

VERSION 1.0

AUGUST, 2023



PEMROGRAMAN MOBILE

FIREBASE AUTHENTICATION DAN MESSAGING – MODUL 4 MATERI

TIM PENYUSUN:

- DIDIH RIZKI CHANDRANEGARA, S.KOM., M.KOM.
- MUHAMMAD ZULFIQOR LILHAQ
- RIYAN PUTRA FIRJATULLAH

PRESENTED BY: LAB. INFORMATIKA UNIVERSITAS MUHAMMADIYAH MALANG

CAPAIAN PEMBELAJARAN PRAKTIKUM

1. Mahasiswa dapat memahami penggunaan Firebase pada Framework Flutter.
 2. Mahasiswa dapat memahami penggunaan Firebase Authentication pada aplikasi Framework Flutter..
 3. Mahasiswa dapat memahami penggunaan Firebase Messaging pada aplikasi Framework Flutter.
-

SUB CAPAIAN PEMBELAJARAN PRAKTIKUM

1. Mahasiswa dapat mengimplementasikan Firebase Authentication pada aplikasi menggunakan Framework Flutter.
 2. Mahasiswa dapat mengimplementasikan Firebase Messaging pada aplikasi menggunakan Framework Flutter.
-

KEBUTUHAN HARDWARE & SOFTWARE

1. PC/Laptop
 2. IDE Android Studio/ Visual Studio Code
 3. Flutter SDK: <https://docs.flutter.dev/release/archive?tab=windows>
-

MATERI POKOK

Pengertian Firebase

Firebase adalah platform pengembangan aplikasi mobile dan web yang dikelola oleh Google. Ini menyediakan berbagai layanan dan alat yang membantu pengembang dalam membangun, mengelola, dan mengembangkan aplikasi yang kuat dengan lebih cepat dan efisien. Firebase mencakup berbagai fitur seperti penyimpanan data, autentikasi pengguna, analitik, pelacakan kinerja, dan banyak lagi. Beberapa fitur dan layanan utama yang ditawarkan oleh Firebase meliputi:

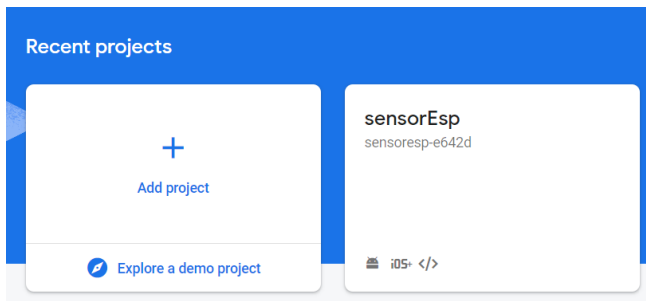
1. **Realtime Database**: Database cloud yang menyimpan data dalam format JSON dan memungkinkan sinkronisasi data secara realtime antara perangkat.
2. **Firestore**: Layanan database cloud yang memberikan skalabilitas tinggi dan struktur koleksi-dokumen yang fleksibel.
3. **Authentication**: Memungkinkan aplikasi untuk mengintegrasikan sistem autentikasi pengguna seperti login dengan email, Google, Facebook, dll.
4. **Storage**: Layanan penyimpanan file yang memungkinkan Anda menyimpan dan mengelola gambar, video, dan file lainnya.
5. **Cloud Functions**: Memungkinkan Anda menulis dan menjalankan kode server tanpa mengelola infrastrukturnya.
6. **Cloud Messaging**: Layanan untuk mengirim notifikasi push ke pengguna.

7. **Crashlytics**: Layanan pelaporan kesalahan dan penanganan crash yang membantu Anda mengidentifikasi masalah dalam aplikasi Anda.
8. **Performance Monitoring**: Memberikan wawasan tentang kinerja aplikasi dan waktu muat.
9. **Analytics**: Membantu Anda memahami bagaimana pengguna berinteraksi dengan aplikasi Anda dan memberikan wawasan tentang kinerja.
10. **Hosting**: Memungkinkan Anda untuk meng-host dan menerbitkan situs web dan aplikasi web Anda.
11. **Machine Learning**: Menawarkan layanan pembelajaran mesin yang dapat diintegrasikan dengan aplikasi Anda.

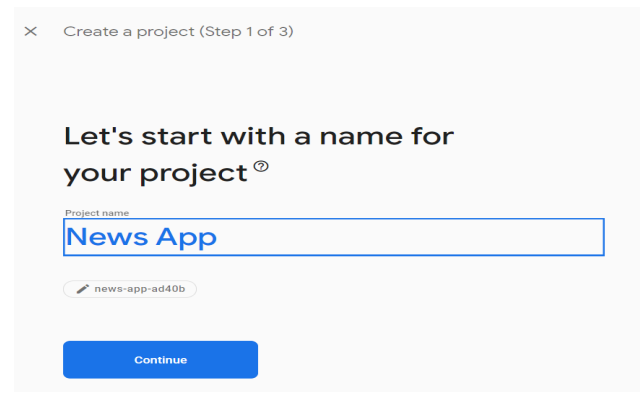
Setup Firebase

Pada setup Firebase kita bisa mengunjungi terlebih dahulu website [Firebase](#) dan login dengan akun Google kita seperti biasa, setelah itu kita bisa klik **Mulai**.

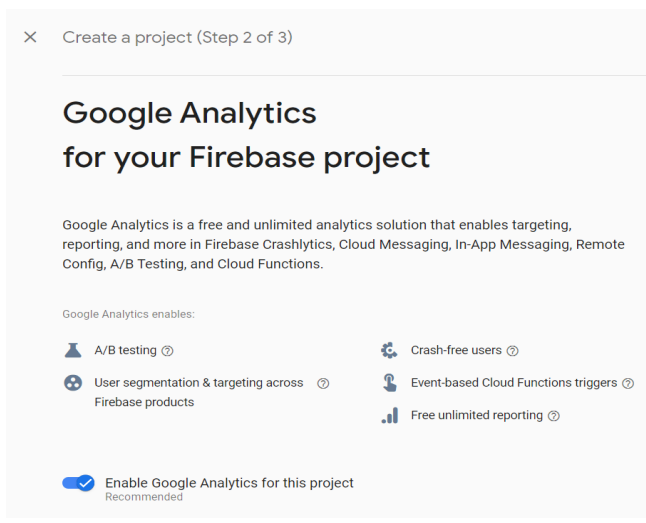
1. Pada halaman Firebase Console, buat proyek baru dengan klik **Add Project**.



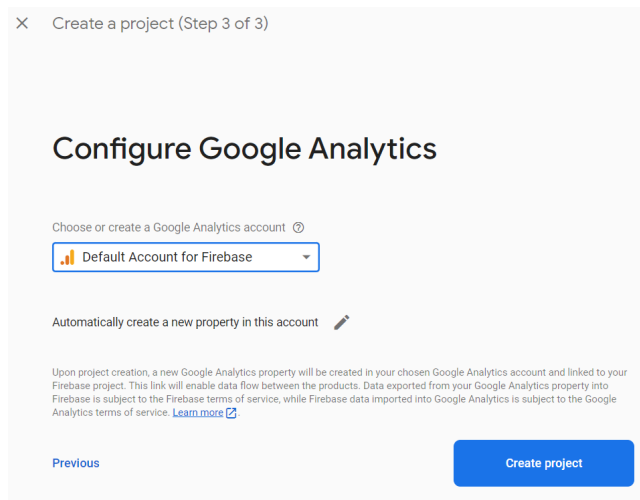
2. Masukkan nama project yang Anda inginkan. Contohnya seperti **News App**. Lalu klik **Continue**.



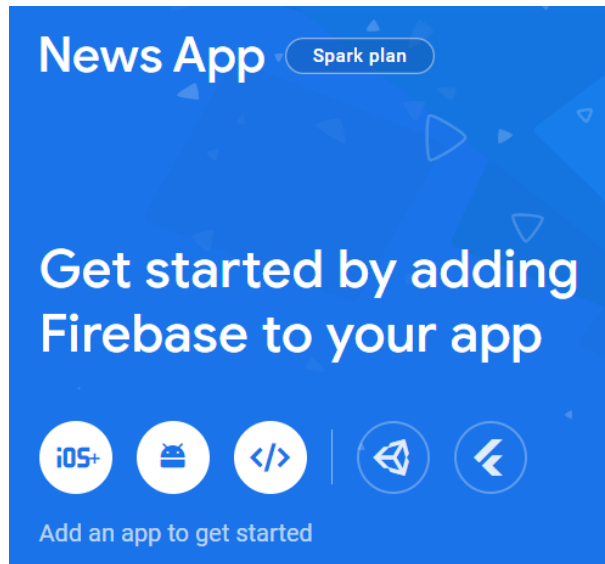
3. Pada bagian ini kita bisa langsung klik **Continue** saja.



4. Selanjutnya pilih akun untuk mengakses Google Analytics. Kita bisa pilih **Default Account for Firebase**. Jika sudah, klik **Create Project**. Firebase akan menyiapkan proyek Anda dalam beberapa saat. Klik **Continue** ketika proyek Anda siap.



5. Setelah itu kita akan diarahkan ke halaman utama project Firebase. Pada bagian ini kita dapat mengatur layanan yang digunakan dan mengintegrasikan Firebase dengan proyek kita. Pada kali ini kita akan mengintegrasikan pada project Flutter, maka klik **logo Flutter**.



6. Setelah mengklik **logo Flutter**, kita akan diarahkan ke halaman mengenai langkah-langkah untuk mengintegrasikan/menambah Firebase ke dalam project Flutter. Kita bisa langsung mengikuti step yang sudah disediakan melalui dokumentasi yang sudah ada disana.

× Add Firebase to your Flutter app

1 Prepare your workspace

The easiest way to get you started is to use the FlutterFire CLI.

Before you continue, make sure to:

- Install the [Firebase CLI](#) and log in (run `firebase login`)
- Install the [Flutter SDK](#)
- Create a Flutter project (run `flutter create`)

Next

Install and run the FlutterFire CLI

Initialize Firebase and add plugins

7. Terdapat beberapa langkah-langkah yang bisa digunakan untuk menambahkan Firebase ke dalam project kita, jika ingin mengikuti cara ini kita harus menginstall [Node.js](#) terlebih dahulu di perangkat kita. Setelah menginstall Node.js, buka terminal pada direktori proyek Flutter, lalu instal Firebase CLI

```
npm install -g firebase-tools
```

Flag **-g** bertujuan untuk instal Firebase CLI secara global sehingga dapat berjalan di berbagai direktori proyek. Pastikan kita login dengan akun Google yang sesuai dengan proyek Firebase sebelumnya. **Apabila tidak sesuai**, Anda dapat lakukan **logout** dengan perintah di bawah ini dan login kembali.

```
firebase logout
```

8. Login ke Firebase dengan akun Google dengan perintah berikut.

```
firebase login
```

9. Berikutnya lakukan install FlutterFire CLI pada terminal. Tambahkan sub-perintah global agar Flutter CLI dapat bekerja di berbagai direktori proyek.

```
dart pub global activate flutterfire_cli
```

10. Pada direktori utama proyek Flutter Anda, lakukan konfigurasi Firebase melalui command line.

```
flutterfire configure
```

11. Setelah itu biasanya kita akan mendapatkan beberapa pertanyaan terkait konfigurasi Firebase pada project Flutter, kita bisa menjawab pertanyaannya seperti berikut:

Pertanyaan: Select a Firebase project to configure your Flutter applications with
Jawaban: Sesuaikan dengan nama project yang sudah kita buat sebelumnya (**News App**)

```
i Found 7 Firebase projects.
? Select a Firebase project to configure your Flutter application with
  coba-85ccc (coba)
  ditonton-11403 (ditonton)
  familyplus-df907 (familyplus)
  login-f3885 (google-login)
  news-app-ad40b (News App)
  reaboks-bbc0c (reaboks)
  sensorexp-e642d (sensorEsp)
  <create a new project>
```

Pertanyaan: Which platforms should your configuration support?
Jawaban: Pilih platform dengan menekan tombol spasi dan Enter jika sudah selesai memilih (**Android, IOS, Web**)

```
i Found 7 Firebase projects.
✓ Select a Firebase project to configure your Flutter application with news-app-ad40b (News App)
? Which platforms should your configuration support (use arrow keys & space to select)?
  ✓ android
  ✓ ios
  macos
  web
```

12. Tunggu sampai inisiasi Firebase berakhir, jika berhasil pada terminal kita akan menampilkan keterangan seperti berikut:

```
i Registered a new Firebase android app on Firebase project news-app-ad40b.
i Registered a new Firebase ios app on Firebase project news-app-ad40b.
i Registered a new Firebase web app on Firebase project news-app-ad40b.
```

13. Setelah proses konfigurasi Firebase, Anda mendapati beberapa berkas baru di dalam proyek Flutter, yaitu:

- **google-service.json**, file ini berada pada folder **android/app** dan berguna sebagai konfigurasi aplikasi Android dengan Firebase.
- **GoogleService-Info.plist**, file ini berada pada folder **ios/Runner** dan berguna sebagai konfigurasi aplikasi iOS dengan Firebase.
- **firebase_app_is_file.json**, file ini berada pada folder **ios** dan berguna sebagai sumber informasi Firebase dengan aplikasi iOS.
- **firebase_options.dart**, file ini berada pada folder **lib** dan berguna sebagai kelas yang mendefinisikan Firebase ke dalam proyek Flutter.

14. Selanjutnya kita konfigurasi pada Android, pertama masuk pada file **build.gradle** pada folder **android/app** dan ubah konfigurasi **minSdkVersion**.

```
android {  
    defaultConfig {  
        // ...  
        minSdkVersion 19  
    }  
}
```

15. Pada berkas **build.gradle** di folder **android**, tambahkan aturan untuk menyertakan plugin Google Services. Periksa juga apakah Anda sudah menyertakan repository Maven Google atau tidak.

```
buildscript {  
  
    repositories {  
        // Periksa apakah sudah menyertakan Maven Google atau tidak.  
        google() // Google's Maven repository  
    }  
  
    dependencies {  
        // ...  
  
        // Tambahkan baris di bawah ini.  
        classpath 'com.google.gms:google-services:4.3.13' // Google Services plugin  
    }  
}  
  
allprojects {  
    // ...  
  
    repositories {  
        // Periksa apakah sudah menyertakan Maven Google atau tidak.  
        google() // Google's Maven repository  
        // ...  
    }  
}
```

16. Selanjutnya, buka berkas **build.gradle** di dalam folder **android/app**. Tambahkan kode di bawah ini untuk menerapkan plugin Google Services.

```



apply plugin: 'com.android.application'
// Tambahkan baris di bawah ini.
apply plugin: 'com.google.gms.google-services' // Google Services plugin

android {
    // ...
}

```

Integrasi Project Flutter

Integrasi pertama dilakukan dengan menambahkan **firebase_core** plugin pada file **pubspec.yaml** [firebase_core](#).

- 1 Buka file **pubspec.yaml** pada project
 -  nama_project
 -  pubspec.yaml
- 2 Ubah dari code **sebelum** (kiri) menjadi **sesudah** (kanan) lalu **save**.

dependencies:

flutter:

 sdk: flutter

The following adds the Cupertino Icons font to your application.

Use with the CupertinoIcons class for iOS style icons.

cupertino_icons: ^1.0.2

dependencies:

flutter:

 sdk: flutter

firebase_core

firebase_core: ^2.15.1

cupertino_icons: ^1.0.2

- 3 Selanjutnya **firebase_core** sudah dapat dipakai oleh project
- 4 Tambahkan kode inisialisasi Firebase di fungsi **main()** pada berkas **main.dart**.

```

void main() async {
    WidgetsFlutterBinding.ensureInitialized();
    await Firebase.initializeApp(
      options: DefaultFirebaseOptions.currentPlatform,
    );
    runApp(const MyApp());
}

```

- 5 Jalankan project seperti biasa.

Firebase Authentication

Firebase Authentication adalah layanan yang disediakan oleh Firebase yang memungkinkan Anda untuk mengintegrasikan sistem autentikasi pengguna ke dalam aplikasi Anda dengan cepat dan aman. Layanan ini memungkinkan pengguna untuk mendaftar, masuk, dan mengelola akun pengguna dengan mudah, serta memberikan berbagai pilihan metode autentikasi yang dapat digunakan, seperti email dan sandi, Google, Facebook, Twitter, dan banyak lagi. [Firebase Authentication Documentation](#) [FlutterFire Authentication Documentation](#). Beberapa fitur pada Firebase Authentication antara lain:

1. **Sign Up (Pendaftaran)**: Pengguna dapat membuat akun baru dengan email dan kata sandi atau menggunakan akun media sosial yang telah ada.
2. **Sign In (Masuk)**: Pengguna yang telah terdaftar dapat masuk ke aplikasi dengan menyediakan informasi autentikasi yang benar.
3. **Password Reset (Pemulihan Kata Sandi)**: Firebase Authentication memungkinkan pengguna untuk mereset kata sandi mereka jika mereka lupa.
4. **Verifikasi Email**: Anda dapat mengirim email verifikasi kepada pengguna setelah mereka mendaftar.
5. **Otentikasi Dua Faktor**: Layanan ini juga mendukung otentikasi dua faktor untuk meningkatkan keamanan.

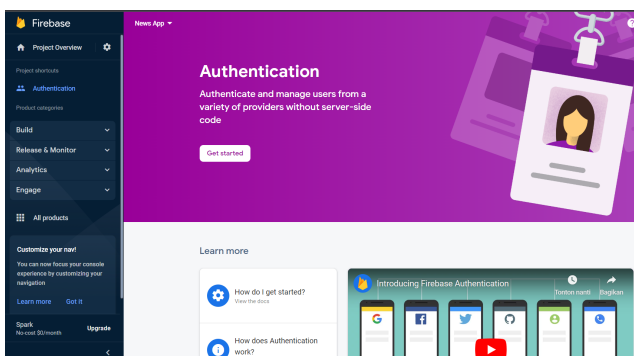
Metode autentikasi yang dapat digunakan menggunakan Firebase Authentication antara lain:

1. **Email dan Password**: Pengguna mendaftar dengan menyediakan alamat email dan kata sandi.
2. **Google Sign-In**: Pengguna dapat masuk menggunakan akun Google mereka.
3. **Facebook Login**: Pengguna dapat masuk menggunakan akun Facebook mereka.
4. **Twitter Login**: Pengguna dapat masuk menggunakan akun Twitter mereka.
5. **Phone Number Authentication**: Pengguna dapat mendaftar dan masuk dengan nomor telepon mereka.
6. **Custom Authentication**: Anda dapat mengintegrasikan sistem autentikasi yang sudah ada dalam aplikasi Anda.

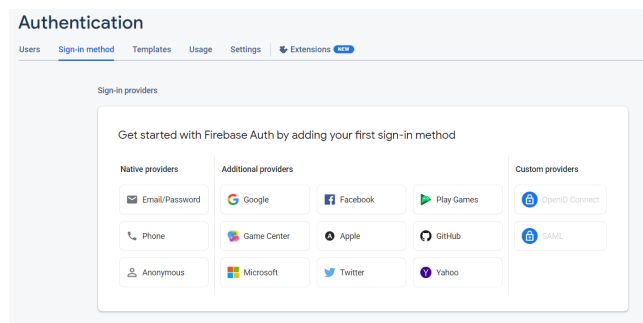
Authentication Email dan Password

Untuk mengaktifkan metode autentikasi pada halaman utama project Firebase sebelah kiri pilih **Build** lalu klik **Authentication**.

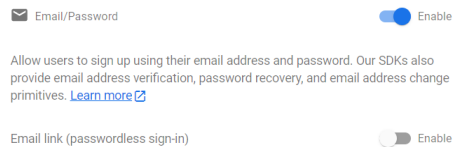
1. Pilih **Build** kemudian **Authentication**. Setelah itu klik **Get Started**.



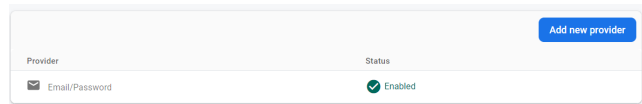
2. Otomatis kita akan diarahkan ke **tab Sign-in method** yang menampilkan beberapa pilihan metode autentikasi yang bisa digunakan. Pilih **Email/Password**.



3. Klik **Enable** di **switch pertama** lalu klik **Save**.



4. Kita bisa menambahkan provider lain seperti login dengan **Google** dengan klik **Add New Provider**.



Installation

Sebelum mengimplementasikan autentikasi pada project kita bisa menambahkan **firebase_auth** plugin pada file **pubspec.yaml** [firebase_auth](#).

- Buka file **pubspec.yaml** pada project
 - nama_project
 - pubspec.yaml
- Ubah dari code **sebelum** (kiri) menjadi **sesudah** (kanan) lalu **save**.

dependencies:

flutter:

 sdk: flutter

The following adds the Cupertino Icons font to your application.

Use with the CupertinoIcons class for iOS style icons.

cupertino_icons: ^1.0.2

dependencies:

flutter:

 sdk: flutter

firebase_core

firebase_core: ^2.15.1

firebase_auth

firebase_auth: ^4.7.3

cupertino_icons: ^1.0.2

- Selanjutnya **firebase_auth** sudah dapat dipakai oleh project.

Register Email dan Password

Setelah melakukan instalasi kita bisa mulai untuk membuat fitur untuk registrasi.

- Pertama kita bisa membuat file dengan nama **auth_controller.dart** untuk mengatur proses autentikasi aplikasi. Kita bisa menerapkan state management GetX disini.

```
class AuthController extends GetxController {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  RxBool isLoading = false.obs;
```

```

Future<void> registerUser(String email, String password) async {
  try {
    isLoading.value = true;
    await _auth.createUserWithEmailAndPassword(
      email: email,
      password: password,
    );
    Get.snackbar('Success', 'Registration successful',
      backgroundColor: Colors.green);
    Get.off(LoginPage()); //Navigate ke Login Page
  } catch (error) {
    Get.snackbar('Error', 'Registration failed: $error',
      backgroundColor: Colors.red);
  } finally {
    isLoading.value = false;
  }
}

```

Metode **createUserWithEmailAndPassword** menerima dua parameter **email** dan **password** yang diinputkan oleh pengguna.

2. Selanjutnya kita bisa untuk membuat tampilan halaman registernya, buat file **register_page.dart** lalu isi codingan seperti berikut:

```

class RegisterPage extends StatefulWidget {
  @override
  State<RegisterPage> createState() => _RegisterPageState();
}

class _RegisterPageState extends State<RegisterPage> {
  final AuthController _authController = Get.put(AuthController());

  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();

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

```

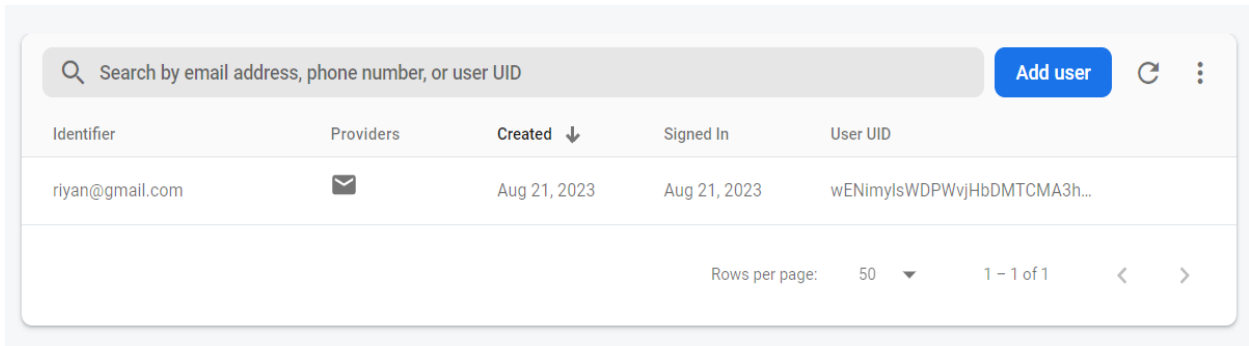
```

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Register'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          TextField(
            controller: _emailController,
            decoration: InputDecoration(labelText: 'Email'),
          ),
          TextField(
            controller: _passwordController,
            obscureText: true,
            decoration: InputDecoration(labelText: 'Password'),
          ),
          SizedBox(height: 16),
          Obx(() {
            return ElevatedButton(
              onPressed: _authController.isLoading.value
                ? null
                : () {
                    _authController.registerUser(
                      _emailController.text,
                      _passwordController.text,
                    );
                  },
              child: _authController.isLoading.value
                ? CircularProgressIndicator()
                : Text('Register'),
            );
          }),
        ],
      ),
    ),
  );
}

```

}

3. Ketika kita berhasil melakukan registrasi pada halaman Firebase daftar pengguna akan muncul.



Login Email dan Password

Setelah membuat fitur registrasi, kita bisa melanjutkan untuk membuat fitur login.

1. Pada file dengan nama **auth_controller.dart** tambahkan fungsi **LoginUser** untuk mengatur proses login.

```
class AuthController extends GetxController {
  final FirebaseAuth _auth = FirebaseAuth.instance;

  RxBool isLoading = false.obs;

  Future<void> registerUser(String email, String password) async {
    try {
      isLoading.value = true;
      await _auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );
      Get.snackbar('Success', 'Registration successful',
        backgroundColor: Colors.green);
      Get.off(LoginPage()); //Navigate ke Login Page
    } catch (error) {
      Get.snackbar('Error', 'Registration failed: $error',
        backgroundColor: Colors.red);
    } finally {
      isLoading.value = false;
    }
  }

  Future<void> loginUser(String email, String password) async {
```

```

try {
  isLoading.value = true;
  await _auth.signInWithEmailAndPassword(
    email: email,
    password: password,
  );
  Get.snackbar('Success', 'Login successful',
    backgroundColor: Colors.green);
} catch (error) {
  Get.snackbar('Error', 'Login failed: $error',
    backgroundColor: Colors.red);
} finally {
  isLoading.value = false;
}
}
}

```

Proses autentikasi akan dijalankan pada backend Firebase ketika Anda memanggil metode **signInWithEmailAndPassword()**. Data pengguna yang digunakan untuk login juga akan disimpan secara lokal pada perangkat Anda. Nantinya Anda dapat menggunakan data ini untuk situasi lain yang berhubungan dengan profil pengguna, seperti mengecek apakah pengguna telah login ketika pertama kali aplikasi dibuka.

2. Selanjutnya kita buat file **login_page.dart** untuk tampilan halaman loginnya.

```

class LoginPage extends StatefulWidget {
  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final AuthController _authController = Get.put(AuthController());

  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();

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

  @override

```

```

Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Login'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          TextField(
            controller: _emailController,
            decoration: InputDecoration(labelText: 'Email'),
          ),
          TextField(
            controller: _passwordController,
            obscureText: true,
            decoration: InputDecoration(labelText: 'Password'),
          ),
          SizedBox(height: 16),
          Obx(() {
            return ElevatedButton(
              onPressed: _authController.isLoading.value
                ? null
                : () {
                    _authController.loginUser(
                      _emailController.text,
                      _passwordController.text,
                    );
                  },
              child: _authController.isLoading.value
                ? CircularProgressIndicator()
                : Text('Login'),
            );
          }),
        ],
      ),
    ),
  );
}

```

Logout

Setelah membuat fitur register dan login, selanjutnya kita bisa membuat fitur logout.

1. Tambahkan fungsi logout pada file **auth_controller.dart** untuk mengatur proses logout.

```
class AuthController extends GetxController {
  // ...

  void logout() async {
    await _auth.signOut();
    Get.offAll(LoginPage()); // Menghapus semua halaman dari tumpukan dan navigasi ke halaman login
  }
}
```

2. Selanjutnya kita bisa memanggil fungsi tersebut pada sebuah button yang kita inginkan.

```
class HomePage extends StatelessWidget {
  final AuthController _authController = Get.put(AuthController());

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Home'),
        actions: [
          IconButton(
            icon: Icon(Icons.exit_to_app),
            onPressed: () {
              _authController.logout();
            },
          ),
        ],
      ),
    );
  }
}
```


Shared Preferences

Shared Preferences adalah cara sederhana untuk menyimpan dan mengambil data dalam bentuk pasangan **key-value** pada penyimpanan lokal perangkat dalam aplikasi Flutter. Ini adalah salah satu cara yang umum digunakan untuk menyimpan pengaturan aplikasi, preferensi pengguna, dan informasi kecil lainnya yang perlu

bertahan antara sesi aplikasi. Dengan SharedPreferences, Anda dapat menyimpan dan mengambil data dengan tipe data yang berbeda seperti int, double, bool, dan string. Ini adalah pilihan yang baik ketika Anda ingin menyimpan data yang tidak terlalu kompleks dan hanya perlu akses yang cepat dan mudah tanpa perlu database. SharedPreferences dapat digunakan dalam sistem login untuk menyimpan informasi tentang status login pengguna atau data yang berhubungan dengan pengguna setelah proses login.

Installation

Tambahkan `shared_preferences` plugin pada file `pubspec.yaml` [shared_preferences](#).

- 1 Buka file `pubspec.yaml` pada project
 -  nama_project
 -  `pubspec.yaml`
- 2 Ubah dari code `sebelum` (kiri) menjadi `sesudah` (kanan) lalu `save`.

`dependencies:`

`flutter:`

`sdk: flutter`

The following adds the Cupertino Icons font to your application.

Use with the CupertinoIcons class for iOS style icons.

`cupertino_icons: ^1.0.2`

`dependencies:`

`flutter:`

`sdk: flutter`

firebase_core

`firebase_core: ^2.15.1`

firebase_auth

`firebase_auth: ^4.7.3`

shared_preferences

`shared_preferences: ^2.2.0`

`cupertino_icons: ^1.0.2`

- 3 Selanjutnya `shared_preferences` sudah dapat dipakai oleh project.

Implementasi Shared Preferences pada Sistem Login

Implementasi ini berguna untuk menyimpan status login pengguna pada aplikasi, jadi pengguna tidak perlu login ulang ketika membuka aplikasi nya lagi.

1. Pada file `auth_controller.dart` tambahkan shared preferences untuk menyimpan token autentikasi dan tambahkan juga fungsi `checkStatusLogin()` untuk mengecek status login.

```
class AuthController extends GetxController {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final SharedPreferences _prefs = Get.find<SharedPreferences>();

  RxBool isLoading = false.obs;
  RxBool isLoggedIn = false.obs;
```



```

@override
void onInit() {
  super.onInit();
  checkLoginStatus(); // Cek status login saat controller diinisialisasi
}

Future<void> checkLoginStatus() async {
  isLoggedIn.value = _prefs.containsKey('user_token');
}

// Fungsi register user
...

Future<void> loginUser(String email, String password) async {
  try {
    isLoading.value = true;
    await _auth.signInWithEmailAndPassword(
      email: email,
      password: password,
    );
    _prefs.setString('user_token', _auth.currentUser!.uid); // Simpan token autentikasi
    Get.snackbar('Success', 'Login successful',
      backgroundColor: Colors.green);
    isLoggedIn.value = true; // Set status login menjadi true
    Get.offAllNamed('/home'); // Navigate ke HomePage dan hapus semua halaman sebelumnya
  } catch (error) {
    Get.snackbar('Error', 'Login failed: $error',
      backgroundColor: Colors.red);
  } finally {
    isLoading.value = false;
  }
}

void logout() {
  _prefs.remove('user_token'); // Hapus token autentikasi dari penyimpanan
  isLoggedIn.value = false; // Set status login menjadi false
  _auth.signOut(); // Sign out dari Firebase Authentication
  Get.offAllNamed('/login'); // Navigate ke HomePage dan hapus semua halaman sebelumnya
}
}

```

2. Kemudian inisialisasi shared preferences pada fungsi **main()** di file **main.dart**.

```

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  await Get.putAsync(() async => await SharedPreferences.getInstance());

  runApp(MyApp());
}

```

3. Pada **MyApp** di **main.dart** kita atur navigasi berdasarkan status login pengguna saat aplikasi dimulai. Di sini, kita menggunakan **initialRoute** untuk menentukan halaman pertama yang akan ditampilkan berdasarkan status login. Jika pengguna telah login, maka aplikasi akan membuka halaman **HomePage**, jika belum, maka akan membuka halaman **LoginPage**.

```

class MyApp extends StatelessWidget {
  MyApp({super.key});
  final AuthController _authController = Get.put(AuthController()); // Buat instance AuthController

  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      initialRoute: _authController.isLoggedIn.value ? '/home' : '/login', // Set initial route based on login status
      getPages: [
        GetPage(name: '/login', page: () => LoginPage()),
        GetPage(name: '/home', page: () => HomePage()),
      ],
    );
  }
}

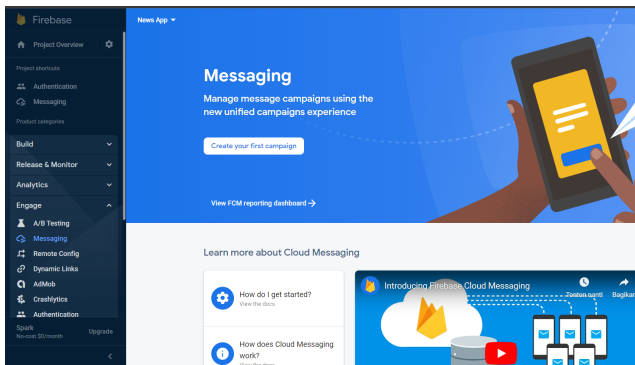
```

Firestore Messaging

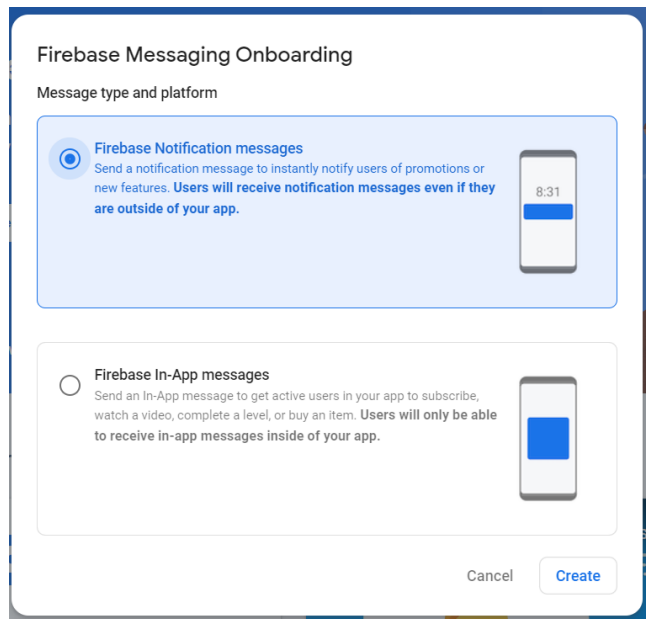
Firestore Cloud Messaging (FCM) adalah layanan yang disediakan oleh Firestore yang memungkinkan Anda mengirimkan notifikasi dan pesan langsung ke perangkat pengguna Anda. Ini adalah solusi untuk mengelola

komunikasi antara server dan klien dalam aplikasi mobile atau web. Firebase Messaging mendukung pengiriman notifikasi push dan pesan data, serta memiliki fitur-fitur yang kuat untuk menyesuaikan dan mengatur cara notifikasi dan pesan dikirimkan. [Firebase FCM Documentation](#) [Flutter Fire FCM Documentation](#)

1. Pilih **Engage** kemudian **Messaging**. Setelah itu klik **Create Your First Campaign**.



2. Terdapat 2 opsi, kali ini kita Pilih **Firestore Notification Messages**. Setelah itu kita akan diarahkan ke halaman **Compose Notification**. Sebelum menggunakan fitur yang sudah ada, kita persiapkan dulu instalasi dan codingannya dulu.



Installation

Instalasi Firebase Messaging cukup dengan menambahkan `firebase_messaging` plugin pada file `pubspec.yaml` [firebase_messaging](#).

1. Buka file `pubspec.yaml` pada project
 - nama_project
 - pubspec.yaml
2. Ubah dari code **sebelum** (kiri) menjadi **sesudah** (kanan) lalu **save**.

```
dependencies:
```

```
flutter:
```

```
  sdk: flutter
```

```
# The following adds the Cupertino Icons font to your application.
```

```
# Use with the CupertinoIcons class for iOS style icons.
```

```
cupertino_icons: ^1.0.2
```

```
dependencies:
```

```
flutter:
```

```
  sdk: flutter
```

```
# firebase_core
```

```
firebase_core: ^2.15.1
```

```
# firebase_auth
```

```
firebase_auth: ^4.7.3
```

```
# shared_preferences
```

```
shared_preferences: ^2.2.0
```

```
# firebase_messaging
```

```
firebase_messaging: ^14.6.6
```

```
cupertino_icons: ^1.0.2
```

- 3 Selanjutnya **firebase_messaging** sudah dapat dipakai oleh project.

Handler Background dan Terminated Message

Pada percobaan ini kita akan membuat kondisi ketika aplikasi dijalankan pada background dan terminated.

1. Pertama buat file **notification_handler.dart** untuk menangani messaging pada tiap-tiap kondisi.

```
Future<void> _firebaseMessagingBackgroundHandler(RemoteMessage message) async {
  print('Message received in background: ${message.notification?.title}');
}

class FirebaseMessagingHandler {
  final FirebaseMessaging _firebaseMessaging = FirebaseMessaging.instance;

  Future<void> initPushNotification() async {

    //allow user to give permission for notification
    NotificationSettings settings = await _firebaseMessaging.requestPermission(
      alert: true,
      announcement: false,
      badge: true,
      carPlay: false,
      criticalAlert: false,
      provisional: false,
      sound: true,
    );
  }
}
```

```

print('User granted permission: ${settings.authorizationStatus}');

//get token messaging
_firebaseMessaging.getToken().then((token) {
  print('FCM Token: $token');
});

//handler terminated message
FirebaseMessaging.instance.getInitialMessage().then((message) {
  print("terminatedNotification : ${message!.notification?.title}");
});

//handler onbackground message
FirebaseMessaging.onBackgroundMessage(_firebaseMessagingBackgroundHandler);

}

```

2. Setelah itu panggil fungsi **initPushNotification()** di fungsi **main()** pada file **main.dart**.

```

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  await Get.putAsync(() async => await SharedPreferences.getInstance());
  await FirebaseMessagingHandler().initPushNotification();

  runApp(const MyApp());
}

```

3. Jalankan Project seperti biasa



4. Setelah project berhasil dijalankan, salin **FCM token** yang ada pada console.

5. Buka kembali Firebase pada halaman **Compose Notification**. Isi **Notification Title** dan **Notification Text** sesuai keinginan. Lalu pada sebelah kanan klik **Send Test Message**.

Pada percobaan ini kita akan membuat kondisi ketika aplikasi dijalankan pada kondisi foreground atau ketika aplikasi sedang berjalan. Disebabkan **foreground message sulit atau lama untuk terpicu** maka disini kita akan menambahkan package **flutter local notifications**.

Flutter Local Notifications adalah sebuah paket atau library yang digunakan dalam pembuatan aplikasi Flutter untuk mengelola dan menampilkan notifikasi secara lokal pada perangkat pengguna. Paket ini memungkinkan Anda membuat, mengatur, dan menampilkan notifikasi yang dapat dilihat oleh pengguna pada perangkat mereka, tanpa perlu terhubung dengan server eksternal seperti yang dilakukan oleh Firebase Cloud Messaging (FCM) untuk notifikasi jarak jauh.

Instalasi dilakukan cukup dengan menambahkan `flutter_local_notifications` plugin pada file `pubspec.yaml` `flutter local notifications`.

- 1 Buka file **pubspec.yaml** pada project
 -  nama_project
 -  pubspec.yaml
- 2 Ubah dari code **sebelum** (kiri) menjadi **sesudah** (kanan) lalu **save**.

dependencies:

flutter:

 sdk: flutter

The following adds the Cupertino Icons font to your application.

Use with the CupertinoIcons class for iOS style icons.

cupertino_icons: ^1.0.2

dependencies:

flutter:

 sdk: flutter

firebase_core

firebase_core: ^2.15.1

firebase_auth

firebase_auth: ^4.7.3

shared_preferences

shared_preferences: ^2.2.0

firebase_messaging

firebase_messaging: ^14.6.6

flutter_local_notifications

flutter_local_notifications: ^15.1.0+1

cupertino_icons: ^1.0.2

- 3 Selanjutnya **flutter_local_notifications** sudah dapat dipakai oleh project.

Implementasi Firebase Messaging dan Local Notifications

Terdapat beberapa settingan yang harus kita ubah dan tambahkan agar notifikasi terpicu.

1. Pada file **notification_handler.dart** tambahkan handler untuk menangani proses message foreground. Tambahkan juga beberapa inisialisasi notification channel dan buat fungsi **initLocalNotification()** untuk local notifications.

```
class FirebaseMessagingHandler {
  final FirebaseMessaging _firebaseMessaging = FirebaseMessaging.instance;

  //inisialisasi notification channel untuk android
  final _androidChannel = const AndroidNotificationChannel(
    'channel_notification',
    'High Importance Notification',
    description: 'Used For Notification',
    importance: Importance.defaultImportance,
```

```

);

final _localNotification = FlutterLocalNotificationsPlugin();

Future<void> initPushNotification() async {
  //allow user to give permission for notification
  ...

  //get token messaging
  ...

  //handler terminated message
  ...

  //handler onbackground message
  ...

  //handler foreground message with local notification
  FirebaseMessaging.onMessage.listen((message) {
    final notification = message.notification;
    if (notification == null) return;
    _localNotification.show(
      notification.hashCode,
      notification.title,
      notification.body,
      NotificationDetails(
        android: AndroidNotificationDetails(
          _androidChannel.id, _androidChannel.name,
          channelDescription: _androidChannel.description,
          icon: '@drawable/ic_launcher'),
        payload: jsonEncode(message.toMap()),
      );
    print(
      'Message received while app is in foreground: ${message.notification?.title}');
  });

  //handler when open the message
  FirebaseMessaging.onMessageOpenedApp.listen((RemoteMessage message) {
    print('Message opened from notification: ${message.notification?.title}');
  });
}

```



```
Future initLocalNotification() async {
  const ios = DarwinInitializationSettings();
  const android = AndroidInitializationSettings('@drawable/ic_launcher');
  const settings = InitializationSettings(android: android, iOS: ios);
  await _localNotification.initialize(settings);
}
}
```

`@drawable/ic_launcher` merupakan path icon untuk notifikasi pada folder **drawable** yang terletak pada folder **android/app/src/main/res**. Untuk mendapat icon **ic_launcher** kita bisa **salin file ic_launcher.png** yang ada pada folder **mipmap-xxhdpi** lalu paste ke dalam folder **drawable**.

2. Panggil fungsi **initLocalNotification()** pada fungsi **main()** di file **main.dart**.

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  await Get.putAsync(() async => await SharedPreferences.getInstance());
  await FirebaseMessagingHandler().initPushNotification();
  await FirebaseMessagingHandler().initLocalNotification();

  runApp(const MyApp());
}
```

3. Tambahkan settingan untuk android notification channel pada **AndroidManifest.xml** yang terletak pada folder **android/app/src/main**.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
  <application
    ...
    <activity
      ...
    </activity>
    <meta-data
      android:name="com.google.firebase.messaging.default_notification_channel_id"
      android:value="channel_notification" />
    ...
  </application>
</manifest>
```

4. Jalankan aplikasi dan lakukan test yang sama seperti sebelumnya dengan kondisi aplikasi sedang berjalan melalui halaman **Compose Notification** di Firebase.

Push Local Notification

Pada percobaan ini kita akan memakai package **flutter_local_notification** saja untuk menampilkan notifikasi pada perangkat kita.

Menampilkan Notifikasi Sederhana

Penerapan notifikasi sederhana ini sudah pernah digunakan sebelumnya pada Implementasi Firebase Messaging dan Local Notification. Berikut kode untuk notifikasi ini:

```
Future<void> showNotification(
  FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin) async {
  var androidPlatformChannelSpecifics = AndroidNotificationDetails(
    _channelId, _channelName, channelDescription: _channelDesc,
    importance: Importance.max, priority: Priority.high, ticker: 'ticker');

  var iOSPlatformChannelSpecifics = DarwinNotificationDetails();

  var platformChannelSpecifics = NotificationDetails(
    android: androidPlatformChannelSpecifics,
    iOS: iOSPlatformChannelSpecifics,
  );

  await flutterLocalNotificationsPlugin.show(
    0,
    'plain title',
    'plain body',
    platformChannelSpecifics,
    payload: 'plain notification',
  );
}
```

Pada Fungsi **showNotification** terdapat beberapa pendeklarasian variabel seperti **androidPlatformChannelSpecifics**, **iOSPlatformChannelSpecifics**, dan **platformChannelSpecifics**. Variabel **androidPlatformChannelSpecifics** merupakan sebuah objek dari kelas **AndroidNotificationDetails** yang berfungsi untuk mengatur konfigurasi notifikasi pada platform Android. Kemudian kita juga bisa lihat di dalam objek kelas **AndroidNotificationDetails** kita memanggil **channel** yang telah kita buat sebelumnya seperti **_channelId**, **_channelName**, dan **_channelDesc**.

Lalu **platformChannelSpecifics** merupakan sebuah objek dari kelas **NotificationDetails** yang bertugas mengatur notifikasi untuk setiap platform yaitu Android dan iOS.

Kemudian setelah berhasil menyiapkan konfigurasi notifikasi untuk setiap platform, Anda akan menampilkan notifikasi dengan memanggil fungsi `.show(...)` dari inputan parameter yaitu `flutterLocalNotificationsPlugin`. Fungsi tersebut akan menampilkan notifikasi sederhana seperti menampilkan *title*, *body*, dan *icon* dari notifikasi.

Kita menentukan pada parameter `payload` yaitu 'plain notification' yang akan diteruskan kembali melalui aplikasi Anda saat pengguna mengetuk notifikasi.

Menampilkan Notifikasi dengan Progres Indikator di Android

Selain menampilkan notifikasi berupa pesan kita juga bisa menampilkan suatu progres pada notifikasi. Contohnya pada saat kita mendownload sesuatu maka notifikasi progres download akan muncul. Namun fungsi ini hanya berjalan pada platform Android saja. Pertama kita perlu melakukan perulangan terlebih dahulu untuk menentukan status dari progres yang akan ditampilkan pada notifikasi. Kemudian kita menambahkan parameter `showProgress`, `maxProgress`, dan `progress`, di mana nilai dari parameter `maxProgress` adalah 5 yang berarti progress tersebut akan terisi selama lima kali sebelum progressnya terisi penuh. Berikut contoh kodenya:

```
Future<void> showProgressNotification(
  FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin) async {
  var maxProgress = 5;

  for (var i = 0; i <= maxProgress; i++) {
    await Future.delayed(Duration(seconds: 1), () async {
      var androidPlatformChannelSpecifics = AndroidNotificationDetails(
        _channelId,
        _channelName,
        channelDescription: _channelDesc,
        channelShowBadge: false,
        importance: Importance.max,
        priority: Priority.high,
        onlyAlertOnce: true,
        showProgress: true,
        maxProgress: maxProgress,
        progress: i,
      );

      var platformChannelSpecifics =
        NotificationDetails(android: androidPlatformChannelSpecifics);

      await flutterLocalNotificationsPlugin.show(
        0,
```

```

    'progress notification title',
    'progress notification body',
    platformChannelSpecifics,
    payload: 'item x',
  );
});
}
}

```

Link Github

Berikut merupakan link github yang akan digunakan pada modul pemrograman mobile:

[Link Github](#)

KEGIATAN PRAKTIKUM

A. Setup dan Instalasi Firebase

1. Lakukan setup dan instalasi firebase ke project Anda.
2. Pastikan muncul keterangan bahwa berhasil menginisiasi Firebase di Terminal.
3. Jalankan project seperti biasa

B. Latihan Registrasi Email dan Password

1. Buatlah halaman registrasi sederhana pada aplikasi Anda.
2. Buatlah controller untuk menangani registrasi dari user seperti email dan password menggunakan Firebase Authentication.
3. Ketika user berhasil registrasi maka email yang sudah terdaftar akan muncul pada halaman Authentication di Firebase.

RUBRIK PENILAIAN MODUL 4 MATERI

Bobot Penilaian Modul 4 Materi (20%)

Berhasil melakukan setup dan instalasi Firebase	60
Berhasil membuat halaman registrasi	20
Proses registrasi berfungsi tanpa ada masalah	20

Total	100
-------	-----