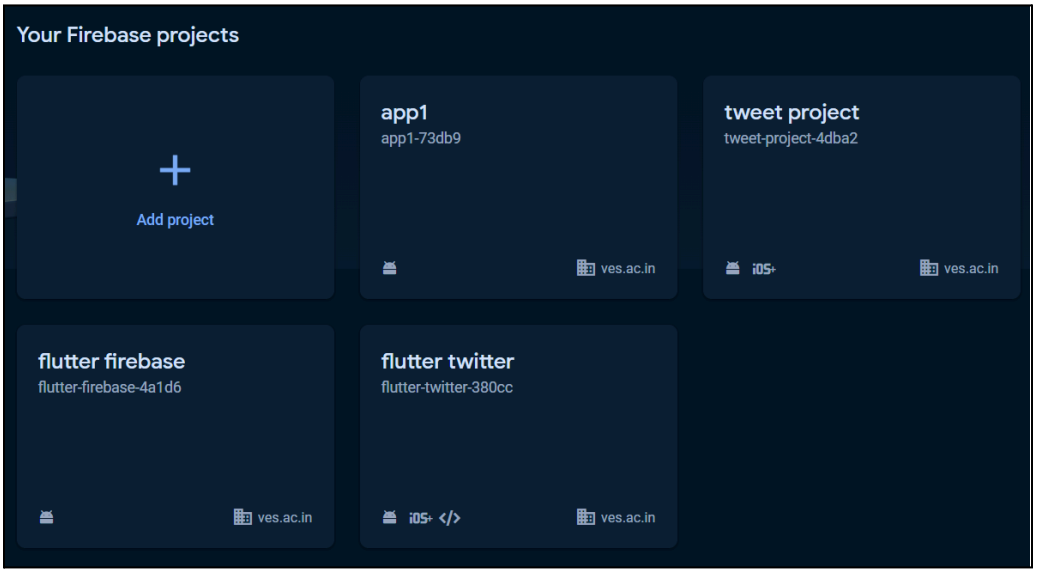
Shravani Pore
D15A_45
Batch B

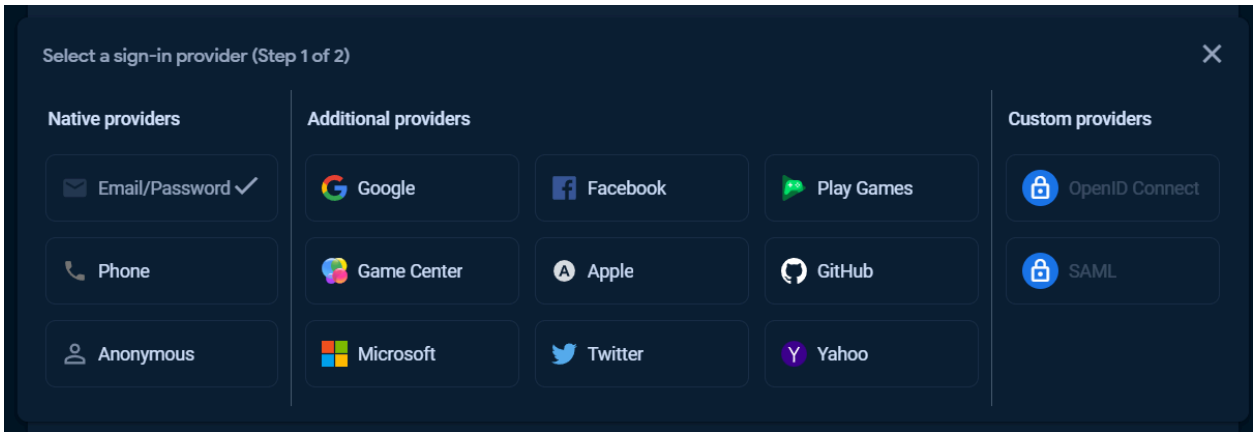
MPL EXPT 6

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

THEORY / STEPS:
(Wherever testing mode is available, I have opted for testing mode.)
1.Create a project in firebase console.



2.Go to authentication section inside the project we have created and enable email/password verification under sign-in method.



3. Similarly enable firestore database.

(Further part is done using Flutter CLI i.e Flutter command line interface)
Following are the commands for the same

4.Installing firebase CLI

npm install -g firebase-tools

5. We have to connect our project with the google account we have made out firebase project on.

firebase login

It would open a browser where we can choose our account.

6. Activation of flutter fire cli

Dart pub global activate flutterfire_cli

7. Configuration of flutterfire

Flutterfire configure

This shows us our firebase projects and we can select our project that we created for this application. I.e app1

8. It would then show us the platforms that we want to make our application compatible for.

9. Configured applications will be now available in firebase.

10. We check the flutter initialization in main function of out application.

```
future: Firebase.initializeApp(),
```

11. We add dependencies:

```
cupertino_icons: ^1.0.2
  firebase_core: ^2.25.4
  firebase_auth: ^4.17.4
  cloud_firestore: 4.15.5
  provider:
```

These should be compatible with the flutter SDK version we are using.
If we get an error with any of these we can try running:
Flutter pub upgrade
Flutter pub get

CODE:
Auth_gate.dart

```dart
// ignore: unused_import
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:twitter/main.dart';

class AuthGate extends StatelessWidget {
  static final GlobalKey<NavigatorState> navigatorKey = GlobalKey();
  const AuthGate({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        body: StreamBuilder(
      stream: FirebaseAuth.instance.authStateChanges(),
      builder: (context, snapshot) {
        //user is logged in
        if (snapshot.hasData) {
          return ProfilePage();
        }
        //user is not logged in
        else {
          return LoginPage();
        }
      },
    ));
  }
}
```

Auth_service.dart

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

class AuthService extends ChangeNotifier {
  final FirebaseAuth _firebaseAuth = FirebaseAuth.instance;

  //sign user in
  Future<UserCredential> signInWithEmailandPassword(
      String email, String password) async {
    try {
```

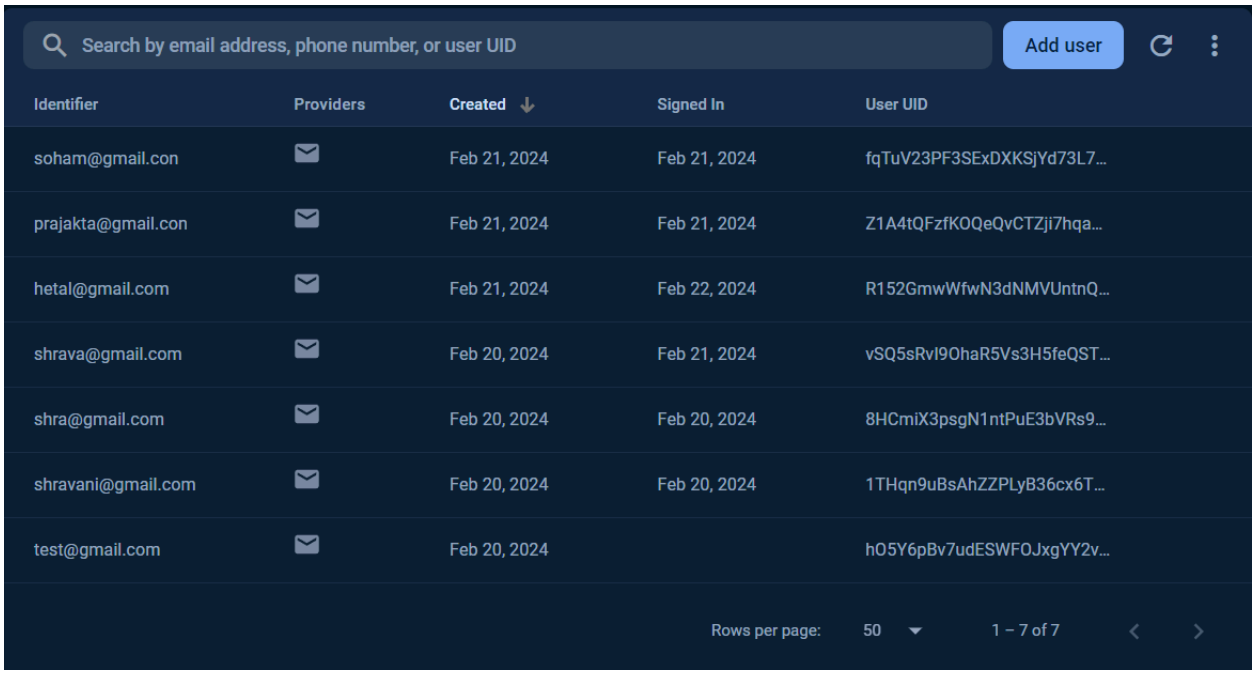
```
      //signin
      UserCredential userCredential =
          await _firebaseAuth.signInWithEmailAndPassword(
        email: email,
        password: password,
      );
      return userCredential;
    } on FirebaseAuthException catch (e) {
      //catch errors
      throw Exception(e.code);
    }
  }

  Future<void> changePassword(String newPassword) async {
    try {
      // Get the currently logged-in user
      User? user = _firebaseAuth.currentUser;

      // Update the password
      await user!.updatePassword(newPassword);

      // Sign out the user after changing the password
      await signOut();

      // You may choose to navigate the user to the sign-in page or any
other page after password change
    } catch (e) {
      // Handle any errors that occur during password change
      print('Error changing password: $e');
      throw Exception(e);
    }
  }

  //create a new user
  Future<UserCredential> signUpWithEmailandPassword(
      String email, String password) async {
    try {
      //signin
      UserCredential userCredential =
          await _firebaseAuth.createUserWithEmailAndPassword(
```

```
        email: email,
        password: password,
      );
      return userCredential;
    } on FirebaseAuthException catch (e) {
      //catch errors
      throw Exception(e.code);
    }
  }

  //sign user out
  Future<void> signOut() async {
    return await FirebaseAuth.instance.signOut();
  }
}

//saving tweets
Future<void> saveTweet(String tweetText) async {
  try {
    String uid = FirebaseAuth.instance.currentUser!.uid;
    await FirebaseFirestore.instance
        .collection('users')
        .doc(uid)
        .collection('tweets')
        .add({
      'text': tweetText,
      'timestamp': DateTime.now(),
    });
    print('Tweet added successfully');
  } catch (e) {
    print('Error adding tweet: $e');
  }
}
```

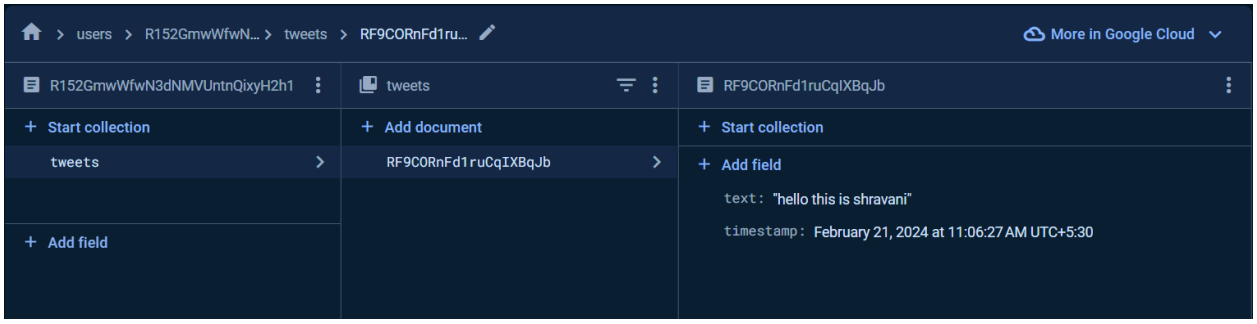
The functions defined here are called in the pages files. Also some code is added to main file itself and not in the Auth_service file.

OUTPUT:

Authentication:



Tweets in firestore database:



With this connection we are able to achieve the following:

- **Saving email and password in database.**
- **Saving tweets and their timestamps in the database under current user's uid.**
- **Fetching tweets from the database to display on the profile page.**
- **Updating the values in the database for password in the authentication database.**

CONCLUSION:

Connected firebase with twitter clone project and handled related errors.