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 Manav Narain Punjabi of **D15A/D15B** semester **VI,** have successfully completed necessary experiments in the **MAD & PWA Lab** under my supervision in **VES Institute of Technology** during the academic year **2024-2025.**

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Project Title: ApnaChat App **Roll No.** 44

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

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

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers : Mrs. Kajal Joseph.

Email : kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: ApnaChat App **Roll No.** 44

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: ApnaChat App **Roll No.** 44

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

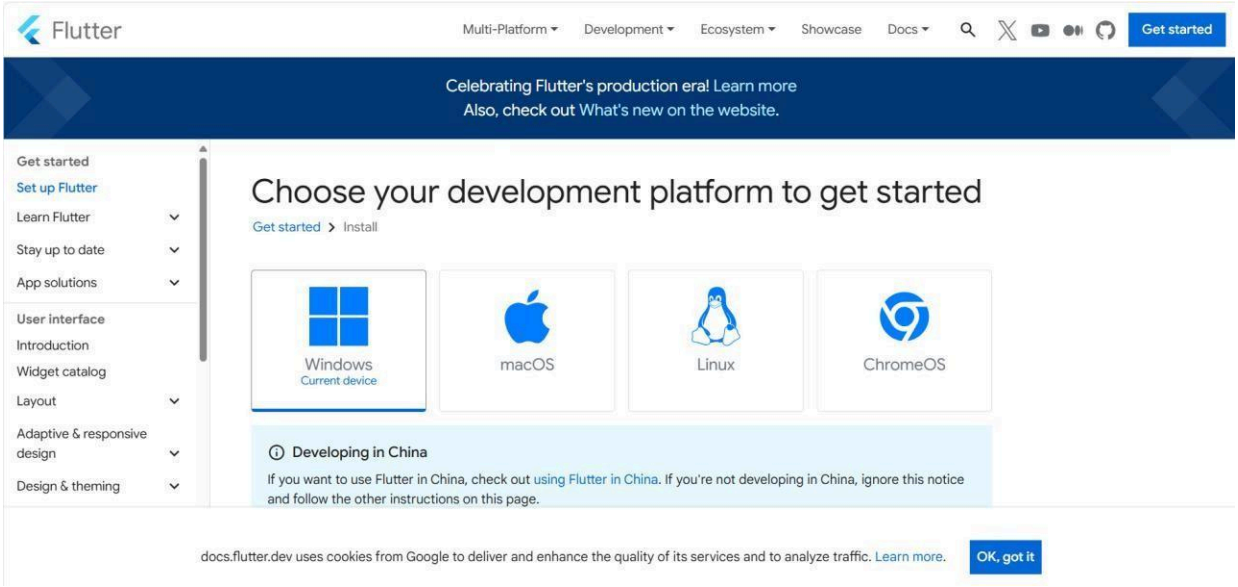
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	44
Name	Manav Punjabi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

EXPERIMENT NO: - 01

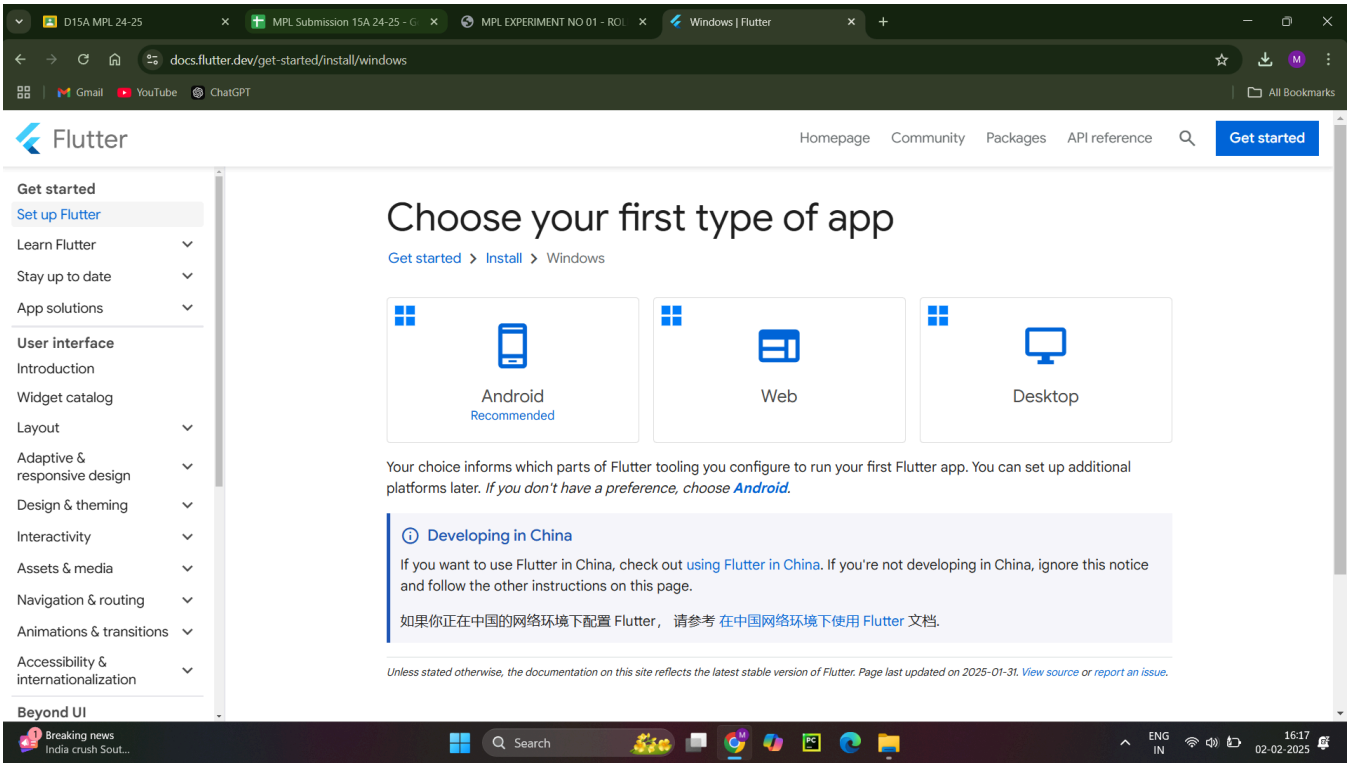
Name:- Manav Punjabi **Class:-** D15A **Roll:No: -** 44

AIM: - Installation and Configuration of Flutter Environment.

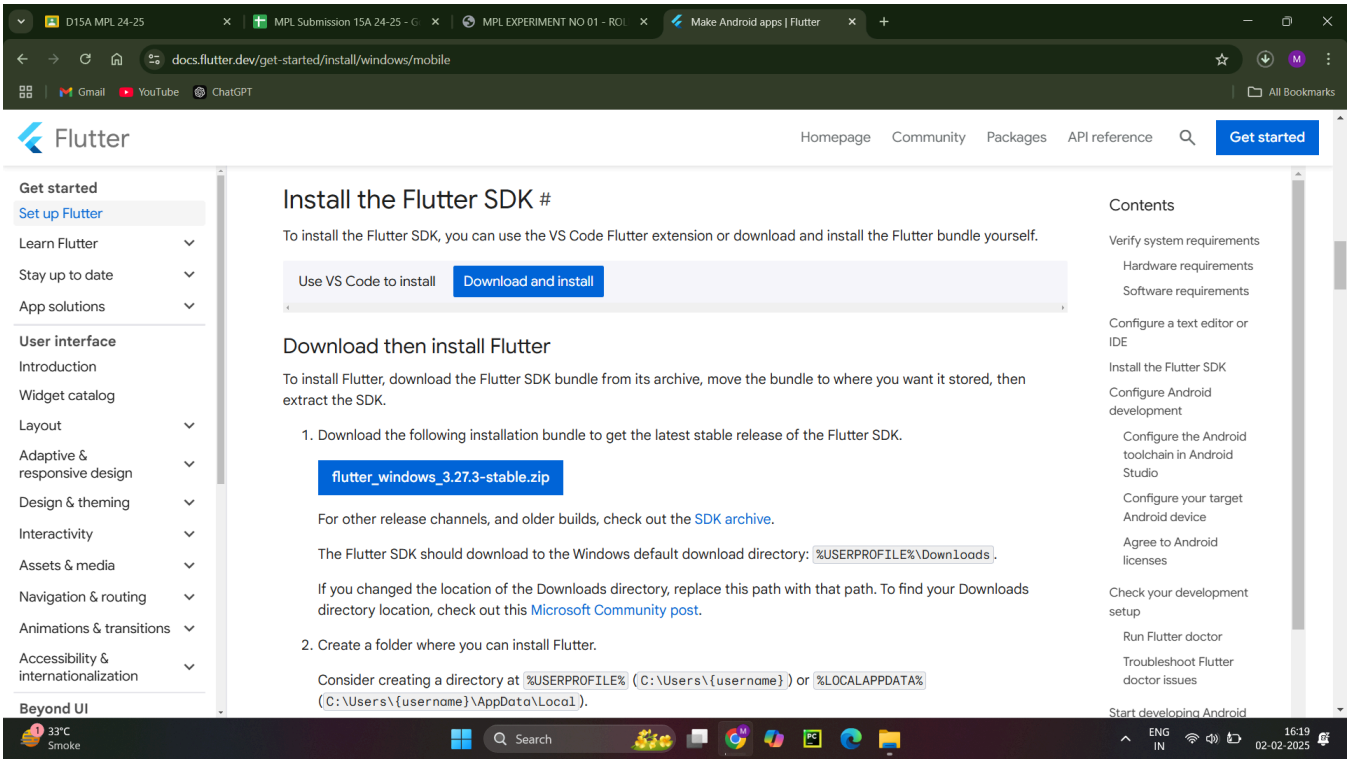
Step 1: Go to the official Flutter website: https://docs.flutter.dev/get-started/install



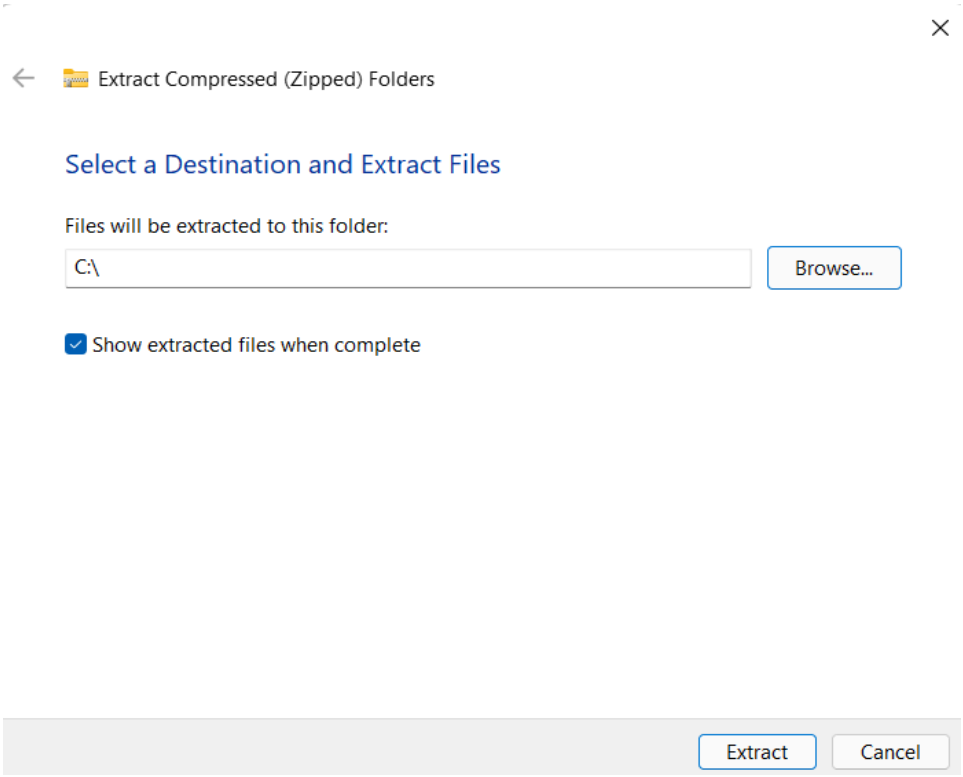
Step 2: To download the latest Flutter SDK, click on the Windows icon > Android



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



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

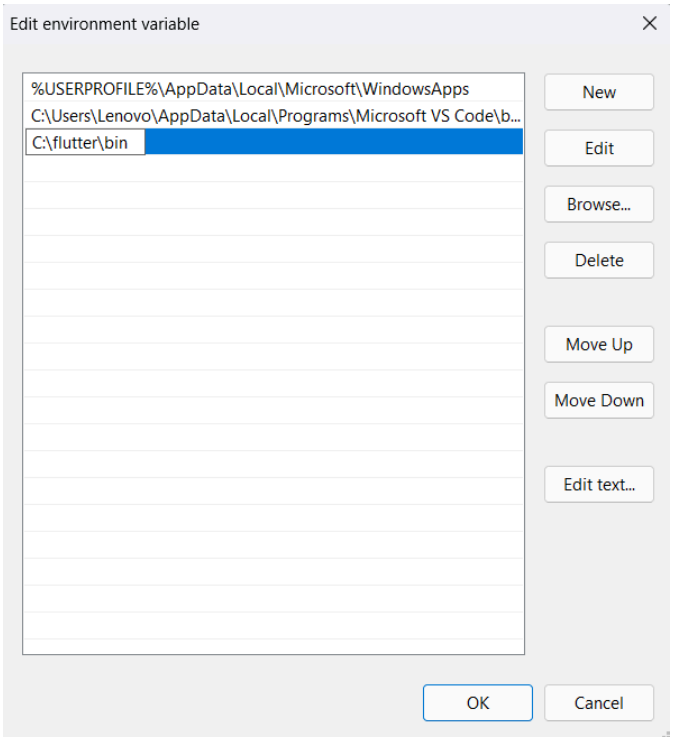


Step 5 :- Add Flutter to System PATH

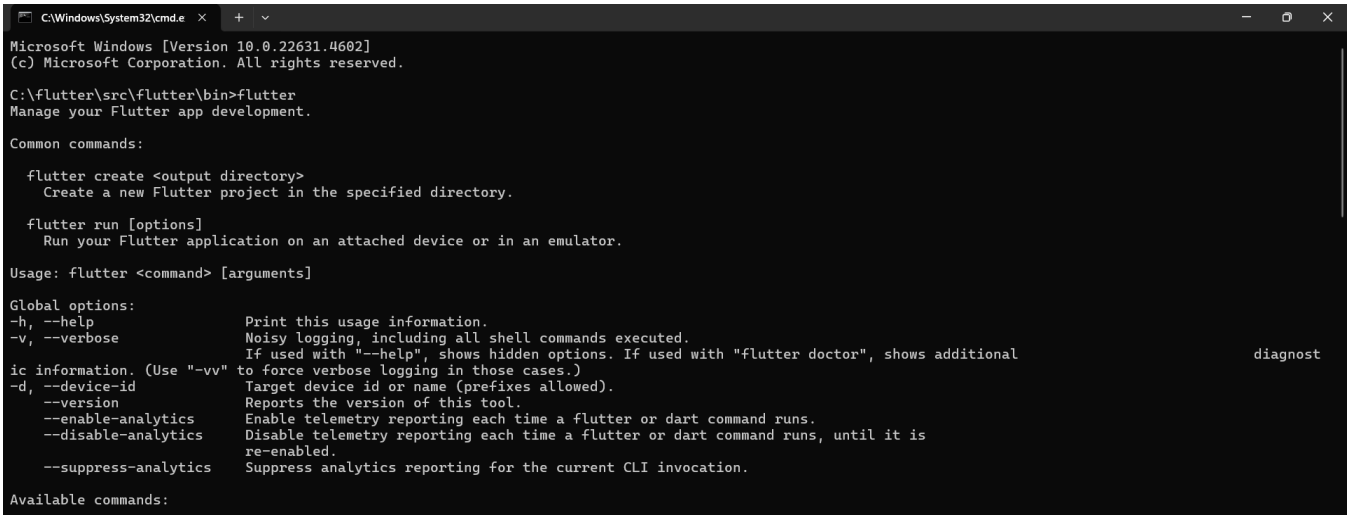
Right-click on the Start Menu > System > Advanced system settings > Environment Variables.

Under System Variables, find Path and click Edit.

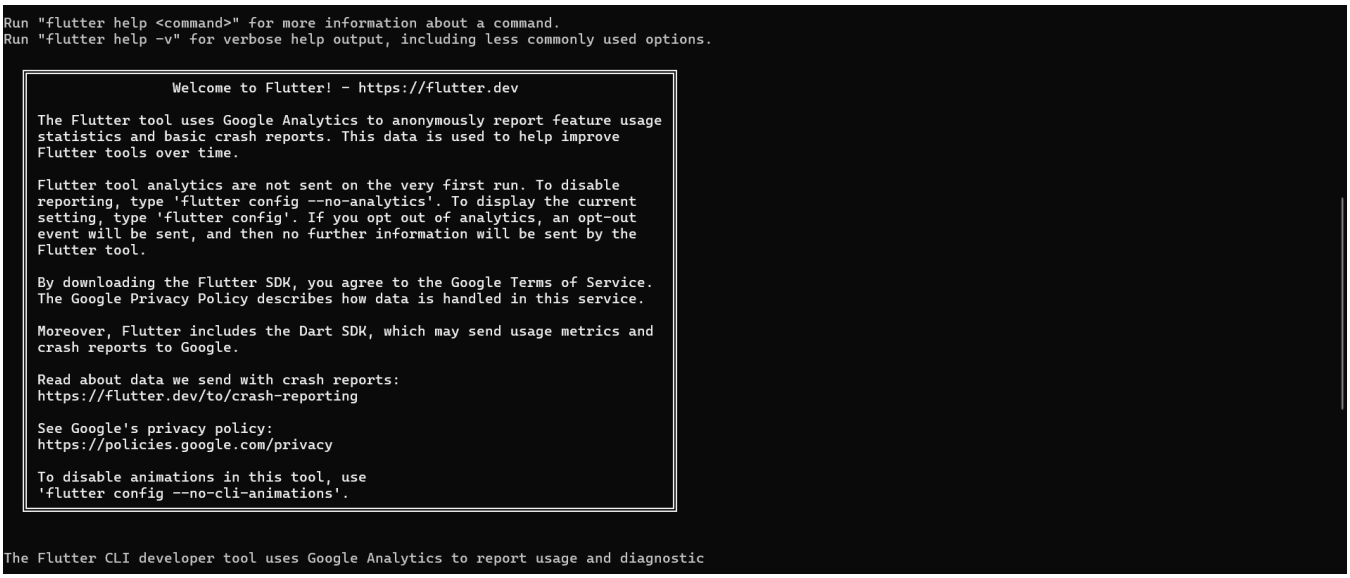
Add the full path to the flutter/bin directory (e.g., C:\flutter\bin).



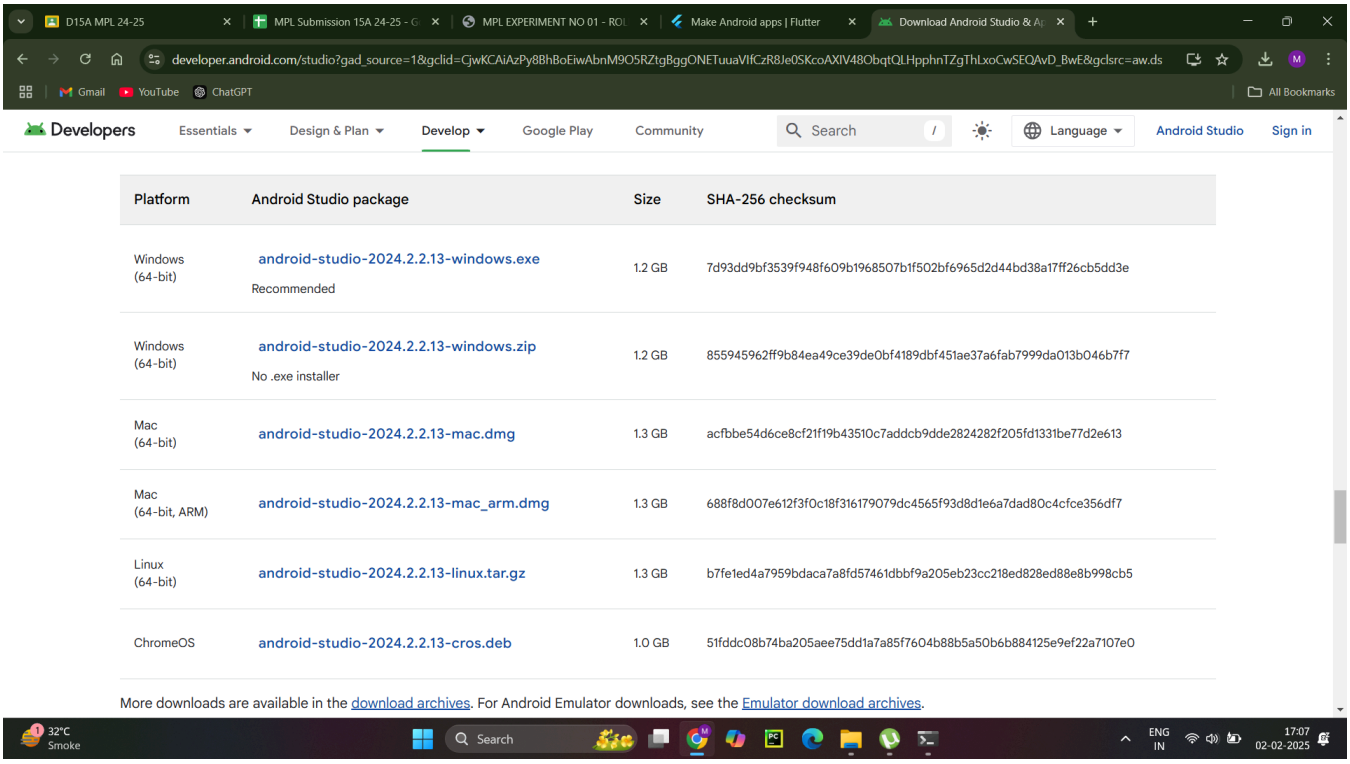
Step 6 : - Now, run the $ flutter command in command prompt.



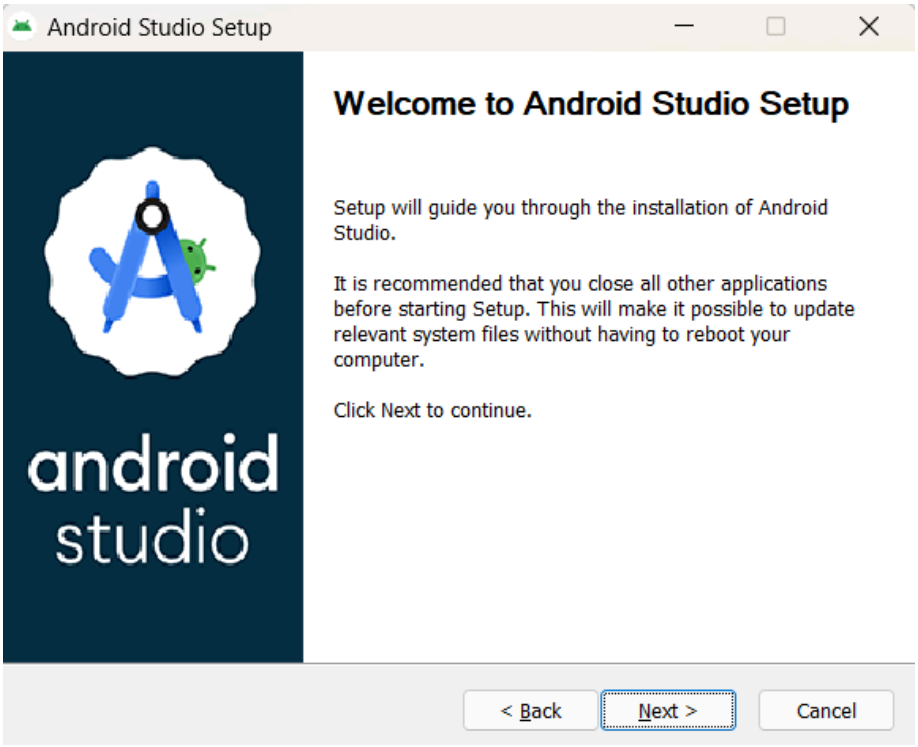
Step 7:- Run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation



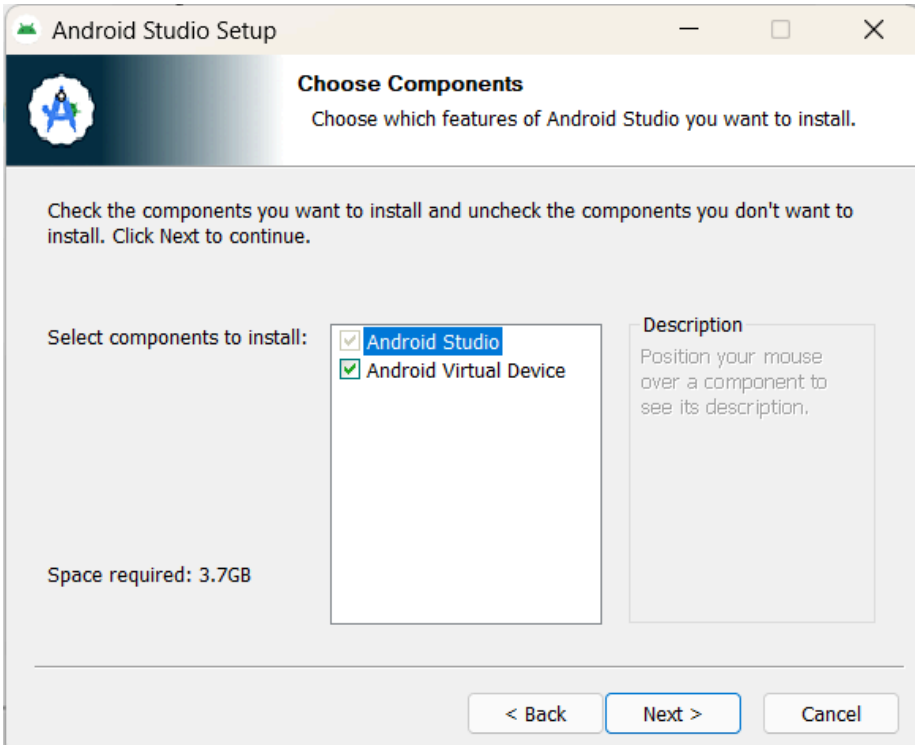
Step 8 : - Go to Android Studio and download the installer.



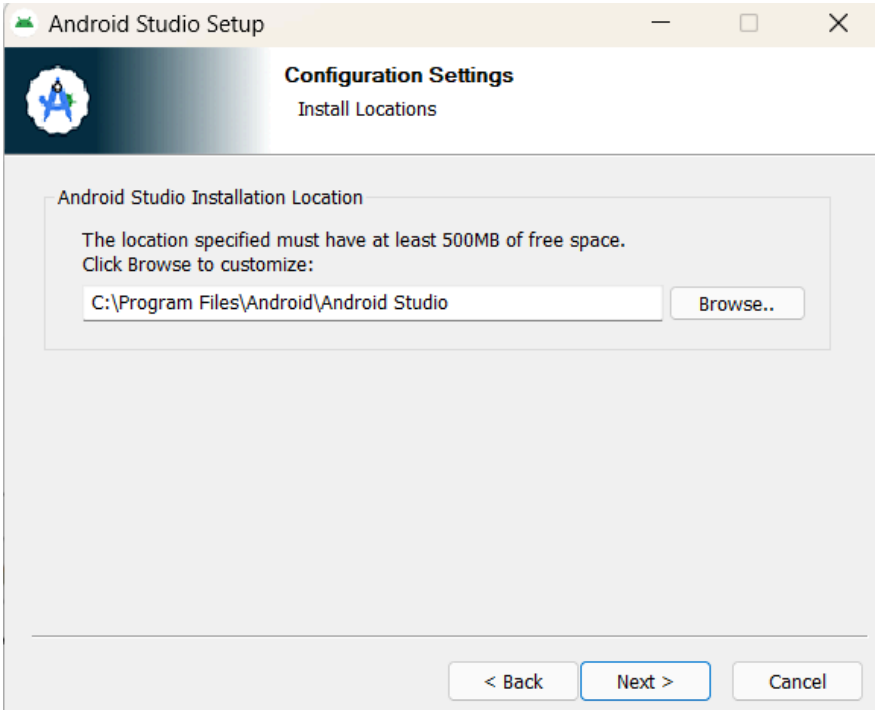
Step 8.1: - When the download is complete, open the .exe file and run it. You will get the following dialog box



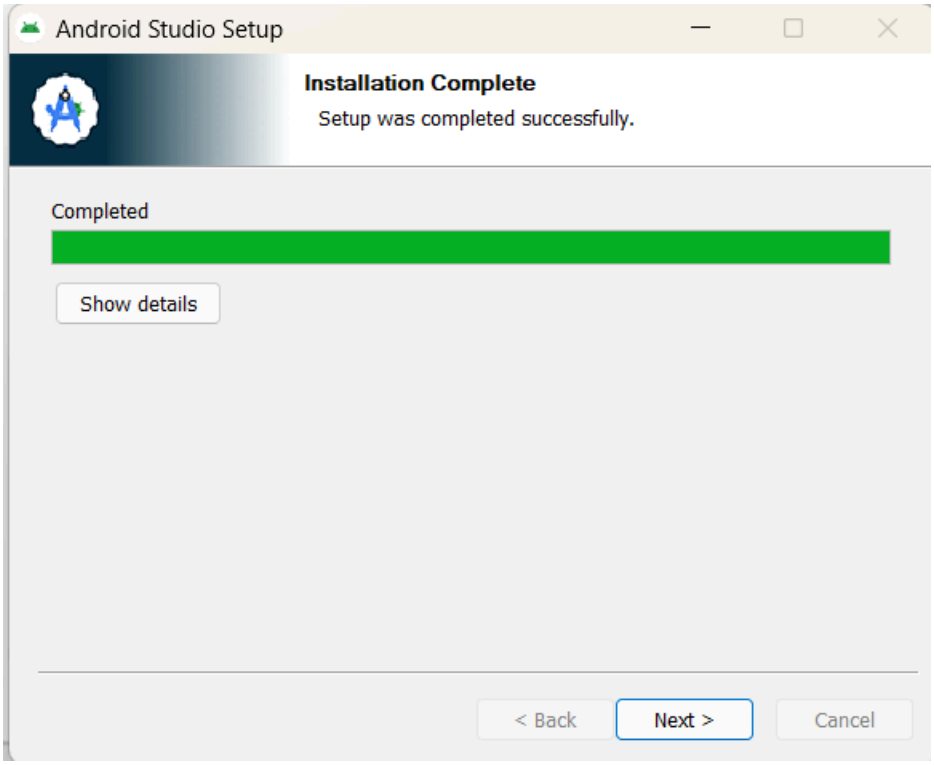
Step 8.2: - Select all the Checkboxes and Click on 'Next' Button.

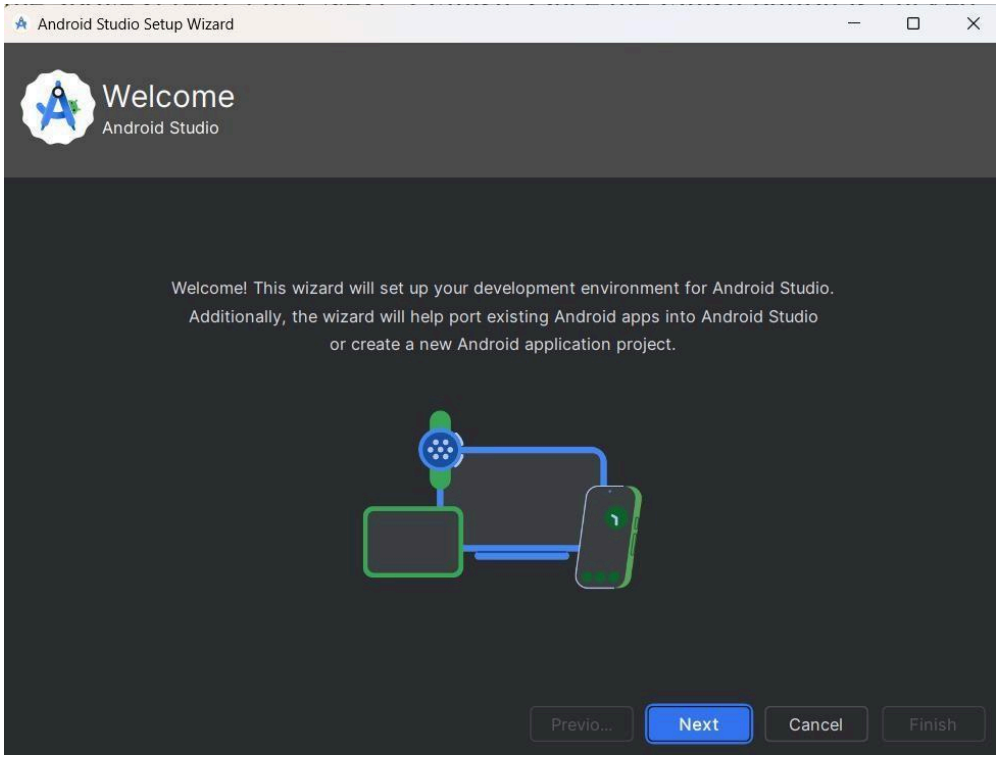


Step 8.3: - Change the destination as per your convenience and click on ‘Next’ Button.

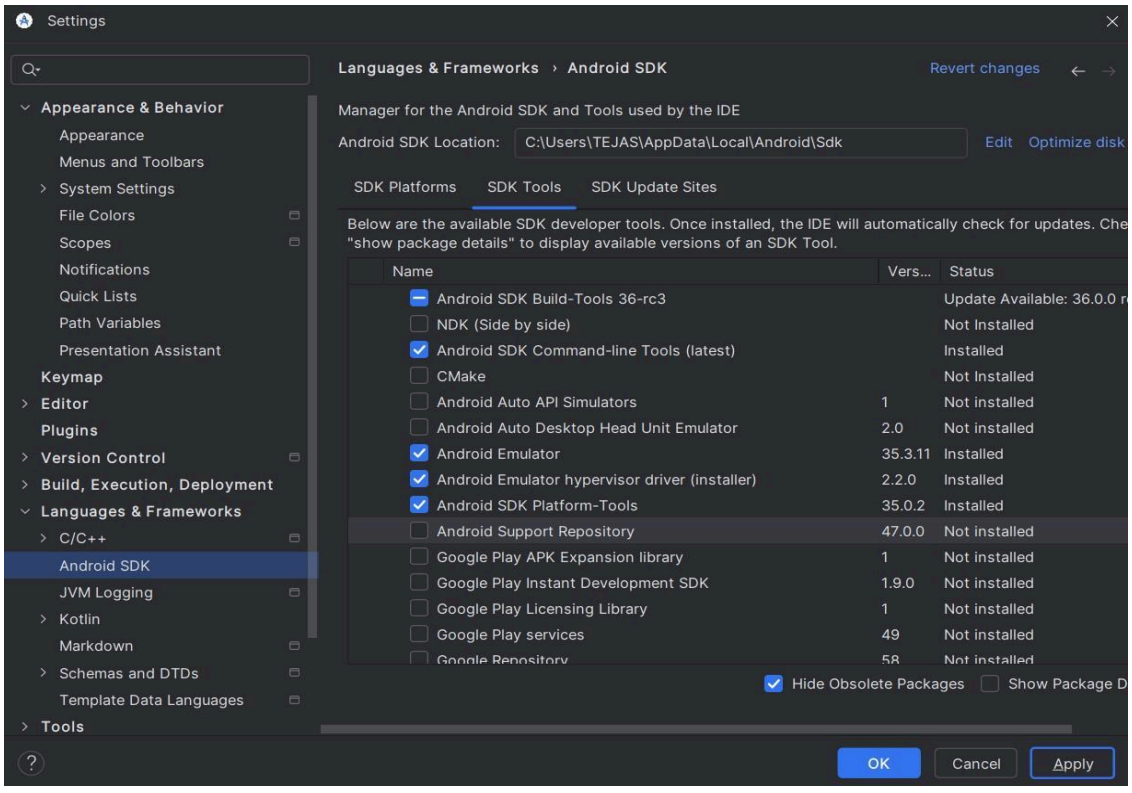


Step 8.4: - Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.

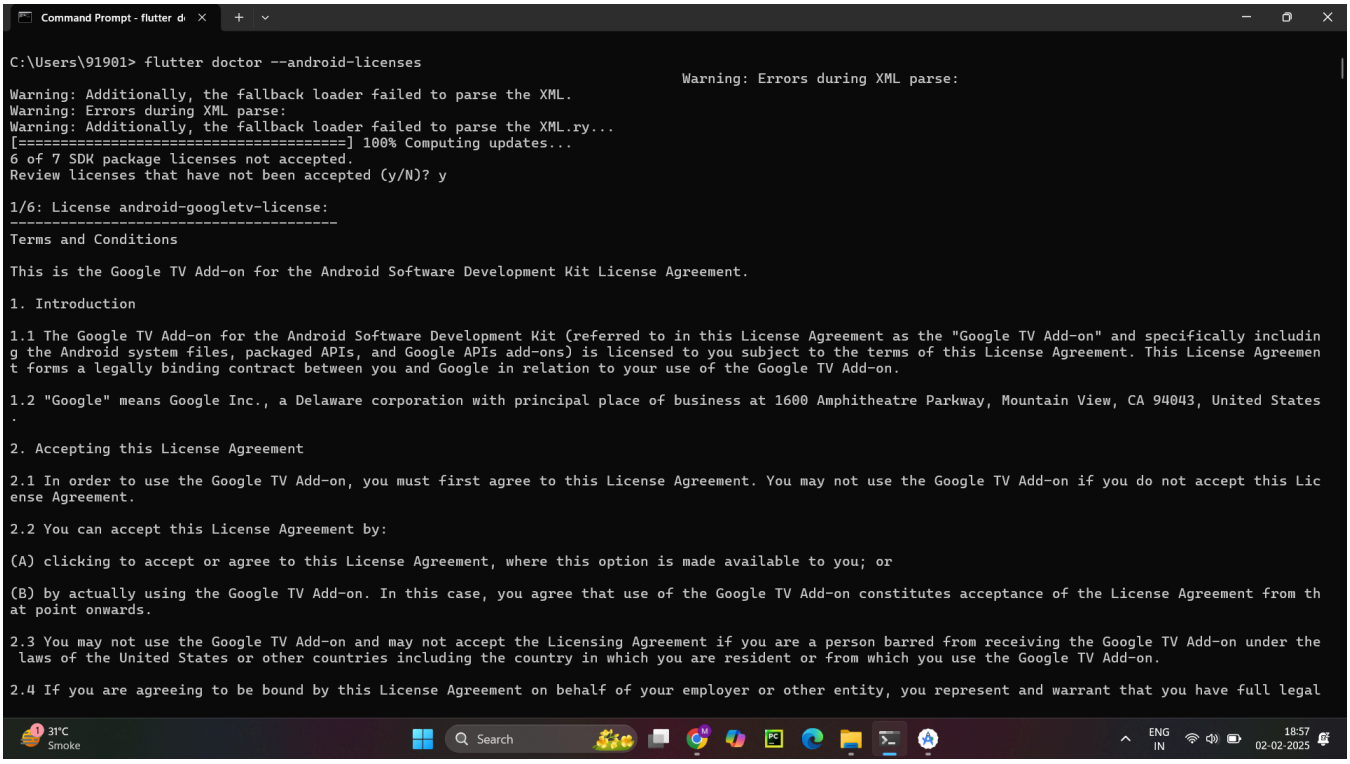


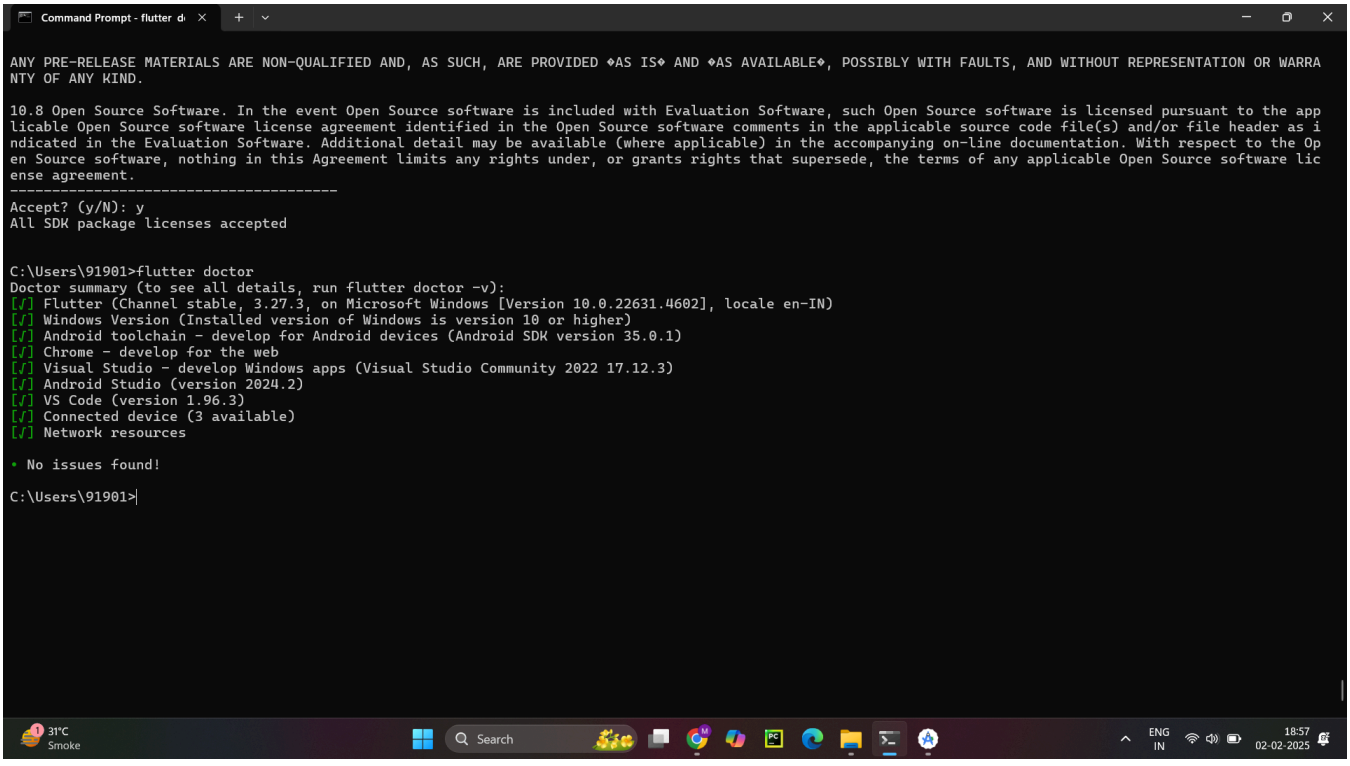


Step 8.5: - Go to Preferences > Appearance & Behavior > System Settings > Android SDK. Select the SDK Tools tab and check Android SDK Command-line Tools and Install it.

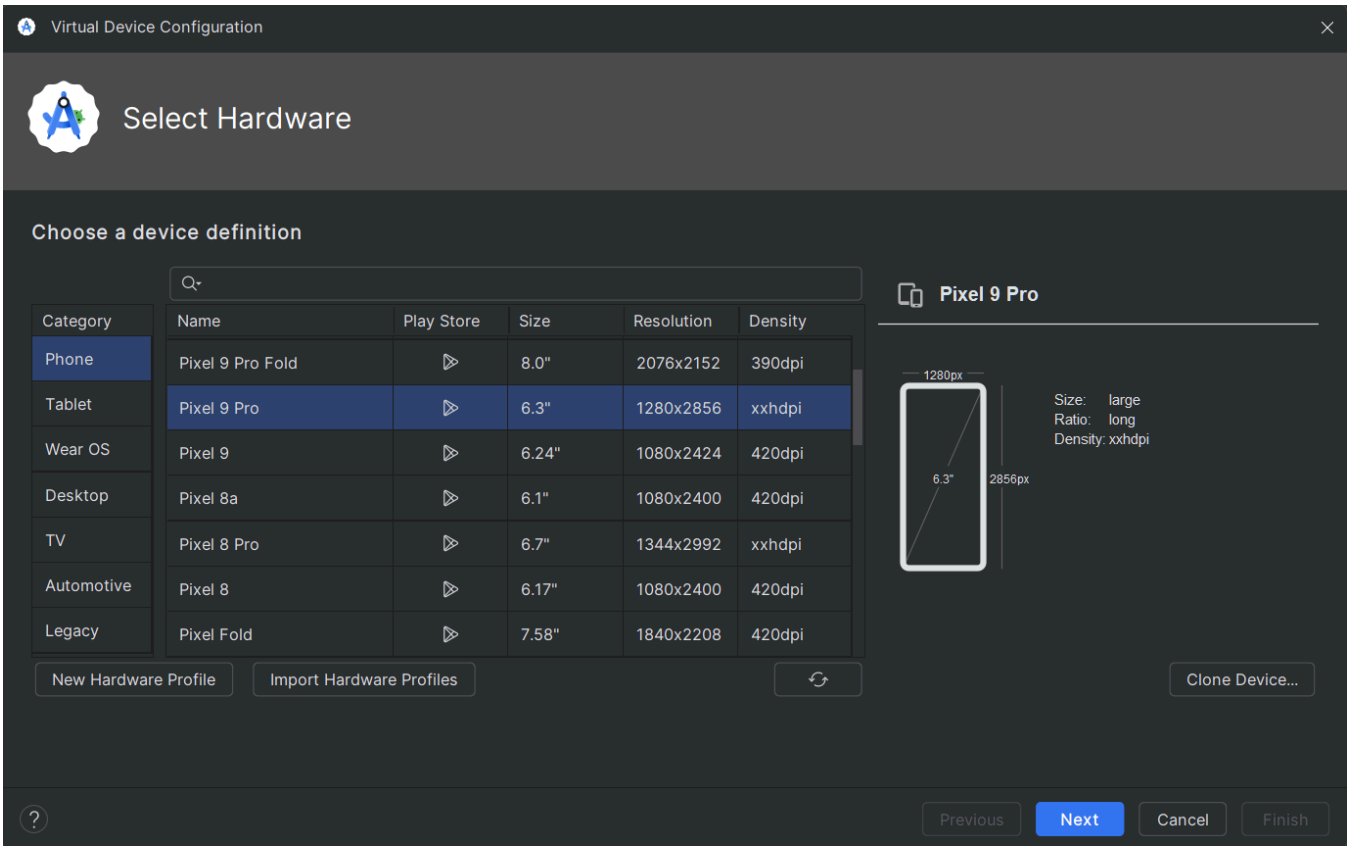


Step 9: - Open a terminal and run the following command

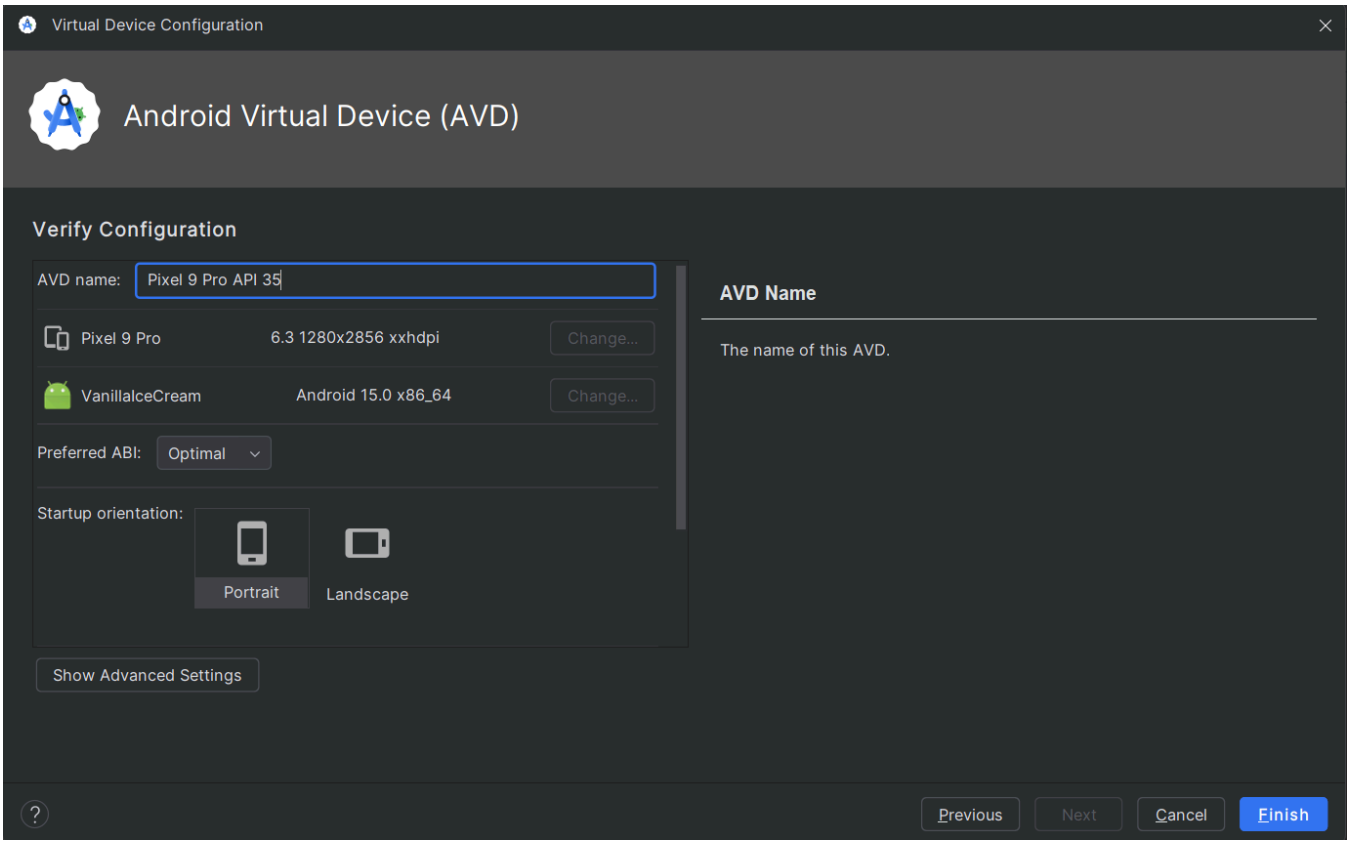




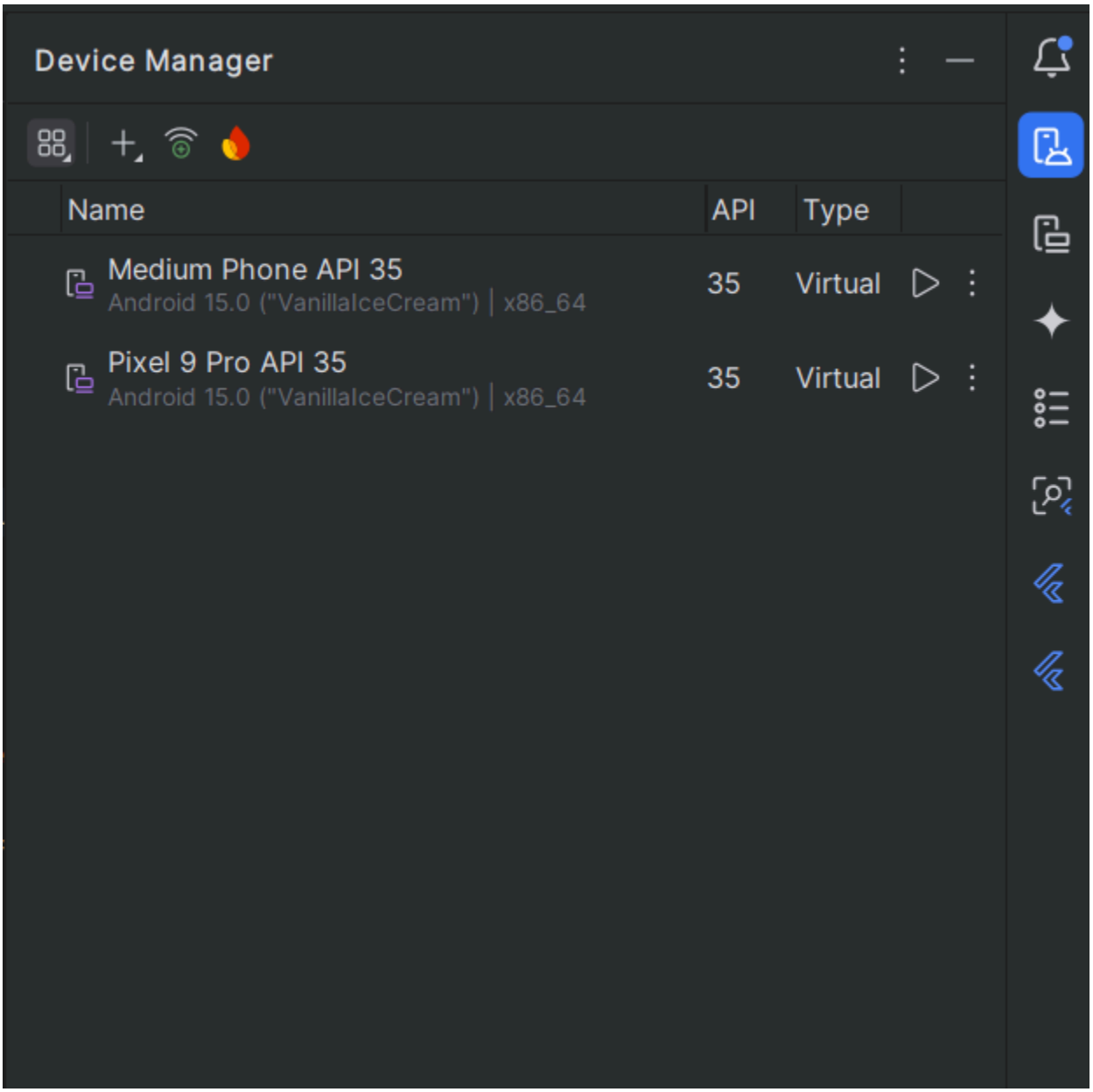
Step 10: - Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application

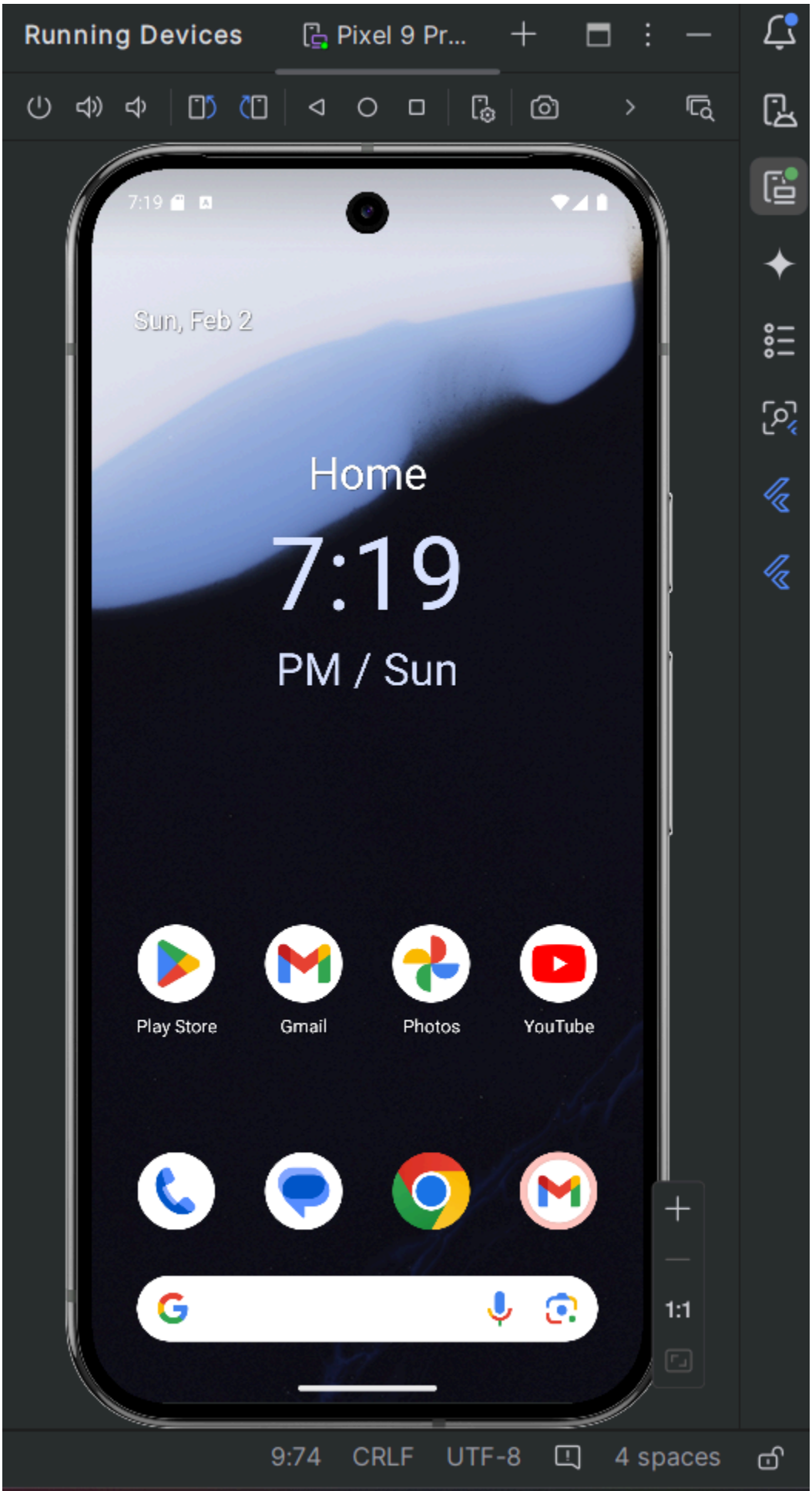


Step 10.1: - Open Android Studio and go to Tools > AVD Manager. Create a new virtual device.


Virtual Device Configuration
Android Virtual Device (AVD)
Verify Configuration
AVD name:
Pixel 9 Pro API 35
Pixel 9 Pro
6.3 1280x2856 xxhdpi
Change...
VanillaIceCream
Android 15.0 x86_64
Change...
Preferred ABI:
Optimal
Startup orientation:
Portrait
Landscape
Show Advanced Settings
AVD Name
The name of this AVD.
Previous
Next
Cancel
Finish

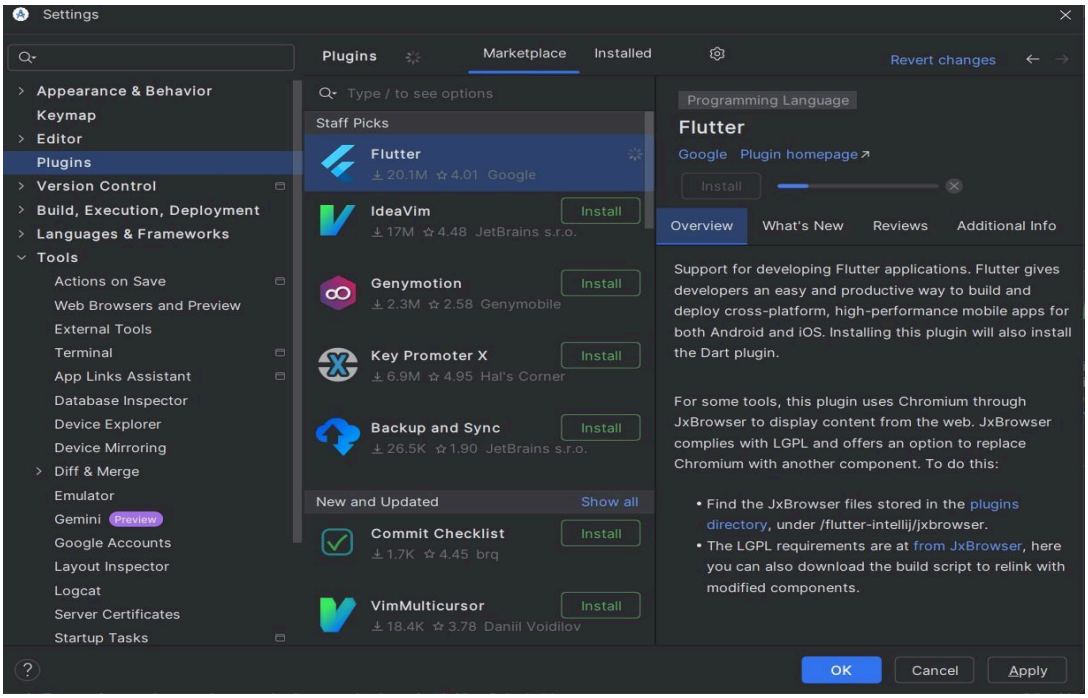
Step 10.2: - Click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen

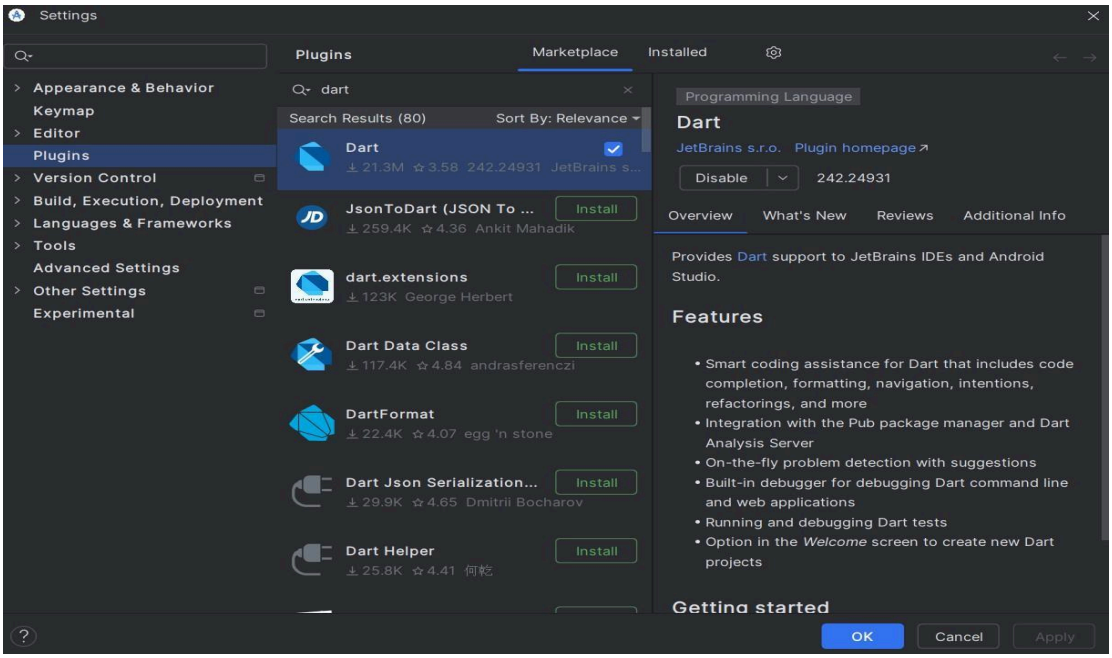



Running Devices
Pixel 9 Pr...
7:19
Sun, Feb 2
Home
7:19
PM / Sun
Play Store
Gmail
Photos
YouTube
1:1
9:74
CRLF
UTF-8
4 spaces

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

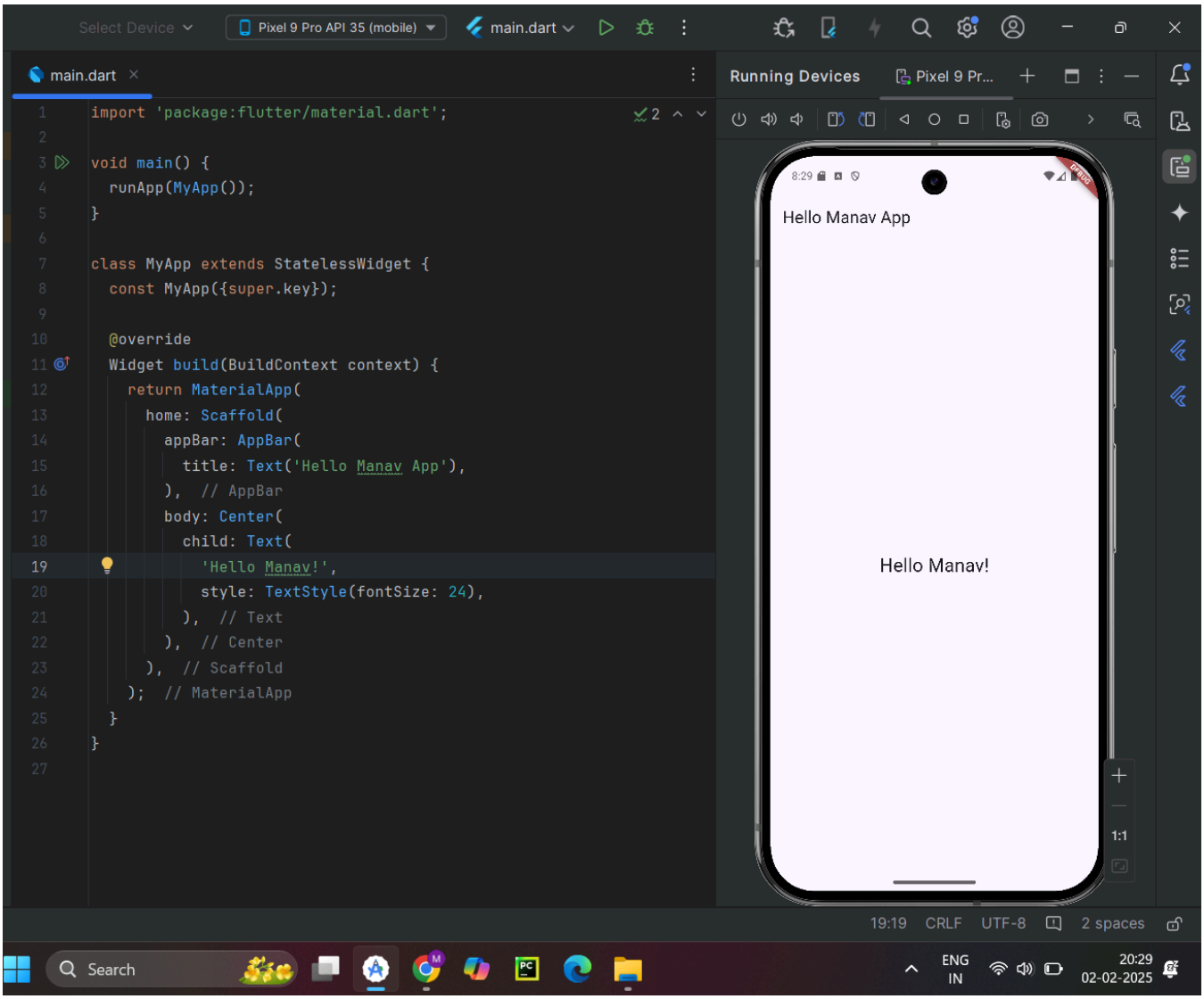
Step 11.1: - Open the Android Studio and then go to File->Settings->Plugins. Now, search the Flutter plugin. If found, select Flutter plugin and click install





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.	44
Name	Manav punjabi
Class	D15A/D15B
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:- Manav Punjabi **Class:-** D15A **Roll:No: -** 44

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.

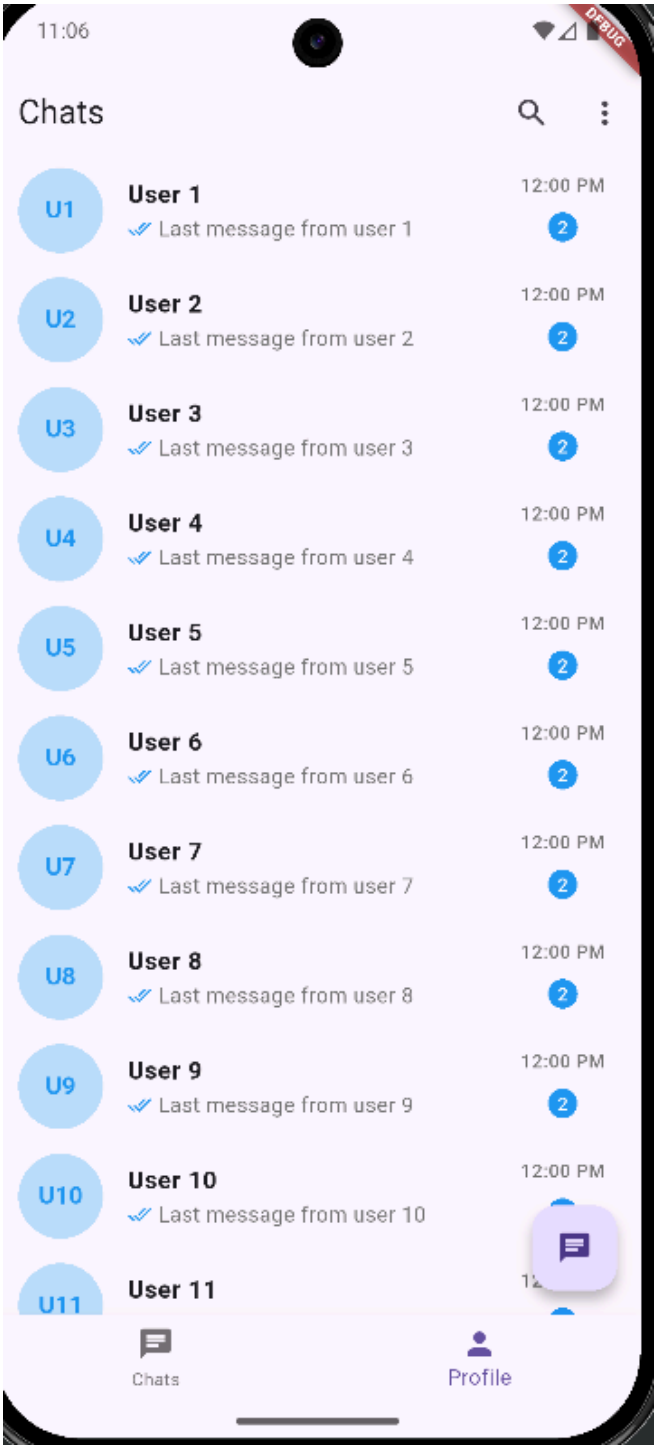
Card – A material design component that displays content inside a box with elevation.

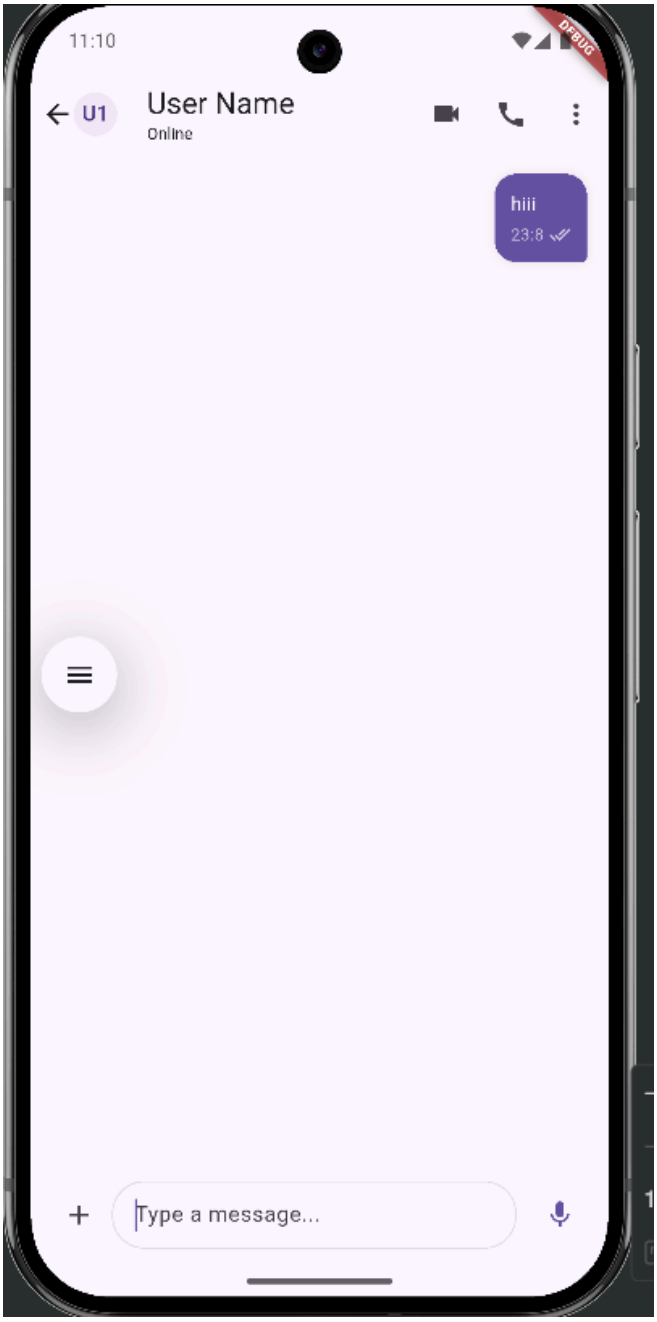
Code: - home_page.dart

```
import 'package:flutter/material.dart';
import 'chat_screen.dart';
import 'profile_screen.dart';
import 'search_screen.dart';

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

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

class _HomeScreenState extends
State<HomeScreen> {
  int _currentIndex = 0;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Chats'),
        actions: [
          IconButton(
            icon: const Icon(Icons.search),
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder:
(context) => const SearchScreen()),
              );
            },
          ),
          IconButton(
            icon: const Icon(Icons.more_vert),
            onPressed: () {
              // Show more options
            },
          ),
        ],
      ),
      body: ListView.builder(
        itemCount: 15, // Dummy data
        itemBuilder: (context, index) {
          return ListTile(
            leading: CircleAvatar(
              radius: 28,
              backgroundColor: Colors.blue[100],
              child: Text(
                'U${index + 1}',
                style: const TextStyle(
                  fontSize: 16,
                  fontWeight: FontWeight.bold,
                  color: Colors.blue,
                ),
              ),
            ),
            title: Text(
              'User ${index + 1}',
              style: const TextStyle(
                fontWeight: FontWeight.bold,
              ),
            ),
            subtitle: Row(
              children: [
                const Icon(
                  Icons.done_all,
                  size: 16,
                  color: Colors.blue,
                ),
                const SizedBox(width: 4),
                Text(
                  'Last message from user ${index
+ 1}',
                  style: TextStyle(
                    color: Colors.grey[600],
                  ),
                ),
              ],
            ),
            trailing: Column(
              mainAxisAlignment:
MainAxisAlignment.spaceAround,
              children: [
                Text(
                  '12:00 PM',
                  style: TextStyle(
                    fontSize: 12,
                    color: Colors.grey[600],
                  ),
                ),
                Container(
                  padding: const EdgeInsets.all(6),
                  decoration: const BoxDecoration(
                    color: Colors.blue,
                    shape: BoxShape.circle,
                  ),
                  child: const Text(
                    '2',
                    style: TextStyle(
                      color: Colors.white,
                      fontSize: 12,
```

```
                ),
              ),
            ),
          ],
        ),
        onTap: () {
          Navigator.push(
            context,
            MaterialPageRoute(builder:
(context) => const ChatScreen()),
          );
        },
      );
    },
  ),
  floatingActionButton:
FloatingActionButton(
    onPressed: () {
      // New chat
    },
    child: const Icon(Icons.chat),
  ),
  bottomNavigationBar:
BottomNavigationBar(
    currentIndex: _currentIndex,
    onTap: (index) {
      setState(() {
        _currentIndex = index;
      });
      if (index == 1) {
        Navigator.push(
          context,
          MaterialPageRoute(builder:
(context) => const ProfileScreen()),
        );
      }
    },
    items: const [
      BottomNavigationBarItem(
        icon: Icon(Icons.chat),
        label: 'Chats',
      ),
      BottomNavigationBarItem(
        icon: Icon(Icons.person),
        label: 'Profile',
      ),
    ],
  ),
 );
 }
}
```

OUTPUT: -


11:06
DEBUG
Chats
U1
User 1
Last message from user 1
12:00 PM
2
U2
User 2
Last message from user 2
12:00 PM
2
U3
User 3
Last message from user 3
12:00 PM
2
U4
User 4
Last message from user 4
12:00 PM
2
U5
User 5
Last message from user 5
12:00 PM
2
U6
User 6
Last message from user 6
12:00 PM
2
U7
User 7
Last message from user 7
12:00 PM
2
U8
User 8
Last message from user 8
12:00 PM
2
U9
User 9
Last message from user 9
12:00 PM
2
U10
User 10
Last message from user 10
12:00 PM
U11
User 11
Chats
Profile


11:10
DEBUG
U1
User Name
Online
hiii
23:8
Type a message...

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	44
Name	Manav punjabi
Class	D15A/D15B
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:- Manav Punjabi **Class:-** D15A **Roll:No: -** 44

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: -

profile_customization_screen.dart:-

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

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

  @override
  State<ProfileCustomizationScreen> createState() =>
_ProfileCustomizationScreenState();
}

class _ProfileCustomizationScreenState extends
State<ProfileCustomizationScreen> {
  File? _imageFile;
  final _nameController = TextEditingController();
  final _bioController = TextEditingController();
  String _selectedTheme = 'Light';
  String _selectedStatus = 'Available';

  @override
  Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: const Text('Edit Profile'),
       actions: [
         IconButton(
           icon: const Icon(Icons.check),
           onPressed: _saveProfile,
         ),
       ],
     ),
     body: SingleChildScrollView(
       padding: const EdgeInsets.all(16),
       child: Column(
         children: [
           // Profile Picture Section
           Center(
             child: Stack(
               children: [
                 CircleAvatar(
                   radius: 60,
                   backgroundColor:
Theme.of(context).colorScheme.primary.withAlpha(30
),
                   backgroundImage: _imageFile != null
                       ? FileImage(_imageFile!)
                       : null,
                   child: _imageFile == null
                       ? const Icon(Icons.person, size: 60)
                       : null,
                 ),
                 Positioned(
                   bottom: 0,
                   right: 0,
                   child: CircleAvatar(
                     backgroundColor:
Theme.of(context).colorScheme.primary,
                     radius: 20,
                     child: IconButton(
                       icon: const Icon(Icons.camera_alt),
                       color: Colors.white,
                       onPressed: _showImagePickerOptions,
                     ),
                   ),
                 ),
               ],
             ),
           ),
           const SizedBox(height: 24),

           // Name Field
           TextField(
             controller: _nameController,
             decoration: InputDecoration(
               labelText: 'Name',
               prefixIcon: const Icon(Icons.person_outline),
               border: OutlineInputBorder(
                 borderRadius: BorderRadius.circular(12),
               ),
             ),
           ),
           const SizedBox(height: 16),

           // Bio Field
           TextField(
             controller: _bioController,
             maxLines: 3,
             decoration: InputDecoration(
               labelText: 'Bio',
               prefixIcon: const Icon(Icons.edit_note),
               border: OutlineInputBorder(
                 borderRadius: BorderRadius.circular(12),
               ),
             ),
           ),
           const SizedBox(height: 24),

           // Theme Selection
           ListTile(
             leading: const Icon(Icons.palette_outlined),
             title: const Text('Theme'),
             subtitle: Text(_selectedTheme),
             trailing: const Icon(Icons.arrow_forward_ios,
size: 16),
             onTap: _showThemeOptions,
           ),
```

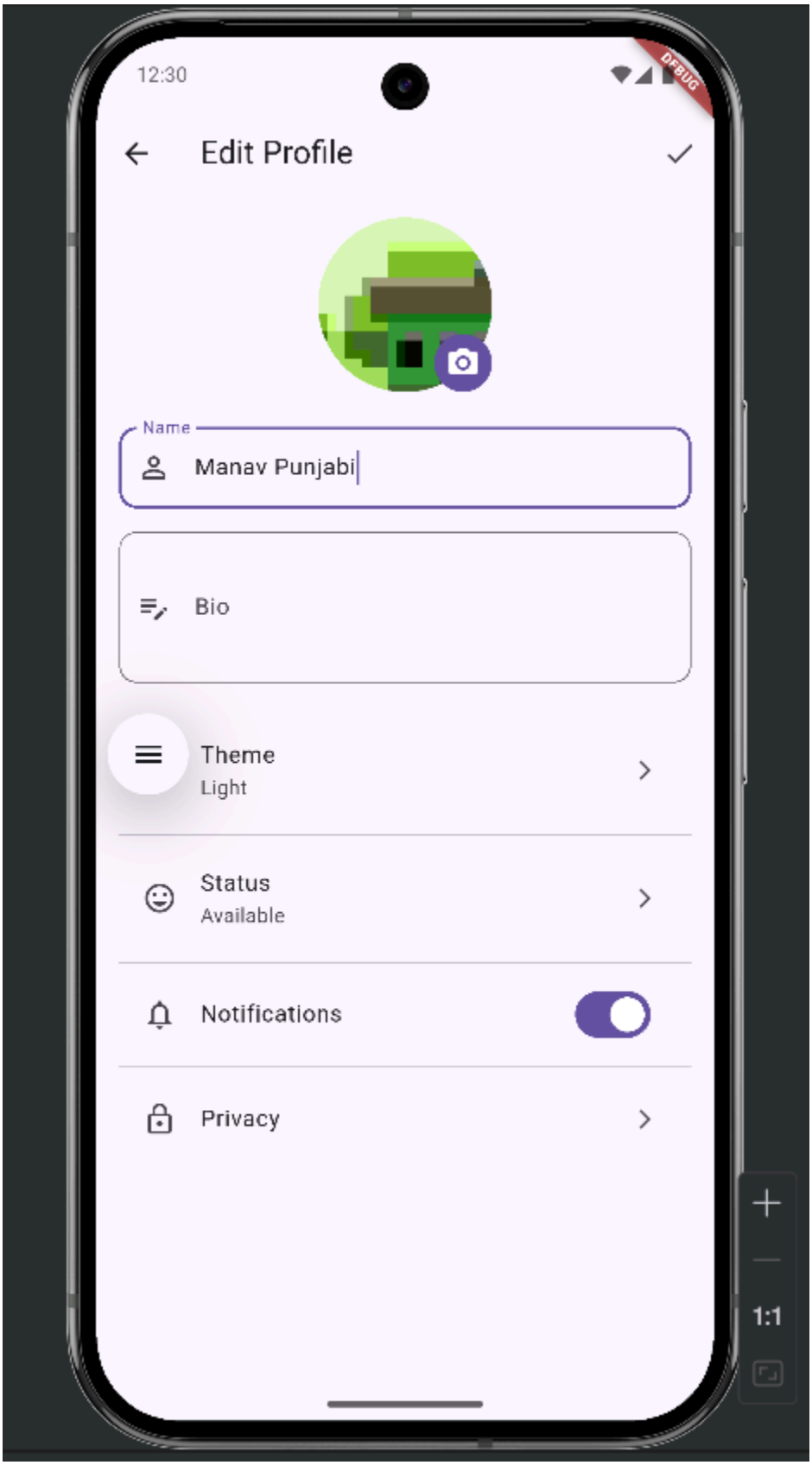
```
          const Divider(),

          // Status Selection
          ListTile(
            leading: const Icon(Icons.mood),
            title: const Text('Status'),
            subtitle: Text(_selectedStatus),
            trailing: const Icon(Icons.arrow_forward_ios,
size: 16),
            onTap: _showStatusOptions,
          ),
          const Divider(),

          // Additional Options
          ListTile(
            leading: const
Icon(Icons.notifications_outlined),
            title: const Text('Notifications'),
            trailing: Switch(
              value: true,
              onChanged: (value) {},
            ),
          ),
          const Divider(),

          ListTile(
            leading: const Icon(Icons.lock_outline),
            title: const Text('Privacy'),
            trailing: const Icon(Icons.arrow_forward_ios,
size: 16),
            onTap: () {},
          ),
        ],
      ),
    ),
  );
 }

 void _showImagePickerOptions() {
  showModalBottomSheet(
    context: context,
    builder: (context) => Container(
      padding: const EdgeInsets.all(20),
      height: 180,
      child: Column(
        children: [
          ListTile(
            leading: const Icon(Icons.camera),
            title: const Text('Take a photo'),
            onTap: () {
              Navigator.pop(context);
              _pickImage(ImageSource.camera);
            },
          ),
          ListTile(
            leading: const Icon(Icons.photo_library),
            title: const Text('Choose from gallery'),
            onTap: () {
              Navigator.pop(context);
              _pickImage(ImageSource.gallery);
            },
          ),
        ],
      ),
    ),
  );
 }

 Future<void> _pickImage(ImageSource source) async
{
  final picker = ImagePicker();
  final pickedFile = await picker.pickImage(source:
source);

  if (pickedFile != null) {
   setState(() {
    _imageFile = File(pickedFile.path);
   });
  }
 }

 void _showThemeOptions() {
  showModalBottomSheet(
   context: context,
   builder: (context) => Container(
    padding: const EdgeInsets.all(20),
    child: Column(
     mainAxisSize: MainAxisSize.min,
     children: [
      ListTile(
       title: const Text('Light'),
       leading: Radio(
        value: 'Light',
        groupValue: _selectedTheme,
        onChanged: (value) {
         setState(() {
          _selectedTheme = value.toString();
          Navigator.pop(context);
         });
        },
       ),
      ),
      ListTile(
       title: const Text('Dark'),
       leading: Radio(
        value: 'Dark',
        groupValue: _selectedTheme,
        onChanged: (value) {
         setState(() {
          _selectedTheme = value.toString();
          Navigator.pop(context);
         });
```

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

void _showStatusOptions() {
 showModalBottomSheet(
  context: context,
  builder: (context) => Container(
   padding: const EdgeInsets.all(20),
   child: Column(
    mainAxisSize: MainAxisSize.min,
    children: [
     ListTile(
      title: const Text('Available'),
      leading: Radio(
       value: 'Available',
       groupValue: _selectedStatus,
       onChanged: (value) {
        setState(() {
         _selectedStatus = value.toString();
         Navigator.pop(context);
        });
       },
      ),
     ),
     ListTile(
      title: const Text('Busy'),
      leading: Radio(
       value: 'Busy',
       groupValue: _selectedStatus,
       onChanged: (value) {
        setState(() {
         _selectedStatus = value.toString();
         Navigator.pop(context);
        });
       },
      ),
     ),
     ListTile(
      title: const Text('Away'),
      leading: Radio(
       value: 'Away',
       groupValue: _selectedStatus,
       onChanged: (value) {
        setState(() {
         _selectedStatus = value.toString();
         Navigator.pop(context);
        });
       },
      ),
     ),
    ],
   ),
  ),
 );
}

void _saveProfile() {
 // Save profile data
 Navigator.pop(context);
}

@override
void dispose() {
 _nameController.dispose();
 _bioController.dispose();
 super.dispose();
}
}
```

pubspec.yaml

```
- name: chatapp
- description: A new Flutter project.
-
- # Specify SDK version
- environment:
-   sdk: '>=3.0.0 <4.0.0'
-   flutter: ">=3.0.0"
-
- dependencies:
-   flutter:
-     sdk: flutter
-   image_picker: ^1.0.7
-   cupertino_icons: ^1.0.2
-
- dev_dependencies:
-   flutter_test:
-     sdk: flutter
-   flutter_lints: ^2.0.0
-
- flutter:
-   uses-material-design: true
```

OUTPUT: -


12:30
DEBUG
Edit Profile
Name
Manav Punjabi
Bio
Theme
Light
Status
Available
Notifications
Privacy
1:1

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	44
Name	Manav Punjabi
Class	D15A/D15B
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:- Manav Punjabi **Class:-** D15A **Roll:No: -** 44

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_page.dart

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

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

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

class _LoginScreenState extends
State<LoginScreen> {
  bool isLogin = true;
  final _emailController =
TextEditingController();
  final _passwordController =
TextEditingController();

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: SingleChildScrollView(
          child: Padding(
            padding: const EdgeInsets.all(16.0),
            child: Column(
              mainAxisAlignment:
MainAxisAlignment.center,
              children: [
                const SizedBox(height: 50),
                // App Logo
                Container(
                  height: 100,
                  width: 100,
                  decoration: BoxDecoration(
                    color: Colors.blue,
                    borderRadius:
BorderRadius.circular(20),
                  ),
                  child: const Icon(
                    Icons.chat,
                    size: 60,
                    color: Colors.white,
                  ),
                ),
                const SizedBox(height: 30),

                // Welcome Text
                Text(
                  isLogin ? 'Welcome Back' : 'Create
Account',
                  style: const TextStyle(
                    fontSize: 28,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                const SizedBox(height: 10),
                Text(
                  isLogin ? 'Login to continue' : 'Sign
up to get started',
                  style: TextStyle(
                    fontSize: 16,
                    color: Colors.grey[600],
                  ),
                ),
                const SizedBox(height: 30),

                // Email Field
                TextField(
                  controller: _emailController,
                  decoration: InputDecoration(
                    labelText: 'Email',
                    prefixIcon: const Icon(Icons.email),
                    border: OutlineInputBorder(
                      borderRadius:
BorderRadius.circular(12),
                    ),
                    enabledBorder: OutlineInputBorder(
                      borderRadius:
BorderRadius.circular(12),
                      borderSide: const
BorderSide(color: Colors.grey),
                    ),
                    focusedBorder: OutlineInputBorder(
                      borderRadius:
BorderRadius.circular(12),
                      borderSide: const
BorderSide(color: Colors.blue),
                    ),
                  ),
                  keyboardType:
TextInputType.emailAddress,
```

```
            ),
            const SizedBox(height: 16),

            // Password Field
            TextField(
              controller: _passwordController,
              decoration: InputDecoration(
                labelText: 'Password',
                prefixIcon: const Icon(Icons.lock),
                border: OutlineInputBorder(
                  borderRadius:
BorderRadius.circular(12),
                ),
                enabledBorder: OutlineInputBorder(
                  borderRadius:
BorderRadius.circular(12),
                  borderSide: const
BorderSide(color: Colors.grey),
                ),
                focusedBorder: OutlineInputBorder(
                  borderRadius:
BorderRadius.circular(12),
                  borderSide: const
BorderSide(color: Colors.blue),
                ),
              ),
              obscureText: true,
            ),
            const SizedBox(height: 20),

            // Forgot Password Button
            if (isLogin)
              Align(
                alignment: Alignment.centerRight,
                child: TextButton(
                  onPressed: () {
                    // Forgot password logic
                  },
                  child: const Text('Forgot
Password?'),
                ),
              ),
            const SizedBox(height: 20),

            // Login/Signup Button
            SizedBox(
              width: double.infinity,
              height: 50,
              child: ElevatedButton(
                onPressed: () {
                  // Login/Signup logic ke baad
                  Navigator.pushReplacement(
                    context,
                    MaterialPageRoute(builder:
(context) => const HomeScreen()),
                  );
                },
                style: ElevatedButton.styleFrom(
                  shape: RoundedRectangleBorder(
                    borderRadius:
BorderRadius.circular(12),
                  ),
                ),
                child: Text(
                  isLogin ? 'Login' : 'Sign Up',
                  style: const TextStyle(fontSize:
16),
                ),
              ),
            ),
            const SizedBox(height: 20),

            // Toggle Login/Signup
            Row(
              mainAxisAlignment:
MainAxisAlignment.center,
              children: [
                Text(
                  isLogin ? 'Don\'t have an account?'
: 'Already have an account?',
                  style: TextStyle(color:
Colors.grey[600]),
                ),
                TextButton(
                  onPressed: () {
                    setState(() {
                      isLogin = !isLogin;
                    });
                  },
                  child: Text(
                    isLogin ? 'Sign Up' : 'Login',
                    style: const TextStyle(fontWeight:
FontWeight.bold),
                  ),
                ),
              ],
            ),
          ],
        ),
      ),
    ),
  );
}
```

```
}
```

signup_screen.dart

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

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

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

class _SignupScreenState extends State<SignupScreen> {
  final _formKey = GlobalKey<FormState>();
  final _nameController = TextEditingController();
  final _emailController = TextEditingController();
  final _phoneController = TextEditingController();
  final _passwordController = TextEditingController();
  String? _selectedGender;
  DateTime? _selectedDate;
  String? _profileImage;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Create Account'),
      ),
      body: SingleChildScrollView(
        padding: const EdgeInsets.all(16),
        child: Form(
          key: _formKey,
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Center(
                child: Stack(
                  children: [
                    CircleAvatar(
                      radius: 50,
                      backgroundColor: Theme.of(context).colorScheme.primary.withAlpha(30),
                      backgroundImage: _profileImage != null
                          ? NetworkImage(_profileImage!)
                          : null,
                      child: _profileImage == null
                          ? const Icon(Icons.person, size: 50)
                          : null,
                    ),
                    Positioned(
                      bottom: 0,
                      right: 0,
                      child: CircleAvatar(
```

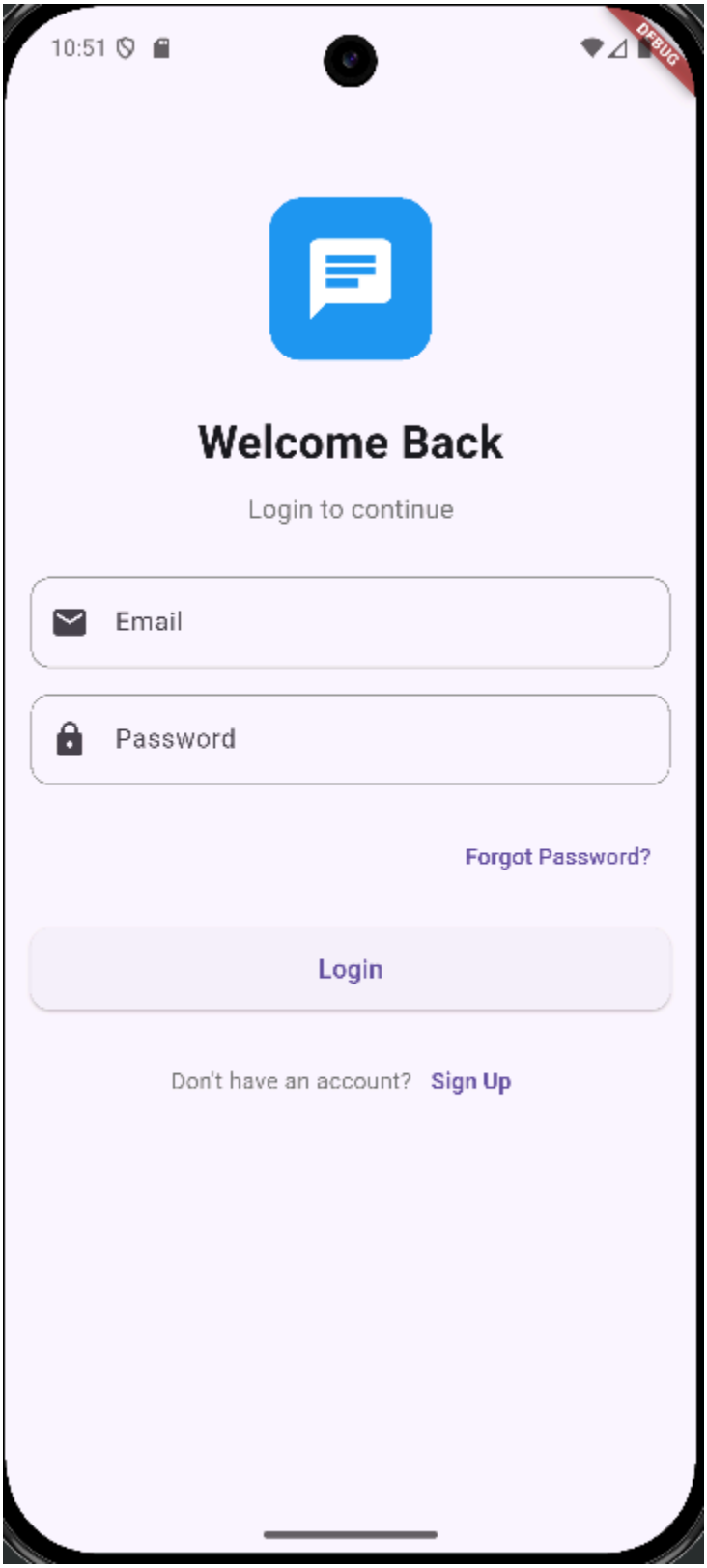
```
          backgroundColor: Theme.of(context).colorScheme.primary,
          radius: 18,
          child: IconButton(
            icon: const Icon(Icons.camera_alt, size: 18),
            color: Colors.white,
            onPressed: _pickImage,
          ),
        ),
      ),
    ],
  ),
),
const SizedBox(height: 24),
TextFormField(
  controller: _nameController,
  decoration: const InputDecoration(
    labelText: 'Full Name',
    prefixIcon: Icon(Icons.person_outline),
  ),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your name';
    }
    return null;
  },
),
const SizedBox(height: 16),
TextFormField(
  controller: _emailController,
  decoration: const InputDecoration(
    labelText: 'Email',
    prefixIcon: Icon(Icons.email_outlined),
  ),
  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),
TextFormField(
  controller: _phoneController,
  decoration: const InputDecoration(
    labelText: 'Phone Number',
    prefixIcon: Icon(Icons.phone_outlined),
  ),
  keyboardType: TextInputType.phone,
```

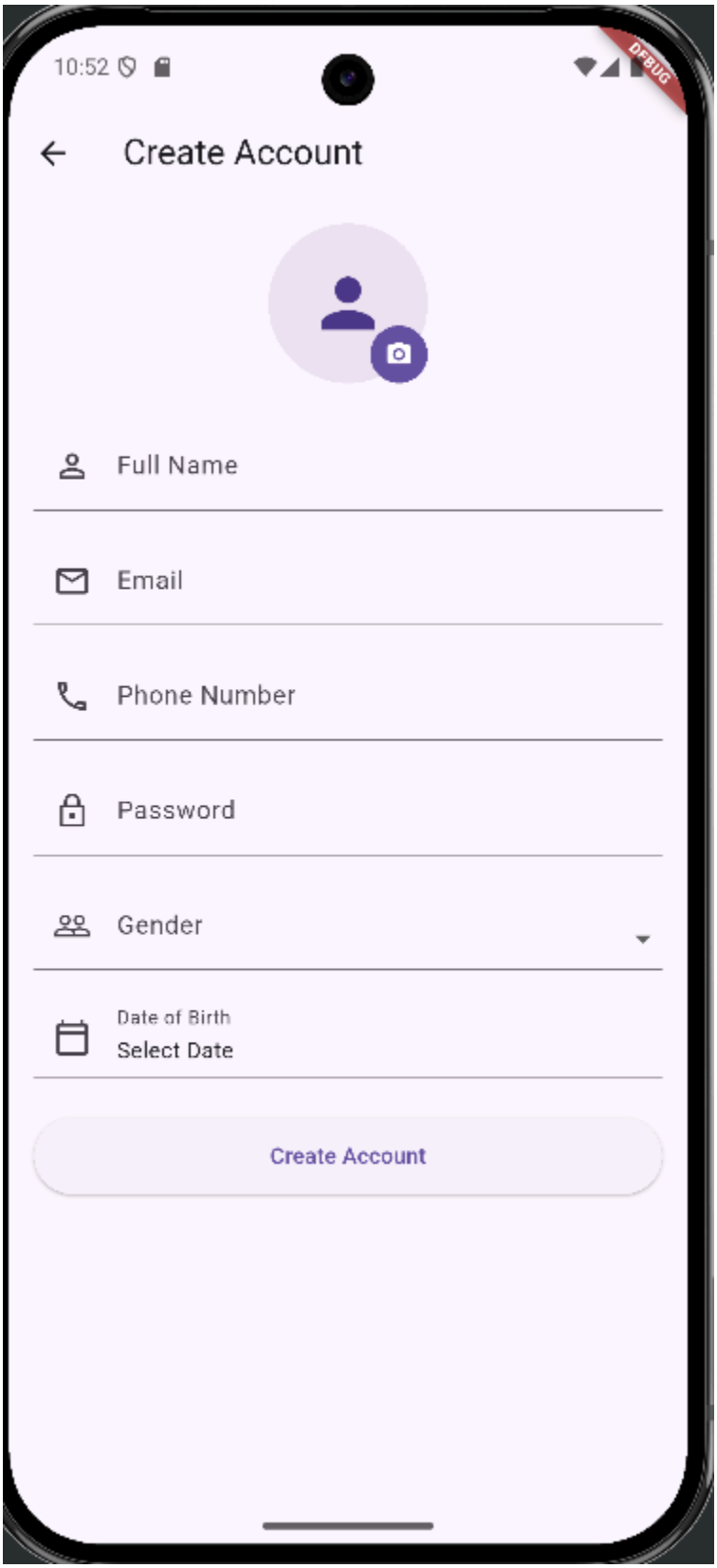
```
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your phone number';
    }
    if (value.length < 10) {
      return 'Please enter a valid phone number';
    }
    return null;
  },
),
const SizedBox(height: 16),
TextFormField(
  controller: _passwordController,
  decoration: const InputDecoration(
    labelText: 'Password',
    prefixIcon: Icon(Icons.lock_outline),
  ),
  obscureText: true,
  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: 16),
DropdownButtonFormField<String>(
  value: _selectedGender,
  decoration: const InputDecoration(
    labelText: 'Gender',
    prefixIcon: Icon(Icons.people_outline),
  ),
  items: ['Male', 'Female', 'Other'].map((gender) {
    return DropdownMenuItem(
      value: gender,
      child: Text(gender),
    );
  }).toList(),
  onChanged: (value) {
    setState(() {
      _selectedGender = value;
    });
  },
  validator: (value) {
    if (value == null) {
      return 'Please select your gender';
    }
    return null;
```

```
            },
          ),
          const SizedBox(height: 16),
          InkWell(
            onTap: _selectDate,
            child: InputDecorator(
              decoration: const InputDecoration(
                labelText: 'Date of Birth',
                prefixIcon: Icon(Icons.calendar_today_outlined),
              ),
              child: Text(
                _selectedDate != null
                    ? '${_selectedDate!.day}/${_selectedDate!.month}/${_selectedDate!.year}'
                    : 'Select Date',
              ),
            ),
          ),
          const SizedBox(height: 24),
          SizedBox(
            width: double.infinity,
            height: 48,
            child: ElevatedButton(
              onPressed: _submitForm,
              child: const Text('Create Account'),
            ),
          ),
        ],
      ),
    ),
  ),
 );
}

Future<void> _pickImage() async {
  // Implement image picker
}

Future<void> _selectDate() async {
  final DateTime? picked = await showDatePicker(
    context: context,
    initialDate: DateTime.now(),
    firstDate: DateTime(1900),
    lastDate: DateTime.now(),
  );
  if (picked != null) {
    setState(() {
      _selectedDate = picked;
    });
  }
}
```

```
  void _submitForm() {
    if (_formKey.currentState!.validate()) {
      // Create user object
      final user = {
        'name': _nameController.text,
        'email': _emailController.text,
        'phone': _phoneController.text,
        'gender': _selectedGender,
        'dateOfBirth': _selectedDate?.toIso8601String(),
        'profileImage': _profileImage,
      };

      // Save user data and navigate
      print(user); // Replace with actual API call
      Navigator.pushReplacementNamed(context, '/home');
    }
  }

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


10:51
DEBUG
Welcome Back
Login to continue
Email
Password
Forgot Password?
Login
Don't have an account? Sign Up


10:52
DEBUG
Create Account
Full Name
Email
Phone Number
Password
Gender
Date of Birth
Select Date
Create Account

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	44
Name	Manav Punjabi
Class	D15A/D15B
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:- Manav Punjabi **Class:-** D15A **Roll:No: -** 44

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';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Chat App',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: const LoginScreen(),
    );
  }
}
```

Login_page.dart:-

```
import 'package:flutter/material.dart';
import 'home_screen.dart';
import 'signup_screen.dart';  // Added this import

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

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

class _LoginScreenState extends State<LoginScreen>
{
  bool isLogin = true;
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: SingleChildScrollView(
          child: Padding(
            padding: const EdgeInsets.all(16.0),
            child: Column(
              mainAxisAlignment:
MainAxisAlignment.center,
              children: [
                const SizedBox(height: 50),
                // App Logo
                Container(
                  height: 100,
                  width: 100,
                  decoration: BoxDecoration(
                    color: Colors.blue,
                    borderRadius: BorderRadius.circular(20),
                  ),
                  child: const Icon(
                    Icons.chat,
                    size: 60,
                    color: Colors.white,
                  ),
                ),
                const SizedBox(height: 30),

                // Welcome Text
                Text(
                  isLogin ? 'Welcome Back' : 'Create
Account',
                  style: const TextStyle(
                    fontSize: 28,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                const SizedBox(height: 10),
                Text(
                  isLogin ? 'Login to continue' : 'Sign up to
get started',
                  style: TextStyle(
                    fontSize: 16,
                    color: Colors.grey[600],
                  ),
                ),
                const SizedBox(height: 30),

                // Email Field
                TextField(
                  controller: _emailController,
                  decoration: InputDecoration(
                    labelText: 'Email',
                    prefixIcon: const Icon(Icons.email),
                    border: OutlineInputBorder(
                      borderRadius:
BorderRadius.circular(12),
                    ),
                    enabledBorder: OutlineInputBorder(
                      borderRadius:
BorderRadius.circular(12),
                      borderSide: const BorderSide(color:
Colors.grey),
                    ),
                    focusedBorder: OutlineInputBorder(
                      borderRadius:
BorderRadius.circular(12),
                      borderSide: const BorderSide(color:
Colors.blue),
                    ),
                  ),
                  keyboardType:
TextInputType.emailAddress,
                ),
                const SizedBox(height: 16),

                // Password Field
                TextField(
                  controller: _passwordController,
                  decoration: InputDecoration(
                    labelText: 'Password',
                    prefixIcon: const Icon(Icons.lock),
                    border: OutlineInputBorder(
```

```
                borderRadius:
BorderRadius.circular(12),
              ),
              enabledBorder: OutlineInputBorder(
                borderRadius:
BorderRadius.circular(12),
                borderSide: const BorderSide(color:
Colors.grey),
              ),
              focusedBorder: OutlineInputBorder(
                borderRadius:
BorderRadius.circular(12),
                borderSide: const BorderSide(color:
Colors.blue),
              ),
            ),
            obscureText: true,
          ),
          const SizedBox(height: 20),

          // Forgot Password Button
          if (isLogin)
            Align(
              alignment: Alignment.centerRight,
              child: TextButton(
                onPressed: () {
                  // Forgot password logic
                },
                child: const Text('Forgot Password?'),
              ),
            ),
          const SizedBox(height: 20),

          // Login/Signup Button
          SizedBox(
            width: double.infinity,
            height: 50,
            child: ElevatedButton(
              onPressed: () {
                // Login/Signup logic ke baad
                Navigator.pushReplacement(
                  context,
                  MaterialPageRoute(builder: (context)
=> const HomeScreen()),
                );
              },
              style: ElevatedButton.styleFrom(
                shape: RoundedRectangleBorder(
                  borderRadius:
BorderRadius.circular(12),
                ),
              ),
              child: Text(
                isLogin ? 'Login' : 'Sign Up',
                style: const TextStyle(fontSize: 16),
              ),
            ),
          ),
          const SizedBox(height: 20),

          // Toggle Login/Signup
          Row(
            mainAxisAlignment:
MainAxisAlignment.center,
            children: [
              Text(
                isLogin ? 'Don\'t have an account?' :
'Already have an account?',
                style: TextStyle(color:
Colors.grey[600]),
              ),
              TextButton(
                onPressed: () {
                  if (isLogin) {
                    // If on login, go to signup screen
                    Navigator.push(
                      context,
                      MaterialPageRoute(builder:
(context) => const SignupScreen()),
                    );
                  } else {
                    // If seeing signup text, toggle back to
login
                    setState(() {
                      isLogin = !isLogin;
                    });
                  }
                },
                child: Text(
                  isLogin ? 'Sign Up' : 'Login',
                  style: const TextStyle(fontWeight:
FontWeight.bold),
                ),
              ),
            ],
          ),
        ],
      ),
    ),
  );
 }
}
```

profile_screen.dart

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

import 'home_screen.dart';

import 'signup_screen.dart';  // Added this import


class LoginScreen extends StatefulWidget {

 const LoginScreen({super.key});


 @override

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

}


class _LoginScreenState extends State<LoginScreen> {

 bool isLogin = true;

        final _emailController = TextEditingController();

       final _passwordController = TextEditingController();


 @override

 void dispose() {

  _emailController.dispose();

  _passwordController.dispose();

  super.dispose();

 }


 @override

 Widget build(BuildContext context) {

  return Scaffold(

    body: SafeArea(

     child: SingleChildScrollView(

      child: Padding(

                    padding: const EdgeInsets.all(16.0),

        child: Column(

                  mainAxisAlignment: MainAxisAlignment.center,

          children: [

           const SizedBox(height: 50),

           // App Logo

           Container(

            height: 100,

            width: 100,

            decoration: BoxDecoration(

             color: Colors.blue,

                        borderRadius: BorderRadius.circular(20),

            ),

            child: const Icon(

             Icons.chat,

             size: 60,

             color: Colors.white,

            ),

           ),

           const SizedBox(height: 30),


           // Welcome Text

           Text(

             isLogin ? 'Welcome Back' : 'Create Account',

            style: const TextStyle(

             fontSize: 28,
```

```
                fontWeight:
FontWeight.bold,
              ),
            ),
            const SizedBox(height: 10),
            Text(
                isLogin ? 'Login to
continue' : 'Sign up to get started',
              style: TextStyle(
                fontSize: 16,
                color: Colors.grey[600],
              ),
            ),
            const SizedBox(height: 30),

            // Email Field
            TextField(
                controller:
_emailController,
                decoration:
InputDecoration(
              labelText: 'Email',
                prefixIcon: const
Icon(Icons.email),
                border:
OutlineInputBorder(
                borderRadius:
BorderRadius.circular(12),
              ),
                enabledBorder:
OutlineInputBorder(
                borderRadius:
BorderRadius.circular(12),
                borderSide: const
BorderSide(color: Colors.grey),
              ),
                focusedBorder:
OutlineInputBorder(
                borderRadius:
BorderRadius.circular(12),
                borderSide: const
BorderSide(color: Colors.blue),
              ),
            ),
                keyboardType:
TextInputType.emailAddress,
            ),
            const SizedBox(height: 16),

            // Password Field
            TextField(
                controller:
_passwordController,
                decoration:
InputDecoration(
              labelText: 'Password',
                prefixIcon: const
Icon(Icons.lock),
                border:
OutlineInputBorder(
                borderRadius:
BorderRadius.circular(12),
              ),
                enabledBorder:
OutlineInputBorder(
                borderRadius:
BorderRadius.circular(12),
                borderSide: const
BorderSide(color: Colors.grey),
              ),
                focusedBorder:
```

```
OutlineInputBorder(
                        borderRadius:
BorderRadius.circular(12),
                      borderSide: const
BorderSide(color: Colors.blue),
                ),
              ),
              obscureText: true,
            ),
            const SizedBox(height: 20),

            // Forgot Password Button
            if (isLogin)
              Align(
                          alignment:
Alignment.centerRight,
                child: TextButton(
                  onPressed: () {
                    // Forgot password logic
                  },
                    child: const Text('Forgot
Password?'),
                ),
              ),
            const SizedBox(height: 20),

            // Login/Signup Button
            SizedBox(
              width: double.infinity,
              height: 50,
              child: ElevatedButton(
                onPressed: () {
                  // Login/Signup logic ke
baad

                  Navigator.pushReplacement(
                    context,

MaterialPageRoute(builder:  (context)
=> const HomeScreen()),
                  );
                },
                          style:
ElevatedButton.styleFrom(
                          shape:
RoundedRectangleBorder(
                      borderRadius:
BorderRadius.circular(12),
                    ),
                  ),
                  child: Text(
                    isLogin ? 'Login' : 'Sign
Up',
                          style:  const
TextStyle(fontSize: 16),
                  ),
                ),
              ),
            const SizedBox(height: 20),

            // Toggle Login/Signup
            Row(
                    mainAxisAlignment:
MainAxisAlignment.center,
              children: [
                Text(
                  isLogin ? 'Don\'t have an
account?' : 'Already have an account?',
```

```
                    style: TextStyle(color:
Colors.grey[600]),
                ),
                TextButton(
                  onPressed: () {
                    if (isLogin) {
                          // If on login, go to
signup screen
                      Navigator.push(
                        context,

MaterialPageRoute(builder:    (context)
=> const SignupScreen()),
                      );
                    } else {
                      // If seeing signup text,
toggle back to login
                      setState(() {
                        isLogin = !isLogin;
                      });
                    }
                  },
                  child: Text(
                        isLogin ? 'Sign Up' :
'Login',
                                  style:  const
TextStyle(fontWeight:
FontWeight.bold),
                  ),
                ),
              ],
            ),
          ],
        ),
                ),
              ),
            ),
          );
        }
      }
```

search_screen.dart

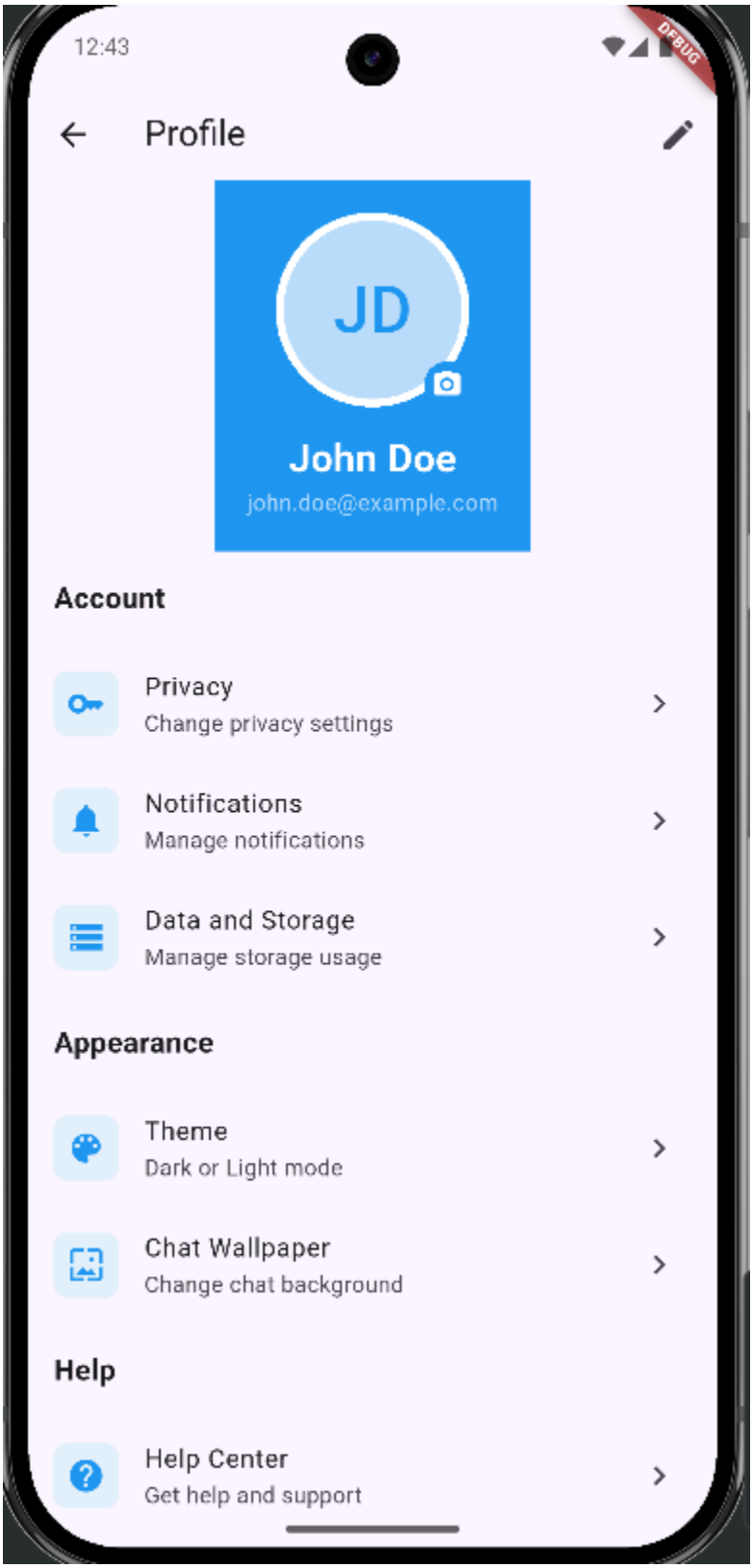
```
import 'package:flutter/material.dart';

import 'chat_screen.dart';


class SearchScreen extends StatefulWidget {

 const SearchScreen({super.key});


 @override

   State<SearchScreen> createState() => _SearchScreenState();

}


class _SearchScreenState extends State<SearchScreen> {

          final TextEditingController _searchController = TextEditingController();

 bool _isSearching = false;


 // Dummy data for demonstration

      final List<Map<String, String>> _recentSearches = [

   {'name': 'John Doe', 'status': 'Online'},

    {'name': 'Jane Smith', 'status': 'Last seen 2h ago'},

       {'name': 'Mike Johnson', 'status': 'Online'},

 ];


      final List<Map<String, String>> _suggestedUsers = [

   {'name': 'Alice Brown', 'status': 'Online'},

    {'name': 'Bob Wilson', 'status': 'Last seen yesterday'},

   {'name': 'Carol White', 'status': 'Online'},

   {'name': 'David Black', 'status': 'Last seen 30m ago'},

 ];


 @override

 Widget build(BuildContext context) {

  return Scaffold(

   appBar: AppBar(

    title: _isSearching

       ? _buildSearchField()

       : const Text('Search'),

    actions: [

     IconButton(

                icon: Icon(_isSearching ? Icons.close : Icons.search),

       onPressed: () {

        setState(() {

         _isSearching = !_isSearching;

         if (!_isSearching) {

          _searchController.clear();

         }

        });

       },

      ),

    ],

   ),

   body: Column(

                crossAxisAlignment: CrossAxisAlignment.start,

    children: [

     if (!_isSearching) ...[
```

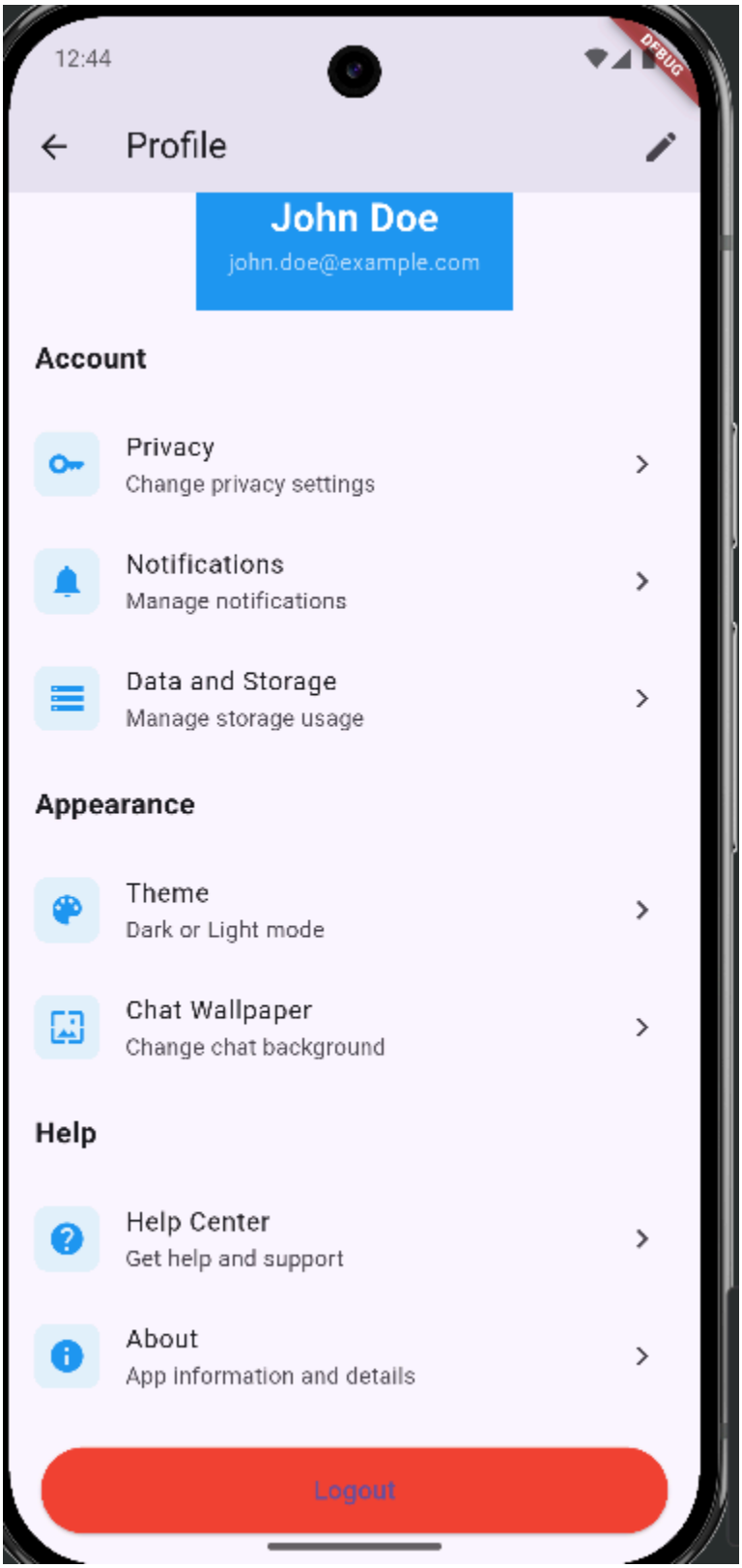
```
        // Recent Searches
        const Padding(
          padding: EdgeInsets.all(16),
          child: Text(
            'Recent Searches',
            style: TextStyle(
              fontSize: 18,
              fontWeight: FontWeight.bold,
            ),
          ),
        ),
        ListView.builder(
          shrinkWrap: true,
                                    itemCount:
_recentSearches.length,
          itemBuilder: (context, index) {
            return _buildUserTile(

_recentSearches[index]['name']!,

_recentSearches[index]['status']!,
              isRecent: true,
            );
          },
        ),

        // Suggested Users
        const Padding(
          padding: EdgeInsets.all(16),
          child: Text(
            'Suggested Users',
            style: TextStyle(
              fontSize: 18,
              fontWeight: FontWeight.bold,
            ),
          ),
        ),
        Expanded(
          child: ListView.builder(
                                    itemCount:
_suggestedUsers.length,
            itemBuilder: (context, index) {
              return _buildUserTile(

_suggestedUsers[index]['name']!,

_suggestedUsers[index]['status']!,
              );
            },
          ),
        ),
      ],
      if (_isSearching)
        Expanded(
          child: _buildSearchResults(),
        ),
    ],
  ),
                          floatingActionButton:
FloatingActionButton(
    onPressed: () {
      _showAddContactDialog();
    },
    child: const Icon(Icons.person_add),
  ),
```

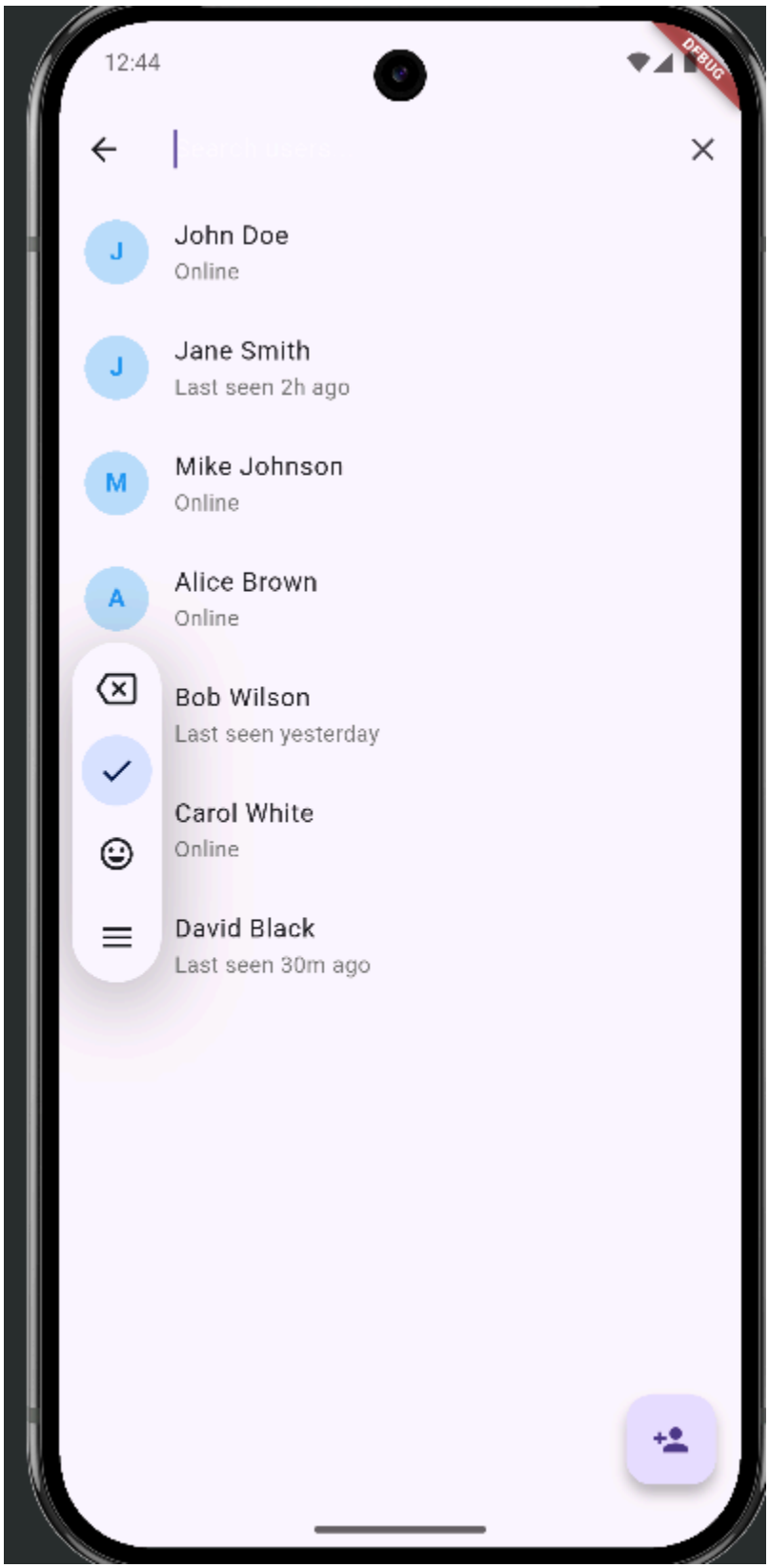
```
    );
  }

  Widget _buildSearchField() {
    return TextField(
      controller: _searchController,
      autofocus: true,
      decoration: const InputDecoration(
        hintText: 'Search users...',
        border: InputBorder.none,
        hintStyle: TextStyle(color: Colors.white70),
      ),
      style: const TextStyle(color: Colors.white),
      onChanged: (value) {
        setState(() {});
      },
    );
  }

  Widget _buildSearchResults() {
    // Filter users based on search query
    final searchQuery = _searchController.text.toLowerCase();
    final searchResults = [..._recentSearches, ..._suggestedUsers]
        .where((user) => user['name']!.toLowerCase().contains(searchQuery))
        .toList();

    return ListView.builder(
      itemCount: searchResults.length,
      itemBuilder: (context, index) {
        return _buildUserTile(
          searchResults[index]['name']!,
          searchResults[index]['status']!,
        );
      },
    );
  }

  Widget _buildUserTile(String name, String status, {bool isRecent = false}) {
    return ListTile(
      leading: CircleAvatar(
        backgroundColor: Colors.blue[100],
        child: Text(
          name[0],
          style: const TextStyle(
            color: Colors.blue,
            fontWeight: FontWeight.bold,
          ),
        ),
      ),
      title: Text(name),
      subtitle: Text(
        status,
        style: TextStyle(color: Colors.grey[600]),
      ),
      trailing: isRecent
          ? IconButton(
        icon: const Icon(Icons.close),
```

```
      onPressed: () {
        // Remove from recent searches
      },
    )
        : null,
    onTap: () {
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context)
=> const ChatScreen()),
      );
    },
  );
 }


 void _showAddContactDialog() {
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      title: const Text('Add New Contact'),
      content: Column(
        mainAxisSize: MainAxisSize.min,
        children: [
          TextField(
            decoration: const InputDecoration(
              labelText: 'Name',
              prefixIcon: Icon(Icons.person),
            ),
          ),
          const SizedBox(height: 16),
          TextField(
            decoration: const InputDecoration(
              labelText: 'Email or Phone',
              prefixIcon: Icon(Icons.email),
            ),
          ),
        ],
      ),
      actions: [
        TextButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: const Text('Cancel'),
        ),
        ElevatedButton(
          onPressed: () {
            // Add contact logic
            Navigator.pop(context);
          },
          child: const Text('Add'),
        ),
      ],
    ),
  );
 }
}
```

OUTPUT: -


12:43
DEBUG
Profile
JD
John Doe
john.doe@example.com
Account
Privacy
Change privacy settings
Notifications
Manage notifications
Data and Storage
Manage storage usage
Appearance
Theme
Dark or Light mode
Chat Wallpaper
Change chat background
Help
Help Center
Get help and support


12:44
DEBUG
Profile
John Doe
john.doe@example.com
Account
Privacy
Change privacy settings
Notifications
Manage notifications
Data and Storage
Manage storage usage
Appearance
Theme
Dark or Light mode
Chat Wallpaper
Change chat background
Help
Help Center
Get help and support
About
App information and details
Logout


12:44
DEBUG
Search users
J
John Doe
Online
J
Jane Smith
Last seen 2h ago
M
Mike Johnson
Online
A
Alice Brown
Online
Bob Wilson
Last seen yesterday
Carol White
Online
David Black
Last seen 30m ago

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	44
Name	Manav Punjabi
Class	D15A/D15B
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:- Manav Punjabi Class:- D15A Roll:No: - 44

AIM: - To connect Flutter UI with Firebase database.

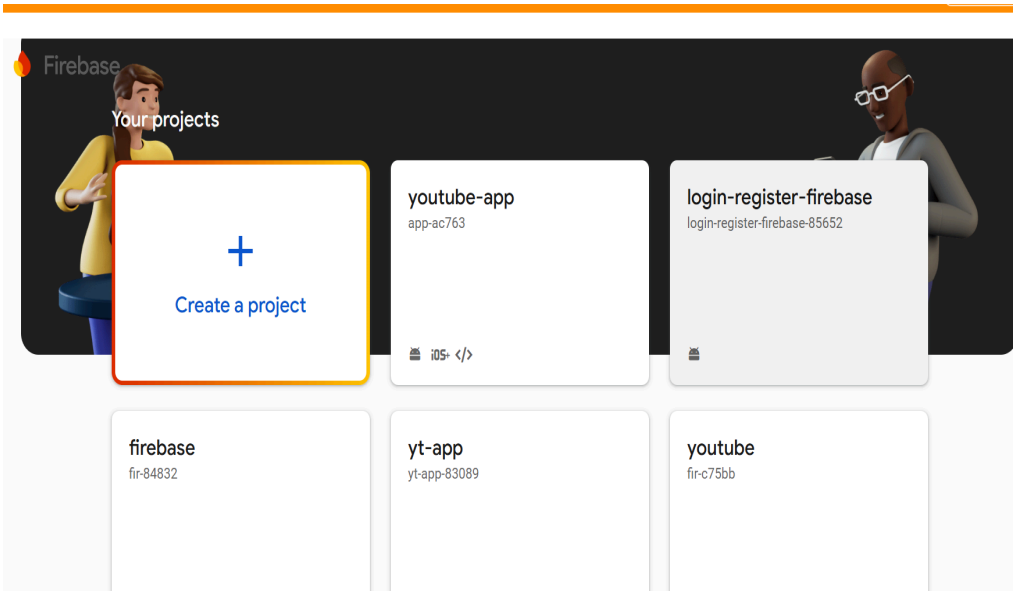
Theory: -
Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase.

Firebase, a Backend-as-aService (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**

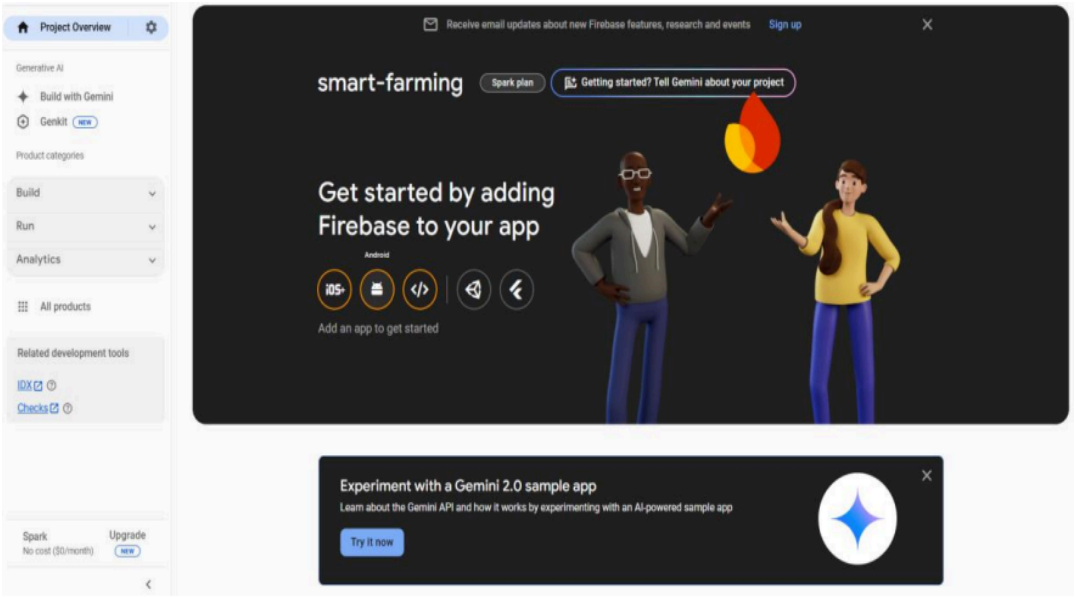
Step 1:

1.1) Go to Firebase Console and Create a Firebase Project

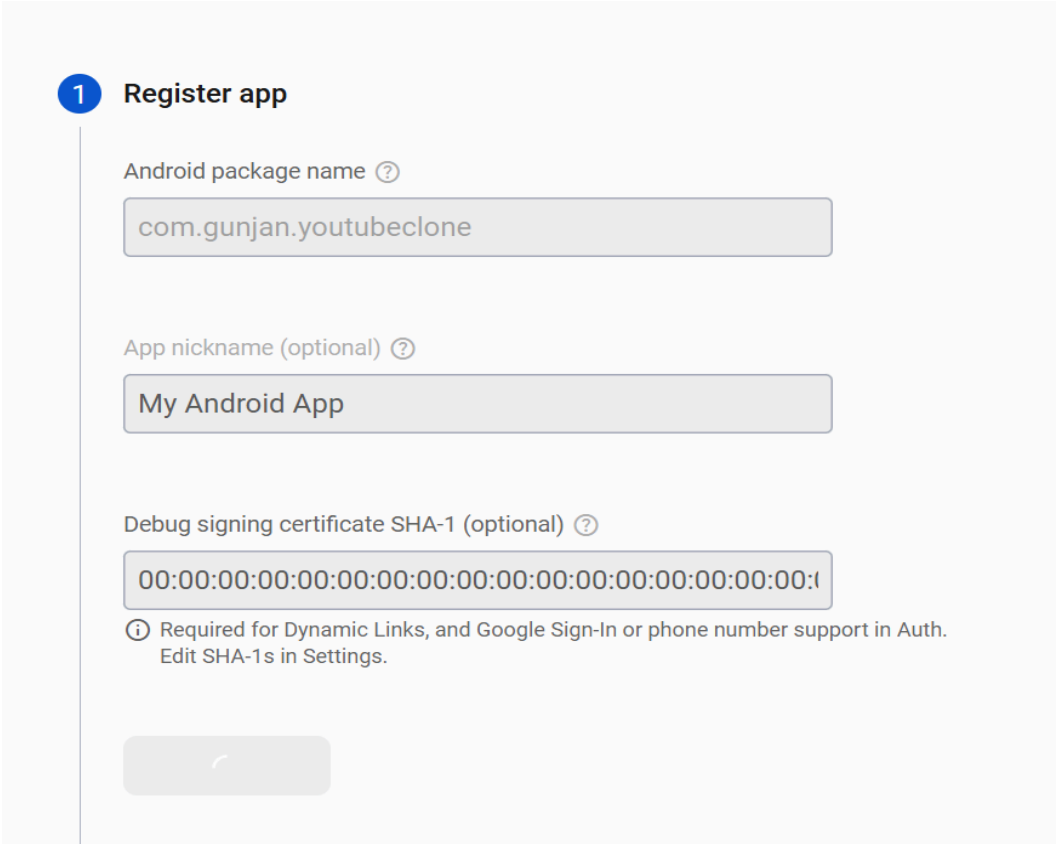


Step 2:- Add Firebase to Your Flutter App

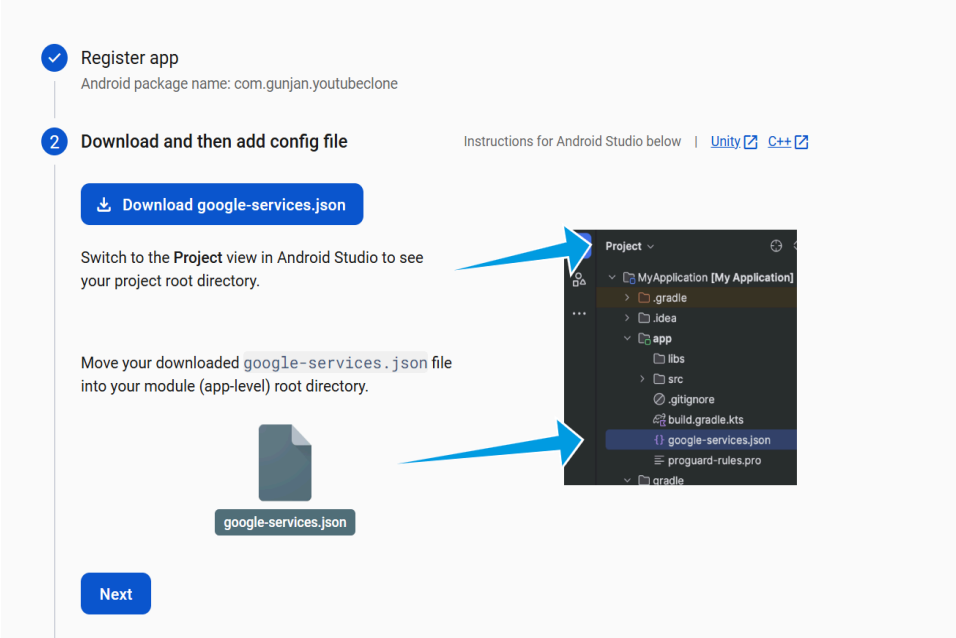
2.1) Click on Android/iOS/Web based on your Flutter application



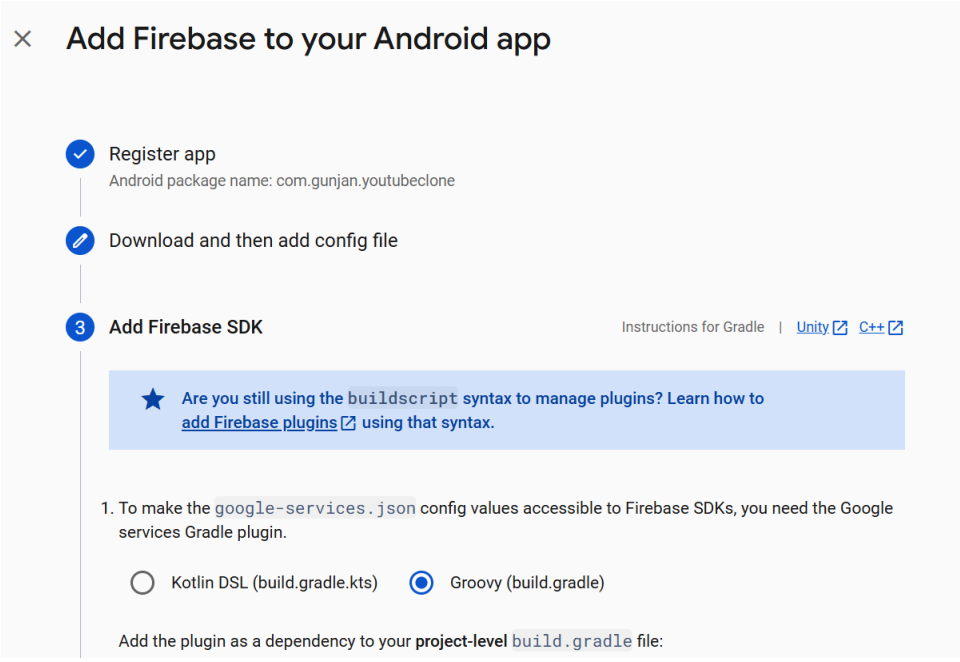
2.2) Register your app with a unique package name (found in android/app/build.gradle for Android)

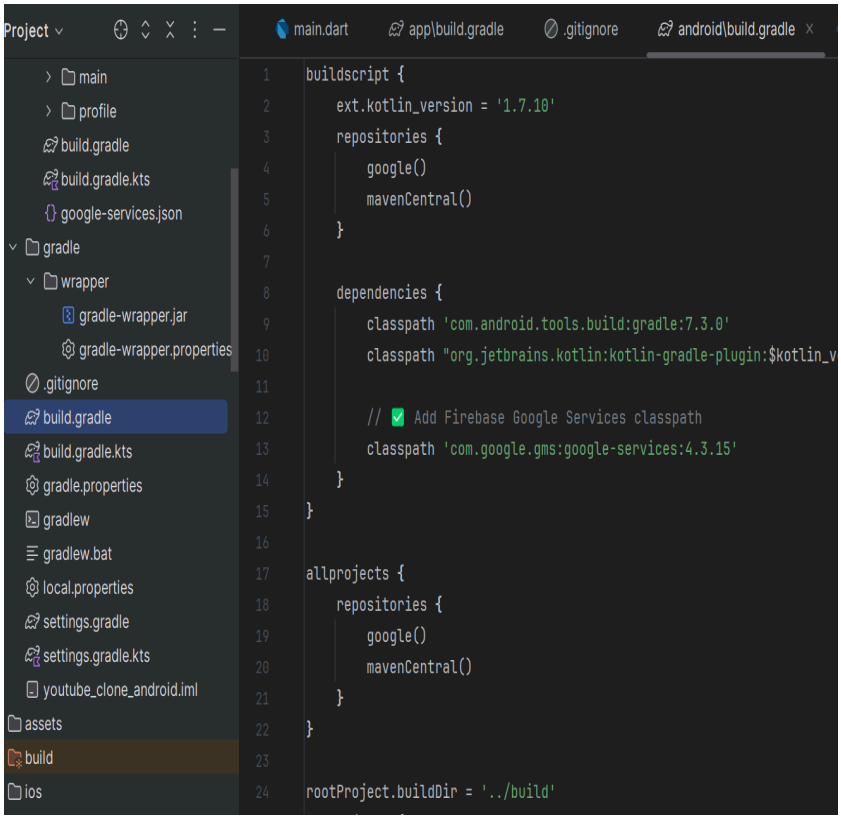


2.3) Download the google-services.json (for Android) & place the JSON file inside android/app/ directory.



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



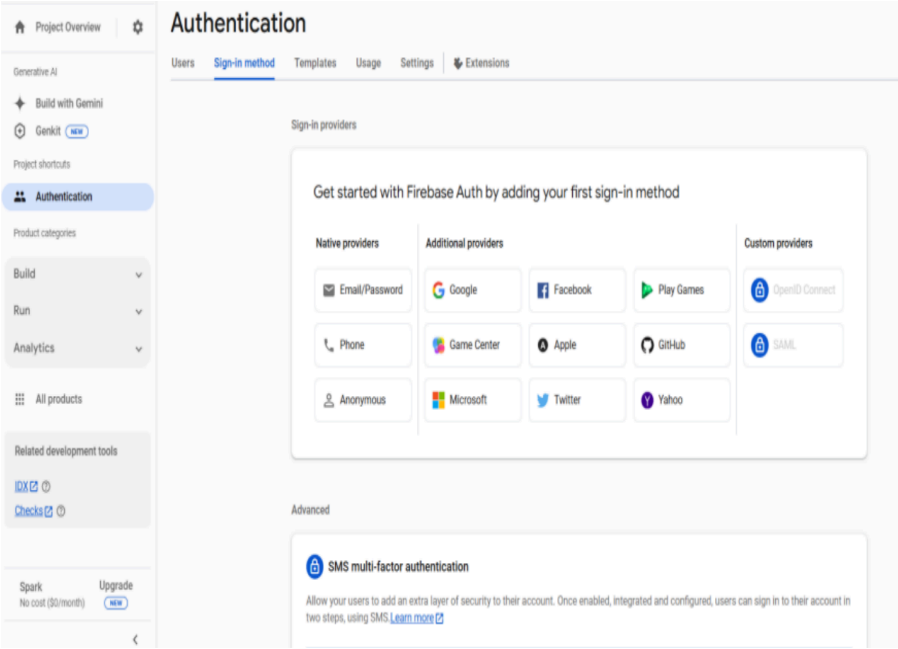


Step 3: - Add Firebase Authentication to Your App

3.1) Add Firebase Authentication Dependencies

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

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



3.3) Implement Authentication in **Flutter Modify main.dart**

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

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

Step 4: -Configure Firebase Realtime Database

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:

login_page.dart

```dart
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:youtube_clone/features/auth/repository/auth_service.dart';

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

 @override
 Widget build(BuildContext context, WidgetRef ref) {
   return Scaffold(
     body: SafeArea(
       child: Center(
         child: Column(
           children: [
             Padding(
               padding: const EdgeInsets.only(
                 top: 20,
                 bottom: 25,
               ),
               child: Image.asset(
                 "assets/images/youtube-signin.jpg",
                 height: 150,
               ),
             ),
             const Text(
               "Welcome To Youtube",
               style: TextStyle(
                 fontSize: 30,
                 fontWeight: FontWeight.bold,
                 color: Colors.blueGrey,
               ),
             ),
             const Spacer(),
             Padding(
               padding: const EdgeInsets.only(bottom: 55),
               child: GestureDetector(
                 onTap: () async {
                   await ref.read(authServiceProvider).signInWithGoogle();
                 },
                 child: Image.asset(
                   "assets/images/signinwithgoogle.png",
                   height: 60,
                 ),
               ),
             ),
           ],
         ),
```

firebase_option.dart:

```dart
// File generated by FlutterFire CLI.
// ignore_for_file: lines_longer_than_80_chars,
avoid_classes_with_only_static_members
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.
///
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
///   options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
 static FirebaseOptions get currentPlatform {
   if (kIsWeb) {
     return web;
   }
   switch (defaultTargetPlatform) {
     case TargetPlatform.android:
       return android;
     case TargetPlatform.iOS:
       return ios;
     case TargetPlatform.macOS:
       return macos;
     case TargetPlatform.windows:
       return windows;
     case TargetPlatform.linux:
       throw UnsupportedError(
         'DefaultFirebaseOptions have not been configured for linux - '
         'you can reconfigure this by running the FlutterFire CLI again.',
       );
     default:
       throw UnsupportedError(
         'DefaultFirebaseOptions are not supported for this platform.',
       );
   }
 }

 static const FirebaseOptions android = FirebaseOptions(
   apiKey: 'AIzaSyDc-JOUuH78RpTHXObsa0RDVmia8cRBsjQ',
   appId: '1:188407165111:android:b564cebc78ad2e861edd32',
   messagingSenderId: '188407165111',
   projectId: 'app-ac763',
```

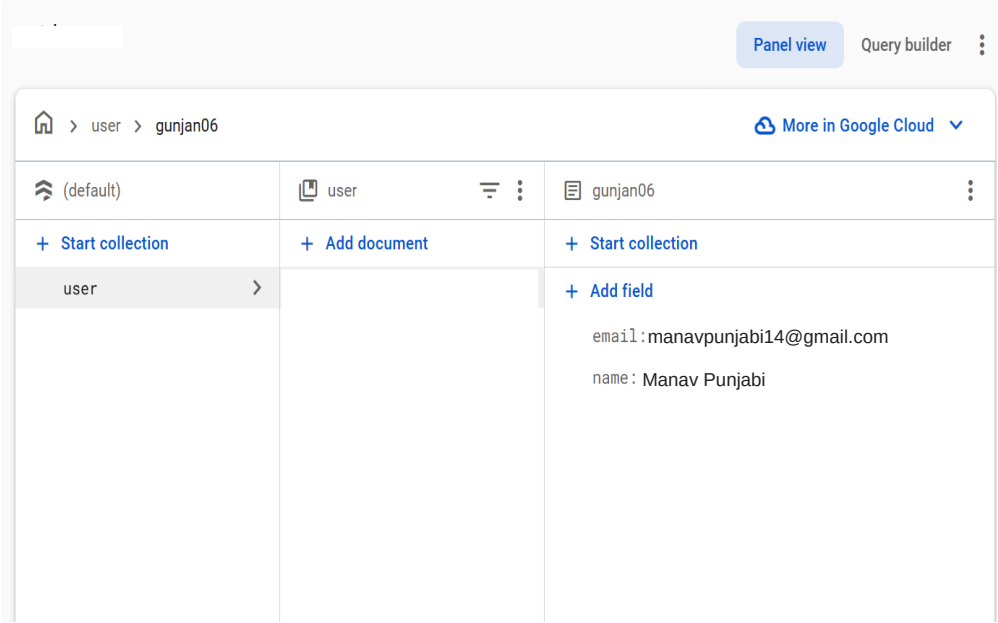
```
    storageBucket: 'app-ac763.firebasestorage.app',
  );

  static const FirebaseOptions ios = FirebaseOptions(
    apiKey: 'AIzaSyB_FXM7S3pqeGvJE0e6DTmAL-Uyfd16agc',
    appId: '1:188407165111:ios:404e3cf1890c2b481edd32',
    messagingSenderId: '188407165111',
    projectId: 'app-ac763',
    storageBucket: 'app-ac763.firebasestorage.app',
    iosBundleId: 'com.example.youtubeClone',
  );

  static const FirebaseOptions web = FirebaseOptions(
    apiKey: 'AIzaSyBSVAuYp50tr_S8-O60E7o5dKNKW-1Ejew',
    appId: '1:188407165111:web:4daa739182e168b11edd32',
    messagingSenderId: '188407165111',
    projectId: 'app-ac763',
    authDomain: 'app-ac763.firebaseapp.com',
    storageBucket: 'app-ac763.firebasestorage.app',
    measurementId: 'G-LVMQ4SZ270',
  );

  static const FirebaseOptions macos = FirebaseOptions(
    apiKey: 'AIzaSyB_FXM7S3pqeGvJE0e6DTmAL-Uyfd16agc',
    appId: '1:188407165111:ios:404e3cf1890c2b481edd32',
    messagingSenderId: '188407165111',
    projectId: 'app-ac763',
    storageBucket: 'app-ac763.firebasestorage.app',
    iosBundleId: 'com.example.youtubeClone',
  );

  static const FirebaseOptions windows = FirebaseOptions(
    apiKey: 'AIzaSyBSVAuYp50tr_S8-O60E7o5dKNKW-1Ejew',
    appId: '1:188407165111:web:0e4eb752f4c3873e1edd32',
    messagingSenderId: '188407165111',
    projectId: 'app-ac763',
    authDomain: 'app-ac763.firebaseapp.com',
    storageBucket: 'app-ac763.firebasestorage.app',
    measurementId: 'G-86VN7VQZJX',
  );

}
```

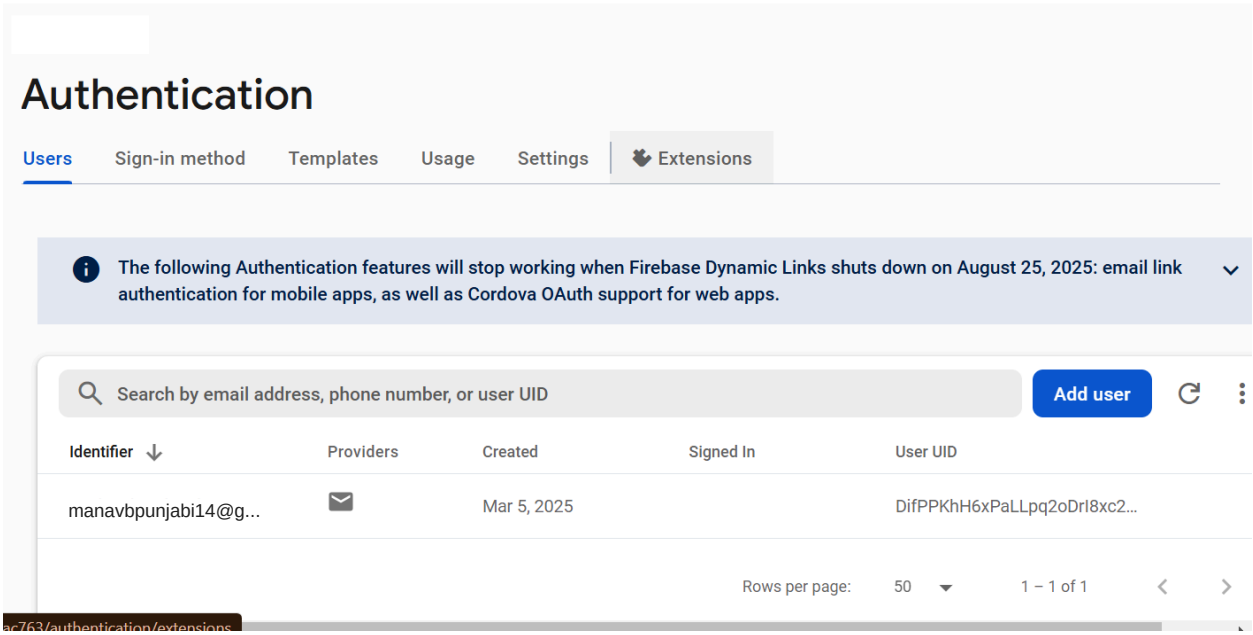
home_page.dart:

```dart
// ignore_for_file: invalid_use_of_visible_for_testing_member

import 'package:cached_network_image/cached_network_image.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:path/path.dart';
import 'package:youtube_clone/cores/screens/error_page.dart';
import 'package:youtube_clone/cores/screens/loader.dart';
import 'package:youtube_clone/cores/widgets/image_button.dart';
import 'package:youtube_clone/features/account/account_page.dart';
import 'package:youtube_clone/features/auth/provider/user_provider.dart';
import 'package:youtube_clone/features/content/bottom_navigation.dart';
import 'package:youtube_clone/features/upload/upload_bottom_sheet.dart';
import 'package:youtube_clone/pages_list.dart';

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

 @override
 State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
 int currentIndex = 0;
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     backgroundColor: const Color(0xffFFFFFF),
     body: SafeArea(
       child: Column(
         children: [
           Row(
             children: [
               Image.asset(
                 "assets/images/youtube.jpg",
                 height: 36,
               ),
               const SizedBox(width: 4),
               const Spacer(),
               Padding(
                 padding: const EdgeInsets.only(right: 12),
                 child: SizedBox(
                   height: 42,
                   child: ImageButton(
                     image: "cast.png",
                     onPressed: () {},
                     haveColor: false,
                   ),
```

```
            ),
          ),
          SizedBox(
            height: 38,
            child: ImageButton(
              image: "notification.png",
              onPressed: () {},
              haveColor: false,
            ),
          ),
          Padding(
            padding: const EdgeInsets.only(left: 12, right: 15),
            child: SizedBox(
              height: 41.5,
              child: ImageButton(
                image: "search.png",
                onPressed: () {},
                haveColor: false,
              ),
            ),
          ),
          Consumer(
            builder: (context, ref, child) {
              return ref.watch(currentUserProvider).when(
                    data: (currentUser) => Padding(
                      padding: const EdgeInsets.only(right: 12),
                      child: GestureDetector(
                        onTap: () {
                          Navigator.push(
                            context,
                            MaterialPageRoute(
                              builder: (context) => AccountPage(
                                user: currentUser,
                              ),
                            ),
                          );
                        },
                        child: CircleAvatar(
                          radius: 14,
                          backgroundColor: Colors.grey,
                          backgroundImage: CachedNetworkImageProvider(
                            currentUser.profilePic,
                          ),
                        ),
                      ),
                    ),
                    error: (error, stackTrace) => const ErrorPage(),
                    loading: () => const Loader(),
                  );
```

```
                },
              ),
            ],
          ),
          Expanded(
            child: pages[currentIndex],
          ),
        ],
      ),
    ),
    bottomNavigationBar: BottomNavigation(
      onPressed: (index) {
        if (index != 2) {
          currentIndex = index;
          setState(() {});
        } else {
          showModalBottomSheet(
            context: context,
            builder: (context) => const CreateBottomSheet(),
          );
        }
      },
    ),
  );
 }
}
```


Panel view
Query builder
user
gunjan06
More in Google Cloud
(default)
user
gunjan06
Start collection
Add document
Start collection
user
Add field
email: manavpunjabi14@gmail.com
name: Manav Punjabi


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
manavbpunjabi14@g...
Mar 5, 2025
DifPPKhH6xPaLLpq2oDrI8xc2...
Rows per page:
50
1 – 1 of 1

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.	44
Name	Manav Punjabi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

PWA Experiment -7

Devansh Wadhwani - 64
Kunal Punjabi - 43
Manav Punjabi - 44
Hitesh Rohra - 46

Class : D15A

❖ **AIM**

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

❖ **Theory**:-

Regular Web Application

A regular web application is a website designed to be accessible on various devices, ensuring that content adjusts dynamically to different screen sizes. Built using web technologies such as HTML, CSS, JavaScript, and Ruby, these applications function through a browser. While they can leverage some native device features, their functionality is dependent on browser compatibility. For instance, a feature may work on Google Chrome but not on Safari or Mozilla Firefox due to browser limitations.

Progressive Web Application (PWA)

A Progressive Web Application (PWA) is an advanced version of a regular web app, integrating additional features to enhance the user experience. PWAs combine the best elements of desktop and mobile applications, delivering a seamless experience across platforms.

❖ Key Differences Between PWAs and Regular Web Apps

1. Native-Like Experience

While both PWAs and regular web apps utilize standard web technologies like HTML, CSS, and JavaScript, PWAs offer a user experience similar to native mobile applications. PWAs can access device-specific features such as push notifications without relying on a particular browser, creating a smoother, more integrated experience.

2. Ease of Access

Unlike traditional mobile apps that require time-consuming downloads and storage space, PWAs can be installed directly via a URL. Users can add a PWA to their home screen with a simple link, eliminating installation complexities and ensuring easy access while keeping brand presence strong.

3. Enhanced Performance

PWAs utilize caching mechanisms to pre-load content, such as text, stylesheets, and images, allowing for faster loading times. This significantly improves user engagement and retention by reducing waiting periods, ultimately benefiting businesses by increasing interaction rates.

4. Improved User Engagement

PWAs efficiently leverage push notifications and native device features to keep users engaged. Unlike regular web apps, their functionality is not restricted by browser dependencies, enabling businesses to notify users about updates, offers, and promotions without disruptions.

5. Real-Time Updates

A major advantage of PWAs is their ability to update automatically without requiring users to download new versions from an app store. Developers can push updates directly from the server, ensuring that users always access the latest features and improvements instantly.

6. Search Engine Optimization (SEO) Benefits

Since PWAs function within web browsers, they can be indexed by search engines, improving their visibility in search results. This gives them a strategic advantage over native apps, which are limited to app store searches.

7. Cost-Effective Development

Unlike native mobile apps, PWAs do not require approval or submission to app stores, reducing development and maintenance costs.

❖ Advantages and Limitations of PWAs

➢ **Advantages:**

- **Progressive:** Compatible with all browsers and devices, following the principle of progressive enhancement.
- **Responsive:** Adapts to different screen sizes, including desktops, tablets, and smartphones.
- **App-Like Feel:** Mimics the experience of native applications in navigation and interaction.
- **Always Updated:** Service workers ensure real-time updates without requiring user intervention.
- **Secure:** Delivered over HTTPS, ensuring secure data transfer and protection against cyber threats.
- **SEO-Friendly:** Can be indexed by search engines, enhancing discoverability.
- **Re-Engagement:** Enables push notifications to encourage continued user interaction.
- **Installable:** Allows users to add the app to their home screen without app store downloads.
- **Offline Functionality:** Can function in low or no connectivity conditions using cached content.

➢ **Limitations:**

- **Higher Battery Consumption:** PWAs tend to consume more battery due to constant background processes.
- **Limited Hardware Access:** Some device features, such as advanced sensors and Bluetooth, may not be fully accessible.
- **Offline Mode Constraints:** Some offline capabilities remain limited, depending on browser support.
- **No App Store Presence:** PWAs cannot generate traffic from app store searches.
- **Lack of Centralized Control:** Unlike native apps, PWAs do not undergo an official approval process, potentially affecting credibility.

CODE:

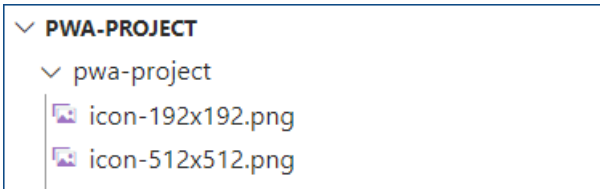
Index.html :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta name="description" content="Web site created using create-react-app" />

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

    <link  rel="apple-touch-icon"  href="%PUBLIC_URL%/apple-touch-icon.png">
    <link rel="apple-touch-icon" sizes="192x192" href="%PUBLIC_URL%/logo.png">
    <link rel="apple-touch-icon" sizes="512x512" href="%PUBLIC_URL%/logo.png">

    <link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico" type="image/x-icon"
/>

    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.15.4/css/all.css"
        integrity="sha384-
DyZ88mC6Up2uqS4h/KRgHuoeGwBcD4Ng9SiP4dIRy0EXTlnuz47vAwmeGwVChigm
"
        crossorigin="anonymous" />

    <title>Delicia Food App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

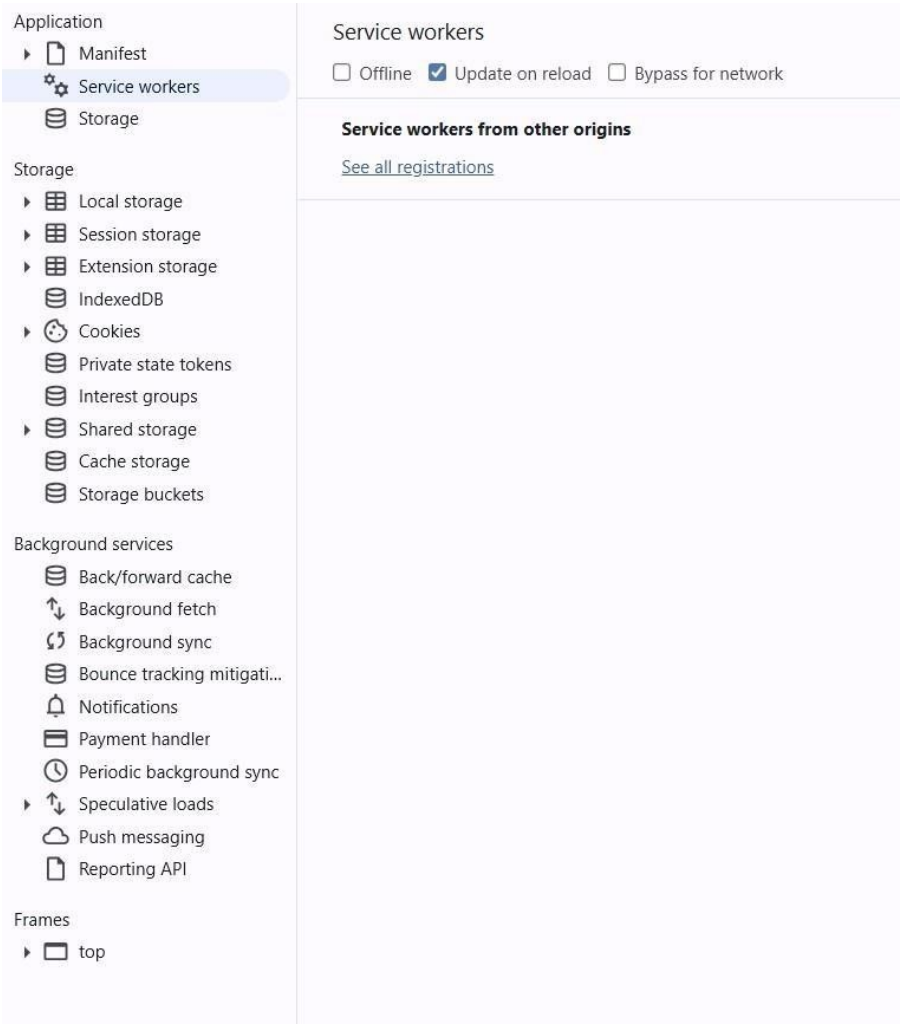
Manifest.json

```
  "name": "Delice Food App",
  "short_name": "Delice",
  "start_url": "./index.html",
  "scope": "./",
  "icons": [
    {
      "src": "icon-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "icon-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ],
  "theme_color": "#ffd31d",
  "background_color": "#333",
  "display": "standalone"
}
```

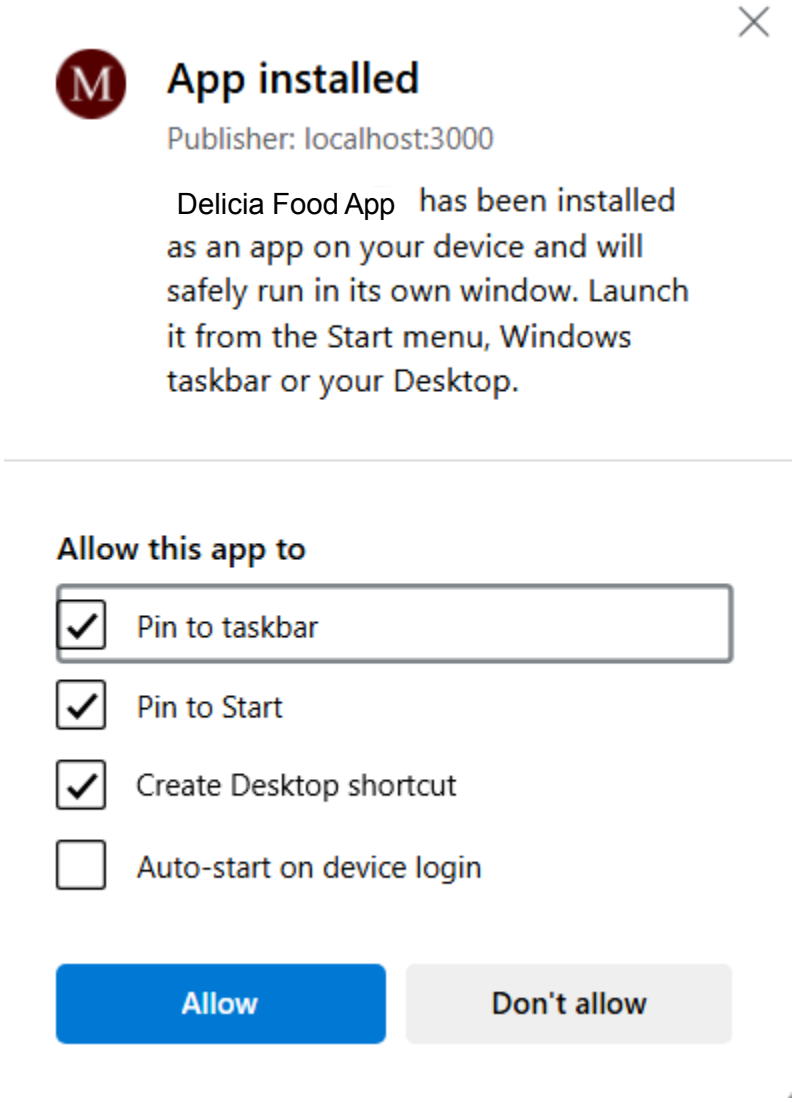
Icons

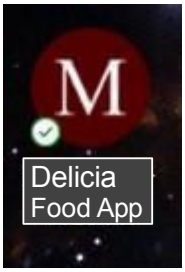


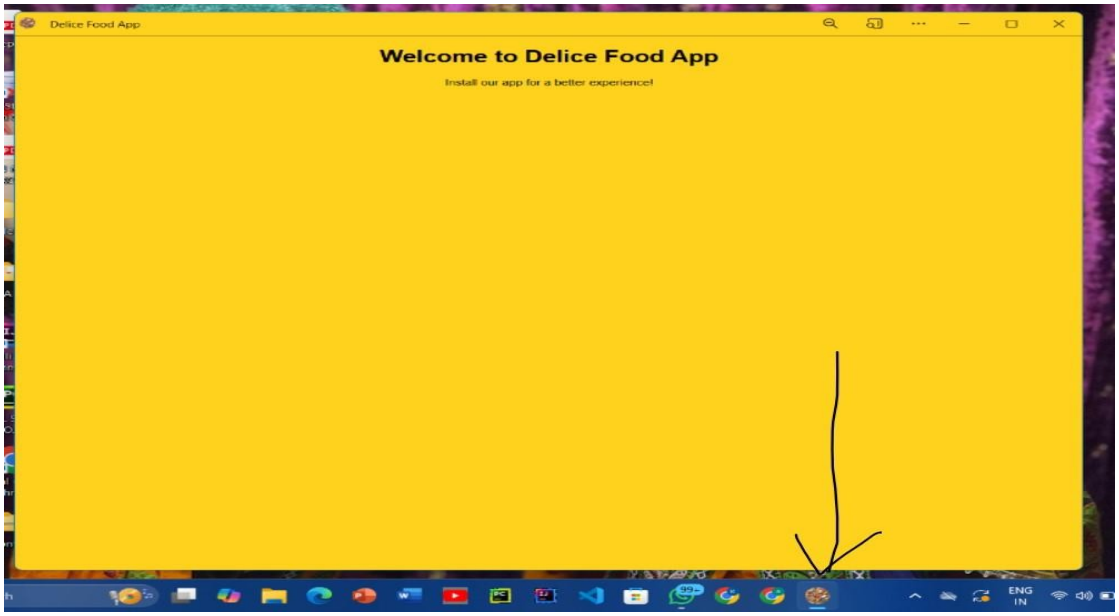
Google Dev

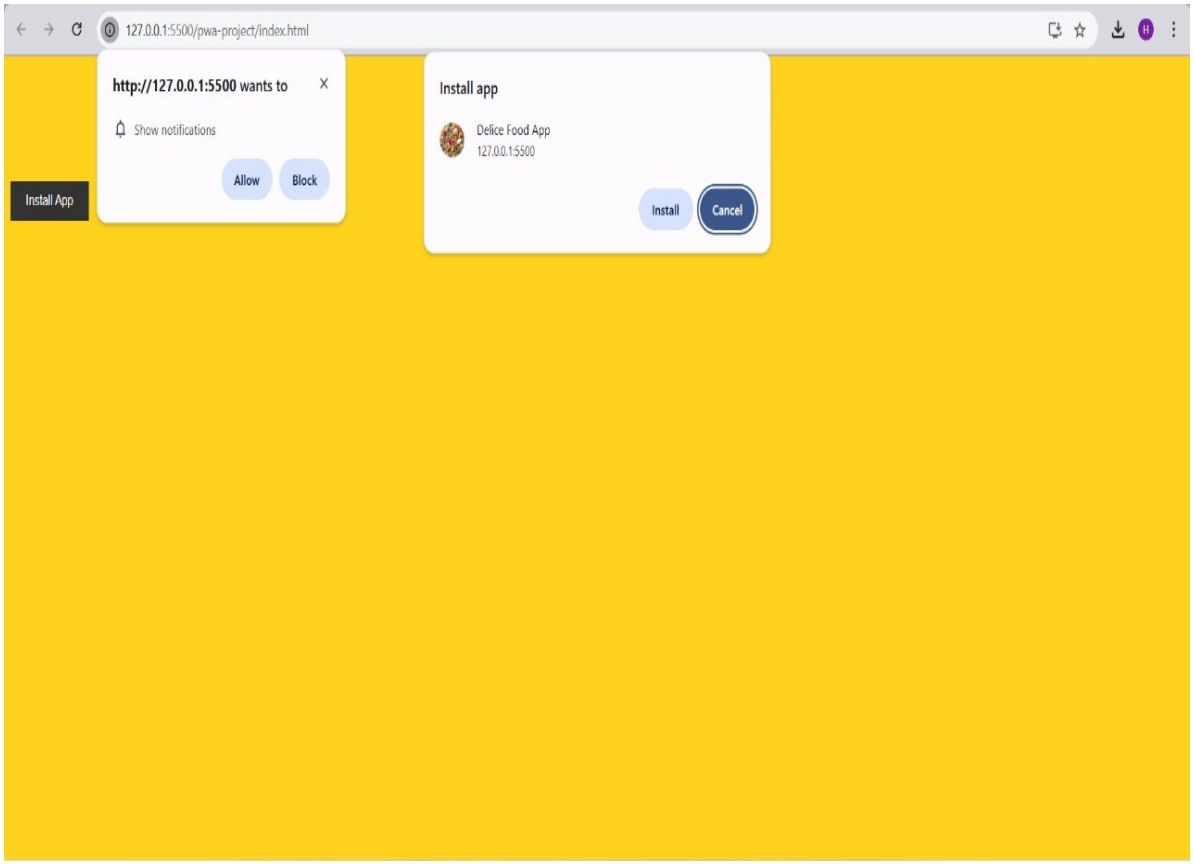


❖ **Output:-**


App installed
Publisher: localhost:3000
Delicia Food App has been installed as an app on your device and will safely run in its own window. Launch it from the Start menu, Windows taskbar or your Desktop.
Allow this app to
Pin to taskbar
Pin to Start
Create Desktop shortcut
Auto-start on device login
Allow
Don't allow


Delicia
Food App


Delice Food App
Welcome to Delice Food App
Install our app for a better experience!


127.0.0.1:5500/pwa-project/index.html
http://127.0.0.1:5500 wants to
Show notifications
Allow
Block
Install App
Install app
Delice Food App
127.0.0.1:5500
Install
Cancel

Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	08
Experiment Title.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA
Roll No.	
Name	
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

MPL Experiment 8 (PWA)

Name: Devansh Wadhwani - 64
Kunal Punjabi - 43
Manav Punjabi - 44
Hitesh Rohra - 46 **Class:** D15A

Aim: To code and register a service worker, and complete the install and activation process for a new service worker for the Food App 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 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.

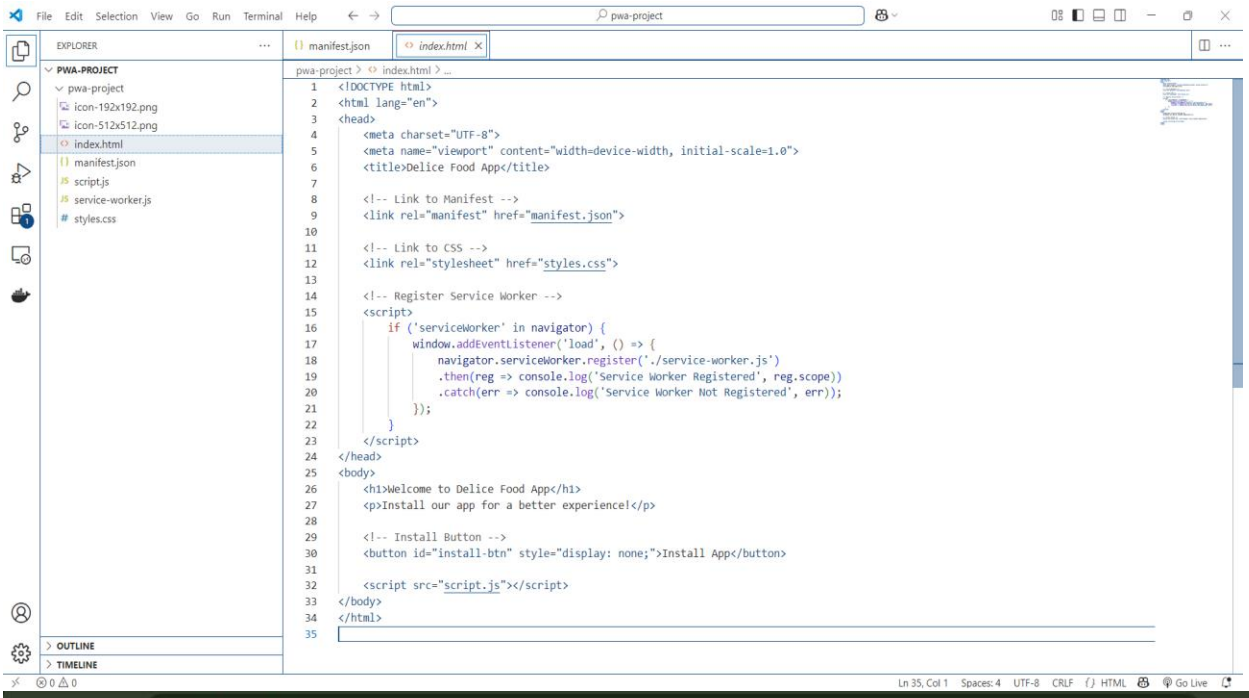
- You can’t work it on 80 Port
  Service Worker just can work on HTTPS protocol. But you can work on localhost during development.

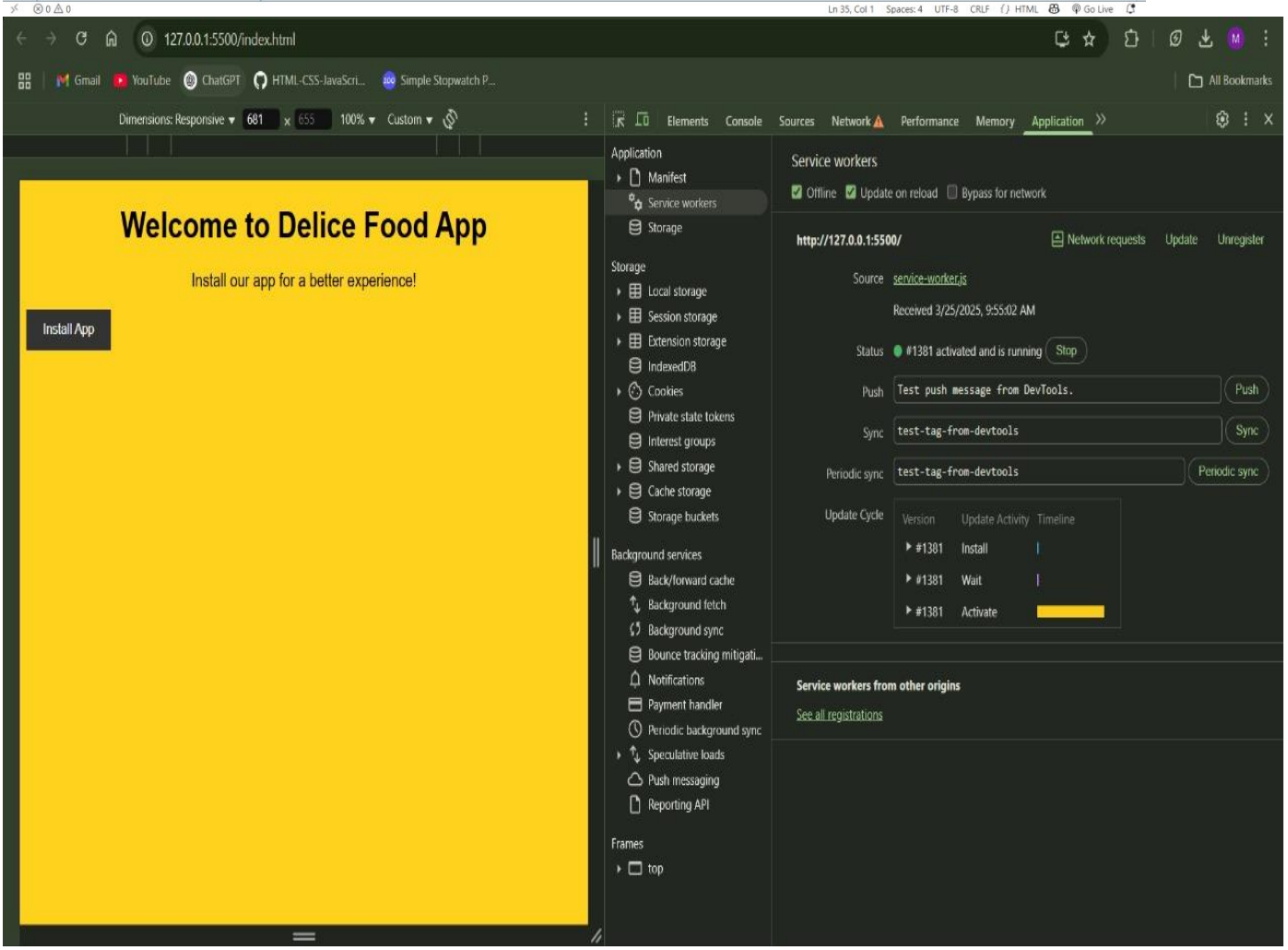
Codes:

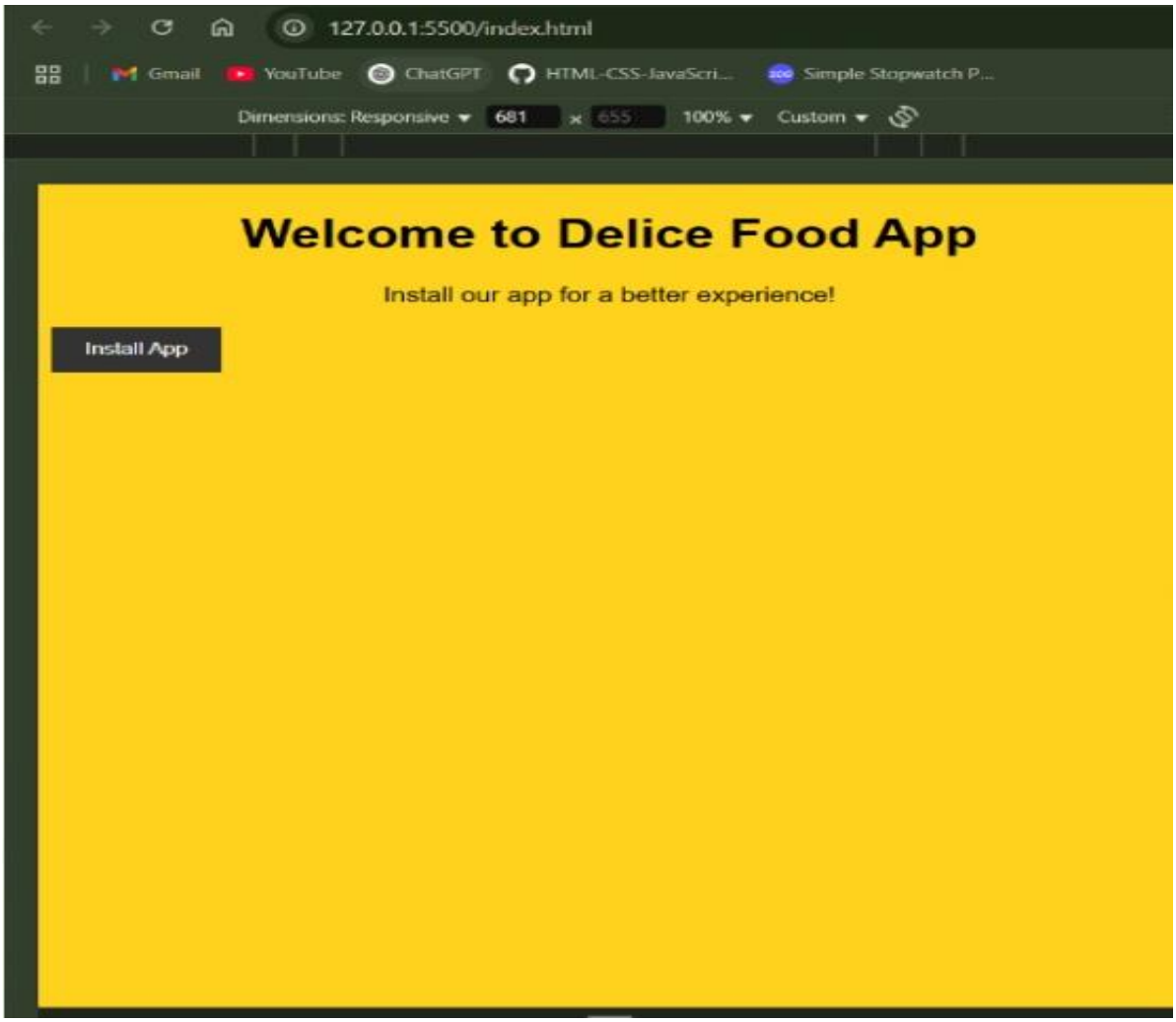
```
//Serviceworker.js


const CACHE_NAME = 'delice-cache-v1';
const urlsToCache = [
    './',
    './index.html',
    './styles.css',
    './script.js',
    './icon-192x192.png',
    './icon-512x512.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 (Old cache cleanup)
self.addEventListener('activate', event => {
    event.waitUntil(
        caches.keys().then(cacheNames => {
            return Promise.all(
                cacheNames.filter(cache => cache !==
CACHE_NAME).map(cache => caches.delete(cache))
            );
        })
    );
});

// Fetch Requests with Network Fallback
self.addEventListener('fetch', event => {
    event.respondWith(
        caches.match(event.request).then(response => {
            return response || fetch(event.request)
                .then(networkResponse => {
                    return caches.open(CACHE_NAME).then(cache => {
                        cache.put(event.request,
networkResponse.clone());
```

```
                        return networkResponse;
                    });
                });
        }).catch(() => {
            return caches.match('./index.html'); // Offline fallback
        })
    );
});
```

Output:


File Edit Selection View Go Run Terminal Help
pwa-project
EXPLORER
PWA-PROJECT
pwa-project
icon-192x192.png
icon-512x512.png
index.html
manifest.json
script.js
service-worker.js
styles.css
manifest.json
index.html
pwa-project > index.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Delice Food App</title>
7
8 <!-- Link to Manifest -->
9 <link rel="manifest" href="manifest.json">
10
11 <!-- Link to CSS -->
12 <link rel="stylesheet" href="styles.css">
13
14 <!-- Register Service Worker -->
15 <script>
16 if ('serviceWorker' in navigator) {
17 window.addEventListener('load', () => {
18 navigator.serviceWorker.register('./service-worker.js')
19 .then(reg => console.log('Service Worker Registered', reg.scope))
20 .catch(err => console.log('Service Worker Not Registered', err));
21 });
22 }
23 </script>
24 </head>
25 <body>
26 <h1>Welcome to Delice Food App</h1>
27 Install our app for a better experience!
28
29 <!-- Install Button -->
30 <button id="install-btn" style="display: none;">Install App</button>
31
32 <script src="script.js"></script>
33 </body>
34 </html>
35
OUTLINE
TIMELINE
Ln 35, Col 1 Spaces: 4 UTF-8 CRLF HTML Go Live


127.0.0.1:5500/index.html
Gmail YouTube ChatGPT HTML-CSS-JavaScri... Simple Stopwatch P...
All Bookmarks
Dimensions: Responsive 681 x 655 100% Custom
Welcome to Delice Food App
Install our app for a better experience!
Install App
Elements Console Sources Network Performance Memory Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigati...
Notifications
Payment handler
Periodic background sync
Speculative loads
Push messaging
Reporting API
Frames
top
Service workers
Offline Update on reload Bypass for network
http://127.0.0.1:5500/
Network requests Update Unregister
Source service-worker.js
Received 3/25/2025, 9:55:02 AM
Status #1381 activated and is running Stop
Push Test push message from DevTools. Push
Sync test-tag-from-devtools Sync
Periodic sync test-tag-from-devtools Periodic sync
Update Cycle
Version Update Activity Timeline
#1381 Install
#1381 Wait
#1381 Activate
Service workers from other origins
See all registrations


127.0.0.1:5500/index.html
Gmail
YouTube
ChatGPT
HTML-CSS-JavaScri...
Simple Stopwatch P...
Dimensions: Responsive
681
655
100%
Custom
Welcome to Delice Food App
Install our app for a better experience!
Install App

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.	44
Name	Manav Punjabi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

MPL Experiment 9 (PWA)

Name: Devansh Wadhwani - 64

Kunal Punjabi - 43

Manav Punjabi – 44

Hitesh Rohra - 46 **Class:** D15A

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.

Push Event

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

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

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

Code:

Serviceworker.js

```
const CACHE_NAME = 'delice-cache-v1';
const urlsToCache = [
    './',
    './index.html',
    './styles.css',
    './script.js',
    './icon-192x192.png',
    './icon-512x512.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 (Old cache cleanup)
self.addEventListener('activate', event => {
    event.waitUntil(
        caches.keys().then(cacheNames => {
            return Promise.all(
                cacheNames.filter(cache => cache !== CACHE_NAME).map(cache =>
caches.delete(cache))
            );
        })
    );
});

// Fetch Requests with Network Fallback
self.addEventListener('fetch', event => {
    event.respondWith(
        caches.match(event.request).then(response => {
            return response || fetch(event.request)
                .then(networkResponse => {
                    return caches.open(CACHE_NAME).then(cache => {
                        cache.put(event.request, networkResponse.clone());
                        return networkResponse;
                    });
                });
        }).catch(() => {
            return caches.match('./index.html'); // Offline fallback
        })
    );
```

});

Push Notification Code:

1. Push Notification code:

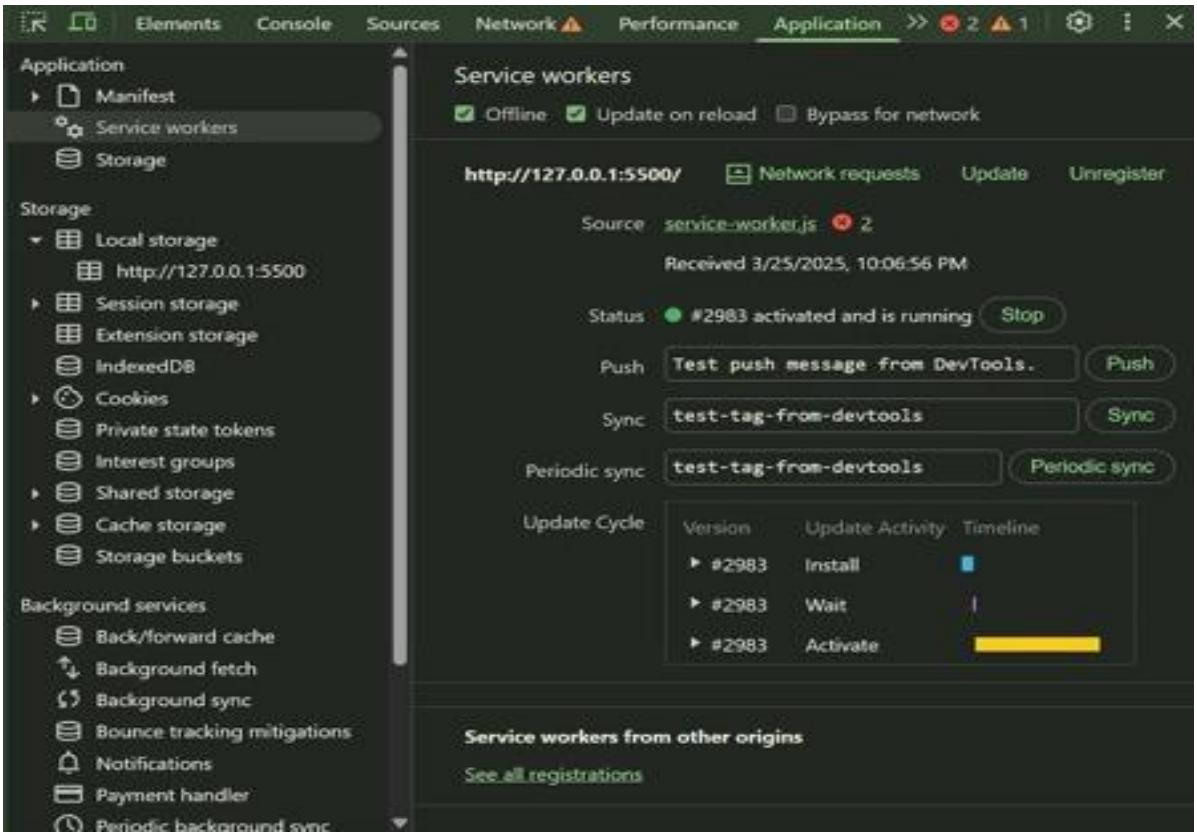
```
// Push Notification Handling
self.addEventListener("push", (event) => {
    console.log("Push notification received:", event);

    let data = { title: "New Notification", body: "Hello, this is a default message!" };

    if (event.data) {
        try {
            data = event.data.json();
        } catch (error) {
            console.warn("Push message is not JSON, using text instead.");
            data.body = event.data.text();
        }
    }

    const options = {
        body: data.body || "You have a new update!",
        icon: "./assets/images/logo_tourly_192.png",
        badge: "./assets/images/logo_tourly_192.png",
        vibrate: [200, 100, 200],
    };

    event.waitUntil(
        self.registration.showNotification(data.title || "Hello Sannidhi", options)
    );
});

// Click Event for Push Notifications
self.addEventListener("notificationclick", (event) => {
    event.notification.close();
    event.waitUntil(
        clients.openWindow("http://localhost:5500/")
    );
});
```

Fetch Code :

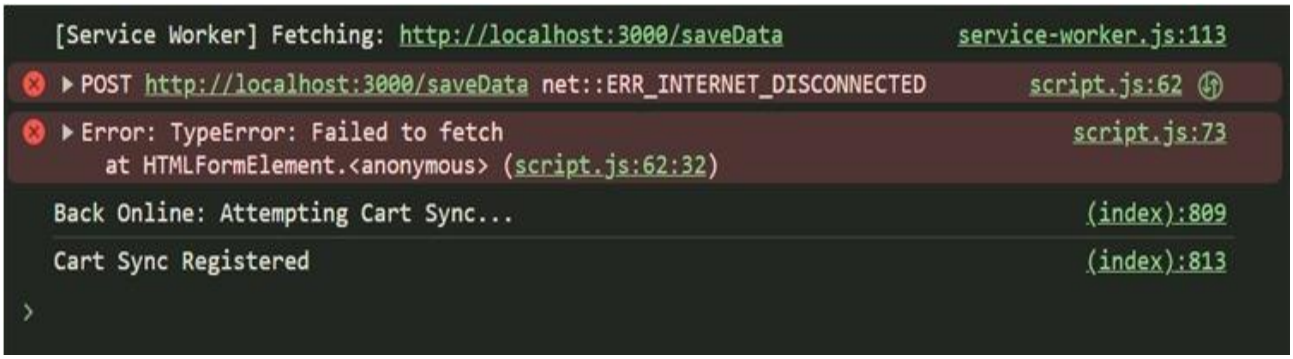
```
self.addEventListener("fetch", (event) => {
   console.log("[Service Worker] Fetching:", event.request.url);

   if (event.request.method !== "GET") return; // Skip non-GET requests

   event.respondWith(
      caches.match(event.request).then((response) => {
         return response || fetch(event.request).then((networkResponse) => {
            return networkResponse;
         });
      }).catch(() => console.warn("[Service Worker] Network request failed:",
event.request.url))
   );
});

// Sync Event - Send Stored Data When Online
self.addEventListener("sync", (event) => {
   if (event.tag === SYNC_TAG) {
      console.log("[Service Worker] Sync event triggered:", SYNC_TAG);
      event.waitUntil(syncOfflineData());
   }
});

// Function to Sync Offline Data & Save to JSON
async function syncOfflineData() {
   const storedData = await getFromLocalStorage();
   if (!storedData.length) {
      console.log("[Service Worker] No offline data to sync.");
      return;
   }

   console.log("[Service Worker] Attempting to sync offline data...", storedData);

   try {
      const response = await fetch("http://localhost:3000/saveData", {
         method: "POST",
         headers: { "Content-Type": "application/json" },
         body: JSON.stringify(storedData)
      });

      if (response.ok) {
         console.log("[Service Worker] Successfully synced data:", storedData);
         await saveDataToJson(storedData); // Save data to JSON file
         clearLocalStorage(); // Clear after successful sync
      } else {
         console.warn("[Service Worker] Sync failed. Server response not OK.");
      }
   } catch (error) {
      console.error("[Service Worker] Sync failed:", error);
   }
}
// Function to Retrieve Data from Local Storage
function getFromLocalStorage() {
```

```
    return new Promise((resolve) => {
        const data = localStorage.getItem("offlineData");
        resolve(data ? JSON.parse(data) : []);
    });
}

// Function to Clear Local Storage after Sync
function clearLocalStorage() {
    localStorage.removeItem("offlineData");
    console.log("[Service Worker] Local storage cleared after sync.");
}

// Save Synced Data to a JSON File
async function saveDataToJson(data) {
    try {
        const response = await fetch(JSON_FILE_URL, {
            method: "PUT", // Use PUT or POST based on server support
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify(data)
        });

        if (response.ok) {
            console.log("[Service Worker] Data saved to JSON file:", JSON_FILE_URL);
        } else {
            console.error("[Service Worker] Failed to save data to JSON.");
        }
    } catch (error) {
        console.error("[Service Worker] Error saving to JSON:", error);
    }
}
```

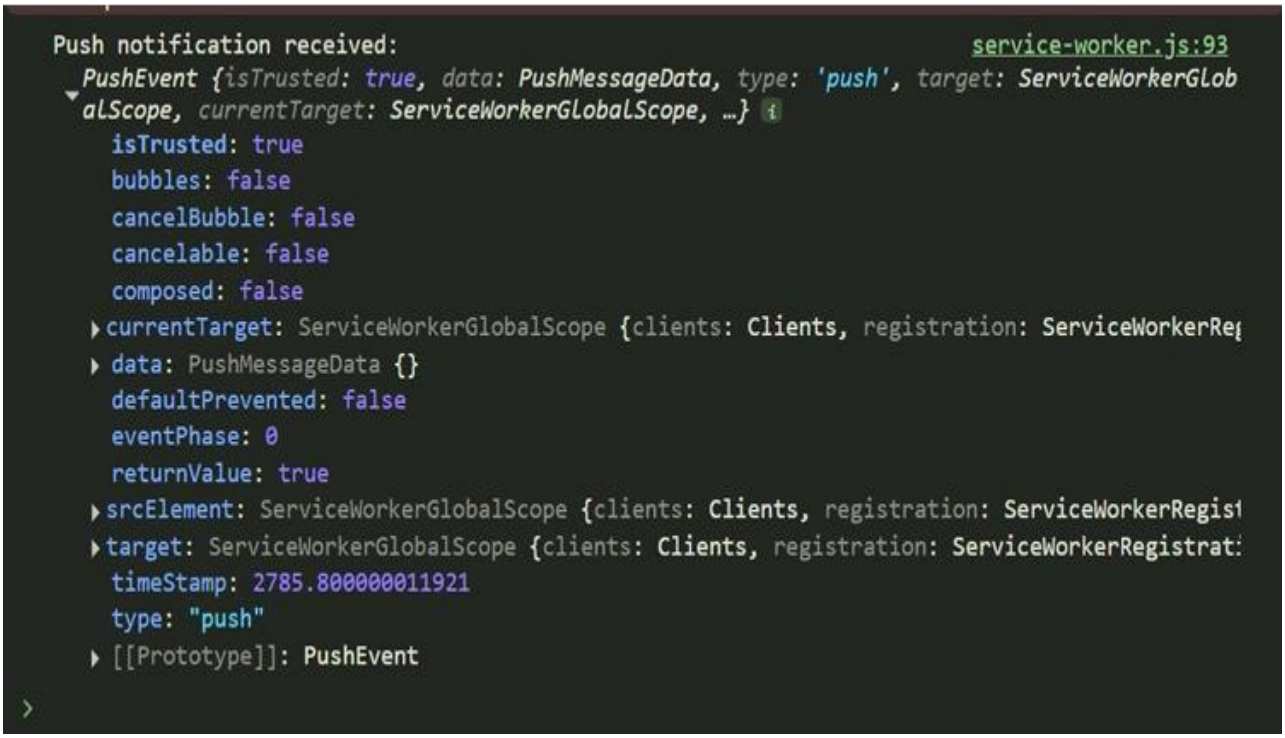
Output:

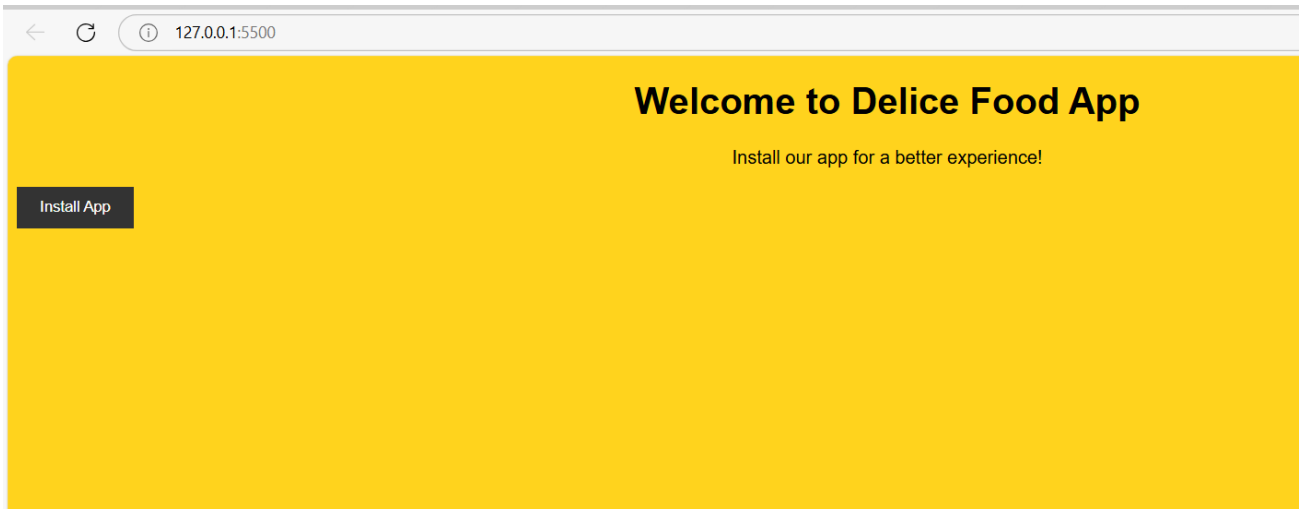

Elements
Console
Sources
Network
Performance
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
http://127.0.0.1:5500
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigations
Notifications
Payment handler
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Network requests
Update
Unregister
Source
service-worker.js
2
Received 3/25/2025, 10:06:56 PM
Status
#2983 activated and is running
Stop
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#2983
Install
#2983
Wait
#2983
Activate
Service workers from other origins
See all registrations

Sync Event



Push event




127.0.0.1:5500
Welcome to Delice Food App
Install our app for a better experience!
Install App

MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	44
Name	Manav Punjabi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

MPL Experiment 10 (PWA)

Class: D15A

Name:

Devansh Wadhwani – 64

Kunal Punjabi – 43

Manav Punjabi – 44

Hitesh Rohra - 46

Aim: To study and implement deployment of Food App PWA to GitHub Page.

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:

- Blogging with Jekyll
- Custom URL
- Automatic Page Generator

Reasons for favoring this over Firebase:

- Free to use
- Right out of github
- 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

- Very familiar interface if you are already using GitHub for your projects.
- Easy to set up. Just push your static website to the gh-pages branch and your website is ready.
- Supports Jekyll out of the box.
- 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

- The code of your website will be public, unless you pay for a private repository.
- Currently, there is no support for HTTPS for custom domains. It’s probably coming soon though.
- 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:

- 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.
- 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.
- 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:

- Realtime backend made easy
- 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

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

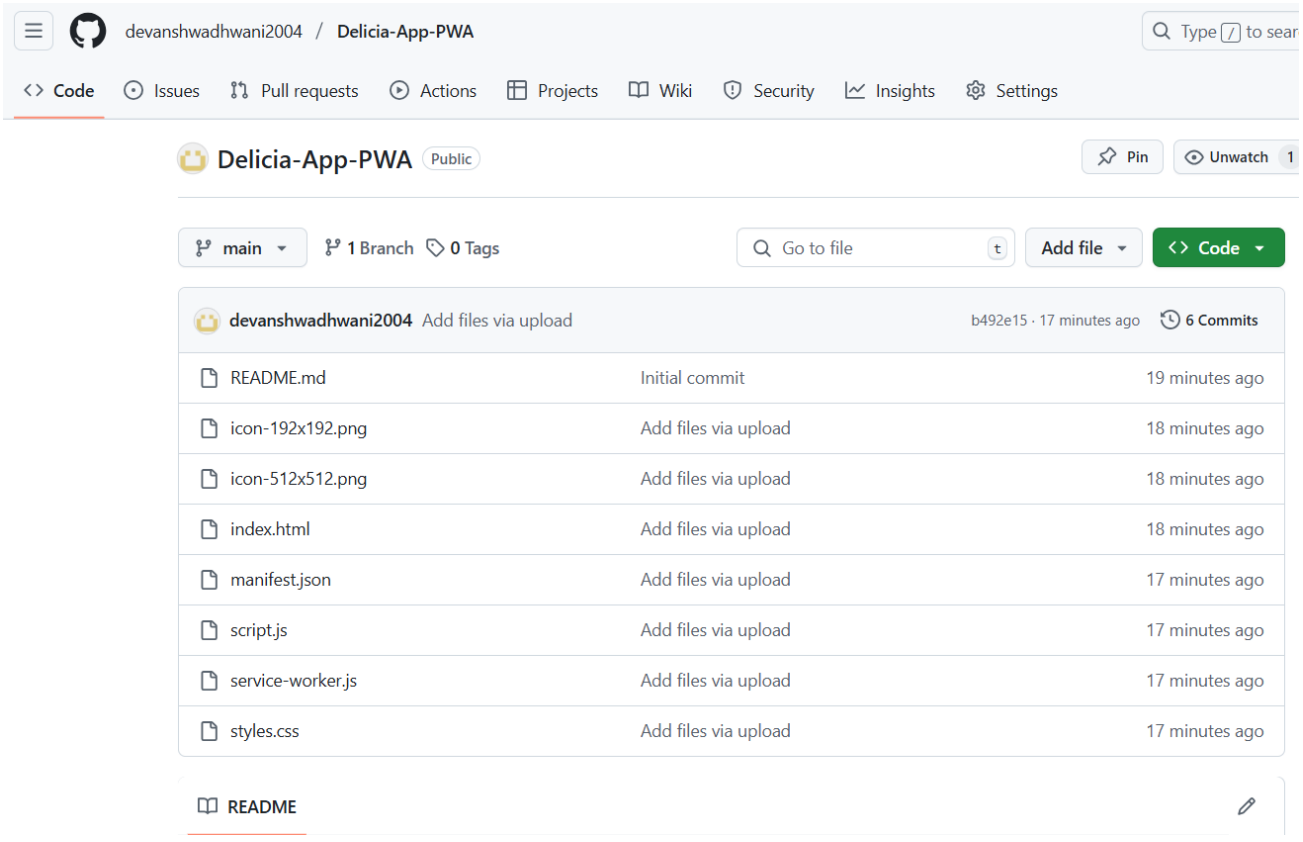
Cons

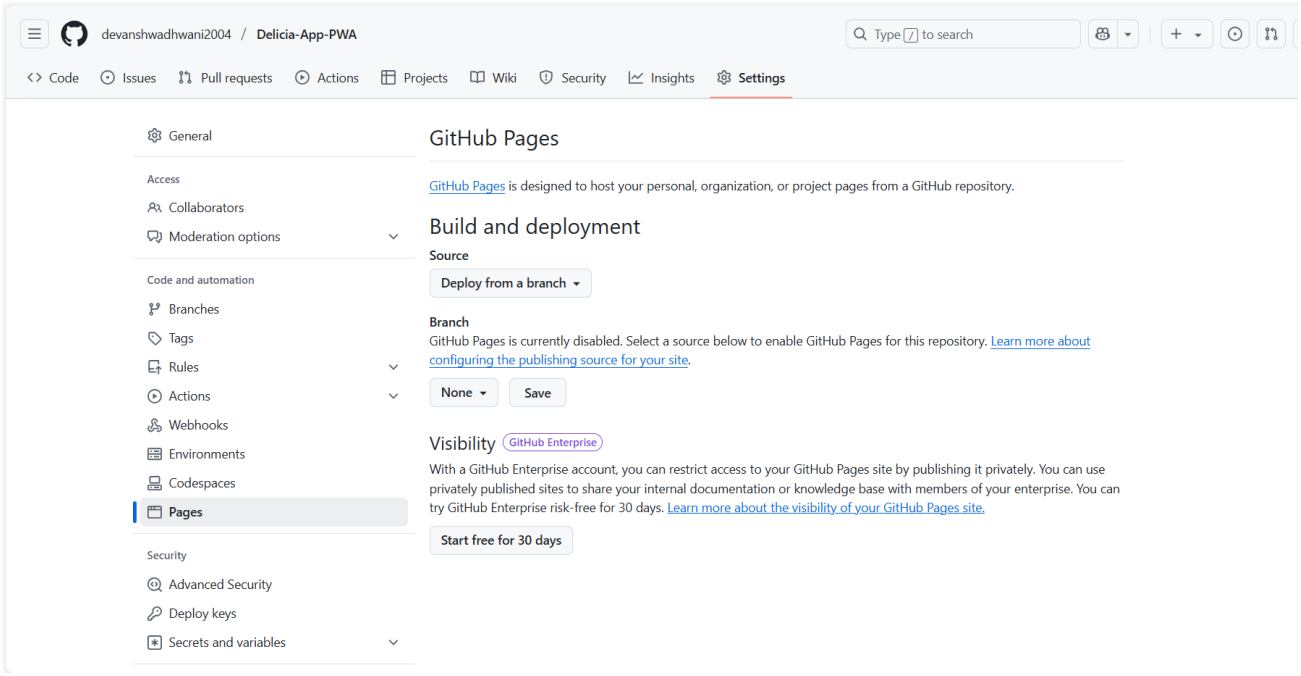
- Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
- Command-line interface only.
- No in-built support for any static site generator.

Link to our GitHub repository:
https://devanshwadhwani2004/Delicia-App-PWA/

Link to our Hosted website:
https://devanshwadhwani2004/Delicia-App-PWA/

Github Screenshot:




devanshwadhwani2004 / Delicia-App-PWA
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
Security
Advanced Security
Deploy keys
Secrets and variables
GitHub Pages
GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.
Build and deployment
Source
Deploy from a branch
Branch
GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. Learn more about configuring the publishing source for your site.
None
Save
Visibility
GitHub Enterprise
With a GitHub Enterprise account, you can restrict access to your GitHub Pages site by publishing it privately. You can use privately published sites to share your internal documentation or knowledge base with members of your enterprise. You can try GitHub Enterprise risk-free for 30 days. Learn more about the visibility of your GitHub Pages site.
Start free for 30 days

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	44
Name	Manav Punjabi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

MPL Experiment 11 (PWA)

Name: Devansh Wadhwani – 64

Kunal Punjabi – 43

Manav Punjabi – 44

Hitesh Rohra - 46 **Class:** D15A

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:

- **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.

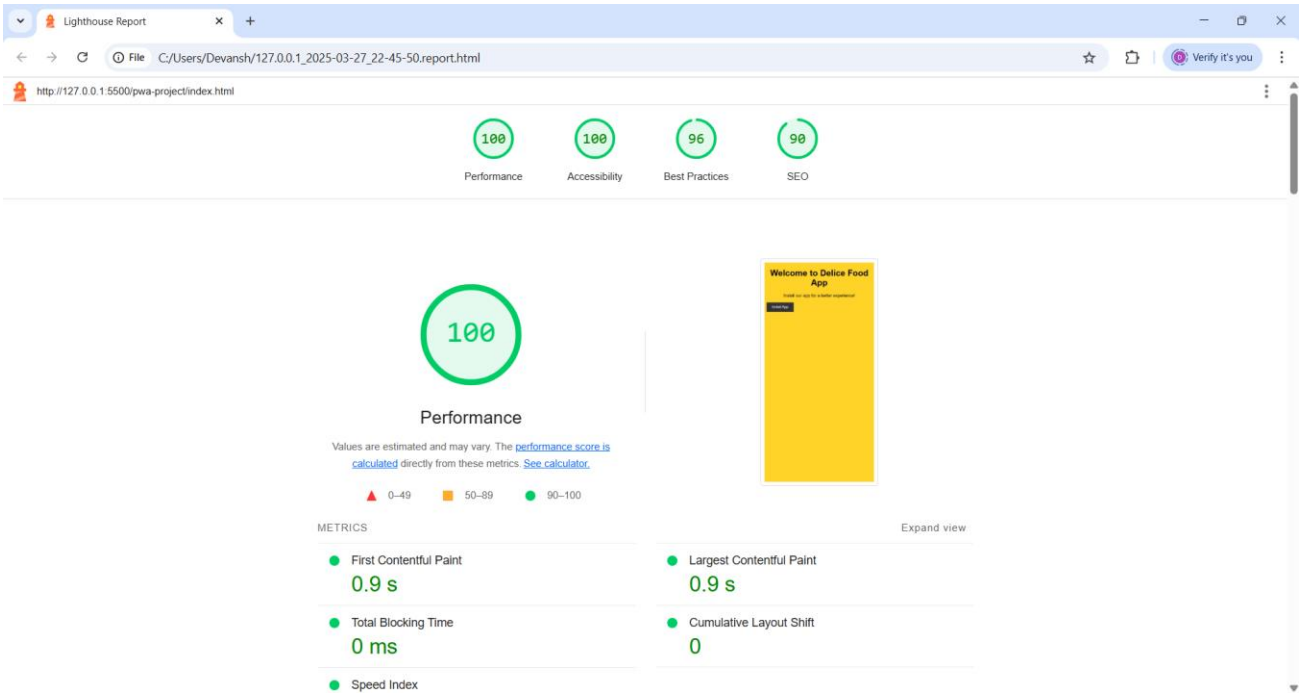
- **PWA Score (Mobile):**

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.

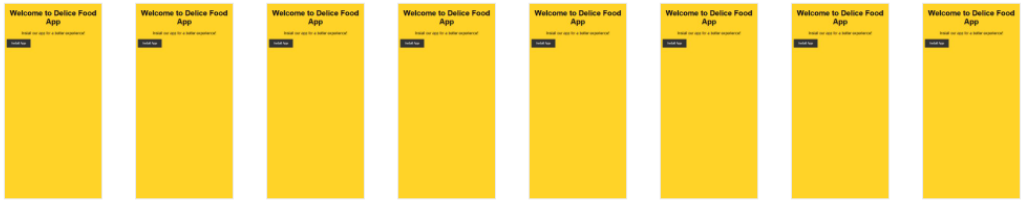
- **Accessibility:**
  As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics 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.
  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

0utput:


Lighthouse Report
File C:/Users/Devansh/127.0.0.1_2025-03-27_22-45-50.report.html
Verify it's you
http://127.0.0.1:5500/pwa-project/index.html
100
Performance
100
Accessibility
96
Best Practices
90
SEO
100
Performance
Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.
0–49
50–89
90–100
Welcome to Delice Food App
METRICS
Expand view
First Contentful Paint
0.9 s
Largest Contentful Paint
0.9 s
Total Blocking Time
0 ms
Cumulative Layout Shift
0
Speed Index

0.9 s

View Treemap



Show audits relevant to: All FCP LCP

DIAGNOSTICS

- Eliminate render-blocking resources — Potential savings of 290 ms
- Page prevented back/forward cache restoration — 1 failure reason
- Avoid chaining critical requests — 2 chains found
- Largest Contentful Paint element — 910 ms

More information about the performance of your application. These numbers don't directly affect the Performance score.

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so manual testing is also encouraged.

ADDITIONAL ITEMS TO MANUALLY CHECK (10) Hide

- Interactive controls are keyboard focusable
- Interactive elements indicate their purpose and state
- The page has a logical tab order
- Visual order on the page follows DOM order
- User focus is not accidentally trapped in a region
- The user's focus is directed to new content added to the page

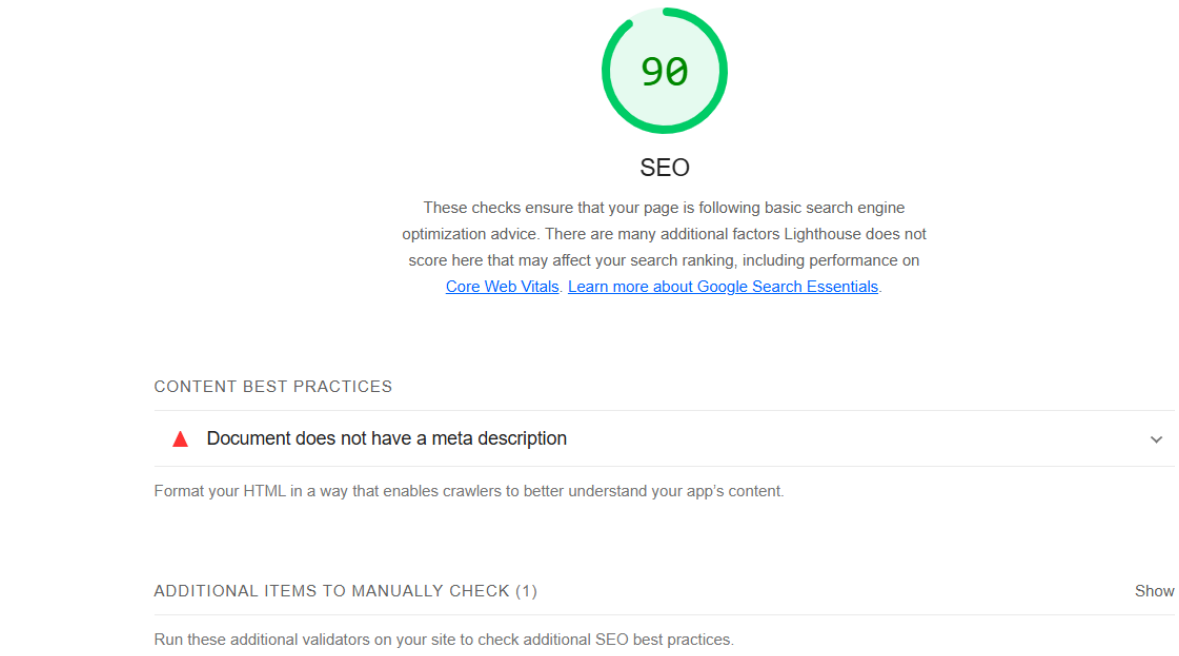
Best Practices

GENERAL

- Browser errors were logged to the console

TRUST AND SAFETY

- Ensure CSP is effective against XSS attacks
- Use a strong HSTS policy
- Ensure proper origin isolation with COOP
- Mitigate clickjacking with XFO or CSP



Conclusion:

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