Name: Abujaid Ansari **Batch: A**

Class: D15B **Roll No. 02**

Experiment No. 6

Aim: To connect Flutter UI with firebase.

Theory:

Google Firebase is a set of cloud-based development tools that helps mobile app developers build, deploy and scale their apps.

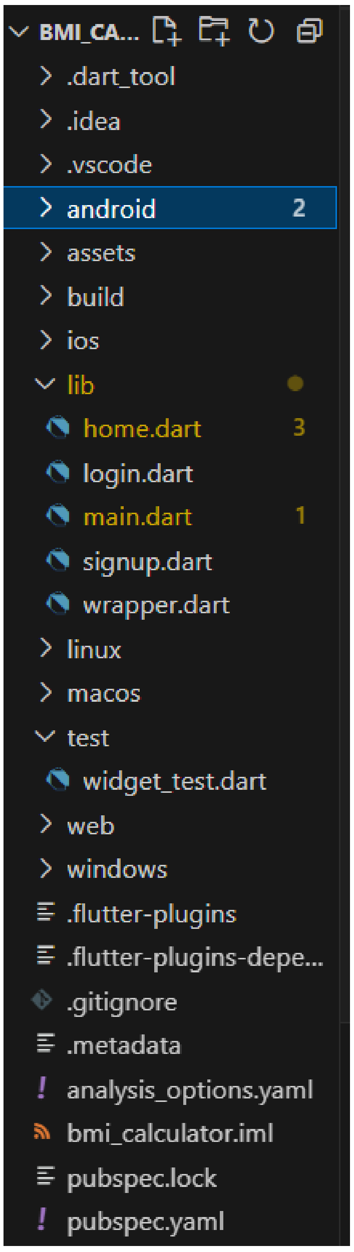
Firebase provides a variety of features, including the following:

- Authentication: Firebase provides a secure and easy way for users to sign into their app. Developers can use Firebase Authentication to support email and password login, Google Sign-In, Facebook Login and more.
- Realtime Database: The Firebase Realtime Database is a cloud-hosted NoSQL database that lets organizations store and sync data in real time across all of their users' devices. This makes it easy to build apps that are always up to date, even when users are offline.
- Cloud Messaging: Firebase Cloud Messaging (FCM) is a service that lets businesses send messages to their users' devices, even if they're not using the app. Developers can use FCM to send push notifications, update app content, and more.
- Crashlytics: Firebase Crashlytics is a service that helps organizations track and fix crashes in their app. Crashlytics provides detailed reports on crashes, so they can quickly identify the root cause and fix the problem.
- Performance Monitoring: Firebase Performance Monitoring provides insights into the performance of their app. Organizations can use Performance Monitoring to track metrics like CPU usage, memory usage and network traffic.
- Test Lab: Firebase Test Lab is a cloud-based service that lets developers test their app on a variety of devices and configurations. This helps them ensure the app works well on a variety of devices and in different network conditions.

Steps to connect firebase with flutter app:

Step-1: create a flutter app

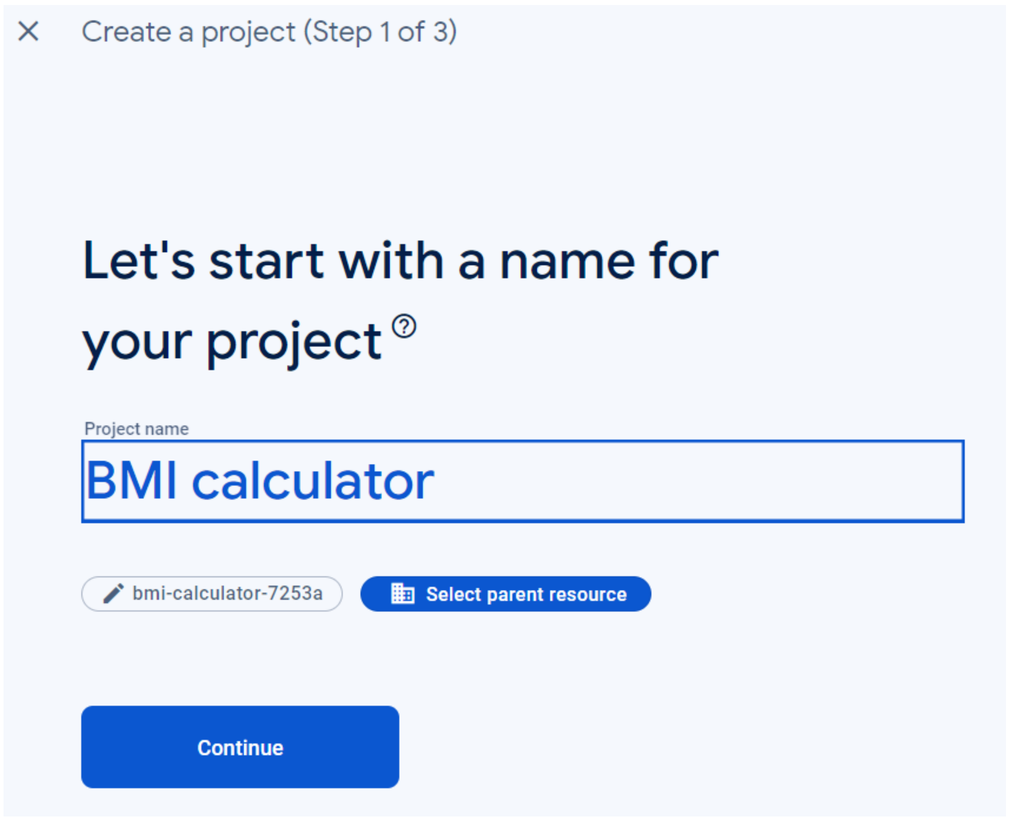
We have created a new flutter project with name bmi calculator.



```
BMI_CA...
  > .dart_tool
  > .idea
  > .vscode
  > android                 2
  > assets
  > build
  > ios
  v lib                     ●
      home.dart             3
      login.dart
      main.dart             1
      signup.dart
      wrapper.dart
  > linux
  > macos
  v test
      widget_test.dart
  > web
  > windows
    .flutter-plugins
    .flutter-plugins-depe...
    .gitignore
    .metadata
  ! analysis_options.yaml
    bmi_calculator.iml
    pubspec.lock
  ! pubspec.yaml
```

Step 2: create 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.

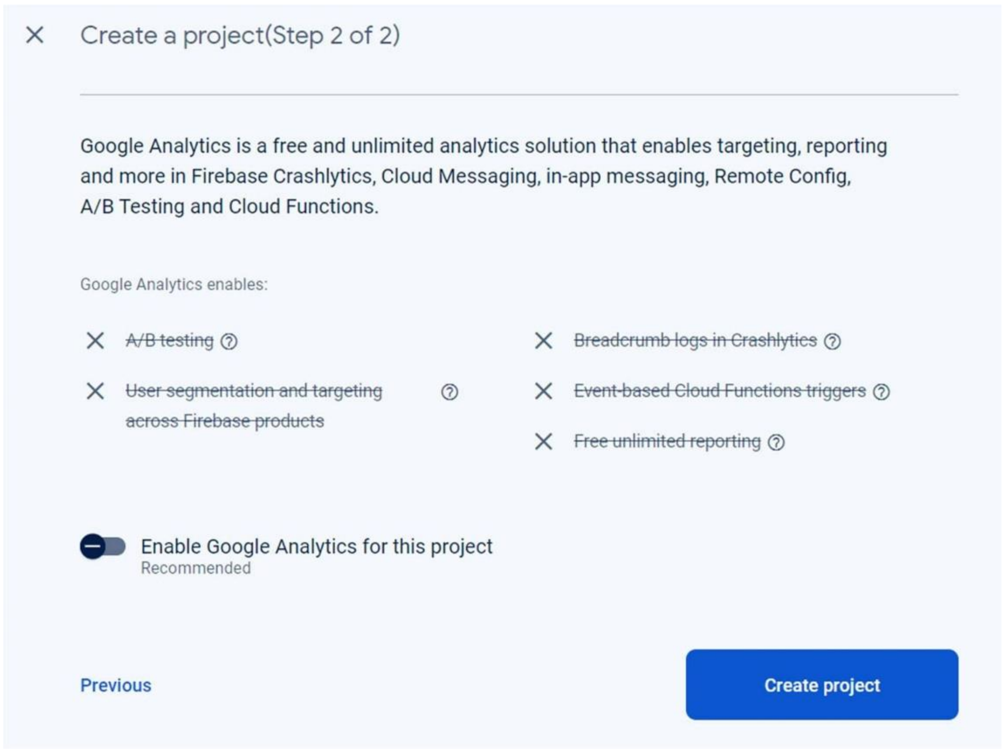

Create a project (Step 1 of 3)

Let's start with a name for your project

Project name: BMI calculator

bmi-calculator-7253a

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.


Create a project(Step 2 of 2)

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting and more in Firebase Crashlytics, Cloud Messaging, in-app messaging, Remote Config, A/B Testing and Cloud Functions.

Google Analytics enables:

- ~~A/B testing~~
- ~~User segmentation and targeting across Firebase products~~
- ~~Breadcrumb logs in Crashlytics~~
- ~~Event-based Cloud Functions triggers~~
- ~~Free unlimited reporting~~

Enable Google Analytics for this project
Recommended

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

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.BMI_calculator”

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:

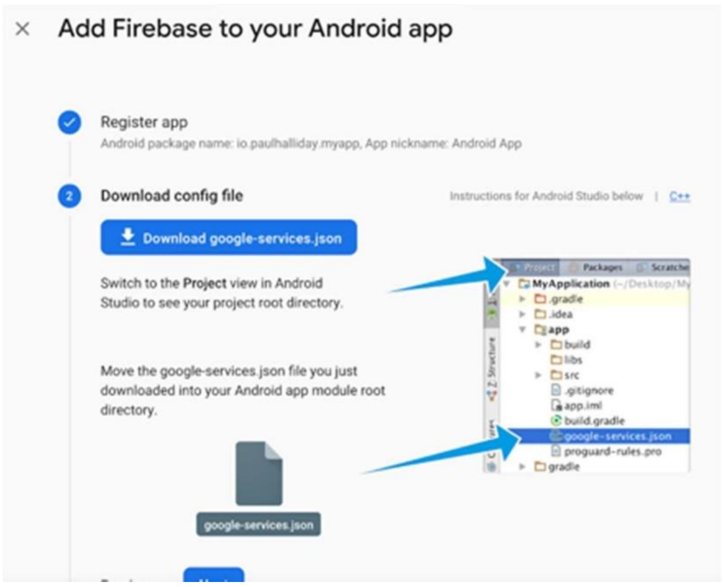
android/app/build.gradle

```
defaultConfig {
        // TODO: Specify your own unique Application ID
(https://developer.android.com/studio/build/application-id.html).
        applicationId "com.example.recipe_app"
        // You can update the following values to match your application needs.
```

You can skip the app nickname and debug signing keys at this stage. Select Register app to continue.

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:



Add Firebase to your Android app

Register app
Android package name: io.paulhalliday.myapp, App nickname: Android App

Download config file
Instructions for Android Studio below | C++

Download google-services.json

Switch to the Project view in Android Studio to see your project root directory.

Move the google-services.json file you just downloaded into your Android app module root directory.

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

Step 3: Adding the Firebase SDK:
We'll now need to update our Gradle configuration to include the Google Services plugin.

Openandroid/build.gradle in your code editor and modify it to include the following:

android/build.gradle

```
buildscript {
ext.kotlin_version = '1.7.10'
repositories {         google()

        mavenCentral()

    }


    dependencies {         classpath "org.jetbrains.kotlin:kotlin-gradle-
plugin:$kotlin_version"             classpath  'com.google.gms:google-
services:4.3.15'
    }

}
```

Finally, update the app level file at android/app/build.gradle to include the following:

android/app/build.gradle

```
plugins {     id
"com.android.application"     id
```

"kotlin-android"   id
"dev.flutter.flutter-gradle-plugin"   id
'com.google.gms.google-services'

}

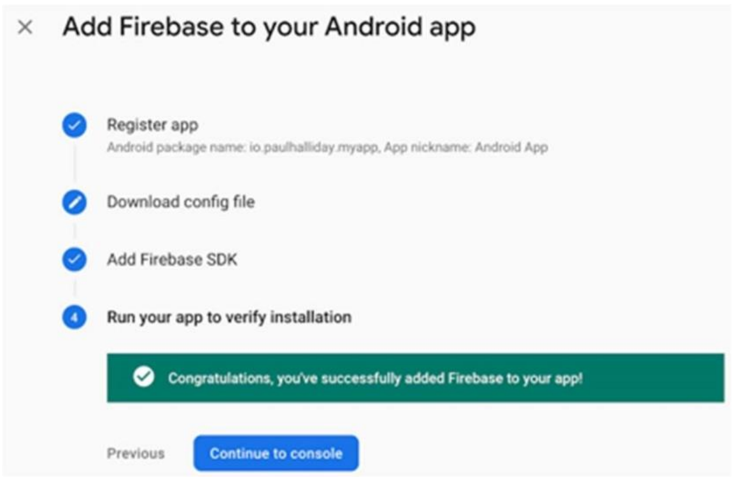
 ...

dependencies {   implementation
platform('com.google.firebase:firebase-bom:32.7.3')

}

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.

From here, run your application on an Android device or simulator. If everything has worked correctly, you should get the following message in the dashboard:


Add Firebase to your Android app

Register app
Android package name: io.paulhalliday.myapp, App nickname: Android App

Download config file

Add Firebase SDK

Run your app to verify installation

Congratulations, you've successfully added Firebase to your app!

Step 4: Login/ SignUp using firebase authentication:

We make a wrapper file in which it checks whether user is login or not.

wrapper.dart

```dart
import 'package:bmi_calculator/home.dart';
import 'package:bmi_calculator/login.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';

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

  @override
  State<Wrapper> createState() => _WrapperState();
}
```

```
class _WrapperState extends State<Wrapper> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: StreamBuilder(stream: FirebaseAuth.instance.authStateChanges(),
      builder: (context,snapshot){
        if (snapshot.hasData){
          return Home();
        }
        else{
          return Login();
        }
      }),
    );
  }
}
```

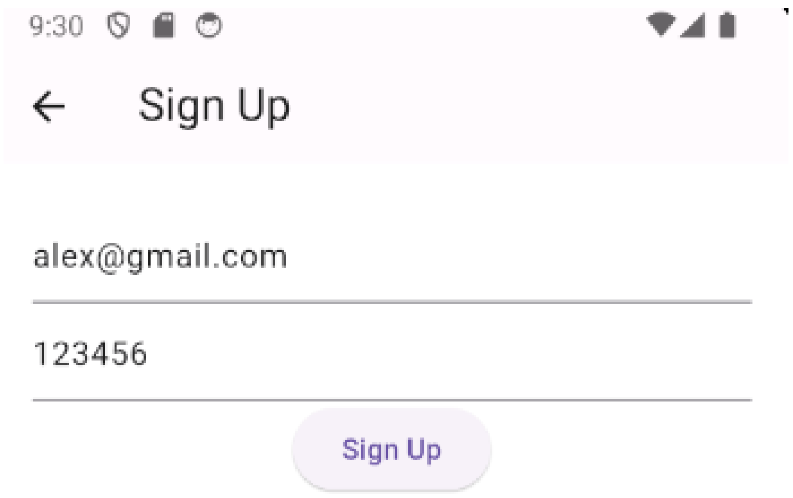
signin.dart

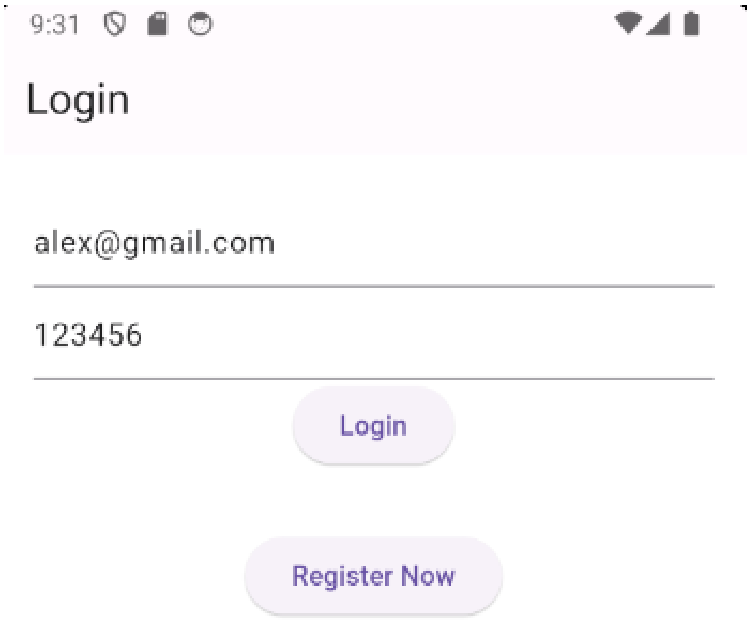
```
import 'package:bmi_calculator/signup.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

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

  @override
  State<Login> createState() => _LoginState();
}

class _LoginState extends State<Login> {

  TextEditingController email=TextEditingController();
  TextEditingController password=TextEditingController();

  signIn() async {
    await FirebaseAuth.instance.signInWithEmailAndPassword(email: email.text,
password: password.text);
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Login"),),
      body:Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          children: [
            TextField(
              controller:email,
              decoration: InputDecoration(hintText: 'Enter email'),
            ),
            TextField(
              controller:password,
              decoration: InputDecoration(hintText: 'Enter password'),
            ),
            ElevatedButton(onPressed: (()=>signIn()), child: Text("Login")),
            SizedBox(height: 30,),
            ElevatedButton(onPressed: (()=>Get.to(SignUp())), child: Text("Register
Now"))
          ],),
      )
    );
  }
}
```

signup.dart

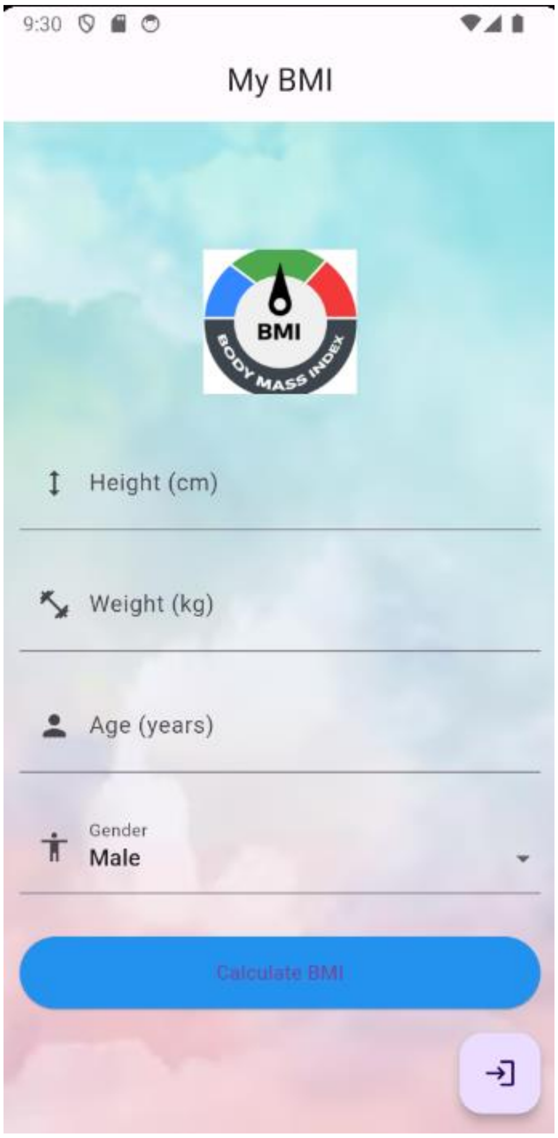
```
import 'package:bmi_calculator/wrapper.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

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

  @override
  State<SignUp> createState() => _SignUpState();
}

class _SignUpState extends State<SignUp> {

  TextEditingController email=TextEditingController();
  TextEditingController password=TextEditingController();

  signUp() async {
    await FirebaseAuth.instance.createUserWithEmailAndPassword(email: email.text,
password: password.text);
    Get.offAll(()=>Wrapper());
  }
  @override
  Widget build(BuildContext context) {
return Scaffold(
      appBar: AppBar(title: Text("Sign Up"),),
      body:Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          children: [
            TextField(
              controller:email,
              decoration: InputDecoration(hintText: 'Enter email'),
            ),
            TextField(
              controller:password,
              decoration: InputDecoration(hintText: 'Enter password'),
            ),
            ElevatedButton(onPressed: (()=>signUp()), child: Text("Sign Up"))
          ],),
      )
    );
  }
}
```

Output:



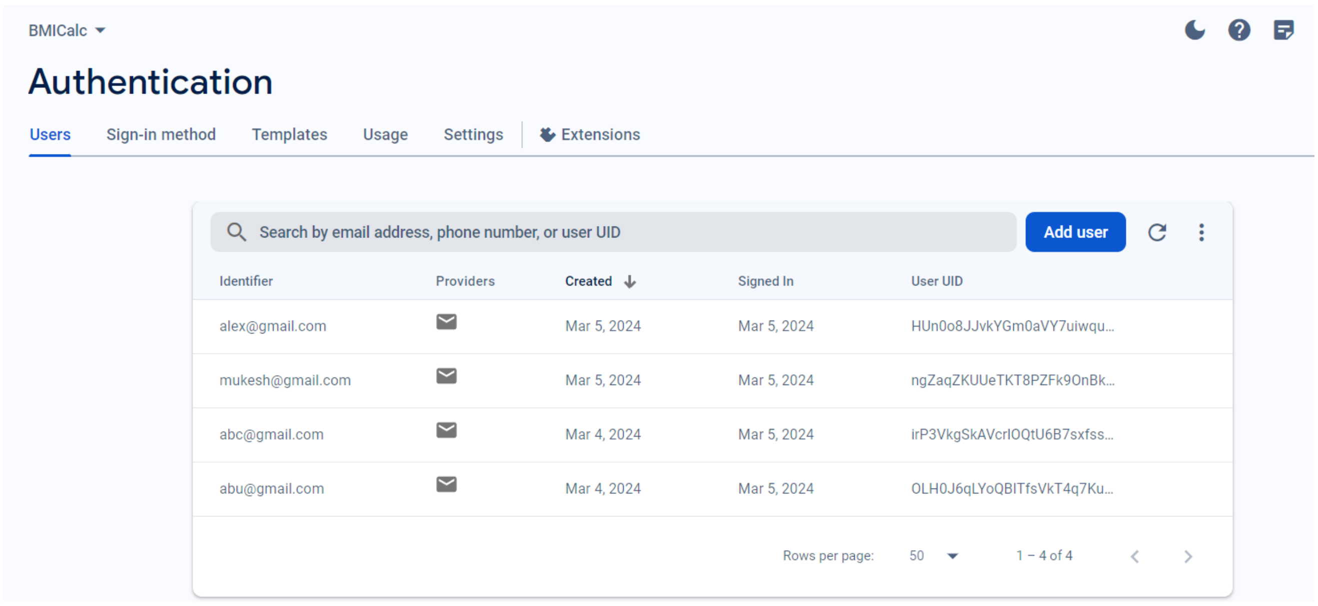


After login, we route to home page.
After sign up/ register in our app the authentication adds the user in firebase authentication services:



Gender
Male

Firebase authenticates the entered credentials of the user from the database and allows the login.



Identifier	Providers	Created	Signed In	User UID
alex@gmail.com	✉	Mar 5, 2024	Mar 5, 2024	HUn0o8JJvkYGm0aVY7uiwqu...
mukesh@gmail.com	✉	Mar 5, 2024	Mar 5, 2024	ngZaqZKUUeTKT8PZFk9OnBk...
abc@gmail.com	✉	Mar 4, 2024	Mar 5, 2024	irP3VkgSkAVcrlOQtU6B7sxfss...
abu@gmail.com	✉	Mar 4, 2024	Mar 5, 2024	OLH0J6qLYoQBITfsVkT4q7Ku...

Rows per page: 50 1 – 4 of 4

Conclusion:

Hence in this experiment, we understood what is firebase and connected it to our flutter app by installing packages and adding required dependencies. Then, we created a login page, signup page and a wrapper file to check whether the user is logged in or not and give him a signout option. Then using firebase authentication we added the login functionality which directs to the home page of our bmi calculator once the firebase authenticates the user.