

LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK



JUDUL:

Authentikasi Firebase

Disusun oleh:

Faiq Mufrih Hakim (21102245)

TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024

Pembahasan



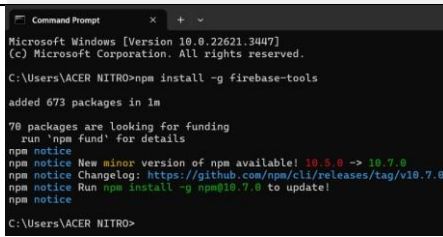
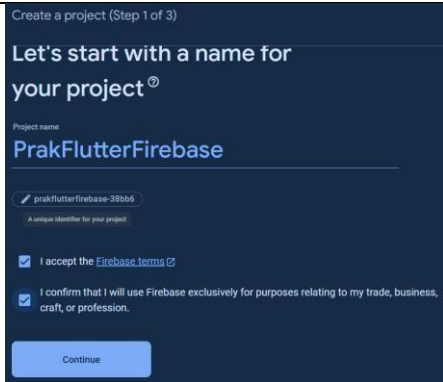


Firestore

Firestore Authentication adalah layanan yang disediakan oleh Google Firestore untuk mengelola otentikasi pengguna di aplikasi Anda. Dengan menggunakan Firestore Authentication, pengembang dapat dengan mudah menambahkan fitur otentikasi menggunakan email dan kata sandi, nomor telepon, serta metode otentikasi berbasis identitas pihak ketiga seperti Google, Facebook, dan Twitter. Firestore menyediakan SDK yang memungkinkan integrasi otentikasi menjadi lebih sederhana dan konsisten di berbagai platform seperti web, iOS, dan Android. Hal ini mengurangi beban pengembang dalam mengelola infrastruktur otentikasi dan keamanan pengguna secara manual.

Pada inti operasinya, Firestore Authentication menggunakan token ID berbasis JSON Web Token (JWT) untuk mengidentifikasi pengguna secara aman. Setelah pengguna berhasil masuk, Firestore Authentication mengeluarkan token ID yang mencakup informasi identitas pengguna. Token ini kemudian dapat digunakan untuk mengautentikasi permintaan yang dibuat oleh aplikasi ke server backend. Firestore Authentication juga menyediakan kemampuan untuk mengelola pengguna seperti membuat pengguna baru, menghapus pengguna, memperbarui informasi pengguna, dan mengelola kata sandi.

Keunggulan utama dari Firestore Authentication adalah kemudahan integrasi dan keamanan yang disediakan. Firestore menangani banyak aspek kompleks dari otentikasi pengguna seperti penyimpanan data pengguna dengan aman, perlindungan terhadap serangan keamanan seperti brute force, dan pemulihan akun pengguna yang lupa kata sandi. Selain itu, Firestore Authentication terintegrasi dengan layanan Firestore lainnya seperti Firestore dan Realtime Database, memungkinkan pengembang untuk dengan mudah membangun aplikasi yang aman dan skalabel dengan otentikasi pengguna yang solid.

Langkah-langkah Praktikum

Langkah Praktikum	Pembahasan									
	Membuat akun di Firebase website									
<p>Windows</p> <p>Anda dapat menginstal Firebase CLI untuk Windows menggunakan salah satu opsi berikut:</p> <table><thead><tr><th>Opsi</th><th>Deskripsi</th><th>Disarankan untuk...</th></tr></thead><tbody><tr><td>biner mandiri</td><td>Download biner mandiri untuk CLI. Selanjutnya, Anda dapat mengimpor file yang dapat dieksekusi untuk membuka shell guna menjalankan perintah Firebase.</td><td>Developer baru Developer yang tidak menggunakan atau belum familiar dengan Node.js</td></tr><tr><td>npm</td><td>Gunakan npm (Pengelola Paket Node) untuk menginstal CLI dan mengaktifkan perintah Firebase yang tersedia secara global.</td><td>Developer yang menggunakan Node.js</td></tr></tbody></table>	Opsi	Deskripsi	Disarankan untuk...	biner mandiri	Download biner mandiri untuk CLI. Selanjutnya, Anda dapat mengimpor file yang dapat dieksekusi untuk membuka shell guna menjalankan perintah Firebase.	Developer baru Developer yang tidak menggunakan atau belum familiar dengan Node.js	npm	Gunakan npm (Pengelola Paket Node) untuk menginstal CLI dan mengaktifkan perintah Firebase yang tersedia secara global.	Developer yang menggunakan Node.js	Mengunduh firebase CLI dan NodeJs
Opsi	Deskripsi	Disarankan untuk...								
biner mandiri	Download biner mandiri untuk CLI. Selanjutnya, Anda dapat mengimpor file yang dapat dieksekusi untuk membuka shell guna menjalankan perintah Firebase.	Developer baru Developer yang tidak menggunakan atau belum familiar dengan Node.js								
npm	Gunakan npm (Pengelola Paket Node) untuk menginstal CLI dan mengaktifkan perintah Firebase yang tersedia secara global.	Developer yang menggunakan Node.js								
	Install node.js									
	Instal Firebase CLI melalui npm dengan menjalankan perintah berikut: npm install -g firebase-tools									
	Membuat Project baru di firebase.									
	Menambahkan flutter ke firebase.									
	Langkah pertama dalam menambahkan flutter ke firebase									

<pre> Future<void> main() async { WidgetsFlutterBinding.ensureInitialized(); await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform,); runApp(const MyApp()); } </pre>	<p>Mengubah void main</p>
	<p>Membuat kode untuk mengimplementasikan desain antarmuka pengguna pada gambar di samping.</p>
	<p>Membuat folder baru yang diberi nama “ui”, dan di dalam folder tersebut buatlah 4 file dart sesuai pada gambar di samping.</p>
<pre> import 'dart:async'; import 'package:flutter/material.dart'; import 'package:flutter/widgets.dart'; class SplashScreen extends StatefulWidget { const SplashScreen({Key? key}) : super(key: key); @override State<SplashScreen> createState() => _SplashScreenState(); } class _SplashScreenState extends State<SplashScreen> { @override void initState() { Timer(Duration(seconds: 3), () => Navigator.pushNamed(context, '/login')); super.initState(); } @override Widget build(BuildContext context) { return Scaffold(body: Center(child: Column(mainAxisAlignment: MainAxisAlignment.center, children: [Text("Praktikum", style: TextStyle(fontSize: 40, fontWeight: FontWeight.bold, color: Color(0xff3D4DE0), // TextStyle), // Text), // Text Text("PPB", style: TextStyle(fontSize: 40, fontWeight: FontWeight.bold, color: Color(0xff3D4DE0), // TextStyle), // Text), // Column], // Center), // Scaffold),); } } </pre>	<p>Tambahkan kode di samping pada file splash.dart</p>

```

lib > ui > login.dart > ...
DOKULAH SEMESTER 01/PEMROGRAMAN PERANGKAT
BERGURUKA/authlibbase/praktikumauthlib
3 class LoginScreen extends StatefulWidget {
4   const LoginScreen({key? key}) : super(key: key);
5   @override
6   State<LoginScreen> createState() => _LoginScreenState();
7 }
8
9 class _LoginScreenState extends State<LoginScreen> {
10   final emailEdc = TextEditingController();
11   final passEdc = TextEditingController();
12   bool passInvisible = false;
13   @override
14   Widget build(BuildContext context) {
15     return Scaffold(
16       body: Container(
17         margin: EdgeInsets.symmetric(horizontal: 30, vertical: 70),
18         child: ListView(
19           children: [
20             Text(
21               "Login",
22               style: TextStyle(
23                 fontSize: 40,
24                 fontWeight: FontWeight.bold,
25                 color: Color(0xff3D4DE0), // TextStyle
26               ), // Text

```

```

27             SizedBox(
28               height: 15,
29             ), // SizedBox
30             Text(
31               "Silahkan masukan e-mail dan password anda",
32               style: TextStyle(
33                 fontSize: 16,
34               ), // TextStyle
35             ), // Text
36             SizedBox(
37               height: 25,
38             ), // SizedBox
39             Text(
40               "e-mail",
41               style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
42             ), // Text
43             TextFormField(
44               controller: emailEdc,
45             ), // TextFormField
46             SizedBox(
47               height: 10,
48             ), // SizedBox
49             Text(
50               "password",

```

```

51             ), // Text
52             TextFormField(
53               controller: passEdc,
54               decoration: InputDecoration(
55                 suffixIcon: IconButton(
56                   icon: Icon(
57                     passInvisible ? Icons.visibility : Icons.visibility_off,
58                     onPressed: () {
59                       setState(() {
60                         passInvisible = !passInvisible;
61                       });
62                     }, // IconButton
63                   ), // InputDecoration
64                 obscureText: !passInvisible,
65               ), // TextFormField
66             ), // SizedBox
67             ElevatedButton(
68               onPressed: () {
69                 Navigator.pushNamed(context, '/home');
70               },

```

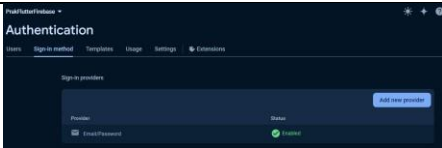
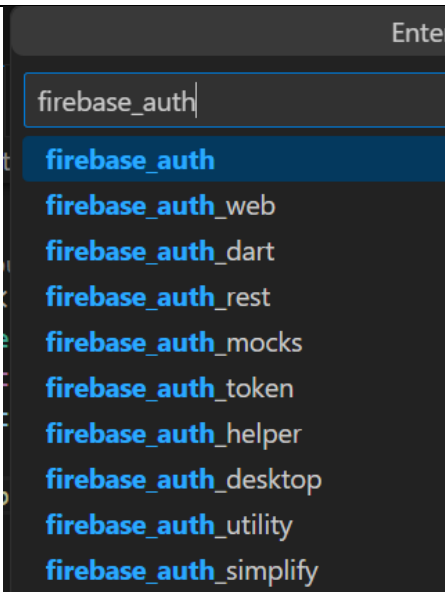
```

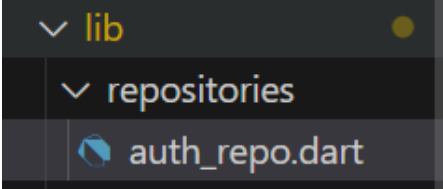
71             ), // Text
72             Row(
73               mainAxisAlignment: MainAxisAlignment.center,
74               children: [
75                 Text("Belum punya akun?"),
76                 TextButton(
77                   onPressed: () {
78                     Navigator.pushNamed(context, '/register');
79                   },
80                   child: Text(
81                     "Daftar",
82                     style: TextStyle(
83                       fontWeight: FontWeight.bold,
84                       color: Color(0xff3D4DE0), // TextStyle
85                     ), // Text // TextButton
86                 ), // Row
87               ], // ListView
88             ), // Container
89           ], // Scaffold
90         ),

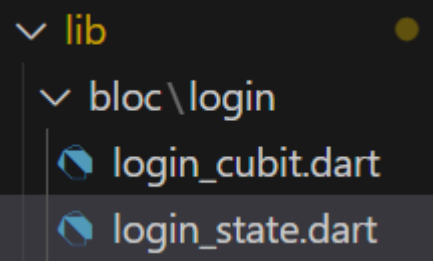
```

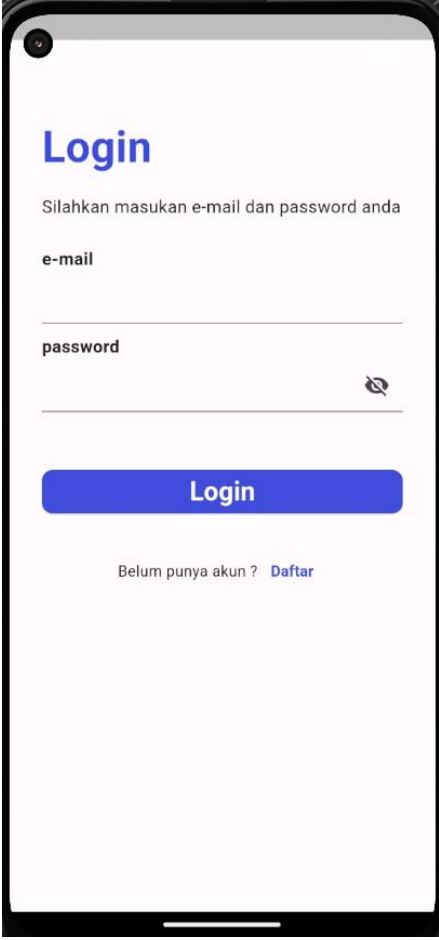

Tambahkan kode di samping pada bagian file login.dart

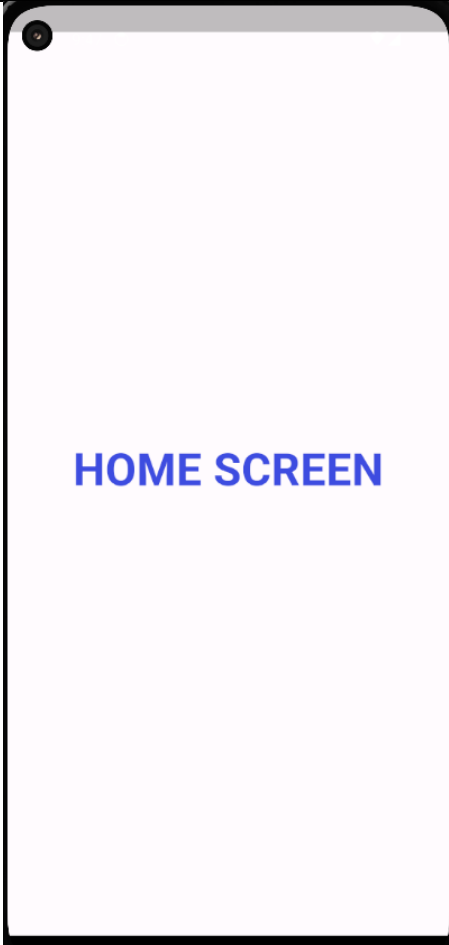
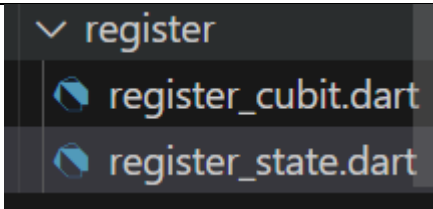
<pre>import 'package:flutter/material.dart'; class RegisterScreen extends StatefulWidget { const RegisterScreen({Key? key}) : super(key: key); @override State<RegisterScreen> createState() => _RegisterScreenState(); } class _RegisterScreenState extends State<RegisterScreen> { final emailEdc = TextEditingController(); final passEdc = TextEditingController(); bool passInvisible = false; @override Widget build(BuildContext context) { return Scaffold(body: Container(margin: EdgeInsets.symmetric(horizontal: 30, vertical: 70), child: ListView(children: [Text("Register", style: TextStyle(fontSize: 40, fontWeight: FontWeight.bold, color: Color(0xff3D4DE0)), // TextStyle), // Text TextFormField(controller: passEdc, decoration: InputDecoration(suffixIcon: IconButton(icon: Icon(passInvisible ? Icons.visibility : Icons.visibility_off), onPressed: () { setState(() { passInvisible = !passInvisible; // Toggle isPasswordVisible ketika }); }, // IconButton), // InputDecoration obscureText: !passInvisible, // Atur obscureText berdasarkan _isPassword), // TextFormField SizedBox(height: 50,), // SizedBox ElevatedButton(onPressed: () {},), // ElevatedButton SizedBox(height: 25,), // SizedBox Row(mainAxisAlignment: MainAxisAlignment.center, children: [Text("Sudah punya akun ?"), TextButton(onPressed: () { Navigator.pushNamed(context, '/login'); }, child: Text("Login", style: TextStyle(fontWeight: FontWeight.bold, color: Color(0xff3D4DE0)), // TextStyle), // Text // TextButton], // Row), // ListView], // Container); // Scaffold)); } }</pre>	<p>Tambahkan kode di samping pada bagian file register.dart</p>
<pre>import 'package:flutter/material.dart'; class HomeScreen extends StatelessWidget { const HomeScreen({Key? key}) : super(key: key); @override Widget build(BuildContext context) { return Scaffold(body: Center(child: Text("HOME SCREEN", style: TextStyle(fontSize: 40, fontWeight: FontWeight.bold, color: Color(0xff3D4DE0)), // TextStyle), // Text // Center); // Scaffold); } }</pre>	<p>Tambahkan kode di samping pada bagian home_screen.dart</p>
	<p>Selanjutnya kita perlu mendaftarkan state yang sudah kita buat dalam route dengan cara buat folder “utils” di</p>

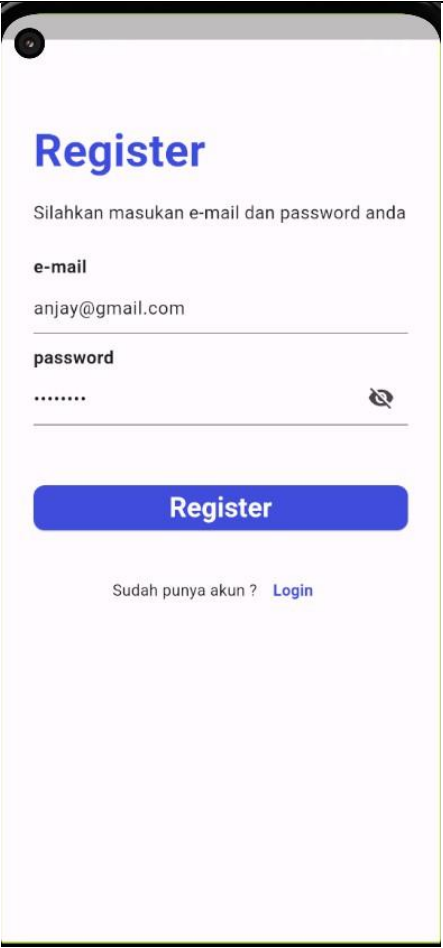

		dalam folder "lib", dan buat file "routes.dart" di dalam folder "utils".
<pre> MaterialPageRoute _pageRoute({required Widget body, required RouteSettings settings}) => MaterialPageRoute(builder: (_) => body, settings: settings); Route? generateRoute(RouteSettings settings) { Route? _route; final _args = settings.arguments; switch (settings.name) { case rLogin: _route = _pageRoute(body: LoginScreen(), settings: settings); break; case rRegister: _route = _pageRoute(body: RegisterScreen(), settings: settings); break; case rHome: _route = _pageRoute(body: HomeScreen(), settings: settings); break; } return _route; } final NAV_KEY = GlobalKey<NavigatorState>(); const String rLogin = '/login'; const String rRegister = '/register'; const String rHome = '/home'; </pre>		Tambahkan kode di samping pada bagian file routes.dart
<pre> class MyApp extends StatelessWidget { const MyApp({Key? key}) : super(key: key); @override Widget build(BuildContext context) { return MaterialApp(title: "Praktikum 6", debugShowCheckedModeBanner: false, navigatorKey: NAV_KEY, generateRoute: generateRoute, home: SplashScreen(),); // MaterialApp } } </pre>		Tambahkan kode di samping pada bagian file main.dart
		Menambahkan autentikasi email dan password pada dashboard firebase di project kita.
		Menginstal dependensi firebase_auth

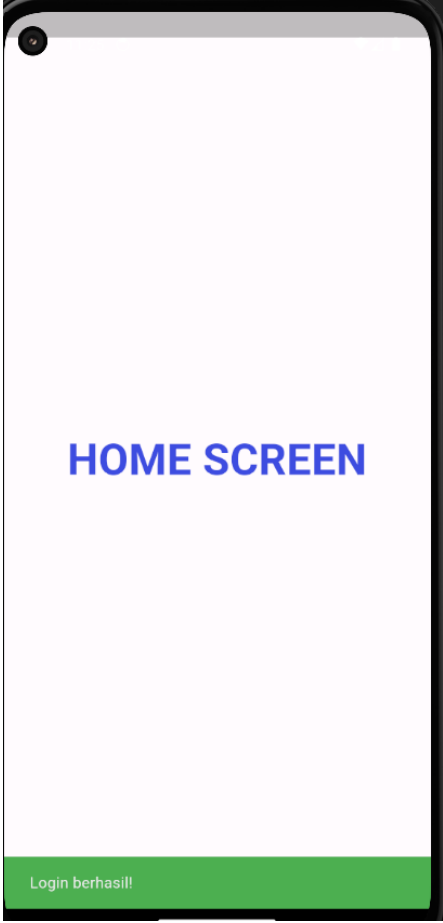
<pre>cupertino_icons: ^1.0.6 firebase_core: ^2.30.1 firebase_auth: ^4.19.4 firebase_auth_rest: ^2.0.4</pre>	<p>Cek ketersediaan dependensi firebase_auth pada pubspec.yaml</p>
<pre>bloc: ^8.1.2 flutter_bloc: ^8.1.3</pre>	<p>Untuk memproses logika bisnis pada aplikasi kita akan menggunakan flutter bloc, sehingga perlu menambahkan 2 dependensi yaitu :</p> <p>bloc: ^8.1.2</p> <p>flutter_bloc: ^8.1.3</p> <p>jangan lupa untuk menjalankan perintah “flutter pub upgrade”</p>
	<p>Buat folder “repositories” pada folder “lib”, kemudian buat file “auth_repo.dart”.</p> <p>folder repositories ini akan kita gunakan untuk berkomunikasi dengan firebase/API dari back end.</p>
<pre>import 'package:firebase_auth/firebase_auth.dart'; import 'package:firebase_core/firebase_core.dart'; class AuthRepo { final _auth = FirebaseAuth.instance; Future<void> login((required String email, required String password)) async { try { final user = await _auth.signInWithEmailAndPassword(email: email, password: password); } on FirebaseAuthException catch (e) { throw e.message ?? 'Something wrong!'; } catch (e) {} throw e; } Future<void> register({required String email, required String password}) async { try { final user = await _auth.createUserWithEmailAndPassword(email: email, password: password); } on FirebaseAuthException catch (e) { throw e.message ?? 'Something wrong!'; } catch (e) { throw e; } } }</pre>	<p>Kita akan membuat 2 buah Future<void> dalam class AuthRepo untuk login dan register pada file auth_repo.dart.</p>

	<p>Kemudian kita akan mengatur bloc untuk login. Buat folder dengan struktur seperti gambar di samping.</p>
<pre>part of 'login_cubit.dart'; import 'package:flutter/material.dart'; @immutable abstract class LoginState {} class LoginInitial extends LoginState {} class LoginLoading extends LoginState {} class LoginSuccess extends LoginState { final String msg; LoginSuccess(this.msg); } class LoginFailure extends LoginState { final String msg; LoginFailure(this.msg); }</pre>	<p>Tambahkan kode di samping pada file login_state.dart</p>
<pre>import 'package:bloc/bloc.dart'; import 'package:meta/meta.dart'; import '../repositories/auth_repo.dart'; part 'login_state.dart'; class LoginCubit extends Cubit<LoginState> { LoginCubit() : super(LoginInitial()); final _repo = AuthRepo(); void login({required String email, required String password}) async { emit(LoginLoading()); try { _repo.login(email: email, password: password); emit(LoginSuccess('Login berhasil!')); } catch (e) { print(e); emit(LoginFailure(e.toString())); } } }</pre>	<p>Tambahkan kode di samping pada file login_cubit.dart</p>

	<p>Jalankan program. inputan email dan password akan disimpan pada emailEdc dan passEdc kemudian dengan menggunakan bloc inputan tersebut dikirim ke firebase melalui repositori yang sudah kita buat, apabila berhasil maka akan masuk ke home screen, apabila gagal akan muncul pesan gagal.</p>
	<p>Mengatur untuk bagian register di firebase</p>

	<p>Data yang baru ditambahkan, kita cek untuk mengisi pada bagian login.</p>
	<p>Login sudah berhasil kita buat, maka selanjutnya kita akan membuat bagian register.</p> <p>Karena kita sudah membuat Future<void> register. Selanjutnya kita atur untuk blocnya. Buat folder “register” di dalam folder bloc.</p>
<pre> part of 'register_cubit.dart'; import 'package:flutter/material.dart'; @immutable abstract class RegisterState {} class RegisterInitial extends RegisterState {} class RegisterLoading extends RegisterState {} class RegisterSuccess extends RegisterState { final String msg; RegisterSuccess(this.msg); } class RegisterFailure extends RegisterState { final String msg; RegisterFailure(this.msg); } </pre>	<p>Tambahkan kode di samping pada bagian file register_state.dart</p>

<pre>import 'package:bloc/bloc.dart'; import 'package:meta/meta.dart'; import '../repositories/auth_repo.dart'; part 'register_state.dart'; class RegisterCubit extends Cubit<RegisterState> { RegisterCubit() : super(RegisterInitial()); final _repo = AuthRepo(); Future<void> register({required String email, required String password}) async { emit(RegisterLoading()); try { await _repo.register(email: email, password: password); emit(RegisterSuccess('Berhasil!')); } catch (e) { print(e); emit(RegisterFailure(e.toString())); } } }</pre>	<p>Tambahkan kode di samping pada bagian file register_cubit.dart</p>
	<p>Mencoba register</p>
	<p>User yang baru didaftarkan berhasil ditambahkan</p>

	<p>Mencoba login dengan akun yang baru saja di tambahkan</p>
--	--