Modul 7

Integrating Firebase into a Flutter App (2)

Firebase Authentication

Module Overview

Membuat proyek dengan memanfaatkan Firebase Authentication sebagai metode otentikasi untuk masuk ke dalam aplikasi.

Module Objectives

Setelah mempelajari dan mempraktikkan modul ini, mahasiswa diharapkan dapat:

- Mengetahui mengenai fitur Firebase Authentication
- Menerapkan otentikasi pada aplikasi menggunakan fitur Firebase Authentication

Pada minggu sebelumnya Anda telah belajar cara mengintegrasikan project Flutter Anda dengan project Firebase Anda secara khusus Anda telah belajar cara menggunakan fitur Firebase Analytics di mana dengan fitur ini Anda dapat melihat statistik penggunaan dari aplikasi Anda secara keseluruhan. Selain fitur analisis yang sudah Anda coba gunakan sebelumnya, Firebase juga menyediakan beragam macam fitur yang dapat Anda gunakan untuk mempermudah Anda dalam melakukan pengembangan aplikasi.

Di minggu ini Anda akan belajar cara menggunakan Firebase Authentication di mana fitur ini akan membantu Anda dalam menyiapkan metode otentikasi pengguna untuk masuk ke dalam aplikasi Anda. Dengan adanya fitur ini Anda tidak perlu lagi dipusingkan untuk mengatur metode otentikasi pada aplikasi Anda karena semuanya sudah dikerjakan oleh Firebase.

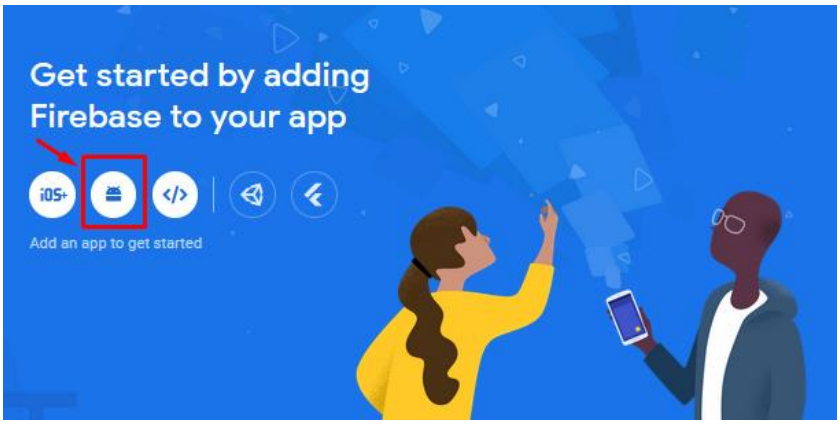
Firebase Authentication menyediakan beberapa fitur yang dapat Anda gunakan untuk menyediakan otentikasi ke aplikasi Anda, beberapa di antaranya:

- Otentikasi menggunakan nama pengguna dan password, atau penyedia layanan otentikasi lain seperti Google, Microsoft, Facebook, dan lain sebagainya.
- Pembuatan identitas pengguna.
- Metode login, logout, sign up, dan reset password.
- Integrasi dengan fitur-fitur lain di Firebase, sehingga Anda dapat dengan mudah menangani aturan otorisasi setelah identitas dibuat.

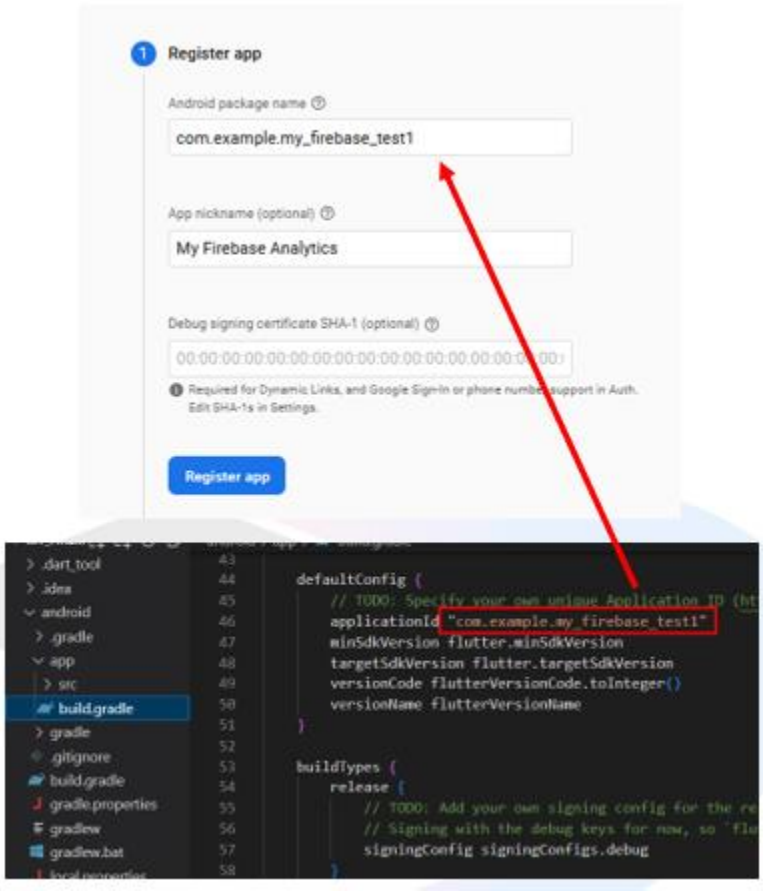
Integrasi Firebase

Sebagai reminder pertemuan sebelumnya, pertama sekali yang harus Anda lakukan adalah mengintegrasikan project Flutter Anda dengan project Firebase Anda. Adapun tahapan-tahapannya yaitu:

1. Silahkan login ke https://console.firebase.google.com/, lalu pilih atau buat sebuah project Firebase (untuk tahapan lengkapnya silahkan buka kembali **modul M06** sebelumnya) hingga Anda sampai di tahapan menghubungkan aplikasi Anda dengan Firebase seperti di bawah ini.

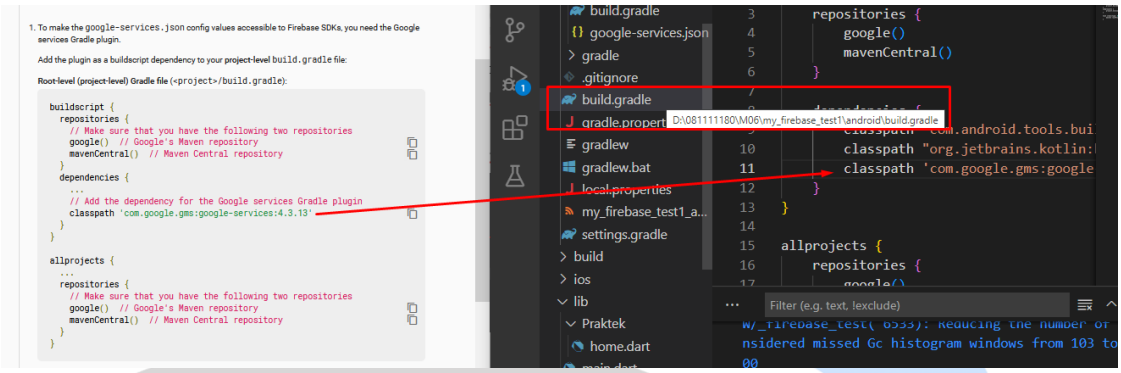


2. Pada tahapan Register App, isikan bagian **Android package name** dengan **applicationID** yang ada di dalam file **Android/app/build.gradle** (gradle level app) serta ganti **minSdkVersion** menjadi **19**. Kemudian klik Register app.

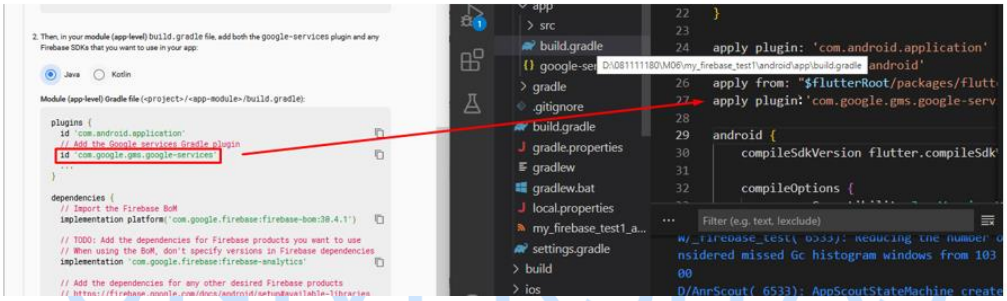


3. Silahkan download file json project Anda dan tempatkan di dalam folder **Android/app**. Kemudian klik Next.

4. Pada file **Android/build.gradle** (gradle level project), tambahkan **classpath 'com.google.gms:google-services:4.3.13'** di bagian **dependencies**. Jangan lupa untuk menyimpan perubahan.



5. Pada file **Android/app/build.gradle** (gradle level app), tambahkan **apply plugin: 'com.google.gms.google-services'** di atas bagian **android**. Jangan lupa untuk menyimpan perubahan.



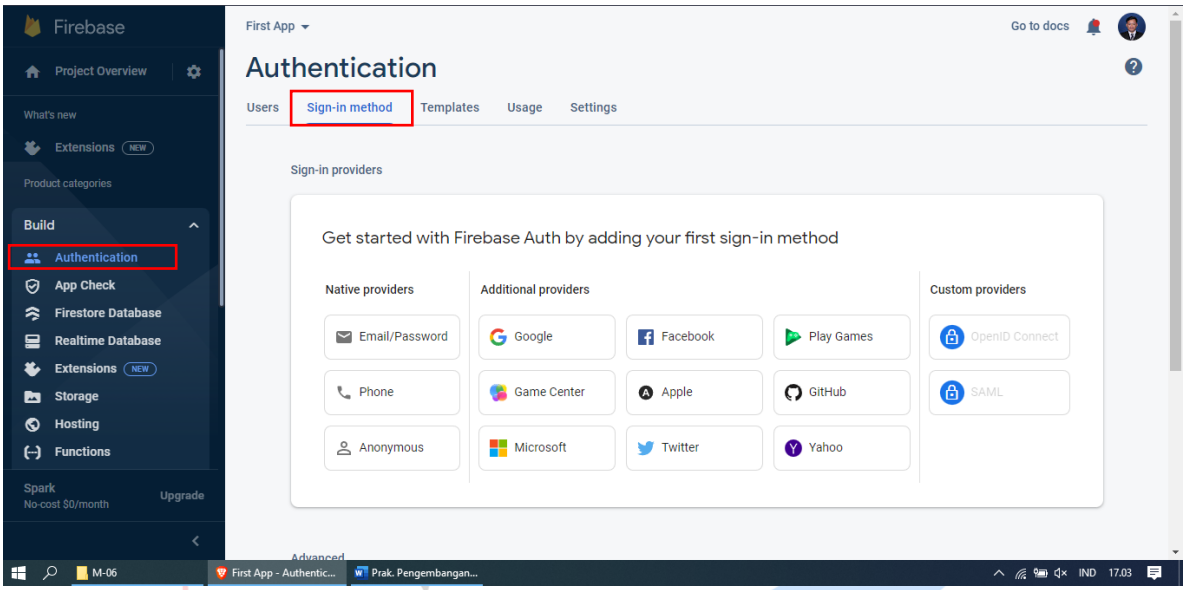
6. Setelah semua setup telah dilakukan silahkan kembali ke console Firebase lalu klik **continue to console**.

Mengaktifkan Fitur Authentication

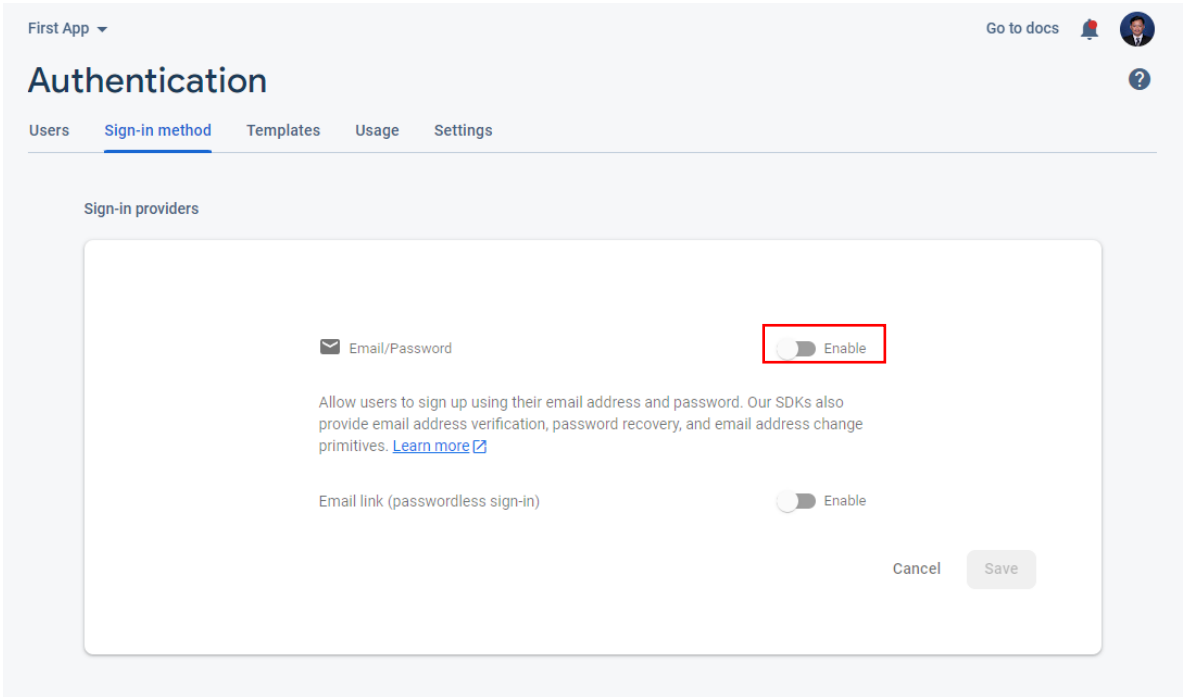
Untuk mulai menggunakan fitur Authentification Anda harus mengaktifkannya terlebih dahulu pada **Firebase Console** project Anda. Adapun tahapan-tahapan untuk mengaktifkan fitur ini adalah sebagai berikut:

1. Silahkan masuk kembali di https://console.firebase.google.com/ menggunakan akun google Anda lalu pilih project Firebase yang akan Anda pergunakan.

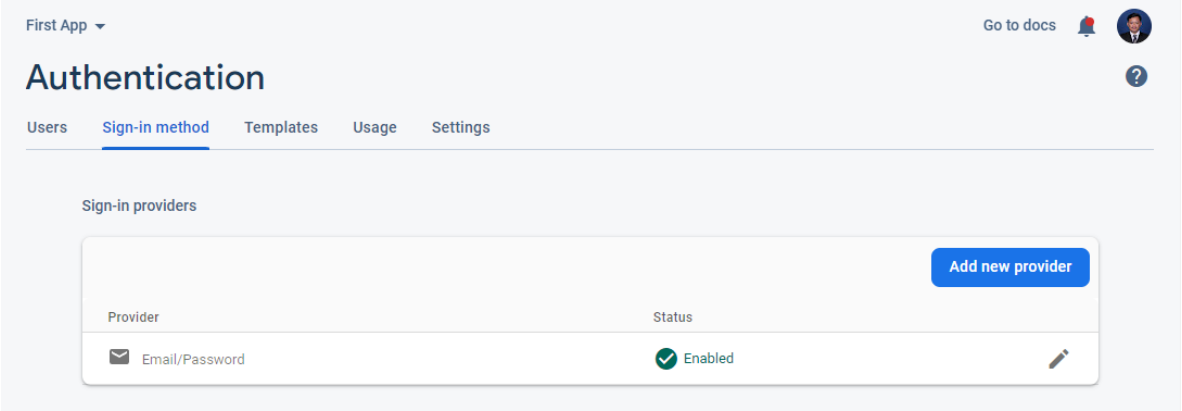
2. Masuk ke menu **Build > Authentication > Sign-in method** untuk memilih metode otentikasi aplikasi Anda. Secara default semua metode otentikasi yang disediakan di sini statusnya adalah **Disable**.



3. Di sini Anda akan belajar menggunakan metode otentikasi yang sangat umum yaitu menggunakan email dan password. Untuk itu silahkan pilih metode **Email/Password** lalu pilih **Enable** untuk mengaktifkan metode yang Anda pilih.



Setelah metode otentikasi diaktifkan maka tampilan dari **Sign-in method** sebelumnya akan menampilkan metode-metode otentikasi yang Anda aktifkan.



Sampai di tahap ini Anda telah siap untuk mulai menggunakan fitur Firebase Authentication pada project Flutter Anda.

Penggunaan Firebase Authentication

1. Lakukan konfigurasi pada **pubspec.yaml**, dengan menambahkan library-library yang akan Anda pergunakan yaitu:

```
firebase_core: ^2.1.1
firebase_auth: ^4.1.0
percent_indicator: ^4.2.2
flutter_login: ^4.0.0
font_awesome_flutter: ^10.2.1
```

Adapun penjelasan dari masing-masing library di atas adalah:

- Firebase core: library untuk menggunakan Firebase API
- Firebase auth: library untuk menggunakan fitur Authentication Firebase
- Percent indikator: library untuk menampilkan progress pada Flutter
- Flutter_login : library yang menyediakan tampilan lengkap untuk proses login, register sampai dengan recovery password.
- Font_awesome_flutter : Library yang berisikan Ikon dan Font gratis

2. Buat sebuah file dart baru bernama **loginScreen.dart**. Nantinya file ini akan Anda pergunakan untuk menampilkan tampilan login maupun sign in dengan menerima masukan berupa email dan password. Untuk membantu merancang tampilan ini, anda dapat menggunakan library **flutter_login**. Library ini memberikan widget berupa **FlutterLogin** yang telah dipersiapkan untuk bekerja dengan proses login pada sebuah aplikasi.

```dart
import 'package:flutter/material.dart';
import 'package:flutter_login/flutter_login.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';

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

  @override
  State<LoginScreen> createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  @override
  Widget build(BuildContext context) {
    return FlutterLogin(
      onLogin: _loginUser,
      onRecoverPassword: _recoverPassword,
      onSignup: _onSignup,
      passwordValidator: (value) {
        if (value != null) {
          if (value.length < 6) {
            return "Password Must Be 6 Characters";
          }
        }
      },
      loginProviders: <LoginProvider>[
        LoginProvider(
          icon: FontAwesomeIcons.google,
          label: 'Google',
          callback: _onLoginGoogle,
        ),
      ],
      onSubmitAnimationCompleted: () {},
    );
  }

  Future<String?>? _loginUser(LoginData data) {
    return null;
  }

  Future<String>? _recoverPassword(String name) {
    return null;
  }

```

```
45   Future<String?>? _onSignup(SignupData data) {
46     return null;
47   }
48
49   Future<String?>? _onLoginGoogle() {
50     return null;
51   }
52 }
53
```

Property dari widget **FlutterLogin** adalah:

- onLogin **(required)**: digunakan untuk memproses login dengan mengirimkan sebuah fungsi Future dengan parameter **LoginData** yang berisikan data **user name/email** dan **password** yang diisikan oleh user pada proses login.
- onRecoverPassword **(required)**: digunakan untuk memproses lupa sandi dengan mengirimkan sebuah fungsi Future dengan parameter **String** yang berisikan data **user name/email** diisikan oleh user pada lupa sandi.
- onSignUp **(required)**: digunakan untuk memproses pendaftaran dengan mengirimkan sebuah fungsi Future dengan parameter **SignupData** yang berisikan data **name**, **password** dan **termsOfService** yang diisikan oleh user pada proses pendaftaran.
- **passwordValidator** : digunakan untuk menentukan regex dari kata sandi yang diterima
- **loginProviders** : digunakan untuk, menyediakan tombol tambahan Ketika user ingin melakukan login menggunakan mekanisme **kredensial** seperti Google, Facebook, Twitter, Microsoft, dan lainnya.
- **onSubmitAnimationCompleted** : Merupakan listener yang dijalankan ketika pengguna menekan tombol **submit**

Anda dapat mempelajari properti lainnya pada https://pub.dev/packages/flutter_login

3. Buat sebuah file baru bernama **home.dart**. Nantinya file ini akan Anda pergunakan untuk menampilkan data user yang berhasil login.

```dart
import 'package:flutter/material.dart';
import 'package:flutter_auth_1/Praktek/auth.dart';
import 'package:flutter_login/flutter_login.dart';

import 'loginscreen.dart';

class MyHome extends StatefulWidget {
  final String wid;
  const MyHome({super.key, required this.wid});

  @override
  State<MyHome> createState() => _MyHomeState();
}

class _MyHomeState extends State<MyHome> {
  String? email;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(actions: [
        IconButton(
          icon: const Icon(Icons.logout_sharp),
          tooltip: 'Logout',
          onPressed: () {},
        ),
      ]),
      body: Center(
          child: Column(
        crossAxisAlignment: CrossAxisAlignment.center,
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text("Welcome $email"),
          Text("ID ${widget.wid}"),
        ],
      )),
    );
  }
}

```

4. Terakhir buat sebuah file **auth.dart** yang berisi class **AuthFirebase**. Pada class ini Anda akan menyimpan setiap fungsi yang nantinya akan Anda butuhkan dalam menggunakan Firebase Authentication. Lakukan inisiasi variabel **FirebaseAuth** sehingga Anda dapat menggunakan seluruh instance dari **FirebaseAuth**.

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

class AuthFirebase {
  final FirebaseAuth _firebaseAuth = FirebaseAuth.instance;
```

5. Buat sebuah fungsi Future untuk melakukan pendaftaran (sign up) menggunakan fungsi **FirebaseAuth** yaitu **createUserWithEmailAndPassword()** dengan menerima parameter berupa email dan password seperti di bawah ini.

```dart
  Future<String?> signUp(String email, String password) async {
    UserCredential authResult = await _firebaseAuth
        .createUserWithEmailAndPassword(email: email, password: password);
    User? user = authResult.user;
    return user?.uid;
  }

```

Fungsi tersebut akan membuat (menambahkan) user baru dengan email dan password yang diberikan pada Firebase Authentication. Jika pendaftaran berhasil maka Firebase akan mengembalikan data berupa data **User Unique ID**.

6. Setelah membuat fungsi pendaftaran, saatnya Anda membuat fungsi untuk masuk (login) menggunakan fungsi **FirebaseAuth** yaitu **signInUserWithEmailAndPassword()** dengan menerima parameter berupa email dan password seperti di bawah ini.

```dart
  Future<String?> login(String email, String password) async {
    try {
      UserCredential authResult = await _firebaseAuth
          .signInWithEmailAndPassword(email: email, password: password);
      User? user = authResult.user;
      return user?.uid;
    } catch (e) {
      print(e.toString());
      return null;
    }
  }

```

Fungsi di atas akan melakukan pengecekan berdasarkan email dan password yang diberikan pada Firebase Authentication. Jika otentikasi berhasil maka Firebase akan mengembalikan data berupa data **User Unique ID**.

7. Setelah user login, kita dapat mengambil data user yang sedang login dengan membuat sebuah fungsi lain seperti di bawah ini.

```
27  Future<User?> getUser() async {
28    User? user = await _firebaseAuth.currentUser;
29    return user;
30  }
```

Fungsi tersebut akan mengambil data user yang sedang login pada Firebase dan mengembalikan data berupa data **User**. Jika tidak ada user yang sedang login, maka Firebase akan mengembalikan data **null**.

8. Pada **Login Screen**, lakukan pengecekan apakah ada user yang telah login atau tidak pada aplikasi Anda dengan memanggil fungsi **getUser()** yang telah kita buat sebelumnya pada class **AuthFirebase**. Gunakan fungsi **getUser()** pada **initState**. Jika seandainya user telah login, maka user tersebut akan langsung diarahkan pada **Home Screen** sedangkan jika belum maka user tersebut akan diarahkan menuju **Login Screen**.

```
14 class _LoginScreenState extends State<LoginScreen> {
15   late AuthFirebase auth;
16
17   @override
18   void initState() {
19     auth = AuthFirebase();
20     auth.getUser().then((value) {
21       MaterialPageRoute route;
22       if (value != null) {
23         route = MaterialPageRoute(builder: (context) => MyHome(wid: value.uid));
24         Navigator.pushReplacement(context, route);
25       }
26     }).catchError((err) => print(err));
27   }
```

9. Pada **LoginScreen** tambahkan isi fungsi **_loginUser()** untuk melakukan fungsi masuk ke dalam aplikasi (login) menggunakan fungsi pada kelas **AuthFirebase** yaitu **login**(). Jika proses berhasil maka arahkan ke **Home Screen.** Jika gagal, maka pastikan anda memberikan pesan error (pada contoh ini kita menggunakan **snackBar**) dan mengarahkan kembali user ke **LoginScreen()**

```
Future<String?> _loginUser(LoginData data) {
  return auth.login(data.name, data.password).then((value) {
    if (value != null) {
      MaterialPageRoute(builder: (context) => MyHome(wid: value));
    } else {
      final snackBar = SnackBar(
        content: const Text('Login Failed, User Not Found'),
        action: SnackBarAction(
          label: 'OK',
          onPressed: () {
            // Some code to undo the change.
          },
        ),
      );
      ScaffoldMessenger.of(context).showSnackBar(snackBar);
      Navigator.of(context).pushReplacement(
          MaterialPageRoute(builder: (context) => LoginScreen()));
    }
  });
}
```

10. Berikutnya pada **LoginScreen** tambahkan isi fungsi **_onSignUp()** untuk melakukan fungsi pendaftaran ke dalam aplikasi (sign up) menggunakan fungsi pada kelas **AuthFirebase** yaitu **signUp**(). Jika proses berhasil maka arahkan ke **Home Screen** dan berikan pesan sukses (pada contoh ini kita menggunakan **snackBar**).

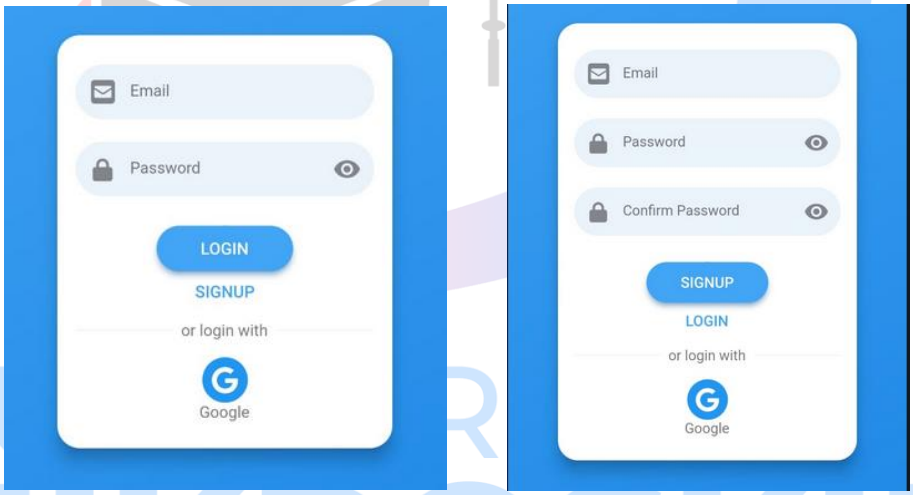
```
Future<String?> _onSignup(SignupData data) {
  return auth.signUp(data.name!, data.password!).then((value) {
    if (value != null) {
      final snackBar = SnackBar(
        content: const Text('Sign Up Successful'),
        action: SnackBarAction(
          label: 'OK',
          onPressed: () {
            // Some code to undo the change.
          },
        ),
      );
      ScaffoldMessenger.of(context).showSnackBar(snackBar);
    }
  });
}
```

11. Untuk memproses button **submit** pada **FlutterLogin,** anda akan mengecek, apakah user telah ada atau tidak menggunakan cara mengecek keberadaan user.

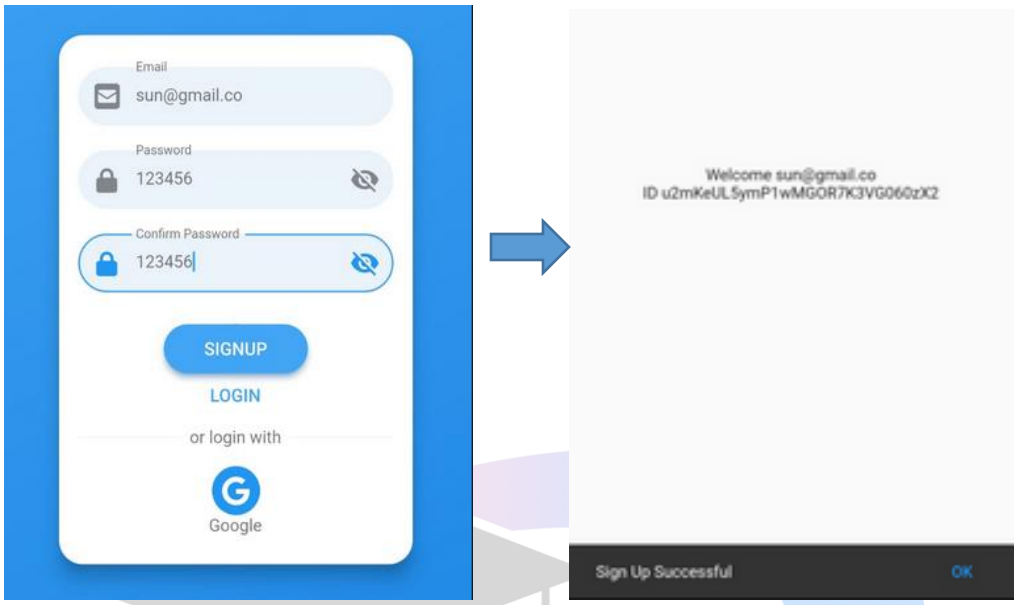
Jika user telah terotentikasi, maka arahkan user ke **Home Sceen,** jika tidak**,** maka kembalikan pada **Login Screen**

```
50      onSubmitAnimationCompleted: () {
51        auth.getUser().then((value) {
52          MaterialPageRoute route;
53          if (value != null) {
54            route =
55                MaterialPageRoute(builder: (context) => MyHome(wid: value.uid));
56          } else {
57            route = MaterialPageRoute(builder: (context) => LoginScreen());
58          }
59          Navigator.pushReplacement(context, route);
60        }).catchError((err) => print(err));
61      },
```

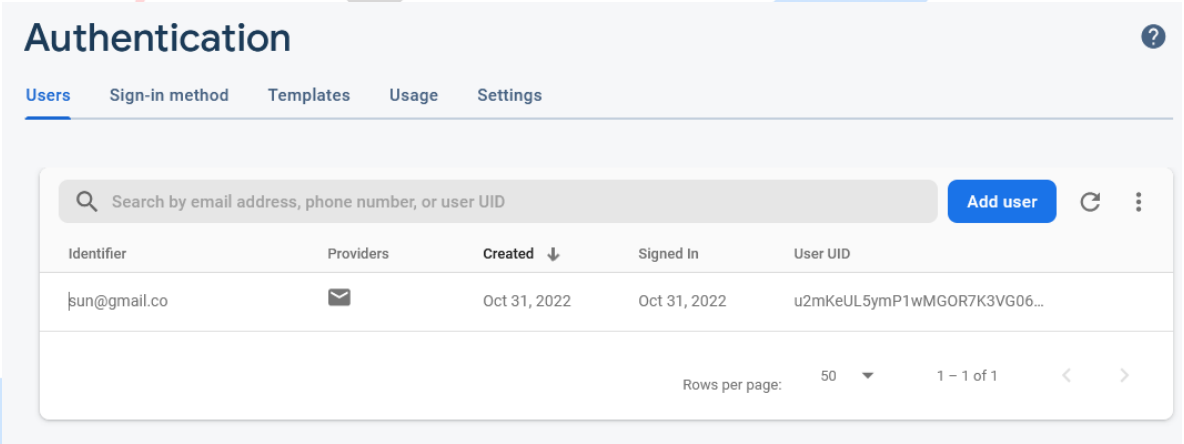
Jalankan aplikasi, dan lihat hasilnya, anda telah berhasil membuat otentikasi pengguna pada aplikasi anda.



Lakukanlah proses sign up dengan email dan password anda, dan jika berhasil anda akan langung di arahkan pada home sceen anda.



Anda juga dapat melihat user yang telah aktif di aplikasi anda melalui **console firebase** pada bagian **Authentication >> Users**



Dokumentasi lebih lengkap untuk penggunaan Firebase Authentication dapat Anda baca pada link berikut https://firebase.google.com/docs/auth/flutter/start.

Latihan

1. Setelah membuat proyek yang terhubung dengan Firebase Authentication, tambahkan sebuah fitur untuk melakukan Sign Out sehingga user dapat keluar dari aplikasi.

2. Lakukan Otentikasi menggunakan kredensial seperti menggunakan Facebook, Google, atau lainnya.