Experiment No. 6

AIM : To connect flutter UI with firebase database

Theory:

Introduction to Firebase and Flutter Integration

Firebase is a comprehensive platform developed by Google, designed to help developers build high-quality applications for both mobile and web. It provides essential services such as real-time databases, authentication, cloud storage, hosting, and much more. One of the most widely used Firebase services is the Firebase Realtime Database, which is a NoSQL cloud database that allows data to be stored and synced in real-time across all connected devices. Flutter, on the other hand, is an open-source UI software development kit created by Google, which allows developers to build natively compiled applications for mobile, web, and desktop from a single codebase. Its rich set of pre-designed widgets and powerful tools makes Flutter an attractive option for developing visually appealing and performant applications. Integrating Firebase with Flutter allows developers to leverage the full potential of Firebase services in their applications. By using Firebase's Realtime Database, Flutter apps can achieve features such as real-time data synchronization, secure authentication, and cloud-based storage. This combination enables developers to create powerful, scalable, and feature-rich mobile and web applications.

Firebase Realtime Database Overview

Firebase Realtime Database is a cloud-hosted NoSQL database that stores data in a JSON-like format. The key characteristic of this database is its real-time synchronization feature, meaning that any changes made to the database are instantly reflected on all clients (i.e., devices) connected to it. This makes it an ideal solution for applications that require frequent updates and need to maintain synchronized data across multiple users or devices, such as messaging apps, social media platforms, or collaborative tools. The Firebase Realtime Database is structured as a tree of data, where each node in the tree can contain key value pairs. This structure allows for easy data retrieval and modification. Firebase's real-time capabilities enable apps to immediately receive updates to the data whenever it changes, without the need to refresh or reload the page. Additionally, the database supports offline data persistence, meaning that even if the user's device loses its internet connection, the app can still function by using the locally cached data.

Setting Up Firebase in Flutter :

To connect a Flutter app with Firebase, the following steps are typically followed:

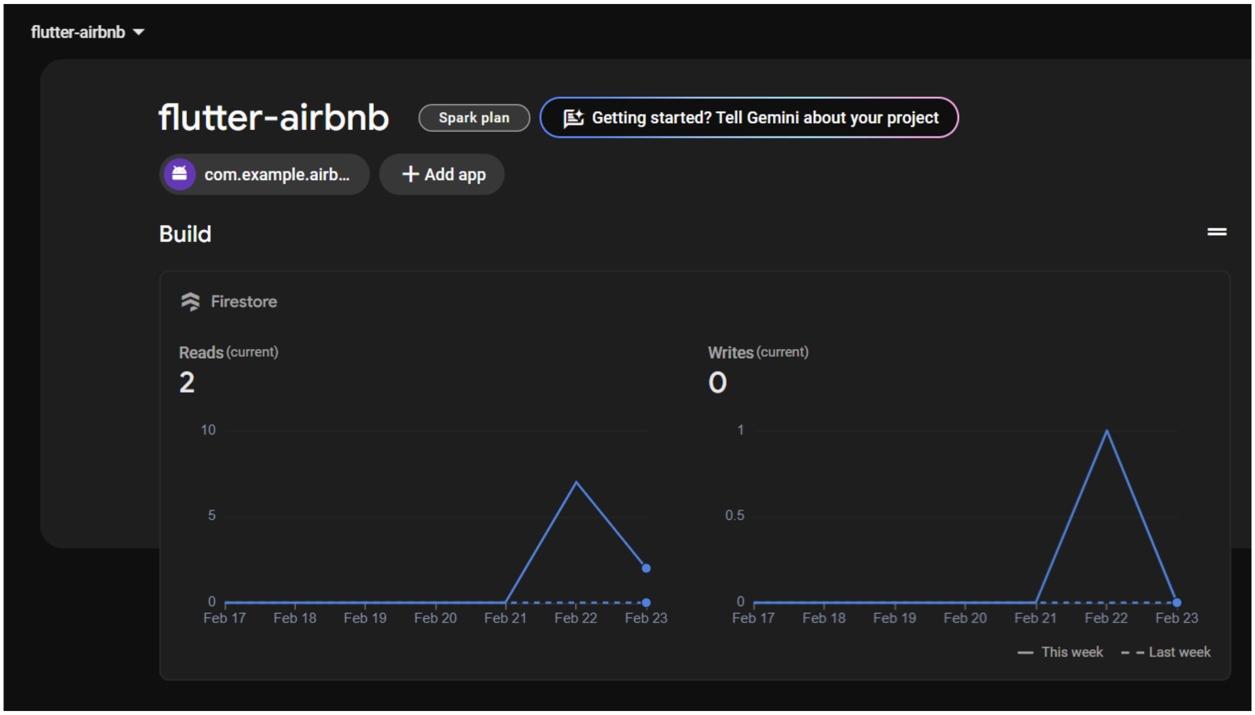
1. Creating a Firebase Project: To start using Firebase with Flutter, the first step is to create a Firebase project in the Firebase Console. Once the project is created, developers can associate their Flutter app with the Firebase project by following the platform-specific instructions for Android or iOS. This usually involves configuring API keys, downloading configuration files, and adding them to the Flutter project.

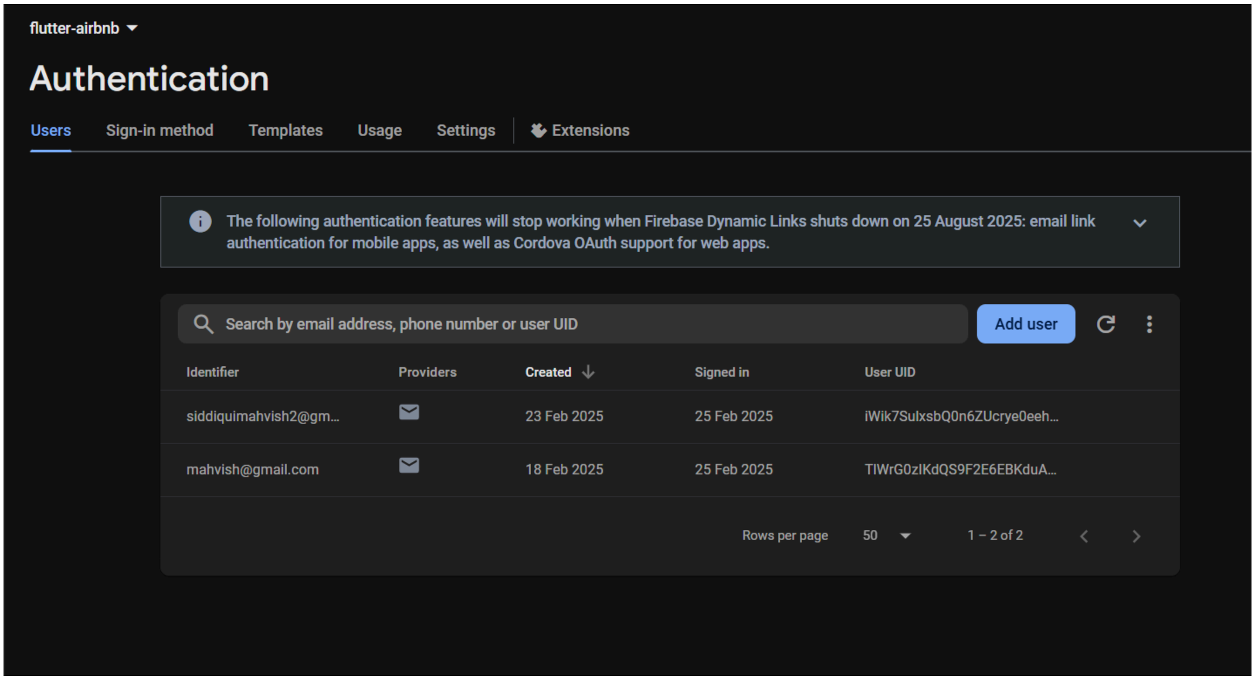
2. Integrating Firebase SDK in Flutter: After the Firebase project is set up, developers need to integrate Firebase's SDK into the Flutter app. This involves adding the necessary dependencies to the Flutter project's pubspec.yaml file. For Firebase's Realtime Database, the package firebase_database is used. Additionally, Firebase's core SDK (firebase_core) must also be included to initialize Firebase services.

3. Initializing Firebase: Before any Firebase functionality can be used, it is essential to initialize Firebase in the Flutter app. This is done by calling Firebase.initializeApp() in the main entry point of the app (usually in the main.dart file). Firebase needs to be initialized before interacting with any Firebase services, such as the Realtime Database, Cloud Firestore, or Authentication.

Connecting Firebase to a Flutter app enables developers to create robust, scalable, and real-time applications with ease. Firebase's Realtime Database offers a powerful, cloud-based solution for managing data in realtime, while Firebase Authentication ensures secure access control. By integrating Firebase with Flutter, developers can take advantage of real-time data synchronization, offline support, and a wide range of other Firebase features, allowing them to build feature-rich apps that meet modern user expectations.

Screenshots:

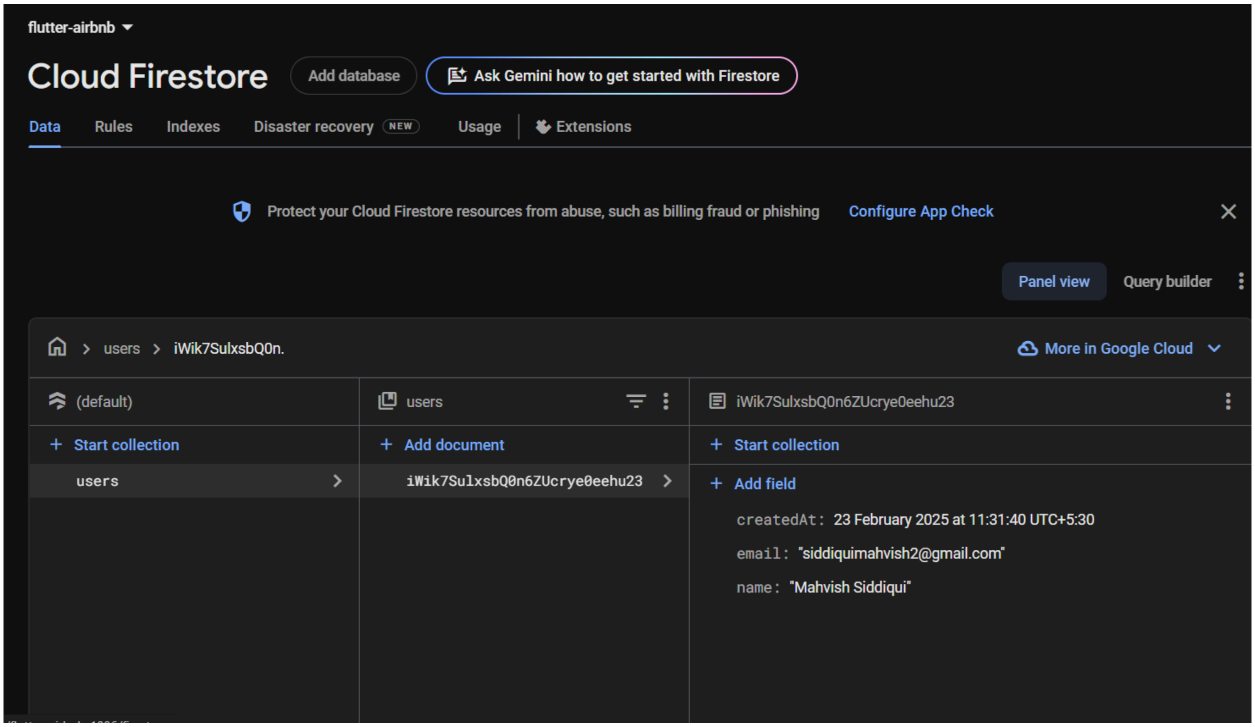




The following authentication features will stop working when Firebase Dynamic Links shuts down on 25 August 2025: email link authentication for mobile apps, as well as Cordova OAuth support for web apps.

Identifier	Providers	Created ↓	Signed in	User UID
siddiquimahvish2@gm...	✉	23 Feb 2025	25 Feb 2025	iWik7SulxsbQ0n6ZUcrye0eeh...
mahvish@gmail.com	✉	18 Feb 2025	25 Feb 2025	TIWrG0zIKdQS9F2E6EBKduA...

Rows per page 50 1 – 2 of 2

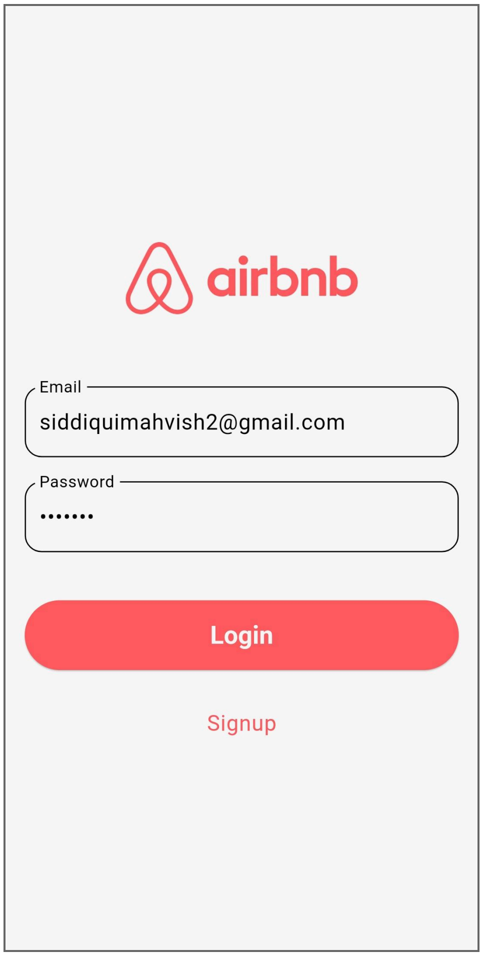


Protect your Cloud Firestore resources from abuse, such as billing fraud or phishing Configure App Check

createdAt: 23 February 2025 at 11:31:40 UTC+5:30

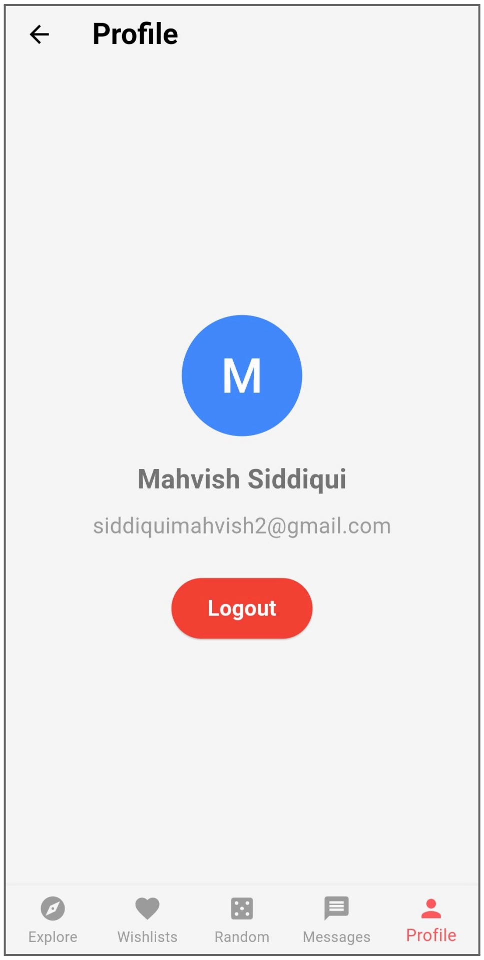
email: "siddiquimahvish2@gmail.com"

name: "Mahvish Siddiqui"



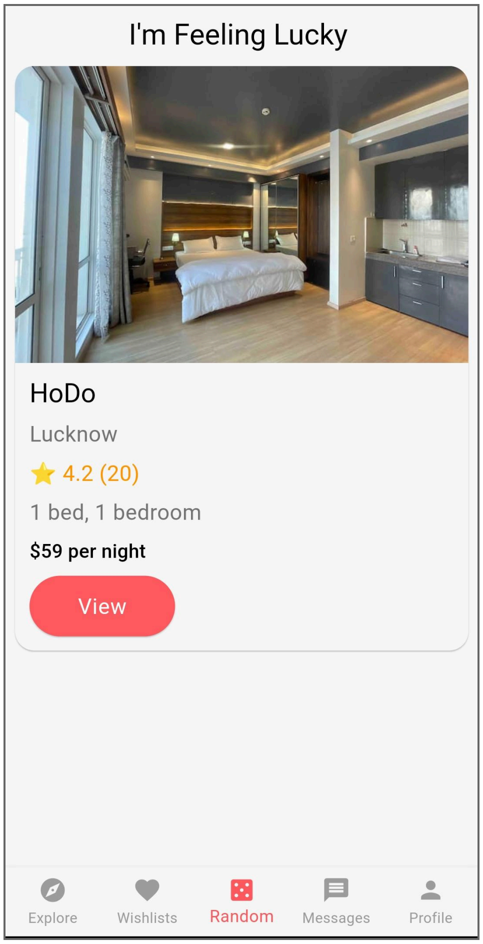
Email: siddiquimahvish2@gmail.com

Password: •••••••



Mahvish Siddiqui

siddiquimahvish2@gmail.com



I'm Feeling Lucky

HoDo

Lucknow

⭐ 4.2 (20)

1 bed, 1 bedroom

$59 per night

Code Snippets:

Setup in main.dart

```
import 'package:airbnb/firebase_options.dart';
import 'package:firebase_core/firebase_core.dart';

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

Login Function

```
Future<void> _handleLogin() async {
    String email = _emailController.text.trim();
    String password = _passwordController.text.trim();

    try {
      await FirebaseAuth.instance.signInWithEmailAndPassword(
        email: email,
        password: password,
      );

      Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (context) => BottomNavBar()),
    );
    } catch (e) {

      setState(() {
        _errorText = "Invalid email or password. Please try again.";
      });
    }
  }
```

Signup Function

```
Future<void> _handleLogin() async {
    String email = _emailController.text.trim();
    String password = _passwordController.text.trim();

    try {
      await FirebaseAuth.instance.signInWithEmailAndPassword(
        email: email,
        password: password,
      );

      Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (context) => BottomNavBar()),
    );
    } catch (e) {

      setState(() {
        _errorText = "Invalid email or password. Please try again.";
      });
    }
  }
```

Profile Page - Logout

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

class ProfileScreen extends StatelessWidget {
  Future<String?> _fetchUserName(String uid) async {
    try {
      DocumentSnapshot userDoc =
          await FirebaseFirestore.instance.collection('users').doc(uid).get();

      return userDoc.exists ? userDoc['name'] : null;
    } catch (e) {
      print("Error fetching name: $e");
      return null;
    }
  }

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

    return Scaffold(
      appBar: AppBar(
        title: const Text("Profile"),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              CircleAvatar(
                radius: 50,
                backgroundColor: Colors.blueAccent,
                child: FutureBuilder<String?>(
                  future: _fetchUserName(user!.uid),
                  builder: (context, snapshot) {
                    if (snapshot.connectionState == ConnectionState.waiting) {
                      return const CircularProgressIndicator(color: Colors.white);
                    }
                    String userName = snapshot.data ?? "?";
                    return Text(
                      userName.isNotEmpty ? userName[0].toUpperCase() : "?",
                      style: const TextStyle(fontSize: 40, color: Colors.white),
                    );
                  },
                ),
              ),
              const SizedBox(height: 20),
              FutureBuilder<String?>(
                future: _fetchUserName(user.uid),
                builder: (context, snapshot) {
                  if (snapshot.connectionState == ConnectionState.waiting) {
                    return const CircularProgressIndicator();
                  }
                  return Text(
                    snapshot.data ?? "No Name",
                    style: const TextStyle(fontSize: 22, fontWeight: FontWeight.bold),
                  );
                },
              ),
              const SizedBox(height: 10),
              Text(
                user.email ?? "No Email",
                style: const TextStyle(fontSize: 18, color: Colors.grey),
              ),
              const SizedBox(height: 30),
              ElevatedButton(
                onPressed: () async {
                  await FirebaseAuth.instance.signOut();
                  Navigator.of(context).pushAndRemoveUntil(
                    MaterialPageRoute(builder: (context) => LoginPage()),
                    (Route<dynamic> route) => false,
                  );
                },
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.red,
                  padding: const EdgeInsets.symmetric(horizontal: 30, vertical: 12),
                ),
                child: const Text("Logout",
                    style: TextStyle(fontSize: 18, color: Colors.white)),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```