Name: Mohit Ailani Roll No: 2 Div: D15B

MAD Lab Experiment No 6

Aim: To connect Flutter UI with Firebase.

Theory: Firebase is a Backend-as-a-Service (BaaS) platform from Google that provides a suite of tools to simplify mobile app development. Integrating Firebase with Flutter allows you to leverage these tools within your Flutter application for functionalities like authentication, database management, storage, analytics, and more.
Here's a breakdown of key concepts related to Firebase in Flutter:

1. Firebase Services:

- Authentication: Provides features for user signup, login, password reset, social logins (Google, Facebook, etc.), and user management.
- Firestore: A NoSQL cloud database for storing and retrieving structured data in a flexible and scalable manner.
- Storage: Cloud storage for images, videos, and other files, accessible from your app.
- Cloud Functions: Serverless functions that you can write and deploy to handle backend logic without managing servers.
- Real-time Database: A database that synchronizes data across devices in real-time, ideal for collaborative apps or live updates.
- Analytics: Tracks user behavior and app usage to gather insights into user engagement and app performance.
- Remote Config: Allows you to dynamically change app configurations without app updates, suitable for A/B testing or feature flags.
- Crashlytics: Provides crash reporting and analysis tools to help identify and fix bugs in your app.

2. Integration with Flutter:

- FlutterFire Plugins: Firebase provides official Flutter plugins (e.g., firebase_auth, cloud_firestore) for each service that simplify integration with your Flutter app.
- Setting Up Firebase Project: You'll need to create a Firebase project and configure it in your Flutter code using the provided web interface and configuration files.
- Adding Dependencies: Include the necessary FlutterFire plugins for the Firebase services you want to use in your pubspec.yaml file.

3. Using Firebase Services in Your App:

- Authentication Example:
    1. Initialize Firebase Auth using FirebaseAuth.instance.

    2. Implement user signup or login logic using methods like createUserWithEmailAndPassword or signInWithEmailAndPassword.
    3. Securely store user credentials or tokens using secure storage mechanisms.
- Firestore Example:
    1. Initialize Firestore using FirebaseFirestore.instance.
    2. Access collections and documents within your Firestore database.
    3. Use methods like add, get, set, and update to manage data.
- Storage Example:
    1. Initialize Cloud Storage using FirebaseStorage.instance.
    2. Upload or download files using methods like putFile and downloadURL.

<u>4. Benefits of Using Firebase in Flutter:</u>

- Reduced Development Time: Firebase provides pre-built services, saving you time on writing backend code.
- Scalability: Firebase services are automatically scalable, handling increased app usage without infrastructure management.
- Offline Support (Firestore and Realtime Database): Certain Firebase services offer offline capabilities for a seamless user experience.
- Realtime Features: Firebase Realtime Database enables real-time data synchronization across devices.
- Analytics and Monitoring: Firebase provides tools to track app usage and identify issues.

Steps to Set Up Firebase:

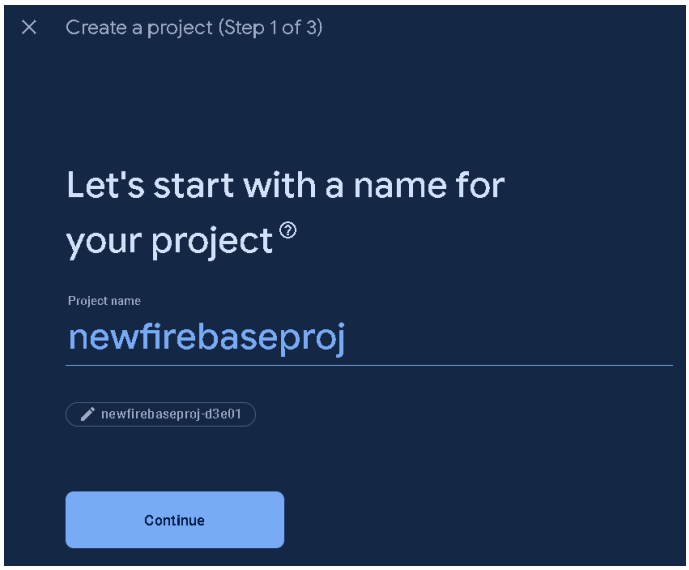
Prerequisites

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

Creating a New Flutter Project:

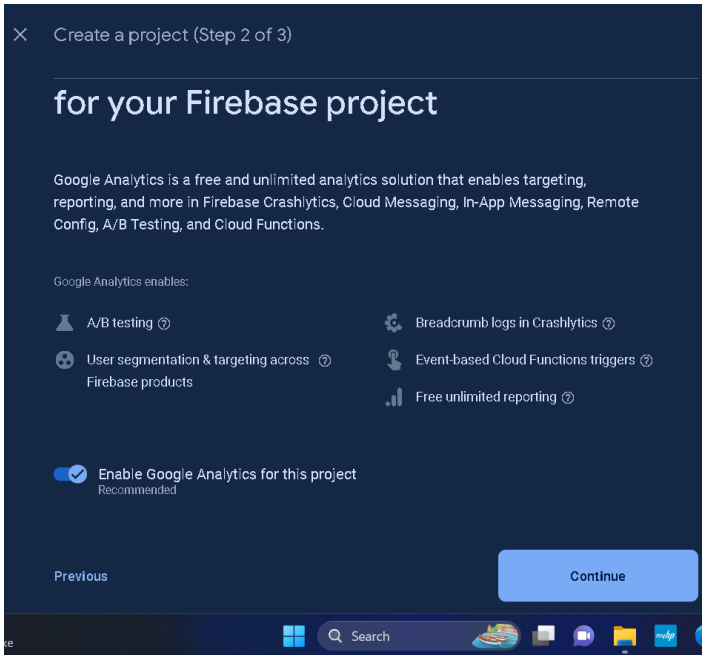
This tutorial will require the creation of an example Flutter app.
Once you have your environment set up for Flutter, you can run “flutter create <appname>” command.

Creating a New 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:

Next, we're given the option to enable Google Analytics. This tutorial will not require Google Analytics, but you can also choose to add it to your project.



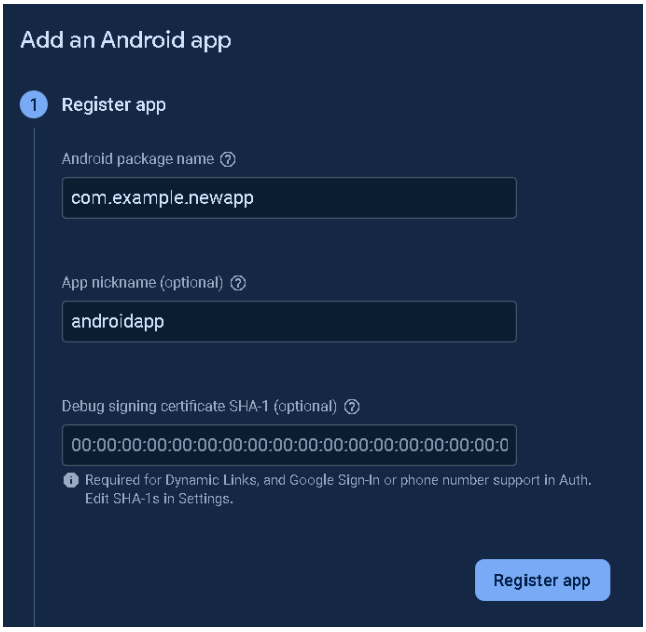
If you choose to use Google Analytics, you will need to review and accept the terms and conditions prior to project creation.

After pressing Continue, your project will be created and resources will be provisioned. You will then be directed to the dashboard for the new project.

Adding Android support:

1. Registering the App

In order to add Android support to our Flutter application, select the Android logo from the dashboard. This brings us to the following screen:



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

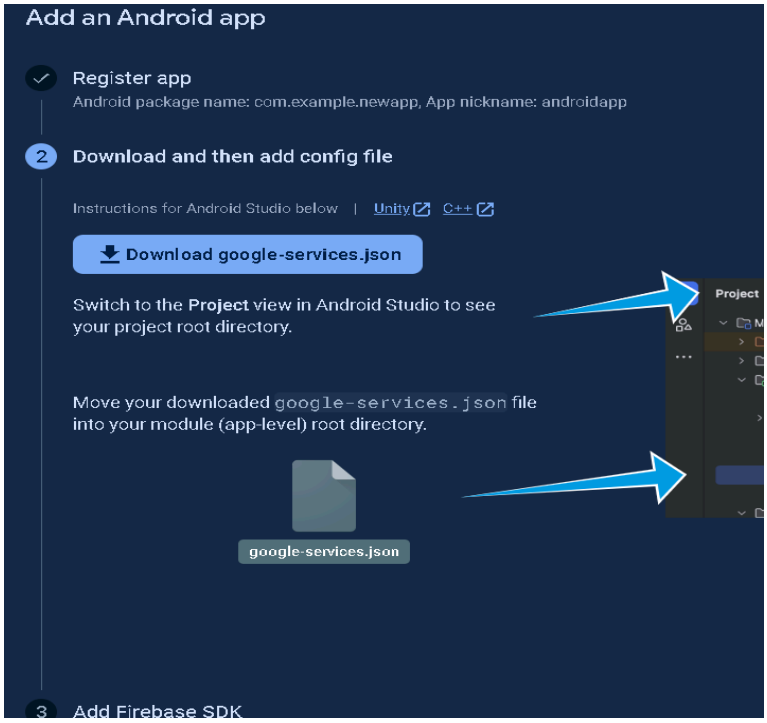
The structure consists of at least two segments. A common pattern is to use a domain name, a company name, and the application name: com.example.flutterfirebaseexample

Once you've decided on a name, open android/app/build.gradle in your code editor and update the applicationId to match the Android package name.Select Register app to continue.

2. Downloading the Config File

The next step is to add the Firebase configuration file into our Flutter project. This is important as it contains the API keys and other critical information for Firebase to use.

Select Download google-services.json from this page:



Next, move the google-services.json file to the android/app directory within the Flutter project.

3. Adding the Firebase SDK

We'll now need to update our Gradle configuration to include the Google Services plugin.

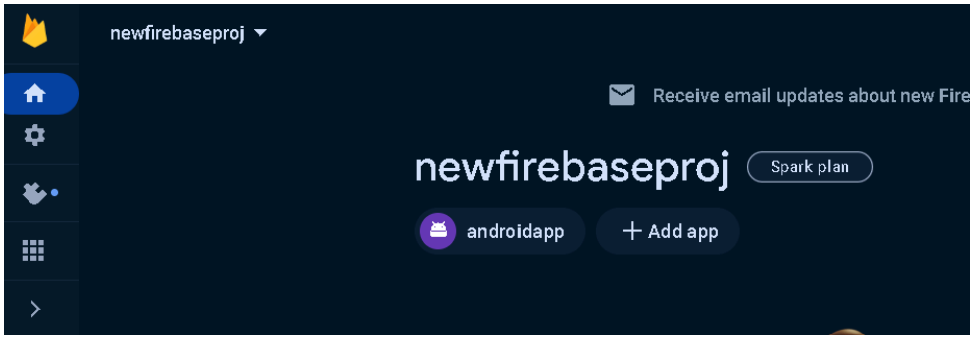
2. Then, in your module (app-level) `build.gradle` file, add both the `google-services` plugin and any Firebase SDKs that you want to use in your app:

Module (app-level) Gradle file (`<project>/<app-module>/build.gradle`):

```
plugins {
  id 'com.android.application'
  // Add the Google services Gradle plugin
  id 'com.google.gms.google-services'
  ...
}

dependencies {
  // Import the Firebase BoM
  implementation platform('com.google.firebase:firebase-bom:32.7.2')

  // TODO: Add the dependencies for Firebase products you want to use
  // When using the BoM, don't specify versions in Firebase dependencies
  implementation 'com.google.firebase:firebase-analytics'

  // Add the dependencies for any other desired Firebase products
  // https://firebase.google.com/docs/android/setup#available-libraries
}
```

By using the Firebase Android BoM, your app will always use compatible Firebase library versions. Learn more

With this update, we're essentially applying the Google Services plugin as well as looking at how other Flutter Firebase plugins can be activated such as Analytics.



Using Firebase Authentication Functionality

Get Started with Firebase Authentication on Flutter

Send feedback

Connect your app to Firebase

Install and initialize the Firebase SDKs for Flutter if you haven't already done so.

Add Firebase Authentication to your app

1. From the root of your Flutter project, run the following command to install the plugin:

```
flutter pub add firebase_auth
```

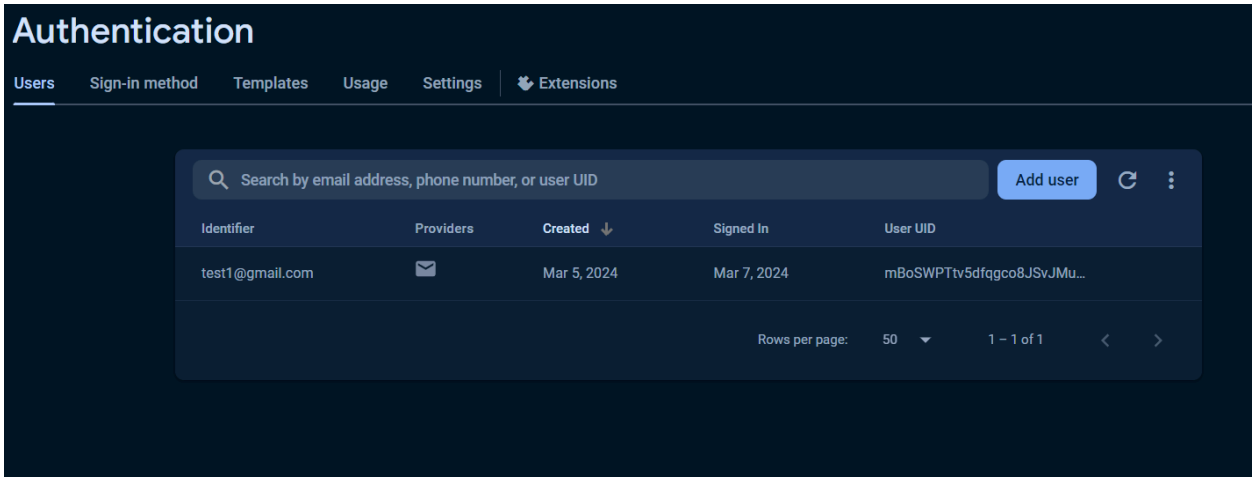
2. Once complete, rebuild your Flutter application:

```
flutter run
```

3. Import the plugin in your Dart code:

```
import 'package:firebase_auth/firebase_auth.dart';
```

Creating a test case for authentication purpose:




```
dependencies:
  flutter:
    sdk: flutter

  firebase_core: ^2.26.0
  firebase_auth: ^4.17.7
```

Code:

```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flashcards_quiz/views/home_page.dart';

class LoginPage extends StatefulWidget {
  const LoginPage({super.key});

  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final _usernameController = TextEditingController();
  final _passwordController = TextEditingController();
  final FirebaseAuth _auth = FirebaseAuth.instance;
  String _errorText = '';

  @override
  void dispose() {
    _usernameController.dispose();
    _passwordController.dispose();
    super.dispose();
  }

  Future<void> _signInWithEmailAndPassword() async {
    try {
      final UserCredential userCredential =
          await _auth.signInWithEmailAndPassword(
        email: _usernameController.text.trim(),
        password: _passwordController.text,
      );
      if (userCredential.user != null) {
        // Navigate to home page if authentication successful
        // ignore: use_build_context_synchronously
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => const HomePage1()),
```

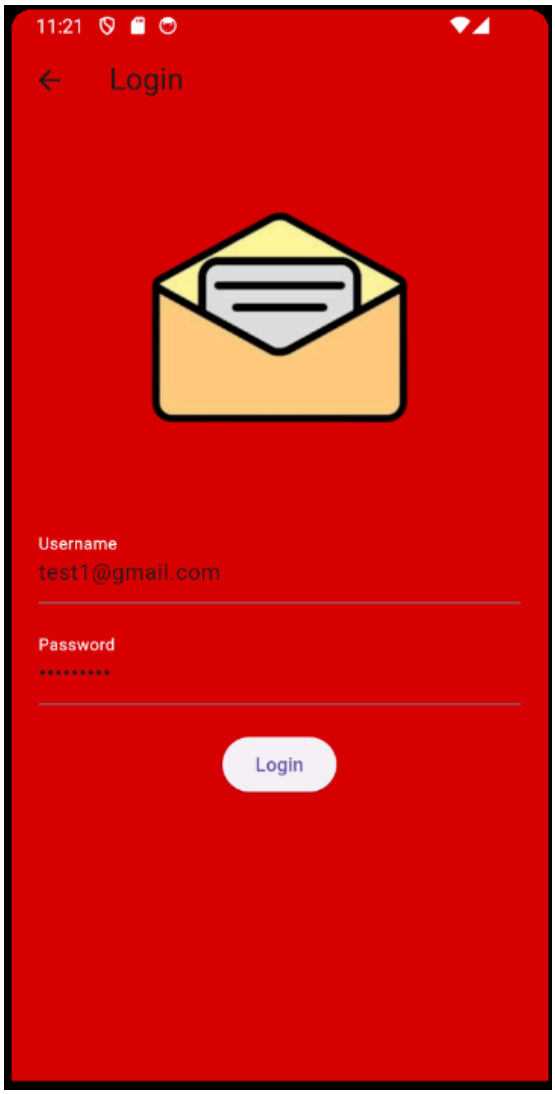
```
      );
    }
  } catch (e) {
    setState(() {
      _errorText =
          'Invalid email or password'; // Set error message for invalid credentials
    });
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text("Login"),
      backgroundColor: const Color.fromARGB(255, 215, 1, 11),
    ),
    backgroundColor: const Color.fromARGB(255, 215, 1, 11),
    body: SingleChildScrollView(
      // Make content scrollable
      padding: const EdgeInsets.all(20.0),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          // Add your image widget here
          Image.asset(
            "assets/login_imgg.png",
            width: 250,
            height: 250,
          ),
          const SizedBox(height: 20),
          TextField(
            controller: _usernameController,
            decoration: const InputDecoration(
              labelText: "Username",
              labelStyle: TextStyle(
                color: Colors.white, // Set text color
              ),
```

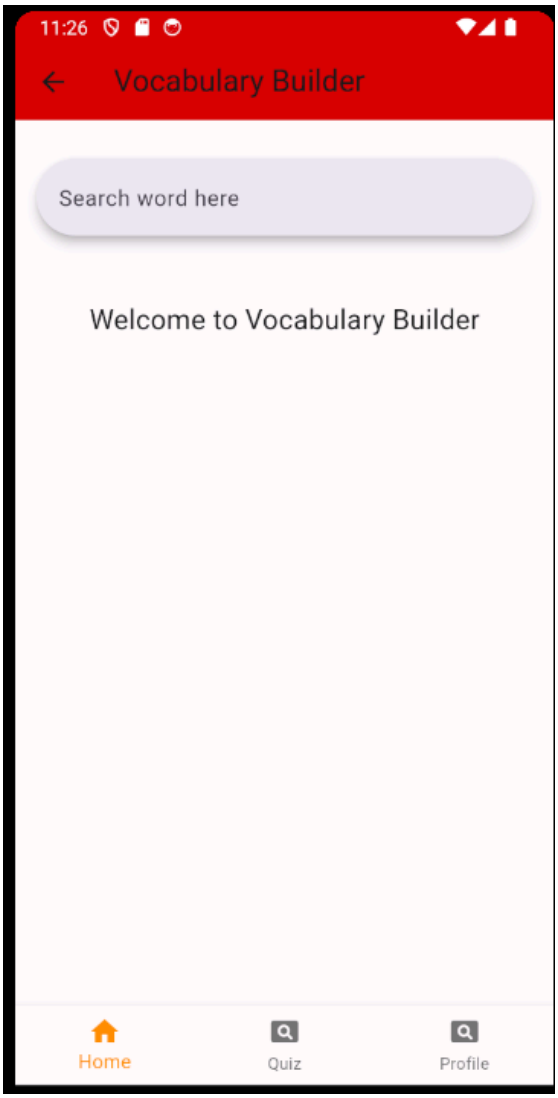
```
          ),
        ),
        const SizedBox(height: 10),
        TextField(
          controller: _passwordController,
          obscureText: true,
          decoration: InputDecoration(
            labelText: "Password",
            labelStyle: const TextStyle(
              color: Colors.white, // Set text color
            ),
            errorText: _errorText.isNotEmpty ? _errorText : null,
            errorStyle: const TextStyle(
                color: Colors.white), // Set error text color
          ),
        ),
        const SizedBox(height: 20),
        ElevatedButton(
          onPressed:
              _signInWithEmailAndPassword, // Call method for authentication
          child: const Text("Login"),
        ),
      ],
    ),
  ),
);
}
}
```

After running the code check whether the firebase connection is established with the project or not:

```
DEBUG CONSOLE   PORTS   PROBLEMS 7   OUTPUT   TERMINAL                    Filter
Launching lib\main.dart on sdk gphone64 x86 64 in debug mode...
Parameter format not correct -
√  Built build\app\outputs\flutter-apk\app-debug.apk.
Connecting to VM Service at ws://127.0.0.1:52125/PyYflatVEZM=/ws
D/CompatibilityChangeReporter(30302): Compat change id reported: 3400644; UID 10193; state: DISABLED
D/FirebaseAuth(30302): Notifying id token listeners about user ( mBoSWPTtv5dfqgco8JSvJMuNO1o2 ).
```

Output:





Conclusion : From this experiment, first of all we studied how to setup firebase. Then we created a firebase project, added multiple dependencies and packages to our flutter project so as to integrate our flutter project with firebase. Next step we authenticate the input fields from our app like email and password using Firebase and store it.