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 **Arnav Santosh Sawant** of **D15A** semester **VI,** has successfully completed necessary experiments in the **MAD & PWA Lab** under my supervision in **VES Institute of Technology** during the academic year **2024-2025.**

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Name of the Course : MAD & PWA Lab

Course Code : ITL604

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

Faculty Incharge : Mrs. Kajal Joseph.

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

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

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: Inshorts clone app **Roll No. 52**

Lab Objectives:

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

Lab Outcomes:

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

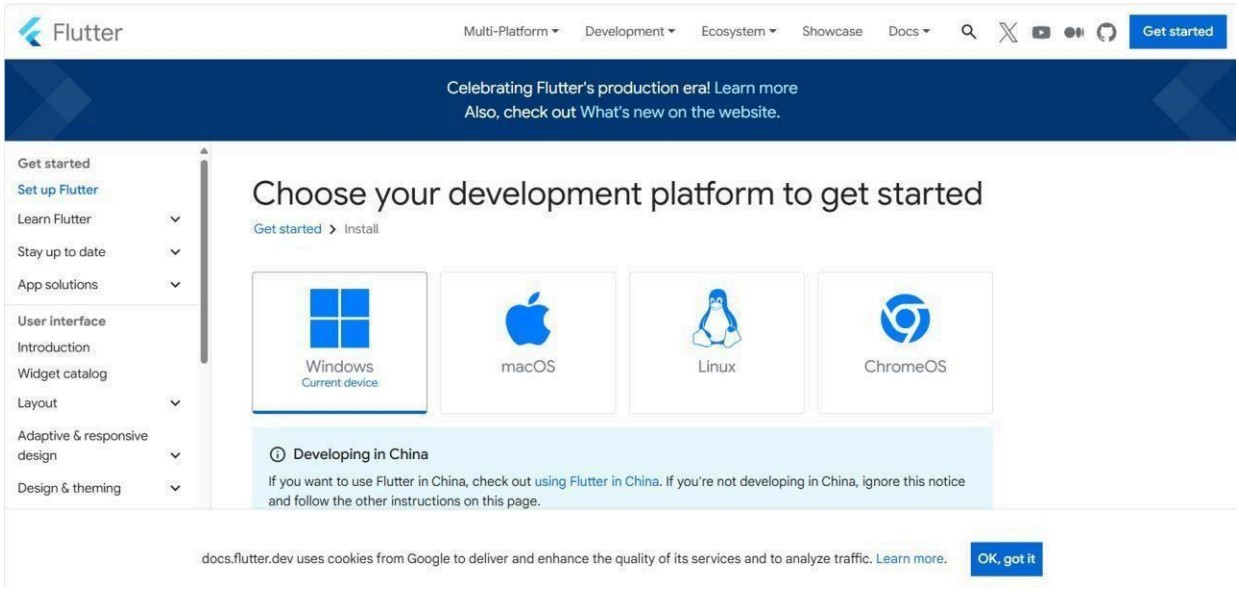
Index

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

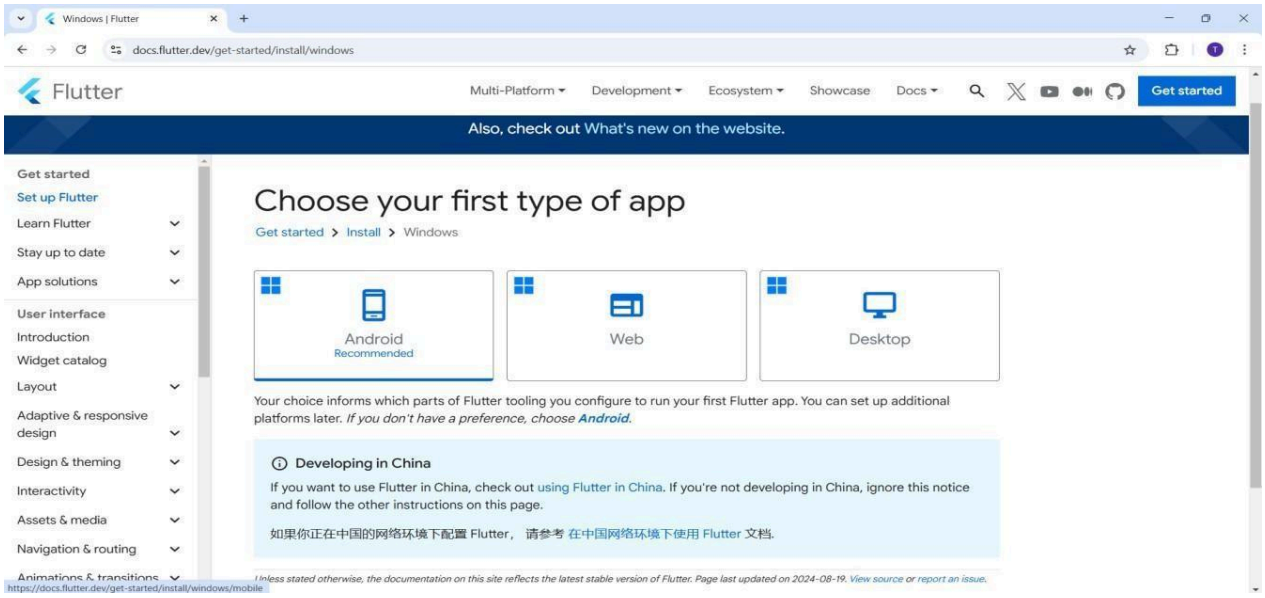
MAD & PWA Lab
Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	52
Name	Arnav Santosh Sawant
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

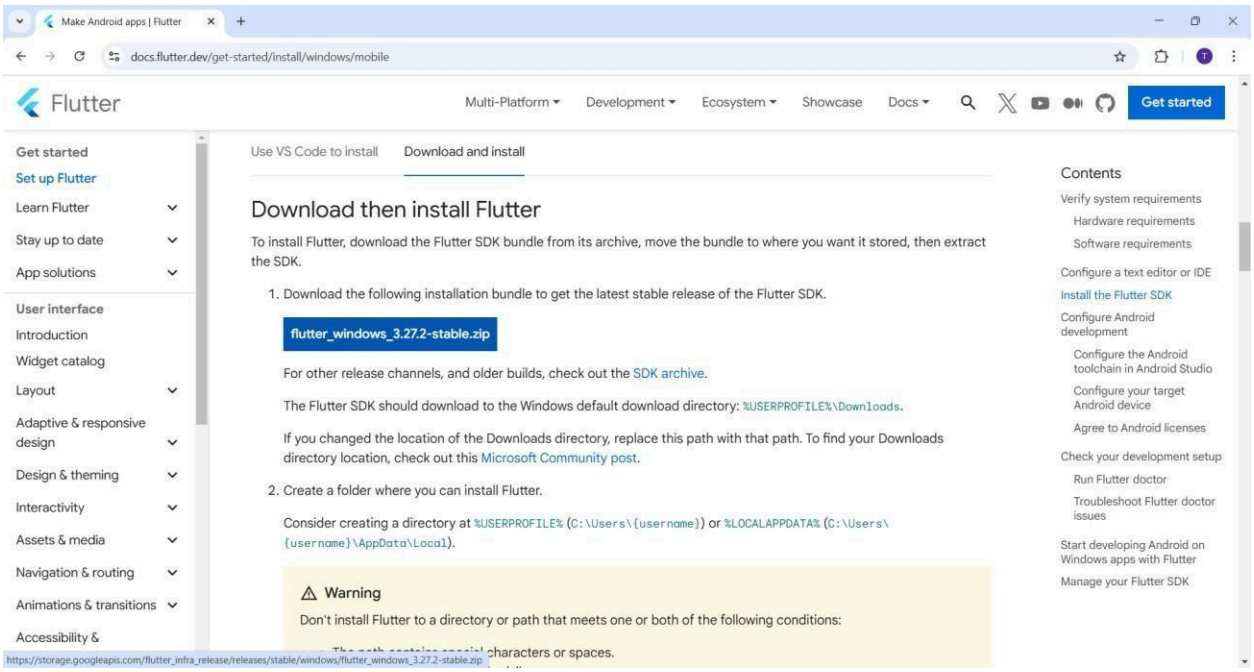
Step 1: Go to the official Flutter website: https://docs.flutter.dev/get-started/install



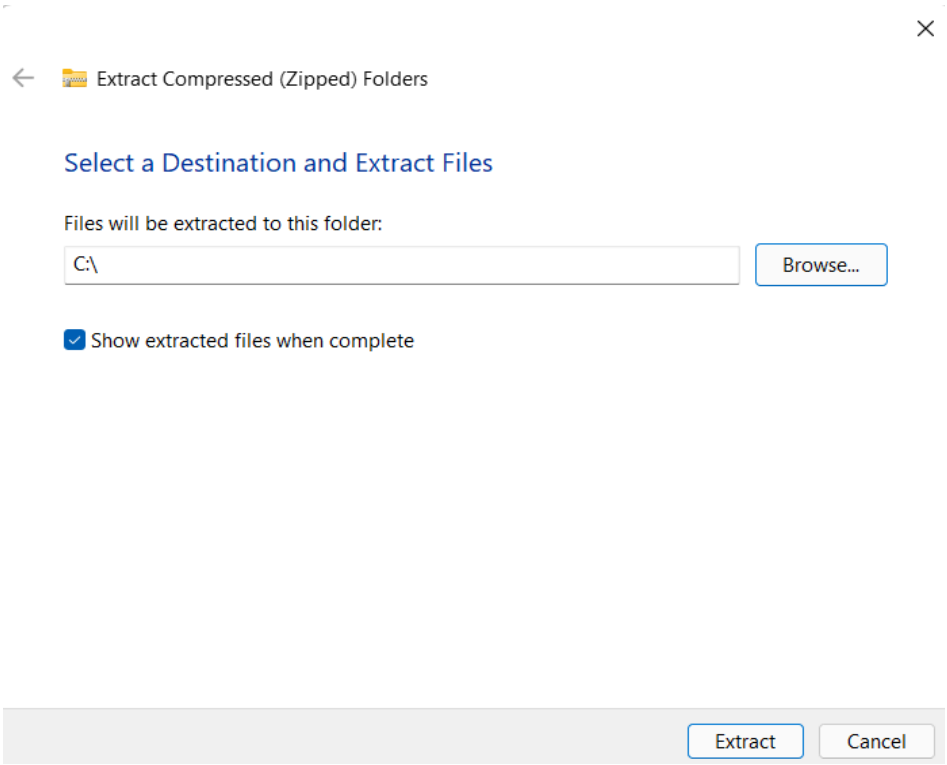
Step 2: To download the latest Flutter SDK, click on the Windows icon > Android



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



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

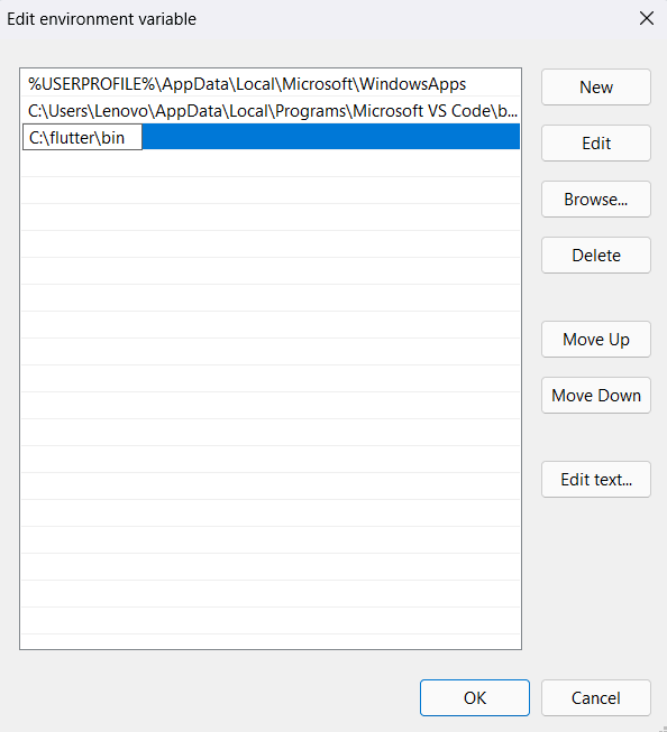


Step 5 :- Add Flutter to System PATH

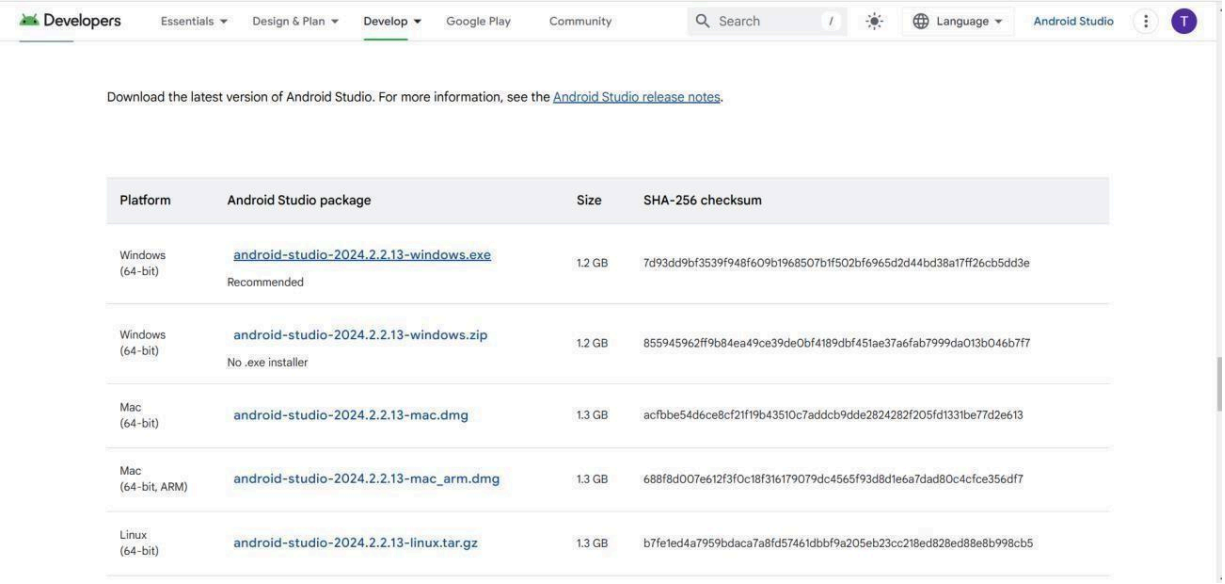
Right-click on the Start Menu > System > Advanced system settings > Environment Variables.

Under System Variables, find Path and click Edit.

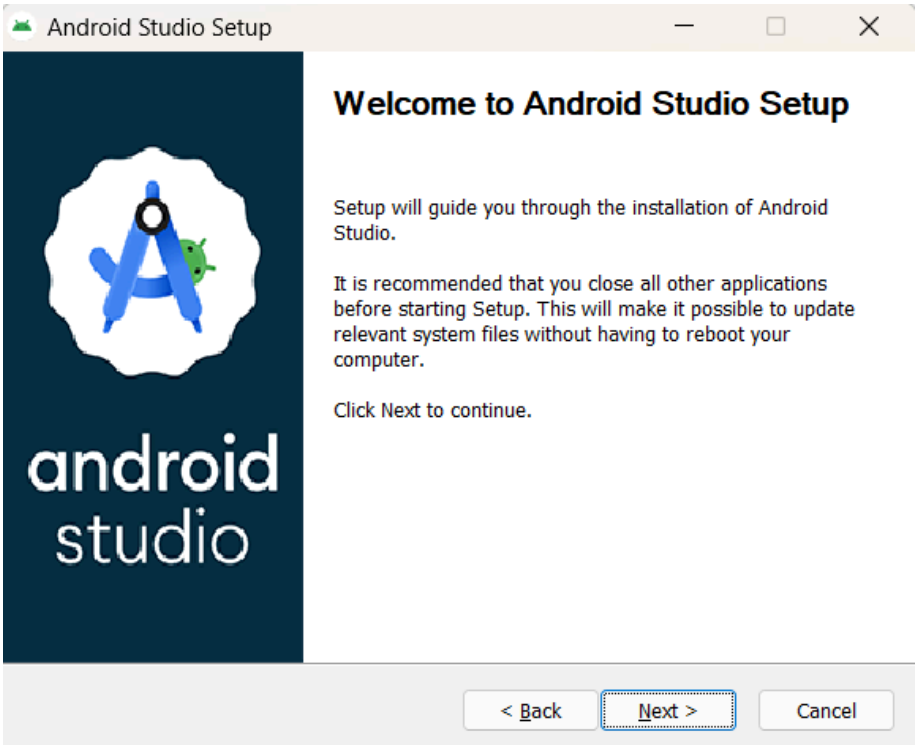
Add the full path to the flutter/bin directory (e.g., C:\flutter\bin).



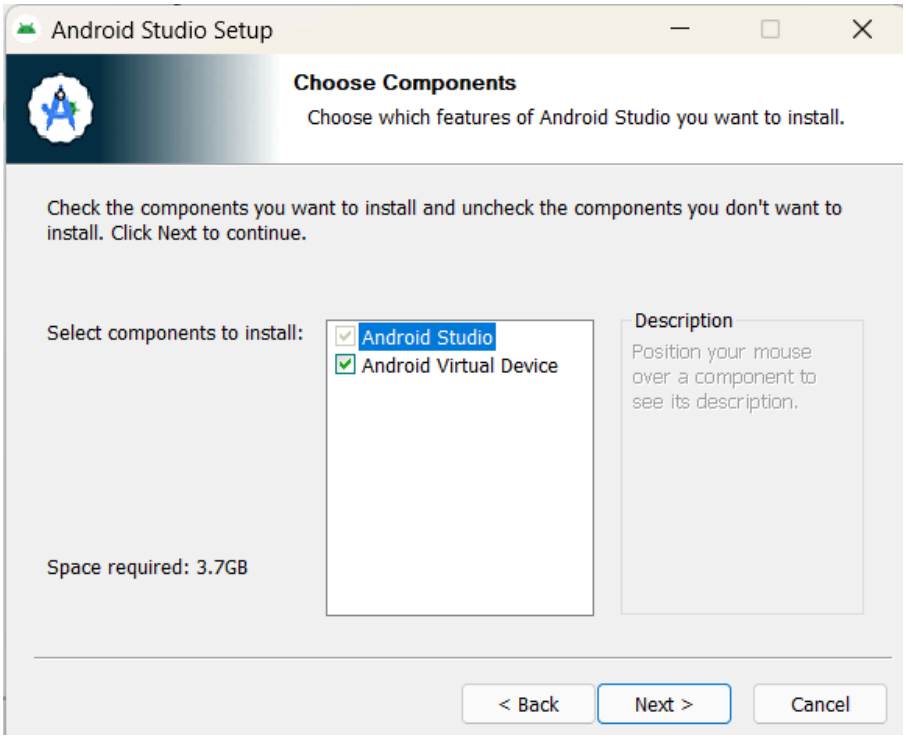
Step 6 : - Go to Android Studio and download the installer.



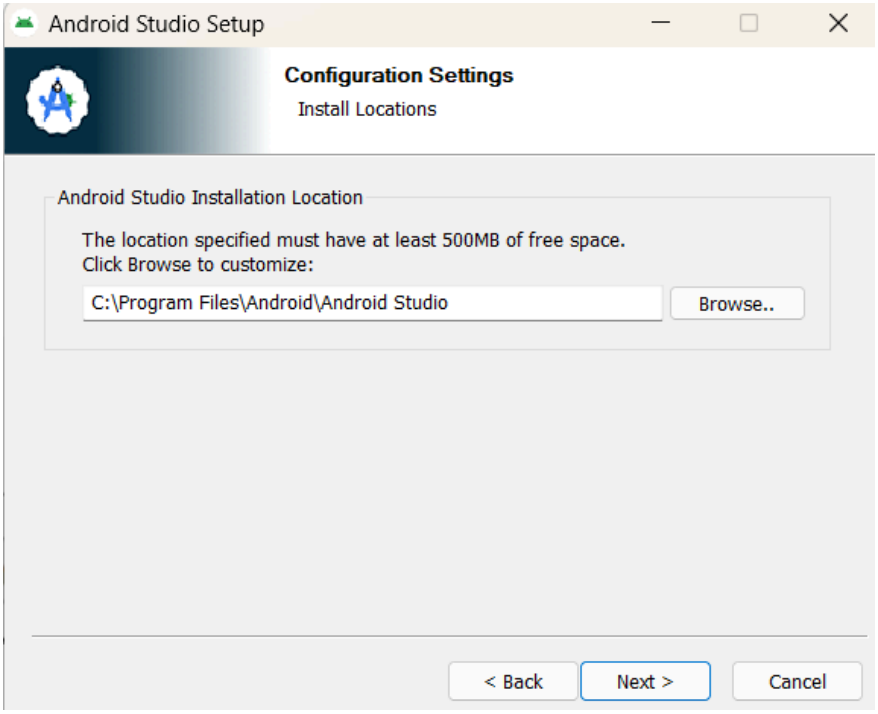
Step 6.1: - When the download is complete, open the .exe file and run it. You will get the following dialog box



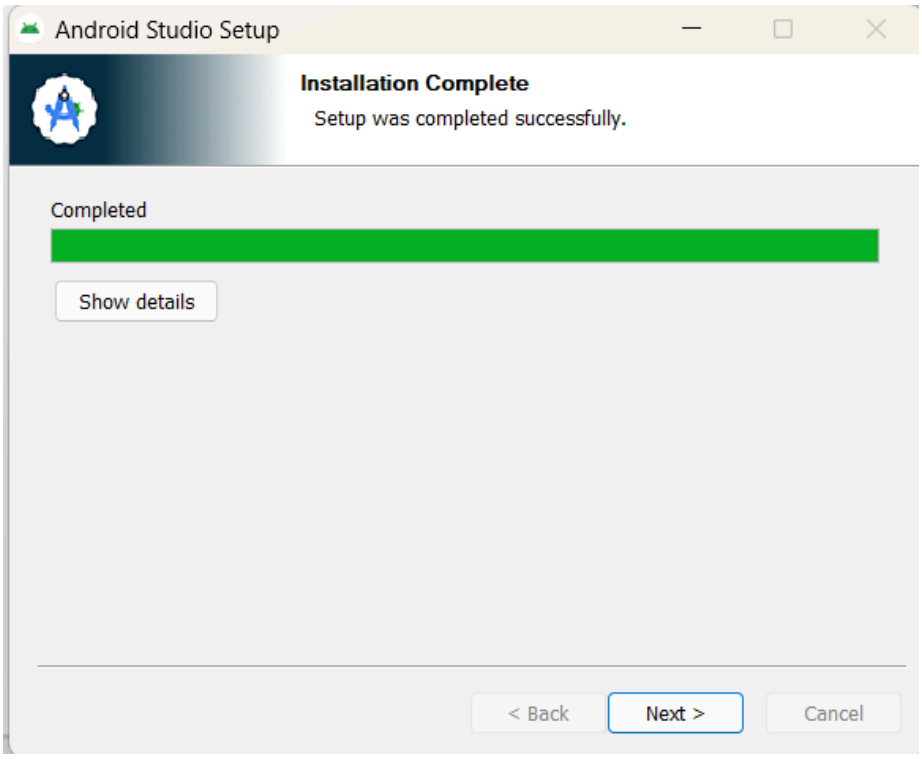
Step 6.2: - Select all the Checkboxes and Click on 'Next' Button.

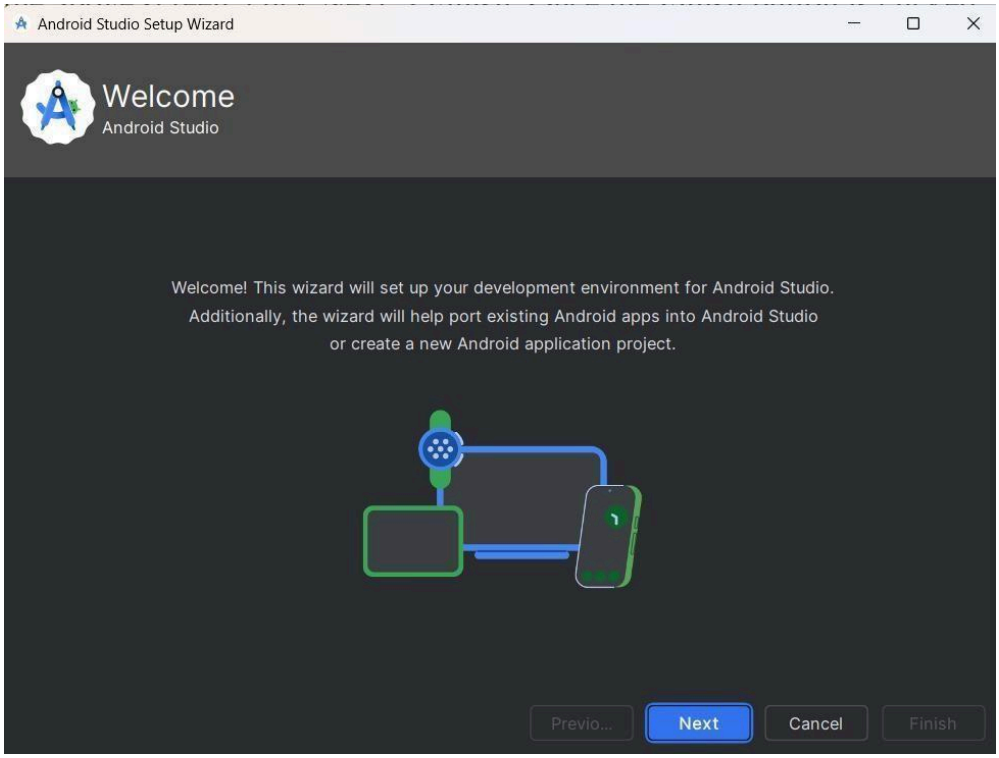


Step 6.3: - Change the destination as per your convenience and click on 'Next' Button.

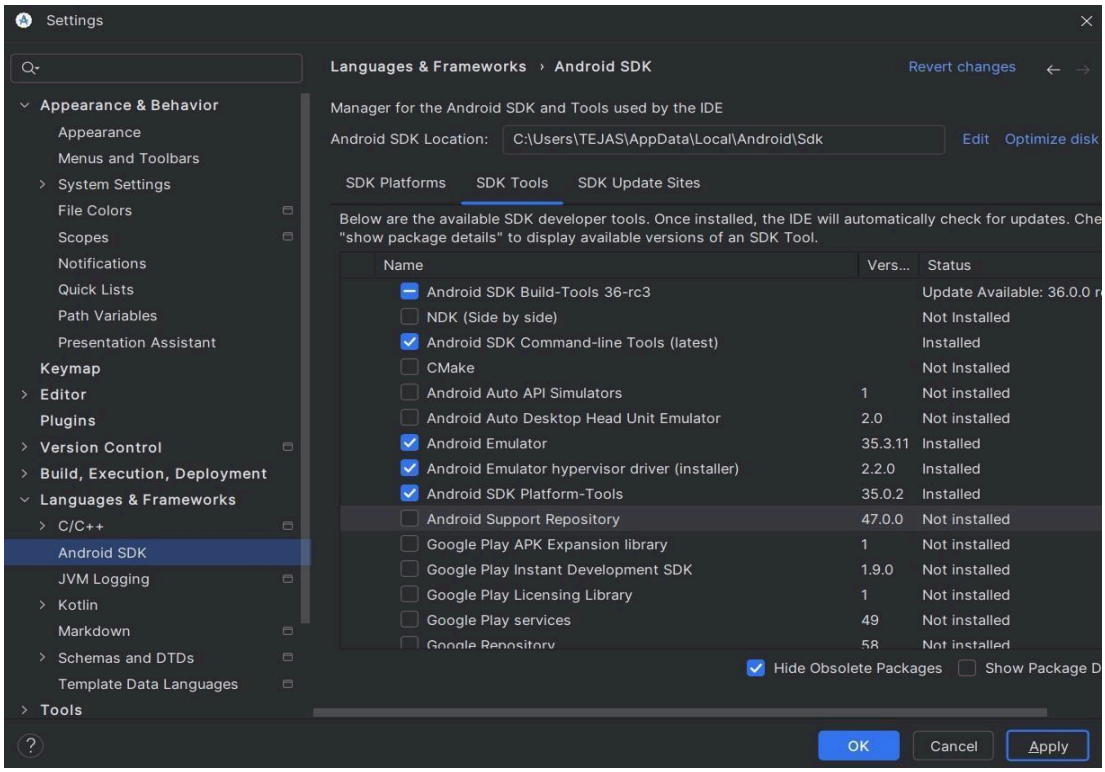


Step 6.4: - Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.





Step 6.5: - Go to Preferences > Appearance & Behavior > System Settings > Android SDK. Select the SDK Tools tab and check Android SDK Command-line Tools and Install it.

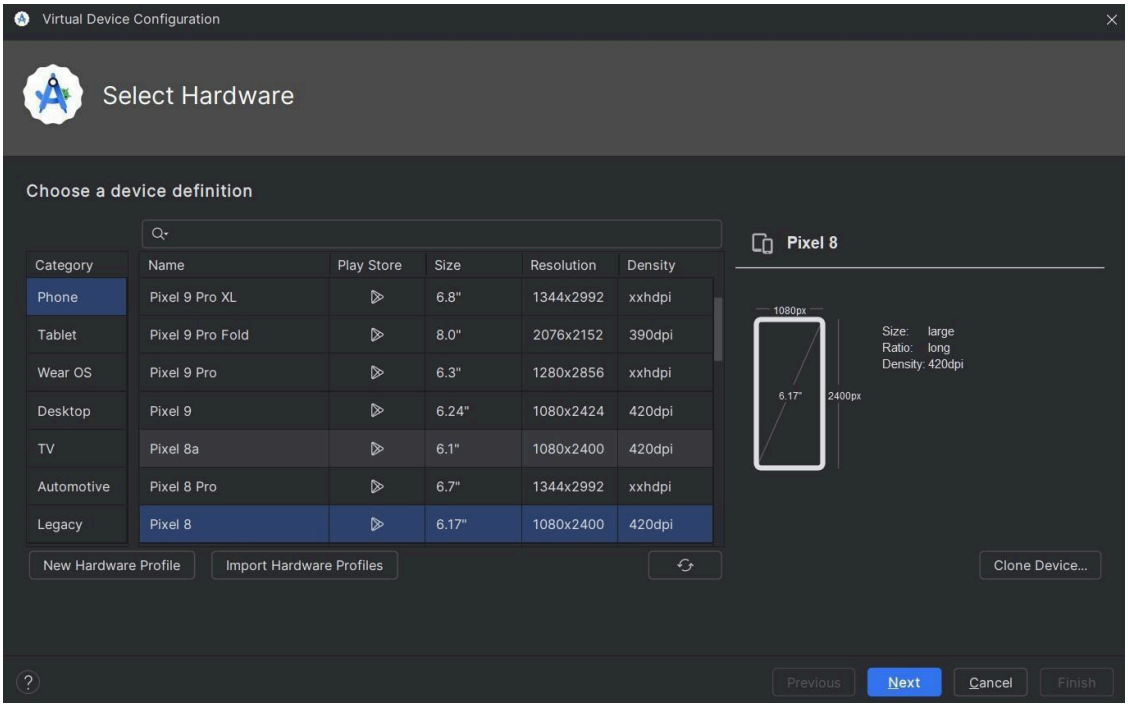


Step 7: - Open a terminal and run the following command

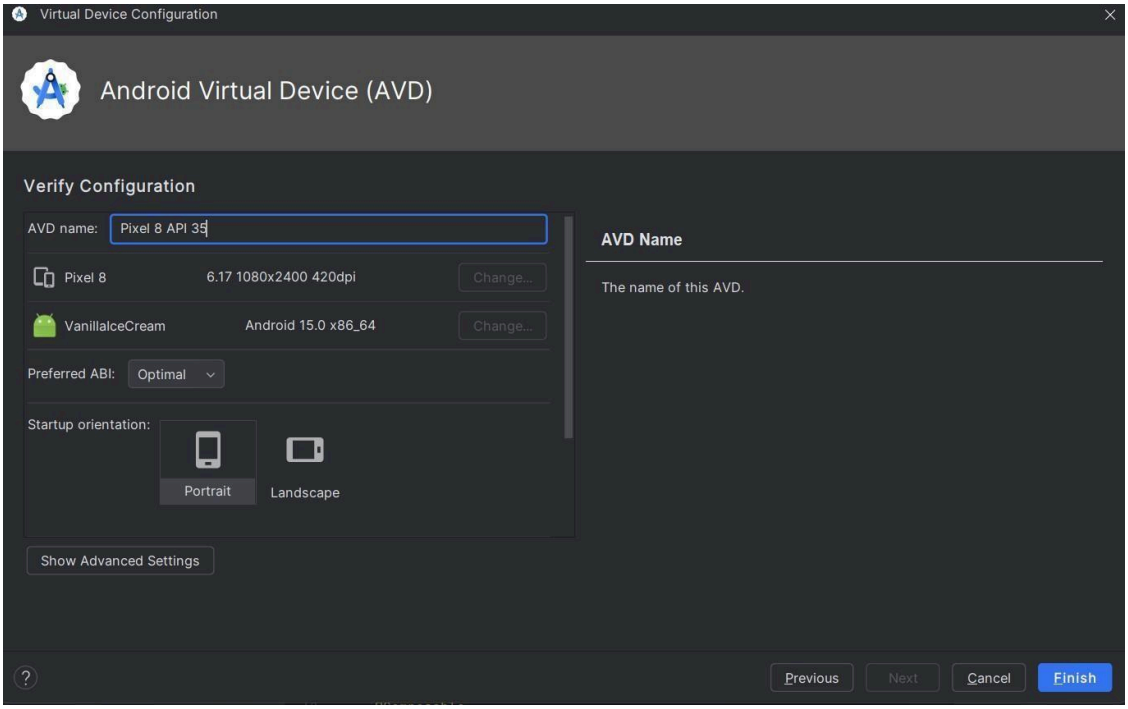
```
C:\Users\ARNAV SAWANT>flutter doctor --android-licenses
Warning: Errors during XML parse:
Warning: Additionally, the fallback loader failed to parse the XML.ry...
Warning: Errors during XML parse:       ] 61% Fetch remote repository...
Warning: Additionally, the fallback loader failed to parse the XML.
[======================================] 100% Computing updates...
6 of 7 SDK package licenses not accepted.
Review licenses that have not been accepted (y/N)? y

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

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

1. Introduction

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

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

2. Accepting this License Agreement

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

2.2 You can accept this License Agreement by:

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

(B) by actually using the Google TV Add-on. In this case, you agree that use of the Google TV Add-on constitutes acceptance of the License Agreement from th
at point onwards.

2.3 You may not use the Google TV Add-on and may not accept the Licensing Agreement if you are a person barred from receiving the Google TV Add-on under the
 laws of the United States or other countries including the country in which you are resident or from which you use the Google TV Add-on.

2.4 If you are agreeing to be bound by this License Agreement on behalf of your employer or other entity, you represent and warrant that you have full legal
```

```
C:\Users\ARNAV SAWANT>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.26100.2894], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[√] Android toolchain - develop for Android devices (Android SDK version 35.0.1)
[√] Chrome - develop for the web
[X] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows apps.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[√] Android Studio (version 2024.2)
[√] VS Code (version 1.96.4)
[√] Connected device (3 available)
[√] Network resources

! Doctor found issues in 1 category.
```

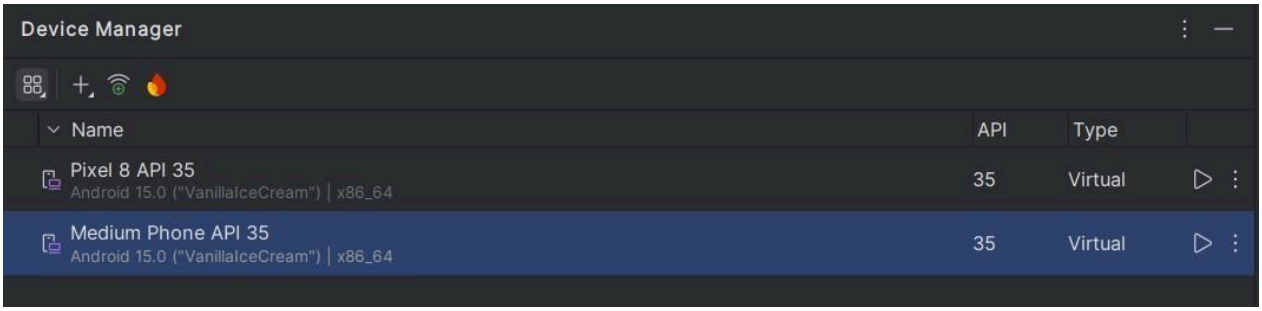
Step 8: - Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application

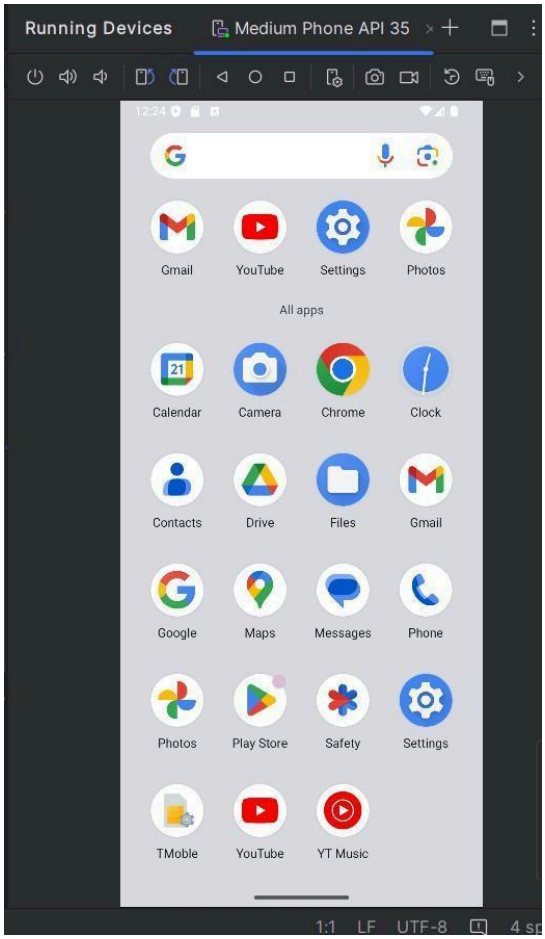


Step 8.1: - Open Android Studio and go to Tools > AVD Manager. Create a new virtual device.



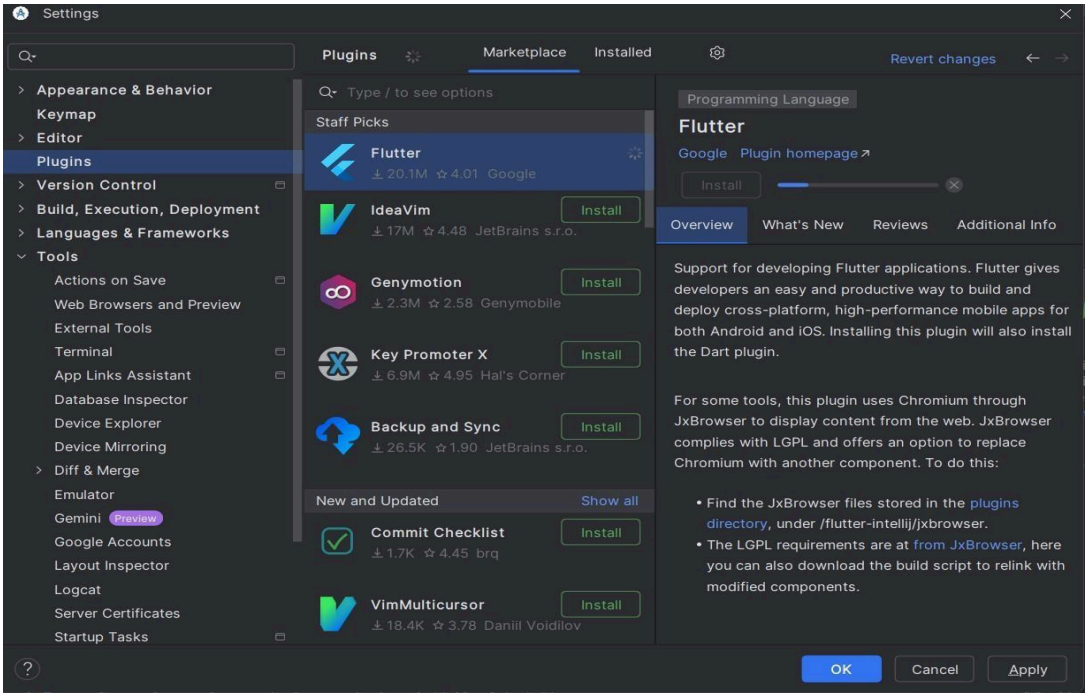
Step 8.2: - Click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen

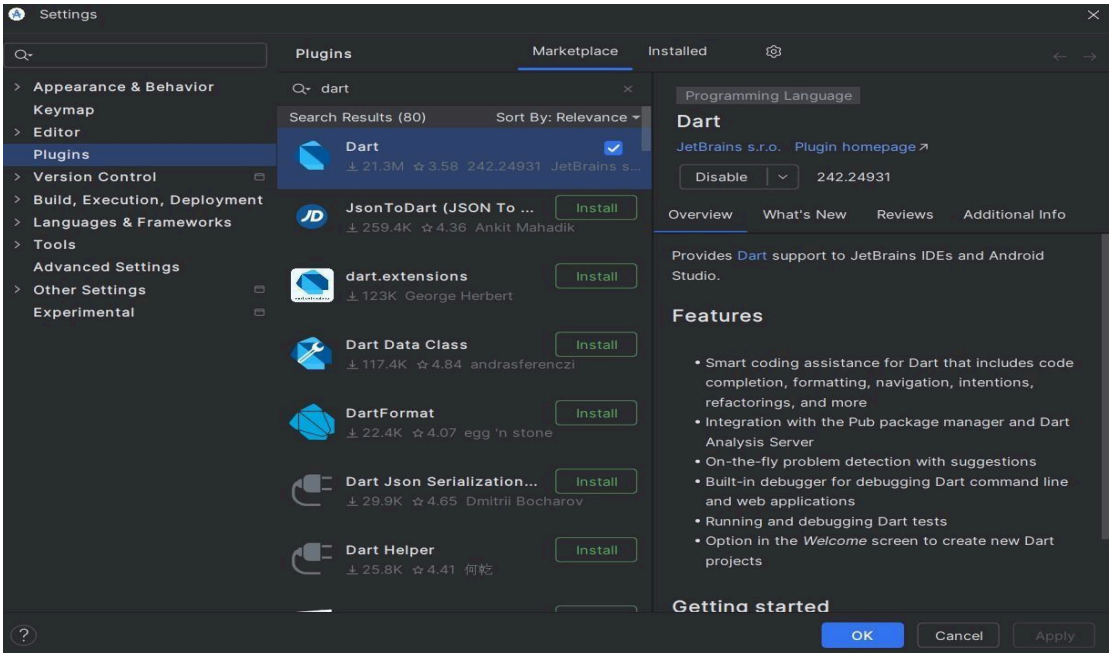




Step 9: - Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself

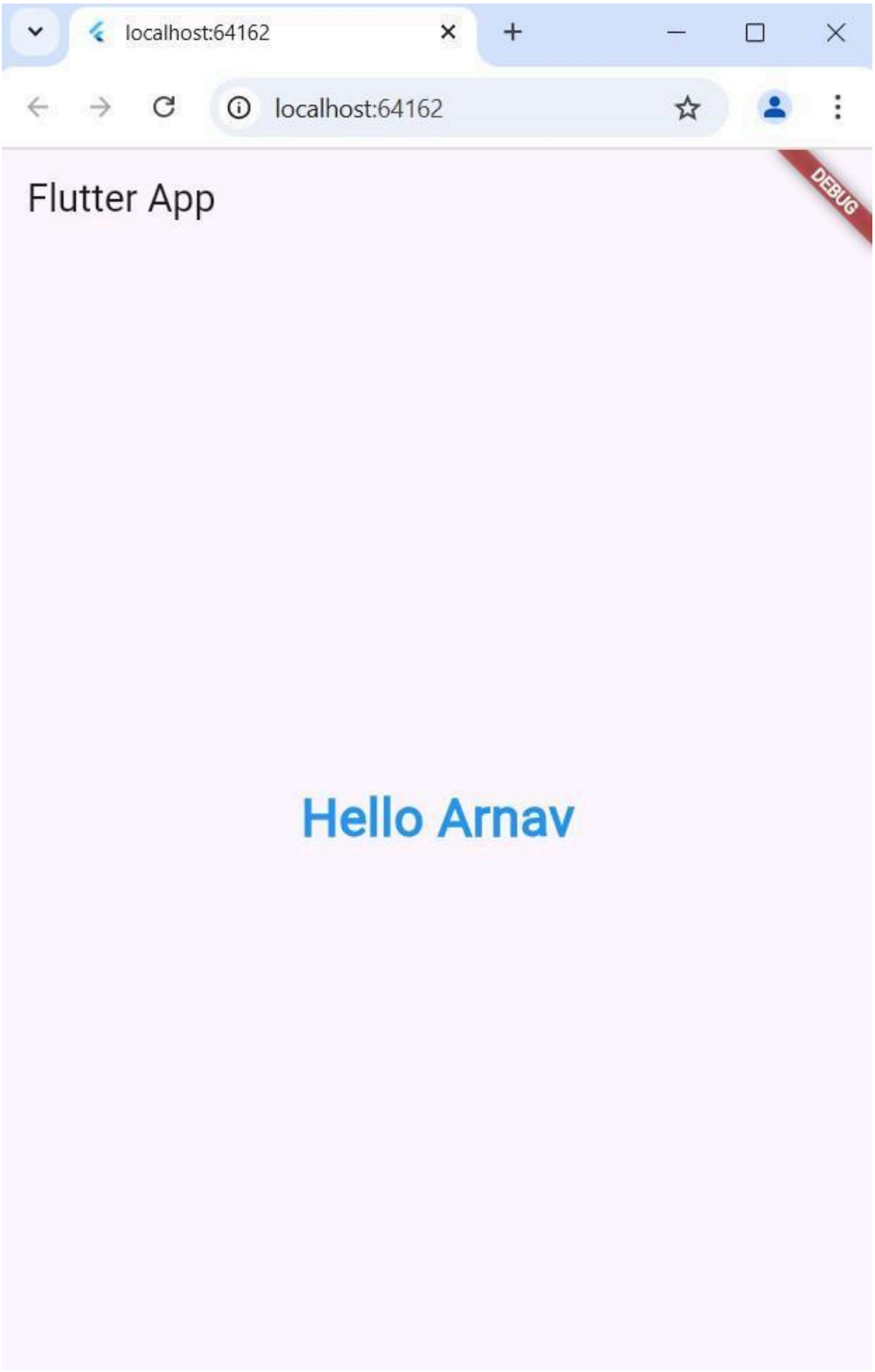
Step 9.1: - Open the Android Studio and then go to File->Settings->Plugins. Now, search the Flutter plugin. If found, select Flutter plugin and click install





Step 9.2: - Restart the Android Studio

Step -10: - Go to File > New Project > Create Flutter Project, then select the project name and location, and click Next to proceed.



MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	52
Name	Arnav Santosh Sawant
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Theory:

The multiple child widgets are a type of widget that contains more than one child widget, and the layout of these widgets is unique. For example:

- **Row widget** lays out its child widgets in a horizontal direction.
- **Column widget** lays out its child widgets in a vertical direction.

By combining the **Row** and **Column** widgets, we can build complex widget structures.

Types of Widgets

- Each element on the screen of a Flutter app is a widget. The view of the screen completely depends on the choice and sequence of widgets used to build the app. The structure of an app's code forms a **tree of widgets**.
- When a change is made in the code, the widget rebuilds its description by calculating the difference between the previous and current widget to determine minimal changes required for rendering in the UI. Widgets are nested within each other to build the app, meaning the **root of the app is itself a widget, and everything inside is also a widget**.

Single Child Layout Widget

A single child layout widget can have **only one child widget inside the parent layout widget**. These widgets can also contain special layout functionalities. Flutter provides many **single child widgets** to make the app UI more attractive. Using these widgets properly saves time and makes the app code more readable.

StatefulWidget

A **StatefulWidget** has state information. It contains **two main classes**:

1. **The state object**
2. **The widget**

It is **dynamic** because it can change its internal data during the widget's lifetime. This widget does not have a **build()** method. Instead, it has a **createState()** method, which returns a class extending Flutter's **State** class.

Examples of Stateful Widgets:

- Checkbox
- Radio
- Slider
- InkWell
- Form
- TextField

StatelessWidget

A **StatelessWidget** does not have any state information and remains **static** throughout its lifecycle.

Examples of Stateless Widgets:

- Text
- Row
- Column
- Container

Commonly Used Widgets

- **Container** – A box widget used for styling with padding, margins, colors, borders, and constraints. It helps in layout structuring and positioning.
- **Row & Column** – Used to arrange widgets in horizontal (**Row**) or vertical (**Column**) orientation. They manage spacing, alignment, and distribution of child widgets.
- **Stack** – Overlaps widgets on top of each other, useful for creating layered UIs like banners, tooltips, or floating elements.
- **Text** – Displays text on the screen with customizable font size, color, alignment, and styling options.
- **Image** – Loads and displays images from assets, network, or memory with scaling and fit properties.
- **Scaffold** – Provides a basic layout structure with an app bar, body, floating action button, and bottom navigation.
- **ListView** – A scrollable list widget that efficiently renders large amounts of dynamic content. Supports both vertical and horizontal scrolling.
- **GridView** – Displays widgets in a grid format, useful for galleries, product listings, or dashboards. It supports dynamic column adjustments.
- **SizedBox** – Used to create space between widgets or define fixed width and height for layout adjustments.
- **ElevatedButton** – A button with elevation that provides a raised effect, customizable with color, shape, and click actions.
- **TextField** – A user input field that supports text entry, keyboard configurations, and validation.

Code:-

searchpage.dart

- *-the common widgets used are- scaffold, appbar, text, iconbutton, column, card*

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

class SearchPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
```

```
  return Scaffold(
    appBar: AppBar(
      title: Text('Search Page'),
      centerTitle: true,
      actions: [
        IconButton(
          icon: Icon(Icons.notifications),
          onPressed: () {
            // Handle notifications
          },
        ),
      ],
    ),
    body: SingleChildScrollView(
      padding: EdgeInsets.all(16),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          // My Feed Section
          _buildSectionHeader('My Feed'),
          _buildHorizontalScroll([
            'Roundups',
            'Video Shorts',
            'Insights',
            'WORD',
            'WHEEL',
            'GAMES',
          ]),
          SizedBox(height: 20),

          // All News Section
          _buildSectionHeader('All News'),
          _buildNewsItem(
            title: 'Who all are in the core team preparing the Budget?',
          ),
          _buildNewsItem(
            title: 'Stampede breaks out on biggest bathing day at Maha Kumbh, several injured',
          ),
          _buildNewsItem(
            title: 'India lose a T20I at home after 426 days',
          ),
          SizedBox(height: 20),

          // Top Stories Section
```

```
        _buildSectionHeader('Top Stories'),
        _buildNewsItem(
          title: 'Top Story 1',
        ),
        _buildNewsItem(
          title: 'Top Story 2',
        ),
        SizedBox(height: 20),

        // Trending Section

_buildSectionHeader('Trending'),
        _buildNewsItem(
          title: 'Trending News 1',
        ),
        _buildNewsItem(
          title: 'Trending News 2',
        ),
      ],
    ),
  ),
 );
}
```

```
// Helper method to build a section header
Widget _buildSectionHeader(String title) {
  return Padding(
    padding: const EdgeInsets.symmetric(vertical: 8.0),
    child: Text(
      title,
      style: TextStyle(
        fontSize: 18,
        fontWeight: FontWeight.bold,
      ),
    ),
  );
}

// Helper method to build a horizontal scrollable list
Widget _buildHorizontalScroll(List<String> items) {
  return Container(
    height: 50,
    child: ListView.builder(
      scrollDirection: Axis.horizontal,
```

```
      itemCount: items.length,
      itemBuilder: (context, index) {
        return Padding(
          padding: const EdgeInsets.only(right: 8.0),
          child: Chip(
            label: Text(items[index]),
            backgroundColor: Colors.blue[50],
          ),
        );
      },
    ),
  );
 }

 // Helper method to build a news item
  Widget _buildNewsItem({required String title}) {
    return Card(
      margin: EdgeInsets.symmetric(vertical: 4),
      child: ListTile(
        title: Text(title),
        trailing: Icon(Icons.arrow_forward_ios, size: 16),
        onTap: () {
          // Handle news item tap
        },
      ),
    );
  }
}
```

profile.dart

-the common widgets used in this flutter code is- scaffold, appbar, iconbutton, sizedbox, center

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

class ProfileScreen extends StatelessWidget {
  @override
```

```
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    backgroundColor: Colors.white,
    elevation: 0,
    leading: IconButton(
     icon: Icon(Icons.arrow_back, color: Colors.black),
     onPressed: () => Navigator.pop(context),
    ),
    actions: [
     TextButton(
      onPressed: () {},
      child: Text("Feedback", style: TextStyle(color: Colors.black)),
     ),
     IconButton(
      icon: Icon(Icons.settings, color: Colors.black),
      onPressed: () {},
     ),
    ],
   ),
   body: Column(
    crossAxisAlignment: CrossAxisAlignment.center,
    children: [
     SizedBox(height: 10),

     // Profile Picture
     Center(
      child: Column(
       children: [
        CircleAvatar(
         radius: 40,
         backgroundImage: AssetImage('assets/profile.jpg'), // Replace with your image
        ),
        SizedBox(height: 10),
        Text("Arnav", style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold)),
        SizedBox(height: 5),

        // Edit Profile Button
        ElevatedButton.icon(
         onPressed: () {},
         style: ElevatedButton.styleFrom(
```

```
            backgroundColor:
Colors.grey[200],
            foregroundColor:
Colors.black,
            padding:
EdgeInsets.symmetric(horizontal: 12,
vertical: 5),
            shape:
RoundedRectangleBorder(
              borderRadius:
BorderRadius.circular(20),
            ),
          ),
          icon: Icon(Icons.edit, size:
16),
          label: Text("Edit Profile"),
        ),
      ],
    ),
  ),

  SizedBox(height: 20),

  // My Bookmarks Section
  Padding(
    padding:
EdgeInsets.symmetric(horizontal: 20),
    child: Column(
      crossAxisAlignment:
CrossAxisAlignment.start,
      children: [
        Text("My Bookmarks",
          style: TextStyle(fontSize: 16,
fontWeight: FontWeight.bold)),
        Container(
          width: 40,
          height: 2,
          color: Colors.blue,
          margin:
EdgeInsets.only(top: 3, bottom: 10),
        ),

        SizedBox(height: 20),

        // Empty Bookmarks
Message
        Center(
          child: Column(
            children: [
              Text("It's Empty Here",
                style: TextStyle(fontSize:
16, fontWeight: FontWeight.bold)),
              SizedBox(height: 5),
```

```
                Text(
                  "Click on the bookmark icon to bookmark a short",
                  textAlign: TextAlign.center,
                  style: TextStyle(color: Colors.grey),
                ),
              ],
            ),
          ),
        ],
      ),
    );
  }
}
```

editprofile.dart

-the common widgets used in this code is – scaffold, appbar, text, padding, column, sizedbox

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

class EditProfileScreen extends StatefulWidget {
  @override
  _EditProfileScreenState createState() => _EditProfileScreenState();
}

class _EditProfileScreenState extends State<EditProfileScreen> {
  TextEditingController nameController = TextEditingController(text: "Arnav");
  TextEditingController bioController = TextEditingController();

  int nameMaxLength = 20;
  int bioMaxLength = 120;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

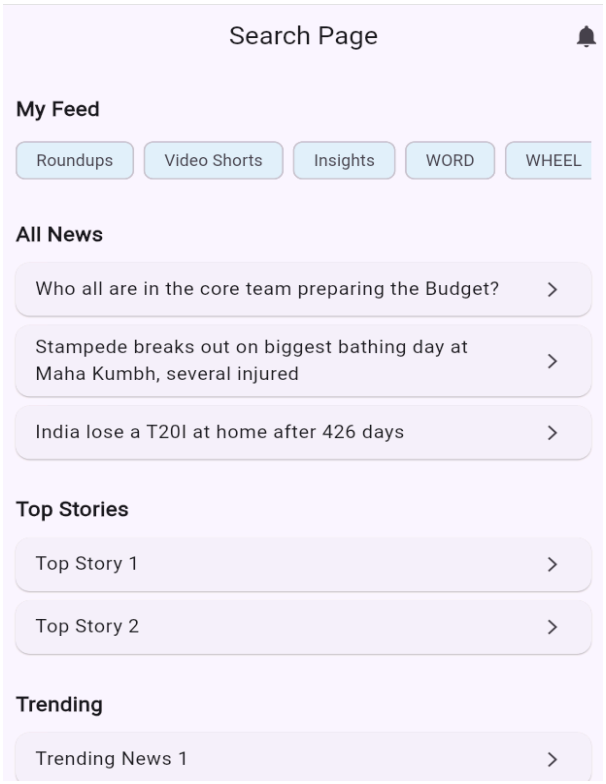
```
    appBar: AppBar(
      leading: IconButton(
        icon: Icon(Icons.arrow_back, color: Colors.black),
        onPressed: () => Navigator.pop(context),
      ),
      title: Text("Edit Profile", style: TextStyle(color: Colors.black)),
      backgroundColor: Colors.white,
      elevation: 0,
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          // Profile Picture
          Center(
            child: CircleAvatar(
              radius: 40,
              backgroundImage: AssetImage('assets/profile.jpg'), // Replace with your image
            ),
          ),
          SizedBox(height: 20),

          // Name Field
          _buildLabel("Name"),
          TextField(
            controller: nameController,
            maxLength: nameMaxLength,
            decoration: InputDecoration(
              border: OutlineInputBorder(),
            ),
          ),

_buildCharacterLimit(nameController.text.length, nameMaxLength),

          SizedBox(height: 16),

          // Bio Field
          _buildLabel("Your Bio"),
          TextField(
            controller: bioController,
            maxLength: bioMaxLength,
```

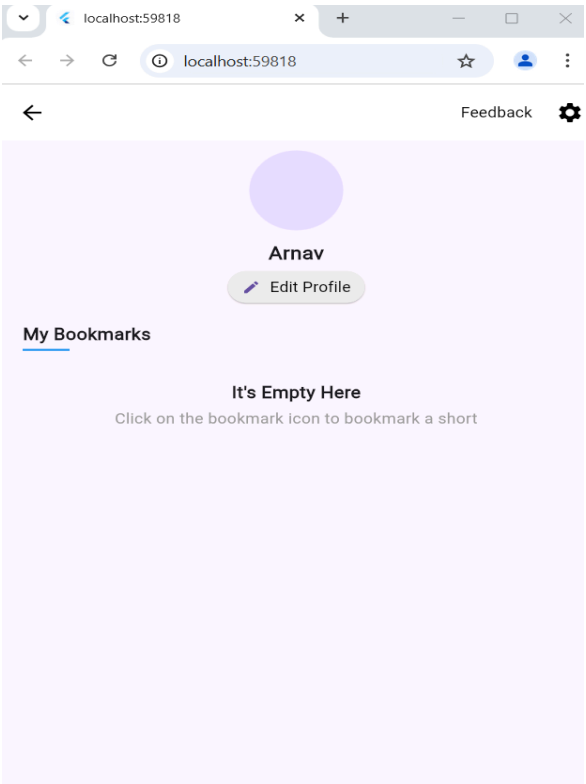
```
          maxLines: 3,
          decoration: InputDecoration(
            border: OutlineInputBorder(),
            hintText: "Enter your bio here",
          ),
        ),

_buildCharacterLimit(bioController.text.length, bioMaxLength),

        SizedBox(height: 20),

        // Submit Button
        SizedBox(
          width: double.infinity,
          child: ElevatedButton(
            onPressed: () {},
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.blue,
              padding: EdgeInsets.symmetric(vertical: 14),
            ),
            child: Text("Submit", style: TextStyle(fontSize: 16, color: Colors.white)),
          ),
        ),
      ],
    ),
  ),
 );
}

 // Helper to create labels
 Widget _buildLabel(String text) {
  return Padding(
    padding: const EdgeInsets.only(bottom: 8),
    child: Text(text, style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold)),
  );
 }

 // Character counter
 Widget _buildCharacterLimit(int current, int max) {
  return Align(
```

```
    alignment: Alignment.centerRight,
    child: Text("$current/$max", style:
TextStyle(color: Colors.grey, fontSize:
12)),
                                    );
                                  }
                                }
```

Screenshots:-

searchpage.dart





editprofile.dart

profile.dart

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	52
Name	Arnav Santosh Sawant
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Flutter is a versatile open-source UI framework , which allows developers to build natively compiled applications for mobile, web, and desktop platforms from a single codebase. One of the key strengths of Flutter is its flexibility in creating highly customizable UIs. This practical focuses on incorporating essential visual elements—icons, images, and custom fonts—into a Flutter application. These elements enhance the visual appeal and usability of the app, providing an engaging experience for users.

A flutter app when built has both assets (resources) and code. Assets are available and deployed during runtime. The asset is a file that can include static data, configuration files, icons, and images. The Flutter app supports many image formats, such as JPEG, WebP, PNG, GIF, animated WebP/GIF, BMP, and WBMP.

Visual elements play a significant role in app development.

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

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

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

- **Adding Images in Flutter**

  Flutter supports images from three sources:

  1. **Assets** (Stored locally in the project)

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

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

     - Display the image in the app

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

2. **Network** (Fetched from the internet)

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

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

3. **Memory or File** (Stored on the device)

 Adding Custom Fonts in Flutter

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

- Download the font and place it in the assets/fonts/ folder.
- Declare the font in pubspec.yaml
- Use the font in the
  app Text(
  'Custom Font Example',
  style: TextStyle(fontFamily: 'CustomFont', fontSize: 24),
  );

Code: -

profile.dart

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

class ProfileScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    backgroundColor: Colors.white,
    elevation: 0,
    leading: IconButton(
     icon: Icon(Icons.arrow_back, color:
Colors.black),
     onPressed: () =>
Navigator.pop(context),
    ),
    actions: [
     TextButton(
      onPressed: () {},
      child: Text("Feedback", style:
TextStyle(color: Colors.black)),
     ),
     IconButton(
      icon: Icon(Icons.settings, color:
Colors.black),
      onPressed: () {},
     ),
    ],
   ),
   body: Column(
    crossAxisAlignment:
CrossAxisAlignment.center,
    children: [
     SizedBox(height: 10),

     // Profile Picture
     Center(
      child: Column(
       children: [
        CircleAvatar(
         radius: 40,
         backgroundImage:
AssetImage('assets/logo.jpg'), // Replace with
your image
        ),
        SizedBox(height: 10),
        Text("Arnav", style:
TextStyle(fontSize: 18, fontWeight:
FontWeight.bold)),
        SizedBox(height: 5),

        // Edit Profile Button
        ElevatedButton.icon(
         onPressed: () {
          Navigator.push(
         context,
        MaterialPageRoute(builder:
(context) => EditProfileScreen()),
        );
          },
         style: ElevatedButton.styleFrom(
          backgroundColor:
Colors.grey[200],
          foregroundColor: Colors.black,
          padding:
EdgeInsets.symmetric(horizontal: 12, vertical:
5),
          shape: RoundedRectangleBorder(
           borderRadius:
BorderRadius.circular(20),
          ),
         ),
         icon: Icon(Icons.edit, size: 16),
         label: Text("Edit Profile"),
        ),
       ],
      ),
     ),

     SizedBox(height: 20),

     // My Bookmarks Section
     Padding(
      padding:
```

```
EdgeInsets.symmetric(horizontal: 20),
          child: Column(
            crossAxisAlignment:
CrossAxisAlignment.start,
            children: [
              Text("My Bookmarks",
                style: TextStyle(fontSize: 16,
fontWeight: FontWeight.bold)),
              Container(
                width: 40,
                height: 2,
                color: Colors.blue,
                margin: EdgeInsets.only(top: 3,
bottom: 10),
              ),

              SizedBox(height: 20),

              // Empty Bookmarks Message
              Center(
                child: Column(
                  children: [
                    Text("It's Empty Here",
                      style: TextStyle(fontSize: 16,
fontWeight: FontWeight.bold)),
                    SizedBox(height: 5),
                    Text(
                      "Click on the bookmark icon to
bookmark a short",
                      textAlign: TextAlign.center,
                      style: TextStyle(color:
Colors.grey, fontFamily: 'GrechenFuemen',
fontSize: 25),
                    ),
                  ],
                ),
              ),

Image.network('https://www.shutterstock.com
/image-vector/vector-flat-illustration-no-book
marks-260nw-2298489655.jpg', width:
250,height: 250,fit: BoxFit.fitWidth,),
            ],
          ),
        ),
      ],
    ),
  );
 }
}
```

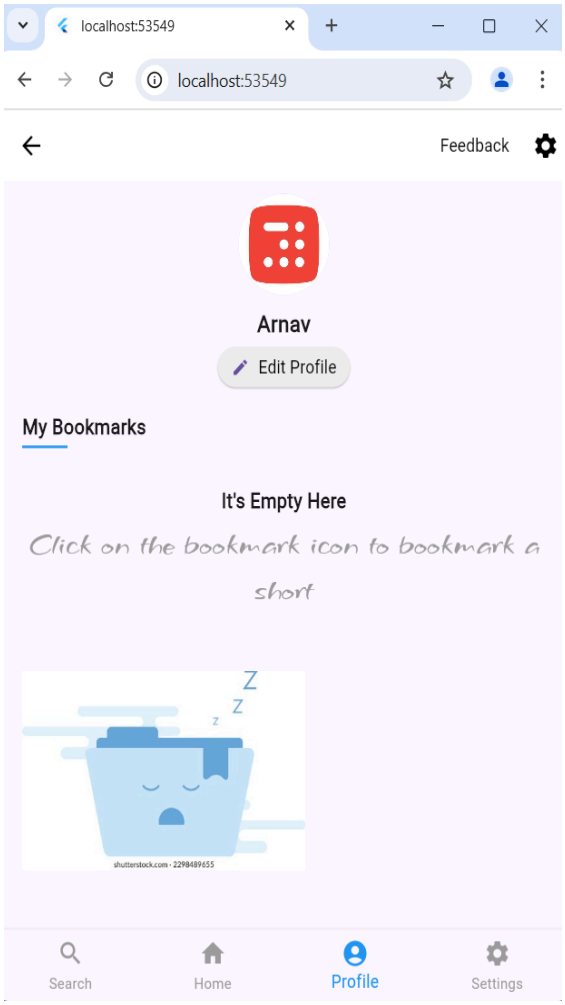
settings.dart

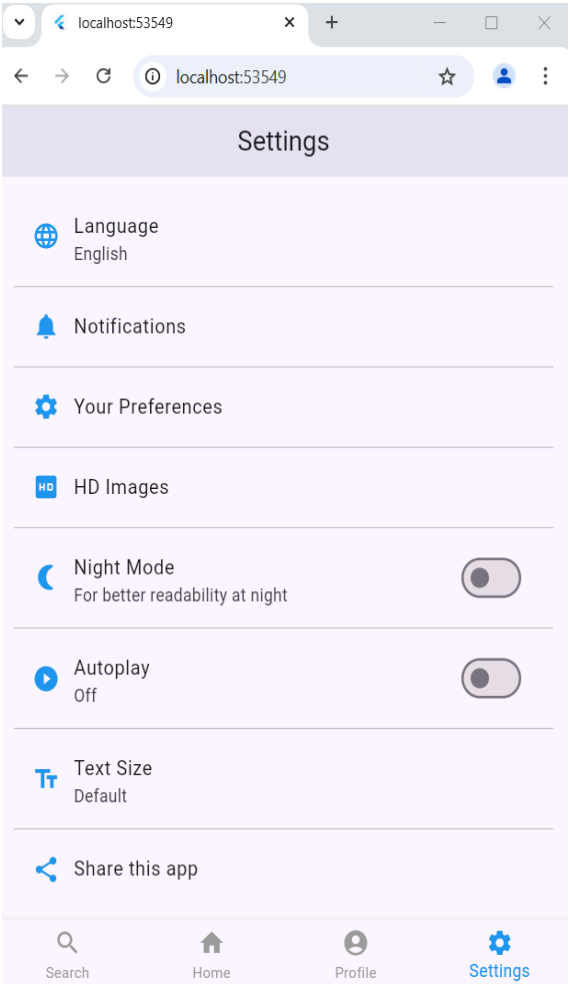
```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   debugShowCheckedModeBanner: false,
   home: SettingsPage(),
  );
 }
}

class SettingsPage extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: Text('Settings'),
    centerTitle: true,
   ),
   body: ListView(
    padding: EdgeInsets.all(16),
    children: [
     // Language
     _buildSettingsItem(
      icon: Icons.language,
```

```
        title: 'Language',
        subtitle: 'English',
        onTap: () {
          // Handle language selection
        },
      ),
      Divider(),

      // Notifications
      _buildSettingsItem(
        icon: Icons.notifications,
        title: 'Notifications',
        onTap: () {
          // Handle notifications settings
        },
      ),
      Divider(),

      // Your Preferences
      _buildSettingsItem(
        icon: Icons.settings,
        title: 'Your Preferences',
        onTap: () {
          // Handle preferences
        },
      ),
      Divider(),

      // HD Images
      _buildSettingsItem(
        icon: Icons.hd,
        title: 'HD Images',
        onTap: () {
          // Handle HD images settings
        },
      ),
      Divider(),

      // Night Mode
      _buildSwitchItem(
        icon: Icons.nightlight_round,
        title: 'Night Mode',
        subtitle: 'For better readability at
night',
        value: false, // Default value
        onChanged: (value) {
          // Handle night mode toggle
        },
      ),
      Divider(),

      // Autoplay
      _buildSwitchItem(
        icon: Icons.play_circle_filled,
        title: 'Autoplay',
        subtitle: 'Off',
        value: false, // Default value
        onChanged: (value) {
          // Handle autoplay toggle
        },
      ),
      Divider(),

      // Text Size
      _buildSettingsItem(
        icon: Icons.text_fields,
        title: 'Text Size',
        subtitle: 'Default',
        onTap: () {
          // Handle text size settings
        },
      ),
      Divider(),

      // Share this app
      _buildSettingsItem(
        icon: Icons.share,
        title: 'Share this app',
        onTap: () {
          // Handle share functionality
        },
      ),
    ],
  ),
 );
}

// Helper method to build a settings item with
an icon and text
Widget _buildSettingsItem({
  required IconData icon,
  required String title,
  String? subtitle,
  VoidCallback? onTap,
```

```
  }) {
    return ListTile(
      leading: Icon(icon, color: Colors.blue),
      title: Text(title, style: TextStyle(fontSize:
16)),
      subtitle: subtitle != null ? Text(subtitle) :
null,
      onTap: onTap,
    );
  }

  // Helper method to build a settings item with
a switch
  Widget _buildSwitchItem({
    required IconData icon,
    required String title,
    String? subtitle,
    required bool value,
    required ValueChanged<bool> onChanged,
  }) {
    return ListTile(
      leading: Icon(icon, color: Colors.blue),
      title: Text(title, style: TextStyle(fontSize:
16)),
      subtitle: subtitle != null ? Text(subtitle) :
null,
      trailing: Switch(
        value: value,
        onChanged: onChanged,
      ),
    );
  }
}
```

Project Title: Inshorts clone app **Roll No. 52**





MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	52
Name	Arnav Santosh Sawant
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

Forms in Flutter are essential components used to collect and manage user input efficiently. They are widely used in login screens, registration pages, and feedback forms. Flutter provides a **Form** widget that works alongside **TextFormField** and other input elements, offering features such as validation, error handling, and state management to improve user experience.

Key Components of a Form in Flutter

1. Form Widget

The **Form** widget acts as a container that groups multiple input fields and manages their validation.

- Requires a **GlobalKey<FormState>** to uniquely identify the form and interact with it.
- Helps in structuring form fields and handling user input efficiently.

2. Form Fields (TextFormField)

The **TextFormField** widget is used for user input, such as entering names, emails, or phone numbers.

- It supports input validation using the **validator** property.
- Allows customization with **InputDecoration** (e.g., labels, icons, borders, hint text).
- Different **TextInputType** options can be set for appropriate keyboard input (e.g., **TextInputType.emailAddress** for emails).

3. Validation in Forms

Ensuring valid user input is crucial. The **validator** property in **TextFormField** helps check whether the entered data meets specified criteria before submission.

- Validation can be triggered manually using formKey.currentState!.validate().
- The **autovalidateMode** property can enable automatic validation during user input.

4. State Management in Forms

To ensure data persistence and processing, proper state management is required.

- The **FormState** class provides methods like validate(), save(), and reset() to manage form behavior.
- The save() method stores user input when validation is successful.
- The reset() method clears the form fields and restores the initial state.

5. Submit Button

A **submit button** is necessary to trigger form validation and submit user data.

- When pressed, it checks validation using formKey.currentState!.validate().
- If validation succeeds, the form data is saved and processed accordingly.

Important Properties & Methods of Form Widget

Properties

- **key** → A GlobalKey<FormState> that uniquely identifies the form.
- **child** → Contains form fields, typically wrapped in a Column or ListView.
- **autovalidateMode** → Defines when the form should auto-validate.

Methods

- **validate()** → Checks if all form fields are valid and returns true or false.
- **save()** → Stores the current values of form fields after successful validation.
- **reset()** → Clears user input and resets the form to its initial state.
- **currentState** → Provides access to the form's state for validation, saving, or resetting.

Code:-

login.dart

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

class LoginPage extends StatefulWidget {
 @override
 _LoginPageState createState() =>
_LoginPageState();
}

class _LoginPageState extends
State<LoginPage> {
 final _formKey = GlobalKey<FormState>();
 TextEditingController emailController =
TextEditingController();
 TextEditingController passwordController =
TextEditingController();
 TextEditingController usernameController =
TextEditingController();

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   backgroundColor: Colors.white,
   body: Center(
    child: SingleChildScrollView(
     padding: const
EdgeInsets.symmetric(horizontal: 32.0),
     child: Form(
      key: _formKey,
      child: Column(
       mainAxisAlignment:
MainAxisAlignment.center,
       children: [
        // Logo
        Image.asset(
         'assets/logo.jpg', // Replace with your
logo asset
         height: 80,
        ),
        SizedBox(height: 40),

        // Username Field
        _buildTextField(controller:
usernameController, label: "Username"),
        SizedBox(height: 10),

        // Email Field
```

```
            _buildTextField(controller:
emailController, label: "Email", keyboardType:
TextInputType.emailAddress),
            SizedBox(height: 10),

            // Password Field
            _buildTextField(controller:
passwordController, label: "Password",
obscureText: true),
            SizedBox(height: 20),

            // Login Button
            SizedBox(
              width: double.infinity,
              child: ElevatedButton(
                onPressed: () {
                  if
(_formKey.currentState!.validate()) {
                    // Handle login logic
                  }
                },
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.blue,
                  padding:
EdgeInsets.symmetric(vertical: 14),
                ),
                child: Text("Login", style:
TextStyle(fontSize: 16, color: Colors.white)),
              ),
            ),
            SizedBox(height: 20),

            // Social Login Buttons
            _buildLoginButton(
              color: Color(0xFF1877F2),
              text: "Sign in with Facebook",
              icon: Icons.facebook,
            ),
            SizedBox(height: 10),
            _buildLoginButton(
              color: Colors.white,
              text: "Sign in with Google",
              icon: Icons.g_mobiledata,
              borderColor: Colors.grey,
              textColor: Colors.black,
            ),
            SizedBox(height: 10),
            _buildLoginButton(
              color: Colors.blue,
              text: "Sign in with phone",
              icon: Icons.phone,
            ),
          ],
        ),
      ),
    ),
  );
}

Widget _buildTextField({
  required TextEditingController controller,
  required String label,
  bool obscureText = false,
  TextInputType keyboardType =
TextInputType.text,
}) {
  return TextFormField(
    controller: controller,
    obscureText: obscureText,
    keyboardType: keyboardType,
    decoration: InputDecoration(
      labelText: label,
      border: OutlineInputBorder(),
    ),
    validator: (value) {
      if (value == null || value.isEmpty) {
        return "$label cannot be empty";
      }
      return null;
    },
  );
}

Widget _buildLoginButton({
  required Color color,
  required String text,
  required IconData icon,
  Color textColor = Colors.white,
  Color? borderColor,
}) {
  return Container(
    width: double.infinity,
    child: ElevatedButton.icon(
      icon: Icon(icon, color: textColor),
      label: Text(
        text,
        style: TextStyle(color: textColor),
```

```
      ),
      onPressed: () {},
      style: ElevatedButton.styleFrom(
        backgroundColor: color,
        padding: EdgeInsets.symmetric(vertical:
14),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(8),
          side: borderColor != null ?
BorderSide(color: borderColor) :
BorderSide.none,
        ),
      ),
    ),
   );
  }
}
```

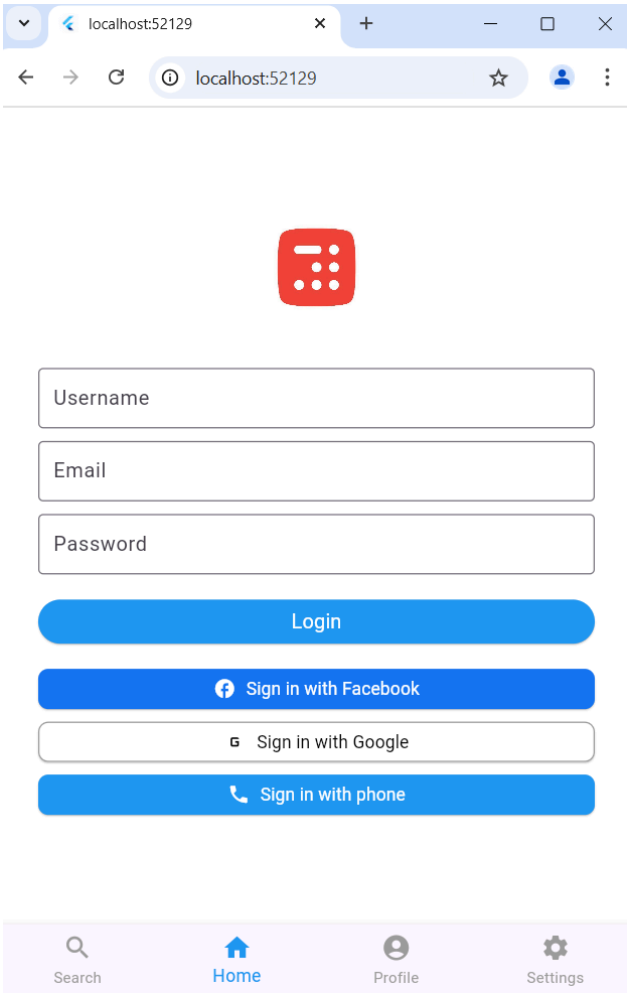
editprofile.dart

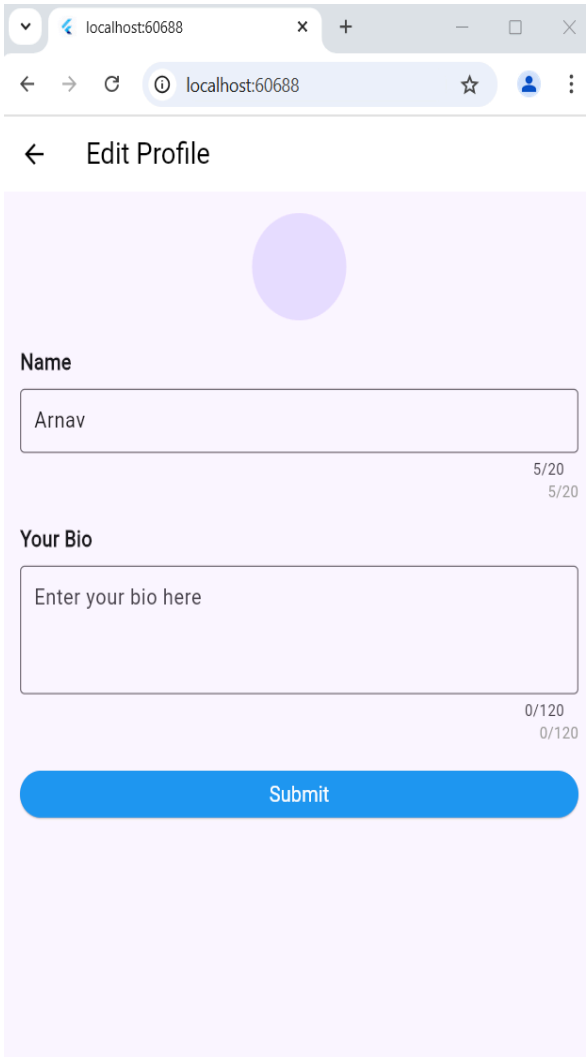
```
import 'package:flutter/material.dart';

class EditProfileScreen extends StatefulWidget
{
  @override
  _EditProfileScreenState createState() =>
_EditProfileScreenState();
}

class _EditProfileScreenState extends
State<EditProfileScreen> {
  TextEditingController nameController =
TextEditingController(text: "Arnav");
  TextEditingController bioController =
TextEditingController();

  int nameMaxLength = 20;
  int bioMaxLength = 120;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: IconButton(
          icon: Icon(Icons.arrow_back, color:
Colors.black),
          onPressed: () => Navigator.pop(context),
        ),
        title: Text("Edit Profile", style:
TextStyle(color: Colors.black)),
        backgroundColor: Colors.white,
        elevation: 0,
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment:
CrossAxisAlignment.start,
          children: [
            // Profile Picture
            Center(
              child: CircleAvatar(
                radius: 40,
                backgroundImage:
AssetImage('assets/profile.jpg'), // Replace with
your image
              ),
            ),
            SizedBox(height: 20),

            // Name Field
            _buildLabel("Name"),
            TextField(
              controller: nameController,
              maxLength: nameMaxLength,
              decoration: InputDecoration(
                border: OutlineInputBorder(),
              ),
            ),

_buildCharacterLimit(nameController.text.lengt
h, nameMaxLength),

            SizedBox(height: 16),

            // Bio Field
            _buildLabel("Your Bio"),
            TextField(
              controller: bioController,
              maxLength: bioMaxLength,
              maxLines: 3,
              decoration: InputDecoration(
```

```
            border: OutlineInputBorder(),
            hintText: "Enter your bio here",
          ),
        ),

_buildCharacterLimit(bioController.text.length,
bioMaxLength),

        SizedBox(height: 20),

        // Submit Button
        SizedBox(
          width: double.infinity,
          child: ElevatedButton(
            onPressed: () {},
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.blue,
              padding:
EdgeInsets.symmetric(vertical: 14),
            ),
            child: Text("Submit", style:
TextStyle(fontSize: 16, color: Colors.white)),
          ),
        ),
          ],
        ),
      ),
    );
  }

  // Helper to create labels
  Widget _buildLabel(String text) {
    return Padding(
      padding: const EdgeInsets.only(bottom: 8),
      child: Text(text, style: TextStyle(fontSize:
16, fontWeight: FontWeight.bold)),
    );
  }

  // Character counter
  Widget _buildCharacterLimit(int current, int
max) {
    return Align(
      alignment: Alignment.centerRight,
      child: Text("$current/$max", style:
TextStyle(color: Colors.grey, fontSize: 12)),
    );
  }
}
```

Screenshots:-

Project Title: Inshorts clone app **Roll No. 52**





MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	52
Name	Arnav Santosh Sawant
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Theory: -

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

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

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

- **Navigation and Routing in Flutter**

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

  1. **Using Navigator Widget**

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

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

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

  2. **Using a list for Named routes**
     Flutter allows the use of a list to manage named routes dynamically, making navigation cleaner and more scalable, especially in larger applications. Instead of writing multiple Navigator.pushNamed() calls, routes can be stored in the list and can be accessed by the index
     Eg:-

```
final List<String> _routes = ['/search', '/login', '/profile', '/settings'];
void _onItemTapped(int index) {
```

```
  Navigator.pushReplacementNamed(context, _routes[index]);
}
```

Handling Gestures in Flutter

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

Tap Gestures

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

Long Press Gesture

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

Swipe and Drag Gestures

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

Code: -

main.dart

```
import 'package:flutter/material.dart';
import 'login/login.dart';
import 'profile/editprofile.dart';
import 'profile/profile.dart';
import 'settings.dart';
import 'search/searchpage.dart';
import 'home.dart'; // Import HomeScreen

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: HomeScreenWrapper(), // Use a
wrapper for bottom navigation
    );
  }
}

class HomeScreenWrapper extends
StatefulWidget {
  @override
  _HomeScreenWrapperState createState() =>
_HomeScreenWrapperState();
}

class _HomeScreenWrapperState extends
State<HomeScreenWrapper> {
  int _selectedIndex = 1;

  final List<Widget> _pages = [
    SearchPage(),
    LoginPage(), // Home Page
    ProfileScreen(),
    SettingsPage(),
  ];

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: IndexedStack(
        index: _selectedIndex,
        children: _pages,
      ),
      bottomNavigationBar:
BottomNavigationBar(
        items: const <BottomNavigationBarItem>[
          BottomNavigationBarItem(
            icon: Icon(Icons.search),
            label: 'Search',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.home),
            label: 'Home',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.account_circle),
            label: 'Profile',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.settings),
            label: 'Settings',
          ),
        ],
        currentIndex: _selectedIndex,
        selectedItemColor: Colors.blue,
        unselectedItemColor: Colors.grey,
        onTap: _onItemTapped,
        type: BottomNavigationBarType.fixed,
      ),
    );
  }
}
```

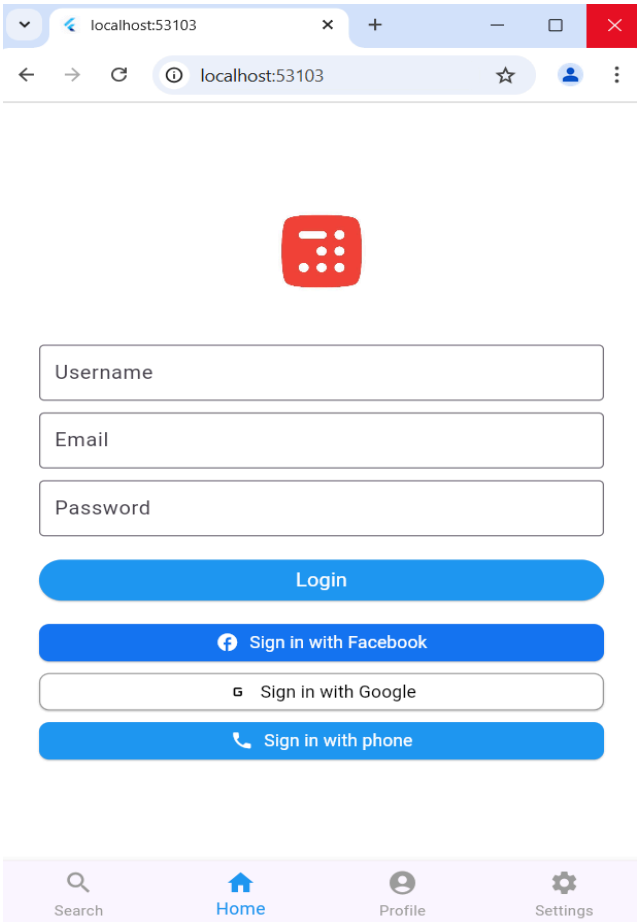
editprofile.dart

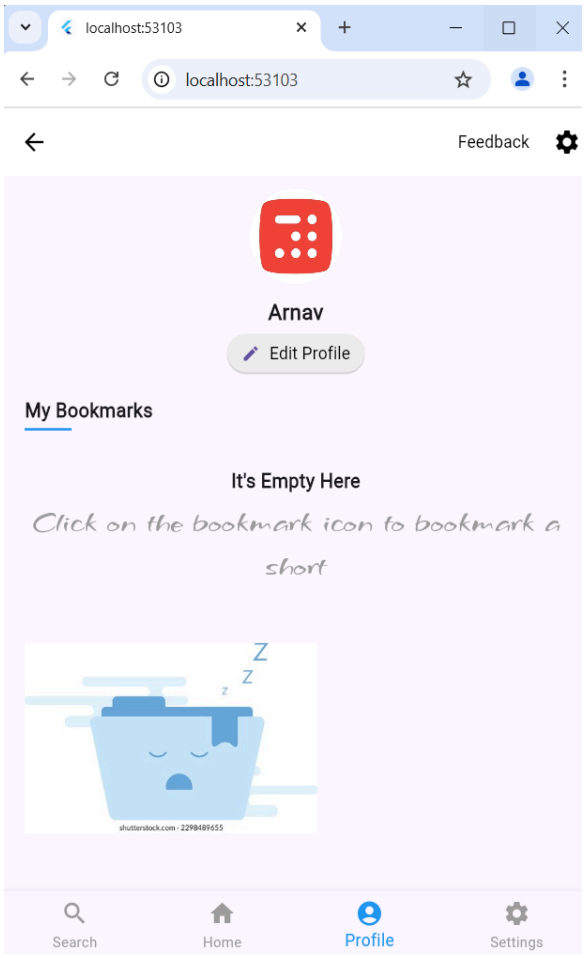
```
import 'package:flutter/material.dart';

class EditProfileScreen extends StatefulWidget
{
 @override
 _EditProfileScreenState createState() =>
_EditProfileScreenState();
}

class _EditProfileScreenState extends
State<EditProfileScreen> {
 TextEditingController nameController =
TextEditingController(text: "Arnav");
 TextEditingController bioController =
TextEditingController();

 int nameMaxLength = 20;
 int bioMaxLength = 120;

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    leading: IconButton(
     icon: Icon(Icons.arrow_back, color:
Colors.black),
     onPressed: () => Navigator.pop(context),
    ),
    title: Text("Edit Profile", style:
TextStyle(color: Colors.black)),
    backgroundColor: Colors.white,
    elevation: 0,
   ),
   body: Padding(
    padding: const EdgeInsets.all(16.0),
    child: Column(
     crossAxisAlignment:
CrossAxisAlignment.start,
     children: [
      // Profile Picture
      Center(
       child: CircleAvatar(
        radius: 40,
        backgroundImage:
AssetImage('assets/profile.jpg'), // Replace with
your image
       ),
      ),
      SizedBox(height: 20),

      // Name Field
      _buildLabel("Name"),
      TextField(
       controller: nameController,
       maxLength: nameMaxLength,
       decoration: InputDecoration(
        border: OutlineInputBorder(),
       ),
      ),

_buildCharacterLimit(nameController.text.lengt
h, nameMaxLength),

      SizedBox(height: 16),

      // Bio Field
      _buildLabel("Your Bio"),
      TextField(
       controller: bioController,
       maxLength: bioMaxLength,
       maxLines: 3,
       decoration: InputDecoration(
        border: OutlineInputBorder(),
        hintText: "Enter your bio here",
       ),
      ),

_buildCharacterLimit(bioController.text.length,
bioMaxLength),

      SizedBox(height: 20),

      // Submit Button
      SizedBox(
       width: double.infinity,
       child: ElevatedButton(
        onPressed: () {},
        style: ElevatedButton.styleFrom(
         backgroundColor: Colors.blue,
         padding:
EdgeInsets.symmetric(vertical: 14),
        ),
        child: Text("Submit", style:
TextStyle(fontSize: 16, color: Colors.white)),
       ),
      ),
```

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

 // Helper to create labels
 Widget _buildLabel(String text) {
  return Padding(
    padding: const EdgeInsets.only(bottom: 8),
    child: Text(text, style: TextStyle(fontSize:
16, fontWeight: FontWeight.bold)),
  );
 }

 // Character counter
 Widget _buildCharacterLimit(int current, int
max) {
   return Align(
     alignment: Alignment.centerRight,
     child: Text("$current/$max", style:
TextStyle(color: Colors.grey, fontSize: 12)),
   );
 }
}
```

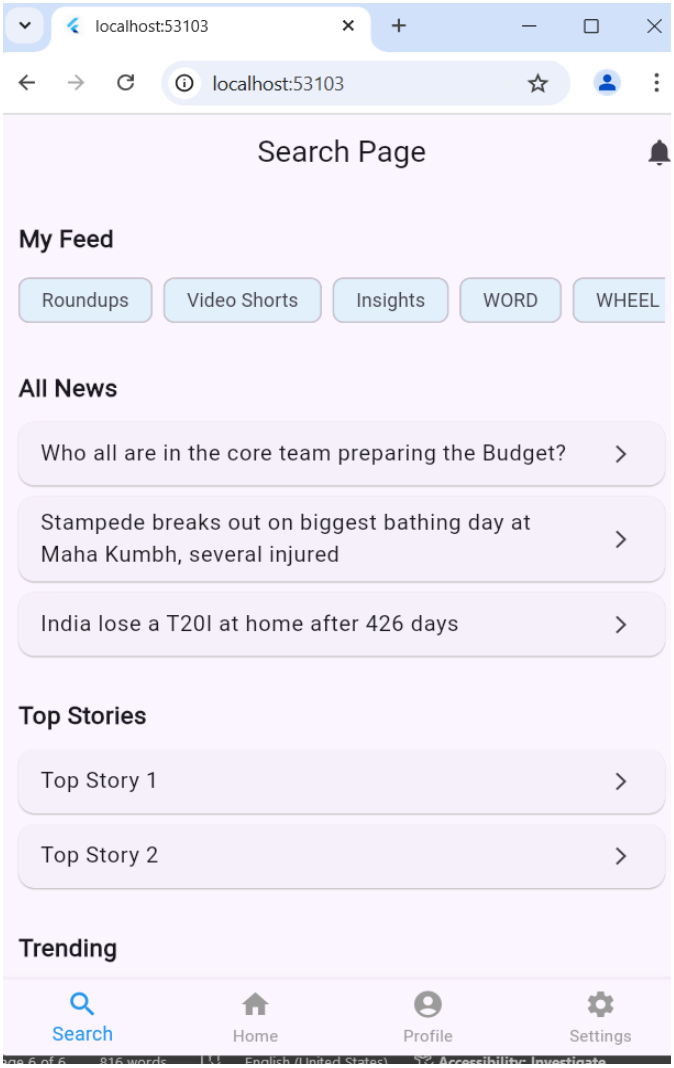
Screenshots:-

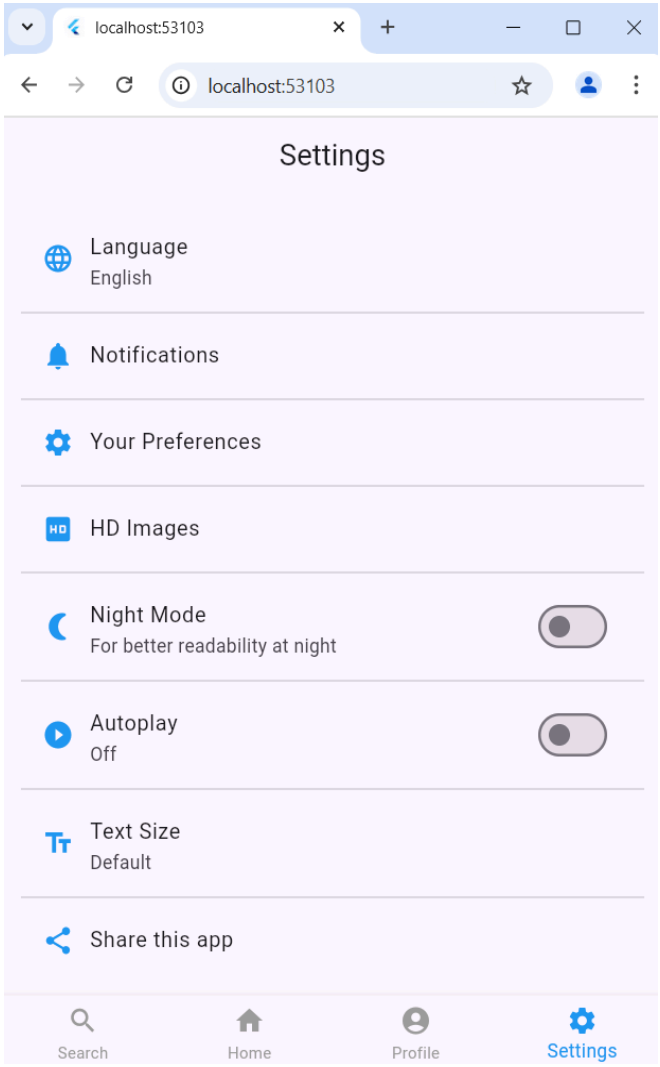
1. The first image is the home screen, when I click on profile it would go to the second image.
2. Then when I click on search, it would go to the third image, then when I click on settings it would go on 4th image
3. Then when I click on 5th image it would go on the 6th Image of editprofile.

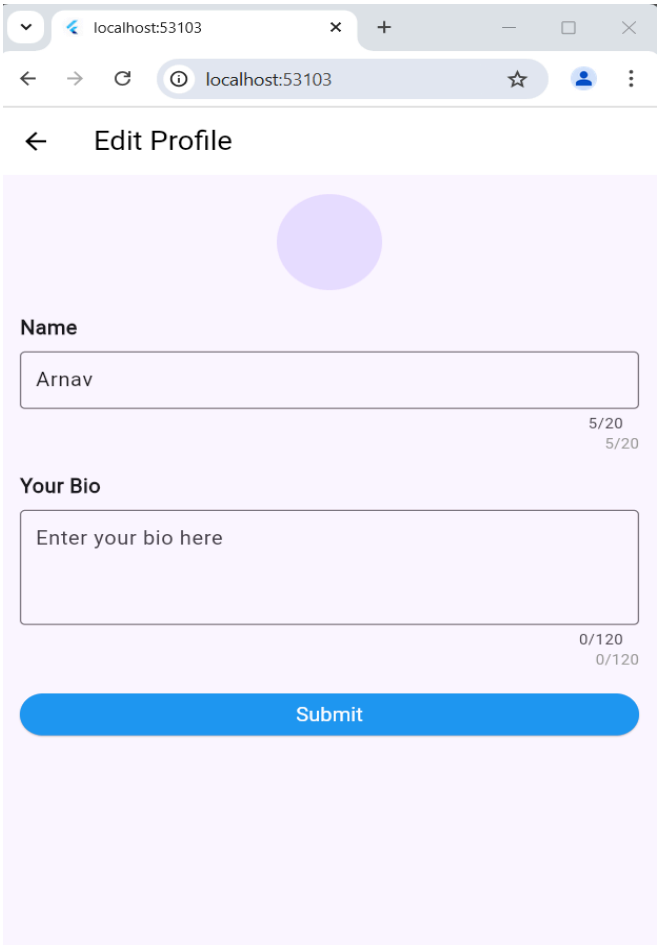




Project Title: Inshorts clone app **Roll No. 52**



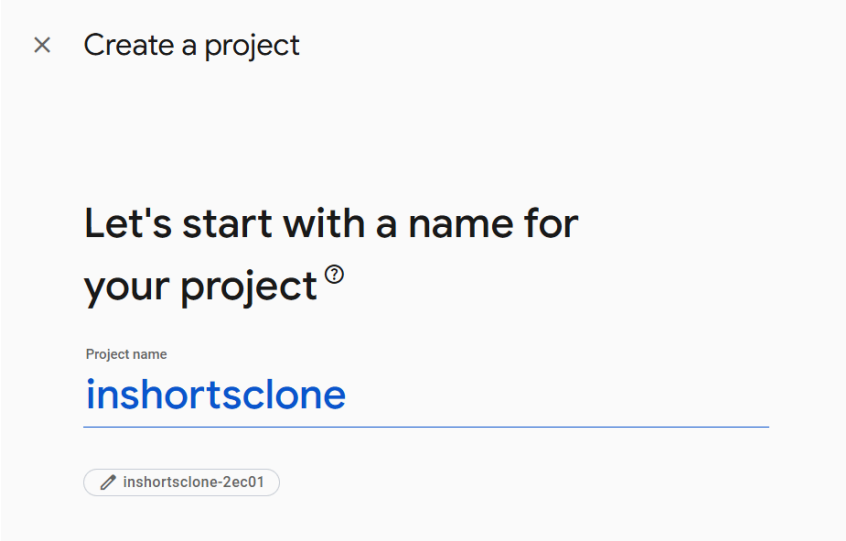



localhost:53103
localhost:53103
Edit Profile
Name
Arnav
5/20
5/20
Your Bio
Enter your bio here
0/120
0/120
Submit

MAD & PWA Lab
Journal

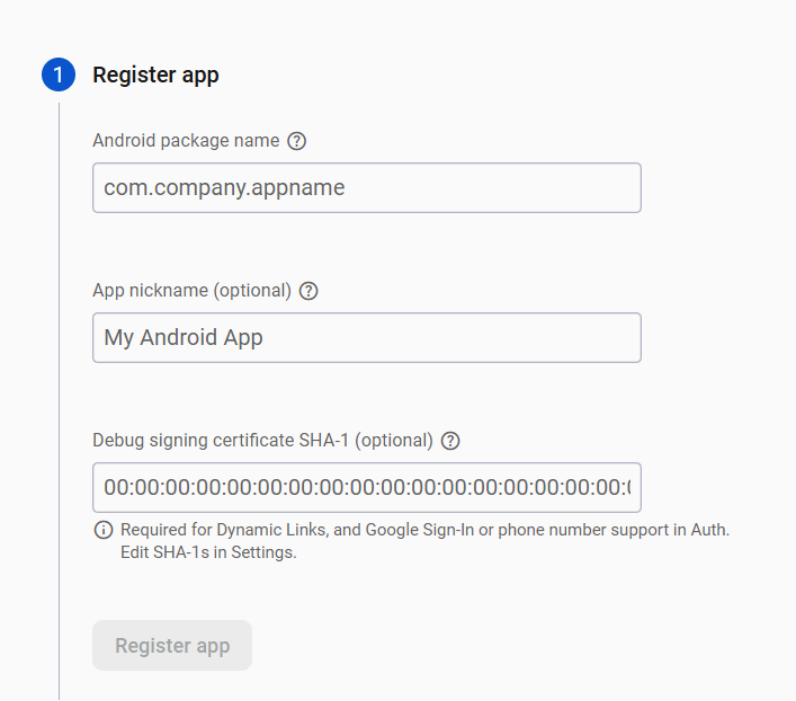
Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	52
Name	Arnav Santosh Sawant
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

Creating a New Firebase Project



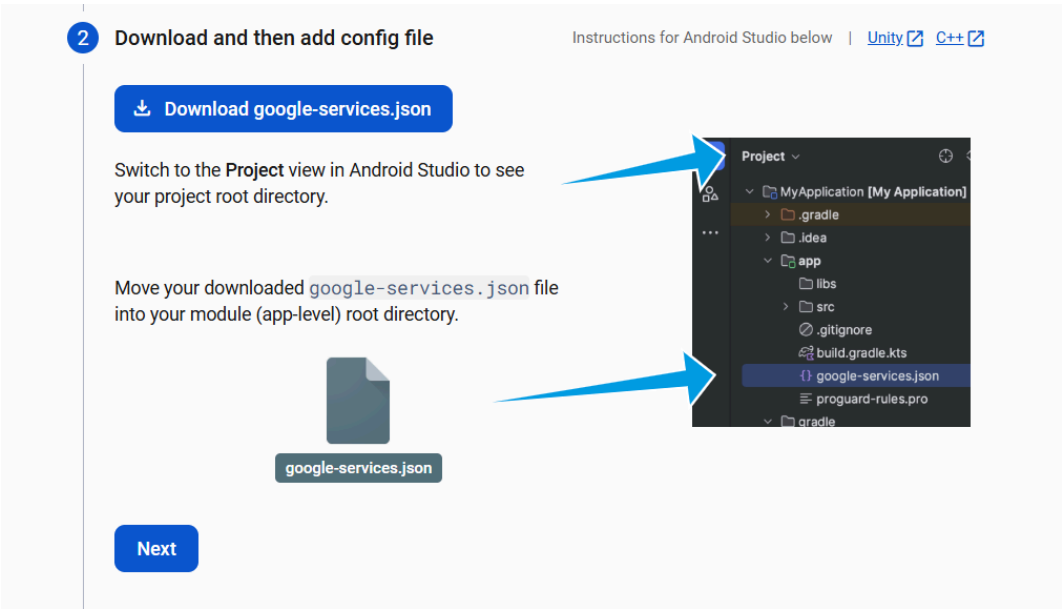
First, log in with your Google account to manage your Firebase projects. From within the Firebase dashboard, select the Create new project button and give it a name

In order to add Android support to our Flutter application, select the Android logo from the dashboard. This brings us to the following screen:

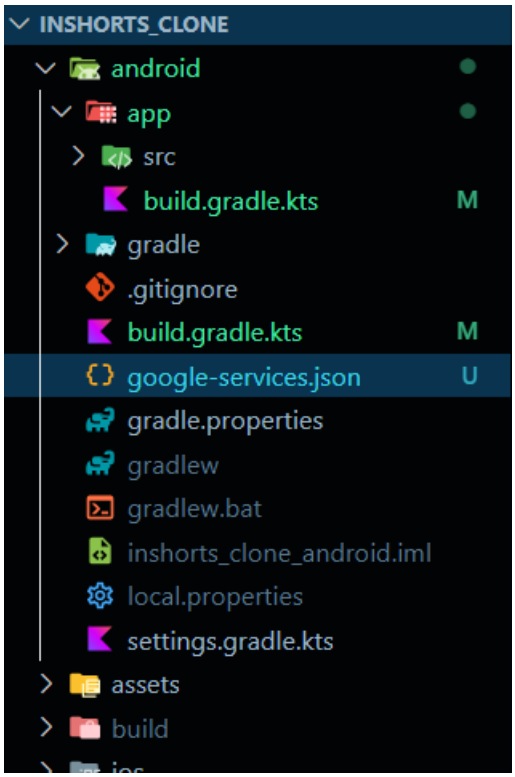


The most important thing here is to match up the Android package name that you choose here with the one inside of our application.

Then download the google-services.json file, that you will get.



put that file in the android folder (root level)



then select the build.gradle.kts (Kotlin DSL) part, and then follow the rest instructions

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.

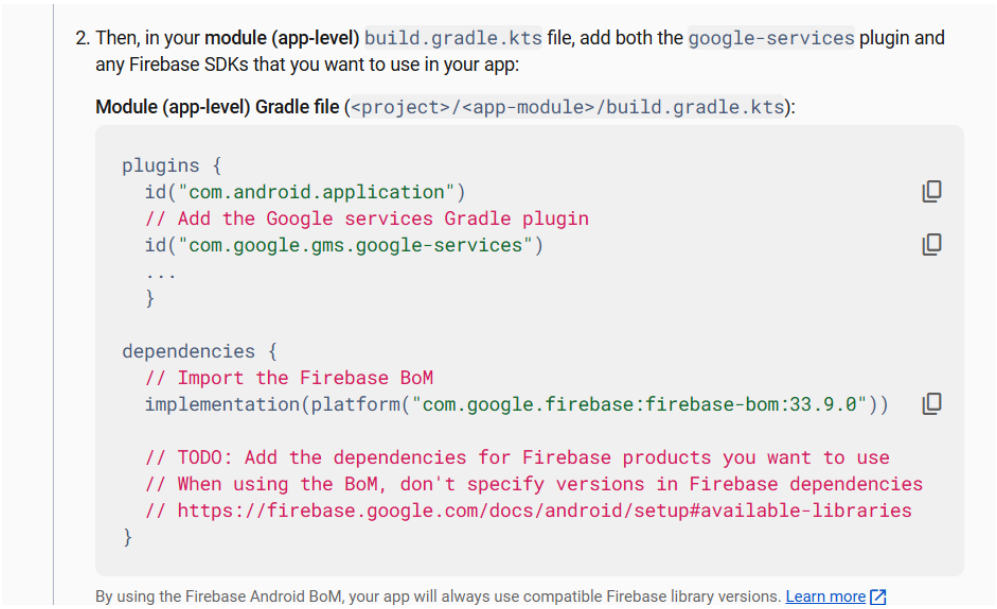
1. To make the `google-services.json` config values accessible to Firebase SDKs, you need the Google services Gradle plugin.

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

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

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

```
plugins {
  // ...

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

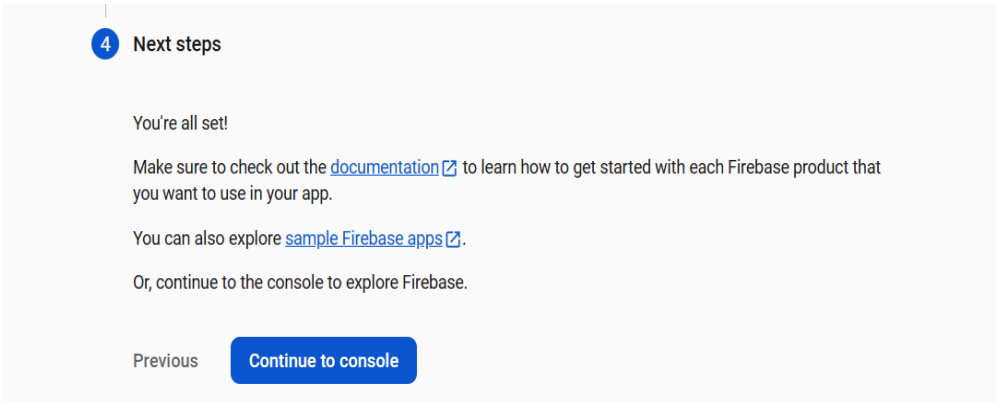


2. Then, in your **module (app-level)** `build.gradle.kts` file, add both the `google-services` plugin and any Firebase SDKs that you want to use in your app:

Module (app-level) Gradle file (`<project>/<app-module>/build.gradle.kts`):

```
plugins {
  id("com.android.application")
  // Add the Google services Gradle plugin
  id("com.google.gms.google-services")
  ...
  }

dependencies {
  // Import the Firebase BoM
  implementation(platform("com.google.firebase:firebase-bom:33.9.0"))

  // TODO: Add the dependencies for Firebase products you want to use
  // When using the BoM, don't specify versions in Firebase dependencies
  // https://firebase.google.com/docs/android/setup#available-libraries
}
```

By using the Firebase Android BoM, your app will always use compatible Firebase library versions. Learn more



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.

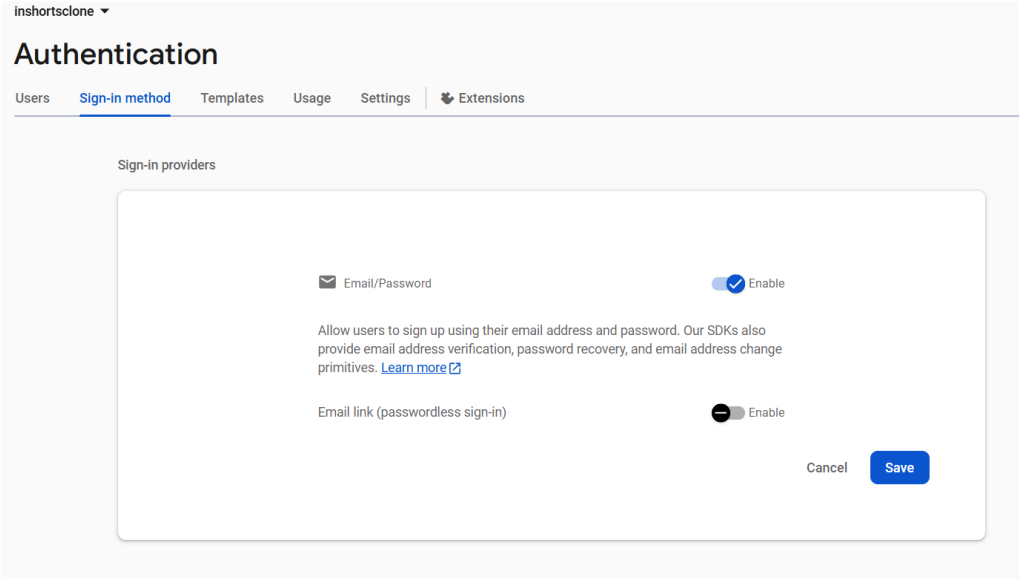
Or, continue to the console to explore Firebase.

Previous Continue to console

Generate the firebase_options.dart file, based on the google-services.json file

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

class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    return const FirebaseOptions(
      apiKey: "AIzaSyA_VIR7fj4d-b6tNZhW9qJ6GRRx5EXKqs0",
      appId: "1:388272292768:android:33180b2382688b18781ac5",
      messagingSenderId: "388272292768",
      projectId: "inshortsclone-848a9",
      storageBucket: "inshortsclone-848a9.firebasestorage.app",
      androidClientId: "1:388272292768:android:33180b2382688b18781ac5",
    );
  }
}
```

In your part select the sign-in method and enable it.



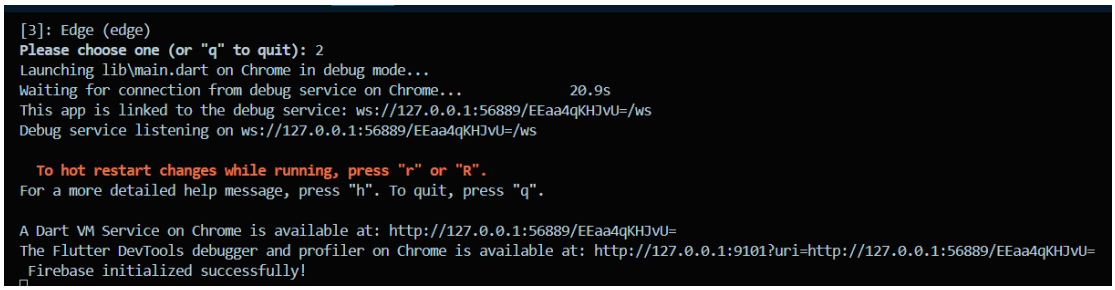
Code:-

```dart
import 'package:flutter/material.dart';
import 'login/login.dart';
import 'profile/editprofile.dart';
import 'profile/profile.dart';
import 'settings.dart';
import 'search/searchpage.dart';
import 'home.dart'; // Import HomeScreen
import 'package:firebase_core/firebase_core.dart';
import 'firebase_options.dart';

void main() async {

WidgetsFlutterBinding.ensureInitialized();
  try {
    await Firebase.initializeApp(
      options: DefaultFirebaseOptions.currentPlatform,
    );
    debugPrint('🔥 Firebase initialized
```

```
successfully!');
  } catch (e) {
    debugPrint('❌ Firebase
initialization failed: $e');
  }
  runApp(MyApp());
}

class MyApp extends StatelessWidget
{
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner:
false,
      home: HomeScreenWrapper(), //
Use a wrapper for bottom navigation
    );
  }
}

class HomeScreenWrapper extends
StatefulWidget {
  @override
  _HomeScreenWrapperState
createState() =>
_HomeScreenWrapperState();
}

class _HomeScreenWrapperState
extends State<HomeScreenWrapper>
{
  int _selectedIndex = 1;

  final List<Widget> _pages = [
    SearchPage(),
    HomeScreen(), // Changed
LoginPage to HomeScreen
    ProfileScreen(),
    SettingsPage(),
  ];

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: IndexedStack(
        index: _selectedIndex,
        children: _pages,
      ),
      bottomNavigationBar:
BottomNavigationBar(
        items: const
<BottomNavigationBarItem>[
          BottomNavigationBarItem(
            icon: Icon(Icons.search),
            label: 'Search',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.home),
            label: 'Home',
          ),
          BottomNavigationBarItem(
            icon:
Icon(Icons.account_circle),
            label: 'Profile',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.settings),
            label: 'Settings',
          ),
        ],
        currentIndex: _selectedIndex,
        selectedItemColor: Colors.blue,
        unselectedItemColor:
Colors.grey,
        onTap: _onItemTapped,
        type:
```

```
BottomNavigationBarType.fixed,
    ),
                                    );
                                  }
                                }
```

Once you run this code this should be your output, if the firebase setup is successful , it would print “Firebase initialized successfully!”.



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.	52
Name	Arnav Santosh Sawant
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":"GlobeNews",
        "short_name":"News",
        "start_url":"index.html",
        "display":"standalone",
        "background_color":"#5900b3
        ", "theme_color":"black",
        "scope": ".",
        "description":"Global news at one stop.",
```

```
        "icons":[
        {
        "src":"assets/newspaperlogo192.png",
        "sizes":"192x192",
        "type":"image/png"
        },
        {
        "src":"assets/newspaperlogo512.png",
        "sizes":"512x512",
        "type":"image/png"
        }
]
}
```

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

```
<html>
   <head>
      <link rel="manifest" href="manifest.json">
      <title>Globenews</title>
      <style>
```

Output:-

Install nodejs

https://nodejs.org/en/download/

In cmd

npm -v

npm install -g browser-sync

```
npm install -g live-server
```

Open vs code

Select proper folder where all files are available

Install extensions node js

In vs code terminal type following commands

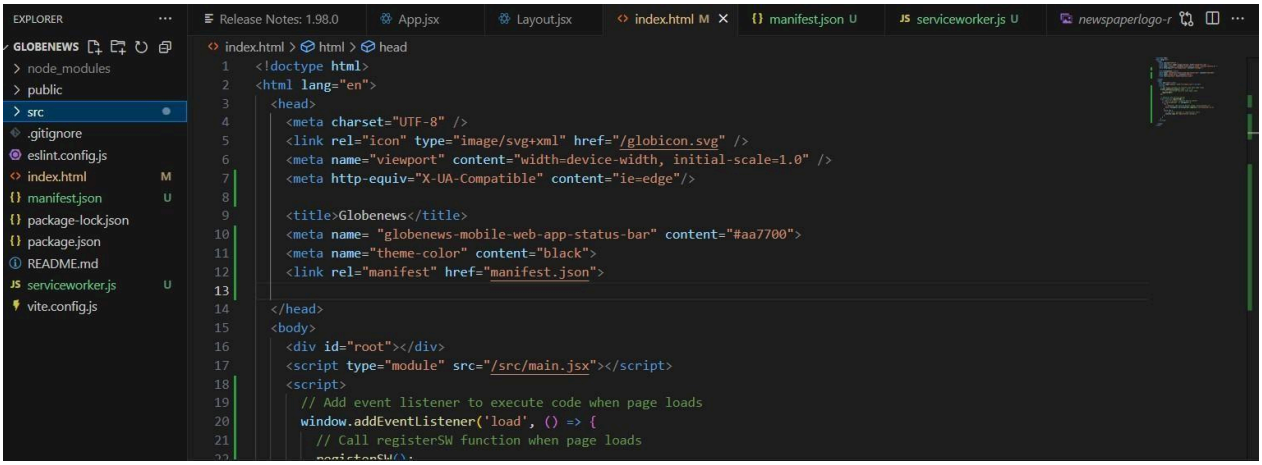
node -v

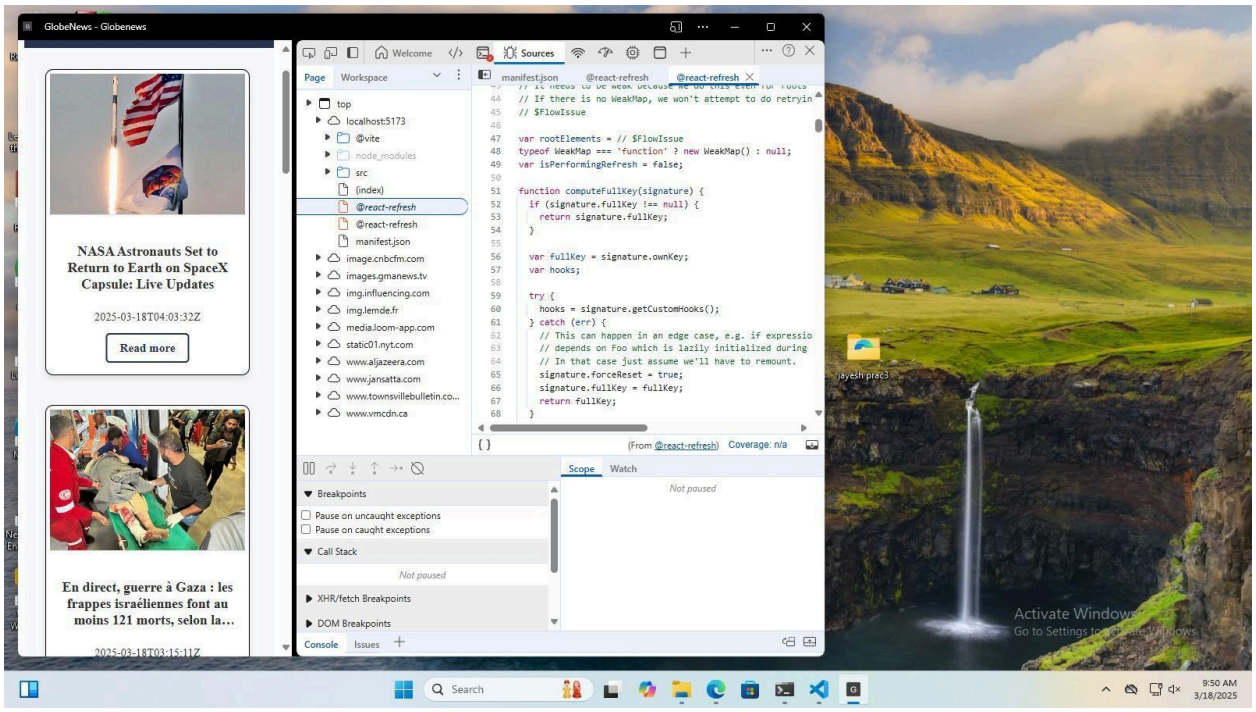
Npx .

Reference https://developer.mozilla.org/en-US/docs/Web/Manifest

https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps

Project Title: Inshorts clone app **Roll No. 52**





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.

https://medium.com/@svinkle/start-a-local-live-reload-web-server-with-one-command-72f99bc6e855

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.	52
Name	Arnav Santosh Sawant
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Service Worker

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

Things to note about Service Worker:

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

What can we do with Service Workers?

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

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

- You can manage **Push Notifications**
  You can manage push notifications with Service Worker and show any information

message to the user.

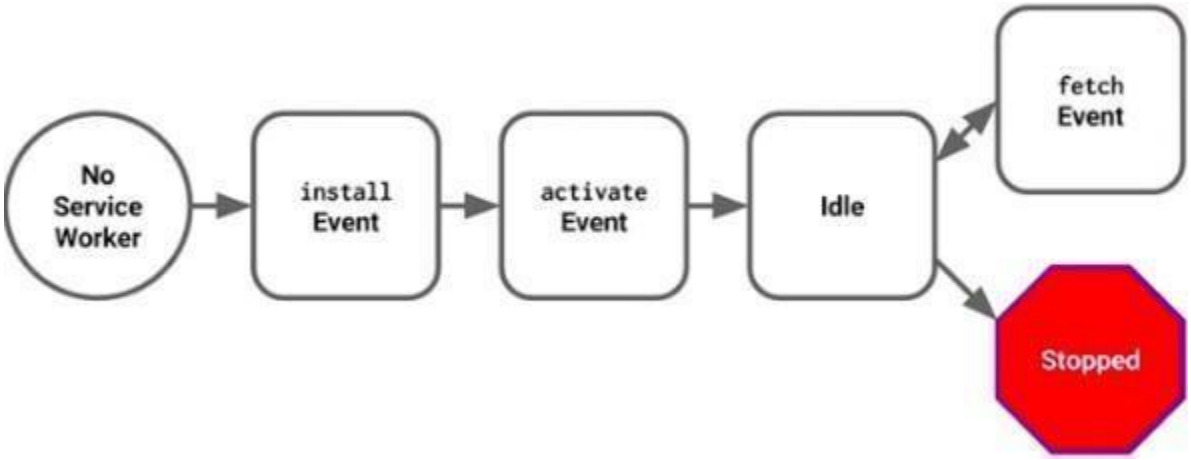
- You can **Continue**
  Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

What can't we do with Service Workers?

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

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

Service Worker Cycle



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

- Registration
- Installation
- Activation

Registration

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

main.js:-

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

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

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

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

example: main.js

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

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

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

main.js

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

Installation

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

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

service-worker.js

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

Activation

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

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

service-worker.js

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

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

Code:

Index.html

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

main.jsx

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

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
// Service Worker Registration
if ('serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker.register('public/sw.js') // Path to your service worker
file
      .then((registration) => {
        console.log('[Service Worker] Registered with scope:', registration.scope);

        // Check if the service worker is active
        if (registration.active) {
        console.log('[Service Worker] Active');
```

```
      }

      // Check if the service worker is installing
      if (registration.installing) {
      console.log('[Service Worker] Installing');
      }

      // Check if the service worker is waiting
      if (registration.waiting) {
      console.log('[Service Worker] Waiting');
      }

      // Listen for state changes
      registration.addEventListener('updatefound', () =>
      { console.log('[Service Worker] Update found');
      const installingWorker = registration.installing;
        if (installingWorker) {
          installingWorker.addEventListener('statechange', () => {
          if (installingWorker.state === 'installed') {
              if (navigator.serviceWorker.controller) {
                console.log('[Service Worker] New content is available and will be used when all tabs for this page are
closed');                                                                    // You can prompt the user to reload here if needed
                                                                           } else {
                                                                             console.log('[Service Worker] Content is cached for
                                                                             offline use.');
                                                                            }
                                                                           }
                                                                          });
        }
       });
     })

     .catch((error) => {
       console.error('[Service Worker] Registration failed:', error);
     });
  });
} else {
  console.warn('[Service Worker] Not supported in this browser.');
}
```

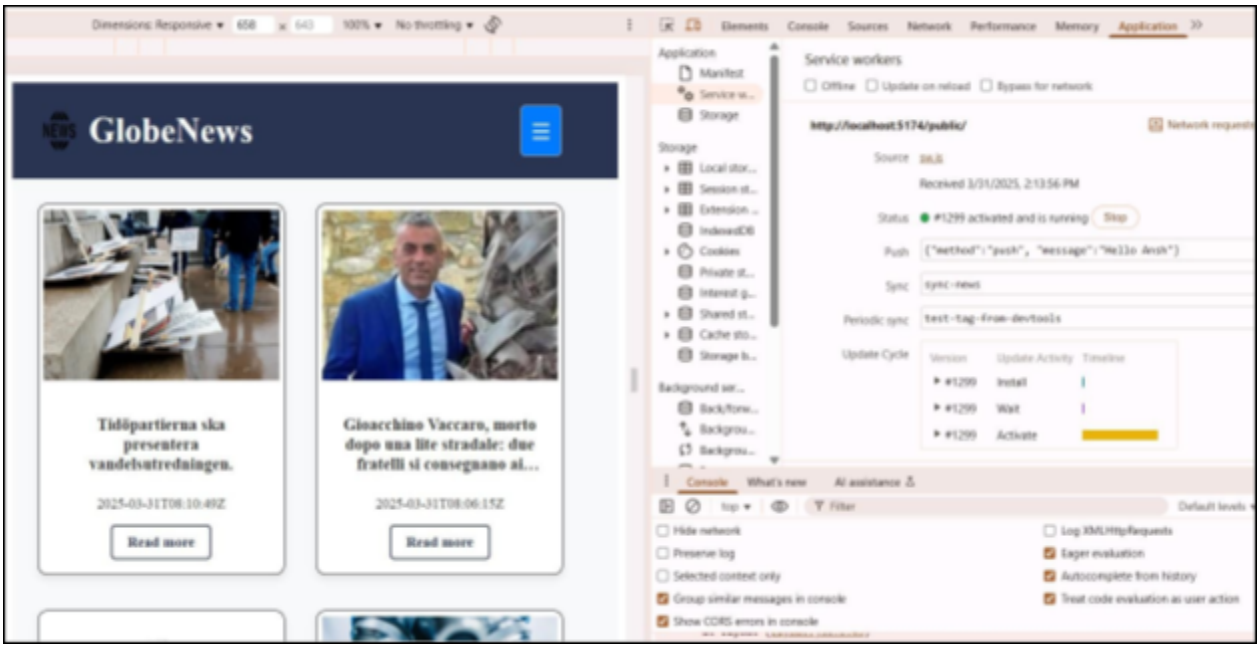
sw.js

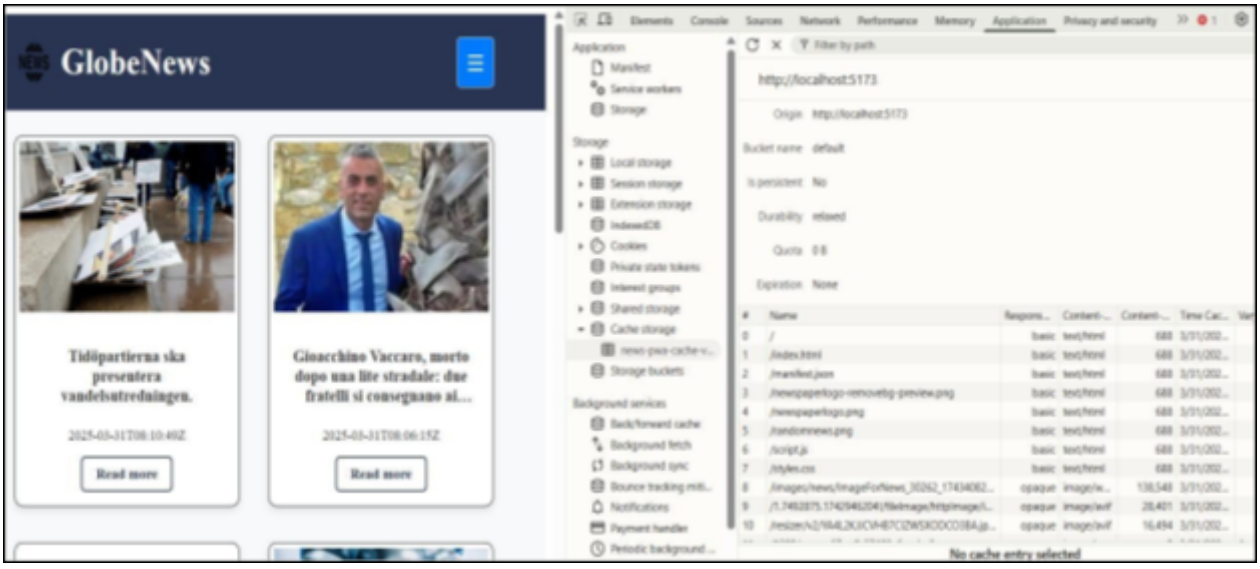
```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import App from './App.jsx'
import './index.css'

createRoot(document.getElementById('root')).render(
  <StrictMode>
```

```
    <App />
  </StrictMode>,
)
// Service Worker Registration
if ('serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker.register('public/sw.js') // Path to your service worker
file
      .then((registration) => {
        console.log('[Service Worker] Registered with scope:', registration.scope);
        // Check if the service worker is active
        if (registration.active) {
        console.log('[Service Worker] Active');
        }
        // Check if the service worker is installing
        if (registration.installing) {
        console.log('[Service Worker] Installing');
        }
        // Check if the service worker is waiting
        if (registration.waiting) {
        console.log('[Service Worker] Waiting');
        }
        // Listen for state changes
        registration.addEventListener('updatefound', () =>
        { console.log('[Service Worker] Update found');
        const installingWorker = registration.installing;
          if (installingWorker) {
            installingWorker.addEventListener('statechange', () => {
            if (installingWorker.state === 'installed') {
                if (navigator.serviceWorker.controller) {
                  console.log('[Service Worker] New content is available and will be used when all tabs for this page are
closed');                                                                    // You can prompt the user to reload here if needed
                                                                             } else {
                                                                               console.log('[Service Worker] Content is cached for
                                                                               offline use.');
                                                                               }
                                                                             }
                                                                           });
          }
        });
      })

      .catch((error) => {
        console.error('[Service Worker] Registration failed:', error);
      });
  });
} else {
  console.warn('[Service Worker] Not supported in this browser.');
}
```

Project Title: Inshorts clone app **Roll No. 52**

Output:





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.	52
Name	Arnav Santosh Sawant
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

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

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

messages to the page.

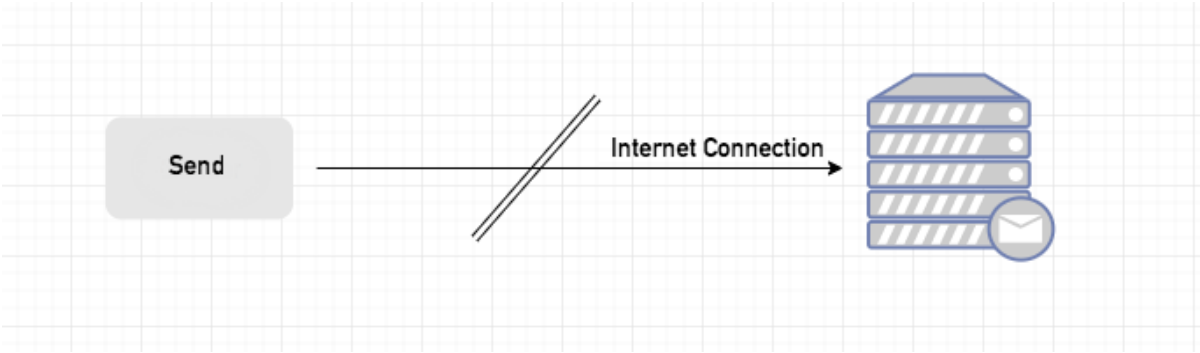
```
self.addEventListener("fetch", function (event) {
    const req = event.request;
    const url = new URL(req.url);

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

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

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

Sync Event

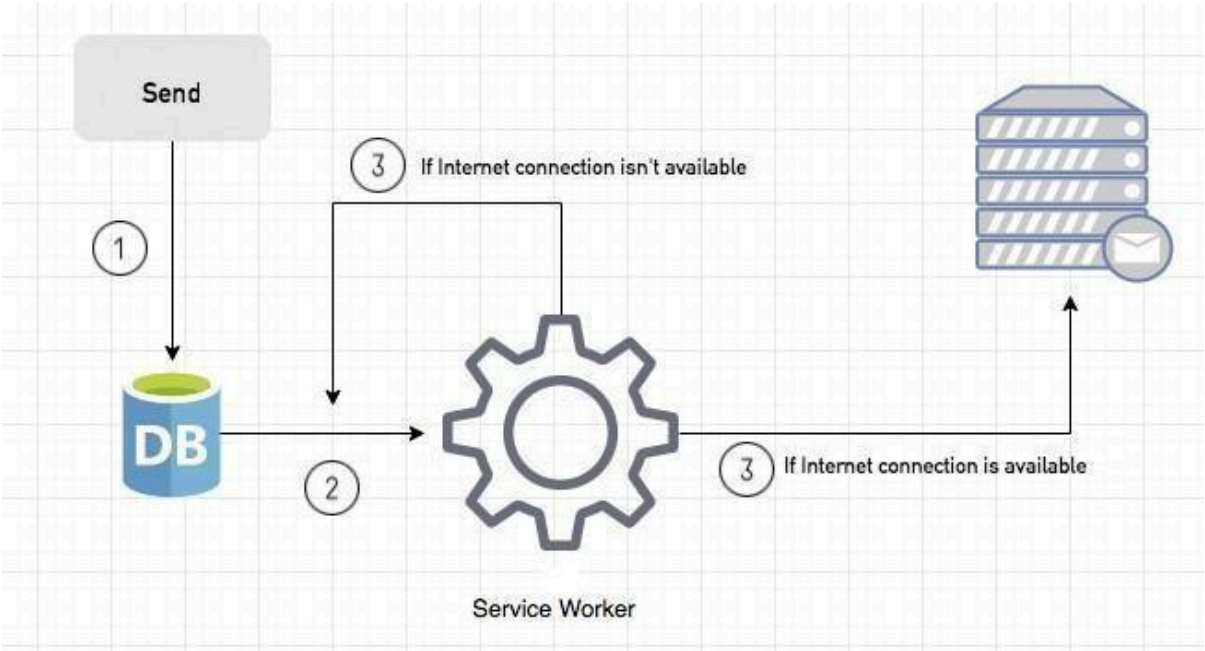
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

Here is a job for the Background Sync.

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



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



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

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

Event Listener for Background Sync Registration

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

Event Listener for sw.js

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

Push Event

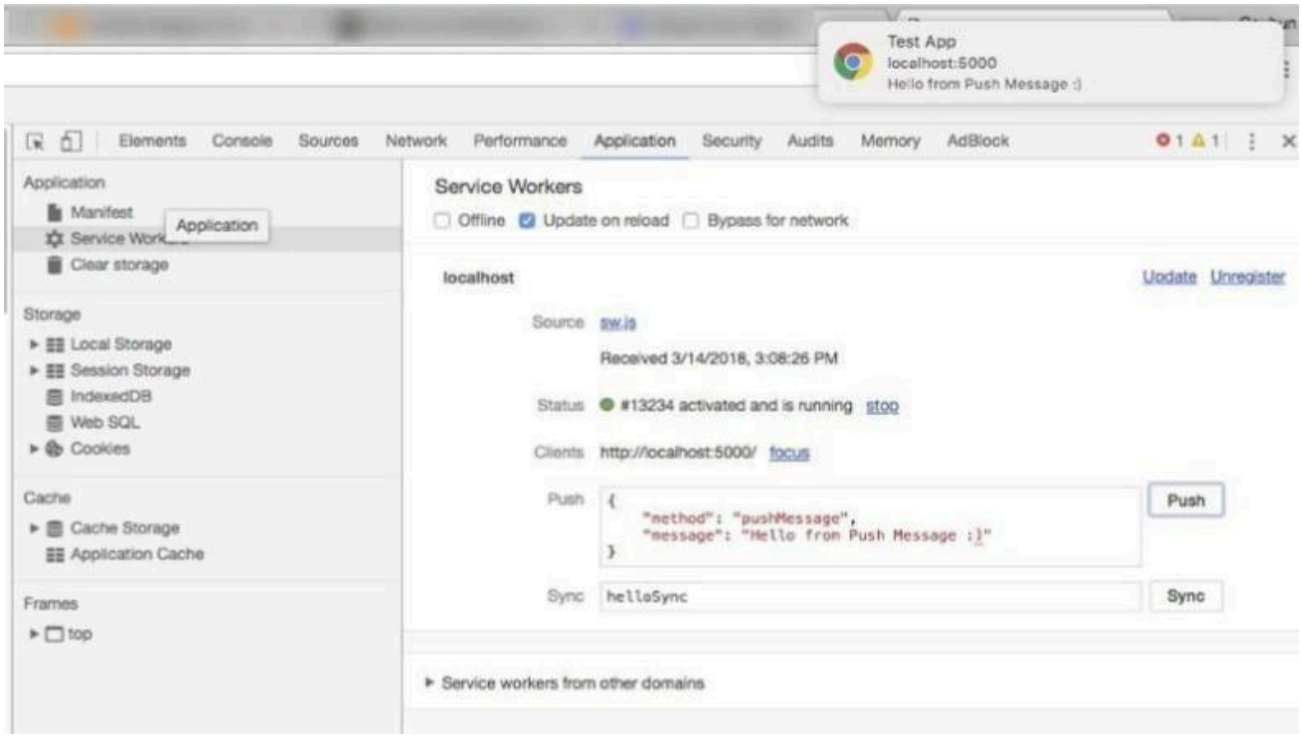
This is the event that handles push notifications that are received from the server. You can apply any method with received data.

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

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

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

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



Code:

service-worker.js const CACHE_NAME = 'news-pwa-cache-v1';
const urlsToCache = [
'/',
'/index.html',
'/styles.css',
'/script.js',
'/manifest.json',
'/newspaperlogo.png',
'/randomnews.png',
'/newspaperlogo-removebg-preview.png'
];

// Install event: Cache static assets
self.addEventListener('install', (event) => {
console.log('[Service Worker] Installing...');
event.waitUntil(
caches.open(CACHE_NAME).then((cache) => {
console.log('[Service Worker] Caching assets...');
return cache.addAll(urlsToCache);
})
);
self.skipWaiting();
});

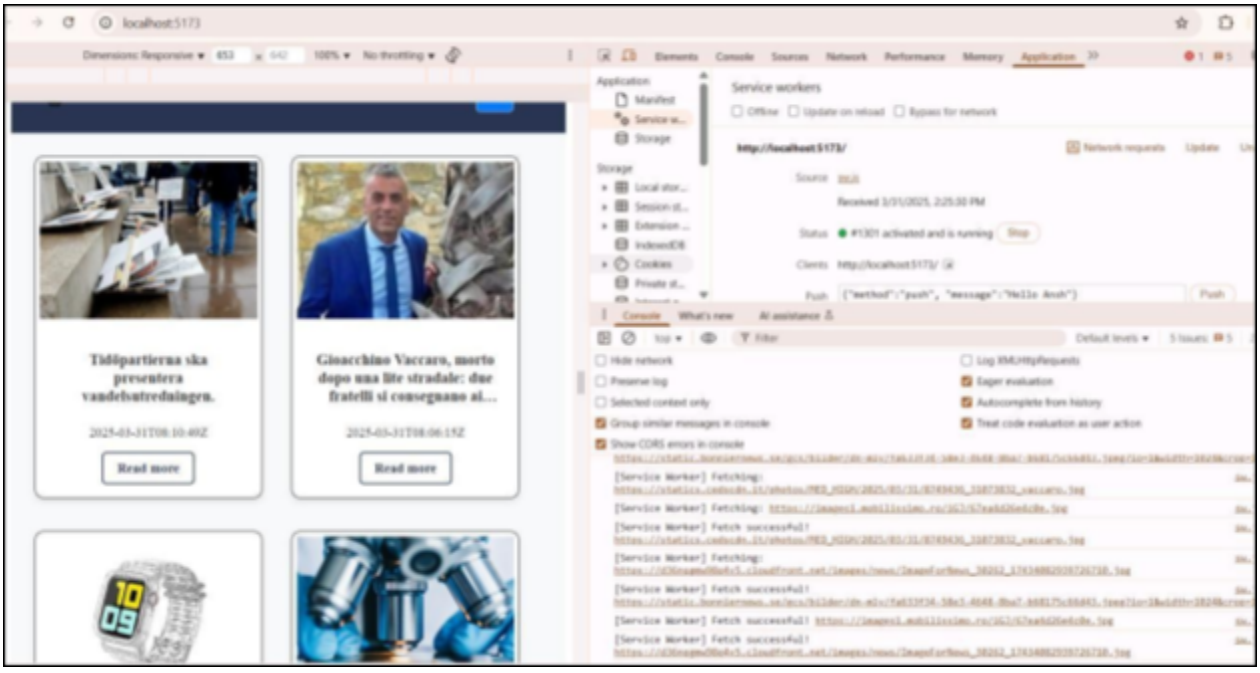
// Activate event: Cleanup old caches
self.addEventListener('activate', (event) => {

```
  console.log('[Service Worker] Activating...');
  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);
         }
       })
     );
   })
  );
  self.clients.claim();
});
// Fetch event: Cache-first strategy
// Fetch event: Cache-first strategy with logging
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] Serving from cache:', event.request.url);
        return cachedResponse;
       }

       return fetch(event.request)
         .then((networkResponse) => {
           console.log('[Service Worker] Fetch successful!', event.request.url);
           return caches.open(CACHE_NAME).then((cache) => {
           cache.put(event.request, networkResponse.clone());
             return networkResponse;
           });
         })
         .catch((error) => {
           console.error('[Service Worker] Fetch failed:', error);
           return new Response('Network error!', { status: 408 });
         });
     })
   );
 });

// Handle Push Notifications
self.addEventListener('push', (event) => {
   if (!event.data) {
     console.error('[Service Worker] Push event has NO data!');
     return;
   }

   const data = event.data.json();
```

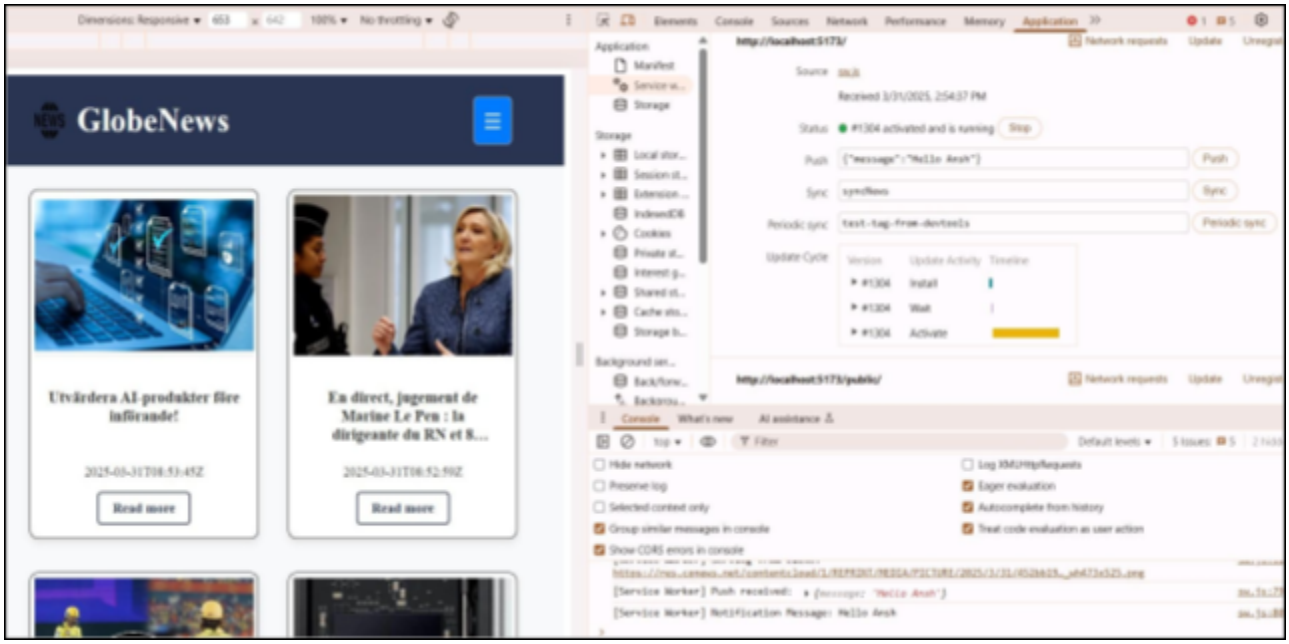
```
    console.log('[Service Worker] Push received:', data); // Log full push data
    console.log('[Service Worker] Notification Message:', data.message); // Log message only

    const options = {
      body: data.message || 'Breaking News!',
      icon: '/newspaperlogo.png',
      badge: '/newspaperlogo-removebg-preview.png'
    };

    event.waitUntil(
      self.registration.showNotification(data.title || 'News Alert', options)
    );
  });
// Handle Notification Click
self.addEventListener('notificationclick', (event) => {
event.notification.close();
event.waitUntil(clients.openWindow('/'));
});
// Background Sync
self.addEventListener('sync', (event) => {
if (event.tag === 'syncNews') {
    console.log('[Service Worker] Sync successful!');
  }
});
```

Output:

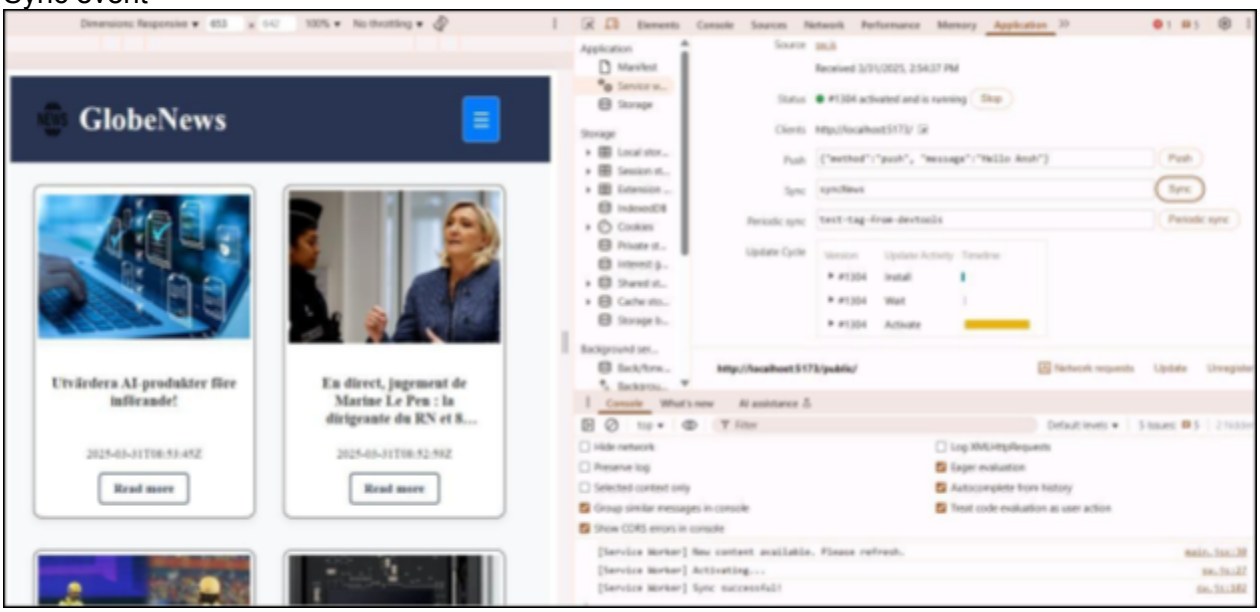
Fetch event



Push event


GlobeNews
Utvärdera AI-produkter före införande!
Read more
En direct, jugement de Marine Le Pen : la dirigeante du RN et s...
Read more

Sync event


GlobeNews
Utvärdera AI-produkter före införande!
Read more
En direct, jugement de Marine Le Pen : la dirigeante du RN et s...
Read more

MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	52
Name	Arnav Santosh Sawant
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

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

GitHub Pages provides the following key features:

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

Reasons for favoring this over Firebase:

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

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

Pros

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

Cons

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

Firebase

The Realtime App Platform. Firebase is a cloud service designed to power real-time, collaborative applications. Simply add the Firebase library to your application to gain access to a

shared data structure; any changes you make to that data are automatically synchronized with the Firebase cloud and with other clients within milliseconds.

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

Pros

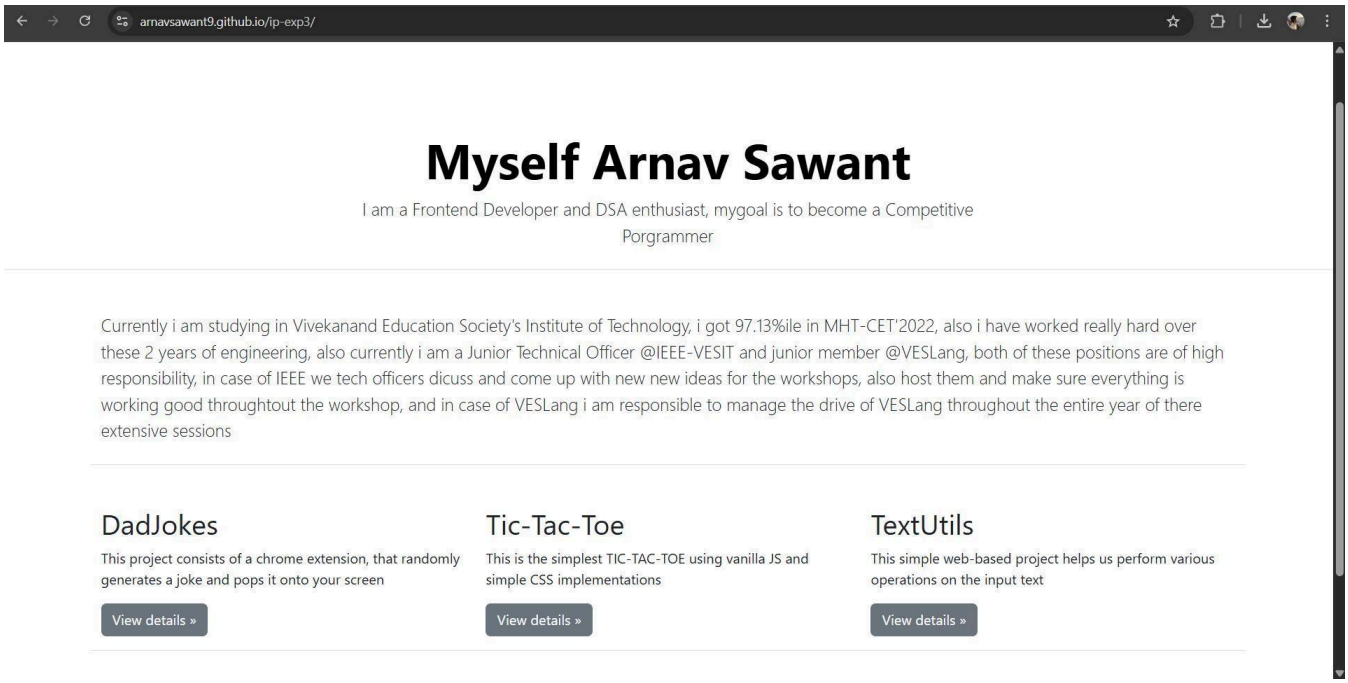
1. Hosted by Google. Enough said.
2. Authentication, Cloud Messaging, and a whole lot of other handy services will be available to you.
3. A real-time database will be available to you, which can store 1 GB of data.
4. You'll also have access to a blob store, which can store another 1 GB of data.
5.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

Cons

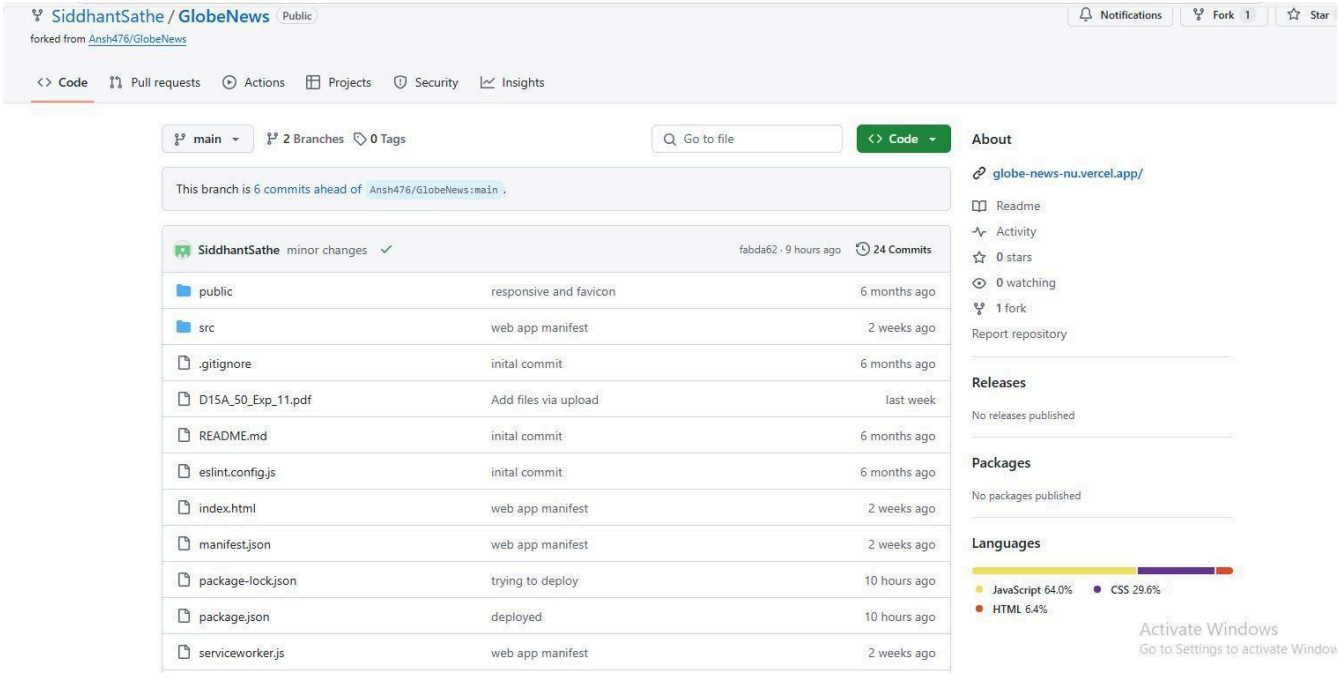
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator.

Link to our GitHub repository:

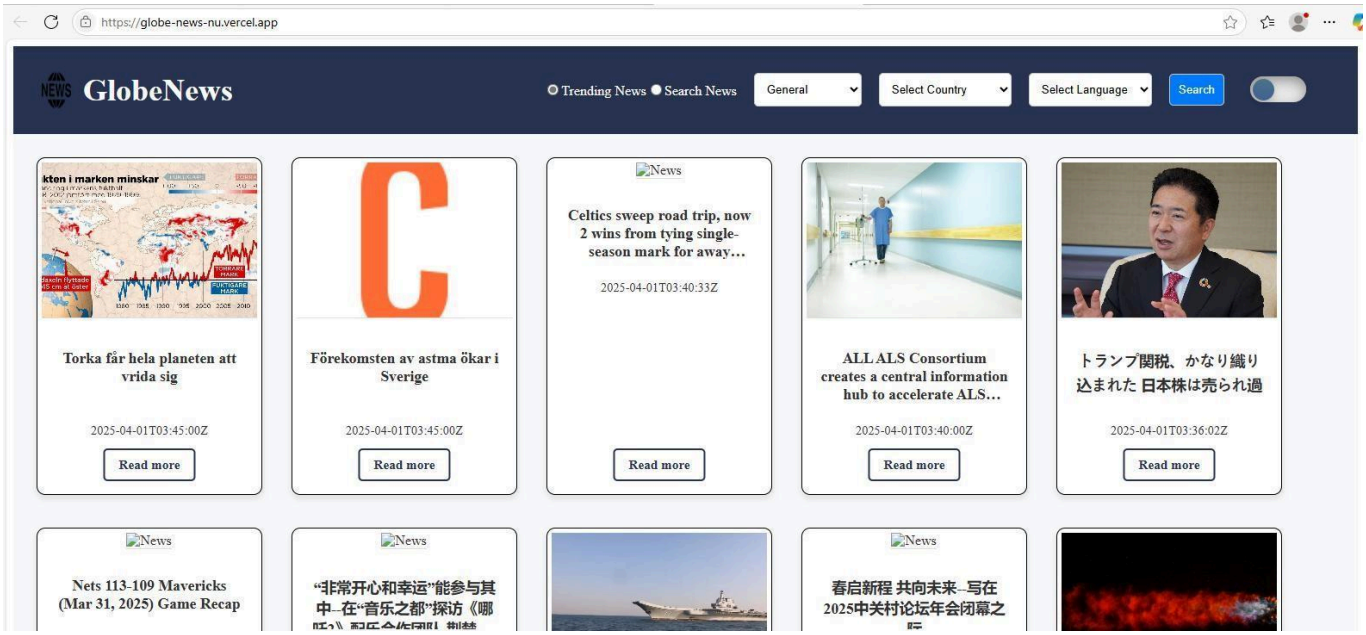
Github Screenshot:


arnavsawant9.github.io/ip-exp3/
Myself Arnav Sawant
I am a Frontend Developer and DSA enthusiast, mygoal is to become a Competitive Porgrammer
Currently i am studying in Vivekanand Education Society's Institute of Technology, i got 97.13%ile in MHT-CET'2022, also i have worked really hard over these 2 years of engineering, also currently i am a Junior Technical Officer @IEEE-VESIT and junior member @VESLang, both of these positions are of high responsibility, in case of IEEE we tech officers dicuss and come up with new new ideas for the workshops, also host them and make sure everything is working good throughtout the workshop, and in case of VESLang i am responsible to manage the drive of VESLang throughout the entire year of there extensive sessions
DadJokes
This project consists of a chrome extension, that randomly generates a joke and pops it onto your screen
View details »
Tic-Tac-Toe
This is the simplest TIC-TAC-TOE using vanilla JS and simple CSS implementations
View details »
TextUtils
This simple web-based project helps us perform various operations on the input text
View details »

PWA Website:


SiddhantSathe / GlobeNews Public
forked from Ansh476/GlobeNews
Notifications Fork 1 Star
Code Pull requests Actions Projects Security Insights
main 2 Branches 0 Tags Go to file Code
This branch is 6 commits ahead of Ansh476/GlobeNews:main .
SiddhantSathe minor changes fabda62 · 9 hours ago 24 Commits
public responsive and favicon 6 months ago
src web app manifest 2 weeks ago
.gitignore inital commit 6 months ago
D15A_50_Exp_11.pdf Add files via upload last week
README.md inital commit 6 months ago
eslint.config.js inital commit 6 months ago
index.html web app manifest 2 weeks ago
manifest.json web app manifest 2 weeks ago
package-lock.json trying to deploy 10 hours ago
package.json deployed 10 hours ago
serviceworker.js web app manifest 2 weeks ago
About
globe-news-nu.vercel.app/
Readme
Activity
0 stars
0 watching
1 fork
Report repository
Releases
No releases published
Packages
No packages published
Languages
JavaScript 64.0%
CSS 29.6%
HTML 6.4%
Activate Windows
Go to Settings to activate Windows

Project Title: Inshorts clone app **Roll No. 52**



Github Website: [GitHub - Ansh476/GlobeNews]

Hosted Website: globe-news-nu.vercel.app/

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	52
Name	Arnav Santosh Sawant
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.

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

Key Features and Audit Metrics

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

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

Service Worker implementation(s), viewport handling, offline functionality, performance in script-disabled environments, etc.

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

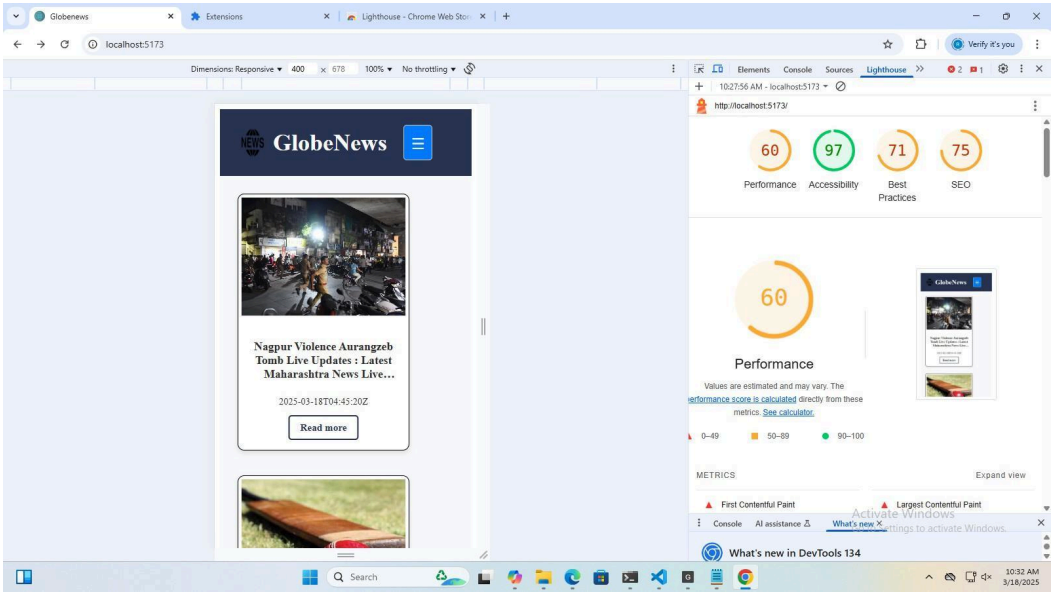
i.e. if all possible elements of the page are not screen-reader friendly (HTML5 introduced features that would make pages easy to interpret for screen readers used by visually challenged people like tag names, tags such as <section>, <article>, etc.), you get a 0 on that score. The aggregate of these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

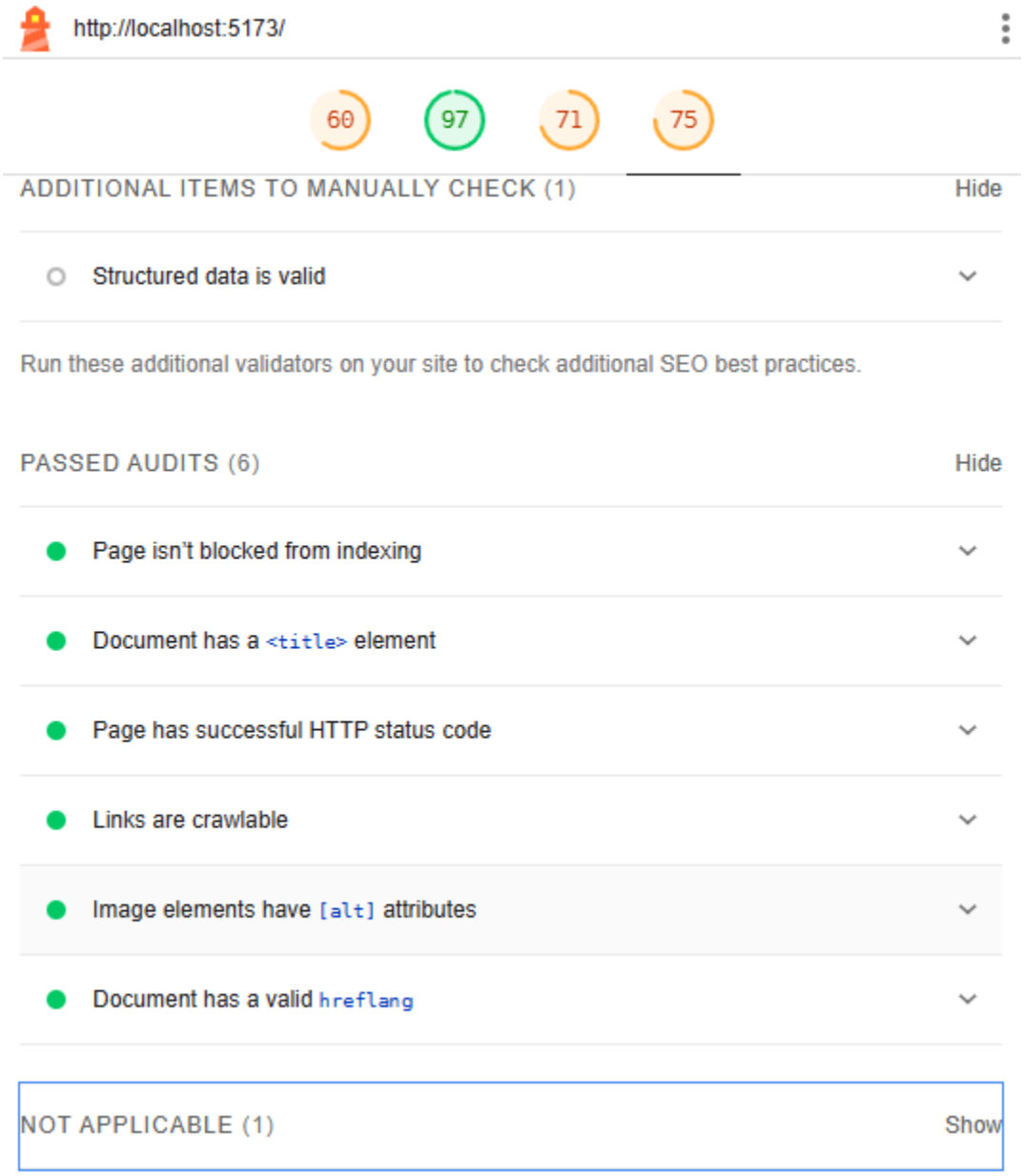
1. **Best Practices:** As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS

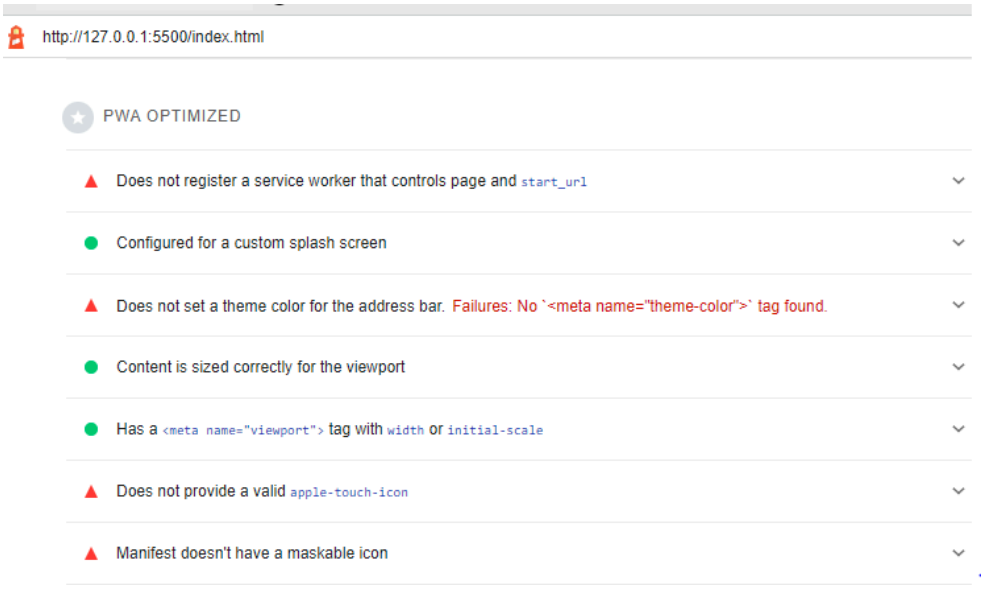
   Avoiding the use of deprecated code elements like tags, directives, libraries, etc. Password input with paste-into disabled

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

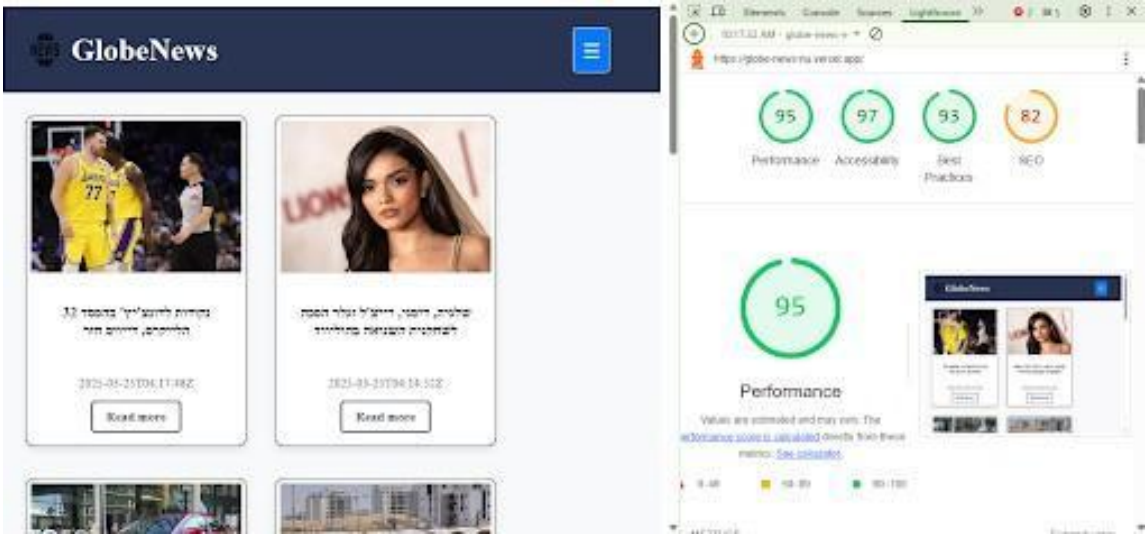
Before :

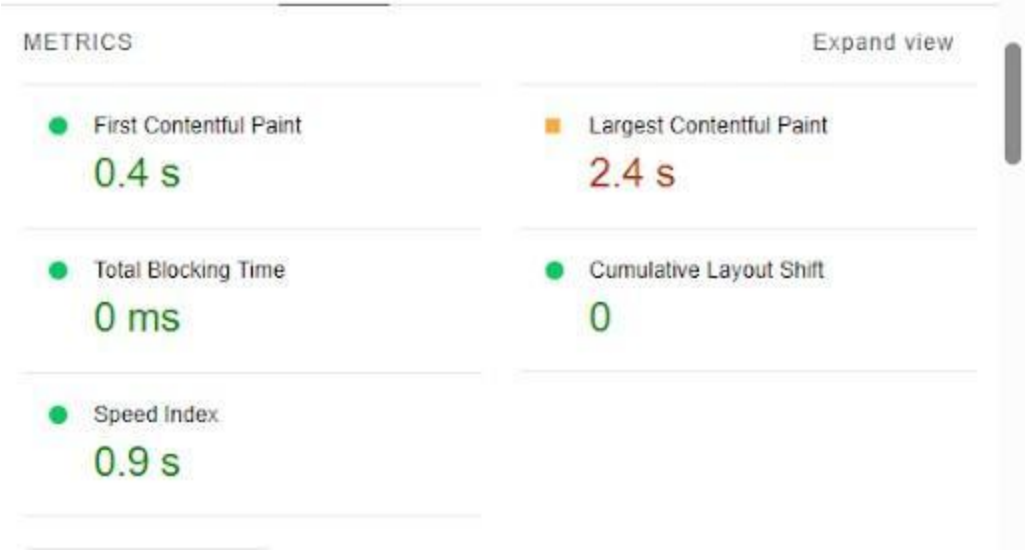



http://localhost:5173/
60
97
71
75
ADDITIONAL ITEMS TO MANUALLY CHECK (1)
Hide
Structured data is valid
Run these additional validators on your site to check additional SEO best practices.
PASSED AUDITS (6)
Hide
Page isn't blocked from indexing
Document has a <title> element
Page has successful HTTP status code
Links are crawlable
Image elements have [alt] attributes
Document has a valid hreflang
NOT APPLICABLE (1)
Show


http://127.0.0.1:5500/index.html
PWA OPTIMIZED
Does not register a service worker that controls page and start_url
Configured for a custom splash screen
Does not set a theme color for the address bar. Failures: No `<meta name="theme-color">` tag found.
Content is sized correctly for the viewport
Has a <meta name="viewport"> tag with width or initial-scale
Does not provide a valid apple-touch-icon
Manifest doesn't have a maskable icon

After:





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