Name : Naikwadi Yash Shivdas

MPL Idea Summary

GreenFund Connect: A Renewable Energy Crowdfunding Platform

Problem Statement:

Many people want to set up renewable energy projects like solar panels and wind turbines, but they do not have enough money. At the same time, investors who want to support green energy do not have a simple way to find such projects. Due to this, many good ideas for clean energy never get started.

Solution:

GreenFund Connect is a crowdfunding platform that helps people raise money for renewable energy projects. It will connect project creators with investors who are interested in funding green initiatives. The platform will:

- Allow project creators to share their ideas and raise funds from investors.
- Give investors details about projects, including how they will help the environment.
- Involve the community in supporting clean energy projects.

Example:

Deepak lives in a small town and wants to install a solar power grid, but he does not have enough money. He creates a project on GreenFund Connect, where investors like Priya can see his idea. Priya and others contribute money after reading about the benefits of the project. Once Deepak raises enough funds, he sets up the solar power grid, which provides clean energy to his town.

Impact:

This platform will help bring more renewable energy projects to life by making it easy to get funding. It will support clean energy, reduce pollution, and benefit communities by providing sustainable solutions.

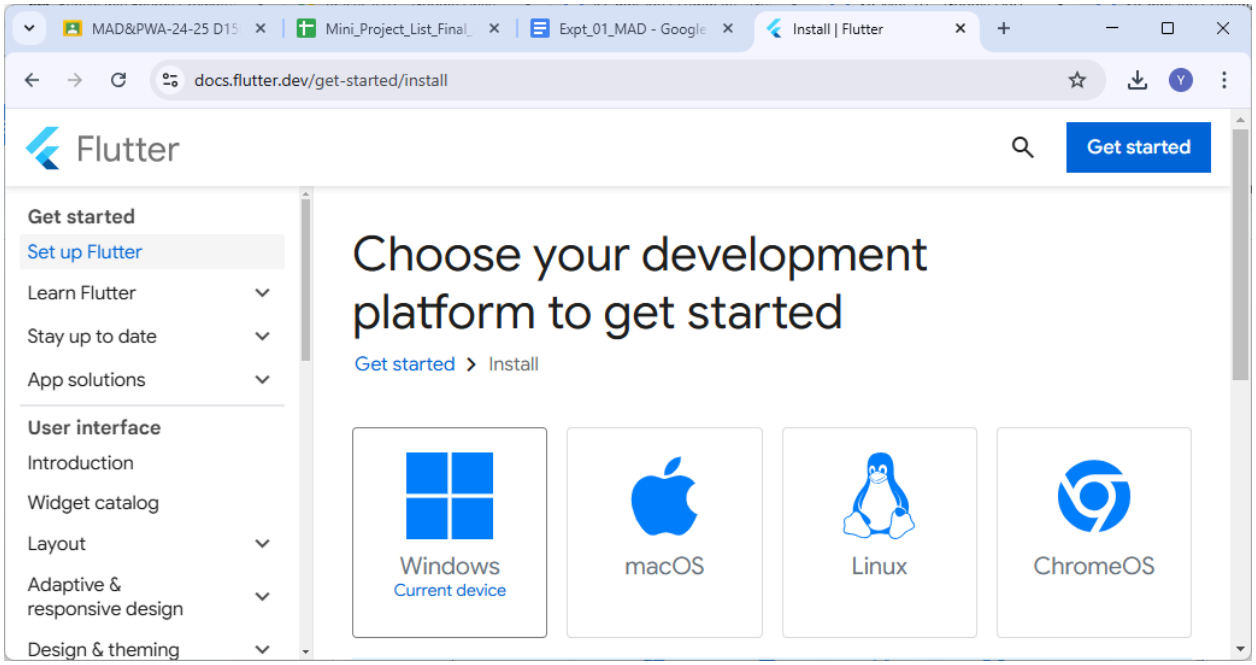
Name : Naikwadi Yash Shivdas

MPL Practical 01

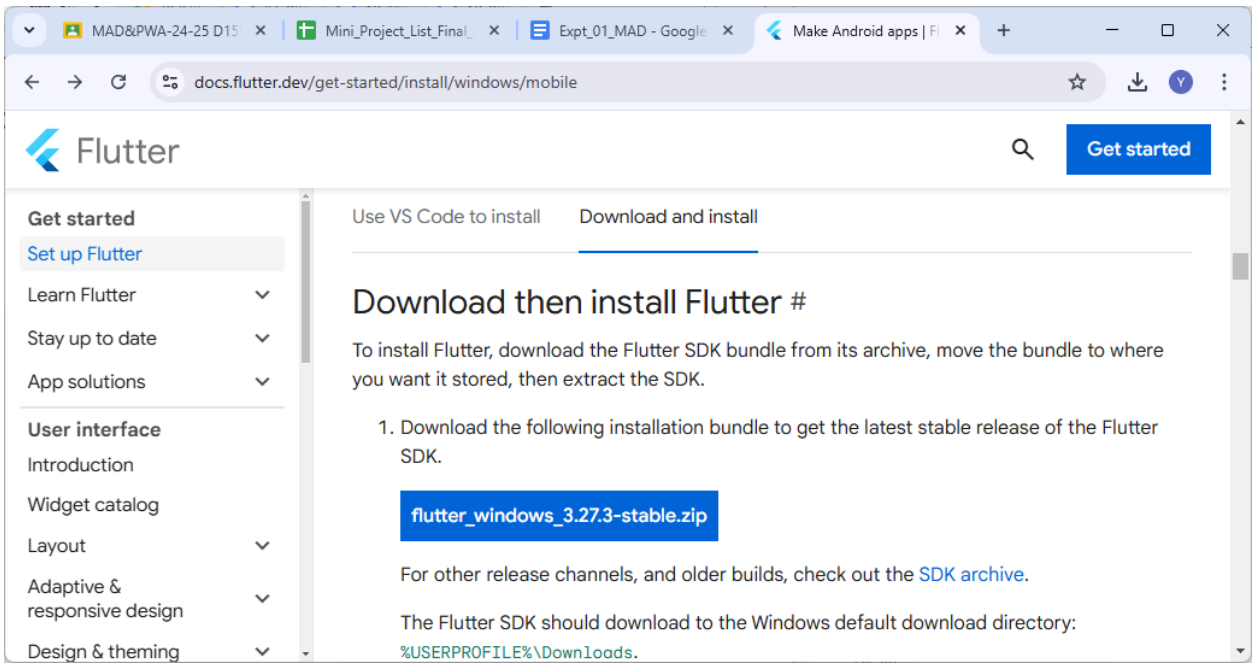
Aim: Installation and Configuration of Flutter Environment.

Install the Flutter SDK

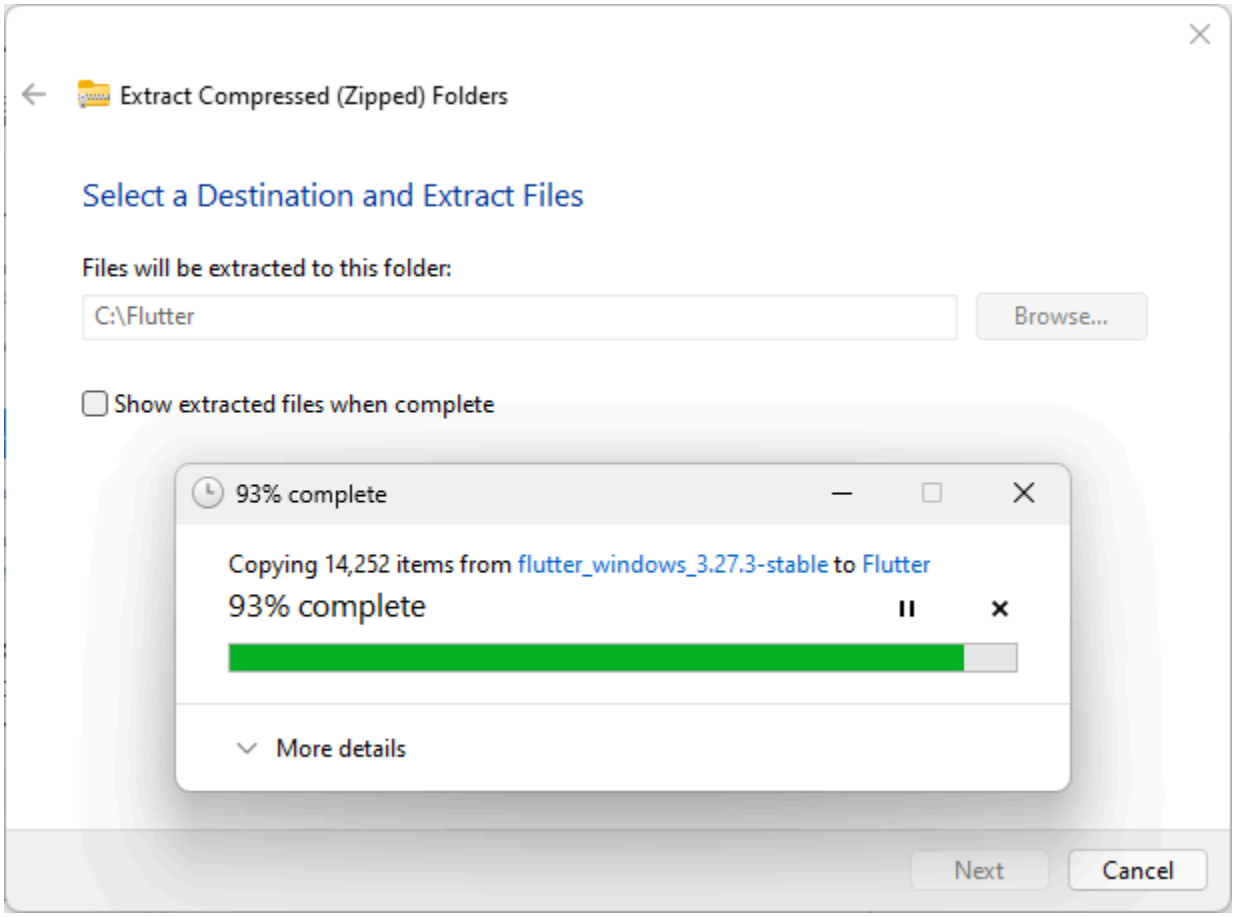
Step 1: Download the installation bundle of the Flutter Software Development Kit for windows. To download Flutter SDK, Go to its official website https://docs.flutter.dev/get-started/install , you will get the following screen.



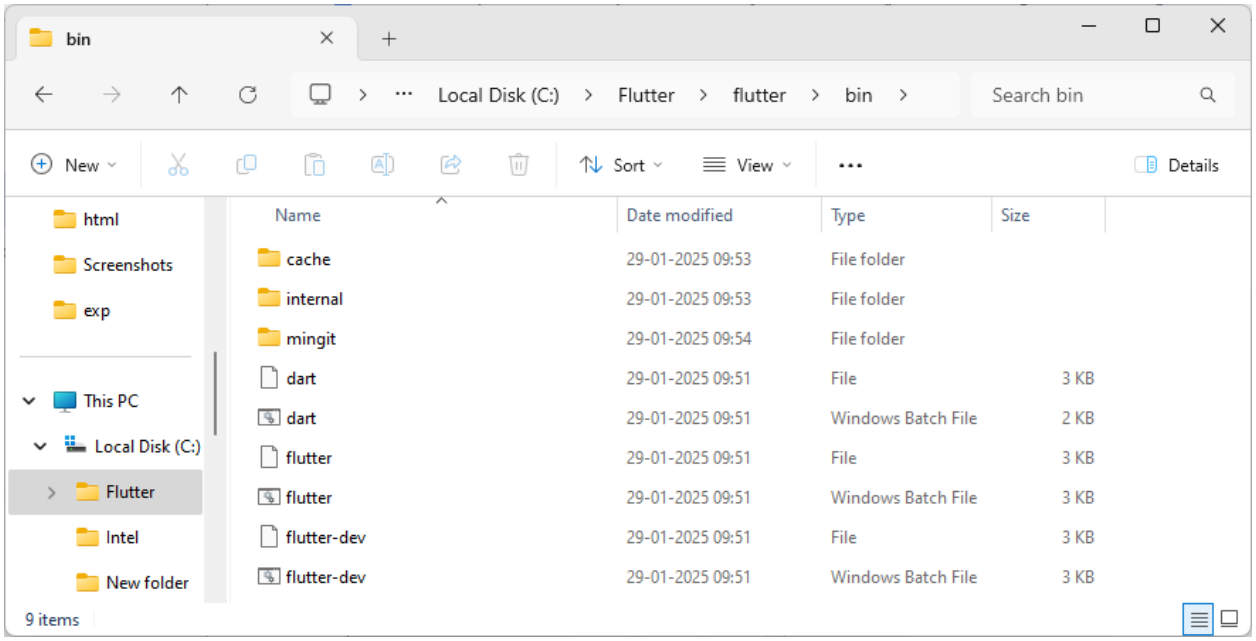
Step 2: Next, to download the latest Flutter SDK, click on the Windows icon. Here, you will find the download link for SDK.



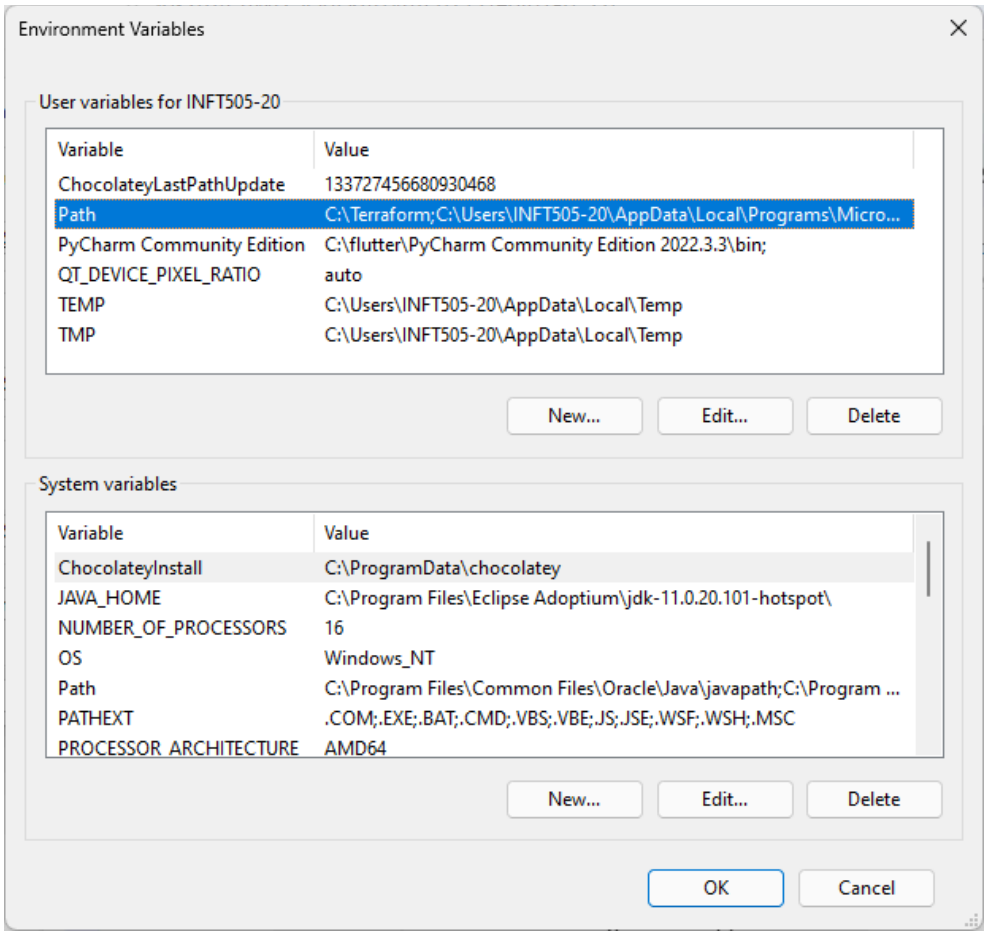
Step 3: When your download is complete, extract the zip file and place it in the desired installation folder or location, for example, C: /Flutter.



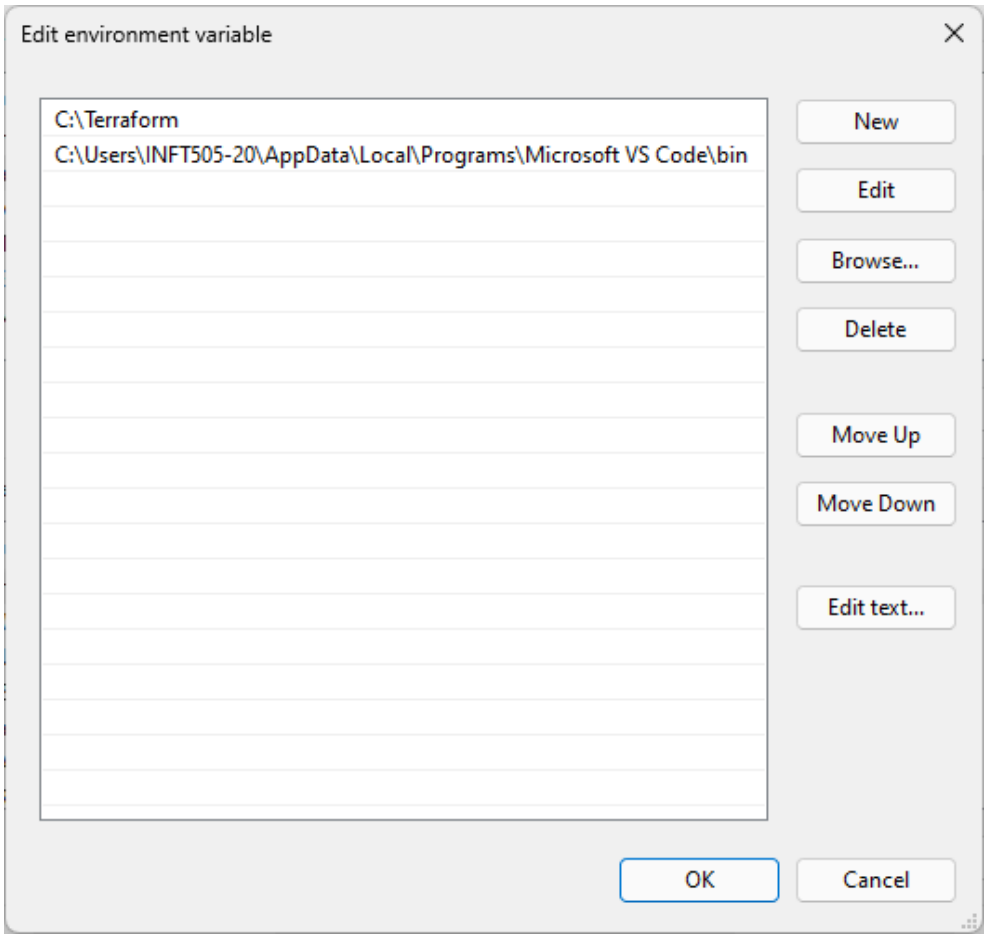
Step 4: To run the Flutter command in regular windows console, you need to update the system path to include the flutter bin directory. The following steps are required to do this:



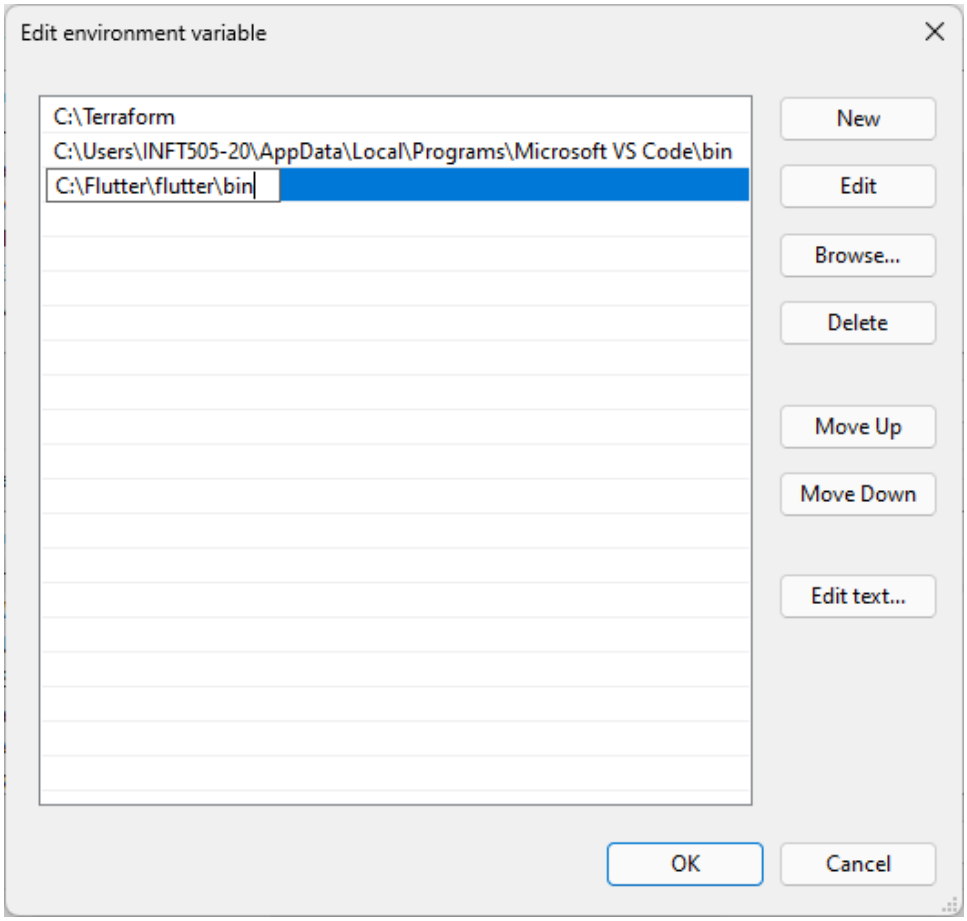
Step 4.1: Go to MyComputer properties -> advanced tab -> environment variables. You will get the following screen.



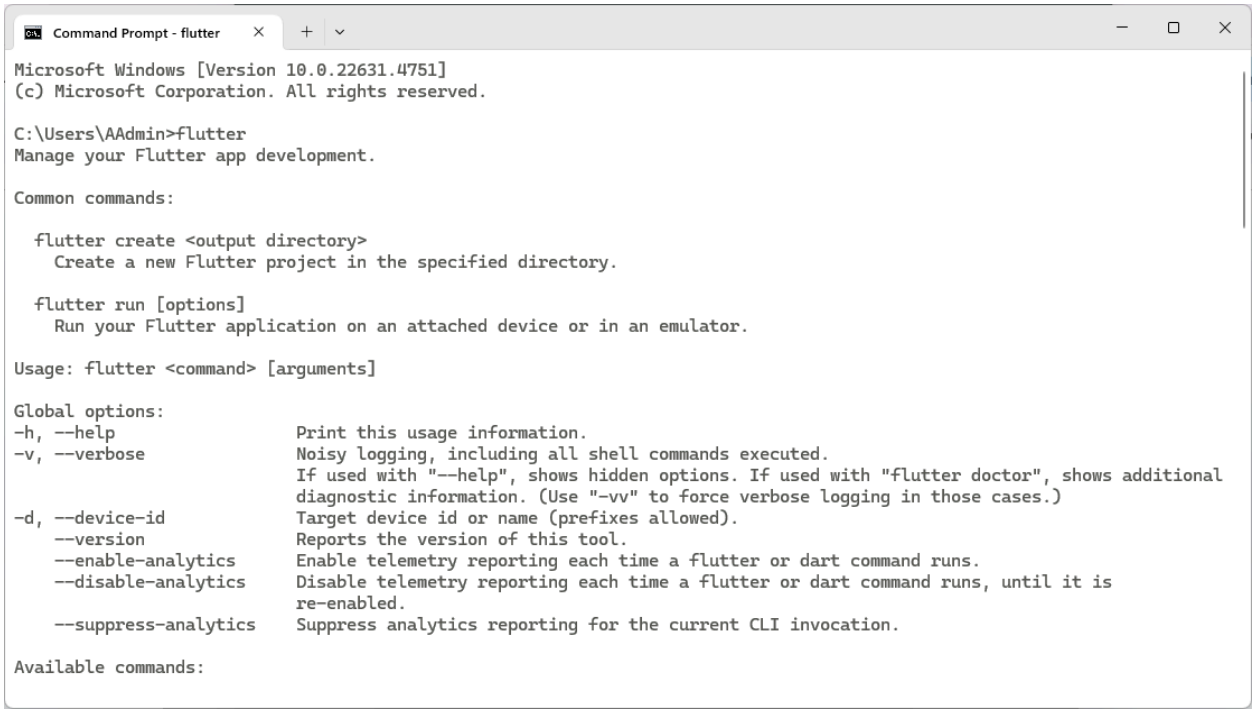
Step 4.2: Now, select path -> click on edit. The following screen appears



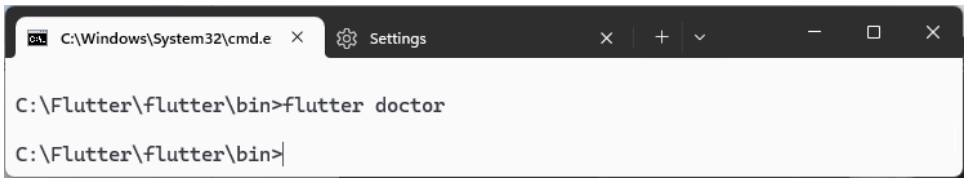
Step 4.3: In the above window, click on New->write path of Flutter bin folder in variable value -> ok -> ok -> ok.



Step 5: Now, run the $ flutter command in the command prompt.
Now, 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.

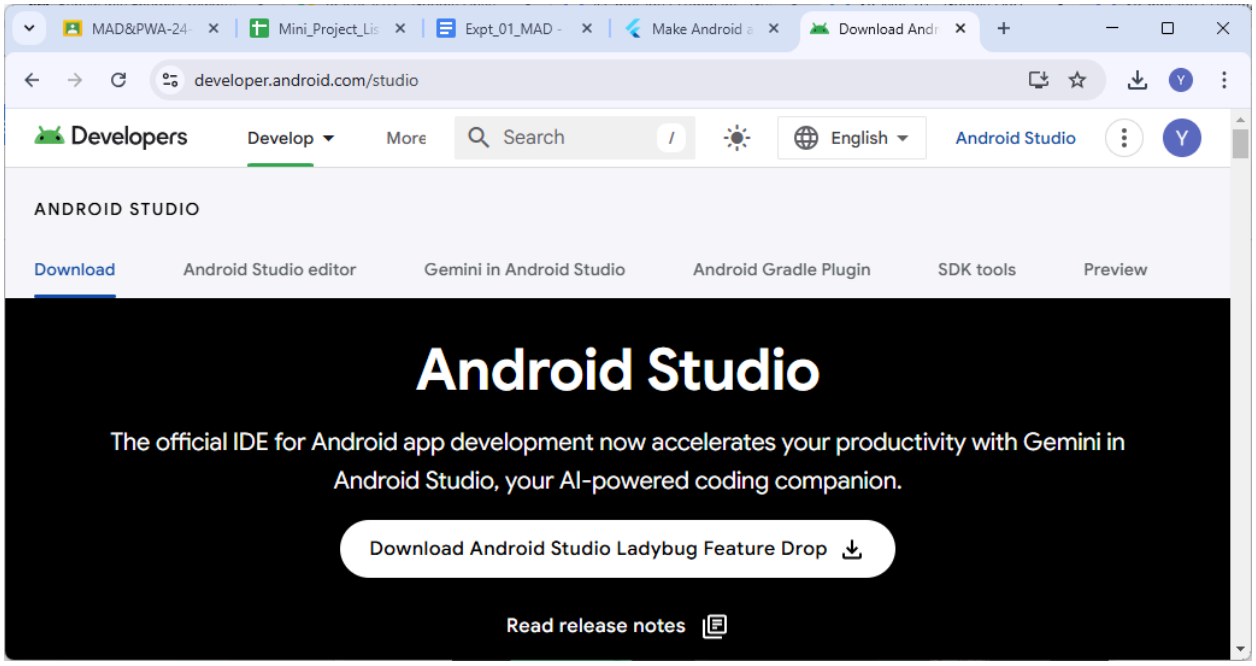


Step 6: When you run the above command, it will analyze the system and show its report, as shown in the below image. Here, you will find the details of all missing tools, which required to run Flutter as well as the development tools that are available but not connected with the device.

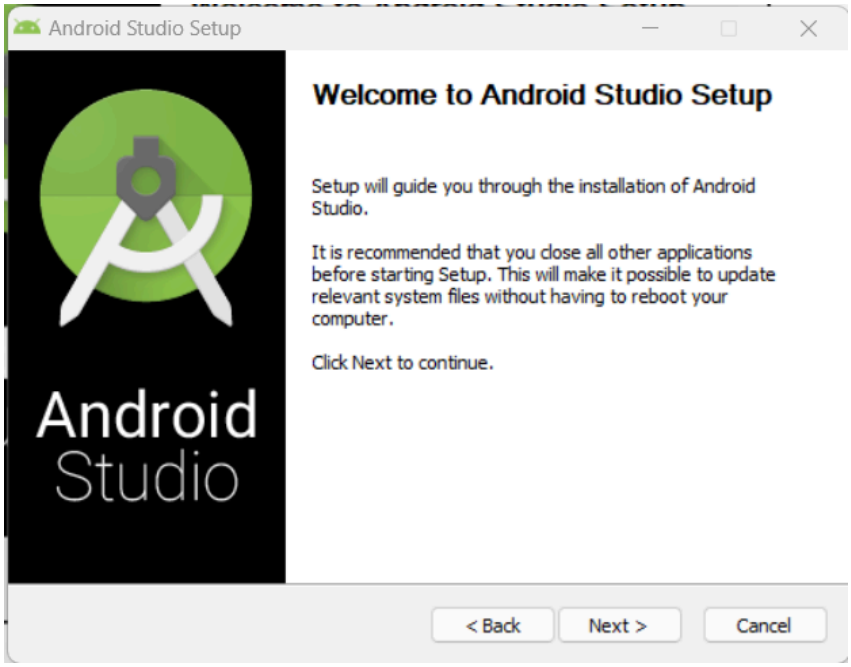


Step 7: Install the Android SDK. If the flutter doctor command does not find the Android SDK tool in your system, then you need first to install the Android Studio IDE. To install Android Studio IDE, do the following steps.

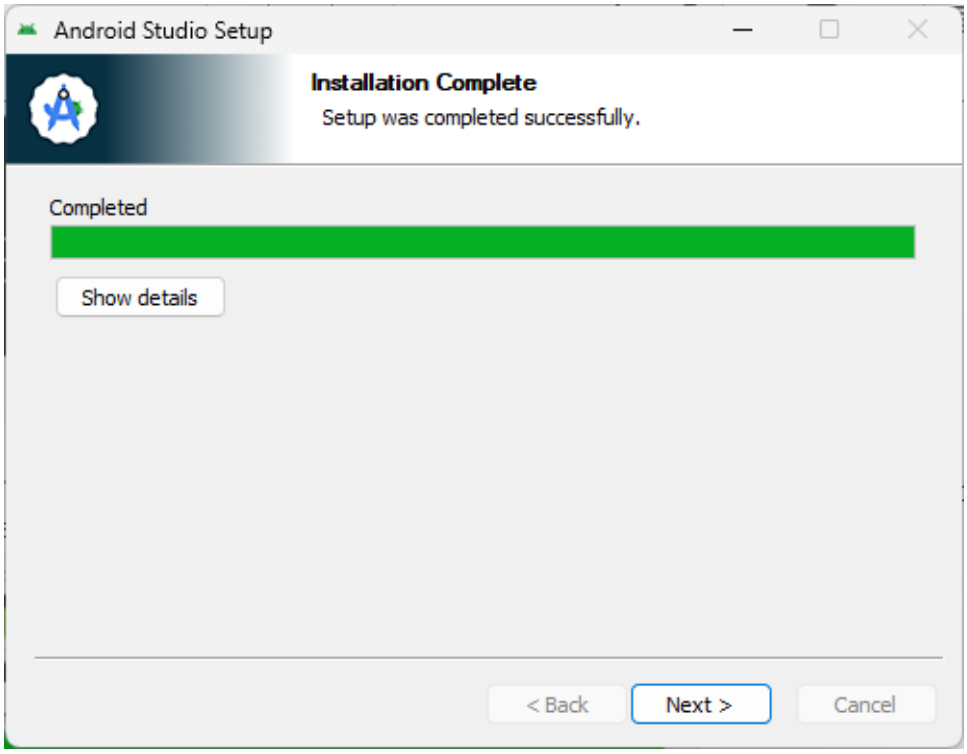
Step 7.1: Download the latest Android Studio executable or zip file from the official site.



Step 7.2: When the download is complete, open the .exe file and run it. You will get the following dialog box.

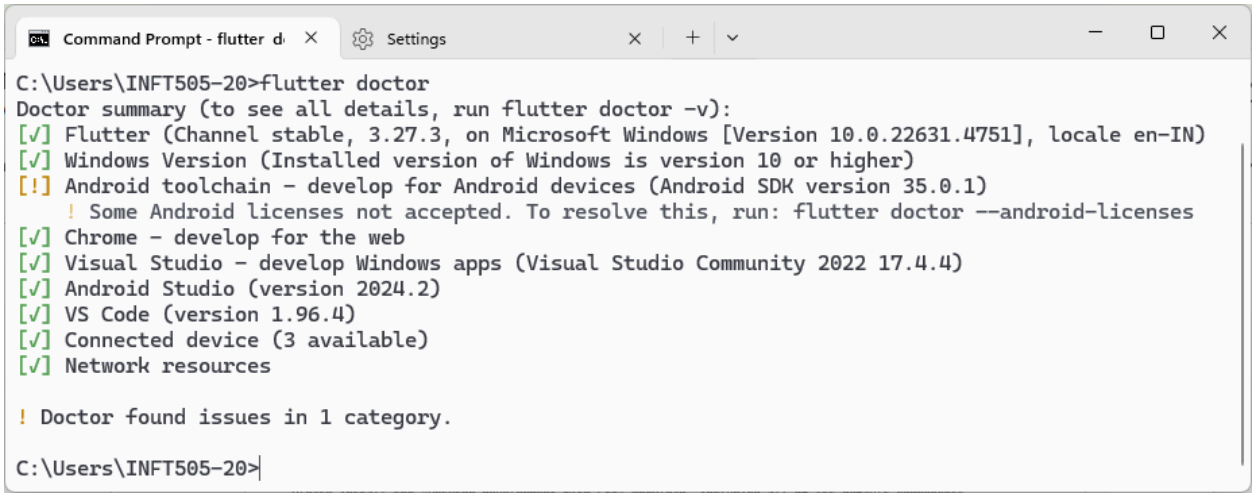


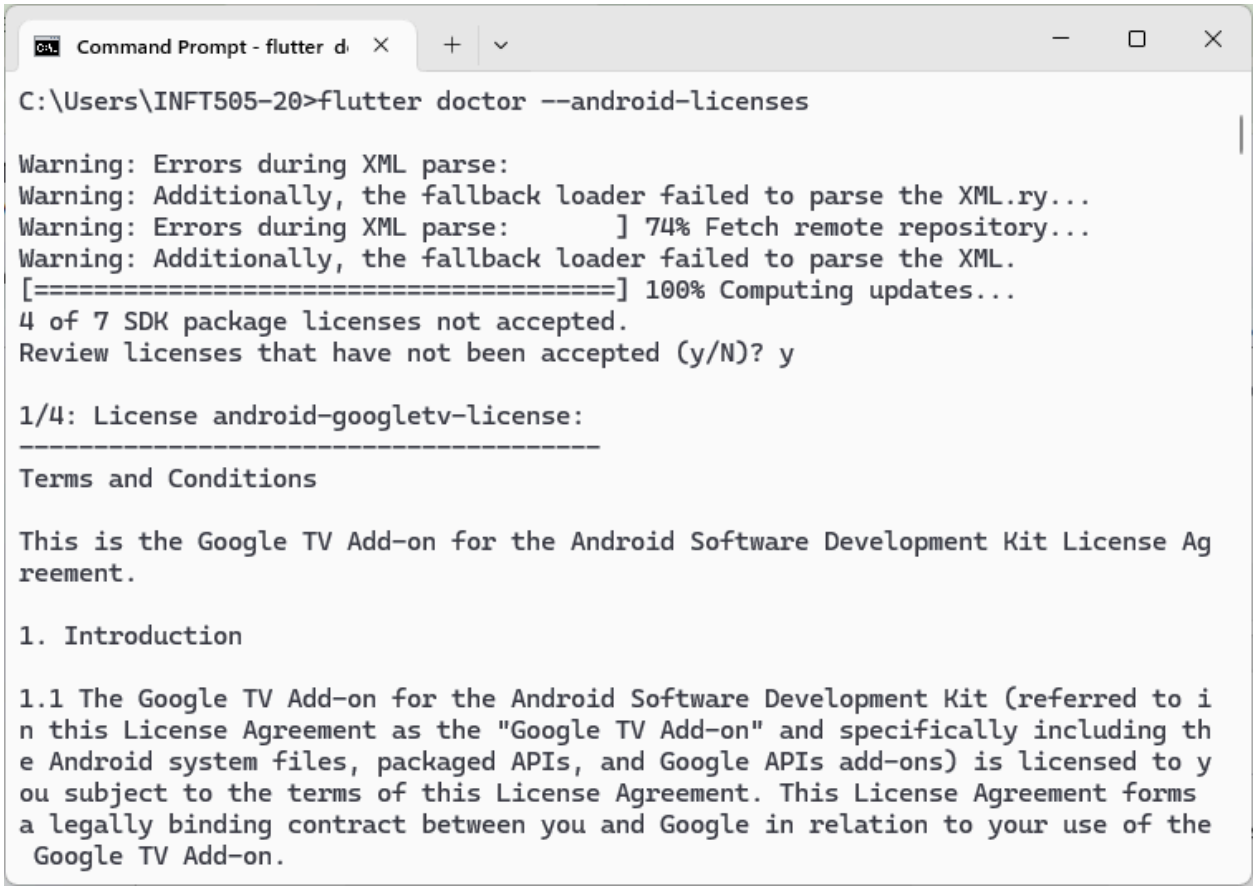
Step 7.3: Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.

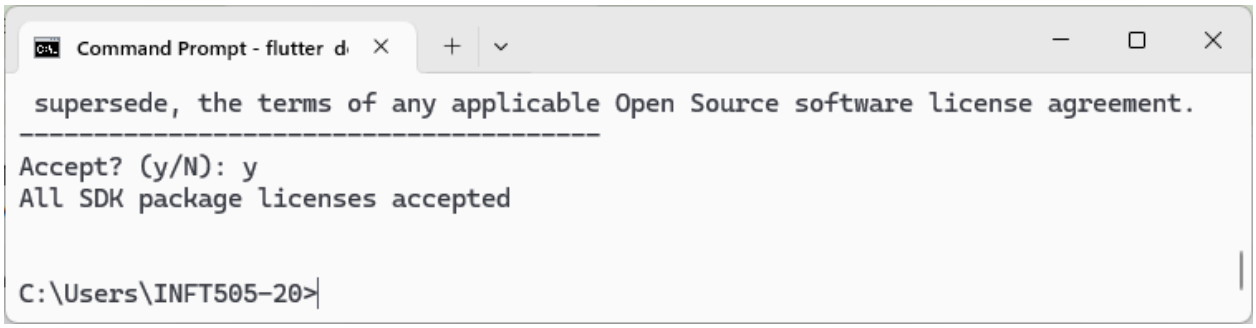


Step 7.4: In the above screen, click Next-> Finish. Once the Finish button is clicked, you need to choose the 'Don't import Settings option' and click OK. It will start the Android Studio.

Step 7.5: run the $ flutter doctor command and Run flutter doctor --android-licenses command.

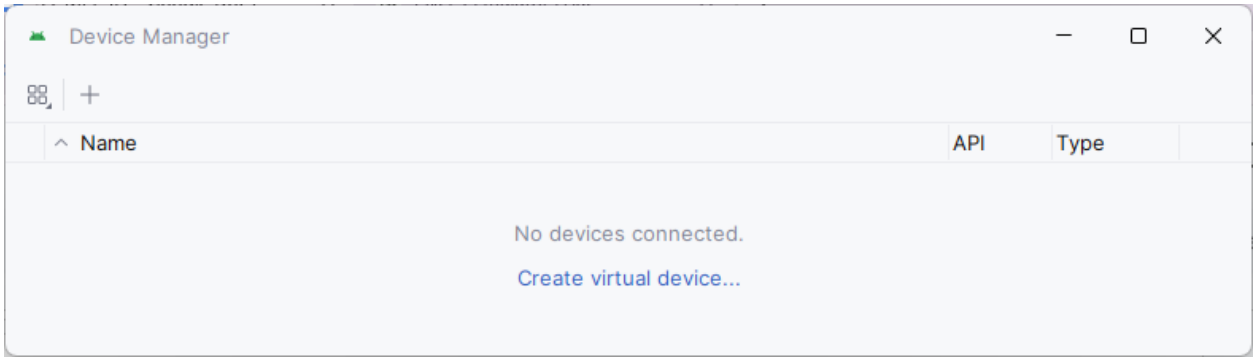




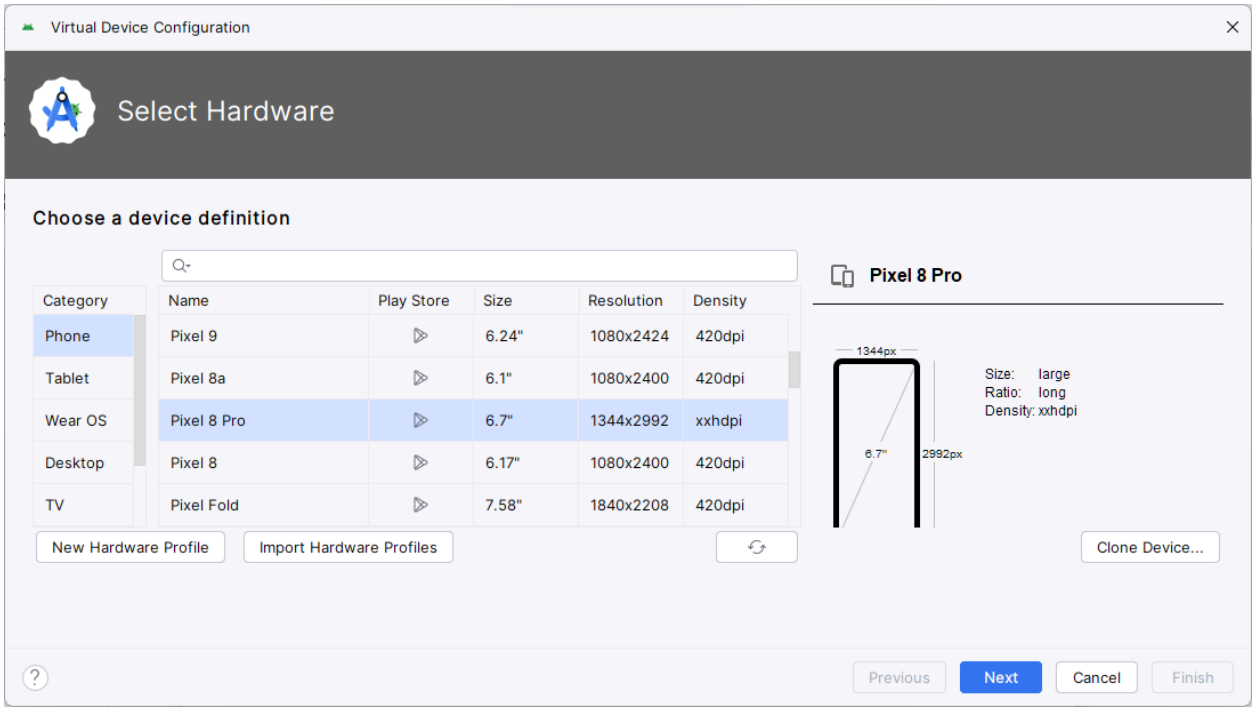


Step 8: Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application.

Step 8.1: To set an Android emulator, go to Android Studio > Tools > Android > AVD Manager and select Create Virtual Device. Or, go to Help->Find Action->Type Emulator in the search box. You will get the following screen.

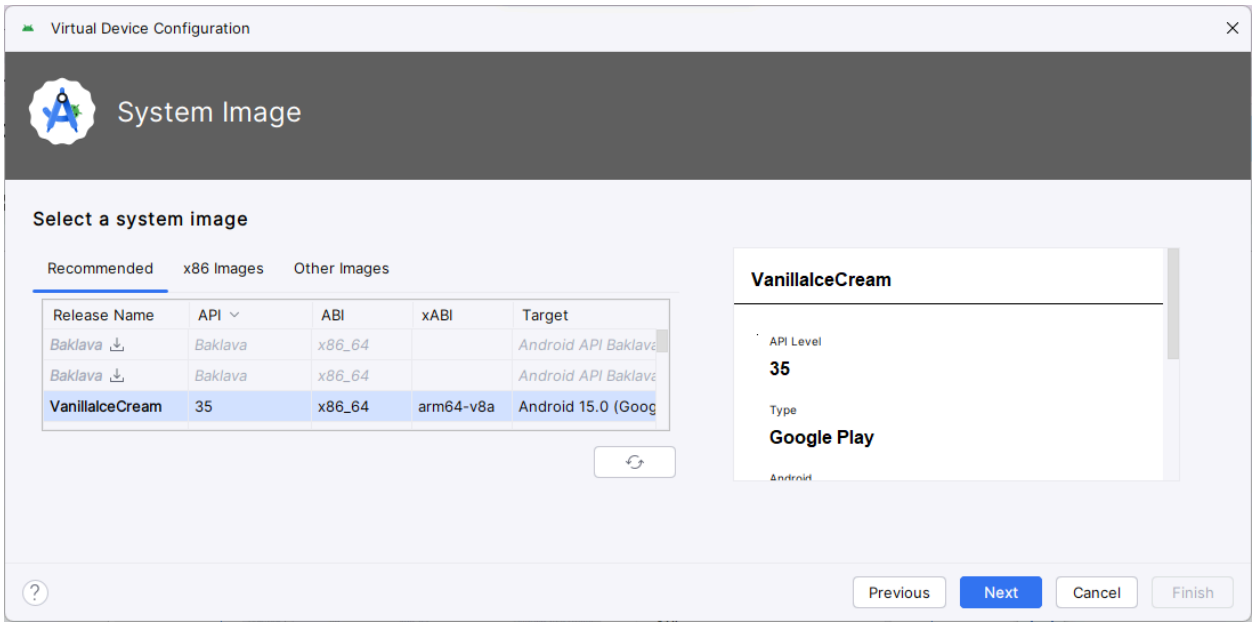


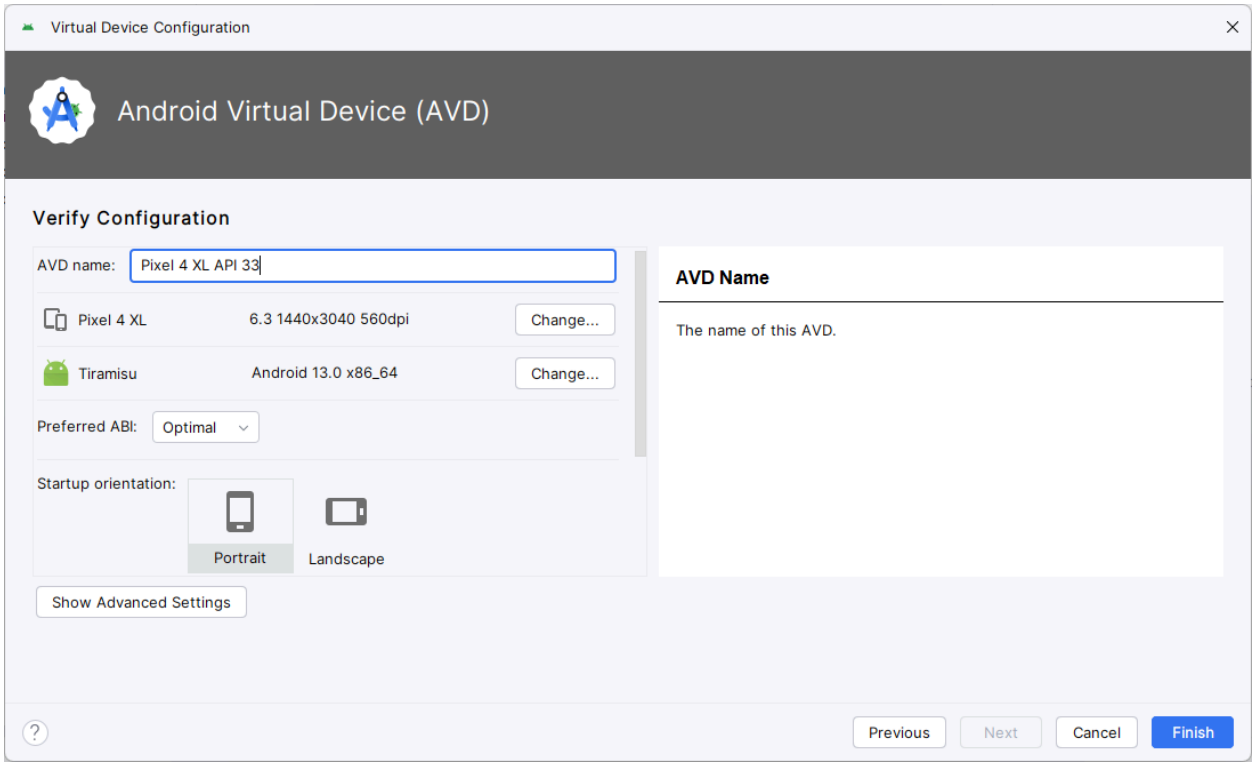
Step 8.2: Choose your device definition and click on Next.



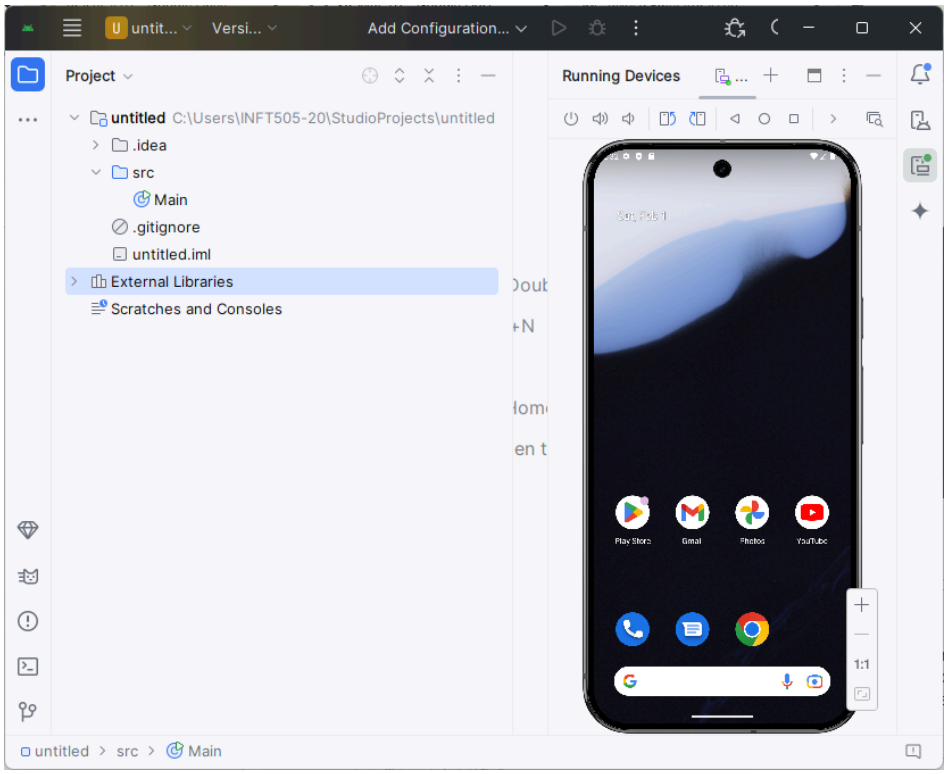
Step 8.3: Select the system image for the latest Android version and click on Next.

Step 8.4: Now, verify the all AVD configuration. If it is correct, click on Finish. The following screen appears.



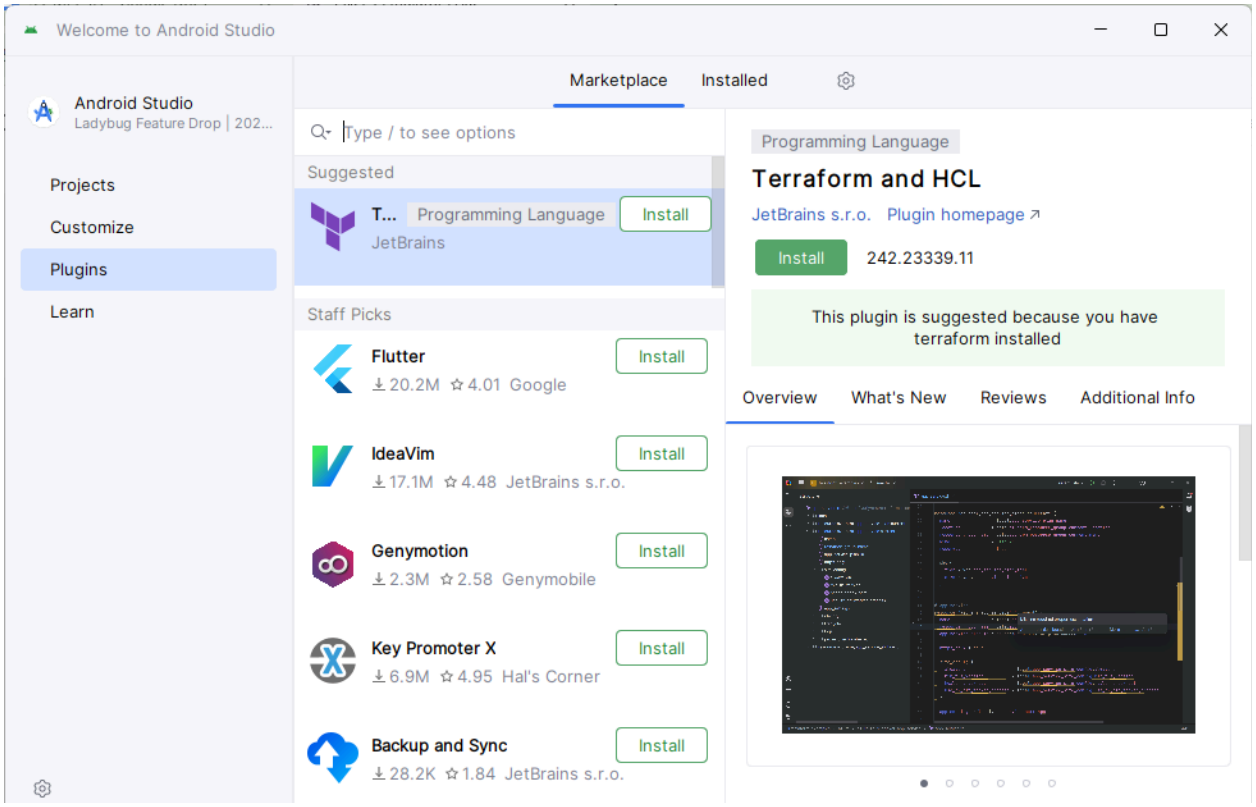


Step 8.5: Last, click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen.

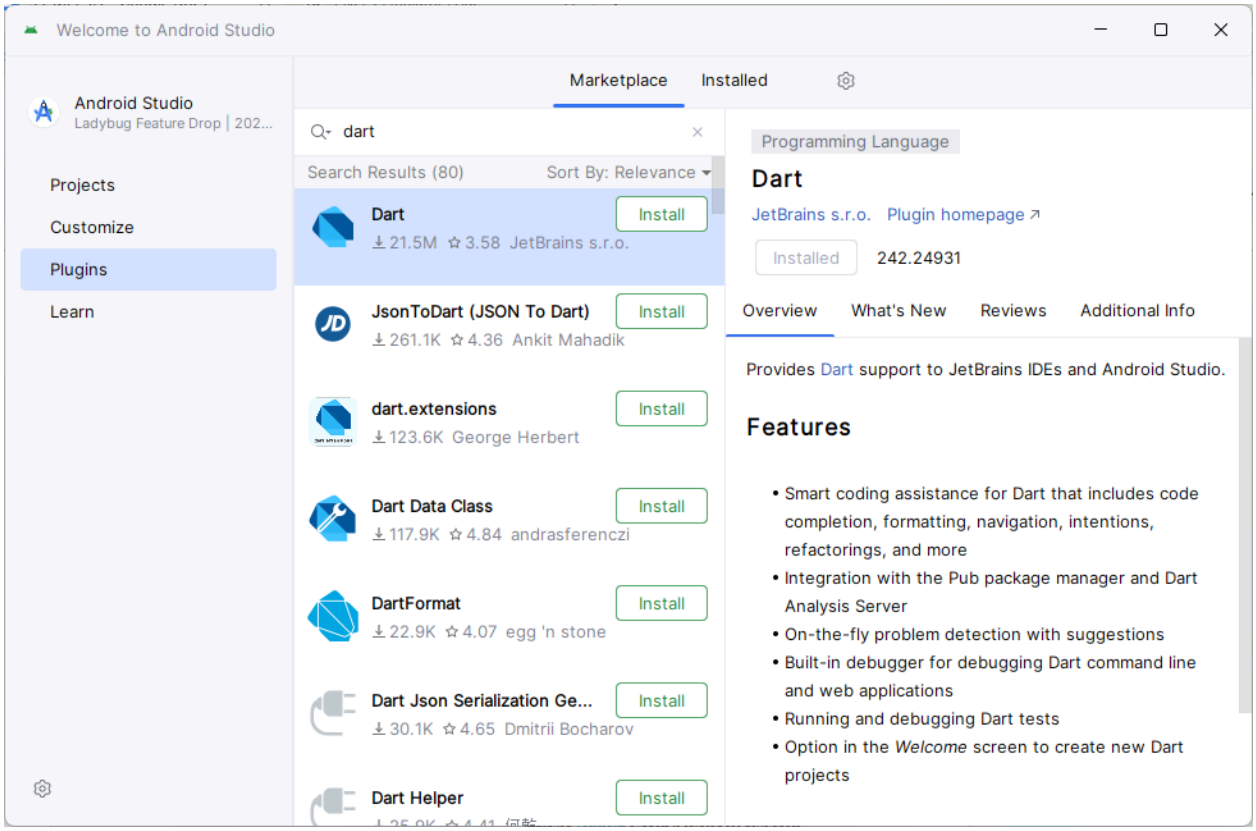


Step 9: Now, install the Flutter and Dart plugin for building Flutter applications in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself. Do the following steps to install these plugins.

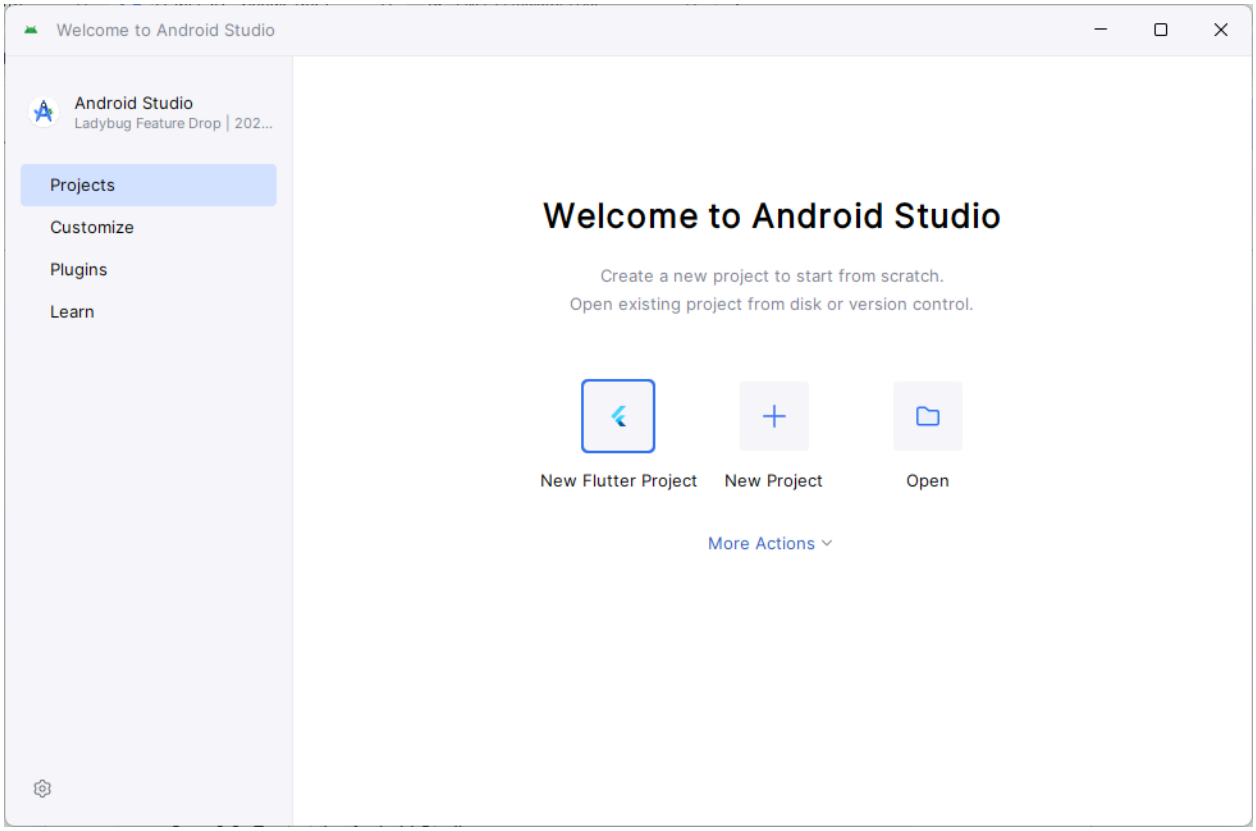
Step 9.1: Open the Android Studio and then go to File->Settings->Plugins.



Step 9.2: Now, search the Flutter plugin. If found, select Flutter plugin and click install. When you click on install, it will ask you to install Dart plugin as below screen. Click yes to proceed.



Step 9.3: Restart the Android Studio.



Name : Naikwadi Yash Shivdas

MPL Practical 02

Aim: To design a Flutter UI by including common widgets.

Introduction

Flutter is an open-source UI toolkit by Google that allows developers to build cross-platform applications for Android, iOS, web, and desktop using a single codebase. It uses the Dart programming language and provides a rich set of pre-built widgets to create beautiful and responsive user interfaces.

A Flutter UI is structured using widgets, which can be broadly classified into:

- StatelessWidget: A widget that does not change over time.
- StatefulWidget: A widget that can update dynamically based on user interaction.

Flutter applications typically use a Scaffold widget, which provides a structure that includes an AppBar, body, floating action buttons, bottom navigation bars, and other UI elements.

Implementation in GreenFund Connect

"GreenFund Connect" is a renewable energy crowdfunding platform where users can explore and support various green energy projects. The UI is designed to be simple, user-friendly, and visually appealing.

Key Features in the UI

1. AppBar – Displays the application title at the top.
2. BottomNavigationBar – Allows users to switch between "Projects" and "About" sections.
3. GridView – Used to display renewable energy projects in a structured way.
4. Card Widget – Displays each project with a clear layout.
5. FloatingActionButton – Provides an option for quick actions.

Code Explanation

1. Main Application (main.dart)

- The MaterialApp widget initializes the app with a green theme.
- The HomeScreen widget is defined as a StatefulWidget because the bottom navigation bar requires state management.

2. HomeScreen Widget

- Contains a BottomNavigationBar with two sections: Projects and About.
- Uses setState to switch between the two sections dynamically.
- Includes a FloatingActionButton for additional actions.

3. ProjectGrid Widget

- Uses GridView.builder to create a grid layout for displaying projects.
- Each project is shown inside a Card widget, ensuring a clean UI.

4. AboutSection Widget

- Displays a brief description of the platform.

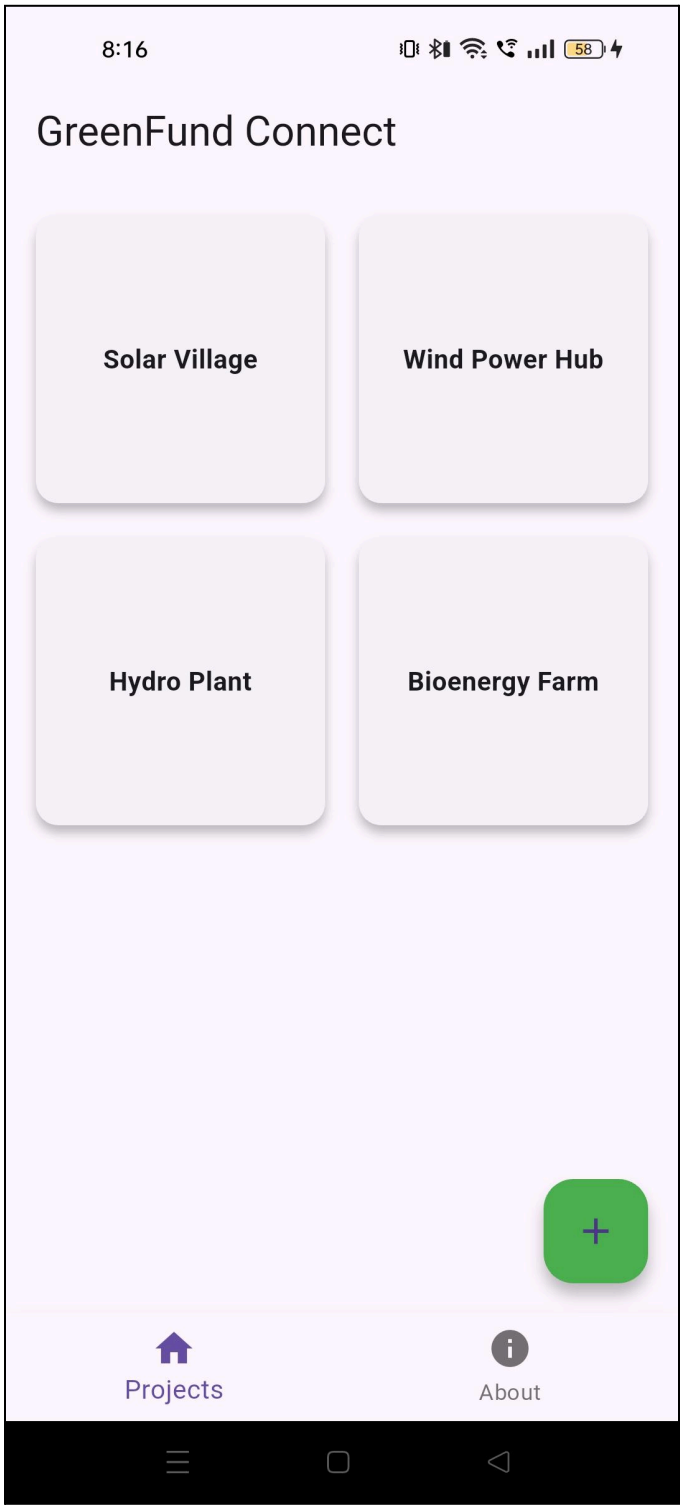
Code:

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

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

class GreenFundConnect extends StatelessWidget {
  const GreenFundConnect({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false, // Removes the Debug Banner
      title: 'GreenFund Connect',
      theme: ThemeData(
        primarySwatch: Colors.green,
      ),
      home: const HomeScreen(),
    );
  }
}

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

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

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

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('GreenFund Connect')),
      body: Padding(
```

```
      padding: const EdgeInsets.all(12.0),
      child: _selectedIndex == 0 ? const ProjectGrid() : const AboutSection(),
    ),
    floatingActionButton: FloatingActionButton(
      backgroundColor: Colors.green,
      onPressed: () {},
      child: const Icon(Icons.add),
    ),
    bottomNavigationBar: BottomNavigationBar(
      currentIndex: _selectedIndex,
      onTap: _onItemTapped,
      items: const [
        BottomNavigationBarItem(icon: Icon(Icons.home), label: 'Projects'),
        BottomNavigationBarItem(icon: Icon(Icons.info), label: 'About'),
      ],
    ),
  );
 }
}

class ProjectGrid extends StatelessWidget {
 const ProjectGrid({Key? key}) : super(key: key);

 final List<String> projects = const [
  "Solar Village",
  "Wind Power Hub",
  "Hydro Plant",
  "Bioenergy Farm"
 ];

 @override
 Widget build(BuildContext context) {
  return GridView.builder(
   gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
    crossAxisCount: 2,
    childAspectRatio: 1,
    crossAxisSpacing: 10,
    mainAxisSpacing: 10,
   ),
   itemCount: projects.length,
   itemBuilder: (context, index) {
    return Card(
     elevation: 4,
     child: Center(
```

```
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Text(
              projects[index],
              style: const TextStyle(fontWeight: FontWeight.bold),
              textAlign: TextAlign.center,
            ),
          ),
        ),
      );
    },
  );
 }
}

class AboutSection extends StatelessWidget {
 const AboutSection({Key? key}) : super(key: key);

 @override
 Widget build(BuildContext context) {
  return const Center(
    child: Text(
      "GreenFund Connect aims to support renewable energy projects worldwide.",
      style: TextStyle(fontSize: 18),
      textAlign: TextAlign.center,
    ),
  );
 }
}
```

Screenshot:



Conclusion

In this project, we implemented a unique Flutter UI using BottomNavigationBar, GridView, and FloatingActionButton to create an interactive layout for GreenFund Connect. Initially, we faced issues with GridView alignment and state management in BottomNavigationBar, but we resolved them by adjusting childAspectRatio and correctly updating the selected index using setState.

Name : Naikwadi Yash Shivdas

MPL Practical 03

Aim: To include icons, images, fonts in Flutter app.

Theory:

In Flutter, visual elements such as icons, images, and fonts play a crucial role in designing an interactive interface. These elements can be integrated as follows:

1. Icons:
    - Flutter provides a built-in Icons class that contains material design icons.
    - Custom icons can also be used by including external icon packs.
2. Images:
    - Images can be loaded from assets, network URLs, or user devices.
    - For asset images, the path must be defined in pubspec.yaml.
3. Fonts:
    - Custom fonts can be added by specifying the font family in pubspec.yaml.
    - The GoogleFonts package allows easy usage of Google Fonts in Flutter apps.

Implementation in Our Code:

In our Flutter project, we implemented these features as follows:

- Icons: Used material design icons like Icons.wb_sunny, Icons.eco, and Icons.house to represent different projects.
- Images: Stored project images in the assets/images/ directory and displayed them using Image.asset().
- Fonts: Used the GoogleFonts package to apply modern typography to project titles, descriptions, and buttons.

Code:

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

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

class GreenFundConnect extends StatelessWidget {
 const GreenFundConnect({Key? key}) : super(key: key);

 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   debugShowCheckedModeBanner: false,
   title: 'GreenFund Connect',
   theme: ThemeData(
    primarySwatch: Colors.green,
```

```
      primaryColor: const Color(0xFF2E7D32), // Dark green
      secondaryHeaderColor: const Color(0xFF81C784), // Light green
      fontFamily: GoogleFonts.poppins().fontFamily,
      textTheme: TextTheme(
       headlineLarge: GoogleFonts.montserrat(
        fontWeight: FontWeight.bold,
        color: const Color(0xFF2E7D32),
       ),
       titleLarge: GoogleFonts.poppins(
        fontWeight: FontWeight.w600,
        color: Colors.black87,
       ),
       bodyLarge: GoogleFonts.poppins(
        color: Colors.black87,
       ),
       bodyMedium: GoogleFonts.poppins(
        color: Colors.black87,
       ),
      ),
      elevatedButtonTheme: ElevatedButtonThemeData(
       style: ElevatedButton.styleFrom(
        backgroundColor: const Color(0xFF2E7D32),
        foregroundColor: Colors.white,
        shape: RoundedRectangleBorder(
         borderRadius: BorderRadius.circular(10),
        ),
        padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 12),
       ),
      ),
     ),
     home: const HomeScreen(),
    );
  }
}

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

  @override
  _HomeScreenState createState() => _HomeScreenState();
}
```

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

 final List<Widget> _screens = [
  const ProjectsScreen(),
  const MyInvestmentsScreen(),
  const ProfileScreen(),
 ];

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

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: Row(
     children: [
      Image.asset(
       'assets/images/logo.png',
       height: 32,
      ),
      const SizedBox(width: 12),
      Text(
       'GreenFund Connect',
       style: GoogleFonts.montserrat(
        fontWeight: FontWeight.bold,
        color: Colors.green,
       ),
      ),
     ],
    ),
    actions: [
     IconButton(
      icon: const Icon(Icons.notifications_outlined),
      onPressed: () {
       // Notifications action
      },
     ),
```

```
        IconButton(
          icon: const Icon(Icons.search),
          onPressed: () {
            // Search action
          },
        ),
      ],
    ),
    body: _screens[_selectedIndex],
    floatingActionButton: FloatingActionButton(
      backgroundColor: Theme.of(context).primaryColor,
      onPressed: () {
        // Add project action
      },
      child: const Icon(Icons.add),
      tooltip: 'Add new project',
    ),
    bottomNavigationBar: BottomNavigationBar(
      currentIndex: _selectedIndex,
      onTap: _onItemTapped,
      selectedItemColor: Theme.of(context).primaryColor,
      unselectedItemColor: Colors.grey,
      items: const [
        BottomNavigationBarItem(
          icon: Icon(Icons.home_outlined),
          activeIcon: Icon(Icons.home),
          label: 'Projects',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.account_balance_wallet_outlined),
          activeIcon: Icon(Icons.account_balance_wallet),
          label: 'My Investments',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.person_outline),
          activeIcon: Icon(Icons.person),
          label: 'Profile',
        ),
      ],
    ),
  );
}
```

```
}

class ProjectsScreen extends StatelessWidget {
  const ProjectsScreen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Padding(
          padding: const EdgeInsets.fromLTRB(16, 16, 16, 8),
          child: Text(
            'Featured Projects',
            style: Theme.of(context).textTheme.titleLarge,
          ),
        ),
        SizedBox(
          height: 250,
          child: ListView(
            scrollDirection: Axis.horizontal,
            padding: const EdgeInsets.symmetric(horizontal: 16),
            children: const [
              FeaturedProjectCard(
                title: 'Solar Village Initiative',
                location: 'Maharashtra, India',
                fundingPercentage: 75,
                imageAsset: 'assets/images/solar_village.png',
              ),
              FeaturedProjectCard(
                title: 'Wind Power Hub',
                location: 'Tamil Nadu, India',
                fundingPercentage: 45,
                imageAsset: 'assets/images/wind_power.png',
              ),
              FeaturedProjectCard(
                title: 'Community Hydro Plant',
                location: 'Himachal Pradesh, India',
                fundingPercentage: 60,
                imageAsset: 'assets/images/hydro_plant.png',
              ),
            ],
```

```
        ),
      ),
      Padding(
        padding: const EdgeInsets.fromLTRB(16, 24, 16, 8),
        child: Text(
          'All Projects',
          style: Theme.of(context).textTheme.titleLarge,
        ),
      ),
      Expanded(child: ProjectGrid()),
    ],
  );
 }
}

class FeaturedProjectCard extends StatelessWidget {
 final String title;
 final String location;
 final int fundingPercentage;
 final String imageAsset;

 const FeaturedProjectCard({
  Key? key,
  required this.title,
  required this.location,
  required this.fundingPercentage,
  required this.imageAsset,
 }) : super(key: key);

 @override
 Widget build(BuildContext context) {
  return Container(
   width: 280,
   margin: const EdgeInsets.only(right: 16),
   decoration: BoxDecoration(
    borderRadius: BorderRadius.circular(12),
    boxShadow: [
     BoxShadow(
      color: Colors.black.withOpacity(0.1),
      blurRadius: 8,
      offset: const Offset(0, 4),
     ),
```

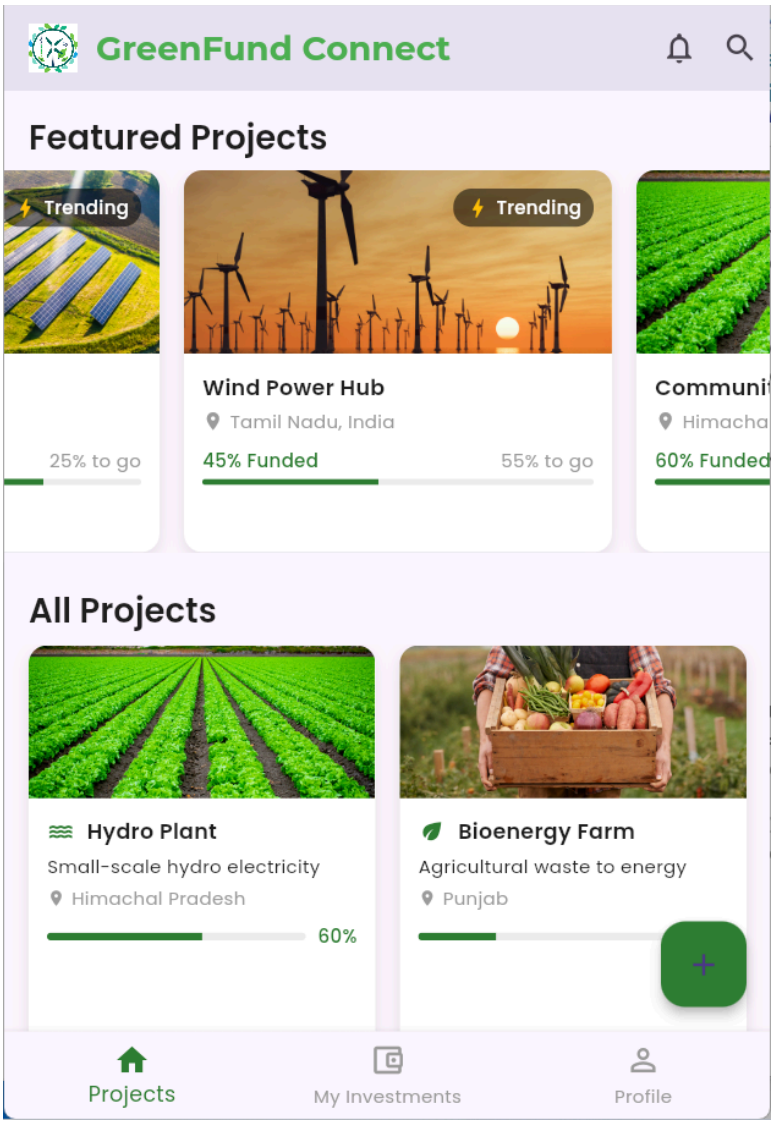
```
    ],
  ),
  child: ClipRRect(
    borderRadius: BorderRadius.circular(12),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Stack(
          children: [
            Image.asset(
              imageAsset,
              height: 120,
              width: double.infinity,
              fit: BoxFit.cover,
            ),
            Positioned(
              top: 12,
              right: 12,
              child: Container(
                padding: const EdgeInsets.symmetric(horizontal: 8, vertical: 4),
                decoration: BoxDecoration(
                  color: Colors.black54,
                  borderRadius: BorderRadius.circular(12),
                ),
                child: Row(
                  children: [
                    const Icon(
                      Icons.bolt,
                      color: Colors.amber,
                      size: 16,
                    ),
                    const SizedBox(width: 4),
                    Text(
                      'Trending',
                      style: GoogleFonts.poppins(
                        color: Colors.white,
                        fontSize: 12,
                        fontWeight: FontWeight.w500,
                      ),
                    ),
                  ],
                ),
              ),
```

```
        ),
      ),
    ],
  ),
  Expanded(
    child: Container(
      padding: const EdgeInsets.all(12),
      color: Colors.white,
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(
            title,
            style: GoogleFonts.poppins(
              fontWeight: FontWeight.w600,
              fontSize: 14,
            ),
            maxLines: 1,
            overflow: TextOverflow.ellipsis,
          ),
          const SizedBox(height: 4),
          Row(
            children: [
              const Icon(
                Icons.location_on,
                size: 14,
                color: Colors.grey,
              ),
              const SizedBox(width: 4),
              Text(
                location,
                style: GoogleFonts.poppins(
                  fontSize: 12,
                  color: Colors.grey,
                ),
              ),
            ],
          ),
          const SizedBox(height: 8),
          Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
```

```
                Row(
                 mainAxisAlignment: MainAxisAlignment.spaceBetween,
                 children: [
                  Text(
                   '$fundingPercentage% Funded',
                   style: GoogleFonts.poppins(
                    fontWeight: FontWeight.w500,
                    fontSize: 12,
                    color: Theme.of(context).primaryColor,
                   ),
                  ),
                  Text(
                   '${100 - fundingPercentage}% to go',
                   style: GoogleFonts.poppins(
                    fontSize: 12,
                    color: Colors.grey,
                   ),
                  ),
                 ],
                ),
                const SizedBox(height: 4),
                ClipRRect(
                 borderRadius: BorderRadius.circular(10),
                 child: LinearProgressIndicator(
                  value: fundingPercentage / 100,
                  backgroundColor: Colors.grey.shade200,
                  valueColor: AlwaysStoppedAnimation<Color>(
                   Theme.of(context).primaryColor,
                  ),
                 ),
                ),
               ],
              ),
             ],
            ),
           ),
          ),
         ],
        ),
       ),
      );
    }
```

}

Screenshot:



Conclusion

In this experiment, we successfully implemented icons, images, and custom fonts to enhance the UI of our Flutter app. Initially, we faced issues like missing asset declarations in pubspec.yaml and incorrect image paths, but we resolved them by properly configuring assets, using hot reload, and checking error logs for debugging.

Name : Naikwadi Yash Shivdas

MPL Practical 04

Aim: To create an interactive Form using a form widget.

Theory:

Forms are an essential part of mobile applications as they allow users to input and submit data. In Flutter, the Form widget provides a structured way to handle user input, validation, and submission. It works along with TextFormField to create fields for user interaction, such as entering text, passwords, or email addresses. Forms also use a GlobalKey<FormState> to manage validation and state.

In this experiment, we implemented a login and signup form using the Form widget. The login form includes fields for email and password, while the signup form additionally has name and confirm password fields. We have applied validation to ensure that:

- The email field accepts only valid email formats.
- The password field requires at least six characters.
- The confirm password field ensures both passwords match.

Navigation is handled properly so that once a user logs in or signs up successfully, they cannot go back to the login screen using the back button. This is achieved using Navigator.pushAndRemoveUntil().

Code:

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

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

class GreenFundConnect extends StatelessWidget {
 const GreenFundConnect({Key? key}) : super(key: key);

 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   debugShowCheckedModeBanner: false,
   title: 'GreenFund Connect',
   theme: ThemeData(
    primarySwatch: Colors.green,
    primaryColor: const Color(0xFF2E7D32), // Dark green
    secondaryHeaderColor: const Color(0xFF81C784), // Light green
    fontFamily: GoogleFonts.poppins().fontFamily,
    textTheme: TextTheme(
     headlineLarge: GoogleFonts.montserrat(
```

```
          fontWeight: FontWeight.bold,
          color: const Color(0xFF2E7D32),
        ),
        titleLarge: GoogleFonts.poppins(
          fontWeight: FontWeight.w600,
          color: Colors.black87,
        ),
        bodyLarge: GoogleFonts.poppins(
          color: Colors.black87,
        ),
        bodyMedium: GoogleFonts.poppins(
          color: Colors.black87,
        ),
      ),
      elevatedButtonTheme: ElevatedButtonThemeData(
        style: ElevatedButton.styleFrom(
          backgroundColor: const Color(0xFF2E7D32),
          foregroundColor: Colors.white,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(10),
          ),
          padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 12),
        ),
      ),
    ),
    home: const LoginScreen(),
  );
 }
}

class LoginScreen extends StatelessWidget {
 const LoginScreen({Key? key}) : super(key: key);

 @override
 Widget build(BuildContext context) {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();

  return Scaffold(
   appBar: AppBar(title: const Text('Login')),
   body: Padding(
```

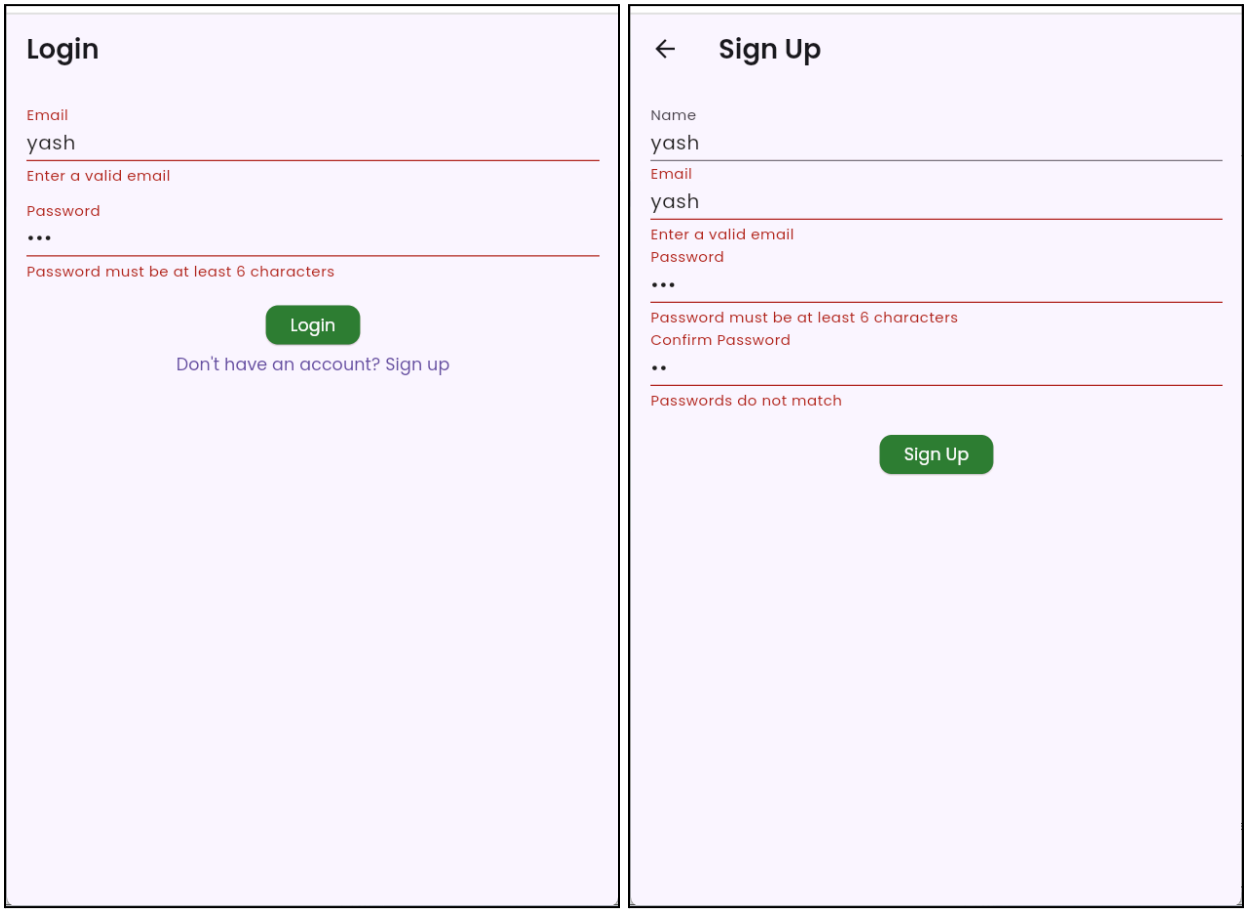
```
padding: const EdgeInsets.all(16.0),
child: Form(
 key: _formKey,
 child: Column(
  children: [
   TextFormField(
    controller: emailController,
    decoration: const InputDecoration(labelText: 'Email'),
    keyboardType: TextInputType.emailAddress,
    validator: (value) {
     if (value == null || value.isEmpty) {
      return 'Please enter an email';
     } else if (!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(value)) {
      return 'Enter a valid email';
     }
     return null;
    },
   ),
   const SizedBox(height: 10),
   TextFormField(
    controller: passwordController,
    decoration: const InputDecoration(labelText: 'Password'),
    obscureText: true,
    validator: (value) {
     if (value == null || value.length < 6) {
      return 'Password must be at least 6 characters';
     }
     return null;
    },
   ),
   const SizedBox(height: 20),
   ElevatedButton(
    onPressed: () {
     if (_formKey.currentState!.validate()) {
      Navigator.push(
       context,
       MaterialPageRoute(builder: (context) => const HomeScreen()),
      );
     }
    },
    child: const Text('Login'),
   ),
```

```
            TextButton(
              onPressed: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(builder: (context) => const SignupScreen()),
                );
              },
              child: const Text('Don\'t have an account? Sign up'),
            ),
          ],
        ),
      ),
    ),
  );
 }
}

class SignupScreen extends StatelessWidget {
 const SignupScreen({Key? key}) : super(key: key);

 @override
 Widget build(BuildContext context) {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController nameController = TextEditingController();
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();
  final TextEditingController confirmPasswordController = TextEditingController();

  return Scaffold(
   appBar: AppBar(title: const Text('Sign Up')),
   body: Padding(
    padding: const EdgeInsets.all(16.0),
    child: Form(
     key: _formKey,
     child: Column(
      children: [
       TextFormField(
        controller: nameController,
        decoration: const InputDecoration(labelText: 'Name'),
        validator: (value) => value!.isEmpty ? 'Enter your name' : null,
       ),
       TextFormField(
```

```
  controller: emailController,
  decoration: const InputDecoration(labelText: 'Email'),
  keyboardType: TextInputType.emailAddress,
  validator: (value) {
   if (value == null || value.isEmpty) {
    return 'Please enter an email';
   } else if (!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(value)) {
    return 'Enter a valid email';
   }
   return null;
  },
 ),
 TextFormField(
  controller: passwordController,
  decoration: const InputDecoration(labelText: 'Password'),
  obscureText: true,
  validator: (value) {
   if (value == null || value.length < 6) {
    return 'Password must be at least 6 characters';
   }
   return null;
  },
 ),
 TextFormField(
  controller: confirmPasswordController,
  decoration: const InputDecoration(labelText: 'Confirm Password'),
  obscureText: true,
  validator: (value) {
   if (value != passwordController.text) {
    return 'Passwords do not match';
   }
   return null;
  },
 ),
 const SizedBox(height: 20),
 ElevatedButton(
  onPressed: () {
   if (_formKey.currentState!.validate()) {
    Navigator.push(
     context,
     MaterialPageRoute(builder: (context) => const HomeScreen()),
    );
```

```
              }
            },
            child: const Text('Sign Up'),
          ),
        ],
      ),
    ),
  ),
 );
 }
}
```

Screenshot:



Conclusion

In this experiment, we successfully implemented interactive login and signup forms with proper validation, ensuring correct user input. Initially, we faced an issue where users could navigate back to the login screen after logging in, but we resolved it by using Navigator.pushAndRemoveUntil() to prevent back navigation.

Name : Naikwadi Yash Shivdas

MPL Practical 05

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

Theory:

1. Navigation & Routing
    - Flutter uses the Navigator widget for screen transitions.
    - Push (Navigator.push()) adds a new screen; Pop (Navigator.pop()) removes it.
    - Named routes simplify navigation by defining paths in MaterialApp.
2. Gestures
    - GestureDetector detects touch interactions like tap, swipe, and long press.
    - Common gestures:
        - Tap – Button clicks.
        - Swipe – Navigation.
        - Long Press – Extra options.

Implementation in Our Code

1. Navigation & Routing:
    - Defined named routes (/, /signup, /home, /project_details, /add_project).
    - Used Navigator.pushNamed() and Navigator.pushReplacementNamed() for transitions.
2. Gestures:
    - GestureDetector detects horizontal swipes to switch tabs.
    - Implemented onHorizontalDragEnd for navigation between sections.

Code:

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

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

class GreenFundConnect extends StatelessWidget {
  const GreenFundConnect({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'GreenFund Connect',
      theme: ThemeData(
        primarySwatch: Colors.green,
        primaryColor: const Color(0xFF2E7D32), // Dark green
        secondaryHeaderColor: const Color(0xFF81C784), // Light green
        fontFamily: GoogleFonts.poppins().fontFamily,
```

```
      textTheme: TextTheme(
        headlineLarge: GoogleFonts.montserrat(
          fontWeight: FontWeight.bold,
          color: const Color(0xFF2E7D32),
        ),
        titleLarge: GoogleFonts.poppins(
          fontWeight: FontWeight.w600,
          color: Colors.black87,
        ),
        bodyLarge: GoogleFonts.poppins(
          color: Colors.black87,
        ),
        bodyMedium: GoogleFonts.poppins(
          color: Colors.black87,
        ),
      ),
      elevatedButtonTheme: ElevatedButtonThemeData(
        style: ElevatedButton.styleFrom(
          backgroundColor: const Color(0xFF2E7D32),
          foregroundColor: Colors.white,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(10),
          ),
          padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 12),
        ),
      ),
    ),
    // Define named routes
    routes: {
      '/': (context) => const LoginScreen(),
      '/signup': (context) => const SignupScreen(),
      '/home': (context) => const HomeScreen(),
      '/project_details': (context) => const ProjectDetailsScreen(),
      '/add_project': (context) => const AddProjectScreen(),
    },
    // Initial route
    initialRoute: '/',
  );
 }
}

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

```
 @override
 _HomeScreenState createState() => _HomeScreenState();
}

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

 final List<Widget> _screens = [
  const ProjectsScreen(),
  const MyInvestmentsScreen(),
  const ProfileScreen(),
 ];

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

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: Row(
     children: [
      Image.asset(
       'assets/images/logo.png',
       height: 32,
      ),
      const SizedBox(width: 12),
      Text(
       'GreenFund Connect',
       style: GoogleFonts.montserrat(
        fontWeight: FontWeight.bold,
        color: Colors.green,
       ),
      ),
     ],
    ),
    actions: [
     IconButton(
      icon: const Icon(Icons.notifications_outlined),
      onPressed: () {
       // Notifications action
```

```
        },
      ),
      IconButton(
        icon: const Icon(Icons.search),
        onPressed: () {
          // Search action
        },
      ),
    ],
  ),
  body: GestureDetector(
    onHorizontalDragEnd: (details) {
      if (details.primaryVelocity! < 0 && _selectedIndex < 2) {
        _onItemTapped(_selectedIndex + 1);
      } else if (details.primaryVelocity! > 0 && _selectedIndex > 0) {
        _onItemTapped(_selectedIndex - 1);
      }
    },
    child: _screens[_selectedIndex],
  ),
  floatingActionButton: FloatingActionButton(
    backgroundColor: Theme.of(context).primaryColor,
    onPressed: () {
      Navigator.pushNamed(context, '/add_project');
    },
    child: const Icon(Icons.add),
    tooltip: 'Add new project',
  ),
  bottomNavigationBar: BottomNavigationBar(
    currentIndex: _selectedIndex,
    onTap: _onItemTapped,
    selectedItemColor: Theme.of(context).primaryColor,
    unselectedItemColor: Colors.grey,
    items: const [
      BottomNavigationBarItem(
        icon: Icon(Icons.home_outlined),
        activeIcon: Icon(Icons.home),
        label: 'Projects',
      ),
      BottomNavigationBarItem(
        icon: Icon(Icons.account_balance_wallet_outlined),
        activeIcon: Icon(Icons.account_balance_wallet),
        label: 'My Investments',
      ),
```

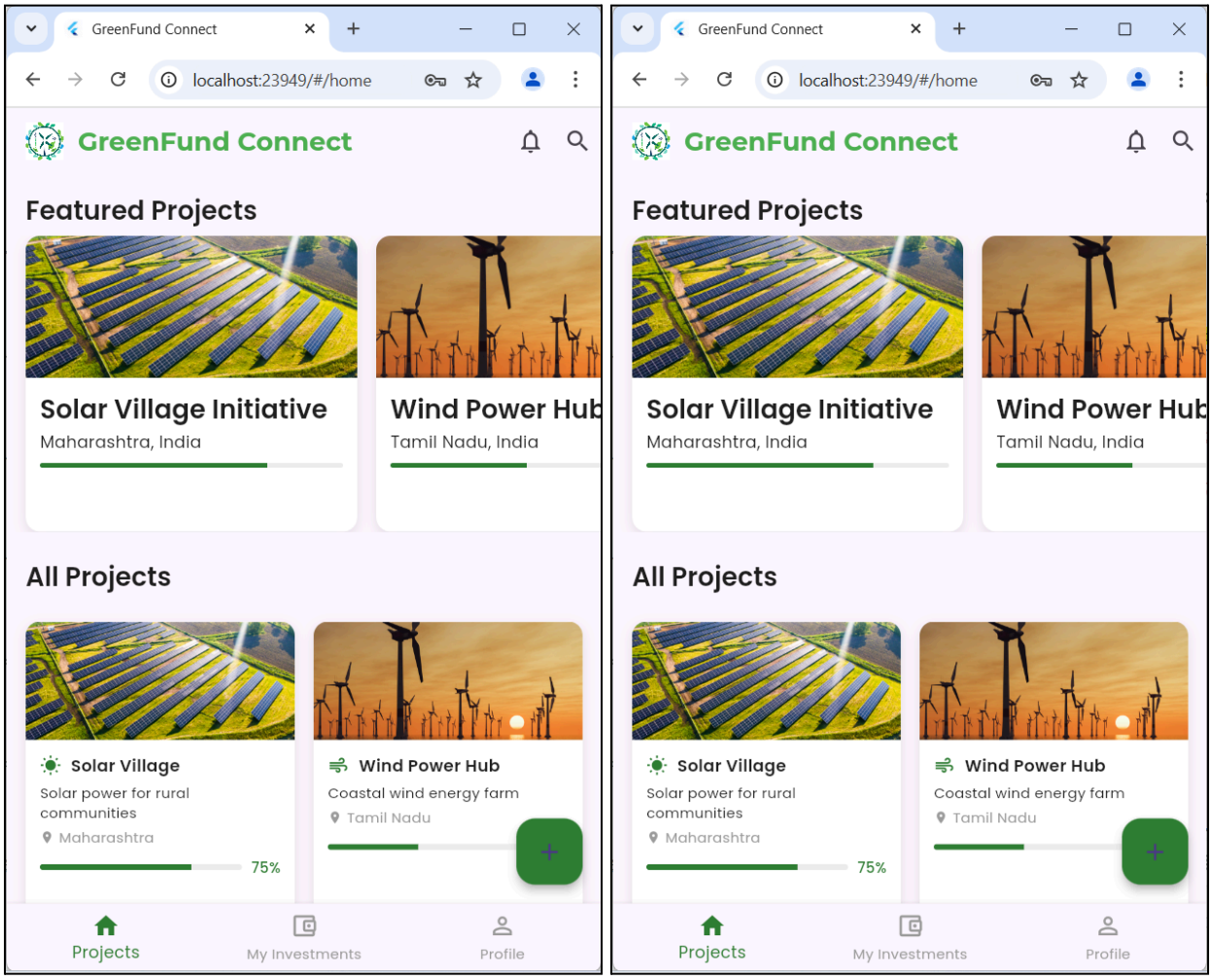
```
        BottomNavigationBarItem(
          icon: Icon(Icons.person_outline),
          activeIcon: Icon(Icons.person),
          label: 'Profile',
        ),
      ],
    ),
  );
 }
}

class ProjectsScreen extends StatelessWidget {
 const ProjectsScreen({Key? key}) : super(key: key);

 @override
 Widget build(BuildContext context) {
  return Column(
   crossAxisAlignment: CrossAxisAlignment.start,
   children: [
    Padding(
     padding: const EdgeInsets.fromLTRB(16, 16, 16, 8),
     child: Text(
      'Featured Projects',
      style: Theme.of(context).textTheme.titleLarge,
     ),
    ),
    SizedBox(
     height: 250,
     child: ListView(
      scrollDirection: Axis.horizontal,
      padding: const EdgeInsets.symmetric(horizontal: 16),
      children: [
       FeaturedProjectCard(
        title: 'Solar Village Initiative',
        location: 'Maharashtra, India',
        fundingPercentage: 75,
        imageAsset: 'assets/images/solar_village.png',
        onTap: () {
         Navigator.pushNamed(context, '/project_details');
        },
       ),
       FeaturedProjectCard(
        title: 'Wind Power Hub',
        location: 'Tamil Nadu, India',
```

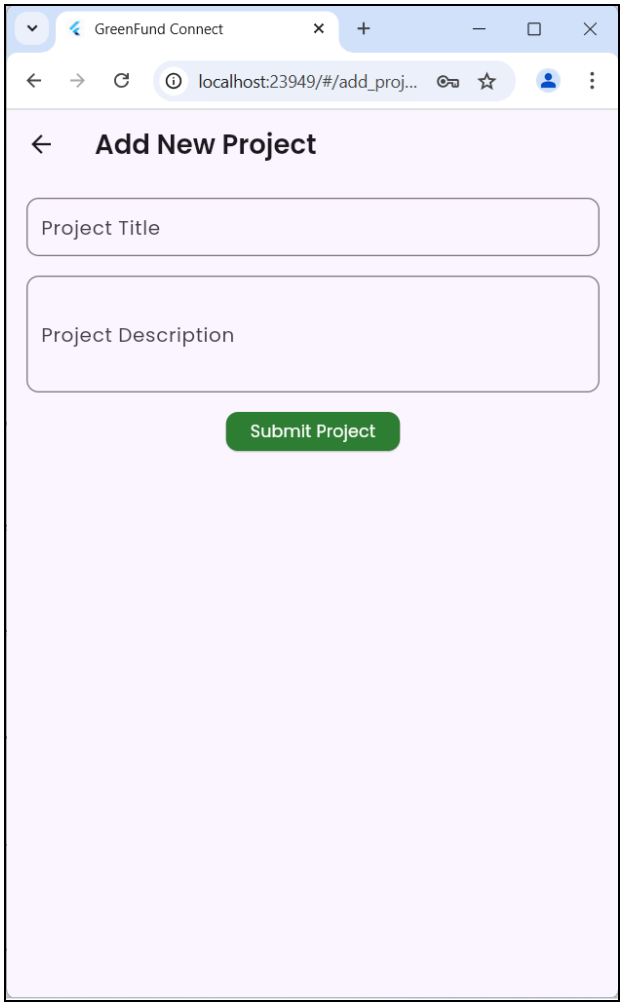
```
            fundingPercentage: 45,
            imageAsset: 'assets/images/wind_power.png',
            onTap: () {
              Navigator.pushNamed(context, '/project_details');
            },
          ),
          FeaturedProjectCard(
            title: 'Community Hydro Plant',
            location: 'Himachal Pradesh, India',
            fundingPercentage: 60,
            imageAsset: 'assets/images/hydro_plant.png',
            onTap: () {
              Navigator.pushNamed(context, '/project_details');
            },
          ),
        ],
      ),
    ),
    Padding(
      padding: const EdgeInsets.fromLTRB(16, 24, 16, 8),
      child: Text(
        'All Projects',
        style: Theme.of(context).textTheme.titleLarge,
      ),
    ),
    Expanded(child: ProjectGrid()),
  ],
 );
 }
}

class AddProjectScreen extends StatelessWidget {
 const AddProjectScreen({Key? key}) : super(key: key);

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(title: const Text('Add New Project')),
   body: Padding(
    padding: const EdgeInsets.all(16.0),
    child: Column(
     children: [
      TextField(
       decoration: InputDecoration(
```

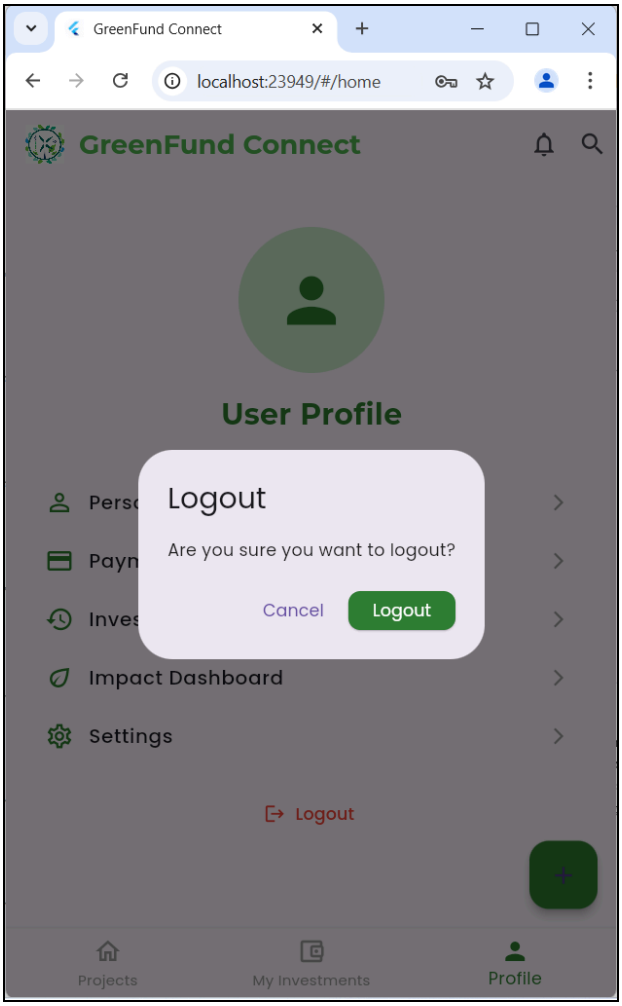
```
            labelText: 'Project Title',
            border: OutlineInputBorder(
              borderRadius: BorderRadius.circular(10),
            ),
          ),
        ),
        const SizedBox(height: 16),
        TextField(
          decoration: InputDecoration(
            labelText: 'Project Description',
            border: OutlineInputBorder(
              borderRadius: BorderRadius.circular(10),
            ),
          ),
          maxLines: 3,
        ),
        const SizedBox(height: 16),
        ElevatedButton(
          onPressed: () {
            // Add project logic
            Navigator.pop(context);
          },
          child: const Text('Submit Project'),
        ),
      ],
    ),
  ),
 );
 }
}

class ProjectDetailsScreen extends StatelessWidget {
 const ProjectDetailsScreen({Key? key}) : super(key: key);

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(title: const Text('Project Details')),
   body: Center(
    child: Column(
     mainAxisAlignment: MainAxisAlignment.center,
     children: [
      const Text(
       'Details about the selected project',
```

```
            style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
          ),
          const SizedBox(height: 20),
          ElevatedButton(
            onPressed: () => Navigator.pop(context),
            child: const Text('Go Back'),
          ),
        ],
      ),
    ),
  );
 }
}
```

Screenshot:







Conclusion

In this experiment, we successfully implemented navigation, routing, and gesture handling in our Flutter app, enabling smooth screen transitions and interactive user interactions. Initially, we faced issues like incorrect route mapping and unresponsive gestures, which we resolved by properly defining named routes and debugging gesture detection using GestureDetector.

Name : Naikwadi Yash Shivdas

MPL Practical 06

Aim: To integrate Firebase Authentication in a Flutter app.

Theory:

Firebase Authentication in Our Code

- User Registration – Users sign up with email & password, and a verification email is sent.
- Email Verification – Users must verify their email before logging in.
- User Login – Users can log in only after verifying their email.
- Google Sign-In – Users can log in using their Google account.
- Password Reset – Users receive a reset link via email if they forget their password.
- Logout – Users can securely log out.

Firebase Integration Steps

- Configured Firebase Project and enabled authentication.
- Initialized Firebase in Flutter and added required dependencies.
- Implemented Firebase Authentication in auth_service.dart for sign-up, login, password reset, and logout.

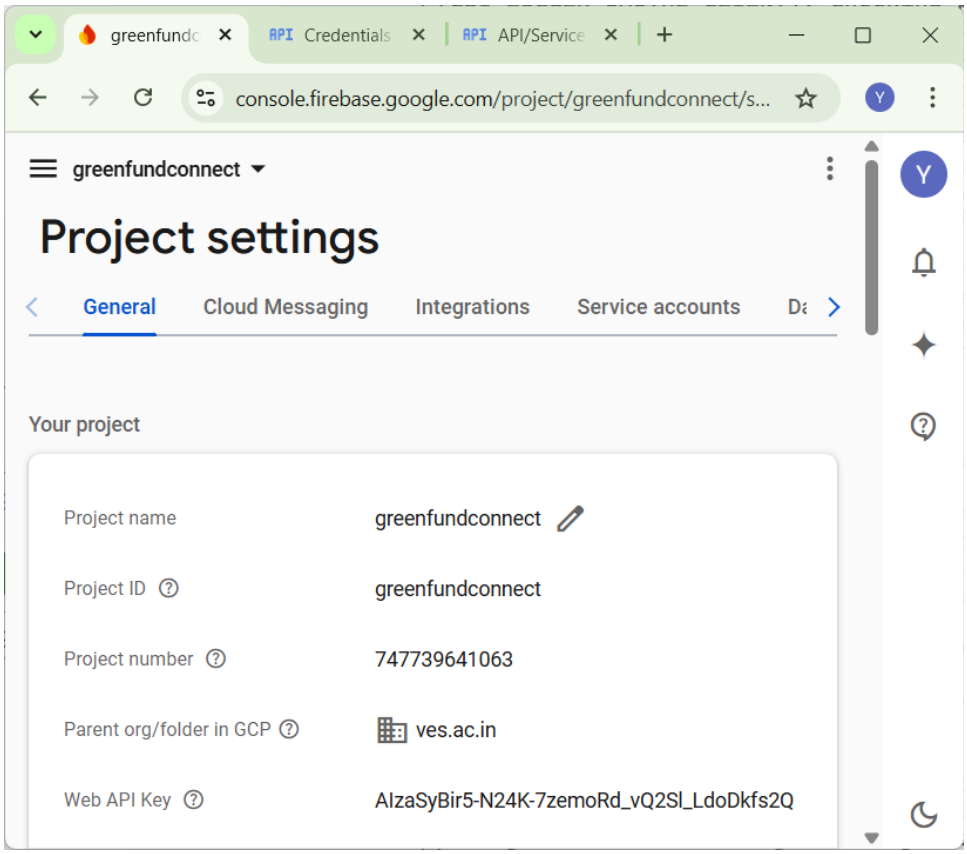
Code:

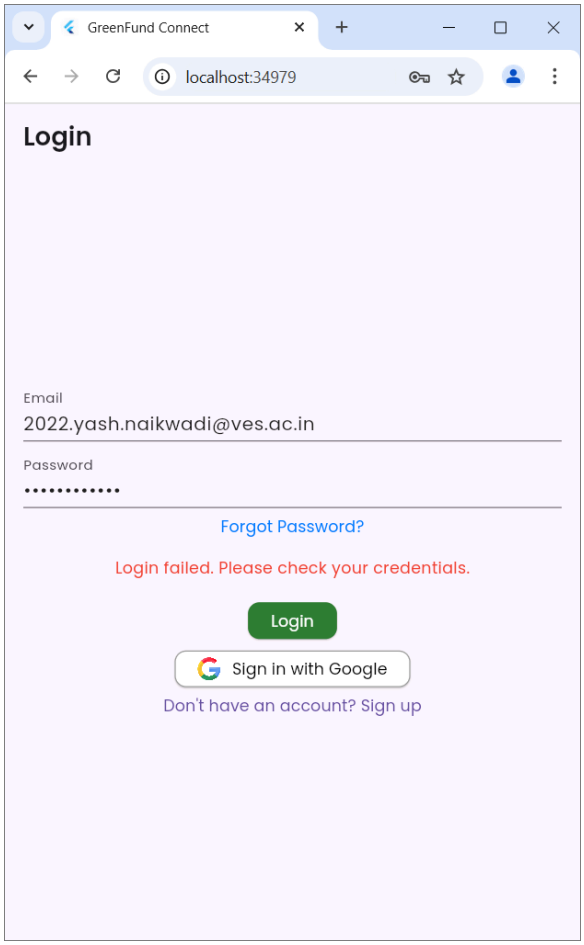
```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:google_sign_in/google_sign_in.dart';

class AuthService {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final GoogleSignIn _googleSignIn = GoogleSignIn(
    clientId: "906491462662-tlb834jjlnn3beter7oapo8d0j1hbvch.apps.googleusercontent.com",
  );

  // Sign Up with Email & Password and Send Verification Email
  Future<User?> signUp(String email, String password) async {
    try {
      UserCredential userCredential = await _auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );

      User? user = userCredential.user;

      if (user != null) {
        await user.sendEmailVerification(); // Send verification email
      }
```

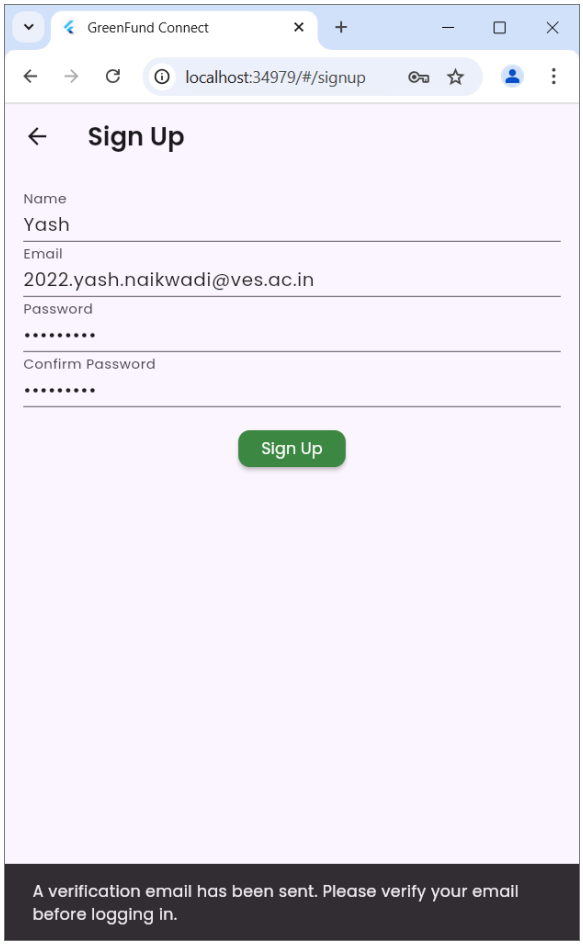
```
    return user;
  } catch (e) {
    print("Sign Up Error: $e");
    return null;
  }
}

// Check if email is verified
Future<bool> isEmailVerified() async {
  User? user = _auth.currentUser;
  await user?.reload(); // Refresh user data
  return user?.emailVerified ?? false;
}

// Login with Email & Password (Only if email is verified)
Future<User?> signIn(String email, String password) async {
  try {
    UserCredential userCredential = await _auth.signInWithEmailAndPassword(
      email: email,
      password: password,
    );

    User? user = userCredential.user;

    if (user != null && user.emailVerified) {
      return user;
    } else {
      print("Email not verified");
      return null;
    }
  } catch (e) {
    print("Login Error: $e");
    return null;
  }
}

// Google Sign-In
Future<User?> signInWithGoogle() async {
  try {
    final GoogleSignInAccount? googleUser = await _googleSignIn.signIn();
    if (googleUser == null) return null;

    final GoogleSignInAuthentication googleAuth = await googleUser.authentication;
    final AuthCredential credential = GoogleAuthProvider.credential(
```

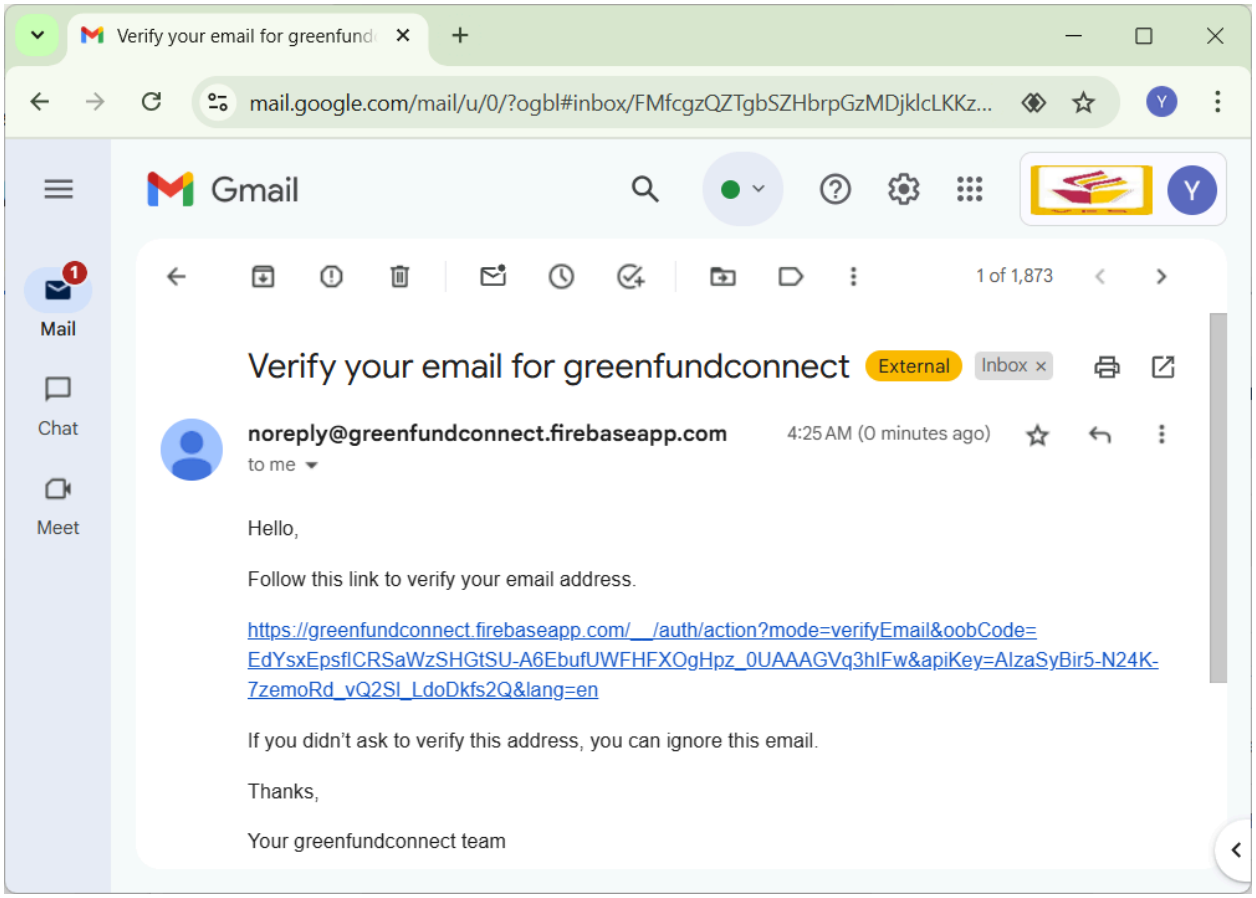
```
      accessToken: googleAuth.accessToken,
      idToken: googleAuth.idToken,
    );

    UserCredential userCredential = await _auth.signInWithCredential(credential);
    return userCredential.user;
  } catch (e) {
    print("Google Sign-In Error: $e");
    return null;
  }
 }

 // Sign Out
 Future<void> signOut() async {
  try {
    await _auth.signOut();
    await _googleSignIn.signOut();
  } catch (e) {
    print("Sign Out Error: $e");
  }
 }

 // Reset Password
 Future<bool> resetPassword(String email) async {
  try {
    await _auth.sendPasswordResetEmail(email: email);
    return true;
  } catch (e) {
    print("Password Reset Error: $e");
    return false;
  }
 }
}
```

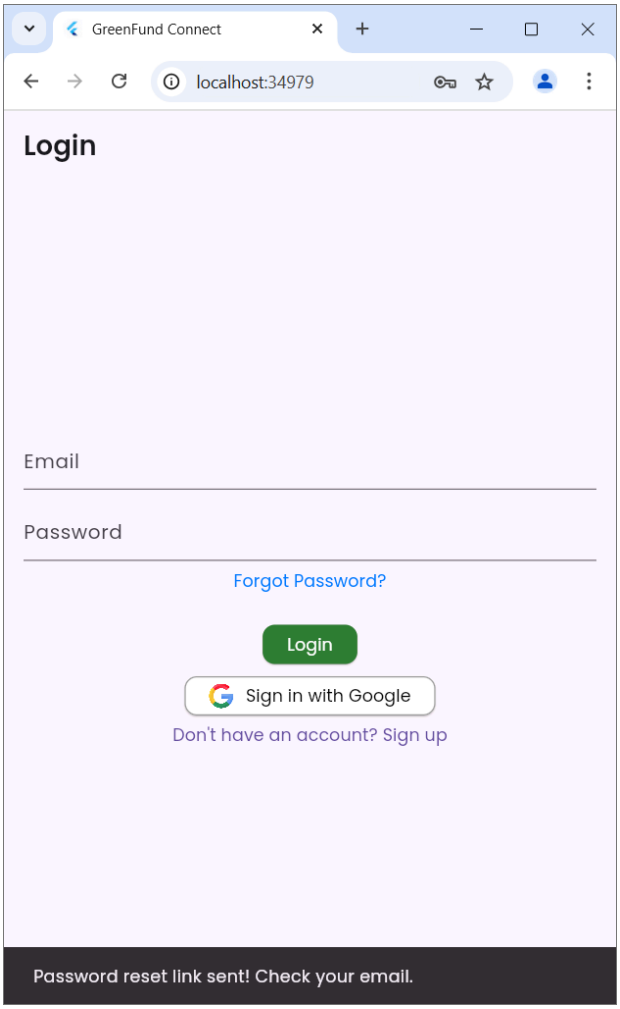
Screenshot:

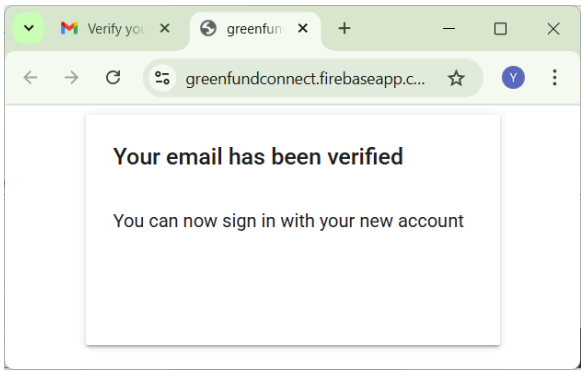

greenfundc
API Credentials
API API/Service
console.firebase.google.com/project/greenfundconnect/s...
greenfundconnect
Project settings
General
Cloud Messaging
Integrations
Service accounts
Your project
Project name
greenfundconnect
Project ID
greenfundconnect
Project number
747739641063
Parent org/folder in GCP
ves.ac.in
Web API Key
AIzaSyBir5-N24K-7zemoRd_vQ2Sl_LdoDkfs2Q

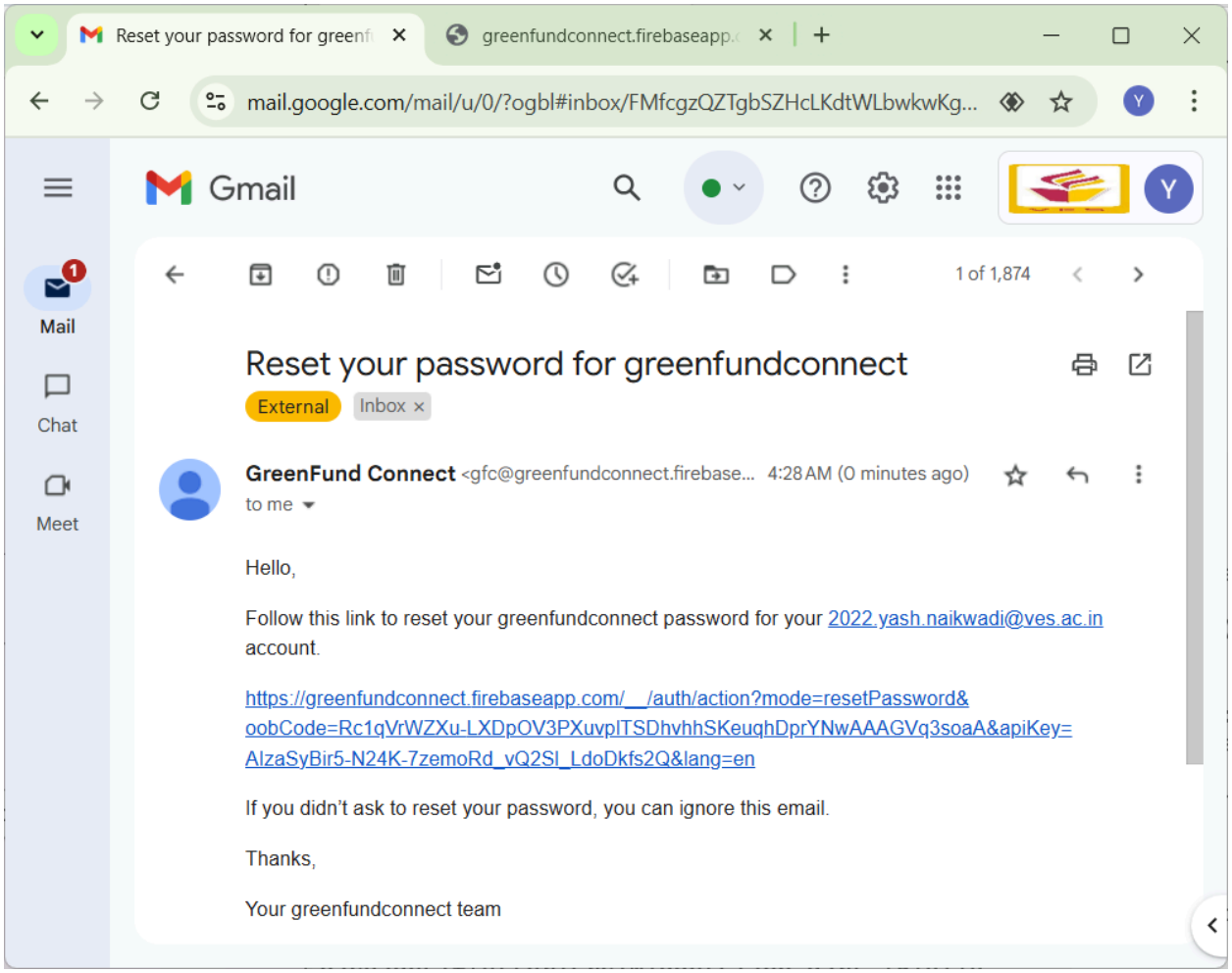

GreenFund Connect
localhost:34979
Login
Email
2022.yash.naikwadi@ves.ac.in
Password
Forgot Password?
Login failed. Please check your credentials.
Login
Sign in with Google
Don't have an account? Sign up

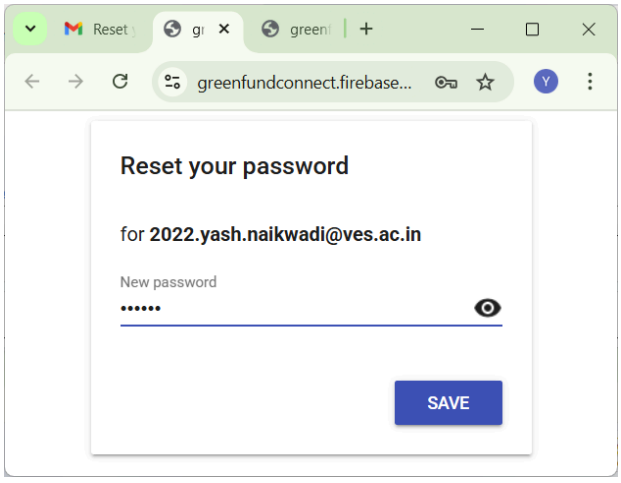

GreenFund Connect
localhost:34979/#/signup
Sign Up
Name
Yash
Email
2022.yash.naikwadi@ves.ac.in
Password
Confirm Password
Sign Up
A verification email has been sent. Please verify your email before logging in.

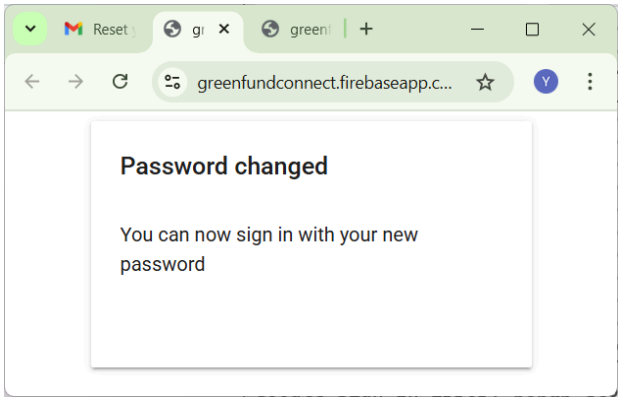

Verify your email for greenfund
mail.google.com/mail/u/0/?ogbl#inbox/FMfcgzQZTgbSZHbrpGzMDjklcLKKz...
Gmail
Mail
Chat
Meet
1 of 1,873
Verify your email for greenfundconnect
External
Inbox ×
noreply@greenfundconnect.firebaseapp.com
4:25 AM (0 minutes ago)
to me
Hello,
Follow this link to verify your email address.
https://greenfundconnect.firebaseapp.com/__/auth/action?mode=verifyEmail&oobCode=EdYsxEpsfICRSaWzSHGtSU-A6EbufUWFHFXOgHpz_0UAAAGVq3hIFw&apiKey=AIzaSyBir5-N24K-7zemoRd_vQ2SI_LdoDkfs2Q&lang=en
If you didn't ask to verify this address, you can ignore this email.
Thanks,
Your greenfundconnect team

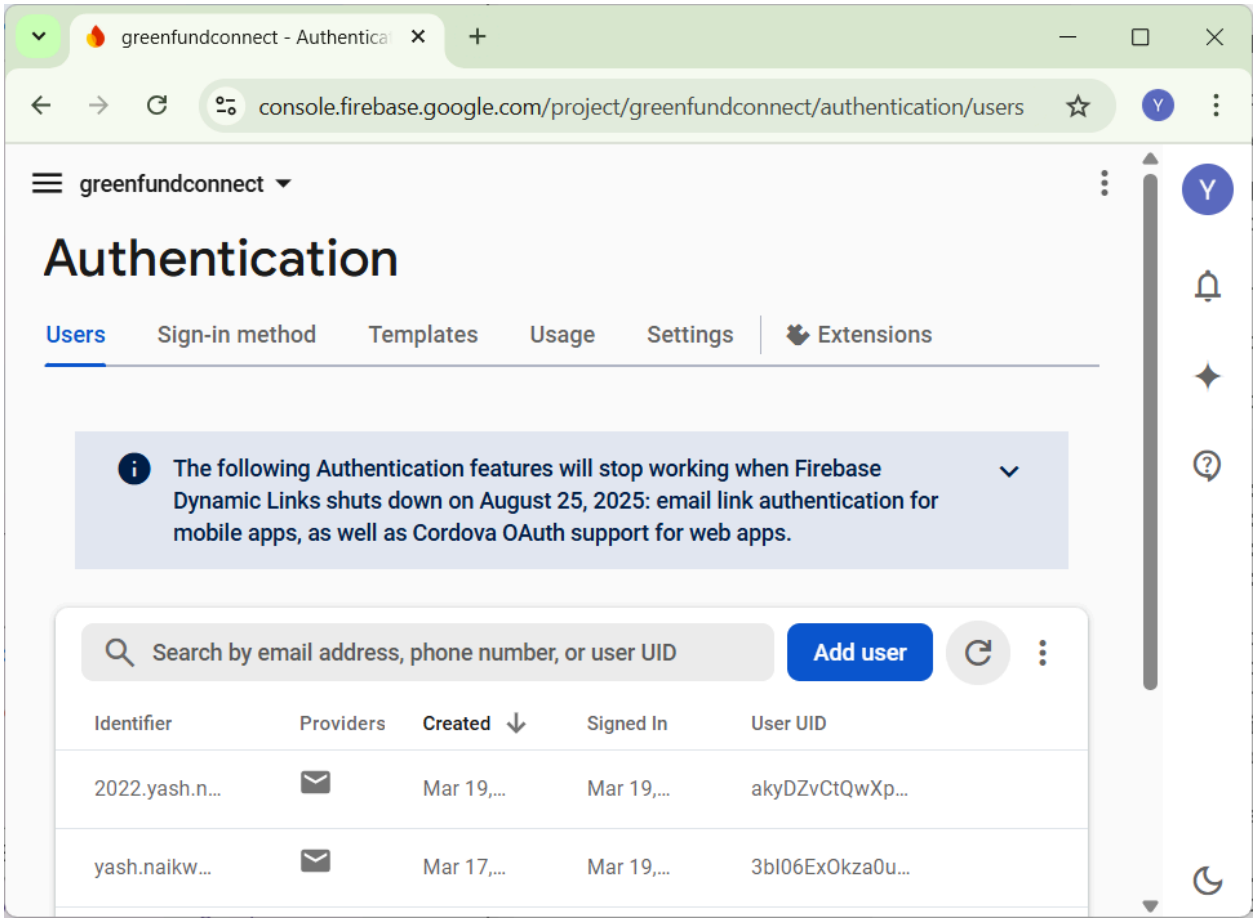

GreenFund Connect
localhost:34979
Login
Email
Password
Forgot Password?
Login
Sign in with Google
Don't have an account? Sign up
Password reset link sent! Check your email.

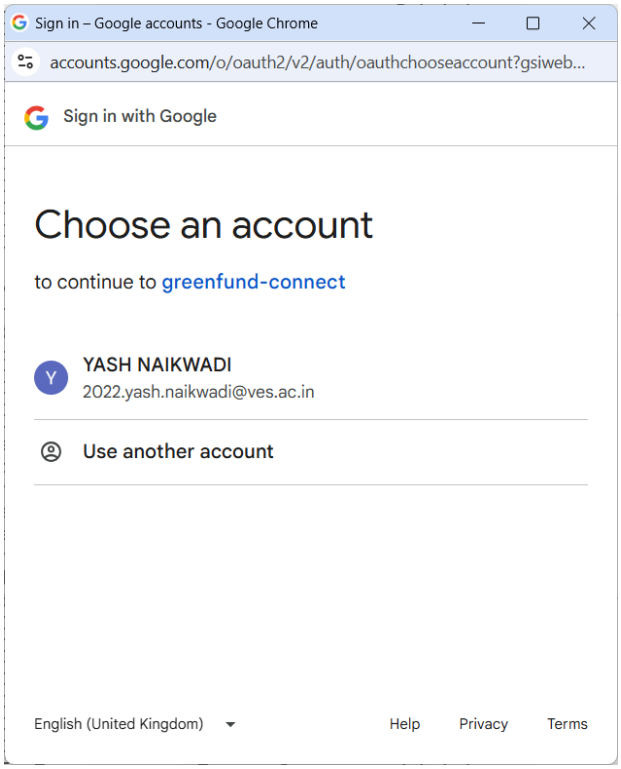

greenfundconnect.firebaseapp.c...
Your email has been verified
You can now sign in with your new account

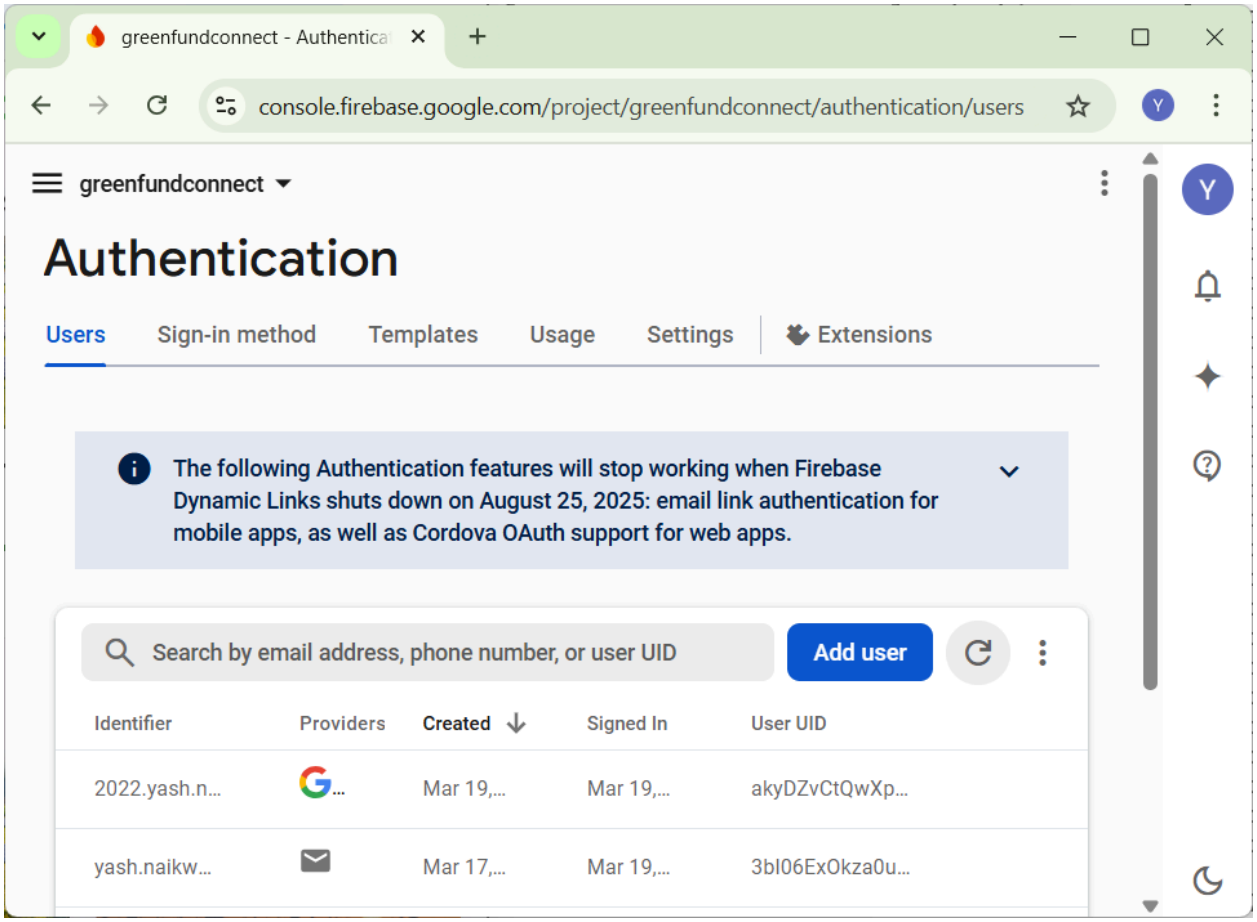

Reset your password for greenf
greenfundconnect.firebaseapp.
mail.google.com/mail/u/0/?ogbl#inbox/FMfcgzQZTgbSZHcLKdtWLbwkwKg...
Gmail
Mail
Chat
Meet
1 of 1,874
Reset your password for greenfundconnect
External
Inbox ×
GreenFund Connect <gfc@greenfundconnect.firebase... 4:28 AM (0 minutes ago)
to me
Hello,
Follow this link to reset your greenfundconnect password for your 2022.yash.naikwadi@ves.ac.in account.
https://greenfundconnect.firebaseapp.com/__/auth/action?mode=resetPassword&oobCode=Rc1qVrWZXu-LXDpOV3PXuvplTSDhvhhSKeuqhDprYNwAAAGVq3soaA&apiKey=AIzaSyBir5-N24K-7zemoRd_vQ2SI_LdoDkfs2Q&lang=en
If you didn't ask to reset your password, you can ignore this email.
Thanks,
Your greenfundconnect team


greenfundconnect.firebase...
Reset your password
for 2022.yash.naikwadi@ves.ac.in
New password
••••••
SAVE


greenfundconnect.firebaseapp.c...
Password changed
You can now sign in with your new password


greenfundconnect - Authentica
console.firebase.google.com/project/greenfundconnect/authentication/users
greenfundconnect
Authentication
Users
Sign-in method
Templates
Usage
Settings
Extensions
The following Authentication features will stop working when Firebase Dynamic Links shuts down on August 25, 2025: email link authentication for mobile apps, as well as Cordova OAuth support for web apps.
Search by email address, phone number, or user UID
Add user
Identifier
Providers
Created
Signed In
User UID
2022.yash.n...
Mar 19,...
Mar 19,...
akyDZvCtQwXp...
yash.naikw...
Mar 17,...
Mar 19,...
3bI06ExOkza0u...


Sign in – Google accounts - Google Chrome
accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount?gsiweb...
Sign in with Google
Choose an account
to continue to greenfund-connect
YASH NAIKWADI
2022.yash.naikwadi@ves.ac.in
Use another account
English (United Kingdom)
Help
Privacy
Terms



Conclusion

In GreenFund-Connect, we successfully integrated Firebase Authentication, implementing user sign-up, email verification, login, Google Sign-In, password reset, and logout. During development, we faced issues like Google Sign-In popup closing unexpectedly and email verification delays, which we resolved by correcting OAuth credentials, enabling the People API, and ensuring proper Firebase authentication settings.

Name : Naikwadi Yash Shivdas

MPL Practical 07

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

Theory:

In Progressive Web Apps (PWAs), a Web App Manifest is a JSON file that provides important information about the web application. This file allows the browser to understand how the app should behave when installed on a user’s device. It plays a key role in enabling the "Add to Homescreen" feature, which makes the website feel like a native mobile app.
The manifest file typically includes:

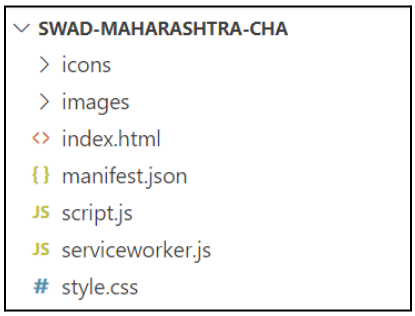
- The name and short name of the app
- The start URL (i.e., where the app opens from the home screen)
- The display mode (e.g., standalone, fullscreen)
- Theme color and background color
- App icons in various sizes for mobile and desktop compatibility
- A description to tell users what the app does

What We Implemented
In our eCommerce website "Swad Maharashtra Cha", we created a manifest.json file to define the essential metadata for our PWA. Here's what we included:

- App Name: "Swad Maharashtra Cha" to represent our brand offering authentic Maharashtrian food.
- Short Name: "SwadMaha" which appears under the app icon when installed.
- Start URL: "index.html" so that the app always starts from the homepage.
- Display Mode: "standalone" to make it look like a native app (no browser UI).
- Theme Color: #b22222 matching our brand's rich red palette.
- Background Color: #ffffff for a clean, neutral background when loading.
- Icons: We added two icons (192x192 and 512x512) so that the app icon looks good on all screen sizes.
- Description: To describe the purpose of our app — delivering traditional Maharashtrian spices, sweets, and snacks.

By linking the manifest in our index.html and ensuring proper metadata, our website now supports the "Add to Homescreen" feature on supported browsers. When a user visits our site on mobile, they can install it to their home screen and enjoy an app-like experience.

Folder Structure:



Code:

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <!-- PWA Theme Color -->
    <meta name="theme-color" content="#b22222" />

    <!-- SEO Meta Tags -->
    <meta
      name="description"
      content="Authentic Maharashtrian Spices & Food – Buy famous spices, masalas, and traditional items like Puran Poli, Bhakarwadi, Modaks, and Chitale Bandhu snacks online."
    />
    <meta
      name="keywords"
      content="Maharashtrian food, Goda masala, Bhakarwadi, Modaks, Puran Poli, Chitale Bandhu snacks, Malvani masala, online spice store"
    />
    <meta name="author" content="Swad Maharashtra Cha" />

    <title>Swad Maharashtra Cha - Authentic Maharashtrian Food</title>

    <!-- Manifest File for PWA -->
    <link rel="manifest" href="manifest.json" />

    <!-- Stylesheet -->
    <link rel="stylesheet" href="style.css" />

    <!-- Google Fonts -->
    <link

href="https://fonts.googleapis.com/css2?family=Playfair+Display:wght@400;700&family=Poppins:wght@300;400;500;600&display=swap"
      rel="stylesheet"
    />
  </head>
  <body>

    <script>
      if ("serviceWorker" in navigator) {
        navigator.serviceWorker
          .register("serviceworker.js")
          .then(() => console.log("Service Worker Registered"))
          .catch((error) =>
```

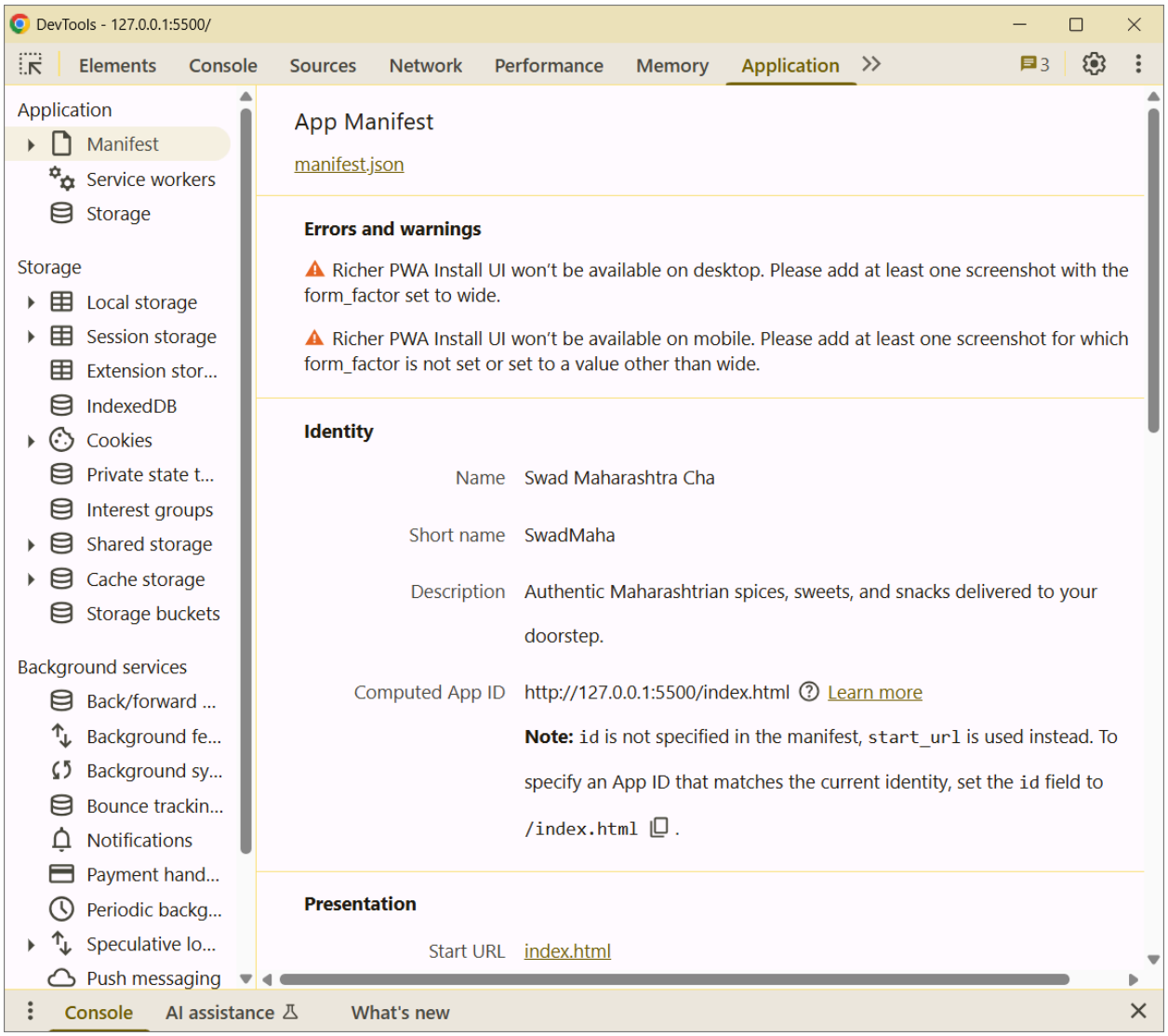
```
          console.log("Service Worker Registration Failed", error)
        );
    }
  </script>

  <script src="script.js"></script>
 </body>
</html>
```

manifest.json

```
{
 "name": "Swad Maharashtra Cha",
 "short_name": "SwadMaha",
 "start_url": "index.html",
 "display": "standalone",
 "background_color": "#ffffff",
 "theme_color": "#b22222",
 "description": "Authentic Maharashtrian spices, sweets, and snacks delivered to your doorstep.",
 "icons": [
  {
   "src": "icons/icon-192x192.png",
   "sizes": "192x192",
   "type": "image/png"
  },
  {
   "src": "icons/icon-512x512.png",
   "sizes": "512x512",
   "type": "image/png"
  }
 ]
}
```

Screenshot:


Swad Maharashtra Cha - Auther
127.0.0.1:5500
Swad Maharashtra Cha
Home
Spices
Sweets
Snacks
Contact
0
Authentic Taste of Maharashtra IN
Explore our curated selection of traditional spices, sweets, and snacks shipped straight from local Maharashtrian kitchens to your doorstep.
Start Tasting


DevTools - 127.0.0.1:5500/
Elements
Console
Sources
Network
Performance
Memory
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension stor...
IndexedDB
Cookies
Private state t...
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward ...
Background fe...
Background sy...
Bounce trackin...
Notifications
Payment hand...
Periodic backg...
Speculative lo...
Push messaging
App Manifest
manifest.json
Errors and warnings
Richer PWA Install UI won't be available on desktop. Please add at least one screenshot with the form_factor set to wide.
Richer PWA Install UI won't be available on mobile. Please add at least one screenshot for which form_factor is not set or set to a value other than wide.
Identity
Name
Swad Maharashtra Cha
Short name
SwadMaha
Description
Authentic Maharashtrian spices, sweets, and snacks delivered to your doorstep.
Computed App ID
http://127.0.0.1:5500/index.html
Learn more
Note: id is not specified in the manifest, start_url is used instead. To specify an App ID that matches the current identity, set the id field to /index.html .
Presentation
Start URL
index.html
Console
AI assistance
What's new

DevTools - 127.0.0.1:5500/

Elements Console Sources Network Performance Memory Application

Application
- Manifest
- Service workers
- Storage

Storage
- Local storage
- Session storage
- Extension stor...
- IndexedDB
- Cookies
- Private state t...
- Interest groups
- Shared storage
- Cache storage
- Storage buckets

Background services
- Back/forward ...
- Background fe...
- Background sy...
- Bounce trackin...
- Notifications
- Payment hand...
- Periodic backg...
- Speculative lo...
- Push messaging

Theme color #b22222

Background color #ffffff

Orientation

Display standalone

Protocol Handlers

Define protocol handlers in the manifest to register your app as a handler for custom protocols when your app is installed.

Need help? Read URL protocol handler registration for PWAs.

Icons

Show only the minimum safe area for maskable icons

Need help? Read the documentation on maskable icons.

192×192px

image/png

Console AI assistance What's new

Swad Maharashtra Cha - Authe

127.0.0.1:5500

Swad Mal Cha

Contact 0

Install app

Swad Maharashtra Cha

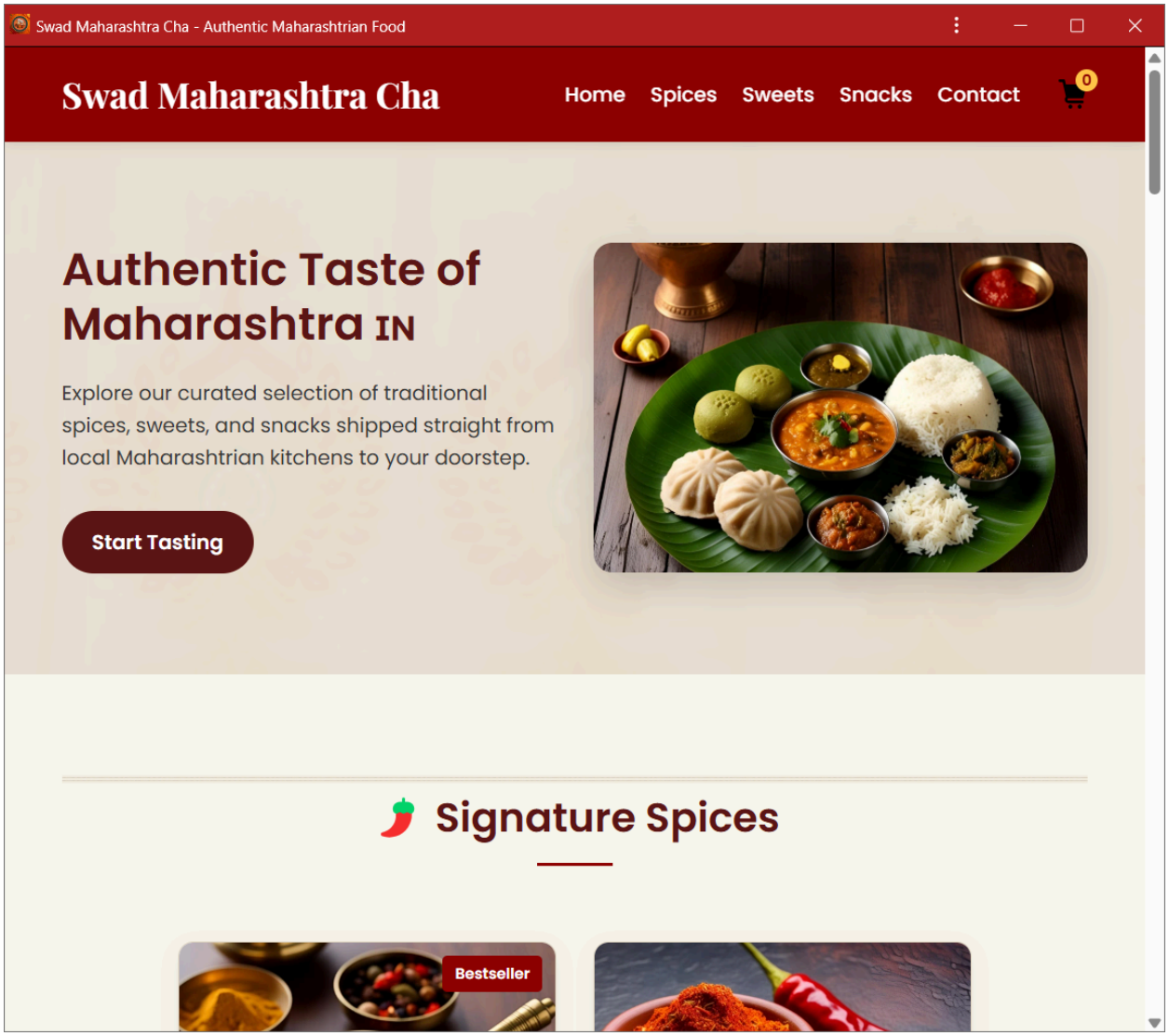
127.0.0.1:5500

Install Cancel

Authentic Taste of Maharashtra IN

Explore our curated selection of traditional spices, sweets, and snacks shipped straight from local Maharashtrian kitchens to your doorstep.

Start Tasting



Conclusion:

In this experiment, we successfully implemented the manifest.json file for our Swad Maharashtra Cha PWA, enabling the "Add to Homescreen" feature with proper app metadata and icons. Initially, we faced an error where the app was not installable due to a missing icon path, but we resolved it by correctly placing the icons in the specified folder and updating the manifest.

Name : Naikwadi Yash Shivdas

MPL Practical 08

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:

A Service Worker is a background script that runs independently of the web page and plays a key role in making a website function like a Progressive Web App (PWA). It allows the app to cache files, serve content offline, and load faster even in low or no internet connection.

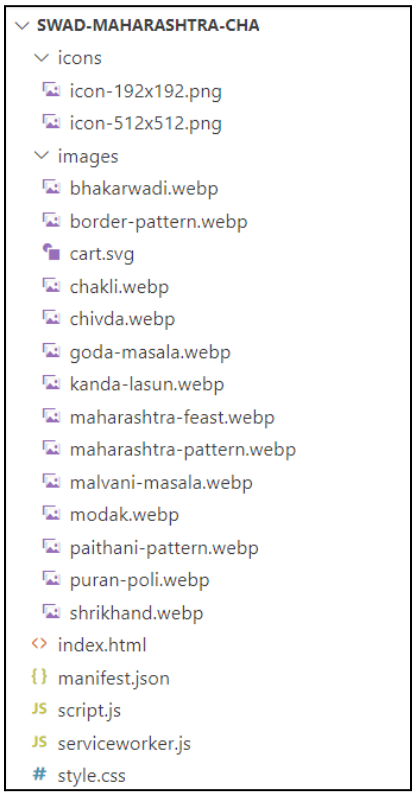
In this experiment, we learned how to register, install, and activate a service worker. The service worker listens for events like "install" and "activate" to manage caching of important files such as HTML, CSS, JavaScript, images, icons, and the manifest. Once installed, it stores these files in a cache so that the app can work offline and load faster on future visits.

What We Implemented

For our project "Swad Maharashtra Cha", we created a serviceworker.js file that caches all the necessary files of the website including product images, stylesheets, scripts, and icons. In the install event, the files are added to the cache. During the activate event, old caches are cleared to avoid unnecessary storage. In our index.html, we registered the service worker to ensure it runs when the site loads.

With this setup, our PWA can now work offline using cached data, which improves user experience and performance.

Folder Structure



Code:

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <!-- PWA Theme Color -->
    <meta name="theme-color" content="#b22222" />

    <!-- SEO Meta Tags -->
    <meta
      name="description"
      content="Authentic Maharashtrian Spices & Food – Buy famous spices, masalas, and traditional items like Puran Poli, Bhakarwadi, Modaks, and Chitale Bandhu snacks online."
    />
    <meta
      name="keywords"
      content="Maharashtrian food, Goda masala, Bhakarwadi, Modaks, Puran Poli, Chitale Bandhu snacks, Malvani masala, online spice store"
    />
    <meta name="author" content="Swad Maharashtra Cha" />

    <title>Swad Maharashtra Cha - Authentic Maharashtrian Food</title>

    <!-- Manifest File for PWA -->
    <link rel="manifest" href="manifest.json" />

    <!-- Stylesheet -->
    <link rel="stylesheet" href="style.css" />

    <!-- Google Fonts -->
    <link

href="https://fonts.googleapis.com/css2?family=Playfair+Display:wght@400;700&family=Poppins:wght@300;400;500;600&display=swap"
      rel="stylesheet"
    />
  </head>
  <body>

    <script>
      if ("serviceWorker" in navigator) {
        window.addEventListener("load", () => {
          navigator.serviceWorker
            .register("/serviceworker.js")
            .then((registration) => {
```

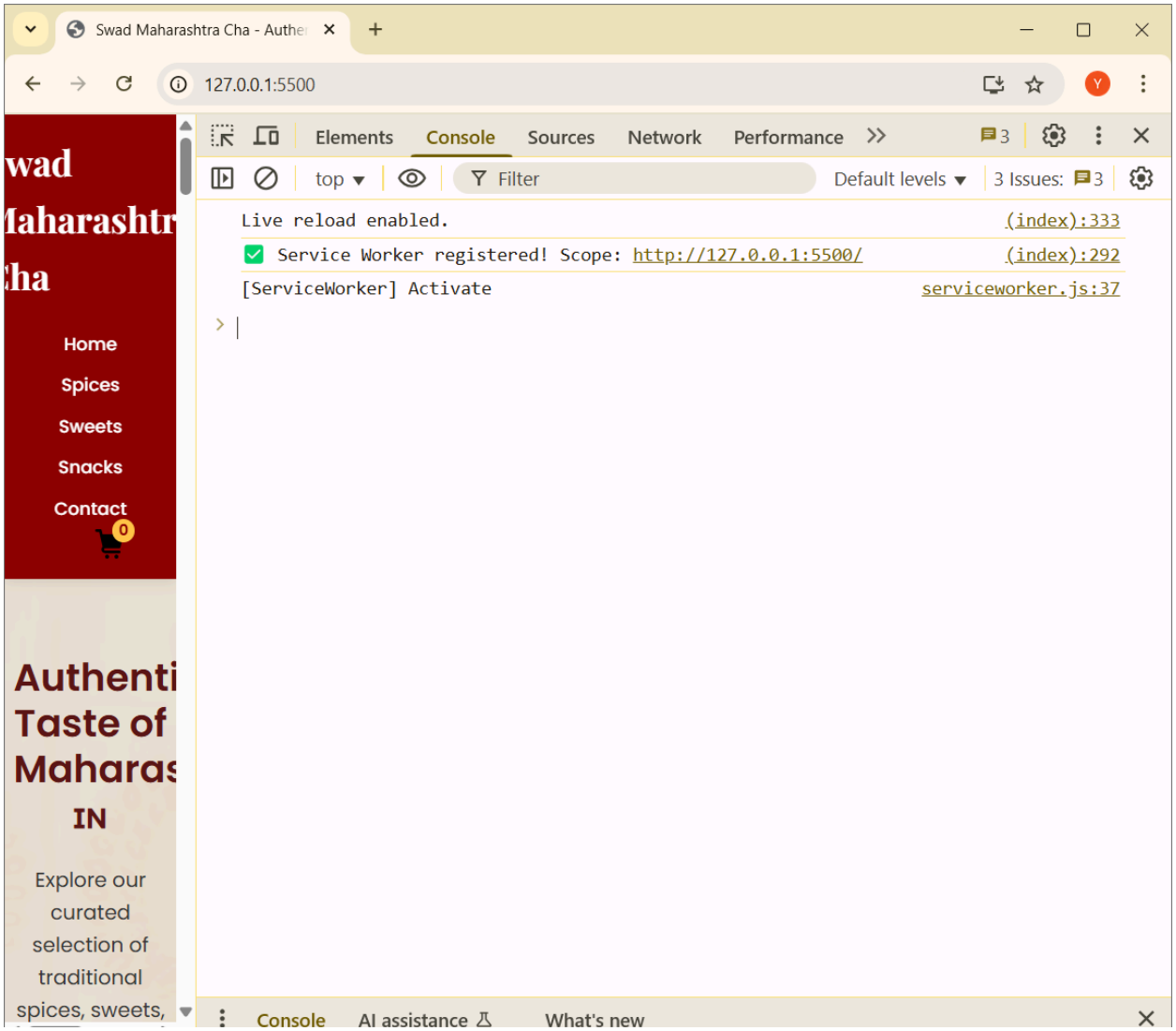
```
            console.log(
              "✅ Service Worker registered! Scope:",
              registration.scope
            );
          })
          .catch((error) => {
            console.log("❌ Service Worker registration failed:", error);
          });
      });
    }
  </script>

  <script src="script.js"></script>
 </body>
</html>
```

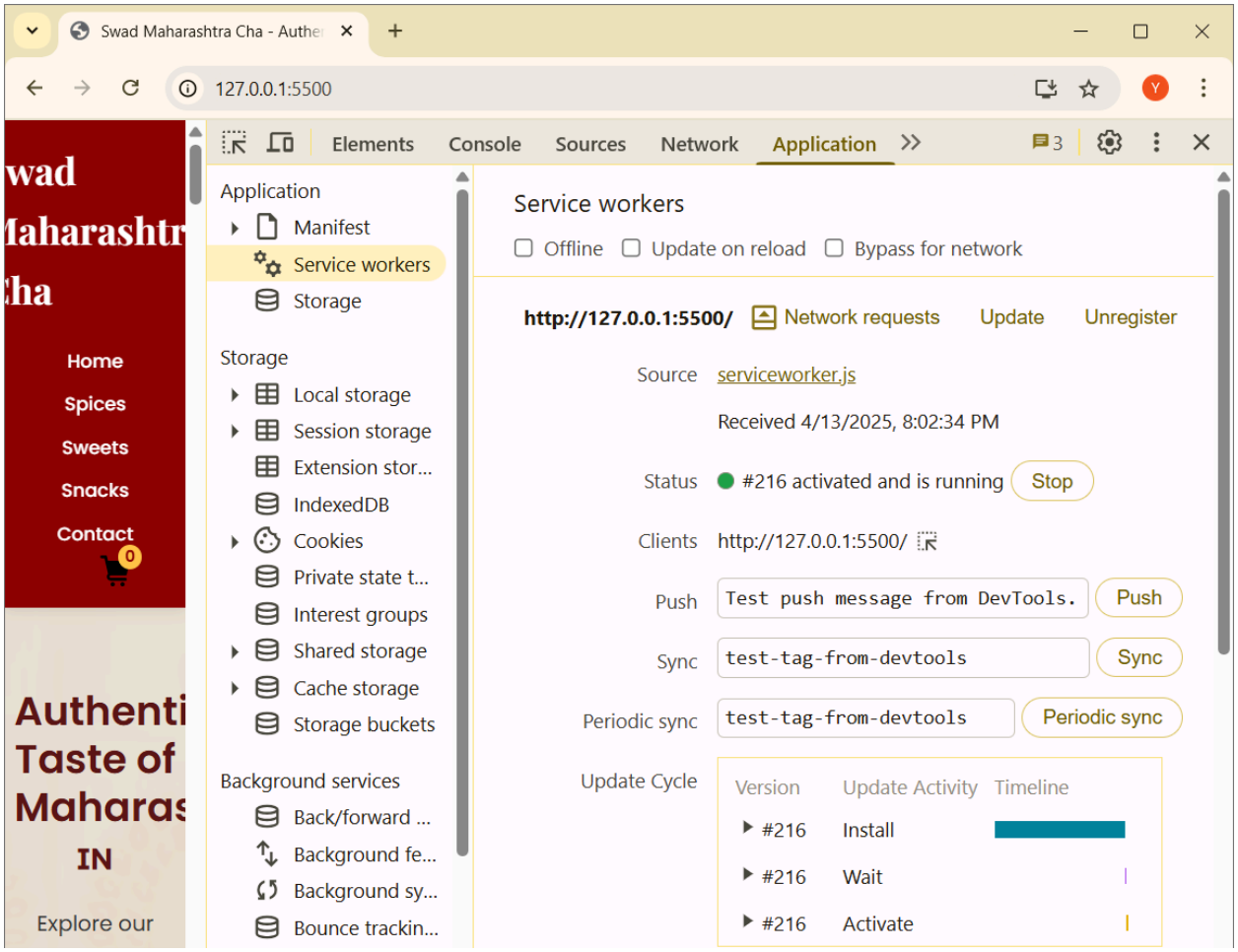
serviceworker.js

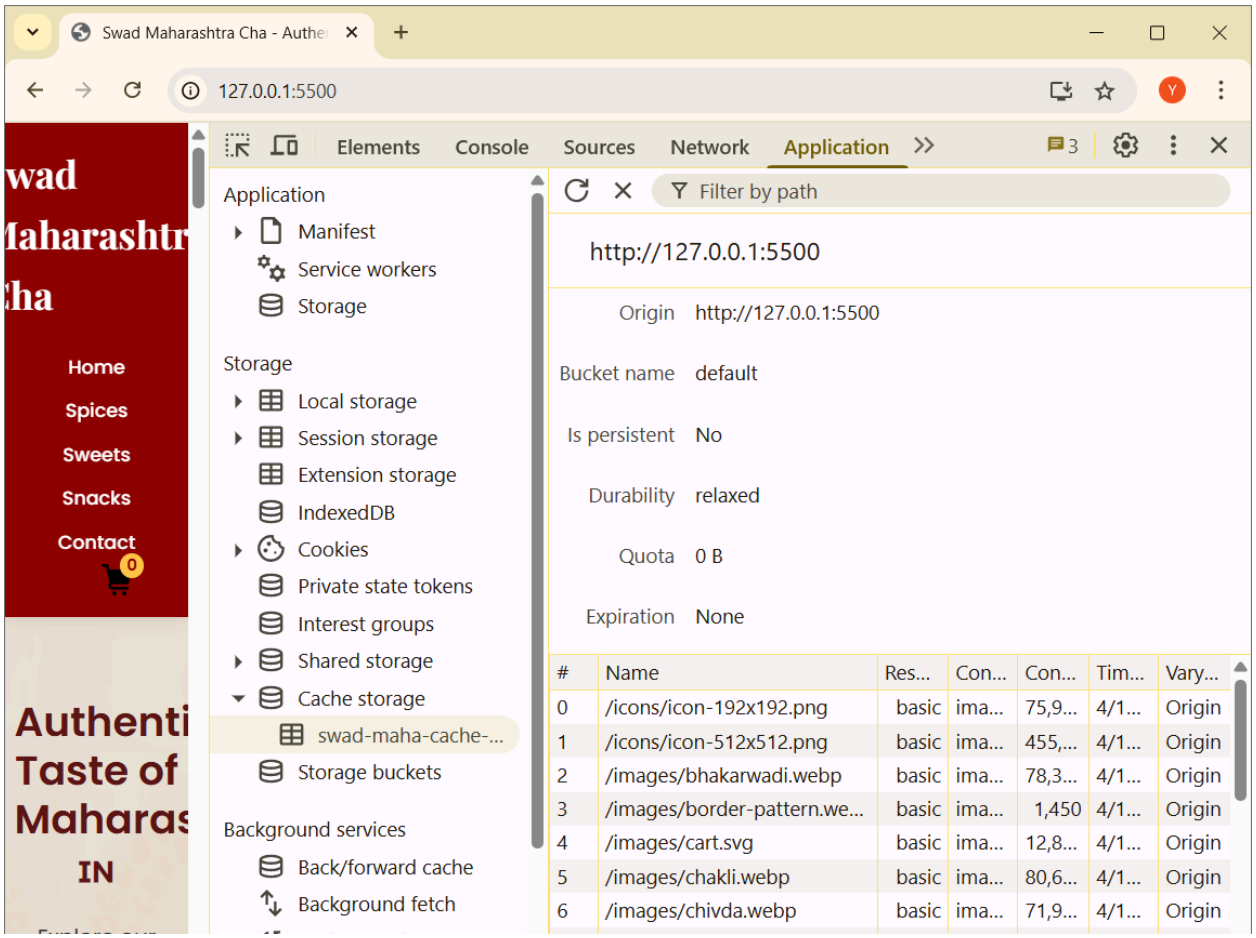
```
const CACHE_NAME = "swad-maha-cache-v1";
const FILES_TO_CACHE = [
  "index.html",
  "style.css",
  "script.js",
  "serviceworker.js",
  "manifest.json",
  "images/maharashtra-feast.webp",
  "images/cart.svg",
  "images/goda-masala.webp",
  "images/malvani-masala.webp",
  "images/kanda-lasun.webp",
  "images/puran-poli.webp",
  "images/modak.webp",
  "images/shrikhand.webp",
  "images/bhakarwadi.webp",
  "images/chivda.webp",
  "images/chakli.webp",
  "images/border-pattern.webp",
  "images/maharashtra-pattern.webp",
  "images/paithani-pattern.webp",
  "icons/icon-192x192.png",
  "icons/icon-512x512.png",
];

// Install Event
self.addEventListener("install", (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => {
      return cache.addAll(FILES_TO_CACHE);
    })
  );
```

```
});

// Activate Event
self.addEventListener("activate", (event) => {
  console.log("[ServiceWorker] Activate");
  event.waitUntil(
    caches.keys().then((keyList) =>
      Promise.all(
        keyList.map((key) => {
          if (key !== CACHE_NAME) {
            console.log("[ServiceWorker] Removing old cache", key);
            return caches.delete(key);
          }
        })
      )
    )
  );
  return self.clients.claim();
});
```

Screenshot:




Swad Maharashtra Cha - Auther
127.0.0.1:5500
Elements
Console
Sources
Network
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension stor...
IndexedDB
Cookies
Private state t...
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward ...
Background fe...
Background sy...
Bounce trackin...
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Network requests
Update
Unregister
Source
serviceworker.js
Received 4/13/2025, 8:02:34 PM
Status
#216 activated and is running
Stop
Clients
http://127.0.0.1:5500/
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#216
Install
#216
Wait
#216
Activate
wad
Maharashtr
Cha
Home
Spices
Sweets
Snacks
Contact
Authenti
Taste of
Maharas
IN
Explore our


Swad Maharashtra Cha - Auther
127.0.0.1:5500
Elements
Console
Sources
Network
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
swad-maha-cache-...
Storage buckets
Background services
Back/forward cache
Background fetch
Filter by path
http://127.0.0.1:5500
Origin
http://127.0.0.1:5500
Bucket name
default
Is persistent
No
Durability
relaxed
Quota
0 B
Expiration
None
Name Res... Con... Con... Tim... Vary...
0 /icons/icon-192x192.png basic ima... 75,9... 4/1... Origin
1 /icons/icon-512x512.png basic ima... 455,... 4/1... Origin
2 /images/bhakarwadi.webp basic ima... 78,3... 4/1... Origin
3 /images/border-pattern.we... basic ima... 1,450 4/1... Origin
4 /images/cart.svg basic ima... 12,8... 4/1... Origin
5 /images/chakli.webp basic ima... 80,6... 4/1... Origin
6 /images/chivda.webp basic ima... 71,9... 4/1... Origin
wad
Maharashtr
Cha
Home
Spices
Sweets
Snacks
Contact
Authenti
Taste of
Maharas
IN

Conclusion:

In this experiment, we successfully registered and activated a service worker to cache essential files and enable offline access for our Swad Maharashtra Cha PWA. Initially, we faced an issue where some images weren't loading offline due to incorrect file paths, which we fixed by double-checking and updating the cache list in the service worker.

Name : Naikwadi Yash Shivdas

MPL Practical 09

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

Theory:

In Progressive Web Apps (PWAs), service workers allow us to handle advanced background tasks that improve performance, reliability, and user engagement. Three important events supported by service workers are fetch, sync, and push.

- The fetch event lets us intercept network requests and serve cached content when offline, helping the app work even without the internet.
- The sync event allows us to delay data synchronization with the server until the user is back online, ensuring reliability.
- The push event enables the app to receive push notifications in the background, even when the web page isn't open.

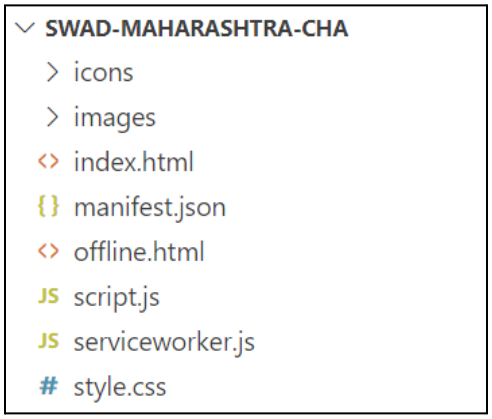
What We Implemented

In our "Swad Maharashtra Cha" PWA, we enhanced the service worker to support these three events:

- Fetch Event: We used a cache-first strategy for same-origin requests and a network-first approach for others. If a fetch fails, we show a custom offline.html page.
- Sync Event: We registered a background sync with the tag "sync-data" to simulate background syncing once connectivity is restored.
- Push Event: We implemented push notification handling, so the service worker can display alerts (like offers or updates) when a message is received.

This makes our eCommerce app more reliable, responsive, and user-friendly — even with unstable or no internet connection.

Folder Structure:



Code:

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
```

```
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />

  <!-- PWA Theme Color -->
  <meta name="theme-color" content="#b22222" />

  <!-- SEO Meta Tags -->
  <meta
    name="description"
    content="Authentic Maharashtrian Spices & Food – Buy famous spices, masalas, and traditional items like Puran Poli, Bhakarwadi, Modaks, and Chitale Bandhu snacks online."
  />
  <meta
    name="keywords"
    content="Maharashtrian food, Goda masala, Bhakarwadi, Modaks, Puran Poli, Chitale Bandhu snacks, Malvani masala, online spice store"
  />
  <meta name="author" content="Swad Maharashtra Cha" />

  <title>Swad Maharashtra Cha - Authentic Maharashtrian Food</title>

  <!-- Manifest File for PWA -->
  <link rel="manifest" href="manifest.json" />

  <!-- Stylesheet -->
  <link rel="stylesheet" href="style.css" />

  <!-- Google Fonts -->
  <link

href="https://fonts.googleapis.com/css2?family=Playfair+Display:wght@400;700&family=Poppins:wght@300;400;500;600&display=swap"
    rel="stylesheet"
  />
 </head>
 <body>

  <script>
    if ("serviceWorker" in navigator) {
      window.addEventListener("load", () => {
        navigator.serviceWorker
          .register("/serviceworker.js")
          .then((registration) => {
            console.log("✅ Service Worker registered! Scope:", registration.scope);

            // 🔔 Request Push Notification Permission
            if ("PushManager" in window) {
              Notification.requestPermission().then((permission) => {
                if (permission === "granted") {
```

```
            console.log("🔔 Push notifications granted.");
          } else {
            console.log("🔕 Push notifications denied.");
          }
        });
      }

      // 🔄 Register Background Sync
      if ("SyncManager" in window) {
        navigator.serviceWorker.ready.then((swReg) => {
          swReg.sync.register("sync-data").then(() => {
            console.log("🔄 Sync registered");
          }).catch((err) => {
            console.log("❌ Sync registration failed:", err);
          });
        });
      }
    })
    .catch((error) => {
      console.log("❌ Service Worker registration failed:", error);
    });
  });
}
</script>
</body>
</html>
```

offline.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />

  <!-- PWA Theme Color -->
  <meta name="theme-color" content="#8c0303" />

  <title>Offline - Swad Maharashtra Cha</title>
</head>
<body>

  <h1>You're Currently Offline</h1>

  <p>Looks like your connection to the flavorful world of Maharashtra has been temporarily lost. Please check your internet connection and try again.</p>

  <p>Don't worry, our delicious spices and treats will be waiting for you when you're back online!</p>

  <button class="btn" onclick="window.location.reload()">Try Again</button>
```

```
    <div class="decorative-line"></div>
  </div>
</body>
</html>
```

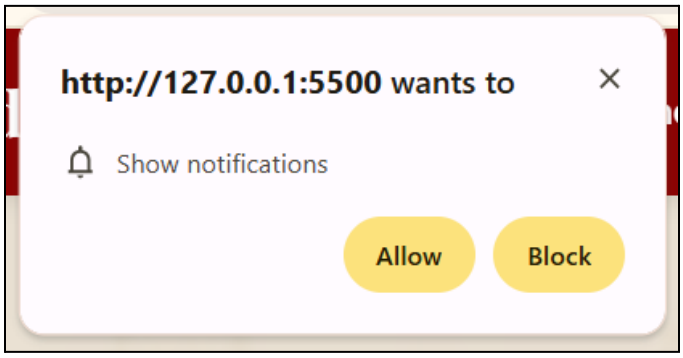
serviceworker.js

```
const CACHE_NAME = "swad-maha-cache-v1";
const FILES_TO_CACHE = [
  "index.html",
  "style.css",
  "script.js",
  "serviceworker.js",
  "manifest.json",
  "images/maharashtra-feast.webp",
  "images/cart.svg",
  "images/goda-masala.webp",
  "images/malvani-masala.webp",
  "images/kanda-lasun.webp",
  "images/puran-poli.webp",
  "images/modak.webp",
  "images/shrikhand.webp",
  "images/bhakarwadi.webp",
  "images/chivda.webp",
  "images/chakli.webp",
  "images/border-pattern.webp",
  "images/maharashtra-pattern.webp",
  "images/paithani-pattern.webp",
  "icons/icon-192x192.png",
  "icons/icon-512x512.png",
  "offline.html"
];

// Install Event
self.addEventListener("install", (event) => {
  console.log("[ServiceWorker] Install");
  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => {
      console.log("[ServiceWorker] Caching files");
      return cache.addAll(FILES_TO_CACHE);
    })
  );
});

// Activate Event
self.addEventListener("activate", (event) => {
  console.log("[ServiceWorker] Activate");
  event.waitUntil(
```

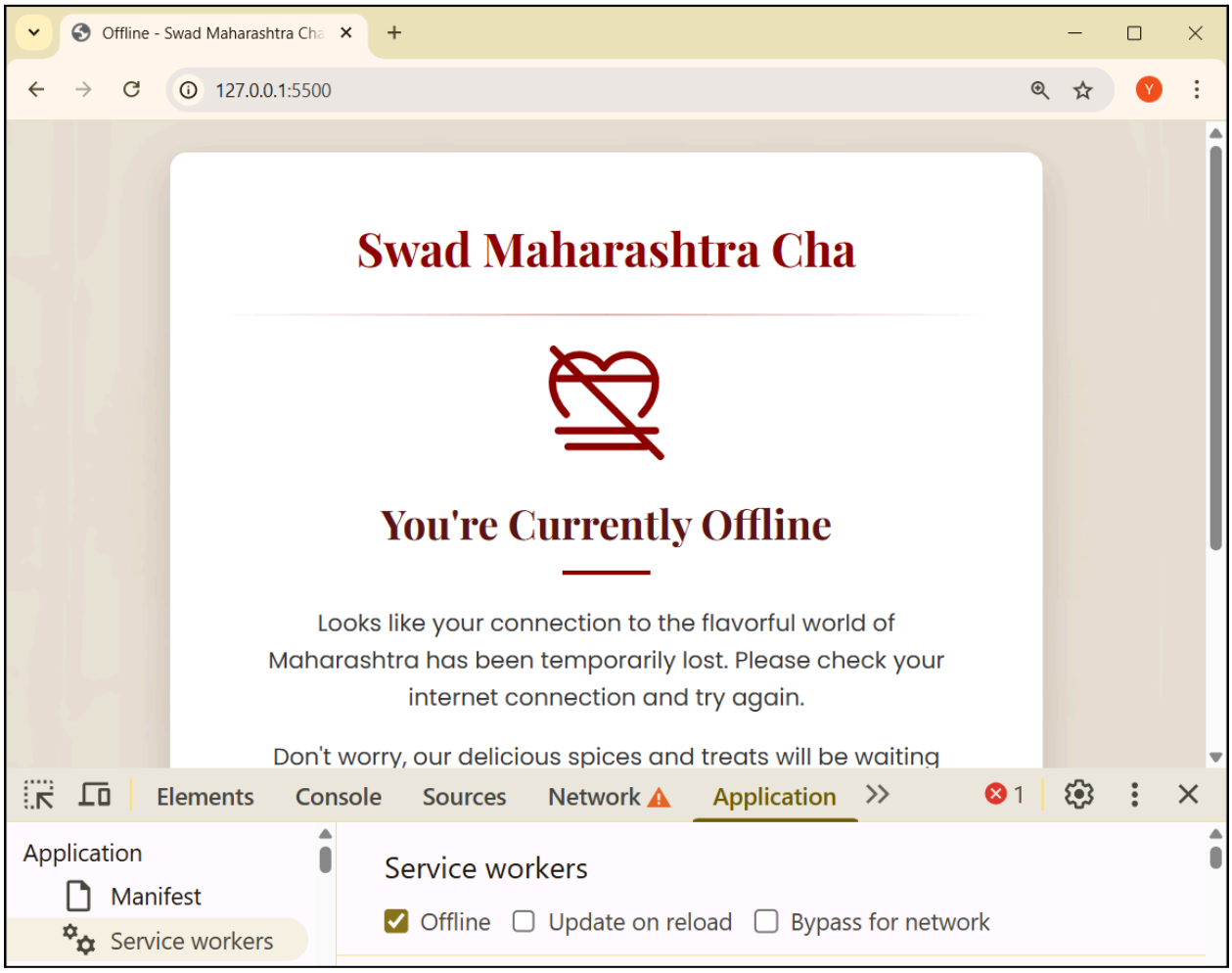
```
    caches.keys().then((keyList) =>
      Promise.all(
        keyList.map((key) => {
          if (key !== CACHE_NAME) {
            console.log("[ServiceWorker] Removing old cache", key);
            return caches.delete(key);
          }
        })
      )
    )
  );
  return self.clients.claim();
});

// Enhanced Fetch Event
self.addEventListener("fetch", (event) => {
  console.log("[ServiceWorker] Fetch", event.request.url);
  const requestURL = new URL(event.request.url);

  // If request is same-origin, use Cache First
  if (requestURL.origin === location.origin) {
    event.respondWith(
      caches.match(event.request).then((cachedResponse) => {
        return (
          cachedResponse ||
          fetch(event.request).catch(() => caches.match("offline.html"))
        );
      })
    );
  } else {
    // Else, use Network First
    event.respondWith(
      fetch(event.request)
        .then((response) => {
          return response;
        })
        .catch(() =>
          caches.match(event.request).then((res) => {
            return res || caches.match("offline.html");
          })
        )
    );
  }
});

// Sync Event (simulation)
self.addEventListener("sync", (event) => {
  if (event.tag === "sync-data") {
    event.waitUntil(
```

```
    (async () => {
      console.log("Sync event triggered: 'sync-data'");
      // Here you can sync data with server when online
    })()
  );
 }
});

// Push Event
self.addEventListener("push", function (event) {
 if (event && event.data) {
  let data = {};
  try {
   data = event.data.json();
  } catch (e) {
   data = {
    method: "pushMessage",
    message: event.data.text(),
   };
  }

  if (data.method === "pushMessage") {
   console.log("Push notification sent");
   event.waitUntil(
    self.registration.showNotification("Swad Maharashtra Cha", {
     body: data.message,
    })
   );
  }
 }
});
```

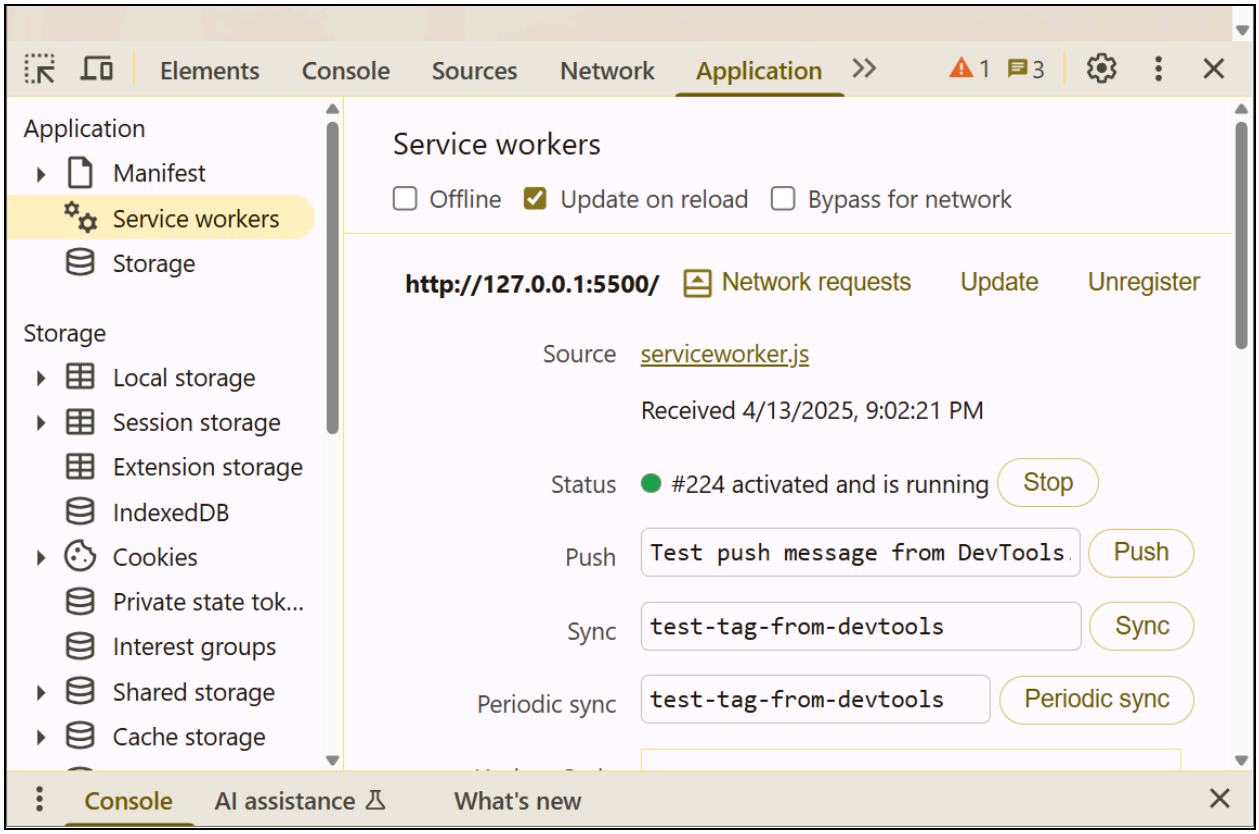
Screenshot:

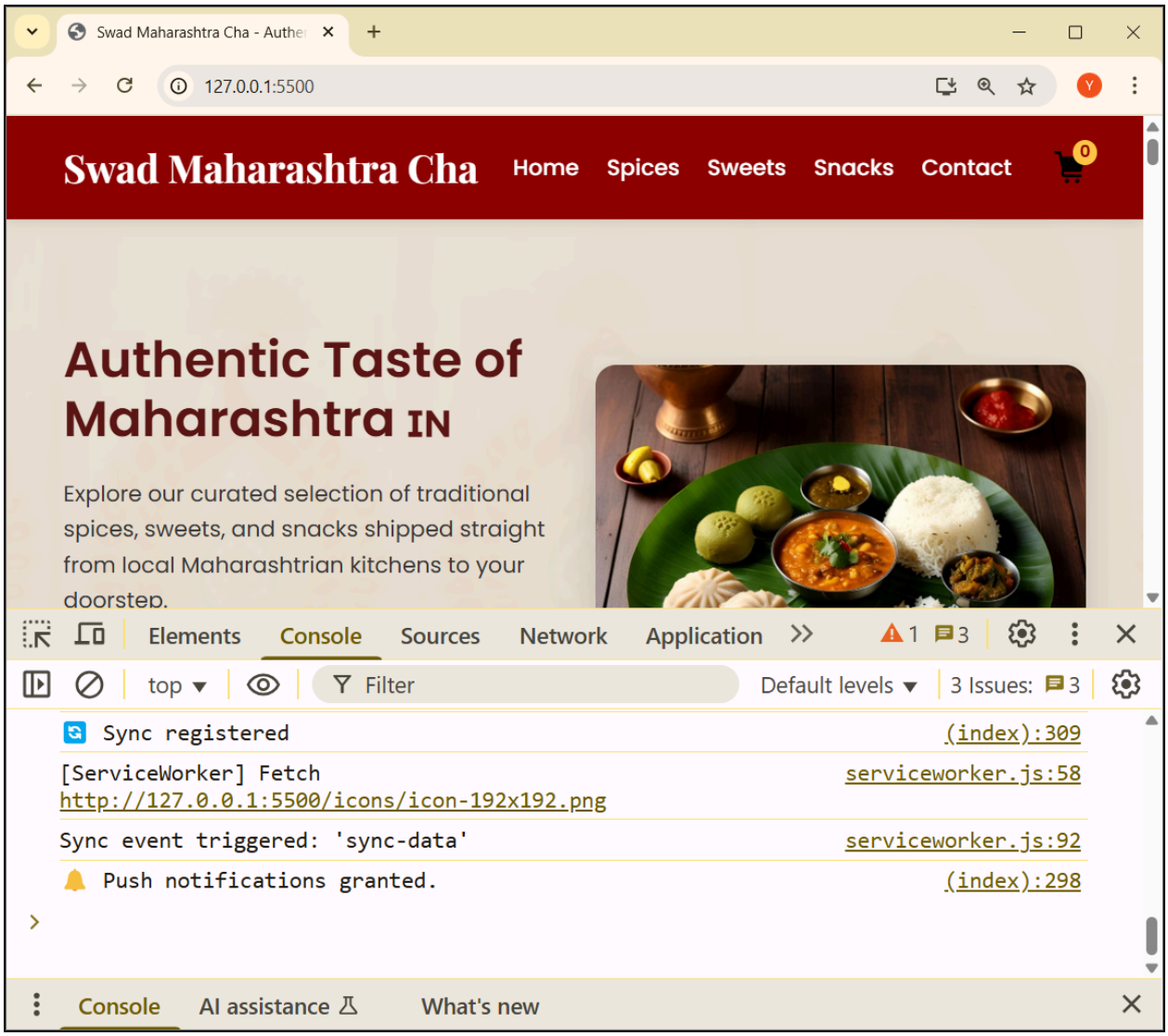


1. Test: Fetch Event (Offline Support)

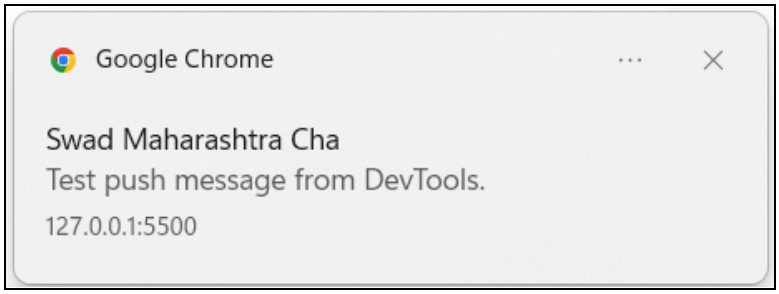


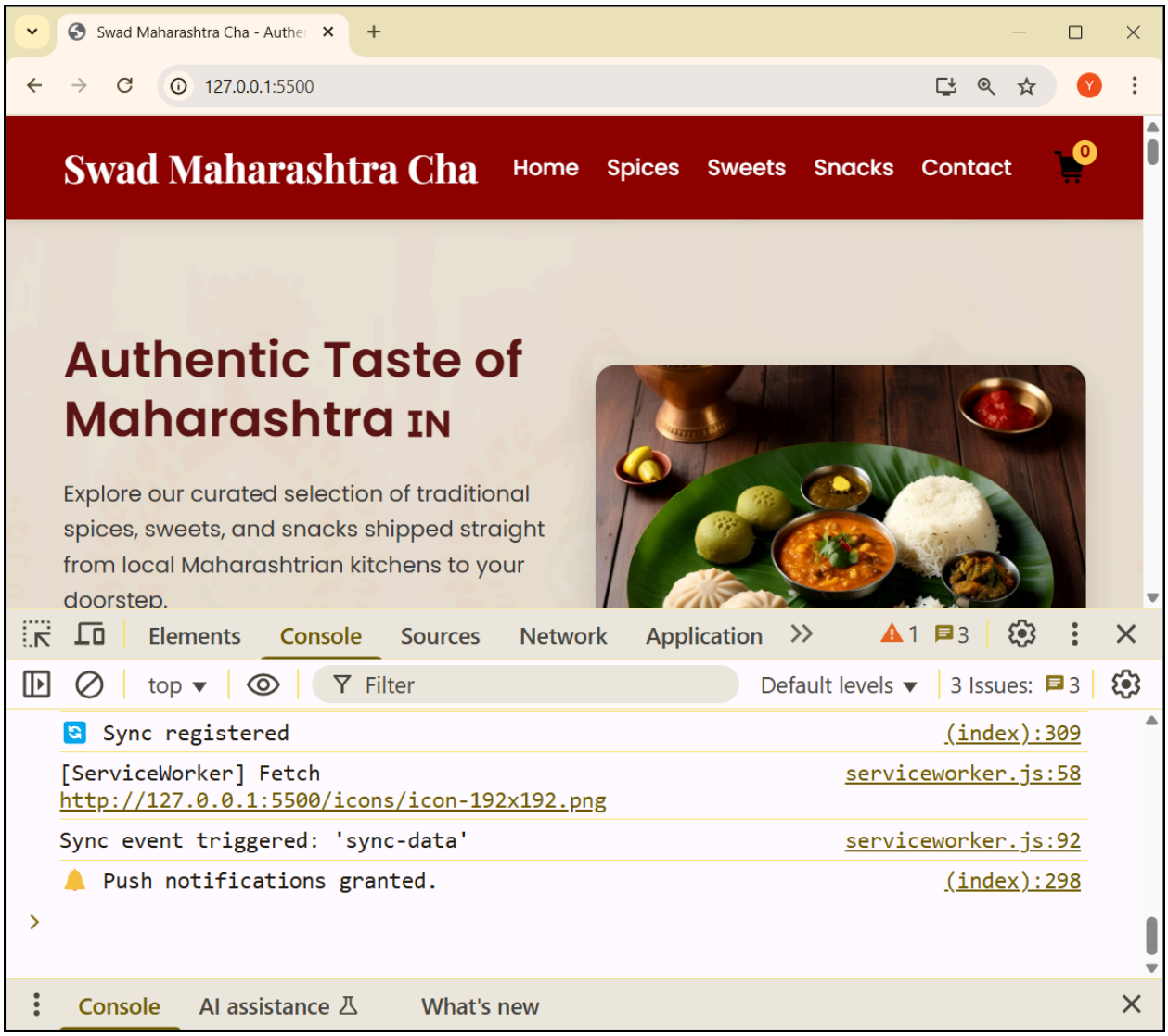
2. Test: Background Sync Event





3. Test: Push Notification Event





Conclusion:

In this experiment, we implemented fetch, sync, and push events in our Swad Maharashtra Cha PWA to enable offline support, background syncing, and push notifications. Initially, push notifications were not showing due to missing permission handling in the browser, which we resolved by adding a proper Notification.requestPermission() block during service worker registration.

Name : Naikwadi Yash Shivdas

MPL Practical 10

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

Github Link: https://yash-naikwadi.github.io/Swad-Maharashtra-Cha-PWA/

Theory:

Deployment is the final step in the web development process where we make our website available online for users to access. GitHub Pages is a free and reliable hosting platform provided by GitHub that lets us publish static websites directly from a repository.

To deploy a Progressive Web App (PWA) like ours, we must ensure that:

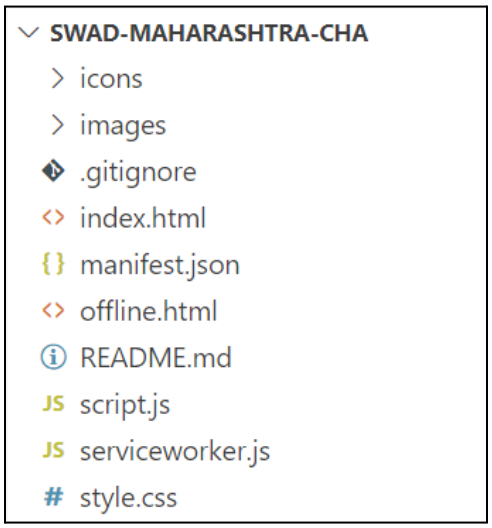
- All file paths are correct (especially for service workers and cached files).
- The service worker properly uses the subfolder paths (since GitHub Pages hosts sites from /username/repo-name/).
- Our project is pushed to the GitHub repository and GitHub Pages is enabled under the repository settings.

What We Implemented

For our Swad Maharashtra Cha eCommerce PWA, we created a GitHub repository and pushed our entire project to it. We updated all the file paths in serviceworker.js to include the repository name /Swad-Maharashtra-Cha-PWA/ so the files could be correctly cached and accessed. We also used relative paths in the HTML to ensure smooth service worker registration and offline support. After this, we enabled GitHub Pages in the repository settings and deployed the app successfully at:
https://yash-naikwadi.github.io/Swad-Maharashtra-Cha-PWA/

This allowed our PWA to be accessible to users worldwide with offline functionality, push notifications, and an installable app-like experience.

Folder Structure

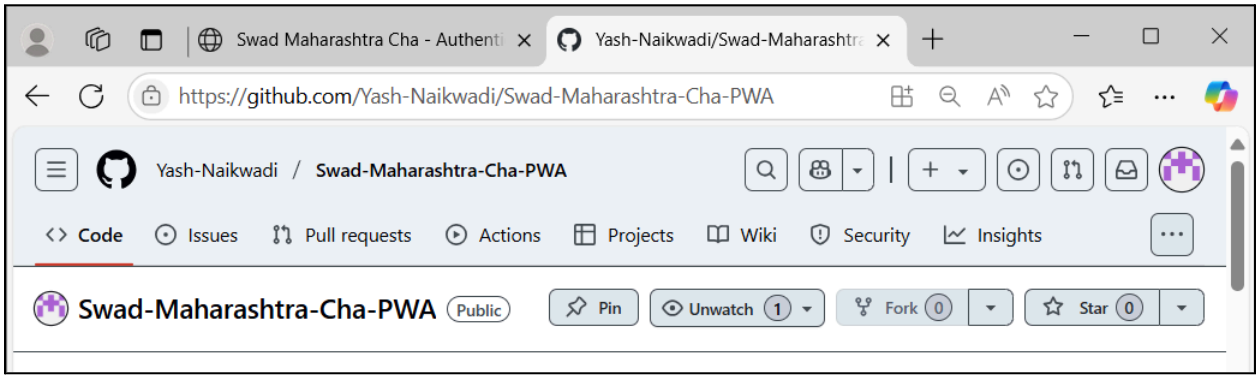


Code:

serviceworker.js

```
const CACHE_NAME = "swad-maha-cache-v1";
const FILES_TO_CACHE = [
  "/Swad-Maharashtra-Cha-PWA/index.html",
  "/Swad-Maharashtra-Cha-PWA/",
  "/Swad-Maharashtra-Cha-PWA/style.css",
  "/Swad-Maharashtra-Cha-PWA/script.js",
  "/Swad-Maharashtra-Cha-PWA/serviceworker.js",
  "/Swad-Maharashtra-Cha-PWA/manifest.json",
  "/Swad-Maharashtra-Cha-PWA/images/maharashtra-feast.webp",
  "/Swad-Maharashtra-Cha-PWA/images/cart.svg",
  "/Swad-Maharashtra-Cha-PWA/images/goda-masala.webp",
  "/Swad-Maharashtra-Cha-PWA/images/malvani-masala.webp",
  "/Swad-Maharashtra-Cha-PWA/images/kanda-lasun.webp",
  "/Swad-Maharashtra-Cha-PWA/images/puran-poli.webp",
  "/Swad-Maharashtra-Cha-PWA/images/modak.webp",
  "/Swad-Maharashtra-Cha-PWA/images/shrikhand.webp",
  "/Swad-Maharashtra-Cha-PWA/images/bhakarwadi.webp",
  "/Swad-Maharashtra-Cha-PWA/images/chivda.webp",
  "/Swad-Maharashtra-Cha-PWA/images/chakli.webp",
  "/Swad-Maharashtra-Cha-PWA/images/border-pattern.webp",
  "/Swad-Maharashtra-Cha-PWA/images/maharashtra-pattern.webp",
  "/Swad-Maharashtra-Cha-PWA/images/paithani-pattern.webp",
  "/Swad-Maharashtra-Cha-PWA/icons/icon-192x192.png",
  "/Swad-Maharashtra-Cha-PWA/icons/icon-512x512.png",
  "/Swad-Maharashtra-Cha-PWA/offline.html"
];
```

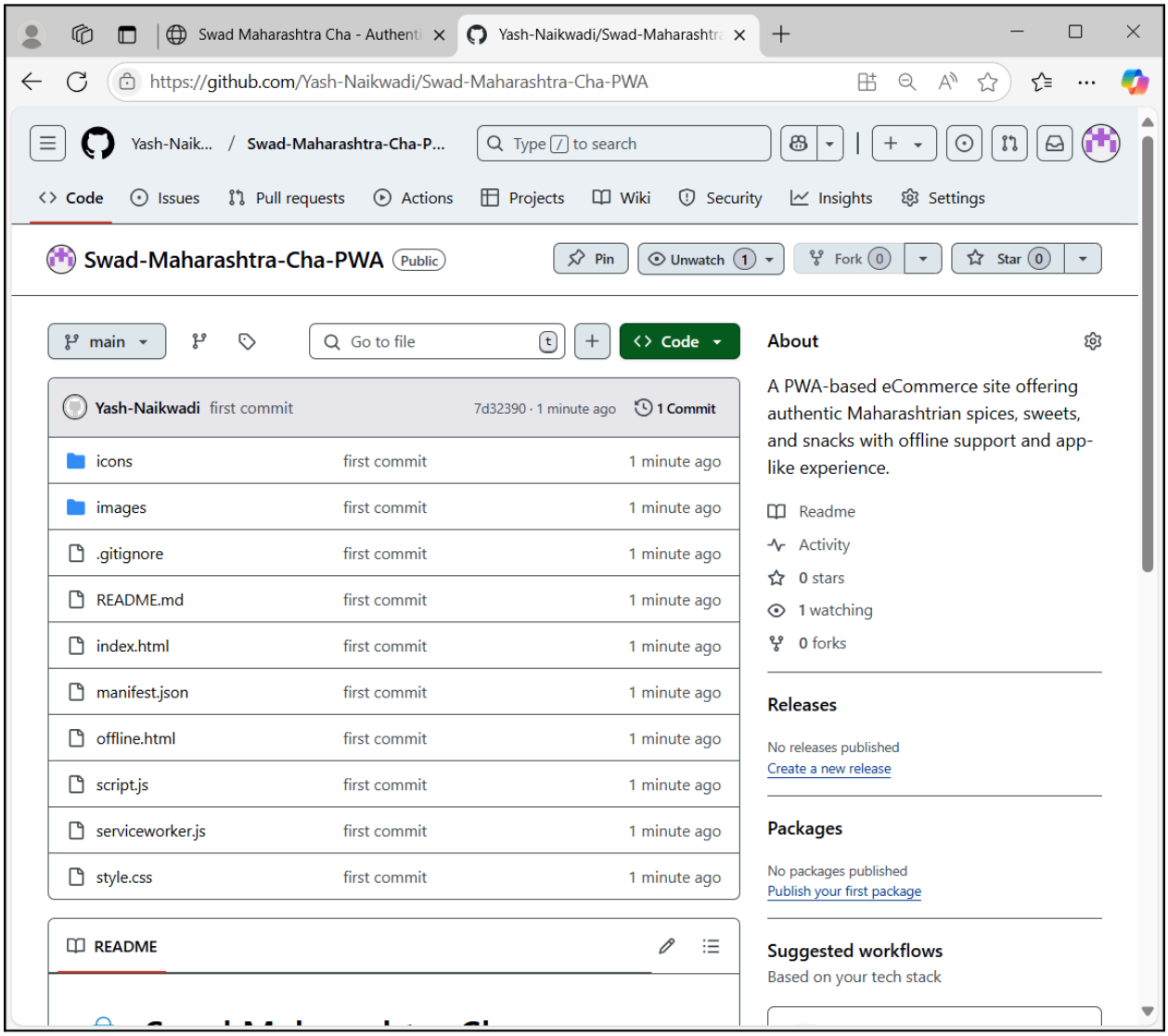
Create a GitHub Repository

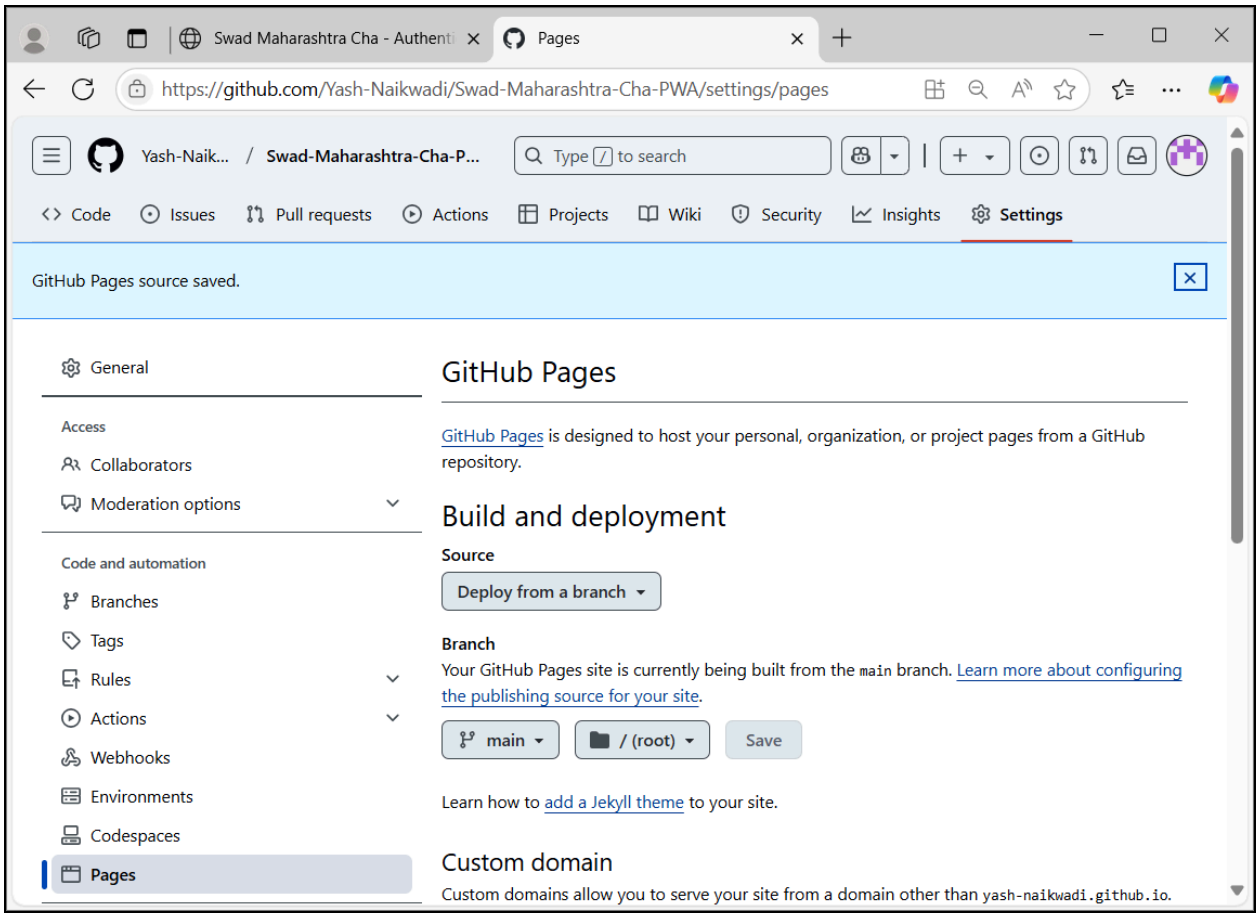


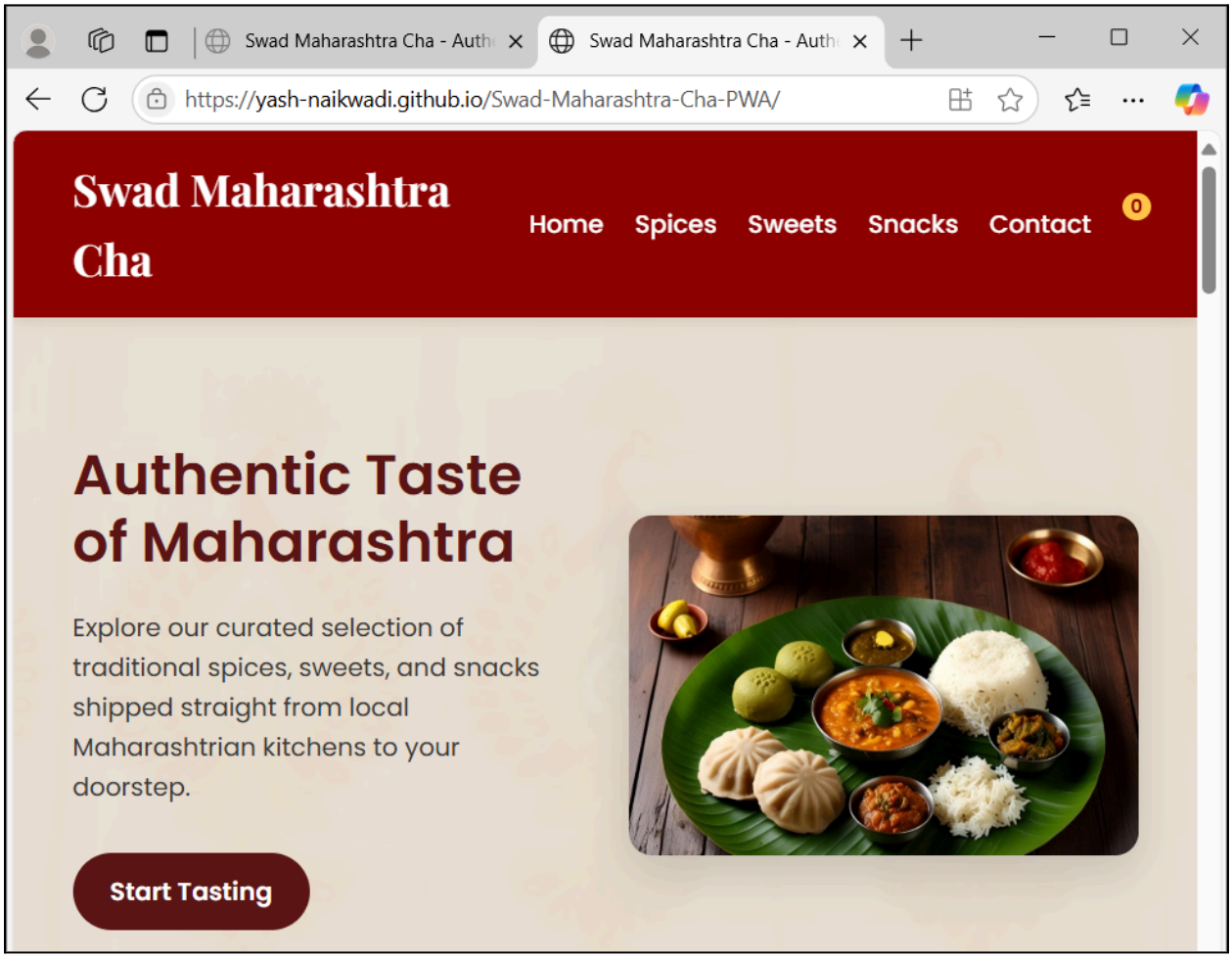
Link Your Local Project to GitHub and Push Changes.

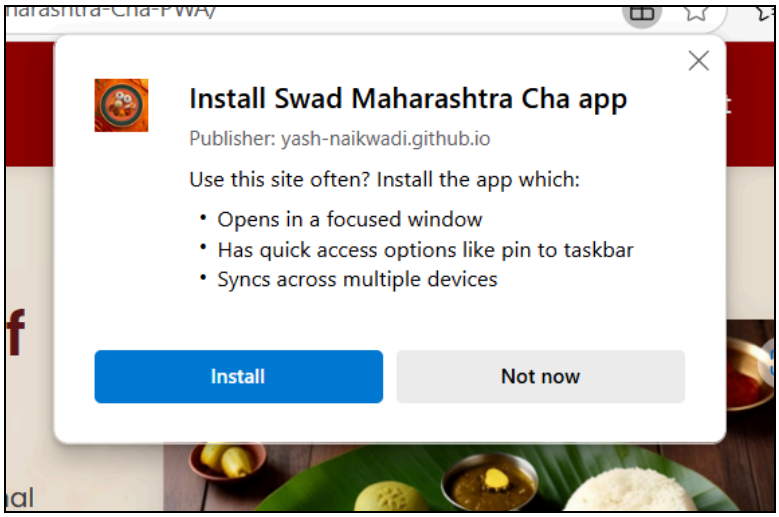
```
PS D:\Users\Yash\Documents\Swad-Maharashtra-Cha> git init
Initialized empty Git repository in D:/Users/Yash/Documents/Swad-Maharashtra-Cha/.git/
PS D:\Users\Yash\Documents\Swad-Maharashtra-Cha> git add .
PS D:\Users\Yash\Documents\Swad-Maharashtra-Cha> git commit -m "first commit"
[main (root-commit) 7d32390] first commit
 24 files changed, 1420 insertions(+)
```

```
PS D:\Users\Yash\Documents\Swad-Maharashtra-Cha> git remote add origin https
://github.com/Yash-Naikwadi/Swad-Maharashtra-Cha-PWA.git
PS D:\Users\Yash\Documents\Swad-Maharashtra-Cha> git push -u origin main
Enumerating objects: 28, done.
Counting objects: 100% (28/28), done.
Delta compression using up to 4 threads
Compressing objects: 100% (28/28), done.
Writing objects: 100% (28/28), 1.23 MiB | 1.27 MiB/s, done.
Total 28 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Yash-Naikwadi/Swad-Maharashtra-Cha-PWA.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
PS D:\Users\Yash\Documents\Swad-Maharashtra-Cha>
```




Swad Maharashtra Cha - Authenti
Pages
https://github.com/Yash-Naikwadi/Swad-Maharashtra-Cha-PWA/settings/pages
Yash-Naik... / Swad-Maharashtra-Cha-P...
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
GitHub Pages source saved.
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
GitHub Pages
GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.
Build and deployment
Source
Deploy from a branch
Branch
Your GitHub Pages site is currently being built from the main branch. Learn more about configuring the publishing source for your site.
main
/ (root)
Save
Learn how to add a Jekyll theme to your site.
Custom domain
Custom domains allow you to serve your site from a domain other than yash-naikwadi.github.io.


Swad Maharashtra Cha - Auth
Swad Maharashtra Cha - Auth
https://yash-naikwadi.github.io/Swad-Maharashtra-Cha-PWA/
Swad Maharashtra Cha
Home
Spices
Sweets
Snacks
Contact
0
Authentic Taste of Maharashtra
Explore our curated selection of traditional spices, sweets, and snacks shipped straight from local Maharashtrian kitchens to your doorstep.
Start Tasting



Conclusion:

In this experiment, we successfully deployed our Swad Maharashtra Cha PWA to GitHub Pages by configuring correct file paths and ensuring the service worker used the full repository path. Initially, we faced a 404 error during service worker registration, which we fixed by replacing absolute paths with relative ones and prefixing all cache URLs with the repository name.

Name : Naikwadi Yash Shivdas

MPL Practical 11

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

Theory:

Google Lighthouse is a powerful, open-source auditing tool built into Chrome DevTools that helps developers analyze and improve the quality of web applications. It evaluates web apps based on multiple parameters such as Performance, Accessibility, Best Practices, SEO, and PWA compliance. This tool is especially useful for verifying whether a web app qualifies as a proper Progressive Web App.

During a Lighthouse test, the tool checks for:

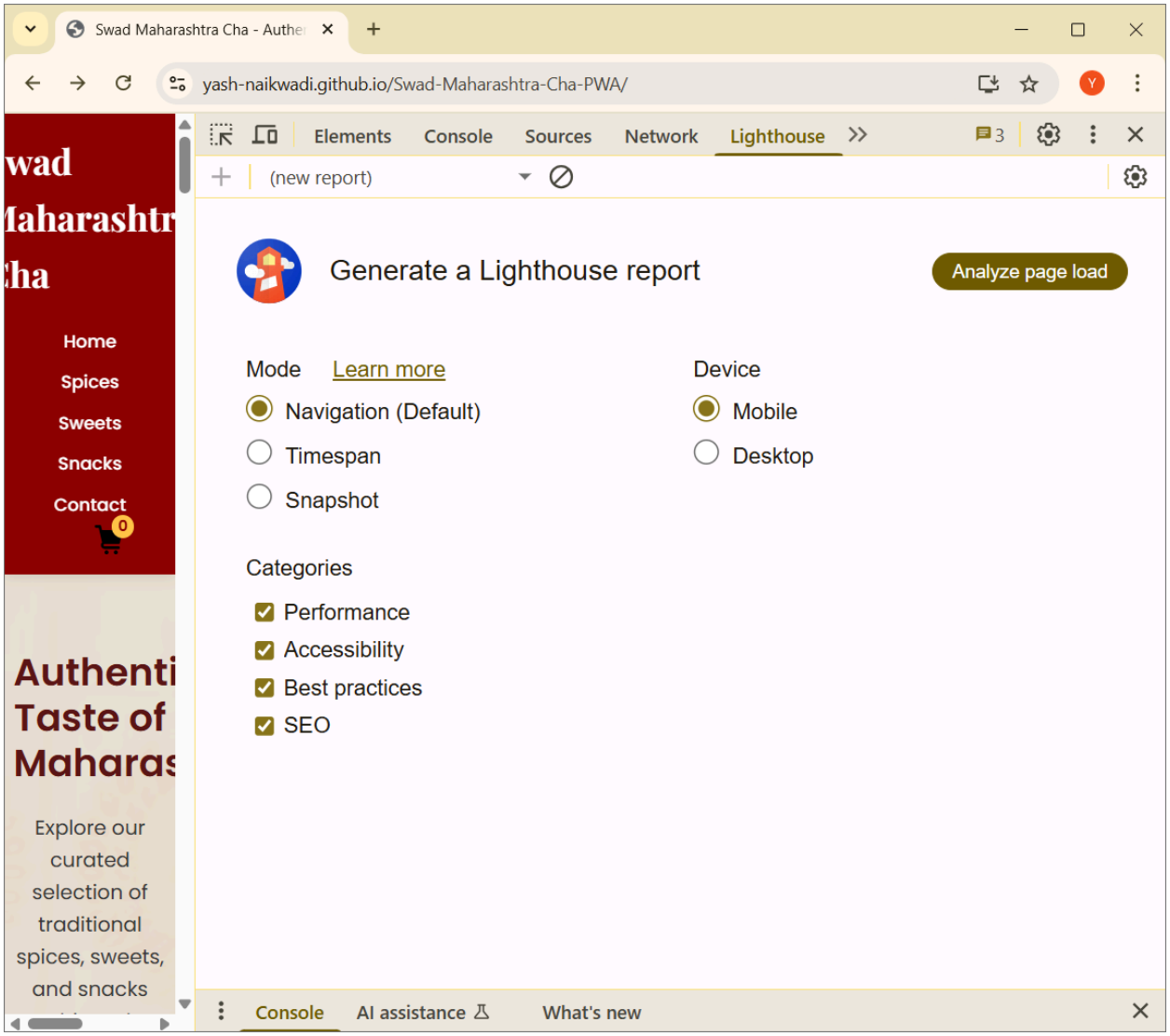
- Fast and responsive loading (Performance)
- Accessible design for all users (Accessibility)
- Secure and modern coding practices (Best Practices)
- Search engine optimization readiness (SEO)
- Proper PWA features like manifest, service worker, and offline capabilities

What We Implemented and Tested

For our project "Swad Maharashtra Cha", we ran a Lighthouse test on the deployed GitHub Pages link:
https://yash-naikwadi.github.io/Swad-Maharashtra-Cha-PWA/
The audit results showed excellent scores:

- Performance: 93
- Accessibility: 92
- Best Practices: 96
- SEO: 100

These scores confirm that our PWA is highly optimized, accessible, and production-ready. The few suggestions we received (like specifying image dimensions and improving contrast) were noted for further improvement.


Swad Maharashtra Cha - Auther
yash-naikwadi.github.io/Swad-Maharashtra-Cha-PWA/
wad
Maharashtr
Cha
Home
Spices
Sweets
Snacks
Contact
0
Authenti
Taste of
Maharas
Explore our curated selection of traditional spices, sweets, and snacks
Elements
Console
Sources
Network
Lighthouse
3
(new report)
Generate a Lighthouse report
Analyze page load
Mode
Learn more
Navigation (Default)
Timespan
Snapshot
Device
Mobile
Desktop
Categories
Performance
Accessibility
Best practices
SEO
Console
AI assistance
What's new

Conclusion:

In this experiment, we used Google Lighthouse to test the performance and PWA compliance of our Swad Maharashtra Cha website, where we achieved high scores in all categories. Initially, we faced issues like missing image dimensions and low text contrast, which we resolved by updating the HTML and CSS to match Lighthouse's accessibility and performance recommendations.

https://yash-naikwadi.github.io/Swad-Maharashtra-Cha-PWA/

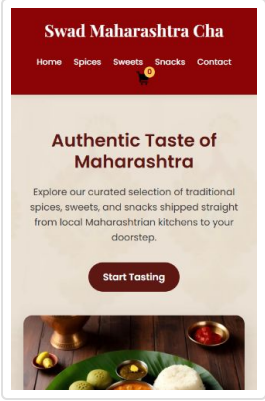
Performance	Accessibility	Best Practices	SEO
93	92	96	100

93

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 50–89 90–100



METRICS Expand view

First Contentful Paint
2.1 s

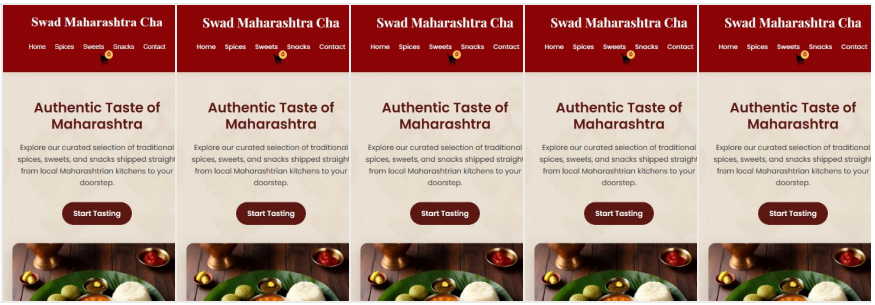
Largest Contentful Paint
2.9 s

Total Blocking Time
0 ms

Cumulative Layout Shift
0.02

Speed Index
2.6 s

View Treemap



Show audits relevant to: All FCP LCP TBT CLS

DIAGNOSTICS

- ▲ Properly size images — Potential savings of 136 KiB
- ▲ Largest Contentful Paint element — 2,940 ms
- ▲ Eliminate render-blocking resources — Potential savings of 390 ms
- Image elements do not have explicit `width` and `height`
- Serve static assets with an efficient cache policy — 15 resources found
- Minimize main-thread work — 2.8 s
- ○ Avoid large layout shifts — 4 layout shifts found
- ○ Avoid chaining critical requests — 7 chains found
- ○ Minimize third-party usage — Third-party code blocked the main thread for 0 ms
- ○ Avoid long main-thread tasks — 6 long tasks found

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

PASSED AUDITS (28) Show

Accessibility

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

CONTRAST

▲ Background and foreground colors do not have a sufficient contrast ratio.

These are opportunities to improve the legibility of your content.

ADDITIONAL ITEMS TO MANUALLY CHECK (10)

Show

These items address areas which an automated testing tool cannot cover. Learn more in our guide on conducting an accessibility review.

PASSED AUDITS (12)

Show

NOT APPLICABLE (44)

Show

Best Practices

TRUST AND SAFETY

▲ Requests the notification permission on page load

○ Ensure CSP is effective against XSS attacks

○ Use a strong HSTS policy

- Ensure proper origin isolation with COOP
- Mitigate clickjacking with XFO or CSP

PASSED AUDITS (14) Show

NOT APPLICABLE (2) Show

SEO

These checks ensure that your page is following basic search engine optimization advice. There are many additional factors Lighthouse does not score here that may affect your search ranking, including performance on Core Web Vitals. Learn more about Google Search Essentials.

ADDITIONAL ITEMS TO MANUALLY CHECK (1) Show

Run these additional validators on your site to check additional SEO best practices.

PASSED AUDITS (8) Show

NOT APPLICABLE (2) Show

Captured at Apr 13, 2025, 10:18 PM GMT+5:30
Initial page load

Emulated Moto G Power with Lighthouse 12.4.0
Slow 4G throttling

Single page session

Using Chromium 135.0.0.0 with devtools

Generated by **Lighthouse** 12.4.0 | File an issue

Naikwadi Yash Shivdas / D15B / 39

SDS | Page No.
Date / /

MPL Assignment 01

Q1 a) Explain the key features & advantages of using Flutter for mobile app development.

Soln. Key features of Flutter :-

1) Single Codebase : Write one code for both Android & IOS

2) Fast Performance : Uses Dart language & a high performance rendering engine

3) Hot Reload : See changes instantly without restarting the app

4) Rich UI Components : Comes with customizable widgets for ~~support~~ smooth UI design.

5) Native-like Experience : Provides high quality animations & fast execution.

6) Cross-Platform Support : Can be used for mobile, web & desktop apps.

7) Open-Source : Free to use & has a strong developer community.

Advantages of Using Flutter :-

1) Saves Time & Effort : Single codebase for multiple platforms.

2) High Speed Development : Hot Reload feature speed up coding

3) Cost - Effective : Reduces development cost & time

4) Attractive UI : Provides beautiful & customizable widgets

5) Good Performance : Uses Dart & Skia for fast & smooth rendering

6) Easy Integration : Supports third-party plugins & native code integration.

b) Discuss how the Flutter framework differs from traditional approaches & why it has gained popularity in the developer community

Soln. How Flutter differs from Traditional Approaches :-

1) Single Codebase : Traditional methods need separate code for Android (Java / Kotlin) & iOS (Swift / Objective-C), but Flutter uses one code for both.

2) Hot Reload : Traditional apps require full restart after changes, but Flutter updates instantly

3) UI Rendering : Traditional apps use native components, while Flutter has its own rendering engine (Skia) for faster performance.

4) Performance : Flutter compiles directly to native machine code, making it faster than frameworks that use a bridge (e.g. React Native)

5) Customization : Traditional UI design depends on platform-specific components, but Flutter provides fully customizable widgets

Why Flutter is popular Among Developers :

1) Fast Development : Hot Reload & single code base save time

2) Cross-Platform Support : Works on mobile, web & desktop

3) Beautiful UI : Rich, customizable widgets for modern designs

4) High Performance : Runs smoothly without a bridge like React Native

5) Active Community & Google Support : Regular updates & strong community help developers

Q2 a) Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces.

Soln. Concept of Widget Tree in Flutter :-

In Flutter, everything is a widget. Widgets are arranged in a tree structure, called the widget tree. This tree represents the UI of the app, where parents widget contain child widgets.

For example, a Scaffold widget can have a Column widget, which contains Text & Button widgets. Changes in widgets update the tree dynamically.

- Widget Composition for Complex UI:

Flutter uses small, reusable widgets to build complex UI. Instead of creating a single large UI block, developers combine multiple small widgets like Rows, Columns, Containers & Buttons.

For Example :-

1) A ListView can contain multiple Card widgets.
2) A Column can hold Text, Images & Buttons.

This modular approach makes the UI flexible, readable & easy to manage

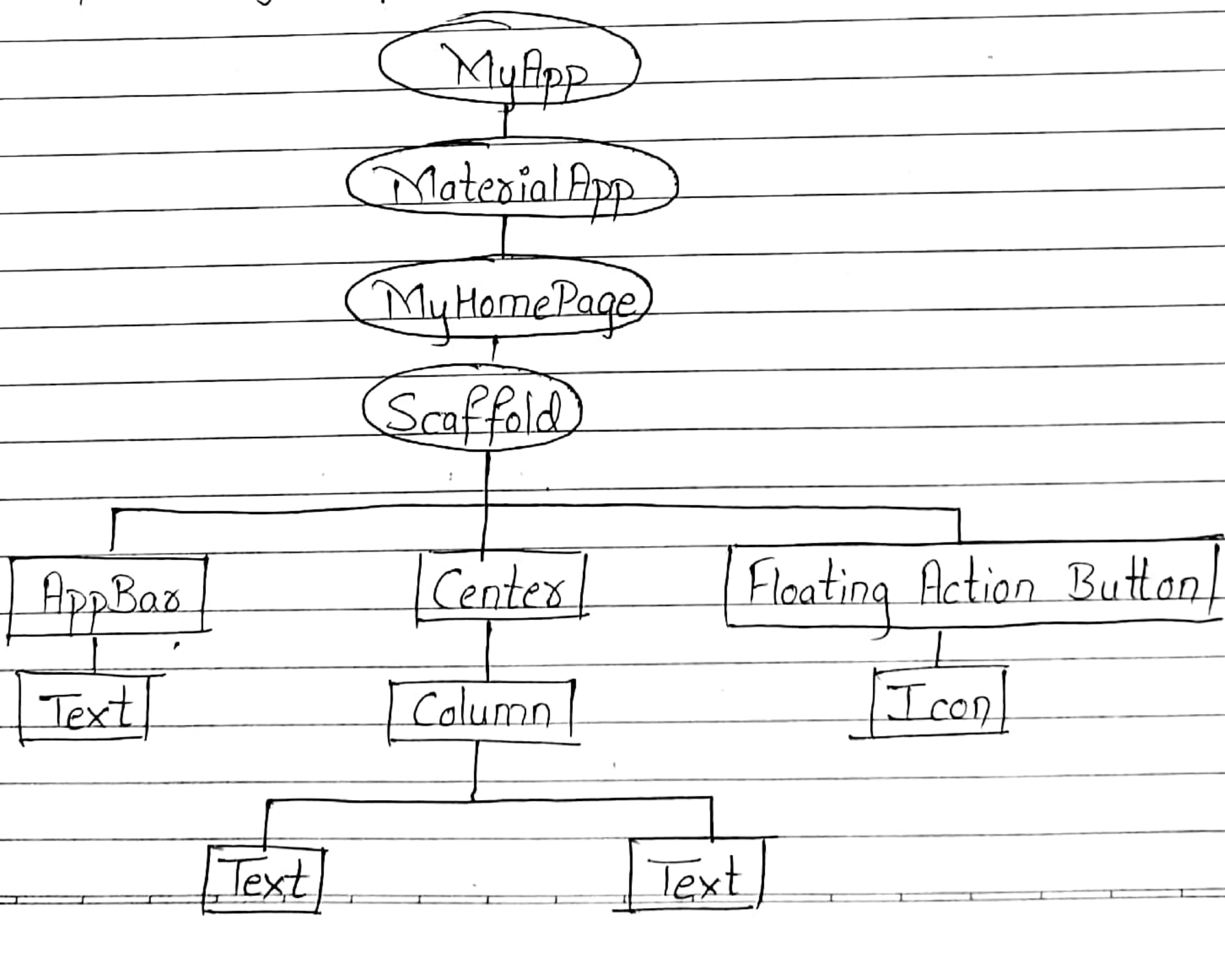
5) Provide examples of community used widgets & their roles in creating a widget tree.

Soln. Commonly used widgets & their Roles in a widget tree :-

1) Scaffold : Provides the basic layout structure (Appbar, Body, Floating Button)

2) AppBar : Displays the top navigation bar with a title
3) Text : Displays simple text on the screen.
4) Image : Shows images from assets or URLs
5) Container : Used for styling (background color, padding, margin)
6) Row : Arranges child widgets ~~vertically~~ horizontally.
7) Column : Arranges child widgets vertically.
8) ListView : Displays scrollable lists.
9) ElevatedButton : A clickable button with elevation
10) Textfield : Used for user input (typing text)
11) Card : Creates a styled box for displaying content
12) Stack : Overlays widgets on top of each other.

Example Widget Tree :-



This tree structure helps in organizing & managing the UI efficiently.

Q3 a) Discuss the importance of state management in Flutter applications

Soln. Importance of State Management in Flutter Applications:

State management is important because it controls how the app stores, updates & displays data when the user interacts with it.

Why State Management is Needed?

1) Keeps UI Updated: Ensures that the app reflects changes (e.g. button clicks, text inputs)

2) Improves Performance: Updates only necessary parts of the UI instead of reloading everything.

3) Manages Complex Data: Helps handle user inputs, API data, & navigation efficiently

4) Ensures Smooth User Experience: Keeps the app responsive & interactive

Types of State in Flutter:

1) Local State: Managed within a single widget using StatefulWidget

2) Global State : Shared across multiple screens using Provider, Riverpod, Bloc, or Redux

Without proper state management, the app may behave unpredictably or show outdated data.

b) Compare & contrast the different state management approaches available in Flutter, such as setState, Provider & Riverpod. Provide scenarios where each approach is suitable

Soln.

Approach	How it works	When to use
setState	Updates UI by calling setState() in a StatefulWidget	Best for small apps or managing state within a single widget. Example: Toggling a button color
Provider	Uses InheritedWidget to share state across widgets efficiently	Suitable for medium sized apps where data needs to be shared between multiple widgets. Example: Managing user authentication
Riverpod	An improved version ~~for~~ of Provider with better performance & simpler syntax	Best for large apps that need complex state management with dependency injection. Example: Handling API data & app-wide themes

SDS | Page No.
Date / /

Choosing the Right Approach :

- Use setState for simple UI updates
- Use Provider for moderate state sharing across widgets
- Use Riverpod for scalable, well-structured applications.

Q4 a) Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution.

Soln: Process of Integrating Firebase with a Flutter Application :-

1) Create a Firebase Project : Go to [Firebase Console] (https://console.firebase.google.com/), create a new project

2) Add Firebase to Flutter App : Register the app (Android / iOS) & download the google-services.json (Android) & or GoogleService-Info.plist (iOS)

3) Install Firebase Packages : Add dependencies like 'firebase-core' & 'firebase-auth' in 'pubspec.yaml'

4) Initialize Firebase : Import Firebase in 'main.dart' & call 'Firebase.initializeApp()'.

5) Use Firebase Services : Implement authentication, database or cloud functions as needed

Benefits of using Firebase as a Backend Solution

1) Real time Database : Syncs data instantly across devices

2) Authentication : Provides ready-to-use sign-in options (Google, Email, etc.)

3) Cloud Firestore : Stores structured data efficiently

4) Hosting & Storage : Hosts web apps & stores files securely

5) Scalability : Handles large users bases without managing servers

6) Push Notifications : Sends alerts & updates to users.

b) Highlight the Firebase services commonly used in Flutter development & provide a brief overview of how data synchronization is achieved.

Soln. Common Firebase Services Used in Flutter Development :

1) Authentication : Provides user sign-in methods (Google, Email, Facebook, etc.)

2) Cloud Firestore : A NoSQL database that stores & syncs data in real time

3) Firebase Realtime Database : Stores & update data instantly across all connected devices

4) Firebase Cloud Storage : Used for storing & retrieving files like images & videos.

5) Firebase Cloud Messaging (FCM) : Seads push notifications to users

6) Firebase Hosting : Deploys web apps with fast & secure hosting

7) Firebase Analytics : Tracks user behaviour & app performance

& How Data Synchronization is Achieved :

1) Real time Updates : Firestore & Realtime Database sync data across devices instantly

2) Listeners & Streams : Widgets listen for changes & update the UI automatically.

3) Offline Support : Firebase catches data, allowing apps to work offline & sync when online.

This ensures fast, smooth & automatic data updates in Flutter apps

Naikwadi Yash Shivdas / D15B / 39

MPL Assignment 02

Define Progressive Web App (PWA) & explain its significance in modern web development. Discuss the key characteristics that differentiate PWAs from traditional mobile apps.

A Progressive Web App (PWA) is a type of web application that works like a mobile app but runs in a browser. It can be installed on a device, works offline & provides a fast & smooth user experience.

Significance of PWA in Modern Web Development :

1) Cross Platform Compatibility : Works on both mobile & desktop with a single codebase
2) Offline Support : Can function without the internet using cached data.
3) Fast Performance : Loads quickly, even on slow networks.
4) No App Store Required : Users can install it directly from the browser
5) Lower Development Cost : One PWA can replace separate Android & iOS apps.

Key Differences Between PWA & Traditional Mobile Apps :-

Feature	PWA	Traditional Mobile App
Installation	Direct from browser	Download from App Store

Internet Required	Works offline with caching	Usually requires internet
Performance	Fast with service workers	Faster but need installation
Updates	Automatic, no app store approval	Manual update needed.
Development Cost	Lower (one codebase for all)	Higher (separate apps for each p

PWAs combine the best of web & mobile apps, ma them efficient & user-friendly.

Q2 Define responsive web design & explain its imp ance in the context of Progressive Web Apps. C & contrast responsive, fluid & adaptive web design approaches.

Soln. Definition of Responsive Web Design :

Respective Web Design (RWD) is a technique th makes web pages adjust automatically to differe screen sizes & devices. It ensures a good use experience on mobiles, tablets & desktops withou needing separate versions of a website.

Importance of Responsive Design in PWAs :

1) Better User Experience : PWAs work smoothly any device.

2) Faster load Time : Optimized design improves speed.

3) SEO Benefits : Google ranks responsive sites higher

4) Cost-effective : No need to build multiple versions for different screens.

Comparison ~~for~~ of Web Design Approaches :-

Approach	How it works	Pros	Cons
Responsive	Uses flexible grids & CSS media queries to adjust layout	Works on all devices, improves SEO	Can be complex to design
Fluid	Uses percent-based widths instead of fixed pixels, so elements resize smoothly	Works well on different screen sizes, easy to implement	less control over layout on large screens
Adaptive	Uses fixed layouts that change at specific breakpoints	Optimized for known screen sizes	More effort required to design for each screen size

Key Differences :

- Responsive adapts dynamically to all screens
- Fluid resizes smoothly but may not be fully optimized

- Adaptive loads different layouts based on type

Responsive design is best for PWAs because it a seamless experience on all devices.

Q3 Describe the lifecycle of Service Workers, inc registration, installation, & activation phases.

Soln. Lifecycle of Service Workers

A Service Workers is a script that runs in th background & helps a web app work offline, l faster & send push notifications. Its lifecyc has three main phases :-

1) Registration Phase :- The browser registers Service Workers using JavaScript.

Code Example :-

```
if ('serviceWorker' in navigator) {
    navigator.serviceWorker.register ('/sw.js')
        .then (() => console.log ('service worker registe
        .catch (error => console.log ('failed :', error));
}
```

- This tells the browser to install & activate Service Worker

SDS Page No.
Date / /

2) Installation Phase
- The Service Worker downloads necessary files (HTML, CSS, JS) & stores them in cache
- If successful, it moves to the activation phase

Code Example :-

```
self.addEventListener('install', event => {
    event.waitUntil(
        caches.open('app-cache').then(cache => {
            return cache.addAll(['/', '/index.html',
                                 '/styles.css']);
        })
    );
}
```

- This ensures the app loads even without the internet

3) Activation Phase

- The old Service Worker is replaced with a new one
- Unused cache files from the previous version are deleted.

Code Example :-

```
self.addEventListener('activate', event => {
    event.waitUntil(
        caches.keys().then(keys.map(key => {
            if(key !== 'app-cache'){
                return caches.delete(key);
```

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

- The Service Worker is now fully active & controls network requests.

Final Step : Fetch & Sync

Once activated, the Service Worker intercep network requests, serves cached files, & sync data when the internet is available.

This lifecycle makes PWAs faster, more relia & capable of working offline.

Q4 Explain the use of IndexedDB in the Servic Worker for data storage

Soln. Use of IndexedDB in Service Worker for D Storage

IndexedDB is a browser database that stores large amounts of structured data like JSON o It helps PWAs work offline by saving & re ing data efficiently.

Why Use IndexedDB in Service Workers?

1) Offline Support : Stores data when offline & syncs it later.
2) Efficient Storage : Saves ~~stor~~ structured data like user settings, cart items, or form inputs.
3) Faster Access : Retrieves data quickly without needing a network request.
4) Persistent Data : Data remains saved even after the browser is closed.

How Service Workers Use IndexedDB?

Opening the Database

```
let db;
let request = indexedDB.open('MyDatabase', 1);

request.onsuccess = function (event) {
    db = event.target.result;
};
```

Creating a Store & Adding Data

```
request.onupgradeneeded = function (event) {
    let db = event.target.result;
    let store = db.createObjectStore('Users', {keyPath:
                                              'id'});
    store.add ({id: 1, name: 'John Doe', age: 25});
};
```

Fetching Data in Service Worker

```
let transaction = db.transaction ('Users', 'readonly');
let store = transaction.objectStore ('Users');
```

```
let getUser = store.get(1);

getUser.onsuccess = function (){
    console.log (getUser.result);
};
```