MOJADIPRO

INTEGRASI APLIKASI FLUTTER DENGAN FIREBASE

MOJADIPRO

Materi

01 | Named Routes

02 | Static Modifier Dart

03 | Hero Animation

04 | Membuat Firebase Project

05 | Paket Firebase Firestore

06 | Firebase Project

Firebase Cloud Firestore

Adalah layanan database NoSQL yang dapat digunakan untuk menyimpan, menyinkronkan, dan membuat kueri data yang disediakan oleh Google.

MOJADIPRO

Firebase Cloud Firestore

1. Menyediakan penyimpanan data aplikasi di cloud yang dapat diakses secara real-time.
2. Dapat mengakses data dari perangkat apa pun secara online maupun offline.
3. Mem-build aplikasi tanpa server, dengan aman dalam skala global.
4. Dioptimalkan untuk penggunaan offline menggunakan cache lokal.

MOJADIPRO

01

Named Routes

MOJADIPRO

Named Routes

Named Routes adalah cara untuk mengatur dan menavigasi antara berbagai halaman atau layar dalam aplikasi Flutter dengan menggunakan nama yang diberikan.

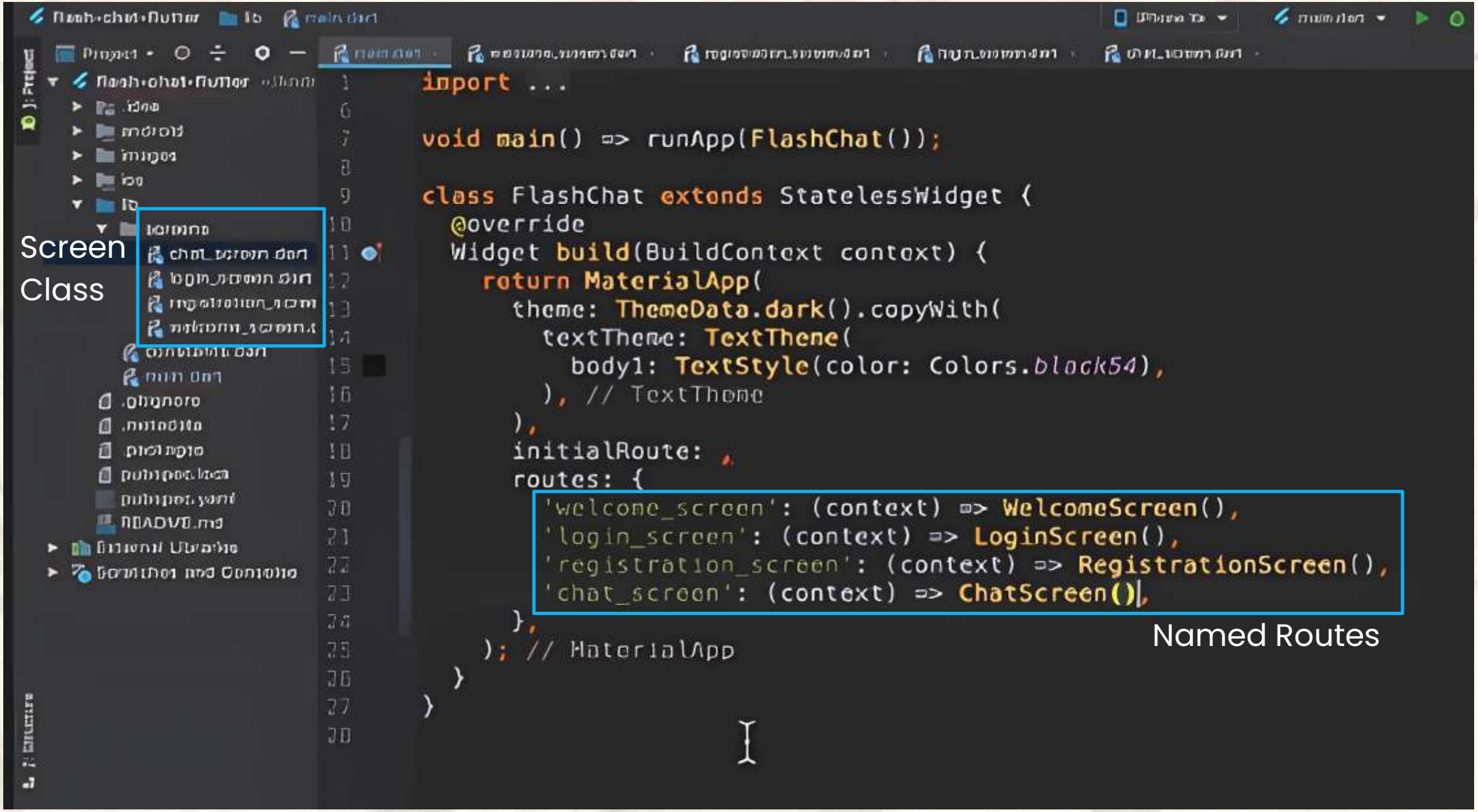
MOJADIPRO

Cara Menggunakan Named Routes

1. Menentukan named routes. Ini biasanya dilakukan di dalam objek MaterialApp, dalam properti routes. Contohnya :

```
MaterialApp(
    initialRoute: '/',
    routes: {
      '/': (context) => HomeScreen(),
      '/second': (context) => SecondScreen(),
      '/third': (context) => ThirdScreen(),
    },
  )
```

MOJADIPRO


Screen Class
import ...
void main() => runApp(FlashChat());
class FlashChat extends StatelessWidget {
@override
Widget build(BuildContext context) {
return MaterialApp(
theme: ThemeData.dark().copyWith(
textTheme: TextTheme(
body1: TextStyle(color: Colors.black54),
), // TextTheme
),
initialRoute: ,
routes: {
'welcome_screen': (context) => WelcomeScreen(),
'login_screen': (context) => LoginScreen(),
'registration_screen': (context) => RegistrationScreen(),
'chat_screen': (context) => ChatScreen(),
},
); // MaterialApp
}
}
Named Routes

MOJADIPRO

Cara Menggunakan Named Routes

2. Buat kelas-kelas untuk setiap layar atau rute dalam aplikasi Anda. Misalnya, dalam contoh di atas, kita perlu membuat kelas HomeScreen, SecondScreen, dan ThirdScreen.

3. Navigasi menggunakan named routes: Untuk melakukan navigasi ke layar atau halaman yang ditentukan menggunakan named route, Anda dapat menggunakan Navigator.pushNamed() atau Navigator.pushReplacementNamed()

MOJADIPRO

Cara Menggunakan Named Routes

Contoh kode berikut menunjukkan cara menavigasi dari HomeScreen ke ProfileScreen :

```
Navigator.pushNamed(context, '/profile');
```

Anda juga dapat mengirimkan argumen tambahan saat melakukan navigasi. Misalnya, jika Anda ingin mengirimkan data pengguna dari HomeScreen ke ProfileScreen :

```
Navigator.pushNamed(context, '/profile', arguments: userData);
```

MOJADIPRO

Cara Menggunakan Named Routes

4. Menerima argumen pada tujuan rute.
   Untuk menerima argumen yang dikirimkan dari rute sebelumnya,
   Anda perlu menambahkan kode berikut di dalam widget tujuan rute tersebut:

```
class ProfileScreen extends StatelessWidget {
@override
Widget build(BuildContext context) {
  final userData = ModalRoute.of(context).settings.arguments;
  // Gunakan data pengguna yang diterima
  // ...
}
}
```

MOJADIPRO

02

Static Modifier Dart

MOJADIPRO

Static Modifier Dart

Static digunakan sebagai modifier untuk menyatakan bahwa suatu anggota (metode, variabel, atau property) dalam sebuah kelas bersifat statis.
Ketika sebuah anggota diberi modifier "static", berarti anggota tersebut terkait dengan kelas itu sendiri, bukan dengan instance (objek) dari kelas tersebut.

MOJADIPRO

1. Metode Statis (Static Methods):

Ketika sebuah metode dideklarasikan sebagai statis, metode tersebut dapat dipanggil langsung dari kelas tanpa membuat objek dari kelas tersebut. Contoh :

```
class MathUtils {
  static int sum(int a, int b) {
    return a + b;
  }
}

void main() {
  int result = MathUtils.sum(5, 3);
  print(result); // Output: 8
}
```

MOJADIPRO

2. Variabel Statis (Static Variables):

Variabel statis adalah variabel yang terkait dengan kelas, bukan dengan objek kelas. Nilai variabel statis diinisialisasi hanya sekali, dan nilainya tetap sama untuk semua instance kelas. Contoh:

```
class Counter {
  static int count = 0;

  Counter() {
    count++;
  }
}

void main() {
  Counter c1 = Counter();
  Counter c2 = Counter();

  print(Counter.count); // Output: 2
}
```

MOJADIPRO

Refactoring Routes

Refactoring" mengacu pada proses memperbaiki kode yang sudah ada tanpa mengubah perilaku fungsionalitasnya. "Routes" adalah istilah yang digunakan dalam pengembangan aplikasi web atau mobile untuk merujuk pada definisi rute atau alamat URL yang digunakan untuk mengarahkan pengguna ke halaman atau tampilan yang tepat.

MOJADIPRO

Static Const

Dalam Dart, penggunaan "static const" pada routes adalah salah satu cara untuk menyimpan definisi rute sebagai konstanta yang tetap di seluruh aplikasi.
Dengan menggunakan "static const", kita dapat mendeklarasikan rute sebagai variabel statis yang bersifat tetap dan dapat diakses tanpa membuat instance dari kelasnya.

MOJADIPRO

Static Const

Berikut adalah contoh penggunaan "static const" untuk merombak rute dalam aplikasi Dart:

```
class Routes {
  static const home = '/';
  static const login = '/login';
  static const profile = '/profile';
  static const settings = '/settings';
}
```

MOJADIPRO

Contoh penggunaan rute yang didefinisikan dengan menggunakan "static const":

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

import 'routes.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'My App',
      initialRoute: Routes.home,
      routes: {
        Routes.home: (context) => HomePage(),
        Routes.login: (context) => LoginPage(),
        Routes.profile: (context) => ProfilePage(),
        Routes.settings: (context) => SettingsPage(),
      },
    );
  }
}
```

```dart
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Home')),
      body: Center(child: Text('Home Page')),
    );
  }
}

// Implementasi kelas halaman lainnya (LoginPage, ProfilePage, dan SettingsPage) di sini...
```

MOJADIPRO

03

Hero Animation

MOJADIPRO

Hero Animation

Hero animation adalah teknik animasi yang digunakan untuk memberikan efek visual yang menarik pada elemen-elemen antarmuka pengguna (user interface) dalam aplikasi atau situs web.

Tujuan utama dari hero animation adalah untuk menciptakan transisi yang halus dan mengesankan antara elemen-elemen antarmuka yang berbeda.

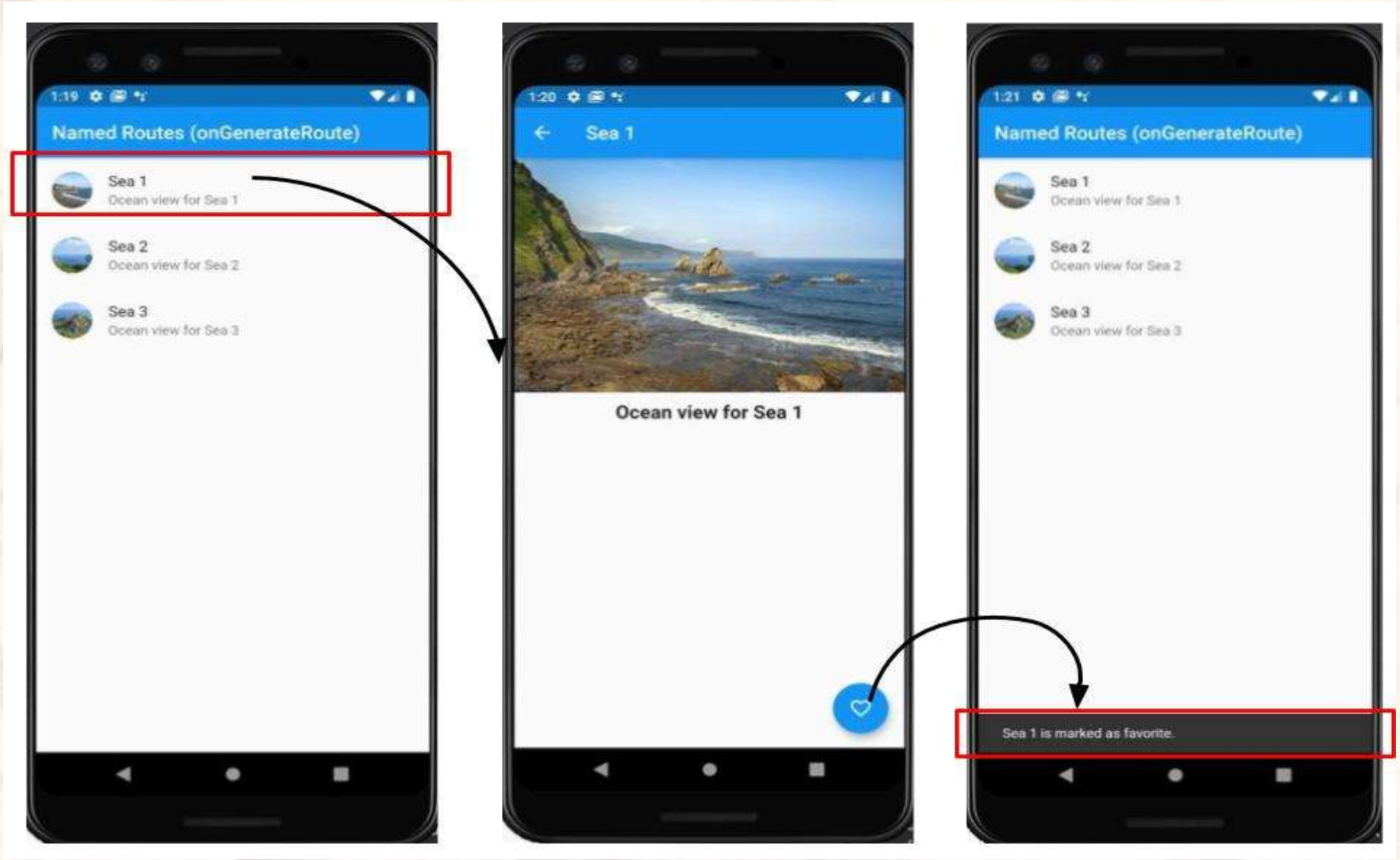
MOJADIPRO

Hero Animation

Dalam hero animation, sebuah elemen antarmuka, seperti tombol atau gambar, diberikan perhatian khusus dengan memperbesar, memudar, bergerak, atau mengubah bentuknya ketika terjadi perubahan dalam antarmuka pengguna.

MOJADIPRO

Contoh Hero Animation



MOJADIPRO

04

Dart Mixin & Dart Stream

MOJADIPRO

Dart Mixin

Dart Mixin adalah mekanisme yang memungkinkan Anda untuk membagikan kode antara berbagai kelas yang tidak saling berhubungan secara hierarkis.
Dalam mixin, Anda dapat mendefinisikan metode dan properti yang dapat digunakan oleh kelas-kelas lain tanpa harus mewarisi dari kelas yang sama.

Berikut adalah contoh sederhana untuk memahami penggunaan mixin dalam Dart:

```
// Definisikan mixin
mixin LoggingMixin {
  void log(String message) {
    print('Log: $message');
  }
}

// Contoh kelas yang menggunakan mixin
class MyClass with LoggingMixin {
  void doSomething() {
    log('Doing something...');
  }
}

void main() {
  var obj = MyClass();
  obj.doSomething(); // Output: Log: Doing something...
}
```

MOJADIPRO

Dart Stream

Dart Stream adalah sebuah konsep yang digunakan dalam bahasa pemrograman Dart untuk memanipulasi data berurutan secara efisien. Stream adalah aliran berkelanjutan dari data yang dapat diproses secara asinkron. Dart Stream dapat digunakan untuk melakukan operasi seperti filter, transformasi, penggabungan, dan reduksi pada data berurutan.

MOJADIPRO

Contoh Penggunaan Dart Stream

1. Membuat Stream:
   Untuk membuat Stream di Dart, dapat menggunakan constructor kelas Stream atau Stream.fromIterable().
   Berikut adalah contoh penggunaannya:

```
// Membuat Stream dari sebuah list
var numbers = [1, 2, 3, 4, 5];
var stream = Stream.fromIterable(numbers);

// Membuat Stream kosong
var emptyStream = Stream<int>.empty();
```

2. Menggunakan Operasi Stream:
   Setelah membuat Stream, Anda dapat menggunakan berbagai operasi pada Stream seperti map, where, reduce, dan lainnya.
   Berikut adalah contoh penggunaan beberapa operasi Stream :

```
// Contoh penggunaan map untuk mengalikan setiap elemen dengan 2
stream.map((number) => number * 2).forEach((result) {
  print(result);
});

// Contoh penggunaan where untuk mengambil elemen yang lebih besar dari 3
stream.where((number) => number > 3).forEach((result) {
  print(result);
});

// Contoh penggunaan reduce untuk menjumlahkan semua elemen
stream.reduce((value, element) => value + element).then((result) {
  print(result);
});
```

3. Menggunakan Stream Controller:
   Stream Controller adalah objek yang digunakan untuk mengontrol dan mengirim data ke Stream. Anda dapat menggunakan Stream Controller untuk menambahkan data ke Stream dan menandai Stream selesai.
   Berikut adalah contoh penggunaan Stream Controller:

```
import 'dart:async';

// Membuat Stream Controller
var controller = StreamController<int>();

// Mengirim data ke Stream menggunakan Stream Controller
controller.add(1);
controller.add(2);
controller.add(3);

// Menandai Stream selesai
controller.close();

// Mendapatkan Stream dari Stream Controller
var stream = controller.stream;

// Menggunakan operasi Stream pada Stream yang diterima dari Stream Controller
stream.forEach((result) {
  print(result);
});
```

MOJADIPRO

Stream Builder

DaDalam konteks Flutter, StreamBuilder memungkinkan kita untuk mengikuti perubahan data dalam stream dan secara otomatis membangun ulang tampilan aplikasi saat ada perubahan data. Hal ini sangat berguna ketika kita ingin menampilkan data secara dinamis dalam aplikasi kita, seperti data yang diperbarui secara real-time atau data yang diambil dari sumber eksternal.

MOJADIPRO

Contoh penggunaan Stream Builder :

Buat kelas atau widget stateful yang berisi widget StreamBuilder:

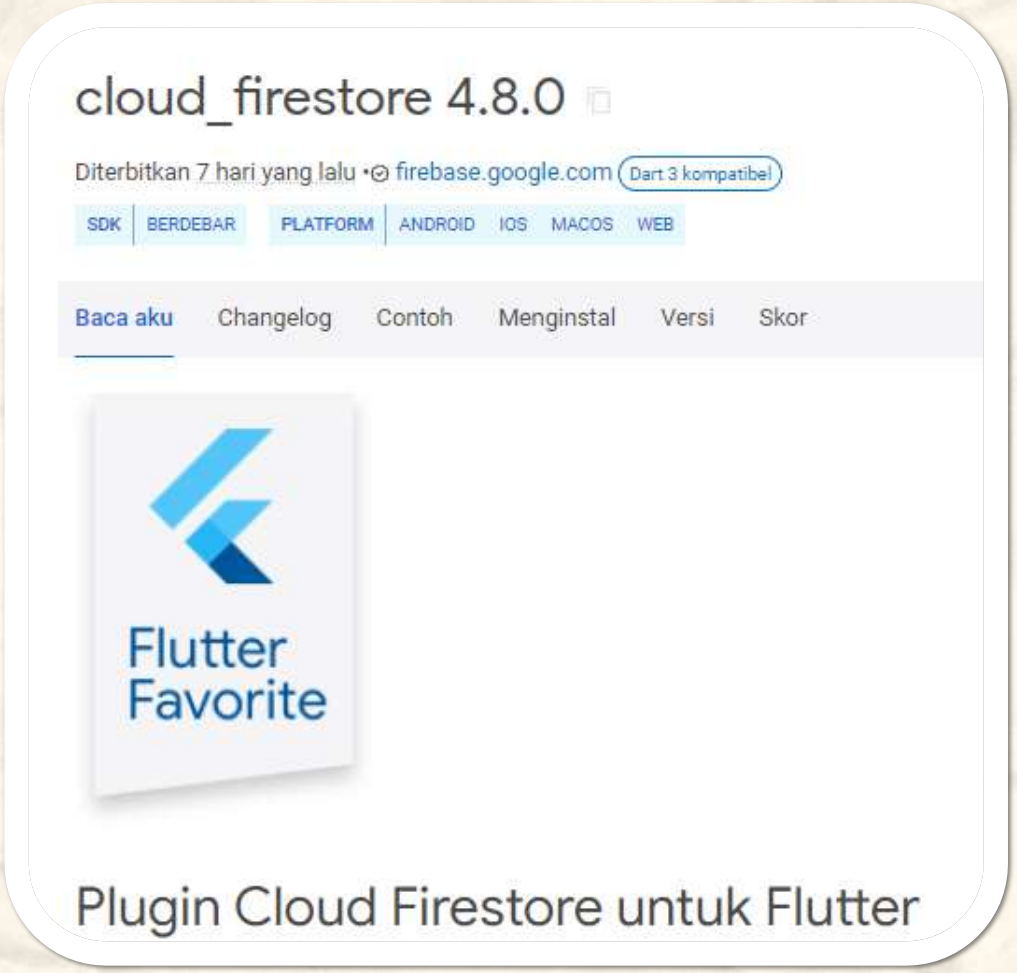
```
import 'dart:async';
import 'package:flutter/material.dart';

class MyWidget extends StatelessWidget {
  // Create a stream controller
  final StreamController<String> _streamController = StreamController<String>();

  @override
  Widget build(BuildContext context) {
    return StreamBuilder<String>(
      stream: _streamController.stream, // Set the stream to listen to
      initialData: '', // Set initial data for the stream
      builder: (BuildContext context, AsyncSnapshot<String> snapshot) {
        // Check the connection state of the stream
        if (snapshot.connectionState == ConnectionState.waiting) {
          return CircularProgressIndicator(); // Show a loading indicator while waiting for data
        } else if (snapshot.hasError) {
          return Text('Error: ${snapshot.error}'); // Show an error message if there is an error in the stream
        } else {
          // Display the data received from the stream
          return Text('Data: ${snapshot.data}');
        }
      },
    );
  }
}
```

MOJADIPRO

05

Paket Firebase Firestore

MOJADIPRO



Paket firebase firestore

1. Cloud firestore
   Paket ini menyediakan API yang memungkinkan Anda untuk mengakses, menambahkan, mengubah, dan menghapus data dalam Firestore.

MOJADIPRO

Paket Firebase Firestore

2. Firebase core ver 2.13.1
   Plugin Flutter untuk menggunakan Firebase Core API, yang memungkinkan koneksi ke beberapa aplikasi Firebase.

3. Firebase auth ver 4.6.2
   Plugin Flutter untuk menggunakan Firebase Authentication API.

4. Animated text kit ver 4.2.2
   Paket flutter yang berisi kumpulan beberapa animasi teks yang keren

MOJADIPRO

Cara Menginstall Paket Firebase Firestore

Menginstal

1. Tergantung padanya

Tambahkan ini ke file paket Anda `pubspec.yaml`:

```
dependencies:
  animated_text_kit: ^4.2.2
```

2. Instal

Anda dapat menginstal paket dari baris perintah:

dengan `pub`:

```
$ pub get
```

dengan `Flutter`:

```
$ flutter pub get
```

3. Impor

Sekarang dalam `Dart` kode Anda, Anda dapat menggunakan:

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

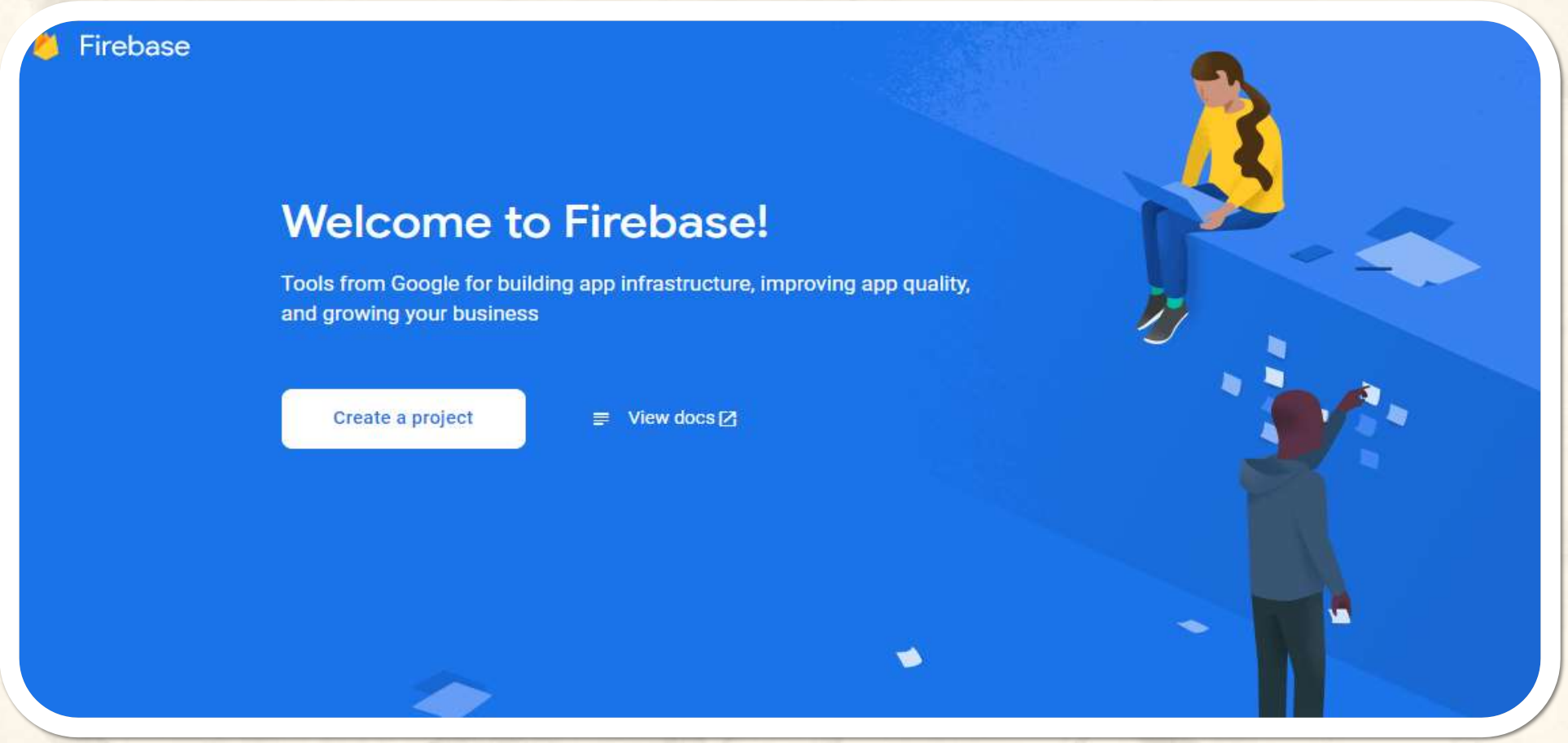
MOJADIPRO

06

Firebase Project

MOJADIPRO

Membuat Firebase Project



Buka laman https://firebase.google.com/?hl=id di Google, lalu klik mulai, lalu klik create a project

MOJADIPRO

Create a project (Step 1 of 3)

Let's start with a name for your project

Project name

Flash Chat

flash-chat-aac43

I accept the Firebase terms

I confirm that I will use Firebase exclusively for purposes relating to my trade, business, craft, or profession.

Continue

Create a project (Step 2 of 3)

Google Analytics for your Firebase project

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 & targeting across Firebase products
- Crash-free users
- Event-based Cloud Functions triggers
- Free unlimited reporting

Enable Google Analytics for this project
Recommended

Previous

Continue

MOJADIPRO

Create a project (Step 3 of 3)

Configure Google Analytics

Analytics location

Indonesia

Google Analytics is a business tool. Use it exclusively for purposes related to your trade, business, craft, or profession.

Data sharing settings and Google Analytics terms

- [x] Use the default settings for sharing Google Analytics data. Learn more
  - [ ] Share your Analytics data with Google to improve Google Products and Services
  - [x] Share your Analytics data with Google to enable Benchmarking
  - [x] Share your Analytics data with Google to enable Technical Support
  - [x] Share your Analytics data with Google Account Specialists

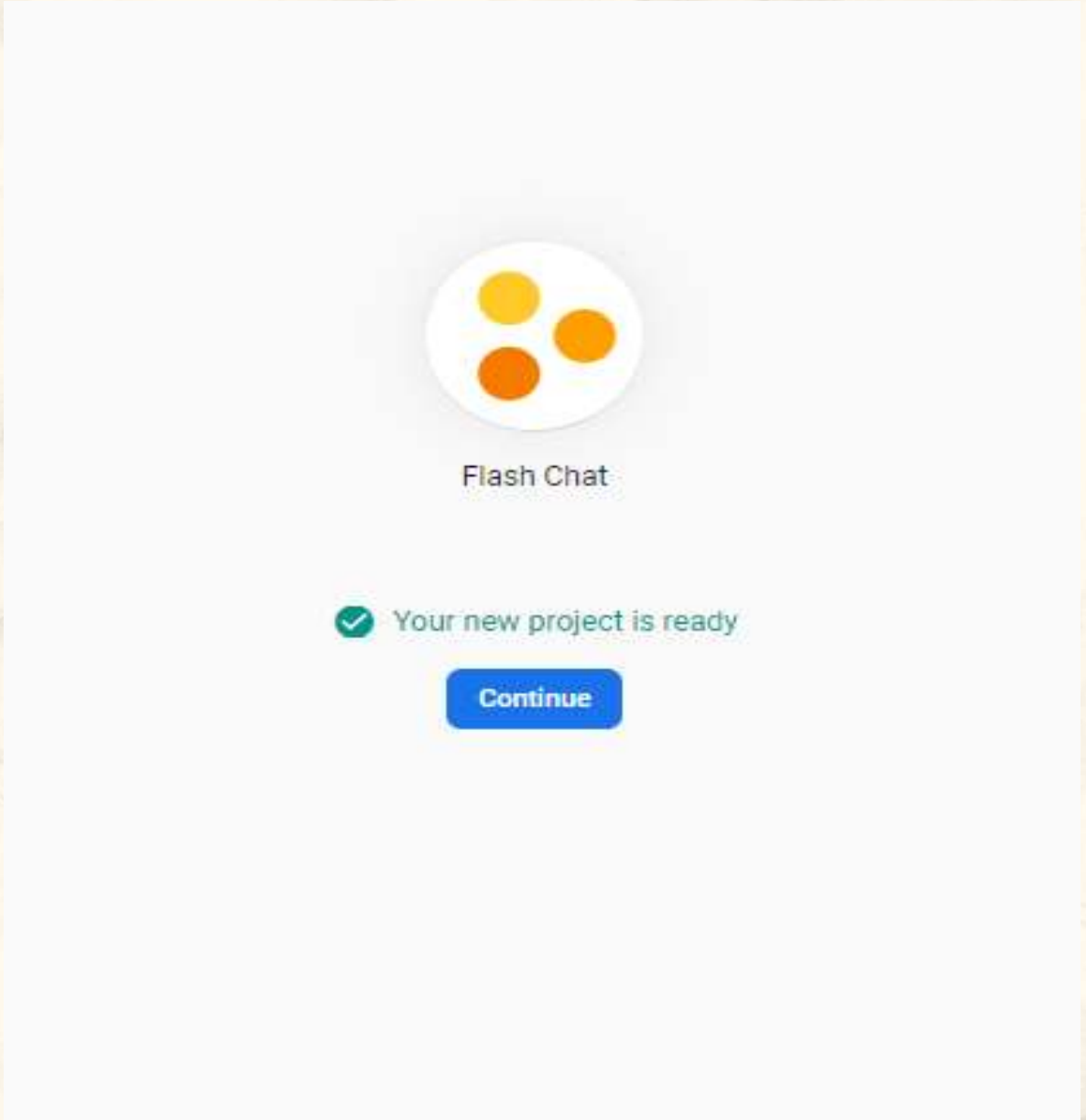
- [x] I accept the Google Analytics terms

Upon project creation, a new Google Analytics property will be created and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. Learn more.

Previous

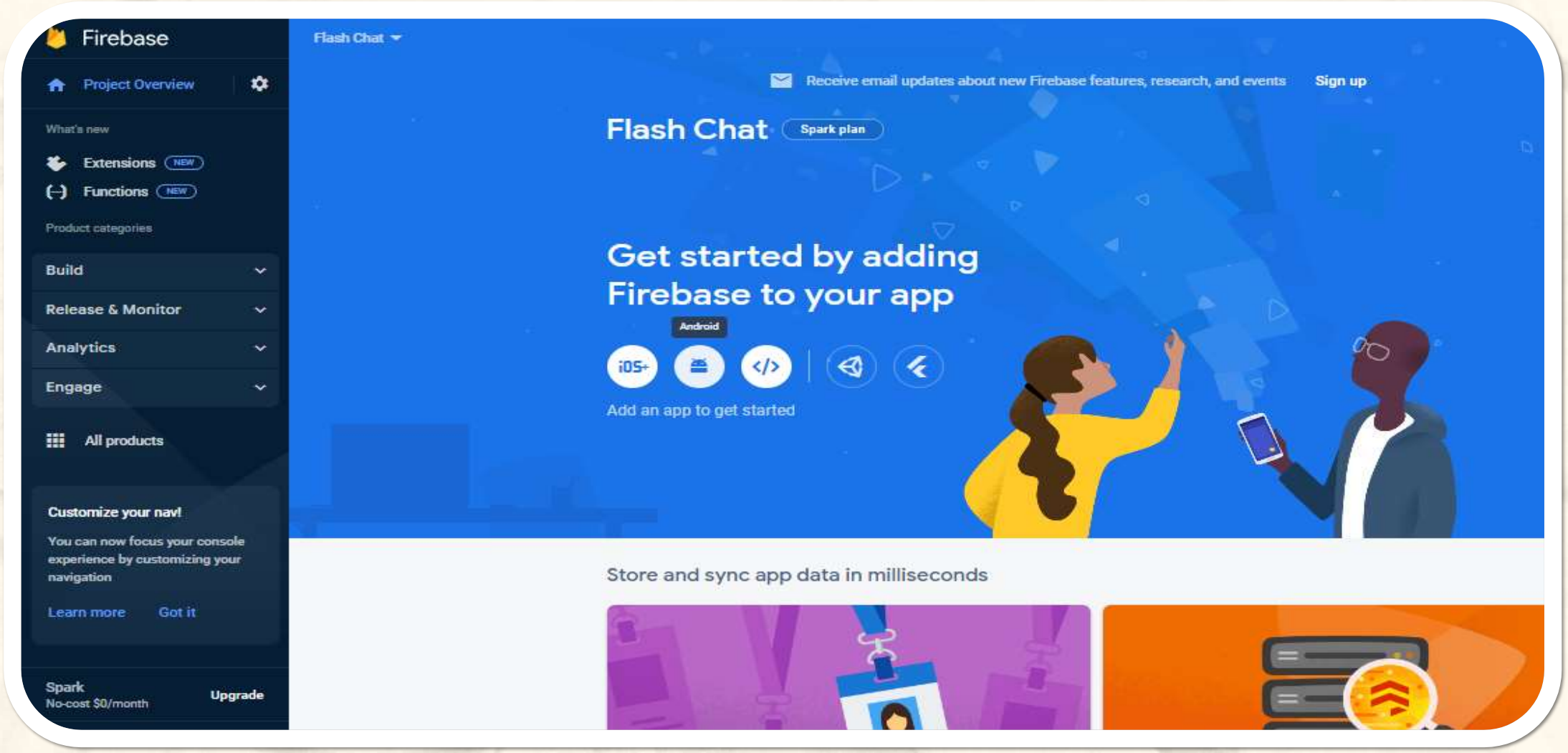
Create project

MOJADIPRO

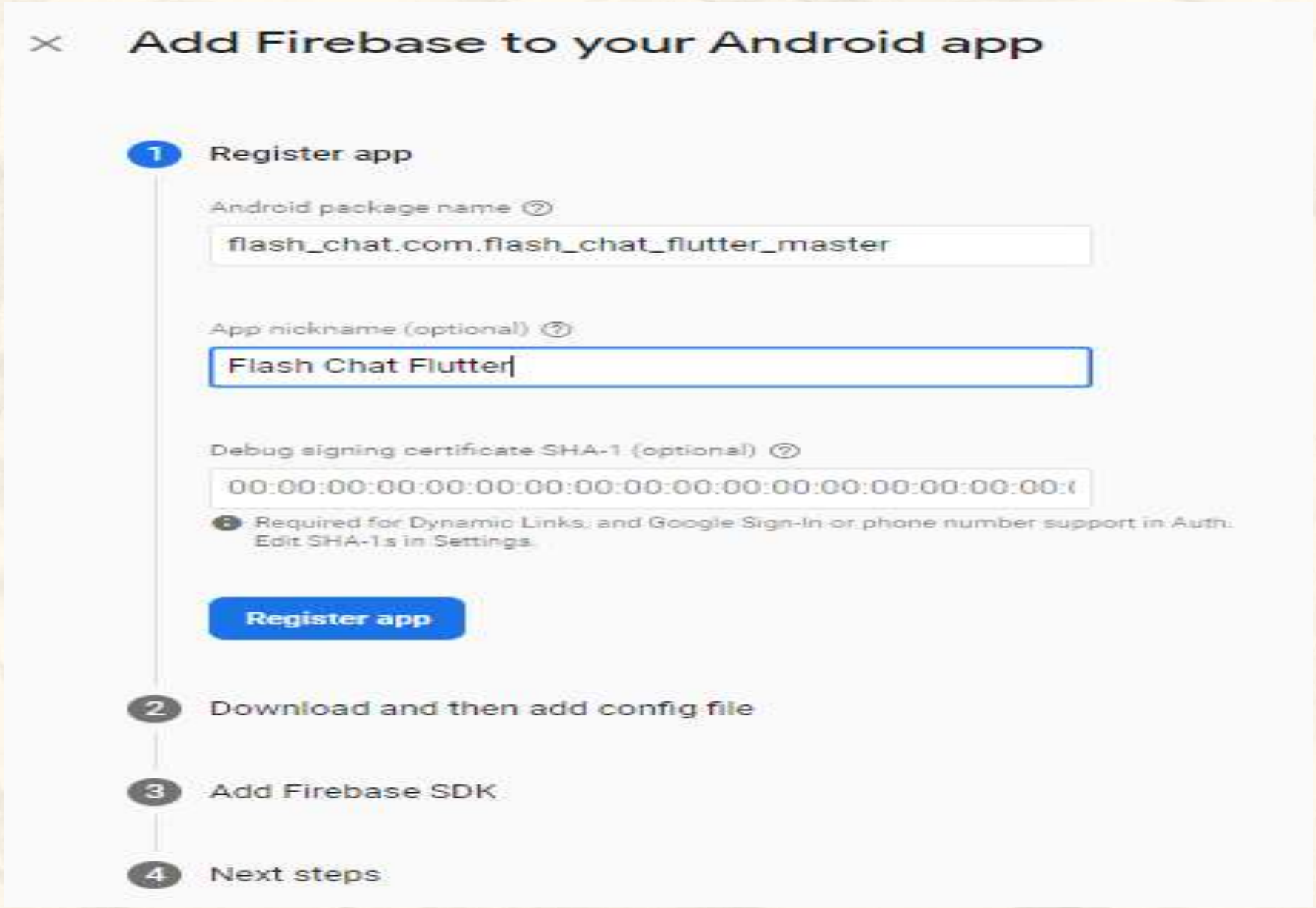


MOJADIPRO

Android Firebase Project Setup



MOJADIPRO


Add Firebase to your Android app
1
Register app
Android package name
flash_chat.com.flash_chat_flutter_master
App nickname (optional)
Flash Chat Flutter
Debug signing certificate SHA-1 (optional)
00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
Required for Dynamic Links, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.
Register app
2
Download and then add config file
3
Add Firebase SDK
4
Next steps

MOJADIPRO

Add Firebase to your Android app

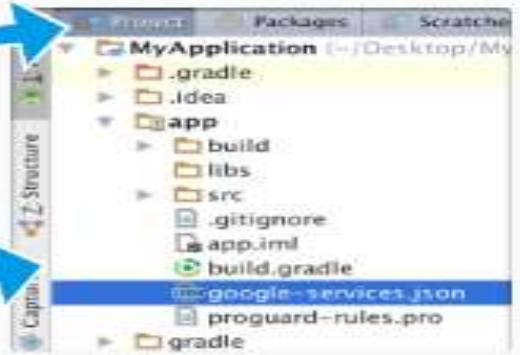
1. Register app
   Android package name: flash_chat.com.flash_chat_flutter_master, App nickname: Flash Chat Flutter

2. **Download and then add config file**
   Instructions for Android Studio below | Unity C++

   Download google-services.json

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

   Move your downloaded `google-services.json` file into your module (app-level) root directory.

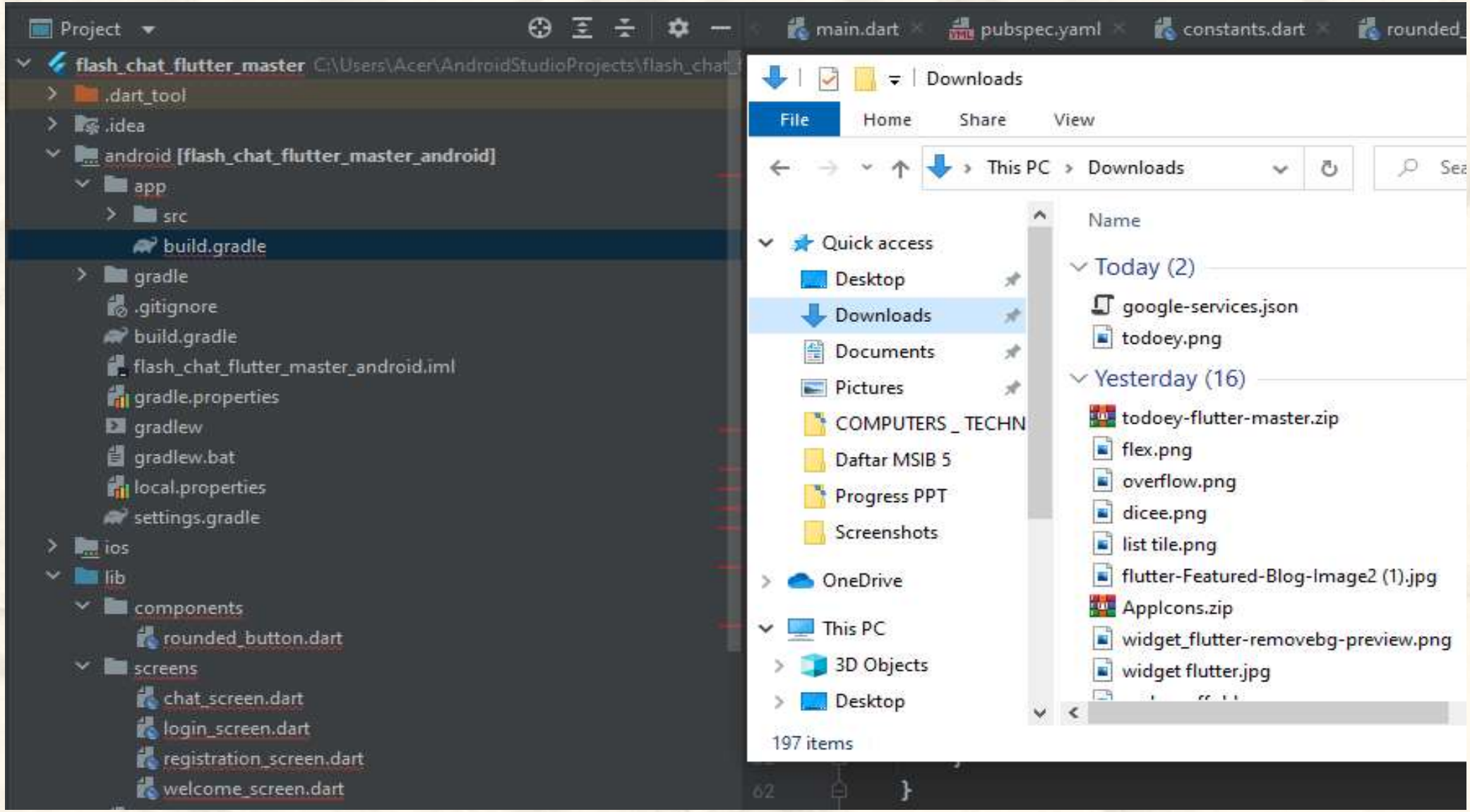


   Next

3. Add Firebase SDK

4. Next steps

MOJADIPRO



Masukkan file google-services.json ke dalam folder app

MOJADIPRO

Add Firebase to your Android app

✓ Register app

Android package name: flash_chat.com.flash_chat_flutter_master, App nickname: Flash Chat Flutter

✓ Download and then add config file

3 Add Firebase SDK

Instructions for Gradle | Unity | C++

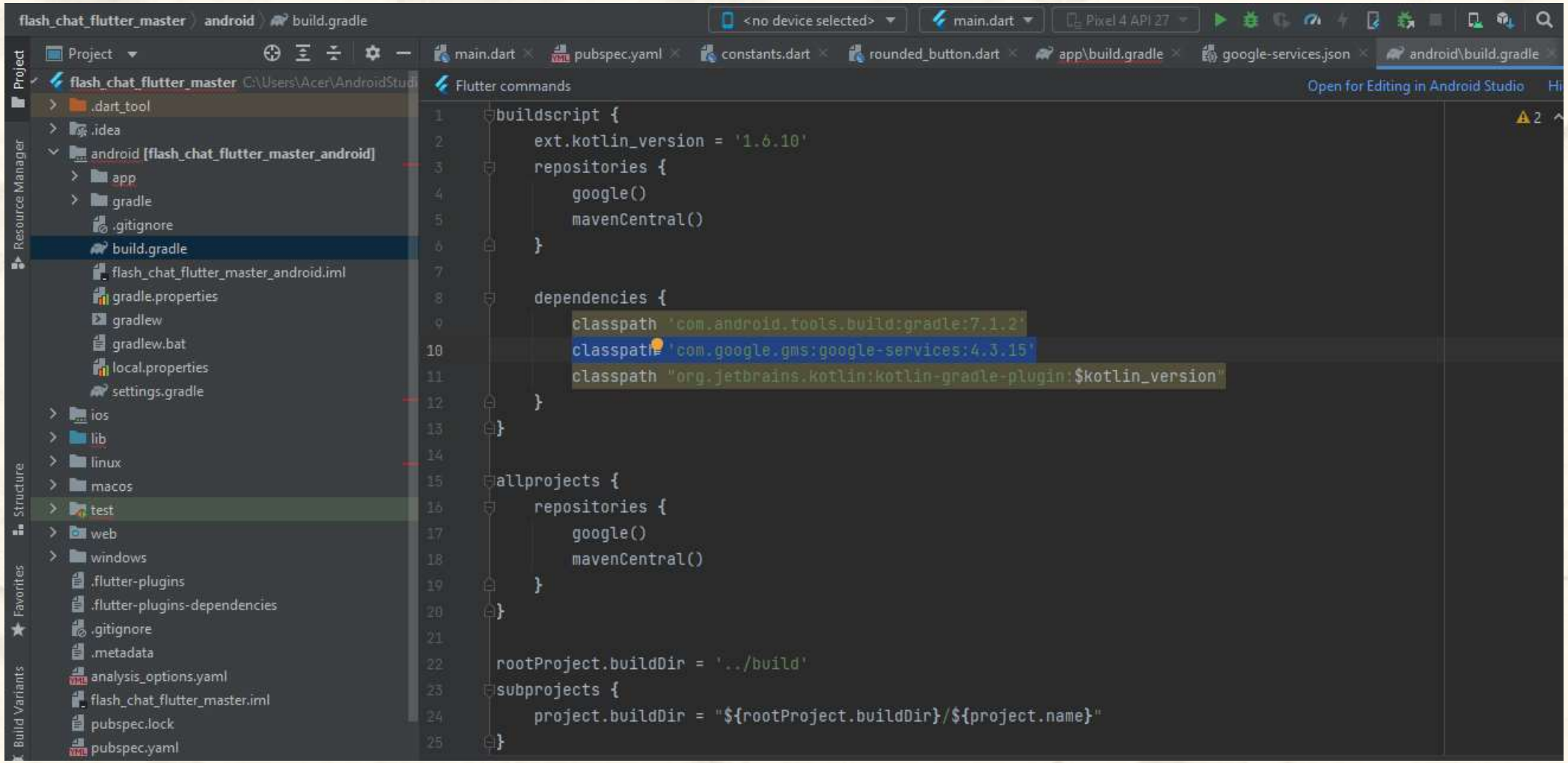
1. To make the `google-services.json` config values accessible to Firebase SDKs, you need the Google services Gradle plugin.

   Add the plugin as a buildscript dependency to your project-level `build.gradle` file:

   Root-level (project-level) Gradle file (`<project>/build.gradle`):

```
buildscript {
  repositories {
    // Make sure that you have the following two repositories
    google()  // Google's Maven repository
    mavenCentral()  // Maven Central repository
  }
  dependencies {
    ...
    // Add the dependency for the Google services Gradle plugin
    classpath 'com.google.gms:google-services:4.3.15'
  }
}

allprojects {
  ...
  repositories {
    // Make sure that you have the following two repositories
    google()  // Google's Maven repository
    mavenCentral()  // Maven Central repository
  }
}
```

MOJADIPRO



Masukkan dependencies pada folder android → build.gradle

MOJADIPRO

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

(•) Kotlin ( ) Java

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

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

dependencies {
  // Import the Firebase BoM
  implementation platform('com.google.firebase:firebase-bom:32.1.0')

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

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

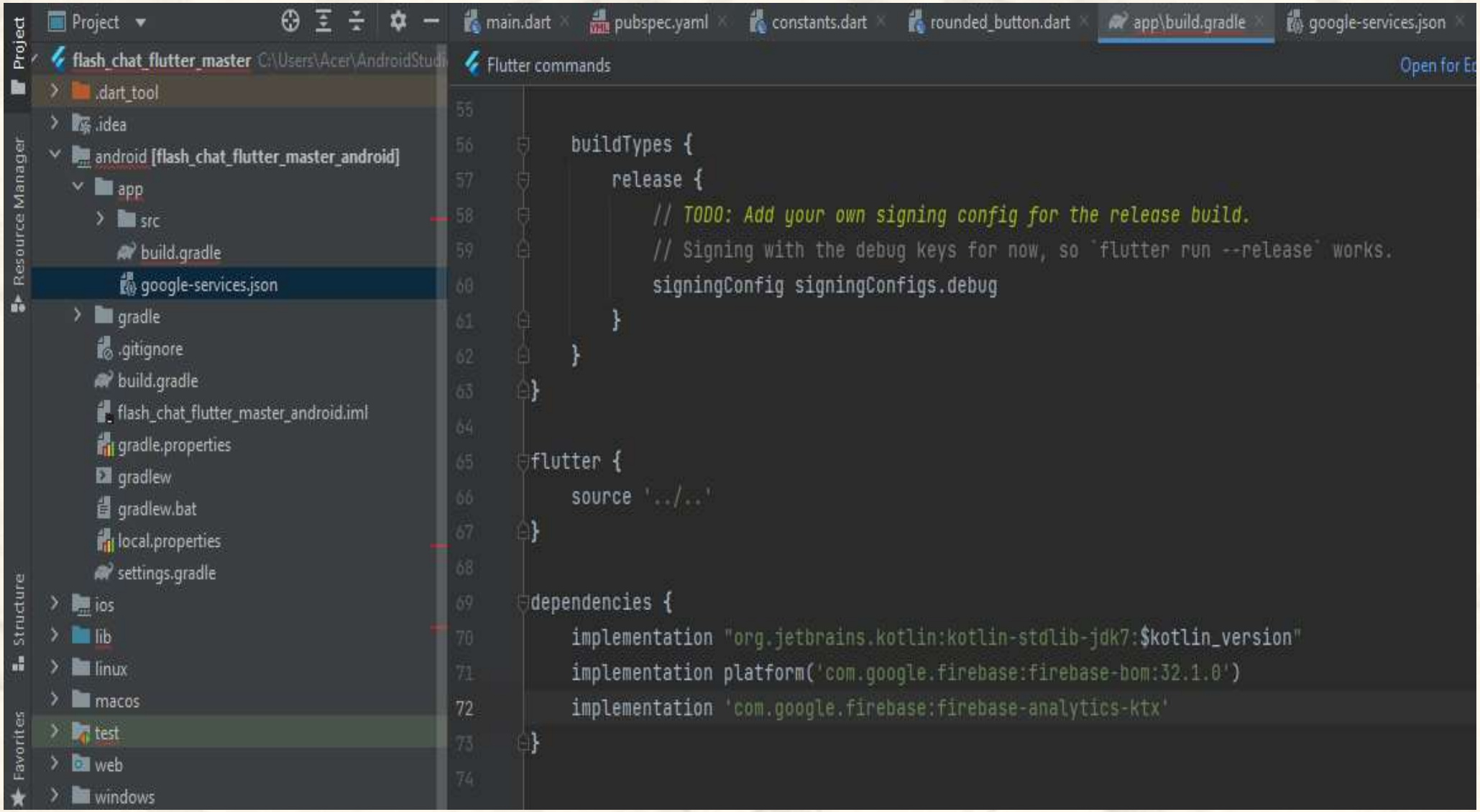
By using the Firebase Android BoM, your app will always use compatible Firebase library versions. Learn more

3. After adding the plugin and the desired SDKs, sync your Android project with Gradle files.

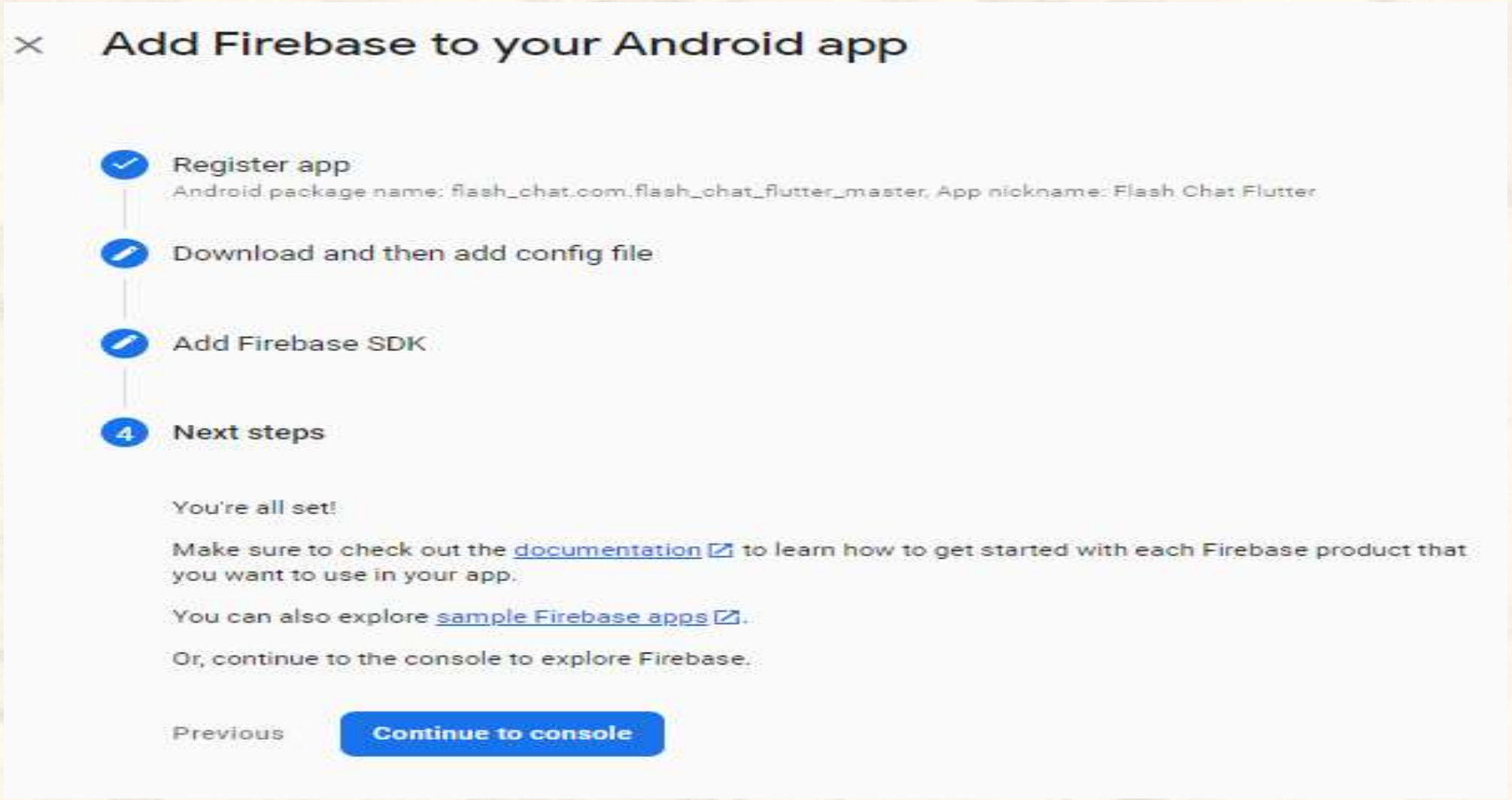
Previous **Next**

4 Next steps

MOJADIPRO



Masukkan plugin pada folder android → build.gradle



Jika sudah run aplikasi dan klik continue to console

MOJADIPRO

TERIMA KASIH