

EXPERIMENT NO: - 06

Name:- Mahi Jodhani

Class:- D15A

Roll:No: - 21

AIM:- To connect Flutter UI with Firebase database.

Theory: -

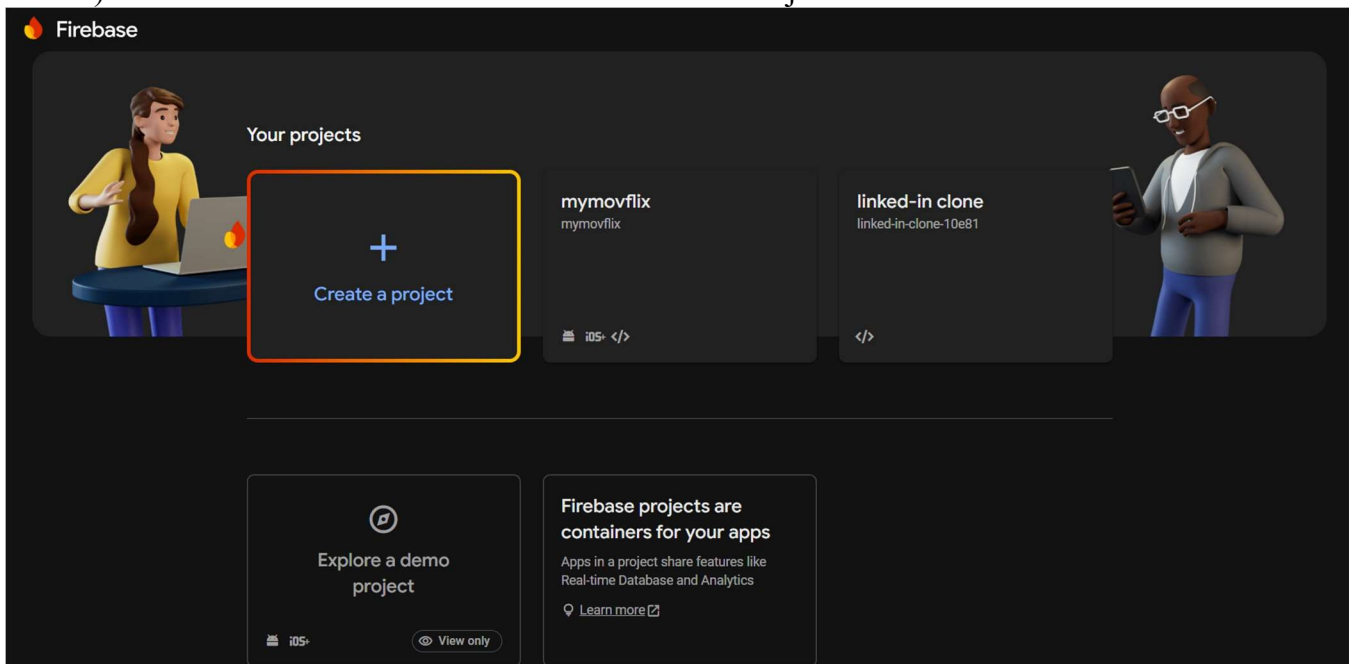
Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. Firebase, a Backend-as-a-Service (BaaS) platform, provides real-time database, authentication, and cloud storage services, making it a powerful backend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

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

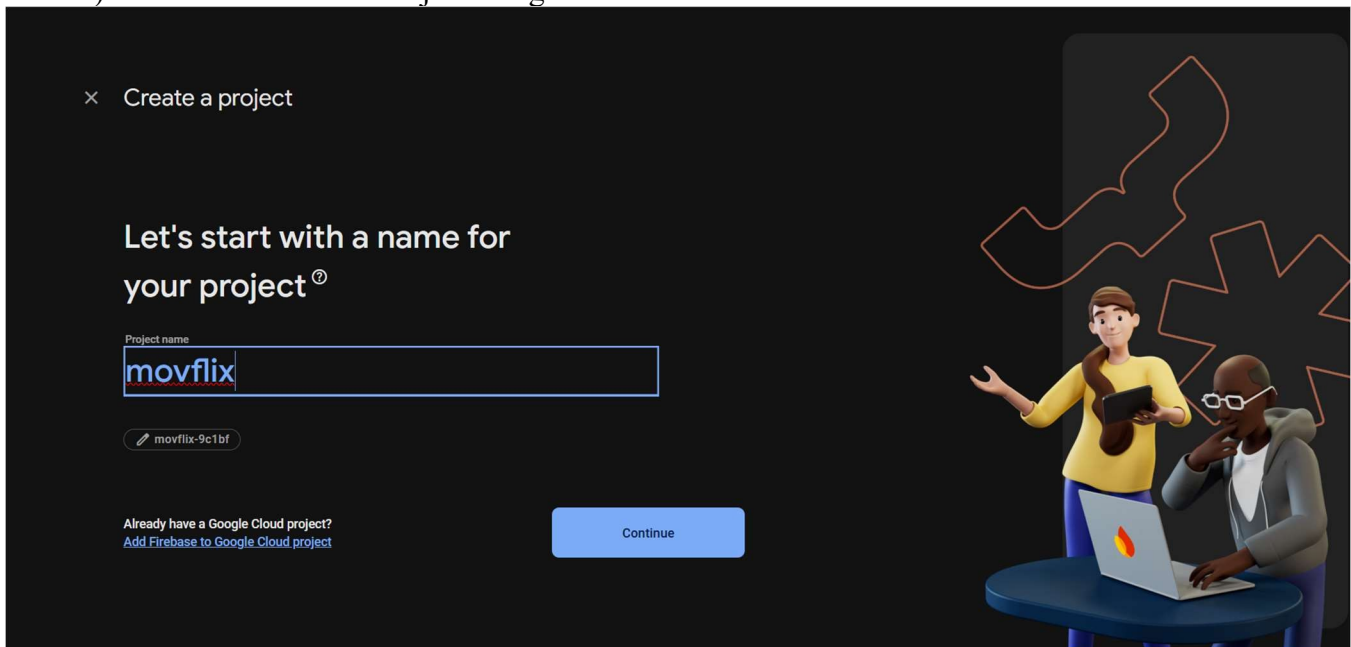
Steps to Connect Flutter UI with Firebase Database

Step 1:

1.1) Go to Firebase Console and Create a Firebase Project

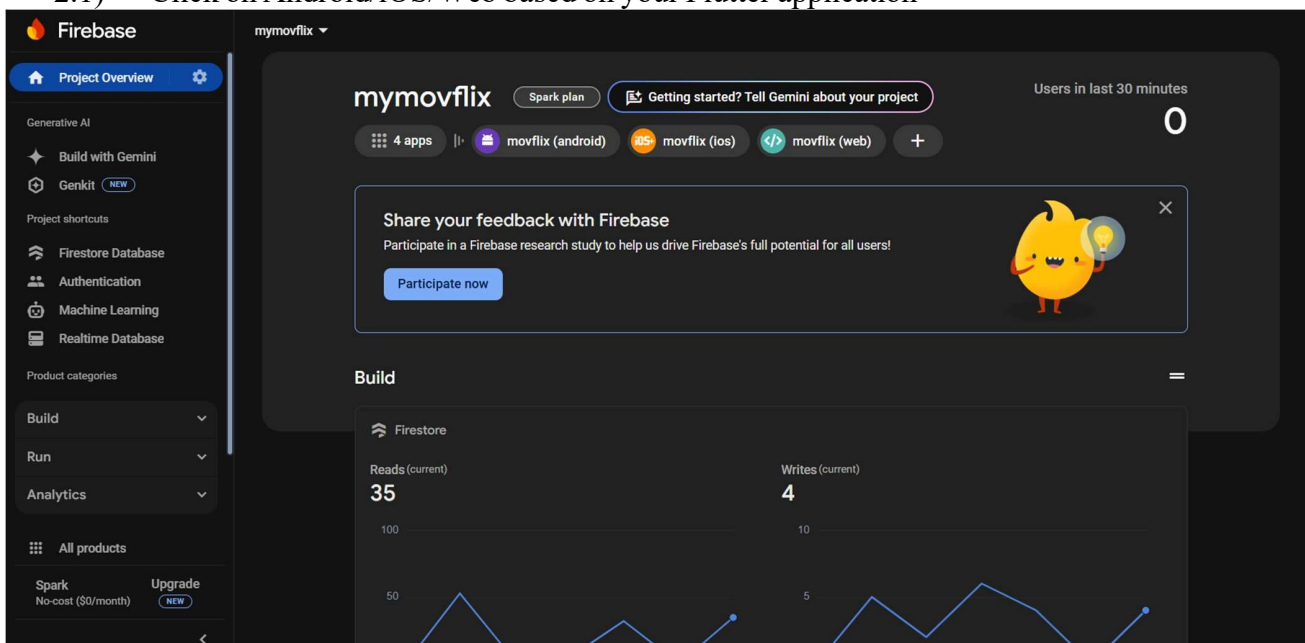


1.2) Click on Create a Project and give it a suitable name.



Step 2:- Add Firebase to Your Flutter App

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



Step 3: - Add Firebase Authentication to Your App

3.1) Add Firebase Authentication Dependencies

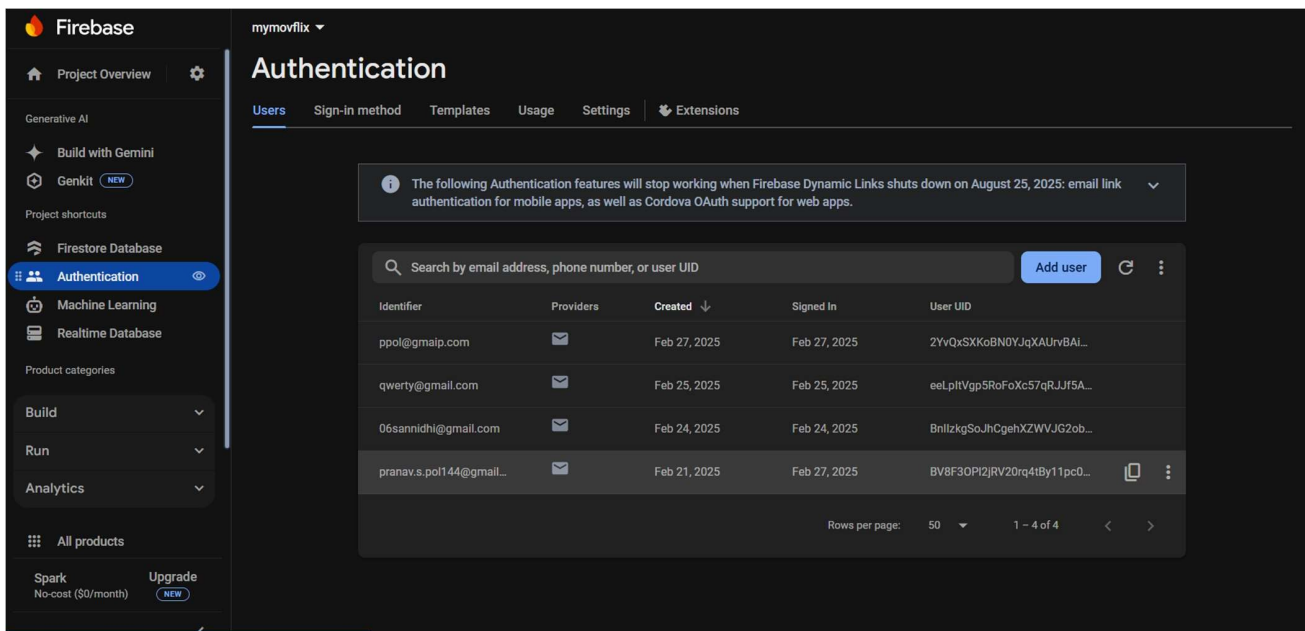
```
dependencies:  
  flutter:  
    sdk: flutter  
  firebase_core: ^3.11.0  
  firebase_auth: ^5.4.2 # For authentication  
  cloud_firestore: ^5.6.3 # For Firestore, if you need it  
  firebase_messaging: ^15.2.2  
  http: ^0.13.3  
  image_picker: ^1.0.4  
  tflite_flutter: ^0.11.0  
  image: ^3.2.0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

3.2) Enable Authentication in Firebase Console

Go to **Firebase Console** → **Authentication**.

Click on **Sign-in method** and enable **Email/Password** (or any other method like Google).

Click Save



The screenshot shows the Firebase Authentication console for a project named 'mymovflix'. The left sidebar contains navigation options like Project Overview, Generative AI, Build with Gemini, Genkit, Project shortcuts, Firestore Database, Authentication (selected), Machine Learning, Realtime Database, Product categories, Build, Run, Analytics, All products, Spark, and Upgrade. The main content area is titled 'Authentication' and has tabs for Users, Sign-in method, Templates, Usage, Settings, and Extensions. A warning message at the top states: '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.' Below this is a search bar with the text 'Search by email address, phone number, or user UID' and an 'Add user' button. A table lists four users with columns for Identifier, Providers, Created, Signed In, and User UID. The table data is as follows:

Identifier	Providers	Created	Signed In	User UID
ppol@gmail.com	📧	Feb 27, 2025	Feb 27, 2025	2YvQxSXKoBN0YJqXAUrvBAI...
qwerty@gmail.com	📧	Feb 25, 2025	Feb 25, 2025	eeLpltVgp5RoFoXc57qRJf5A...
06sannidhi@gmail.com	📧	Feb 24, 2025	Feb 24, 2025	BnltzkgSoJhCgehXZWWJG2ob...
pranav.s.pot144@gmail...	📧	Feb 21, 2025	Feb 27, 2025	BV8F30Pi2jRV20rq4tBy11pc0...

At the bottom right of the table, it shows 'Rows per page: 50' and '1 - 4 of 4'.

3.3) Implement
Authentication in
Flutter Modify
main.dart

```
import  
'package:firebase_core/firebase_core.dar  
t'; import  
'package:firebase_auth/firebase_auth.dar  
t';  
  
void main() async  
{ WidgetsFlutterBinding.ensure  
  Initialized(); await  
  Firebase.initializeApp();  
  runApp(MyApp());  
}
```

Step 4: -Configure Firebase Realtime Database

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

Code:-

Sign_in_page.dart

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

import 'sign_up_page.dart';
import
'package:movflix/screens/homescreen.dart';
import
'package:movflix/widgets/bottom_bar_nav.dart';

class SignInPage extends StatefulWidget
{
  @override
  _SignInPageState createState() =>
  _SignInPageState();
}

class _SignInPageState extends
State<SignInPage> {
  final FirebaseAuth _auth =
FirebaseAuth.instance;
  final FirebaseFirestore _firestore =
FirebaseFirestore.instance;
  final TextEditingController _emailController =
TextEditingController();
  final TextEditingController
_passwordController = TextEditingController();
  String _errorMessage = "";
  bool _isLoading = false;

  Future<void> _signIn() async
  {
    setState(() {
      _isLoading = true;
      _errorMessage = "";
    });

    try {
      // Firebase Authentication
      UserCredential userCredential = await
_auth.signInWithEmailAndPassword(
```

```
email: _emailController.text.trim(),
password: _passwordController.text.trim(),
);

User? user = userCredential.user;
if (user != null) {
  // Ensure user document exists before
updating
  var userDoc = await
_firestore.collection("users").doc(user.uid).get()
;
  if (userDoc.exists)
  {
    await
_firestore.collection("users").doc(user.uid).update(
{
  "lastLogin":
FieldValue.serverTimestamp(),
});
  }

  Navigator.pushAndRemoveUntil( context,
  MaterialPageRoute(builder: (context) =>
HomeScreen()),
  (route) => false,
);
  Navigator.of(context).pushReplacement( M
aterialPageRoute(builder: (context) =>
const BottomNavBar()),
);
}
} on FirebaseAuthException catch (e)
{
  setState(() {
    _errorMessage = e.message ?? "Error
signing in";
    _isLoading = false;
  });
} finally
{
  setState(()
  {
    _isLoading = false;
  });
}
```

mainAxisAlignment:

}

@override

Widget build(BuildContext context)

{ return Scaffold(

body: Container(

padding: EdgeInsets.symmetric(horizontal:

15, vertical: 15),

child:

Column(chil

dren: [

_headerWidget(),

SizedBox(heigh

t: 10,

),

_formWidget(),

],

),

),

);

}

Widget _headerWidget()

{ return Row(

children:

[InkWell(onT

ap: () {

Navigator.pop(context);

},

child: Icon(Icons.arrow_back),

),

SizedBox(wid

th: 10,

),

Container(hei

ght: 40,

child: Image.asset('assets/logo.png'),

)

],

);

}

Widget _formWidget()

{ return

Expanded(child:

Column(

```

MainAxisAlignment.center, children: [
  Container(
    padding:
      EdgeInsets.symmetric(horizontal: 8),
    decoration:
      BoxDecoration( color:
        Colors.grey[800],
        borderRadius:
          BorderRadius.all(
            Radius.circular(5),
          ),
        ),
    child:
      TextFormField( controller:
        _emailController,
        decoration:
          InputDecoration(
            labelStyle: TextStyle(fontSize:
14, color: Colors.white),
            border: InputBorder.none,
            labelText: "Email or phone number",
          ),
        ),
      SizedBox( height: 10,
        ),
      Container(
        padding:
          EdgeInsets.symmetric(horizontal: 8),
        decoration:
          BoxDecoration( color:
            Colors.grey[800],
            borderRadius:
              BorderRadius.all(
                Radius.circular(5),
              ),
            ),
        child: TextFormField(
          controller:
            _passwordController,
          obscureText: true,

```

```

        decoration:
          InputDecoration( labelStyle:
            TextStyle(fontSize: 14,
              color: Colors.white),
            border: InputBorder.none,
            labelText: "Password",
          ),
        ),
      ),
      SizedBox( height:
        t: 15,
        ),
    ),

```

```

InkWell(
  onTap: _isLoading ? null : _signIn,
  child: Container(
    alignment: Alignment.center,
    padding:
EdgeInsets.symmetric(vertical: 15),
    width: double.maxFinite,
    decoration: BoxDecoration(
      color: _isLoading ? Colors.grey :
Colors.transparent,
      border: Border.all(
        color: Colors.grey[600] ??
Colors.grey, width: 2),
    ),
    child: _isLoading
      ? CircularProgressIndicator(color:
Colors.white)
      : Text("Sign In"),
    ),
  ),
  if(_errorMessage.isNotEmpty)
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: Text(
        _errorMessage,
        style: TextStyle(color: Colors.red),
      ),
    ),
  SizedBox( height
    t: 15,
  ),
  Text("Need Help?"),
  SizedBox(

```

```

    height: 15,
  ),
  GestureDetector( onTap: ()
    { Navigator.push(
      context,
      MaterialPageRoute(builder: (context)
=> SignUpPage()),
    );
  },
  child: Text(
    "New to Netflix? Sign up now.",
    style: TextStyle(fontWeight:
FontWeight.bold, color: Colors.blue),
  ),
  ),
  SizedBox( height
    t: 20,
  ),
  Text(
    "Sign-in is protected by Google
reCAPTCHA to ensure you're not a bot. Learn
more.",
    style:
      TextStyle( font
        Size: 12,
      ),
    textAlign: TextAlign.center,
  )
],
),
);
}
}

```


Sign_up_page.dart

```
import 'package:flutter/material.dart';
import 'package:flutter/rendering.dart';
import
'package:movflix/widgets/header_widget.dart';
import 'sign_in_page.dart';
import
'package:movflix/screens/homescreen.dart';
import
'package:movflix/widgets/bottom_bar_nav.dart';
import
'package:firebase_auth/firebase_auth.dart';
import
'package:cloud_firestore/cloud_firestore.dart';
```

```
class SignUpPage extends StatefulWidget
{
  @override
  _SignUpPageState createState() =>
  _SignUpPageState();
}
```

```
class _SignUpPageState extends
State<SignUpPage> {
  final FirebaseAuth _auth =
FirebaseAuth.instance;
  final FirebaseFirestore _firestore =
FirebaseFirestore.instance;
  final TextEditingController
_emailController = TextEditingController();
  final TextEditingController
_passwordController =
TextEditingController();
  bool _isChecked = false;
  String _errorMessage = "";
  bool _isLoading = false;
```

```
Future<void> _signUp() async
{
  setState(() {
    _isLoading = true;
    _errorMessage = "";
  });

  try {
    // Check if the user already exists in
    Firestore
```

```
    var existingUser = await _firestore
.collection('users')
    .where('email', isEqualTo:
_emailController.text.trim())
    .get();

    if (existingUser.docs.isNotEmpty)
    {
      setState(() {
        _errorMessage = "User already exists.
Please log in.";
        _isLoading = false;
      });
      return;
    }

    // Create user in Firebase Authentication
    UserCredential userCredential = await
_auth.createUserWithEmailAndPassword( email
: _emailController.text.trim(), password:
_passwordController.text.trim(),
);

    // Store user details in Firestore
    await
_firestore.collection('users').doc(userCredential.
user!.uid).set({
      'email': _emailController.text.trim(),
      'password': _passwordController.text.trim(),
      'createdAt': FieldValue.serverTimestamp(),
    });

    // Navigate to Home Page
    Navigator.pushAndRemoveUntil( context
t,
    MaterialPageRoute(builder: (context) =>
HomeScreen()),
    (route) => false,
  );

    Navigator.of(context).pushReplacement( Materia
lPageRoute(builder: (context) =>
const BottomNavBar()),
  );

  } on FirebaseAuthException catch (e)
  {
    setState(() {
      _errorMessage = e.message ?? "Error
signing up";
    });
```

```

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

  @override
  Widget build(BuildContext context)
  {
    return Scaffold(
      body:
        Container(
          margin:
            EdgeInsets.symmetric(horizontal: 15,
              vertical: 10),
          child:
            Column(
              children:
                [
                  HeaderWidget(),
                  _formWidget(),
                ],
            ),
        ),
    );
  }

```

```

Widget _formWidget()
{
  return
    Expanded(
      child:
        Column(
          crossAxisAlignment:
            CrossAxisAlignment.start,
          children:
            [
              SizedBox(
                height: 19,
                child: Text(
                  "Sign up to start your\nmembership.",
                  style: TextStyle(
                    fontSize: 18,
                    fontWeight: FontWeight.bold,
                    color: Colors.white),
                ),
              SizedBox(
                height: 10,
                child: Text(
                  "Create your account",
                  style: TextStyle(
                    fontSize: 18,
                    fontWeight: FontWeight.bold),
                ),
              SizedBox(
                height: 10,
                child: Container(
                  padding:

```

```

          color: Colors.grey[800],
          borderRadius:
            BorderRadius.all(
              Radius.circular(5)),
        ),
        child: TextFormField(
          controller: _emailController,
          decoration: InputDecoration(
            edgeInsets: EdgeInsets.symmetric(
              horizontal: 8),
            decoration: BoxDecoration(

```

```

        labelText:
        "Email",
        border:
        InputBorder.
        none,
        labelStyle:
        TextStyle(fontSize: 14, color:
Colors.white),
      ),
    ),
    ),
    SizedBo
x(height:
10),
    Containe
r(
padding:
EdgeInsets.symmetric(h
orizontal: 8),
decoration:
BoxDecorati
on( color:
Colors.grey[8
00],
borderRadius:
BorderRadius.all(Radius.circular(5)),
),
    child: TextFormField(
controller:
_passwordControll
er, obscureText:
true,
decoration:
InputDecorati
on( labelText:
"Password",
border:
InputBorder.n
one,
labelStyle:
TextStyle(fontSize: 14, color:
Colors.white),
    ),
  ),
),
SizedBo
x(height:
15),
Row(
c
h

```

```

ildren:
[ Checkbo
x(
value: _isChecked,
onChanged: (value)
{ setState(() {
_isCheck = value ?? false;
});
},
),
Text("Please do not email me Netflix
special offers."),
],
),

```

```

    SizedBox(height: 15),
    InkWell(
      onTap: _signUp,
      child: Container(
        alignment: Alignment.center,
        padding:
EdgeInsets.symmetric(vertical: 15),
      width: double.maxFinite,
      decoration:
        BoxDecoration( color:
          Colors.transparent, border:
            Border.all(color:
Colors.grey[600] ?? Colors.grey, width: 2),
        ),
      child: _isLoading
        ? CircularProgressIndicator()
        : Text("Sign Up"),
    ),
  ),
  if(_errorMessage.isNotEmpty)
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: Text(_errorMessage, style:
TextStyle(color: Colors.red)),
    ),
    SizedBox(height: 15),
    GestureDetector( onTap: () {
      Navigator.push(context,
MaterialPageRoute(builder: (context) =>
SignInPage()));
    },
    child: Text(
      "Already have an account? Sign
in.",
      style: TextStyle(fontWeight:
FontWeight.bold, color: Colors.blue),
    ),
  ),
],
),
);
}
}

```

Main.dart

```

import 'package:flutter/material.dart';
import
'package:movflix/widgets/custom_theme.dart';

import
'package:firebase_core/firebase_core.dart';
import 'firebase_options.dart';
import 'screens/splash_screen.dart';

void main()
  async { WidgetsFlutterBinding.ensureInitialize
d(); await Firebase.initializeApp(
    options:
DefaultFirebaseOptions.currentPlatform,
  );
  runApp(const MyApp());
}

class MyApp extends StatelessWidget
{ const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context)
  { return MaterialApp(
    title: 'MovFlix ',
    theme: darkTheme,
    debugShowCheckedModeBanner: false,
    home: SplashScreen(),
  );
  }
}

```

Project Overview

Generative AI

Build with Gemini

Genkit

Project shortcuts

Firestore Database

Authentication

Machine Learning

Realtime Database

Product categories

Build

Run

Analytics

All products

Spark

No-cost (\$0/month)

Upgrade

NEW

mymovflix

Cloud Firestore

Add database

Ask Gemini how to get started with Firestore

Data

Rules

Indexes

Disaster Recovery

Usage

Extensions

Protect your Cloud Firestore resources from abuse, such as billing fraud or phishing

Configure App Check

Panel view

Query builder

More in Google Cloud

(default)

users

2YvQxSXKoBN0YJqXAUrvBAio7x93

+ Start collection

movie_recommendations

+ Add document

2YvQxSXKoBN0YJqXAUrvBAio7x93

+ Start collection

users

BV8F30P12jRV20rq4tBy11pc8nH3

+ Add field

Bn1IzkgSoJhCgehXZWWJG2objse3

createdAt: February 27, 2025 at 1:01:03 AM UTC+5:30

email: "ppol@gmail.com"

lastLogin: February 27, 2025 at 11:20:47 AM UTC+5:30

password: "Pranav"

eeLpItVgp5RoFoXc57qRJf5AQE2

Project Overview

Generative AI

Build with Gemini

Genkit

Project shortcuts

Firestore Database

Authentication

Machine Learning

Realtime Database

Product categories

Build

Run

Analytics

All products

Spark

No-cost (\$0/month)

Upgrade

NEW

mymovflix

Realtime Database

Need help with Realtime Database? Ask Gemini

Data

Rules

Backups

Usage

Extensions

Protect your Realtime Database resources from abuse, such as billing fraud or phishing

Configure App Check

https://mymovflix-default-rtdb.firebaseio.com > messages > -0Jt1857_x6uhF... > -0Jt19VINQPNQ

Your security rules are not secure. Any authenticated user can steal, modify, or delete data in your database.

Learn more

Dismiss

-0Jt19VINQPNQ180cLcC

message: "hi"

sender: "pranav.s.pol144@gmail.com"

timestamp: "2025-02-24T23:07:03.537993"

Database location: United States (us-central1)

