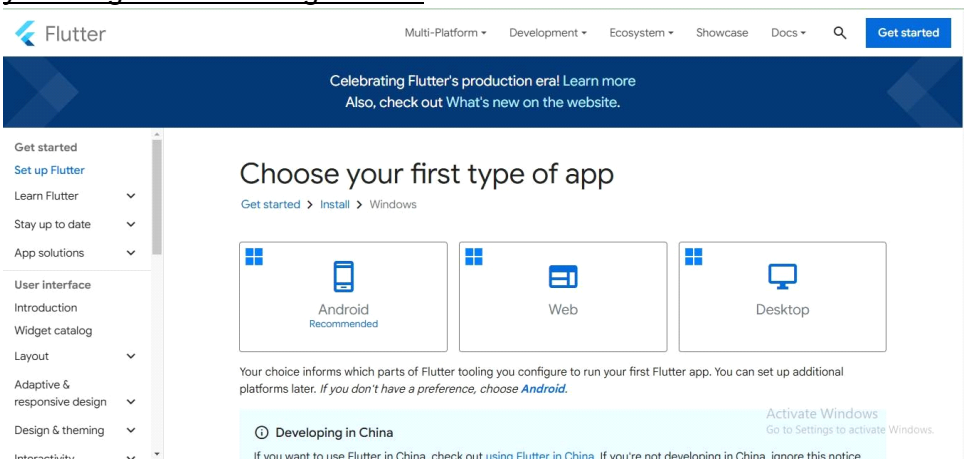
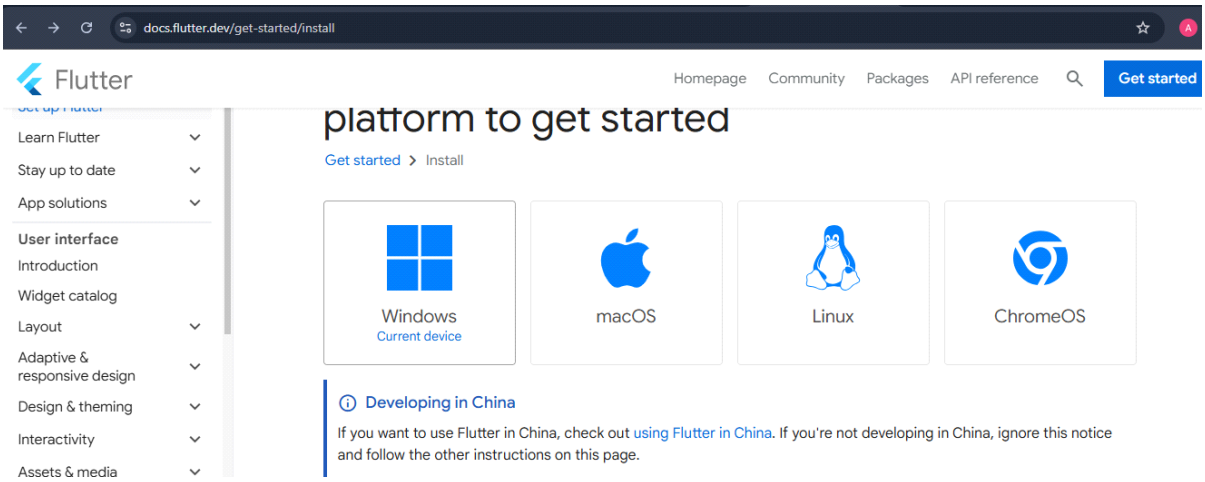
Name: Abhay Gupta
Div: D15B
Roll no: 16

Experiment No:1

Aim:

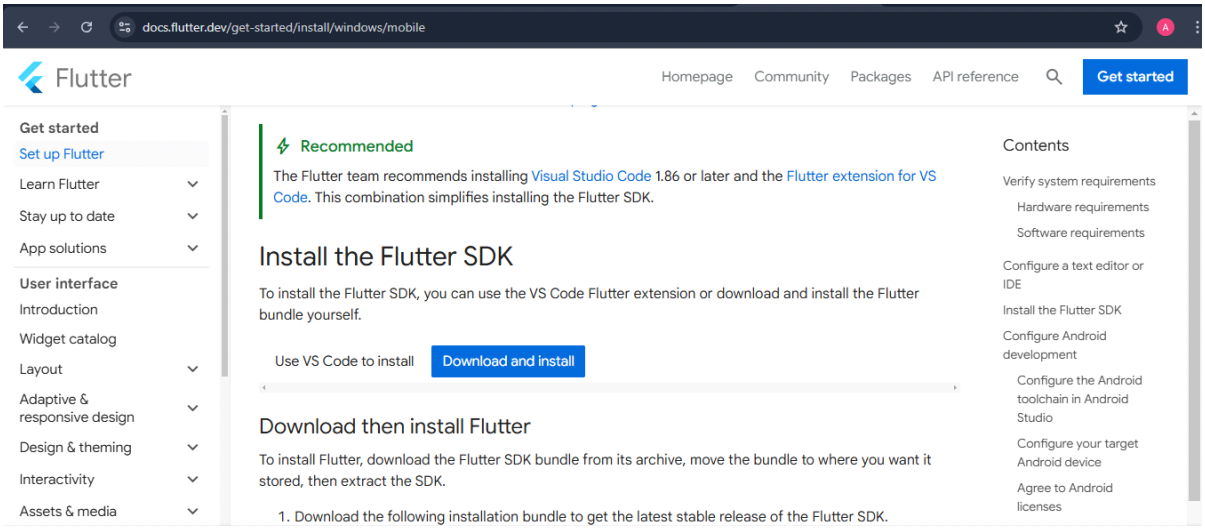
Step 1: Download the installation bundle of the Flutter Software Development Kit for windows. To download Flutter SDK, Go to its official website https://docs.flutter.dev/get-started/install , you will get the following screen.





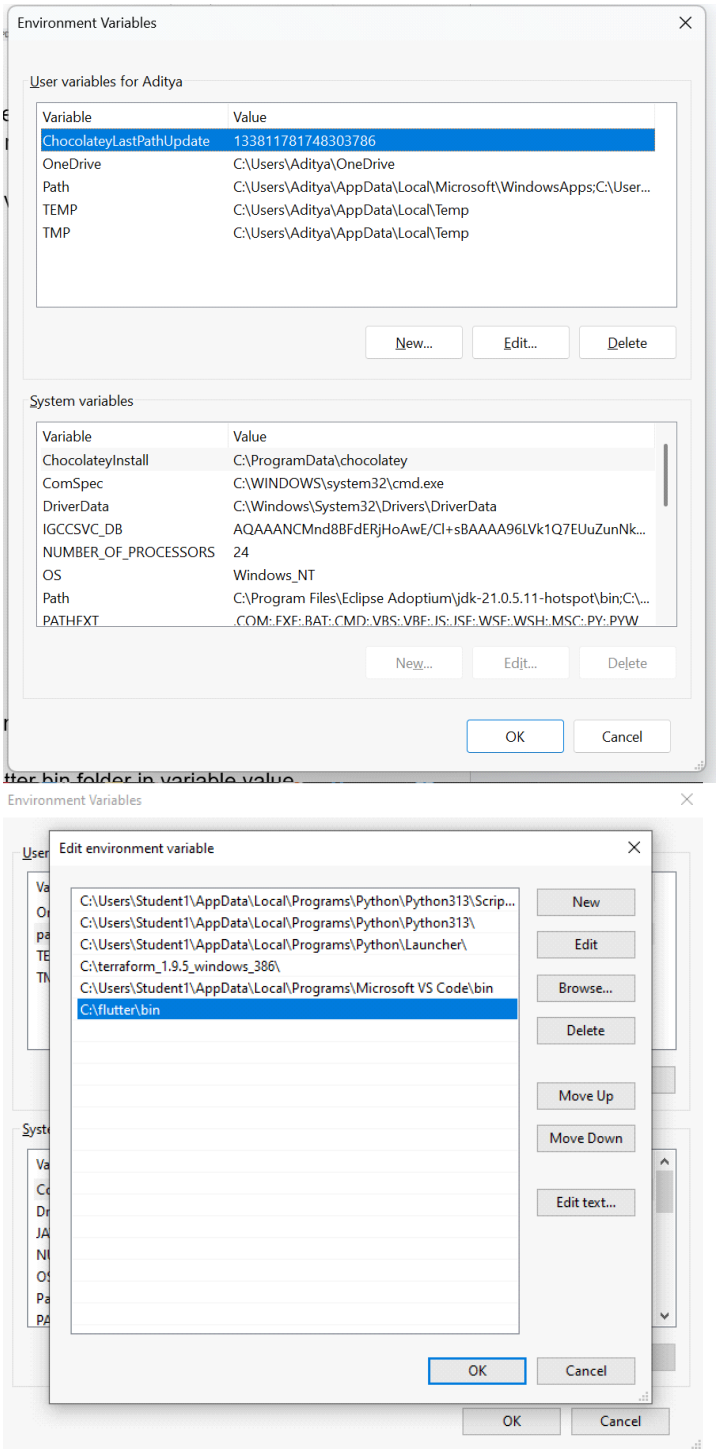
Step 2: Next, to download the latest Flutter SDK, click on the Windows icon. Here, you will find the download link for SDK.

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 command prompt.

```
C:\Users\Student>flutter --version
Flutter 3.19.2 • channel stable • https://github.com/flutter/flutter.git
Framework • revision 7482962148 (12 months ago) • 2024-02-27 16:51:22 -0500
Engine • revision 04817c99c9
Tools • Dart 3.3.0 • DevTools 2.31.1

C:\Users\Student>
```

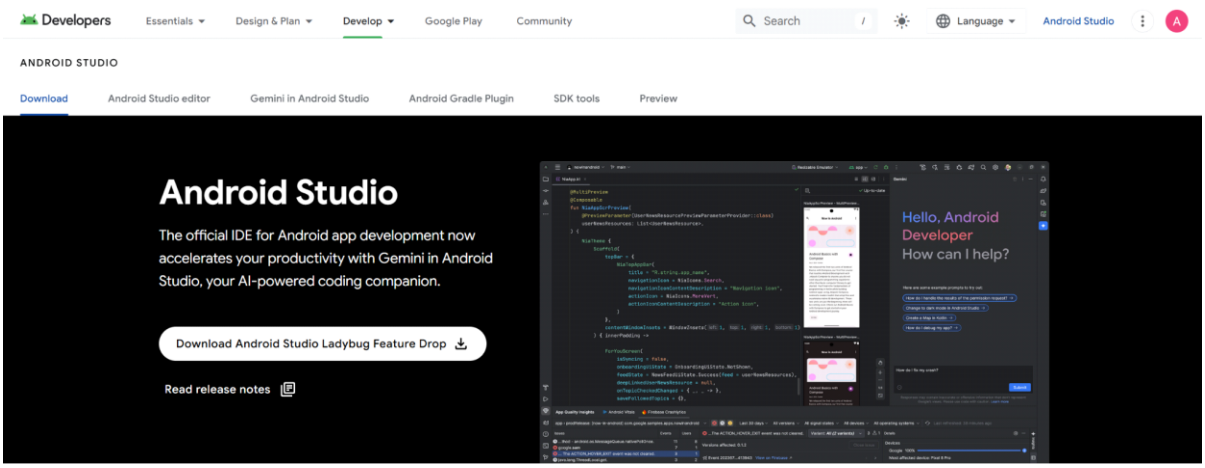
Now, run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation.

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.

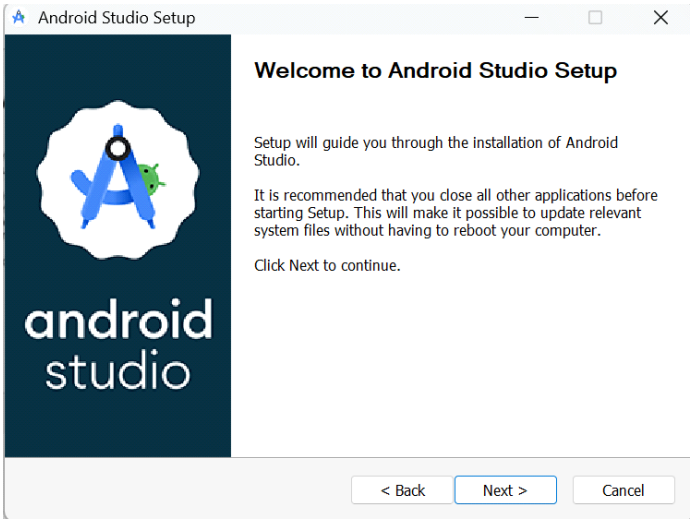
```
C:\Users\Student>echo abhay
abhay

C:\Users\Student>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.19.2, on Microsoft Windows [Version 10.0.22631.4751], locale en-US)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[X] Android toolchain - develop for Android devices
    X Unable to locate Android SDK.
      Install Android Studio from: https://developer.android.com/studio/index.html
      On first launch it will assist you in installing the Android SDK components.
      (or visit https://flutter.dev/docs/get-started/install/windows#android-setup for detailed instructions).
      If the Android SDK has been installed to a custom location, please use
      `flutter config --android-sdk` to update to that location.

[√] Chrome - develop for the web
[√] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.4.3)
[√] Android Studio (version 2024.2)
[√] VS Code (version 1.97.0)
[√] Connected device (3 available)
[√] Network resources

! Doctor found issues in 1 category.
```

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.

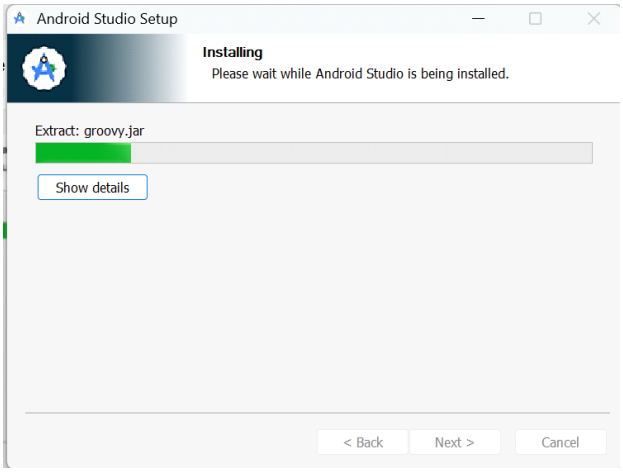
Step 7.1: Download the latest Android Studio executable or zip file from the official site.



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



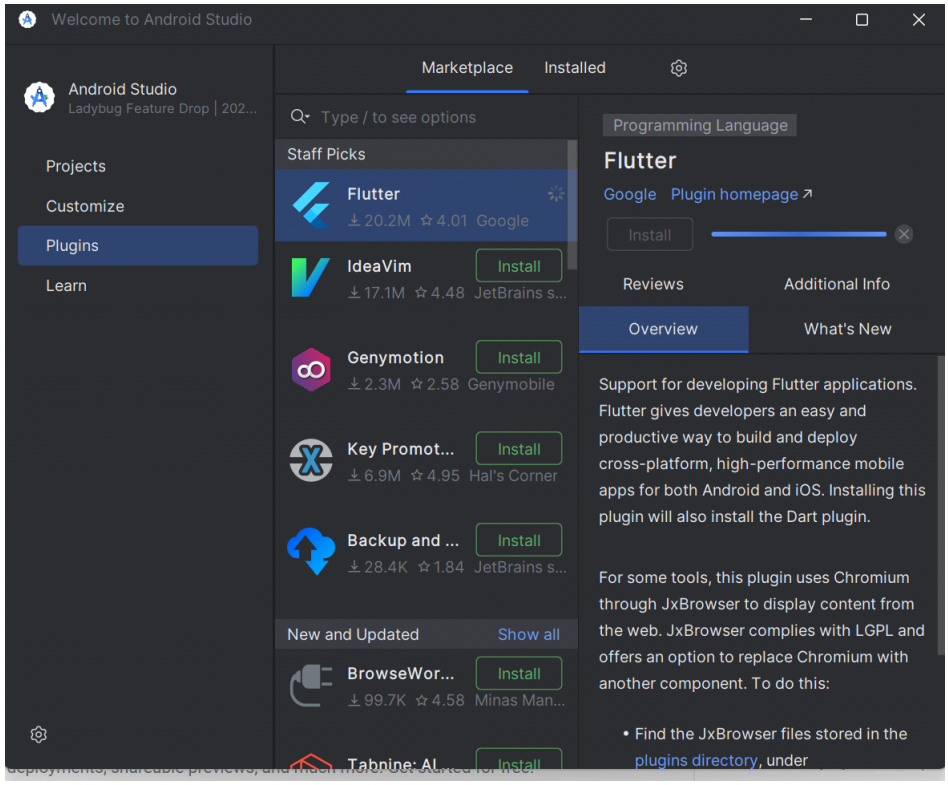
Step 7.3: Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.



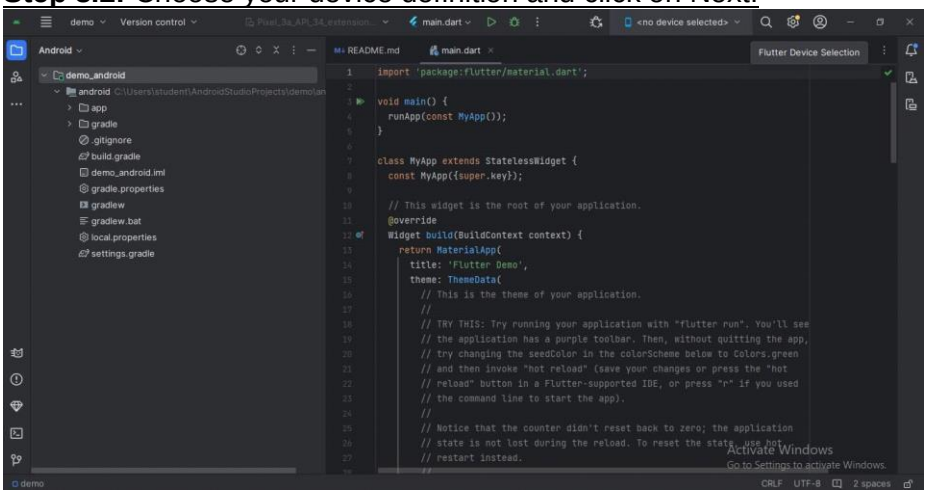
Step 7.4: In the above screen, click Next-> Finish. Once the Finish button is clicked, you need to choose the 'Don't import Settings option' and click OK. It will start the Android Studio.

Step 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.

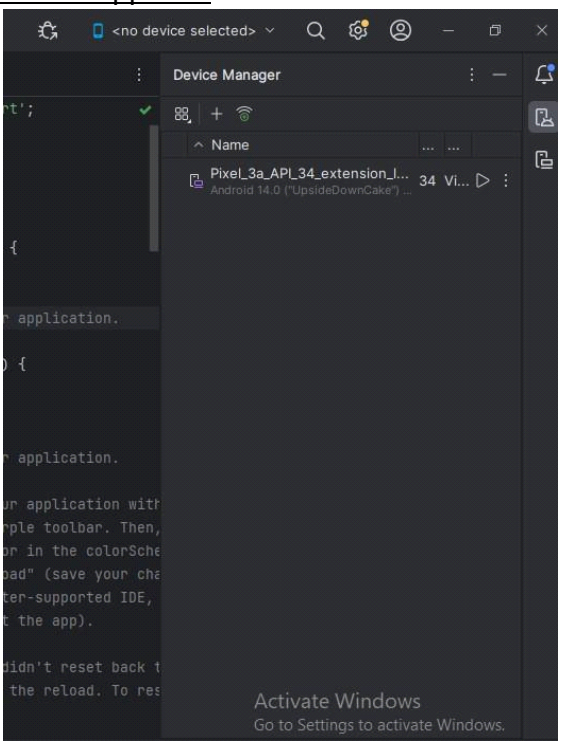


Step 8.2: Choose your device definition and click on Next.

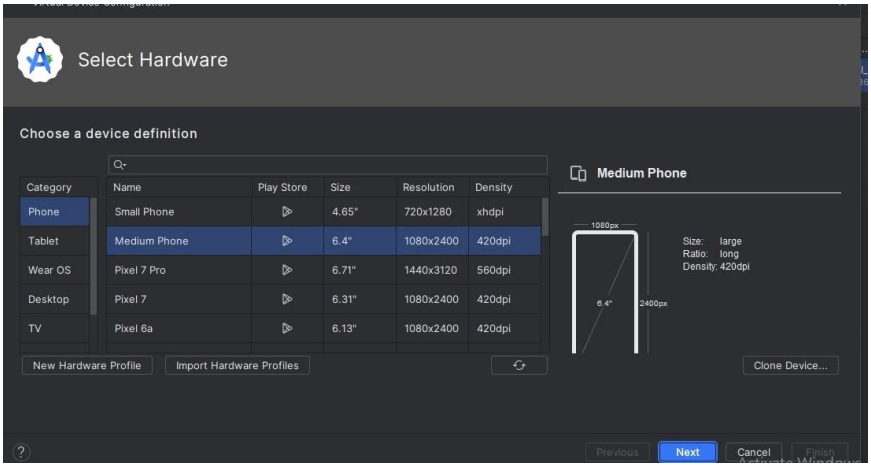


Step 8.3: Select the system image for the latest Android version and click on Next.

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

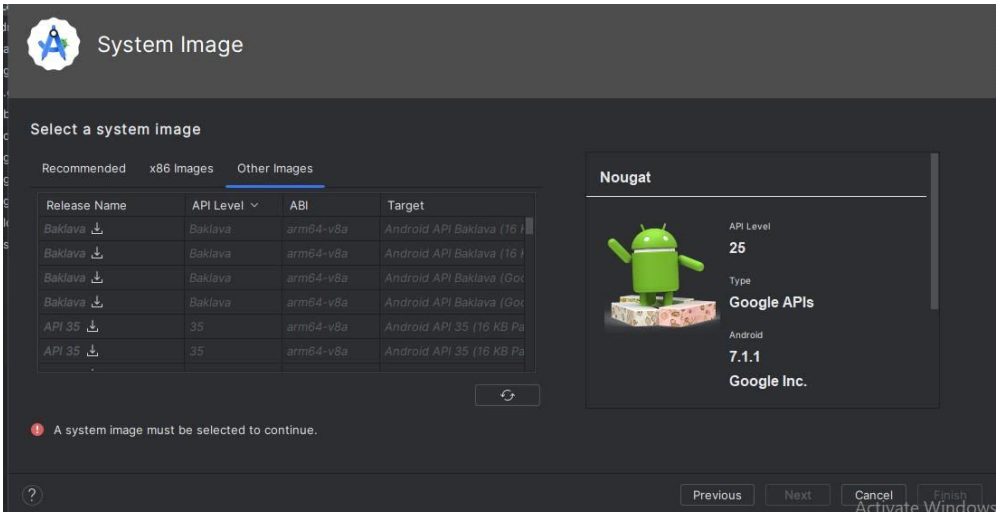


Step 8.5: Last, click on the icon pointed into the red color rectangle. The Android emulator



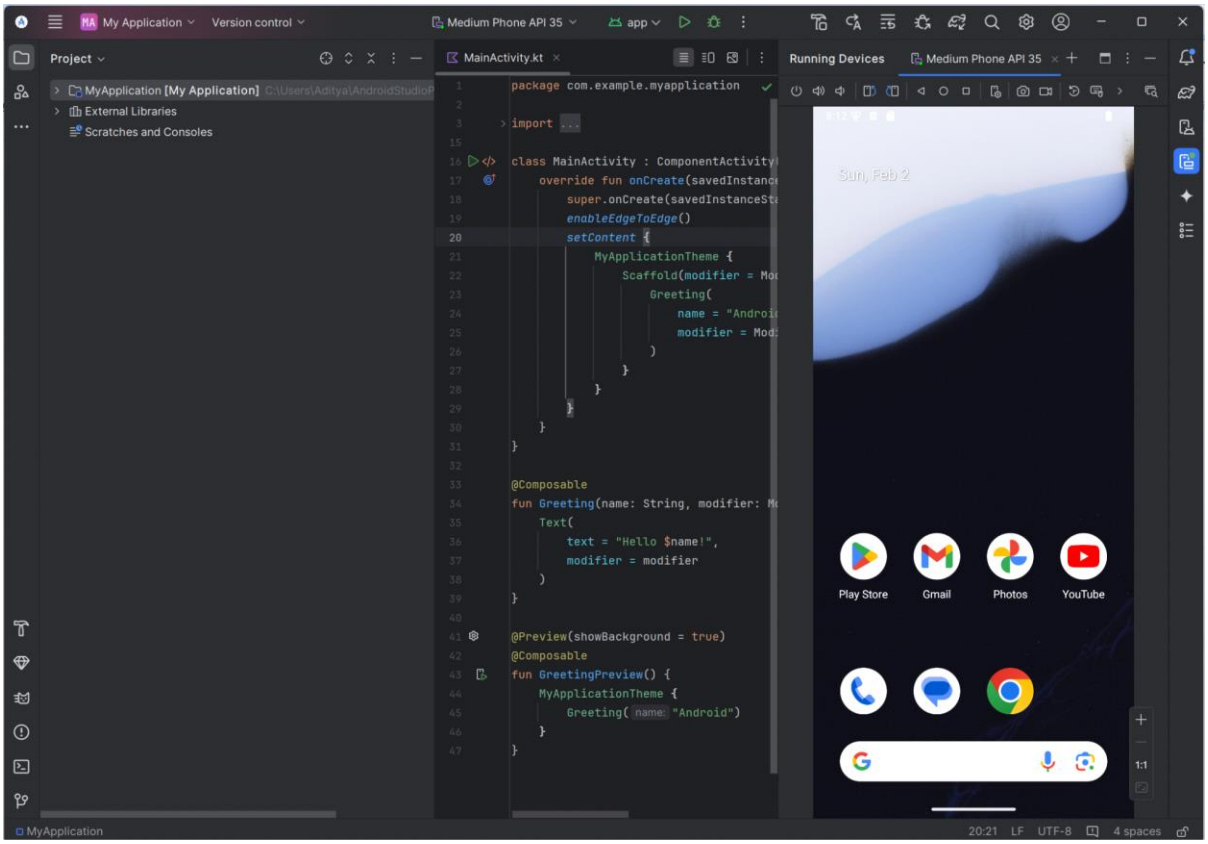
Step 9: Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself. Do the following steps to install these plugins.

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 : Abhay Gupta
Div : D15B
Roll No : 16

MPL Practical 02

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

Theory:

Flutter UI Design using Common Widgets

Flutter is a framework used for building mobile applications for Android and iOS. It allows developers to create beautiful and responsive user interfaces using widgets. Widgets are the basic building blocks of a Flutter app, and everything in Flutter is a widget, including buttons, text fields, and layout structures.

Common Widgets in Flutter

Flutter provides many built-in widgets that help in designing the UI of an application. Some of the commonly used widgets are:

1. Scaffold – Provides a basic structure with an app bar, body, and floating action button.
2. AppBar – Displays the title of the application and actions like buttons.
3. Text – Used to display text content in the app.
4. ListView – Helps in displaying a list of items in a scrollable manner.
5. Card – Used to display information inside a container with a shadow effect.
6. ElevatedButton – A button that performs an action when pressed.
7. TextField – Allows users to input text.
8. AlertDialog – Displays a pop-up dialog with options.

Implementation in Our Code

In our Sports Community Builder app, we have designed a simple user interface using common Flutter widgets. The main objective of this app is to allow users to form teams in their locality for instant play.

Features Implemented:

1. Displaying a List of Teams – We use ListView.builder() to show a list of available teams.
2. Adding a New Team – A TextField inside an AlertDialog allows users to enter a new team name.
3. Interactive Buttons – An ElevatedButton lets users create new teams.
4. Cards for Team Display – Each team name is displayed inside a Card for better presentation.
5. State Management – The list of teams is stored in a List<String> and updated dynamically using setState().

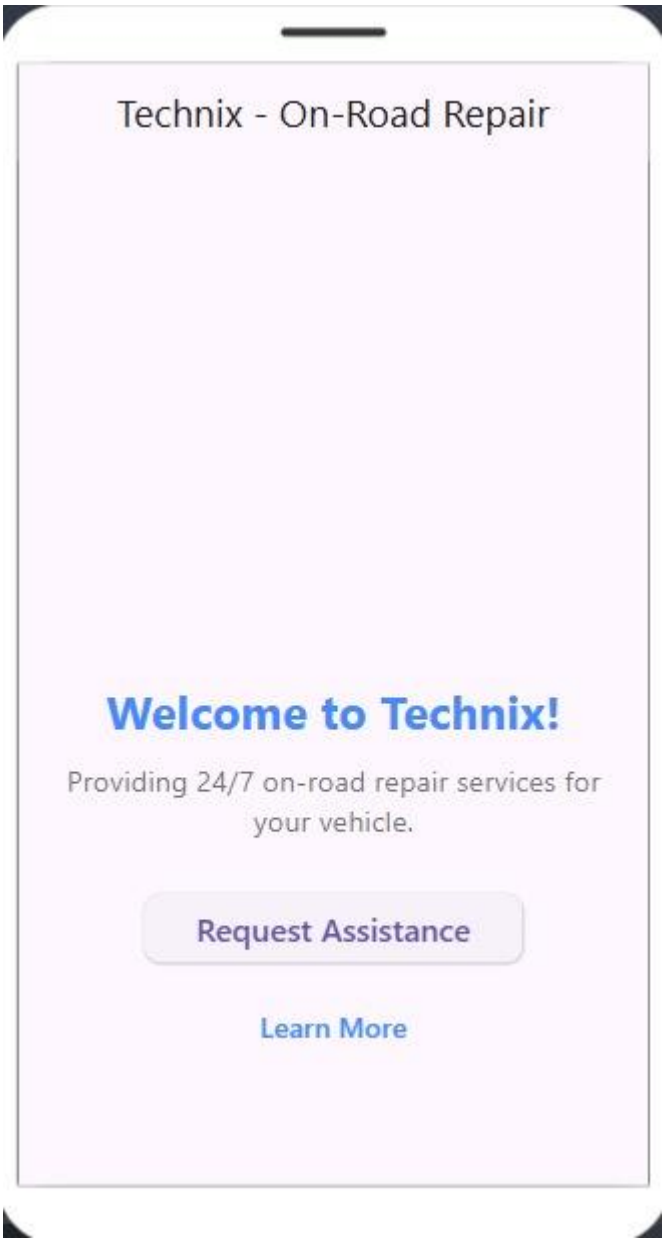
CODE:

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

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

class TechnixApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   debugShowCheckedModeBanner: false,
   theme: ThemeData(
    primarySwatch: Colors.blue,
    textTheme: GoogleFonts.poppinsTextTheme(),
   ),
   home: HomeScreen(),
  );
 }
}

class HomeScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   backgroundColor: Colors.blueAccent,
   appBar: AppBar(
    title: Text("Technix - Roadside Assistance"),
    backgroundColor: Colors.black87,
   ),
   body: Center(
    child: Column(
     mainAxisAlignment: MainAxisAlignment.center,
     children: [
      Text(
       "Need a Mechanic?",
       style: TextStyle(
        fontSize: 24,
        fontWeight: FontWeight.bold,
        color: Colors.white,
       ),
```

```
          ),
          SizedBox(height: 20),
          ElevatedButton(
           style: ElevatedButton.styleFrom(
            backgroundColor: Colors.orange,
            padding: EdgeInsets.symmetric(horizontal: 30, vertical: 15),
            shape: RoundedRectangleBorder(
             borderRadius: BorderRadius.circular(10),
            ),
           ),
           onPressed: () {
            Navigator.push(
             context,
             MaterialPageRoute(builder: (context) => RequestMechanicScreen()),
            );
           },
           child: Text(
            "Request a Mechanic",
            style: TextStyle(fontSize: 18, color: Colors.white),
           ),
          ),
         ],
        ),
       ),
      );
  }
}

class RequestMechanicScreen extends StatefulWidget {
  @override
  _RequestMechanicScreenState createState() => _RequestMechanicScreenState();
}

class _RequestMechanicScreenState extends State<RequestMechanicScreen> {
  TextEditingController _locationController = TextEditingController();

  void _submitRequest() {
    Fluttertoast.showToast(msg: "Mechanic Requested at ${_locationController.text}", toastLength:
Toast.LENGTH_SHORT);
  }

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

```
    return Scaffold(
      appBar: AppBar(
        title: Text("Request a Mechanic"),
        backgroundColor: Colors.black87,
      ),
      body: Padding(
        padding: EdgeInsets.all(20.0),
        child: Column(
          children: [
            TextField(
              controller: _locationController,
              decoration: InputDecoration(
                labelText: "Enter Your Location",
                border: OutlineInputBorder(),
              ),
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: _submitRequest,
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.green,
                padding: EdgeInsets.symmetric(horizontal: 30, vertical: 15),
              ),
              child: Text("Submit Request", style: TextStyle(fontSize: 16, color: Colors.white)),
            ),
          ],
        ),
      ),
    );
  }
}
```

OUTPUT:



Conclusion:

In this experiment, we successfully designed the UI for our Sports Community Builder app using common Flutter widgets like ListView, Card, TextField, and ElevatedButton. Initially, we faced errors related to updating the list dynamically and handling the pop-up dialog, but we resolved them by using setState() for real-time UI updates and ensuring proper text input handling.

Abhay Gupta

D15B

18

MPL Practical-3

Aim:

To learn how to include and use icons, images, and custom fonts in a Flutter application.

Theory:

Flutter allows developers to enhance their app's UI by adding icons, images, and custom fonts. This improves the app's aesthetics and usability.

1. Adding Icons in Flutter

Flutter provides built-in icons through the Icons class and also supports custom icons.

- **Using Built-in Icons:**

dart

CopyEdit

```
Icon(Icons.home, size: 50, color: Colors.blue)
```

- **Using Custom Icons (from Assets, such as FontAwesome):**
    - Add the package to pubspec.yaml:

yaml

CopyEdit

```
dependencies:
 font_awesome_flutter: ^10.4.0
```

    - Use the icon:

dart

CopyEdit

```
Icon(FontAwesomeIcons.car, size: 50, color: Colors.red)
```

---

2. Adding Images in Flutter

Flutter supports images from different sources:

- **Adding Asset Images:**
    1. Place the image inside the assets/images/ folder.

2. Declare it in pubspec.yaml:

yaml

CopyEdit

flutter:

assets:

- assets/images/car.png

3. Use it in the app:

dart

CopyEdit

Image.asset('assets/images/car.png', width: 100, height: 100)

- **Using Network Images:**

dart

CopyEdit

Image.network('https://example.com/image.jpg', width: 100, height: 100)

---

3. Adding Custom Fonts in Flutter

Custom fonts allow better UI customization.

- **Steps to Add Custom Fonts:**
  1. Place the font files inside the assets/fonts/ folder.
  2. Declare them in pubspec.yaml:

yaml

CopyEdit

flutter:

fonts:

- family: CustomFont

fonts:

- asset: assets/fonts/CustomFont-Regular.ttf
- asset: assets/fonts/CustomFont-Bold.ttf

weight: 700

3. Use the font in the app:

dart

CopyEdit

Text('Welcome to Flutter!', style: TextStyle(fontFamily: 'CustomFont', fontSize: 20))

Code:

```
// Add this to your pubspec.yaml under flutter section:
// fonts:
//   - family: TechnixIcons
//     fonts:
//       - asset: assets/fonts/TechnixIcons.ttf

import 'package:flutter/material.dart';

class TechnixIcons {
  // App Bar & Navigation Icons
  static const IconData appLogo = Icons.build_circle; // App logo
  static const IconData menu = Icons.menu; // Drawer menu
  static const IconData home = Icons.home_rounded; // Home screen
  static const IconData search = Icons.search; // Search functionality
  static const IconData notification = Icons.notifications; // Notifications

  // Authentication & Profile Icons
  static const IconData login = Icons.login; // Login
  static const IconData logout = Icons.logout; // Logout
  static const IconData profile = Icons.person; // User profile
  static const IconData email = Icons.email; // Email field
  static const IconData password = Icons.lock; // Password field
  static const IconData eye = Icons.visibility; // Password visibility
  static const IconData eyeOff = Icons.visibility_off; // Password hidden
  static const IconData edit = Icons.edit; // Edit profile

  // Car Service Related Icons
  static const IconData car = Icons.directions_car; // Car icon
```

```
static const IconData repair = Icons.build; // Repair services
static const IconData diagnostic = Icons.memory; // Car diagnostic
static const IconData tire = Icons.tire_repair; // Tire service
static const IconData battery = Icons.battery_full; // Battery service
static const IconData oil = Icons.opacity; // Oil change
static const IconData engine = Icons.settings; // Engine service
static const IconData ac = Icons.ac_unit; // AC service
static const IconData brake = Icons.do_not_step; // Brake service
static const IconData wash = Icons.local_car_wash; // Car wash

// Location & Navigation Icons
static const IconData location = Icons.location_on; // Location marker
static const IconData navigation = Icons.navigation; // Navigation
static const IconData map = Icons.map; // Map view
static const IconData direction = Icons.directions; // Get directions

// Booking & Payment Icons
static const IconData calendar = Icons.calendar_today; // Booking calendar
static const IconData time = Icons.access_time; // Time selection
static const IconData payment = Icons.payment; // Payment
static const IconData wallet = Icons.account_balance_wallet; // Wallet
static const IconData creditCard = Icons.credit_card; // Credit card
static const IconData offer = Icons.local_offer; // Offers/Discounts

// Communication Icons
static const IconData chat = Icons.chat; // Chat with mechanic
static const IconData call = Icons.call; // Call support
static const IconData support = Icons.support_agent; // Customer support
static const IconData feedback = Icons.feedback; // User feedback

// Status & Progress Icons
```

```
  static const IconData pending = Icons.pending; // Pending status
  static const IconData inProgress = Icons.engineering; // Service in progress
  static const IconData complete = Icons.check_circle; // Service complete
  static const IconData cancelled = Icons.cancel; // Cancelled service
  static const IconData warning = Icons.warning; // Warning/Alert

  // Misc Utility Icons
  static const IconData share = Icons.share; // Share functionality
  static const IconData favorite = Icons.favorite; // Favorites/Wishlist
  static const IconData camera = Icons.camera_alt; // Camera for photos
  static const IconData attachment = Icons.attach_file; // File attachments
  static const IconData filter = Icons.filter_list; // Filter options
  static const IconData sort = Icons.sort; // Sort functionality
  static const IconData info = Icons.info; // Information
  static const IconData help = Icons.help; // Help section
}
```

Conclusion:

By following these steps, icons, images, and fonts can be seamlessly integrated into a Flutter app. This enhances the visual appeal, user engagement, and branding of the application, making it more attractive and interactive.

Name: Abhay Gupta

Div: D15 B

Roll No.: 18

MPL Practical 4

Aim:

To design and implement an interactive form using the Form widget in Flutter that validates user input for fields such as email and password.

Theory:

Forms are an essential part of user interaction in mobile applications, allowing users to input and submit data. In Flutter, the Form widget is used to group multiple form fields (TextFormField), enabling validation and submission handling.

- **Form Widget:**
  - A container for grouping multiple form fields.
  - Uses a GlobalKey<FormState> to validate and manage state.
- **TextFormField Widget:**
  - Accepts user input with built-in validation.
  - Provides error messages when validation fails.
- **Form Validation:**
  - Ensures input fields contain valid data before submission.
  - Uses the validator function to check conditions such as email format and password length.
- **Submission Handling:**
  - The FormState class provides methods like validate() and save() to handle form submission.
  - If validation succeeds, the form processes the data; otherwise, error messages are displayed.

Code:

```
import 'package:flutter/material.dart';
void main() {
  runApp(const MyApp());
}
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
```

```
    return MaterialApp(
      debugShowCheckedModeBanner: false, // Removes the debug banner
      title: 'Technix Login',
      theme: ThemeData.dark().copyWith(
        primaryColor: Colors.red,
        scaffoldBackgroundColor: Colors.black,),
      home: const LoginPage(),
    );}}
class LoginPage extends StatefulWidget {
  const LoginPage({Key? key}) : super(key: key);
@override
  State<LoginPage> createState() => _LoginPageState();}
class _LoginPageState extends State<LoginPage> {
  final _formKey = GlobalKey<FormState>();
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();
  bool _isPasswordVisible = false;
  @override
  void dispose() {
    _emailController.dispose();
    _passwordController.dispose();
    super.dispose();
  }
  String? _validateEmail(String? value) {
    if (value == null || value.isEmpty) {
      return 'Email is required';
    }
    final emailRegex = RegExp(r'^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}\$');
    if (!emailRegex.hasMatch(value)) {
      return 'Please enter a valid email';
    }    return null;
  }
```

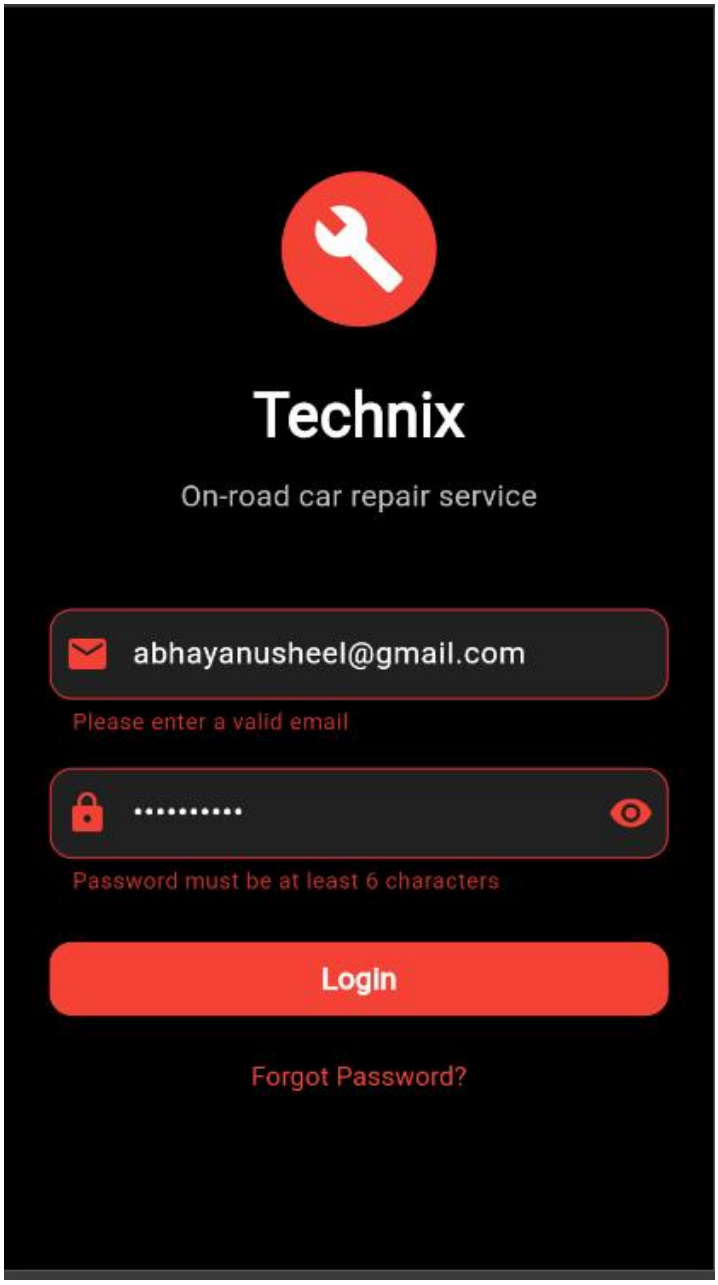
```
  String? _validatePassword(String? value) {
    if (value == null || value.isEmpty) {
      return 'Password is required';
    }
    if (value.length < 6) {
      return 'Password must be at least 6 characters';
    }
    return null;
  }

  void _handleLogin() {
    if (_formKey.currentState!.validate()) {
      print('Email: ${_emailController.text}');
      print('Password: ${_passwordController.text}');
    }
  }
@override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      body: SafeArea(
        child: Center(
          child: SingleChildScrollView(
            padding: const EdgeInsets.all(24.0),
            child: Form(
              key: _formKey,
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Container(
                    padding: const EdgeInsets.all(16.0),
                    decoration: BoxDecoration(
```

```
    color: Colors.red,
    shape: BoxShape.circle,
  ),
  child: Icon(
    Icons.build,
    size: 50,
    color: Colors.white,
  ),
),
const SizedBox(height: 24),
Text(
  'Technix',
  style: TextStyle(
    fontSize: 32,
    fontWeight: FontWeight.bold,
    color: Colors.white,
  ),
),
const SizedBox(height: 8),
Text(
  'On-road car repair service',
  style: TextStyle(
    fontSize: 16,
    color: Colors.grey[400],
  ),
),
const SizedBox(height: 48),
TextFormField(
  controller: _emailController,
  style: TextStyle(color: Colors.white),
  validator: _validateEmail,
  keyboardType: TextInputType.emailAddress,
```

```
  decoration: _inputDecoration('Email', Icons.email),
),
const SizedBox(height: 16),
TextFormField(
  controller: _passwordController,
  obscureText: !_isPasswordVisible,
  style: TextStyle(color: Colors.white),
  validator: _validatePassword,
  decoration: _inputDecoration('Password', Icons.lock).copyWith(
    suffixIcon: IconButton(
      icon: Icon(
        _isPasswordVisible ? Icons.visibility_off : Icons.visibility,
        color: Colors.red,
      ),
      onPressed: () {
        setState(() {
          _isPasswordVisible = !_isPasswordVisible;
        });
      },
    ),
  ),
),
const SizedBox(height: 24),
SizedBox(
  width: double.infinity,
  child: ElevatedButton(
    onPressed: _handleLogin,
    style: ElevatedButton.styleFrom(
      backgroundColor: Colors.red,
      padding: const EdgeInsets.symmetric(vertical: 16),
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(12),
```

```
                ),
              ),
              child: Text(
                'Login',
                style: TextStyle(
                  fontSize: 16,
                  fontWeight: FontWeight.bold,
                  color: Colors.white,
                ),
              ),
            ),
          ),
          const SizedBox(height: 16),
          TextButton(
            onPressed: () {},
            child: Text(
              'Forgot Password?',
              style: TextStyle(
                color: Colors.red,
                fontSize: 14,
              ),),),],),),),),), );  }
InputDecoration _inputDecoration(String hintText, IconData icon) {
  return InputDecoration(
    hintText: hintText,
    hintStyle: TextStyle(color: Colors.grey[400]),
    prefixIcon: Icon(icon, color: Colors.red),
    filled: true,
    fillColor: Colors.grey[900],
    enabledBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.transparent),      ),
    focusedBorder: OutlineInputBorder(
```

);}}



Conclusion:

By using the Form widget in Flutter, we can create interactive and user-friendly forms with validation. This approach enhances user experience by preventing incorrect data entry and ensuring only valid input is processed. The use of GlobalKey<FormState> enables efficient form validation and submission, making it an essential component in Flutter applications requiring user input.

Experiment 5: Applying Navigation, Routing, and Gestures in a Flutter App

Aim:

To implement navigation, routing, and gestures in a Flutter application to enhance user experience and interactivity.

Theory:

Navigation and Routing

Navigation in Flutter enables users to move between different screens (routes) within an application. There are two main approaches:

1. Basic Navigation

Navigator.push(): Moves to a new screen.

Navigator.pop(): Returns to the previous screen.

2. Named Routes Navigation

Routes are defined in the MaterialApp widget.

Navigator.pushNamed(): Navigates to a named route.

Navigator.pop(): Returns to the previous screen.

Code:

Basic Navigation

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

void main() {
  runApp(const MaterialApp(
    title: 'Navigation Basics',
    home: FirstRoute(),
  ));
}
```

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('First Route')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Open route'),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => const SecondRoute()),
            );
          },
        ),
      ),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Second Route')),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: const Text('Go back!'),
        ),
      ),
    );
  }
}
```

Named Routes Navigation Example

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

```
void main() {
  runApp(MaterialApp(
    title: 'Named Route Navigation',
    initialRoute: '/',
    routes: {
      '/': (context) => HomeScreen(),
      '/second': (context) => SecondScreen(),
    },
  ));
}

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Home Screen')),
      body: Center(
        child: ElevatedButton(
          child: Text('Click Here'),
          onPressed: () {
            Navigator.pushNamed(context, '/second');
          },
        ),
      ),
    );
  }
}

class SecondScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Second Screen")),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: Text('Go back!'),
        ),
      ),
    );
  }
```

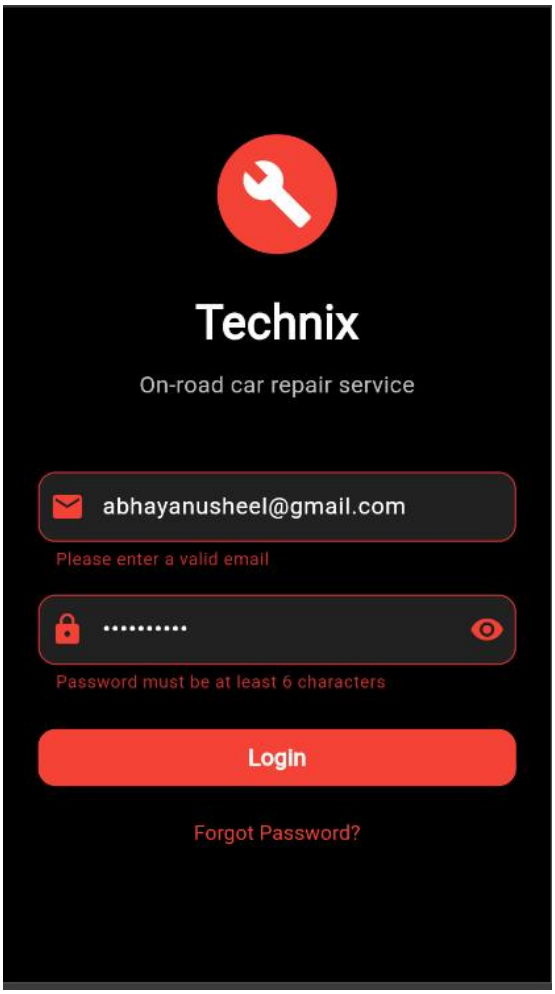
```
}
```

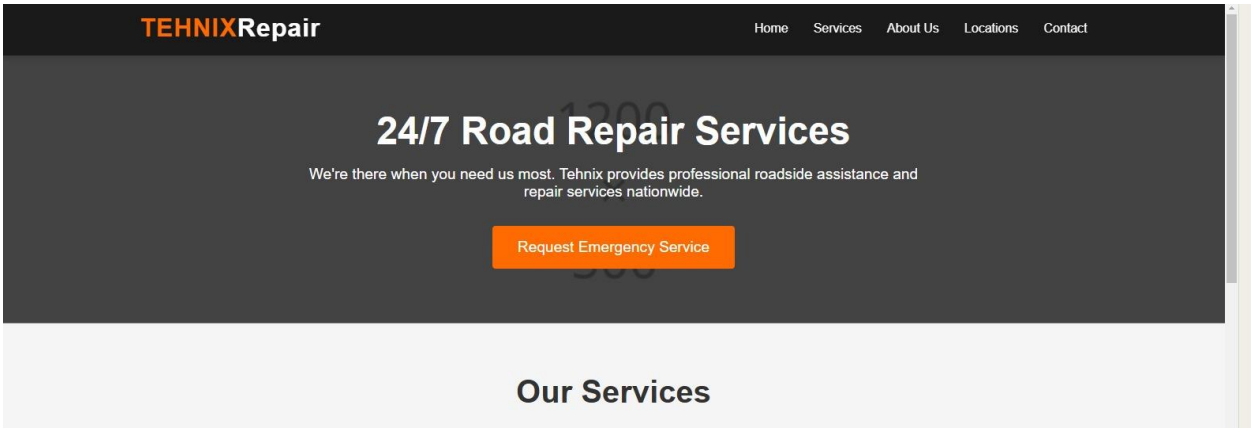
Gesture Handling Example

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Gesture Example',
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  MyHomePageState createState() => new MyHomePageState();
}

class MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Gestures Example')),
      body: Center(
        child: GestureDetector(
          onTap: () {
            print('Box Clicked');
          },
          child: Container(
            height: 60.0,
            width: 120.0,
            decoration: BoxDecoration(
              color: Colors.blueGrey,
              borderRadius: BorderRadius.circular(15.0),
            ),
            child: Center(child: Text('Click Me')),
          ),
        ),
      ),
```

```
    );
  }
}
```

Output:


Technix
On-road car repair service
abhayanusheel@gmail.com
Please enter a valid email
Password must be at least 6 characters
Login
Forgot Password?



Conclusion:

In this experiment, we successfully implemented navigation, routing, and gestures in a Flutter application. We explored both basic and named navigation techniques and used GestureDetector to detect user interactions. These functionalities enhance app usability by providing seamless screen transitions and interactive elements.

Experiment No: 06

How To Set Up Firebase with Flutter for iOS and Android Apps

Firebase is a great backend solution for anyone that wants to use authentication, databases, cloud functions, ads, and countless other features within an app.

In this article, you will create a Firebase project for iOS and Android platforms using

Flutter. Creating a New Flutter Project

This tutorial will require the creation of an example Flutter app.

Once you have your environment set up for Flutter, you can run the following to create a new application:

```
flutter create flutterfirebaseexample
```

Navigate to the new project directory:

```
cd flutterfirebaseexample
```

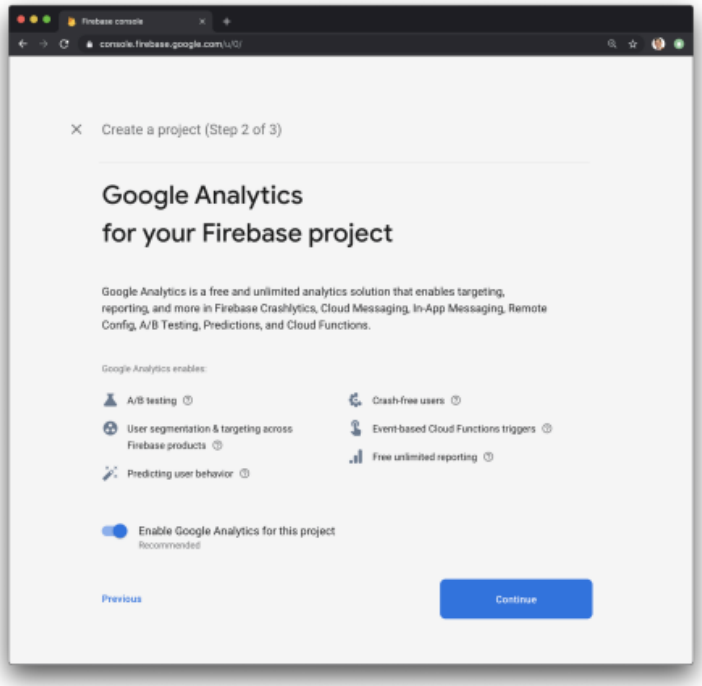
Using `flutter create` will produce a demo application that will display the number of times a button is clicked.

Now that we've got a Flutter project up and running, we can add

Firebase. Creating a New Firebase Project

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

Next, we're given the option to enable Google Analytics. This tutorial will not require Google Analytics, but you can also choose to add it to your project.



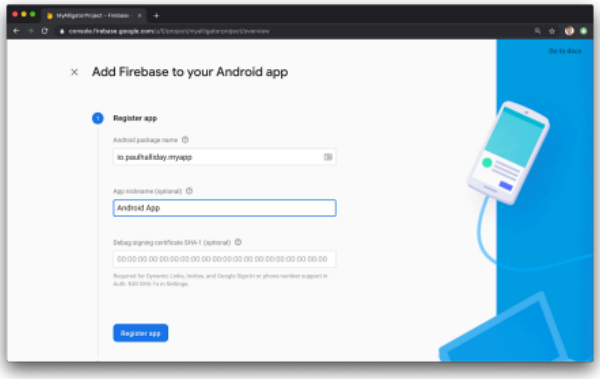
f you choose to use Google Analytics, you will need to review and accept the terms and conditions prior to project creation.

After pressing Continue, your project will be created and resources will be provisioned. You will then be directed to the dashboard for the new project.

Adding Android support

Registering the App

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



The most important thing here is to match up the Android package name that you choose here

with the one inside of our application.

The structure consists of at least two segments. A common pattern is to use a domain name, a company name, and the application name:

`com.example.flutterfirebaseexample`

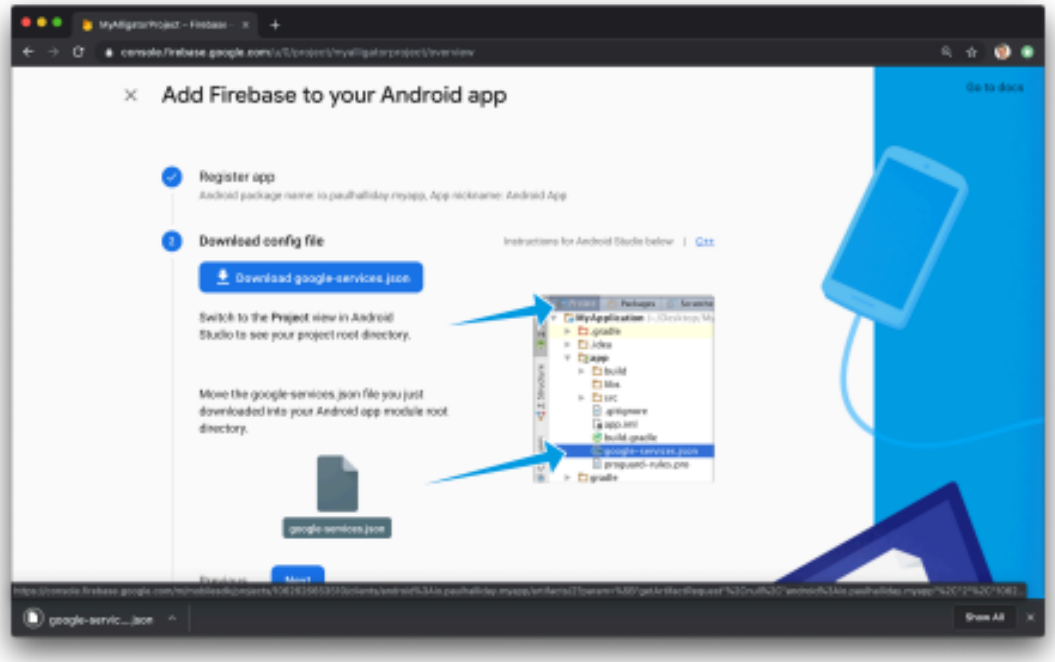
Once you've decided on a name, open `android/app/build.gradle` in your code editor and update the `applicationId` to match the Android package name:

android/app/build.gradle

Downloading the Config File

The next step is to add the Firebase configuration file into our Flutter project. This is important as it contains the API keys and other critical information for Firebase to use.

Select Download `google-services.json` from this page:



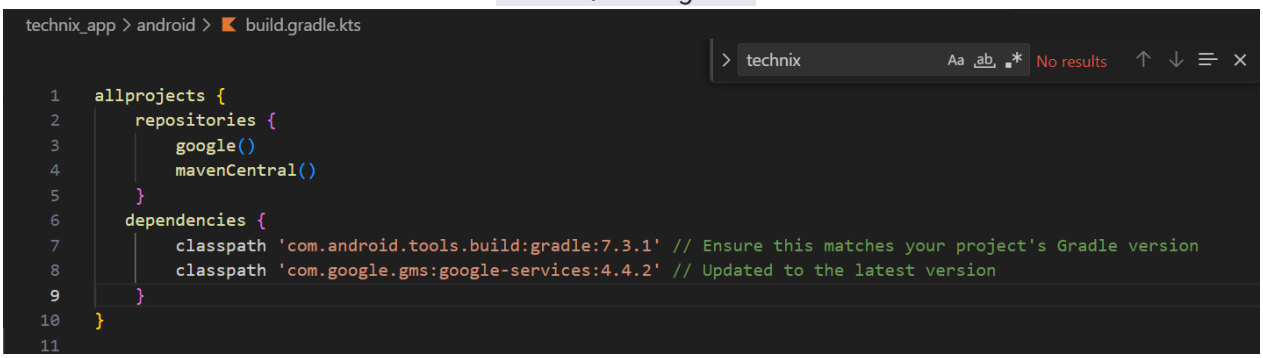
Next, move the `google-services.json` file to the `android/app` directory within the Flutter project.

Adding the Firebase SDK

We'll now need to update our Gradle configuration to include the Google Services plugin.

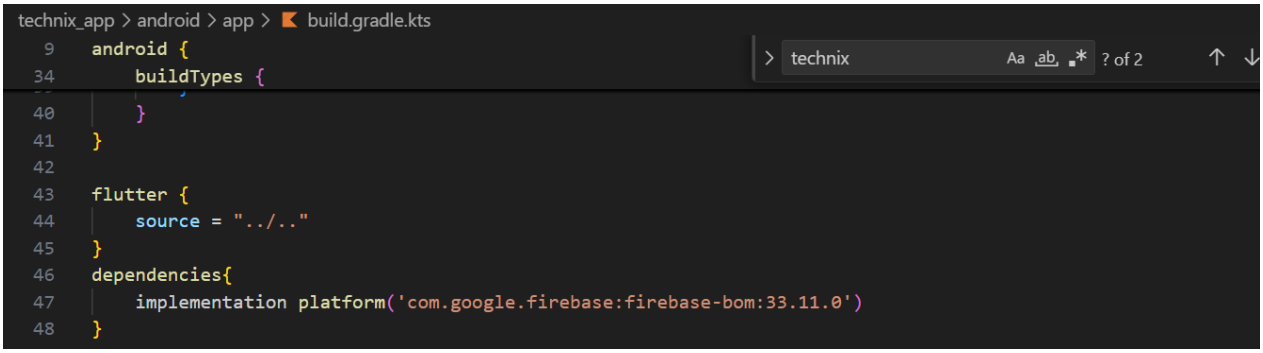
Open `android/build.gradle` in your code editor and modify it to include the following:

android/buiild.gradle



Finally, update the app level file at `android/app/build.gradle` to include the following:

android/app/build.gradle



With this update, we're essentially applying the Google Services plugin as well as looking at how other Flutter Firebase plugins can be activated such as Analytics.

From here, run your application on an Android device or simulator. If everything has worked correctly, you should get the following message in the dashboard:
Next up, let's add iOS support!

Adding iOS Support

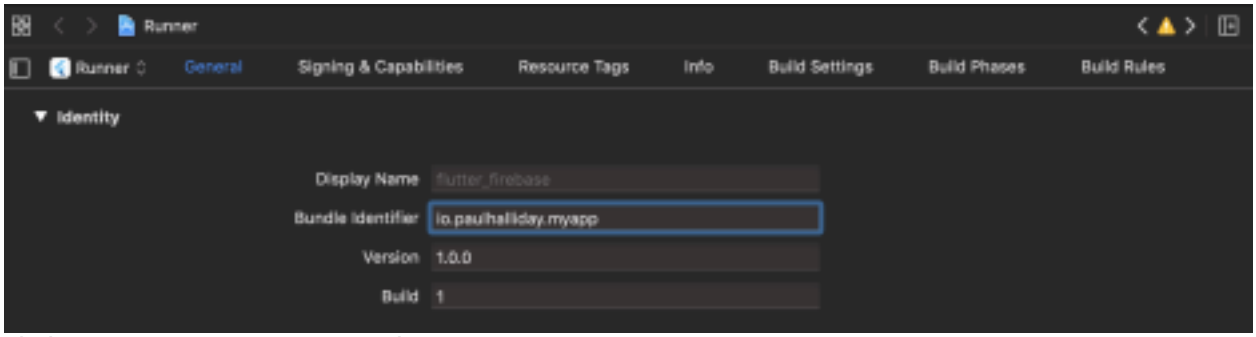
In order to add Firebase support for iOS, we have to follow a similar set of instructions.

Head back over to the dashboard and select Add app and then iOS icon to be navigated to the setup process.

Registering an App

Once again, we'll need to add an "iOS Bundle ID". It is possible to use the "Android package name" for consistency:

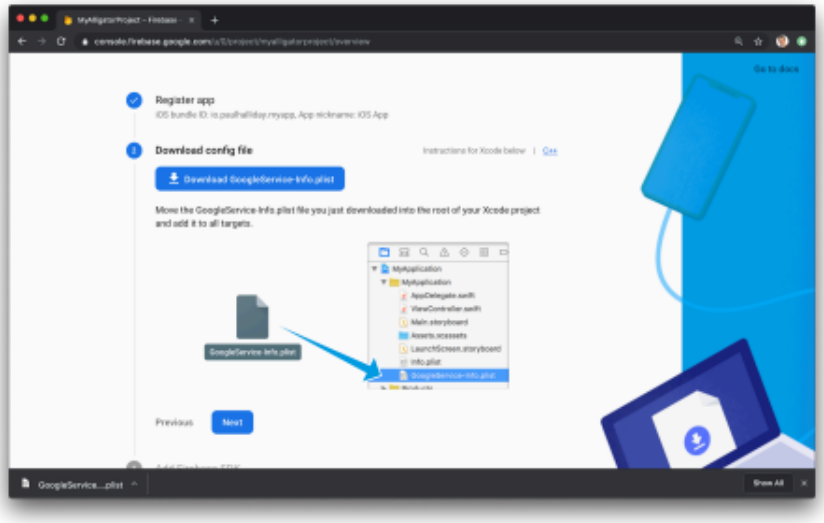
You'll then need to make sure this matches up by opening the iOS project up in Xcode at `ios/Runner/Runner.xcodeproj` and changing the Bundle identifier under General:



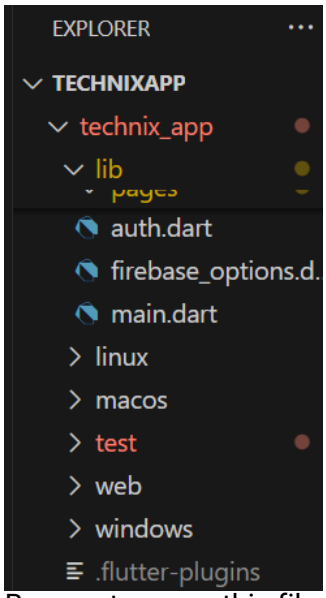
Click Register app to move to the next screen.

Downloading the Config File

In this step, we'll need to download the configuration file and add this to our Xcode project.

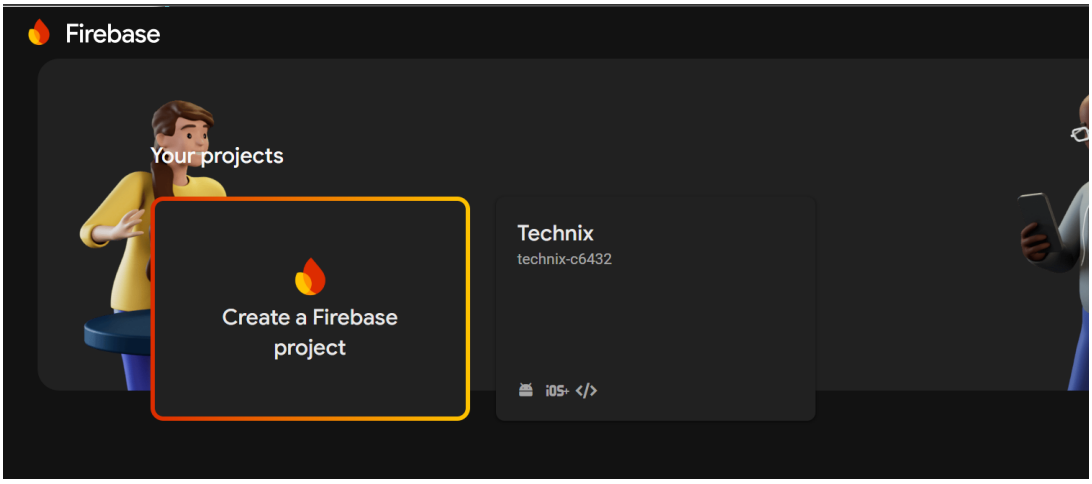


Download `GoogleService-Info.plist` and move this into the root of your Xcode project within `Runner`:



Be sure to move this file within Xcode to create the proper file references.
There are additional steps for installing the Firebase SDK and adding initialization code, but they are not necessary for this tutorial.

That's it!



Conclusion

In this article, you learned how to set up and ready our Flutter applications to be used with Firebase.

Flutter has official support for Firebase with the FlutterFire set of libraries.

Name : Abhay Gupta

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

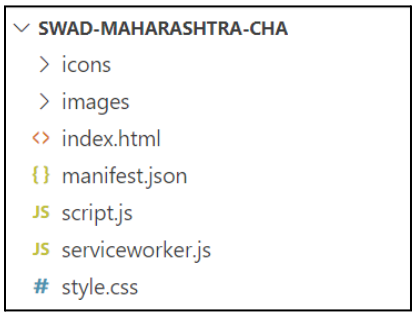
What We Implemented

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

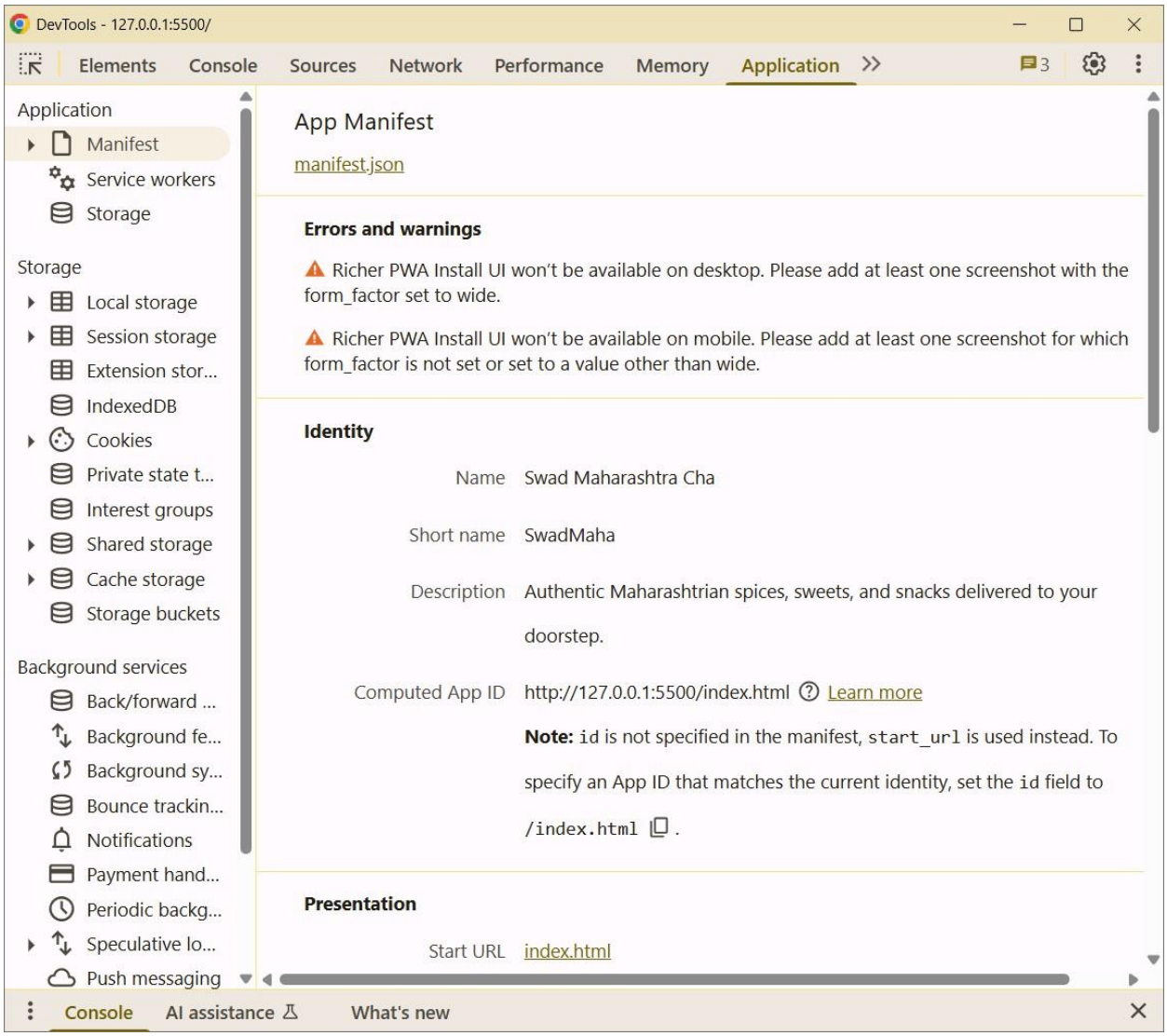
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <!-- 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) =>
```

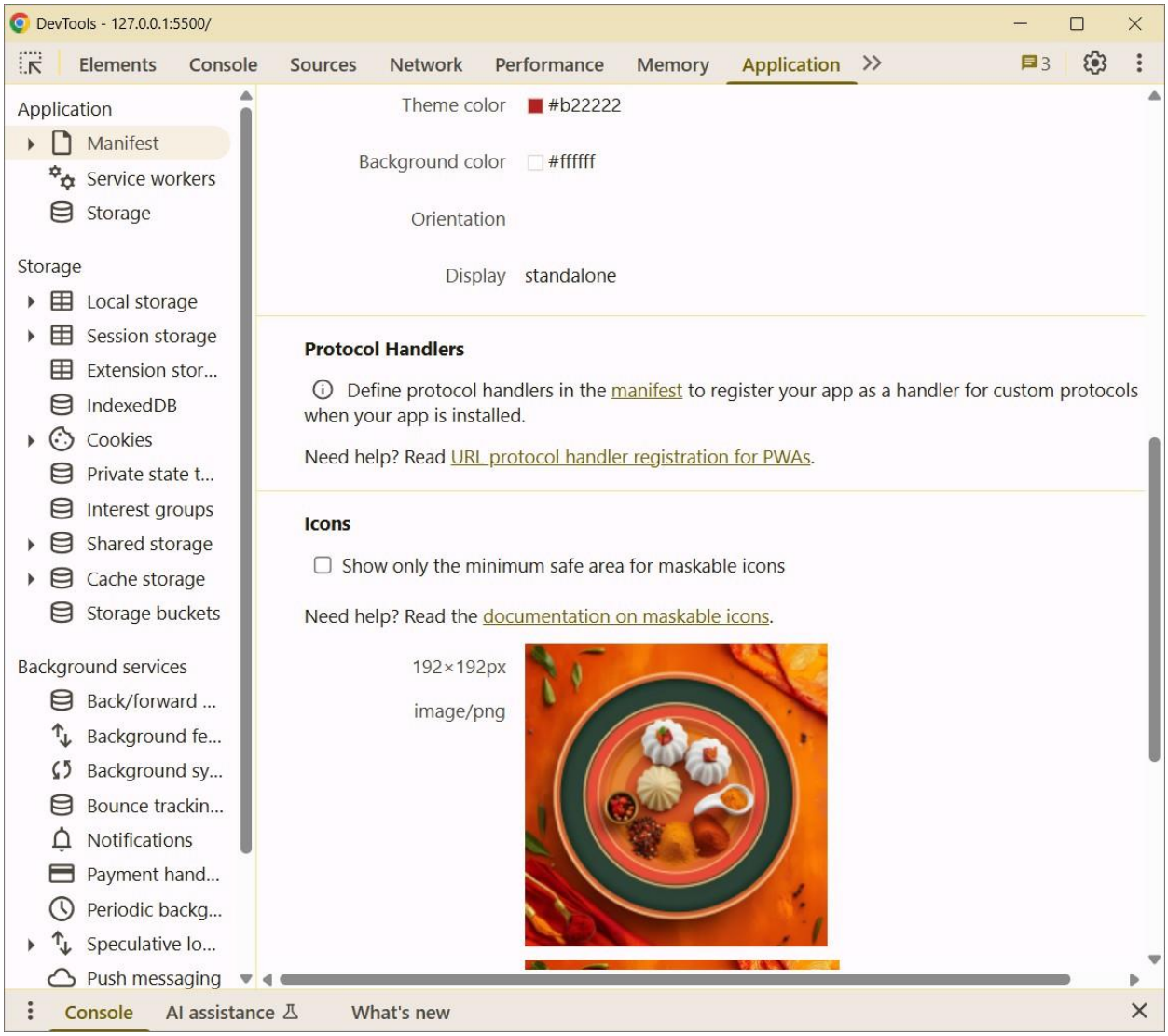
```
          console.log("Service Worker Registration Failed", error)
        );
    }
  </script>

  <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


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

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 : Abhay Gupta

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.

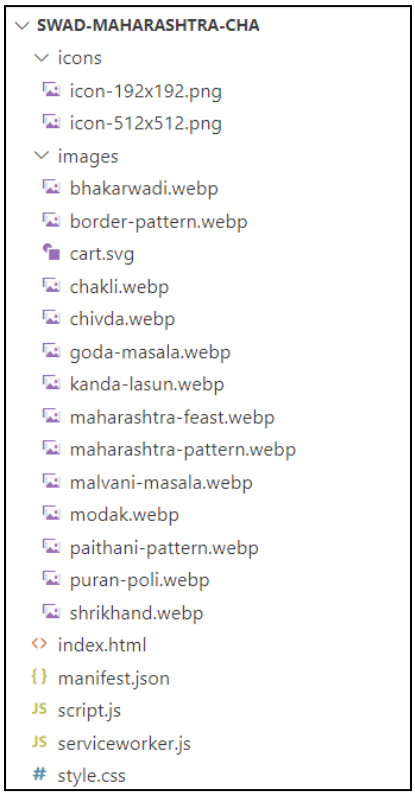
In this experiment, we learned how to register, install, and activate a service worker. The service worker listens for events like "install" and "activate" to manage caching of important files such as HTML, CSS, JavaScript, images, icons, and the manifest. Once installed, it stores these files in a cache 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) => {
```

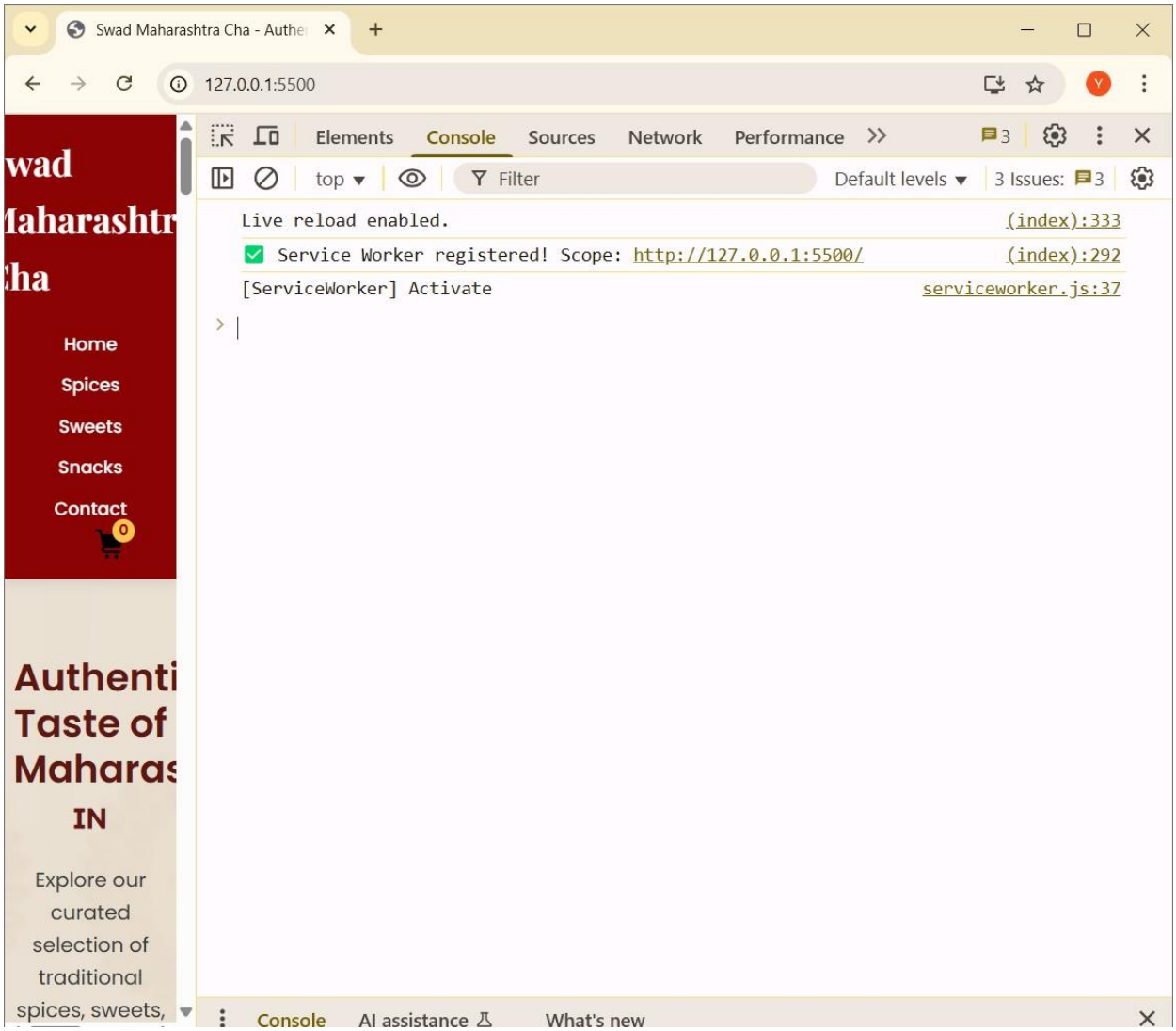
```
            console.log(
              "✅Service Worker registered! Scope:",
              registration.scope
            );
          })
          .catch((error) => {
            console.log("❌Service Worker registration failed:", error);
          });
      });
    }
  </script>

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

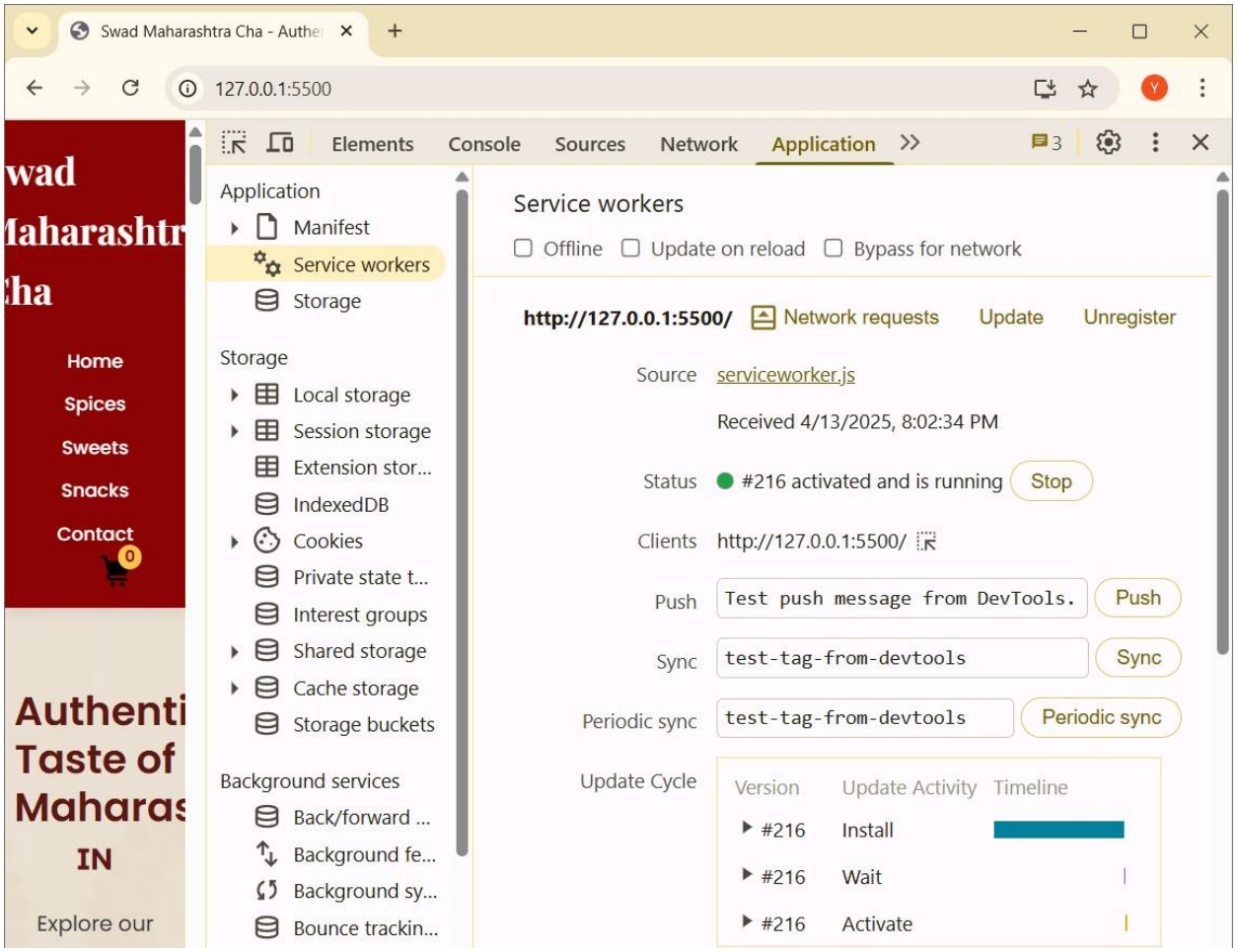
serviceworker.js

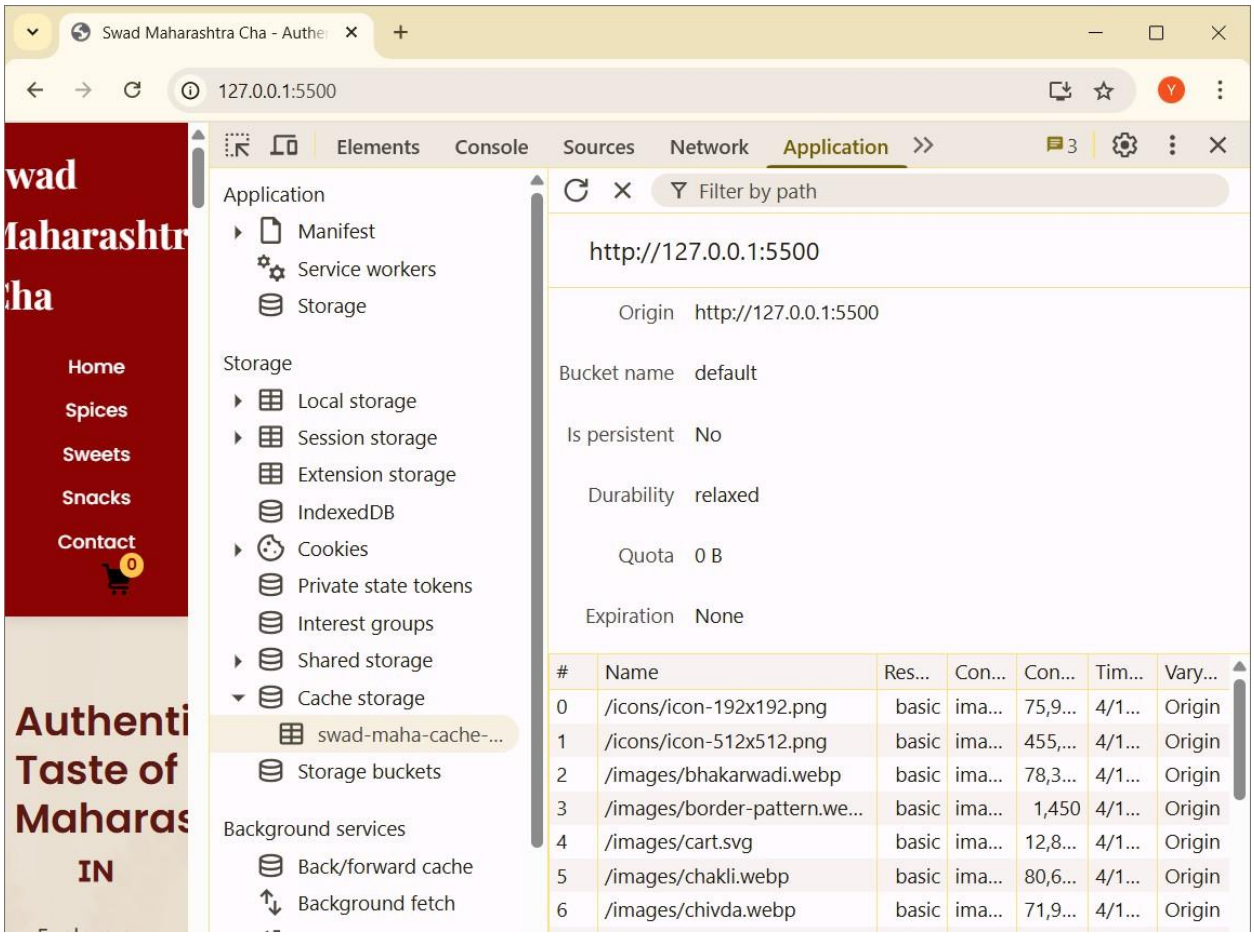
```
const CACHE_NAME = "swad-maha-cache-v1";
const FILES_TO_CACHE = [
  "index.html",
  "style.css",
  "script.js",
  "serviceworker.js",
  "manifest.json",
  "images/maharashtra-feast.webp",
  "images/cart.svg",
  "images/goda-masala.webp",
  "images/malvani-masala.webp",
  "images/kanda-lasun.webp",
  "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 - Authe
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
Home
Spices
Sweets
Snacks
Contact
IN
Explore our


Swad Maharashtra Cha - Authe
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
Home
Spices
Sweets
Snacks
Contact
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 : Abhay Gupta

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.

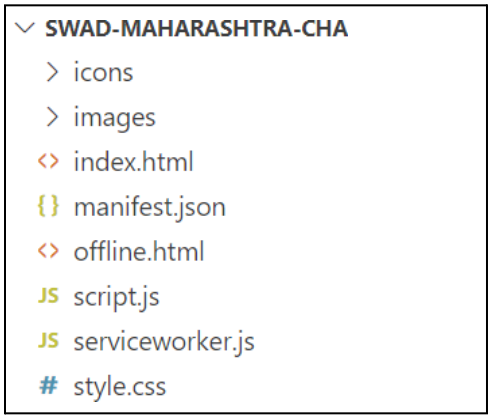
What We Implemented

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>
```

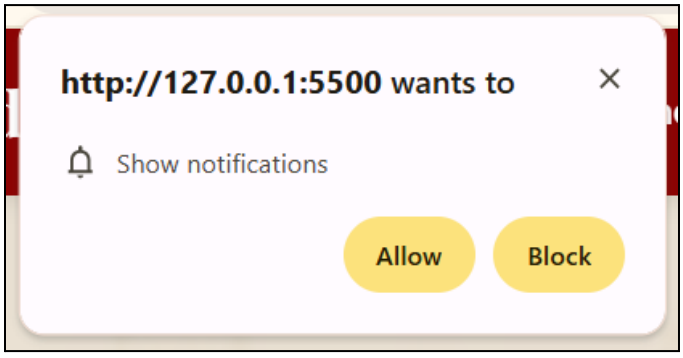
serviceworker.js

```
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(
```

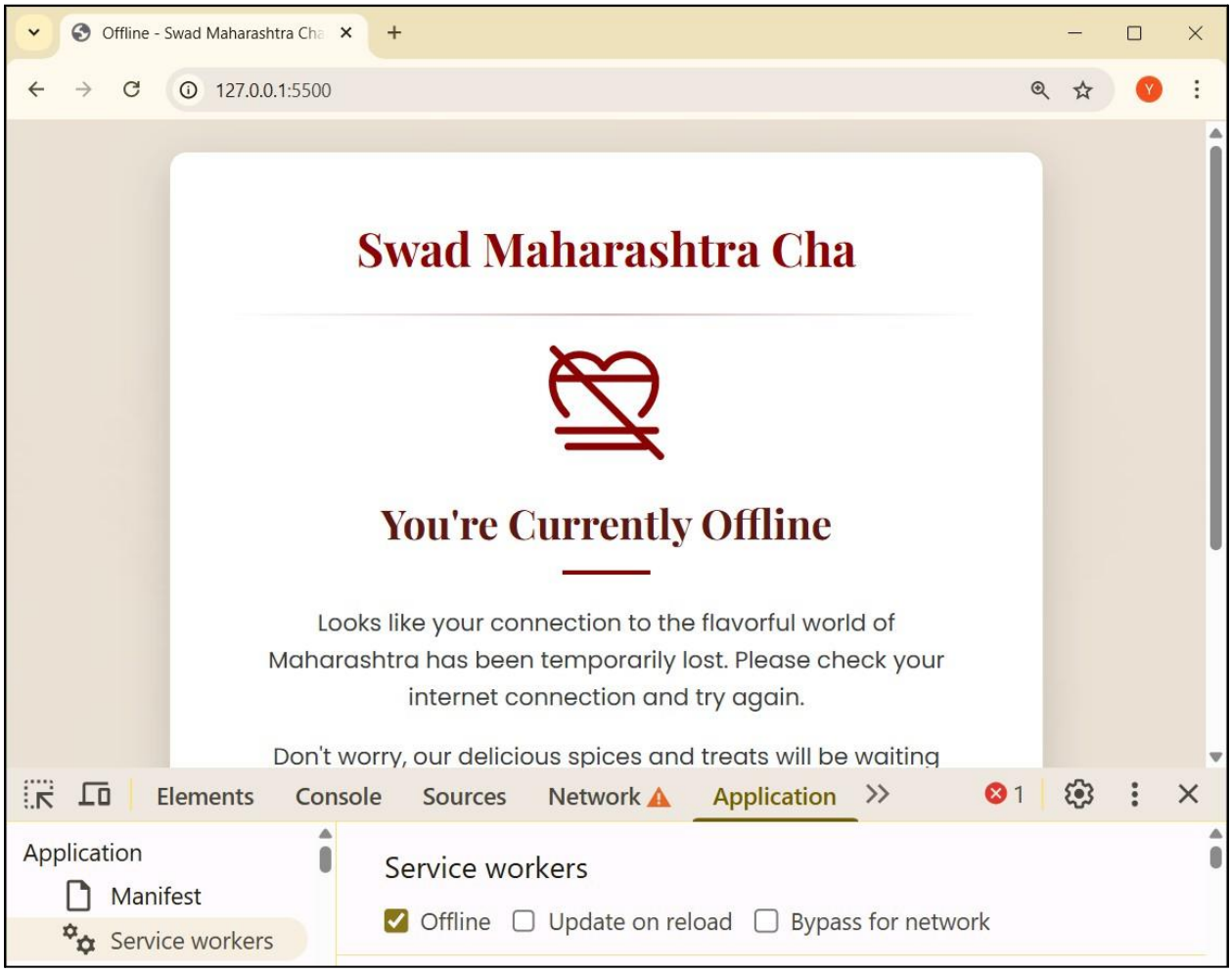
```
    caches.keys().then((keyList) =>
      Promise.all(
      keyList.map((key) => {
          if (key !== CACHE_NAME) {
            console.log("[ServiceWorker] Removing old cache", key);
            return caches.delete(key);
          }
        })
      )
    )
  );
  return self.clients.claim();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

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

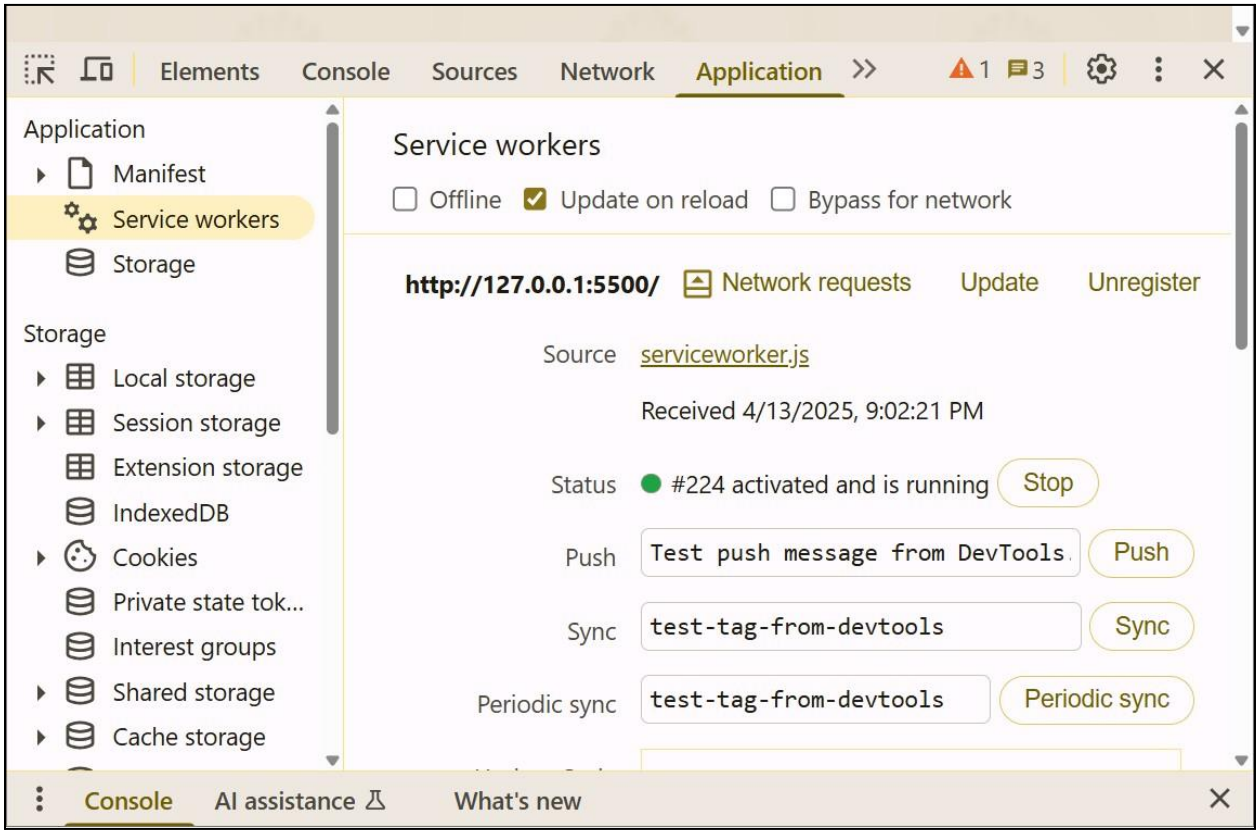
Screenshot:

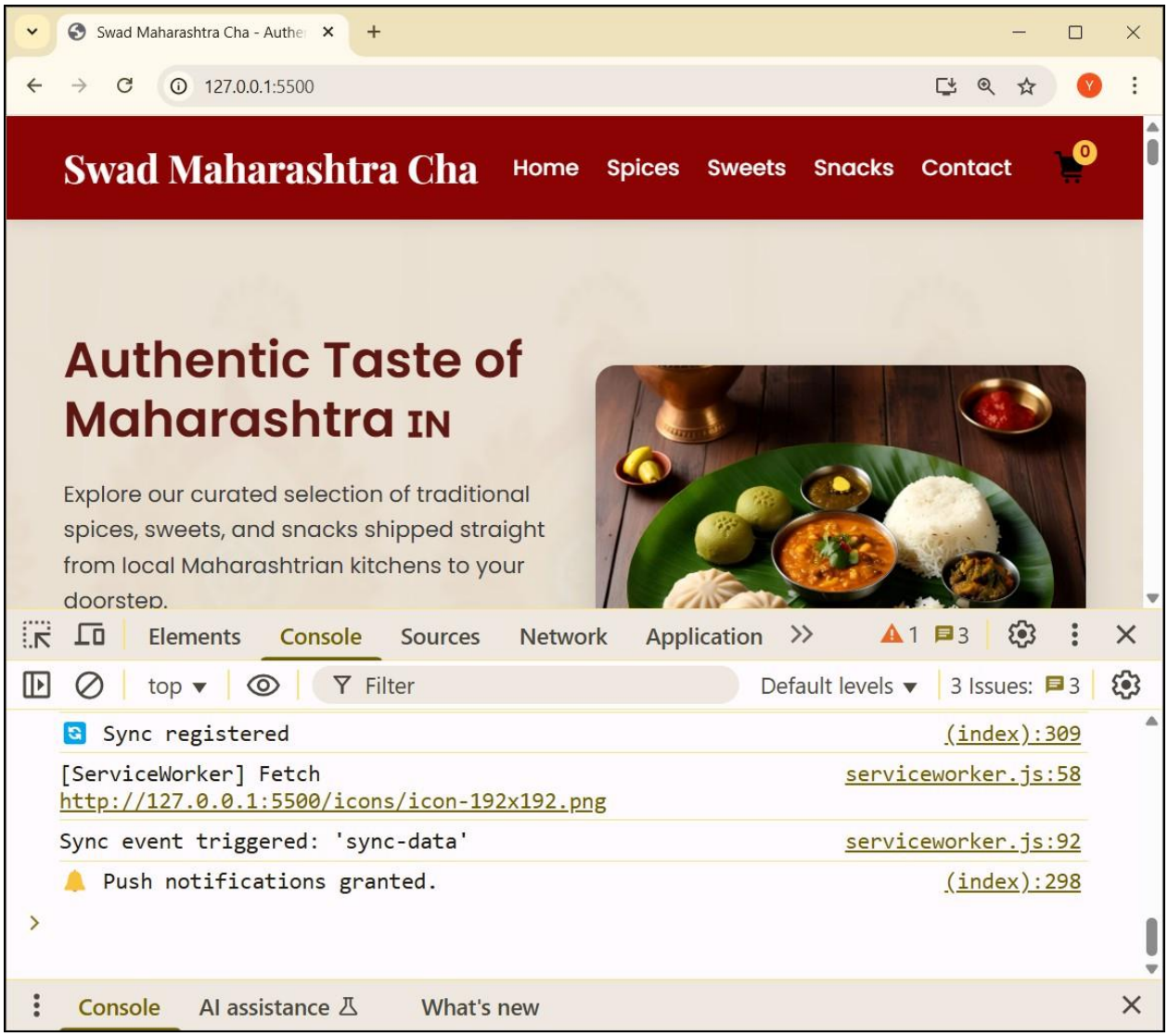


1. Test: Fetch Event (Offline Support)

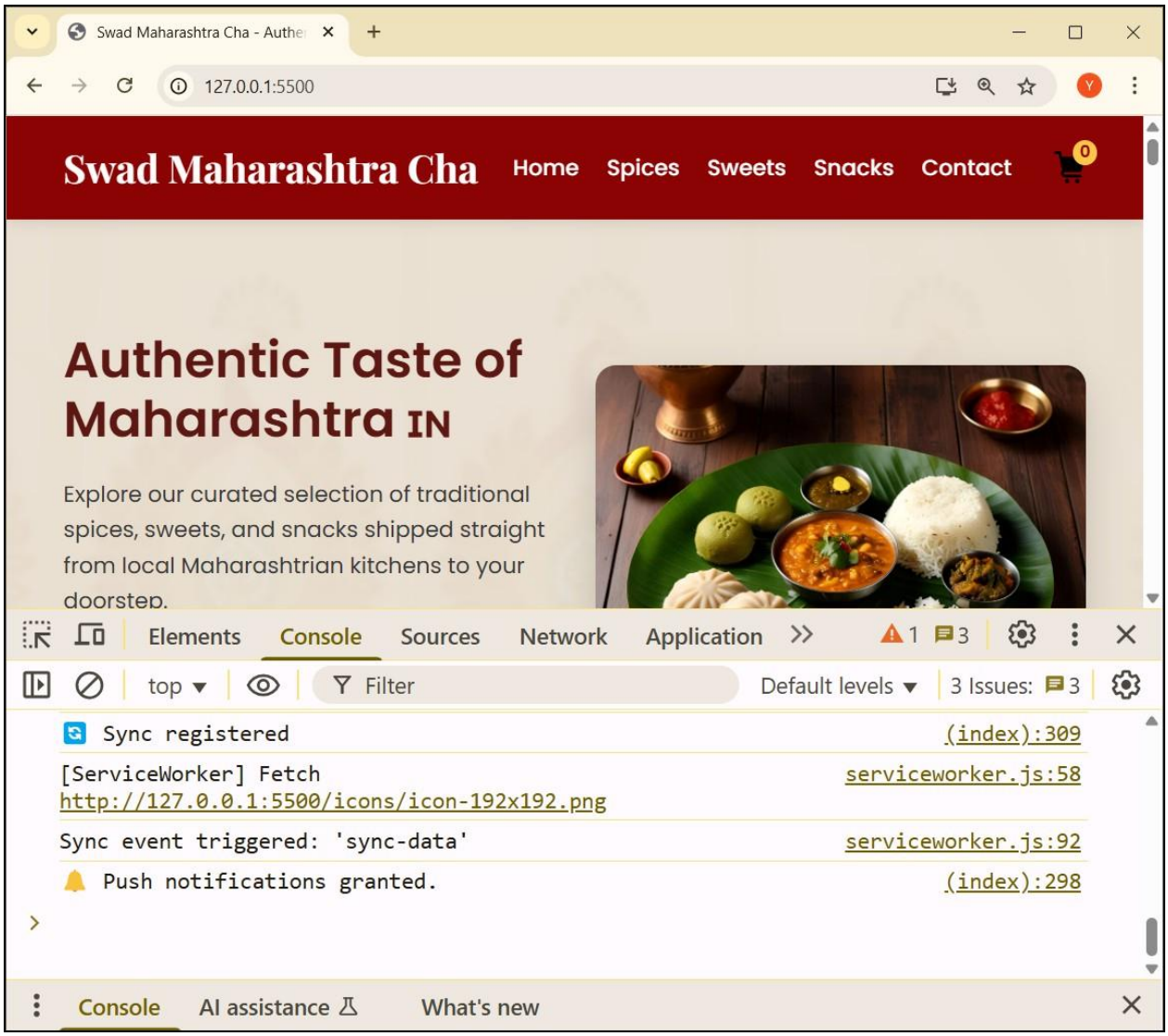


2. Test: Background Sync Event





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 :Abhay Gupta

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:

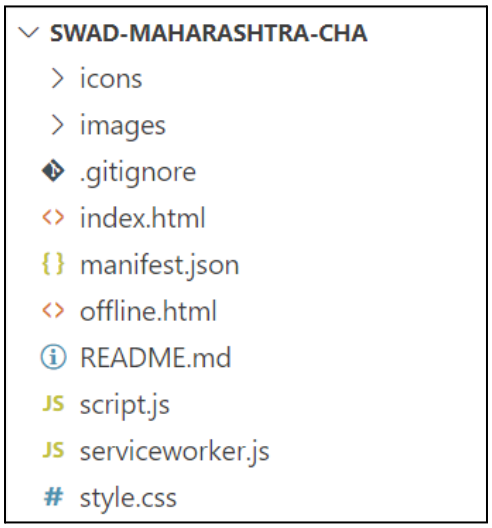
- All file paths are correct (especially for service workers and cached files).
- The service worker properly uses the subfolder paths (since GitHub Pages hosts sites from /username/repo-name/).
- Our project is pushed to the GitHub repository and GitHub Pages is enabled under the repository settings.

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 /Technix-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:

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

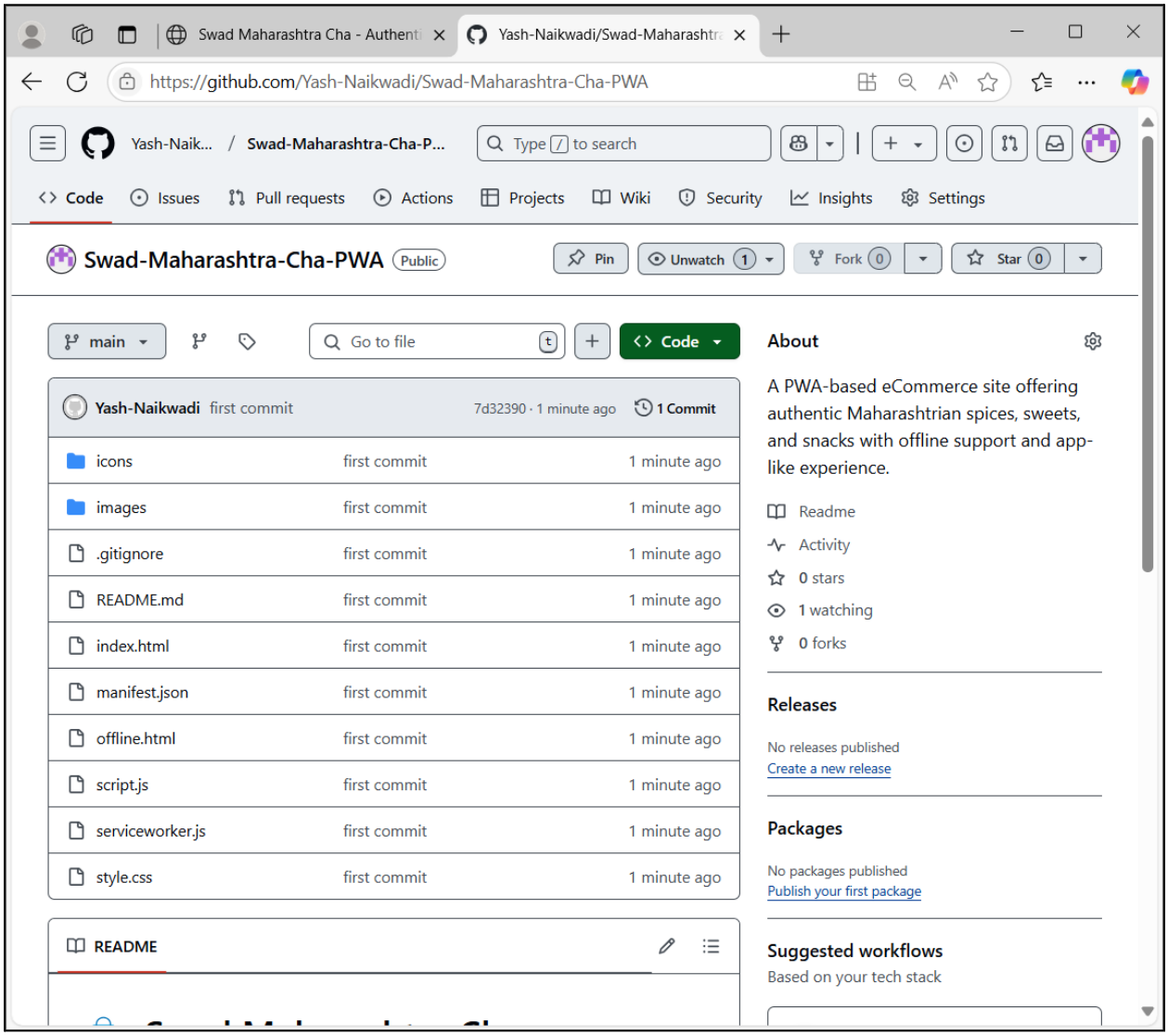
```
const CACHE_NAME = "swad-maha-cache-v1";
const FILES_TO_CACHE = [
  "/Swad-Maharashtra-Cha-PWA/index.html",
  "/Swad-Maharashtra-Cha-PWA/",
  "/Swad-Maharashtra-Cha-PWA/style.css",
  "/Swad-Maharashtra-Cha-PWA/script.js",
  "/Swad-Maharashtra-Cha-PWA/serviceworker.js",
  "/Swad-Maharashtra-Cha-PWA/manifest.json",
  "/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>
```



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 : Abhay Gupta

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:

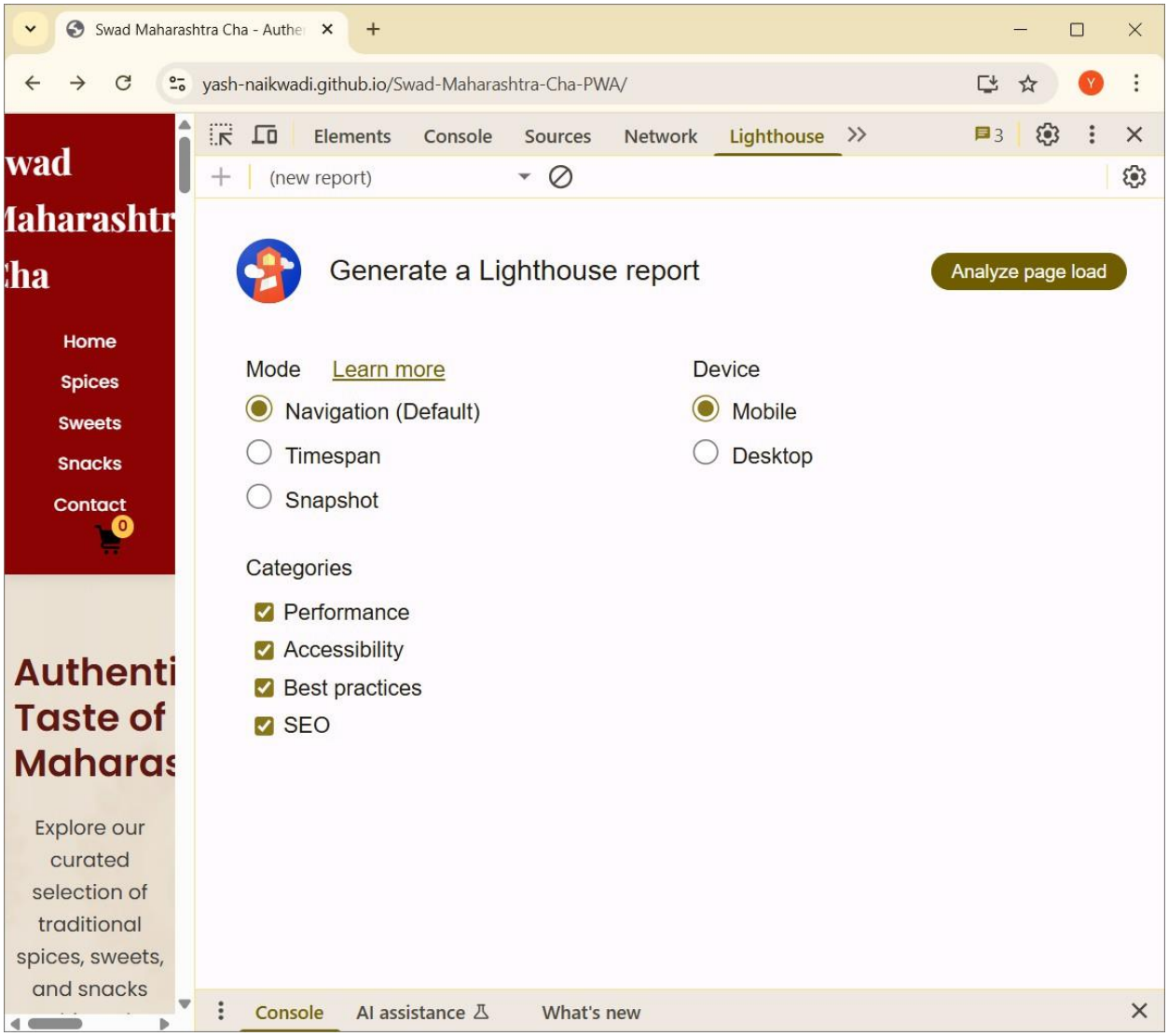
- Fast and responsive loading (Performance)
- Accessible design for all users (Accessibility)
- Secure and modern coding practices (Best Practices)
- Search engine optimization readiness (SEO)
- Proper PWA features like manifest, service worker, and offline capabilities

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 - Authe
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.

Abhay Gupta
D15B
16

DATE 04

MAD Assignment - 1

a) Explain features, advantages of using flutter for mobile app development.

1. Single for multiple platforms: write one codebase for both android and IOS, reducing development effort and maintenance

2. Hot Reload Instantly see changes in the app without restraining making development faster and more interactive

3. Fast performance: Uses Dart language and a compiled approach smooth and high performance apps.

4. Open source & sharing community support, Enforced by google and large developer community. ensuring continuous improvement and resources.

Advantages:-

1. Faster Development Time: Hot reload and single codebase reduce development time significantly

2. Cost Effective: The code runs both android and IOS, business save on development and maintain and cost

b). Discuss how flutter framework work diff from the

traditional approach and why gained popularity in development community.

1. Diff from traditional approaches:

Rendering Engine: Traditional framework rely on pla
specific UI

Declarative UI: UI Updates are defined as part
of state change rather than imperative

Performance: Flutter apps have better performance
than react Native as avoid JS bridge

2. why flutter popular in Dev Community:

Rapid development with hot Reload
Strong support from google and the community
Growing number of plugins and Libraries
Flexibility in UI customization
Support for web mobile and desktop

Q2) a) widget tree and widget Composition in flutter

ans In flutter, the widget tree is fundamental structure
that represent the UI of an application. It's a hierar
arrangements of widget where each widget defines a
part of user Interface

widget Composition in flutter refers to building

Complex UI by combining smaller, reusable widgets. Instead of creating large, monolithic components flutter encourages breaking the UI into smaller, manageable widgets that can be composed together. This approach promotes code reusability, maintainability and performance. Each widget is responsible for rendering a specific part of the UI and can be easily reused in different parts of the app. Flutter efficiently rebuilds and updates only the necessary widgets when changes occur.

Example:

```
class ProfileCard extends StatelessWidget {
  final String name;
  final String imageUrl;

  ProfileCard(required this.name, required this.imageUrl);

  @override
  Widget build(BuildContext context) {
    return Card(
      child: Column(
        children: [
          Image.network(
            imageUrl)
        ]
      )
    );
  }
}
```

Benefits of widgets composition:-

Reusability small widgets can be reused in different parts of the app

Maintainability Breaking UI into smaller widgets makes it easier to debug and update

Performance: Flutter efficiently rebuild only necessary parts

b) Provide examples of commonly used widgets and roles in creating widget tree

1. Structural widgets
These widgets acts as the foundation for buil the UI

- Scaffold Provides basic layout structure, includi or app bar, body floating action button etc

Eg:- MaterialApp(
home: Scaffold(
appBar: AppBar(title:
('Flutter Widget Tree')),
body: Container(
padding: EdgeInsets
));

2. Input & Interaction widgets

ElevationButton: A button with elevation gestures like taps, swipes and long presses

Eg:- Column(
children[
TextField(decoration: InputD
(labelText: "Ente
ElevatedButton(
onPressed: (){
Print("Button Pr
});

3. Display & Styling Widgets

- Text - displays text on the screen
- Image - shows images from assets network or memory
- Icon - Displays Icons
- Card - A material design card with rounded corners and elevation

Eg:- Column(
children[
Text("welcome to flutter", style.
TextStyle(Fontsize: 24,
FontWeight.bold)),
Image.network("https://flutter.dev
images/flutter-logo-sharing")),
];
);

a) In flutter, state refers to data that can change during the lifetime of an application
This includes:
- User input
- UI changes
- Network changes
- Animation states

There are two types of states
1. Ephemral state: small UI, specific state that doesn't affect the whole app
2. App wide state: Data shared across multiple

widgets Importance of state management ensures that only necessary widgets are updated improving perfor

Data Consistency and Synchronization
proper state management ensures that data remains consistent across different screens and widgets

b) set state - Local state
Pros - simple built in easier to use
Cons - Not scalable causes unnecessary re-rend

Best use Cases - small UI updates (eg. toggle s counter)

Riverpod App - wide state CMore scalable than provid
pros - diminates Providers limitation, improves performances
Cons - Requires learning new concepts
Best use case - large apps needing global state

Scenarios for each approach

→ Use set state when managing simple UI Elements with a single widget like toggling dark mode in a setting screen

→ Use provider when sharing state across multiple widgets such as managing user authentication or theme changes, like an ecommerce app with cart management

SDS | Page No.
Date / /

a) Explain process of integrating firebase with a flutter application

Ans Firebase provides a powerful backend solution for flutter application offering services like authentication, real time database, cloud functions, storage and more.

→ Steps to integrate firebase with flutter

Step 1: Create a firebase project
- go to firebase console
- click on "Add project" and enter a
- project name

Step 2: Register the flutter app with firebase
- In the firebase project dashboard click "Add App" and select android or IOS
- For Android: Enter the android package name and download the google service json file and place it in android app

Step 3: Install firebase dependencies
Add firebase dependencies in pubpes.yaml
firebase core
firebase with cloud firestore
Run flutter project

Step 5: Initializing firebase in flutter

SDS | Page No. | Date

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

→ Benefits of using firebase:

1. Fast to setup & scale
2. Authentication
3. Cloud Storage
4. Push notification

b) Firebase provides a suite of backend services th simplify flutter app development

1. Firebase authentication:
   Enable secure authentication using E-mail / password, phone no. and third party providers like google, facebook and google and apple
2. Cloud firebase
   Stores and syncs data in real time ac devices support structured data, queries and offline access
3. Realtime Database:
   A real-time, JSON-based database that auto updates data across databases.

Abhay Gupta
D15B
18

MDL Assignment-2

03/05

1) A = PWA is a web applications that combines both web & mobile apps to deliver a seamless. PWA work offline, load quickly & provide app like experience

→ Platform independence
→ Improved performance
→ Offline functionality
→ No app store dependencies

⇒ Key characteristics :-

(i) Installation : Installed from browser, traditional mobile apps are downloaded from mobile app

(ii) Platform dependencies

(iii) offline support

(iv) Updates

(v) Performance

2) It's an approach that ensures web pages adapt to different screen sizes & orientations using flexible grids

→ Eliminates need for different devices and enhance usability

Feature	Responsive	Fluid	Adapti
Definition	uses CSS media queries to adjust	uses % for elements	uses pr defined layou
Flexibility	Highly flexible	completely flexible	Fixed, specif
Performance	efficient but may need adjustment	smooth	Break pt cause shifts
Best use case	website & PWAs for all screens	App requires scaling	pre- def Lay

(Q3) (i) Lifecycle Phases:

(ii) Registeration:

→ if ('serviceWorker' in navigator) {
navigator.serviceWorker.('/sw.js')
then(() ⇒ console.log('service registered
})

(iii) Installation:

→ Occurs when the service worker is first downloaded

Eg: self.addEventListener('install', event => {
event.waitUntil(
return cache.addAll(
});

(ii) Activation

→ Runs after installation & ensures old cache cleared if necessary

Eg: self.addEventListener('activate', event => {
event.waitUntil(
caches.keys(), then(keys => {
return keys.filter(k => k !== 'v1')
});
});

(iv) Fetching & Updates

→ The service worker intercepts network requests

Indexed DB is a Low-level NoSql database in the browser that allows web pages to store & retrieve

Service workers are JavaScript that run in background separate from main browser thread, enabling features like:

(i) Background Synchronization

(ii) Push Notifications.

(iii) offline functionality

(iv) Caching and data management

Advantages of using Indexed DB with service worker

1. offline - First functionality : Store and serve data from Indexed DB when offline

2. Large storage capacity : Store large number of structured data efficiently

3 Asynchronous API : Improves performance and av blocking the main thread

4. Persistent storage : Unlike Local storage, Inde DB supports complex data types and is non-vol