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

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Name of the Course : MAD & PWA Lab

Course Code : ITL604

Project Title: Behance **Roll No.** 45

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

Faculty Incharge : Mrs. Kajal Joseph.

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

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

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: Behance **Roll No.** 45

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: Behance **Roll No.** 45

Index

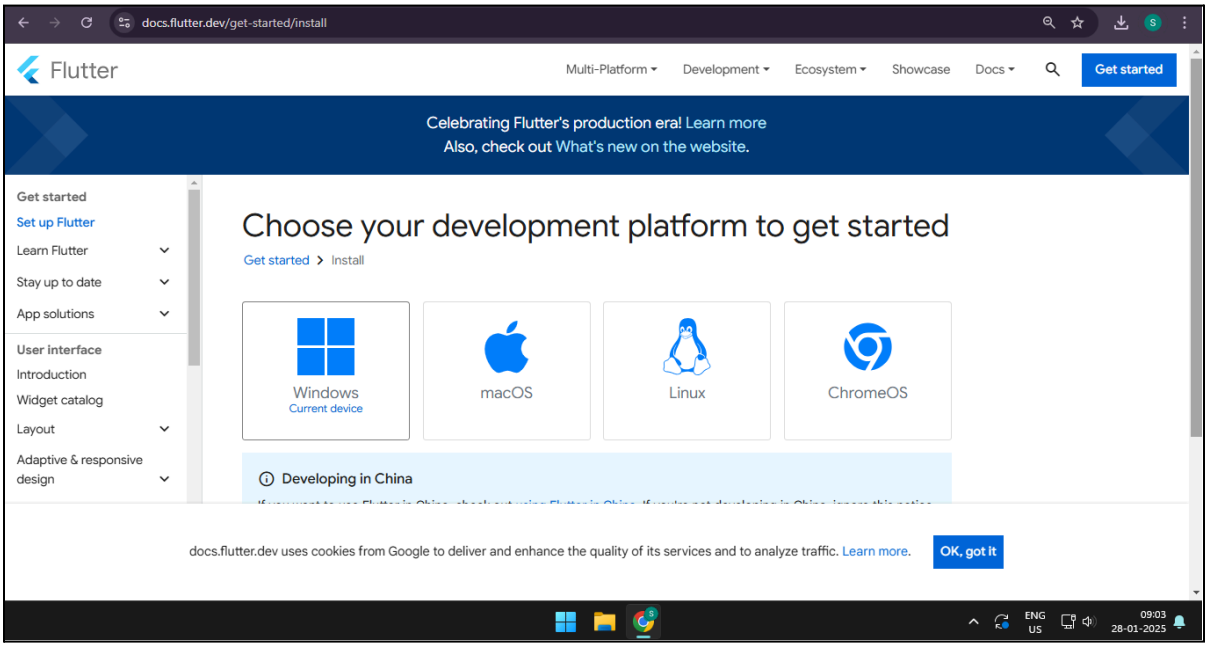
Sr. No	Experiment Title	LO	DOP	DOS	Grade
1.	To install and configure the Flutter Environment	LO1			
2.	To design Flutter UI by including common widgets.	LO2			
3.	To include icons, images, fonts in Flutter app	LO2			
4.	To create an interactive Form using form widget	LO2			
5.	To apply navigation, routing 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

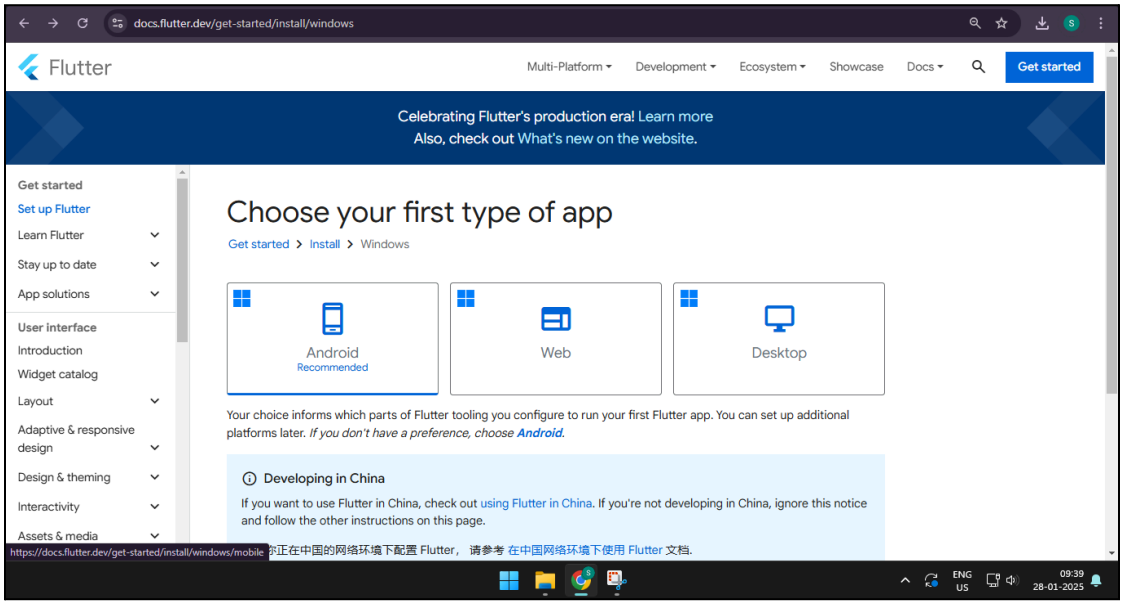
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	45
Name	Shravani Rasam
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

Project Title: Behance **Roll No.** 45

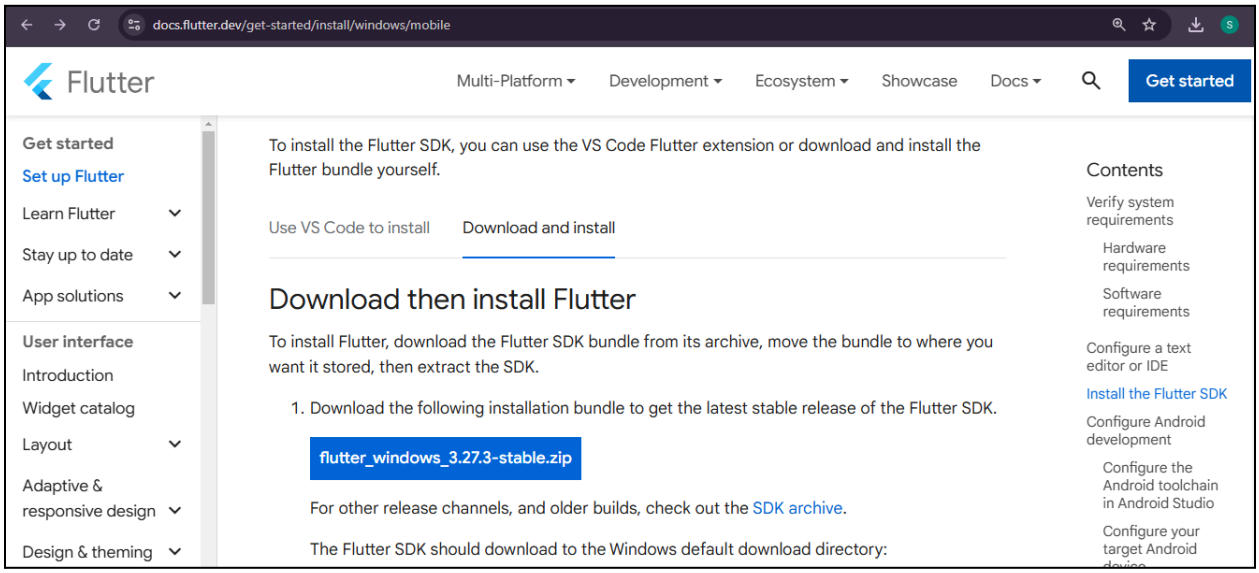
Step 1: Go to the official Flutter website: https://docs.flutter.dev/get-started/install



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

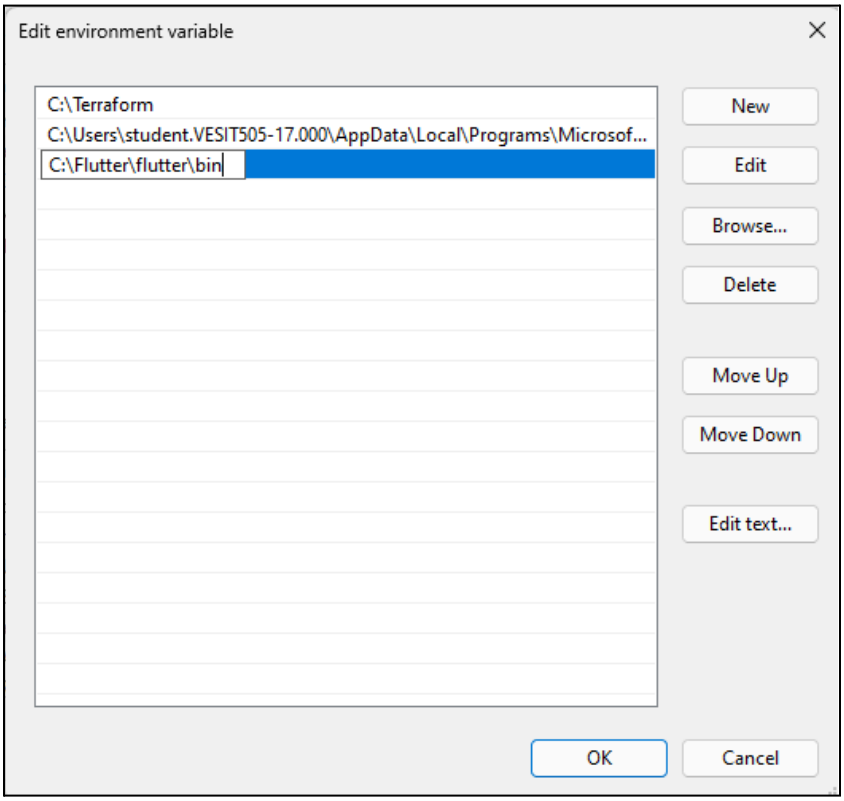


Step 3: For Windows, download the stable release (a .zip file).

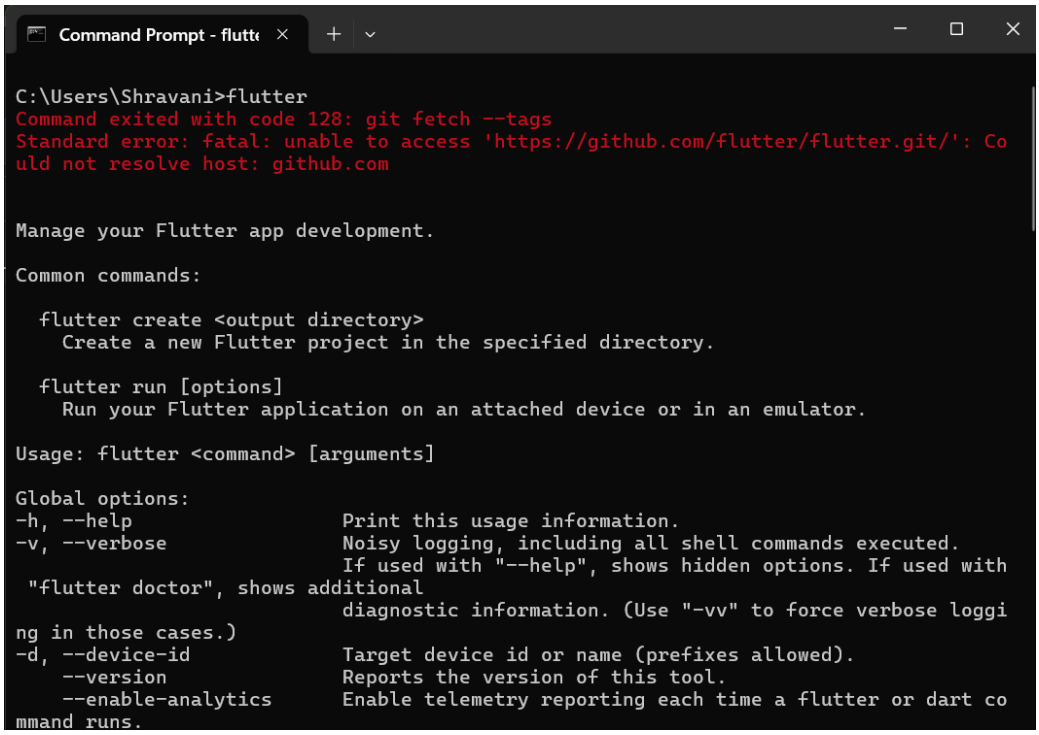


Step 5 :- Add Flutter to System PATH

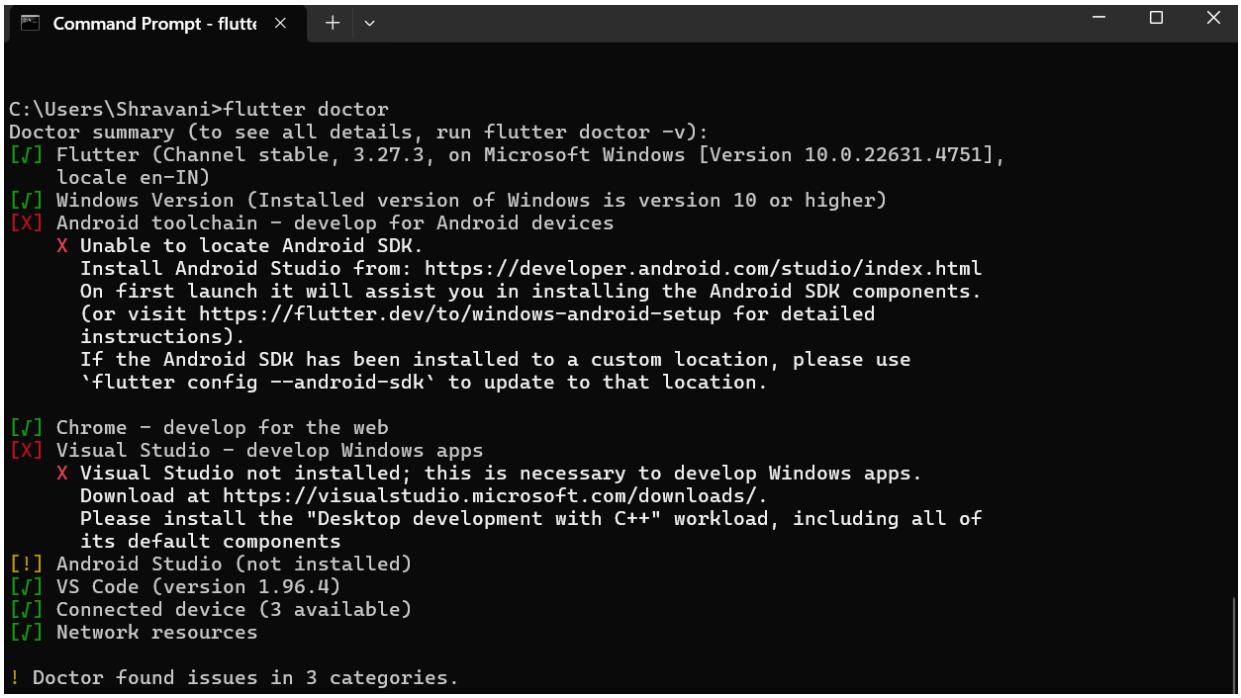
Right-click on the Start Menu > System > Advanced system settings > Environment Variables.
Under System Variables, find Path and click Edit.
Add the full path to the flutter/bin directory (e.g., C:\flutter\bin).



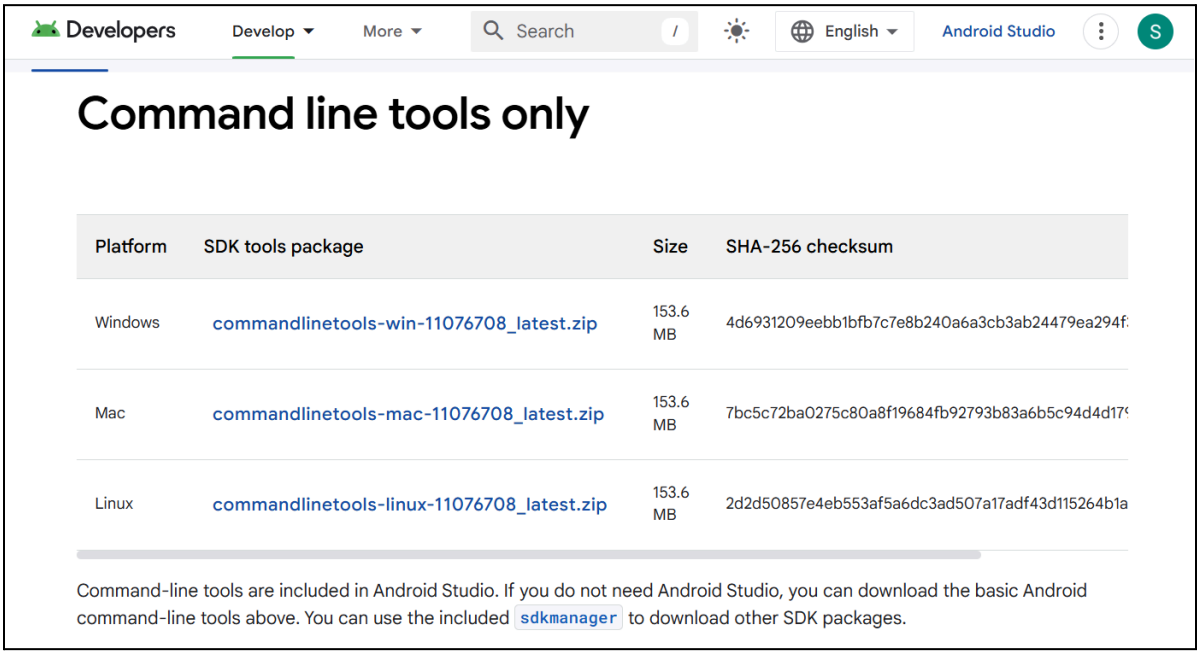
Step 6 : - Now, run the $ flutter command in command prompt



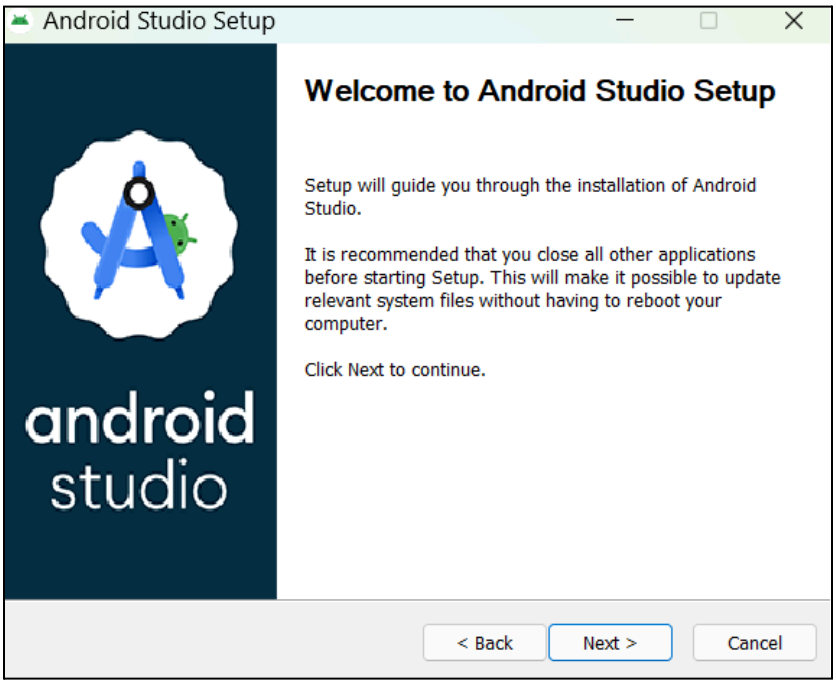
Step 7:- Run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation



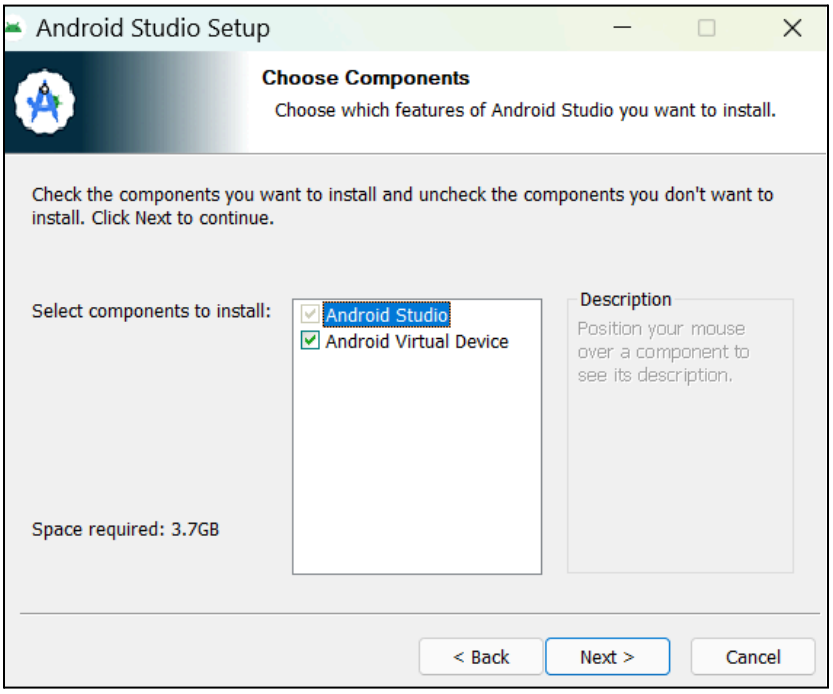
Step 8 : - Go to Android Studio and download the installer



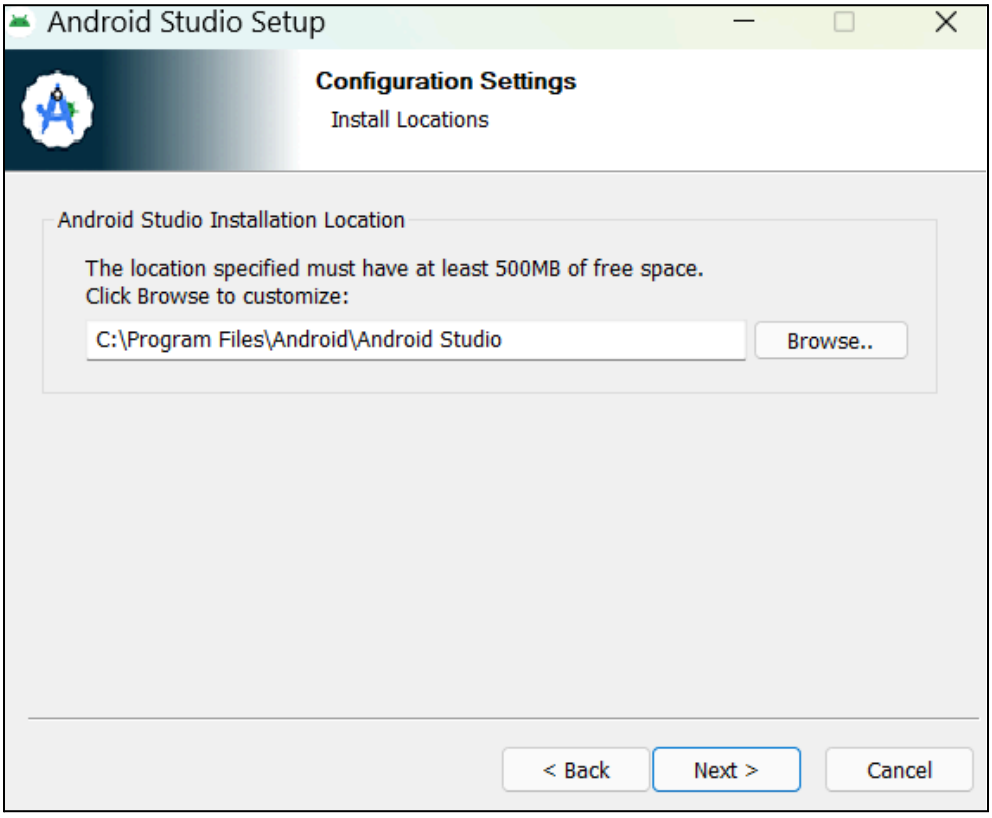
Step 8.1: - When the download is complete, open the .exe file and run it. You will get the following dialog box



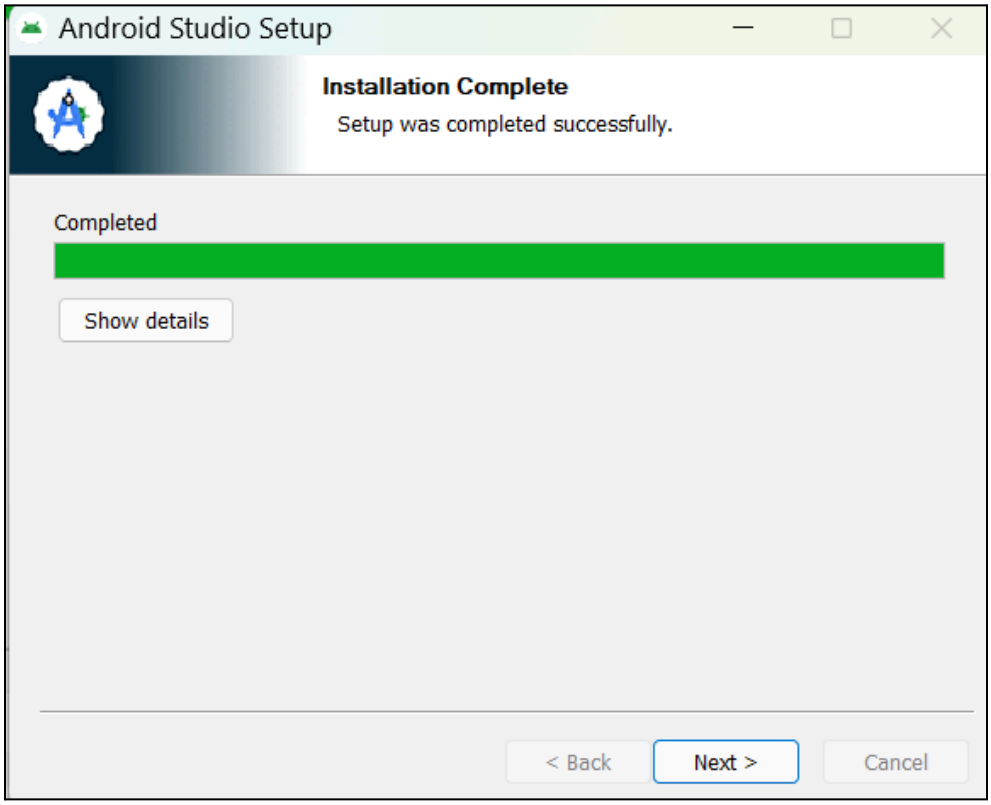
Step 8.2: - Select all the Checkboxes and Click on ‘Next’ Button.

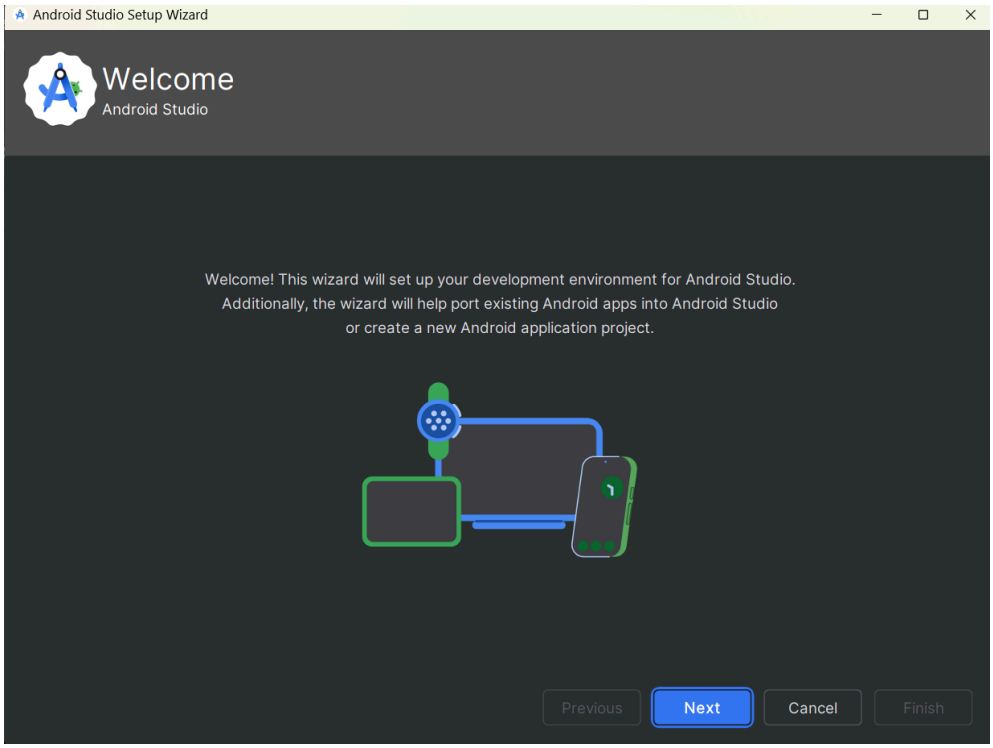


Step 8.3: - Change the destination as per your convenience and click on ‘Next’ Button.

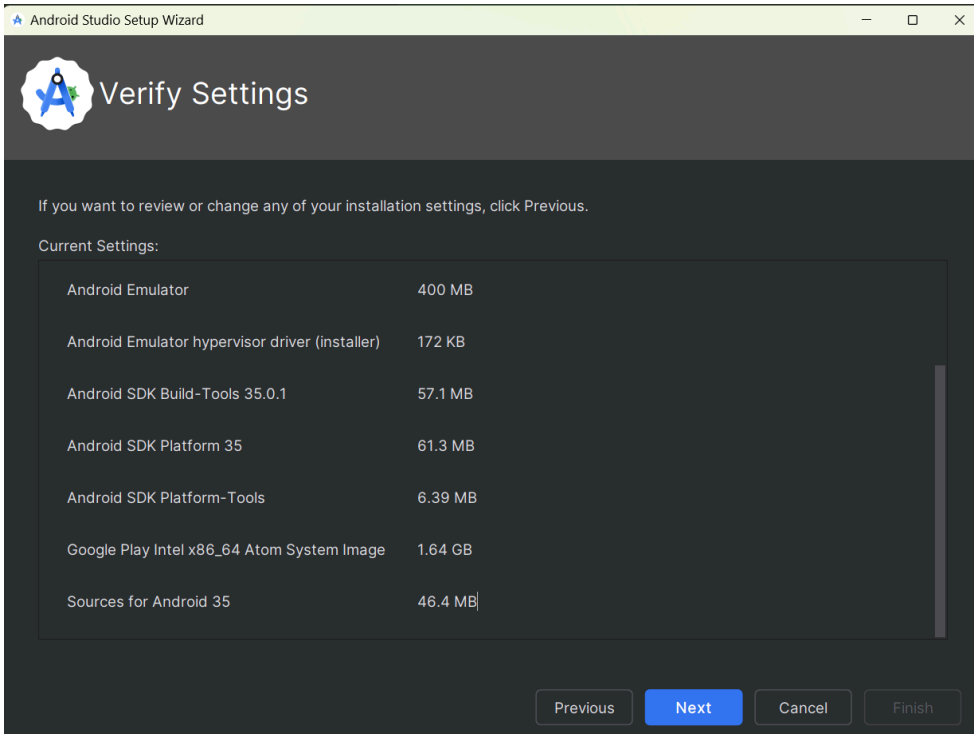


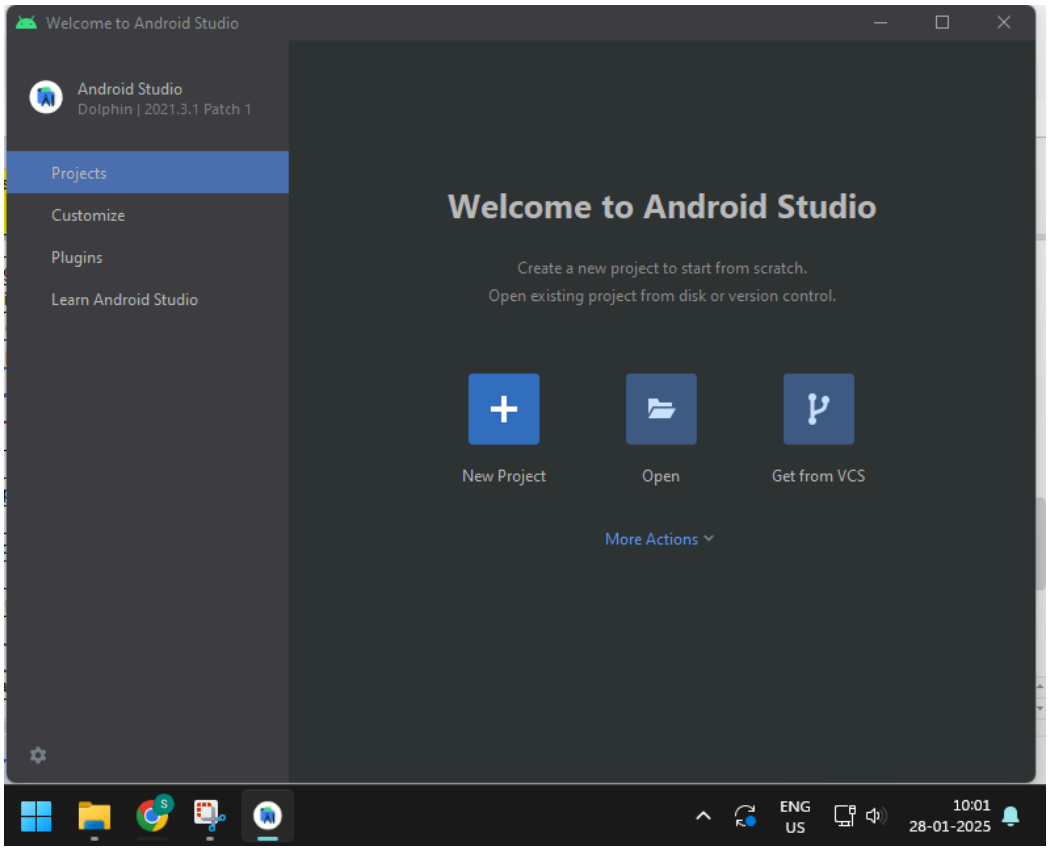
Step 8.4: - Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.



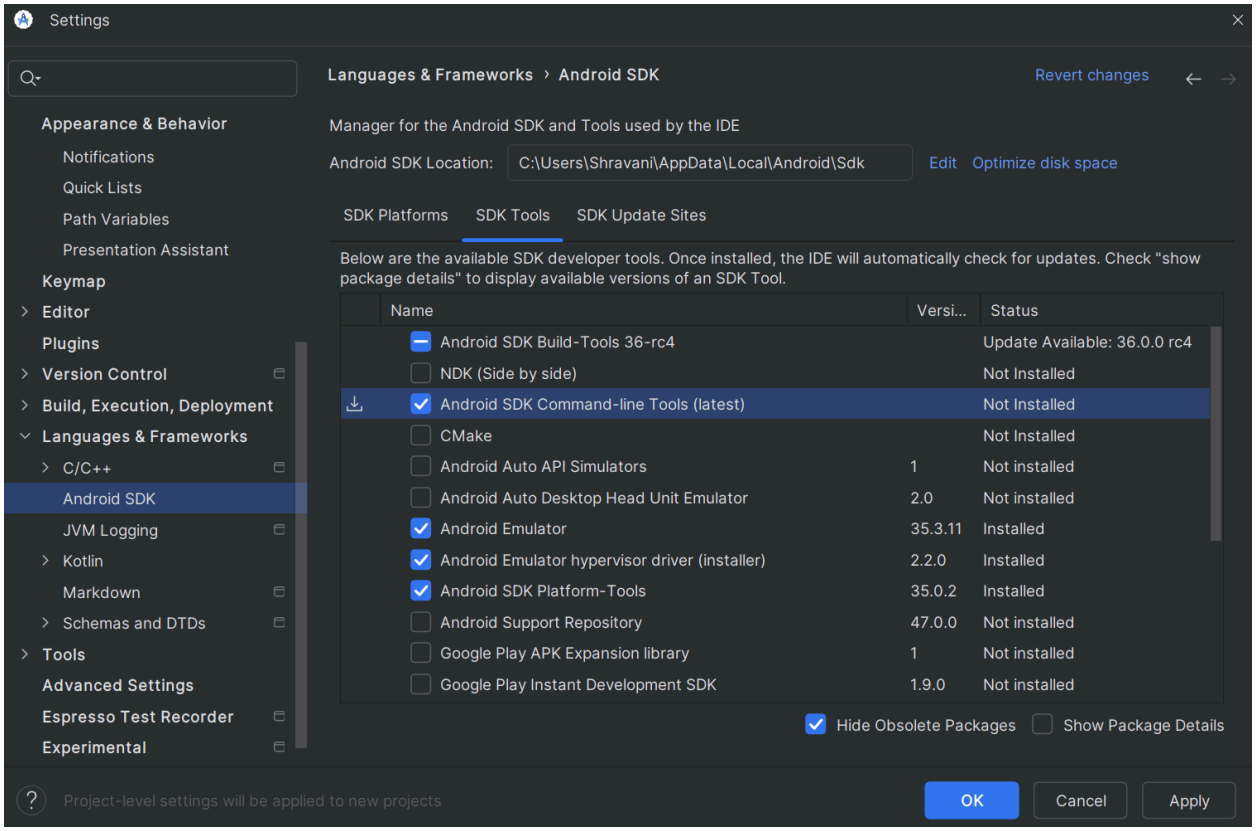


Project Title: Behance **Roll No.** 45





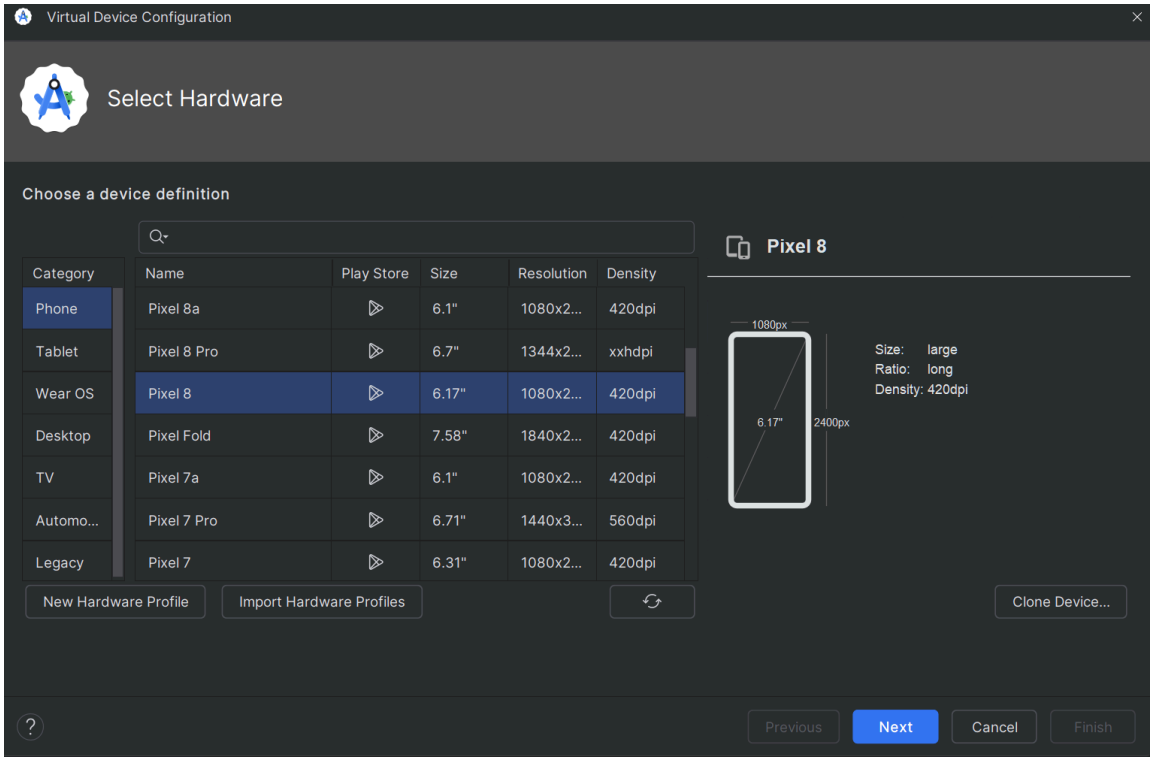
Step 8.5: - Go to Preferences > Appearance & Behavior > System Settings > Android SDK. Select the SDK Tools tab and check Android SDK Command-line Tools and Install it.



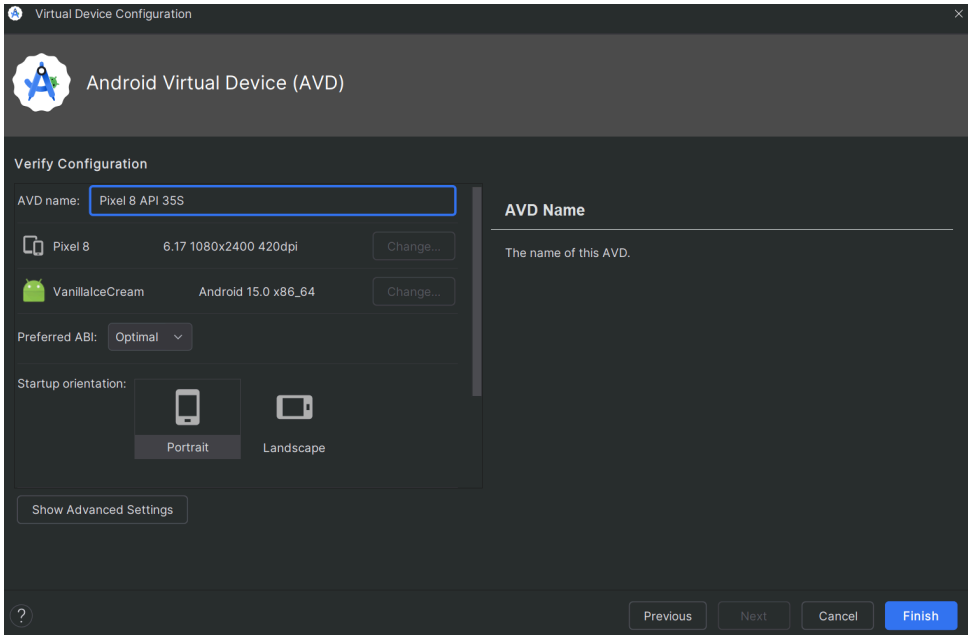
Step 9: - Open a terminal and run the following command

```
C:\Users\Shravani>flutter doctor --android-licenses
                                                                    Warning:

Warning: Additionally, the fallback loader failed to parse the XML.
Warning: Errors during XML parse:       ] 62% Fetch remote repository...
Warning: Additionally, the fallback loader failed to parse the XML.
[======================================] 100% Computing updates...
6 of 7 SDK package licenses not accepted.
Review licenses that have not been accepted (y/N)? y

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

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

1. Introduction

1.1 The Google TV Add-on for the Android Software Development Kit (referred to in this Li
g the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you su
t forms a legally binding contract between you and Google in relation to your use of the
```

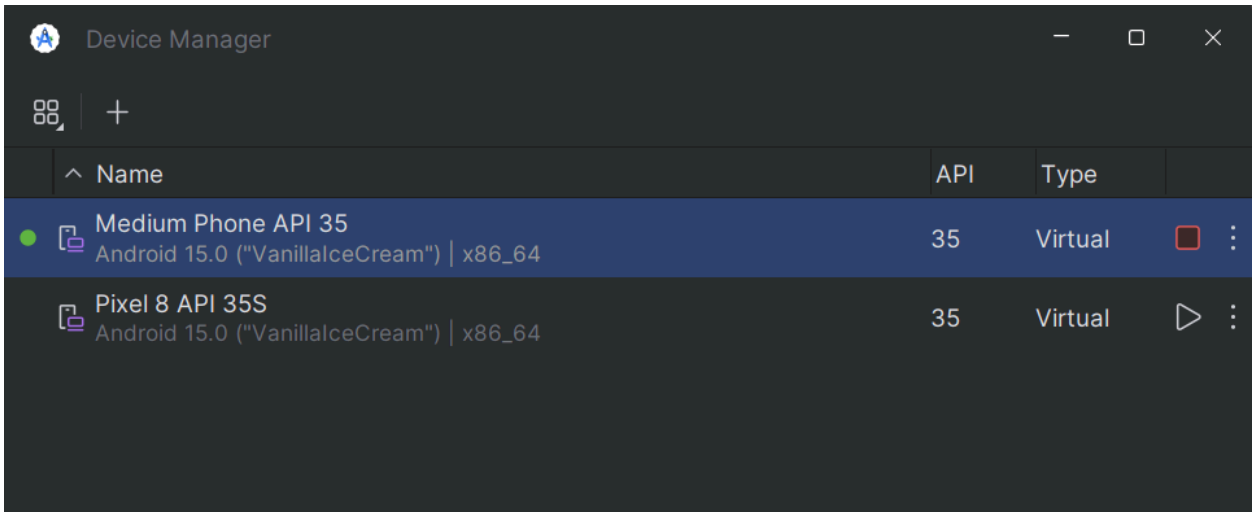
Step 10: - Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application

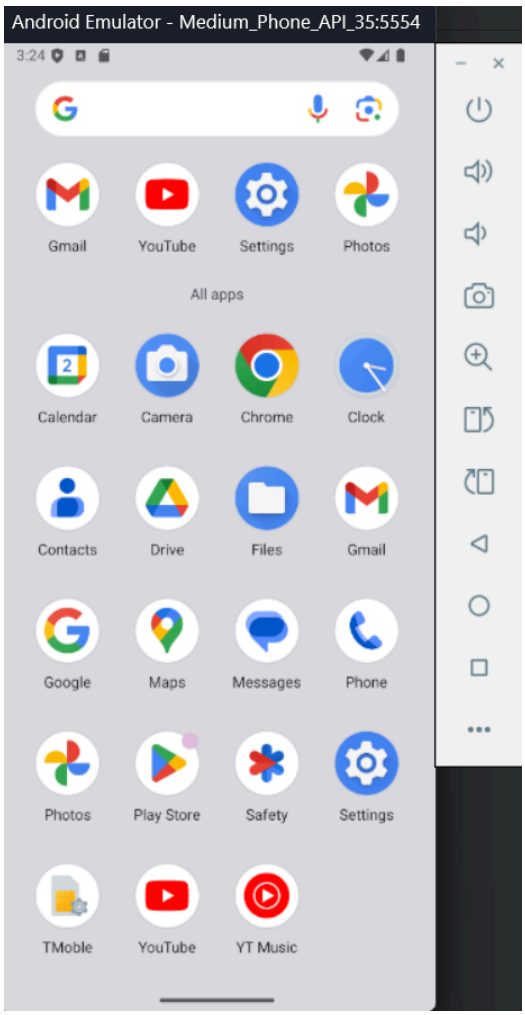


Step 10.1: - Open Android Studio and go to Tools > AVD Manager. Create a new virtual device

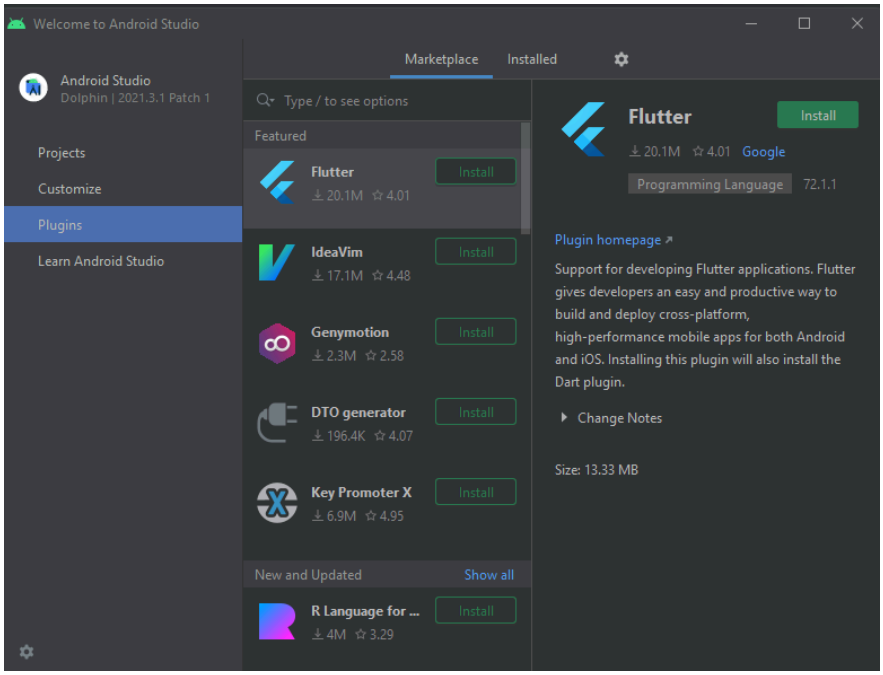


Step 10.2: - Click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen

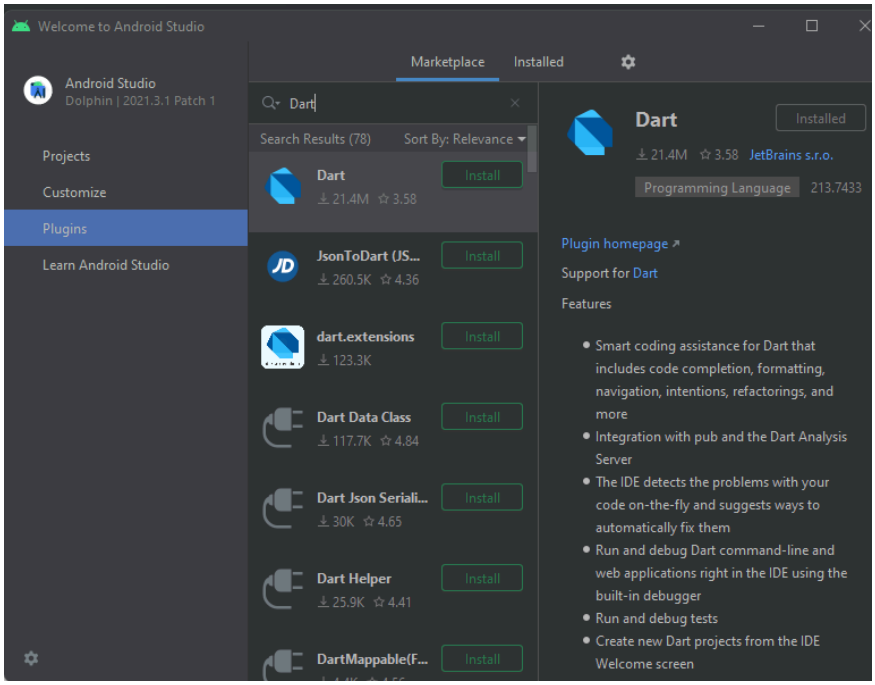




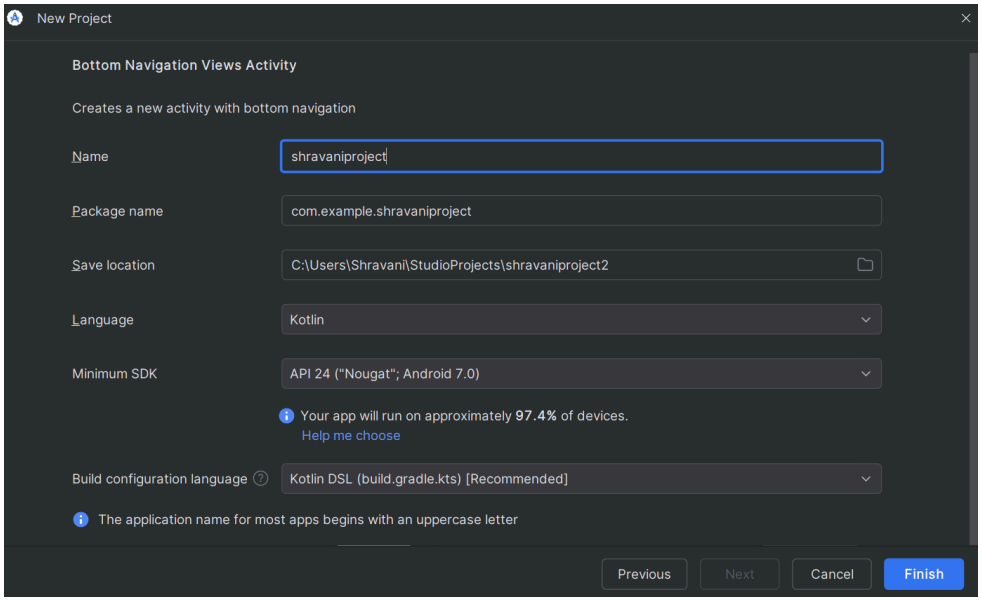
Step 11: - Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself



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



Step 12: Go to File > New Project select project name , location and click next



MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	45
Name	Shravani Rasam
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Flutter Widgets:

In Flutter, everything displayed on the screen is a widget. The layout and structure of your app depend entirely on the sequence and choice of widgets used to build it. Essentially, the app's structure forms a tree of widgets, with each widget potentially containing other widgets, making the UI dynamic and interactive.

Whenever the code is altered, the Flutter framework compares the previous and current widget descriptions to determine the minimal changes needed for rendering the UI, which optimizes performance. Widgets are nested within each other, and from the root of the app to its deepest level, each widget plays a role, whether in displaying content, defining the UI design, or managing user interactions.

Single-Child and Multi-Child Widgets

Flutter has different types of widgets, and one important distinction is between single-child and multi-child widgets.

- **Single-child widgets** allow only one child widget to be nested inside them. These are great for simpler layouts where only a single component needs to be displayed, but often come with additional layout functionalities. Examples of single-child widgets include Padding, Center, and Align.
- **Multi-child widgets** can contain multiple child widgets, allowing for more complex layouts. For instance, the Row widget arranges children horizontally, while the Column widget arranges them vertically. By combining Row and Column, more intricate and hierarchical layouts can be designed.

Types of Widgets in Flutter

Flutter provides two primary types of widgets:

1. **StatefulWidget**
   A StatefulWidget has mutable state and can change during its lifecycle. This type of widget contains two classes: the State object and the Widget class. The state is dynamic and can be updated during the widget’s life. A StatefulWidget doesn’t have a build() method but instead contains a createState() method, which returns an instance of a class that extends Flutter's State class. Examples include widgets like Checkbox, Radio, Slider, InkWell, Form, and TextField.
2. **StatelessWidget**
   A StatelessWidget is immutable and does not change over time. It’s static throughout its lifecycle. These widgets don't maintain any internal state and are usually used for UI

elements that do not need to change after the initial render. Examples include Text, Row, Column, Container, and Icon.

Commonly Used Flutter Widgets

Here are some commonly used Flutter widgets that help structure and design the UI:

- **Container**
  A versatile widget that acts as a box for styling elements such as padding, margins, colors, borders, and constraints. It's often used for layout and positioning of child widgets.
- **Row & Column**
  These are essential for laying out widgets either horizontally (Row) or vertically (Column). They manage the alignment, distribution, and spacing of their child widgets.
- **Stack**
  The Stack widget overlays its child widgets on top of each other, which is useful for creating layered UI elements, such as floating action buttons or tooltips.
- **Text**
  The Text widget displays static or dynamic text on the screen. It supports various styling options like font size, color, weight, and alignment.
- **Image**
  The Image widget is used to display images in the app from sources like assets, networks, or memory. You can control properties like scaling and fitting.
- **Scaffold**
  The Scaffold widget provides the basic structure of an app, including the app bar, body, floating action button, and bottom navigation bar.
- **ListView**
  A scrollable list that efficiently renders a large amount of dynamic content. It can handle both vertical and horizontal scrolling.
- **GridView**
  A grid-based layout that arranges widgets in rows and columns, perfect for displaying items like photos, products, or a dashboard with adjustable columns.
- **SizedBox**
  A widget used to add space between other widgets or to set fixed widths and heights for layout adjustment.
- **ElevatedButton**
  A button with elevation, giving it a raised appearance. It's highly customizable, allowing you to modify its color, shape, and the actions performed when clicked.
- **TextField**
  A user input field that allows text entry. It supports various keyboard types, validation, and configuration options for user input.

- **AppBar**
  A top navigation bar that typically includes the title of the app and action buttons. It's commonly used within the Scaffold widget.
- **BottomNavigationBar**
  A bar placed at the bottom of the screen for navigation. It typically contains icons and labels to switch between different sections or screens of the app.
- **Drawer**
  A side navigation panel that slides out, often used for app menus, settings, or quick navigation links.
- **Card**
  A material design component that presents content in a container with elevation. Cards are often used to display content in a visually distinct manner.

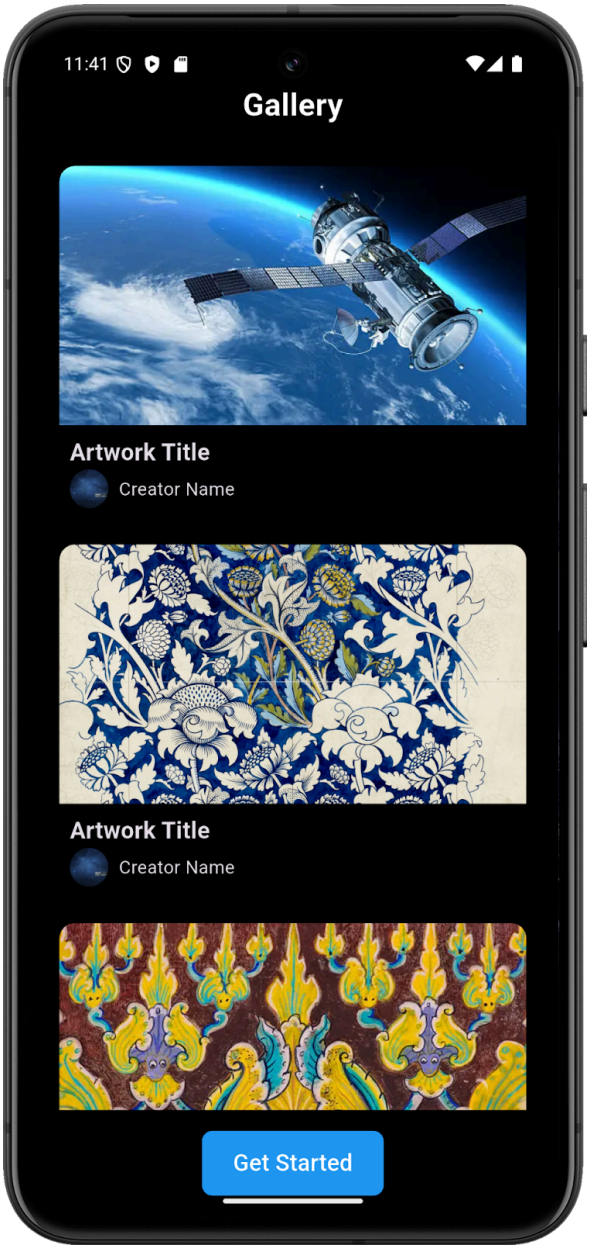
Landing.dart

```
import 'package:flutter/material.dart';
import 'Home.dart';
import 'theme.dart'; // Assuming AppColors is imported from theme.dart

class Landing extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    List<String> imageUrls = [

'https://images.unsplash.com/photo-1577084381380-3b9ea4153664?q=80&w=2012&auto=form
at&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3
D%3D',

'https://media.istockphoto.com/id/452192145/photo/ancient-thai-painting.webp?a=1&b=1&s=612
x612&w=0&k=20&c=VeU1WbFs5ns4TFftXU8ud8PTMq3l9VeNNk-ub2DnWzE=',

'https://images.unsplash.com/photo-1549716679-95380658d5cd?w=600&auto=format&fit=crop
&q=60&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8MjB8fGNhbnZhc3xlbnwwfHwwfHx8
MA%3D%3D',

'https://images.unsplash.com/photo-1471899236350-e3016bf1e69e?q=80&w=2071&auto=form
at&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3
D%3D',

'https://media.istockphoto.com/id/182062885/photo/space-station-in-earth-orbit.webp?a=1&b=1
&s=612x612&w=0&k=20&c=alGFjcWNyVs4fLL9cWWtYZybeef0X1S1iROLA-bbx_8='
```

```
    ];

    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData.dark(),
      home: Scaffold(
        backgroundColor: AppColors.background,
        body: Column(
          children: [
            // Top Bar
            Container(
              margin: const EdgeInsets.only(top: 20),
              padding: const EdgeInsets.symmetric(vertical: 20),
              alignment: Alignment.center,
              color: Colors.black,
              child: Text(
                'Gallery',
                style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold, color: Colors.white),
              ),
            ),

            // Content Feed
            Expanded(
              child: SingleChildScrollView(
                child: Column(
                  children: List.generate(imageUrls.length, (index) => ArtworkCard(imageUrl:
imageUrls[index])),
                ),
              ),
            ),

            // Call-to-Action
            Container(
              margin: const EdgeInsets.all(16),
              child: ElevatedButton(
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.blue,
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(8),
                  ),
                  padding: const EdgeInsets.symmetric(horizontal: 24, vertical: 12),
                ),
                onPressed: () {
                  Navigator.push(
```

```
                context,
                MaterialPageRoute(builder: (context) => HomePage()),
              );
            },
            child: const Text(
              'Get Started',
              style: TextStyle(fontSize: 18, color: Colors.white),
            ),
          ),
        ),
      ],
    ),
  ),
 );
 }
}

class ArtworkCard extends StatelessWidget {
 final String imageUrl;

 ArtworkCard({required this.imageUrl});

 @override
 Widget build(BuildContext context) {
  return Container(
   margin: const EdgeInsets.symmetric(horizontal: 25, vertical: 10),
   decoration: BoxDecoration(
    color: Colors.black54,
    borderRadius: BorderRadius.circular(12),
   ),
   child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
     ClipRRect(
      borderRadius: BorderRadius.vertical(top: Radius.circular(12)),
      child: Image.network(
       imageUrl,
       height: 200,
       width: double.infinity,
       fit: BoxFit.cover,
      ),
     ),
     Padding(
      padding: const EdgeInsets.all(8.0),
```

```
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              const Text(
                'Artwork Title',
                style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
              ),
              Row(
                children: [
                  CircleAvatar(
                    backgroundImage: NetworkImage(

'https://media.istockphoto.com/id/177247796/photo/night-sky-filled-with-stars-and-nebulae.jpg?s
=1024x1024&w=is&k=20&c=iKWp9X2wvjXd5kb0xIV5K_tLzrOSL6FGZggDwDpFXOI='
                    ),
                    radius: 15,
                  ),
                  const SizedBox(width: 8),
                  const Text(
                    'Creator Name',
                    style: TextStyle(fontSize: 14),
                  ),
                ],
              ),
            ],
          ),
        ),
      ],
    ),
  );
 }
}
```



Custome_card.dart

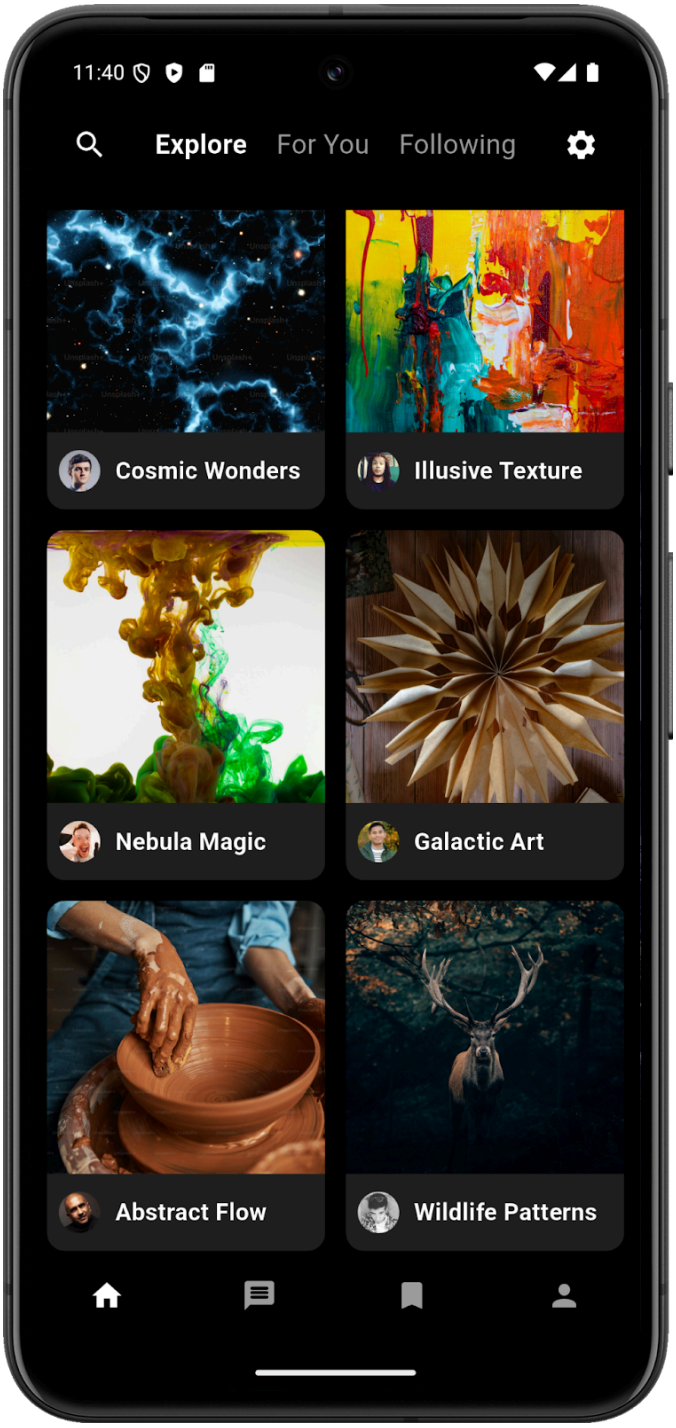
```
import 'package:flutter/material.dart';

class CustomCard extends StatelessWidget {
  final String imageUrl;
  final String title;
  final String avatarUrl;

  const CustomCard({
    Key? key,
    required this.imageUrl,
    required this.title,
    required this.avatarUrl,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
        margin: const EdgeInsets.all(3.0),
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(12),
        color: Colors.black,
      ),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          // Image Section
          Expanded(
            child: Container(
              decoration: const BoxDecoration(
                borderRadius: BorderRadius.only(
                  topLeft: Radius.circular(12),
                  topRight: Radius.circular(12),
                ),
              ),
              child: ClipRRect(
                borderRadius: const BorderRadius.only(
                  topLeft: Radius.circular(12),
                  topRight: Radius.circular(12),
                ),
                child: Image.network(
                  imageUrl,
```

```
                fit: BoxFit.cover,
                width: double.infinity,
              ),
            ),
          ),
        ),
        // Title & Avatar
        Container(
          padding: const EdgeInsets.symmetric(horizontal: 8.0, vertical: 12.0),
          decoration: BoxDecoration(
            color: Colors.grey[900], // Background color added
            borderRadius: const BorderRadius.only(
              bottomLeft: Radius.circular(12),
              bottomRight: Radius.circular(12),
            ),
          ),
          child: Row(
            children: [
              // Avatar
              CircleAvatar(
                radius: 14,
                backgroundImage: NetworkImage(avatarUrl),
              ),
              const SizedBox(width: 10),
              // Title
              Expanded(
                child: Text(
                  title,
                  style: const TextStyle(
                    color: Colors.white,
                    fontSize: 16,
                    fontWeight: FontWeight.bold,
                  ),
                  overflow: TextOverflow.ellipsis,
                ),
              ),
            ],
          ),
        ),
      ],
    ),
  );
 }
}
```

Home.dart

```
import 'package:flutter/material.dart';
import '../widgets/bottom_nav.dart';
import '../widgets/top_nav.dart';
import '../widgets/custom_card.dart';

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

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

class _HomePageState extends State<HomePage> {
  int _selectedIndex = 0;

  void _onNavTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      body: Column(
        children: [
          const TopNavBar(),
          Expanded(
            child: GridView.builder(
              padding: const EdgeInsets.all(8),
              gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
                crossAxisCount: 2,
                crossAxisSpacing: 8,
                mainAxisSpacing: 8,
                childAspectRatio: 0.8,
              ),
              itemCount: 6,
              itemBuilder: (context, index) {
                return CustomCard(
```

```
                imageUrl:
"https://images.unsplash.com/photo-1462331940025-49
6dfbfc7564?q=80&w=2111&aut,
                title: "Gibberish #${index + 1}",
                avatarUrl:
"https://randomuser.me/api/portraits/men/${index +
1}.jpg",
              );
            },
          ),
        ),
      ],
    ),
    bottomNavigationBar:
BottomNavBar(currentIndex: _selectedIndex,
onTap: _onNavTapped),
  );
 }
}
```



Creatives.dart

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

class CreativesPage extends StatelessWidget {
  final List<Map<String, dynamic>> creatives = [
    {
      "name": "Lorenzo 🤘 Bocchi",
      "location": "Australia",
      "responseTime": "Responds quickly",
      "projects": 2,
      "featured": true,
      "profilePic": "https://via.placeholder.com/50",
      "portfolio": [

"https://images.unsplash.com/photo-1547891654-e66ed7ebb968?q=80&w=2070&auto=format&
fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%
3D",

"https://plus.unsplash.com/premium_photo-1673240367277-e1d394465b56?q=80&w=2069&aut
o=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx
8fA%3D%3D",

"https://images.unsplash.com/photo-1547826039-bfc35e0f1ea8?q=80&w=1972&auto=format&fi
t=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3
D"
      ]
    },
    {
      "name": "Jeffrey Dirkse",
      "location": "Netherlands",
      "responseTime": "Responds quickly",
      "projects": 1,
      "featured": true,
      "profilePic": "https://via.placeholder.com/50",
      "portfolio": [

"https://images.unsplash.com/photo-1547891654-e66ed7ebb968?q=80&w=2070&auto=format&
fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%
3D",

"https://plus.unsplash.com/premium_photo-1673240367277-e1d394465b56?q=80&w=2069&aut
```

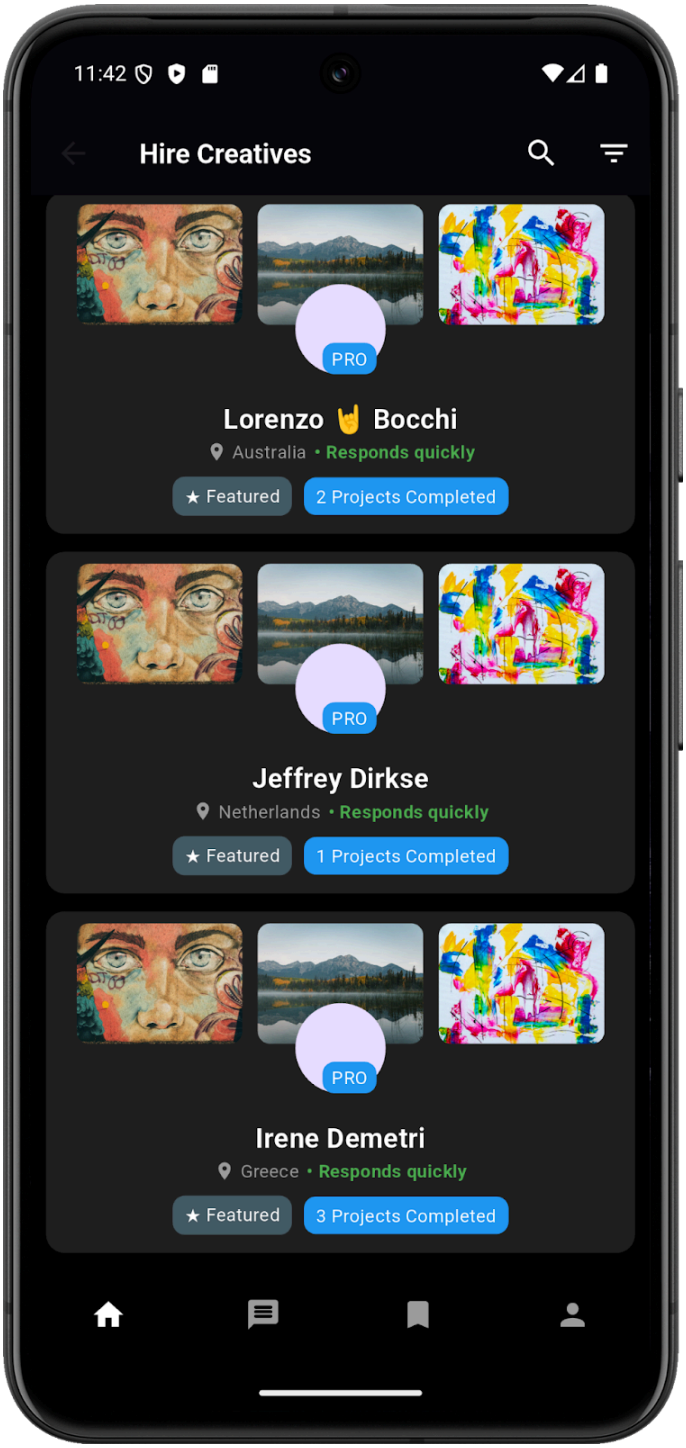
```
o=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx
8fA%3D%3D",

"https://images.unsplash.com/photo-1547826039-bfc35e0f1ea8?q=80&w=1972&auto=format&fi
t=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3
D"
      ]
    },
    {
      "name": "Irene Demetri",
      "location": "Greece",
      "responseTime": "Responds quickly",
      "projects": 3,
      "featured": true,
      "profilePic": "https://via.placeholder.com/50",
      "portfolio": [

"https://images.unsplash.com/photo-1547891654-e66ed7ebb968?q=80&w=2070&auto=format&
fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%
3D",

"https://plus.unsplash.com/premium_photo-1673240367277-e1d394465b56?q=80&w=2069&aut
o=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx
8fA%3D%3D",

"https://images.unsplash.com/photo-1547826039-bfc35e0f1ea8?q=80&w=1972&auto=format&fi
t=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3
D"
      ]
    }
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      appBar: AppBar(
        backgroundColor: Colors.black,
        title: const Text("Hire Creatives",
            style: TextStyle(color: Colors.white,
                fontSize: 18,
                fontWeight: FontWeight.bold)),
        actions: [
          IconButton(
```

```
        icon: const Icon(Icons.search, color: Colors.white),
        onPressed: () {},
      ),
      IconButton(
        icon: const Icon(Icons.filter_list, color: Colors.white),
        onPressed: () {},
      ),
    ],
  ),
  body: ListView.builder(
    padding: const EdgeInsets.all(10),
    itemCount: creatives.length,
    itemBuilder: (context, index) {
      final creative = creatives[index];

      return Container(
        margin: const EdgeInsets.only(bottom: 12),
        decoration: BoxDecoration(
          color: Colors.grey[900],
          borderRadius: BorderRadius.circular(12),
        ),
        child: Column(
          children: [
            // Portfolio
            Stack(
              alignment: Alignment.bottomCenter,
              clipBehavior: Clip.none,
              children: [
                Padding(
                  padding: const EdgeInsets.symmetric(
                      horizontal: 10, vertical: 8),
                  child: Row(
                    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                    children: (creative['portfolio'] as List<dynamic>)
                        .map<Widget>((image) =>
                        Flexible(
                          child: ClipRRect(
                            borderRadius: BorderRadius.circular(8),
                            child: Image.network(image,
                                width: 110, height: 80, fit: BoxFit.cover),
                          ),
                        ))
                        .toList(),
                  ),
```

```
    ),
    // Profile Picture with PRO Badge
    Positioned(
      bottom: -25,
      child: Stack(
        clipBehavior: Clip.none,
        children: [
          CircleAvatar(
            radius: 30,
            backgroundImage: NetworkImage(
                creative['profilePic']),
          ),
          Positioned(
            bottom: 0,
            left: 18,
            child: Container(
              padding: const EdgeInsets.symmetric(
                  horizontal: 6, vertical: 2),
              decoration: BoxDecoration(
                color: Colors.blue,
                borderRadius: BorderRadius.circular(8),
              ),
              child: const Text("PRO",
                  style: TextStyle(
                      color: Colors.white, fontSize: 12)),
            ),
          ),
        ],
      ),
    ),
  ],
),
const SizedBox(height: 30),

// Profile Details
Padding(
  padding: const EdgeInsets.all(12),
  child: Column(
    children: [
      Text(creative['name'],
          style: const TextStyle(
              color: Colors.white,
              fontSize: 18,
              fontWeight: FontWeight.bold)),
```

```
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    const Icon(Icons.location_on, color: Colors.grey,
        size: 14),
    Text(" ${creative['location']}",
        style: const TextStyle(
            color: Colors.grey, fontSize: 12)),
    const SizedBox(width: 5),
    Text("• ${creative['responseTime']}",
        style: const TextStyle(
            color: Colors.green,
            fontSize: 12,
            fontWeight: FontWeight.bold)),
  ],
),
const SizedBox(height: 8),
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    if (creative['featured'])
      Container(
        padding: const EdgeInsets.symmetric(
            horizontal: 8, vertical: 4),
        decoration: BoxDecoration(
          color: Colors.blueGrey[700],
          borderRadius: BorderRadius.circular(8),
        ),
        child: const Text("★ Featured",
            style: TextStyle(
                color: Colors.white, fontSize: 12)),
      ),
    const SizedBox(width: 8),
    Container(
      padding: const EdgeInsets.symmetric(
          horizontal: 8, vertical: 4),
      decoration: BoxDecoration(
        color: Colors.blue,
        borderRadius: BorderRadius.circular(8),
      ),
      child: Text(
          "${creative['projects']} Projects Completed",
          style: const TextStyle(
              color: Colors.white, fontSize: 12)),
```

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


    bottomNavigationBar:
BottomNavBar(currentIndex: 0, onTap: (index)
{}),
   );
  }
}
```



MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	45
Name	Shravani Rasam
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Theory:

Flutter is an open-source UI framework that allows developers to build natively compiled applications for multiple platforms—mobile, web, and desktop—using a single codebase. One of the primary advantages of Flutter is its flexibility in creating highly customizable user interfaces (UIs). This experiment focuses on the integration of essential visual elements—icons, images, and custom fonts—into a Flutter application. These visual elements not only enhance the aesthetic appeal of an app but also contribute to a more engaging and intuitive user experience.

In Flutter, an app consists of both assets (resources) and code. Assets refer to the static resources, such as images, icons, and configuration files, which are available during runtime. Flutter supports a wide variety of image formats, including JPEG, PNG, GIF, WebP, BMP, and others, enabling developers to include rich media content in their apps.

Visual elements serve a crucial role in mobile app development, offering the following benefits:

- **Improved User Experience:** Icons, images, and other visual elements make the app more visually appealing, engaging, and easy to use.
- **Quick Communication**: Visuals can convey complex information instantly. An icon can often represent a concept more effectively than text.
- **Brand Identity**: Custom visuals, such as logos and unique icon designs, contribute to reinforcing the app's branding, helping it stand out in a competitive market.

Adding Icons in Flutter

Flutter offers a range of built-in material design icons accessible through the Icons class. In addition to the default icons, developers can also add custom icons using third-party packages such as flutter_launcher_icons or font_awesome_flutter. Here is an example of adding an icon in Flutter:

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

Adding Images in Flutter

Flutter supports displaying images from three different sources:

1. **Local Assets (Stored within the app project)**
    - Store the image inside the assets/images/ folder of the project.
    - Declare the image path in the pubspec.yaml file.
    - Use the Image.asset widget to display the image in the app.
2. Example:

```
flutter:

    assets:

      - assets/images/sample.png
```

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

Network (Fetched from the internet)

- Displaying images from the internet is simple using the `Image.network` method. This method allows you to provide additional properties such as width, height, and fit for controlling the image's appearance.

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

Adding Custom Fonts in Flutter

While Flutter uses the Roboto font by default, you can add custom fonts to personalize your app's typography.

- Download the font file and place it in the assets/fonts/ folder.
- Declare the custom font in the pubspec.yaml file.
- Apply the custom font to text widgets in your app using the TextStyle widget.

Example:

1. Place the custom font in the assets/fonts/ folder.
2. Declare the font in pubspec.yaml:

yaml

```
flutter:
  fonts:
    - family: 'CustomFont'
      fonts:
        - asset: assets/fonts/CustomFont-Regular.ttf


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

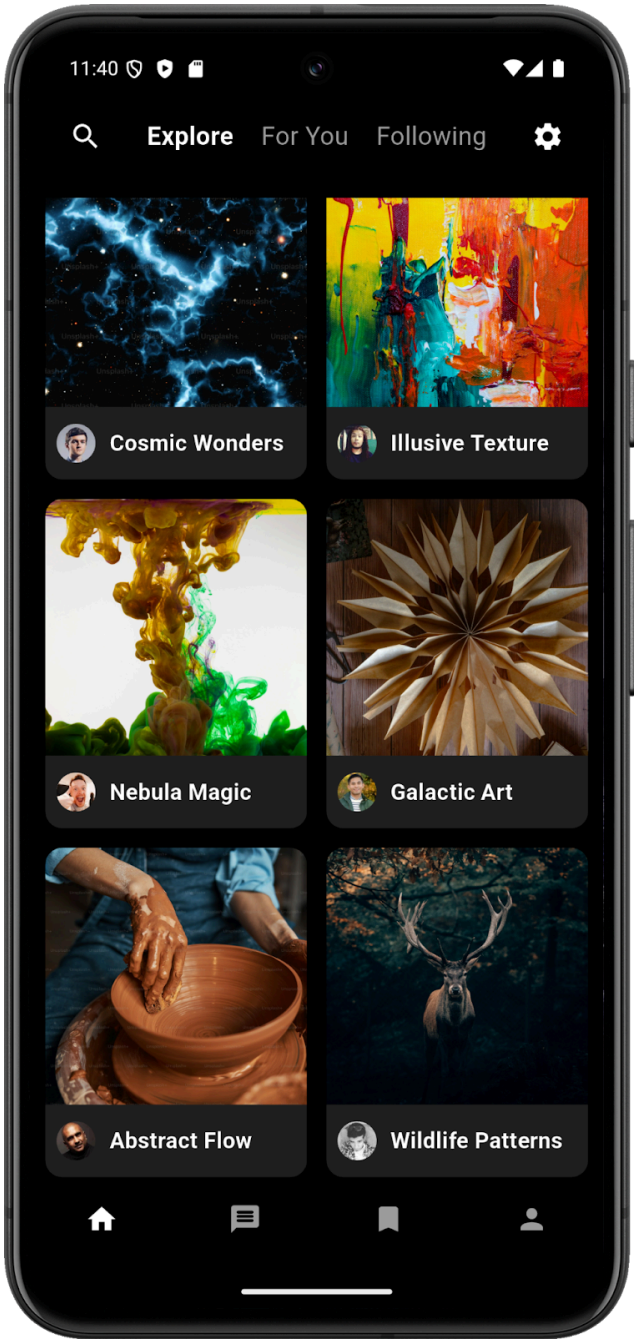
bottom_nav.dart

```
import 'package:flutter/material.dart';
import '../Creatives.dart'; // Ensure the
correct import for CreativesPage
import '../AboutInfo.dart'; // Ensure this
file contains AboutInfoPage
import '../ArtistDetails.dart';
class TopNavBar extends StatelessWidget
{
 const TopNavBar({Key? key}) :
super(key: key);

 @override
 Widget build(BuildContext context) {
  return Container(
   margin: const EdgeInsets.only(top:
30),
   padding: const
EdgeInsets.symmetric(horizontal: 16,
vertical: 20),
   color: Colors.black,
   child: Row(
    mainAxisAlignment:
MainAxisAlignment.spaceBetween,
    children: [
     // Search Icon
     IconButton(
      icon: const Icon(Icons.search,
color: Colors.white),
      onPressed: () {
       // Implement search functionality
here
      },
     ),

     const Text(
      "Explore",
      style: TextStyle(color:
Colors.white, fontSize: 18, fontWeight:
FontWeight.bold),
     ),

     // "For You" with Navigation
     GestureDetector(
      onTap: () {
       Navigator.push(
        context,
        MaterialPageRoute(builder:
(context) => AboutInfoPage()), // Ensure
this page exists
       );
      },
      child: const Text(
       "For You",
       style: TextStyle(color:
Colors.grey, fontSize: 18),
      ),
     ),

     // "Following" Text
     GestureDetector(
      onTap: () {
       Navigator.push(
        context,
        MaterialPageRoute(builder:
(context) => ArtistDetails()), // Ensure
this page exists
       );
      },
      child: const Text(
       "Following",
       style: TextStyle(color:
Colors.grey, fontSize: 18),
      ),
     ),

     // Settings Icon with Navigation
     IconButton(
      icon: const Icon(Icons.settings,
color: Colors.white),
      onPressed: () {
       Navigator.push(
        context,
        MaterialPageRoute(builder:
(context) => CreativesPage()), // Ensure
CreativesPage exists );
```

Home.dart

```
import 'package:flutter/material.dart';
import '../widgets/bottom_nav.dart';
import '../widgets/top_nav.dart';
import '../widgets/custom_card.dart';

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

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

class _HomePageState extends
State<HomePage> {
  int _selectedIndex = 0;

  void _onNavTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }

  final List<Map<String, String>> items
= [
    {
      "title": "Cosmic Wonders",
      "imageUrl":
"https://plus.unsplash.com/premium_p
hoto-1667105168041-3fad1d08b58b?
q=80&w=2127&auto=format&fit=crop
&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8
MHxwaG90by1wYWdlfHx8fGVufDB8f
Hx8fA%3D%3D",
    },
    {
      "title": "Illusive Texture",
      "imageUrl":
"https://images.unsplash.com/photo-1
541961017774-22349e4a1262?q=80
&w=1958&auto=format&fit=crop&ixlib
=rb-4.0.3&ixid=M3wxMjA3fDB8MHxw
aG90by1wYWdlfHx8fGVufDB8fHx8fA
%3D%3D",
    },
    {
      "title": "Nebula Magic",
      "imageUrl":
"https://images.unsplash.com/photo-1
612743138072-675de14392e1?q=80
&w=1972&auto=format&fit=crop&ixlib
=rb-4.0.3&ixid=M3wxMjA3fDB8MHxw
aG90by1wYWdlfHx8fGVufDB8fHx8fA
%3D%3D",
    },
    {
      "title": "Galactic Art",
      "imageUrl":
"https://images.unsplash.com/photo-1
607349877492-8f2410658848?q=80&
w=1974&auto=format&fit=crop&ixlib=r
b-4.0.3&ixid=M3wxMjA3fDB8MHxwa
G90by1wYWdlfHx8fGVufDB8fHx8fA
%3D%3D",
    },
    {
      "title": "Abstract Flow",
      "imageUrl":
"https://plus.unsplash.com/premium_p
hoto-1661683887049-69e8ae0a6c92
?q=80&w=2071&auto=format&fit=cro
p&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8
MHxwaG90by1wYWdlfHx8fGVufDB8f
Hx8fA%3D%3D",
    },
    {
      "title": "Wildlife Patterns",
      "imageUrl":
"https://images.unsplash.com/photo-1
543946207-39bd91e70ca7?q=80&w=
1974&auto=format&fit=crop&ixlib=rb-4
.0.3&ixid=M3wxMjA3fDB8MHxwaG90
```

```
by1wYWdlfHx8fGVufDB8fHx8fA%3D
%3D",
    },
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      body: Column(
        children: [
          const TopNavBar(),
          Expanded(
            child: GridView.builder(
              padding: const
EdgeInsets.all(8),
              gridDelegate: const
SliverGridDelegateWithFixedCrossAxi
sCount(
                crossAxisCount: 2,
                crossAxisSpacing: 8,
                mainAxisSpacing: 8,
                childAspectRatio: 0.8,
              ),
              itemCount: items.length,
              itemBuilder: (context, index) {
                return CustomCard(
                  imageUrl:
items[index]["imageUrl"]!,
                  title: items[index]["title"]!,
                  avatarUrl:
"https://randomuser.me/api/portraits/m
en/${index + 1}.jpg",
                );
              },
            ),
          ),
        ],
      ),
      bottomNavigationBar:
BottomNavBar(currentIndex:
_selectedIndex, onTap:
_onNavTapped),
    );
  }
}
```



Custome_card.dart

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

class CustomCard extends
StatelessWidget {
  final String imageUrl;
  final String title;
  final String avatarUrl;

  const CustomCard({
    Key? key,
    required this.imageUrl,
    required this.title,
    required this.avatarUrl,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      margin: const
EdgeInsets.all(3.0),
     decoration: BoxDecoration(
      borderRadius:
BorderRadius.circular(12),
      color: Colors.black,
     ),
     child: Column(
      crossAxisAlignment:
CrossAxisAlignment.start,
      children: [
       // Image Section
       Expanded(
         child: Container(
           decoration: const
BoxDecoration(
             borderRadius:
BorderRadius.only(
               topLeft:
Radius.circular(12),
               topRight:
Radius.circular(12),
             ),
           ),
           child: ClipRRect(
             borderRadius: const
BorderRadius.only(
               topLeft:
Radius.circular(12),
               topRight:
Radius.circular(12),
             ),
             child: Image.network(
               imageUrl,
               fit: BoxFit.cover,
               width: double.infinity,
             ),
           ),
         ),
       ),
       // Title & Avatar
       Container(
         padding: const
EdgeInsets.symmetric(horizontal: 8.0,
vertical: 12.0),
         decoration: BoxDecoration(
           color: Colors.grey[900], //
Background color added
           borderRadius: const
BorderRadius.only(
             bottomLeft:
Radius.circular(12),
             bottomRight:
Radius.circular(12),
           ),
         ),
         child: Row(
           children: [
             // Avatar
             CircleAvatar(
               radius: 14,
               backgroundImage:
NetworkImage(avatarUrl),
             ),
             const SizedBox(width: 10),
             // Title
             Expanded(
               child: Text(
```

```
                title,
                style: const TextStyle(
                  color: Colors.white,
                  fontSize: 16,
                  fontWeight:
FontWeight.bold,
                ),
                overflow:
TextOverflow.ellipsis,
              ),
            ),
          ],
        ),
      ),
    ],
  ),
 );
 }
}
```

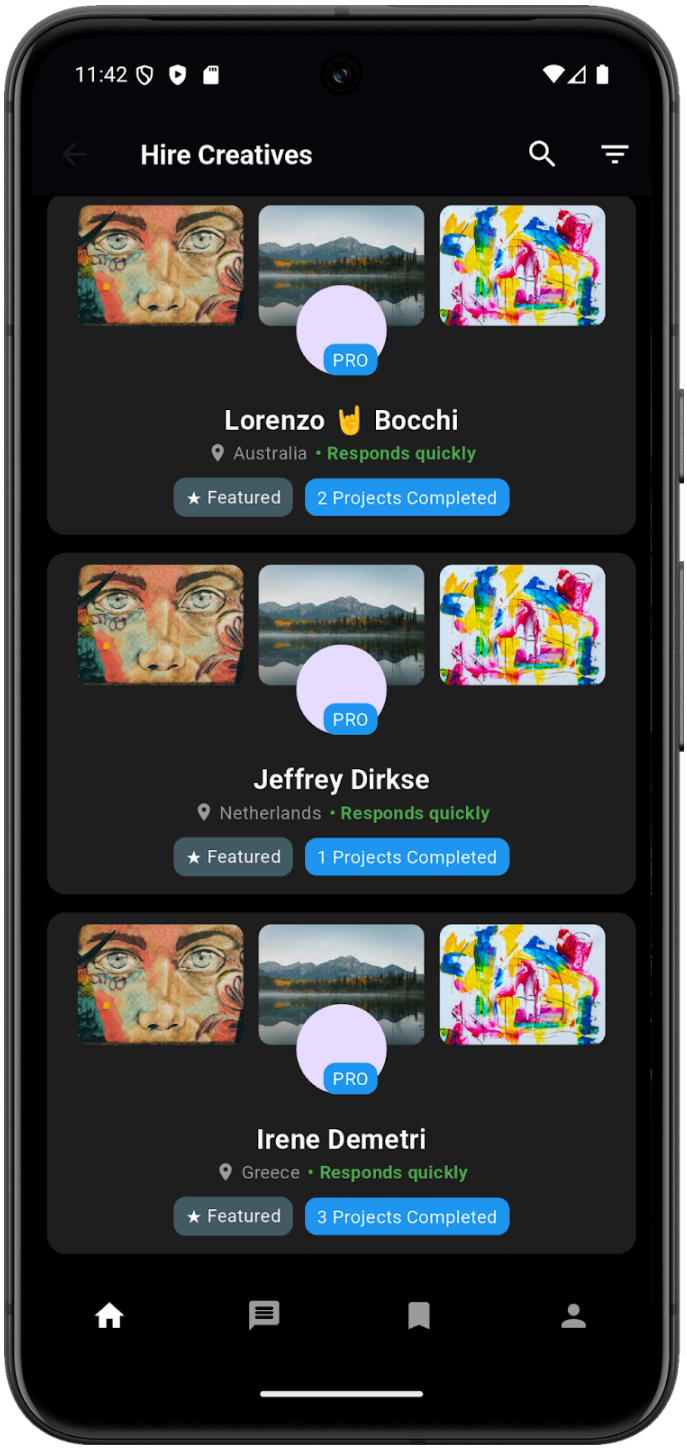
Creatives.dart

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

class CreativesPage extends
StatelessWidget {
  final List<Map<String, dynamic>>
creatives = [
    {
      "name": "Lorenzo 🤘 Bocchi",
      "location": "Australia",
      "responseTime": "Responds quickly",
      "projects": 2,
      "featured": true,
      "profilePic":
"https://via.placeholder.com/50",
      "portfolio": [

"https://images.unsplash.com/photo-1547
91654-e66ed7ebb968?q=80&w=2070&au
o=format&fit=crop&ixlib=rb-4.0.3&ixid=M3
wxMjA3fDB8MHxwaG90by1wYWdlfHx8fG
VufDB8fHx8fA%3D%3D",

"https://plus.unsplash.com/premium_photo
-1673240367277-e1d394465b56?q=80&w
=2069&auto=format&fit=crop&ixlib=rb-4.0.
3&ixid=M3wxMjA3fDB8MHxwaG90by1wY
WdlfHx8fGVufDB8fHx8fA%3D%3D",

"https://images.unsplash.com/photo-1547
26039-bfc35e0f1ea8?q=80&w=1972&auto
=format&fit=crop&ixlib=rb-4.0.3&ixid=M3w
xMjA3fDB8MHxwaG90by1wYWdlfHx8fGV
ufDB8fHx8fA%3D%3D"
      ]
    },
    {
      "name": "Jeffrey Dirkse",
      "location": "Netherlands",
      "responseTime": "Responds quickly",
      "projects": 1,
      "featured": true,
      "profilePic":
"https://via.placeholder.com/50",
      "portfolio": [

"https://images.unsplash.com/photo-15478
91654-e66ed7ebb968?q=80&w=2070&aut
o=format&fit=crop&ixlib=rb-4.0.3&ixid=M3
wxMjA3fDB8MHxwaG90by1wYWdlfHx8fG
VufDB8fHx8fA%3D%3D",

"https://plus.unsplash.com/premium_photo
-1673240367277-e1d394465b56?q=80&w
=2069&auto=format&fit=crop&ixlib=rb-4.0.
3&ixid=M3wxMjA3fDB8MHxwaG90by1wY
WdlfHx8fGVufDB8fHx8fA%3D%3D",

"https://images.unsplash.com/photo-15478
26039-bfc35e0f1ea8?q=80&w=1972&auto
=format&fit=crop&ixlib=rb-4.0.3&ixid=M3w
xMjA3fDB8MHxwaG90by1wYWdlfHx8fGV
ufDB8fHx8fA%3D%3D"
      ]
    },
    {
      "name": "Irene Demetri",
      "location": "Greece",
      "responseTime": "Responds quickly",
      "projects": 3,
      "featured": true,
      "profilePic":
"https://via.placeholder.com/50",
      "portfolio": [

"https://images.unsplash.com/photo-15478
91654-e66ed7ebb968?q=80&w=2070&aut
o=format&fit=crop&ixlib=rb-4.0.3&ixid=M3
wxMjA3fDB8MHxwaG90by1wYWdlfHx8fG
VufDB8fHx8fA%3D%3D",

"https://plus.unsplash.com/premium_photo
-1673240367277-e1d394465b56?q=80&w
=2069&auto=format&fit=crop&ixlib=rb-4.0.
3&ixid=M3wxMjA3fDB8MHxwaG90by1wY
WdlfHx8fGVufDB8fHx8fA%3D%3D",
```

```
"https://images.unsplash.com/photo-1547826039-bfc35e0f1ea8?q=80&w=1972&autc=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D"
    ]
  }
 ];

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   backgroundColor: Colors.black,
   appBar: AppBar(
    backgroundColor: Colors.black,
    title: const Text("Hire Creatives",
      style: TextStyle(color: Colors.white
        fontSize: 18,
        fontWeight: FontWeight.bold)),
    actions: [
     IconButton(
      icon: const Icon(Icons.search, color: Colors.white),
      onPressed: () {},
     ),
     IconButton(
      icon: const Icon(Icons.filter_list, color: Colors.white),
      onPressed: () {},
     ),
    ],
   ),
   body: ListView.builder(
    padding: const EdgeInsets.all(10),
    itemCount: creatives.length,
    itemBuilder: (context, index) {
     final creative = creatives[index];

     return Container(
      margin: const EdgeInsets.only(bottom: 12),
      decoration: BoxDecoration(
       color: Colors.grey[900],
       borderRadius: BorderRadius.circular(12),
      ),
      child: Column(
       children: [
        // Portfolio Preview
        Stack(
         alignment: Alignment.bottomCenter,
         clipBehavior: Clip.none,
         children: [
          Padding(
           padding: const EdgeInsets.symmetric(
             horizontal: 10, vertical: 8),
           child: Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: (creative['portfolio'] as List<dynamic>)
              .map<Widget>((image) =>
              Flexible(
               child: ClipRRect(
                borderRadius: BorderRadius.circular(8),
                child: Image.network(image,
                  width: 110, height: 80, fit: BoxFit.cover),
               ),
              ))
              .toList(),
           ),
          ),
          // Profile Picture with PRO Badge
          Positioned(
           bottom: -25,
           child: Stack(
            clipBehavior: Clip.none,
            children: [
             CircleAvatar(
              radius: 30,
```

```
                    backgroundImage:
NetworkImage(
                      creative['profilePic']),
                  ),
                  Positioned(
                    bottom: 0,
                    left: 18,
                    child: Container(
                      padding: const
EdgeInsets.symmetric(
                        horizontal: 6, vertical
2),
                      decoration:
BoxDecoration(
                        color: Colors.blue,
                        borderRadius:
BorderRadius.circular(8),
                      ),
                      child: const Text("PRO"
                        style: TextStyle(
                          color:
Colors.white, fontSize: 12)),
                      ),
                    ),
                  ],
                ),
              ),
            ],
          ),
          const SizedBox(height: 30),

          // Profile Details
          Padding(
            padding: const
EdgeInsets.all(12),
            child: Column(
              children: [
                Text(creative['name'],
                  style: const TextStyle(
                    color: Colors.white,
                    fontSize: 18,
                    fontWeight:
FontWeight.bold)),
                Row(
                  mainAxisAlignment:
MainAxisAlignment.center,
                  children: [
                    const
Icon(Icons.location_on, color: Colors.grey,
                      size: 14),
                    Text("
${creative['location']}",
                      style: const TextStyle(
                        color: Colors.grey,
fontSize: 12)),
                    const SizedBox(width: 5),
                    Text("•
${creative['responseTime']}",
                      style: const TextStyle(
                        color: Colors.green,
                        fontSize: 12,
                        fontWeight:
FontWeight.bold)),
                  ],
                ),
                const SizedBox(height: 8),
                Row(
                  mainAxisAlignment:
MainAxisAlignment.center,
                  children: [
                    if (creative['featured'])
                      Container(
                        padding: const
EdgeInsets.symmetric(
                          horizontal: 8, vertical:
4),
                        decoration:
BoxDecoration(
                          color:
Colors.blueGrey[700],
                          borderRadius:
BorderRadius.circular(8),
                        ),
                        child: const Text("★
Featured",
                          style: TextStyle(
                            color:
Colors.white, fontSize: 12)),
```

```
                    ),
                  const SizedBox(width: 8),
                  Container(
                    padding: const
EdgeInsets.symmetric(
                        horizontal: 8, vertical:
4),
                    decoration:
BoxDecoration(
                      color: Colors.blue,
                      borderRadius:
BorderRadius.circular(8),
                    ),
                    child: Text(
                      "${creative['projects']}
Projects Completed",
                      style: const TextStyle(
                        color: Colors.white,
fontSize: 12)),
                  ),
                ],
              ),
            ],
          ),
        ),
      ),

      // Bottom Navigation Bar
      bottomNavigationBar:
BottomNavBar(currentIndex: 0, onTap:
(index) {}),
    );
  }
}
```



MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	45
Name	Shravani Rasam
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Forms in Flutter

In Flutter, a form is a container used to gather user input through various fields such as text fields, checkboxes, dropdowns, and buttons. Forms are essential for apps requiring user data entry, like login pages, registration forms, and feedback submissions. Flutter provides the Form widget, which, in combination with TextFormField and other input elements, helps manage validation, state handling, and error messages efficiently. Proper form validation ensures data accuracy, leading to an enhanced user experience.

When creating a form, a GlobalKey is required. This key uniquely identifies the form, allowing you to perform validation and manage form field states. The TextFormField widget is used to render input fields for user text entry, providing built-in support for validation error display.

Creating a Form in Flutter

1. **Form Widget**
   The Form widget acts as a container for multiple form fields and manages validation. It holds child widgets like TextFormField and validates the input fields.
2. **GlobalKey<FormState>**
   This key is essential for uniquely identifying the form, enabling validation, and allowing the retrieval of data from form fields.
3. **TextFormField**
   TextFormField provides input fields where users can enter text, such as names, email addresses, and phone numbers. It also supports validation.
4. **Input Decoration**
   InputDecoration enhances the appearance of input fields, enabling customizations such as labels, icons, borders, and hint text to improve usability.
5. **Validation**
   The validator property within TextFormField is used to ensure user input meets specific criteria, such as proper email format or non-empty values.
6. **Keyboard Types**
   Different types of input require different keyboard types. For instance, TextInputType.number is used for numeric fields, and TextInputType.emailAddress is used for email fields.
7. **State Management**
   Proper state management is vital for storing and retrieving user input, ensuring correct processing of the form data.
8. **Submit Button**
   A submit button triggers form validation and submits the entered data for further processing, such as saving to a database.

Key Properties of the Form Widget

- **key**: A GlobalKey that uniquely identifies the form. This key is used to interact with the form, such as validating, resetting, or saving the form's state.
- **child**: A child widget that contains the form fields. This can be a Column, ListView, or other widget used to arrange form fields vertically or horizontally.
- **autovalidateMode**: Specifies when the form should automatically validate its fields (e.g., on field change or when the form is submitted).

Methods of the Form Widget

- **validate()**: Triggers validation of all form fields. It returns true if all fields are valid, and false if any field fails validation. This method is used before submitting the form to ensure data integrity.
- **save()**: Saves the current values of all form fields and invokes the onSaved callback for each field. This is typically called after validation succeeds.
- **reset()**: Resets the form to its initial state, clearing any entered data.
- **currentState**: A getter that returns the current FormState associated with the form, allowing you to access and manipulate the form’s state.

Login.dart

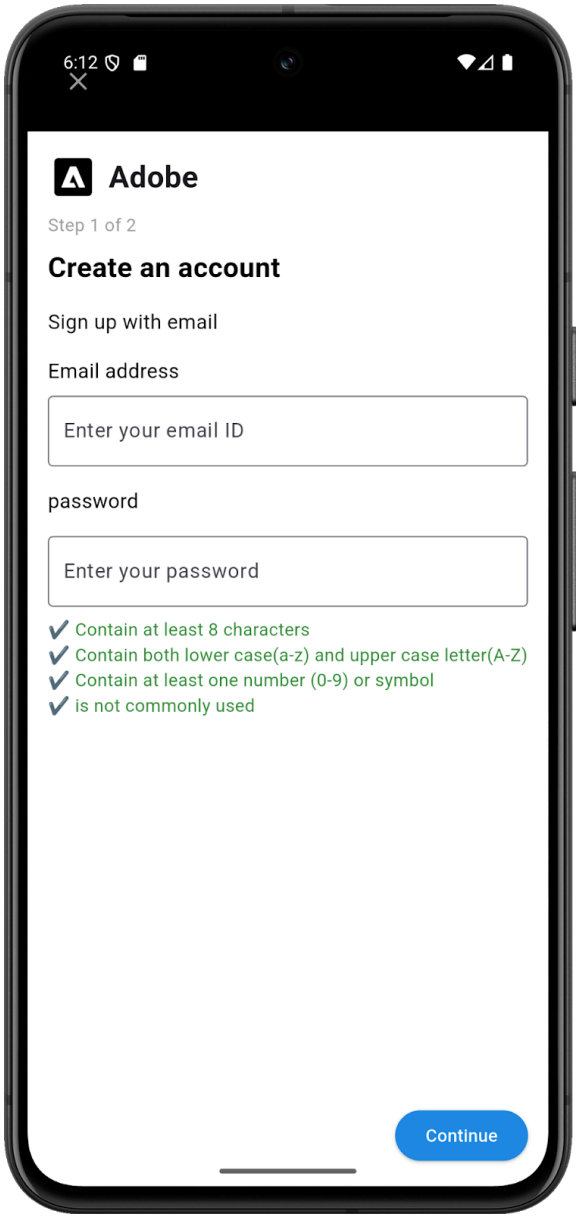
```
import 'package:flutter/material.dart';
import 'theme.dart';
import 'profile.dart';

class Login extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   backgroundColor: Colors.white,
   body: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
     Container(
      color: AppColors.background,
      padding: EdgeInsets.all(16),
      child: Row(
       children: [
```

```
            IconButton(
              icon: Icon(Icons.close, color: AppColors.hint),
              onPressed: () => Navigator.pop(context),
            ),
          ],
        ),
      ),
      Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Row(
              children: [
                Icon(Icons.adobe, size: 40, color: Colors.black),
                SizedBox(width: 8),
                Text(
                  'Adobe',
                  style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
                ),
              ],
            ),
            SizedBox(height: 8),
            Text(
              'Step 1 of 2',
              style: TextStyle(color: AppColors.hint, fontSize: 14),
            ),
            SizedBox(height: 8),
            Text(
              'Create an account',
              style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold, color:
Colors.black),
            ),
            SizedBox(height: 16),
            Text('Sign up with email', style: TextStyle(fontSize: 16)),

            SizedBox(height: 16),
            Text(
              'Email address',
              style: TextStyle(fontSize: 16, color: Colors.black),
```

```
          ),

          SizedBox(height: 8),
          TextField(
            decoration: InputDecoration(
              border: OutlineInputBorder(borderSide: BorderSide(color:
AppColors.hint)),
              hintText: 'Enter your email ID',
            ),
          ),
          SizedBox(height: 16),
          Text(
            'password',
            style: TextStyle(fontSize: 16, color: Colors.black),
          ),

          SizedBox(height: 16),
          TextField(
            obscureText: true,
            decoration: InputDecoration(
              border: OutlineInputBorder(borderSide: BorderSide(color:
AppColors.hint)),
              hintText: 'Enter your password',
            ),

          ),
          SizedBox(height: 8),
          Text(
            '✔️ Contain at least 8 characters',
            style: TextStyle(color: Colors.green[700], fontSize: 14),
          ),
          Text(
            '✔️ Contain both lower case(a-z) and upper case letter(A-Z)',
            style: TextStyle(color: Colors.green[700], fontSize: 14),
          ),
          Text(
            '✔️ Contain at least one number (0-9) or symbol',
            style: TextStyle(color: Colors.green[700], fontSize: 14),
          ),
          Text(
```

```
                '✔️ is not commonly used',
                style: TextStyle(color: Colors.green[700], fontSize: 14),
              ),
            ],
          ),
        ),
        Spacer(),
        Padding(
          padding: const EdgeInsets.all(16.0),
          child: Align(
            alignment: Alignment.bottomRight,
            child: ElevatedButton(
              style: ElevatedButton.styleFrom(
                backgroundColor: AppColors.primary,
                foregroundColor: Colors.white,
              ),
              onPressed: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(builder: (context) => Profile()),
                );
              },
              child: Text('Continue'),
            ),
          ),
        ),
      ],
    ),
  );
 }
}
```

Project Title: Behance **Roll No.** 45



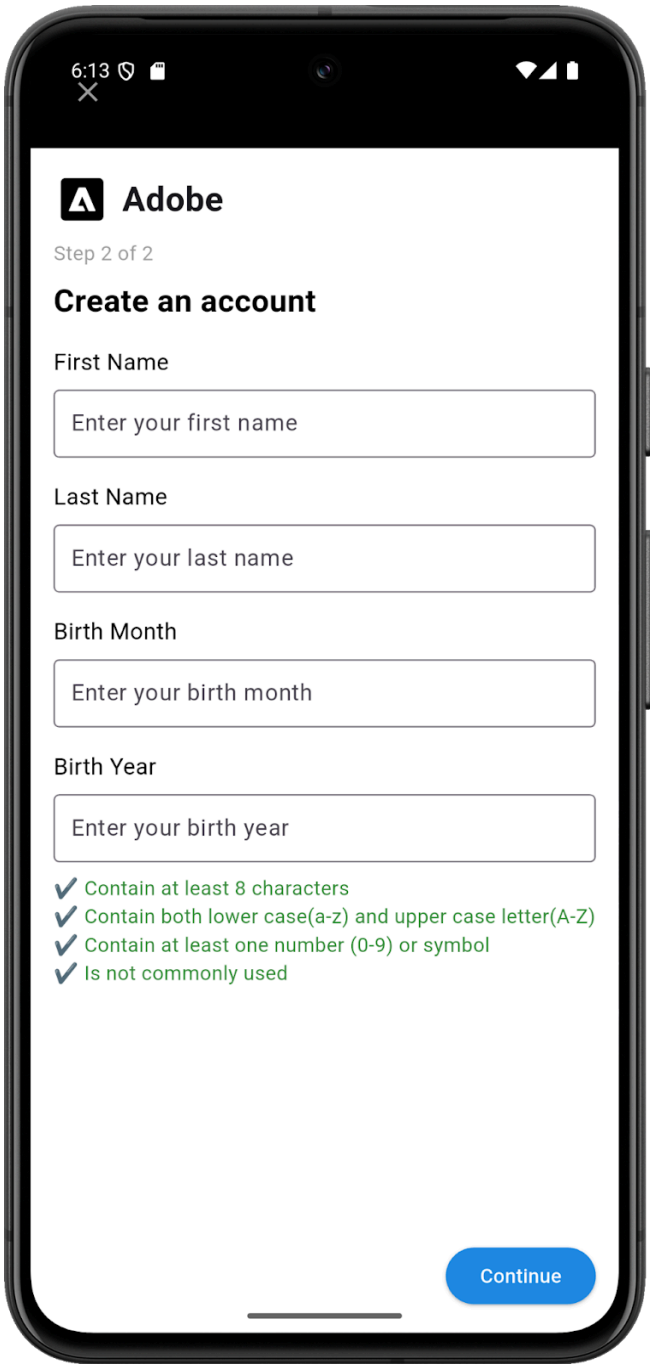
ProfileCreation.dart

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

class Profile extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
   return Scaffold(
     backgroundColor: Colors.white,
     body: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
       Container(
         color: AppColors.background,
         padding: EdgeInsets.all(16),
         child: Row(
           children: [
             IconButton(
               icon: Icon(Icons.close, color: AppColors.hint),
               onPressed: () => Navigator.pop(context),
             ),
           ],
         ),
       ),
       Padding(
         padding: const EdgeInsets.all(16.0),
         child: Column(
           crossAxisAlignment: CrossAxisAlignment.start,
           children: [
             Row(
               children: [
                 Icon(Icons.adobe, size: 40, color: Colors.black),
                 SizedBox(width: 8),
                 Text(
                   'Adobe',
                   style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
                 ),
               ],
```

```
),
SizedBox(height: 8),
Text(
 'Step 2 of 2',
 style: TextStyle(color: AppColors.hint, fontSize: 14),
),
SizedBox(height: 8),
Text(
 'Create an account',
 style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold, color: Colors.black),
),
SizedBox(height: 16),
Text(
 'First Name',
 style: TextStyle(fontSize: 16, color: Colors.black),
),
SizedBox(height: 8),
TextField(
 decoration: InputDecoration(
  border: OutlineInputBorder(borderSide: BorderSide(color: AppColors.hint)),
  hintText: 'Enter your first name',
  contentPadding: EdgeInsets.symmetric(vertical: 8, horizontal: 12), // Reduced padding
 ),
),
SizedBox(height: 16),
Text(
 'Last Name',
 style: TextStyle(fontSize: 16, color: Colors.black),
),
SizedBox(height: 8),
TextField(
 decoration: InputDecoration(
  border: OutlineInputBorder(borderSide: BorderSide(color: AppColors.hint)),
  hintText: 'Enter your last name',
  contentPadding: EdgeInsets.symmetric(vertical: 8, horizontal: 12), // Reduced padding
 ),
),
SizedBox(height: 16),
Text(
 'Birth Month',
```

```
  style: TextStyle(fontSize: 16, color: Colors.black),
),
SizedBox(height: 8),
TextField(
  decoration: InputDecoration(
    border: OutlineInputBorder(borderSide: BorderSide(color: AppColors.hint)),
    hintText: 'Enter your birth month',
    contentPadding: EdgeInsets.symmetric(vertical: 8, horizontal: 12), // Reduced padding
  ),
),
SizedBox(height: 16),
Text(
  'Birth Year',
  style: TextStyle(fontSize: 16, color: Colors.black),
),
SizedBox(height: 8),
TextField(
  decoration: InputDecoration(
    border: OutlineInputBorder(borderSide: BorderSide(color: AppColors.hint)),
    hintText: 'Enter your birth year',
    contentPadding: EdgeInsets.symmetric(vertical: 8, horizontal: 12), // Reduced padding
  ),
),
SizedBox(height: 8),
Text(
  '✓ Contain at least 8 characters',
  style: TextStyle(color: Colors.green[700], fontSize: 14),
),
Text(
  '✓ Contain both lower case(a-z) and upper case letter(A-Z)',
  style: TextStyle(color: Colors.green[700], fontSize: 14),
),
Text(
  '✓ Contain at least one number (0-9) or symbol',
  style: TextStyle(color: Colors.green[700], fontSize: 14),
),
Text(
  '✓ Is not commonly used',
  style: TextStyle(color: Colors.green[700], fontSize: 14),
),
```

```
            ],
          ),
        ),
        Spacer(),
        Padding(
          padding: const EdgeInsets.all(16.0),
          child: Align(
                                        alignment:
Alignment.bottomRight,
            child: ElevatedButton(
              style: ElevatedButton.styleFrom(
                                    backgroundColor:
AppColors.primary,
                foregroundColor: Colors.white,
              ),
              onPressed: () {
                Navigator.push(
                  context,
                      MaterialPageRoute(builder:
(context) => Landing()),
                );
              },
              child: Text('Continue'),
            ),))
      ],
    ),
  );
 }}
```

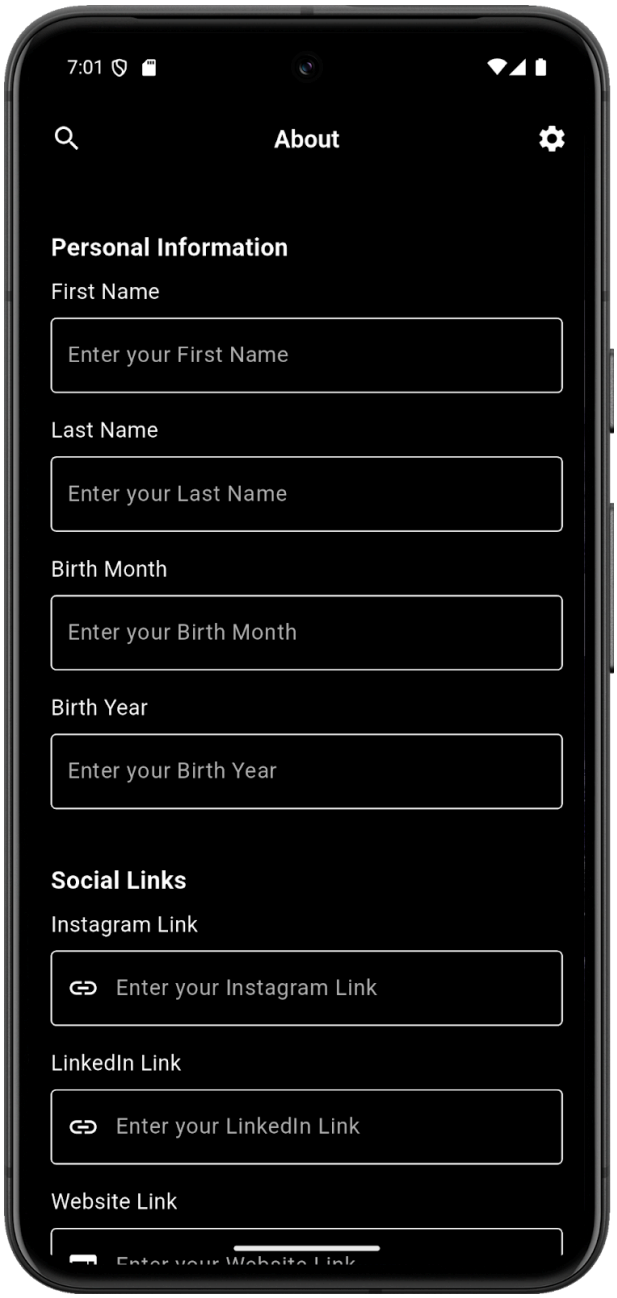


AboutInfo.dart

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

class AboutInfoPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      appBar: AppBar(
        backgroundColor: Colors.black,
        leading: IconButton(
          icon: Icon(Icons.search, color: Colors.white),
          onPressed: () {},
        ),
        title: Text(
          "About",
          style: TextStyle(color: Colors.white, fontSize: 18, fontWeight: FontWeight.bold),
        ),
        centerTitle: true,
        actions: [
          IconButton(
            icon: Icon(Icons.settings, color: Colors.white),
            onPressed: () {},
          ),
        ],
      ),


      body: SingleChildScrollView(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            _buildSectionTitle("Personal Information"),
            _buildInputField('First Name'),
            _buildInputField('Last Name'),
            _buildInputField('Birth Month'),
```

```
        _buildInputField('Birth Year'),

        _buildSectionTitle("Social Links"),
        _buildInputField('Instagram Link', icon: Icons.link),
        _buildInputField('LinkedIn Link', icon: Icons.link),
        _buildInputField('Website Link', icon: Icons.web),

        _buildSectionTitle("Professional Details"),
        _buildInputField('Work Experience', maxLines: 3),
        _buildInputField('Focus Area', maxLines: 2),

        SizedBox(height: 24),
        Center(
          child: ElevatedButton(
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.white,
              foregroundColor: Colors.black,
              padding: EdgeInsets.symmetric(horizontal: 24, vertical: 12),
            ),
            onPressed: () {},
            child: Text("Save"),
          ),
        ),
      ],
    ),
  ),
 );
}

Widget _buildSectionTitle(String title) {
 return Padding(
   padding: const EdgeInsets.only(top: 24, bottom: 8),
   child: Text(
     title,
     style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold, color: Colors.white),
   ),
 );
}

Widget _buildInputField(String label, {IconData? icon, int maxLines = 1}) {
```

```
    return Padding(
      padding: const EdgeInsets.only(bottom: 16),
      child: Column(
                                        crossAxisAlignment:
CrossAxisAlignment.start,
        children: [
          Text(
            label,
                      style: TextStyle(fontSize: 16, color:
Colors.white),
          ),
          SizedBox(height: 8),
          TextField(
            style: TextStyle(color: Colors.white),
            maxLines: maxLines,
            decoration: InputDecoration(
              filled: true,
              fillColor: Colors.transparent,
              enabledBorder: OutlineInputBorder(
                                borderSide: BorderSide(color:
Colors.white),
              ),
              focusedBorder: OutlineInputBorder(
                                borderSide: BorderSide(color:
Colors.white, width: 2),
              ),
                prefixIcon: icon != null ? Icon(icon, color:
Colors.white) : null,
              hintText: 'Enter your $label',
              hintStyle: TextStyle(color: Colors.white70),
            ),
          ),
        ],
      ),
    );
  }
}
```



MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	45
Name	Shravani Rasam
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Theory:

Flutter provides powerful tools to implement navigation, routing, and gestures within your applications. These elements are crucial for building an interactive and user-friendly app. Navigation and routing allow users to move between different screens or pages, while gestures enable intuitive interactions with the UI.

1. Navigation and Routing in Flutter

Navigation in Flutter allows you to move between different screens or routes within your app. Flutter uses a **Navigator** to manage the app's stack of screens. Each screen in the app is referred to as a **route**, and the Navigator pushes and pops these routes based on user actions.

Key Concepts:

- **Navigator**: A widget that manages a stack of routes. You can push new routes onto the stack and pop routes off.
- **Routes**: Each screen in the app is represented by a route, which can be defined using either a `MaterialPageRoute`, `CupertinoPageRoute`, or custom routes.

Types of Navigation in Flutter:

1. **Push and Pop Navigation**: The basic form of navigation, where you "push" a new screen onto the stack or "pop" the current screen off.

```
Navigator.push(
  context,
  MaterialPageRoute(builder: (context) => SecondScreen()),
);
```

2. Gestures in Flutter

Gestures allow users to interact with the app through various touch actions such as tapping, swiping, pinching, and dragging. Flutter provides a rich set of gesture recognizers to detect user input and perform corresponding actions.

Key Gesture Types:

1. **Tap Gesture**: A single tap on a widget. It is the most common gesture.

```
GestureDetector(
  onTap: () {
    print('Tapped!');
  },
```

```
  child: Container(
    color: Colors.blue,
    height: 100,
    width: 100,
  ),
);
```

2 .Swipe Gestures: Swiping across the screen in various directions.

```
GestureDetector(
  onHorizontalDragUpdate: (details) {
    print('Swiping horizontally');
  },
  onVerticalDragUpdate: (details) {
    print('Swiping vertically');
  },
  child: Container(
    height: 100,
    width: 100,
    color: Colors.red,
  ),
);
```

3. Long Press Gesture: Triggered by a long tap on a widget.

```
GestureDetector(
  onLongPress: () {
    print('Long press detected');
  },
  child: Container(
    color: Colors.green,
    height: 100,
    width: 100,
  ),
);
```

Navigation and Routing :
Flutter enables users to move seamlessly between different screens and organize the app's flow. It allows both simple and complex routing, making it easier to manage your app's structure.

Gestures :
enhance user interaction by enabling intuitive touch-based actions, such as tapping, swiping, and pinching, to provide a more dynamic and engaging experience.

top_nav.dart

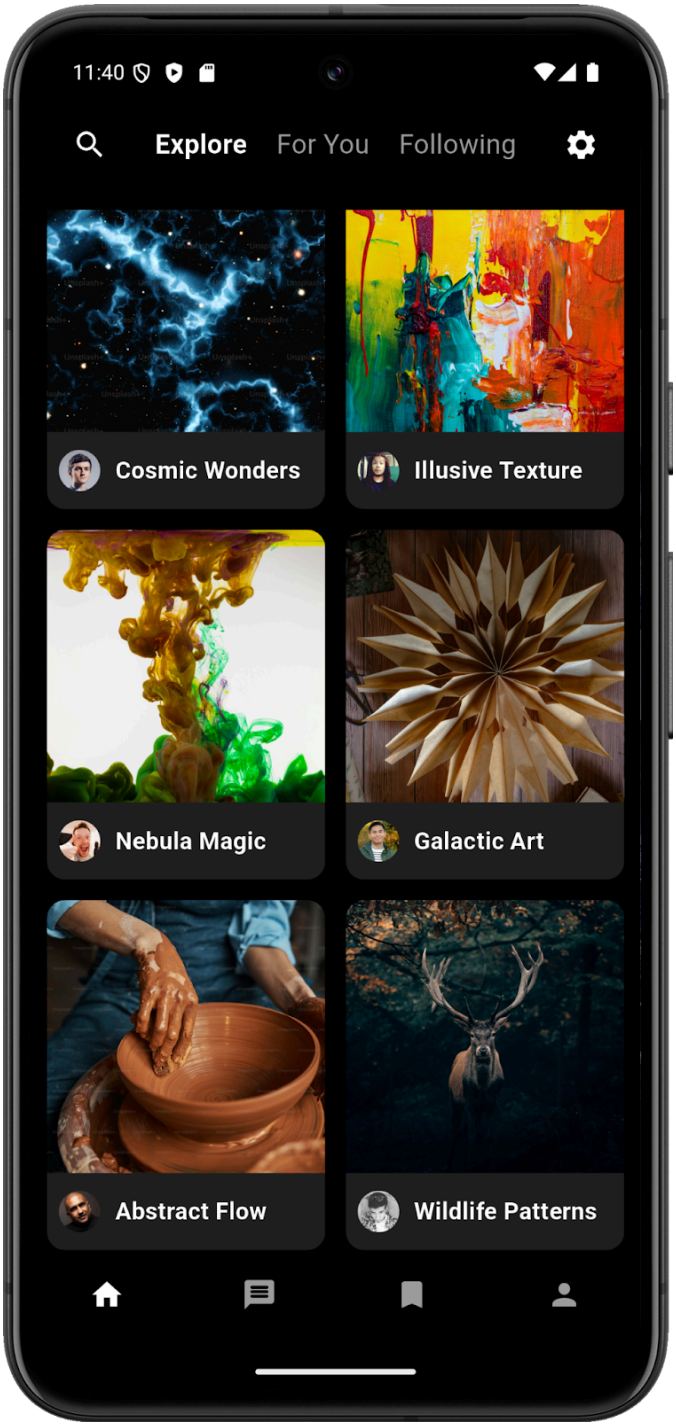
```
import 'package:flutter/material.dart';
import '../Creatives.dart'; // Ensure the correct import for CreativesPage
import '../AboutInfo.dart'; // Ensure this file contains AboutInfoPage
import '../ArtistDetails.dart';
class TopNavBar extends StatelessWidget {
 const TopNavBar({Key? key}) : super(key: key);

 @override
 Widget build(BuildContext context) {
  return Container(
   margin: const EdgeInsets.only(top: 30),
   padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 20),
   color: Colors.black,
   child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [
     // Search Icon
     IconButton(
      icon: const Icon(Icons.search, color: Colors.white),
      onPressed: () {
       // Implement search functionality here
      },
     ),

     // Explore Text
     const Text(
      "Explore",
      style: TextStyle(color: Colors.white, fontSize: 18, fontWeight: FontWeight.bold),
     ),

     // "For You" with Navigation
     GestureDetector(
      onTap: () {
       Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => AboutInfoPage()), // Ensure this page exists
       );
      },
      child: const Text(
       "For You",
       style: TextStyle(color: Colors.grey, fontSize: 18),
      ),
     ),

     // "Following" Text
```

```
        GestureDetector(
          onTap: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder:
(context) => ArtistDetails()), // Ensure
this page exists
            );
          },
          child: const Text(
            "Following",
            style: TextStyle(color:
Colors.grey, fontSize: 18),
          ),
        ),

        // Settings Icon with Navigation
        IconButton(
          icon: const Icon(Icons.settings,
color: Colors.white),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder:
(context) => CreativesPage()), // Ensure
CreativesPage exists
            );
          },
        ),
      ],
    ),
  );
 }
}
```

bottom_nav.dart

```
import 'package:flutter/material.dart';
import '../Creatives.dart'; // Ensure the
correct import for CreativesPage
import '../AboutInfo.dart'; // Ensure this
file contains AboutInfoPage
import '../ArtistDetails.dart';
class TopNavBar extends StatelessWidget
{
 const TopNavBar({Key? key}) :
super(key: key);

 @override
 Widget build(BuildContext context) {
  return Container(
   margin: const EdgeInsets.only(top:
30),
   padding: const
EdgeInsets.symmetric(horizontal: 16,
vertical: 20),
   color: Colors.black,
   child: Row(
    mainAxisAlignment:
MainAxisAlignment.spaceBetween,
    children: [
     // Search Icon
     IconButton(
      icon: const Icon(Icons.search,
color: Colors.white),
      onPressed: () {
       // Implement search functionality
here
      },
     ),

     const Text(
      "Explore",
      style: TextStyle(color:
Colors.white, fontSize: 18, fontWeight:
FontWeight.bold),
     ),

     // "For You" with Navigation
     GestureDetector(
      onTap: () {
       Navigator.push(
        context,
        MaterialPageRoute(builder:
(context) => AboutInfoPage()), // Ensure
this page exists
       );
      },
      child: const Text(
       "For You",
       style: TextStyle(color:
Colors.grey, fontSize: 18),
      ),
     ),

     // "Following" Text
     GestureDetector(
      onTap: () {
       Navigator.push(
        context,
        MaterialPageRoute(builder:
(context) => ArtistDetails()), // Ensure
this page exists
       );
      },
      child: const Text(
       "Following",
       style: TextStyle(color:
Colors.grey, fontSize: 18),
      ),
     ),

     // Settings Icon with Navigation
     IconButton(
      icon: const Icon(Icons.settings,
color: Colors.white),
      onPressed: () {
       Navigator.push(
        context,
        MaterialPageRoute(builder:
(context) => CreativesPage()), // Ensure
CreativesPage exists );
```

Project Title: Behance **Roll No.** 45



Custome_card.dart

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

class CustomCard extends
StatelessWidget {
  final String imageUrl;
  final String title;
  final String avatarUrl;

  const CustomCard({
    Key? key,
    required this.imageUrl,
    required this.title,
    required this.avatarUrl,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      margin: const
EdgeInsets.all(3.0),
     decoration: BoxDecoration(
      borderRadius:
BorderRadius.circular(12),
      color: Colors.black,
     ),
     child: Column(
      crossAxisAlignment:
CrossAxisAlignment.start,
      children: [
       // Image Section
       Expanded(
         child: Container(
           decoration: const
BoxDecoration(
             borderRadius:
BorderRadius.only(
               topLeft:
Radius.circular(12),
               topRight:
Radius.circular(12),
             ),
           ),
           child: ClipRRect(
             borderRadius: const
BorderRadius.only(
               topLeft:
Radius.circular(12),
               topRight:
Radius.circular(12),
             ),
             child: Image.network(
               imageUrl,
               fit: BoxFit.cover,
               width: double.infinity,
             ),
           ),
         ),
       ),
       // Title & Avatar
       Container(
         padding: const
EdgeInsets.symmetric(horizontal: 8.0,
vertical: 12.0),
         decoration: BoxDecoration(
           color: Colors.grey[900], //
Background color added
           borderRadius: const
BorderRadius.only(
             bottomLeft:
Radius.circular(12),
             bottomRight:
Radius.circular(12),
           ),
         ),
         child: Row(
           children: [
             // Avatar
             CircleAvatar(
               radius: 14,
               backgroundImage:
NetworkImage(avatarUrl),
             ),
             const SizedBox(width: 10),
             // Title
             Expanded(
               child: Text(
```

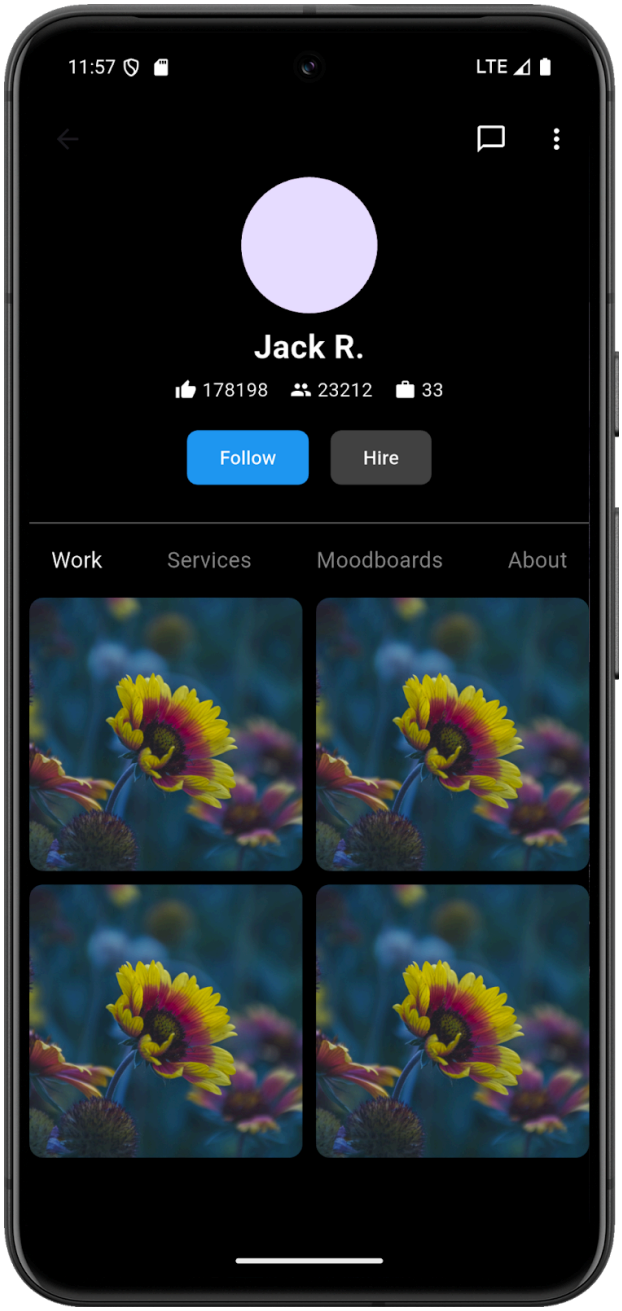
```
                title,
                style: const TextStyle(
                  color: Colors.white,
                  fontSize: 16,
                  fontWeight:
FontWeight.bold,
                ),
                overflow:
TextOverflow.ellipsis,
              ),
            ),
          ],
        ),
      ),
    ],
  ),
 );
 }
}
```

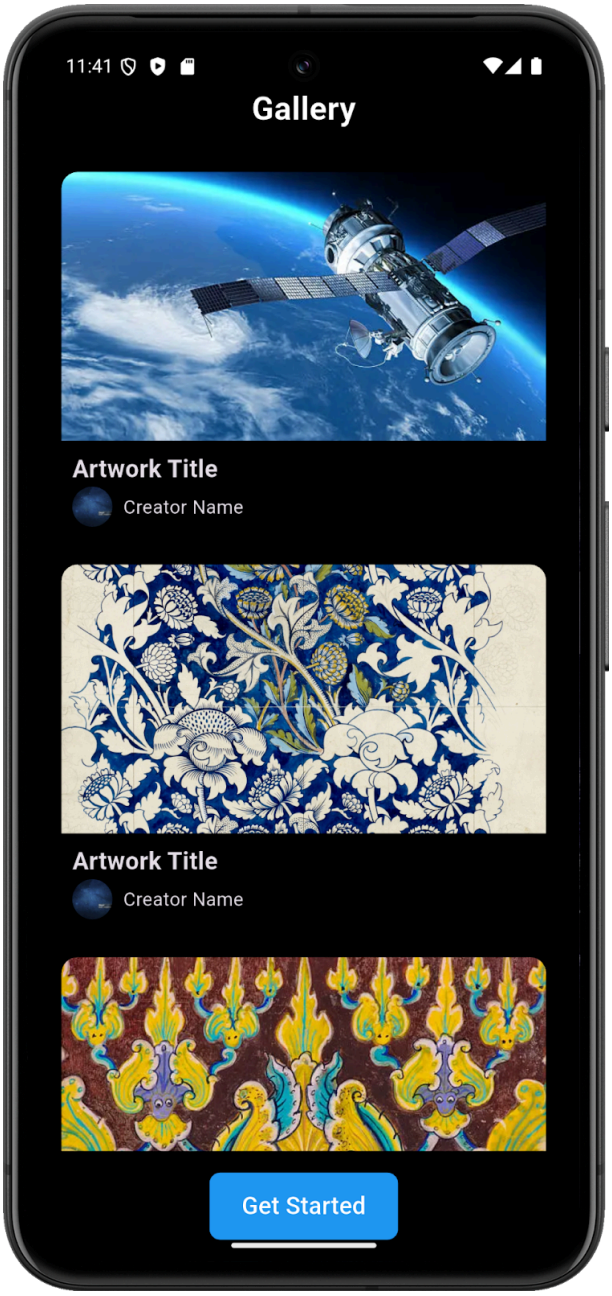
ArtistDetails.dart

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

class ArtistDetails extends
StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      appBar: AppBar(
        backgroundColor: Colors.black,
        elevation: 0,
        actions: [
          IconButton(
            icon:
Icon(Icons.chat_bubble_outline, color:
Colors.white),
            onPressed: () {},
          ),
          IconButton(
            icon: Icon(Icons.more_vert, color:
Colors.white),
            onPressed: () {},
          ),
        ],
      ),
      body: SingleChildScrollView(
        child: Column(
          children: [
            // Profile Header
            Center(
              child: Column(
                children: [
                  CircleAvatar(
                    radius: 50,
                    backgroundImage:
AssetImage('assets/artist.jpg'), // Change
with actual image
                  ),
                  SizedBox(height: 8),
                  Text(
                    'Jack R.',
                    style: TextStyle(color:
Colors.white, fontSize: 24, fontWeight:
FontWeight.bold),
                  ),
                  SizedBox(height: 4),
                  Row(
                    mainAxisAlignment:
MainAxisAlignment.center,
                    children: [
                      Icon(Icons.thumb_up, color:
Colors.white, size: 16),
                      SizedBox(width: 4),
                      Text('178198', style:
TextStyle(color: Colors.white)),
                      SizedBox(width: 16),
                      Icon(Icons.group, color:
Colors.white, size: 16),
                      SizedBox(width: 4),
                      Text('23212', style:
TextStyle(color: Colors.white)),
                      SizedBox(width: 16),
                      Icon(Icons.work, color:
Colors.white, size: 16),
                      SizedBox(width: 4),
                      Text('33', style:
TextStyle(color: Colors.white)),
                    ],
                  ),
                  SizedBox(height: 16),
                  Row(
                    mainAxisAlignment:
MainAxisAlignment.center,
                    children: [
                      ElevatedButton(
                        style:
ElevatedButton.styleFrom(
                          backgroundColor:
Colors.blue,
                          shape:
RoundedRectangleBorder(borderRadius:
BorderRadius.circular(8)),
                        ),
                        onPressed: () {},
```

```
                    child: Text('Follow',style:
TextStyle(color: Colors.white)),
                  ),
                  SizedBox(width: 16),
                  ElevatedButton(
                    style:
ElevatedButton.styleFrom(
                      backgroundColor:
Colors.grey[800],
                      shape:
RoundedRectangleBorder(borderRadius:
BorderRadius.circular(8)),
                    ),
                    onPressed: () {},
                    child: Text('Hire', style:
TextStyle(color: Colors.white)),
                  ),
                ],
              ),
              SizedBox(height: 16),
              Divider(color: Colors.white38),
              SizedBox(height: 8),
            ],
          ),
        ),
        // Tab Bar
        Container(
          padding:
EdgeInsets.symmetric(horizontal: 16),
          child: Row(
            mainAxisAlignment:
MainAxisAlignment.spaceBetween,
            children: [
              Text('Work', style:
TextStyle(color: Colors.white, fontSize:
16)),
              Text('Services', style:
TextStyle(color: Colors.white54, fontSize:
16)),
              Text('Moodboards', style:
TextStyle(color: Colors.white54, fontSize:
16)),
              Text('About', style:
TextStyle(color: Colors.white54, fontSize:
16)),
            ],
          ),
        ),
        SizedBox(height: 16),
        // Portfolio Section
        GridView.builder(
          shrinkWrap: true,
          physics:
NeverScrollableScrollPhysics(),
          gridDelegate:
SliverGridDelegateWithFixedCrossAxisCo
unt(
            crossAxisCount: 2,
            crossAxisSpacing: 10,
            mainAxisSpacing: 10,
            childAspectRatio: 1,
          ),
          itemCount: 4, // Example images
          itemBuilder: (context, index) {
            return ClipRRect(
              borderRadius:
BorderRadius.circular(10),
              child: Image.network(

'https://images.unsplash.com/photo-16624
52212461-6075942a8781?q=80&w=1974
&auto=format&fit=crop&ixlib=rb-4.0.3&ixid
=M3wxMjA3fDB8MHxwaG90by1wYWdlfH
x8fGVufDB8fHx8fA%3D%3D',
                fit: BoxFit.cover,
              ),
            );
          },
        ),
      ],
    ),
  ),
 );
 }
}
```

Project Title: Behance **Roll No.** 45





MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	45
Name	Shravani Rasam
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

THEORY:

Introduction to Firebase and Flutter Integration

Firebase is a comprehensive platform developed by Google, designed to help developers build high-quality applications for both mobile and web. It provides essential services such as real-time databases, authentication, cloud storage, hosting, and much more. One of the most widely used Firebase services is the Firebase Realtime Database, which is a NoSQL cloud database that allows data to be stored and synced in real-time across all connected devices.

Flutter, on the other hand, is an open-source UI software development kit created by Google, which allows developers to build natively compiled applications for mobile, web, and desktop from a single codebase. Its rich set of pre-designed widgets and powerful tools makes Flutter an attractive option for developing visually appealing and performant applications. Integrating Firebase with Flutter allows developers to leverage the full potential of Firebase services in their applications.

By using Firebase's Realtime Database, Flutter apps can achieve features such as real-time data synchronization, secure authentication, and cloud-based storage. This combination enables developers to create powerful, scalable, and feature-rich mobile and web applications.

Setting Up Firebase in Flutter :

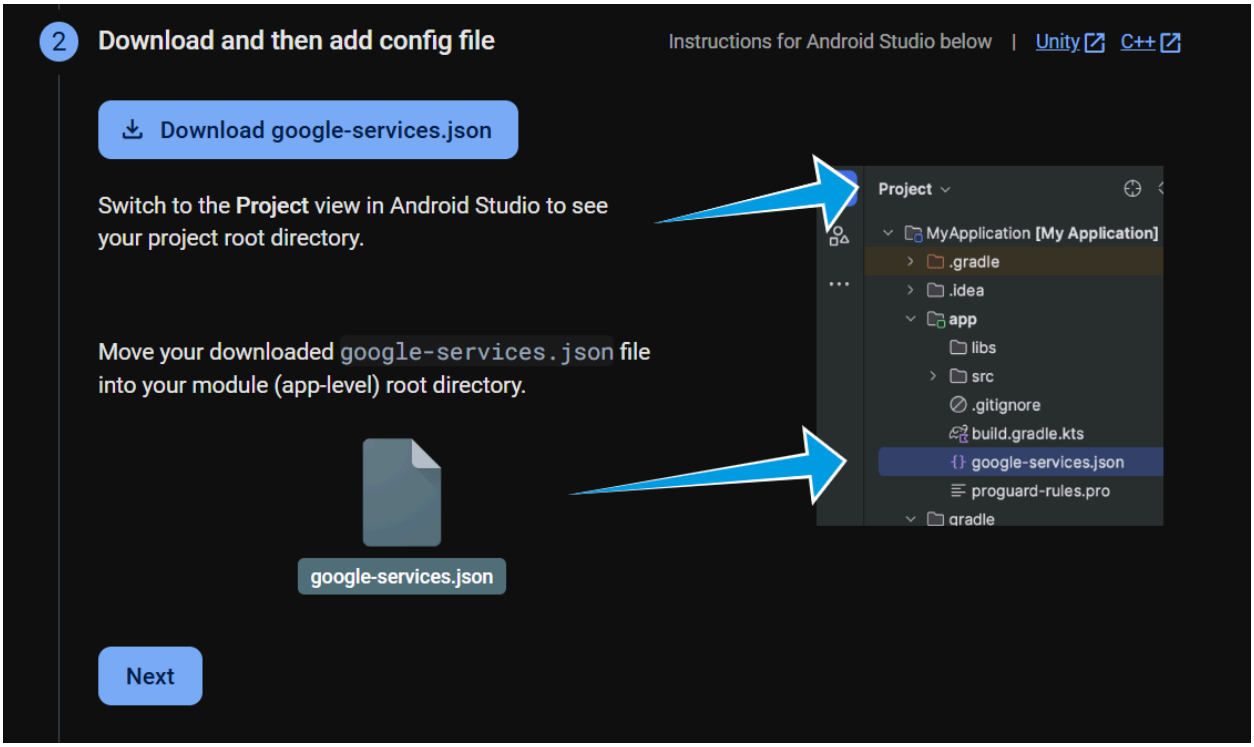
To connect a Flutter app with Firebase, the following steps are typically followed:

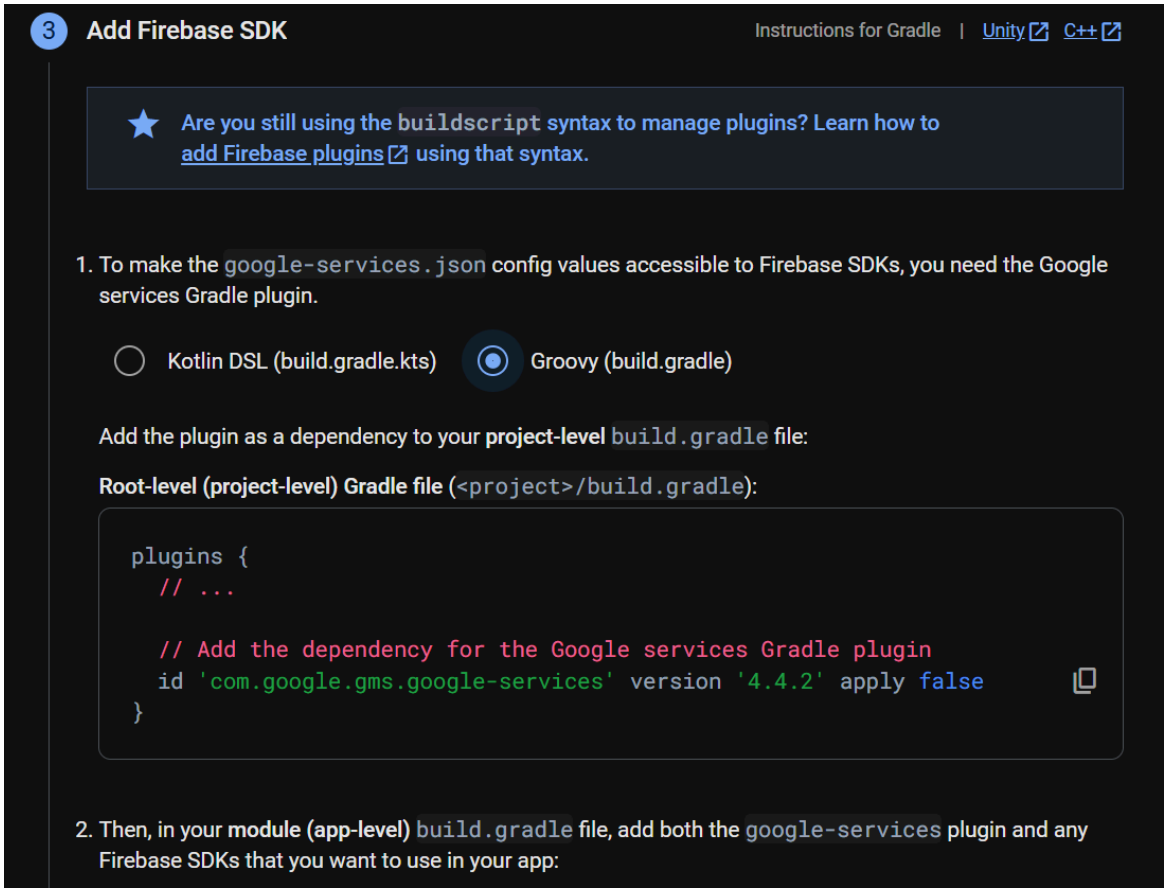
1. Creating a Firebase Project: To start using Firebase with Flutter, the first step is to create a Firebase project in the Firebase Console. Once the project is created, developers can associate their Flutter app with the Firebase project by following the platform-specific instructions for Android or iOS. This usually involves configuring API keys, downloading configuration files, and adding them to the Flutter project.

2. Integrating Firebase SDK in Flutter: After the Firebase project is set up, developers need to integrate Firebase's SDK into the Flutter app. This involves adding the necessary dependencies to the Flutter project's pubspec.yaml file. For Firebase's Realtime Database, the package firebase_database is used. Additionally, Firebase's core SDK (firebase_core) must also be included to initialize Firebase services.

3. Initializing Firebase: Before any Firebase functionality can be used, it is essential to initialize Firebase in the Flutter app. This is done by calling Firebase.initializeApp() in the main entry point of the app (usually in the main.dart file). Firebase needs to be initialized before interacting with any Firebase services, such as the Realtime Database, Cloud Firestore, or Authentication. Connecting Firebase to a Flutter app enables developers to create robust, scalable, and

real-time applications with ease. Firebase's Realtime Database offers a powerful, cloud-based solution for managing data in realtime, while Firebase Authentication ensures secure access control. By integrating Firebase with Flutter, developers can take advantage of real-time data synchronization, offline support, and a wide range of other Firebase features, allowing them to build feature-rich apps that meet modern user expectations.





3 Add Firebase SDK

Instructions for Gradle | Unity C++

Are you still using the `buildscript` syntax to manage plugins? Learn how to add Firebase plugins using that syntax.

1. To make the `google-services.json` config values accessible to Firebase SDKs, you need the Google services Gradle plugin.

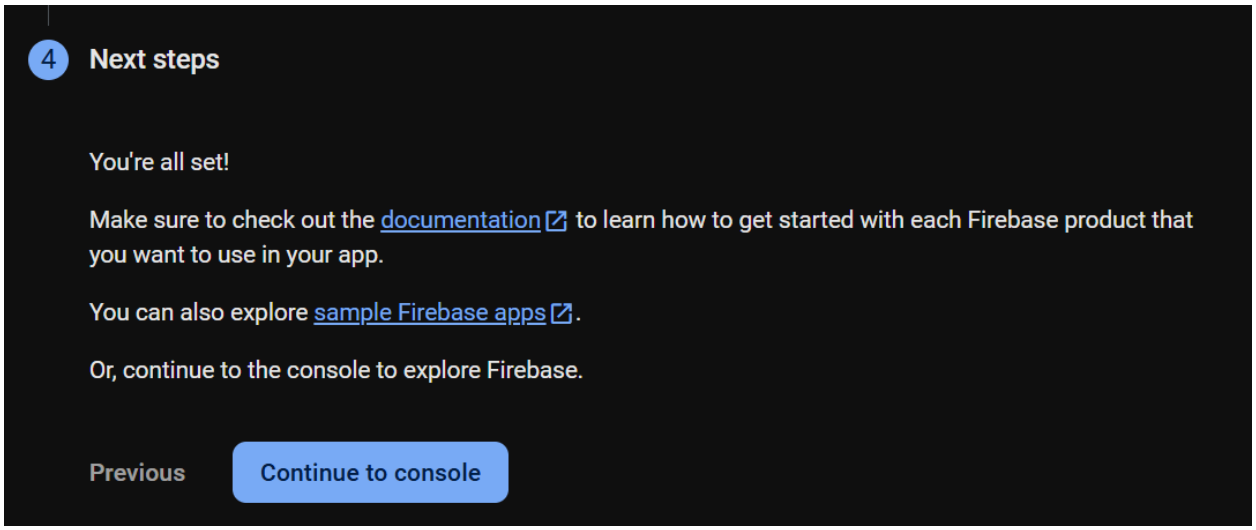
Kotlin DSL (build.gradle.kts) Groovy (build.gradle)

Add the plugin as a dependency to your **project-level** `build.gradle` file:

Root-level (project-level) Gradle file (`<project>/build.gradle`):

```
plugins {
  // ...

  // Add the dependency for the Google services Gradle plugin
  id 'com.google.gms.google-services' version '4.4.2' apply false
}
```

2. Then, in your **module (app-level)** `build.gradle` file, add both the `google-services` plugin and any Firebase SDKs that you want to use in your app:



4 Next steps

You're all set!

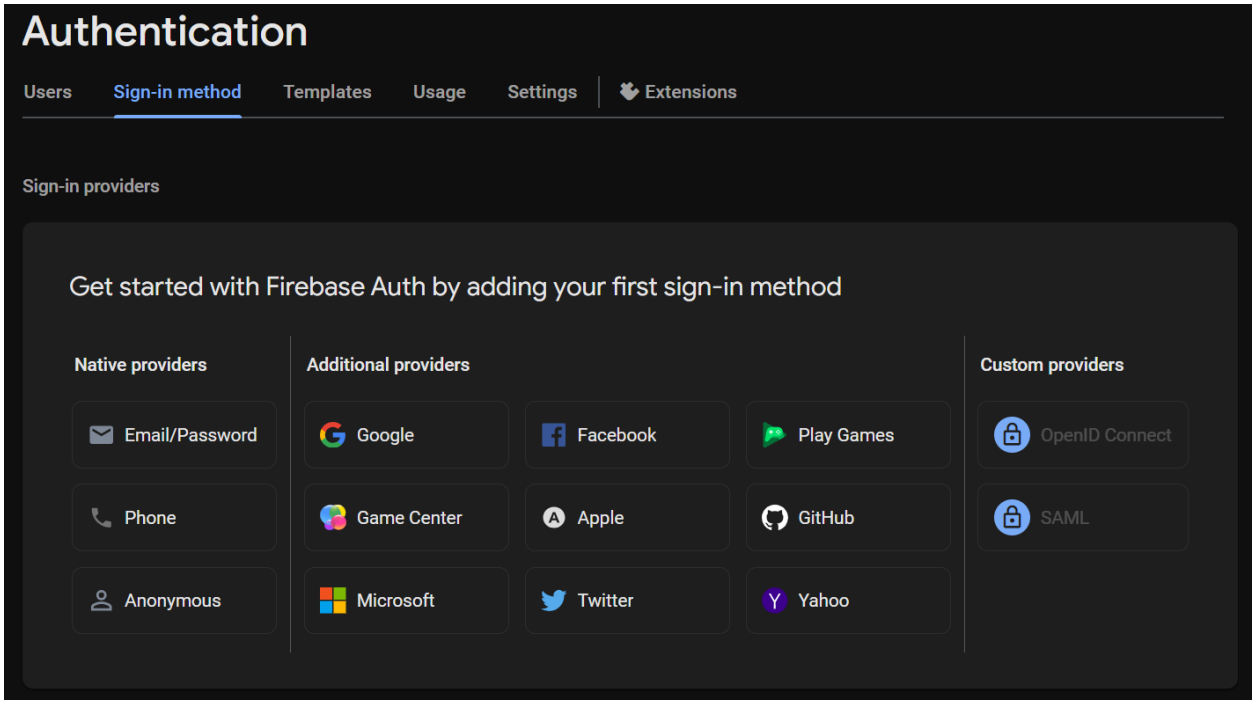
Make sure to check out the documentation to learn how to get started with each Firebase product that you want to use in your app.

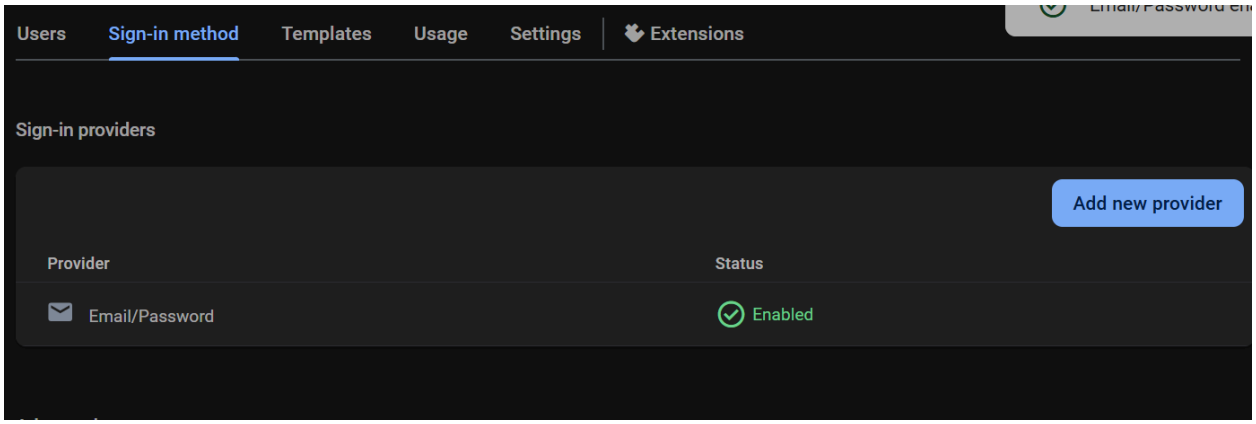
You can also explore sample Firebase apps.

Or, continue to the console to explore Firebase.

Previous Continue to console

Project Title: Behance **Roll No.** 45



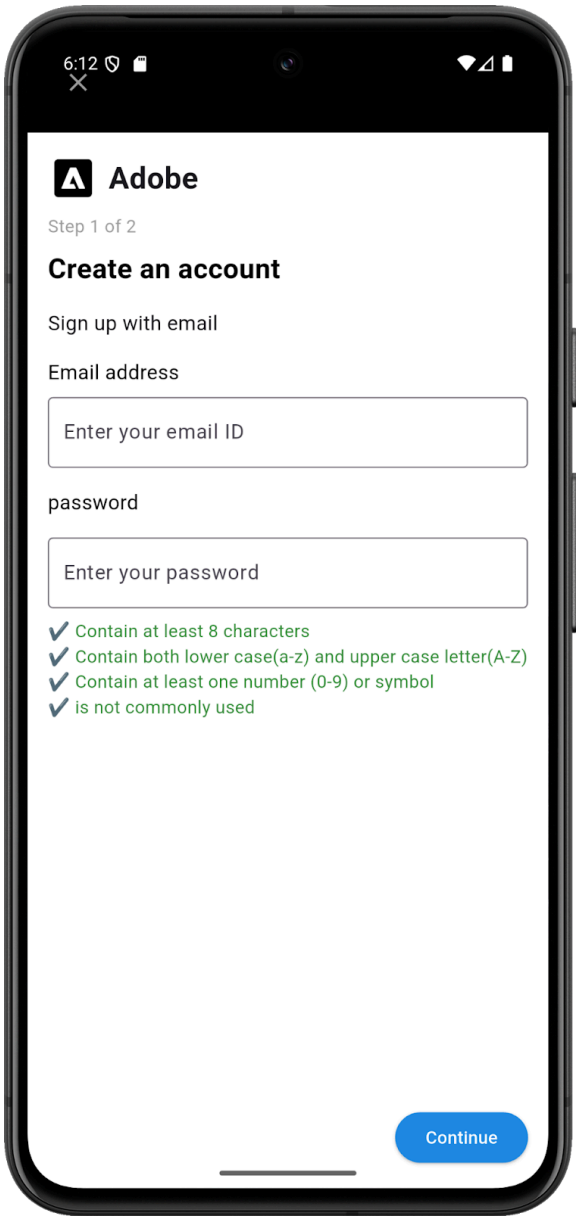


Login.dart

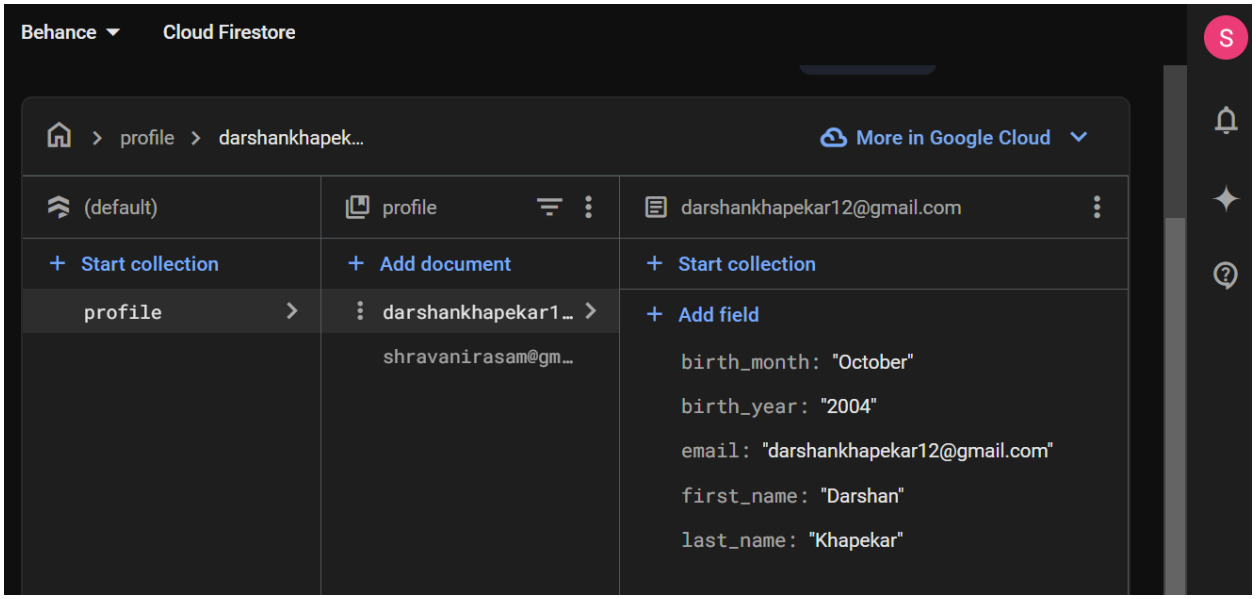
```
import
'package:behnace/Screens/Landing.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import
'package:firebase_auth/firebase_auth.dart';
import '../Widgets/theme.dart';
import '../Widgets/auth_provider.dart';
import  '../Screens/Landing.dart';
// import '../Screens/Profile.dart';

class Login extends StatefulWidget {
  @override
  _LoginState createState() =>
_LoginState();
}

class _LoginState extends State<Login> {
  final TextEditingController _emailController
= TextEditingController();
  final TextEditingController
_passwordController =
TextEditingController();
  bool _isLoading = false;
  String? _errorMessage;

  Future<void> _signInWithEmailPassword()
async {
    setState(() {
      _isLoading = true;
      _errorMessage = null;
    });

    try {
      // Sign in the user
      UserCredential userCredential = await
FirebaseAuth.instance
          .signInWithEmailAndPassword(
        email: _emailController.text.trim(),
        password:
_passwordController.text.trim(),
      );

      // Extract user details correctly
      User? user = userCredential.user;

      if (user != null) {
        // Store email using Provider

Provider.of<AuthProviderFunction>(context,
listen: false).setEmail(
            user.email!);

        // Navigate if login is successful
        Navigator.pushReplacement(
          context,
          MaterialPageRoute(builder: (context)
=> Landing()),
        );
      } else {
        setState(() {
          _errorMessage = "User not found.
Please check your credentials.";
        });
      }
    } catch (e) {
      setState(() {
        _errorMessage = e.toString(); //
Display the actual Firebase error
      });
    } finally {
      setState(() {
        _isLoading = false;
      });
    }
  }



  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: Column(
        crossAxisAlignment:
CrossAxisAlignment.start,
```

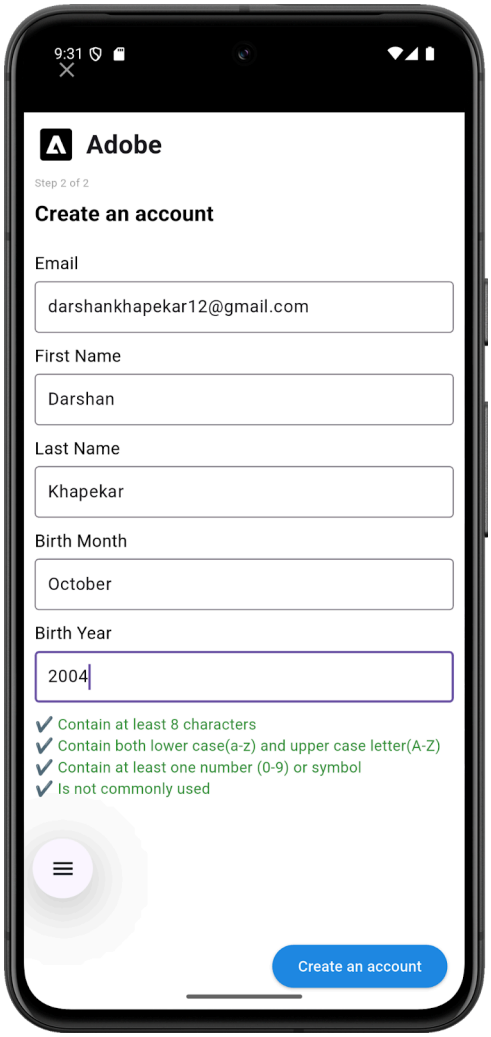
```
        children: [
          Container(
            color: AppColors.background,
            padding: EdgeInsets.all(16),
            child: Row(
              children: [
                IconButton(
                  icon: Icon(Icons.close, color:
AppColors.hint),
                  onPressed: () =>
Navigator.pop(context),
                ),
              ],
            ),
          ),
          Padding(
            padding: const
EdgeInsets.all(16.0),
            child: Column(
              crossAxisAlignment:
CrossAxisAlignment.start,
              children: [
                Row(
                  children: [
                    Icon(Icons.adobe, size: 40,
color: Colors.black),
                    SizedBox(width: 8),
                    Text(
                      'Adobe',
                      style: TextStyle(
                          fontSize: 24, fontWeight:
FontWeight.bold),
                    ),
                  ],
                ),
                SizedBox(height: 8),
                Text('Step 1 of 2',
                    style: TextStyle(color:
AppColors.hint, fontSize: 14)),
                SizedBox(height: 8),
                Text(
                  'Create an account',
                  style:
                  TextStyle(fontSize: 22,
                      fontWeight: FontWeight.bold,
                      color: Colors.black),
                ),
                SizedBox(height: 16),
                Text('Sign up with email', style:
TextStyle(fontSize: 16)),
                SizedBox(height: 16),
                Text('Email address',
                    style: TextStyle(fontSize: 16,
color: Colors.black)),
                SizedBox(height: 8),
                TextField(
                  controller: _emailController,
                  keyboardType:
TextInputType.emailAddress,
                  decoration: InputDecoration(
                    border: OutlineInputBorder(
                        borderSide:
BorderSide(color: AppColors.hint)),
                    hintText: 'Enter your email ID',
                  ),
                ),
                SizedBox(height: 16),
                Text('Password',
                    style: TextStyle(fontSize: 16,
color: Colors.black)),
                SizedBox(height: 8),
                TextField(
                  controller: _passwordController,
                  obscureText: true,
                  decoration: InputDecoration(
                    border: OutlineInputBorder(
                        borderSide:
BorderSide(color: AppColors.hint)),
                    hintText: 'Enter your
password',
                  ),
                ),
                SizedBox(height: 8),
                if (_errorMessage != null)
                  Text("Invalid email or
password",
                      style: TextStyle(color:
Colors.red, fontSize: 14)),
```

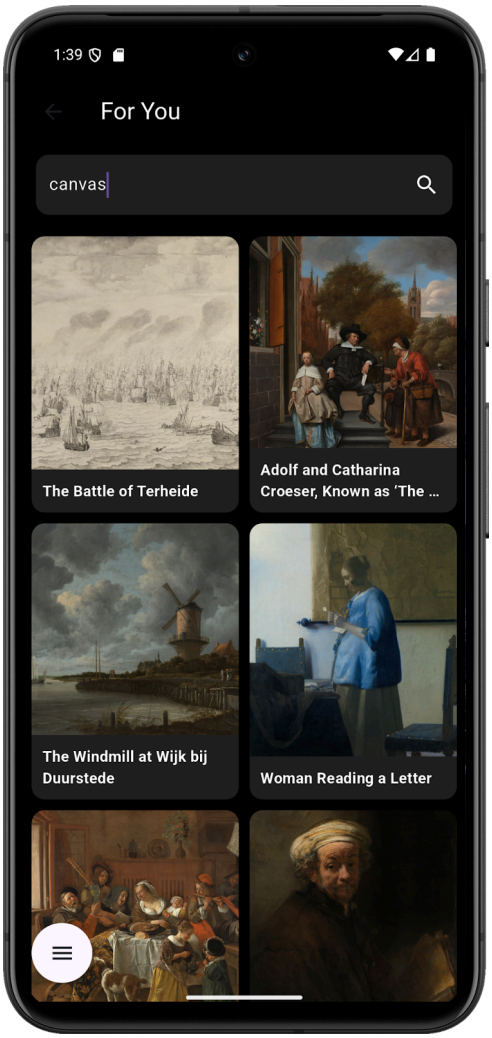
```
              SizedBox(height: 16),
              Text('✓ Contain at least 8
characters',
                  style: TextStyle(color:
Colors.green[700], fontSize: 14)),
              Text('✓ Contain both lower and
upper case letters',
                  style: TextStyle(color:
Colors.green[700], fontSize: 14)),
              Text('✓ Contain at least one
number (0-9) or symbol',
                  style: TextStyle(color:
Colors.green[700], fontSize: 14)),
              Text('✓ Is not commonly used',
                  style: TextStyle(color:
Colors.green[700], fontSize: 14)),
            ],
          ),
        ),
        Spacer(),
        Padding(
          padding: const
EdgeInsets.all(16.0),
          child: Align(
            alignment:
Alignment.bottomRight,
            child: ElevatedButton(
              style: ElevatedButton.styleFrom(
                backgroundColor:
AppColors.primary,
                foregroundColor: Colors.white,
              ),
              onPressed: _isLoading ? null :
_signInWithEmailPassword,
              child: _isLoading
                  ?
CircularProgressIndicator(color:
Colors.white)
                  : Text('Continue'),
            ),
          ),
        ),
      ],
    ),
  );
 }
}
```



Project Title: Behance **Roll No.** 45







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.	45
Name	Shravani Rasam
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

Online Reference:
https://developer.mozilla.org/en-US/docs/Web/Manifest
https://www.geeksforgeeks.org/making-a-simple-pwa-under-5-minutes/

Theory:-

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

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

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

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

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

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

5. Updated Real-Time Data Access
Another plus point of PWAs is that these apps get updated on their own. They do not demand the end-users to go to the App Store or other such platforms to download the update and wait until installed.
In this app type, the web app developers can push the live update from the server, which reaches the apps residing on the user's devices automatically. Therefore, it is easier for the mobile app developer to provide the best of the updated functionalities and services to the end-users without forcing them to update their app.
6. Discoverable
PWAs reside in web browsers. This implies higher chances of optimizing them as per the Search Engine Optimization (SEO) criteria and improving the Google rankings like that in websites and other web apps.

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

Pros and cons of the Progressive Web App
The main features are:

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

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

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

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

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

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

Reactivable — Make it easy to reactivate the application thanks to capabilities such as web notifications.
Installable — They allow the user to “save” the apps that he considers most useful with the corresponding icon on the screen of his mobile terminal (home screen) without having to face all the steps and problems related to the use of the app store.

Linkable — Easily shared via URL without complex installations.

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

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

Lack of presence on the stores (there is no possibility to acquire traffic from that channel);
There is no “body” of control (like the stores) and an approval process;
Limited access to some hardware components of the devices;
Little flexibility regarding “special” content for users (eg loyalty programs, loyalty, etc.).

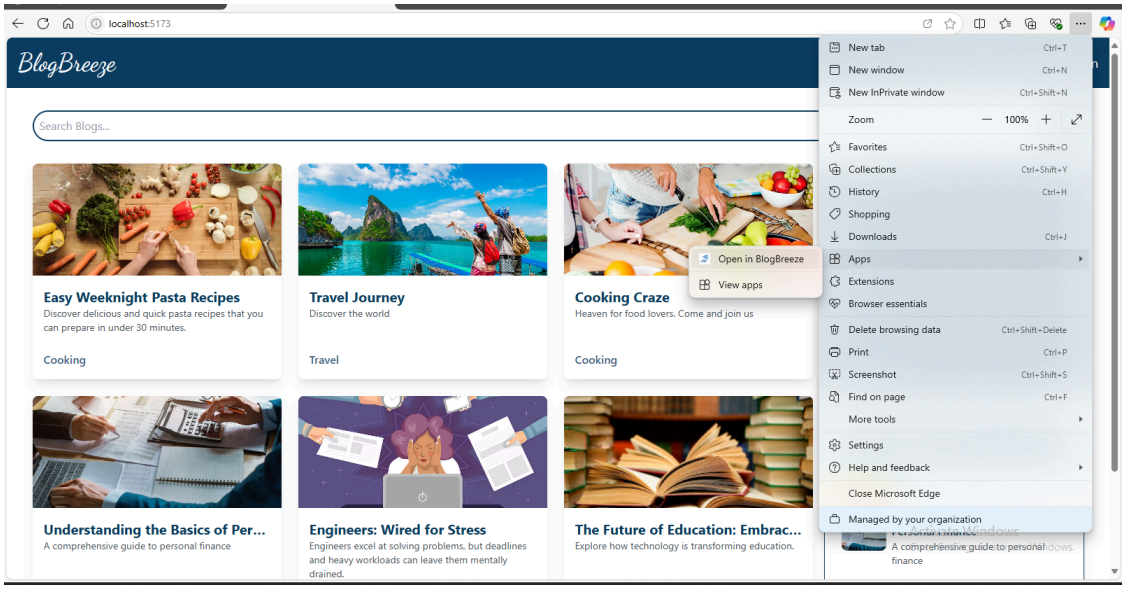
Code:-
manifest.json:-

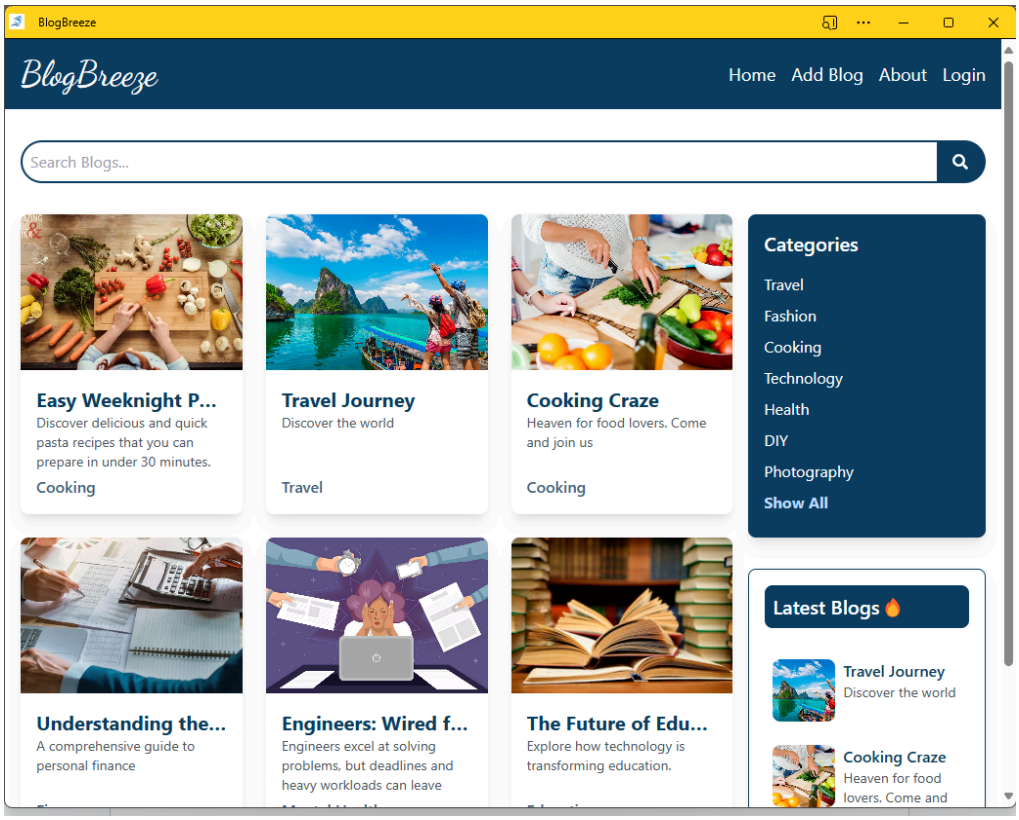
```
{
   "name": "BlogBreeze",
   "short_name": "BlogBreeze",
   "start_url": "ani.html",
   "scope": "./",
   "icons": [
      {
       "src": "/src/assets/react.svg",
       "sizes": "192x192",
       "type": "image/svg"
      }
    ],
    "theme_color": "#ffd31d",
    "background_color": "#333",
   "display": "standalone"
}
```

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

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="manifest" href="manifest.json">
    <link rel="icon" type="image/svg+xml" href="./public/icon.png" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>BlogBreeze</title>

    <!-- Google Fonts Dancing Script -->
    <link href="https://fonts.googleapis.com/css2?family=Dancing+Script:wght@400;500;600;700&display=swap" rel="stylesheet">
<!-- Add this in the <head> section of your index.html -->
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css" />

  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
```

Output:-





Conclusion:-

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

MAD & PWA Lab
Journal

Experiment No.	08
Experiment Title.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA
Roll No.	45
Name	Shravani Rasam
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Theory:

Service Worker

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

Things to note about Service Worker:

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

What can we do with Service Workers?

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

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

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

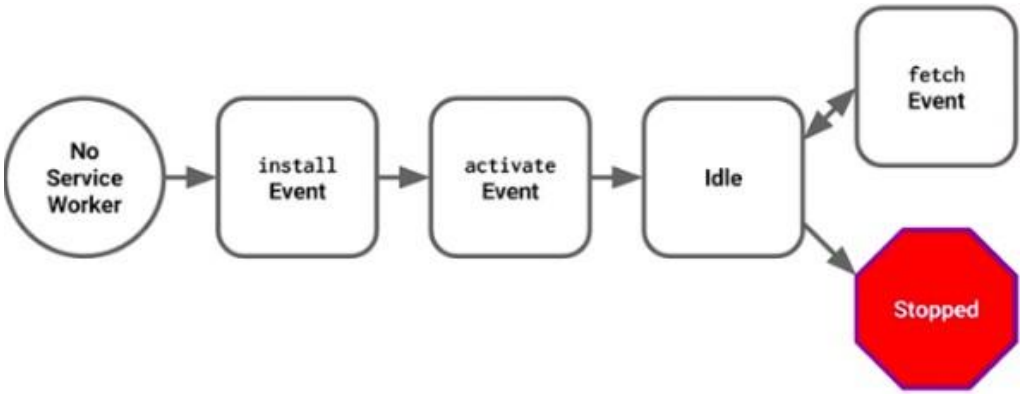
Background Sync of Service Worker.

What can't we do with Service Workers?

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

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

Service Worker Cycle



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

- Registration
- Installation
- Activation

Registration
To install a service worker, you need to register it in your main JavaScript code

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

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

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

You can also set an arbitrary scope by passing in an additional parameter when

registering. For example: main.js

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

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

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

main.js

```
// Service Worker Script

const CACHE_NAME = 'blogbreeze-v1'; // Cache name to identify the version of
cached content
const urlsToCache = [
  '/', // Home page
  'index.html', // Main HTML file
  'ani.html', // Any additional pages you want to cache
  '/src/assets/react.svg', // App icon
  '/public/icon.png', // Favicon
  '/src/main.jsx', // Main JS file for app functionality
//   'https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css' //
FontAwesome for icons
];

// Install Service Worker
self.addEventListener('install', (event) => {
  console.log('Service Worker: Installed');
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => {
        return cache.addAll(urlsToCache);
      })
  );
});

// Activate Service Worker
self.addEventListener('activate', (event) => {
  console.log('Service Worker: Activated');
  // Remove old caches if there are any
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cacheName) => {
          if (cacheName !== CACHE_NAME) {
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
});

// Fetch event: Intercept network requests and serve cached content
self.addEventListener('fetch', (event) => {
  console.log('Service Worker: Fetching', event.request.url);
```

```
  event.respondWith(
    caches.match(event.request)
      .then((cachedResponse) => {
        // Return cached content if found, otherwise fetch from network
        return cachedResponse || fetch(event.request);
      })
  );
})

// Service Worker Script

const CACHE_NAME = 'blogbreeze-v1'; // Cache name to identify the version of
cached content
const urlsToCache = [
  '/', // Home page
  'index.html', // Main HTML file
  'ani.html', // Any additional pages you want to cache
  '/src/assets/react.svg', // App icon
  '/public/icon.png', // Favicon
  '/src/main.jsx', // Main JS file for app functionality
//   'https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css' //
FontAwesome for icons
];

// Install Service Worker
self.addEventListener('install', (event) => {
  console.log('Service Worker: Installed');
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => {
        return cache.addAll(urlsToCache);
      })
  );
});

// Activate Service Worker
self.addEventListener('activate', (event) => {
  console.log('Service Worker: Activated');
  // Remove old caches if there are any
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cacheName) => {
          if (cacheName !== CACHE_NAME) {
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
```

```
});

// Fetch event: Intercept network requests and serve cached content
self.addEventListener('fetch', (event) => {
  console.log('Service Worker: Fetching', event.request.url);
  event.respondWith(
    caches.match(event.request)
      .then((cachedResponse) => {
        // Return cached content if found, otherwise fetch from network
        return cachedResponse || fetch(event.request);
      })
  );
});

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

Installation

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

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

service-worker.js

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

Activation

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

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

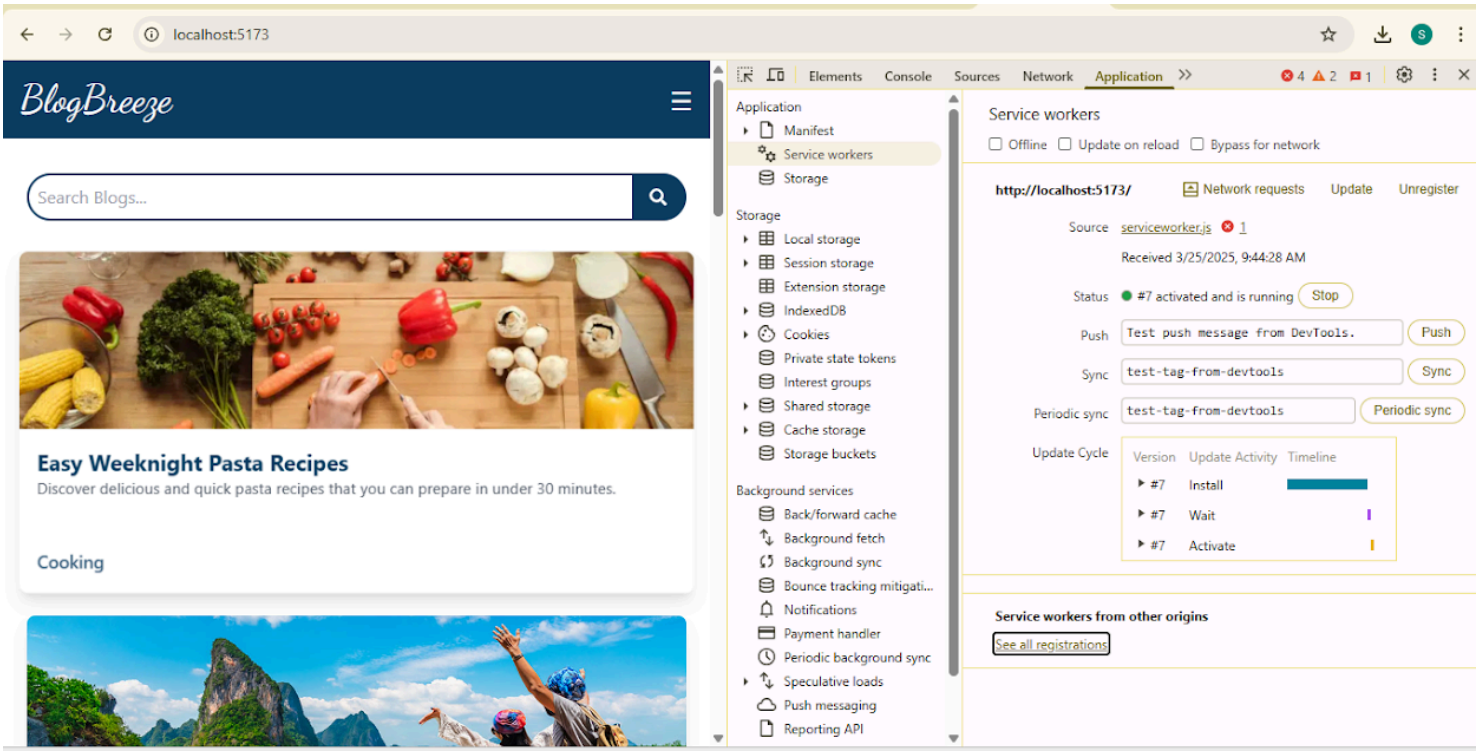
Once activated, the service worker controls all pages that load within its scope, and starts listening for events from those pages. However, pages in your app that were loaded before the service worker activation will not be under service worker control. The new service worker will only take over when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

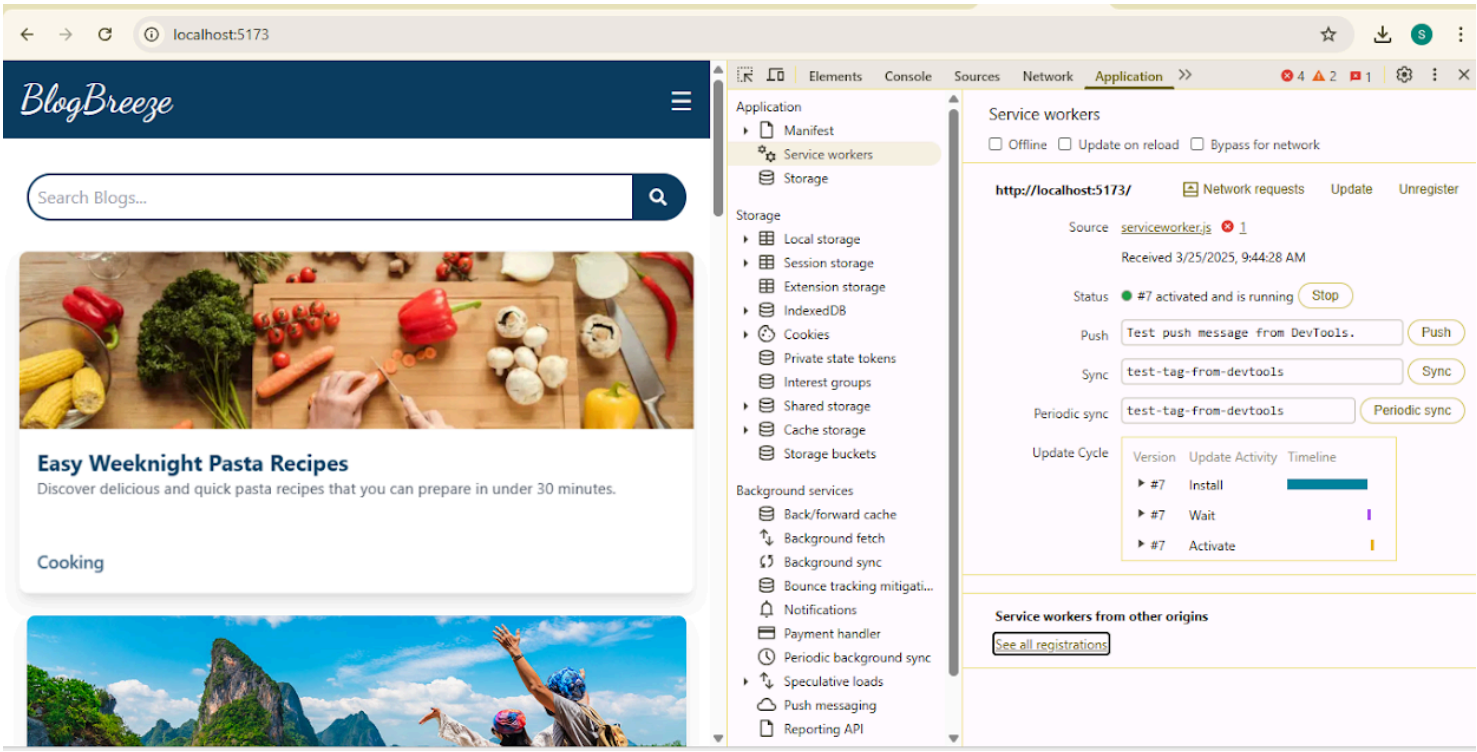
Code :

```
// Service Worker Script

const CACHE_NAME = 'blogbreeze-v1'; // Cache name to identify the version of cached content
const urlsToCache = [
  '/', // Home page
  'index.html', // Main HTML file
  'ani.html', // Any additional pages you want to cache
  '/src/assets/react.svg', // App icon
  '/public/icon.png', // Favicon
  '/src/main.jsx', // Main JS file for app functionality
//            'https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css'    // FontAwesome for icons
];

// Install Service Worker
self.addEventListener('install', (event) => {
  console.log('Service Worker: Installed');
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => {
        return cache.addAll(urlsToCache);
      })
  );
});

// Activate Service Worker
self.addEventListener('activate', (event) => {
  console.log('Service Worker: Activated');
  // Remove old caches if there are any
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cacheName) => {
          if (cacheName !== CACHE_NAME) {
```

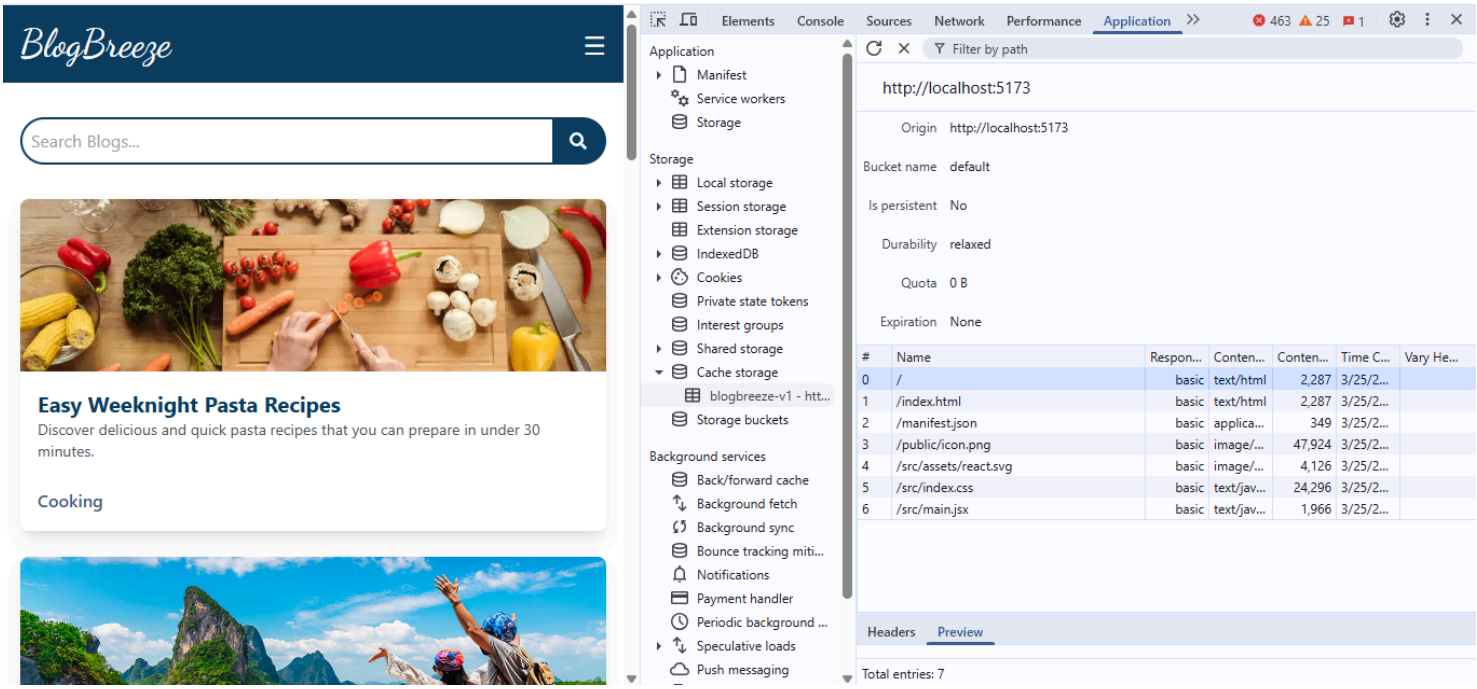
```
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
});

// Fetch event: Intercept network requests and serve cached content
self.addEventListener('fetch', (event) => {
  console.log('Service Worker: Fetching', event.request.url);
  event.respondWith(
    caches.match(event.request)
      .then((cachedResponse) => {
        // Return cached content if found, otherwise fetch from network
        return cachedResponse || fetch(event.request);
      })
  );
});
```

Project Title: Behance **Roll No.** 45





Project Title: Behance **Roll No.** 45



Conclusion : Thus we have learnt to code and register a service worker, and complete the install and activation process for a new service worker for the PWA.

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.	45
Name	Shravani Rasam
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

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

SERVICE WORKER FOR PUSH NOTIFICATIONS :

```
const CACHE_NAME = 'blogbreeze-v1';
const filesToCache = [
  '/',
  'index.html',
  'manifest.json',
  '/src/assets/react.svg',
  '/public/icon.png',
  '/src/index.css',
  '/src/main.jsx',
];

// Install event: Caching important files
self.addEventListener('install', (event) => {
  console.log('Service Worker: Installed');
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => {
        console.log('Service Worker: Caching files');
        return cache.addAll(filesToCache);
      })
  );
});
```

```
// Activate event: Cleaning up old caches
self.addEventListener('activate', (event) => {
  console.log('Service Worker: Activated');
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cacheName) => {
          if (cacheName !== CACHE_NAME) {
            console.log('Service Worker: Deleting old cache', cacheName);
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
});

// Fetch event: Intercepting requests and serving from cache if available
self.addEventListener('fetch', (event) => {
  console.log('Service Worker: Fetching', event.request.url);

  event.respondWith(
    caches.match(event.request)
      .then((cachedResponse) => {
        if (cachedResponse) {
          console.log('Service Worker: Returning cached response');
          return cachedResponse; // Return the cached response if available
        }

        return fetch(event.request)
          .then((networkResponse) => {
```

```
          caches.open(CACHE_NAME).then((cache) => {
            console.log('Service Worker: Caching new response for', event.request.url);
            cache.put(event.request, networkResponse.clone()); // Cache the new response
          });
          return networkResponse;
        });
    })
  );
});

self.addEventListener('push', (event) => {
  console.log('Push notification received:', event);

  if (event && event.data) {
    // Parse the push notification data
    const data = event.data.json();

    if (data.method === 'pushMessage') {
      console.log('Push notification sent with message:', data.message);

      // Set up notification options (like body, icon, etc.)
      const options = {
        body: data.message || 'Hello, this is a default message!', // Default or push message
        icon: '/src/assets/react.svg', // Icon for the notification
        badge: '/src/assets/react.svg', // Badge icon for the notification
      };

      // Check if the notification permission is granted before showing the notification
      if (Notification.permission === 'granted') {
        // Show the notification with a title
        event.waitUntil(
          self.registration.showNotification('Blog Breeze', options)
```

```
        );
      } else {
        console.log('Notification permission not granted yet.');
      }
    }
  }
});
```

SERVICE WORKER SCRIPT FOR SYNC AND FETCH :

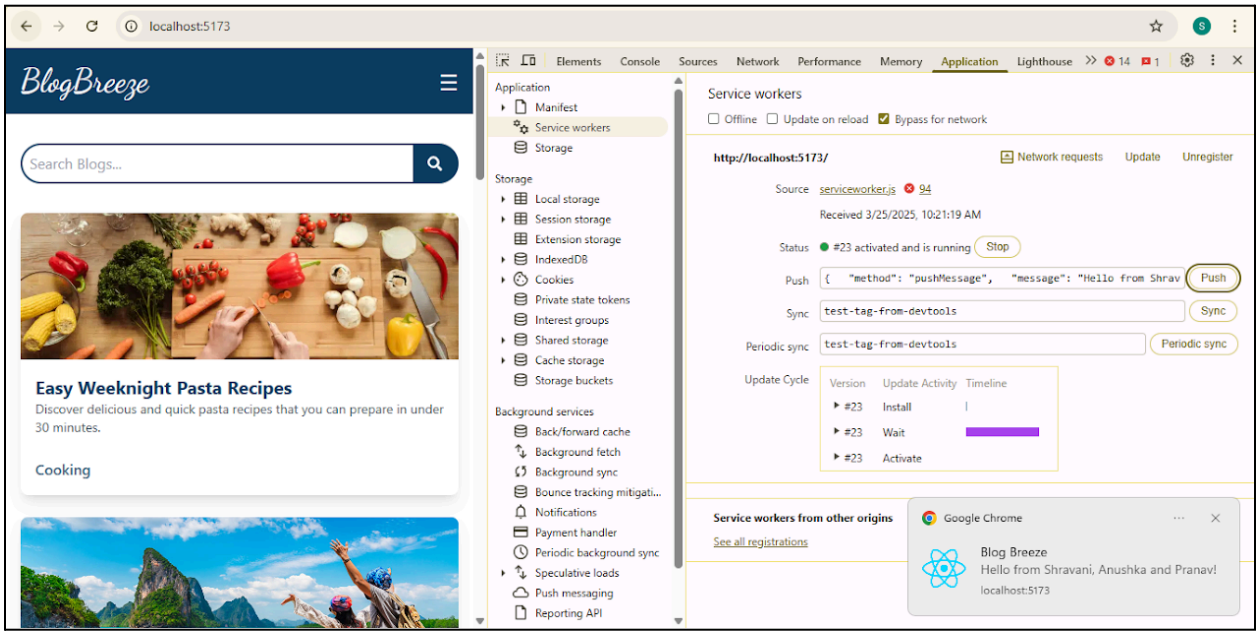
```
const CACHE_NAME = 'blogbreeze-v1';
const filesToCache = [
 '/',
 'index.html',
 'manifest.json',
 '/src/assets/react.svg',
 '/public/icon.png',
 '/src/index.css',
 '/src/main.jsx',

];

// Install event: Caching important files
self.addEventListener('install', (event) => {
 console.log('Service Worker: Installed');
 event.waitUntil(
  caches.open(CACHE_NAME)
    .then((cache) => {
      console.log('Service Worker: Caching files');
      return cache.addAll(filesToCache);
    })
```

```
  );
});

// Activate event: Cleaning up old caches
self.addEventListener('activate', (event) => {
  console.log('Service Worker: Activated');
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cacheName) => {
          if (cacheName !== CACHE_NAME) {
            console.log('Service Worker: Deleting old cache', cacheName);
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
});

// Fetch event: Intercepting requests and serving from cache if available
self.addEventListener('fetch', function (event) {
  console.log('Service Worker: Fetching', event.request.url);

  // Skip caching for Firebase API requests
  if (event.request.url.includes('firestore.googleapis.com')) {
    // Always fetch Firebase data from the network (no cache)
    event.respondWith(fetch(event.request));
    return;
  }

  event.respondWith(
```

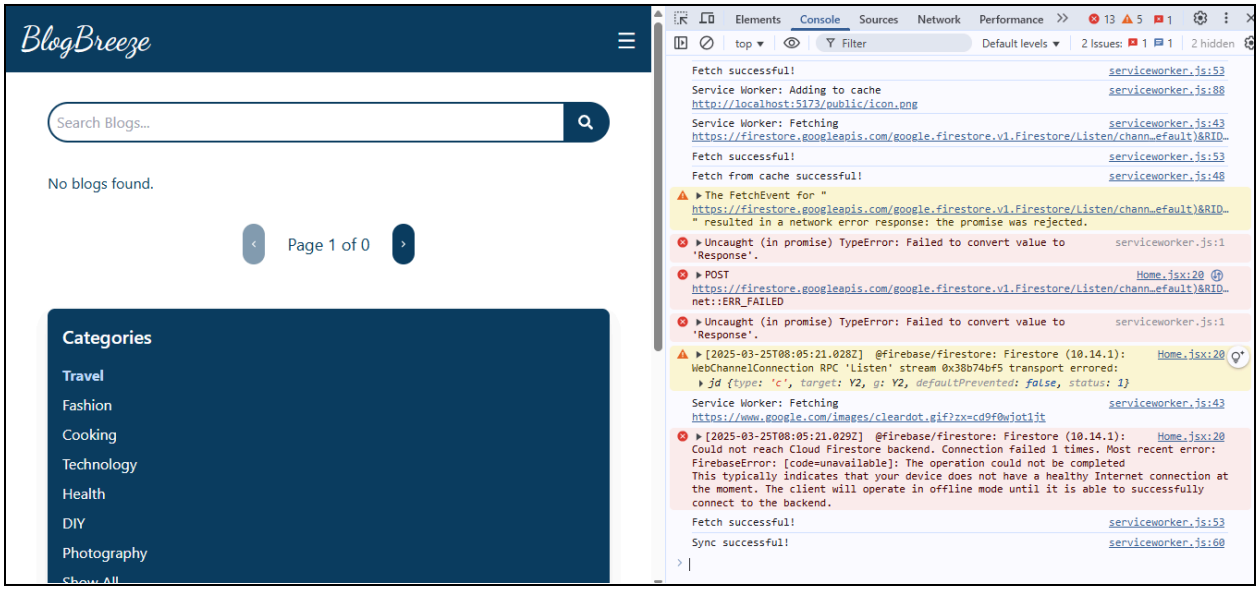
```
    checkResponse(event.request)
      .catch(function () {
        console.log('Fetch from cache successful!');
        return returnFromCache(event.request);
      })
  );

  console.log('Fetch successful!');
  event.waitUntil(addToCache(event.request));
});

// Sync event: Handling background sync
self.addEventListener('sync', function (event) {
  if (event.tag === 'syncMessage') {
    console.log('Sync successful!');
    // You can add more background sync logic here
  }
});

// Placeholder functions for cache and response handling
function checkResponse(request) {
  return caches.match(request)
    .then(function (cachedResponse) {
      if (cachedResponse) {
        return cachedResponse;
      }
      return fetch(request);
    });
}

function returnFromCache(request) {
  return caches.match(request);
```

```
}

function addToCache(request) {
  return fetch(request)
    .then(function (response) {
      if (!response || response.status !== 200 || response.type !== 'basic') {
        return response;
      }
      return caches.open(CACHE_NAME)
        .then(function (cache) {
          console.log('Service Worker: Adding to cache', request.url);
          cache.put(request, response);
          return response;
        });
    });
}
```

OUTPUT :



Project Title: Behance **Roll No.** 45



Conclusion : Thus we learnt to implement service worker events like fetch, sync and push for PWA.

MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	45
Name	Shravani Rasam
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Theory:

GitHub Pages

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

GitHub Pages provides the following key features:

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

Reasons for favoring this over Firebase:

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

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

Pros

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

Cons

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

Firebase

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

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

Pros

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

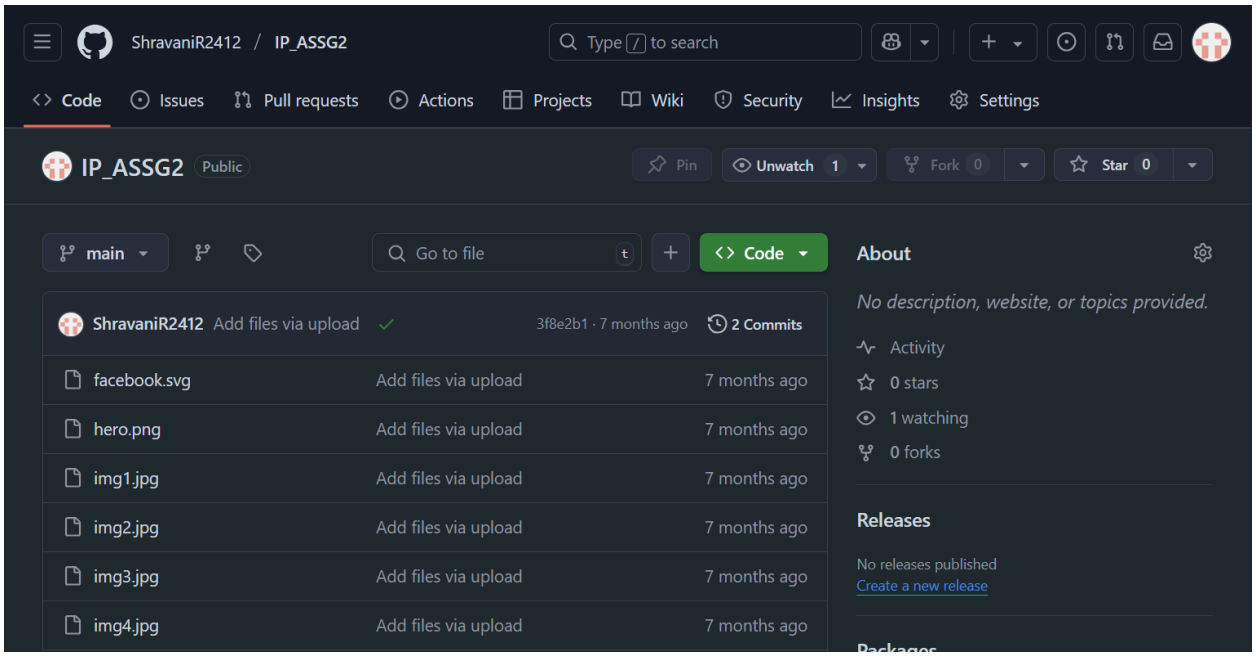
Cons

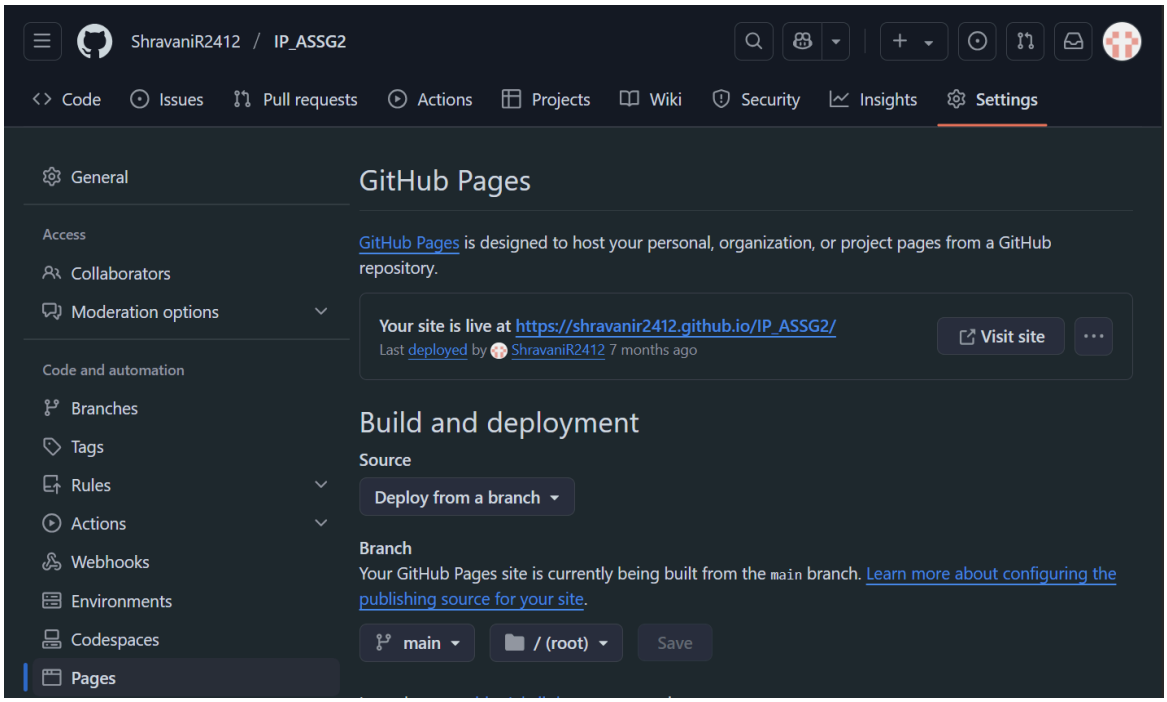
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator.

Project Title: Behance **Roll No.** 45

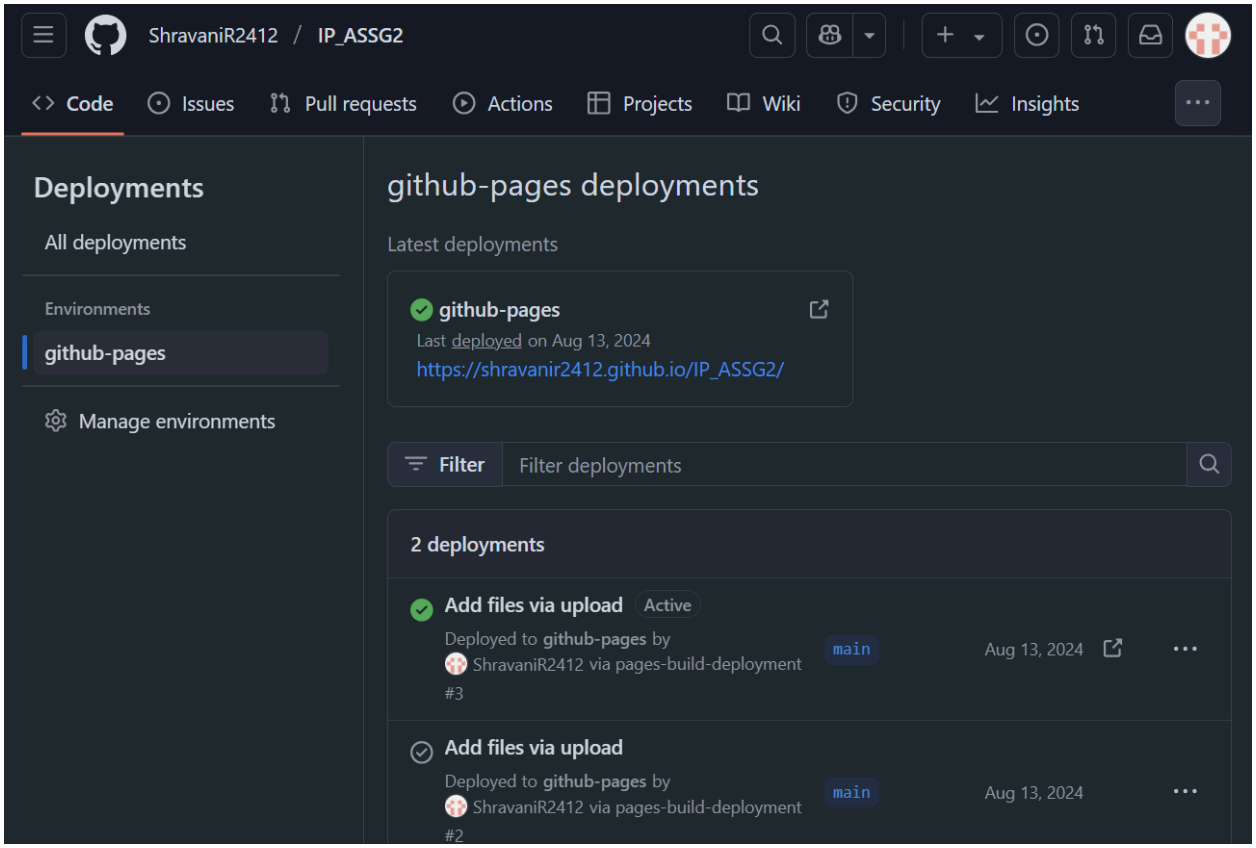
Link to our GitHub repository:
https://github.com/ShravaniR2412/IP_ASSG2

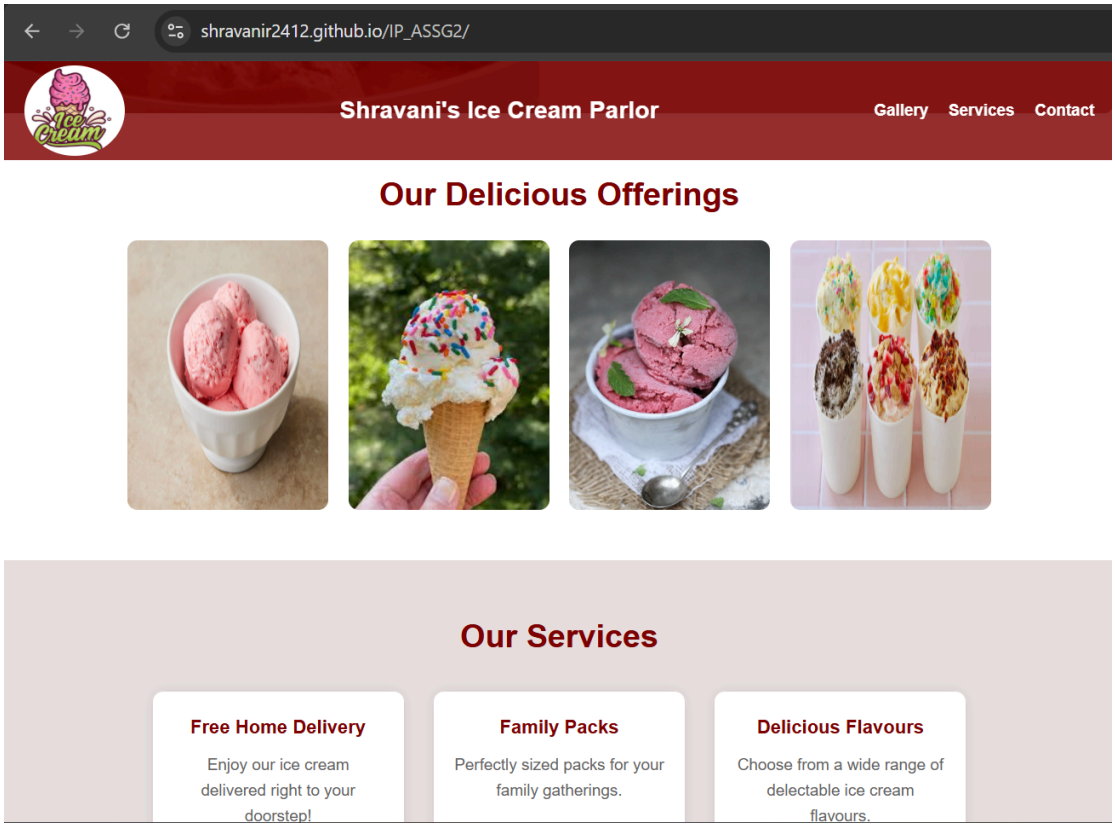
Github Screenshot:





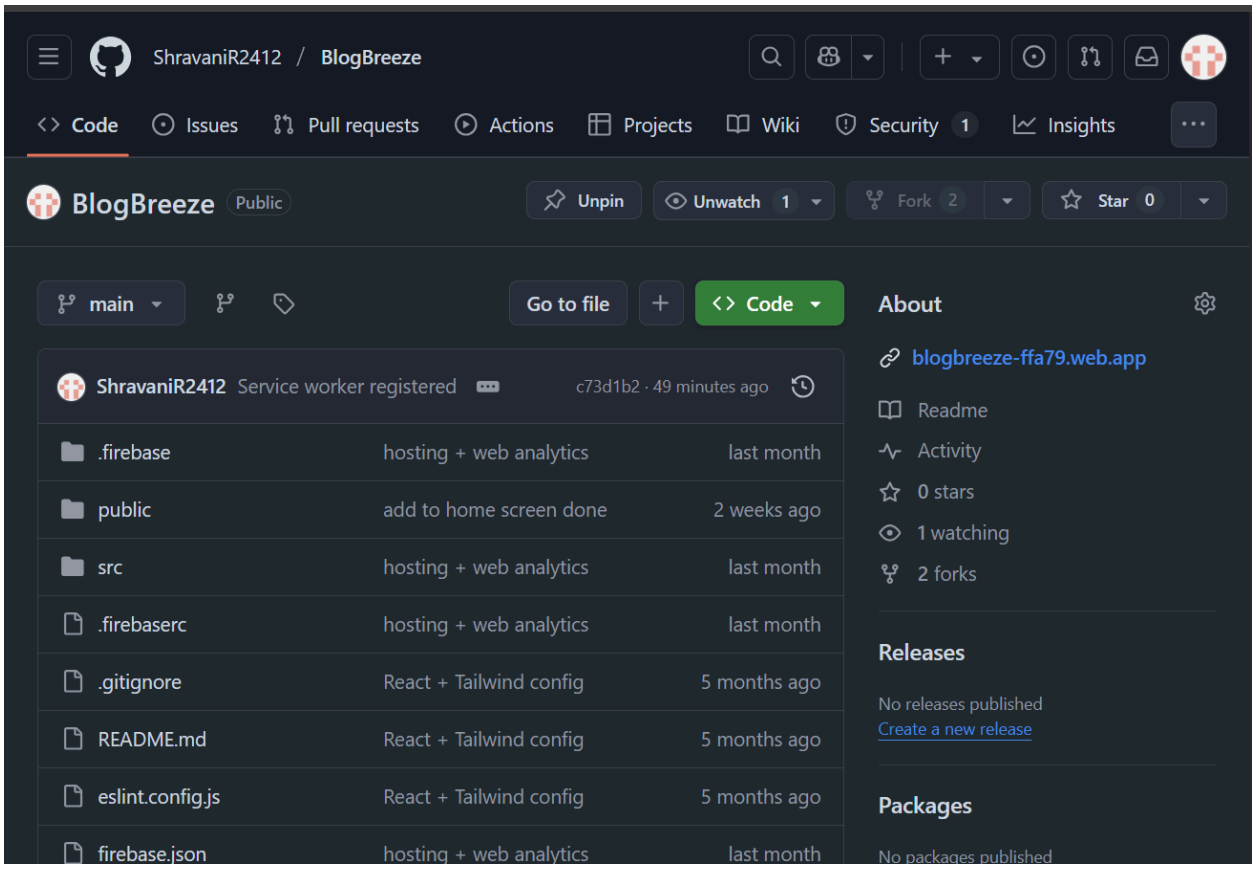
Project Title: Behance **Roll No.** 45





Deployed Link: https://shravanir2412.github.io/IP_ASSG2/

PWA Website:



Github Link: https://github.com/ShravaniR2412/BlogBreeze

Deployed Link: blogbreeze-ffa79.web.app

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	45
Name	Shravani Rasam
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

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.

Key Features and Audit Metrics

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

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

3. **Accessibility:** As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the ‘aria-’ attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics though, Accessibility metrics score on a pass/fail basis i.e. if all possible elements of the page are not screen-reader friendly (HTML5 introduced features that would make pages easy to interpret for screen readers used by visually challenged people like tag names, tags such as <section>, <article>, etc.), you get a 0 on that score. The aggregate of these scores is your Accessibility metric score.

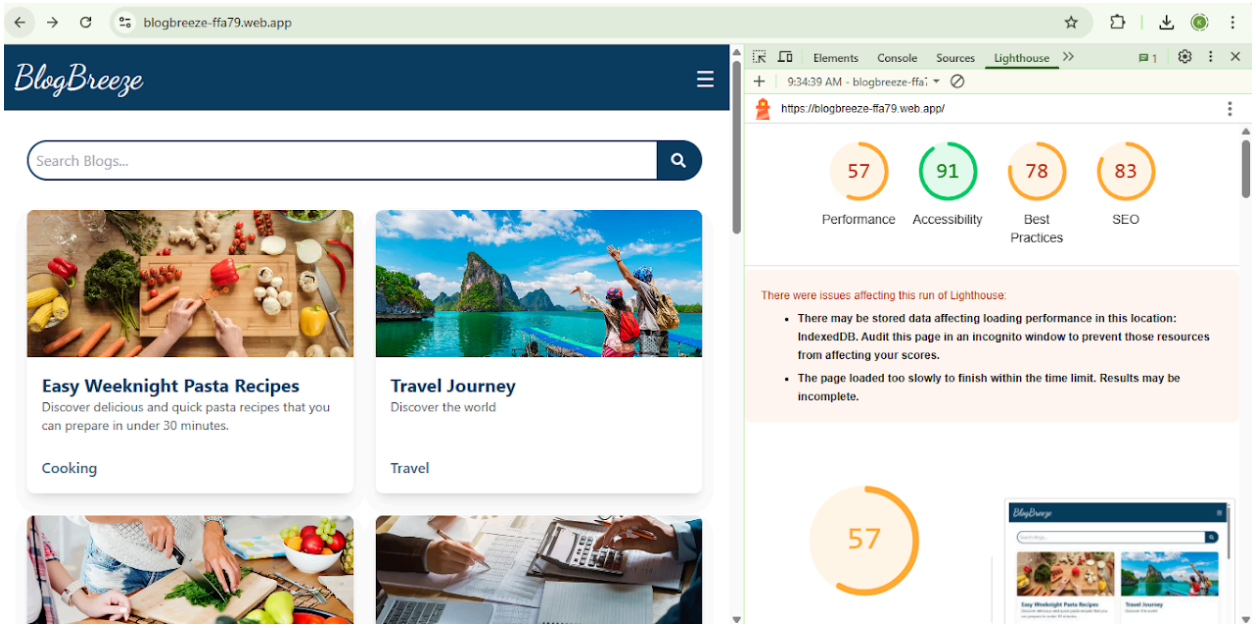
4. **Best Practices:** As any developer would know, there are a number of practices that have been deemed ‘best’ based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS

Avoiding the use of deprecated code elements like tags, directives, libraries, etc. Password input with paste-into disabled

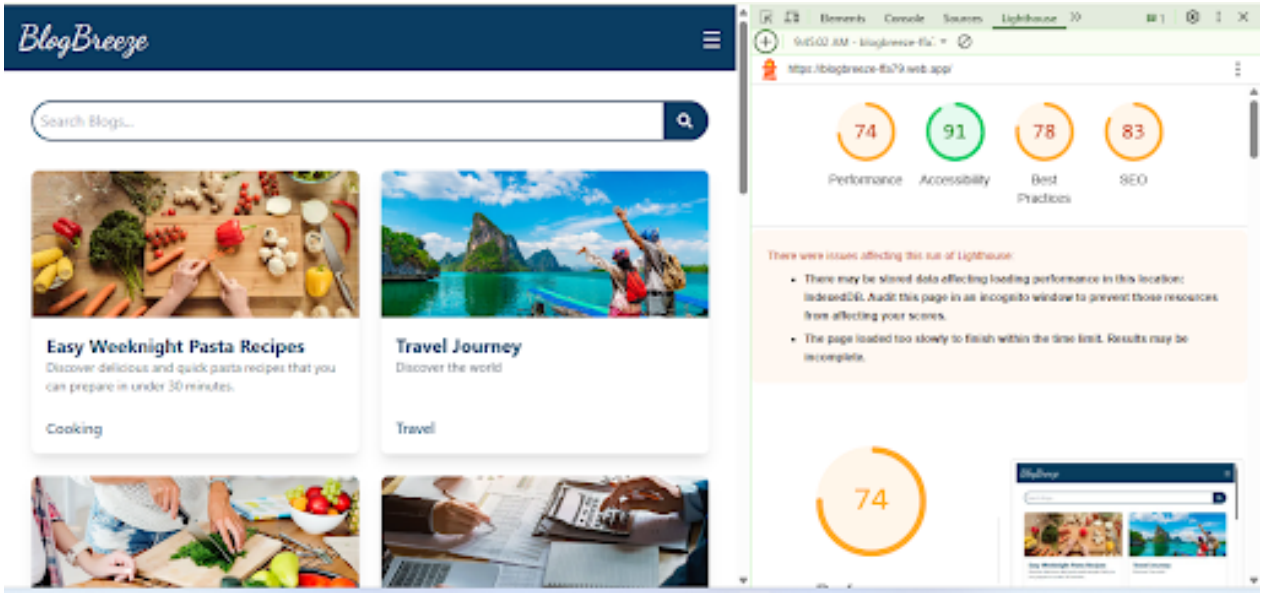
Geo-Location and cookie usage alerts on load, etc.

Project Title: Behance **Roll No.** 45

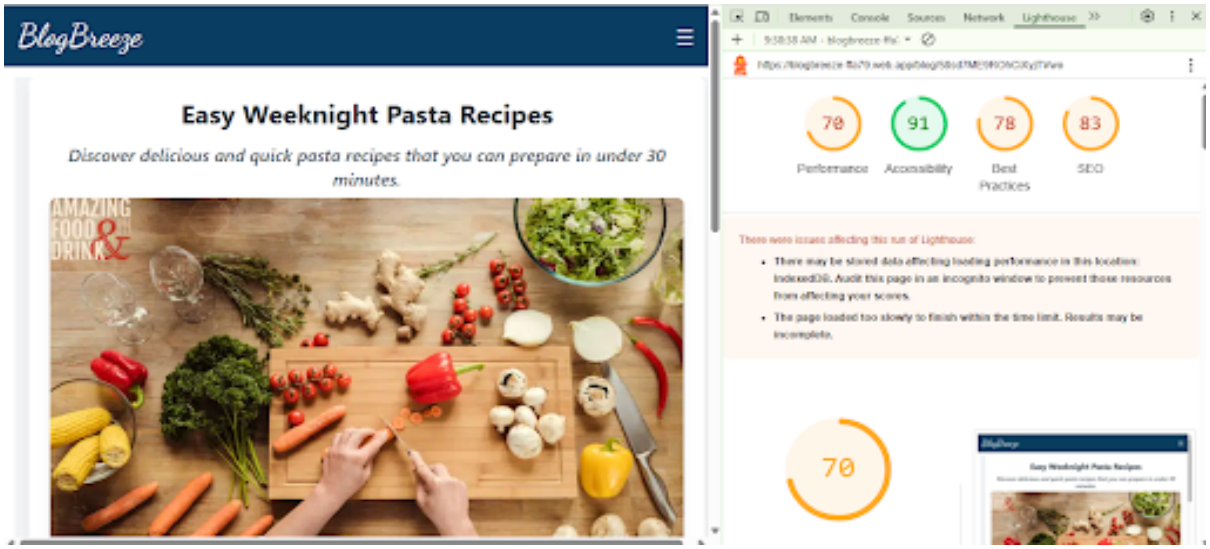
Performance before changes

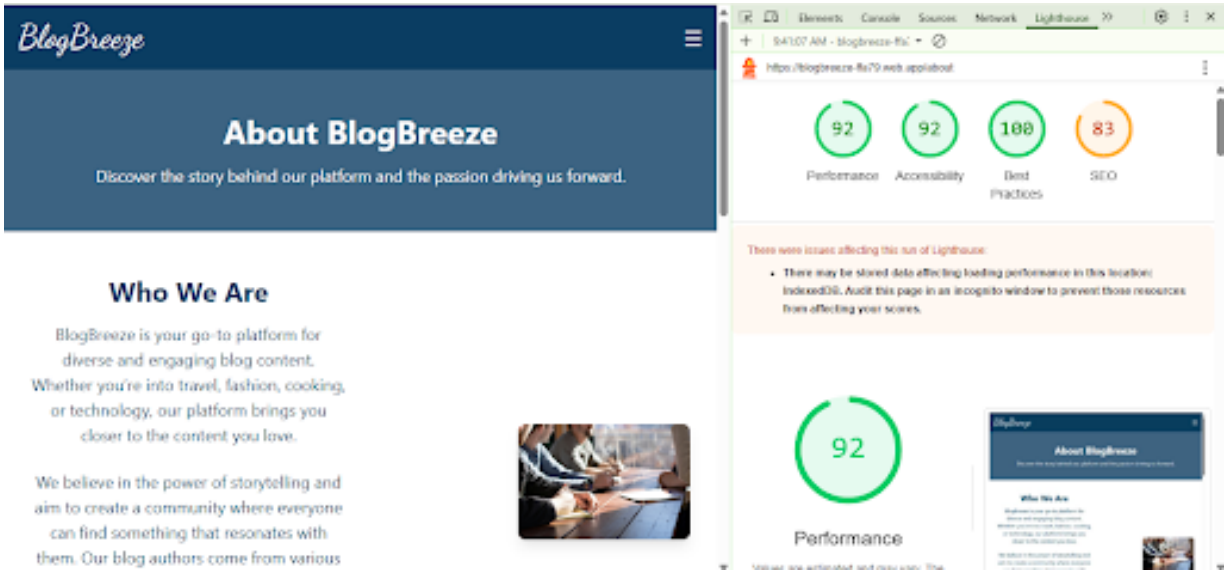


Performance after changes



Project Title: Behance **Roll No.** 45





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