

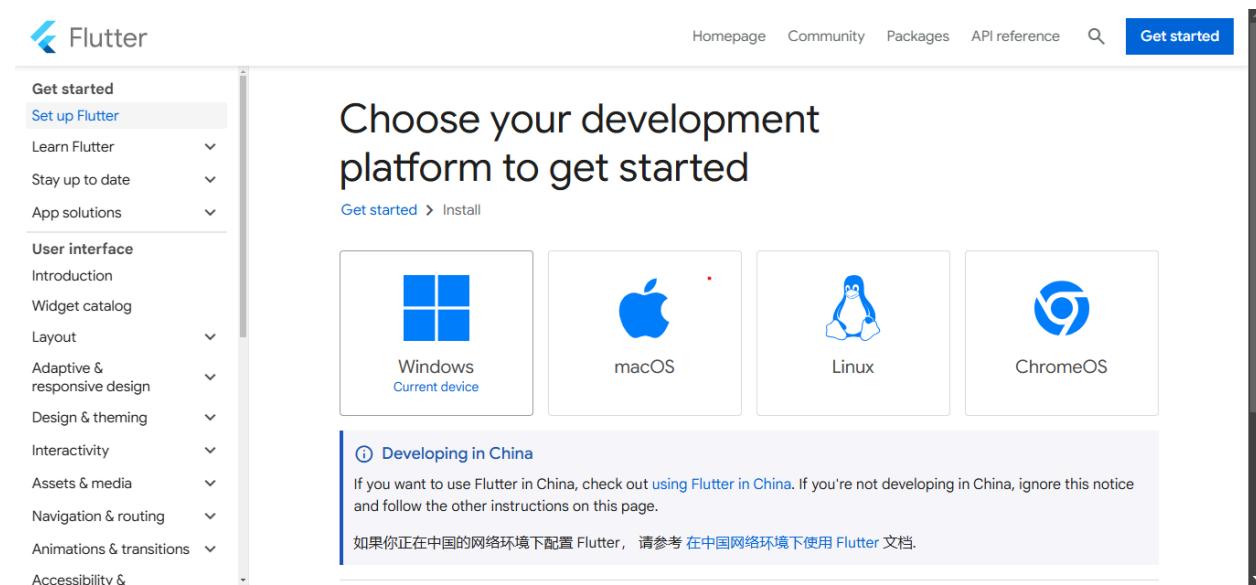
Name:Aarya Gandhi
Class:D15B
Roll no. 14

Aim:Installation and Configuration of Flutter Environment.

Theory:

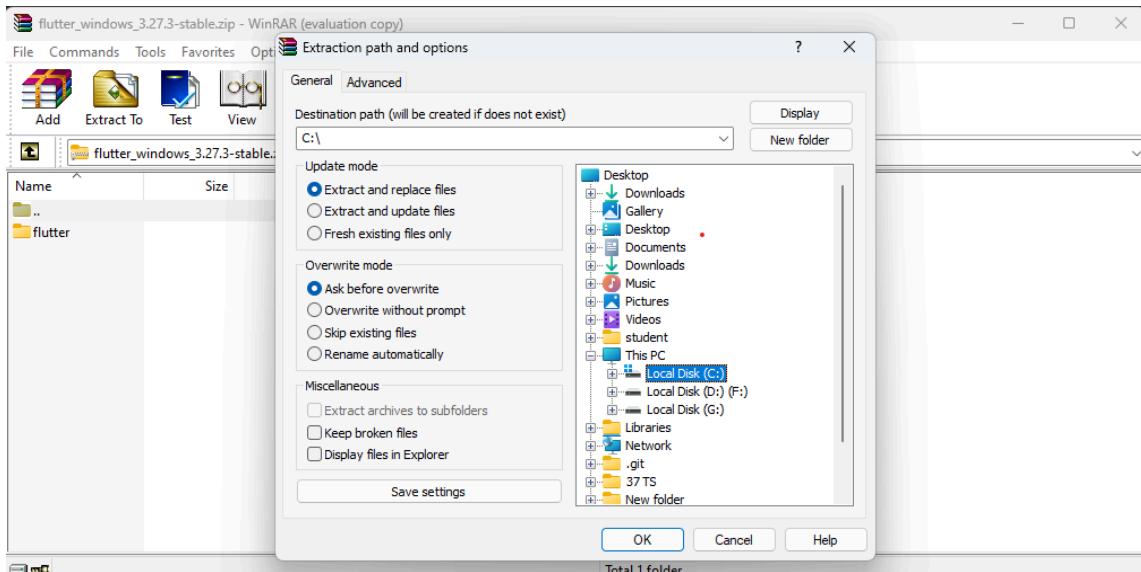
Install the Flutter SDK

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.



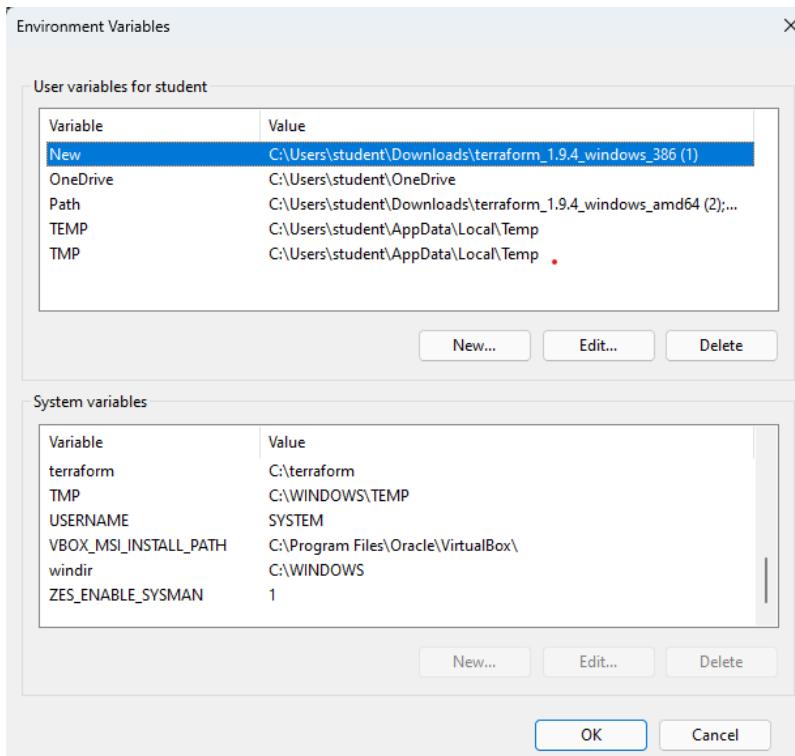
Step 2: Next, to download the latest Flutter SDK, click on the Windows icon. Here, you will find the download link for SDK.

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.

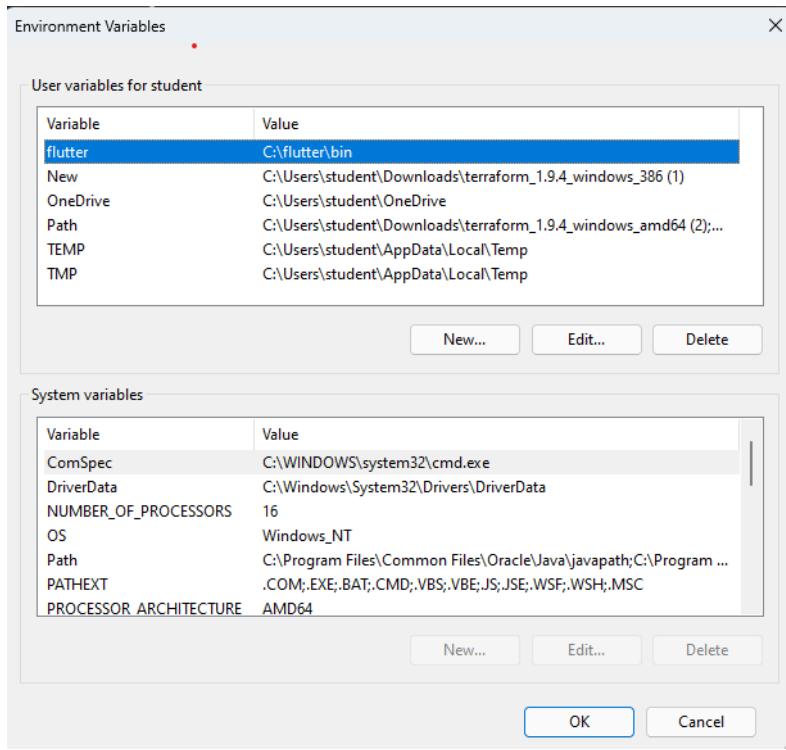


Step 4: To run the Flutter command in regular windows console, you need to update the system path to include the flutter bin directory. The following steps are required to do this:

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 5: Now, run the \$ flutter command in 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.

The screenshot shows a Command Prompt window titled 'Administrator: Command Pro'. The window displays the output of the 'flutter doctor' command. It starts with system information, followed by common commands, global options, and available commands. The 'Global options' section includes detailed descriptions for each option.

```
Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

C:\Users\INFT505-12>flutter
Manage your Flutter app development.

Common commands:

  flutter create <output directory>
    Create a new Flutter project in the specified directory.

  flutter run [options]
    Run your Flutter application on an attached device or in an emulator.

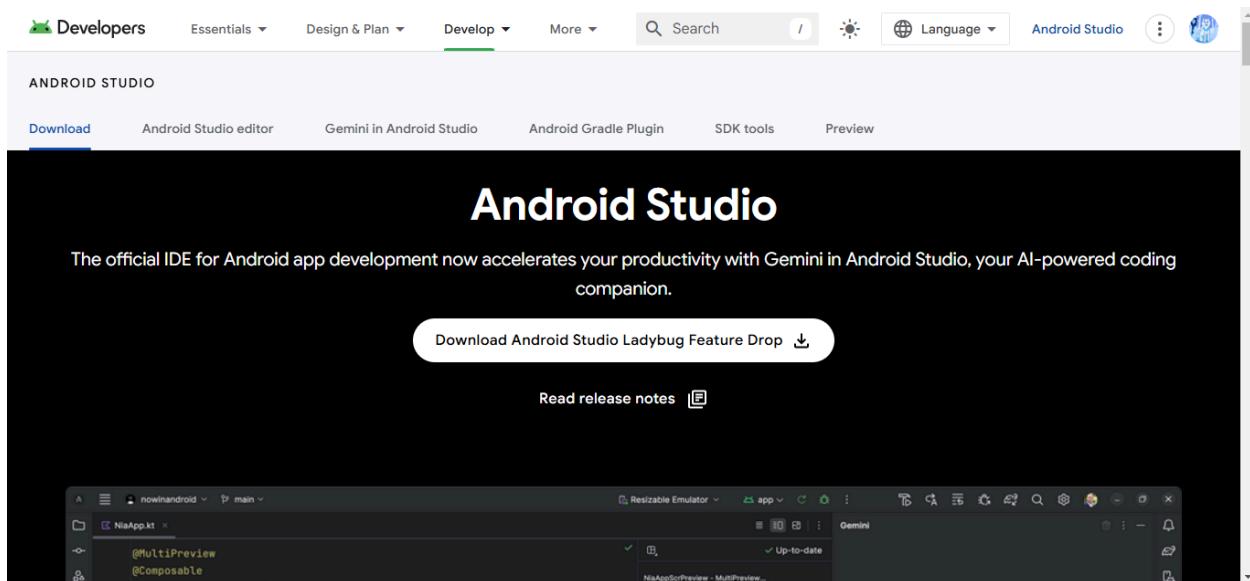
Usage: flutter <command> [arguments]

Global options:
  -h, --help          Print this usage information.
  -v, --verbose       Noisy logging, including all shell commands executed.
                      If used with "--help", shows hidden options. If used with "flutter doctor", shows additional diagnostic information. (Use "-vv" to force verbose logging in those cases.)
  -d, --device-id    Target device id or name (prefixes allowed).
  --version          Reports the version of this tool.
  --enable-analytics Enable telemetry reporting each time a flutter or dart command runs.
  --disable-analytics Disable telemetry reporting each time a flutter or dart command runs, until it is re-enabled.
  --suppress-analytics Suppress analytics reporting for the current CLI invocation.

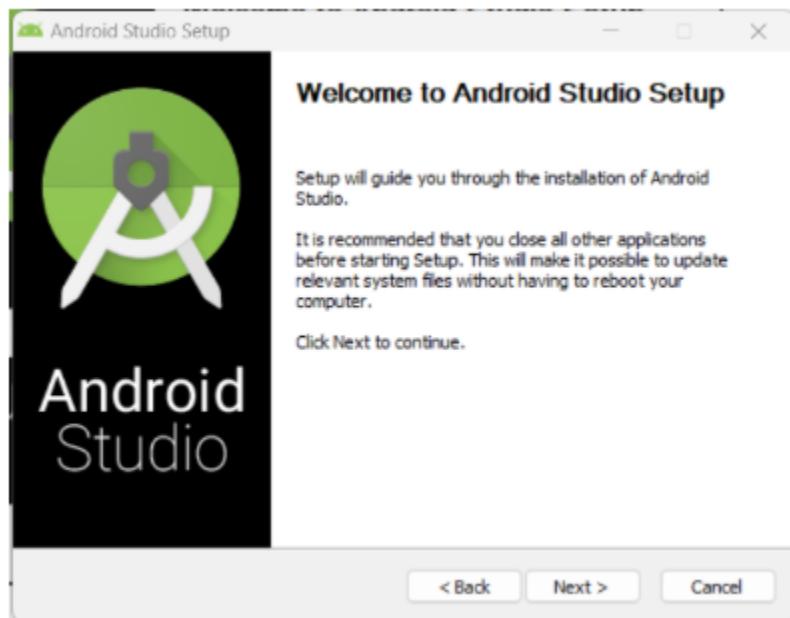
Available commands:
```

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.

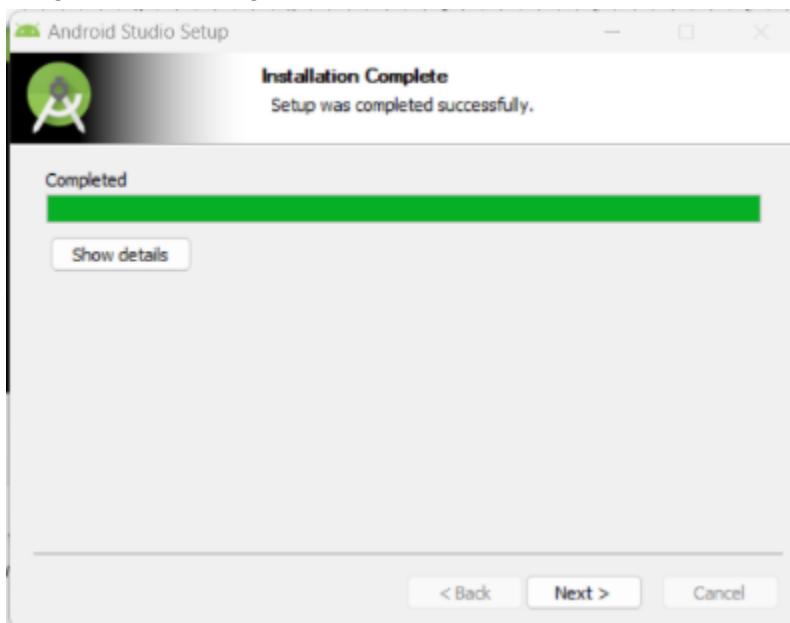
Step 7.1: Download the latest Android Studio executable or zip file from the official site.



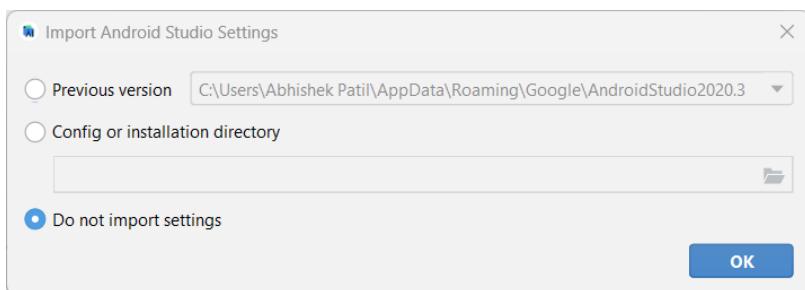
Step 7.2: When the download is complete, open the .exe file and run it. You will get the following dialog box.



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.

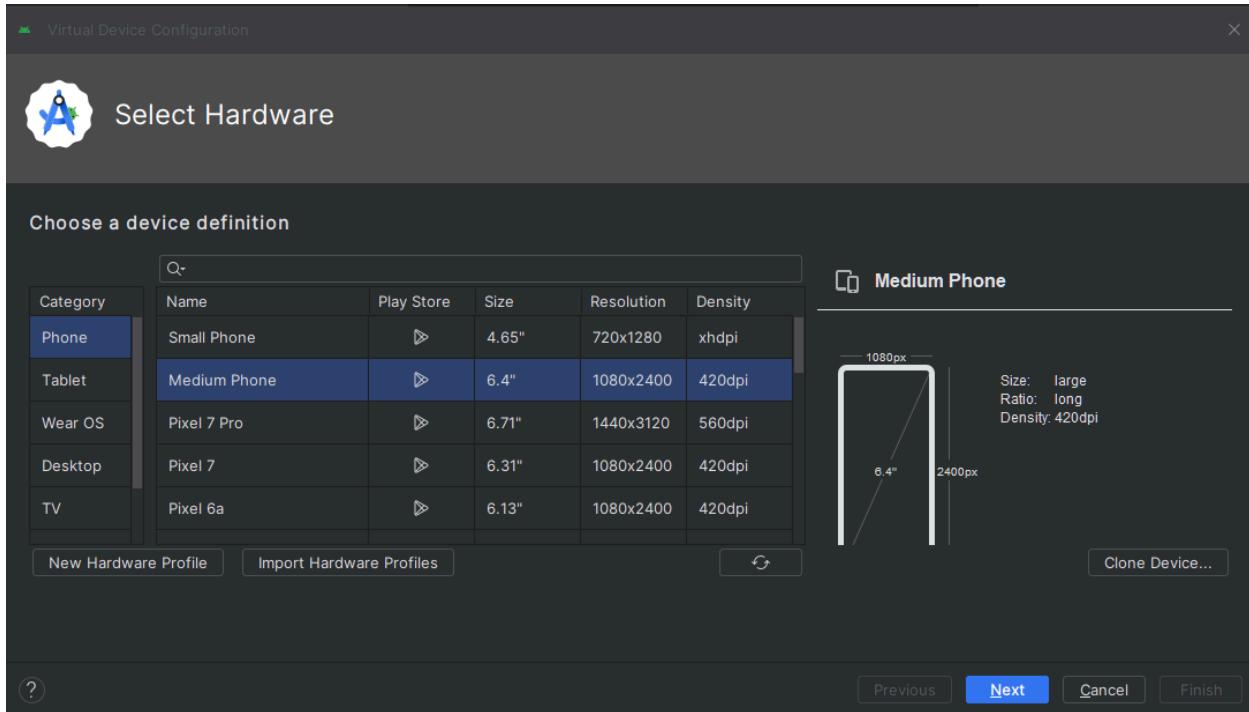


Step 7.5: run the \$ flutter doctor command and Run flutter doctor --android-licenses command.

```
C:\Users\INFT505-12> flutter doctor --android-licenses
Warning: Errors during XML parse:
Warning: Additionally, the fallback loader failed to parse the XML.ry...
Warning: Errors during XML parse:
Warning: Additionally, the fallback loader failed to parse the XML.ry...
[=====] 100% Computing updates...
All SDK package licenses accepted.
```

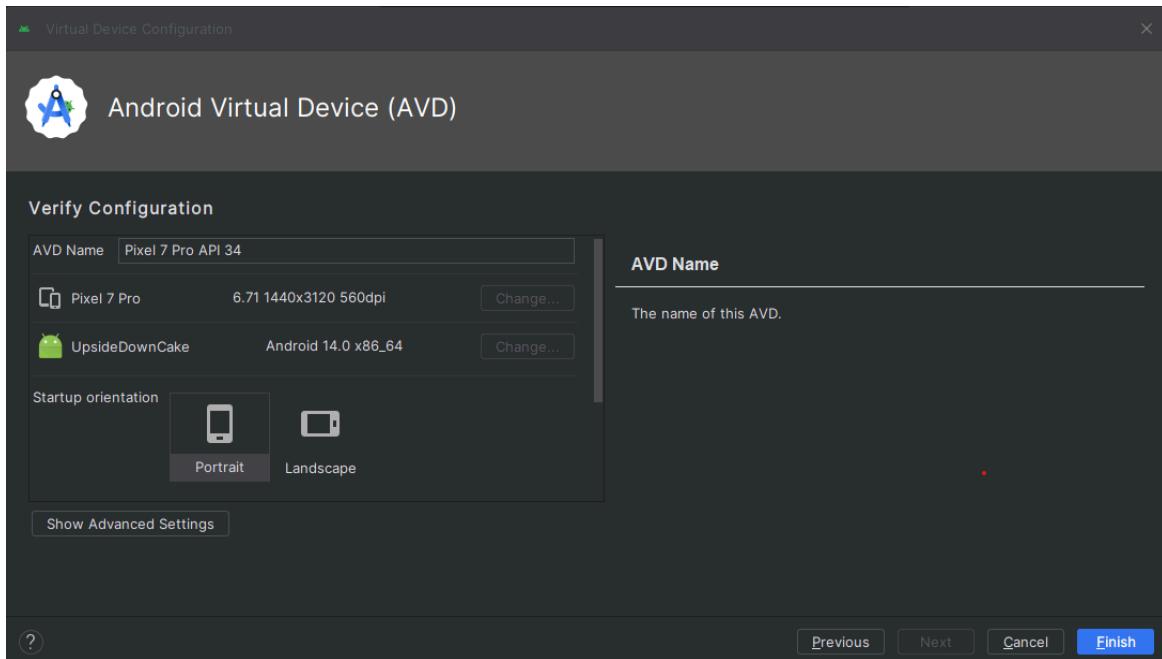
Step 8: Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application.

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.

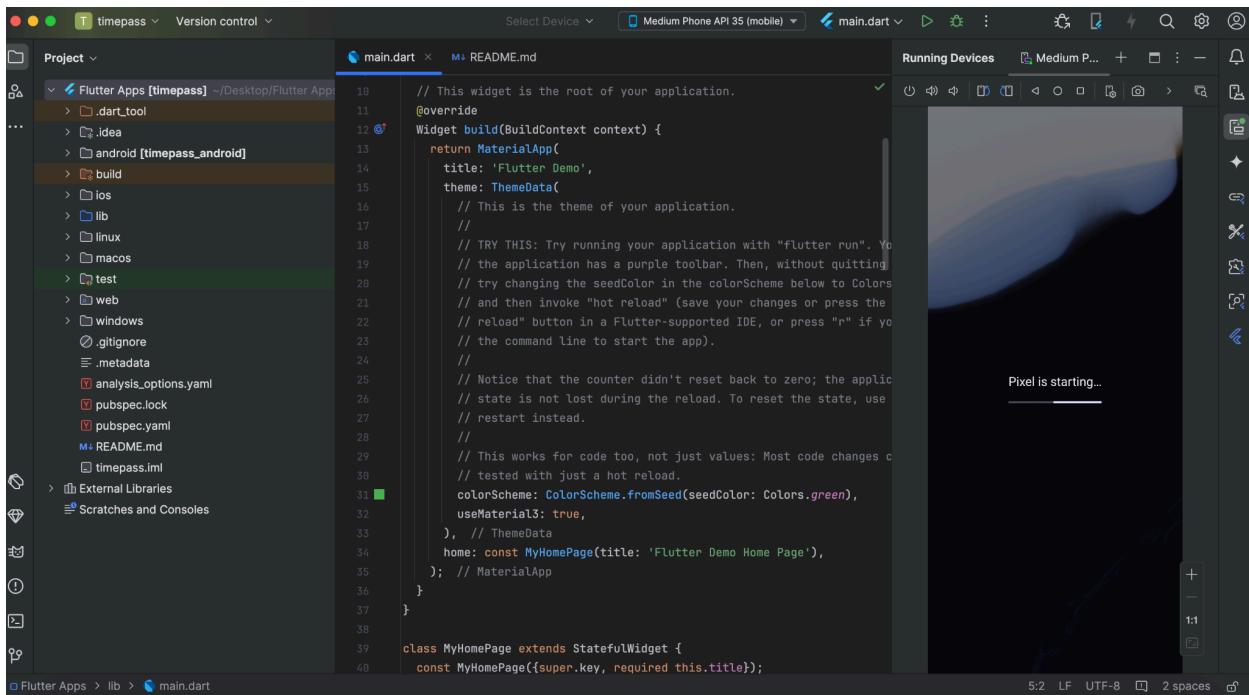


Step 8.3: Select the system image for the latest Android version and click on Next.

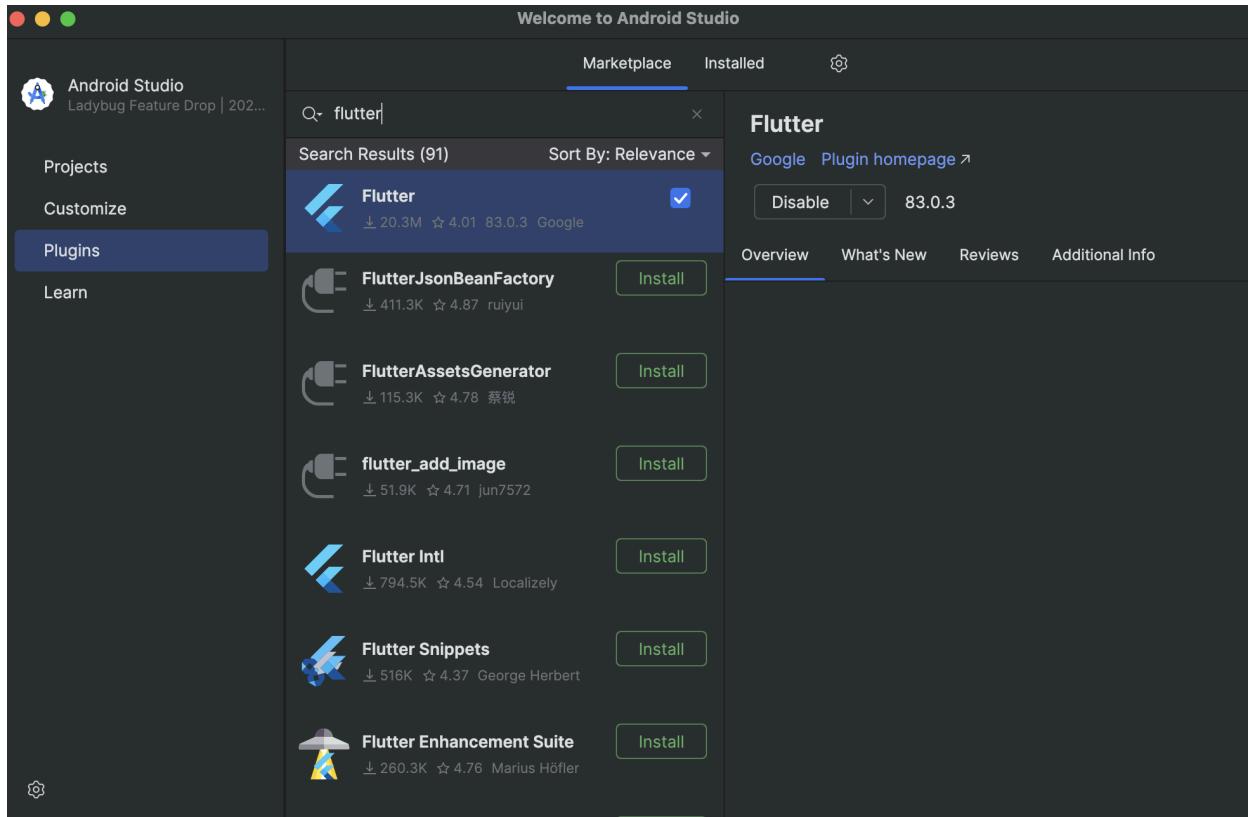
Step 8.4: Now, verify the all AVD configuration. If it is correct, click on Finish. The following screen appears.



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 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.

Conclusion: In conclusion, setting up the Flutter environment ensures a smooth development process by installing the SDK, configuring dependencies, and verifying the setup with flutter doctor.

Once complete, developers can efficiently build and test cross-platform applications.

Name:Aarya Gandhi
Roll No. 14
Class: D15B

MPL Experiment 02

Aim:To design Flutter UI by including common widgets.

Theory:

Flutter UI Design Using Common Widgets

Flutter provides a comprehensive set of widgets to create visually appealing and interactive user interfaces. These widgets are categorized into **stateless** and **stateful** widgets.

- **Stateless Widgets** remain constant once built and do not update dynamically.
- **Stateful Widgets** can change based on user interaction or internal state changes.

Common Flutter Widgets

1. Basic Structural Widgets

These widgets provide the foundational layout structure of the app.

- **Scaffold** – Serves as the main layout structure with built-in support for **AppBar**, **Body**, and **FloatingActionButton**.
- **AppBar** – A top navigation bar that contains a title and optional action buttons.
- **Container** – A box-like widget used for styling, including background color, padding, and margins.
- **Column & Row** – Used to arrange child widgets in a vertical or horizontal direction, respectively.

2. User Input Widgets

These widgets allow users to provide input and interact with the application.

- **TextField** – Accepts user input in the form of text.
- **DropdownButton** – Displays a dropdown menu to select from multiple options.
- **Checkbox & Switch** – Used for enabling/disabling settings or selecting multiple options.

3. Display Widgets

These widgets are used to display text, images, and content.

- **Text** – Displays static or dynamic text content.
- **Image** – Loads images from assets, the network, or memory.
- **Card** – A material design component used to display content in an organized manner.

4. Interactive Widgets

These widgets handle user interactions like button clicks and gestures.

- **ElevatedButton** – A raised button used to perform actions on tap.
- **IconButton** – A button with an icon instead of text.
- **GestureDetector** – Detects gestures such as taps, swipes, and long presses.

5. Lists & Scrolling Widgets

These widgets help manage scrollable content.

- **ListView** – Displays a scrollable list of items.
- **GridView** – Creates a grid layout, ideal for galleries and product listings.
- **SingleChildScrollView** – Allows scrolling for a single child widget when content overflows.

6. Navigation Widgets

These widgets enable seamless navigation between different screens in the app.

- **Navigator** – Manages screen transitions using push and pop methods.
- **BottomNavigationBar** – A bottom menu that allows switching between different sections of the app.

Implementation:

Login Page:

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

class SignupPage extends StatefulWidget {
  @override
  _SignupPageState createState() => _SignupPageState();
}

class _SignupPageState extends State<SignupPage> {
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();
  final FirebaseAuth _auth = FirebaseAuth.instance;

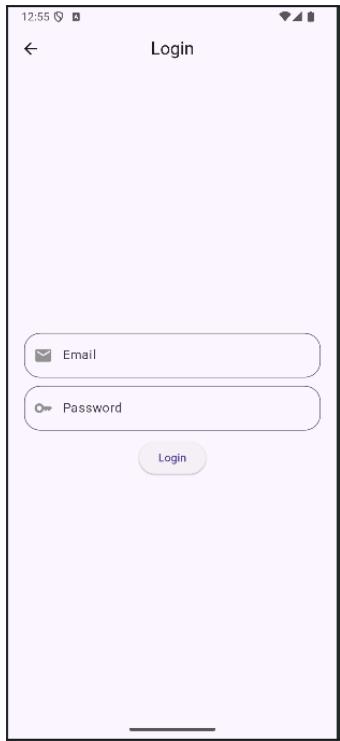
  void signUp() async {
    try {
      await _auth.createUserWithEmailAndPassword(
        email: emailController.text,
        password: passwordController.text,
      );
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("Signup Successful!")),
      );
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("Error: ${e.toString()}")),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(" Sign Up Page",
        style: TextStyle(
          fontWeight: FontWeight.bold
```

```
 ),),
centerTitle: true,
),
body: Center(
child: Padding(
padding: EdgeInsets.all(16),
child: Column(
mainAxisSize: MainAxisSize.min,
children: [
TextField(
controller: emailController,
decoration: InputDecoration(
prefixIcon: Icon(Icons.person,color: Colors.grey,),,
labelText: "Enter Your Username",
border: OutlineInputBorder(
borderRadius: BorderRadius.circular(20),
),
),
),
),
SizedBox(height: 16),
TextField(
controller: emailController,
decoration: InputDecoration(
prefixIcon: Icon(Icons.mail,color: Colors.grey,),,
labelText: "Enter Your Email",
border: OutlineInputBorder(
borderRadius: BorderRadius.circular(20),
),
),
),
),
SizedBox(height: 16),
TextField(
controller: passwordController,
obscureText: true,
decoration: InputDecoration(
prefixIcon: Icon(Icons.key,color: Colors.grey,),,
labelText: "Enter Your Password",
border: OutlineInputBorder(

```

```
        borderRadius: BorderRadius.circular(20),
    ),
),
),
),
SizedBox(height: 16),
ElevatedButton(
    onPressed: signUp,
    child: Text("Sign Up"),
),
],
),
),
),
),
);
}
}
```



Signup Page:

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

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

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

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

class _LoginPageState extends State<LoginPage> {
    final FirebaseAuth _auth = FirebaseAuth.instance;
    final TextEditingController emailController = TextEditingController();
    final TextEditingController passwordController = TextEditingController();
    String errorMessage = "";

    Future<void> loginUser() async {
        try {
            await _auth.signInWithEmailAndPassword(
                email: emailController.text.trim(),
                password: passwordController.text.trim(),
            );
            // Navigate to home page after successful login
            Navigator.pushReplacement(
                context,
                MaterialPageRoute(builder: (context) => HomePage()),
            );
        } catch (error) {

```

```
        setState(() {
            errorMessage = error.toString();
        });
    }
}

@Override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: Text("Login"),
            centerTitle: true,
        ),
        body: Padding(
            padding: EdgeInsets.all(20.0),
            child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                    TextField(
                        controller: emailController,
                        decoration: InputDecoration(
                            prefixIcon: Icon(Icons.mail,color: Colors.grey,),
                            labelText: "Email",
                            border: OutlineInputBorder(
                                borderRadius: BorderRadius.circular(20)
                            ),
                        ),
                    ),
                    SizedBox(height: 10),
                    TextField(
                        controller: passwordController,
                        obscureText: true,
                        decoration: InputDecoration(
                            labelText: "Password",
                            border: OutlineInputBorder(
                                borderRadius: BorderRadius.circular(20)
                            ),
                            prefixIcon: Icon(Icons.key,color: Colors.grey,)
                        ),
                    ),
                ],
            ),
        ),
    );
}
```

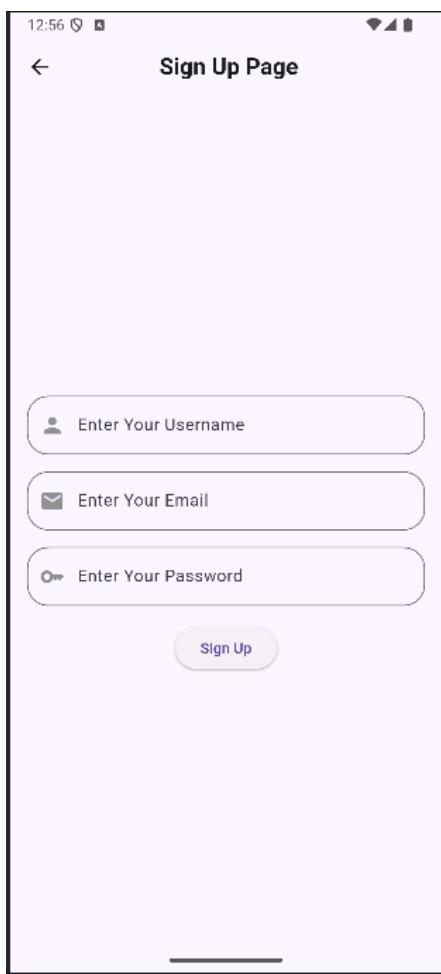
```
        ),
        SizedBox(height: 10),
        ElevatedButton(
            onPressed: loginUser,
            child: Text("Login"),
        ),
        if (errorMessage.isNotEmpty)
            Text(
                errorMessage,
                style: TextStyle(color: Colors.red),
            ),
        ],
    ),
),
),
);
);
}
}

class HomePage extends StatelessWidget {
final FirebaseAuth _auth = FirebaseAuth.instance;

@Override
Widget build(BuildContext context) {
    User? user = _auth.currentUser;

    return Scaffold(
        appBar: AppBar(
            title: Text("Home"),
            actions: [
                IconButton(
                    icon: Icon(Icons.logout),
                    onPressed: () async {
                        await _auth.signOut();
                        Navigator.pushReplacement(
                            context,
                            MaterialPageRoute(builder: (context) => LoginPage()),
                        );
                    },
                ),
            ],
        ),
    );
}
```

```
),
body: Center(
    child: Text(
        "Welcome, ${user?.email ?? 'User'}",
        style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
    ),
),
);
},
);
}
}
```



Conclusion

This app demonstrates Firebase authentication integration in Flutter with a well-structured UI. Using Flutter's Material Design widgets ensures a **responsive, smooth, and visually appealing** user experience.

MPL experiment 3 SS

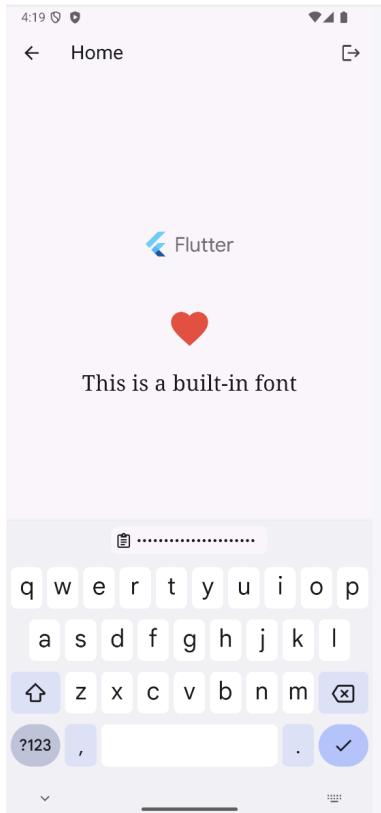
```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'login_page.dart';

class HomePage extends StatelessWidget {
  final FirebaseAuth _auth = FirebaseAuth.instance;

  @override
  Widget build(BuildContext context) {
    User? user = _auth.currentUser;

    return Scaffold(
      appBar: AppBar(
        title: Text("Home"),
        actions: [
          IconButton(
            icon: Icon(Icons.logout),
            onPressed: () async {
              await _auth.signOut();
              Navigator.pushReplacement(
                context,
                MaterialPageRoute(builder: (context) => LoginPage()),
              );
            },
          ),
        ],
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            // Displaying an image from assets
            Image.network(
              'https://upload.wikimedia.org/wikipedia/commons/1/17/Google-flutter-logo.png',
              width: 100,
              height: 100,
            ),
            SizedBox(height: 20),
            // Displaying an icon from material icons
            Icon(Icons.favorite, color: Colors.red, size: 50),
            SizedBox(height: 20),
            // Displaying custom font text
            Text(
              'This is a built-in font',
              style: TextStyle(fontFamily: 'Georgia', fontSize: 24),
            )
          ],
        ),
      ),
    );
  }
}
```

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



MPL Experiment 4

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

class LoginPage extends StatefulWidget {
  const LoginPage({super.key});

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

class _LoginPageState extends State<LoginPage> {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();
  bool isLoading = false;

  Future<void> _login() async {
    final email = emailController.text.trim();
    final password = passwordController.text.trim();

    if (email.isEmpty || password.isEmpty) {
      _showSnackbar('Please enter both email and password');
      return;
    }

    setState(() => isLoading = true);

    try {
      await _auth.signInWithEmailAndPassword(email: email, password: password);
      _showSnackbar('Login Successful');

      // Navigate to HomePage after successful login
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => const HomePage()),
      );
    } on FirebaseAuthException catch (e) {
      _handleFirebaseError(e);
    } finally {
      setState(() => isLoading = false);
    }
  }

  void _handleFirebaseError(FirebaseAuthException e) {
```

```

        String message = 'Login failed';
        if (e.code == 'user-not-found') {
            message = 'No account found for this email';
        } else if (e.code == 'wrong-password') {
            message = 'Incorrect password';
        } else if (e.code == 'invalid-email') {
            message = 'Invalid email format';
        }
        _showSnackbar(message);
    }

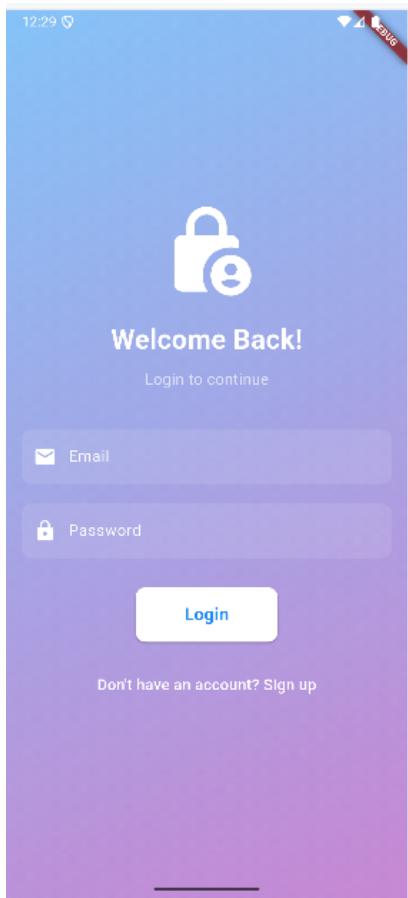
    void _showSnackbar(String message) {
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(content:
Text(message)));
    }

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: SingleChildScrollView(
                child: Container(
                    padding: const EdgeInsets.all(16.0),
                    height: MediaQuery.of(context).size.height,
                    decoration: BoxDecoration(
                        gradient: LinearGradient(
                            colors: [Colors.blue.shade200, Colors.purple.shade200],
                            begin: Alignment.topLeft,
                            end: Alignment.bottomRight,
                        ),
                    ),
                ),
            ),
            child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                    // App Logo or Icon
                    const Icon(
                        Icons.lock_person,
                        size: 100,
                        color: Colors.white,
                    ),
                    const SizedBox(height: 20),
                    const Text(
                        'Welcome Back!',
                        style: TextStyle(
                            fontSize: 28,
                            fontWeight: FontWeight.bold,
                            color: Colors.white,
                        ),
                    ),
                    const SizedBox(height: 10),

```

```
const Text(
  'Login to continue',
  style: TextStyle(
    fontSize: 16,
    color: Colors.white70,
  ),
),
const SizedBox(height: 40),
// Email Field
TextField(
  controller: emailController,
  decoration: InputDecoration(
    labelText: 'Email',
    labelStyle: const TextStyle(color: Colors.white),
    filled: true,
    fillColor: Colors.white.withOpacity(0.1),
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(10),
      borderSide: BorderSide.none,
    ),
    prefixIcon: const Icon(Icons.email, color: Colors.white),
  ),
  style: const TextStyle(color: Colors.white),
  keyboardType: TextInputType.emailAddress,
),
const SizedBox(height: 20),
// Password Field
TextField(
  controller: passwordController,
  obscureText: true,
  decoration: InputDecoration(
    labelText: 'Password',
    labelStyle: const TextStyle(color: Colors.white),
    filled: true,
    fillColor: Colors.white.withOpacity(0.1),
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(10),
      borderSide: BorderSide.none,
    ),
    prefixIcon: const Icon(Icons.lock, color: Colors.white),
  ),
  style: const TextStyle(color: Colors.white),
),
const SizedBox(height: 30),
// Login Button
isLoading
  ? const CircularProgressIndicator(color: Colors.white)
  : ElevatedButton(
    onPressed: _login,
```

```
        style: ElevatedButton.styleFrom(
            backgroundColor: Colors.white,
            padding: const EdgeInsets.symmetric(
                horizontal: 50, vertical: 15),
            shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(10),
            ),
        ),
        child: const Text(
            'Login',
            style: TextStyle(
                fontSize: 18,
                color: Colors.blue,
                fontWeight: FontWeight.bold,
            ),
        ),
    ),
),
const SizedBox(height: 20),
// Signup Link
white,
fontSize: 16,
```



MPL Experiment 5

1. Navigation

Login Page	Signup page
<pre> import 'package:flutter/material.dart'; import 'package:firebase_auth/firebase_auth.dart'; import 'home_page.dart'; import 'signup_page.dart'; class LoginPage extends StatefulWidget { const LoginPage({super.key}); @override _LoginPageState createState() => _LoginPageState(); } class _LoginPageState extends State<LoginPage> { final FirebaseAuth _auth = FirebaseAuth.instance; final TextEditingController emailController = TextEditingController(); final TextEditingController passwordController = TextEditingController(); bool isLoading = false; Future<void> _login() async { final email = emailController.text.trim(); final password = passwordController.text.trim(); if (email.isEmpty password.isEmpty) { _showSnackbar('Please enter both email and password'); return; } setState(() => isLoading = true); try { await _auth.signInWithEmailAndPassword(email: email, password: password); _showSnackbar('Login Successful'); // Navigate to HomePage after successful login Navigator.pushReplacement(context, MaterialPageRoute(builder: (context) => const HomePage()),); } on FirebaseAuthException catch (e) { _handleFirebaseError(e); } finally { setState(() => isLoading = false); } } </pre>	<pre> import 'package:flutter/material.dart'; import 'package:firebase_auth/firebase_auth.dart'; import 'login_page.dart'; class SignupPage extends StatefulWidget { const SignupPage({super.key}); @override _SignupPageState createState() => _SignupPageState(); } class _SignupPageState extends State<SignupPage> { final FirebaseAuth _auth = FirebaseAuth.instance; final TextEditingController emailController = TextEditingController(); final TextEditingController passwordController = TextEditingController(); final TextEditingController confirmPasswordController = TextEditingController(); bool isLoading = false; Future<void> _signup() async { final email = emailController.text.trim(); final password = passwordController.text.trim(); final confirmPassword = confirmPasswordController.text.trim(); if (email.isEmpty password.isEmpty confirmPassword.isEmpty) { _showSnackbar('Please fill all fields'); return; } if (password != confirmPassword) { _showSnackbar('Passwords do not match'); return; } setState(() => isLoading = true); try { await </pre>

```

}

void _handleFirebaseError(FirebaseAuthException e)
{
    String message = 'Login failed';
    if (e.code == 'user-not-found') {
        message = 'No account found for this email';
    } else if (e.code == 'wrong-password') {
        message = 'Incorrect password';
    } else if (e.code == 'invalid-email') {
        message = 'Invalid email format';
    }
    _showSnackbar(message);
}

void _showSnackbar(String message) {
    ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text(message)));
}

@Override
Widget build(BuildContext context) {
    return Scaffold(
        body: SingleChildScrollView(
            child: Container(
                padding: const EdgeInsets.all(16.0),
                height: MediaQuery.of(context).size.height,
                decoration: BoxDecoration(
                    gradient: LinearGradient(
                        colors: [Colors.blue.shade200,
Colors.purple.shade200],
                        begin: Alignment.topLeft,
                        end: Alignment.bottomRight,
                    ),
                ),
                child: Column(
                    mainAxisAlignment:
MainAxisAlignment.center,
                    children: [
                        // App Logo or Icon
                        const Icon(
                            Icons.lock_person,
                            size: 100,
                            color: Colors.white,
                        ),
                        const SizedBox(height: 20),
                        const Text(
                            'Welcome Back!',
                            style: TextStyle(
                                fontSize: 28,
                                fontWeight: FontWeight.bold,
                                color: Colors.white,
                            ),
                        ),
                        const SizedBox(height: 10),
                    ],
                ),
            ),
        ),
    );
}

```

```

_auth.createUserWithEmailAndPassword(email:
email, password: password);
    _showSnackbar('Signup Successful');

    // Navigate to LoginPage after successful
    // signup
    Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) =>
const LoginPage()),
    );
} on FirebaseAuthException catch (e) {
    _handleFirebaseError(e);
} finally {
    setState(() => isLoading = false);
}
}

void _handleFirebaseError(FirebaseAuthException e) {
    String message = 'Signup failed';
    if (e.code == 'email-already-in-use') {
        message = 'Email is already in use';
    } else if (e.code == 'weak-password') {
        message = 'Password is too weak';
    } else if (e.code == 'invalid-email') {
        message = 'Invalid email format';
    }
    _showSnackbar(message);
}

void _showSnackbar(String message) {
    ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text(message)));
}

@Override
Widget build(BuildContext context) {
    return Scaffold(
        body: SingleChildScrollView(
            child: Container(
                padding: const EdgeInsets.all(16.0),
                height:
MediaQuery.of(context).size.height,
                decoration: BoxDecoration(
                    gradient: LinearGradient(
                        colors: [Colors.blue.shade200,
Colors.purple.shade200],
                        begin: Alignment.topLeft,
                        end: Alignment.bottomRight,
                    ),
                ),
                child: Column(
                    mainAxisAlignment:
MainAxisAlignment.center,

```

```

const Text(
  'Login to continue',
  style: TextStyle(
    fontSize: 16,
    color: Colors.white70,
  ),
),
const SizedBox(height: 40),
// Email Field
TextField(
  controller: emailController,
  decoration: InputDecoration(
    labelText: 'Email',
    labelStyle: const TextStyle(color:
Colors.white),
    filled: true,
    fillColor:
Colors.white.withOpacity(0.1),
    border: OutlineInputBorder(
      borderRadius:
BorderRadius.circular(10),
      borderSide: BorderSide.none,
    ),
    prefixIcon: const Icon(Icons.email,
color: Colors.white),
  ),
  style: const TextStyle(color:
Colors.white),
  keyboardType:
TextInputType emailAddress,
),
const SizedBox(height: 20),
// Password Field
TextField(
  controller: passwordController,
  obscureText: true,
  decoration: InputDecoration(
    labelText: 'Password',
    labelStyle: const TextStyle(color:
Colors.white),
    filled: true,
    fillColor:
Colors.white.withOpacity(0.1),
    border: OutlineInputBorder(
      borderRadius:
BorderRadius.circular(10),
      borderSide: BorderSide.none,
    ),
    prefixIcon: const Icon(Icons.lock,
color: Colors.white),
  ),
  style: const TextStyle(color:
Colors.white),
),
const SizedBox(height: 30),
// Login Button

```

```

children: [
  // App Logo or Icon
  const Icon(
    Icons.person_add,
    size: 100,
    color: Colors.white,
  ),
  const SizedBox(height: 20),
  const Text(
    'Create an Account',
    style: TextStyle(
      fontSize: 28,
      fontWeight: FontWeight.bold,
      color: Colors.white,
    ),
  ),
  const SizedBox(height: 10),
  const Text(
    'Sign up to get started',
    style: TextStyle(
      fontSize: 16,
      color: Colors.white70,
    ),
  ),
  const SizedBox(height: 40),
  // Email Field
  TextField(
    controller: emailController,
    decoration: InputDecoration(
      labelText: 'Email',
      labelStyle: const
TextStyle(color: Colors.white),
      filled: true,
      fillColor:
Colors.white.withOpacity(0.1),
      border: OutlineInputBorder(
        borderRadius:
BorderRadius.circular(10),
        borderSide: BorderSide.none,
      ),
      prefixIcon: const
Icon(Icons.email, color: Colors.white),
    ),
    style: const TextStyle(color:
Colors.white),
    keyboardType:
TextInputType emailAddress,
  ),
  const SizedBox(height: 20),
  // Password Field
  TextField(
    controller: passwordController,
    obscureText: true,
    decoration: InputDecoration(
      labelText: 'Password',
      labelStyle: const TextStyle(color:
Colors.white),
      filled: true,
      fillColor:
Colors.white.withOpacity(0.1),
      border: OutlineInputBorder(
        borderRadius:
BorderRadius.circular(10),
        borderSide: BorderSide.none,
      ),
      prefixIcon: const
Icon(Icons.lock, color: Colors.white),
    ),
    style: const TextStyle(color:
Colors.white),
  )
]

```

```

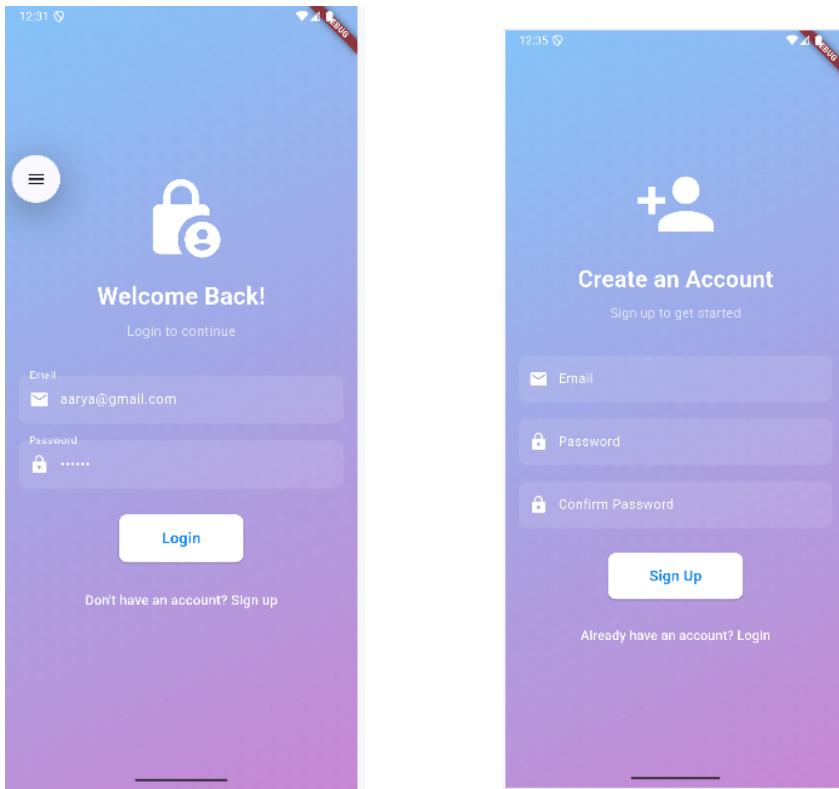
isLoading
    ? const
CircularProgressIndicator(color: Colors.white)
    : ElevatedButton(
        onPressed: _login,
        style: ElevatedButton.styleFrom(
            backgroundColor: Colors.white,
            padding: const
EdgeInsets.symmetric(
            horizontal: 50, vertical: 15),
            shape: RoundedRectangleBorder(
                borderRadius:
BorderRadius.circular(10),
            ),
        ),
        child: const Text(
            'Login',
            style: TextStyle(
                fontSize: 18,
                color: Colors.blue,
                fontWeight: FontWeight.bold,
            ),
        ),
    ),
    const SizedBox(height: 20),
// Signup Link
TextButton(
    onPressed: () {
        Navigator.push(
            context,
            MaterialPageRoute(builder:
(context) => const SignupPage()),
        );
    },
    child: const Text(
        'Don\'t have an account? Sign up',
        style: TextStyle(
            color: Colors.white,
            fontSize: 16,
        ),
    ),
),
],
),
),
),
);
}
}

```

```

TextStyle(color: Colors.white),
filled: true,
fillColor:
Colors.white.withOpacity(0.1),
border: OutlineInputBorder(
    borderRadius:
BorderRadius.circular(10),
    borderSide: BorderSide.none,
),
prefixIcon: const
Icon(Icons.lock, color: Colors.white),
),
style: const TextStyle(color:
Colors.white),
),
const SizedBox(height: 20),
// Confirm Password Field
TextField(
    controller:
confirmPasswordController,
    obscureText: true,
    decoration: InputDecoration(
        labelText: 'Confirm Password',
        labelStyle: const
TextStyle(color: Colors.white),
filled: true,
fillColor:
Colors.white.withOpacity(0.1),
border: OutlineInputBorder(
    borderRadius:
BorderRadius.circular(10),
    borderSide: BorderSide.none,
),
prefixIcon: const
Icon(Icons.lock, color: Colors.white),
),
style: const TextStyle(color:
Colors.white),
),
const SizedBox(height: 30),
// Signup Button
isLoading
    ? const
CircularProgressIndicator(color: Colors.white)
    : ElevatedButton(
        onPressed: _signup,
        style: ElevatedButton.styleFrom(
            backgroundColor: Colors.white,
            padding: const
EdgeInsets.symmetric(
            horizontal: 50, vertical:
15),
            shape: RoundedRectangleBorder(
                borderRadius:
BorderRadius.circular(10),
            )
        )
    )

```



2. Gestures

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

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

class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _MyAppState();
}

class _MyAppState extends State<HomePage> {
  late GoogleMapController mapController;

  final LatLng _center = const LatLng(45.521563, -122.677433);

  void _onMapCreated(GoogleMapController controller) {
    mapController = controller;
  }
}
```

```

        }

    @override
    Widget build(BuildContext context) {
        return MaterialApp(
            theme: ThemeData(
                useMaterial3: true,
                colorSchemeSeed: Colors.green[700],
            ),
            home: Scaffold(
                appBar: AppBar(
                    title: const Text('Maps Sample App'),
                    elevation: 2,
                ),
                body: GoogleMap(
                    onMapCreated: _onMapCreated,
                    initialCameraPosition: CameraPosition(
                        target: _center,
                        zoom: 11.0,
                    ),
                ),
            ),
        );
    }
}

```



MPL Experiment 6

Google-services.json

```
{  
  "project_info": {  
    "project_number": "663509823079",  
    "project_id": "complaint1",  
    "storage_bucket": "complaint1.firebaseio.storage.app"  
  },  
  "client": [  
    {  
      "client_info": {  
        "mobilesdk_app_id": "1:663509823079:android:f919160ed188bc2ee42ec2",  
        "android_client_info": {  
          "package_name": "com.example.new_complaint"  
        }  
      },  
      "oauth_client": [],  
      "api_key": [  
        {  
          "current_key": "AIzaSyALbN6aQftIodwXpZmdIRzO0JDSUhD4ojk"  
        }  
      ],  
      "services": {  
        "appinvite_service": {  
          "other_platform_oauth_client": []  
        }  
      }  
    },  
    {"configuration_version": "1"}  
}
```

Login Page

```
import 'package:flutter/material.dart';  
import 'package:firebase_auth/firebase_auth.dart';  
import 'home_page.dart';  
import 'signup_page.dart';  
  
class LoginPage extends StatefulWidget {  
  const LoginPage({super.key});  
  
  @override  
  _LoginPageState createState() => _LoginPageState();  
}
```

```
class _LoginPageState extends State<LoginPage> {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();
  bool isLoading = false;

  Future<void> _login() async {
    final email = emailController.text.trim();
    final password = passwordController.text.trim();

    if (email.isEmpty || password.isEmpty) {
      _showSnackbar('Please enter both email and password');
      return;
    }

    setState(() => isLoading = true);

    try {
      await _auth.signInWithEmailAndPassword(email: email, password: password);
      _showSnackbar('Login Successful');

      // Navigate to HomePage after successful login
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => const HomePage()),
      );
    } on FirebaseAuthException catch (e) {
      _handleFirebaseError(e);
    } finally {
      setState(() => isLoading = false);
    }
  }

  void _handleFirebaseError(FirebaseAuthException e) {
    String message = 'Login failed';
    if (e.code == 'user-not-found') {
      message = 'No account found for this email';
    } else if (e.code == 'wrong-password') {
      message = 'Incorrect password';
    } else if (e.code == 'invalid-email') {
      message = 'Invalid email format';
    }
    _showSnackbar(message);
  }

  void _showSnackbar(String message) {
    ScaffoldMessenger.of(context).showSnackBar(SnackBar(content:
    Text(message)));
  }
}
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: SingleChildScrollView(
      child: Container(
        padding: const EdgeInsets.all(16.0),
        height: MediaQuery.of(context).size.height,
        decoration: BoxDecoration(
          gradient: LinearGradient(
            colors: [Colors.blue.shade200, Colors.purple.shade200],
            begin: Alignment.topLeft,
            end: Alignment.bottomRight,
          ),
        ),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            // App Logo or Icon
            const Icon(
              Icons.lock_person,
              size: 100,
              color: Colors.white,
            ),
            const SizedBox(height: 20),
            const Text(
              'Welcome Back!',
              style: TextStyle(
                fontSize: 28,
                fontWeight: FontWeight.bold,
                color: Colors.white,
              ),
            ),
            const SizedBox(height: 10),
            const Text(
              'Login to continue',
              style: TextStyle(
                fontSize: 16,
                color: Colors.white70,
              ),
            ),
          ],
        ),
        const SizedBox(height: 40),
        // Email Field
        TextField(
          controller: emailController,
          decoration: InputDecoration(
            labelText: 'Email',
            labelStyle: const TextStyle(color: Colors.white),
            filled: true,
          ),
        ),
      ),
    ),
  );
}
```

```
        fillColor: Colors.white.withOpacity(0.1),
        border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(10),
            borderSide: BorderSide.none,
        ),
        prefixIcon: const Icon(Icons.email, color: Colors.white),
    ),
    style: const TextStyle(color: Colors.white),
    keyboardType: TextInputType.emailAddress,
),
const SizedBox(height: 20),
// Password Field
TextField(
    controller: passwordController,
    obscureText: true,
    decoration: InputDecoration(
        labelText: 'Password',
        labelStyle: const TextStyle(color: Colors.white),
        filled: true,
        fillColor: Colors.white.withOpacity(0.1),
        border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(10),
            borderSide: BorderSide.none,
        ),
        prefixIcon: const Icon(Icons.lock, color: Colors.white),
    ),
    style: const TextStyle(color: Colors.white),
),
const SizedBox(height: 30),
// Login Button
isLoading
? const CircularProgressIndicator(color: Colors.white)
: ElevatedButton(
    onPressed: _login,
    style: ElevatedButton.styleFrom(
        backgroundColor: Colors.white,
        padding: const EdgeInsets.symmetric(
            horizontal: 50, vertical: 15),
        shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(10),
        ),
    ),
    child: const Text(
        'Login',
        style: TextStyle(
            fontSize: 18,
            color: Colors.blue,
            fontWeight: FontWeight.bold,
        ),
    ),
)
```

```
        ),
    ),
    const SizedBox(height: 20),
    // Signup Link
    TextButton(
        onPressed: () {
            Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => const SignupPage()),
            );
        },
        child: const Text(
            'Don\'t have an account? Sign up',
            style: TextStyle(
                color: Colors.white,
                fontSize: 16,
            ),
        ),
    ),
),
],
),
),
),
),
);
}
}
```

Signup Page:

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

class SignupPage extends StatefulWidget {
  const SignupPage({super.key});

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

class _SignupPageState extends State<SignupPage> {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();
  final TextEditingController confirmPasswordController =
  TextEditingController();
  bool isLoading = false;

  Future<void> _signup() async {
    final email = emailController.text.trim();
```

```

final password = passwordController.text.trim();
final confirmPassword = confirmPasswordController.text.trim();

if (email.isEmpty || password.isEmpty || confirmPassword.isEmpty) {
  _showSnackbar('Please fill all fields');
  return;
}

if (password != confirmPassword) {
  _showSnackbar('Passwords do not match');
  return;
}

setState(() => isLoading = true);

try {
  await _auth.createUserWithEmailAndPassword(email: email, password: password);
  _showSnackbar('Signup Successful');

  // Navigate to LoginPage after successful signup
  Navigator.pushReplacement(
    context,
    MaterialPageRoute(builder: (context) => const LoginPage()),
  );
} on FirebaseAuthException catch (e) {
  _handleFirebaseError(e);
} finally {
  setState(() => isLoading = false);
}
}

void _handleFirebaseError(FirebaseAuthException e) {
  String message = 'Signup failed';
  if (e.code == 'email-already-in-use') {
    message = 'Email is already in use';
  } else if (e.code == 'weak-password') {
    message = 'Password is too weak';
  } else if (e.code == 'invalid-email') {
    message = 'Invalid email format';
  }
  _showSnackbar(message);
}

void _showSnackbar(String message) {
  ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text(message)));
}

```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: SingleChildScrollView(
      child: Container(
        padding: const EdgeInsets.all(16.0),
        height: MediaQuery.of(context).size.height,
        decoration: BoxDecoration(
          gradient: LinearGradient(
            colors: [Colors.blue.shade200, Colors.purple.shade200],
            begin: Alignment.topLeft,
            end: Alignment.bottomRight,
          ),
        ),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            // App Logo or Icon
            const Icon(
              Icons.person_add,
              size: 100,
              color: Colors.white,
            ),
            const SizedBox(height: 20),
            const Text(
              'Create an Account',
              style: TextStyle(
                fontSize: 28,
                fontWeight: FontWeight.bold,
                color: Colors.white,
              ),
            ),
            const SizedBox(height: 10),
            const Text(
              'Sign up to get started',
              style: TextStyle(
                fontSize: 16,
                color: Colors.white70,
              ),
            ),
            const SizedBox(height: 40),
            // Email Field
            TextField(
              controller: emailController,
              decoration: InputDecoration(
                labelText: 'Email',
                labelStyle: const TextStyle(color: Colors.white),
                filled: true,
                fillColor: Colors.white.withOpacity(0.1),
              ),
            ),
          ],
        ),
      ),
    ),
  );
}
```

```
        border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(10),
            borderSide: BorderSide.none,
        ),
        prefixIcon: const Icon(Icons.email, color: Colors.white),
    ),
    style: const TextStyle(color: Colors.white),
    keyboardType: TextInputType.emailAddress,
),
const SizedBox(height: 20),
// Password Field
TextField(
    controller: passwordController,
    obscureText: true,
    decoration: InputDecoration(
        labelText: 'Password',
        labelStyle: const TextStyle(color: Colors.white),
        filled: true,
        fillColor: Colors.white.withOpacity(0.1),
        border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(10),
            borderSide: BorderSide.none,
        ),
        prefixIcon: const Icon(Icons.lock, color: Colors.white),
    ),
    style: const TextStyle(color: Colors.white),
),
const SizedBox(height: 20),
// Confirm Password Field
TextField(
    controller: confirmPasswordController,
    obscureText: true,
    decoration: InputDecoration(
        labelText: 'Confirm Password',
        labelStyle: const TextStyle(color: Colors.white),
        filled: true,
        fillColor: Colors.white.withOpacity(0.1),
        border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(10),
            borderSide: BorderSide.none,
        ),
        prefixIcon: const Icon(Icons.lock, color: Colors.white),
    ),
    style: const TextStyle(color: Colors.white),
),
const SizedBox(height: 30),
// Signup Button
isLoading
    ? const CircularProgressIndicator(color: Colors.white)
```

```

        : ElevatedButton(
        onPressed: _signup,
        style: ElevatedButton.styleFrom(
            backgroundColor: Colors.white,
            padding: const EdgeInsets.symmetric(
                horizontal: 50, vertical: 15),
            shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(10),
            ),
        ),
        child: const Text(
            'Sign Up',
            style: TextStyle(
                fontSize: 18,
                color: Colors.blue,
                fontWeight: FontWeight.bold,
            ),
        ),
    ),
),
const SizedBox(height: 20),
// Login Link
TextButton(
    onPressed: () {
        Navigator.pushReplacement(
            context,
            MaterialPageRoute(builder: (context) => const LoginPage()),
        );
    },
    child: const Text(
        'Already have an account? Login',
        style: TextStyle(
            color: Colors.white,
            fontSize: 16,
        ),
    ),
),
),
],
),
),
),
);
}
}
}

```

Firebase Screenshots:

complaint1 ▾

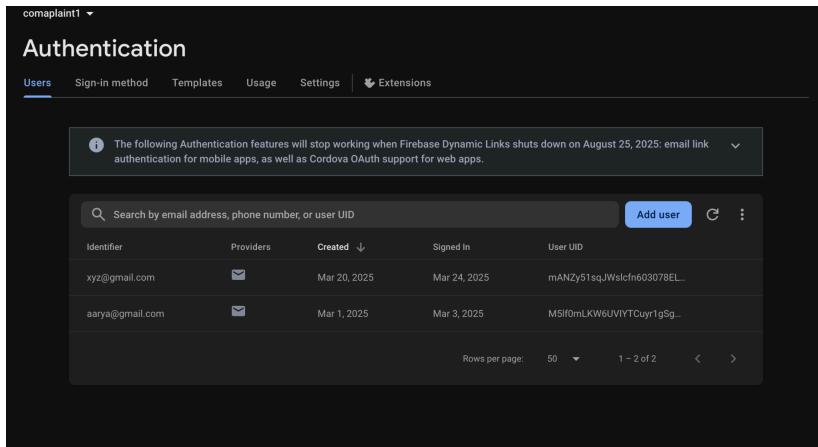
Authentication

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.

Identifier	Providers	Created	Signed In	User UID
xyz@gmail.com	✉️	Mar 20, 2025	Mar 24, 2025	mANZyS1sqJWslcnf603078EL...
aarya@gmail.com	✉️	Mar 1, 2025	Mar 3, 2025	MSif0mLKW6UVIYTCuyr1gSg...

Rows per page: 50 1 – 2 of 2 < >



complaint1 ▾

Project settings

General Cloud Messaging Integrations Service accounts Data privacy Users and permissions

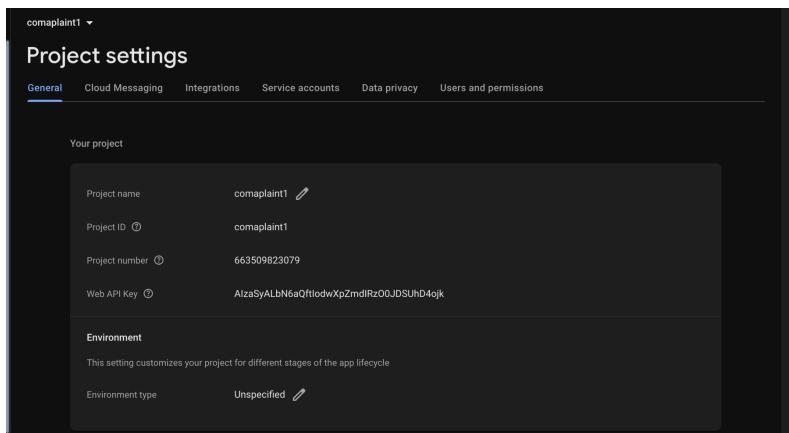
Your project

Project name	complaint1
Project ID	complaint1
Project number	663509823079
Web API Key	AizaSyALbn5aQtfiodwXpZmdlRzO0JDShD4ojk

Environment

This setting customizes your project for different stages of the app lifecycle

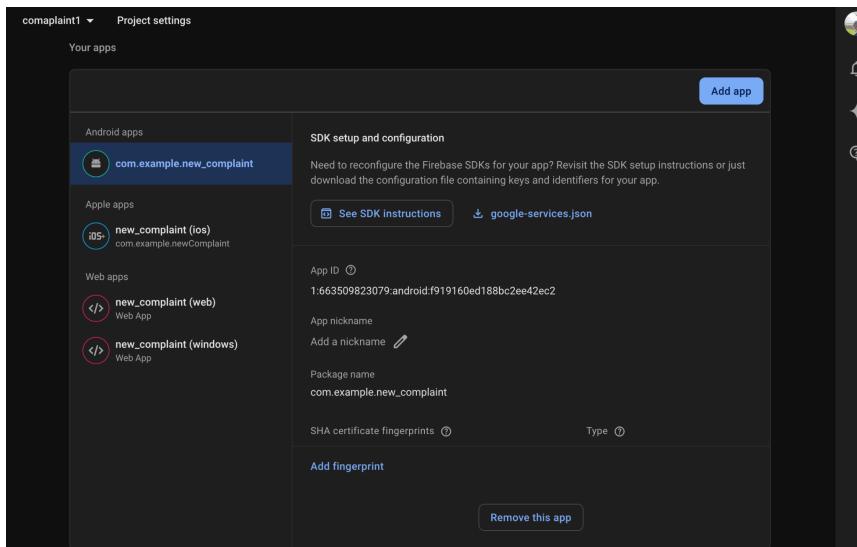
Environment type Unspecified



complaint1 ▾ Project settings

Your apps

Android apps	SDK setup and configuration
 com.example.new_complaint	Need to reconfigure the Firebase SDKs for your app? Revisit the SDK setup instructions or just download the configuration file containing keys and identifiers for your app.
Apple apps	See SDK instructions google-services.json
 new_complaint (ios)	
Web apps	
 new_complaint (web)	App ID 1:663509823079.android:f919160ed188bc2ee42ec2
 new_complaint (windows)	App nickname Add a nickname
	Package name com.example.new_complaint
	SHA certificate fingerprints Type
	Add fingerprint
	Remove this app



MPL Experiment 7

Folder Structure

exp7		Downloads		
Name	Date Modified	Size	Kind	
images	Today at 2:39 PM	--	Folder	
edge.jpg	Today at 2:39 PM	17 KB	JPEG image	
ig.jpg	Today at 2:34 PM	7 KB	JPEG image	
index.html	Today at 2:18 PM	781 bytes	HTML text	
manifest.json	Today at 2:40 PM	445 bytes	JSON	
serviceworker.js	Today at 2:12 PM	467 bytes	JavaSc...t script	

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>PWA Experiment 7</title>

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

    <style>
      body {
        font-family: "Arial", sans-serif;
        background: linear-gradient(to right, #4facfe, #00f2fe);
        color: #fff;
        text-align: center;
        margin: 0;
        padding: 0;
        display: flex;
      }
    </style>
  </head>
  <body>
    <h1>Hello World</h1>
    <p>This is a PWA Experiment 7</p>
  </body>
</html>
```

```
flex-direction: column;
align-items: center;
justify-content: center;
height: 100vh;
}

h1 {
  font-size: 2.5rem;
  font-weight: bold;
  margin-bottom: 10px;
}

h4 {
  font-size: 1.2rem;
  font-weight: normal;
  margin-bottom: 20px;
}

.install-btn {
  background-color: #ff4d6d;
  color: white;
  border: none;
  padding: 12px 20px;
  font-size: 1rem;
  border-radius: 25px;
  cursor: pointer;
  transition: 0.3s;
}

.install-btn:hover {
  background-color: #ff1e4d;
}

.container {
  background: rgba(255, 255, 255, 0.2);
  padding: 20px;
  border-radius: 10px;
  backdrop-filter: blur(10px);
  box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
  max-width: 500px;
}

</style>
```

```

<script>
    window.addEventListener("load", () => {
        registerSW();
    });

    async function registerSW() {
        if ("serviceWorker" in navigator) {
            try {
                await navigator.serviceWorker.register("/serviceworker.js");
                console.log("Service Worker Registered Successfully");
            } catch (error) {
                console.error("Service Worker Registration Failed:", error);
            }
        } else {
            console.warn("Service Worker is not supported in this browser.");
        }
    }
</script>
</head>
<body>
    <div class="container">
        <h1>Hello!!! This is Aarya Gandhi</h1>
        <h4>This is a sample page for the implementation of PWA Experiment 7</h4>
    </div>
</body>
</html>

```

Manifest.json

```
{
    "name": "MPL experiment 7",
    "short_name": "PWA",
    "start_url": "/index.html",
    "display": "standalone",
    "background_color": "#5900b3",
    "theme_color": "black",
    "scope": "/",
    "description": "This is a PWA tutorial.",
    "icons": [

```

```

    {
      "src": "/images/ig.jpg",
      "sizes": "192x192",
      "type": "image/jpg"
    },
    {
      "src": "/images/edge.jpg",
      "sizes": "512x512",
      "type": "image/jpg"
    }
  ]
}

```

ServiceWorker.js

```

var staticCacheName="pwa";

self.addEventListener("Install",function(e){
  e.waitUntil(
    caches.open(staticCacheName).then(function(cache) {
      return cache.addAll(["/"])
    })
  );
}) ;

self.addEventListener("fetch",function(event){
  console.log(event.request.url);

  event.respondWith(
    caches.match(event.request).then(function(response) {
      return response || fetch(event.request);
    })
  );
}) ;

```

Steps to Implement:

1. open Developer tools options->g to applications

The screenshot shows the Chrome DevTools Application tab open for a PWA. The left sidebar displays the app's interface with a header 'FOODMART GROCERY STORE' and a main content area featuring a smoothie bottle and promotional text. The right sidebar is the 'Application' panel, which includes sections for 'Manifest' (containing 'manifest.json'), 'Errors and warnings', 'Installability', 'Identity' (with fields for Name, Short name, Description, and App ID), and 'Presentation' (with fields for Start URL, Theme color, Background color, and Orientation). A note about the 'id' field is present in the 'Installability' section.

- Click on the top right corner of browser from the app option install this site as an app

The screenshot shows a browser menu open on the right side of the screen. The 'More Tools' menu is expanded, and the 'Install This Site as an App' option is highlighted with a green box. The main content area shows the 'FOODMART GROCERY STORE' website with its product listing and promotional banners.

- App icon

Edge Apps			
Name	Date Modified	Size	Kind
⌚ MPL experiment 7	Today at 2:43 PM	2.4 MB	Application

Experiment 8

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.

Output:

1. Service Worker Implementation (serviceworker.js)

```
// Define a cache name for your app
const CACHE_NAME = 'e-commerce-pwa-v1';

// List of files to cache for offline use
const filesToCache = [
  'index.html',
  'manifest.json',
  'images/ig.jpg',
  'images/edge.jpg',
  // Add other assets your app needs
];

// Install event - Cache the essential files
self.addEventListener('install', (event) => {
  console.log('Service Worker: Installing...');
  // waitUntil() ensures the service worker won't install until
  // the code inside has completed
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => {
        console.log('Service Worker: Caching files');
        return cache.addAll(filesToCache);
      })
      .then(() => self.skipWaiting()) // Force waiting service
      // worker to become active
  );
});
```

```

// Activate event - Clean up old caches
self.addEventListener('activate', (event) => {
  console.log('Service Worker: Activated');
  // Remove old caches
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cache) => {
          if (cache !== CACHE_NAME) {
            console.log('Service Worker: Clearing old cache',
cache);
            return caches.delete(cache);
          }
        })
      );
    });
  .then(() => self.clients.claim()) // Take control of all
clients
  );
});

// Fetch event - Serve from cache first, then network
self.addEventListener('fetch', (event) => {
  console.log('Service Worker: Fetching', event.request.url);
  event.respondWith(
    // Try the cache first
    caches.match(event.request)
    .then((response) => {
      // Return cached response if found
      if (response) {
        return response;
      }
    })
    // If not in cache, fetch from network
  );
});

```

```

        return fetch(event.request)
            .then((res) => {
                // Check if valid response
                if (!res || res.status !== 200 || res.type !==
'basic') {
                    return res;
                }

                // Clone the response
                let responseToCache = res.clone();

                // Add the new file to cache
                caches.open(CACHE_NAME)
                    .then((cache) => {
                        cache.put(event.request, responseToCache);
                    });
            }

            // Return the response
            return res;
        })
        .catch(() => {
            // If both cache and network fail, you could return
            a fallback page
            if (event.request.url.indexOf('.html') > -1) {
                return caches.match('index.html');
            }
        });
    );
}
);

```

2. Changes Made in the index.html

```

if ("serviceWorker" in navigator) {
    window.addEventListener("load", () => {

```

```

navigator.serviceWorker
  .register("serviceworker.js")
  .then((registration) => {
    console.log(
      "ServiceWorker registration successful with
      scope: ",
      registration.scope
    );
  })
  .catch((error) => {
    console.log("ServiceWorker registration failed: ", error);
  });
}

```

The screenshot shows the Chrome DevTools Application tab open for a PWA named "FOODMART GROCERY STORE". The main content area displays a promotional banner for "Fresh Smoothie & Summer Juice". The DevTools sidebar on the left lists various service worker components:

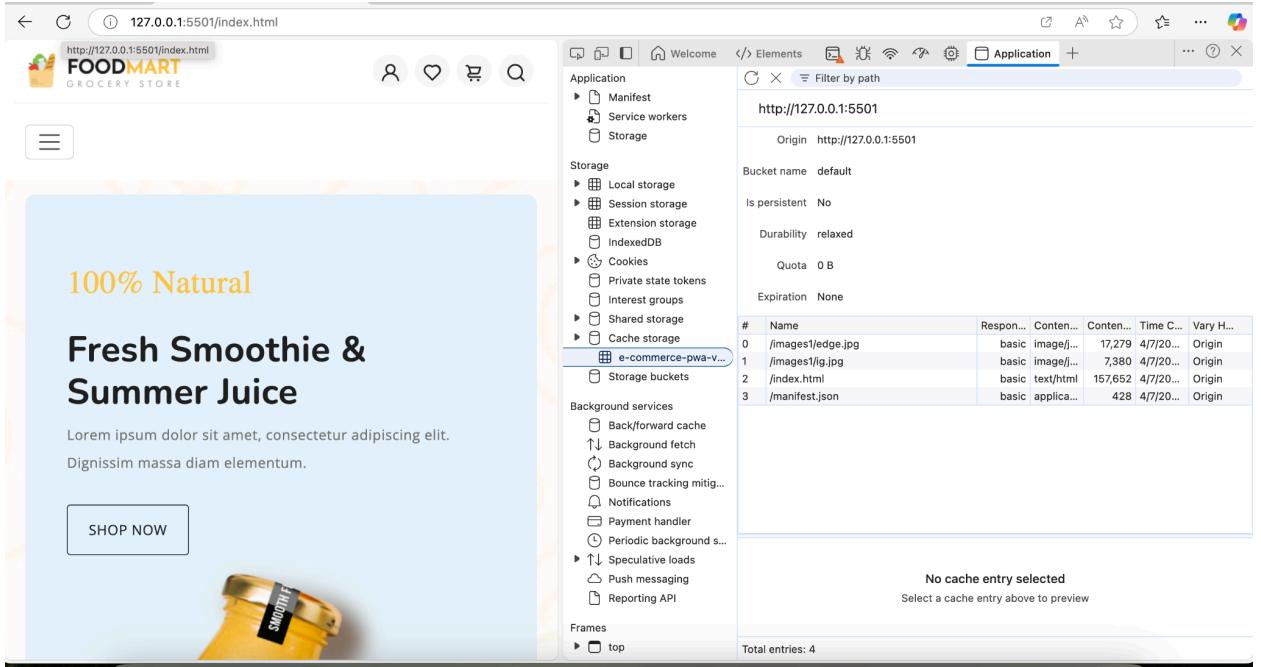
- Application** (selected)
 - Manifest
 - Service workers (selected)
 - Storage
 - Background services
 - Frames
- Storage**
 - Local storage
 - Session storage
 - Extension storage
 - IndexedDB
 - Cookies
 - Private state tokens
 - Interest groups
 - Shared storage
 - Cache storage
 - e-commerce-pwa-v...
 - Storage buckets
- Background services**
 - Back/forward cache
 - Background fetch
 - Background sync
 - Bounce tracking mitig...
 - Notifications
 - Payment handler
 - Periodic background s...
 - Speculative loads
 - Push messaging
 - Reporting API
- Frames**

The **Service workers** section for the URL <http://127.0.0.1:5501/> shows:

- Source:** serviceworker.js
- Status:** #20 activated and is running (Stop)
- Client:** http://127.0.0.1:5501/index.html
- 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
#20	Install		<div style="width: 100%; height: 10px; background-color: #007bff;"></div>
#20	Wait		<div style="width: 100%; height: 10px; background-color: #ffc107;"></div>
#20	Activate		<div style="width: 100%; height: 10px; background-color: #28a745;"></div>

The second instance of the service worker shows it is redundant (#19).



Conclusion

The implementation successfully demonstrates all stages of the service worker lifecycle as outlined in the practical document. The service worker now enables offline access to the application by caching essential resources during installation and serving them when needed, providing a more reliable user experience even without an internet connection.

Experiment 9

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

Output:

Manifest.json

```
{  
  "name": "MPL experiment 7",  
  "short_name": "PWA",  
  "start_url": "/index.html",  
  "display": "standalone",  
  "background_color": "#5900b3",  
  "theme_color": "black",  
  "scope": "/",  
  "description": "This is a PWA tutorial.",  
  "icons": [  
    {  
      "src": "/images1/ig.jpg",  
      "sizes": "192x192",  
      "type": "image/jpg"  
    },  
    {  
      "src": "/images1/edge.jpg",  
      "sizes": "512x512",  
      "type": "image/jpg"  
    }  
  ]  
}
```

Index.html (Script)

```
document.addEventListener("DOMContentLoaded", function () {  
  // Register Service Worker after page load  
  window.addEventListener("load", () => {  
    registerSW();  
  });  
  
  async function registerSW() {  
    if ("serviceWorker" in navigator) {
```

```

try {
    await navigator.serviceWorker.register("/serviceworker.js");
    console.log("Service Worker Registered Successfully");
} catch (error) {
    console.error("Service Worker Registration Failed:", error);
}
} else {
    console.warn("Service Worker is not supported in this browser.");
}
}

// Place Order Button Logic
const placeOrderBtn = document.getElementById("place-order-btn");
if (placeOrderBtn) {
    placeOrderBtn.addEventListener("click", function () {
        if ("serviceWorker" in navigator && "SyncManager" in window) {
            const order = {
                id: Date.now(),
                items: [
                    { name: "Product 1", price: 99.99, quantity: 1 },
                    { name: "Product 2", price: 49.99, quantity: 2 }
                ],
                total: 199.97,
                timestamp: new Date().toISOString()
            };

            console.log("Order saved locally:", order);
            alert("Order placed! It will sync when you're online.");
        }
    });
}

navigator.serviceWorker.ready.then(function (registration) {
    registration.sync.register("order-sync")
        .then(() => {
            console.log("Sync registered for order");
        })
        .catch(err => {
            console.error("Sync registration failed:", err);
        });
});
} else {
    alert("Background Sync not supported in your browser");
}
);
}

```

```
    } else {
        console.warn("Place Order button not found");
    }

// Test Push Notification Logic
const testPushBtn = document.getElementById("test-push-btn");
if (testPushBtn) {
    testPushBtn.addEventListener("click", function () {
        if ("serviceWorker" in navigator) {
            navigator.serviceWorker.ready.then(function (registration) {
                Notification.requestPermission().then(function (permission) {
                    if (permission === "granted") {
                        registration.active?.postMessage({
                            type: "push",
                            data: {
                                method: "pushMessage",
                                message: "Your order is ready for pickup!",
                                url: "/orders"
                            }
                        });
                    } else {
                        alert("Notification permission denied");
                    }
                });
            });
        } else {
            alert("Service Worker not supported in your browser");
        }
    });
} else {
    console.warn("Test Push button not found");
}
});
```

← ⌘ 127.0.0.1:5501/index.html

The screenshot shows a web browser window with the URL 127.0.0.1:5501/index.html. The main content area displays a grocery store website for "FOOD MART GROCERY STORE". The page features a header with a logo, user icons, and a search bar. Below the header is a promotional banner with the text "100% Natural Fresh Smoothie & Summer Juice" and a "SHOP NOW" button. The main content area contains placeholder text: "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Dignissim massa diam elementum." To the right of the main content is the Chrome DevTools Application tab, which is focused on the "Service workers" section. It shows a list of registered service workers, including "serviceworker.js" which is redundant. There are sections for "Push", "Sync", and "Periodic sync" with their respective test messages. The "Update Cycle" section shows the current state as "Install" and "Activate". The "Console" tab at the bottom shows a log of network requests handled by the service worker, including fetches for CSS files, images, and JavaScript files from both local and CDN sources.

FOOD MART
GROCERY STORE

100% Natural

Fresh Smoothie &
Summer Juice

SHOP NOW

Service Worker: Fetching <http://127.0.0.1:5501/css/vendor.css>
Service Worker: Fetching <http://127.0.0.1:5501/style.css>
Service Worker: Fetching <http://127.0.0.1:5501/images/logo.png>
Service Worker: Fetching <http://127.0.0.1:5501/images/product-thumb-1.png>
Found in cache: <http://127.0.0.1:5501/css/vendor.css>
Found in cache: <http://127.0.0.1:5501/style.css>
Found in cache: <http://127.0.0.1:5501/images/logo.png>
Found in cache: <http://127.0.0.1:5501/images/product-thumb-1.png>
Service Worker: Fetching <http://127.0.0.1:5501/images/icon-vegetables-broccoli.png>
Service Worker: Fetching <https://cdn.jsdelivr.net/npm/swiper@9/swiper-bundle.min.css>
Found in cache: <http://127.0.0.1:5501/images/icon-vegetables-broccoli.png>
Service Worker: Fetching <http://127.0.0.1:5501/js/jquery-1.11.0.min.js>
Service Worker: Fetching <https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js>
serviceworker.js:50
serviceworker.js:50
serviceworker.js:50
serviceworker.js:50
serviceworker.js:68
serviceworker.js:50
serviceworker.js:50
serviceworker.js:50
serviceworker.js:50
serviceworker.js:68
serviceworker.js:68
serviceworker.js:50
serviceworker.js:50
serviceworker.js:50
serviceworker.js:50

Experiment 10

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

Output:

Steps to Deploy an E-commerce PWA to GitHub Pages

1. **Create a GitHub Repository** – Set up a new public repository on GitHub.
2. **Initialize Git** – Link your local project to the repository and commit your code.
3. **Push Code to GitHub** – Upload your project files to GitHub.
4. **Enable GitHub Pages** – Go to the repository's **Settings > Pages**, set the source to the **main** branch, and save.
5. **Install Deployment Tool** – Add the necessary package for GitHub Pages deployment.
6. **Update Project Settings** – Modify the project configuration to specify the deployment URL.
7. **Deploy the PWA** – Build the project and publish it to GitHub Pages.
8. **Access Live Site** – Visit the provided GitHub Pages link to view your deployed PWA

The screenshot shows the GitHub repository settings page for 'CrossGo14 / MPL-exp10'. The 'Settings' tab is selected. On the left, the 'Pages' section is highlighted. The main area displays the 'GitHub Pages' configuration, showing that the site is live at <https://crossgo14.github.io/MPL-exp10/>. It also indicates that the site was last deployed by 'CrossGo14' 1 hour ago. Below this, the 'Build and deployment' section shows the 'Source' is set to 'Deploy from a branch' with 'Branch' set to 'main'. There are buttons for 'Save' and 'Deploy'. Further down, sections for 'Custom domain' and 'Integrations' are visible.

Screenshot of a GitHub repository page for 'MPL-exp10' (Public). The repository has 1 branch and 0 tags. The main branch contains 11 files: .vscode, css, images, images1, js, .DS_Store, index.html, manifest.json, serviceworker.js, and style.css. All files are first commits made 1 hour ago by 'Aarya Gandhi and Aarya Gandhi'. The commit message is 'Initial commit: Deploy FoodMart PWA'. The repository has 3 commits and was last updated 1 hour ago.

About

No description provided.

Activity

0 stars

1 watching

0 forks

Releases

No releases published. Create a new release

Packages

No packages published. Publish your first package

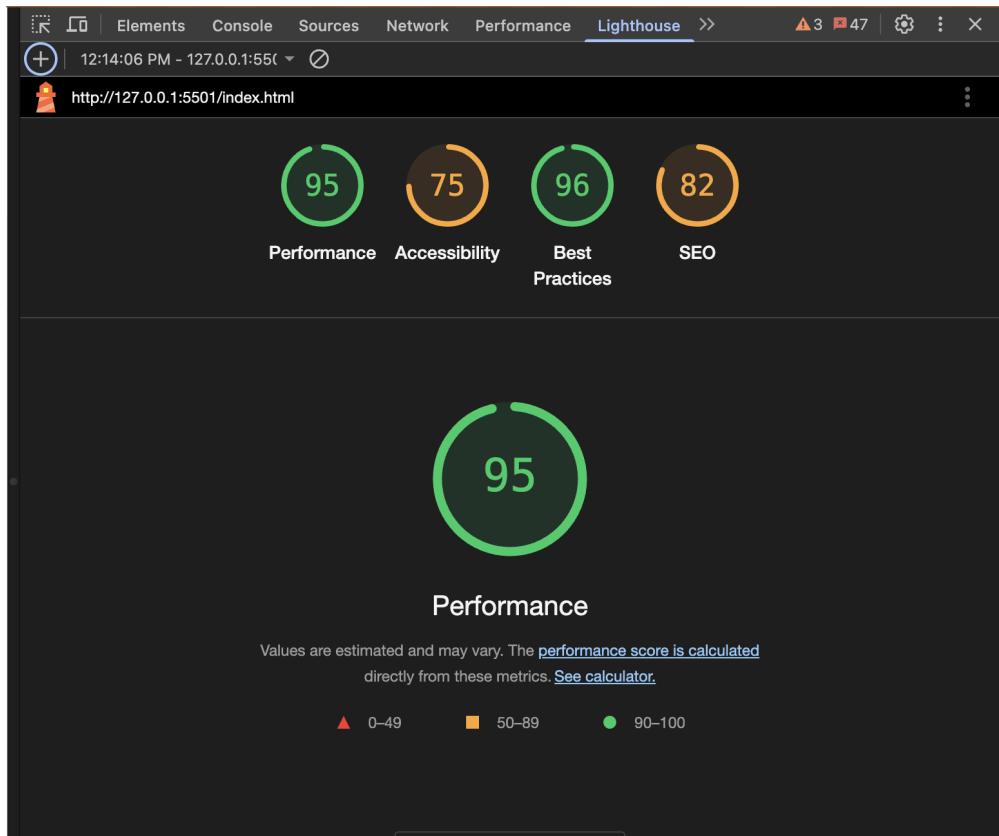
Deployments

github-pages

Screenshot of a web browser showing a mobile-optimized grocery store website, 'FoodMart GROCERY STORE'. The URL is 'crossgo14.github.io'. The cart contains \$1290.00 worth of items. The page features a hero section with a smoothie bottle and text: '100% Natural Fresh Smoothie & Summer Juice'. Below it is a lorem ipsum placeholder. To the right are promotional banners for 'Fruits & Vegetables' (20% off) and 'Baked Products' (15% off).

Experiment 11

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



The screenshot shows the Lighthouse analysis interface for a local development server at 127.0.0.1:5501. The overall score is 95, indicated by a green circle. Below the score, there are four circular icons with scores: 75 (orange), 96 (green), and 82 (orange). The main section is titled "DIAGNOSTICS" and lists several performance and optimization suggestions:

- ▲ Serve images in next-gen formats — Potential savings of 2,211 KIB
- ▲ Enable text compression — Potential savings of 271 KIB
- ▲ Efficiently encode images — Potential savings of 818 KIB
- ▲ Largest Contentful Paint element — 1,420 ms
- ▲ Reduce unused JavaScript — Potential savings of 84 KIB
- ▲ Eliminate render-blocking resources — Potential savings of 270 ms
- ▲ Reduce unused CSS — Potential savings of 59 KIB
- ▲ Page prevented back/forward cache restoration — 3 failure reasons
- Image elements do not have explicit `width` and `height`
- Minify CSS — Potential savings of 5 KIB

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