Divesh Punjabi
D15A 46
Batch B

Experiment no 6

Aim:To Connect Flutter UI with FireBase database

Theory:

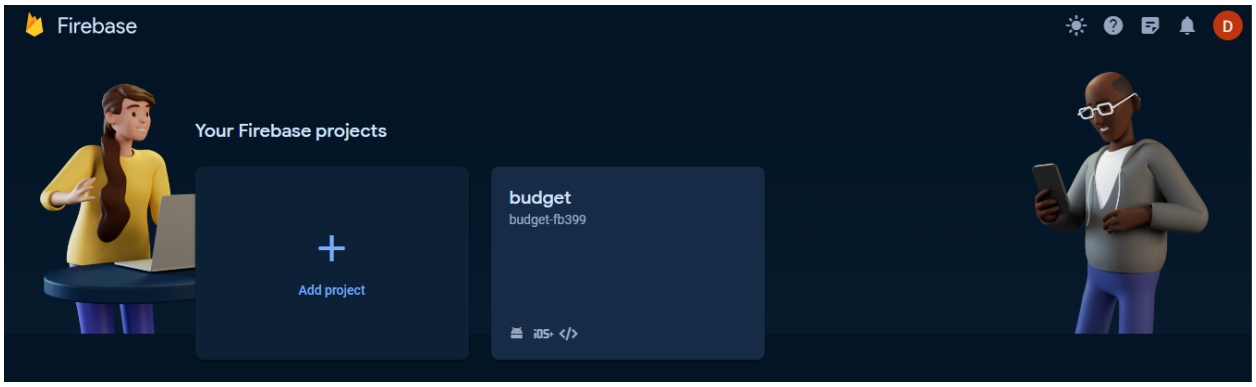
Prerequisites

To complete this tutorial, you will need:

- A Google account to use Firebase.
- Developing for iOS will require XCode.
- To download and install Flutter.
- To download and install Android Studio and Visual Studio Code.
- It is recommended to install plugins for your code editor:
  - Flutter and Dart plugins installed for Android Studio.
  - Flutter extension installed for Visual Studio Code.

1)Create a Firebase Project:

First, log in with your Google account to manage your Firebase projects. From within the Firebase dashboard,
select the Create new project button and give it a name:



2)Go to the Firebase Console and create a new project.

Add your Flutter app to the Firebase project:

Register your app in the Firebase project, and follow the instructions to download the configuration files (google-services.json for Android, GoogleService-Info.plist for iOS).

The most important thing here is to match up the Android package name that you choose here with the one
inside of our application.

The structure consists of at least two segments. A common pattern is to use a domain name, a company
name, and the application name:budget

3) Add Firebase to your Flutter project:

Add Dependencies:
dependencies:
  flutter:
    sdk: flutter
  firebase_core: 2.24.2
  cloud_firestore: 4.14.0
  firebase_auth: 4.16.0

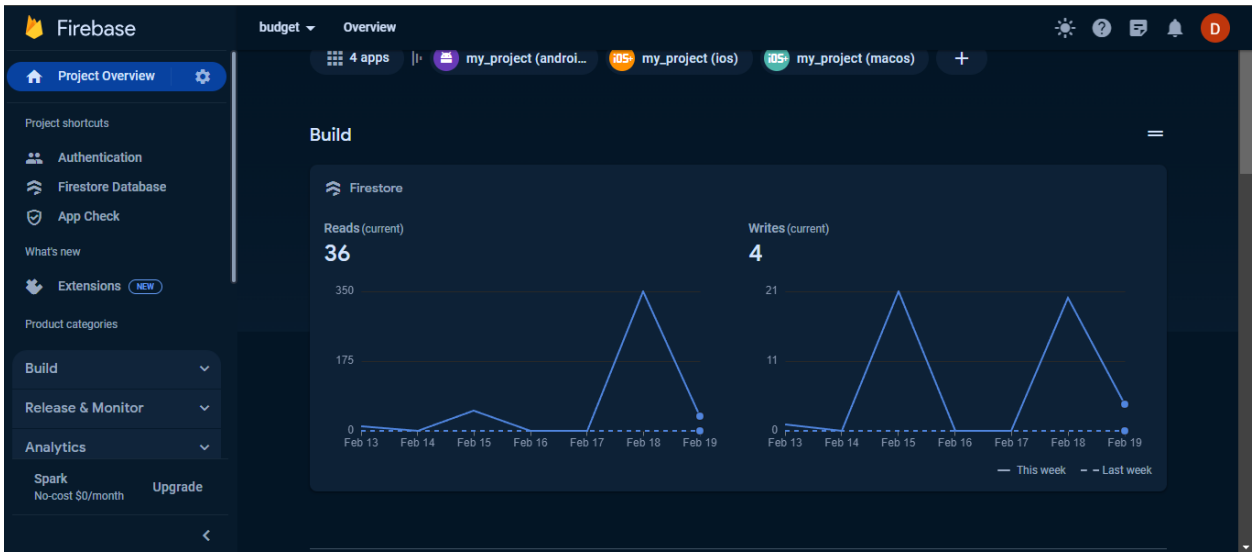
Code:
Db.dart

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

class Db {
  CollectionReference users = FirebaseFirestore.instance.collection('users');

  Future<void> addUser(data, context) async {
    final userId = FirebaseAuth.instance.currentUser!.uid;
    await users
        .doc(userId)
        .set(data)
        .then((value) => print("User Added"))
        .catchError((error) {
      showDialog(
          context: context,
          builder: (context) {
            return AlertDialog(
              title: Text("Sign up Failed"),
              content: Text(error.toString()),
            );
          });
```

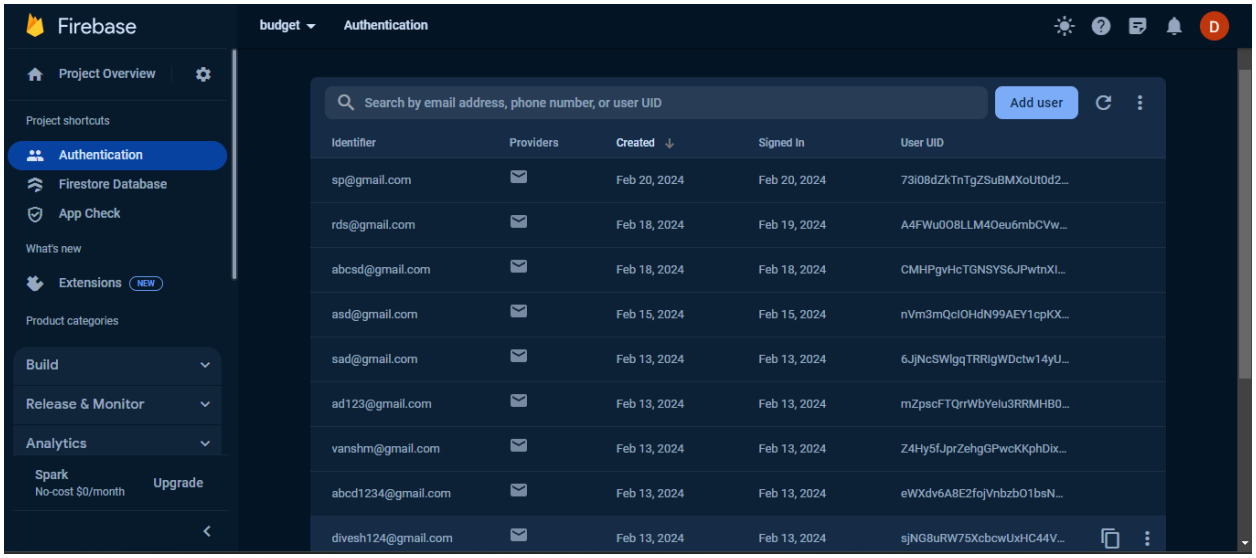
```
    });
  }
}

Auth_services.dart
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:my_project/screens/dashboard.dart';
import 'package:my_project/services/db.dart';

class AuthServices {
  var db = Db();
  createUser(data, context) async {
    try {
      await FirebaseAuth.instance.createUserWithEmailAndPassword(
        email: data['email'],
        password: data['password'],
      );
      await db.addUser(data, context);
      Navigator.of(context).pushReplacement(
          MaterialPageRoute(builder: (context) => Dashboard()));
    } catch (e) {
      showDialog(
          context: context,
          builder: (context) {
            return AlertDialog(
              title: Text("Sign up Failed"),
              content: Text(e.toString()),
            );
          });
    }
  }

  login(data, context) async {
    try {
      await FirebaseAuth.instance.signInWithEmailAndPassword(
        email: data['email'],
        password: data['password'],
      );
      Navigator.of(context).pushReplacement(
          MaterialPageRoute(builder: (context) => Dashboard()));
```

```
    } catch (e) {
      showDialog(
          context: context,
          builder: (context) {
            return AlertDialog(
              title: Text("Login Error"),
              content: Text(e.toString()),
            );
          });
    }
  }
}
```

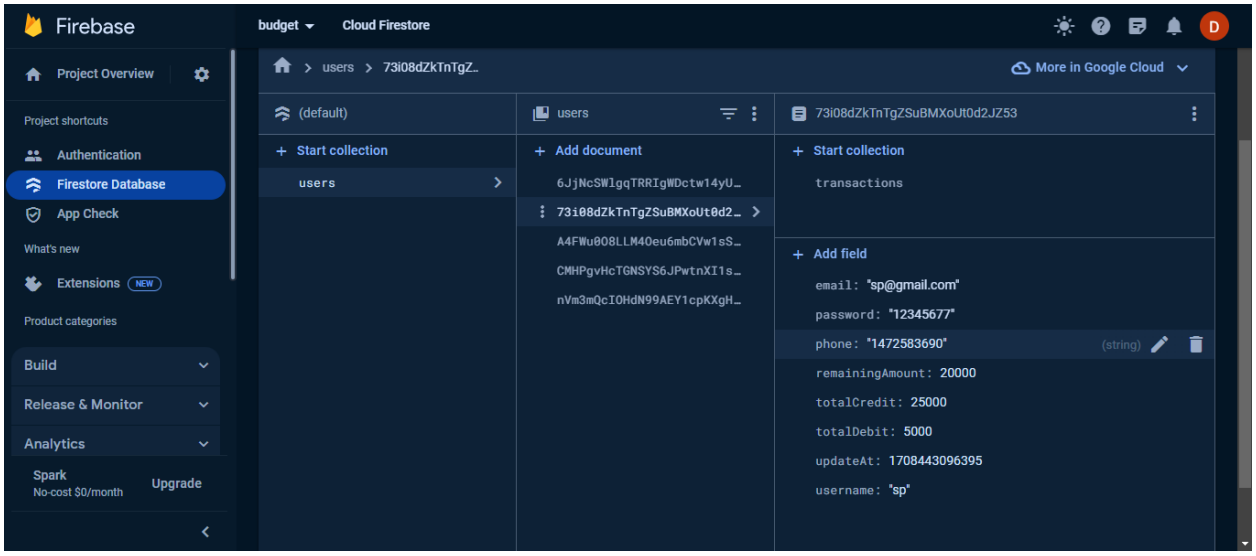
Output:



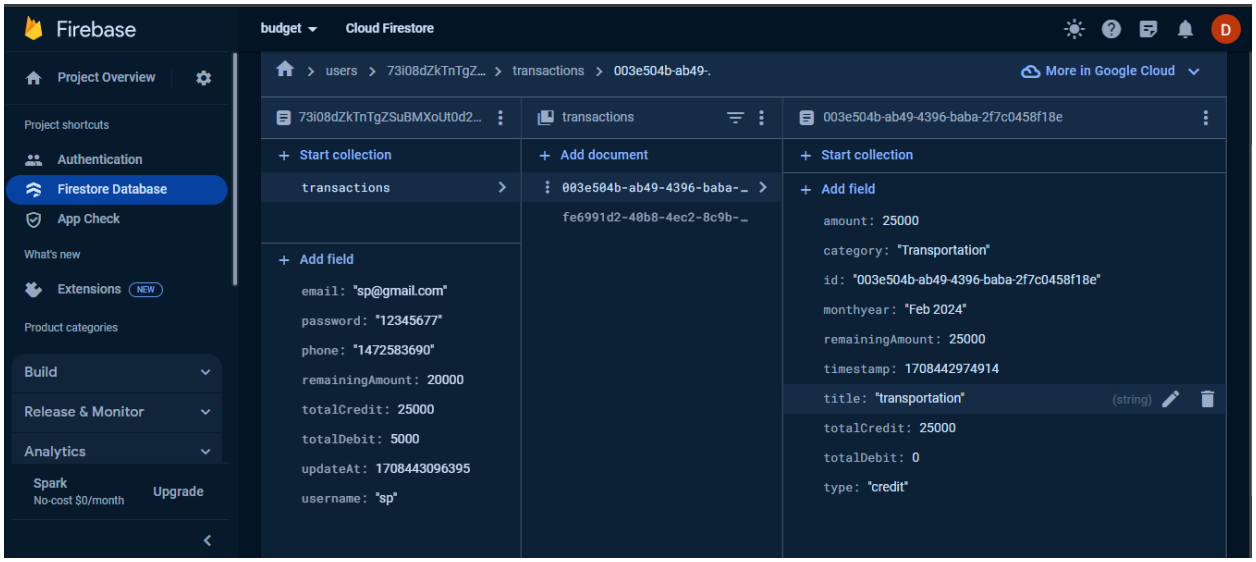
Authentications:



Users and transactions:



Transactions:



Conclusion:

In this experiment, we have successfully connected firebase database and authenticated using signin and email and password without flutter application successfully and also added the users and their transactions successfully