# AIM:

Lab 1: Implementing a Login Screen in Flutter Application

Learning Resources

Flutter Login Tutorial - <https://www.youtube.com/watch?v=Dh-cTQJgM-Q>

Firebase Authentication Guide - <https://www.youtube.com/watch?v=_3W-JuIVFlg>

Practical Task:

Create a login screen using TextField widgets for username and password input.

Integrate Firebase Authentication for user login and registration.

Topics Covered:

Widgets: Text, Button, Layout

Handling User Input and Forms

Firebase Authentication Integration

# THEORY:

**1. Flutter Architecture and Widget System**

Flutter's widget system is the foundation of its UI framework, with **everything being a widget**. These widgets are broadly classified into:

* **StatefulWidget**: Widgets that can change their state during runtime.
* **StatelessWidget**: Widgets that don’t change once rendered.

**Code Example:**

import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(title: Text('Stateless and Stateful Widgets')),

body: Column(

children: [

MyStatelessWidget(),

MyStatefulWidget(),

],

),

),

);

}

}

// StatelessWidget

class MyStatelessWidget extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Center(

child: Text('This is a StatelessWidget'),

);

}

}

// StatefulWidget

class MyStatefulWidget extends StatefulWidget {

@override

\_MyStatefulWidgetState createState() => \_MyStatefulWidgetState();

}

class \_MyStatefulWidgetState extends State<MyStatefulWidget> {

int counter = 0;

@override

Widget build(BuildContext context) {

return Column(

children: [

Text('Counter: $counter'),

ElevatedButton(

onPressed: () => setState(() {

counter++;

}),

child: Text('Increment'),

),

],

);

}

}

**2. Networking in Flutter**

The http package is commonly used for making API requests in Flutter. It supports GET, POST, and other HTTP methods.

**Code Example:**

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'dart:convert';

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

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(home: NetworkingExample());

}

}

class NetworkingExample extends StatefulWidget {

@override

\_NetworkingExampleState createState() => \_NetworkingExampleState();

}

class \_NetworkingExampleState extends State<NetworkingExample> {

String? data;

Future<void> fetchData() async {

final response = await http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts/1'));

if (response.statusCode == 200) {

setState(() {

data = json.decode(response.body)['title'];

});

} else {

throw Exception('Failed to load data');

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('Networking Example')),

body: Center(

child: data == null

? ElevatedButton(

onPressed: fetchData,

child: Text('Fetch Data'),

)

: Text('Data: $data'),

),

);

}

}

**3. Displaying Data in Flutter**

Flutter provides widgets like ListView.builder, Card, and ListTile for rendering lists and dynamic data.

**Code Example:**

import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(home: DataDisplayExample());

}

}

class DataDisplayExample extends StatelessWidget {

final List<String> items = ['Item 1', 'Item 2', 'Item 3', 'Item 4'];

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('Displaying Data')),

body: ListView.builder(

itemCount: items.length,

itemBuilder: (context, index) {

return Card(

child: ListTile(

title: Text(items[index]),

leading: Icon(Icons.label),

),

);

},

),

);

}

}

**4. Error Handling and Loading Indicators**

In Flutter, use CircularProgressIndicator for loading states and try-catch blocks for error handling.

**Code Example:**

Future<void> fetchData() async {

try {

final response = await http.get(Uri.parse('https://example.com/data'));

if (response.statusCode == 200) {

// Handle success

} else {

throw Exception('Failed to fetch data');

}

} catch (e) {

// Handle error

print(e);

}

}

**5. Navigating Between Pages**

Navigation in Flutter uses the Navigator class to manage routes.

**Code Example:**

import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

initialRoute: '/',

routes: {

'/': (context) => HomePage(),

'/details': (context) => DetailsPage(),

},

);

}

}

class HomePage extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('Home Page')),

body: Center(

child: ElevatedButton(

onPressed: () {

Navigator.pushNamed(context, '/details', arguments: 'Hello from Home Page');

},

child: Text('Go to Details'),

),

),

);

}

}

class DetailsPage extends StatelessWidget {

@override

Widget build(BuildContext context) {

final args = ModalRoute.of(context)!.settings.arguments as String;

return Scaffold(

appBar: AppBar(title: Text('Details Page')),

body: Center(child: Text(args)),

);

}

}

**6. WebView in Flutter**

The webview\_flutter package lets you embed web content in Flutter apps.

**Code Example:**

import 'package:flutter/material.dart';

import 'package:webview\_flutter/webview\_flutter.dart';

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

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(title: Text('WebView Example')),

body: WebView(

initialUrl: 'https://flutter.dev',

javascriptMode: JavascriptMode.unrestricted,

),

),

);

}

}

**7. Placeholder and Fallbacks**

Use the errorBuilder in Image widgets to show placeholders.

**Code Example:**

Image.network(

'https://example.com/image.png',

errorBuilder: (context, error, stackTrace) {

return Icon(Icons.error);

},

);

**8. Dependency Management**

Manage dependencies in the pubspec.yaml file.

**Example:**

dependencies:

flutter:

sdk: flutter

http: ^0.15.0

webview\_flutter: ^3.0.0

**9. Best Practices for User Experience**

* Always show a loading indicator while fetching data.
* Provide descriptive error messages.
* Use fallback content for empty or missing data.

**Code Example:**

if (data == null) {

return CircularProgressIndicator();

} else if (data.isEmpty) {

return Text('No data available');

} else {

return Text('Data: $data');

}

This detailed explanation and code should cover your outlined topics comprehensively! Let me know if you'd like deeper dives into any specific area.

# CODE:

**main.dart**

import 'package:flutter/material.dart';

import 'package:modernlogintute/pages/auth\_page.dart';

import 'pages/login\_page.dart';

import 'package:firebase\_core/firebase\_core.dart';

import 'firebase\_options.dart';

void main() async{

  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp(

    options: DefaultFirebaseOptions.currentPlatform,

  );

  runApp(const MyApp());

}

class MyApp extends StatelessWidget {

  const MyApp({super.key});

  @override

  Widget build(BuildContext context) {

    return const MaterialApp(

      debugShowCheckedModeBanner: false,

      home: AuthPage(),

    );

  }

}

**Page/home\_page.dart**

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/material.dart';

class HomePage extends StatefulWidget {

  HomePage({super.key});

  @override

  \_HomePageState createState() => \_HomePageState();

}

class \_HomePageState extends State<HomePage> {

  User? user;

  @override

  void initState() {

    super.initState();

    // Get the current user at the start

    user = FirebaseAuth.instance.currentUser;

  }

  // Sign the user out

  void signUserOut() async {

    await FirebaseAuth.instance.signOut();

    // After sign out, set the user to null and rebuild

    setState(() {

      user = null;

    });

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        actions: [

          IconButton(

            onPressed: signUserOut,

            icon: Icon(Icons.logout),

          ),

        ],

      ),

      body: Center(

        child: user == null

            ? Text("You are logged out.")

            : Text(

                "LOGGED IN AS : ${user!.email}",

                style: TextStyle(fontSize: 20),

              ),

      ),

    );

  }

}

**Pages/login\_page.dart**

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/material.dart';

import 'package:modernlogintute/components/my\_button.dart';

import 'package:modernlogintute/components/my\_textfield.dart';

import 'package:modernlogintute/components/square\_tile.dart';

class LoginPage extends StatefulWidget {

  LoginPage({super.key});

  @override

  State<LoginPage> createState() => \_LoginPageState();

}

class \_LoginPageState extends State<LoginPage> {

  // text editing controllers

  final emailController = TextEditingController();

  final passwordController = TextEditingController();

  // sign user in method

  void signUserIn() async {

    // Show loading circle

    showDialog(

      context: context,

      builder: (context) {

        return const Center(

          child: CircularProgressIndicator(),

        );

      },

    );

    // Try to sign in

    try {

      await FirebaseAuth.instance.signInWithEmailAndPassword(

        email: emailController.text.trim(),

        password: passwordController.text.trim(),

      );

      // Pop the loading circle

      Navigator.pop(context);

    } on FirebaseAuthException catch (e) {

      // Pop the loading circle

      Navigator.pop(context);

      // Debugging: Print the error code

      print('Error code: ${e.code}');

      // Handle specific error codes

      if (e.code == 'invalid-email') {

        wrongEmailMessage();

      } else if (e.code == 'invalid-password') {

        wrongPasswordMessage();

      } else if (e.code == 'invalid-credential') {

        wrongPasswordMessage();

      }

    }

  }

  // Wrong email message popup

  void wrongEmailMessage() {

    showDialog(

      context: context,

      builder: (context) {

        return AlertDialog(

          title: const Text('Wrong Email'),

        );

      },

    );

  }

  // Wrong password message popup

  void wrongPasswordMessage() {

    showDialog(

      context: context,

      builder: (context) {

        return AlertDialog(

          title: const Text('Wrong Password'),

        );

      },

    );

  }

  // Invalid credential message popup

  void invalidCredentialMessage() {

    showDialog(

      context: context,

      builder: (context) {

        return AlertDialog(

          title: const Text('Invalid Credential'),

        );

      },

    );

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      backgroundColor: Colors.grey[300],

      body: SafeArea(

        child: Center(

          child: Column(

            mainAxisAlignment: MainAxisAlignment.center,

            children: [

              const SizedBox(height: 50),

              const Icon(

                Icons.lock,

                size: 100,

              ),

              const SizedBox(height: 50),

              Text(

                'Welcome back you\'ve been missed!',

                style: TextStyle(

                  color: Colors.grey[700],

                  fontSize: 16,

                ),

              ),

              const SizedBox(height: 25),

              MyTextField(

                controller: emailController,

                hintText: 'Email',

                obscureText: false,

              ),

              const SizedBox(height: 10),

              MyTextField(

                controller: passwordController,

                hintText: 'Password',

                obscureText: true,

              ),

              const SizedBox(height: 10),

              Padding(

                padding: const EdgeInsets.symmetric(horizontal: 25.0),

                child: Row(

                  mainAxisAlignment: MainAxisAlignment.end,

                  children: [

                    Text(

                      'Forgot Password?',

                      style: TextStyle(color: Colors.grey[600]),

                    ),

                  ],

                ),

              ),

              const SizedBox(height: 25),

              MyButton(

                onTap: signUserIn,

              ),

              const SizedBox(height: 50),

              Padding(

                padding: const EdgeInsets.symmetric(horizontal: 25.0),

                child: Row(

                  children: [

                    Expanded(

                      child: Divider(

                        thickness: 0.5,

                        color: Colors.grey[400],

                      ),

                    ),

                    Padding(

                      padding: const EdgeInsets.symmetric(horizontal: 10.0),

                      child: Text(

                        'Or continue with',

                        style: TextStyle(color: Colors.grey[700]),

                      ),

                    ),

                    Expanded(

                      child: Divider(

                        thickness: 0.5,

                        color: Colors.grey[400],

                      ),

                    ),

                  ],

                ),

              ),

              const SizedBox(height: 50),

              Row(

                mainAxisAlignment: MainAxisAlignment.center,

                children: const [

                  SquareTile(imagePath: 'lib/images/google.png'),

                  SizedBox(width: 25),

                  SquareTile(imagePath: 'lib/images/apple.png')

                ],

              ),

              const SizedBox(height: 50),

              Row(

                mainAxisAlignment: MainAxisAlignment.center,

                children: [

                  Text(

                    'Not a member?',

                    style: TextStyle(color: Colors.grey[700]),

                  ),

                  const SizedBox(width: 4),

                  const Text(

                    'Register now',

                    style: TextStyle(

                      color: Colors.blue,

                      fontWeight: FontWeight.bold,

                    ),

                  ),

                ],

              )

            ],

          ),

        ),

      ),

    );

  }

}

**Page/auth\_page.dart**

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/material.dart';

import 'package:modernlogintute/pages/home\_page.dart';

import 'package:modernlogintute/pages/login\_page.dart';

class AuthPage extends StatelessWidget {

  const AuthPage({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: StreamBuilder<User?>(

        stream: FirebaseAuth.instance.authStateChanges(),

        builder: (context, snapshot) {

          //User is signed in

          if(snapshot.hasData){

            return HomePage();

          }

          //user is not signed in

          else{

            return LoginPage();

          }

        },

        ),

    );

  }

}

**Firebase\_options.dart**

import 'package:firebase\_core/firebase\_core.dart' show FirebaseOptions;

class DefaultFirebaseOptions {

  static const FirebaseOptions currentPlatform = FirebaseOptions(

    apiKey: 'AIzaSyBJKRhuRliq5QeKaBy\_Le9otE2jIcqUFkc',

    appId: '1:247668320967:web:96692d847bf5052b7b2ce2',

    messagingSenderId: '247668320967',

    projectId: 'smit-77923',

    authDomain: 'smit-77923.firebaseapp.com',

    storageBucket: 'smit-77923.firebasestorage.app',

    measurementId: 'G-DHHRCD7R51',

  );

}

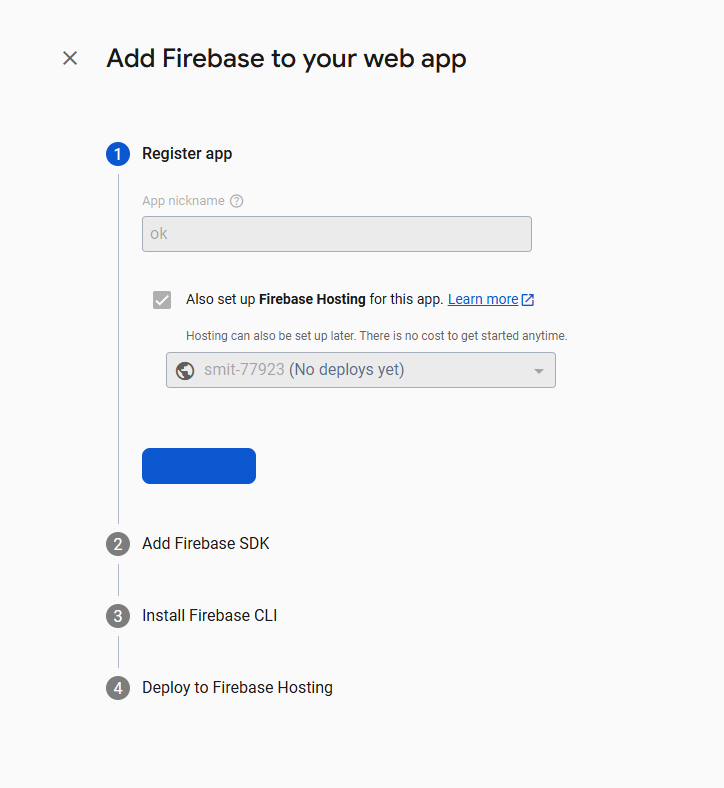
**OUTPUT:** ****

Figure 1 : creating the firebase project

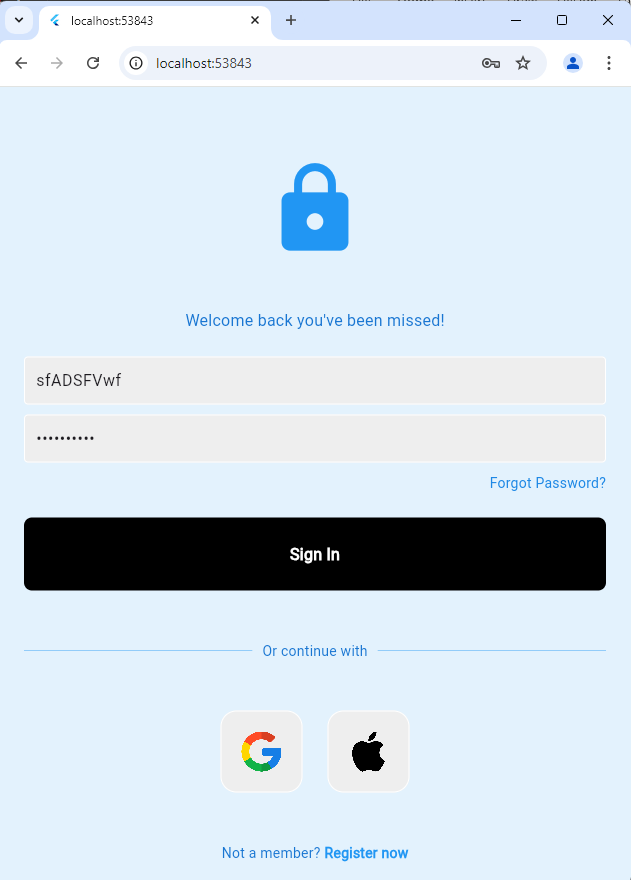


Figure 2:Home Page

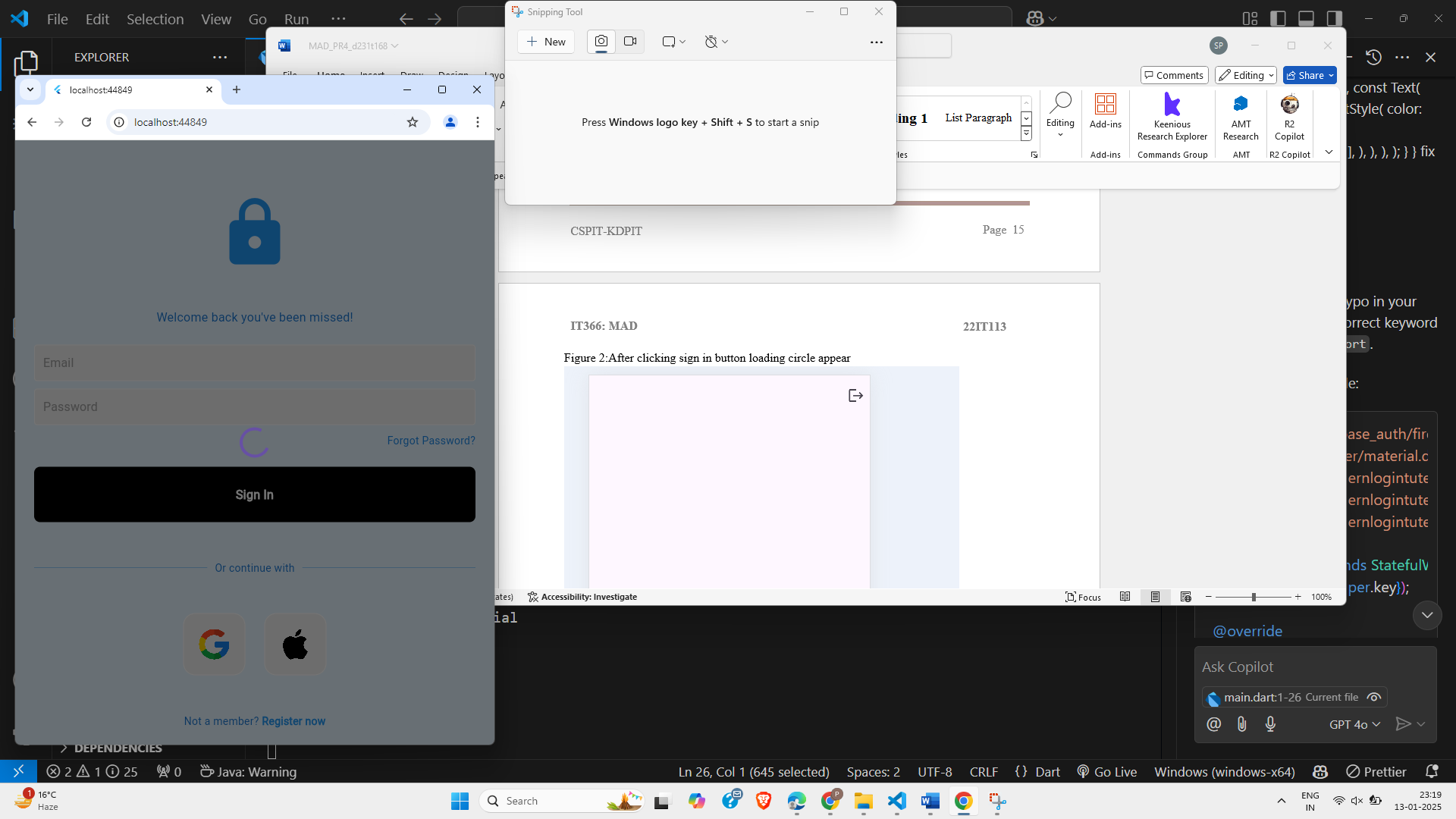


Figure 3:After clicking sign in button loading circle appear

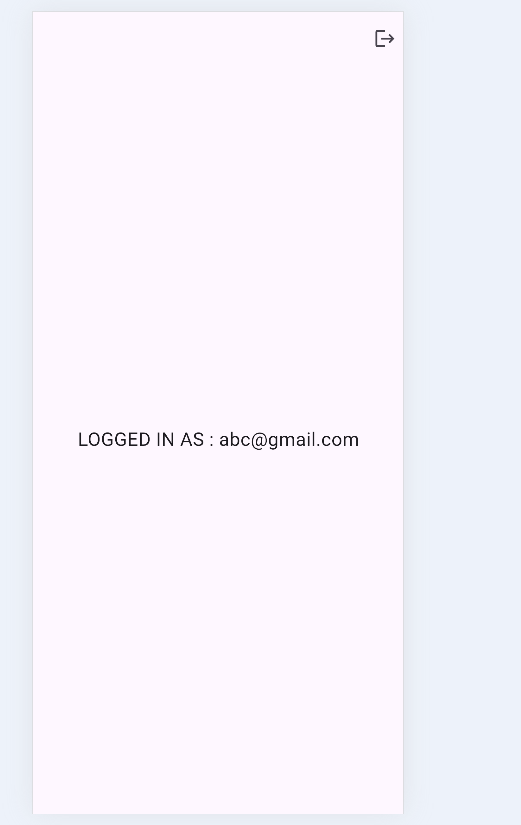


Figure 4: After login there is sign out options right upper side

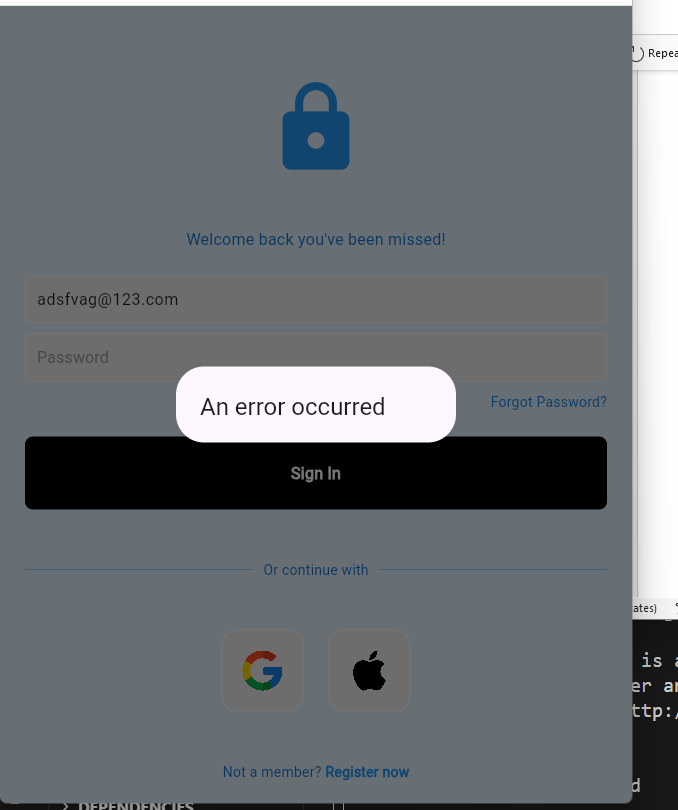


Figure 5:If we enter wrong password

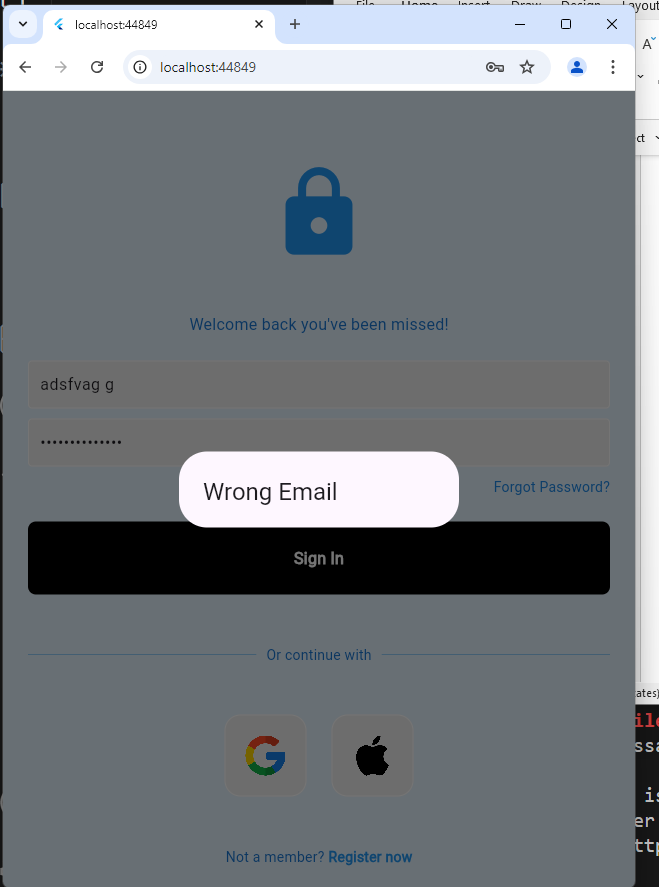


Figure 6:If we enter wrong email

# Latest Applications:

1. **User Authentication:**
   * Email/password-based login and registration.
   * Integration with Firebase Authentication for secure and efficient session management.
2. **UI/UX Design:**
   * Clean and responsive login screen.
   * TextField widgets for input, ElevatedButton for actions, and meaningful error messages for better feedback.
3. **User Input Validation:**
   * Ensures fields are non-empty.
   * Validates email format before proceeding.
4. **State Management:**
   * Dynamically updates the UI based on user actions using StatefulWidget.
5. **Navigation:**
   * Seamless transition to the home screen upon successful authentication.
   * Clear and concise error feedback for failed authentication attempts.
6. **Error Handling:**
   * Captures Firebase errors such as invalid credentials or duplicate registrations.
   * Displays feedback messages to enhance user experience.
7. **Extensibility:**
   * Prepared for future integration of advanced features like:
     + Social login options (Google, Facebook, etc.).
     + Password recovery.
     + Profile management.
8. **Security:**
   * Implements Firebase Authentication to ensure secure handling of sensitive user data.

**Learning Outcomes**

1. **User Input & Validation:**
   * Learned to collect and validate user inputs for login and registration using Flutter's TextField widget.
   * Implemented checks for empty fields and incorrect email formatting.
2. **Firebase Authentication:**
   * Mastered integration of Firebase Authentication for secure user management.
   * Understood session handling and credential verification.
3. **Error Handling:**
   * Gained experience in using try-catch blocks to handle errors.
   * Displayed appropriate messages to users, improving usability.
4. **UI/UX Design:**
   * Designed intuitive screens for login and registration using Flutter widgets.
   * Ensured responsiveness and clean design principles for a pleasant user experience.
5. **State Management:**
   * Learned to use StatefulWidget to update UI dynamically based on application states.
6. **Navigation:**
   * Built basic navigation flows between screens.
   * Ensured smooth transitions with contextual feedback (e.g., success or error messages).
7. **Security Considerations:**
   * Understood and implemented best practices for securely managing user credentials using Firebase's robust authentication system.
8. **Extensibility:**
   * Designed the application to be modular, facilitating the integration of additional features like Google sign-in or profile editing.