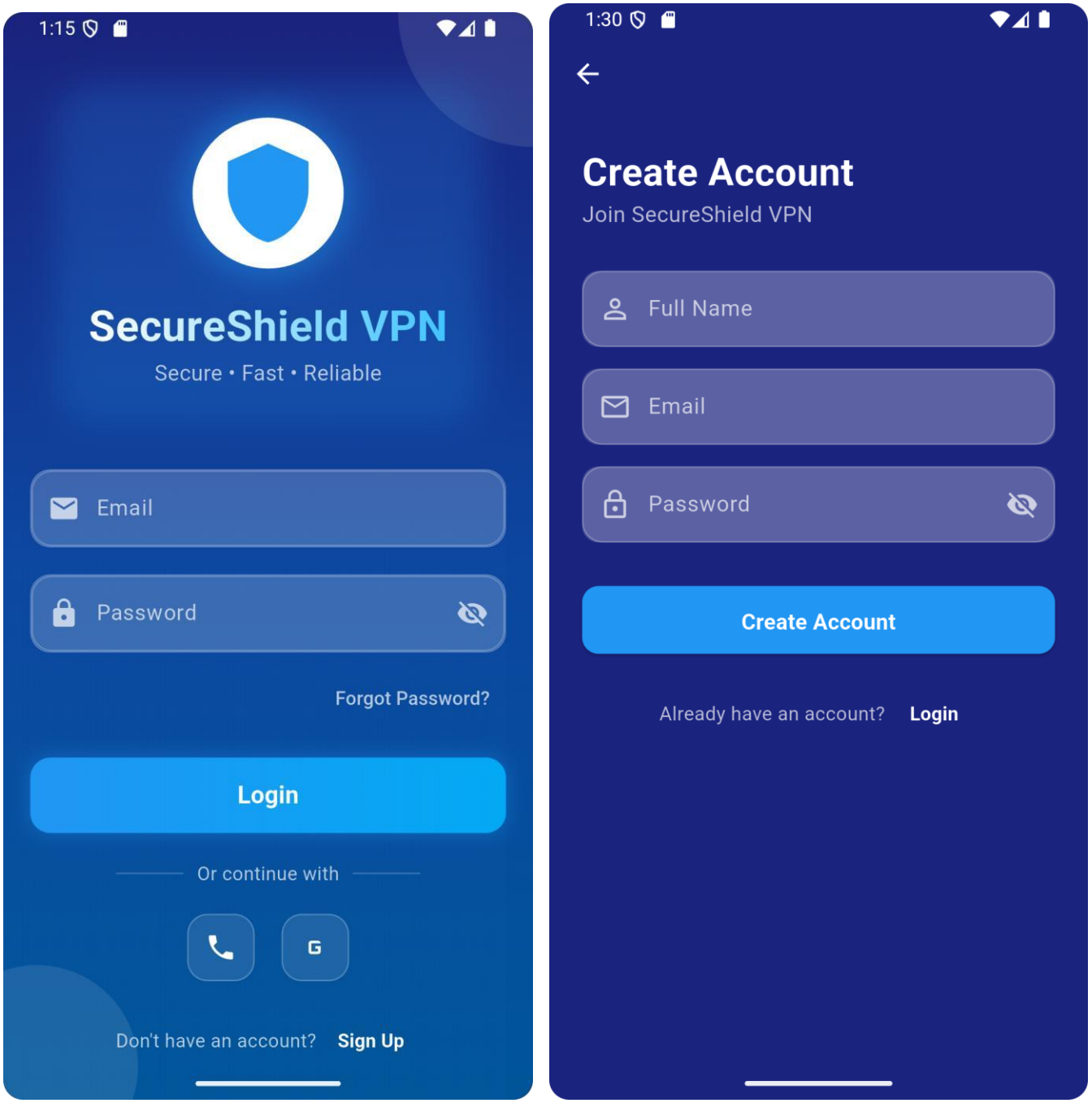
profile_screen.dart

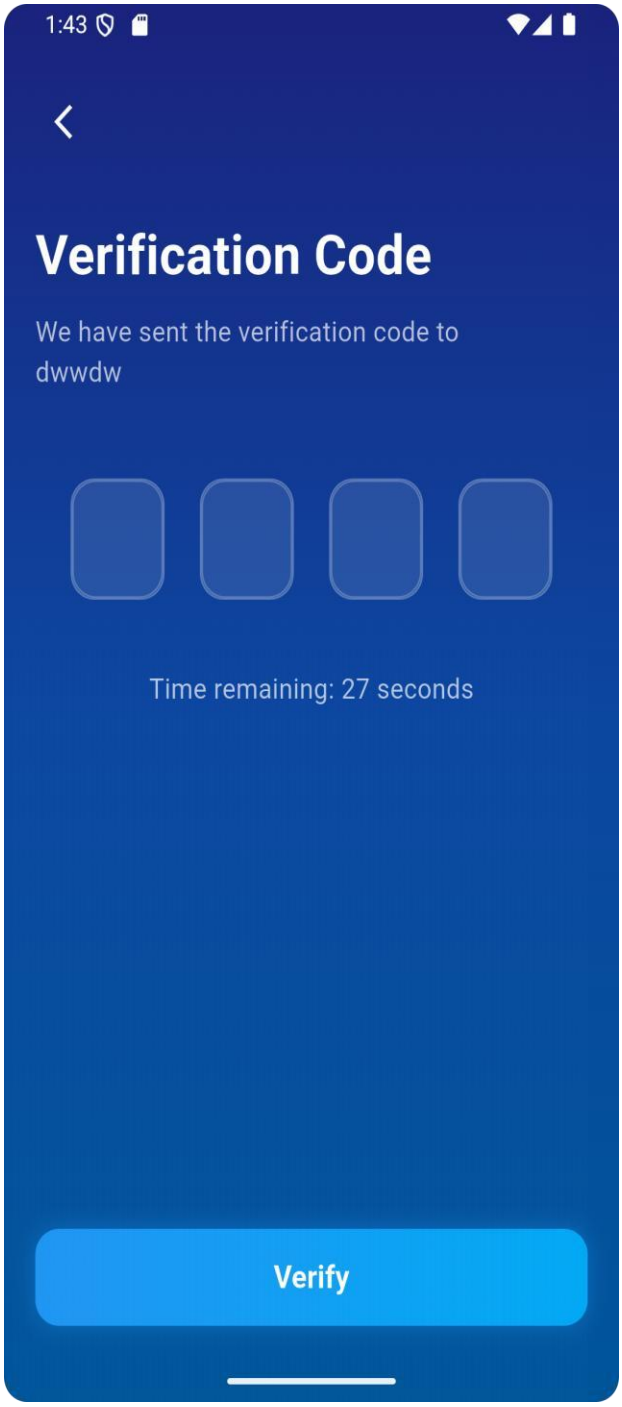
```
import 'package:flutter/material.dart';
import 'dart:math';
import 'dart:async';
class ProfileScreen extends StatelessWidget {
  const ProfileScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.transparent,
      body: SingleChildScrollView(
        child: Column(
          children: [
            _buildProfileHeader(),
            _buildProfileOptions(),
          ],
        ),
      ),
    );
  }

  Widget _buildProfileHeader() {
    return Container(
      padding: const EdgeInsets.all(32),
      decoration: BoxDecoration(
      gradient: LinearGradient(
          colors: [Colors.blue.shade900,
Colors.purple.shade900],
        ),
        borderRadius: const
BorderRadius.vertical(bottom:
Radius.circular(30)),
      ),
      child: SafeArea(
        child: Column(
        children: [
          const CircleAvatar(
            radius: 50,
            backgroundColor: Colors.white,
            child: Icon(Icons.person, size: 50, color:
Colors.blue),
          ),
          const SizedBox(height: 16),
          const Text('Free User',
          style: TextStyle(
              color: Colors.white,
              fontSize: 24,
              fontWeight: FontWeight.bold,
            ),
          ),
          const SizedBox(height: 16),
          _buildPremiumButton(),
          ],
        ),
      ),
    );
  }

  Widget _buildPremiumButton() {
    return Container(
      decoration: BoxDecoration(
        gradient: LinearGradient(
          colors: [Colors.amber.shade400,
Colors.orange.shade400],
        ),
        borderRadius: BorderRadius.circular(15),
        boxShadow: [
          BoxShadow(
            color: Colors.amber.withOpacity(0.3),
            blurRadius: 10,
            spreadRadius: 2,
          ),
        ],
      ),
      child: ElevatedButton(
        onPressed: () {},
        style: ElevatedButton.styleFrom(
          backgroundColor: Colors.transparent,
          shadowColor: Colors.transparent,
          padding: const
EdgeInsets.symmetric(horizontal: 32, vertical: 12),
          shape:
RoundedRectangleBorder(borderRadius:
BorderRadius.circular(15)),
        ),
        child: const Row(
          mainAxisSize: MainAxisSize.min,
          children: [
            Icon(Icons.star, color: Colors.white),
            SizedBox(width: 8),
            Text('Upgrade to Premium',
              style: TextStyle(color: Colors.white,
fontWeight: FontWeight.bold),
            ),
          ],
        ),
      ),
    );
  }

  Widget _buildProfileOptions() {
    return Padding(
      padding: const EdgeInsets.all(16),
      child: Column(
        children: [
          _buildOptionTile('Connection History',
Icons.history),
          _buildOptionTile('Support',
```

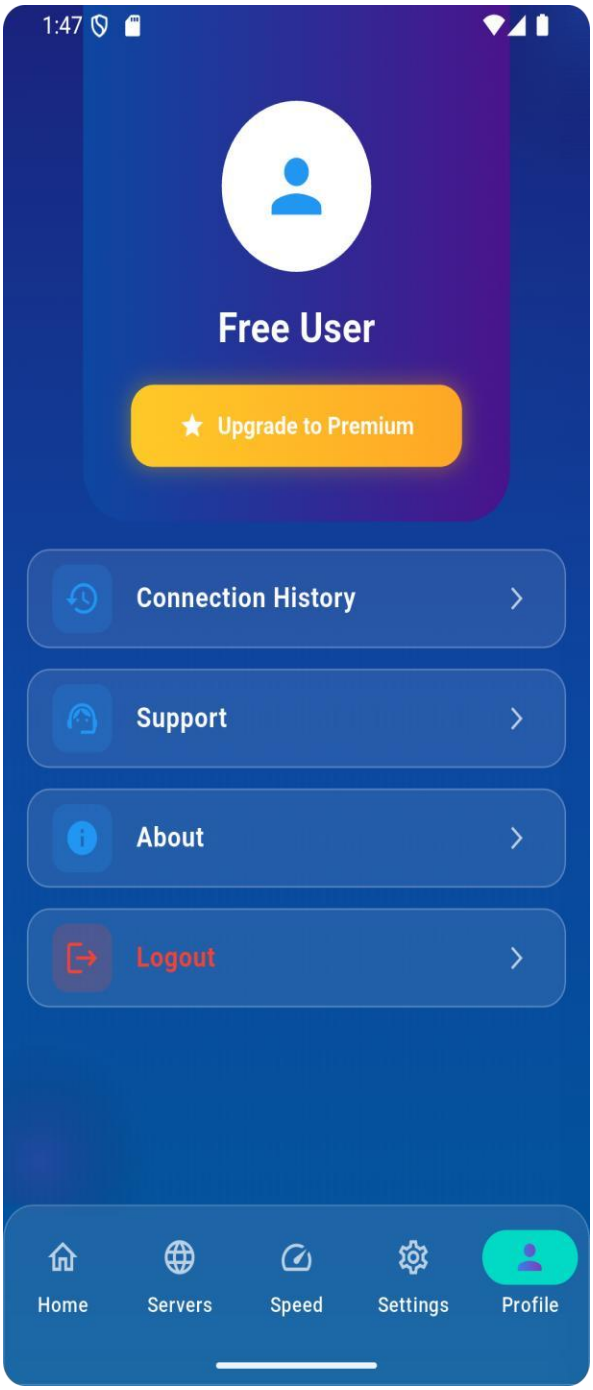
```
Icons.support_agent),
        _buildOptionTile('About', Icons.info),
        _buildOptionTile('Logout', Icons.logout,
isLogout: true),
      ],
    ),
   );
  }

  Widget _buildOptionTile(String title, IconData
icon, {bool isLogout = false}) {
   return Container(
    margin: const EdgeInsets.only(bottom: 12),
    decoration: BoxDecoration(
     color: Colors.white.withOpacity(0.1),
     borderRadius: BorderRadius.circular(15),
     border: Border.all(color: Colors.white24),
    ),
    child: ListTile(
     leading: Container(
      padding: const EdgeInsets.all(8),
      decoration: BoxDecoration(
      color: (isLogout ? Colors.red :
Colors.blue).withOpacity(0.2),
       borderRadius: BorderRadius.circular(10),
      ),
      child: Icon(icon,
       color: isLogout ? Colors.red : Colors.blue,
      ),
     ),
     title: Text(title,
      style: TextStyle(
       color: isLogout ? Colors.red :
Colors.white,
       fontWeight: FontWeight.bold,
      ),
     ),
     trailing: const
Icon(Icons.arrow_forward_ios,
      color: Colors.white70,
      size: 16,
     ),
    ),
   );
  }
}
TextStyle(fontSize:
      18)), Column(
       children:      myCrops.map((crop)      => Text(crop)).toList(),
      ),
     ],
    ),
   );
  }
```

```
Widget _buildTextField({
 required String label,
 required String
 initialValue,
 required Function(String?) onSaved,
}) {
 return TextFormField(
  initialValue: initialValue,
  decoration:
  InputDecoration(
  labelText: label,
   border: OutlineInputBorder(),
  ),
  onSaved: onSaved,

 );
}
```

OUTPUT: -


1:15
SecureShield VPN
Secure • Fast • Reliable
Email
Password
Forgot Password?
Login
Or continue with
G
Don't have an account? Sign Up
1:30
Create Account
Join SecureShield VPN
Full Name
Email
Password
Create Account
Already have an account? Login


1:43
Verification Code
We have sent the verification code to dwwdw
Time remaining: 27 seconds
Verify


1:47
Free User
Upgrade to Premium
Connection History
Support
About
Logout
Home
Servers
Speed
Settings
Profile

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App.
Roll No.	57
Name	Laksh Sodhai
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation.
Grade:	

EXPERIMENT NO: - 05

Name:- Laksh Sodhai **Class:-** D15A **Roll:No: -** 57

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

Theory: -

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

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

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

- **Navigation and Routing in Flutter**

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

  1. Using Navigator Widget

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

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

```
ElevatedButton(
  onPressed: () {
  Navigator.push(
```

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

2. Named Routes

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

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

Navigate to the route using Navigator.pushNamed()

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

Handling Gestures in Flutter

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

Tap Gestures

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

Long Press Gesture

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

Swipe and Drag Gestures

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

Code: -

main.dart

```
import 'package:flutter/material.dart';
import 'screens/login_screen.dart';
import 'screens/register_screen.dart';
import 'screens/main_screen.dart';
import 'screens/home_screen.dart';
import 'screens/servers_screen.dart';
import 'screens/settings_screen.dart';
import 'screens/profile_screen.dart';
import 'screens/speed_test_screen.dart';
import 'screens/vpn_details_screen.dart';
import 'screens/otp_verification_screen.dart';
import 'screens/phone_login_screen.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'SecureShield VPN',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.dark(
          primary: Colors.blue,
          background: const Color(0xFF1A237E),
          secondary: Colors.blue,
          surface: Colors.white.withOpacity(0.1),
        ),
        scaffoldBackgroundColor: const
Color(0xFF1A237E),
        appBarTheme: const AppBarTheme(
          backgroundColor: Colors.transparent,
          elevation: 0,
          iconTheme: IconThemeData(color:
Colors.white),
          titleTextStyle: TextStyle(
            color: Colors.white,
            fontSize: 24,
            fontWeight: FontWeight.bold,
          ),
        ),
        textTheme: const TextTheme(
          headlineLarge: TextStyle(
            color: Colors.white,
            fontSize: 32,
            fontWeight: FontWeight.bold,
          ),
          headlineMedium: TextStyle(
            color: Colors.white,
            fontSize: 24,
            fontWeight: FontWeight.bold,
          ),
          bodyLarge: TextStyle(
            color: Colors.white,
            fontSize: 16,
          ),
          bodyMedium: TextStyle(
            color: Colors.white70,
            fontSize: 14,
          ),
        ),
        elevatedButtonTheme:
ElevatedButtonThemeData(
          style: ElevatedButton.styleFrom(
            backgroundColor: Colors.blue,
            foregroundColor: Colors.white,
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(12),
            ),
            padding: const
EdgeInsets.symmetric(vertical: 16),
          ),
        ),
        inputDecorationTheme:
InputDecorationTheme(
          filled: true,
          fillColor: Colors.white.withOpacity(0.1),
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(12),
            borderSide: BorderSide.none,
          ),
          enabledBorder: OutlineInputBorder(
            borderRadius: BorderRadius.circular(12),
            borderSide: BorderSide(color:
Colors.white.withOpacity(0.1)),
          ),
          focusedBorder: OutlineInputBorder(
            borderRadius: BorderRadius.circular(12),
            borderSide: const BorderSide(color:
Colors.blue),
          ),
          labelStyle: const TextStyle(color:
Colors.white70),
          hintStyle: const TextStyle(color:
Colors.white60),
        ),
      ),
      initialRoute: '/',
      onGenerateRoute: (settings) {
        switch (settings.name) {
          case '/':
            return MaterialPageRoute(builder: (_) =>
const LoginScreen());
```

```
case '/register':
          return MaterialPageRoute(builder: (_) =>
const RegisterScreen());
        case '/main':
          return MaterialPageRoute(builder: (_) =>
const MainScreen());
        case '/home':
          return MaterialPageRoute(builder: (_) =>
const HomeScreen());
        case '/servers':
          return MaterialPageRoute(builder: (_) =>
const ServersScreen());
        case '/settings':
          return MaterialPageRoute(builder: (_) =>
const SettingsScreen());
        case '/profile':
          return MaterialPageRoute(builder: (_) =>
const ProfileScreen());
        case '/speedtest':
          return MaterialPageRoute(builder: (_) =>
const SpeedTestScreen());
        case '/vpn-details':
          return MaterialPageRoute(builder: (_) =>
const VPNDetailsScreen());
        case '/phone-login':
          return MaterialPageRoute(builder: (_) =>
const PhoneLoginScreen());
        case '/otp':
          if (settings.arguments != null) {
            final args = settings.arguments as
Map<String, String?>;
            return MaterialPageRoute(
              builder: (_) => OTPVerificationScreen(
                email: args['email'],
                phoneNumber: args['phoneNumber'],
              ),
            );
          }
          return MaterialPageRoute(builder: (_) =>
const LoginScreen());
        default:
          return MaterialPageRoute(builder: (_) =>
const LoginScreen());
      }
    },
    builder: (context, child) {
      return MediaQuery(
        data:
MediaQuery.of(context).copyWith(textScaleFactor:
1.0),
        child: child!,
      );
    },
  );
 }
}

// App Constants
class VPNConstants {
  static const appName = 'SecureShield VPN';
  static const appVersion = '1.0.0';

  // Colors
  static const primaryColor = Color(0xFF1A237E);
  static const accentColor = Colors.blue;

  // Animation Durations
  static const animationDuration =
Duration(milliseconds: 300);

  // Padding & Sizes
  static const defaultPadding = 16.0;
  static const borderRadius = 12.0;

  // Strings
  static const strConnected = 'Connected';
  static const strDisconnected = 'Not Connected';
  static const strConnecting = 'Connecting...';

  // Server List
  static const List<Map<String, dynamic>> servers =
[
    {'country': 'United States', 'flag': 'US', 'ping': 45},
    {'country': 'United Kingdom', 'flag': 'GB', 'ping':
65},
    {'country': 'Japan', 'flag': 'JP', 'ping': 85},
    {'country': 'Germany', 'flag': 'DE', 'ping': 55},
    {'country': 'Singapore', 'flag': 'SG', 'ping': 75},
  ];
}
```

Login_screen.dart

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

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

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

class _LoginScreenState extends
State<LoginScreen> {
  final _formKey = GlobalKey<FormState>();
  final _emailController = TextEditingController();
  final _passwordController =
TextEditingController();
  bool _isPasswordVisible = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        decoration: const BoxDecoration(
          gradient: LinearGradient(
            begin: Alignment.topCenter,
            end: Alignment.bottomCenter,
            colors: [
              Color(0xFF1A237E),
              Color(0xFF0D47A1),
              Color(0xFF01579B),
            ],
          ),
        ),
        child: Stack(
          children: [
            Positioned(
              top: -100,
              right: -100,
              child: Container(
                width: 200,
                height: 200,
                decoration: BoxDecoration(
                  shape: BoxShape.circle,
                  color: Colors.white.withOpacity(0.1),
                ),
              ),
            ),
            Positioned(
              bottom: -50,
              left: -50,
              child: Container(
                width: 150,
                height: 150,
                decoration: BoxDecoration(
                  shape: BoxShape.circle,
                  color: Colors.white.withOpacity(0.1),
                ),
              ),
            ),
            SingleChildScrollView(
              child: Padding(
                padding: const EdgeInsets.all(20.0),
                child: Form(
                  key: _formKey,
                  child: Column(
                    mainAxisAlignment:
MainAxisAlignment.center,
                    children: [
                      const SizedBox(height: 80),
                      Container(
                        padding: const EdgeInsets.all(20),
                        decoration: BoxDecoration(
                          borderRadius:
BorderRadius.circular(30),
                          boxShadow: [
                            BoxShadow(
                              color:
Colors.blue.withOpacity(0.2),
                              blurRadius: 20,
                              spreadRadius: 5,
                            ),
                          ],
                        ),
                        child: Column(
                          children: [
                            Container(
                              padding: const EdgeInsets.all(16),
                              decoration: BoxDecoration(
                                color: Colors.white,
                                shape: BoxShape.circle,
                                boxShadow: [
                                  BoxShadow(
                                    color:
Colors.blue.withOpacity(0.5),
                                    blurRadius: 20,
                                    spreadRadius: 5,
                                  ),
                                ],
```

```
),
                    child: const Icon(
                     Icons.shield,
                     size: 80,
                     color: Colors.blue,
                    ),
                   ),
                   const SizedBox(height: 20),
                   ShaderMask(
                    shaderCallback: (bounds) =>
const LinearGradient(
                     colors: [Colors.white,
Colors.lightBlueAccent],
                    ).createShader(bounds),
                    child: const Text(
                     'SecureShield VPN',
                     style: TextStyle(
                      fontSize: 32,
                      fontWeight: FontWeight.bold,
                      color: Colors.white,
                     ),
                    ),
                   ),
                   const Text(
                    'Secure • Fast • Reliable',
                    style: TextStyle(
                     color: Colors.white70,
                     fontSize: 16,
                    ),
                   ),
                  ],
                 ),
                ),
                const SizedBox(height: 40),
                Container(
                 decoration: BoxDecoration(
                  color: Colors.white10,
                  borderRadius:
BorderRadius.circular(15),
                  border: Border.all(color:
Colors.white24),
                 ),
                 child: TextFormField(
                  controller: _emailController,
                  style: const TextStyle(color:
Colors.white),
                  decoration: InputDecoration(
                   labelText: 'Email',
                   labelStyle: const TextStyle(color:
Colors.white70),
                   border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(15),
                    borderSide: BorderSide.none,
                   ),
                   prefixIcon: const Icon(Icons.email,
color: Colors.white70),
                  ),
                  validator: (value) {
                   if (value?.isEmpty ?? true) {
                    return 'Please enter email';
                   }
                   return null;
                  },
                 ),
                ),
                const SizedBox(height: 20),
                Container(
                 decoration: BoxDecoration(
                  color: Colors.white10,
                  borderRadius:
BorderRadius.circular(15),
                  border: Border.all(color:
Colors.white24),
                 ),
                 child: TextFormField(
                  controller: _passwordController,
                  style: const TextStyle(color:
Colors.white),
                  obscureText: !_isPasswordVisible,
                  decoration: InputDecoration(
                   labelText: 'Password',
                   labelStyle: const TextStyle(color:
Colors.white70),
                   border: OutlineInputBorder(
                    borderRadius:
BorderRadius.circular(15),
                    borderSide: BorderSide.none,
                   ),
                   prefixIcon: const Icon(Icons.lock,
color: Colors.white70),
                   suffixIcon: IconButton(
                    icon: Icon(
                     _isPasswordVisible
                       ? Icons.visibility
                       : Icons.visibility_off,
                     color: Colors.white70,
                    ),
                    onPressed: () {
                     setState(() {
                      _isPasswordVisible =
!_isPasswordVisible;
                     });
                    },
```

```
                ),
              ),
              validator: (value) {
                if (value?.isEmpty ?? true) {
                  return 'Please enter password';
                }
                return null;
              },
            ),
          ),
          const SizedBox(height: 10),
          Align(
            alignment: Alignment.centerRight,
            child: TextButton(
              onPressed: () {
                // Add forgot password
functionality
              },
              child: const Text(
                'Forgot Password?',
                style: TextStyle(color:
Colors.white70),
              ),
            ),
          ),
          const SizedBox(height: 20),
          Container(
            decoration: BoxDecoration(
              gradient: const LinearGradient(
                colors: [Colors.blue,
Colors.lightBlue],
              ),
              borderRadius:
BorderRadius.circular(15),
              boxShadow: [
                BoxShadow(
                  color:
Colors.blue.withOpacity(0.3),
                  blurRadius: 10,
                  spreadRadius: 2,
                ),
              ],
            ),
            child: ElevatedButton(
              onPressed: () {
                if
(_formKey.currentState?.validate() ?? false) {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) =>
OTPVerificationScreen(
                        email: _emailController.text,
                      ),
                    ),
                  );
                }
              },
              style: ElevatedButton.styleFrom(
                backgroundColor:
Colors.transparent,
                shadowColor: Colors.transparent,
                minimumSize: const
Size.fromHeight(55),
                shape: RoundedRectangleBorder(
                  borderRadius:
BorderRadius.circular(15),
                ),
              ),
              child: const Text(
                'Login',
                style: TextStyle(
                  fontSize: 18,
                  fontWeight: FontWeight.bold,
                  color: Colors.white,
                ),
              ),
            ),
          ),
          const SizedBox(height: 20),
          Row(
            mainAxisAlignment:
MainAxisAlignment.center,
            children: [
              Container(
                height: 1,
                width: 50,
                color: Colors.white24,
              ),
              const Padding(
                padding:
EdgeInsets.symmetric(horizontal: 10),
                child: Text(
                  'Or continue with',
                  style: TextStyle(color:
Colors.white70),
                ),
              ),
              Container(
                height: 1,
                width: 50,
                color: Colors.white24,
```

```
),
                ],
              ),
              const SizedBox(height: 20),
              Row(
                mainAxisAlignment:
MainAxisAlignment.center,
                children: [
                  _buildSocialButton(
                    icon: Icons.phone,
                    onPressed: () {
                      // Add phone login functionality
                    },
                  ),
                  const SizedBox(width: 20),
                  _buildSocialButton(
                    icon: Icons.g_mobiledata,
                    onPressed: () {
                      // Add Google login functionality
                    },
                  ),
                ],
              ),
              const SizedBox(height: 20),
              Row(
                mainAxisAlignment:
MainAxisAlignment.center,
                children: [
                  const Text(
                    "Don't have an account? ",
                    style: TextStyle(color:
Colors.white70),
                  ),
                  TextButton(
                    onPressed: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const
RegisterScreen(),
                        ),
                      );
                    },
                    child: const Text(
                      'Sign Up',
                      style: TextStyle(
                        color: Colors.white,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                  ),
                ],
                        ),
                      ],
                    ),
                  ),
                ),
              ),
            ],
          ),
        ),
      );
    }

    Widget _buildSocialButton({
      required IconData icon,
      required VoidCallback onPressed,
    }) {
      return Container(
        decoration: BoxDecoration(
          color: Colors.white10,
          borderRadius: BorderRadius.circular(15),
          border: Border.all(color: Colors.white24),
        ),
        child: IconButton(
          icon: Icon(icon, color: Colors.white),
          onPressed: onPressed,
        ),
      );
    }
}
```

Speed_test_screen.dart

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

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

  @override
  State<SpeedTestScreen> createState() =>
_SpeedTestScreenState();
}

class _SpeedTestScreenState extends
State<SpeedTestScreen> with
SingleTickerProviderStateMixin {
  late AnimationController _controller;
  double downloadSpeed = 0;
  double uploadSpeed = 0;
  double currentValue = 0;
  bool isTesting = false;
  Timer? _timer;

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      vsync: this,
      duration: const Duration(seconds: 2),
    );
  }

  void startSpeedTest() {
    setState(() {
      isTesting = true;
      downloadSpeed = 0;
      uploadSpeed = 0;
      currentValue = 0;
    });

    _timer = Timer.periodic(const
Duration(milliseconds: 100), (timer) {
      setState(() {
        if (currentValue < 140) {
          currentValue += 2;
          downloadSpeed = 94.999 * (currentValue /
140);
          uploadSpeed = 68.887 * (currentValue / 140);
        } else {
          _timer?.cancel();
          isTesting = false;
        }
      });
    });
  }

  @override
  void dispose() {
    _controller.dispose();
    _timer?.cancel();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.transparent,
      appBar: AppBar(
        title: const Text('Speed Test',
          style: TextStyle(fontSize: 24, fontWeight:
FontWeight.bold),
        ),
        backgroundColor: Colors.transparent,
        elevation: 0,
      ),
      body: Column(
        children: [
          const SizedBox(height: 20),
          _buildServerInfo(),
          const SizedBox(height: 40),
          _buildSpeedometer(),
          const SizedBox(height: 20),
          _buildSpeedResults(),
        ],
      ),
    );
  }

  Widget _buildServerInfo() {
    return Row(
      mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
      children: [
        _buildInfoCard('Server', Icons.router, 'Local
Server', Colors.blue),
        _buildInfoCard('Location', Icons.location_on,
'Current', Colors.purple),
      ],
    );
  }

  Widget _buildInfoCard(String title, IconData icon,
String value, Color color) {
    return Container(
      padding: const EdgeInsets.all(16),
      decoration: BoxDecoration(
        color: Colors.white.withOpacity(0.1),
        borderRadius: BorderRadius.circular(15),
        border: Border.all(color: Colors.white24),
      ),
```

```
child: Column(
      children: [
       Icon(icon, color: color, size: 30),
       const SizedBox(height: 8),
       Text(title,
        style: const TextStyle(color: Colors.white70),
       ),
       Text(value,
        style: const TextStyle(
         color: Colors.white,
         fontWeight: FontWeight.bold,
        ),
       ),
      ],
     ),
    );
  }

  Widget _buildSpeedometer() {
   return SizedBox(
    height: 300,
    child: Stack(
     alignment: Alignment.center,
     children: [
      CustomPaint(
       painter: SpeedometerPainter(
        value: currentValue,
        isTesting: isTesting,
       ),
       size: const Size(300, 300),
      ),
      Column(
       mainAxisSize: MainAxisSize.min,
       children: [
        if (!isTesting && currentValue == 0)
         ElevatedButton(
          onPressed: startSpeedTest,
          style: ElevatedButton.styleFrom(
           backgroundColor: Colors.blue,
           shape: RoundedRectangleBorder(
            borderRadius:
BorderRadius.circular(20),
           ),
           padding: const EdgeInsets.symmetric(
            horizontal: 32,
            vertical: 16,
           ),
          ),
          child: const Text('Start Test'),
         ),
        if (isTesting || currentValue > 0)
         Text(
          '${currentValue.toStringAsFixed(1)}',
          style: const TextStyle(
           fontSize: 32,
           fontWeight: FontWeight.bold,
           color: Colors.white,
          ),
         ),
        ],
       ),
      ],
     ),
    );
   }

   Widget _buildSpeedResults() {
    return Row(
     mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
     children: [
      _buildSpeedCard('Download', Icons.download,
downloadSpeed),
      _buildSpeedCard('Upload', Icons.upload,
uploadSpeed),
     ],
    );
   }

   Widget _buildSpeedCard(String type, IconData
icon, double speed) {
    return Container(
     padding: const EdgeInsets.all(16),
     decoration: BoxDecoration(
      color: Colors.white.withOpacity(0.1),
      borderRadius: BorderRadius.circular(15),
      border: Border.all(color: Colors.white24),
     ),
     child: Column(
      children: [
       Icon(icon, color: Colors.blue, size: 30),
       const SizedBox(height: 8),
       Text('$type Speed',
        style: const TextStyle(color: Colors.white70),
       ),
       Text(
        '${speed.toStringAsFixed(2)} Mbps',
        style: const TextStyle(
         color: Colors.white,
         fontWeight: FontWeight.bold,
         fontSize: 18,
        ),
       ),
      ],
     ),
    );
   }
}

class SpeedometerPainter extends CustomPainter {
```

```
final double value;
  final bool isTesting;

  SpeedometerPainter({required this.value, required
this.isTesting});

  @override
  void paint(Canvas canvas, Size size) {
    final center = Offset(size.width / 2, size.height /
2);
    final radius = min(size.width / 2, size.height / 2) -
20;

    final paint = Paint()
      ..style = PaintingStyle.stroke
      ..strokeWidth = 20;

    paint.color = Colors.grey.withOpacity(0.2);
    canvas.drawArc(
      Rect.fromCircle(center: center, radius: radius),
      pi,
      pi,
      false,
      paint,
    );

    final gradient = SweepGradient(
      startAngle: pi,
      endAngle: 2 * pi,
      colors: const [
        Colors.green,
        Colors.blue,
        Colors.purple,
        Colors.pink,
      ],
    );

    paint.shader = gradient.createShader(
      Rect.fromCircle(center: center, radius: radius),
    );

    final progressAngle = (value / 140) * pi;
    canvas.drawArc(
      Rect.fromCircle(center: center, radius: radius),
      pi,
      progressAngle,
      false,
      paint,
    );
  }

  @override
  bool shouldRepaint(SpeedometerPainter
oldDelegate) =>
      value != oldDelegate.value || isTesting !=
oldDelegate.isTesting;
  }
```

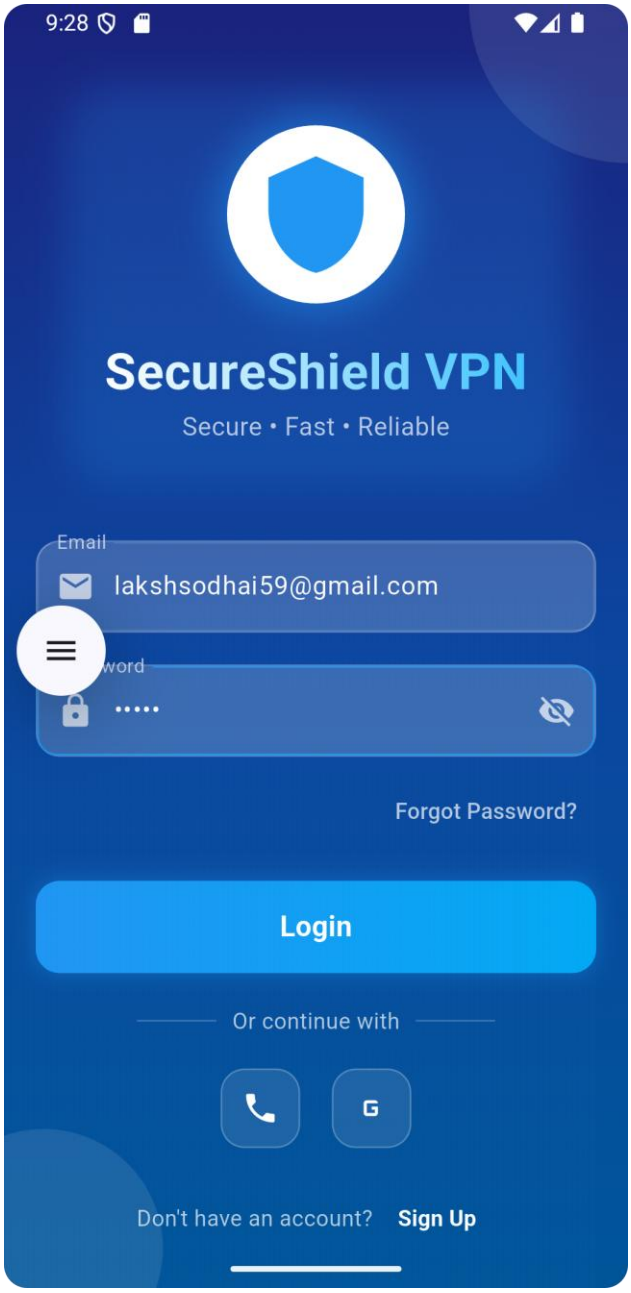
servers_screen.dart

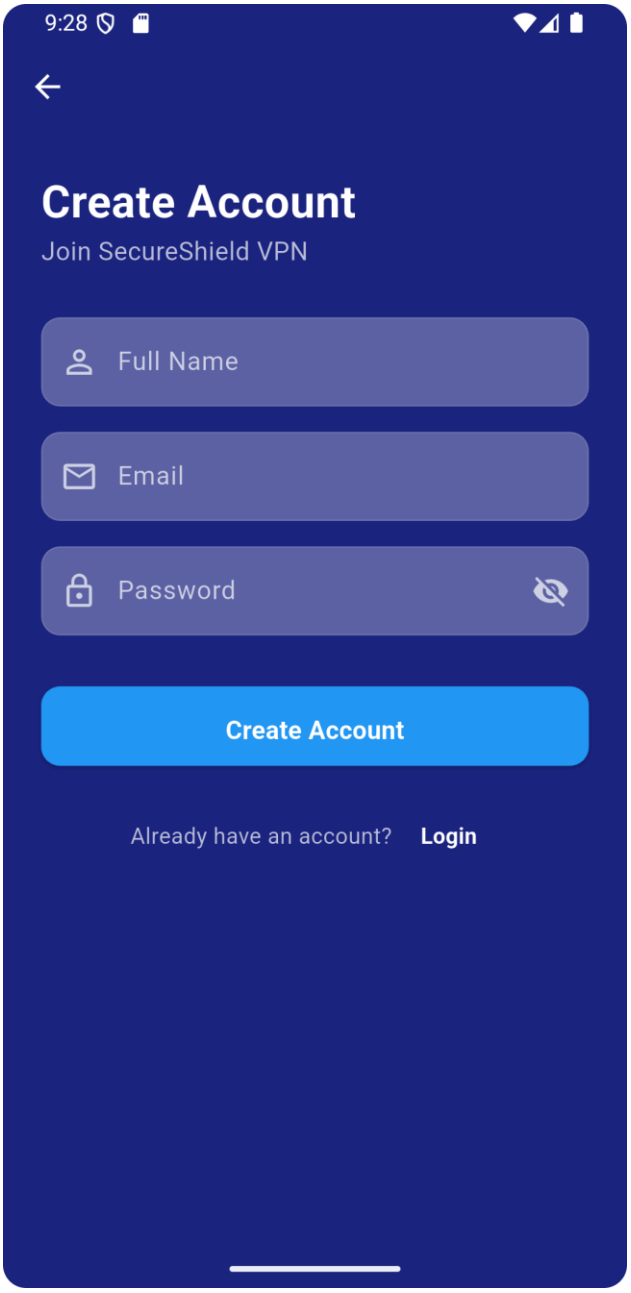
```
import 'package:flutter/material.dart';
import 'dart:math';
import 'dart:async';
class ServersScreen extends StatelessWidget {
  const ServersScreen({super.key});

  @override
  Widget build(BuildContext context) {
    final List<Map<String, dynamic>> servers =
[
      {"country": "United States", "flag": "US",
"ping": 45},
      {"country": "United Kingdom", "flag": "GB",
"ping": 65},
      {"country": "Japan", "flag": "JP", "ping":
85},
      {"country": "Germany", "flag": "DE", "ping":
55},
      {"country": "Singapore", "flag": "SG",
"ping": 75},
      {"country": "Canada", "flag": "CA", "ping":
50},
      {"country": "France", "flag": "FR", "ping":
60},
      {"country": "Australia", "flag": "AU", "ping":
90},
      {"country": "Netherlands", "flag": "NL",
"ping": 58},
      {"country": "India", "flag": "IN", "ping":
95},
    ];

    return Scaffold(
      backgroundColor: Colors.transparent,
      appBar: AppBar(
        title: const Text('Server Locations',
          style: TextStyle(fontSize: 24, fontWeight:
FontWeight.bold),
        ),
        backgroundColor: Colors.transparent,
        elevation: 0,
      ),
      body: ListView.builder(
        padding: const EdgeInsets.all(16),
        itemCount: servers.length,
        itemBuilder: (context, index) {
          final server = servers[index];
          return Container(
            margin: const EdgeInsets.only(bottom:
12),
            decoration: BoxDecoration(
              color: Colors.white.withOpacity(0.1),
              borderRadius:
BorderRadius.circular(15),
              border: Border.all(color:
Colors.white24),
              boxShadow: [
                BoxShadow(
                  color:
Colors.black.withOpacity(0.1),
                  blurRadius: 10,
                ),
              ],
            ),
            child: ListTile(
              contentPadding: const
EdgeInsets.all(16),
              leading: Text(
                server["flag"].toString(),
                style: const TextStyle(fontSize: 30),
              ),
              title: Text(
                server["country"].toString(),
                style: const TextStyle(
                  color: Colors.white,
                  fontWeight: FontWeight.bold,
                ),
              ),
              subtitle: Row(
                children: [
                  Icon(Icons.speed, size: 16, color:
Colors.orange.shade400),
                  const SizedBox(width: 4),
                  Text(
                    '${server["ping"]}ms',
                    style: TextStyle(color:
Colors.orange.shade400),
                  ),
                ],
              ),
              trailing: Container(
                padding: const EdgeInsets.all(8),
                decoration: BoxDecoration(
                  color: Colors.blue.withOpacity(0.2),
                  borderRadius:
BorderRadius.circular(10),
```

```
            ),

 child: const Icon(Icons.power_settings_new,
color: Colors.blue),
            ),
            onTap: () {

ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(
                  content: Text('Connecting to
${server["country"]}...'),
                  duration: const Duration(seconds:
1),
                ),
              );
            },
          ),
        );
      }
    }
```

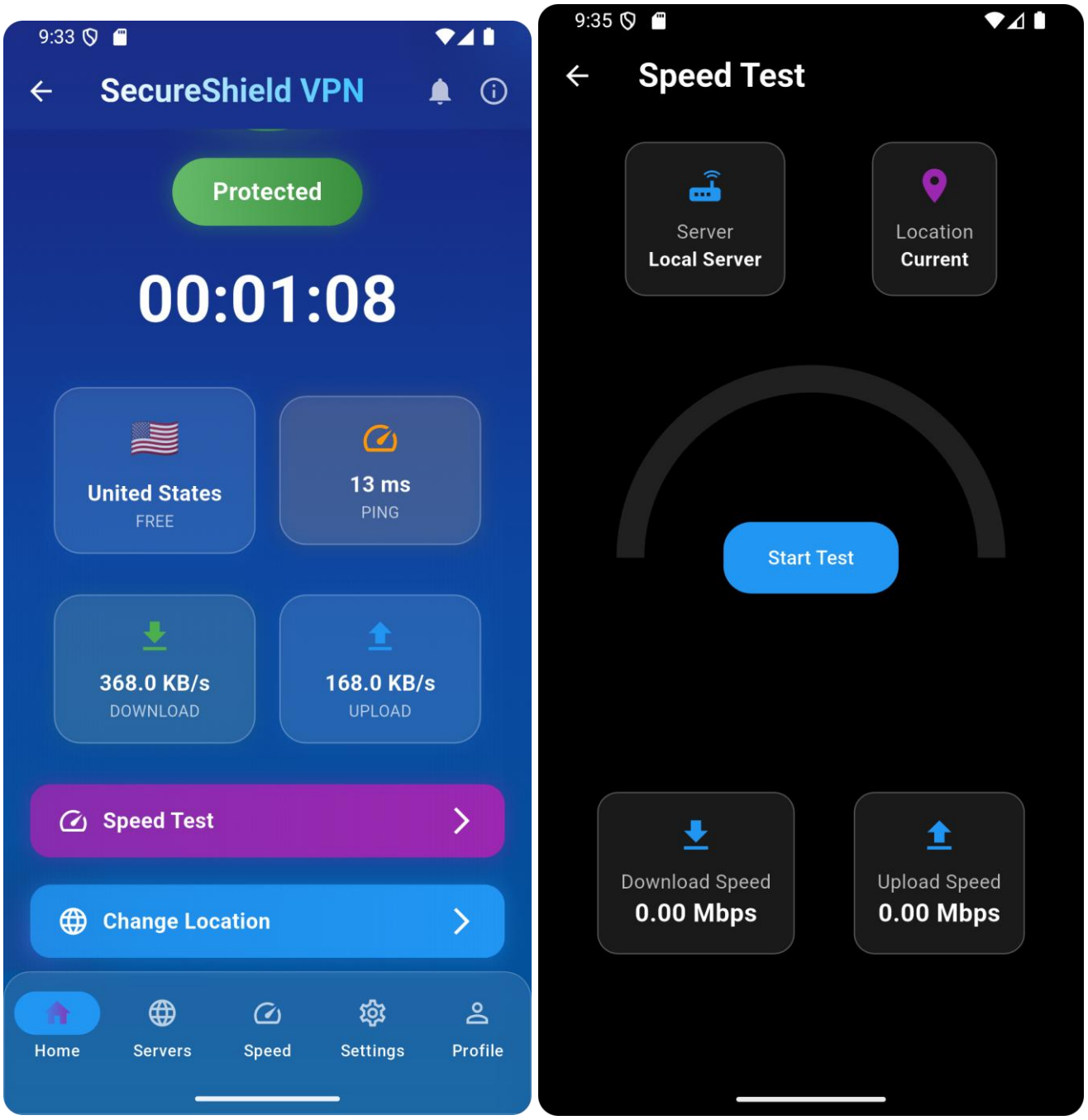
Output-

After clicking on Don’t have an account? It navigates to the registration page.

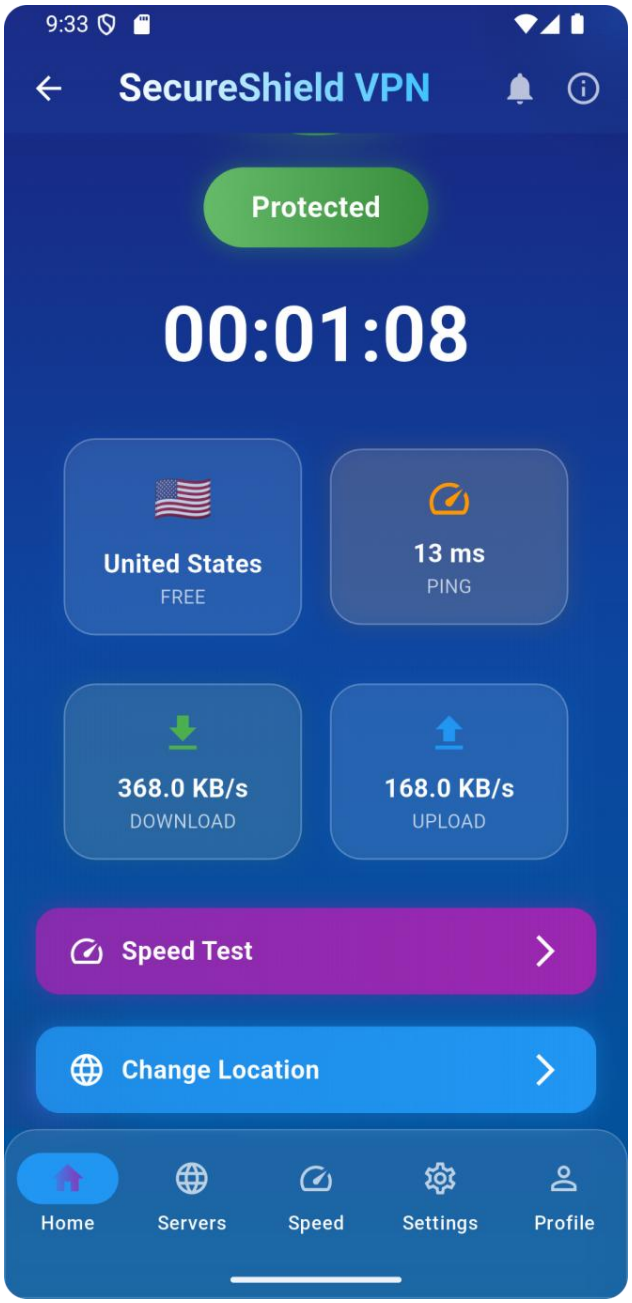


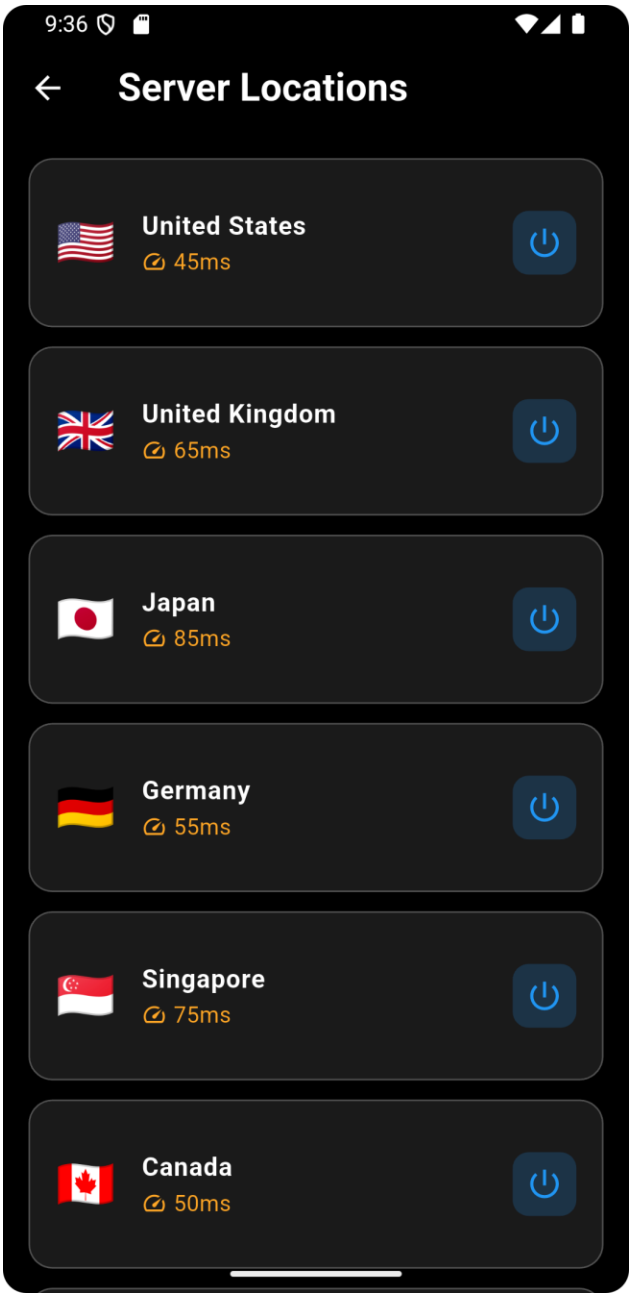


In home page, after clicking on Speed test it navigates to the speed test page.



In home page, after clicking on change location it navigates to the Server locations page.





MAD & PWA Lab

Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database.
Roll No.	57
Name	Laksh Sodhai
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS.
Grade:	

EXPERIMENT NO: - 06

Name:- Laksh Sodhai **Class:-** D15A **Roll:No: -** 57

AIM: - To connect Flutter UI with Firebase database.

Theory: -

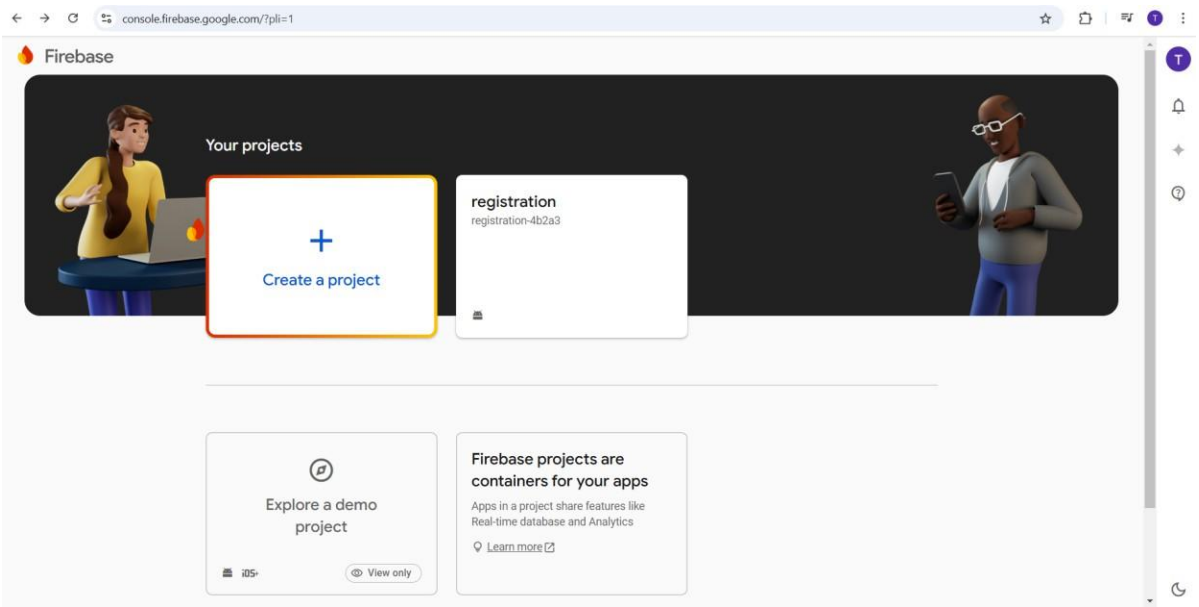
Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. Firebase, a Backend-as-a-Service (BaaS) platform, provides real-time database, authentication, and cloud storage services, making it a powerful backend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

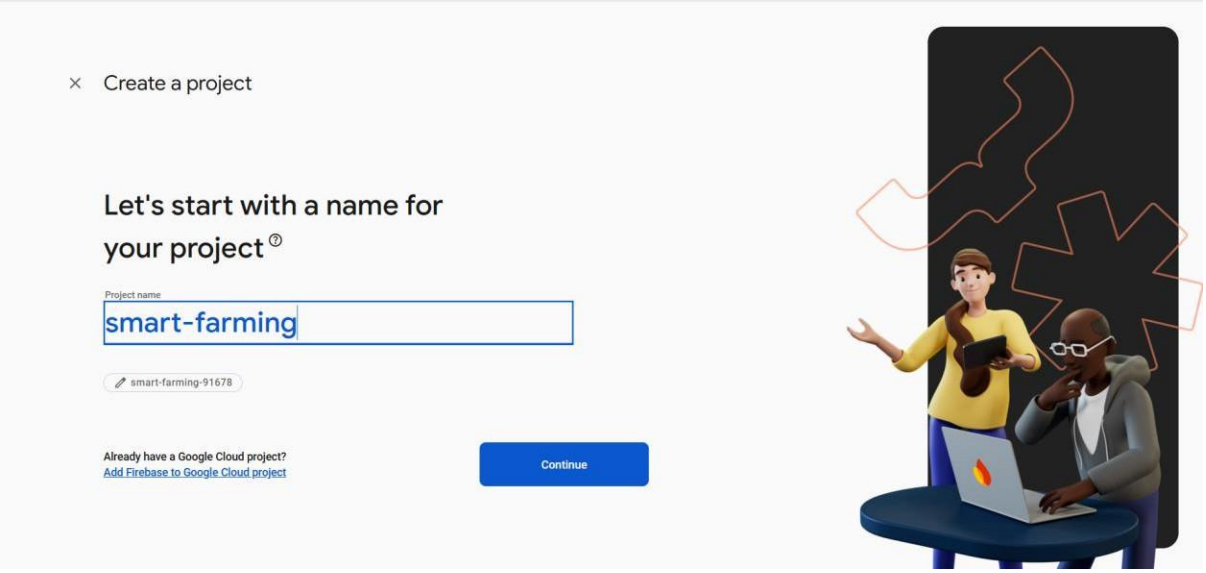
- **Steps to Connect Flutter UI with Firebase Database**

Step 1:

1.1) Go to Firebase Console and Create a Firebase Project

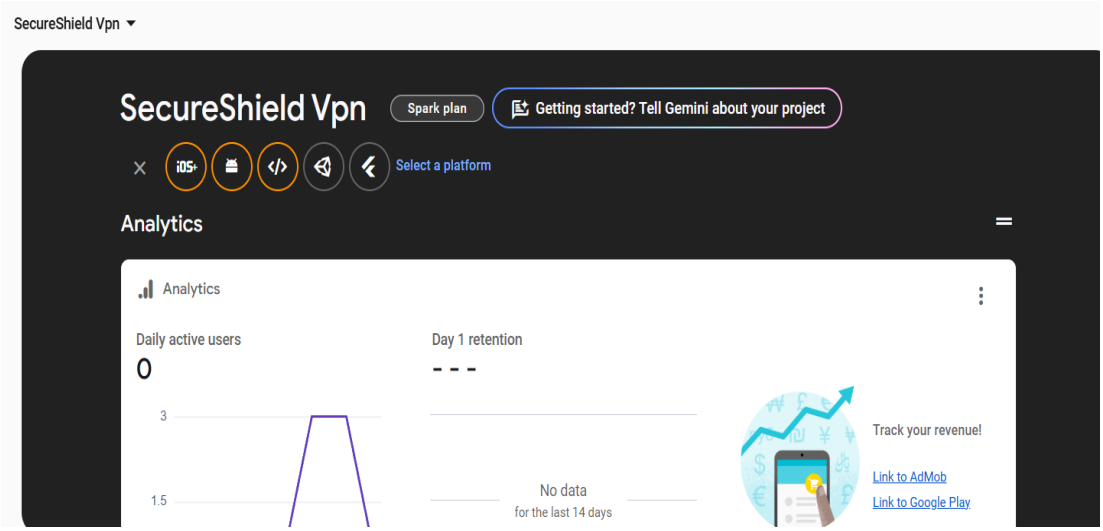


1.2) Click on Create a Project and give it a suitable name.

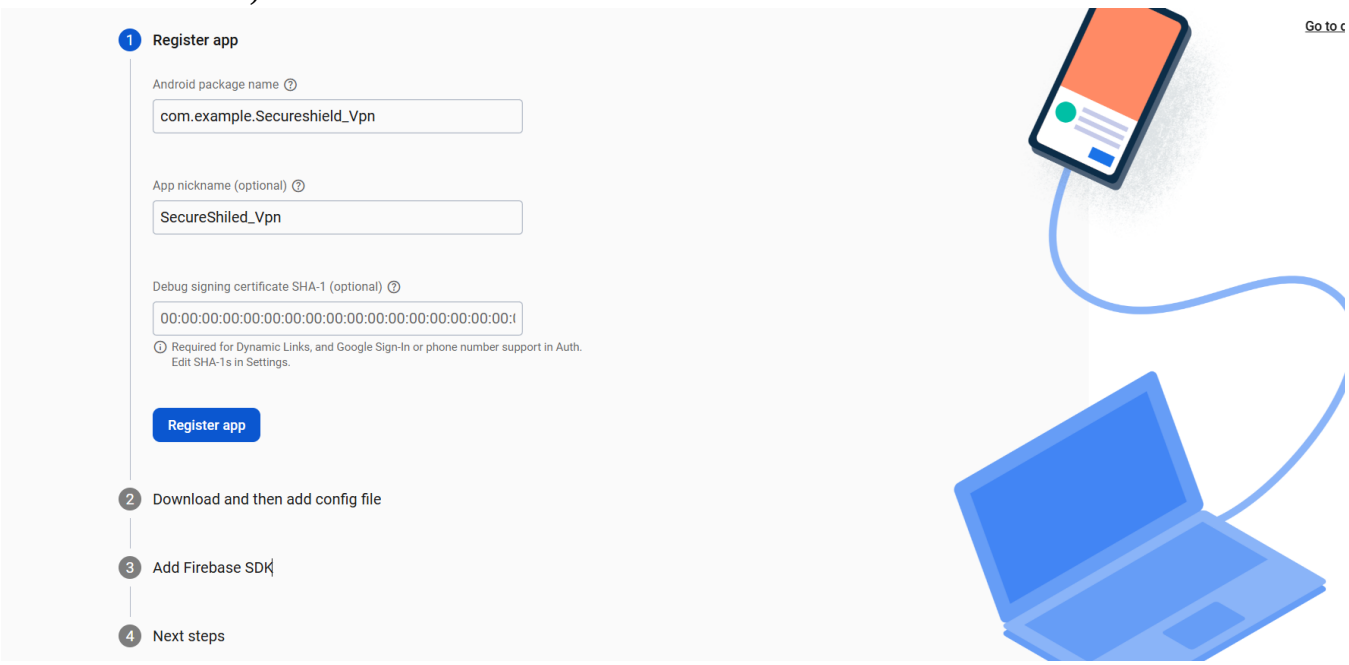


Step 2:- Add Firebase to Your Flutter App

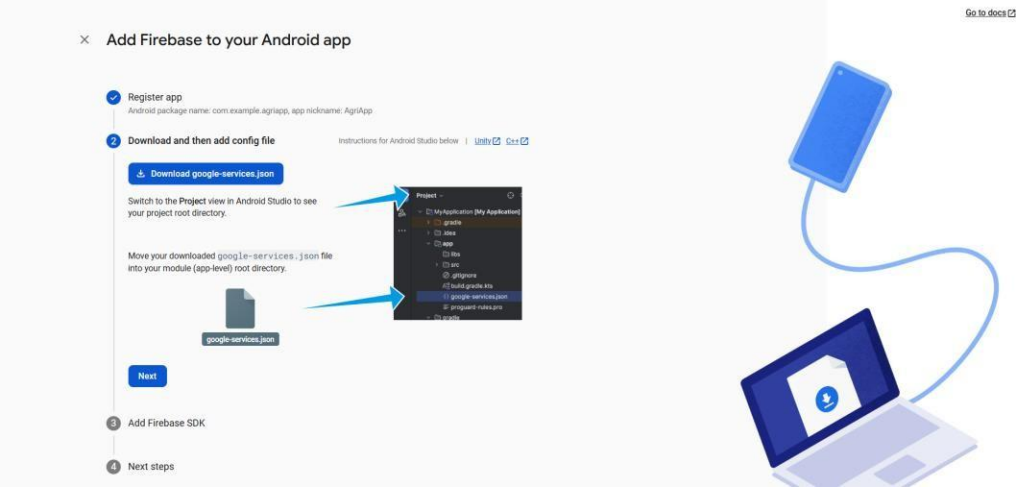
2.1) Click on Android/iOS/Web based on your Flutter application

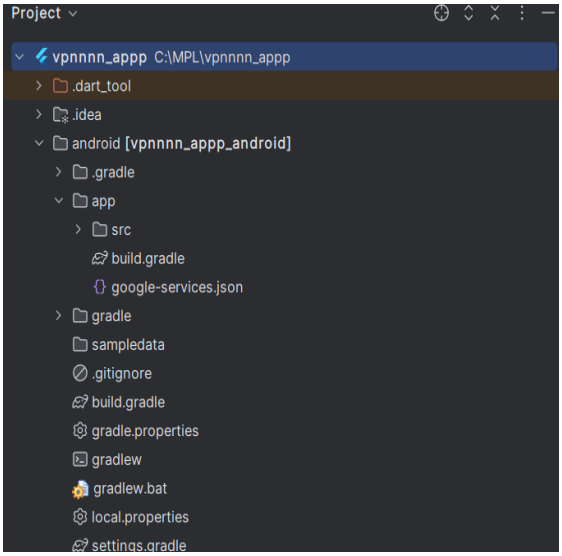


2.2) Register your app with a unique package name (found in android/app/build.gradle for Android).

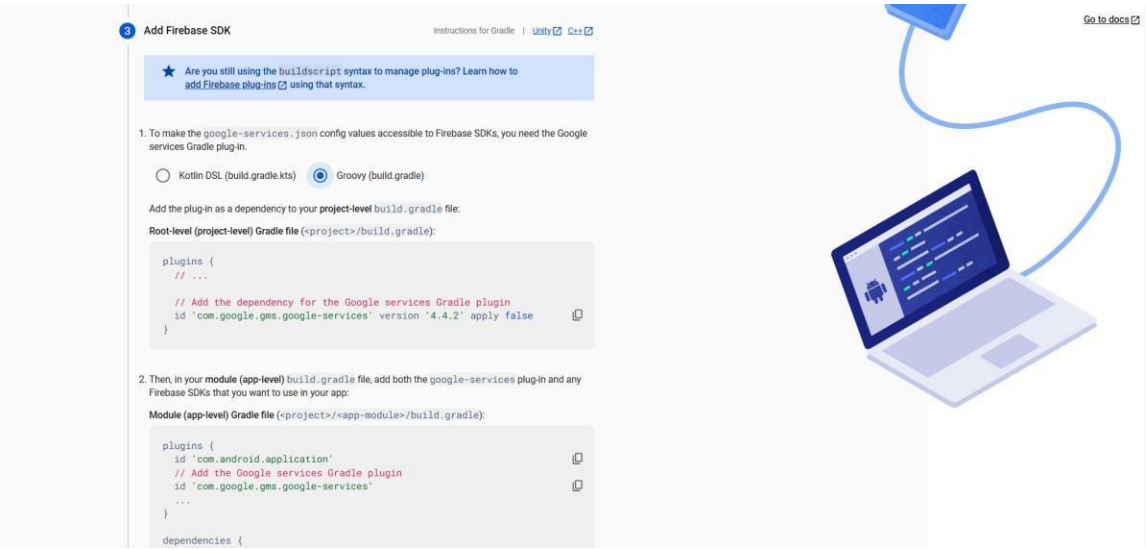


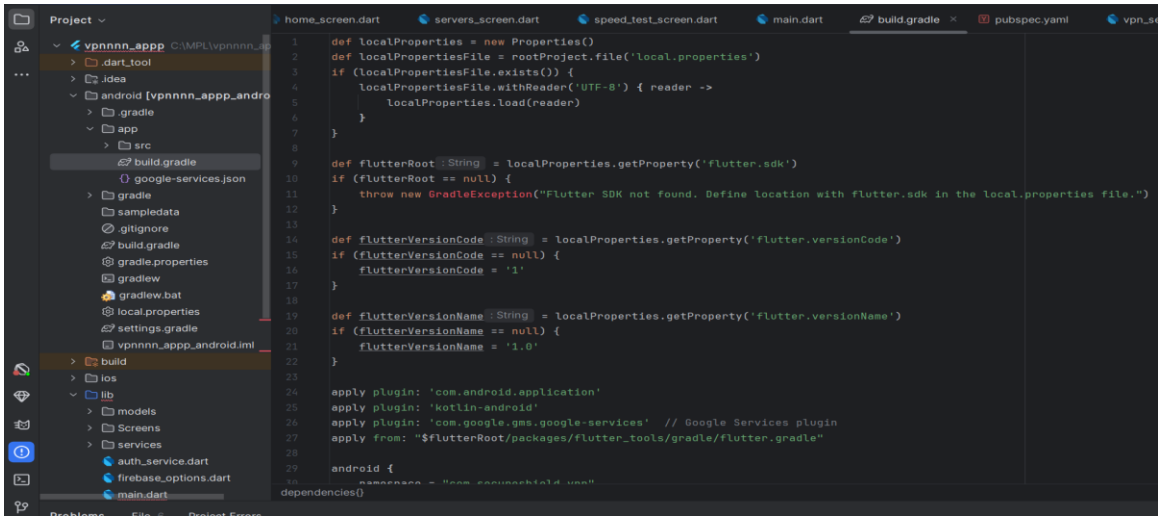
2.3) Download the google-services.json (for Android) & place the JSON file inside android/app/ directory.





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



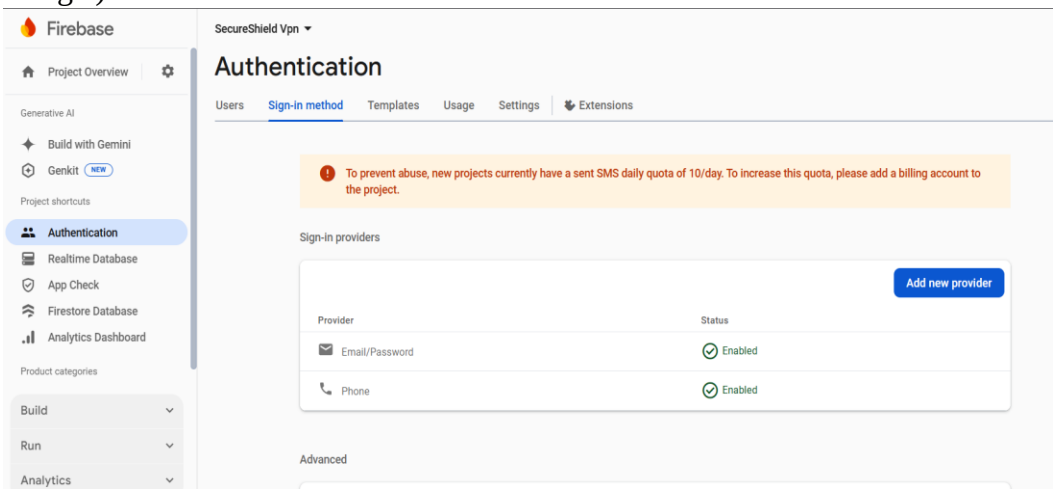


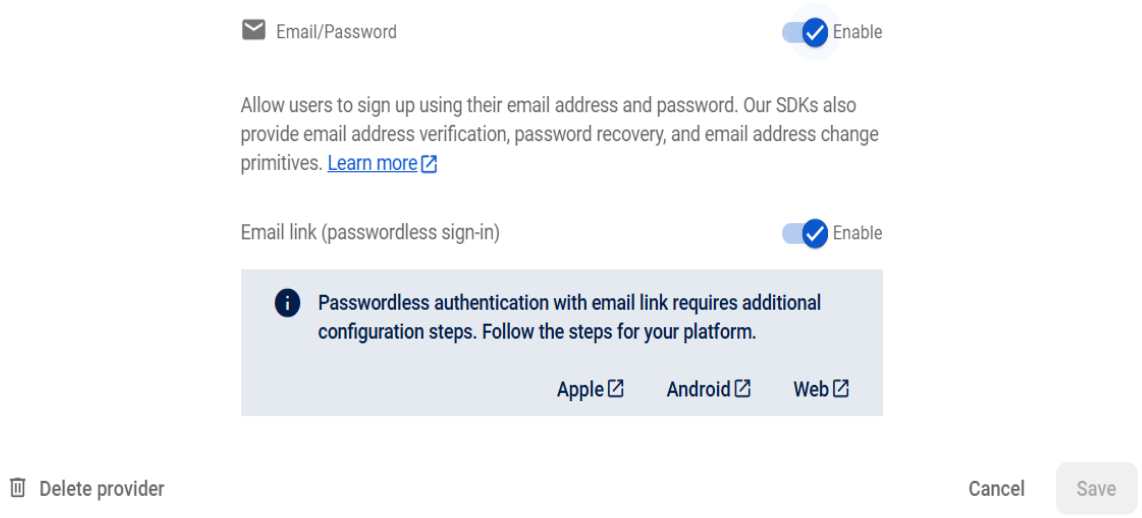
Step 3: - Add Firebase Authentication to Your App

3.1) Add Firebase Authentication Dependencies

```yaml
dependencies:
  flutter:
    sdk: flutter


  firebase_core: ^2.17.0
  firebase_analytics: ^10.5.1
  firebase_auth: ^4.10.1
  cloud_firestore: ^4.9.3


  provider: ^6.0.5


  cupertino_icons: ^1.0.2

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^2.0.0

flutter:
  uses-material-design: true
```

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





3.3) Implement Authentication in Flutter
Modify main.dart

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

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

Step 4: -Configure Firebase Realtime Database

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:-

Register_page.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../services/auth_service.dart';

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

  @override
  State<RegisterScreen> createState() =>
_RegisterScreenState();
}

class _RegisterScreenState extends
State<RegisterScreen> {
  final _formKey = GlobalKey<FormState>();
  final _nameController =
TextEditingController();
  final _emailController =
TextEditingController();
  final _passwordController =
TextEditingController();
  bool _isLoading = false;
  String? _errorMessage;

  @override
  void dispose() {
    _nameController.dispose();
    _emailController.dispose();
    _passwordController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.transparent,
        elevation: 0,
        leading: IconButton(
          icon: const Icon(Icons.arrow_back),
          onPressed: () => Navigator.pop(context),
        ),
      ),

      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(24.0),
          child: Form(
            key: _formKey,
            child: Column(
              crossAxisAlignment:
CrossAxisAlignment.start,
              children: [
                const Text(
                  'Create Account',
                  style: TextStyle(
                    color: Colors.white,
                    fontSize: 32,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                const Text(
                  'Join SecureShield VPN',
                  style: TextStyle(
                    color: Colors.white70,
                    fontSize: 16,
                  ),
                ),
                const SizedBox(height: 40),
                // Name Field
                Container(
                  decoration: BoxDecoration(
                    color:
Colors.white.withOpacity(0.1),
                    borderRadius:
BorderRadius.circular(12),
                  ),
                  child: TextFormField(
                    controller: _nameController,
                    style: const TextStyle(color:
Colors.white),
                    decoration: const InputDecoration(
                      hintText: 'Full Name',
                      prefixIcon:
Icon(Icons.person_outline),
```

```
                  border: InputBorder.none,
                  contentPadding:
EdgeInsets.symmetric(horizontal: 20, vertical:
16),
                ),
                validator: (value) {
                  if (value == null || value.isEmpty)
{
                    return 'Please enter your name';
                  }
                  return null;
                },
              ),
            ),
            const SizedBox(height: 16),
            // Email Field
            Container(
              decoration: BoxDecoration(
                color:
Colors.white.withOpacity(0.1),
                borderRadius:
BorderRadius.circular(12),
              ),
              child: TextFormField(
                controller: _emailController,
                style: const TextStyle(color:
Colors.white),
                decoration: const InputDecoration(
                  hintText: 'Email',
                  prefixIcon:
Icon(Icons.email_outlined),
                  border: InputBorder.none,
                  contentPadding:
EdgeInsets.symmetric(horizontal: 20, vertical:
16),
                ),
                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),
            // Password Field
            Container(
              decoration: BoxDecoration(
                color:
Colors.white.withOpacity(0.1),
                borderRadius:
BorderRadius.circular(12),
              ),
              child: TextFormField(
                controller: _passwordController,
                obscureText: true,
                style: const TextStyle(color:
Colors.white),
                decoration: const InputDecoration(
                  hintText: 'Password',
                  prefixIcon:
Icon(Icons.lock_outline),
                  border: InputBorder.none,
                  contentPadding:
EdgeInsets.symmetric(horizontal: 20, vertical:
16),
                ),
                validator: (value) {
                  if (value == null || value.isEmpty)
{
                    return 'Please enter a password';
                  }
                  if (value.length < 6) {
                    return 'Password must be at least
6 characters';
                  }
                  return null;
                },
              ),
            ),
            const SizedBox(height: 24),
            if (_errorMessage != null)
              Padding(
                padding: const
EdgeInsets.only(bottom: 16),
                child: Text(
                  _errorMessage!,
                  style: TextStyle(color:
Theme.of(context).colorScheme.error),
                  textAlign: TextAlign.center,
```

```
              ),
            ),
            // Register Button
            SizedBox(
              width: double.infinity,
              child: ElevatedButton(
                onPressed: _isLoading ? null :
_register,
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.blue,
                  shape: RoundedRectangleBorder(
                    borderRadius:
BorderRadius.circular(12),
                  ),
                  padding: const
EdgeInsets.symmetric(vertical: 16),
                ),
                child: _isLoading
                    ? const SizedBox(
                  height: 20,
                  width: 20,
                  child: CircularProgressIndicator(
                    color: Colors.white,
                    strokeWidth: 2,
                  ),
                )
                    : const Text(
                  'Create Account',
                  style: TextStyle(
                    fontSize: 16,
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ),
            ),
            const SizedBox(height: 20),
            // Login Option
            Row(
              mainAxisAlignment:
MainAxisAlignment.center,
              children: [
                const Text(
                  'Already have an account? ',
                  style: TextStyle(color:
Colors.white70),
                ),
                TextButton(
                  onPressed: () =>
Navigator.pop(context),
                  child: const Text(
                    'Login',
                    style: TextStyle(
                      color: Colors.white,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                ),
              ],
            ),
          ],
        ),
      ),
    ),
  );
}

Future<void> _register() async {
  print('Register button clicked');

  if (_formKey.currentState!.validate()) {
    setState(() {
      _isLoading = true;
      _errorMessage = null;
    });

    try {
      final authService =
Provider.of<AuthService>(context, listen:
false);

      await authService.signUp(
        _emailController.text,
        _passwordController.text,
        _nameController.text,
      );

      print('Registration successful');

      // No need for manual navigation - the
AuthWrapper will handle it
    } catch (e) {
      print('Registration error: $e');
      setState(() {
```

```
        String message = 'Registration failed';

        if (e.toString().contains('email-already-in-
use')) {
          message = 'This email is already
registered';
        } else if (e.toString().contains('weak-
password')) {
          message = 'Password is too weak';
        } else if (e.toString().contains('invalid-
email')) {
          message = 'Invalid email format';
        }

        _errorMessage = message;
      });
    } finally {
      if (mounted) {
        setState(() {
          _isLoading = false;
        });
      }
    }
  }
}
```

login_page.dart

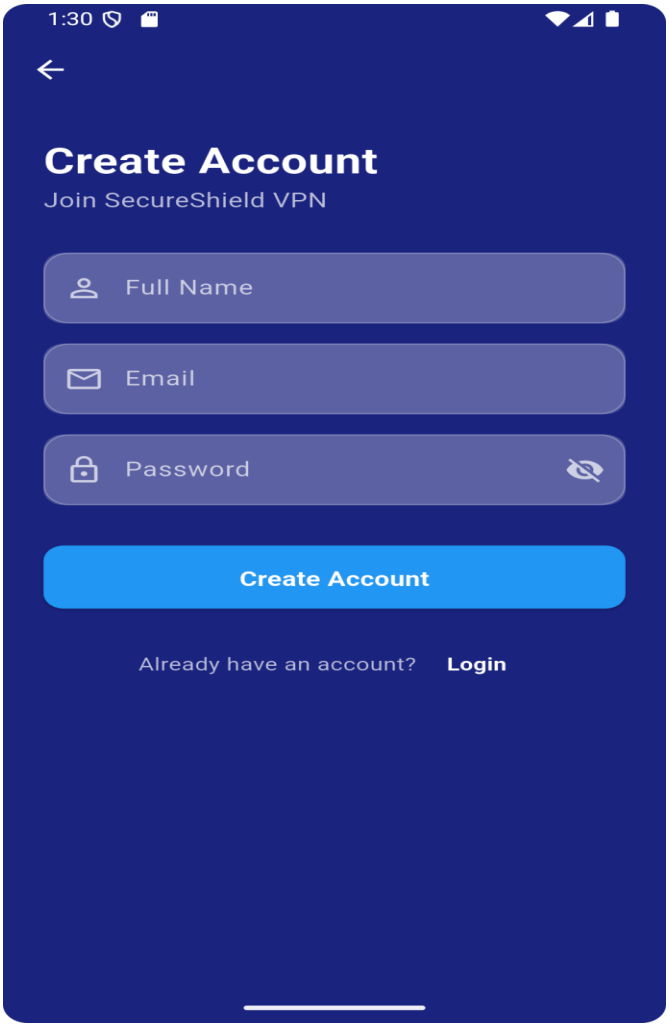
```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'dart:async';
import '../services/vpn_service.dart';

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

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

class _HomeScreenState extends
State<HomeScreen> {
  bool isConnected = false;
  String timerText = "00:00:00";
  Timer? _timer;
  int _seconds = 0;
  double downloadSpeed = 0;
  double uploadSpeed = 0;
  int ping = 0;
  String currentServer = "United States";

  @override
  void initState() {
    super.initState();
    print("HomeScreen initialized");

    // Add delay to handle any initialization
    Future.delayed(Duration.zero, () {
      // Get the VPN service
      final vpnService =
Provider.of<VPNService>(context, listen: false);

      // Check if already connected
      setState(() {
        isConnected = vpnService.isConnected;
        print("Initial connection state:
$isConnected");
      });

      // Start timer if connected
      if (isConnected) {
        startTimer();
      }
    });
  }

  void startTimer() {

  print("Starting timer");
    _timer = Timer.periodic(const Duration(seconds:
1), (timer) {
      setState(() {
        _seconds++;
        int hours = _seconds ~/ 3600;
        int minutes = (_seconds % 3600) ~/ 60;
        int seconds = _seconds % 60;
        timerText =
        '${hours.toString().padLeft(2, '0')}:'
            '${minutes.toString().padLeft(2, '0')}:'
            '${seconds.toString().padLeft(2, '0')}';

        downloadSpeed = (300 + (_seconds %
100)).toDouble();
        uploadSpeed = (150 + (_seconds %
50)).toDouble();
        ping = isConnected ? 8 : 0; // Fixed ping to
match UI mockup
      });
    });
  }

  void stopTimer() {
    print("Stopping timer");
    _timer?.cancel();
    setState(() {
      _seconds = 0;
      timerText = "00:00:00";
      downloadSpeed = 0;
      uploadSpeed = 0;
      ping = 0;
    });
  }

  void toggleConnection() async {
    print("Toggle connection button pressed");
    final vpnService =
Provider.of<VPNService>(context, listen: false);

    try {
      if (isConnected) {
        print("Disconnecting VPN");
        // Try to disconnect using Firestore, fallback to
mock method
        try {
          await vpnService.disconnectVPN(dataUsed:
```

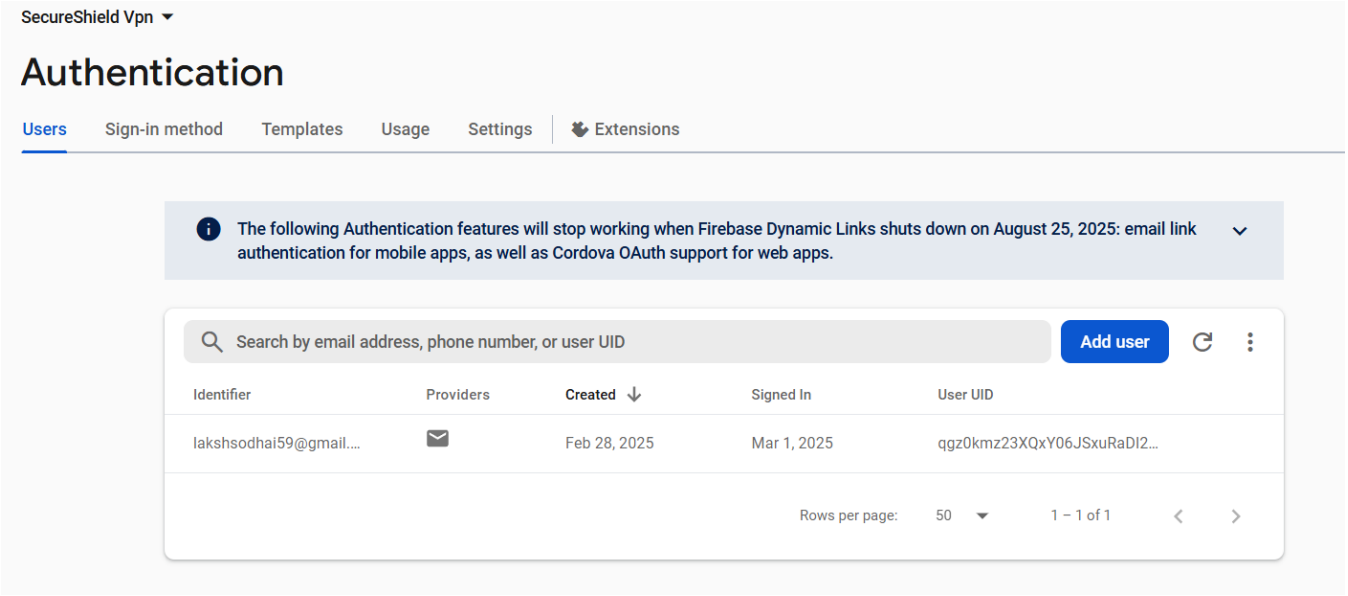
```
downloadSpeed + uploadSpeed);
        } catch (e) {
          print("Using mock disconnect due to
Firestore error: $e");
          await vpnService.mockDisconnect();
        }
        stopTimer();
      } else {
        print("Connecting VPN");
        // Try to connect using Firestore, fallback
to mock method
        try {
          await vpnService.connectVPN(
            serverId: 'us_server',
            serverName: currentServer,
          );
        } catch (e) {
          print("Using mock connect due to
Firestore error: $e");
          await
vpnService.mockConnect(currentServer);
        }
        startTimer();
      }
      setState(() {
        isConnected = !isConnected;
        print("Connection state changed to:
$isConnected");
      });
    } catch (e) {
      print("Error toggling connection: $e");

ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("Connection error:
$e")),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    print("Building HomeScreen, isConnected:
$isConnected");

    return Scaffold(
      appBar: AppBar(
        title: const Text('SecureShield VPN'),
        actions: [
          IconButton(
            icon: const
Icon(Icons.notifications_outlined),
            onPressed: () {},
          ),
          IconButton(
            icon: const Icon(Icons.info_outline),
            onPressed: () =>
Navigator.pushNamed(context, '/vpn-details'),
          ),
        ],
      ),
      backgroundColor: Colors.transparent,
      body: SingleChildScrollView(
        child: Padding(
          padding: const
EdgeInsets.symmetric(horizontal: 16),
          child: Column(
            children: [
              const SizedBox(height: 40),
              _buildConnectionButton(),
              const SizedBox(height: 20),
              _buildConnectionStatus(),
              const SizedBox(height: 20),
              _buildTimer(),
              const SizedBox(height: 40),
              _buildLocationAndPing(),
              const SizedBox(height: 40),
              _buildDataUsage(),
              const SizedBox(height: 40),
            ],
          ),
        ),
      ),
    );
  }

  Widget _buildConnectionButton() {
    return InkWell(
      onTap: toggleConnection,
      child: Container(
        width: 200,
        height: 200,
        decoration: BoxDecoration(
          shape: BoxShape.circle,
          color: isConnected ? Colors.green :
Colors.blue,
          boxShadow: [
            BoxShadow(
              color: isConnected ?
Colors.green.withOpacity(0.3) :
Colors.blue.withOpacity(0.3),
```

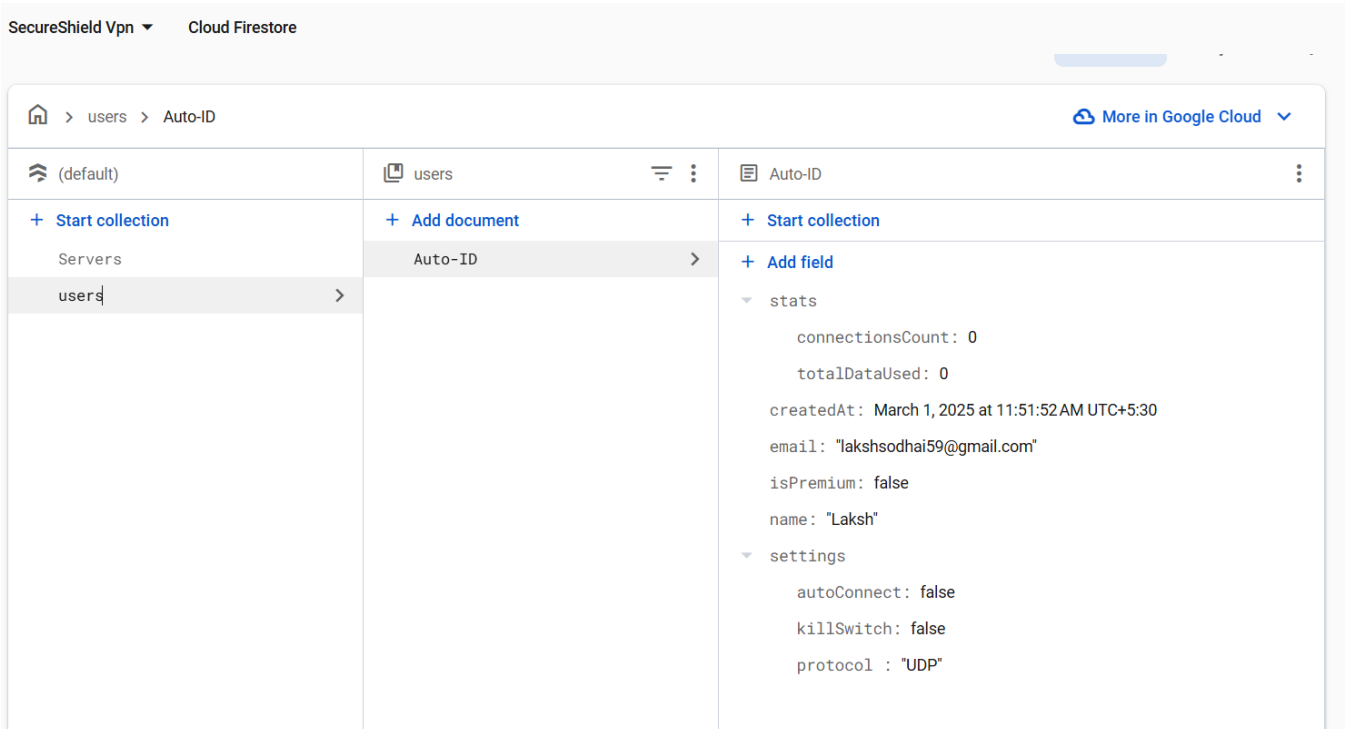
```
  spreadRadius: 10,
            blurRadius: 20,
          ),
        ],
      ),
      child: Column(
        mainAxisAlignment:
MainAxisAlignment.center,
        children: [
          Icon(
            isConnected
                ? Icons.power_settings_new
                : Icons.wifi_off_rounded,
            size: 50,
            color: Colors.white,
          ),
          const SizedBox(height: 8),
          Text(
            isConnected ? 'Connected' : 'Tap to
Connect',
            style: const TextStyle(
              color: Colors.white,
              fontSize: 16,
              fontWeight: FontWeight.bold,
            ),
          ),
        ],
      ),
    ),
  );
 }

 Widget _buildConnectionStatus() {
  return Container(
    padding: const
EdgeInsets.symmetric(horizontal: 30, vertical:
12),
    decoration: BoxDecoration(
      color: isConnected ? Colors.green :
Colors.blue,
      borderRadius: BorderRadius.circular(30),
    ),
    child: Text(
      isConnected ? 'Protected' : 'Not Protected',
      style: const TextStyle(
        color: Colors.white,
        fontWeight: FontWeight.bold,
      ),
    ),
  );
 }

 Widget _buildTimer() {
  return Text(
    timerText,
    style: const TextStyle(
      fontSize: 48,
      fontWeight: FontWeight.bold,
      color: Colors.white,
    ),
  );
 }

 Widget _buildLocationAndPing() {
  return Row(
    mainAxisAlignment:
MainAxisAlignment.center,
    children: [
      Expanded(
        child: Container(
          margin: const EdgeInsets.only(right: 8),
          padding: const
EdgeInsets.symmetric(vertical: 24, horizontal: 16),
          decoration: BoxDecoration(
            color:
Colors.blue.shade800.withOpacity(0.5),
            borderRadius: BorderRadius.circular(15),
          ),
          child: Column(
            children: [
              const Text(
                'us',
                style: TextStyle(fontSize: 30),
              ),
              const SizedBox(height: 8),
              const Text(
                'United States',
                style: TextStyle(
                  color: Colors.white,
                  fontWeight: FontWeight.bold,
                ),
              ),
              const Text(
                'FREE',
                style: TextStyle(color: Colors.white70),
              ),
            ],
          ),
        ),
      ),
```

```
Expanded(
        child: Container(
          margin: const EdgeInsets.only(left: 8),
          padding: const
EdgeInsets.symmetric(vertical: 24, horizontal:
16),
          decoration: BoxDecoration(
            color:
Colors.blue.shade800.withOpacity(0.5),
            borderRadius:
BorderRadius.circular(15),
          ),
          child: Column(
            children: [
              Icon(Icons.speed, color:
Colors.orange, size: 30),
              const SizedBox(height: 8),
              Text(
                '$ping ms',
                style: const TextStyle(
                  color: Colors.white,
                  fontWeight: FontWeight.bold,
                ),
              ),
              const Text(
                'PING',
                style: TextStyle(color:
Colors.white70),
              ),
            ],
          ),
        ),
      ),
    ],
  );
 }

 Widget _buildDataUsage() {
  return Row(
    mainAxisAlignment:
MainAxisAlignment.center,
    children: [
      Expanded(
        child: Container(
          margin: const EdgeInsets.only(right: 8),
          padding: const EdgeInsets.all(16),
          decoration: BoxDecoration(
            color:
Colors.blue.shade800.withOpacity(0.5),
            borderRadius:
BorderRadius.circular(15),
          ),
          child: Column(
            children: [
              Icon(Icons.download, color:
Colors.green, size: 30),
              const SizedBox(height: 8),
              Text(
                '${downloadSpeed.toStringAsFixed(1)}
KB/s',
                style: const TextStyle(
                  color: Colors.white,
                  fontWeight: FontWeight.bold,
                ),
              ),
              const Text(
                'DOWNLOAD',
                style: TextStyle(color: Colors.white70),
              ),
            ],
          ),
        ),
      ),
      Expanded(
        child: Container(
          margin: const EdgeInsets.only(left: 8),
          padding: const EdgeInsets.all(16),
          decoration: BoxDecoration(
            color:
Colors.blue.shade800.withOpacity(0.5),
            borderRadius: BorderRadius.circular(15),
          ),
          child: Column(
            children: [
              Icon(Icons.upload, color: Colors.blue,
size: 30),
              const SizedBox(height: 8),
              Text(
                '${uploadSpeed.toStringAsFixed(1)}
KB/s',
                style: const TextStyle(
                  color: Colors.white,
                  fontWeight: FontWeight.bold,
                ),
              ),
              const Text(
                'UPLOAD',
                style: TextStyle(color: Colors.white70),
              ),
            ],
```


1:15
SecureShield VPN
Secure • Fast • Reliable
Email
Password
Forgot Password?
Login
Or continue with
G
Don't have an account? Sign Up


1:30
Create Account
Join SecureShield VPN
Full Name
Email
Password
Create Account
Already have an account? Login


SecureShield Vpn
Authentication
Users
Sign-in method
Templates
Usage
Settings
Extensions
The following Authentication features will stop working when Firebase Dynamic Links shuts down on August 25, 2025: email link authentication for mobile apps, as well as Cordova OAuth support for web apps.
Search by email address, phone number, or user UID
Add user
Identifier
Providers
Created
Signed In
User UID
lakshsodhai59@gmail....
Feb 28, 2025
Mar 1, 2025
qgz0kmz23XQxY06JSxuRaDI2...
Rows per page:
50
1 – 1 of 1


SecureShield Vpn
Cloud Firestore
users
Auto-ID
More in Google Cloud
(default)
Start collection
Servers
users
users
Add document
Auto-ID
Auto-ID
Start collection
Add field
stats
connectionsCount: 0
totalDataUsed: 0
createdAt: March 1, 2025 at 11:51:52 AM UTC+5:30
email: "lakshsodhai59@gmail.com"
isPremium: false
name: "Laksh"
settings
autoConnect: false
killSwitch: false
protocol : "UDP"

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.	57
Name	Laksh Sodhai
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.

<u>Weaknesses:</u>

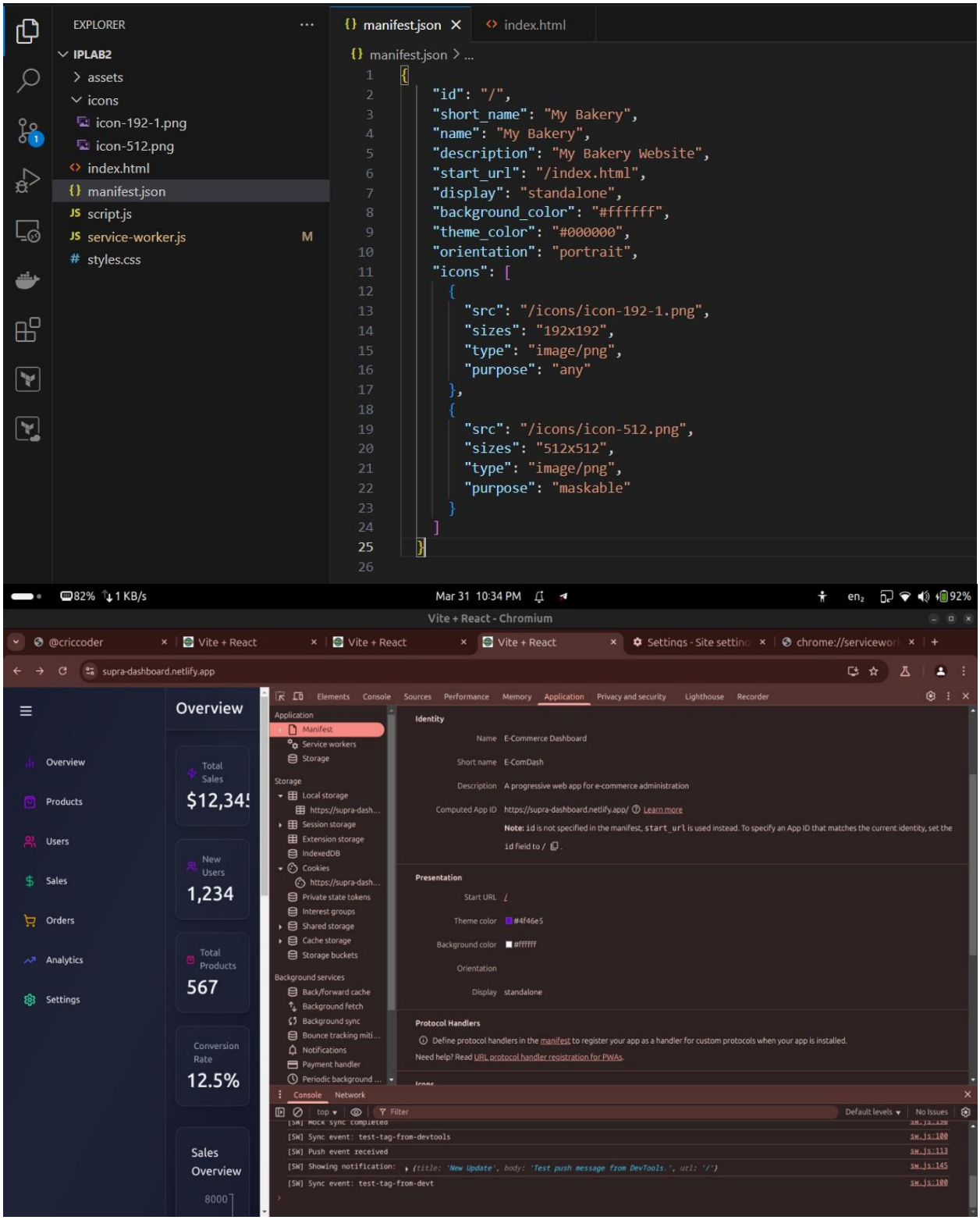
- IOS support from version 11.3 onwards
- Greater use of the device battery
- Not all devices support the full range of PWA features (same speech for iOS and Android operating systems)
- It is not possible to establish a strong re-engagement for iOS users (URL scheme, standard web notifications)
- Support 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


EXPLORER
IPLAB2
assets
icons
icon-192-1.png
icon-512.png
index.html
manifest.json
script.js
service-worker.js M
styles.css
manifest.json
index.html
{
"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"
}
]
}
82% 1 KB/s
Mar 31 10:34 PM
en2
92%
Vite + React - Chromium
@criccoder
Vite + React
Vite + React
Vite + React
Settings - Site setting
chrome://servicewor
supra-dashboard.netlify.app
Overview
Overview
Products
Users
Sales
Orders
Analytics
Settings
Total Sales
$12,345
New Users
1,234
Total Products
567
Conversion Rate
12.5%
Sales Overview
8000
Elements
Console
Sources
Performance
Memory
Application
Privacy and security
Lighthouse
Recorder
Application
Manifest
Service workers
Storage
Storage
Local storage
https://supra-dash...
Session storage
Extension storage
IndexedDB
Cookies
https://supra-dash...
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking miti...
Notifications
Payment handler
Periodic background ...
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
top
Filter
Default levels
No Issues
[SW] Sync event: test-tag-from-devtools sw.js:100
[SW] Push event received sw.js:113
[SW] Showing notification: {title: 'New Update', body: 'Test push message from DevTools.', url: '/'} sw.js:145
[SW] Sync event: test-tag-from-devt sw.js:100

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.	57
Name	Laksh Sodhai
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.

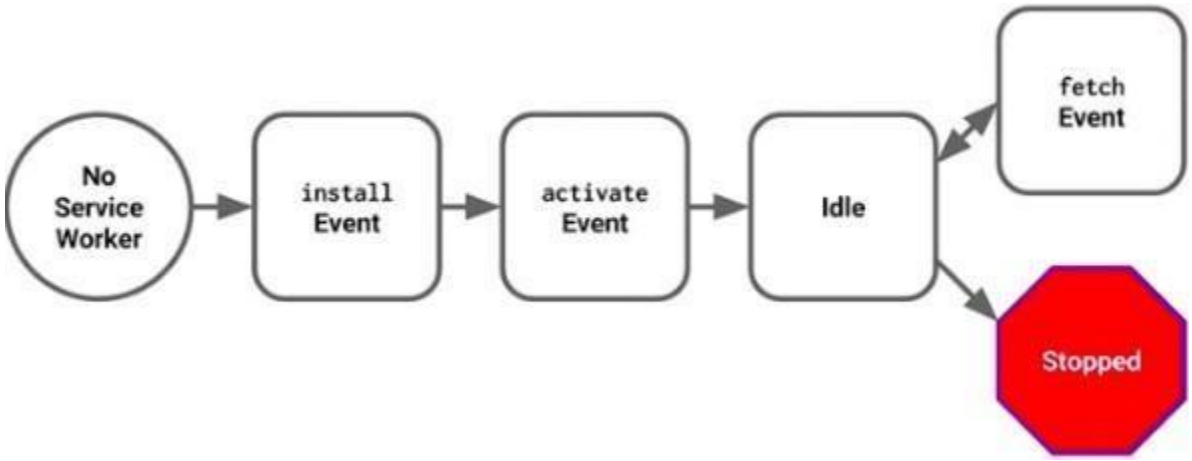
- You can manage **Push Notifications**
  You can manage push notifications with Service Worker and show any information message to the user.
- You can **Continue**
  Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

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.

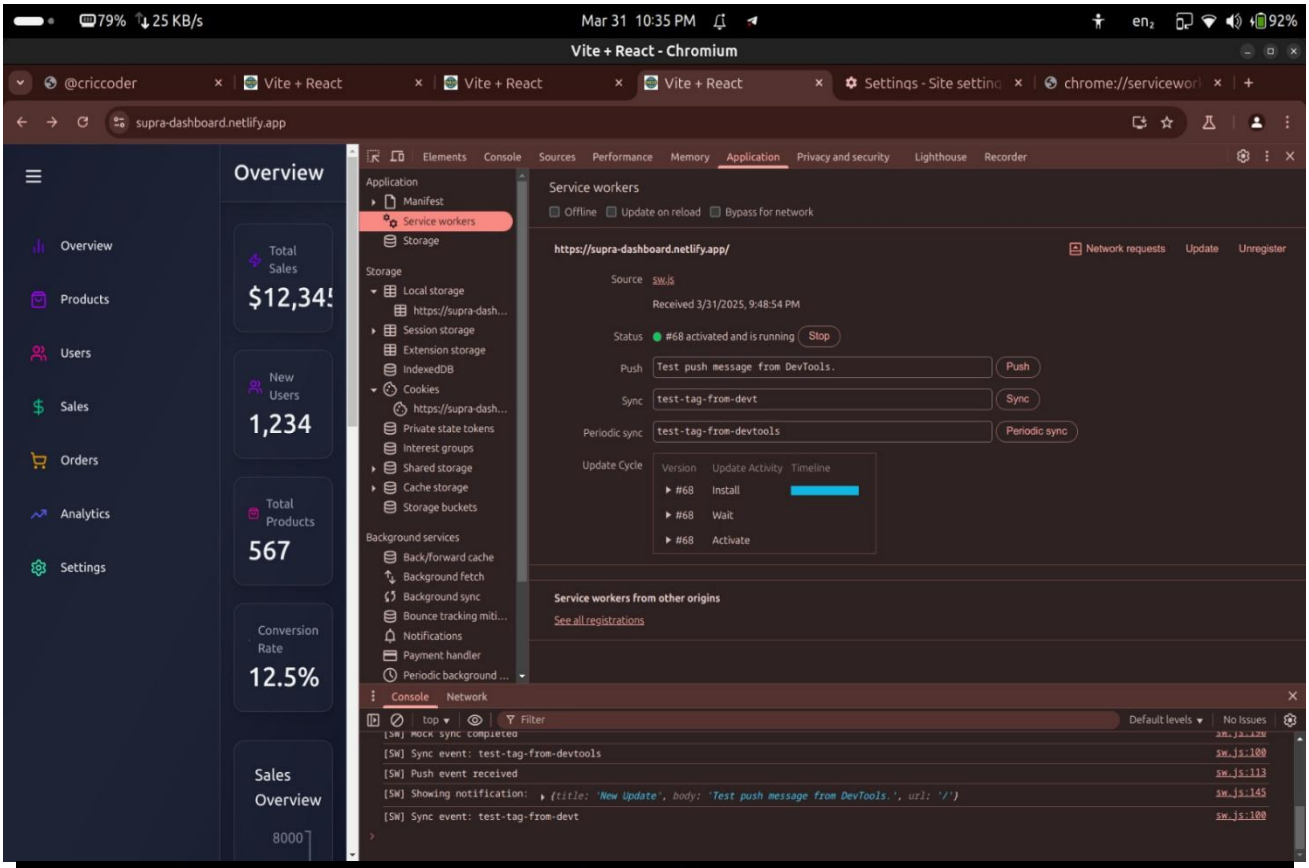
Code:

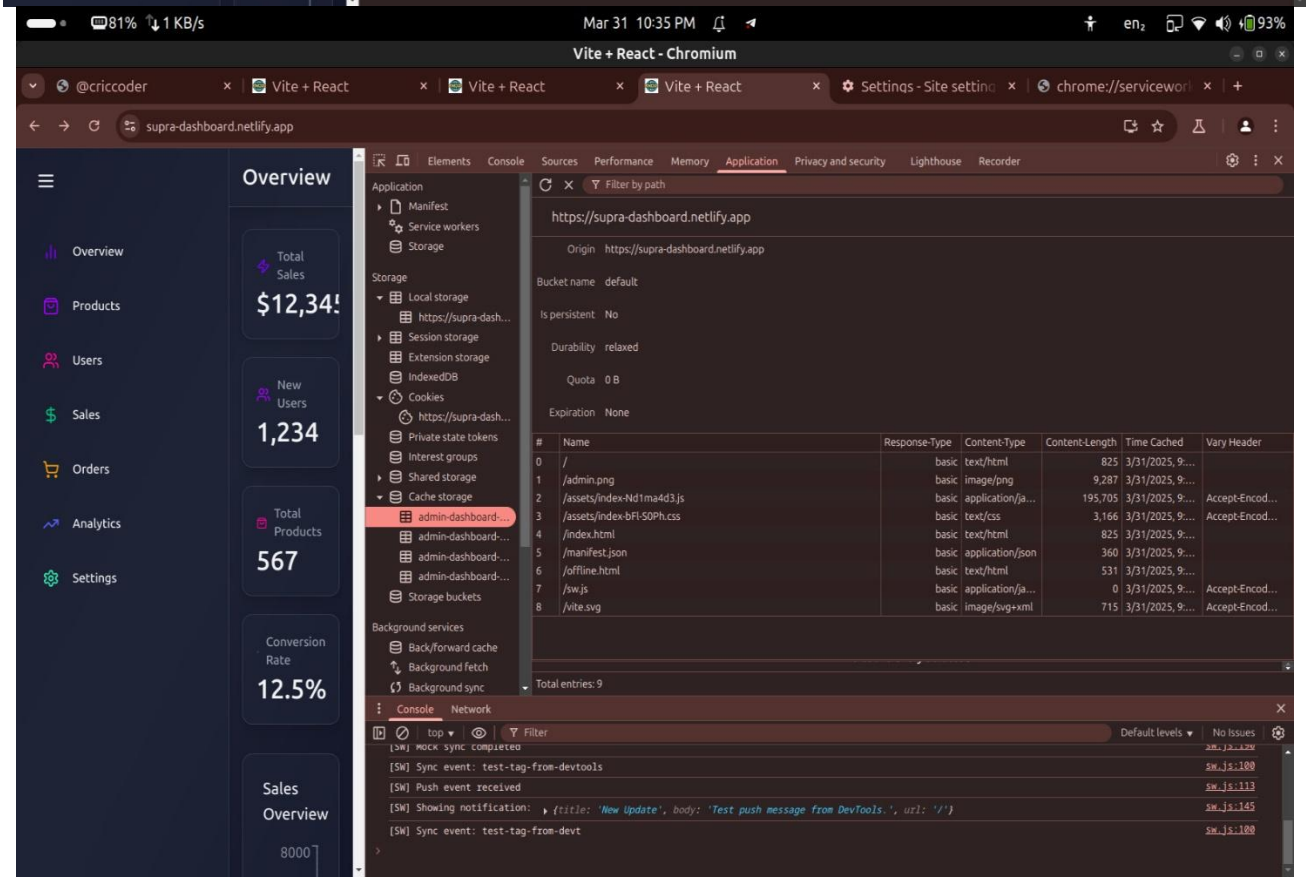
```
<script>
      if ('serviceWorker' in navigator) {
        window.addEventListener('load', () => {
        navigator.serviceWorker.register('/service-worker.js')
           .then((registration) => {
             console.log('Service Worker registered with scope: ', registration.scope);
           })
           .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
92%
Vite + React - Chromium
@criccoder
Vite + React
Settings - Site setting
chrome://serviceworl
supra-dashboard.netlify.app
Overview
Total Sales
$12,345
New Users
1,234
Total Products
567
Conversion Rate
12.5%
Sales Overview
8000
Products
Users
Sales
Orders
Analytics
Settings
Elements
Console
Sources
Performance
Memory
Application
Privacy and security
Lighthouse
Recorder
Service workers
Offline
Update on reload
Bypass for network
https://supra-dashboard.netlify.app/
Network requests
Update
Unregister
Source sw.js
Received 3/31/2025, 9:48:54 PM
Status #68 activated and is running
Stop
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devt
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#68 Install
#68 Wait
#68 Activate
Service workers from other origins
See all registrations
Manifest
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking miti...
Notifications
Payment handler
Periodic background ...
Console
Network
Filter
Default levels
No Issues
[SW] Sync event: test-tag-from-devtools
sw.js:100
[SW] Push event received
sw.js:113
[SW] Showing notification:
{title: 'New Update', body: 'Test push message from DevTools.', url: '/'}
sw.js:145
[SW] Sync event: test-tag-from-devt
sw.js:100


81% 1 KB/s
Mar 31 10:35 PM
93%
Vite + React - Chromium
@criccoder
Vite + React
Settings - Site setting
chrome://serviceworl
supra-dashboard.netlify.app
Overview
Total Sales
$12,345
New Users
1,234
Total Products
567
Conversion Rate
12.5%
Sales Overview
8000
Products
Users
Sales
Orders
Analytics
Settings
Elements
Console
Sources
Performance
Memory
Application
Privacy and security
Lighthouse
Recorder
Filter by path
https://supra-dashboard.netlify.app
Origin https://supra-dashboard.netlify.app
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-Nd1ma4d3.js basic application/ja... 195,705 3/31/2025, 9:... Accept-Encod...
3 /assets/index-bFl-S0Ph.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/ja... 0 3/31/2025, 9:... Accept-Encod...
8 /vite.svg basic image/svg+xml 715 3/31/2025, 9:... Accept-Encod...
Total entries: 9
Manifest
Service workers
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
admin-dashboard-...
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Console
Network
Filter
Default levels
No Issues
[SW] Sync event: test-tag-from-devtools
sw.js:100
[SW] Push event received
sw.js:113
[SW] Showing notification:
{title: 'New Update', body: 'Test push message from DevTools.', url: '/'}
sw.js:145
[SW] Sync event: test-tag-from-devt
sw.js:100

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.	57
Name	Laksh Sodhai
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”.

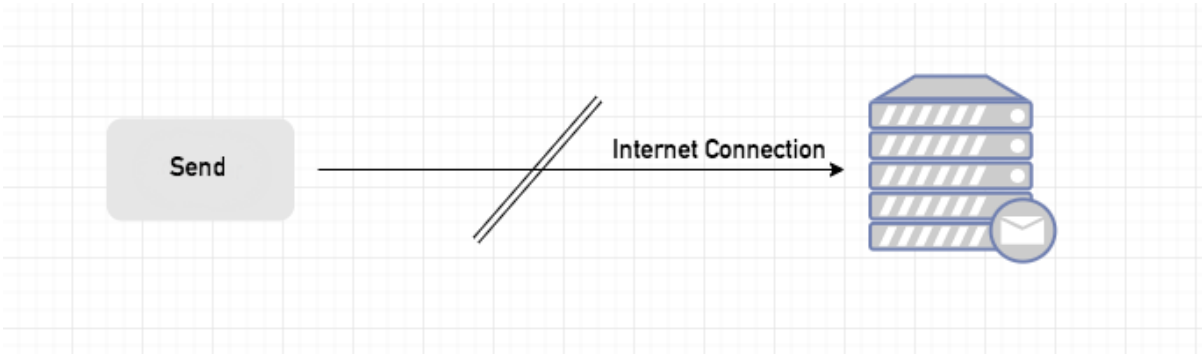
- **CacheFirst** - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.
- **NetworkFirst** - In this function, firstly we can try getting an updated response from the network, if this process completed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

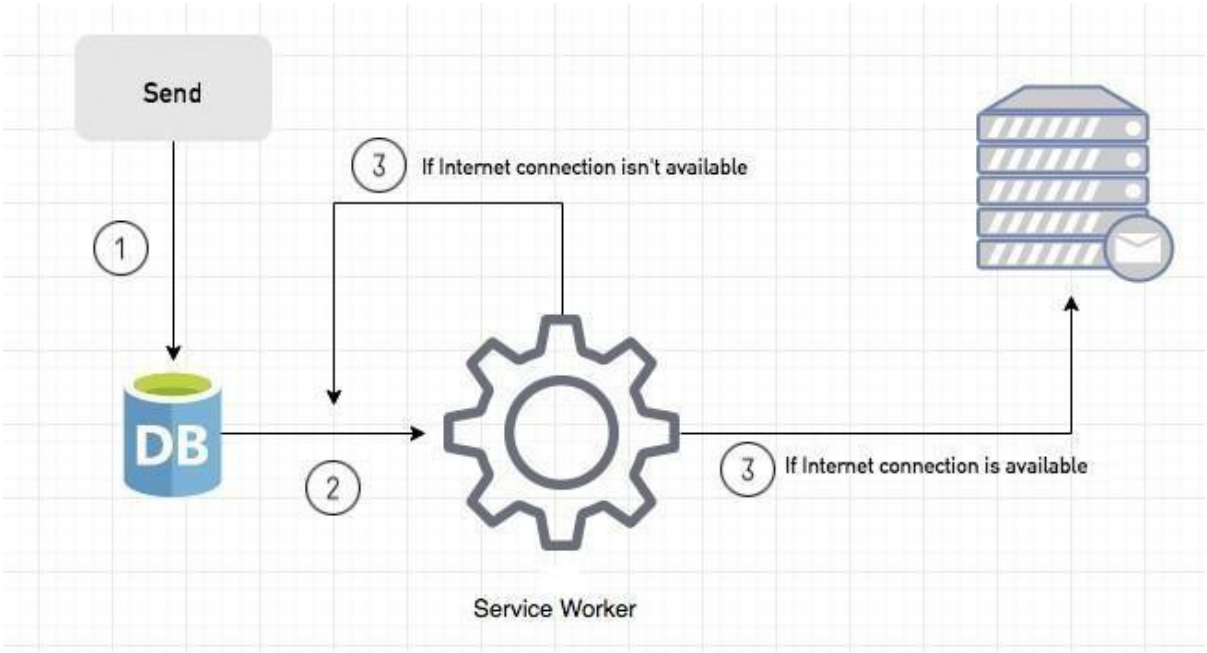
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are 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

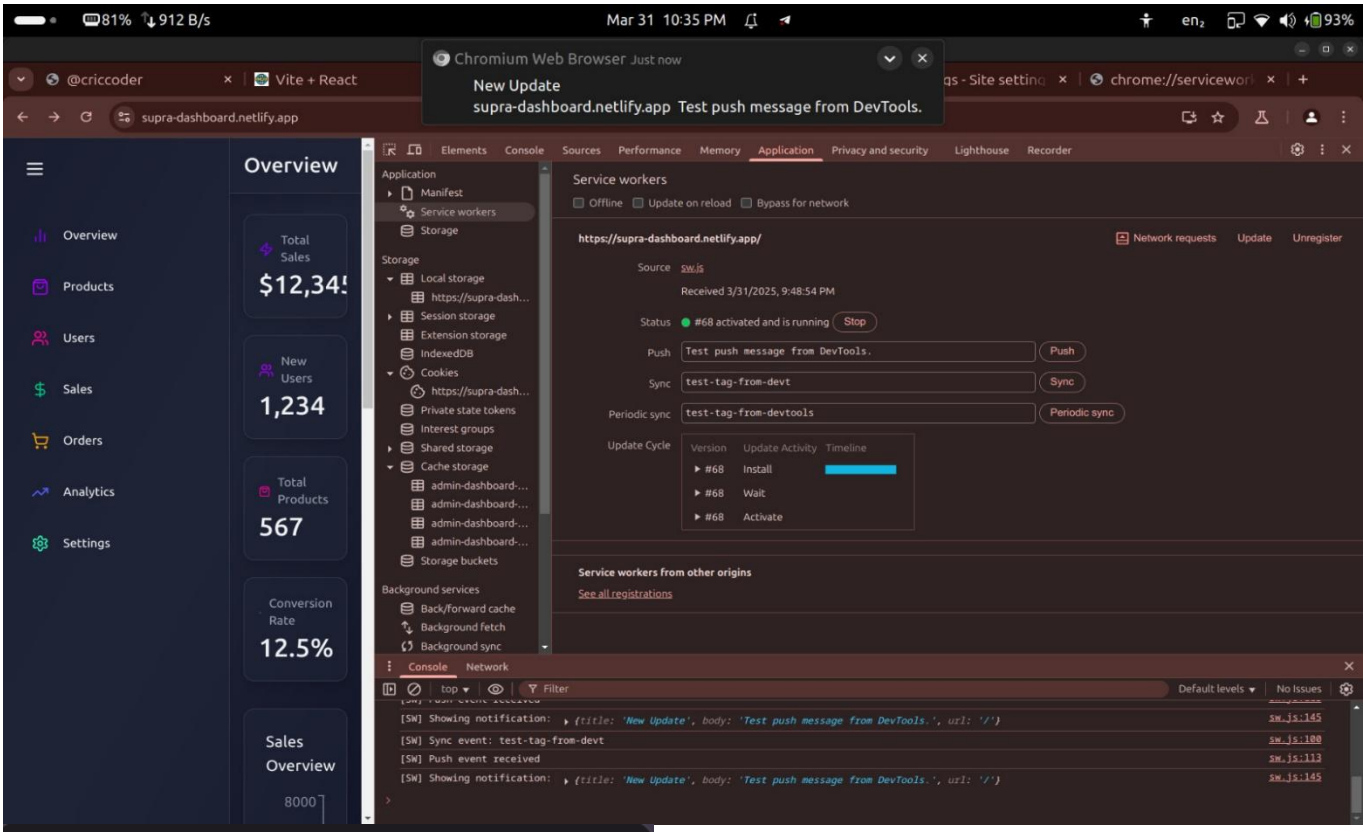
```
self.addEventListener('sync', event => {
    if (event.tag == 'helloSync') {
        console.log("helloSync [sw.js]");
    }
});
```

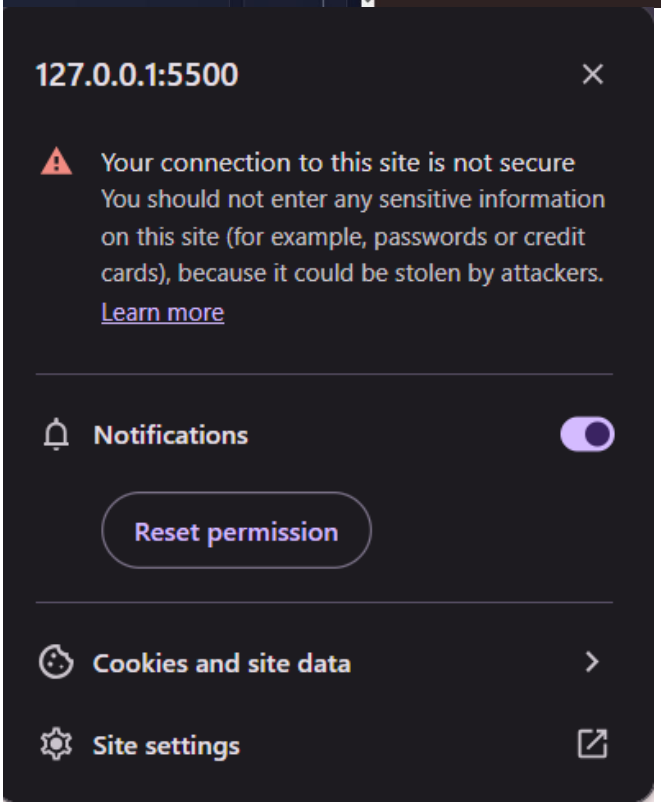
Push Event

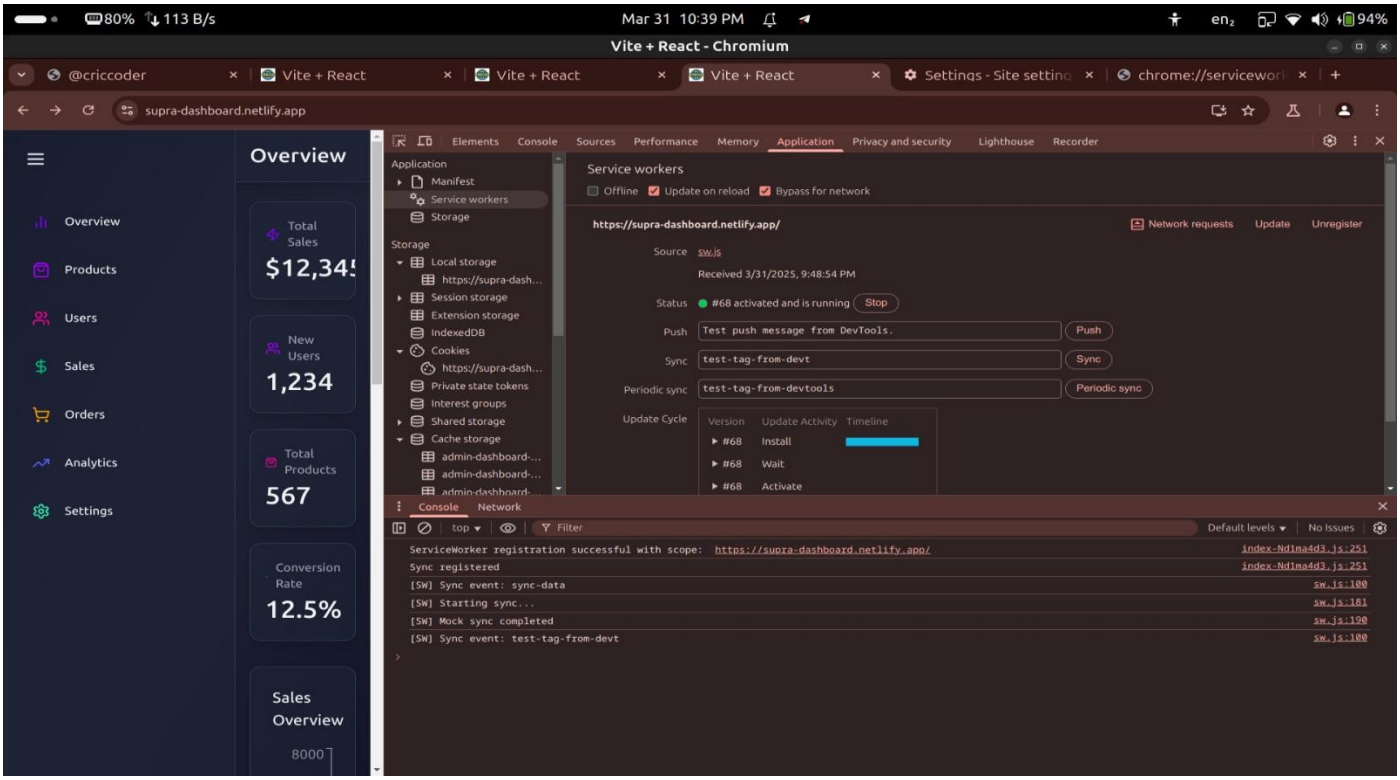
This is the event that handles push notifications that are received from the server. You can apply any method with received data.

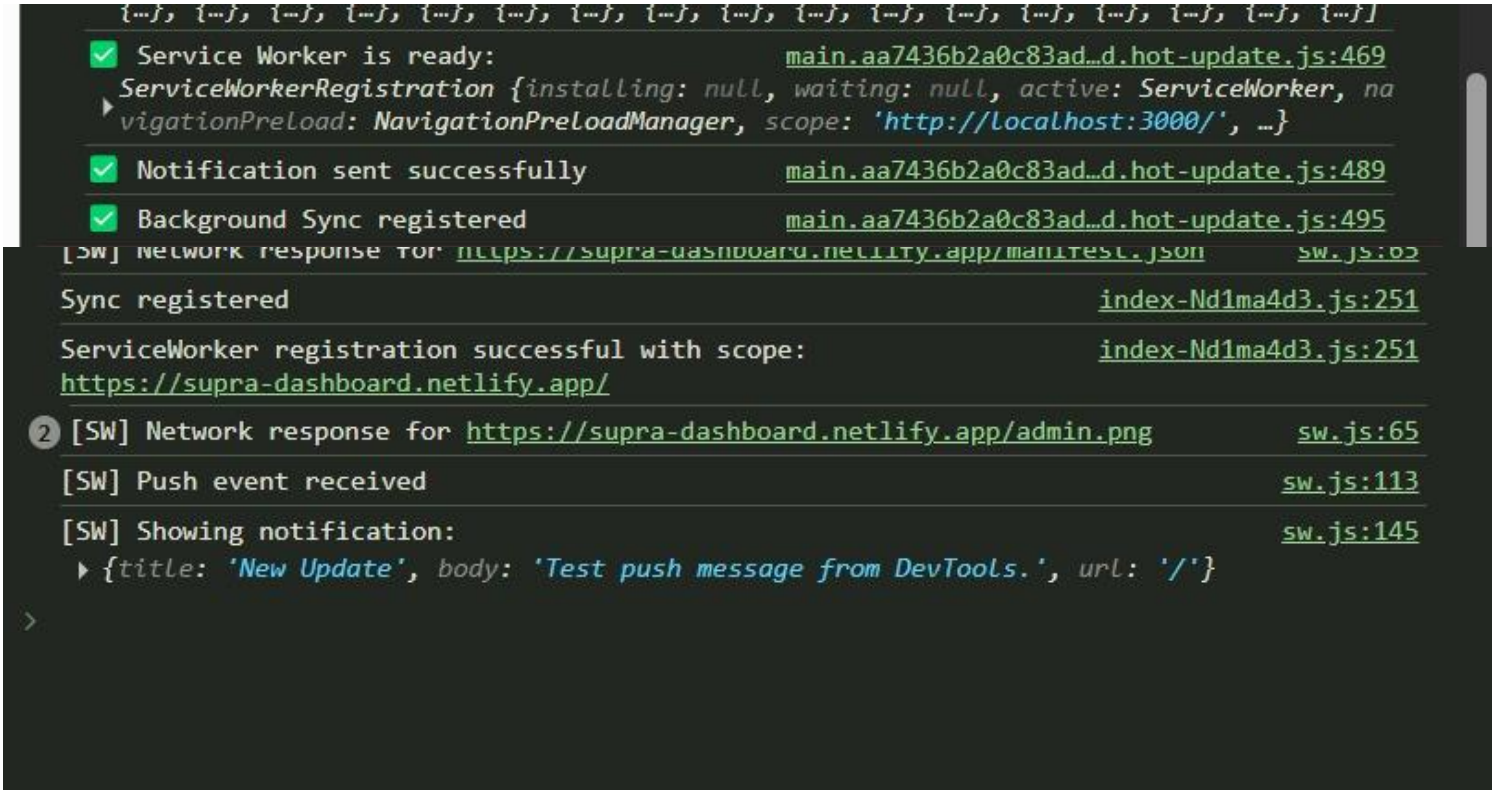
We can check in the following example.
“Notification.requestPermission();” is the necessary line to show notification to the user. If you don’t want to show any notification, you don’t need this line.

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


81% 912 B/s
Mar 31 10:35 PM
en2
93%
Chromium Web Browser Just now
New Update
supra-dashboard.netlify.app Test push message from DevTools.
@criccoder
Vite + React
chrome://serviceworkers
supra-dashboard.netlify.app
Overview
Overview
Products
Users
Sales
Orders
Analytics
Settings
Total Sales
$12,345
New Users
1,234
Total Products
567
Conversion Rate
12.5%
Sales Overview
8000
Elements
Console
Sources
Performance
Memory
Application
Privacy and security
Lighthouse
Recorder
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Service workers
Offline
Update on reload
Bypass for network
https://supra-dashboard.netlify.app/
Network requests
Update
Unregister
Source sw.js
Received 3/31/2025, 9:48:54 PM
Status #68 activated and is running
Stop
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devt
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#68 Install
#68 Wait
#68 Activate
Service workers from other origins
See all registrations
Console
Network
top
Filter
Default levels
No Issues
[SW] Showing notification: {title: 'New Update', body: 'Test push message from DevTools.', url: '/'}
sw.js:145
[SW] Sync event: test-tag-from-devt
sw.js:100
[SW] Push event received
sw.js:113
[SW] Showing notification: {title: 'New Update', body: 'Test push message from DevTools.', url: '/'}
sw.js:145


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
Notifications
Reset permission
Cookies and site data
Site settings


Vite + React - Chromium
supra-dashboard.netlify.app
Overview
Total Sales
$12,345
New Users
1,234
Total Products
567
Conversion Rate
12.5%
Sales Overview
Products
Users
Sales
Orders
Analytics
Settings
Service workers
Offline
Update on reload
Bypass for network
https://supra-dashboard.netlify.app/
Network requests
Update
Unregister
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
ServiceWorker registration successful with scope: https://supra-dashboard.netlify.app/
Sync registered
[SW] Sync event: sync-data
[SW] Starting sync...
[SW] Mock sync completed
[SW] Sync event: test-tag-from-devt


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
Sync registered
index-Nd1ma4d3.js:251
ServiceWorker registration successful with scope:
https://supra-dashboard.netlify.app/
index-Nd1ma4d3.js:251
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.	57
Name	Laksh Sodhai
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 developers stacks

Pros

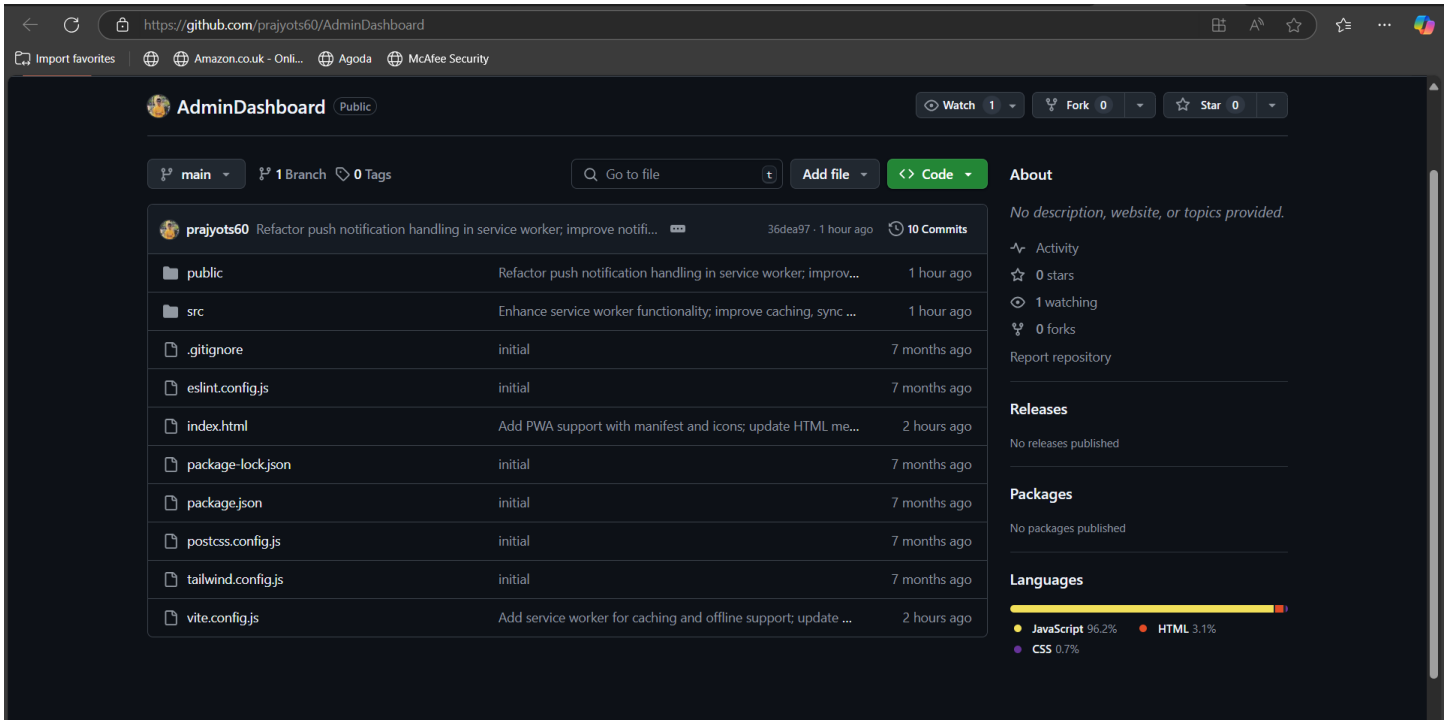
1. Hosted by Google. Enough said.
2. Authentication, Cloud Messaging, and a whole lot of other handy services will be available to you.
3. A real-time database will be available to you, which can store 1 GB of data.
4. You'll also have access to a blob store, which can store another 1 GB of data.
5.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

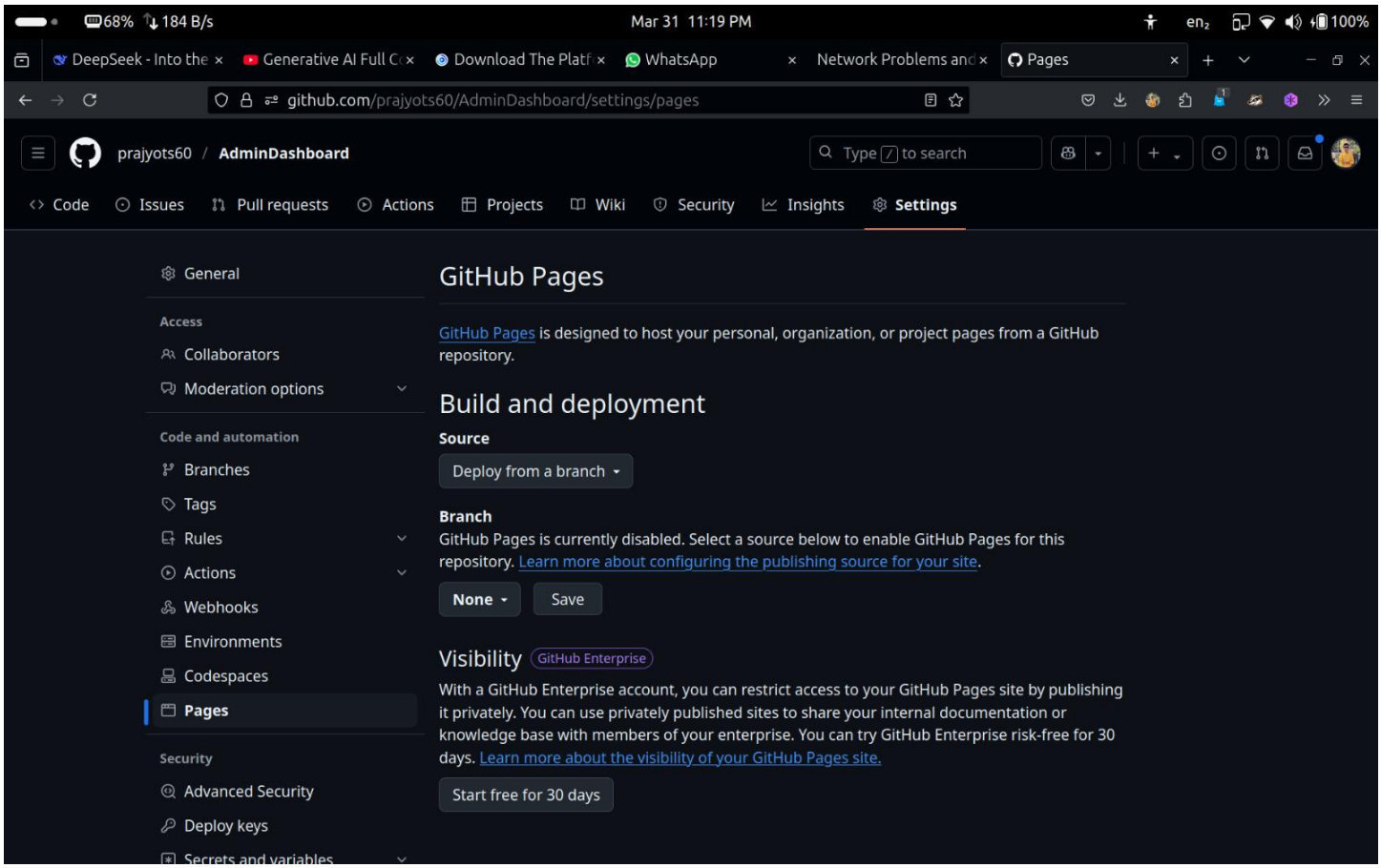
Cons

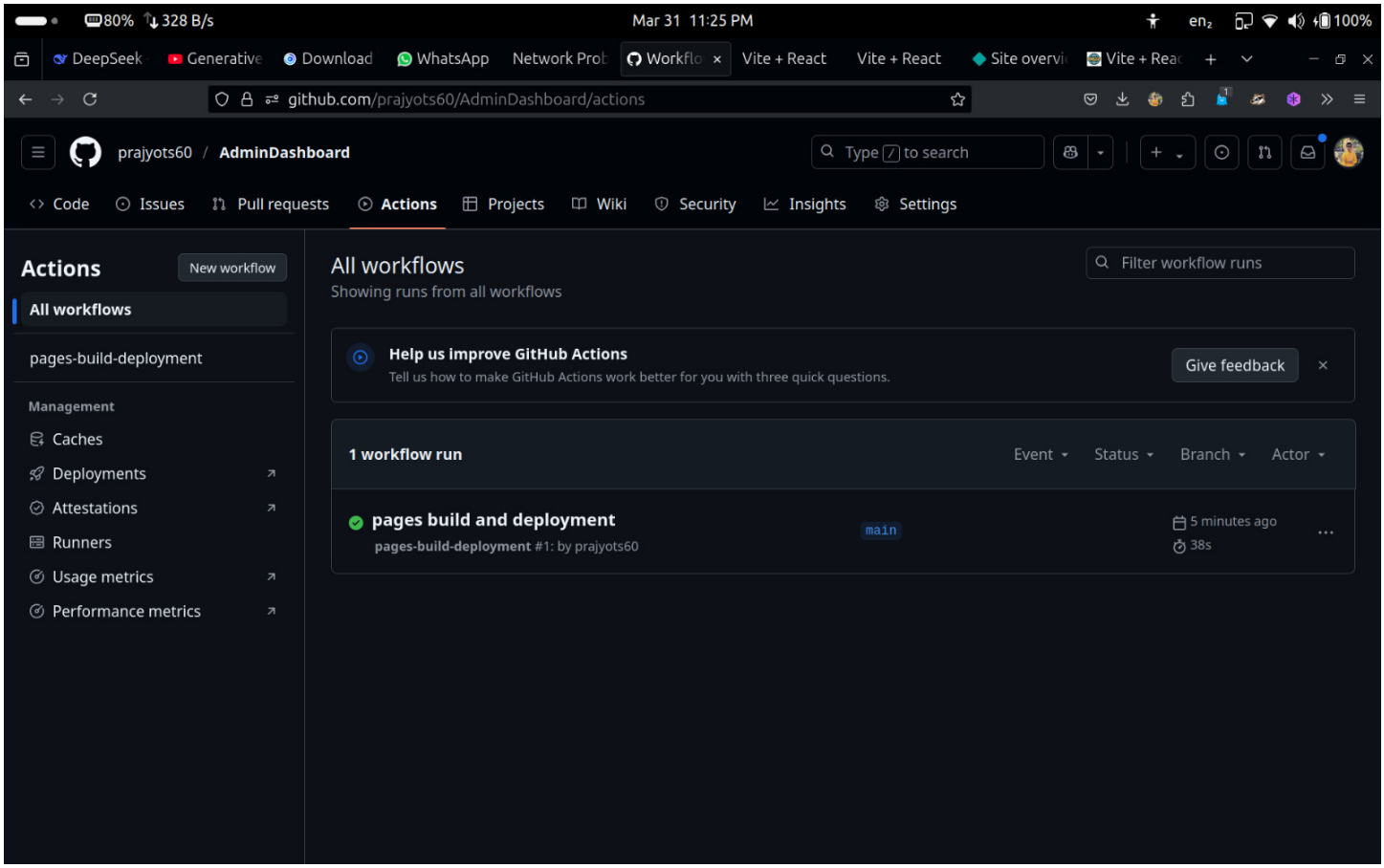
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator.

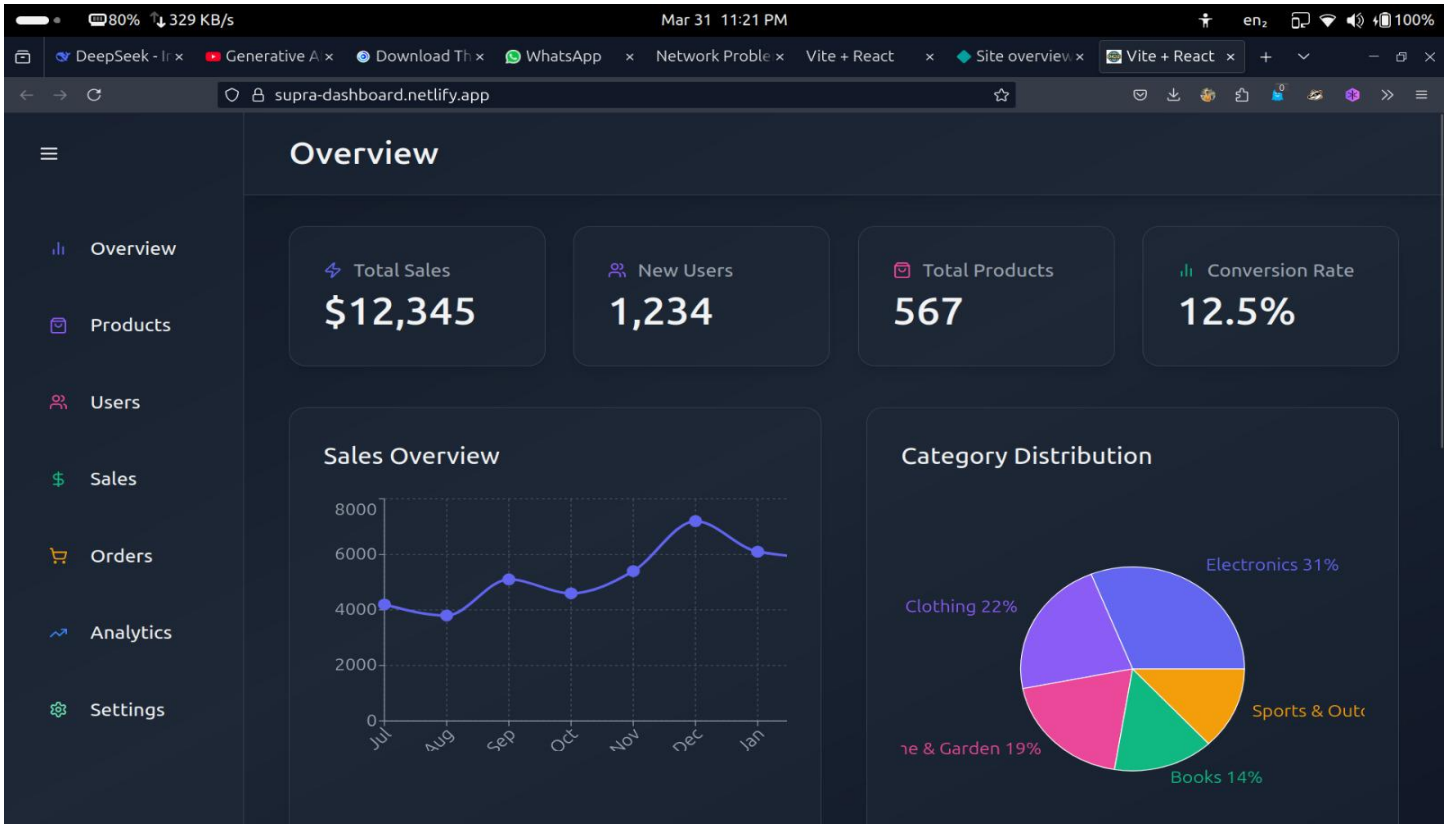
Link to our GitHub repository: https://github.com/prajyots60/AdminDashboard

Github Screenshot:









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.	57
Name	Laksh Sodhai
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:

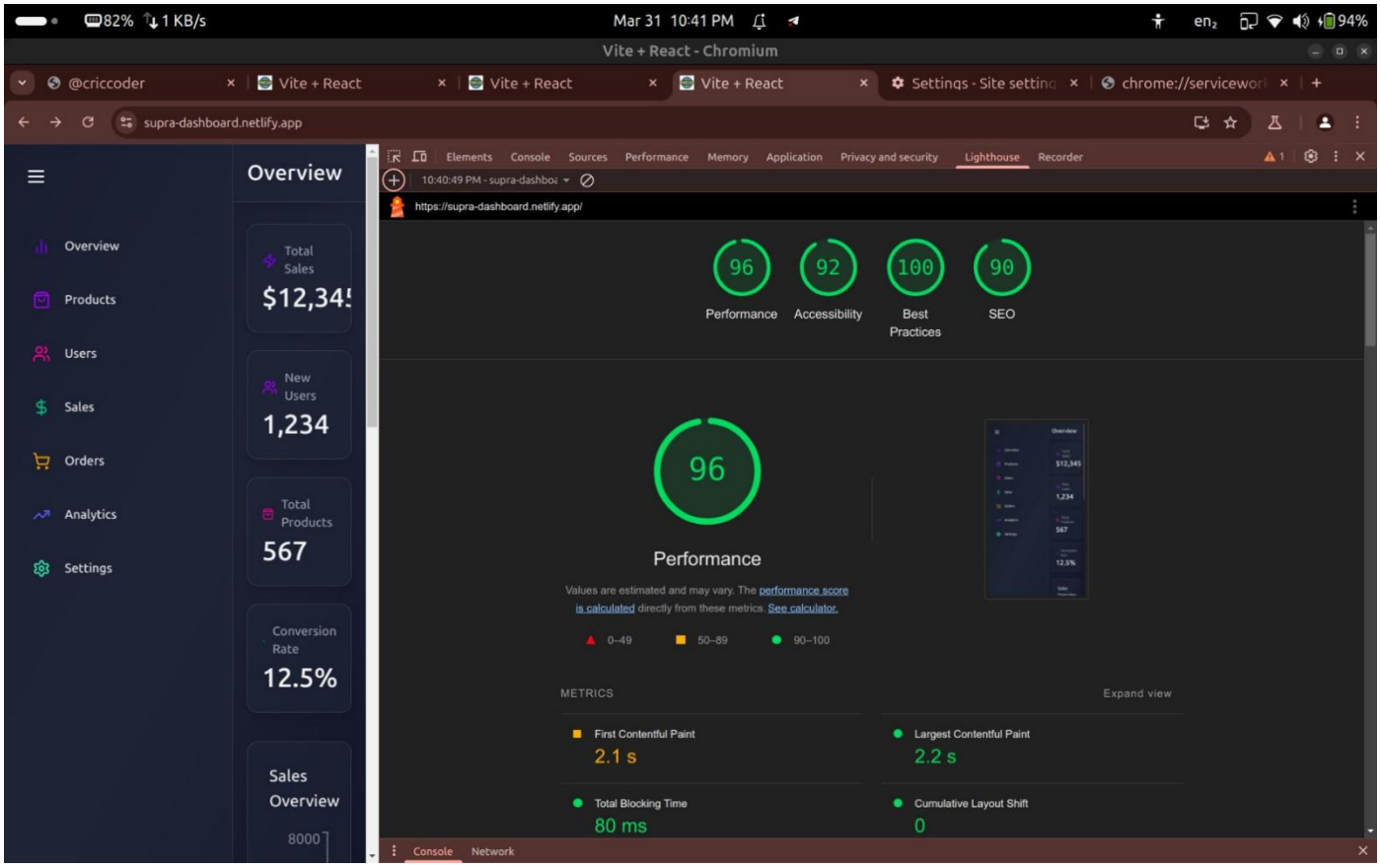
1. **Performance:** This score is an aggregation of how the page fared in aspects such as (but not limited to) loading speed, time taken for loading for basic frame(s), displaying meaningful content to the user, etc. To a layman, this score is indicative of how decently the site performs, with a score 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.

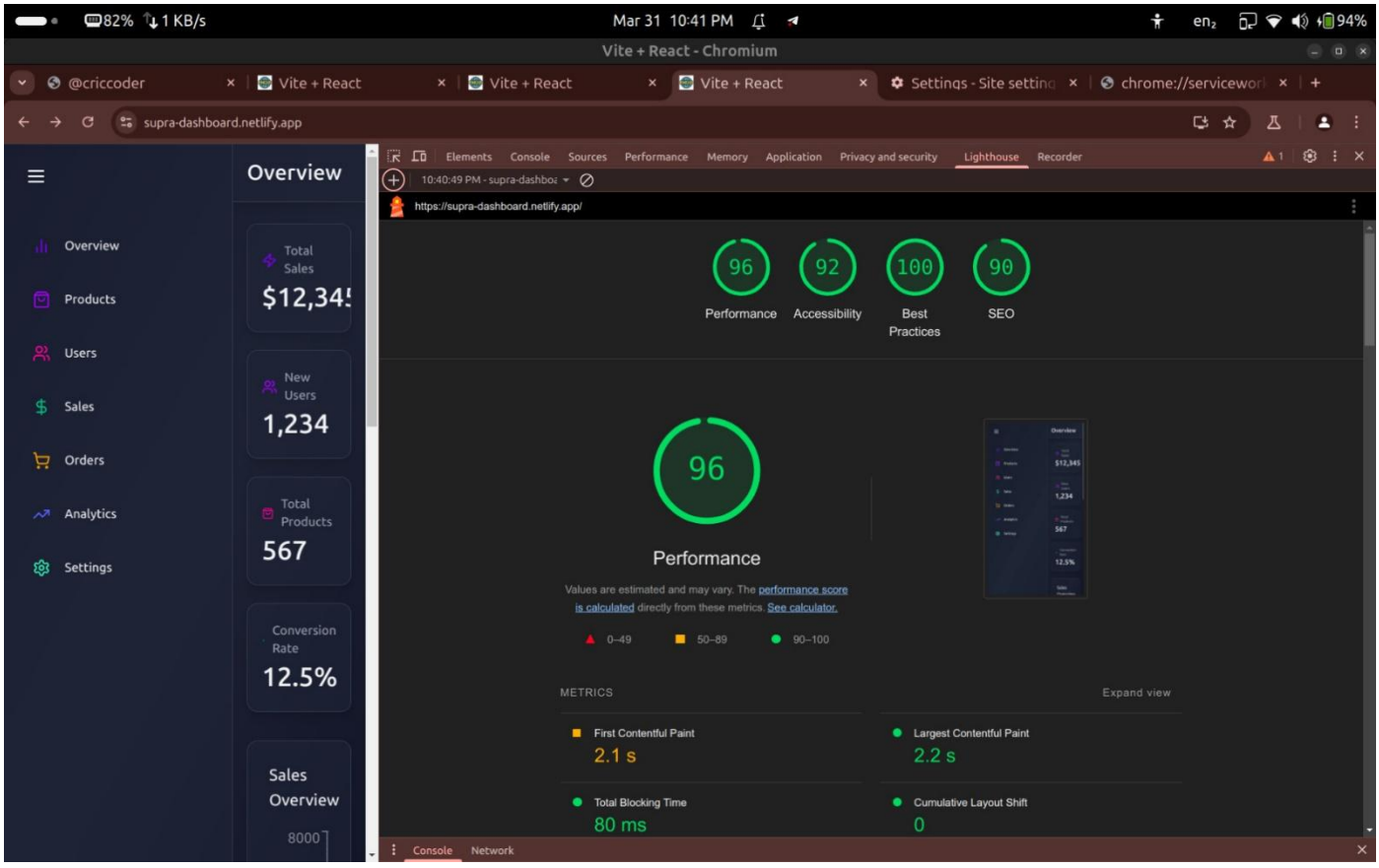
4. **Best Practices:** As any developer would know, there are a number of practices that have been deemed ‘best’ based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS
   Avoiding the use of deprecated code elements like tags, directives, 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.

MAD & PWA Lab
Journal

Experiment No.	Assignment-1
Assignment 1 Questions	1. Flutter Overview:Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community. 2. Widget Tree and Composition: Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree. 3. State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable. 4. Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.
Roll No.	**57**
Name	**Laksh Vijay Sodhai**
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

MAD & PWA Lab

Journal

Experiment No.	Assignment-2
Assignment 2 Questions	1. Define Progressive Web App (PWA) and explain its significance in modern web development.Discuss the key characteristics that differentiate PWAs from traditional mobile apps 2. Define responsive web design and explain its importance in the context of Progressive Web Apps.Compare and contrast responsive, fluid, and adaptive web design approaches. 3. Describe the lifecycle of Service Workers, including registration, installation, and activation phases. 4. Explain the use of IndexedDB in the Service Worker for data storage.
Roll No.	**57**
Name	Laksh Vijay Sodhai
Class	D15A/D15B
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:	