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

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Project Title: Smart Farming App (MAD) & College Website (PWA) **Roll No. 18**

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

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

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers : Mrs. Kajal Joseph.

Email : kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: Smart Farming App (MAD) & College Website (PWA) **Roll No. 18**

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

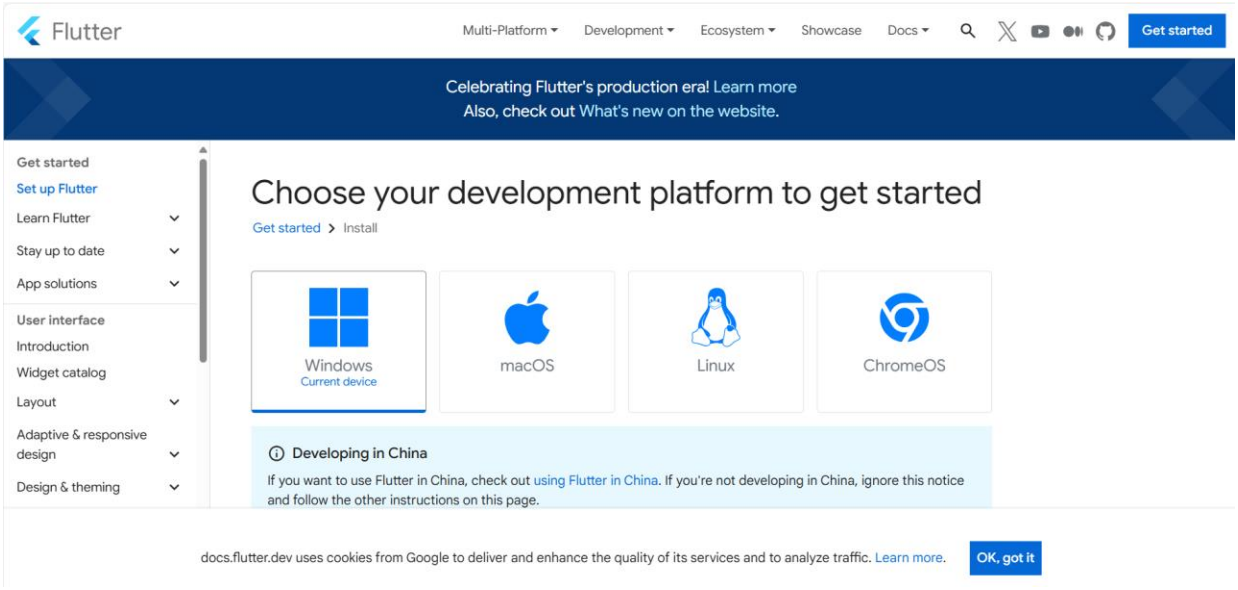
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	18
Name	Tejas Dhondibhau Gunjal
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

EXPERIMENT NO: - 01

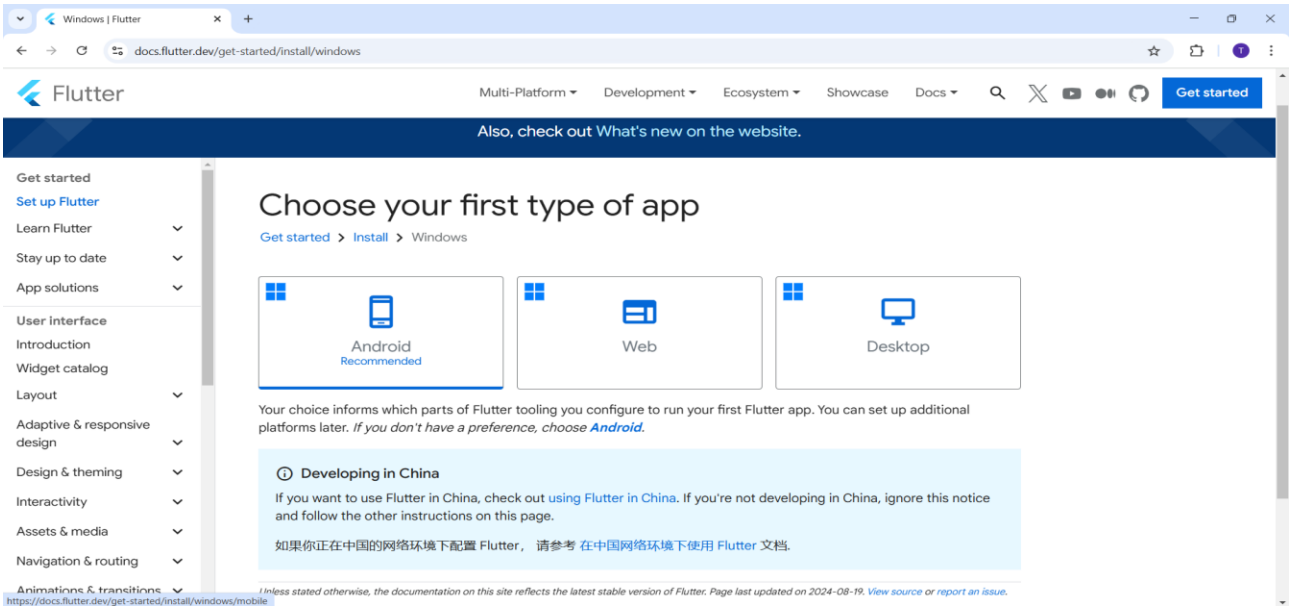
Name:- Tejas Gunjal **Class:-** D15A **Roll:No: -** 18

AIM: - Installation and Configuration of Flutter Environment.

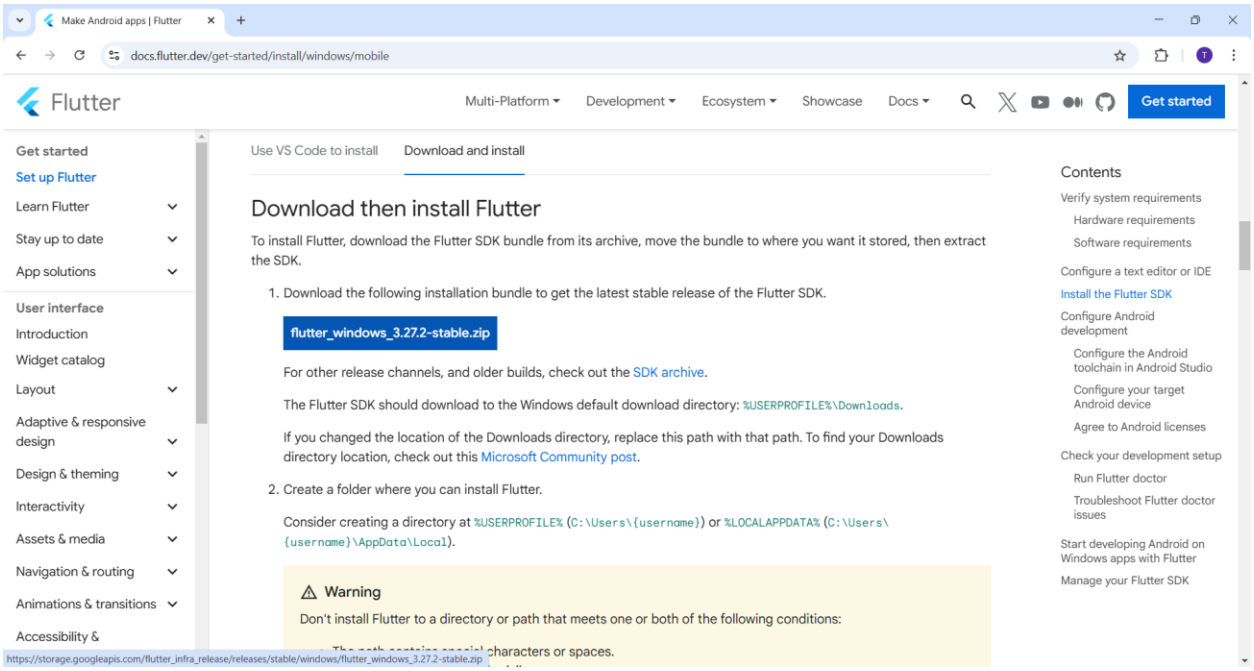
Step 1: Go to the official Flutter website: https://docs.flutter.dev/get-started/install



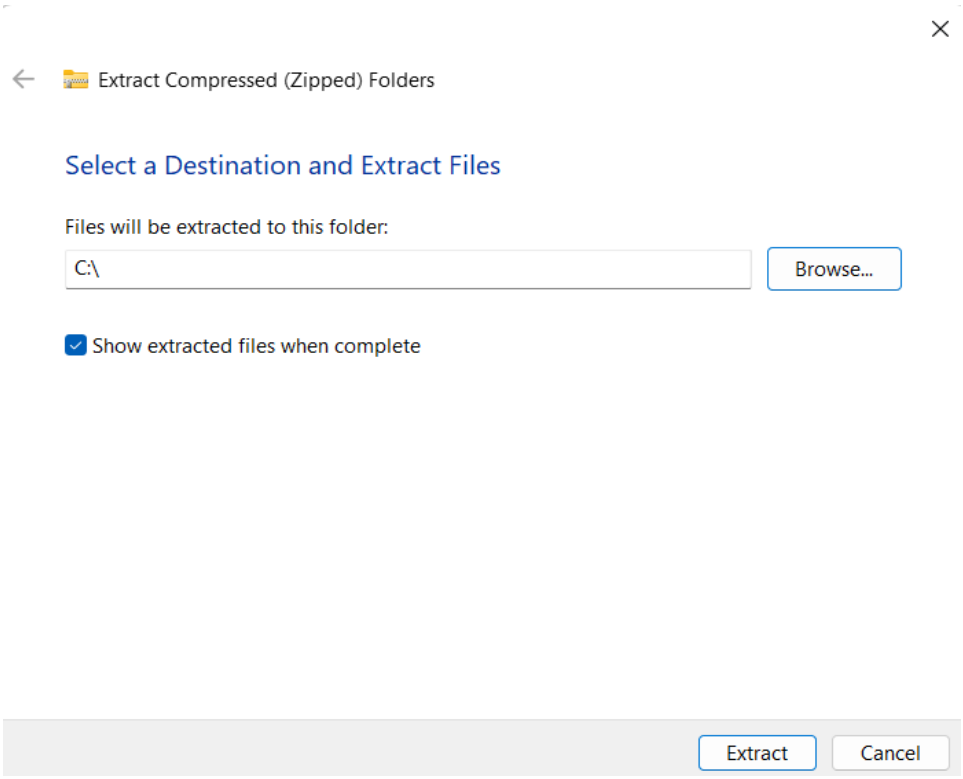
Step 2: To download the latest Flutter SDK, click on the Windows icon > Android



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



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

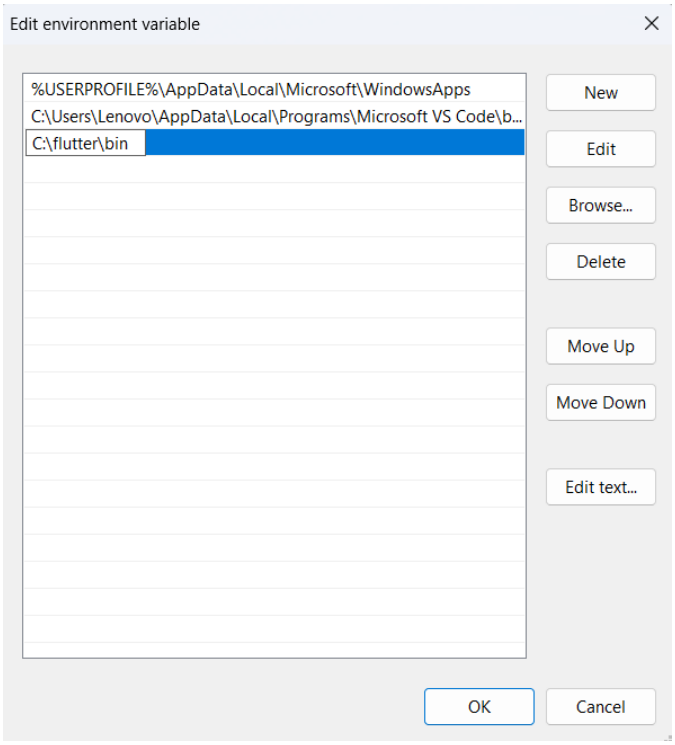


Step 5 :- Add Flutter to System PATH

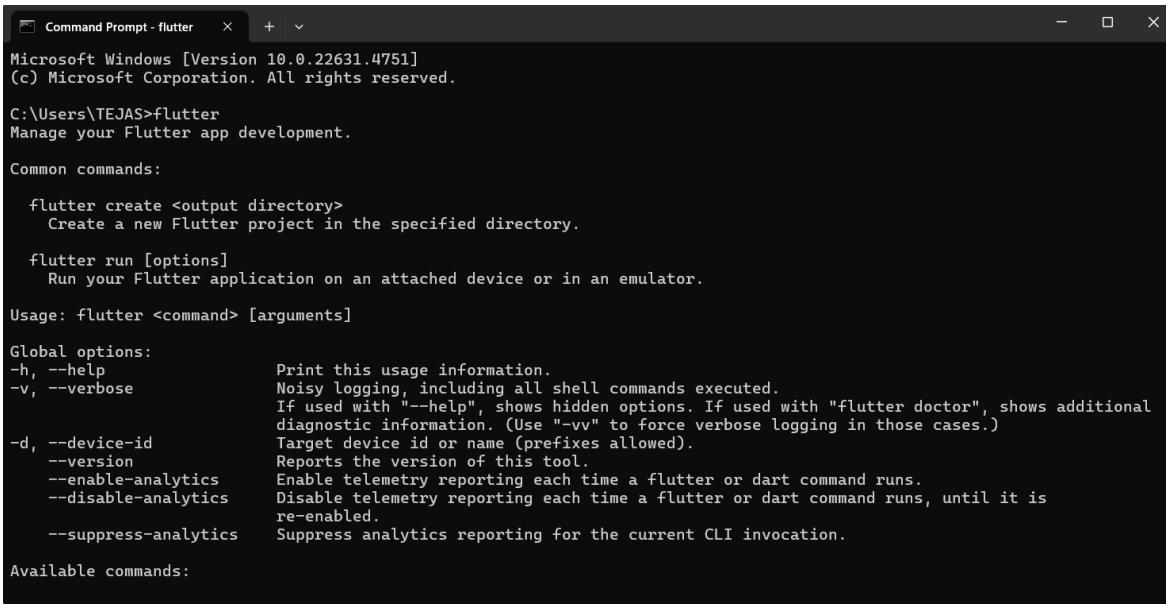
Right-click on the Start Menu > System > Advanced system settings > Environment Variables.

Under System Variables, find Path and click Edit.

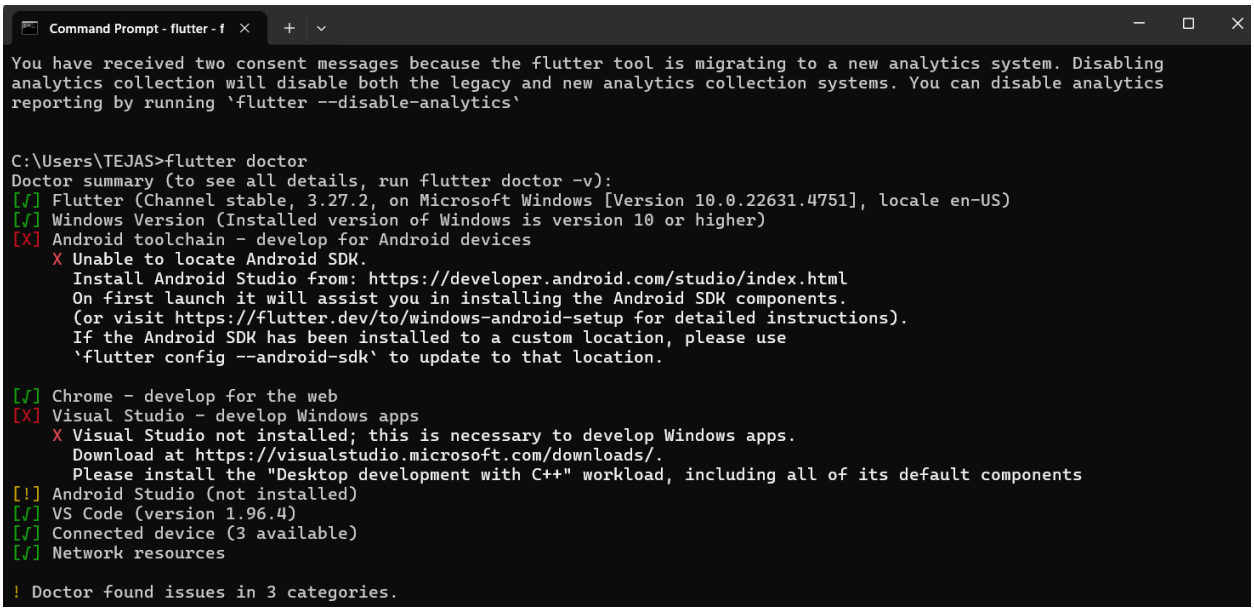
Add the full path to the flutter/bin directory (e.g., C:\flutter\bin).



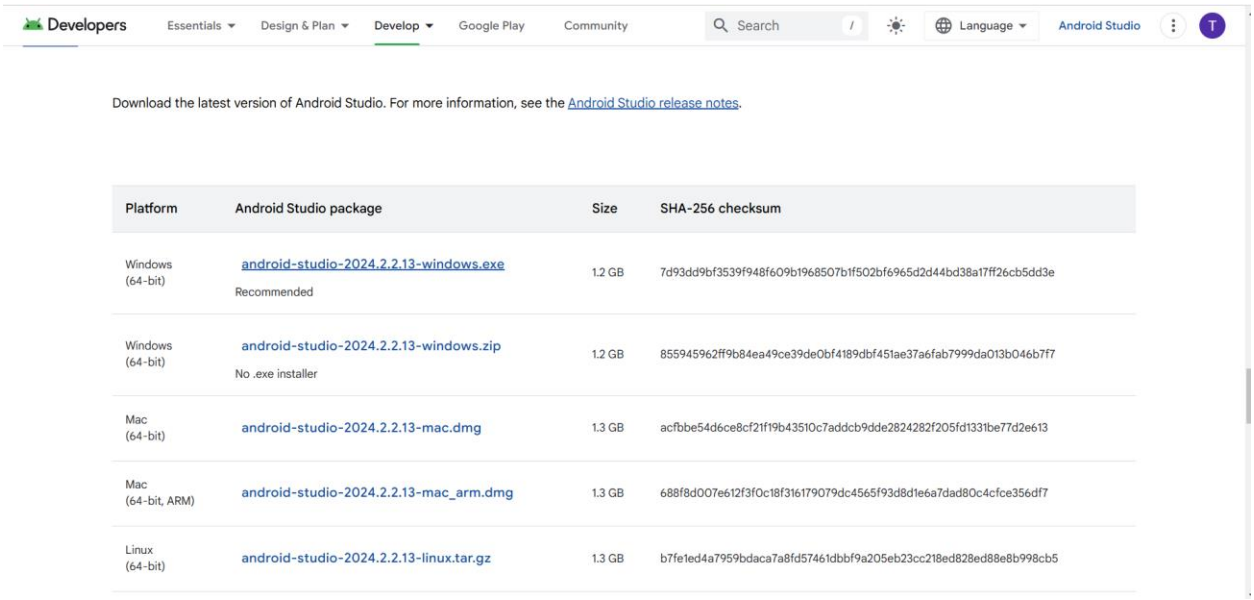
Step 6 : - Now, run the $ flutter command in command prompt.



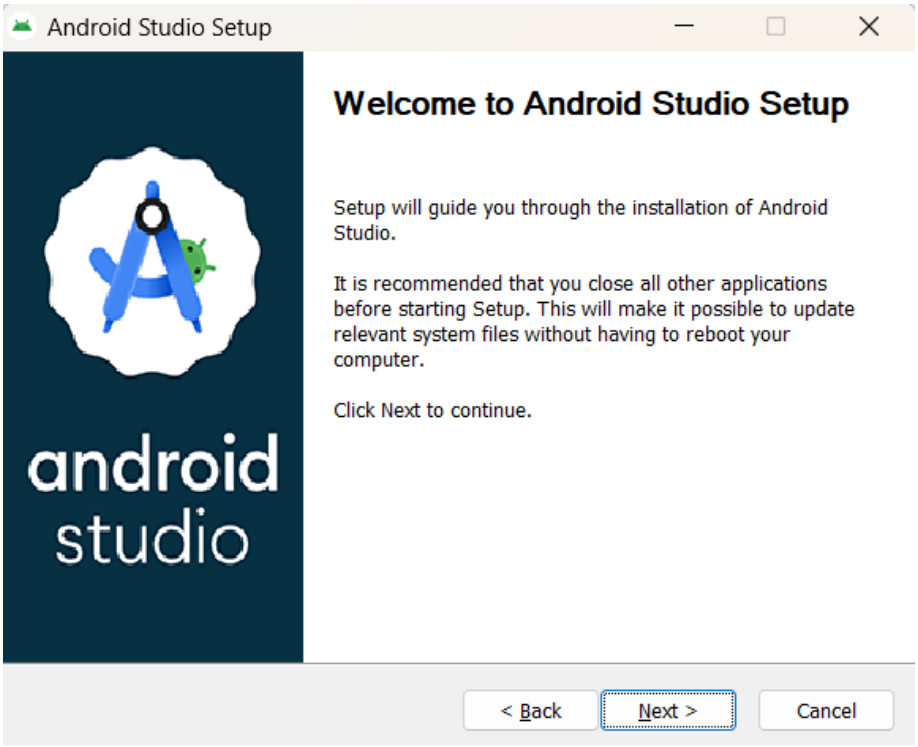
Step 7:- Run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation



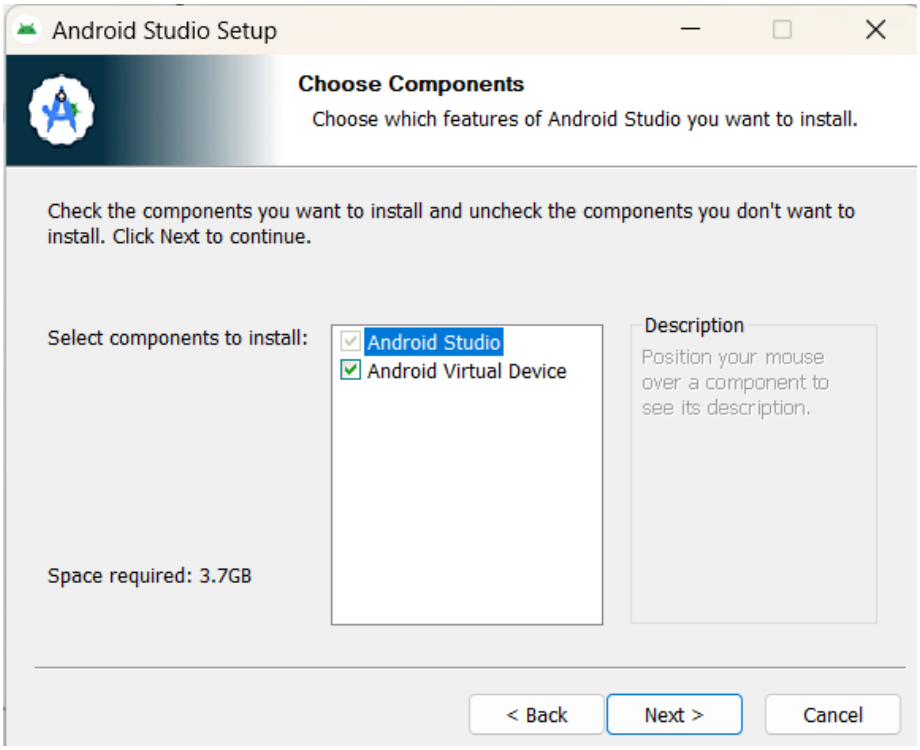
Step 8 : - Go to Android Studio and download the installer.



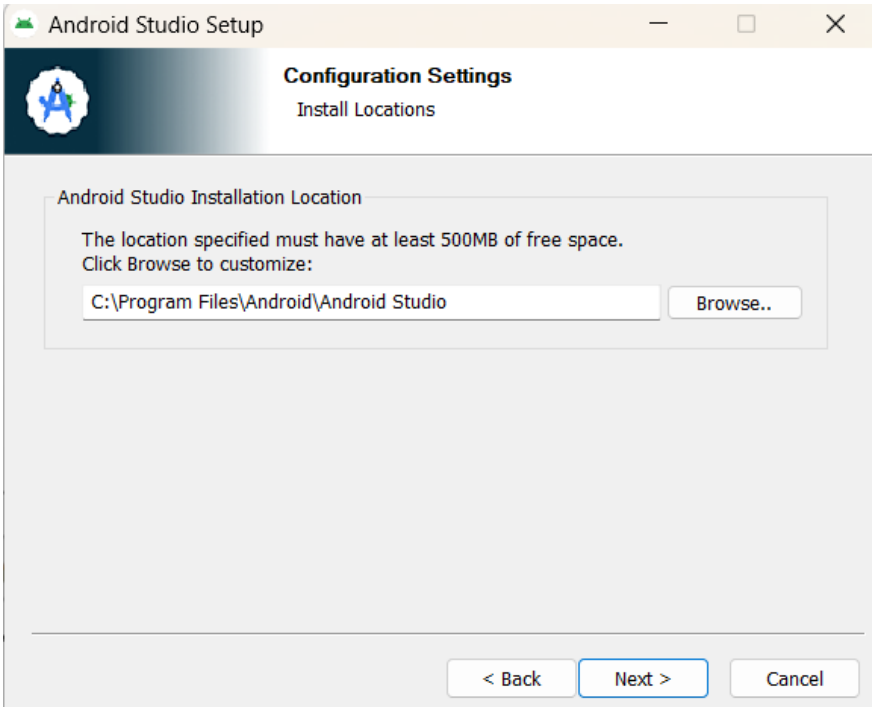
Step 8.1: - When the download is complete, open the .exe file and run it. You will get the following dialog box



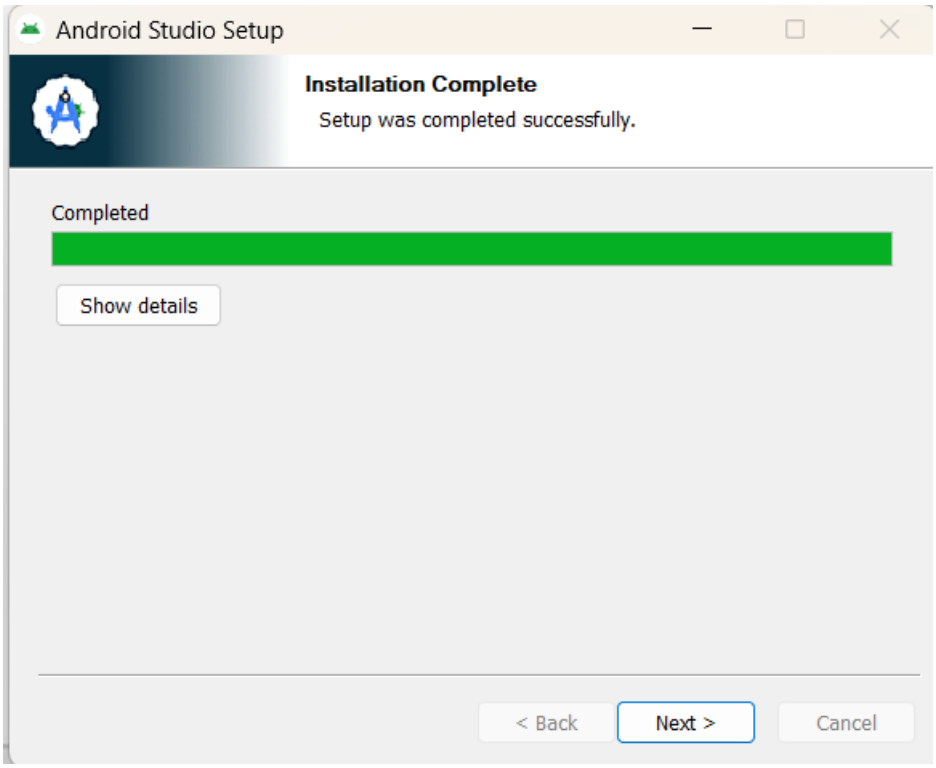
Step 8.2: - Select all the Checkboxes and Click on 'Next' Button.

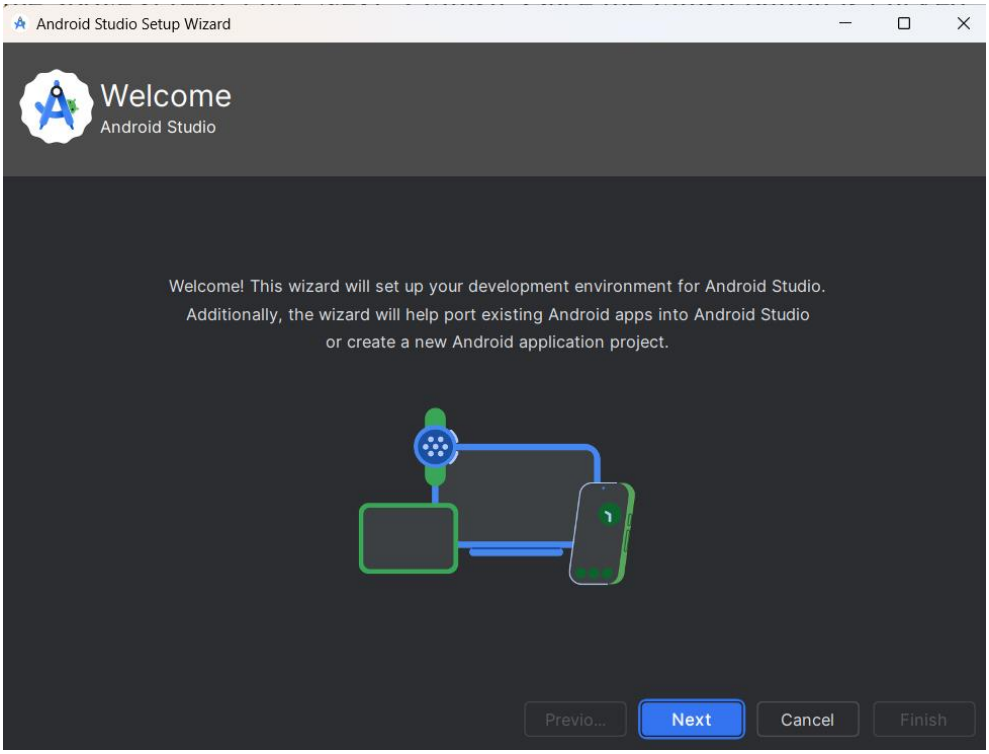


Step 8.3: - Change the destination as per your convenience and click on ‘Next’ Button.

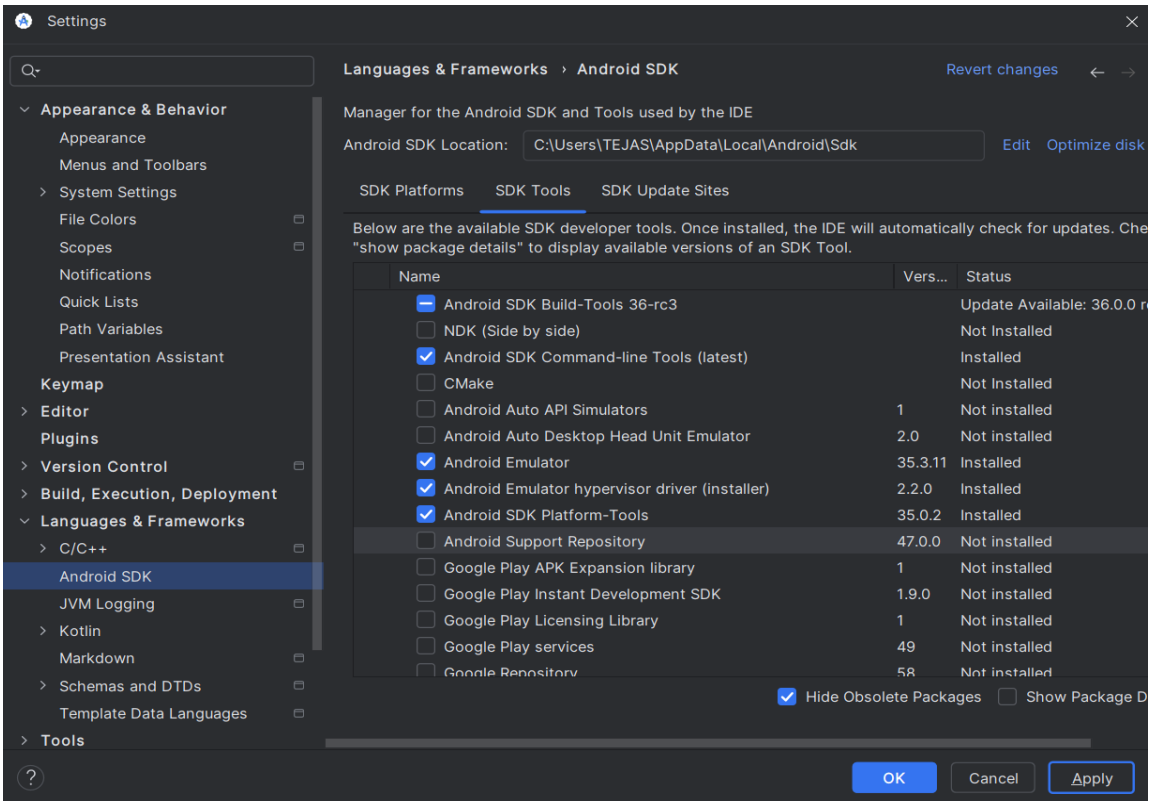


Step 8.4: - Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.

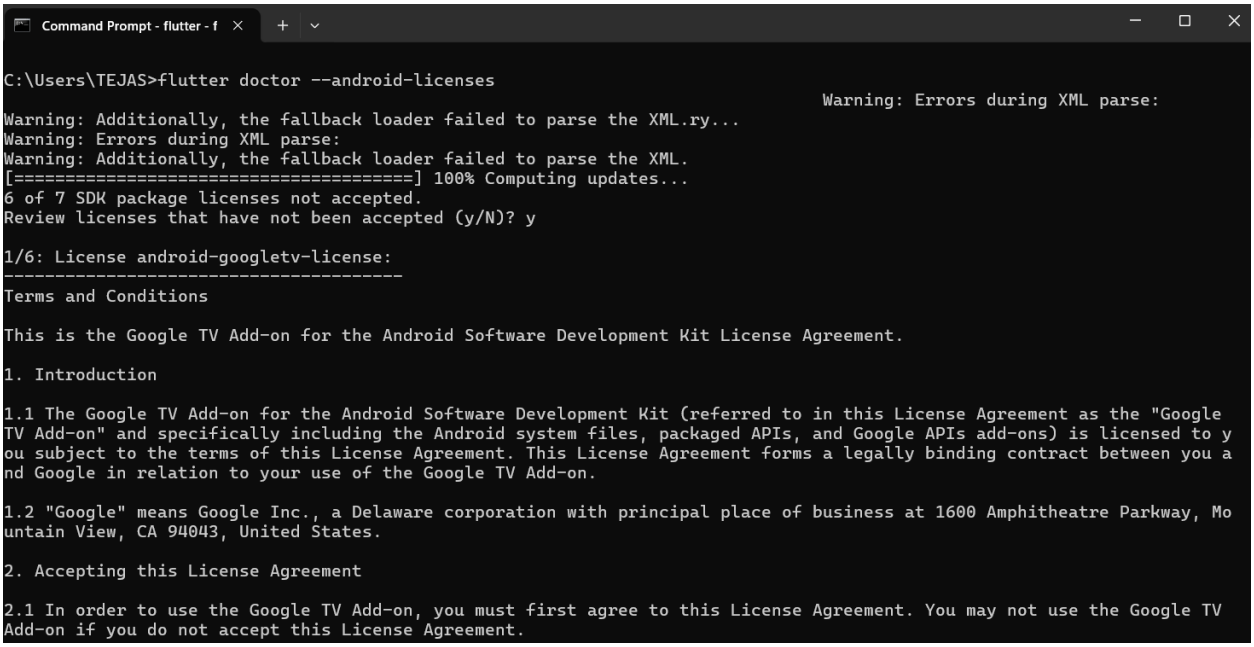


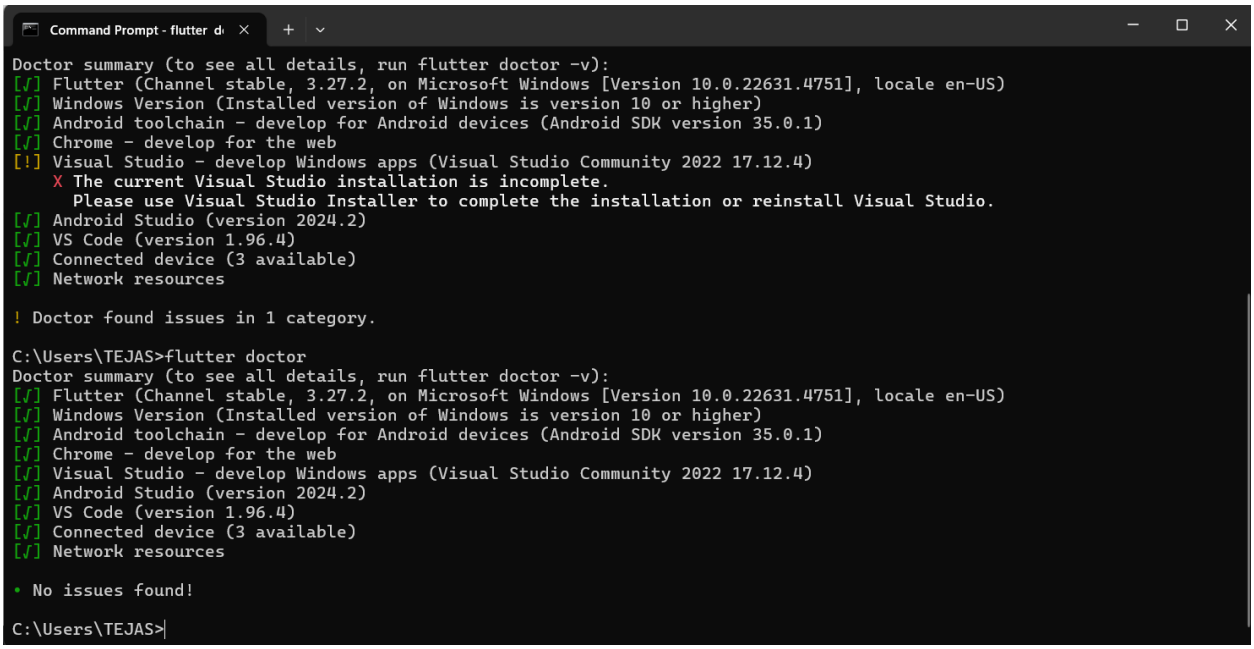


Step 8.5: - Go to Preferences > Appearance & Behavior > System Settings > Android SDK. Select the SDK Tools tab and check Android SDK Command-line Tools and Install it.

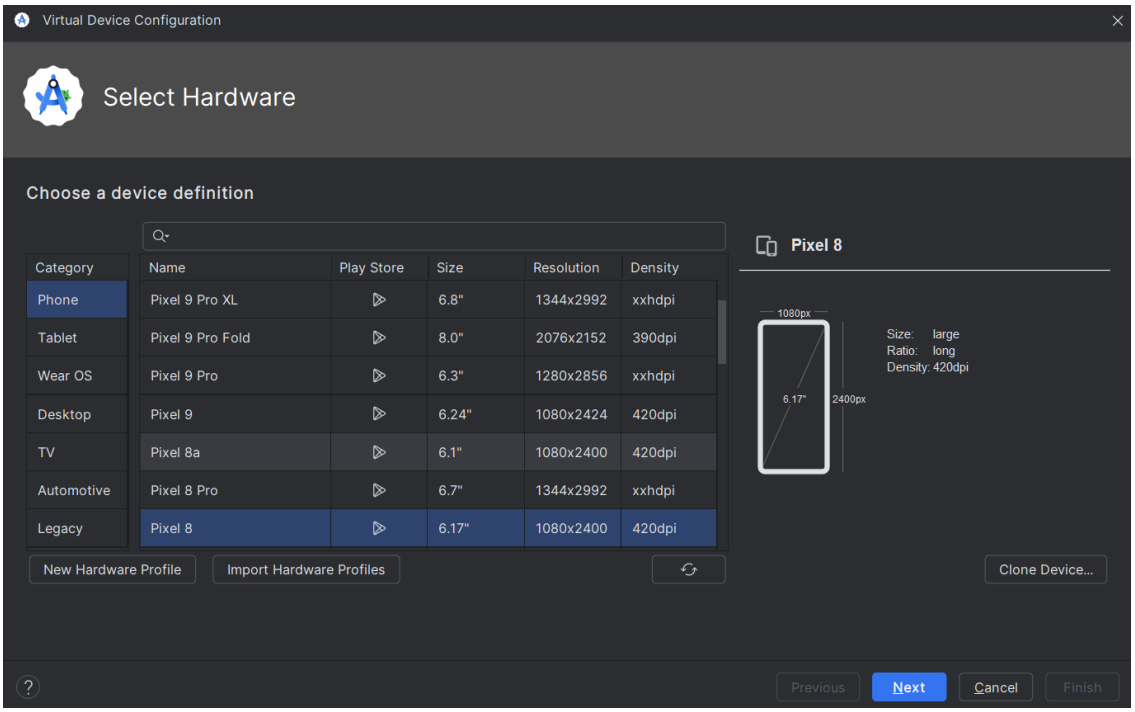


Step 9: - Open a terminal and run the following command

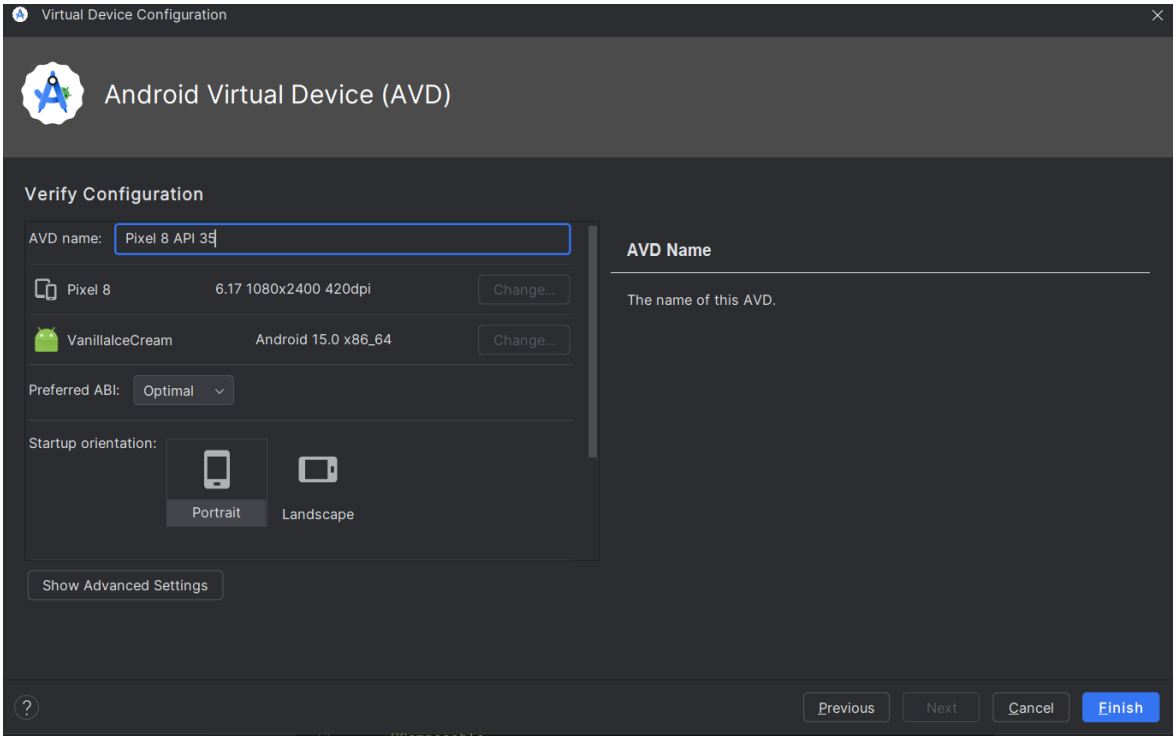




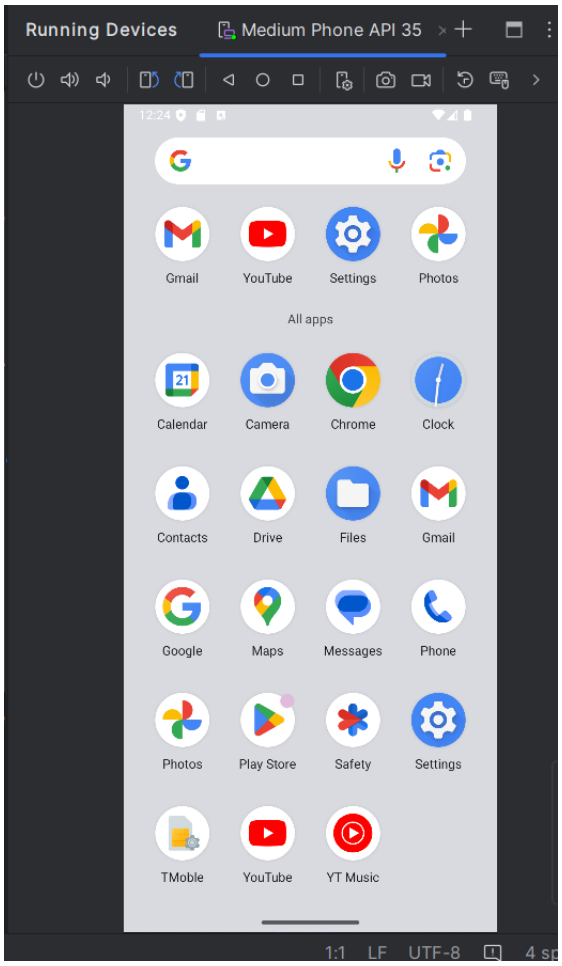
Step 10: - Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application



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

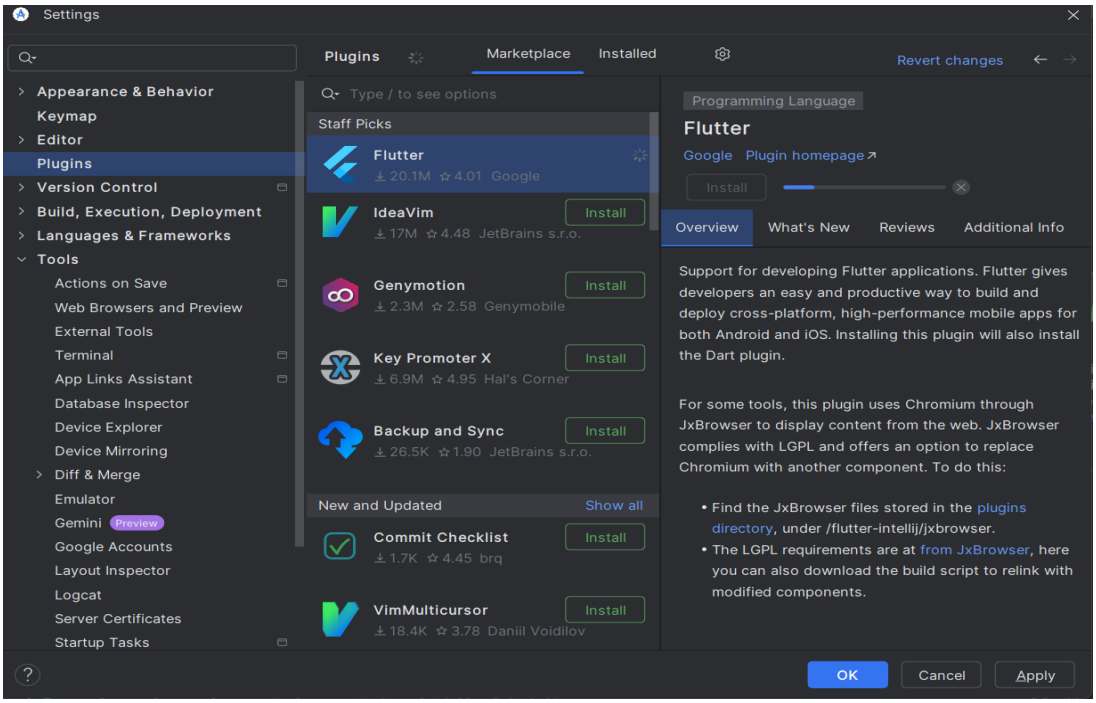


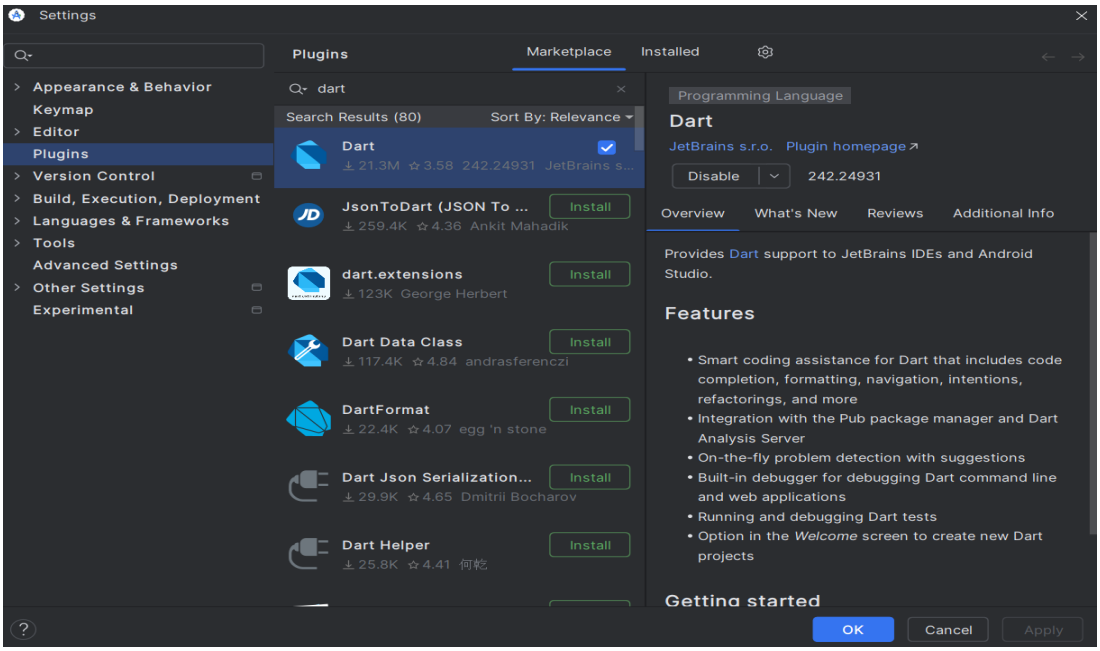
Step 10.2: - Click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen



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

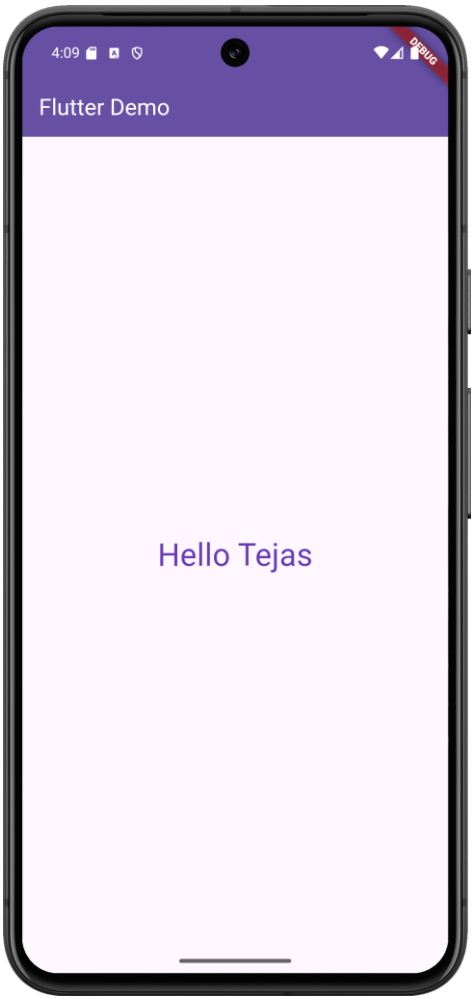
Step 11.1: - Open the Android Studio and then go to File->Settings->Plugins. Now, search the Flutter plugin. If found, select Flutter plugin and click install





Step 11.2: - Restart the Android Studio

Step 12: - Go to File > New Project > Create Flutter Project, then select the project name and location, and click Next to proceed.



MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	18
Name	Tejas Dhondibhau Gunjal
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: - 02

Name:- Tejas Gunjal **Class:-** D15A **Roll:No: -** 18

AIM: - To design Flutter UI by including common widgets.

Theory: -

Each element on the screen of the Flutter app is a widget. The view of the screen completely depends upon the choice and sequence of the widgets used to build the apps. And the structure of the code of apps is a tree of widgets.

When you made any alteration in the code, the widget rebuilds its description by calculating the difference of previous and current widget to determine the minimal changes for rendering in UI of the app.Widgets are nested with each other to build the app. It means the root of your app is itself a widget, and all the way down is a widget also. For example, a widget can display something, can define design, can handle interaction, etc.

The single child layout widget is a type of widget, which can have only **one child widget** inside the parent layout widget. These widgets can also contain special layout functionality. Flutter provides us many single child widgets to make the app UI attractive. If we use these widgets appropriately, it can save our time and makes the app code more readable.

The multiple child widgets are a type of widget, which contains **more than one child widget**, and the layout of these widgets are **unique**. For example, Row widget laying out of its child widget in a horizontal direction, and Column widget laying out of its child widget in a vertical direction. If we combine the Row and Column widget, then it can build any level of the complex widget.

Type of Widgetss

- **StatefulWidget**

  A StatefulWidget has state information. It contains mainly two classes: the state object and the widget. It is dynamic because it can change the inner data during the widget lifetime. This widget does not have a build() method. It has createState() method, which returns a class that extends the Flutters State Class. The examples of the StatefulWidget are Checkbox, Radio, Slider, InkWell, Form, and TextField.

- **StatelessWidget**

  The StatelessWidget does not have any state information. It remains static throughout its lifecycle. The examples of the StatelessWidget are Text, Row, Column, Container, etc.

Some of the commonly used widgets

Container – A box widget used for styling with padding, margins, colors, borders, and constraints. It helps in layout structuring and positioning.

Row & Column – Used to arrange widgets in horizontal (Row) or vertical (Column) orientation. They manage spacing, alignment, and distribution of child widgets.

Stack – Overlaps widgets on top of each other, useful for creating layered UIs like banners, tooltips, or floating elements.

Text – Displays text on the screen with customizable font size, color, alignment, and styling options

Image – Loads and displays images from assets, network, or memory with scaling, fit, properties.

Scaffold – Provides a basic layout structure with an app bar, body, floating action button, and bottom navigation.

ListView – A scrollable list widget that efficiently renders large amounts of dynamic content. Supports both vertical and horizontal scrolling.

GridView – Displays widgets in a grid format, useful for galleries, product listings, or dashboards. It supports dynamic column adjustments.

SizedBox – Used to create space between widgets or define fixed width and height for layout adjustments.

ElevatedButton – A button with elevation that provides a raised effect, customizable with color, shape, and click actions.

TextField – A user input field that supports text entry, keyboard configurations, validation.

AppBar – A top navigation bar that includes a title, actions, and menu icons, commonly used in Scaffold.

BottomNavigationBar – A bar at the bottom of the screen used for navigation between different app sections with icons and labels.

Drawer – A side navigation panel that slides out from the left, typically used for app menus and quick navigation.

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

Code: -

home_page.dart

```
import 'package:flutter/material.dart';
import 'myaccountpage.dart';
import 'news_section.dart';
import 'crop_list_page.dart';
import 'weather_forecast_page.dart';

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

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

class HomePage extends StatefulWidget {
  @override
  _HomePageState createState() =>
_HomePageState();
}

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

  // Method to handle the bottom navigation item
taps
  void _onItemTapped(int index) {
    setState(() {
      _currentIndex = index;
    });
    if (index == 1) {
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) =>
WeatherPage()),
      );
    }
    if (index == 3) {
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) =>
CropListPage()),
      );
    }
    if (index == 4) {
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) =>
MyAccountPage()),
      );
    }
  }



  // Returns the widget corresponding to the
selected index
  Widget _getSelectedPage() {
    switch (_currentIndex) {
      case 0:
        return Center(child: Text('Home Page'));
      case 1:
        return Center(child: Text('Weather Page'));
      case 2:
        return Center(child: Text('Disease
Detection Page'));
      case 3:
        return Center(child: Text('Crop'));
      case 4:
        return Center(child: Text('Profile Page'));
      default:
        return Center(child: Text('Home Page'));
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        backgroundColor: Colors.white,
        title: Row(
          children: [
```

```
          Image.asset('assets/images/logo.png',
height: 40), // Your logo here
          SizedBox(width: 10),
          Text('AgriApp'),
         ],
        ),
        actions: [
         IconButton(
          icon: Icon(Icons.search),
          onPressed: () {
           // Add search functionality
          },
         ),
        ],
       ),
       body: SingleChildScrollView(
        child: Column(
         children: [
          BannerSection(),
          CategorySection(),
          DealsOfTheDaySection(),
          FreeDeliverySection(),
          NewsSection(),
         ],
        ),
       ),
       bottomNavigationBar:
BottomNavigationBar(
        currentIndex: _currentIndex, // Keep track
of the selected index
        onTap: _onItemTapped, // Update the index
when an item is tapped
        selectedItemColor: Colors.black, // Color
for the selected item
        unselectedItemColor: Colors.black54, //
Color for unselected items
        type: BottomNavigationBarType.fixed, //
Ensures icons stay the same size
        items: const <BottomNavigationBarItem>[
         BottomNavigationBarItem(
          icon: Icon(Icons.home),
          label: 'Home',
         ),
         BottomNavigationBarItem(
          icon: Icon(Icons.cloud), // Weather icon
          label: 'Weather',
         ),
         BottomNavigationBarItem(
          icon: Icon(Icons.camera_alt), // Disease
detection icon
          label: 'Detect',
         ),
         BottomNavigationBarItem(
          icon: Icon(Icons.local_florist), // Crop
practices icon
          label: 'Crops',
         ),
         BottomNavigationBarItem(
          icon: Icon(Icons.account_circle), //
Profile icon
          label: 'Profile',
         ),
        ],
       ),
      );
     }
    }

    class BannerSection extends StatelessWidget {
     @override
     Widget build(BuildContext context) {
      return Padding(
       padding: EdgeInsets.all(14.0),
       child: ClipRRect(
        borderRadius: BorderRadius.circular(20),
        child: Container(
         width: double.infinity,
         height: 200,
         decoration: BoxDecoration(
          image: DecorationImage(
           image:
AssetImage('assets/images/banner.jpg'),
           //fit: BoxFit.cover,
          ),
         ),
        ),
       ),
      );
     }
    }

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

```
    return Container(
      padding: EdgeInsets.symmetric(vertical: 20),
      child: GridView.count(
        crossAxisCount: 4,
        shrinkWrap: true,
        physics: NeverScrollableScrollPhysics(),
        children: [
          CategoryItem('Offers',
'assets/images/offers.png'),
          CategoryItem('Crop Tonics',
'assets/images/crop_tonics.png'),
          CategoryItem('Fertilizer',
'assets/images/fertilizer.png'),
          CategoryItem('Pesticides',
'assets/images/pesticides.png'),
        ],
      ),
    );
  }
}

class CategoryItem extends StatelessWidget {
  final String title;
  final String imagePath; // Changed from
IconData to String

  CategoryItem(this.title, this.imagePath);

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Container(
          width:80,
          height:80,
          decoration: BoxDecoration(
            color: Colors.green.withAlpha(50), //
Light green background
            shape: BoxShape.circle,
          ),
          padding: EdgeInsets.all(16),
          child: Image.asset(imagePath, width:
double.infinity, height: double.infinity,
fit:BoxFit.cover), // Load image instead of icon
        ),
        SizedBox(height: 5),
        Text(title, textAlign: TextAlign.center,
style: TextStyle(fontSize: 12, fontWeight:
FontWeight.bold)),
      ],
    );
  }
}

class DealsOfTheDaySection extends
StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.symmetric(vertical: 10),
      child: Column(
        crossAxisAlignment:
CrossAxisAlignment.start,
        children: [
          // Row for title and "See All" button
          Padding(
            padding: EdgeInsets.only(left: 20),
            child: Row(
              mainAxisAlignment:
MainAxisAlignment.spaceBetween,
              children: [
                Text(
                  'Deals of the day!',
                  style: TextStyle(fontSize: 24,
fontWeight: FontWeight.bold),
                ),
                TextButton(
                  onPressed: () {
                    // Navigate to All Deals screen
                    Navigator.push(
                      context,
                      MaterialPageRoute(builder:
(context) => AllDealsScreen()),
                    );
                  },
                  child: Text(
                    'See All >',
                    style: TextStyle(color: Colors.blue,
fontWeight: FontWeight.bold),
                  ),
                ),
              ],
            ),
          ),
```

```
        SizedBox(height: 10),
        // Grid of products for Deals of the Day
        GridView.count(
          crossAxisCount: 2,
          shrinkWrap: true,
          physics: NeverScrollableScrollPhysics(),
          children: [
            ProductCard('00-00-50', '₹240.00',
'₹270.00', 'assets/images/product1.png'),
            ProductCard('12-61-00', '₹277.2',
'₹355.00', 'assets/images/product2.png'),
          ],
        ),
      ],
    ),
  );
 }
}

// Product Card widget
class ProductCard extends StatelessWidget {
 final String productName;
 final String price;
 final String originalPrice;
 final String imagePath;

 ProductCard(this.productName, this.price,
this.originalPrice, this.imagePath);

 @override
 Widget build(BuildContext context) {
  return Card(
    margin: EdgeInsets.all(8),
    //color: Colors.white, // White background
for card
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(8),
      side: BorderSide(color: Colors.grey, width:
1), // Black border
    ),
    child: Column(
      children: [
        Container(
          height: 100,
          width: 100,
          child: Image.asset(
            imagePath,
            fit: BoxFit.contain,
          ),
        ),
        Text(productName),
        Text(price, style: TextStyle(fontWeight:
FontWeight.bold)),
        Text(originalPrice, style:
TextStyle(decoration:
TextDecoration.lineThrough)),
      ],
    ),
  );
 }
}

class AllDealsScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('All Deals'),
    ),
    body: GridView.count(
      crossAxisCount: 2,
      padding: EdgeInsets.all(10),
      children: [
        // List of all product cards
        ProductCard('00-00-50', '₹240.00',
'₹270.00', 'assets/images/product1.png'),
        ProductCard('12-61-00', '₹277.2',
'₹355.00', 'assets/images/product2.png'),
        ProductCard('23-45-67', '₹500.00',
'₹600.00', 'assets/images/product3.png'),
        ProductCard('34-56-78', '₹350.00',
'₹400.00', 'assets/images/product4.png'),
        ProductCard('45-67-89', '₹220.00',
'₹280.00', 'assets/images/product5.png'),
        ProductCard('56-78-90', '₹310.00',
'₹370.00', 'assets/images/product6.png'),
        ProductCard('67-89-01', '₹450.00',
'₹520.00', 'assets/images/product7.png'),
        ProductCard('78-90-12', '₹200.00',
'₹250.00', 'assets/images/product8.png'),
        ProductCard('89-01-23', '₹600.00',
'₹700.00', 'assets/images/product9.png'),
        ProductCard('90-12-34', '₹150.00',
'₹200.00', 'assets/images/product10.png'),
```

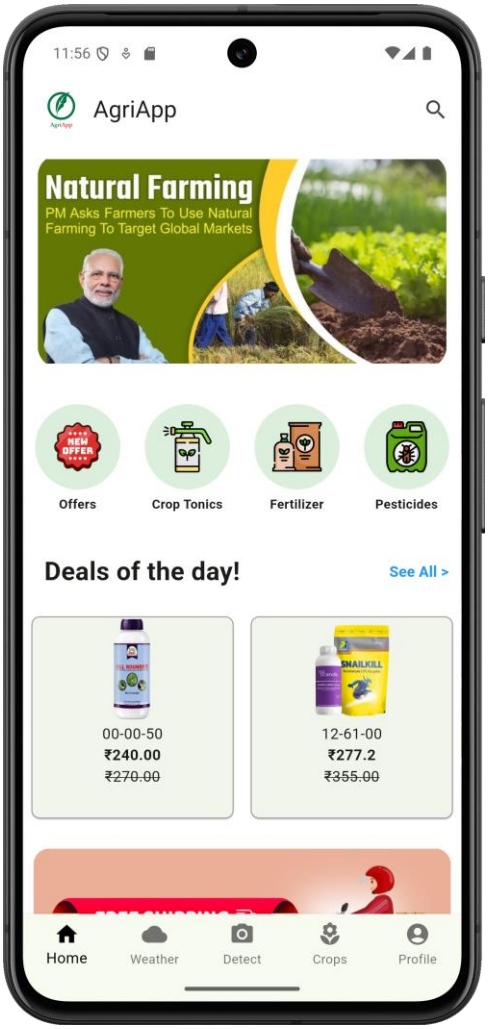
```
        ProductCard('01-23-45', '₹180.00',
'₹230.00', 'assets/images/product11.png'),
        ProductCard('12-34-56', '₹420.00',
'₹480.00', 'assets/images/product12.png'),
        // Add more ProductCard widgets here
       ],
     ),
   );
  }
}


class FreeDeliverySection extends
StatelessWidget {
  @override
  Widget build(BuildContext context) {
   return Container(
     padding: EdgeInsets.symmetric(vertical: 10),
// Vertical padding
     child: Center(
       child: Padding(
         padding:
EdgeInsets.symmetric(horizontal: 10), // Add
horizontal space (left & right)
         child: ClipRRect(
           borderRadius:
BorderRadius.circular(15), // Adjust the radius
as needed
           child: Image.asset(
             'assets/images/delivery.png',
             fit: BoxFit.cover,
           ),
         ),
       ),
     ),
   );
  }
}
```

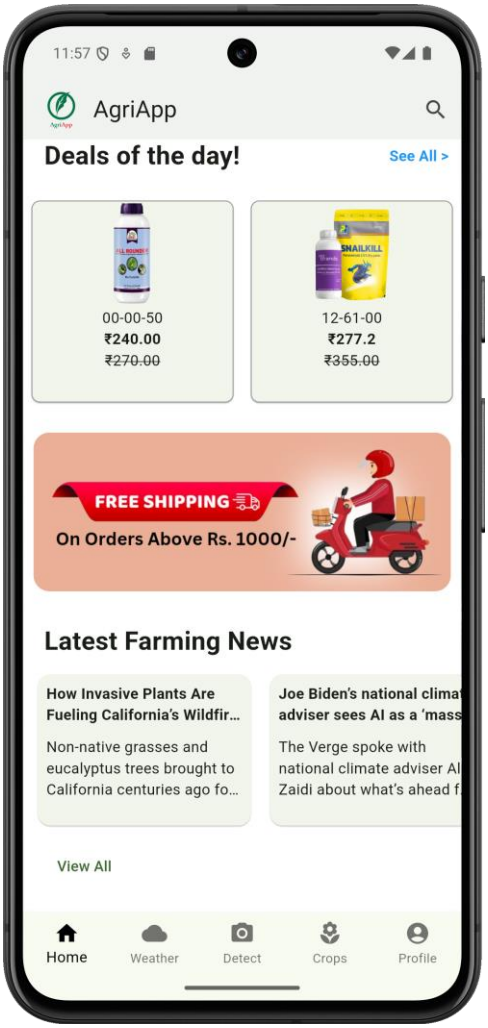
news_section.dart

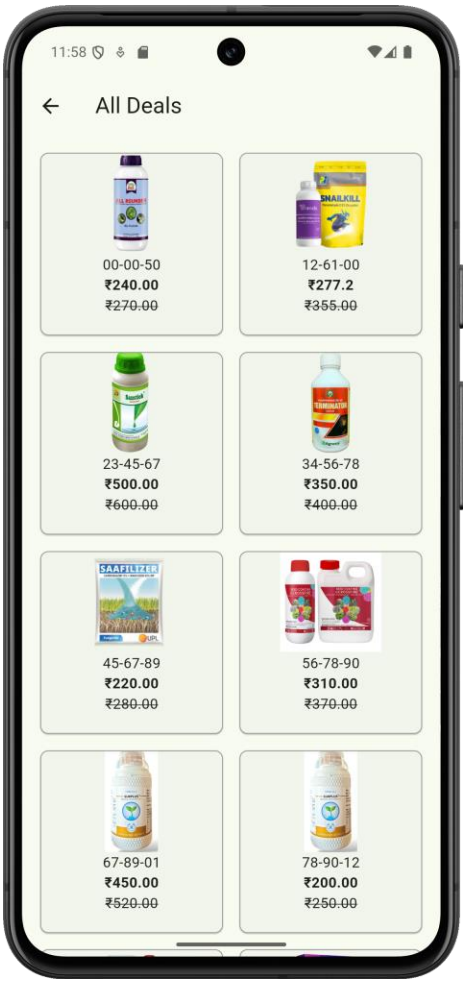
```
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';

// Define the API endpoint
const String apiKey =
"7b40c2993a264008904439ac295309b0";
const String newsApiUrl =

"https://newsapi.org/v2/everything?q=Agricultur
e&sortBy=popularity&apiKey=7b40c2993a264
008904439ac295309b0";

// News Section widget
class NewsSection extends StatefulWidget {
  @override
  _NewsSectionState createState() =>
_NewsSectionState();
}

class _NewsSectionState extends
State<NewsSection> {
  List<dynamic> farmingNews = [];

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

  // Fetch the news from the API
  Future<void> fetchNews() async {
    final response = await
http.get(Uri.parse(newsApiUrl));

    if (response.statusCode == 200) {
      setState(() {
        farmingNews =
json.decode(response.body)['articles']; // Adjust
based on the response structure
      });
    } else {
      // Handle the error if the request fails
      throw Exception('Failed to load news');
    }
  }

  @override
  Widget build(BuildContext context) {
    return SingleChildScrollView( // Wrap the
column in a scroll view
      child: Padding(
        padding: EdgeInsets.symmetric(vertical:
20),
        child: Column(
          crossAxisAlignment:
CrossAxisAlignment.start,
          children: [
            Padding(
              padding: EdgeInsets.only(left: 20),
              child: Text(
                'Latest Farming News',
                style: TextStyle(fontSize: 24,
fontWeight: FontWeight.bold),
              ),
            ),
            SizedBox(height: 10),
            // Container with horizontal scrolling for
news
            Container(
              height: 150, // Set height for horizontal
scroll
              child: ListView.builder(
                scrollDirection: Axis.horizontal,
                itemCount: farmingNews.length > 4 ?
4 : farmingNews.length, // Limit to 4 items
                itemBuilder: (context, index) {
                  return Padding(
                    padding: EdgeInsets.only(left: 10),
                    child: Card(
                      child: Container(
                        width: 200,
                        padding: EdgeInsets.all(8),
                        child: Column(
                          crossAxisAlignment:
CrossAxisAlignment.start,
                          children: [
                            Text(
                              farmingNews[index]['title'],
                              style: TextStyle(fontWeight:
FontWeight.bold),
                              maxLines: 2,
```

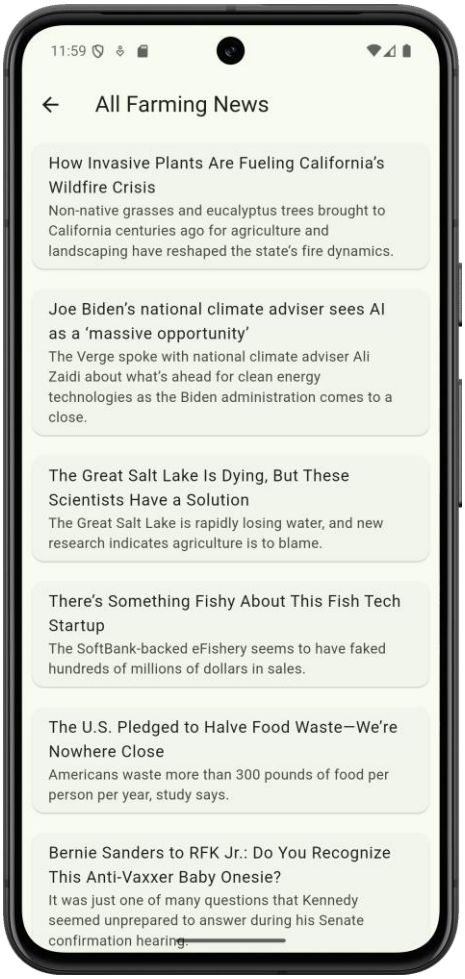
```
                        overflow:
TextOverflow.ellipsis,
                      ),
                      SizedBox(height: 10),
                      Text(

farmingNews[index]['description'] ?? 'No
description available.',
                        maxLines: 3,
                        overflow:
TextOverflow.ellipsis,
                      ),
                    ],
                  ),
                ),
              ),
            );
          },
        ),
      ),
      // View All Button
      Padding(
        padding: EdgeInsets.only(left: 20, top:
10),
        child: TextButton(
          onPressed: () {
            // Navigate to the next screen to show
all the news
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) =>
AllNewsScreen(news: farmingNews),
              ),
            );
          },
          child: Text('View All'),
        ),
      ),
    ],
  ),
 ),
);
 }
}

class AllNewsScreen extends StatelessWidget {
 final List<dynamic> news;

 AllNewsScreen({required this.news});

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(title: Text('All Farming
News')),
   body: ListView.builder(
    itemCount: news.length,
    itemBuilder: (context, index) {
     return Card(
      margin: EdgeInsets.all(10),
      child: ListTile(
       title: Text(news[index]['title']),
       subtitle: Text(news[index]['description']
?? 'No description available.'),
      ),
     );
    },
   ),
  );
 }
}
```

OUTPUT: -


11:56
AgriApp
Natural Farming
PM Asks Farmers To Use Natural Farming To Target Global Markets
Offers
Crop Tonics
Fertilizer
Pesticides
Deals of the day!
See All >
00-00-50
₹240.00
₹270.00
12-61-00
₹277.2
₹355.00
Home
Weather
Detect
Crops
Profile


11:57
AgriApp
Deals of the day!
See All >
00-00-50
₹240.00
₹270.00
12-61-00
₹277.2
₹355.00
FREE SHIPPING
On Orders Above Rs. 1000/-
Latest Farming News
How Invasive Plants Are Fueling California's Wildfir...
Non-native grasses and eucalyptus trees brought to California centuries ago fo...
Joe Biden's national clima
adviser sees AI as a 'mass
The Verge spoke with national climate adviser Al Zaidi about what's ahead f
View All
Home
Weather
Detect
Crops
Profile


11:58
All Deals
00-00-50
₹240.00
₹270.00
12-61-00
₹277.2
₹355.00
23-45-67
₹500.00
₹600.00
34-56-78
₹350.00
₹400.00
45-67-89
₹220.00
₹280.00
56-78-90
₹310.00
₹370.00
67-89-01
₹450.00
₹520.00
78-90-12
₹200.00
₹250.00


11:59
All Farming News
How Invasive Plants Are Fueling California's Wildfire Crisis
Non-native grasses and eucalyptus trees brought to California centuries ago for agriculture and landscaping have reshaped the state's fire dynamics.
Joe Biden's national climate adviser sees AI as a 'massive opportunity'
The Verge spoke with national climate adviser Ali Zaidi about what's ahead for clean energy technologies as the Biden administration comes to a close.
The Great Salt Lake Is Dying, But These Scientists Have a Solution
The Great Salt Lake is rapidly losing water, and new research indicates agriculture is to blame.
There's Something Fishy About This Fish Tech Startup
The SoftBank-backed eFishery seems to have faked hundreds of millions of dollars in sales.
The U.S. Pledged to Halve Food Waste—We're Nowhere Close
Americans waste more than 300 pounds of food per person per year, study says.
Bernie Sanders to RFK Jr.: Do You Recognize This Anti-Vaxxer Baby Onesie?
It was just one of many questions that Kennedy seemed unprepared to answer during his Senate confirmation hearing.

MAD & PWA Lab

Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	18
Name	Tejas Dhondibhau Gunjal
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: - 03

Name:- Tejas Gunjal **Class:-** D15A **Roll:No: -** 18

AIM: - To include icons, images, fonts in Flutter app.

Theory: -

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

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

Visual elements play a significant role in app development.

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

➢ **Adding Icons in Flutter**

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

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

➢ Adding Images in Flutter

Flutter supports images from three sources:

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

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

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

   - Display the image in the app

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

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

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

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

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

➢ Adding Custom Fonts in Flutter

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

- Download the font and place it in the assets/fonts/ folder.
- Declare the font in pubspec.yaml
- Use the font in the app

  ```
  Text(
    'Custom Font Example',
    style: TextStyle(fontFamily: 'CustomFont', fontSize: 24),
  );
  ```

Code: -

crop_list_page.dart

```
import 'dart:convert';
import 'package:flutter/services.dart';
import 'package:flutter/material.dart';
import 'crop_detail_page.dart';

class CropListPage extends StatelessWidget {
  Future<List<Map<String, dynamic>>>
loadCrops() async {
    // Load the JSON file
    String jsonString = await
rootBundle.loadString('assets/crops.json');
    List<dynamic> jsonResponse =
json.decode(jsonString);
    return jsonResponse.map((crop) => crop as
Map<String, dynamic>).toList();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Crop
Practices')),
      body: FutureBuilder<List<Map<String,
dynamic>>>(
        future: loadCrops(),  // Load crops data
from JSON
        builder: (context, snapshot) {
          if (snapshot.connectionState ==
ConnectionState.waiting) {
            return Center(child:
CircularProgressIndicator());
          } else if (snapshot.hasError) {
            return Center(child: Text('Error:
${snapshot.error}'));
          } else if (!snapshot.hasData ||
snapshot.data!.isEmpty) {
            return Center(child: Text('No crops
available.'));
          } else {
            var crops = snapshot.data!;
            return GridView.builder(
              padding: EdgeInsets.all(10),
              gridDelegate:
SliverGridDelegateWithFixedCrossAxisCount(
                crossAxisCount: 2,
                crossAxisSpacing: 10,
                mainAxisSpacing: 15,
                childAspectRatio: 1.2,
              ),
              itemCount: crops.length,
              itemBuilder: (context, index) {
                return GestureDetector(
                  onTap: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) =>
CropDetailPage(crop: crops[index]),
                      ),
                    );
                  },
                  child: Column(
                    crossAxisAlignment:
CrossAxisAlignment.center,
                    children: [
                      ClipRRect(
                        borderRadius:
BorderRadius.circular(15),
                        child: Image.asset(
                          crops[index]["image"],
                          height: 130,
                          width: 180,
                          fit: BoxFit.cover,
                        ),
                      ),
                      SizedBox(height: 5),
                      Text(
                        crops[index]["name"],
                        style: TextStyle(fontSize: 16,
fontWeight: FontWeight.bold),
                      ),
                    ],
```

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

crop_details_page.dart

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

class CropDetailPage extends StatelessWidget {
 final Map<String, dynamic> crop;

 CropDetailPage({required this.crop});

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(title: Text(crop["name"])),
   body: ListView(
    children: [
     Image.asset(
      crop["image"],
      height: 250,
      width: double.infinity,
      fit: BoxFit.cover,
     ),
     SizedBox(height: 20),
     buildListTile(context, "Introduction",
crop["introduction"]),
     buildListTile(context, "Climate",
crop["climate"]),
     buildListTile(context, "Soil",
crop["soil"]),
     buildListTile(context, "Sowing",
crop["sowing"]),
     buildListTile(context, "Land Preparation",
crop["land preparation"]),
     buildListTile(context, "Fertilizers",
crop["fertilizers"]),
     buildListTile(context, "Major Diseases
and their Management", crop["disease"]),
     buildListTile(context, "Harvesting",
crop["harvesting"]),
     buildListTile(context, "Yield",
crop["yield"]),
    ],
   ),
  );
 }

 Widget buildListTile(BuildContext context,
String label, String content) {
  return ListTile(
   title: Text(label),
   onTap: () {
    Navigator.push(
     context,
     MaterialPageRoute(
      builder: (context) => DetailsPage(label:
label, content: content),
     ),
    );
   },
  );
 }
}
```

details_page.dart

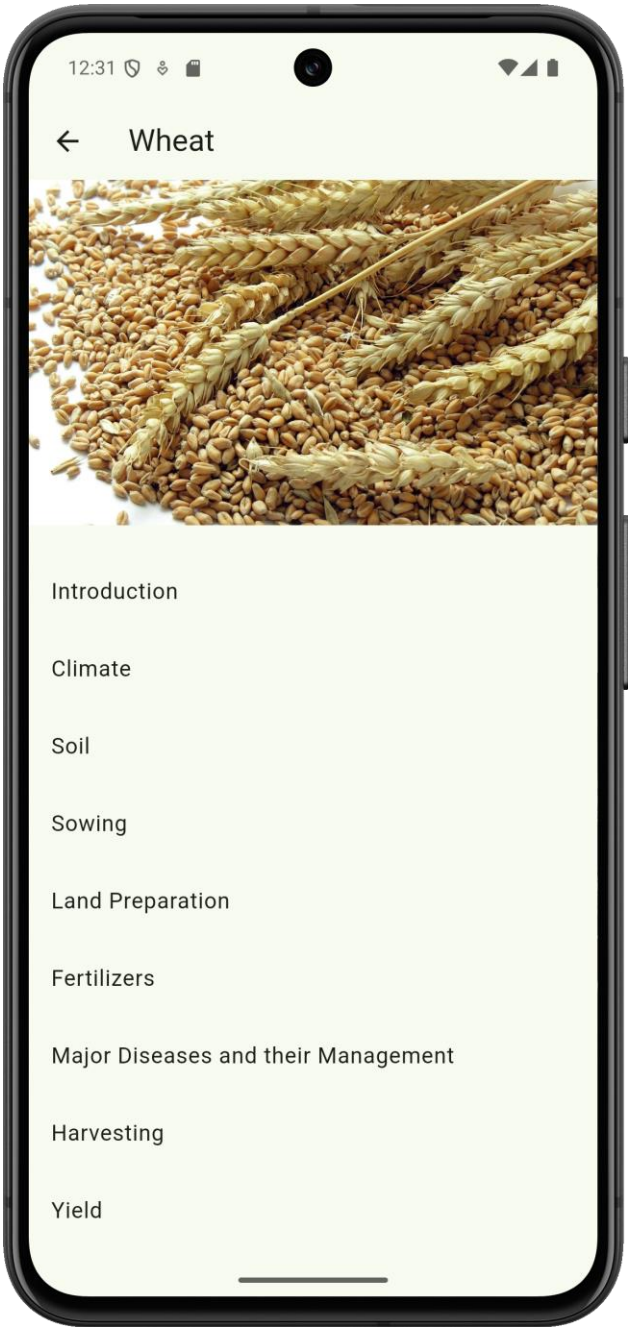
```
import 'package:flutter/material.dart';

class DetailsPage extends StatelessWidget {
  final String label;
  final String content;

  DetailsPage({required this.label, required
this.content});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('$label')),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: SingleChildScrollView(
          child: Text(
            content,
            style: TextStyle(fontSize: 18),
          ),
        ),
      ),
    );
  }
}
```

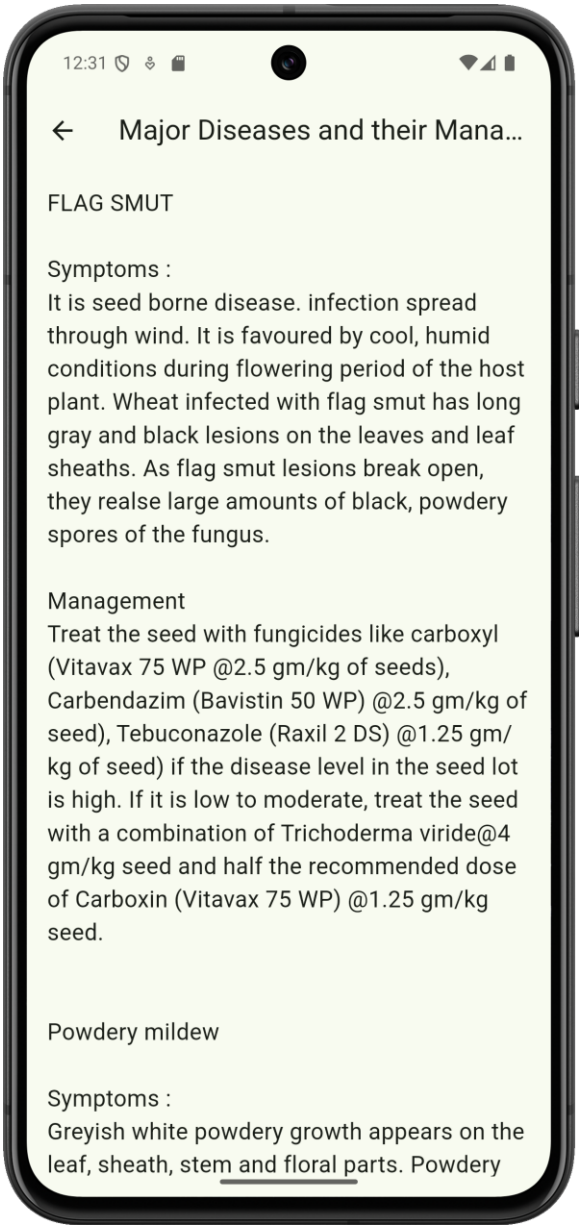
pubspec.yaml

```
flutter:
  assets:
    - assets/crops.json
    - assets/images/logo.png
    - assets/images/banner.jpg
    - assets/images/offers.png
    - assets/images/crop_tonics.png
    - assets/images/fertilizer.png
    - assets/images/pesticides.png
    - assets/images/delivery.png
    - assets/images/product1.png
    - assets/images/product2.png
    - assets/images/product3.png
    - assets/images/product4.png
    - assets/images/product5.png
    - assets/images/product6.png
    - assets/images/product7.png
    - assets/images/product8.png
    - assets/images/product9.png
    - assets/images/product10.png
    - assets/images/product11.png
    - assets/images/product12.png
    - assets/images/wheat.png
    - assets/images/paddy.png
    - assets/images/maize.jpg
    - assets/images/pearl_millet.png
    - assets/images/tomato.jpg
    - assets/images/cauliflower.jpg
    - assets/images/potato.jpg
    - assets/images/brinjal.jpg
    - assets/images/cabbage.jpg
    - assets/images/chilli.jpg
    - assets/images/onion.jpg
    - assets/images/coriander.jpg
    - assets/images/garlic.png
    - assets/images/sugarcane.png
```

OUTPUT: -


12:30
Crop Practices
Wheat
Paddy
Maize
Pearl Millet
Tomato
Cauliflower
Potato
Brinjal


12:31
Wheat
Introduction
Climate
Soil
Sowing
Land Preparation
Fertilizers
Major Diseases and their Management
Harvesting
Yield


12:31
Major Diseases and their Mana...
FLAG SMUT
Symptoms :
It is seed borne disease. infection spread through wind. It is favoured by cool, humid conditions during flowering period of the host plant. Wheat infected with flag smut has long gray and black lesions on the leaves and leaf sheaths. As flag smut lesions break open, they realse large amounts of black, powdery spores of the fungus.
Management
Treat the seed with fungicides like carboxyl (Vitavax 75 WP @2.5 gm/kg of seeds), Carbendazim (Bavistin 50 WP) @2.5 gm/kg of seed), Tebuconazole (Raxil 2 DS) @1.25 gm/ kg of seed) if the disease level in the seed lot is high. If it is low to moderate, treat the seed with a combination of Trichoderma viride@4 gm/kg seed and half the recommended dose of Carboxin (Vitavax 75 WP) @1.25 gm/kg seed.
Powdery mildew
Symptoms :
Greyish white powdery growth appears on the leaf, sheath, stem and floral parts. Powdery

MAD & PWA Lab

Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	18
Name	Tejas Dhondibhau Gunjal
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: - 04

Name:- Tejas Gunjal **Class:-** D15A **Roll:No: -** 18

AIM: - To create an interactive Form using form widget.

Theory: -

A form in Flutter is a structured container that collects user input through various fields like text fields, dropdowns, checkboxes, and buttons. It plays a crucial role in applications that require user data entry, such as login pages, registration forms, and feedback submissions. Flutter provides the Form widget, which works alongside TextFormField and other input elements to manage validation, state handling, and error messages efficiently. By using form validation techniques, developers can ensure data accuracy and enhance user experience.

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

Creation of a Form

- While creating a form in Flutter, the **Form widget** is essential as it acts as a container for grouping multiple form fields and managing validation.
- A **GlobalKey<FormState>** is required to uniquely identify the form and enable validation or data retrieval from the form fields.
- The **TextFormField widget** is used to provide input fields where users can enter data such as names, phone numbers, or email addresses.
- To enhance the appearance and usability of input fields, **InputDecoration** is used, allowing customization of labels, icons, borders, and hint text.
- Validation plays a crucial role in forms, and the **validator property** within **TextFormField** ensures user input meets specific criteria before submission.

- Different types of input require appropriate **keyboard types**, such as TextInputType.number for numeric fields or TextInputType.emailAddress for email fields.

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

- A **submit button** is necessary to trigger form validation and submit the collected data for further processing.

Some Properties of Form Widget

- **key:** A GlobalKey that uniquely identifies the Form. You can use this key to interact with the form, such as validating, resetting, or saving its state.
- **child:** The child widget that contains the form fields. Typically, this is a Column, ListView, or another widget that allows you to arrange the form fields vertically.
- **autovalidateMode:** An enum that specifies when the form should automatically validate its fields.

Some Methods of Form Widget

- **validate():** This method is used to trigger the validation of all the form fields within the Form. It returns true if all fields are valid, otherwise false. You can use it to check the overall validity of the form before submitting it.
- **save():** This method is used to save the current values of all form fields. It invokes the onSaved callback for each field. Typically, this method is called after validation succeeds.
- **reset():** Resets the form to its initial state, clearing any user-entered data.
- **currentState:** A getter that returns the current FormState associated with the Form.

Code: -

login_page.dart

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

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

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

  @override
  Widget build(BuildContext context) {
   return Scaffold(
     backgroundColor: Colors.white,
     body: SafeArea(
       child: SingleChildScrollView(
         child: Padding(
           padding: const EdgeInsets.all(16.0),
           child: Column(
            crossAxisAlignment:
CrossAxisAlignment.stretch,
            children: [
             SizedBox(height: 50),
             // Logo
             Center(
              child: Image.asset(
               'assets/images/logo.png',
               height: 150,
              ),
             ),
             SizedBox(height: 16),
             // Title
             Text(
              'AgriApp: The mix of Agriculture &
Smart, Scientific, Sustainable, Modern
Technology Methods for Precision Farming.',
              textAlign: TextAlign.center,
              style: TextStyle(
               fontSize: 15,
               color: Colors.black,
              ),
             ),
             SizedBox(height: 32),
             // Form
             Form(
              key: _formKey,
              child: Column(
               children: [
                _buildTextField(
                 label: 'Email',
                 hint: 'Enter your email',
                 icon: Icons.email,
                 validator: (value) {
                  if (value == null ||
value.isEmpty) {
                   return 'Email is required';
                  } else if
(!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(
value)) {
                   return 'Enter a valid email
address';
                  }
                  return null;
                 },
                ),
                SizedBox(height: 16),
                _buildTextField(
                 label: 'Password',
                 hint: 'Enter your password',
                 icon: Icons.lock,
                 isPassword: true,
                 validator: (value) {
                  if (value == null ||
value.isEmpty) {
                   return 'Password is required';
                  } else if (value.length < 6) {
                   return 'Password must be at
least 6 characters';
                  }
                  return null;
                 },
                ),
               ],
              ),
             ),
             SizedBox(height: 32),
```

```
            // Login Button
            ElevatedButton(
              onPressed: () {
                if
(_formKey.currentState!.validate()) {

Navigator.pushReplacementNamed(context,
'/home');
                }
              },
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.green,
                minimumSize: Size(double.infinity,
50),
                shape: RoundedRectangleBorder(
                  borderRadius:
BorderRadius.circular(8),
                ),
              ),
              child: Text(
                'Login',
                style: TextStyle(fontSize: 18, color:
Colors.white),
              ),
            ),
            SizedBox(height: 16),
            // Registration link
            Center(
              child: TextButton(
                onPressed: () {
                  Navigator.pushNamed(context,
'/register'); // Add registration route later
                },
                child: Text(
                  'Don’t have an account? Register',
                  style: TextStyle(color:
Colors.green), // Green color applied
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
 }
```

```
 Widget _buildTextField({
  required String label,
  required String hint,
  required IconData icon,
  bool isPassword = false,
  required String? Function(String?) validator,
 }) {
  return TextFormField(
   obscureText: isPassword,
   decoration: InputDecoration(
    labelText: label,
    hintText: hint,
    prefixIcon: Icon(icon, color: Colors.green),
    border: OutlineInputBorder(
     borderRadius: BorderRadius.circular(8),
    ),
    focusedBorder: OutlineInputBorder(
     borderSide: BorderSide(color:
Colors.green), // Green color applied
     borderRadius: BorderRadius.circular(8),
    ),
   ),
   validator: validator,
  );
 }
}
```

register_page.dart

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

class RegistrationPage extends StatefulWidget {
  @override
  _RegistrationPageState createState() =>
_RegistrationPageState();
}

class _RegistrationPageState extends
State<RegistrationPage> {
  final _formKey = GlobalKey<FormState>();
  bool _isLoading = false; // State to track
loading status

  String _name = '';
  String _email = '';
  String _state = '';
  String _district = '';
  String _password = '';

  void _handleRegistration() async {
    if (_formKey.currentState!.validate()) {
      setState(() {
        _isLoading = true; // Set loading to true
      });

      // Simulate a network request
      await Future.delayed(Duration(seconds: 2));

      setState(() {
        _isLoading = false; // Set loading to false
after request is done
      });

      // Navigate to the home screen after
successful registration
      Navigator.pushReplacementNamed(context,
'/otp');
    }
  }

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

backgroundColor: Colors.white,
      body: SafeArea(
        child: SingleChildScrollView(
          child: Padding(
            padding: const EdgeInsets.all(16.0),
            child: Column(
              crossAxisAlignment:
CrossAxisAlignment.stretch,
              children: [
                SizedBox(height: 50),
                // Logo
                Center(
                  child: Image.asset(
                    'assets/images/logo.png',
                    height: 150,
                  ),
                ),
                SizedBox(height: 16),

                Text(
                  'AgriApp: The mix of Agriculture &
Smart, Scientific, Sustainable, Modern
Technology Methods for Precision Farming.',
                  textAlign: TextAlign.center,
                  style: TextStyle(
                    fontSize: 15,
                    color: Colors.black,
                  ),
                ),
                SizedBox(height: 32),

                Form(
                  key: _formKey,
                  child: Column(
                    children: [
                      _buildTextField(
                        label: 'Name',
                        hint: 'Enter your name',
                        icon: Icons.person,
                        validator: (value) {
                          if (value == null ||
value.isEmpty) {
```

```
                    return 'Name is required';
                  }
                  return null;
                },
                onChanged: (value) {
                  _name = value;
                },
              ),
              SizedBox(height: 16),

              // Email Field
              _buildTextField(
                label: 'Email',
                hint: 'Enter your email',
                icon: Icons.email,
                validator: (value) {
                  if (value == null ||
value.isEmpty) {
                    return 'Email is required';
                  } else if
(!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(
value)) {
                    return 'Enter a valid email
address';
                  }
                  return null;
                },
                onChanged: (value) {
                  _email = value;
                },
              ),
              SizedBox(height: 16),

              // State Field (TextField or
Dropdown)
              _buildTextField(
                label: 'State',
                hint: 'Enter your state',
                icon: Icons.location_on,
                validator: (value) {
                  if (value == null ||
value.isEmpty) {
                    return 'State is required';
                  }
                  return null;
                },
                onChanged: (value) {
                  _state = value;
                },
              ),
              SizedBox(height: 16),

              // District Field (TextField or
Dropdown)
              _buildTextField(
                label: 'District',
                hint: 'Enter your district',
                icon: Icons.location_city,
                validator: (value) {
                  if (value == null ||
value.isEmpty) {
                    return 'District is required';
                  }
                  return null;
                },
                onChanged: (value) {
                  _district = value;
                },
              ),
              SizedBox(height: 16),

              // Password Field
              _buildTextField(
                label: 'Password',
                hint: 'Enter your password',
                icon: Icons.lock,
                isPassword: true,
                validator: (value) {
                  if (value == null ||
value.isEmpty) {
                    return 'Password is required';
                  } else if (value.length < 6) {
                    return 'Password must be at
least 6 characters';
                  }
                  return null;
                },
                onChanged: (value) {
                  _password = value;
                },
              ),
            ],
          ),
        ),
```

```
            SizedBox(height: 32),
            // Register Button
            ElevatedButton(
              onPressed: _handleRegistration,
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.green, // Green color applied
                minimumSize: Size(double.infinity, 50),
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(8),
                ),
              ),
              child: _isLoading
                  ? CircularProgressIndicator(color: Colors.white) // Show loading spinner
                  : Text(
                'Next',
                style: TextStyle(fontSize: 18, color: Colors.white),
              ),
            ),
            SizedBox(height: 16),
            // Login link (If user already has an account)
            Center(
              child: TextButton(
                onPressed: () {
                  Navigator.pushNamed(context, '/login'); // Add login route later
                },
                child: Text(
                  'Already have an account? Login',
                  style: TextStyle(color: Colors.green), // Green color applied
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
 }
```

```
 Widget _buildTextField({
  required String label,
  required String hint,
  required IconData icon,
  bool isPassword = false,
  required String? Function(String?) validator,
  required Function(String) onChanged,
 }) {
  return TextFormField(
   obscureText: isPassword,
   decoration: InputDecoration(
    labelText: label,
    hintText: hint,
    prefixIcon: Icon(icon, color: Colors.green),
    border: OutlineInputBorder(
     borderRadius: BorderRadius.circular(8),
    ),
    focusedBorder: OutlineInputBorder(
     borderSide: BorderSide(color: Colors.green),
     borderRadius: BorderRadius.circular(8),
    ),
   ),
   validator: validator,
   onChanged: onChanged,
  );
 }
}
```

otp_verification_page.dart

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

class OtpVerificationPage extends
StatefulWidget {
  @override
  _OtpVerificationPageState createState() =>
_OtpVerificationPageState();
}

class _OtpVerificationPageState extends
State<OtpVerificationPage> {
  final _formKey = GlobalKey<FormState>();
  String _otp = '';

  void _verifyOtp() {
    if (_formKey.currentState!.validate()) {


ScaffoldMessenger.of(context).showSnackBar(
       SnackBar(content: Text('OTP Verified!')),
     );

 Navigator.pushReplacementNamed(context,
'/home');
    }
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            crossAxisAlignment:
CrossAxisAlignment.stretch,
            children: [
              SizedBox(height: 50),

              Center(
                child: Image.asset(
                  'assets/images/logo.png',
                  height: 100,
                ),
              ),
              SizedBox(height: 16),
              // App Description
              Center(
                child: Text(
                  'AgriApp: The mix of Agriculture &
Smart, Scientific, Sustainable, Modern
Technology Methods for Precision Farming.',
                  textAlign: TextAlign.center,
                  style: TextStyle(
                    fontSize: 15,
                    color: Colors.black,
                  ),
                ),
              ),
              SizedBox(height: 8),

              SizedBox(height: 40),
              // Title
              Text(
                'OTP Verification',
                textAlign: TextAlign.center,
                style: TextStyle(
                  fontSize: 24,
                  fontWeight: FontWeight.bold,
                  color: Colors.green, // Green color
applied
                ),
              ),
              SizedBox(height: 24),
              // Instructions
              Text(
                'Enter the OTP sent to your registered
email or phone number.',
                textAlign: TextAlign.center,
                style: TextStyle(fontSize: 16, color:
Colors.black),
              ),
              SizedBox(height: 24),
              Form(
                key: _formKey,
                child: TextFormField(
                  maxLength: 6,
                  keyboardType:
TextInputType.number,
                  decoration: InputDecoration(
                    labelText: 'OTP',
```

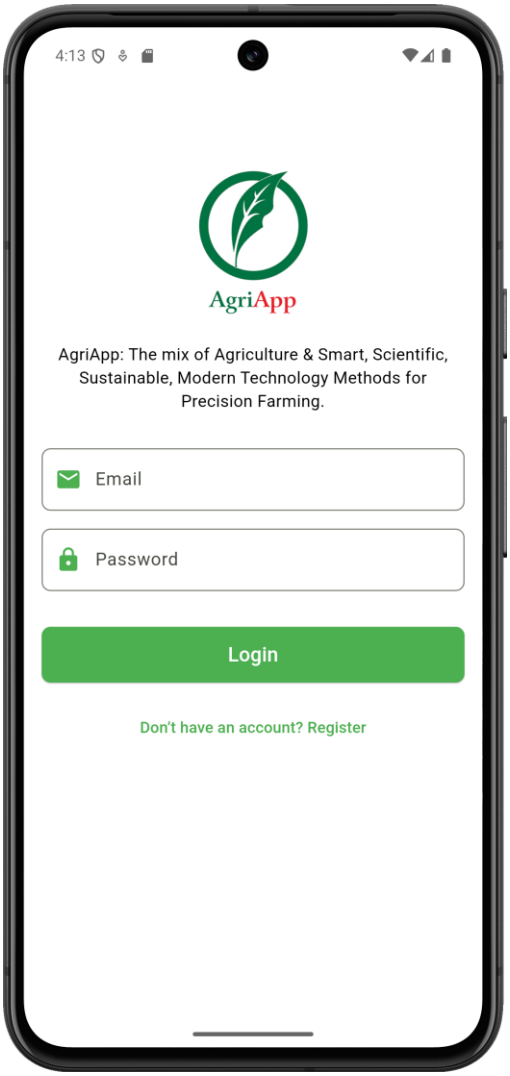
```
                hintText: 'Enter 6-digit OTP',
                prefixIcon: Icon(Icons.lock, color:
Colors.green),
                border: OutlineInputBorder(
                  borderRadius:
BorderRadius.circular(8),
                ),
                focusedBorder: OutlineInputBorder(
                  borderSide: BorderSide(color:
Colors.green),
                  borderRadius:
BorderRadius.circular(8),
                ),
              ),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'OTP is required';
                } else if (value.length != 6) {
                  return 'Enter a valid 6-digit OTP';
                }
                return null;
              },
              onChanged: (value) {
                _otp = value;
              },
            ),
          ),
          SizedBox(height: 24), ElevatedButton(
            onPressed: _verifyOtp,
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.green,
              minimumSize: Size(double.infinity,
50),
              shape: RoundedRectangleBorder(
                borderRadius:
BorderRadius.circular(8),
              ),
            ),
            child: Text(
              'Verify OTP',
              style: TextStyle(fontSize: 18, color:
Colors.white),
            ),
          ),
          SizedBox(height: 16),

          Center(
            child: TextButton(
              onPressed: () {

ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(content: Text('OTP
Resent!')),
                );
              },
              child: Text(
                'Resend OTP',
                style: TextStyle(color:
Colors.green), // Green color applied
              ),
            ),
          ),
        ],
      ),
    ),
  );
}
}
```

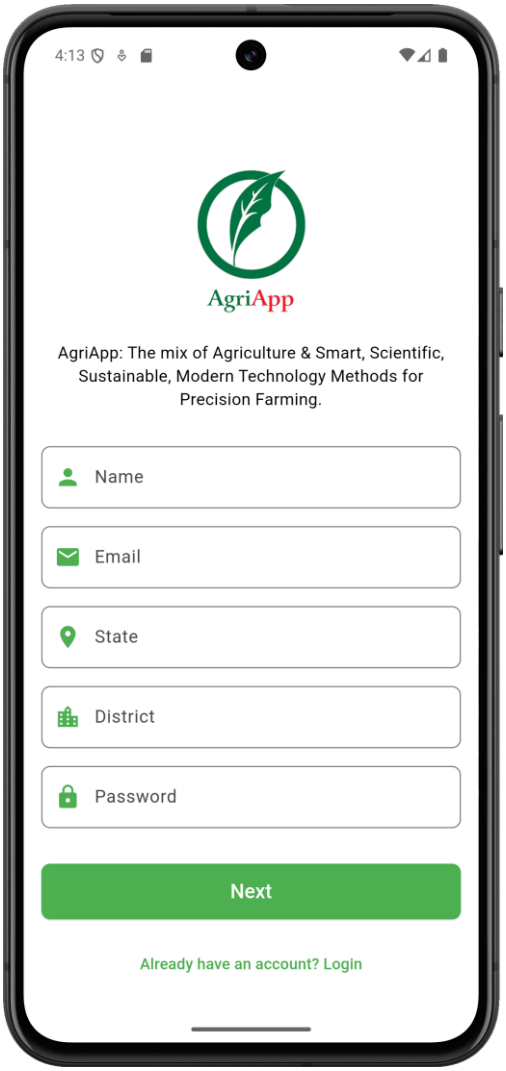
myaccountpage.dart

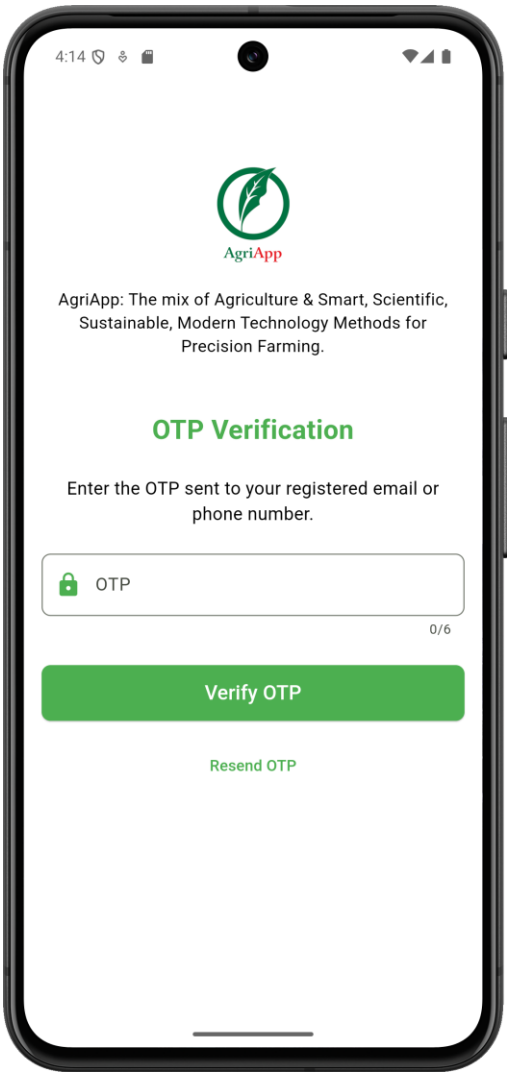
```
import 'package:flutter/material.dart';

class MyAccountPage extends StatefulWidget {
  @override
  _MyAccountPageState createState() =>
_MyAccountPageState();
}

class _MyAccountPageState extends
State<MyAccountPage> {
  final _formKey = GlobalKey<FormState>();
  String name = 'Tejas';
  String phone = '9082949011';
  String email = 'tejas@example.com';
  String state = 'Maharashtra';
  String language = 'English';
  List<String> myCrops = ['Beetroot', 'Brinjal',
'Onion', 'Soybean', 'Sugarcane'];

  @override
  Widget build(BuildContext context) {
   return Scaffold(
    backgroundColor: Colors.white,
    appBar: AppBar(
     title: Text('My Account'),
     leading: IconButton(
      icon: Icon(Icons.arrow_back),
      onPressed: () {
       Navigator.pop(context);
      },
     ),
    ),
    body: Padding(
     padding: const EdgeInsets.all(16.0),
     child: SingleChildScrollView(
      child: Column(
       crossAxisAlignment:
CrossAxisAlignment.start,
       children: [
        Center(
         child: CircleAvatar(
          radius: 50,
          backgroundColor: Colors.grey,
          child: Icon(Icons.person, size: 50,
color: Colors.white),
         ),
        ),
        SizedBox(height: 20),
        _buildForm(),
        SizedBox(height: 50),
        Row(
         mainAxisAlignment:
MainAxisAlignment.spaceBetween,
         children: [
          ElevatedButton(
           onPressed: () {
            // Reset changes logic here
(optional)
           },
           style: ElevatedButton.styleFrom(
            backgroundColor: Colors.grey, //
Corrected here
           ),
           child: Text('Reset'),
          ),
          ElevatedButton(
           onPressed: () {

           },
           style: ElevatedButton.styleFrom(
            backgroundColor: Colors.green,
           ),
           child: Text('Save Changes'),
          ),
         ],
        ),
       ],
      ),
     ),
    ),
   );
  }

  Widget _buildForm() {
   return Form(
    key: _formKey,
    child: Column(
     children: [
      _buildTextField(
       label: 'Name',
       initialValue: name,
       onSaved: (value) {
```

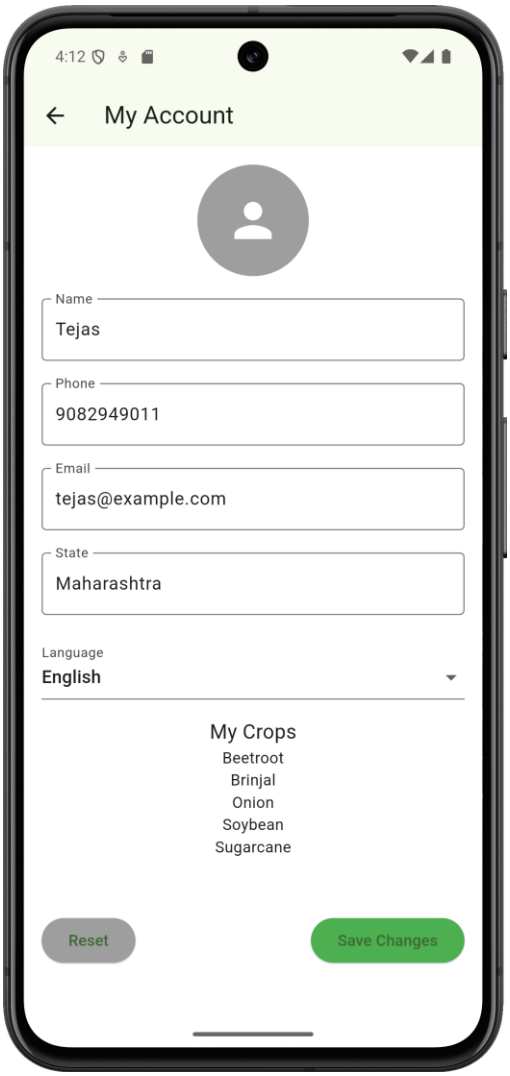
```
            name = value!;
          },
        ),
        SizedBox(height: 20),
        _buildTextField(
          label: 'Phone',
          initialValue: phone,
          onSaved: (value) {
            phone = value!;
          },
        ),
        SizedBox(height: 20),
        _buildTextField(
          label: 'Email',
          initialValue: email,
          onSaved: (value) {
            email = value!;
          },
        ),
        SizedBox(height: 20),
        _buildTextField(
          label: 'State',
          initialValue: state,
          onSaved: (value) {
            state = value!;
          },
        ),
        SizedBox(height: 20),
        DropdownButtonFormField<String>(
          value: language,
          items: ['English', 'Hindi',
'Marathi'].map((String language) {
            return DropdownMenuItem<String>(
              value: language,
              child: Text(language),
            );
          }).toList(),
          onChanged: (value) {
            setState(() {
              language = value!;
            });
          },
          decoration: InputDecoration(labelText:
'Language'),
        ),
        SizedBox(height: 16),
        Text('My Crops', style:
TextStyle(fontSize: 18)),
          Column(
            children: myCrops.map((crop) =>
Text(crop)).toList(),
          ),
        ],
      ),
    );
  }

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

    );
  }
}
```

OUTPUT: -


4:13
AgriApp
AgriApp: The mix of Agriculture & Smart, Scientific, Sustainable, Modern Technology Methods for Precision Farming.
Email
Password
Login
Don't have an account? Register


4:13
AgriApp
AgriApp: The mix of Agriculture & Smart, Scientific, Sustainable, Modern Technology Methods for Precision Farming.
Name
Email
State
District
Password
Next
Already have an account? Login


4:14
AgriApp
AgriApp: The mix of Agriculture & Smart, Scientific, Sustainable, Modern Technology Methods for Precision Farming.
OTP Verification
Enter the OTP sent to your registered email or phone number.
OTP
0/6
Verify OTP
Resend OTP


4:12
My Account
Name
Tejas
Phone
9082949011
Email
tejas@example.com
State
Maharashtra
Language
English
My Crops
Beetroot
Brinjal
Onion
Soybean
Sugarcane
Reset
Save Changes

MAD & PWA Lab

Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	18
Name	Tejas Dhondibhau Gunjal
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: - 05

Name:- Tejas Gunjal **Class:-** D15A **Roll:No: -** 18

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

Theory: -

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

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

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

- **Navigation and Routing in Flutter**

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

  1. Using Navigator Widget

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

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

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

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

2. **Named Routes**

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

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

Navigate to the route using Navigator.pushNamed()

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

Handling Gestures in Flutter

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

Tap Gestures

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

Long Press Gesture

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

Swipe and Drag Gestures

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

Code: -

main.dart

```
import 'package:flutter/material.dart';
import 'pages/login_page.dart';
import 'pages/register_page.dart';
import 'pages/otp_verification_page.dart';
import 'pages/myaccountpage.dart';
import 'pages/home_page.dart';


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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'AgriApp',
      theme: ThemeData(
        primarySwatch: Colors.green,
        colorScheme: ColorScheme.fromSeed(seedColor:
Color(0xFF6A9A5B)),
      ),
      initialRoute: '/login', // Set initial page
      routes: {
        '/login': (context) => LoginPage(),
        '/register': (context) => RegistrationPage(),
        '/otp': (context) => OtpVerificationPage(),
        '/myaccount': (context) => MyAccountPage(),
        '/home': (context) => HomePage(),
      },
    );
  }
}
```

Login_page.dart

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

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

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: SingleChildScrollView(
          child: Padding(
            padding: const EdgeInsets.all(16.0),
            child: Column(
              crossAxisAlignment:
CrossAxisAlignment.stretch,
              children: [
                SizedBox(height: 50),
                // Logo
                Center(
                  child: Image.asset(
                    'assets/images/logo.png',
                    height: 150,
                  ),
                ),
                SizedBox(height: 16),
                // Title
                Text(
                  'AgriApp: The mix of Agriculture &
Smart, Scientific, Sustainable, Modern
Technology Methods for Precision Farming.',
                  textAlign: TextAlign.center,
                  style: TextStyle(
                    fontSize: 15,
                    color: Colors.black,
                  ),
                ),
                SizedBox(height: 32),
                // Form
                Form(
                  key: _formKey,
                  child: Column(
                    children: [
                      _buildTextField(
                        label: 'Email',
                        hint: 'Enter your email',
                        icon: Icons.email,
                        validator: (value) {
                          if (value == null ||
value.isEmpty) {
                            return 'Email is required';
                          } else if
(!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(
value)) {
                            return 'Enter a valid email
address';
                          }
                          return null;
                        },
                      ),
                      SizedBox(height: 16),
                      _buildTextField(
                        label: 'Password',
                        hint: 'Enter your password',
                        icon: Icons.lock,
                        isPassword: true,
                        validator: (value) {
                          if (value == null ||
value.isEmpty) {
                            return 'Password is required';
                          } else if (value.length < 6) {
                            return 'Password must be at
least 6 characters';
                          }
                          return null;
                        },
                      ),
                    ],
                  ),
                ),
                SizedBox(height: 32),
                // Login Button
                ElevatedButton(
                  onPressed: () {
                    if
```

```
(_formKey.currentState!.validate()) {

Navigator.pushReplacementNamed(context,
'/home');
                }
              },
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.green,
                minimumSize: Size(double.infinity,
50),
                shape: RoundedRectangleBorder(
                  borderRadius:
BorderRadius.circular(8),
                ),
              ),
              child: Text(
                'Login',
                style: TextStyle(fontSize: 18, color:
Colors.white),
              ),
            ),
            SizedBox(height: 16),
            // Registration link
            Center(
              child: TextButton(
                onPressed: () {
                  Navigator.pushNamed(context,
'/register'); // Add registration route later
                },
                child: Text(
                  'Don’t have an account? Register',
                  style: TextStyle(color:
Colors.green), // Green color applied
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
 }

 Widget _buildTextField({
  required String label,
  required String hint,
  required IconData icon,
  bool isPassword = false,
  required String? Function(String?) validator,
 }) {
  return TextFormField(
   obscureText: isPassword,
   decoration: InputDecoration(
    labelText: label,
    hintText: hint,
    prefixIcon: Icon(icon, color: Colors.green),
    border: OutlineInputBorder(
     borderRadius: BorderRadius.circular(8),
    ),
    focusedBorder: OutlineInputBorder(
     borderSide: BorderSide(color:
Colors.green), // Green color applied
     borderRadius: BorderRadius.circular(8),
    ),
   ),
   validator: validator,
  );
 }
}
```

weather_forecast.dart

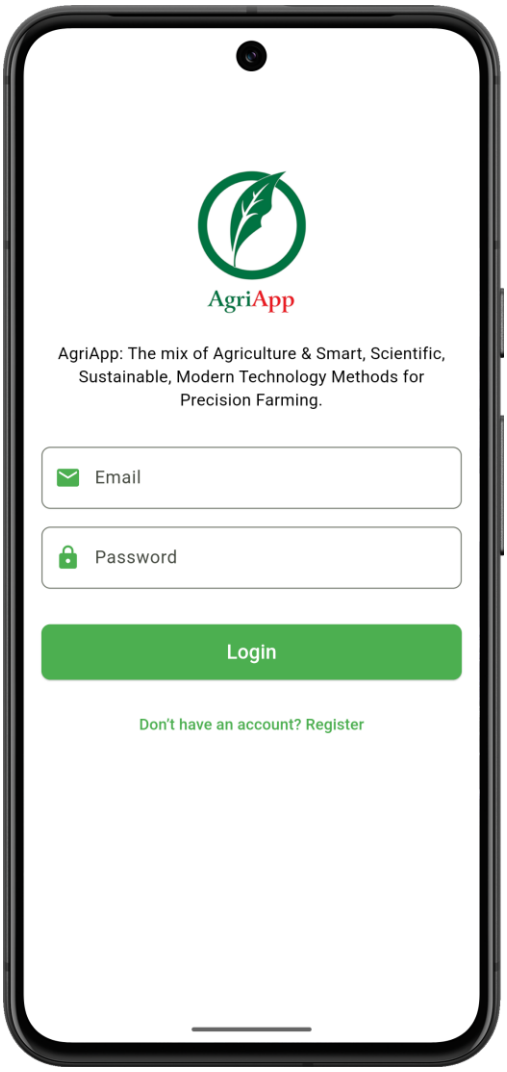
```
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';

class WeatherPage extends StatefulWidget {
  @override
  _WeatherPageState createState() =>
_WeatherPageState();
}

class _WeatherPageState extends
State<WeatherPage> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController _cityController =
TextEditingController();

  String? _city;
  String? _address;
  String? _updatedAt;
  String? _status;
  String? _temp;
  String? _tempMin;
  String? _tempMax;
  String? _windSpeed;
  String? _pressure;
  String? _humidity;
  String? _sunrise;
  String? _sunset;
  bool _isLoading = false;
  bool _isError = false;
  bool _isDataFetched = false; // Add this flag to
control the visibility of weather details

  Future<void> _fetchWeather() async {
    setState(() {
      _isLoading = true;
      _isError = false;
      _isDataFetched = false; // Reset this flag
when fetching new data
    });

    final response = await http.get(Uri.parse(

'https://api.openweathermap.org/data/2.5/weathe
r?q=$_city&units=metric&appid=73cbebdd0322
acd49bda6ede059b2b18'));

    if (response.statusCode == 200) {
      final data = jsonDecode(response.body);

      setState(() {
        _address = '${data['name']},
${data['sys']['country']}';
        _updatedAt = 'Updated At:
${DateTime.fromMillisecondsSinceEpoch(data['
dt'] * 1000).toString()}';
        _status =
data['weather'][0]['description'].toUpperCase();
        _temp = '${data['main']['temp']}°C';
        _tempMin = 'Min Temp:
${data['main']['temp_min']}°C';
        _tempMax = 'Max Temp:
${data['main']['temp_max']}°C';
        _pressure = 'Pressure:
${data['main']['pressure']} hPa';
        _humidity = 'Humidity:
${data['main']['humidity']}%';
        _windSpeed = 'Wind Speed:
${data['wind']['speed']} m/s';
        _sunrise =
DateTime.fromMillisecondsSinceEpoch(data['sy
s']['sunrise'] * 1000).toString();
        _sunset =
DateTime.fromMillisecondsSinceEpoch(data['sy
s']['sunset'] * 1000).toString();
        _isLoading = false;
        _isDataFetched = true; // Set the flag to true
after data is fetched
      });
    } else {
      setState(() {
        _isLoading = false;
        _isError = true;
        _isDataFetched = false; // Reset the flag if
there is an error
      });
    }
  }

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

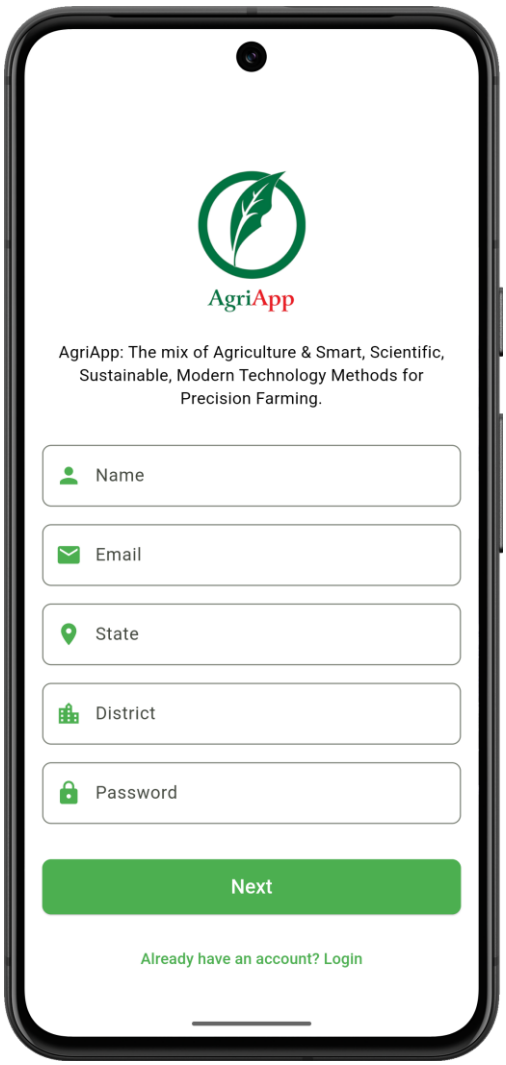
```
    appBar: AppBar(
      title: Text('Weather Forecasting'),
      //backgroundColor: Colors.white,
    ),
    //backgroundColor: Colors.white,
    body: SafeArea(
      child: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            crossAxisAlignment:
CrossAxisAlignment.stretch,
            children: [
              SizedBox(height: 50),
              // Logo
              Center(
                child: Image.asset(
                  'assets/images/logo.png', // Replace
with your logo
                  height: 150,
                ),
              ),
              SizedBox(height: 16),
              // Title
              Text(
                'Weather Forecasting',
                textAlign: TextAlign.center,
                style: TextStyle(
                  fontSize: 18,
                  color: Colors.black,
                ),
              ),
              SizedBox(height: 32),
              // Form
              Form(
                key: _formKey,
                child: Column(
                  children: [
                    TextFormField(
                      controller: _cityController,
                      decoration: InputDecoration(
                        labelText: 'City',
                        hintText: 'Enter the city name',
                        prefixIcon:
Icon(Icons.location_city, color: Colors.green),
                        border: OutlineInputBorder(
                          borderRadius:
BorderRadius.circular(8),
                        ),
                        focusedBorder:
OutlineInputBorder(
                          borderSide: BorderSide(color:
Colors.green),
                          borderRadius:
BorderRadius.circular(8),
                        ),
                      ),
                      validator: (value) {
                        if (value == null ||
value.isEmpty) {
                          return 'City is required';
                        }
                        return null;
                      },
                    ),
                    SizedBox(height: 16),
                    ElevatedButton(
                      onPressed: () {
                        if
(_formKey.currentState!.validate()) {
                          setState(() {
                            _city = _cityController.text;
                          });
                          _fetchWeather();
                        }
                      },
                      style: ElevatedButton.styleFrom(
                        backgroundColor: Colors.green,
                        minimumSize:
Size(double.infinity, 50),
                        shape:
RoundedRectangleBorder(
                          borderRadius:
BorderRadius.circular(8),
                        ),
                      ),
                      child: Text(
                        'Get Weather',
                        style: TextStyle(fontSize: 18,
color: Colors.white),
                      ),
                    ),
                  ],
                ),
```

```
            ),
            SizedBox(height: 32),
            _isLoading
                ? CircularProgressIndicator()
                : _isError
                ? Text(
              'Error fetching weather data.',
              style: TextStyle(color: Colors.red),
              textAlign: TextAlign.center,
            )
                : _isDataFetched
                ? Column(
              crossAxisAlignment:
CrossAxisAlignment.start,
              children: [
                Text(
                  'Location: $_address',
                  style: TextStyle(fontSize: 16,
fontWeight: FontWeight.bold),
                ),
                SizedBox(height: 8),
                Text('$_updatedAt'),
                SizedBox(height: 16),
                Text('Status: $_status'),
                SizedBox(height: 8),
                Text('Temperature: $_temp'),
                SizedBox(height: 8),
                Text('$_tempMin'),
                SizedBox(height: 8),
                Text('$_tempMax'),
                SizedBox(height: 8),
                Text('$_windSpeed'),
                SizedBox(height: 8),
                Text('$_pressure'),
                SizedBox(height: 8),
                Text('$_humidity'),
                SizedBox(height: 8),
                Text('Sunrise: $_sunrise'),
                SizedBox(height: 8),
                Text('Sunset: $_sunset'),
              ],
            )
                : Container(),
          ],
        ),
      ),
    ),
                ),
              );
            }
          }
```

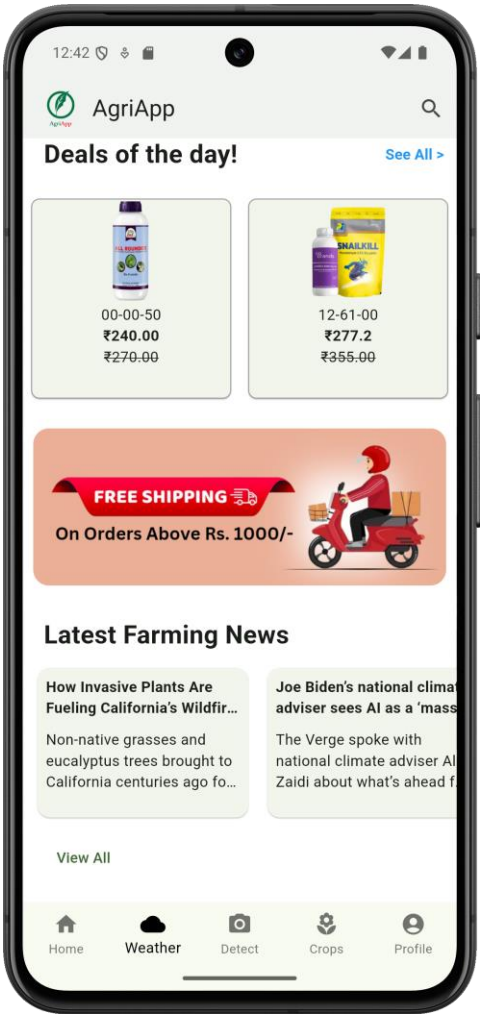
OUTPUT: -

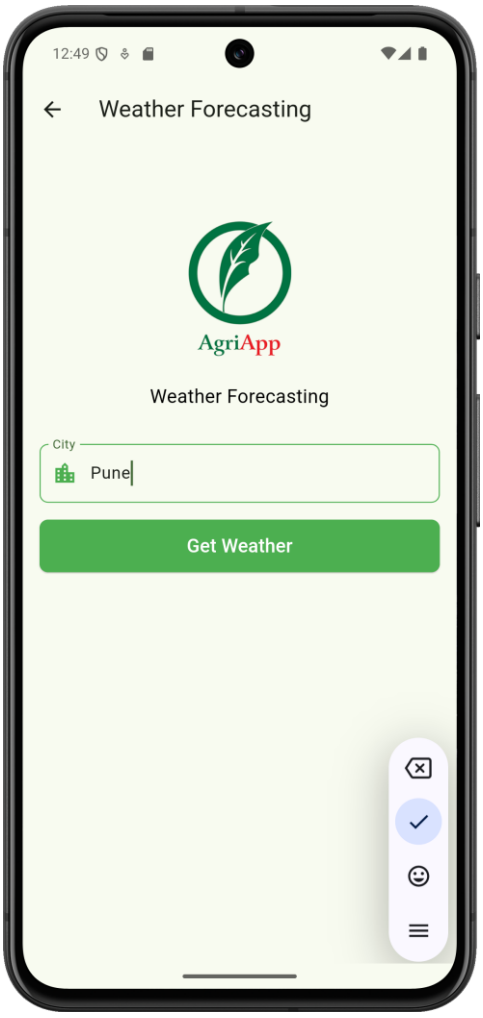
After clicking on Don’t have an account? it navigates to the registration page.





In home page, after clicking on Weather icon it navigates to the Weather page.





MAD & PWA Lab

Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	18
Name	Tejas Dhondibhau Gunjal
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

EXPERIMENT NO: - 06

Name:- Tejas Gunjal **Class:-** D15A **Roll:No: -** 18

AIM: - To connect Flutter UI with Firebase database.

Theory: -

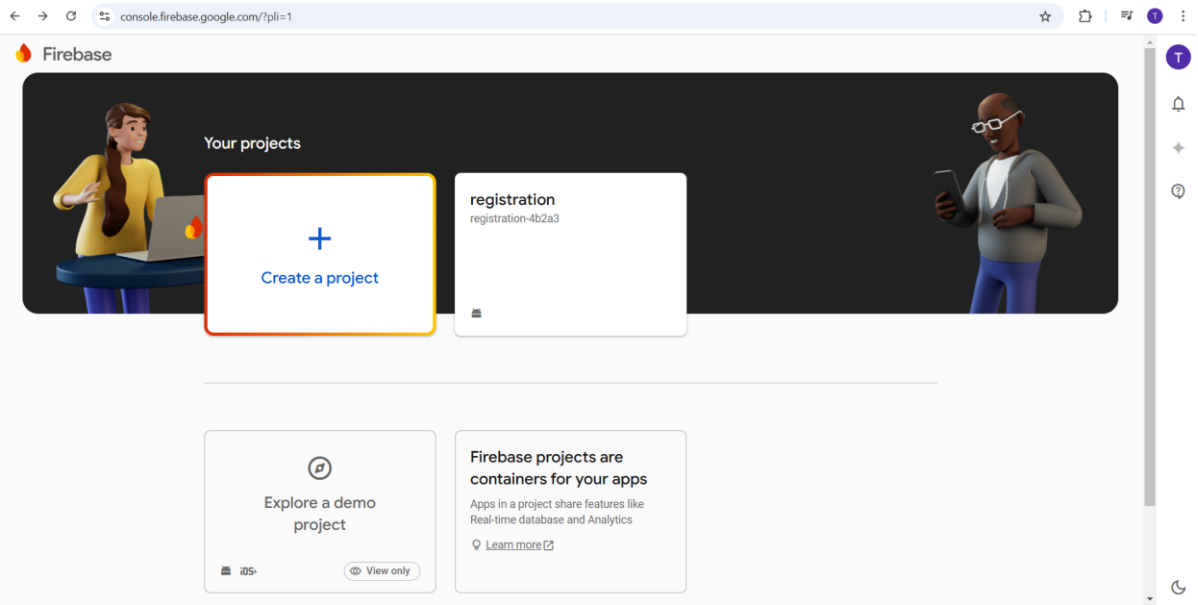
Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. Firebase, a Backend-as-a-Service (BaaS) platform, provides real-time database, authentication, and cloud storage services, making it a powerful backend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

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

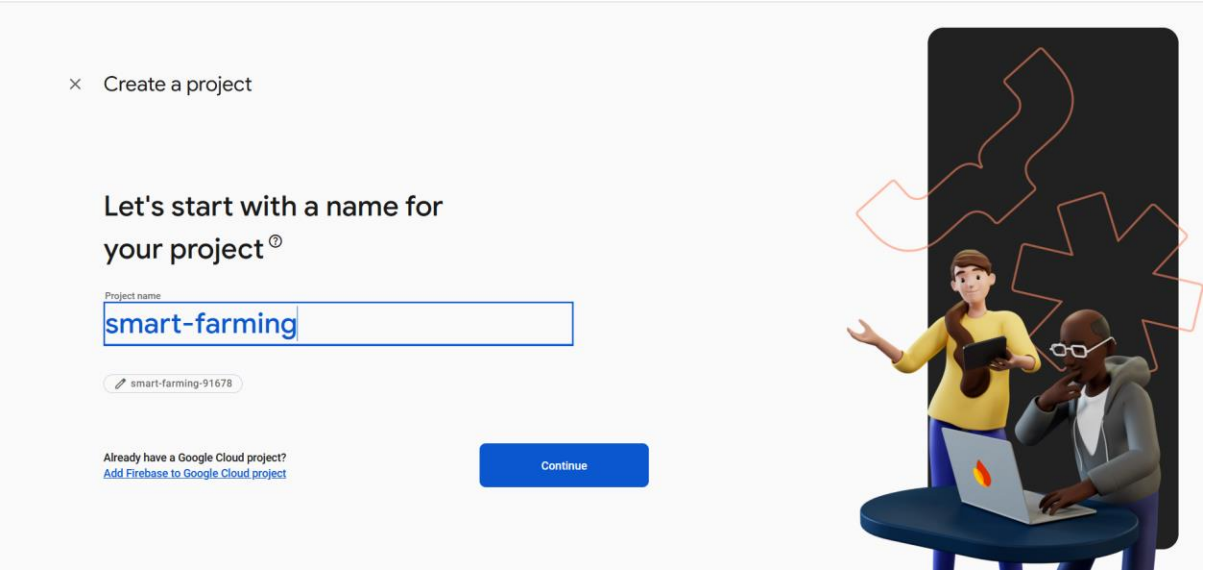
➢ Steps to Connect Flutter UI with Firebase Database

Step 1:

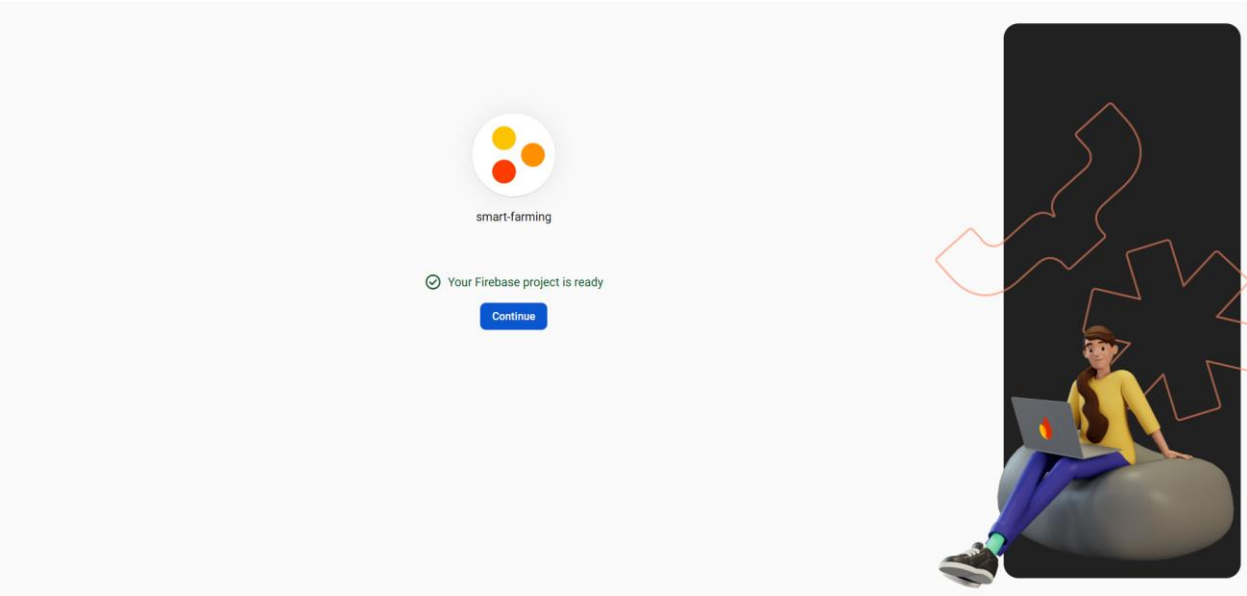
1.1) Go to Firebase Console and Create a Firebase Project



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

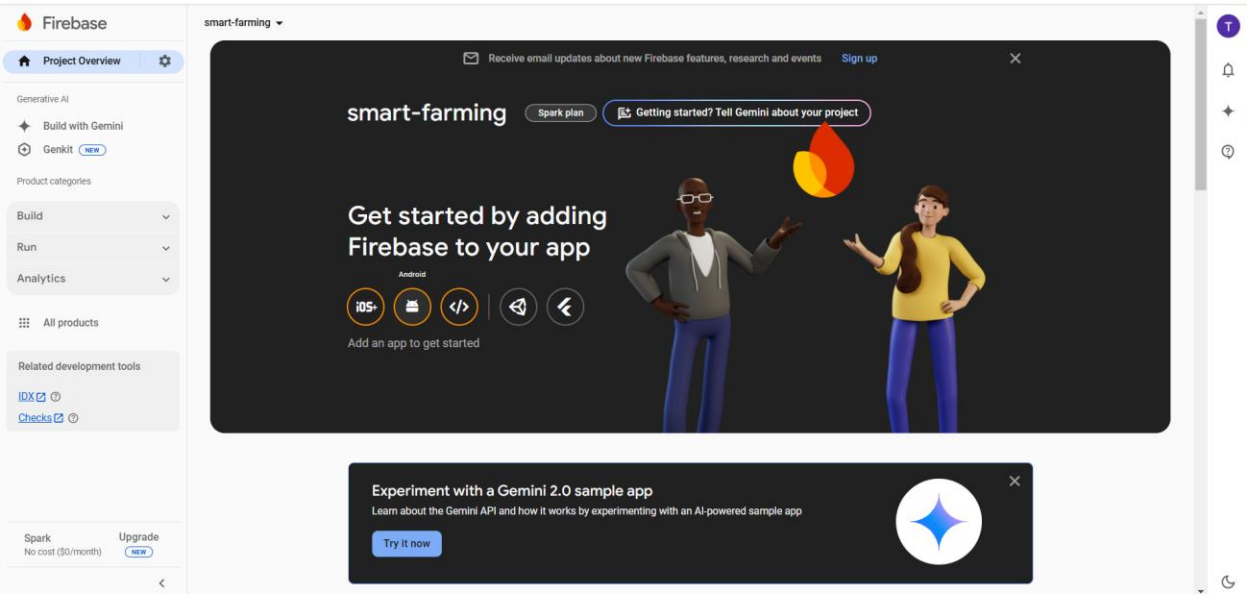


1.3) Enable Google Analytics (optional) & Click continue and complete the setup

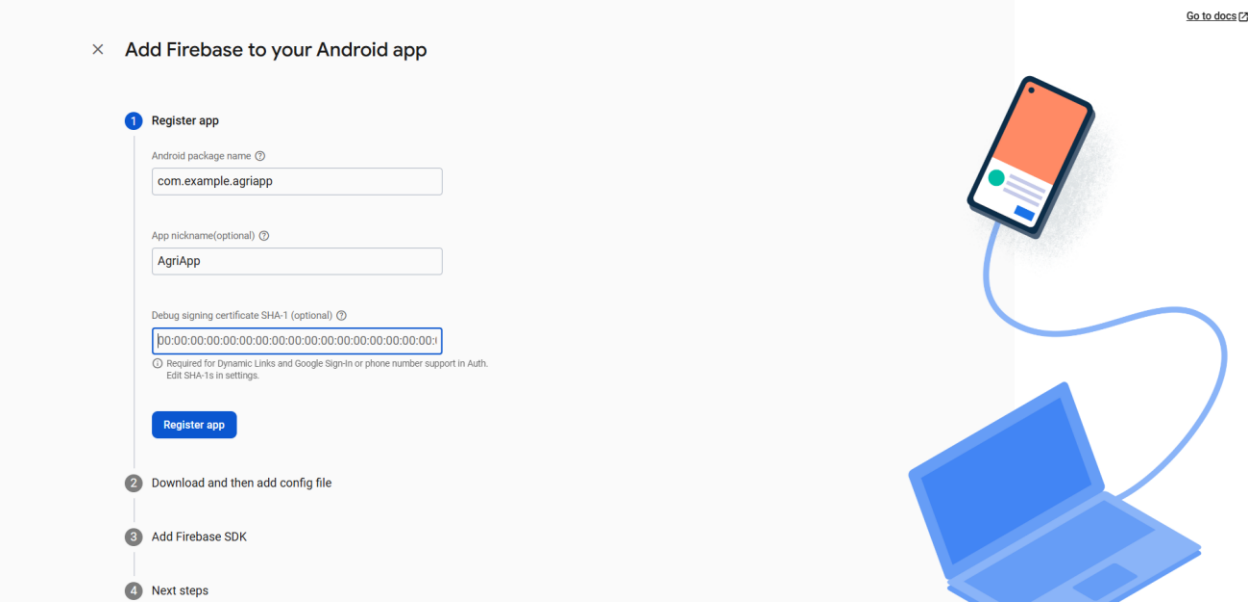


Step 2:- Add Firebase to Your Flutter App

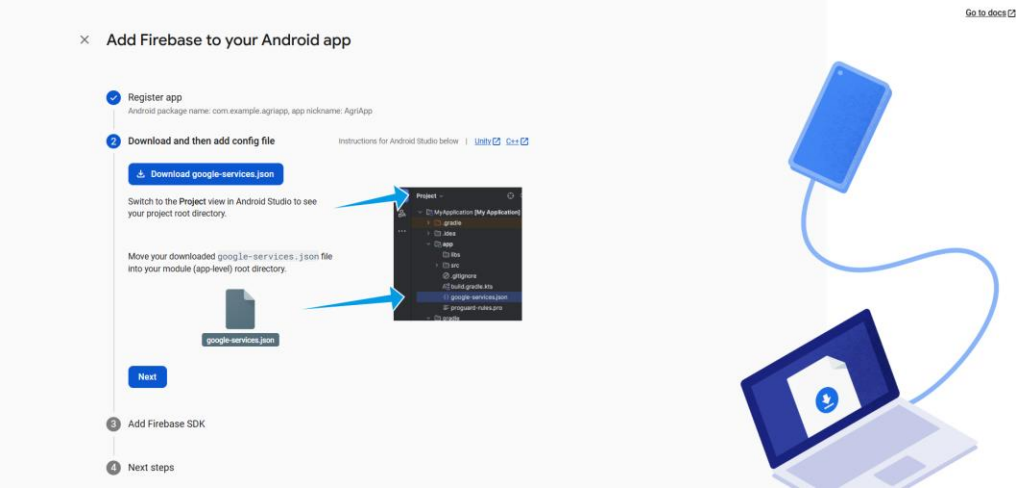
2.1) Click on Android/iOS/Web based on your Flutter application

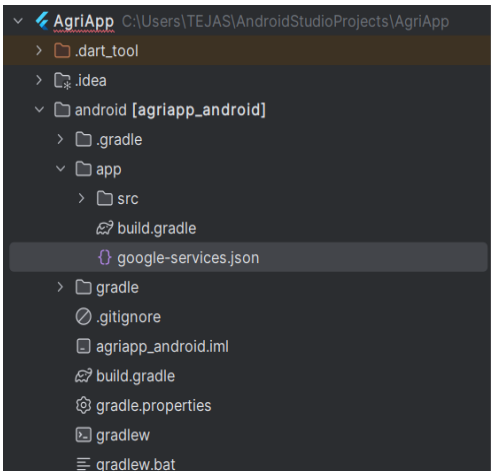


2.2) Register your app with a unique package name (found in android/app/build.gradle for Android).

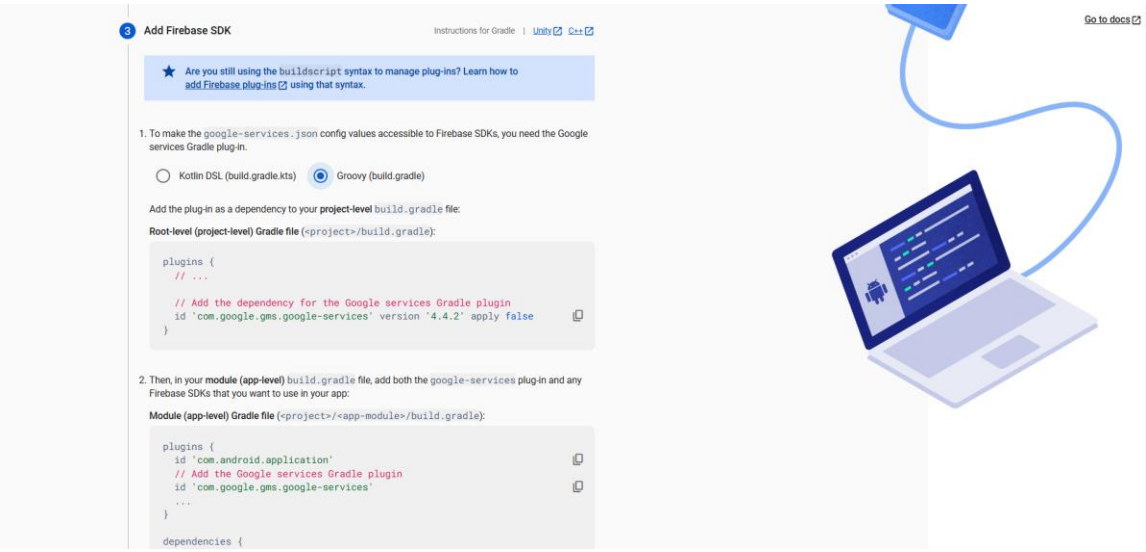


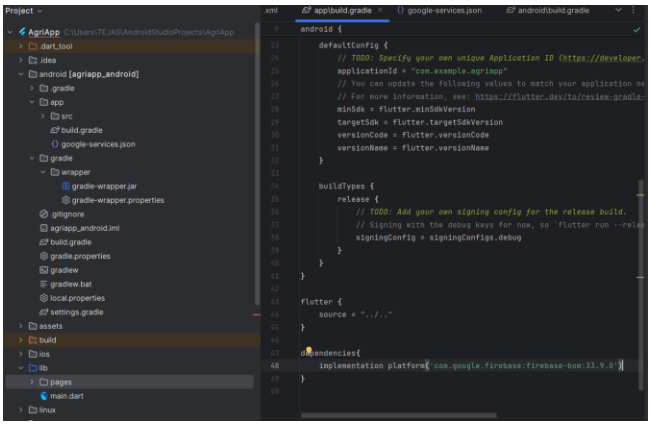
2.3) Download the google-services.json (for Android) & place the JSON file inside android/app/ directory.

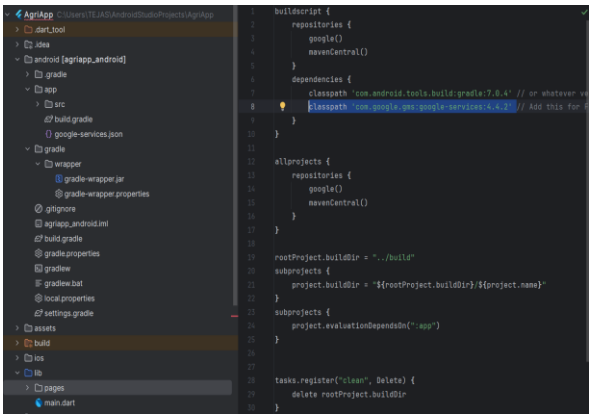




2.4) Add Firebase SDK dependencies to android/build.gradle





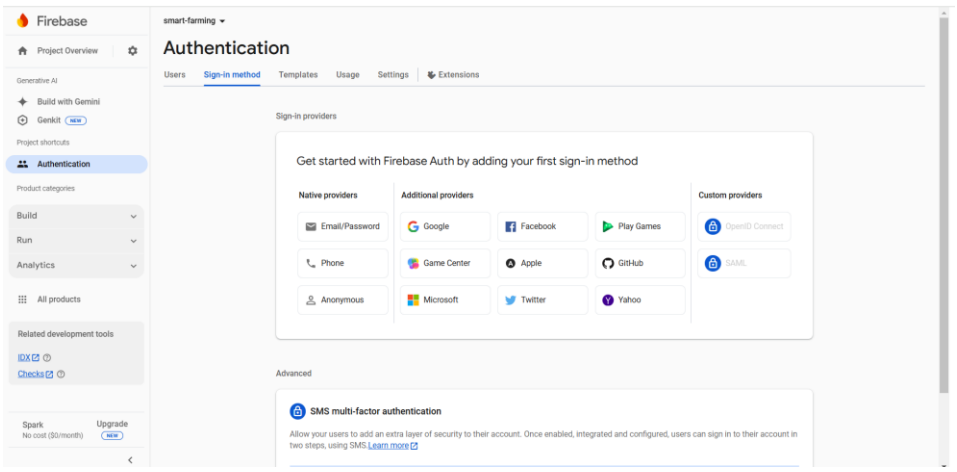


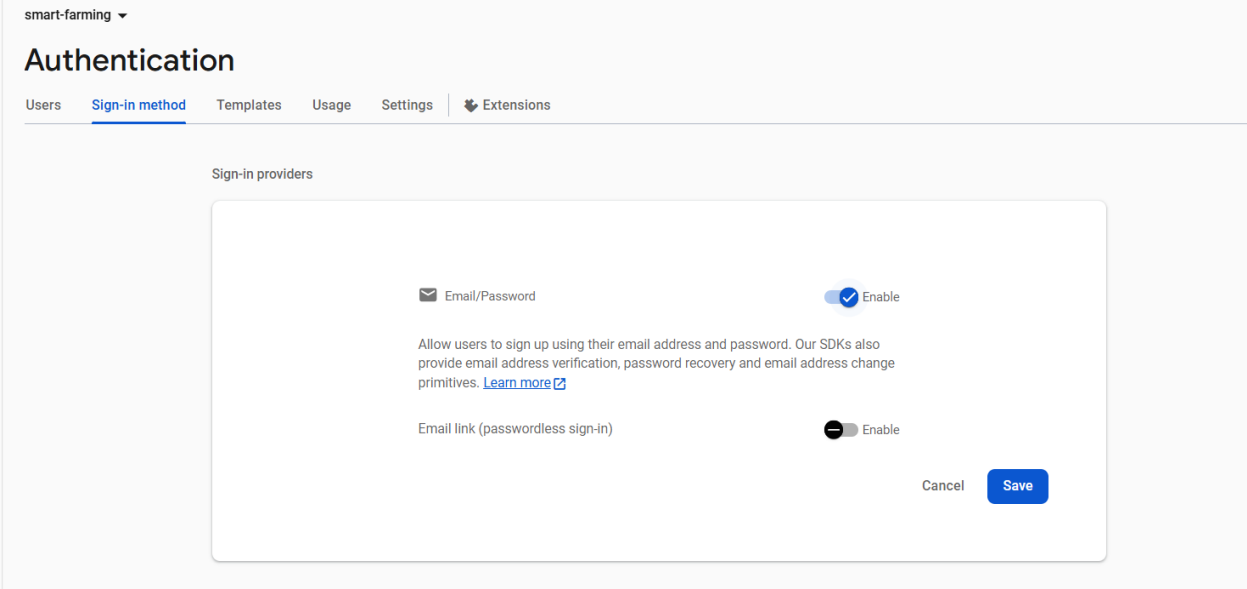
Step 3: - Add Firebase Authentication to Your App

3.1) Add Firebase Authentication Dependencies

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

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





3.3) Implement Authentication in Flutter
Modify main.dart

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

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

Step 4: -Configure Firebase Realtime Database

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

Code:-

Register_page.dart

```
class _RegistrationPageState extends
State<RegistrationPage> {
  final _formKey =
GlobalKey<FormState>();
  bool _isLoading = false;
  String _name = '';
  String _email = '';
  String _state = '';
  String _district = '';
  String _password = '';
  String _phone = '';

  final FirebaseAuth _auth =
FirebaseAuth.instance;

  // Firebase Registration
  Future<void> _handleRegistration() async
{
    if (_formKey.currentState!.validate()) {
      setState(() {
        _isLoading = true;
      });

      try {
        // Register user with Firebase
Authentication (Email and Password)
        UserCredential userCredential = await
_auth.createUserWithEmailAndPassword(
          email: _email,
          password: _password,
        );

        // Get the user ID from Firebase
Authentication user
        String userId =
userCredential.user!.uid;

        // Store extra user information in
Firestore
        await
FirebaseFirestore.instance.collection('users')
.doc(userId).set({
          'name': _name,
          'email': _email,
          'phone': _phone,
          'state': _state,
          'district': _district,
          // You can add other fields here
        });

        setState(() {
          _isLoading = false;
        });

        // Navigate to OTP verification page
after successful registration
        Navigator.pushReplacementNamed(
          context,
          '/otp',
          arguments: _phone, // Pass the phone
number here
        );
      } catch (e) {
        setState(() {
          _isLoading = false;
        });

ScaffoldMessenger.of(context).showSnackB
ar(
          SnackBar(content: Text("Registration
failed: $e")),
        );
      }
    }
  }
```

login_page.dart

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

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

class _LoginPageState extends
State<LoginPage> {
  final _formKey =
GlobalKey<FormState>();
  final FirebaseAuth _auth =
FirebaseAuth.instance;

  TextEditingController _emailController =
TextEditingController();
  TextEditingController _passwordController
= TextEditingController();

  bool _isLoading = false;
  String _errorMessage = '';

  Future<void> _loginUser() async {
    if (!_formKey.currentState!.validate())
return;

    setState(() {
      _isLoading = true;
      _errorMessage = '';
    });

    try {
      UserCredential userCredential = await
_auth.signInWithEmailAndPassword(
        email: _emailController.text.trim(),
        password:
_passwordController.text.trim(),
      );
      User? user = userCredential.user;

      if (user != null) {

Navigator.pushReplacementNamed(context,
'/home');
      } else {
        setState(() {
          _errorMessage = "Something went
wrong. Please try again.";
        });
      }
    } on FirebaseAuthException catch (e) {
      setState(() {
        _errorMessage = e.message ?? "An
error occurred. Please try again.";
      });
    }

    setState(() {
      _isLoading = false;
    });
  }
```

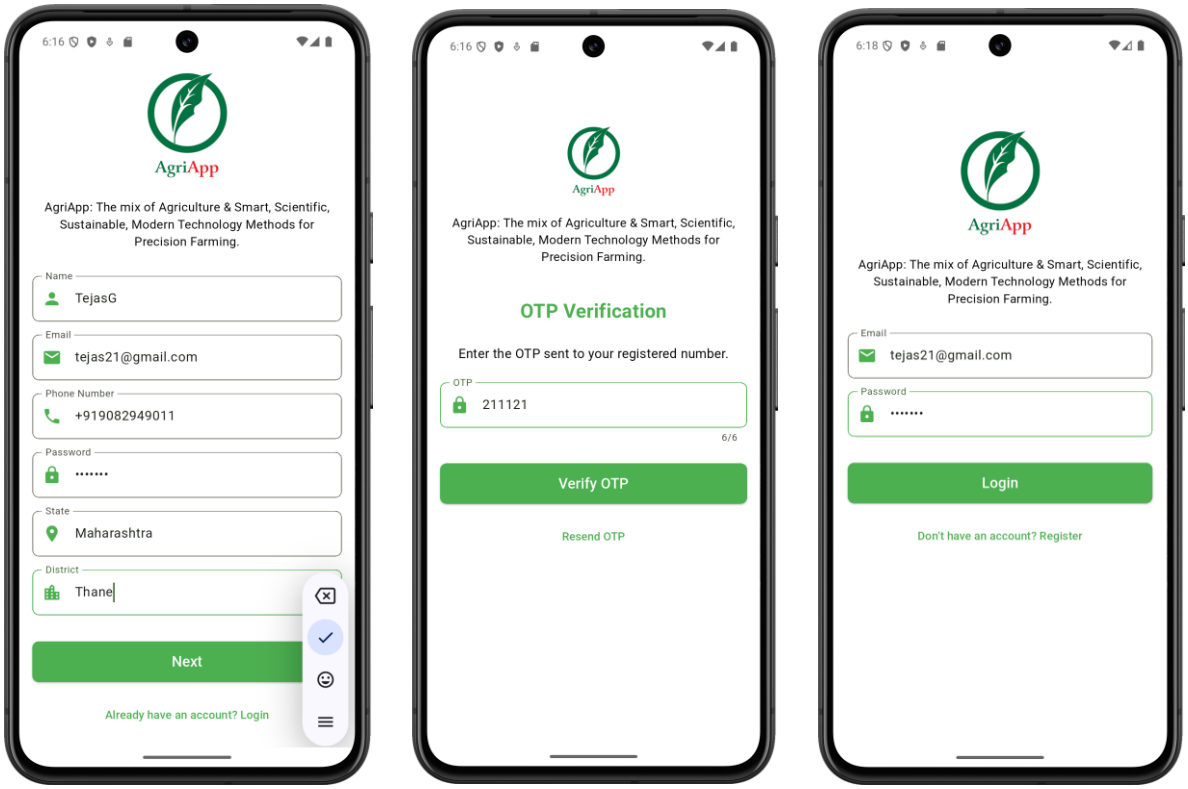
Myaccountpage.dart

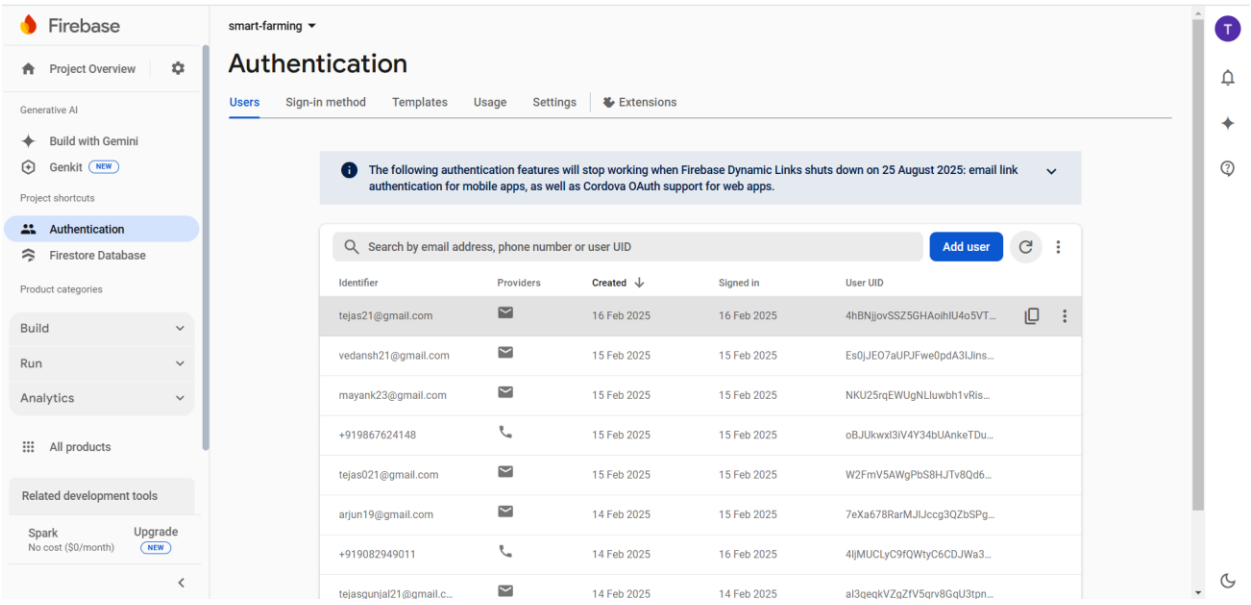
```
import
'package:cloud_firestore/cloud_firestore.dart
';
import
'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';

class MyAccountPage extends
StatefulWidget {
  @override
  _MyAccountPageState createState() =>
_MyAccountPageState();
}

class _MyAccountPageState extends
State<MyAccountPage> {
  final _formKey =
GlobalKey<FormState>();
  String name = '';
  String phone = '';
  String email = '';
  String state = '';
  String language = 'English';
  List<String> myCrops = [];
  bool isLoading = true;

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

  // Fetch data from Firestore
  _fetchUserData() async {
    User? user =
FirebaseAuth.instance.currentUser;

    if (user != null) {
      try {
        DocumentSnapshot doc = await
FirebaseFirestore.instance
            .collection('users')
            .doc(user.uid)
            .get();

        if (doc.exists) {
          setState(() {
            name = doc['name'];
            phone = doc['phone'];
            email = doc['email'];
            state = doc['state'];
            language = doc['language'];
            myCrops =
List<String>.from(doc['myCrops']);
            isLoading = false;
          });
        }
      } catch (e) {
        print("Error fetching user data: $e");
        setState(() {
          isLoading = false;
        });
      }
    }
  }

  // Save updated data to Firestore
  _saveUserData() async {
    User? user =
FirebaseAuth.instance.currentUser;

    if (user != null) {
      try {
        await
FirebaseFirestore.instance.collection('users')
.doc(user.uid).update({
          'name': name,
          'phone': phone,
          'email': email,
          'state': state,
          'language': language,
          'myCrops': myCrops,
        });
```

```
      // Show success message

ScaffoldMessenger.of(context).showSnackB
ar(SnackBar(content: Text('Changes
saved!')));
    } catch (e) {
      print("Error saving user data: $e");

ScaffoldMessenger.of(context).showSnackB
ar(SnackBar(content: Text('Error saving
changes')));
    }
  }
 }
```

Main.dart

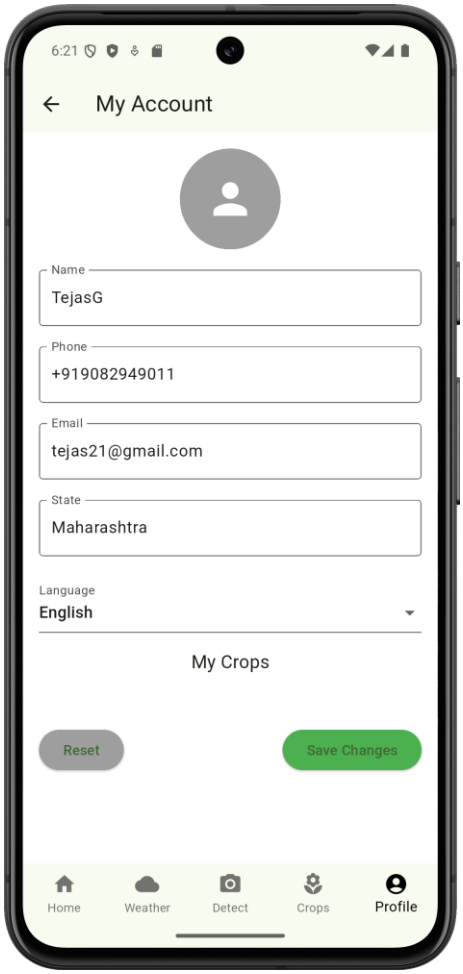
```
void main() async {
 WidgetsFlutterBinding.ensureInitialized();
 await Firebase.initializeApp();
 runApp(MyApp());
}

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   debugShowCheckedModeBanner: false,
   title: 'AgriApp',
   theme: ThemeData(
    primarySwatch: Colors.green,
    colorScheme:
ColorScheme.fromSeed(seedColor:
Color(0xFF6A9A5B)),
   ),
   home:
FirebaseAuth.instance.currentUser == null ?
LoginPage() : MainScreen(),
   routes: {
    '/login': (context) => LoginPage(),
    '/register': (context) =>
RegistrationPage(),
    '/otp': (context) =>
OtpVerificationPage(
     phoneNumber:
ModalRoute.of(context)!.settings.arguments
as String,
    ),
    '/home': (context) => MainScreen(),
   },
  );
 }
}

class MainScreen extends StatefulWidget {
 @override
 _MainScreenState createState() =>
_MainScreenState();
}

class _MainScreenState extends
State<MainScreen> {
 int _currentIndex = 0;

 final List<Widget> _pages = [
  HomePage(),
  WeatherPage(),
  DiseaseDetectionPage(),
  CropListPage(),
  MyAccountPage(),
 ];
```

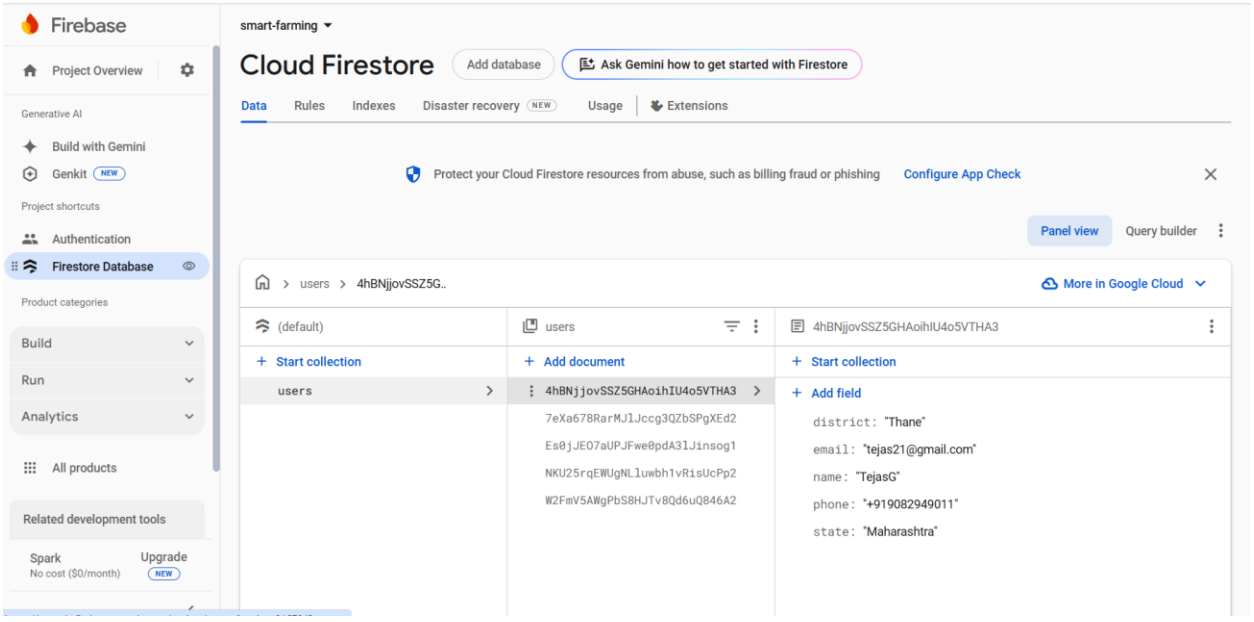




After Registering, the user details is saved in the database.

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





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.	18
Name	Tejas Dhondibhau Gunjal
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

EXPERIMENT NO. 7

Name:- Tejas Gunjal **Class:-** D15A **Roll:No: -** 18

Aim: To write metadata of your E-commerce Progressive Web App (PWA) in a Web App Manifest file to enable the “Add to Home Screen” feature.

Theory :

A **Progressive Web App (PWA)** is a web application that combines the best features of both websites and mobile apps. PWAs are designed to be fast, reliable, and engaging, offering features such as offline access, push notifications, and the ability to be installed on a user's device.

One of the key features of a PWA is the **"Add to Home Screen" (A2HS) functionality**, which allows users to install the web app on their mobile or desktop devices, just like a native app. To enable this feature, a **Web App Manifest** file is required.

PWAs leverage modern web technologies to deliver an app-like experience, eliminating the need for users to download applications from app stores. They provide seamless performance across different devices and platforms, ensuring accessibility and responsiveness. By utilizing service workers for caching, PWAs can function even in low or no-network conditions, making them an excellent choice for enhancing user engagement and retention.

WEB APP MANIFEST

The **Web App Manifest** is a JSON file (manifest.json) that provides important metadata about the PWA, such as its name, icons, colors, and display settings. This file is essential for enabling the **"Add to Home Screen"** feature and ensuring the PWA appears like a native app when installed.

Importance of the Web App Manifest:

1. **Enables App Installation** – Allows users to install the PWA on their home screen or desktop.
2. **Defines App Appearance** – Controls how the app is displayed (standalone, fullscreen, or minimal UI).
3. **Enhances User Experience** – Provides a consistent theme, splash screen, and branding.
4. **Supports Cross-Platform Compatibility** – Works on multiple devices and browsers.

Key Components of a Web App Manifest File (manifest.json)

1. **name and short_name**
   - Defines the full name and a shorter version of the app name used in installation prompts.
2. **start_url**
   - Specifies the URL to open when the app is launched.
3. **display**
   - Defines how the PWA will appear (e.g., standalone, fullscreen, minimal UI, or browser mode).
4. **background_color and theme_color**
   - Controls the splash screen and browser theme when the app is launched.
5. **icons**
   - Provides different icon sizes for various devices.

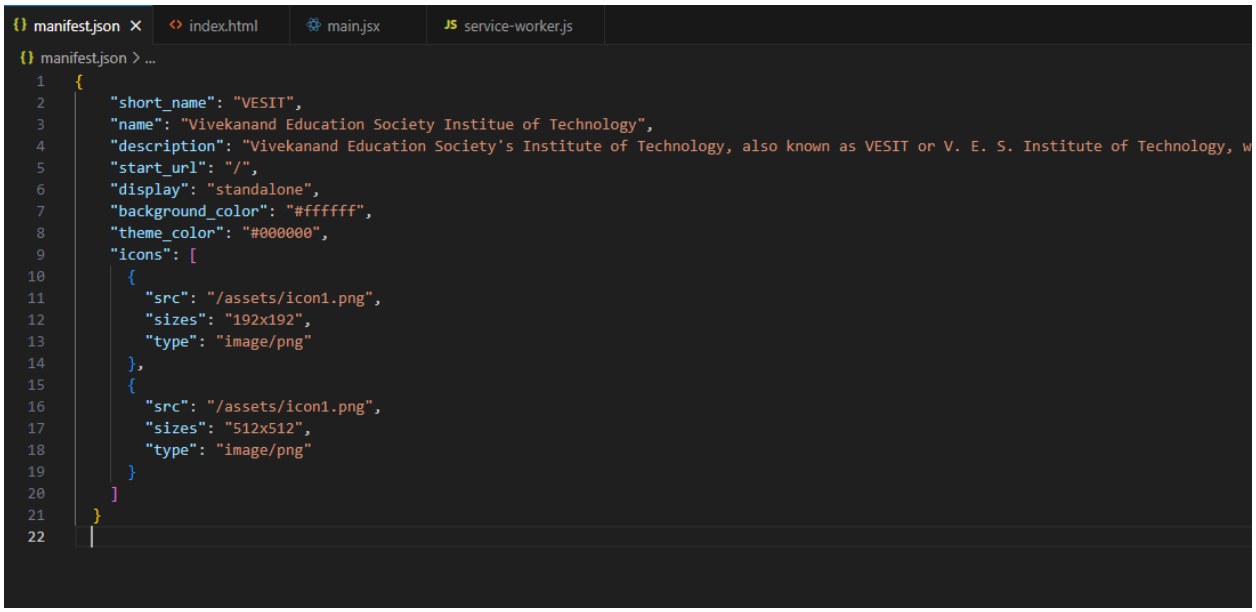
Advantages of Adding a PWA to Home Screen

1. **Fast Access** – Users can launch the app directly from their home screen.
2. **Offline Support** – PWAs can work offline using Service Workers.
3. **Push Notifications** – Engage users with real-time notifications.
4. **Better Performance** – Cached assets make loading faster.
5. **No App Store Required** – Users can install the app without visiting an app store.

Steps to “Add to HomeScreen” feature

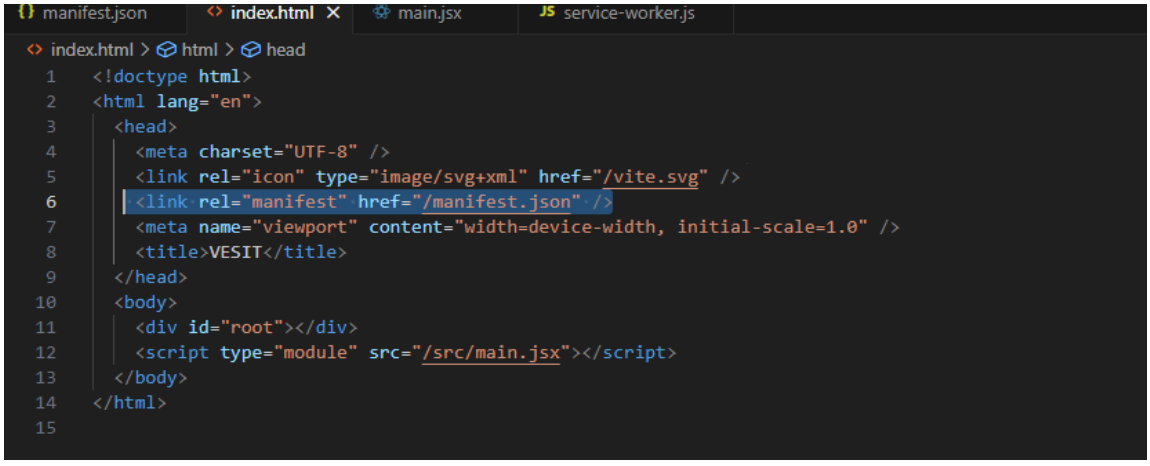
1) Create a Web App Manifest File

Create a manifest.json file in your project’s root directory.
Add metadata required for PWA installation



2) Link the Manifest in index.html

Open index.html and add this line inside the <head> tag



3) Register a Service Worker

Create a new file service-worker.js and add the following code to enable caching

```
var staticCacheName = "pwa-v1"; // Versioned cache name

self.addEventListener("install", function (e) {
  e.waitUntil(
    caches.open(staticCacheName).then(function (cache) {
      return cache.addAll([
        "/",
        "/index.html",
        "/style.css",
        "/script.js",
        "/logo.png",
        "/icon1.png",
        "/offline.html"  // Offline fallback page
      ]);
    })
  );
});

self.addEventListener("activate", function (event) {
    var cacheWhitelist = ["staticCacheName"];

    event.waitUntil(
      caches.keys().then(function (cacheNames) {
        return Promise.all(
          cacheNames.map(function (cacheName) {
            if (cacheWhitelist.indexOf(cacheName) === -1) {
              return caches.delete(cacheName);
            }
          })
        );
      })
    );
  });


self.addEventListener("fetch", function (event) {
```

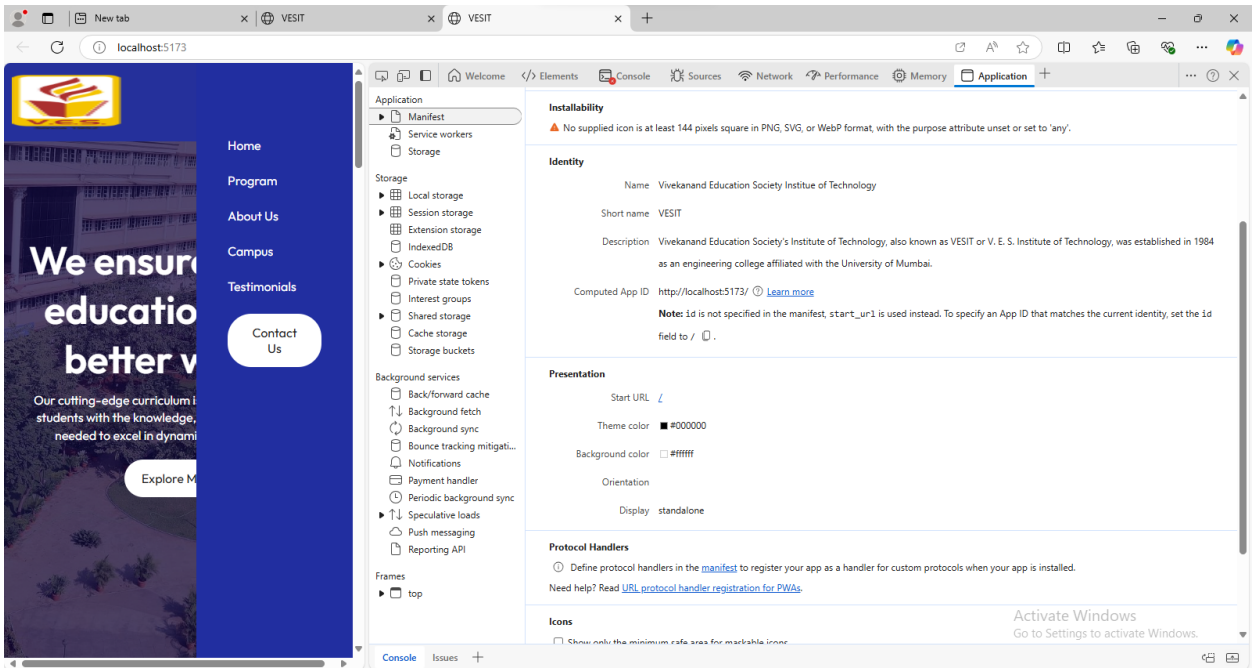
```
  event.respondWith(
    caches.match(event.request).then(function (response) {
      if (response) {
        return response;  // Serve from cache
      }

      return fetch(event.request).catch(function () {
        return caches.match("/offline.html");  // Serve offline fallback
      });
    })
  );
});

self.addEventListener('message', (event) => {
    if (event.data.action === 'skipWaiting') {
      self.skipWaiting();
    }
  });
```

4) <u>Register this service worker in index.html</u>

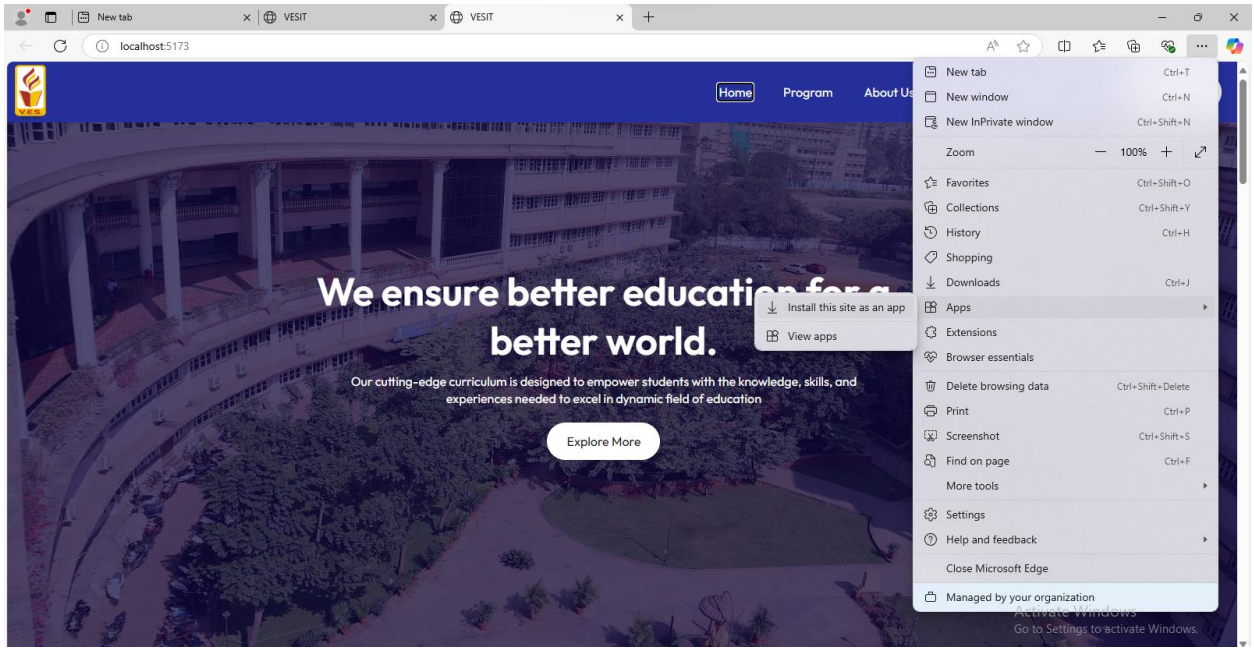
```
// src/index.js (or src/main.jsx)
if ('serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker
      .register('/service-worker.js')
      .then((registration) => {
        console.log('Service Worker registered with scope:', registration.scope);
    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

5) Test the PWA Installation

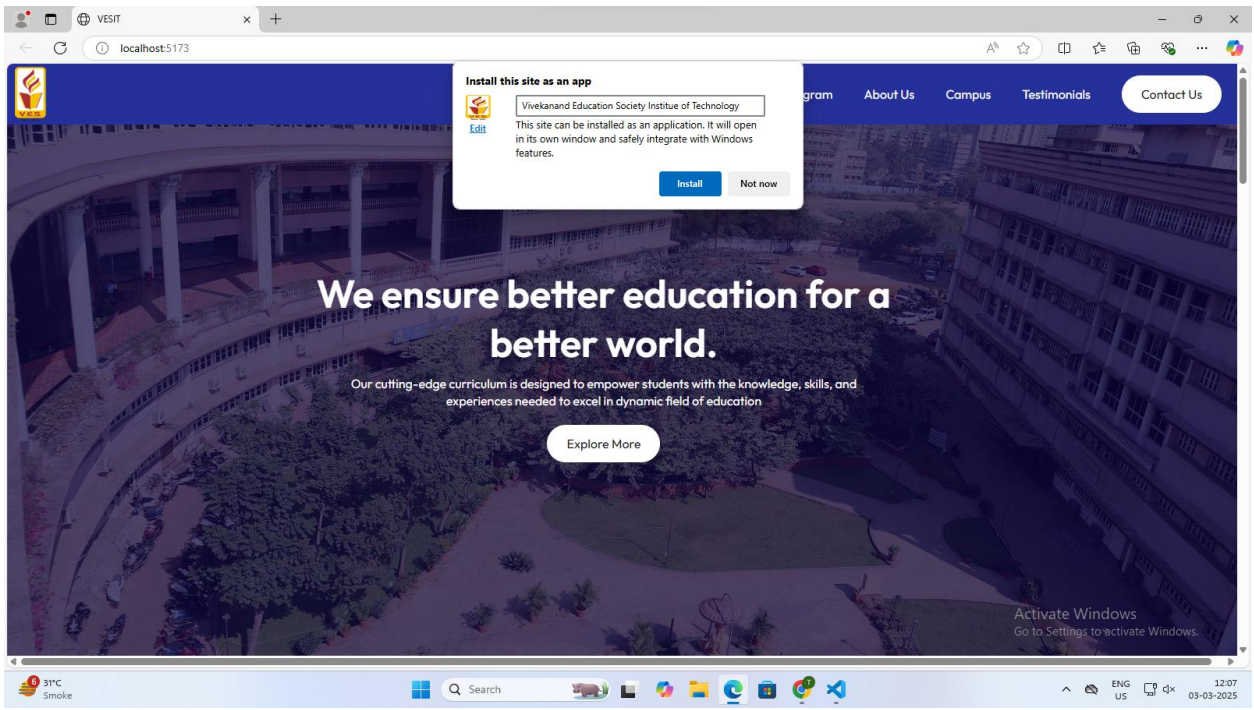
5.1) Open Google Chrome and go to Developer Tools > Application > Manifest to verify your manifest file.



5.2) Installing the PWA Using Microsoft Edge's "Add to Home Screen" Feature

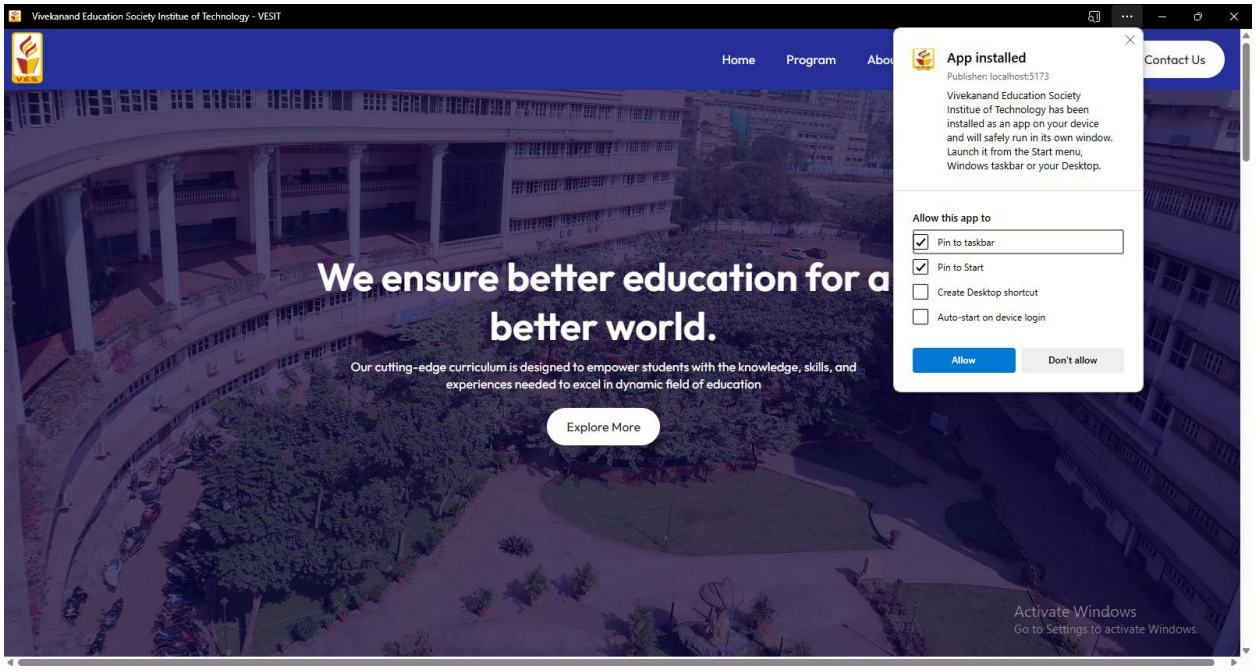


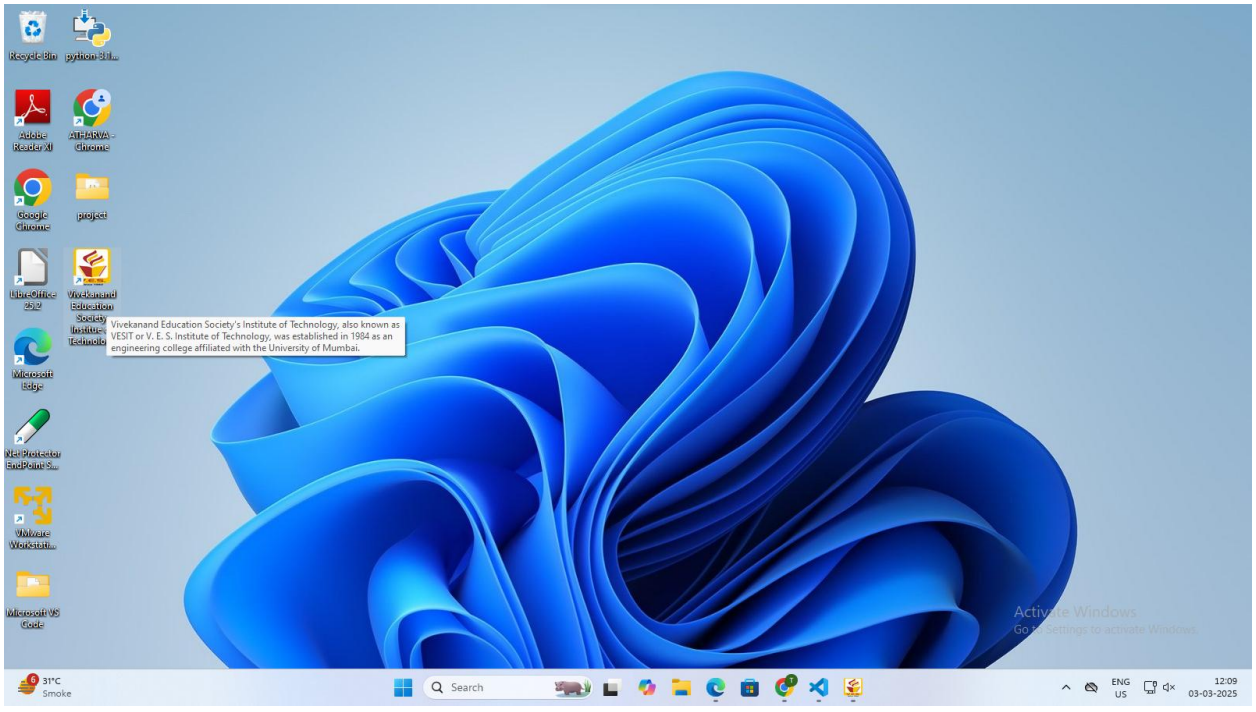
5.3) App icon along with name of the app will appear



5.4) PWA Successfully Installed

Users can choose to **pin the app to the taskbar, Start menu, or create a desktop shortcut** for easy access.





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.	18
Name	Tejas Dhondibhau Gunjal
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

EXPERIMENT NO: - 08

Name: - Tejas Gunjal **Class: -** D15A **Roll:No: -** 18

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

Theory: -

Service Worker

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

Things to note about Service Worker:

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

What can we do with Service Workers?

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

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

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

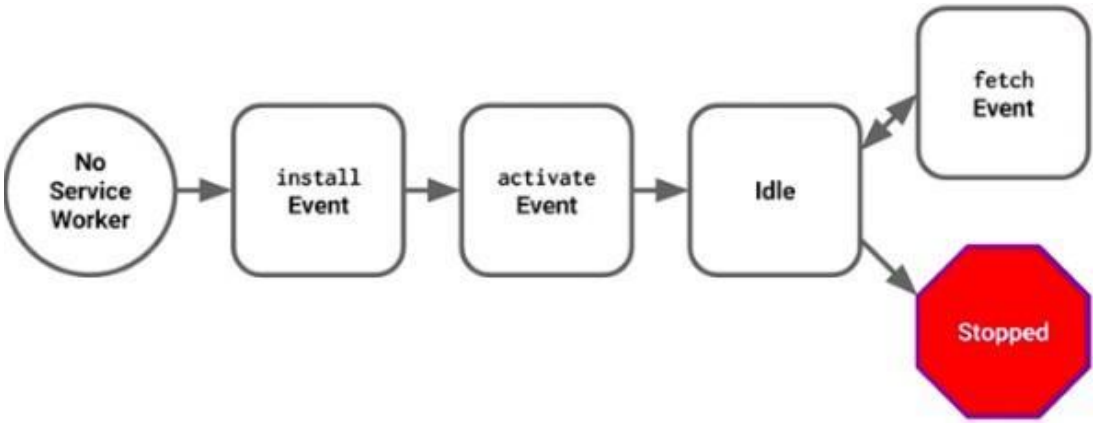
- You can **Continue**
  Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

What can't we do with Service Workers?

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

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

Service Worker Cycle



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

- Registration
- Installation
- Activation

Registration

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

main.js

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

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

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

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

main.js

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

});
```

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: -

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

service-worker.js

```
var staticCacheName = "pwa-v1"; // Versioned cache name

self.addEventListener("install", function (e) {
  e.waitUntil(
    caches.open(staticCacheName).then(function (cache) {
      return cache.addAll([
        "/",
        "/index.html",
        "/style.css",
        "/script.js",
        "/logo.png",
        "/icon1.png",
        "/offline.html"  // Offline fallback page
      ]);
    })
  );
});

self.addEventListener("activate", function (event) {
    var cacheWhitelist = [staticCacheName];  // Use staticCacheName directly

    event.waitUntil(
      caches.keys().then(function (cacheNames) {
        return Promise.all(
          cacheNames.map(function (cacheName) {
            if (cacheWhitelist.indexOf(cacheName) === -1) {
              return caches.delete(cacheName);  // Delete outdated caches
            }
          })
        );
      })
      .then(function() {
        // After the activation and cache cleaning, take control of the page immediately
        self.clients.claim();  // This ensures the new service worker takes control
      })
    );
});

self.addEventListener("fetch", function (event) {
  event.respondWith(
    caches.match(event.request).then(function (response) {
      if (response) {
        return response;  // Serve from cache
      }
```

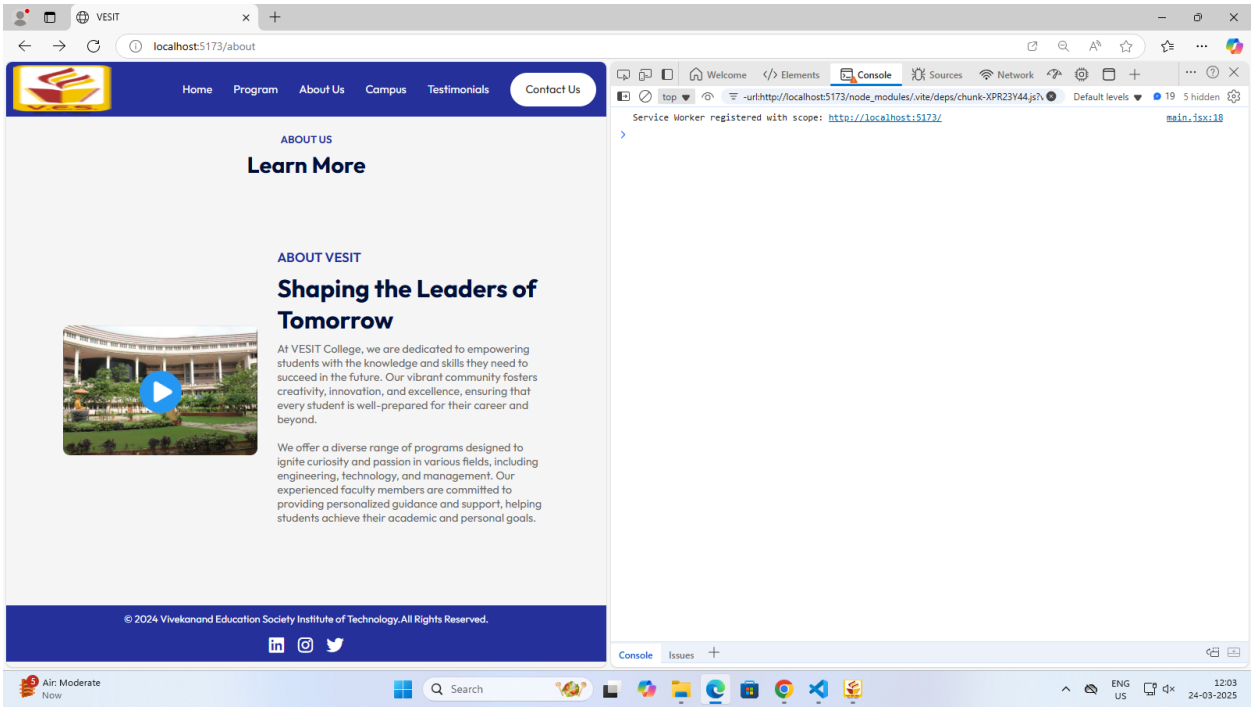
```
      return fetch(event.request).catch(function () {
        return caches.match("/offline.html");  // Serve offline fallback
      });
    })
  );
});

self.addEventListener('message', (event) => {
  if (event.data.action === 'skipWaiting') {
    self.skipWaiting();
  }
});
```

Index.html

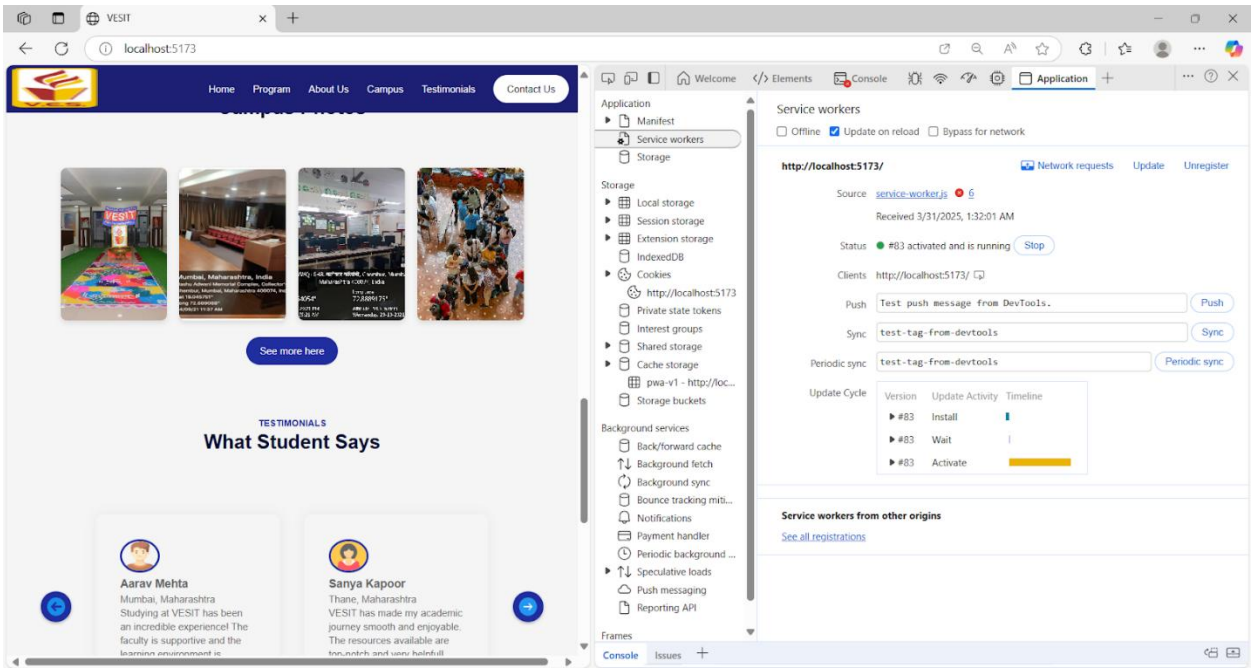
```
<!doctype html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <link rel="icon" type="image/svg+xml" href="/vite.svg" />

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

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>VESIT</title>

  </head>

  <body>

    <div id="root"></div>

    <script type="module" src="/src/main.jsx"></script>

  </body>

</html>
```

OUTPUT: -

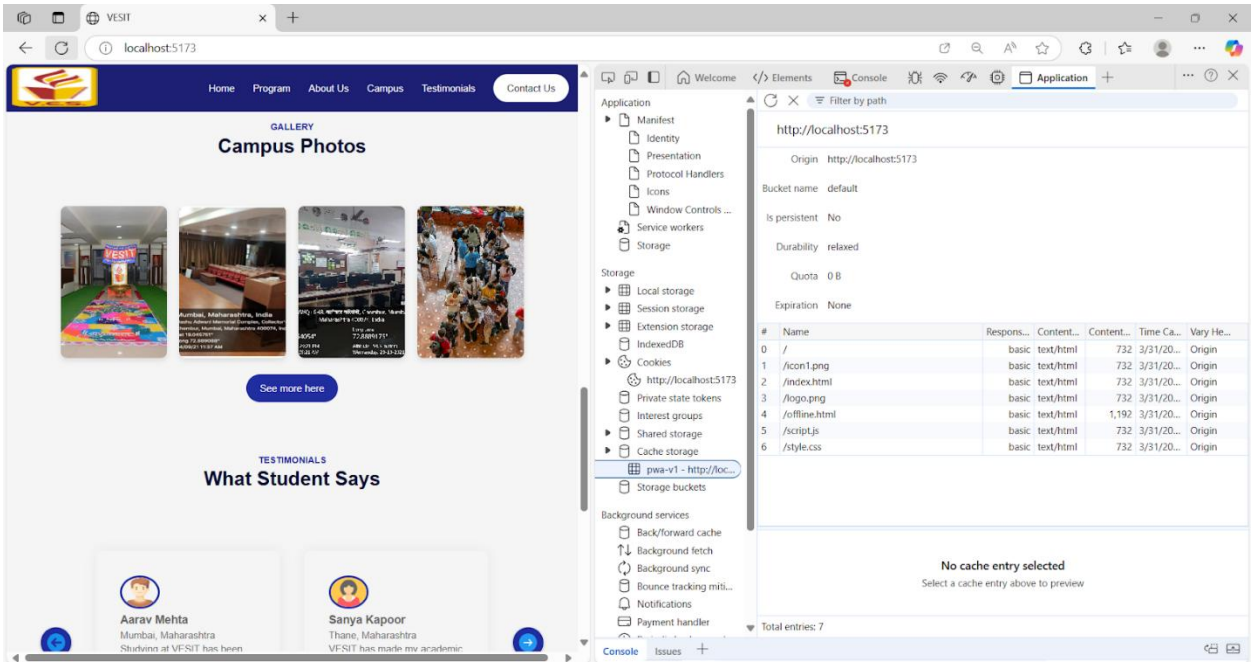
- Service Worker successfully registered



- Service Worker Activate status



- Cache storage of the application



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.	18
Name	Tejas Dhondibhau Gunjal
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

EXPERIMENT NO: - 09

Name:- Tejas Gunjal **Class:-** D15A **Roll:No: -** 18

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

Theory: -

Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

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

- **CacheFirst** - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.

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

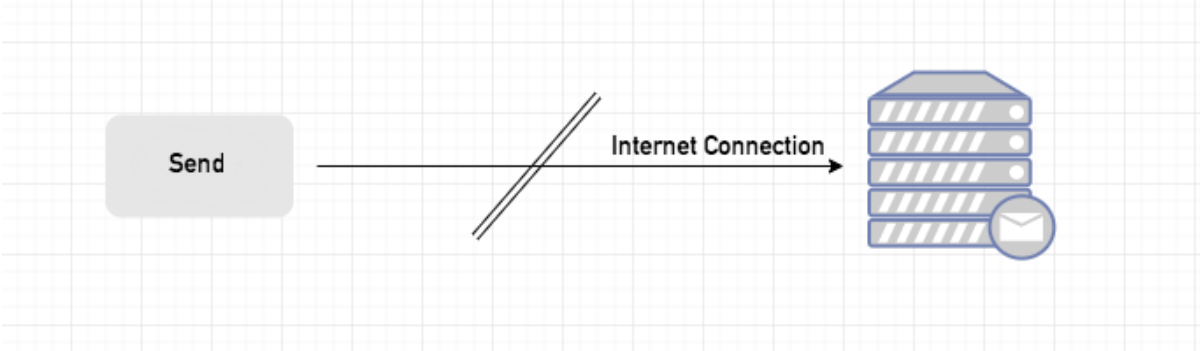
```
self.addEventListener("fetch", function (event) {
    const req = event.request;
    const url = new URL(req.url);

    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

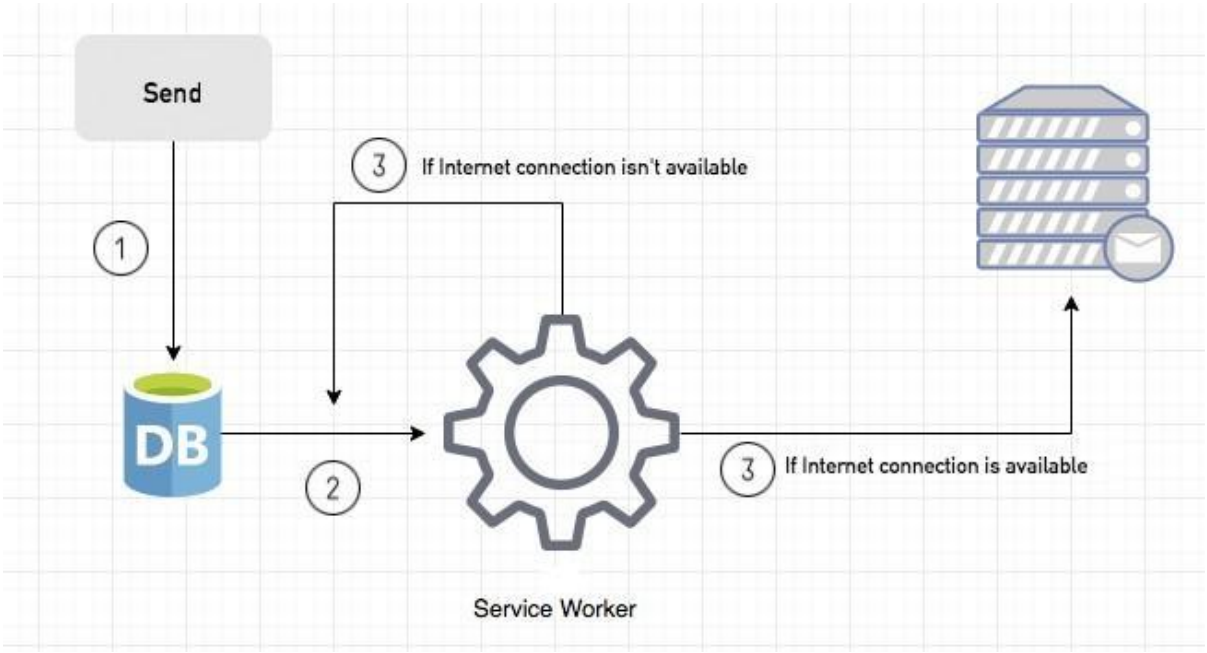
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

```
var staticCacheName = "pwa-v1"; // Versioned cache name

// Install event: Caching static assets
self.addEventListener("install", function (e) {
  console.log("Service Worker: Installing...");
  e.waitUntil(
    caches.open(staticCacheName).then(function (cache) {
      console.log("Service Worker: Caching files...");
      return cache.addAll([
        "/",
        "/index.html",
        "/styles.css",
        "/app.js",
        "/logo.png",
        "/icon.png",
        "/offline.html" // Offline fallback page
      ]);
    })
  );
});

// Activate event: Cleanup old caches
self.addEventListener("activate", function (event) {
  console.log("Service Worker: Activating...");
  var cacheWhitelist = [staticCacheName];

  event.waitUntil(
    caches.keys().then(function (cacheNames) {
      return Promise.all(
        cacheNames.map(function (cacheName) {
          if (!cacheWhitelist.includes(cacheName)) {
            console.log("Service Worker: Deleting old cache:", cacheName);
            return caches.delete(cacheName);
          }
        })
      );
    }).then(() => {
      console.log("Service Worker: Now controlling all clients.");
      self.clients.claim();
    })
  );
});
```

```
// Fetch event: Serve from cache, then network, then offline fallback
self.addEventListener("fetch", function (event) {
  console.log("Fetching:", event.request.url);
  event.respondWith(
    caches.match(event.request).then(function (response) {
      if (response) {
        console.log("Fetch successful (from cache):", event.request.url);
        return response; // Serve from cache
      }
      return fetch(event.request, { credentials: "include" })
        .then((networkResponse) => {
          console.log("Fetch successful (from network):", event.request.url);
          return networkResponse;
        })
        .catch(() => {
          console.log("Fetch failed, serving offline page.");
          return caches.match("/offline.html");
        });
    })
  );
});

// Push Notification Event
self.addEventListener("push", function (event) {
  console.log("Push event received.");
  const data = event.data ? event.data.json() : { message: "Default notification" };

  const options = {
    body: data.message,
    icon: "/logo.png"
  };

  event.waitUntil(
    self.registration.showNotification("VESIT", options).then(() => {
      console.log("Push Notification displayed successfully.");
    })
  );
});

// Background Sync Event
self.addEventListener("sync", function (event) {
  console.log("Sync event received:", event.tag);

  if (event.tag === "syncMessage") {
    event.waitUntil(
      (async () => {
```

```
      console.log("Processing sync event...");

      // Simulate an actual sync operation (like posting data to server)
      try {
        let response = await fetch("/sync-endpoint", { method: "POST" });
        console.log("Sync request sent:", response.status);
      } catch (error) {
        console.error("Sync request failed:", error);
      }

      // Show notification
      await self.registration.showNotification("VESIT", {
        body: "Sync successful!",
        icon: "/logo.png"
      });

      console.log("Sync event handled successfully.");
    })()
  );
 }
});

// Message event for skipWaiting
self.addEventListener("message", (event) => {
 if (event.data.action === "skipWaiting") {
   console.log("Skipping waiting phase...");
   self.skipWaiting();
 }
});
```

Main.jsx

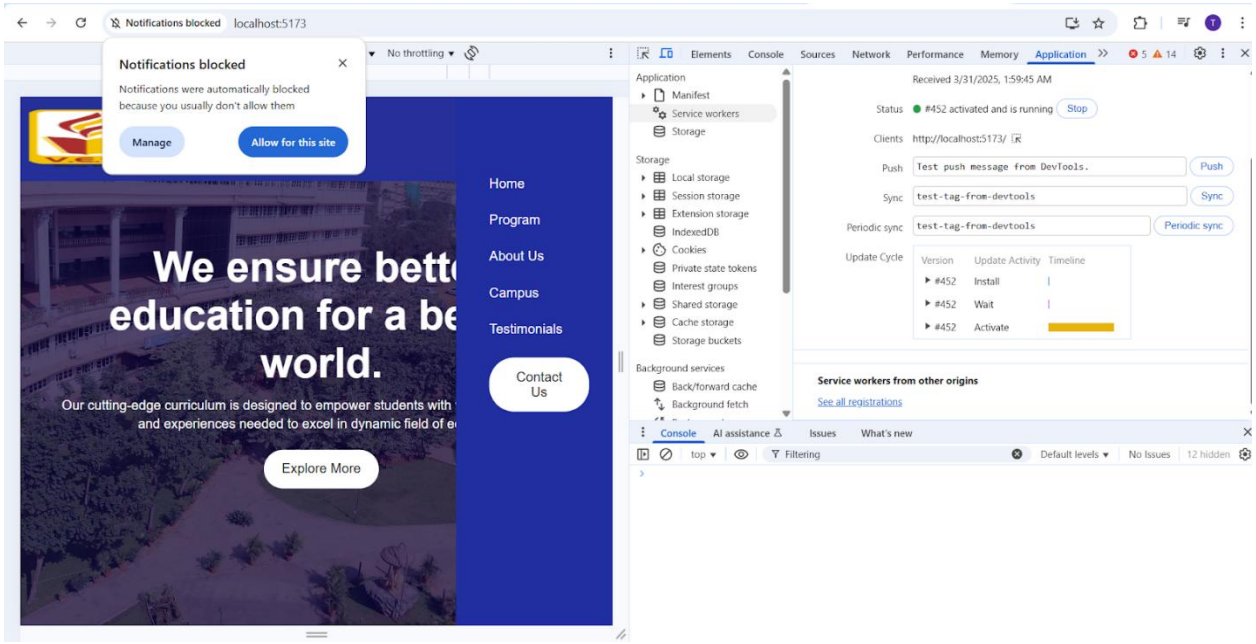
```
import { StrictMode } from "react";
import { createRoot } from "react-dom/client";
import App from "./App.jsx";
import "./index.css";

createRoot(document.getElementById("root")).render(
 <StrictMode>
   <App />
 </StrictMode>
);

// Register Service Worker
if ("serviceWorker" in navigator) {
 window.addEventListener("load", () => {
```

```
navigator.serviceWorker
  .register("/service-worker.js")
  .then((registration) => {
    console.log("Service Worker registered successfully with scope:", registration.scope);

    // Request Notification Permission
    Notification.requestPermission().then((permission) => {
      if (permission === "granted") {
        console.log("Notification permission granted.");
      } else {
        console.warn("Notification permission denied.");
      }
    });

    if ('serviceWorker' in navigator && 'SyncManager' in window) {
      navigator.serviceWorker.ready.then(registration => {
          return registration.sync.register('syncMessage')
              .then(() => console.log('Sync event registered'))
              .catch(err => console.error('Sync registration failed', err));
      });
  }


    // Push Notification Setup
    navigator.serviceWorker.ready.then((reg) => {
      reg.pushManager.subscribe({
        userVisibleOnly: true,
        applicationServerKey: "BD13Iv9g2jOfJ-
7JtItR7smZV7rk5DgFWfB43GLh7HgjrATPzJKDS8jCGYFNnaTutJM-
5oN885EjYB0k1CfOG2s" // Replace with actual VAPID key
      }).then((subscription) => {
        console.log("Push Notification Subscription successful:", subscription);
      }).catch((error) => {
        console.error("Push Notification Subscription failed:", error);
      });
    });

    // Background Sync Setup
    navigator.serviceWorker.ready.then((reg) => {
      if ("sync" in reg) {
        reg.sync.register("syncMessage").then(() => {
          console.log("Background sync registered successfully.");
        }).catch((error) => {
          console.error("Background sync registration failed:", error);
        });
      } else {
```

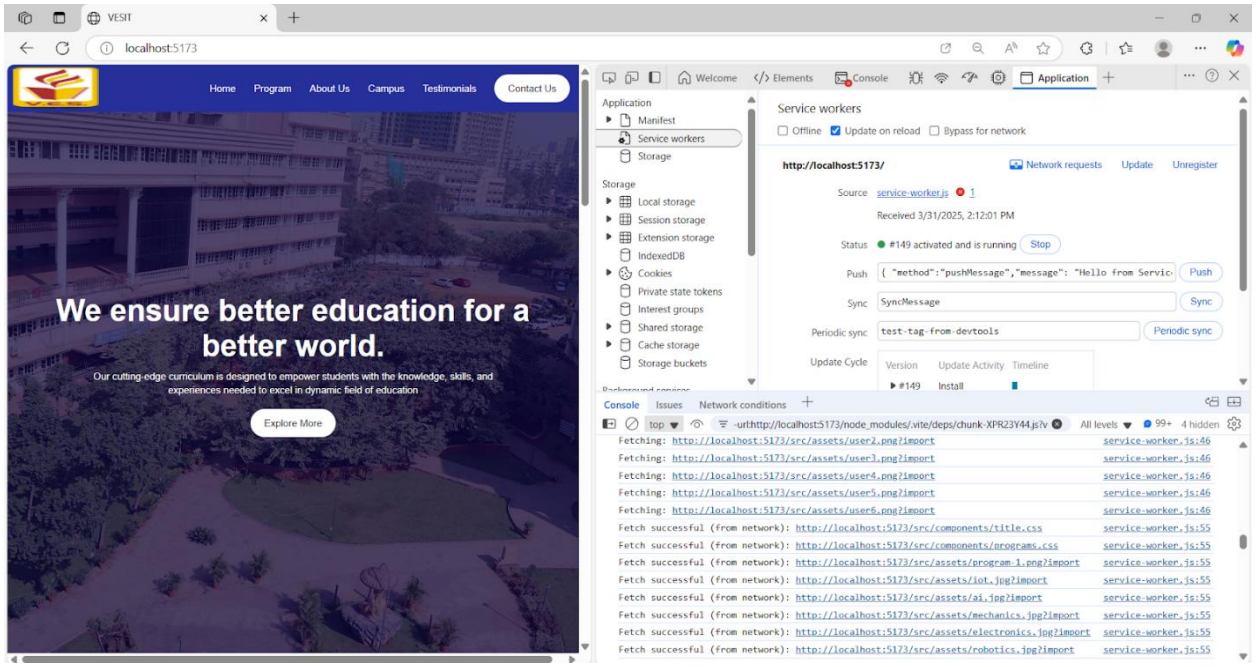
```
        console.warn("SyncManager is not supported.");
      }
    });

  })
  .catch((error) => {
    console.error("Service Worker registration failed:", error);
  });
});
}
```

OUTPUT: -

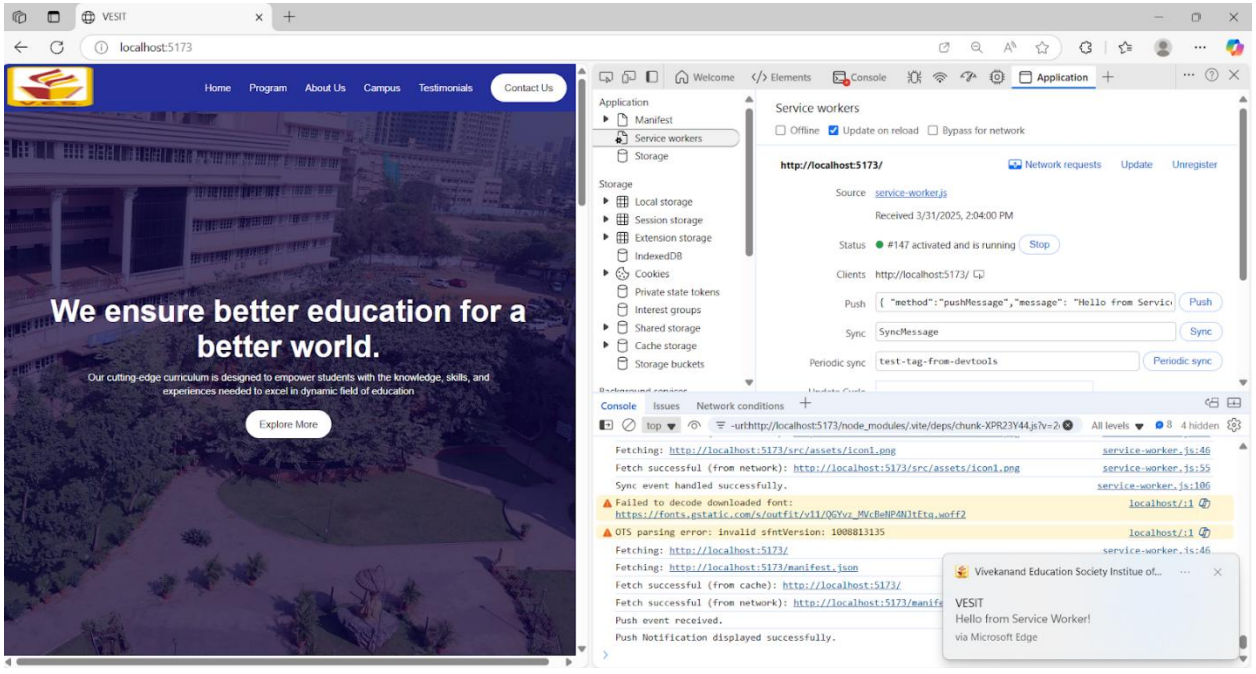
- **Notification Permission**



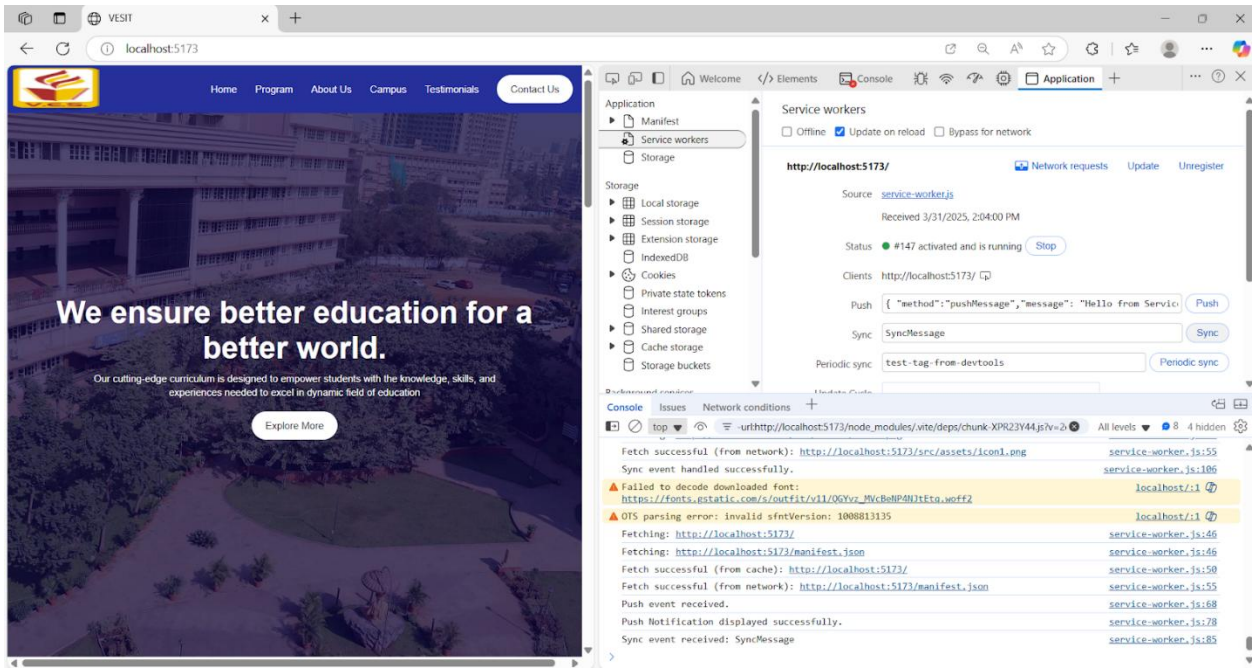
➢ Fetch Event

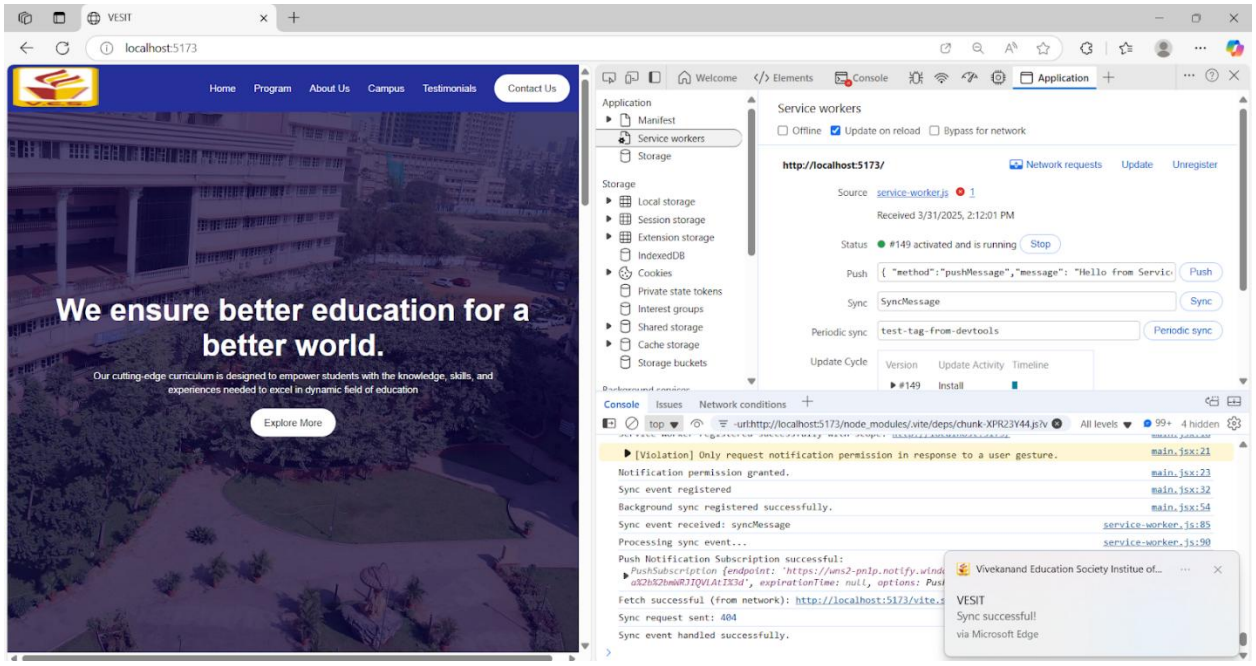


➢ Push Event



➢ Sync Event


VESIT
localhost:5173
Home
Program
About Us
Campus
Testimonials
Contact Us
We ensure better education for a better world.
Our cutting-edge curriculum is designed to empower students with the knowledge, skills, and experiences needed to excel in dynamic field of education
Explore More
Application
Service workers
Offline
Update on reload
Bypass for network
http://localhost:5173/
Network requests
Update
Unregister
Source service-worker.js
Received 3/31/2025, 2:04:00 PM
Status #147 activated and is running
Stop
Clients http://localhost:5173/
Push
{ "method":"pushMessage","message": "Hello from Servic
Push
Sync
SyncMessage
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Console
Issues
Network conditions
Fetch successful (from network): http://localhost:5173/src/assets/icon1.png service-worker.js:55
Sync event handled successfully. service-worker.js:106
Failed to decode downloaded font: localhost/:1
OTS parsing error: invalid sfntVersion: 1008813135 localhost/:1
Fetching: http://localhost:5173/ service-worker.js:46
Fetching: http://localhost:5173/manifest.json service-worker.js:46
Fetch successful (from cache): http://localhost:5173/ service-worker.js:50
Fetch successful (from network): http://localhost:5173/manifest.json service-worker.js:55
Push event received. service-worker.js:68
Push Notification displayed successfully. service-worker.js:78
Sync event received: SyncMessage service-worker.js:85


VESIT
localhost:5173
Home
Program
About Us
Campus
Testimonials
Contact Us
We ensure better education for a better world.
Our cutting-edge curriculum is designed to empower students with the knowledge, skills, and experiences needed to excel in dynamic field of education
Explore More
Application
Service workers
Offline
Update on reload
Bypass for network
http://localhost:5173/
Network requests
Update
Unregister
Source service-worker.js 1
Received 3/31/2025, 2:12:01 PM
Status #149 activated and is running
Stop
Push
{ "method":"pushMessage","message": "Hello from Servic
Push
Sync
SyncMessage
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#149
Install
Console
Issues
Network conditions
[Violation] Only request notification permission in response to a user gesture. main.jsx:21
Notification permission granted. main.jsx:23
Sync event registered main.jsx:32
Background sync registered successfully. main.jsx:54
Sync event received: syncMessage service-worker.js:85
Processing sync event... service-worker.js:90
Push Notification Subscription successful:
Fetch successful (from network): http://localhost:5173/vite.
Sync request sent: 404
Sync event handled successfully.
Vivekanand Education Society Institue of...
VESIT
Sync successful!
via Microsoft Edge

MAD & PWA Lab

Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	18
Name	Tejas Dhondibhau Gunjal
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

EXPERIMENT NO: - 10

Name:- Tejas Gunjal **Class:-** D15A **Roll:No: -** 18

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

Theory: -

GitHub Pages

GitHub Pages is a free hosting service that allows users to publish public webpages directly from a GitHub repository. It is particularly useful for static websites, project documentation, and blogs. It seamlessly integrates with Git version control, ensuring automatic updates with every commit. With built-in support for Jekyll, custom domains, and HTTPS (for GitHub subdomains), it provides an easy and efficient way to deploy static sites.

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 , watch it update in real-time.

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

Pros

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

Cons

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

Link to our Github Repository: -

https://github.com/tejasgunjal021/college_pwa_website

Link to our Hosted Website : -

https://tejasgunjal021.github.io/college_pwa_website/

Method Followed [gh- pages] : -

We deployed a **React-based PWA** to **GitHub Pages** using the gh-pages package. The process involved:

1. **Setting up GitHub Pages** – Configuring the repository to support deployment.
2. **Adding homepage and scripts in package.json** – Ensuring proper routing support.
3. **Building the React App** – Creating the production-ready files.
4. **Deploying using gh-pages** – Pushing the build folder to the gh-pages branch.
5. **Verifying the Deployment** – Ensuring the website works correctly on GitHub Pages.

Github Screenshots: -

- **Pushing my PWA application into the github repository**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    powershell

PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master> git init
Initialized empty Git repository in C:/Users/TEJAS/Desktop/ip-exp6-master/ip-exp6-master/.git/
PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master> git add .
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'eslint.config.js', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/App.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/About.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Campus.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Contact.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Footer.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Hero.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Navbar.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Programcard.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Programs.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Testimonials.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Title.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/VideoPlayer.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/about.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/campus.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/contact.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/footer.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/hero.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/navbar.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/programs.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/testimonials.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/title.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/videoplayer.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/index.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/main.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'vite.config.js', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master> git commit -m "Initial Commit"
[master (root-commit) 24ee19e] Initial Commit
 63 files changed, 5926 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 README.md
 create mode 100644 eslint.config.js
 create mode 100644 index.html
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    powershell

PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master> git remote add origin https://github.com/tejasgunjal021/college_pwa_website.git
PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master> git branch -M main
PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master> git push -u origin main
Enumerating objects: 67, done.
Counting objects: 100% (67/67), done.
Delta compression using up to 8 threads
Compressing objects: 100% (67/67), done.
Writing objects: 100% (67/67), 80.32 MiB | 3.83 MiB/s, done.
Total 67 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/tejasgunjal021/college_pwa_website.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master>
```

- **Installing gh-pages Package** – Setting up the deployment tool.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    powershell

PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master> npm install gh-pages --save-dev
>>

added 44 packages, and audited 308 packages in 4s

116 packages are looking for funding
  run `npm fund` for details

3 moderate severity vulnerabilities

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
PS C:\Users\TEJAS\Desktop\ip-exp6-master\ip-exp6-master>
```

- **Adding homepage and Deployment scripts in package.json**

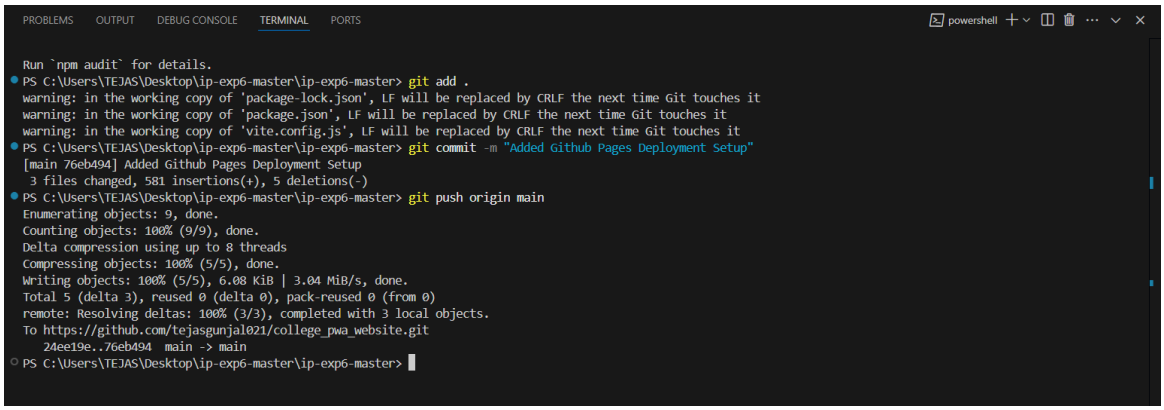
```
{
 "name": "college-website",
 "homepage": "https://tejasgunjal021.github.io/college_pwa_website",

 "private": true,
 "version": "0.0.0",
 "type": "module",
 "scripts": {
  "dev": "vite",
  "build": "vite build",
  "lint": "eslint .",
  "preview": "vite preview",
  "predeploy": "vite build",
  "deploy": "gh-pages -d dist"
 },
 "dependencies": {
  "@fortawesome/fontawesome-free": "^6.6.0",
  "react": "^18.3.1",
  "react-dom": "^18.3.1",
  "react-router-dom": "^6.27.0"
 },
 "devDependencies": {
  "@eslint/js": "^9.11.1",
  "@types/react": "^18.3.10",
  "@types/react-dom": "^18.3.0",
  "@vitejs/plugin-react": "^4.3.2",
  "eslint": "^9.11.1",
  "eslint-plugin-react": "^7.37.0",
  "eslint-plugin-react-hooks": "^5.1.0-rc.0",
  "eslint-plugin-react-refresh": "^0.4.12",
  "gh-pages": "^6.3.0",
  "globals": "^15.9.0",
  "vite": "^5.4.8"
 }
}
```

- **Update vite.config.js**

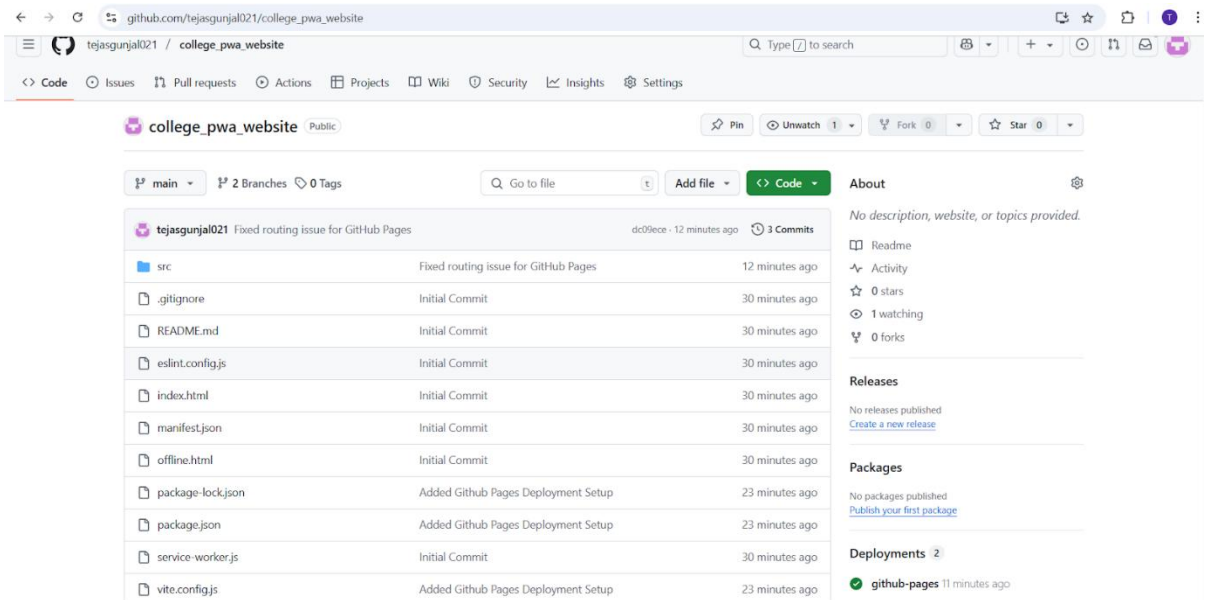
  Since Vite serves the app from / by default, you must configure it to serve from the correct subdirectory.

```
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react";

export default defineConfig({
  plugins: [react()],
  base: "/college_pwa_website/",  // Add this line
});
```

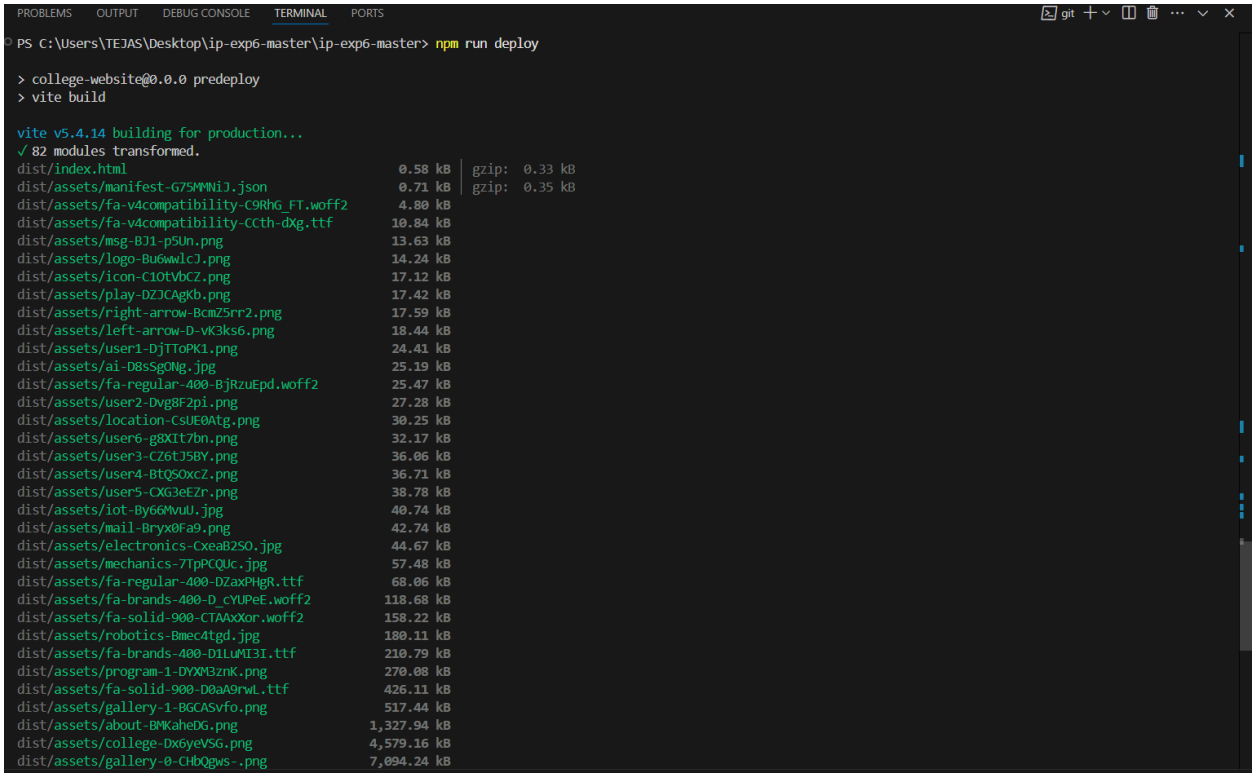
- **Updating the changes in the repository**



- **Code pushed to the repository**

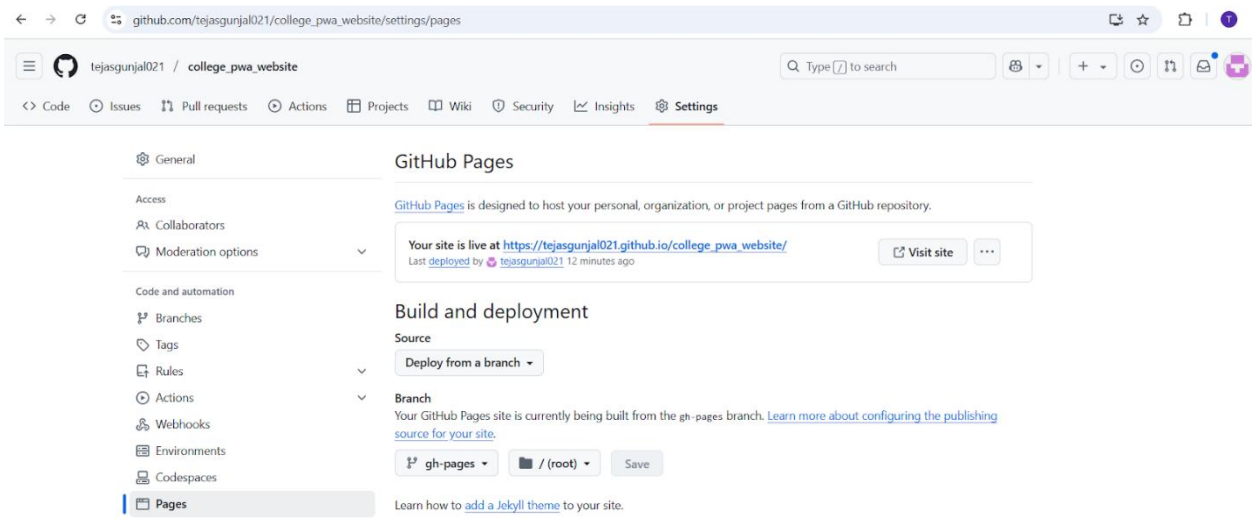


➢ Deploy to github pages

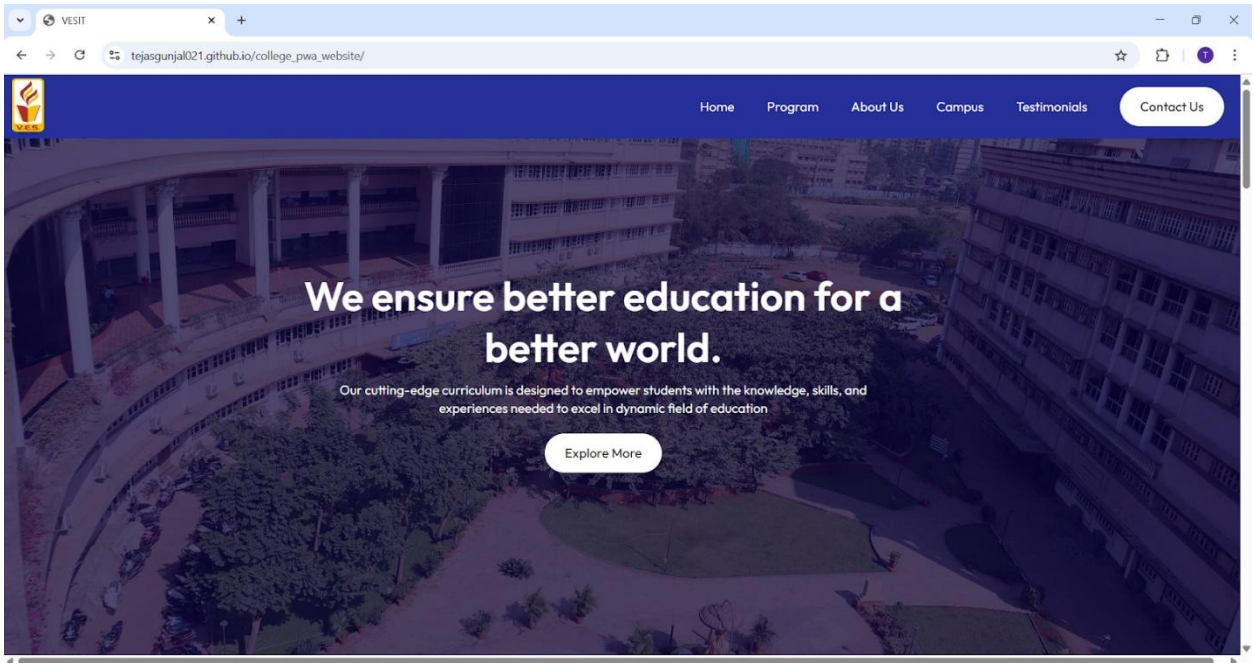


➢ Enable Github Pages

1. Go to your repository on GitHub.
2. Click on **Settings** > **Pages**.
3. In that you will see the github hosted link for your application.



- **Hosted Website: -** https://tejasgunjal021.github.io/college_pwa_website/



MAD & PWA Lab

Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	18
Name	Tejas Dhondibhau Gunjal
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

EXPERIMENT NO: - 11

Name:- Tejas Gunjal **Class:-** D15A **Roll:No: -** 18

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

Theory: -

Google Lighthouse is an open-source tool designed to evaluate and optimize web applications based on multiple key parameters, including performance, accessibility, Progressive Web App (PWA) implementation, and best practices. It provides a detailed automated audit that helps developers improve their websites efficiently. Unlike traditional manual audits that can take days or weeks, Lighthouse generates insights within minutes with minimal setup.

One of Lighthouse's biggest advantages is its ease of use—simply run it on a webpage or provide a URL, and it will generate a comprehensive report. Lighthouse supports audits for both desktop and mobile versions of a website, ensuring an optimal user experience across different devices.

Key Features and Audit Metrics

1. Performance

This metric evaluates how efficiently a webpage loads and becomes interactive. It considers several factors, including:

- Page Load Speed – Measures how quickly content appears to users.
- First Contentful Paint (FCP) – Time taken for the first visible content to load.
- Largest Contentful Paint (LCP) – Measures how long the largest visible content takes to fully render.
- Cumulative Layout Shift (CLS) – Ensures content does not shift unexpectedly, improving user experience.
- Time to Interactive (TTI) – The time taken for the webpage to become fully functional.

Lighthouse assigns a performance score from 0 to 100, where:

- 100 → Top 2% of websites (98th percentile)
- 50 → Around the 75th percentile
- Lower scores → Indicate areas needing optimization

2. Progressive Web App (PWA) Score

With the increasing adoption of PWAs, web applications now strive to deliver an app-like experience. Lighthouse evaluates a website's PWA readiness based on Google's Baseline PWA Checklist, which includes:

- Service Worker Implementation – Enables offline functionality and background synchronization.
- App Manifest Compliance – Provides metadata for mobile app-like integration.
- Viewport Configuration – Ensures responsiveness across different screen sizes.
- Performance in Script-Disabled Environments – Verifies that the page functions properly even if JavaScript is disabled.

A high PWA score indicates that the application meets essential criteria for speed, reliability, and mobile usability.

3. Accessibility

This metric determines how well a website supports users with **visual, cognitive, or physical disabilities**. Lighthouse evaluates accessibility based on:

- **ARIA Attributes** – Enhances screen reader support (e.g., aria-required).
- **Text Alternatives for Media** – Ensures images, audio, and video content are accessible.
- **Semantic HTML Usage** – Encourages proper use of elements like <section>, <article>, and <button> for better screen-reader compatibility.

Unlike other metrics, **accessibility follows a pass/fail approach**—missing key features can **significantly impact the overall score**. Improving accessibility ensures **inclusivity for all users**.

4. Best Practices

Lighthouse also assesses whether the website follows **modern web development best practices**, including:

- **Use of HTTPS** – Ensures secure data transmission.
- **Avoiding Deprecated Code** – Prevents the use of outdated elements, directives, and libraries.
- **Secure Password Inputs** – Disables paste-into fields to reduce security risks.
- **User Security Alerts** – Provides notifications for **geo-location access and cookie usage**.

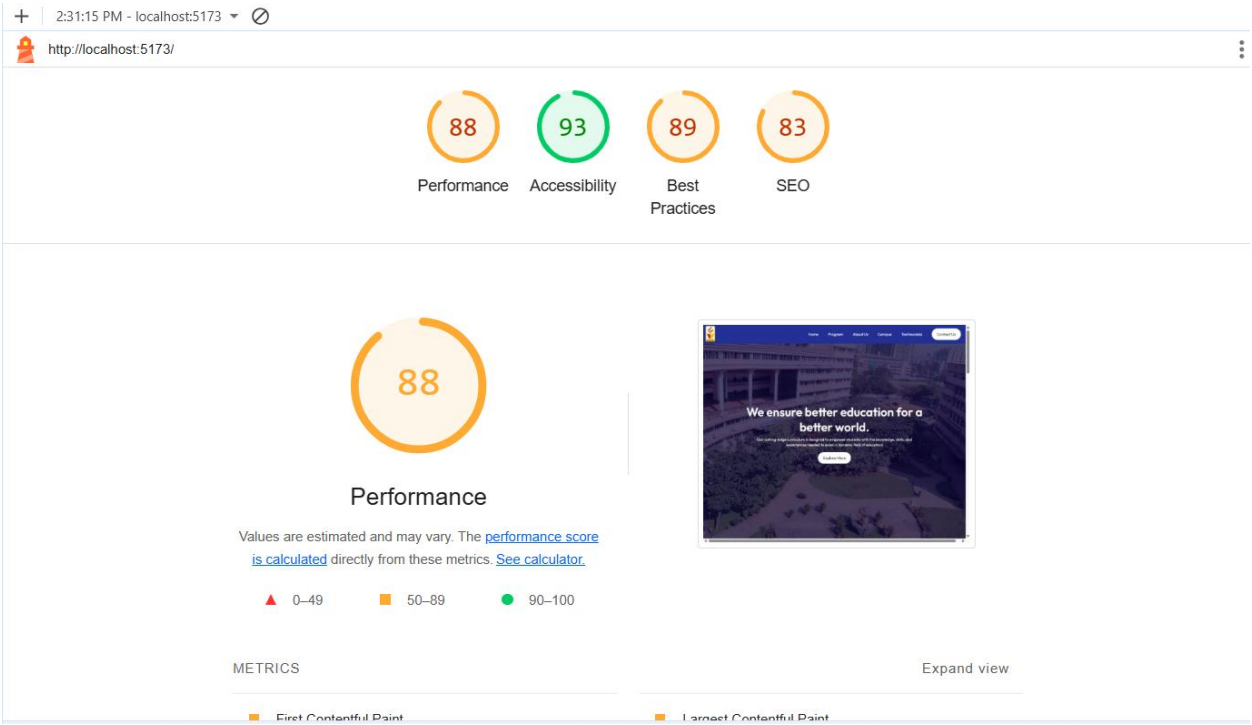
A high **Best Practices score** means the website meets industry standards, leading to better **security, maintainability, and overall performance**.

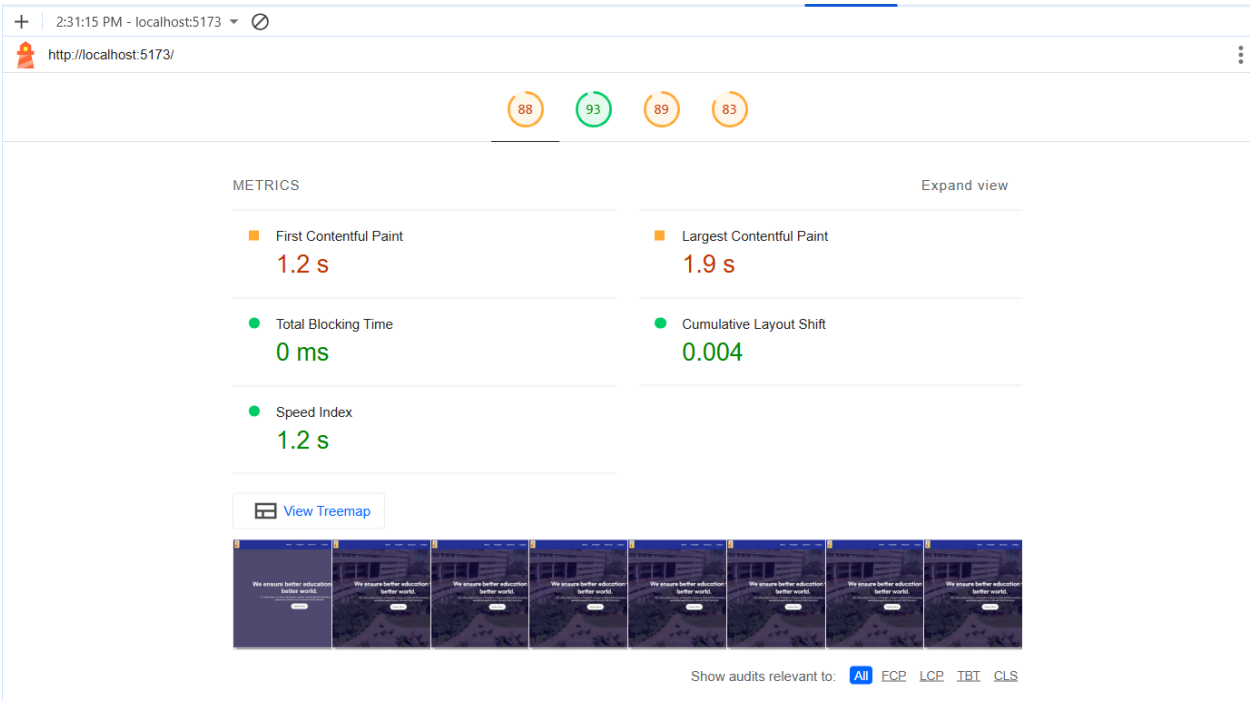
Manifest.json

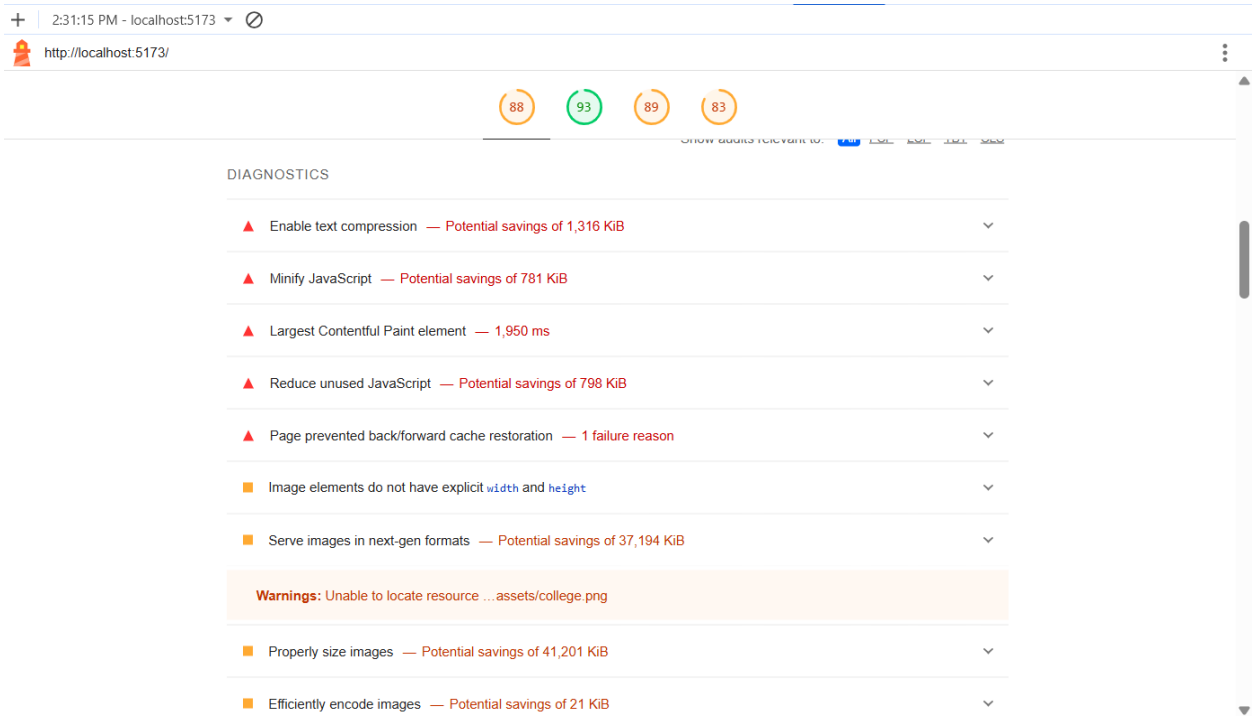
```
{
  "short_name": "VESIT",
  "name": "Vivekanand Education Society Institue of Technology",
  "description": "Vivekanand Education Society's Institute of Technology, also known as VESIT or V. E. S. Institute of Technology, was established in 1984 as an engineering college affiliated with the University of Mumbai.",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#000000",
  "icons": [
    {
      "src": "/src/assets/icon1.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/src/assets/icon1.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

Output: -

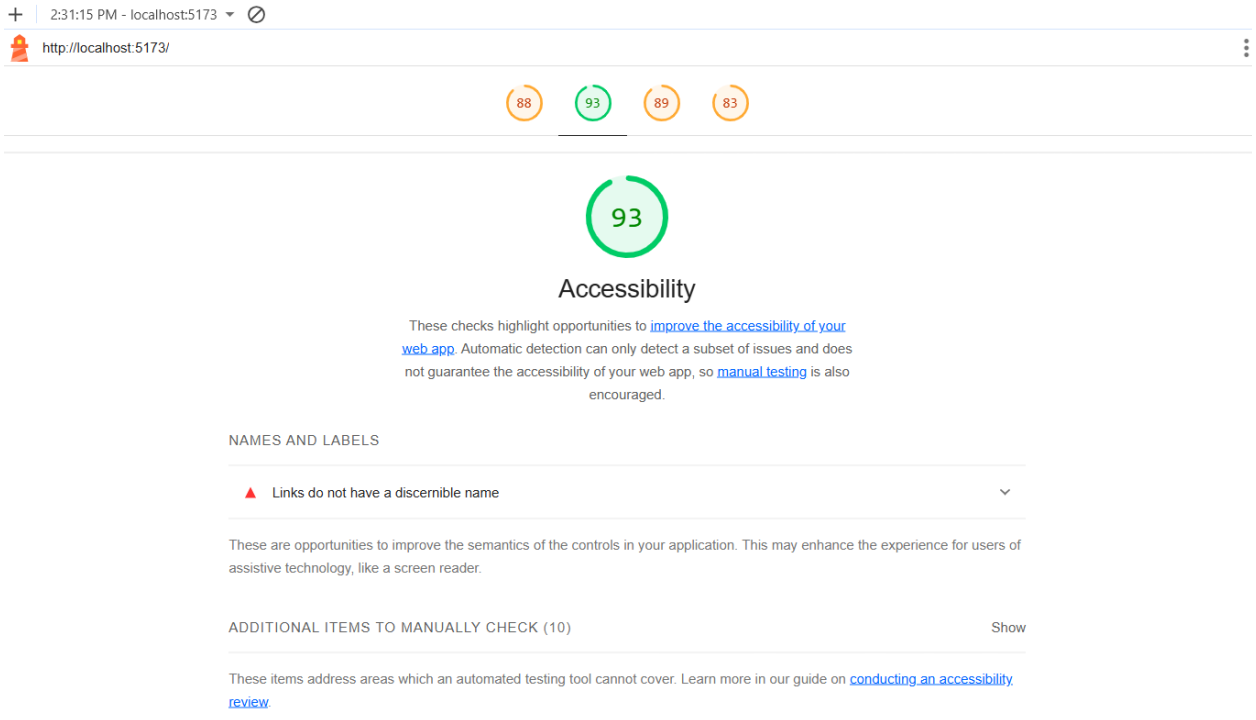
a) Performance



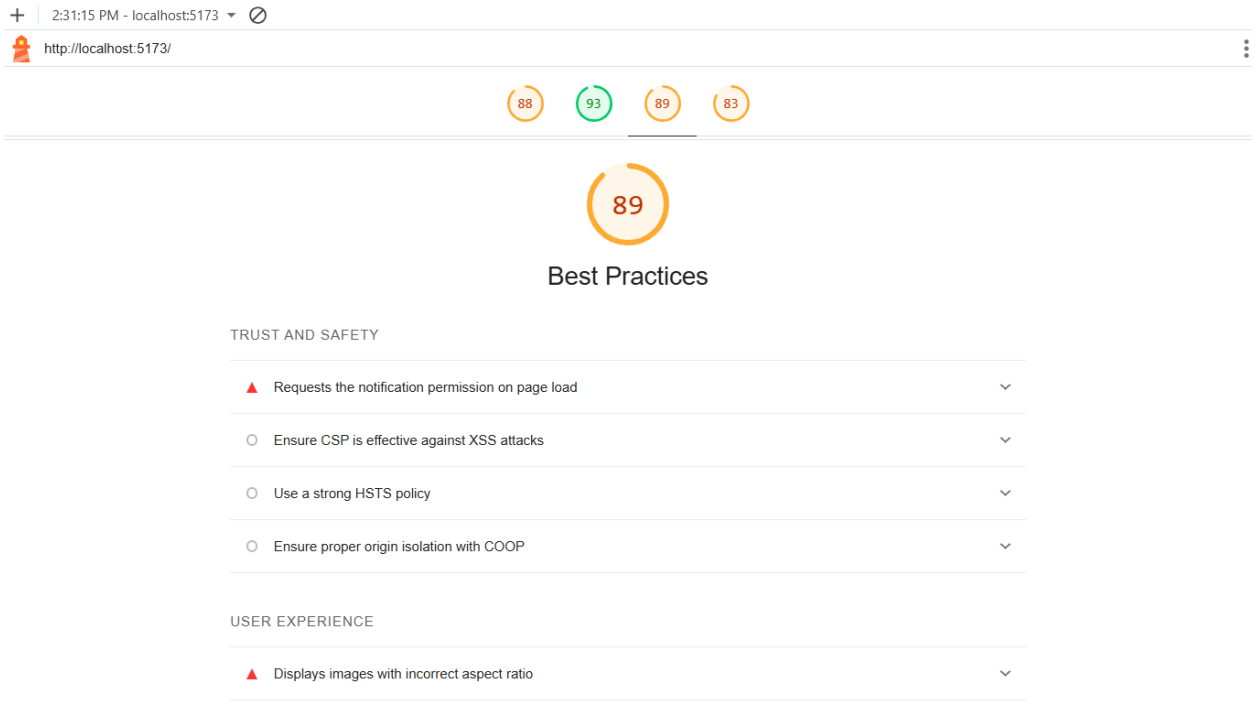




b) Accessibility



c) Best Practices



d) SEO [Search Engine Optimization]