

Modul 9

Integrating Firebase into a Flutter App (3)

Firestore, Writing and Uploading

Module Overview

Membuat proyek yang terhubung dengan Firestore

Module Objectives

Setelah mempelajari dan mempraktikkan modul ini, mahasiswa diharapkan dapat:

- Mengetahui mengenai firestore
- Dapat memproses data document pada Firestore

Firestore database merupakan *NoSQL database* yang artinya sistem manajemen data non-relasional yang tidak memerlukan skema tetap. Pada Firebaase, terdapat 2 cara penyimpanan data dengan perbedaannya masing-masing dalam menangani cara penyimpan data dan penyimpanannya, yaitu *Cloud Firestore* dan *Realtime Database*. Keduanya merupakan *NoSQL database*, tetapi keduanya memiliki arsitektur yang berbeda. Pemilihan penggunaan basis data akan bergantung pada banyak faktor seperti peran database, operasi yang dilakukan pada data, model data, ketersediaan akses, dan jumlah objek dalam database. Akan tetapi kedua database tersebut dapat menawarkan:

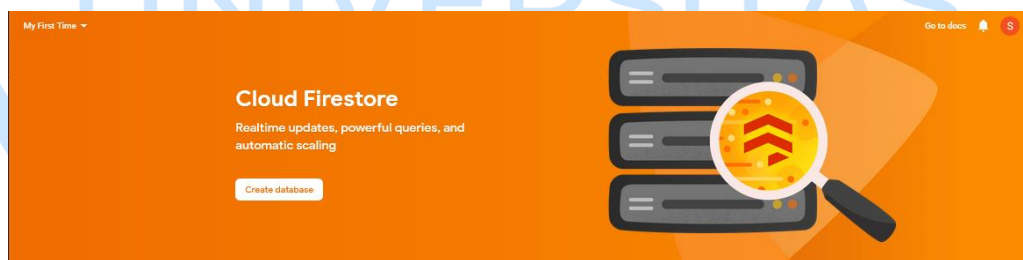
1. Tidak memerlukan server untuk di-deploy dan dipelihara
2. Update bersifat real-time
3. Gratis dalam Batasan penggunaan tertentu

Cloud Firestore adalah pilihan terbaru dan merupakan pilihan yang direkomendasikan untuk sebagian besar proyek baru karena fitur model data yang lebih intuitif, dengan kueri yang lebih cepat dan opsi penskalaan yang lebih ditingkatkan.

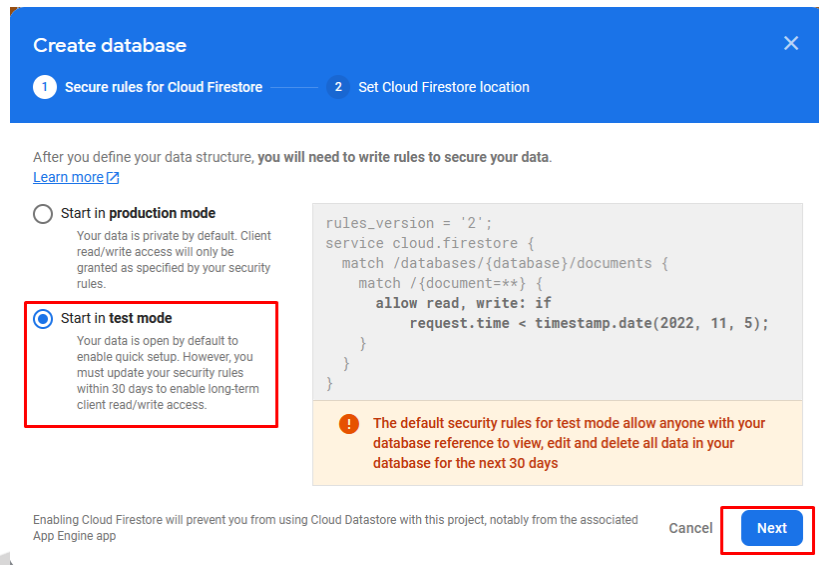
Membuat Pengaturan Firestore

Setelah membuat proyek yang terhubung dengan Firebase Analytics, tambahkan Untuk membuat database Cloud Firestore sebagai cara penyimpanan data dalam database NoSQL, lakukanlah langkah-langkah berikut:

1. Jika anda belum memiliki project Firebase: Di Firebase console, klik Add project dan ikuti petunjuk di layar untuk membuat project Firebase (**Pertemuan 6**)
2. Buka bagian Cloud Firestore di Firebase console.



3. **(jika Proyek lebih dari 1)** Anda akan diminta untuk memilih project Firebase yang ada.
4. Pilih mode awal Cloud Firestore Security Rules Anda menjadi **test mode**, sehingga anda dapat bekerja dengan database selama 30 hari. Selama 30 hari, database anda dapat dilihat, diubah dan dihapus oleh siapa saja. Dan Klik **Next**.



Create database

1 Secure rules for Cloud Firestore — 2 Set Cloud Firestore location

After you define your data structure, you will need to write rules to secure your data.
[Learn more](#)

☐ Start in production mode
Your data is private by default. Client read/write access will only be granted as specified by your security rules.

☒ Start in test mode
Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access.

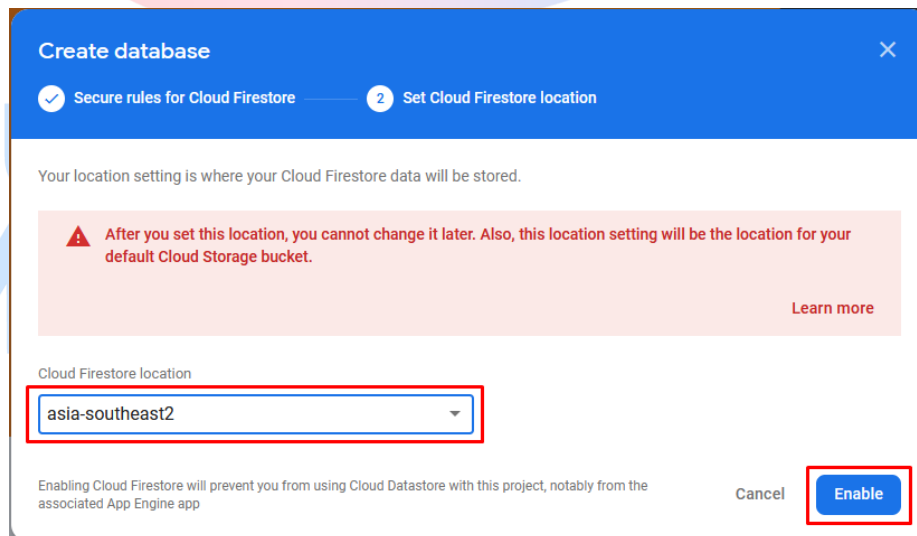
```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if
        request.time < timestamp.date(2022, 11, 5);
    }
  }
}
```

The default security rules for test mode allow anyone with your database reference to view, edit and delete all data in your database for the next 30 days

Enabling Cloud Firestore will prevent you from using Cloud Datastore with this project, notably from the associated App Engine app

Cancel **Next**

5. Tentukan di mana server database yang akan anda gunakan. Anda tidak dapat mengubah posisi server Ketika telah memilih lokasi server anda. Untuk itu, pastikan anda memilih lokasi server terdekat untuk pengguna anda. Pada contoh ini, anda dapat memilih **asia-southeast1 (Singapore)** atau **asia-southeast2 (Jakarta)**. Dan klik **Next**.



Create database

✓ Secure rules for Cloud Firestore — 2 Set Cloud Firestore location

Your location setting is where your Cloud Firestore data will be stored.

After you set this location, you cannot change it later. Also, this location setting will be the location for your default Cloud Storage bucket.

[Learn more](#)

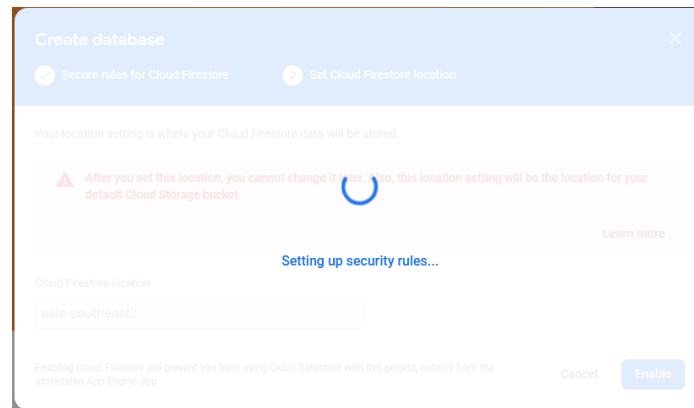
Cloud Firestore location

asia-southeast2

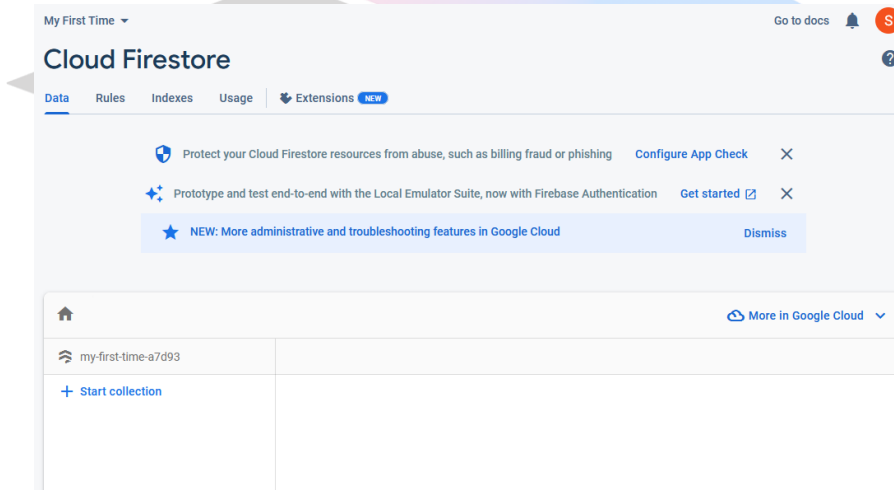
Enabling Cloud Firestore will prevent you from using Cloud Datastore with this project, notably from the associated App Engine app

Cancel **Enable**

6. Tunggu hingga pengaturan Cloud Firestore anda selesai

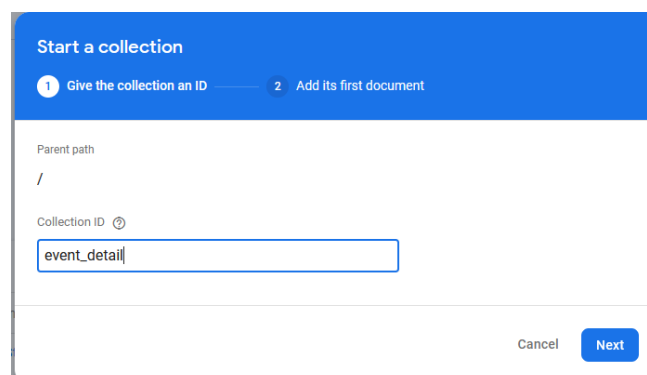


7. Maka tampilan proyek firebase anda akan terlihat seperti ini.

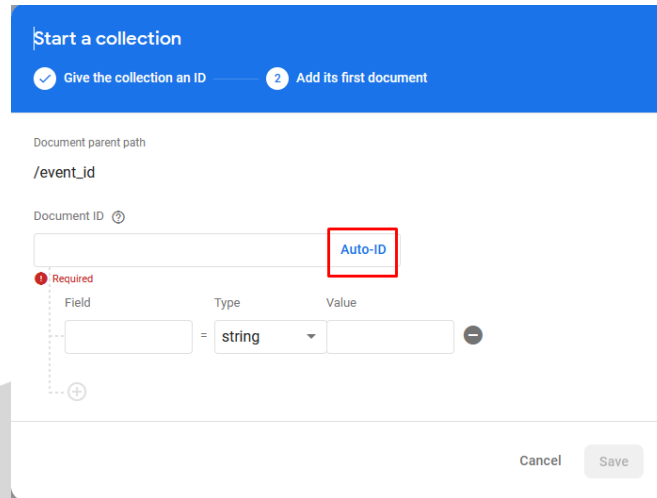


Anda telah berhasil membuat sebuah proyek firebase yang dilengkapi dengan pengolahan data NoSQL Cloud Firestore. Untuk penggunaannya pada Flutter, kita akan mengisikan data di dalam Cloud Firestore terlebih dahulu dengan langkah:

1. Klik pada **Start collection**, dan masukkan **Collection ID** dengan **event_detail** dan klik next



2. Pada Opsi **Document ID**, klik **Auto-ID**, ini akan memudahkan anda dalam membuat id unik untuk setiap data yang anda buat pada *Cloud Firestore*.



Start a collection

✓ Give the collection an ID 2 Add its first document

Document parent path
/event_id

Document ID ⓘ

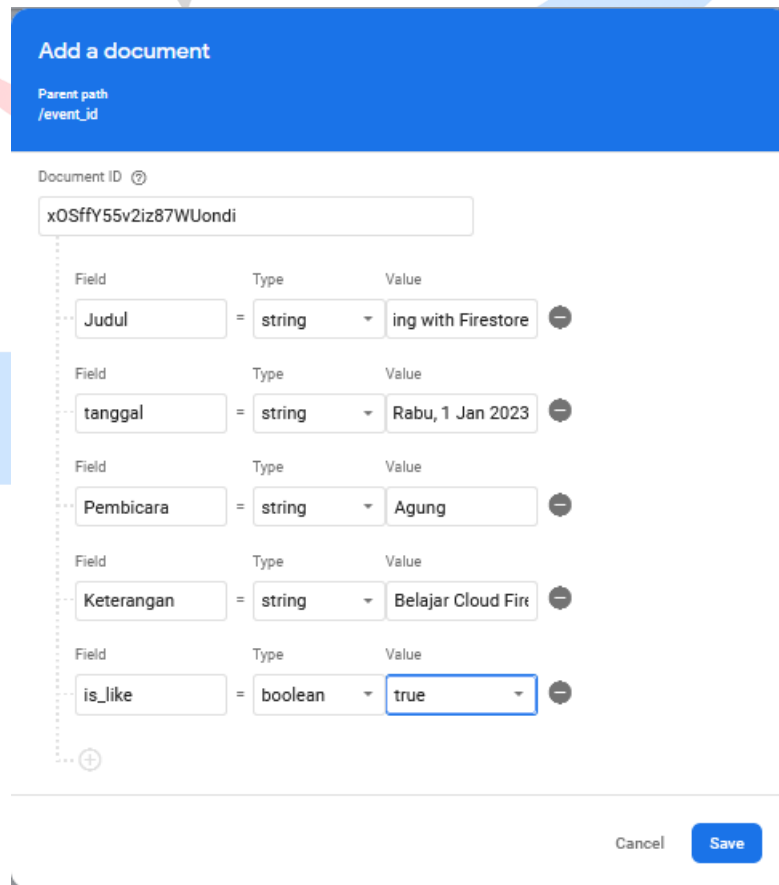
Auto-ID

Required

Field	Type	Value
	string	

Cancel Save

3. lalu tambahkan beberapa kolom, tipe data dan nilai, seperti yang ditunjukkan pada tangkapan layar berikut, lalu klik **Simpan**



Add a document

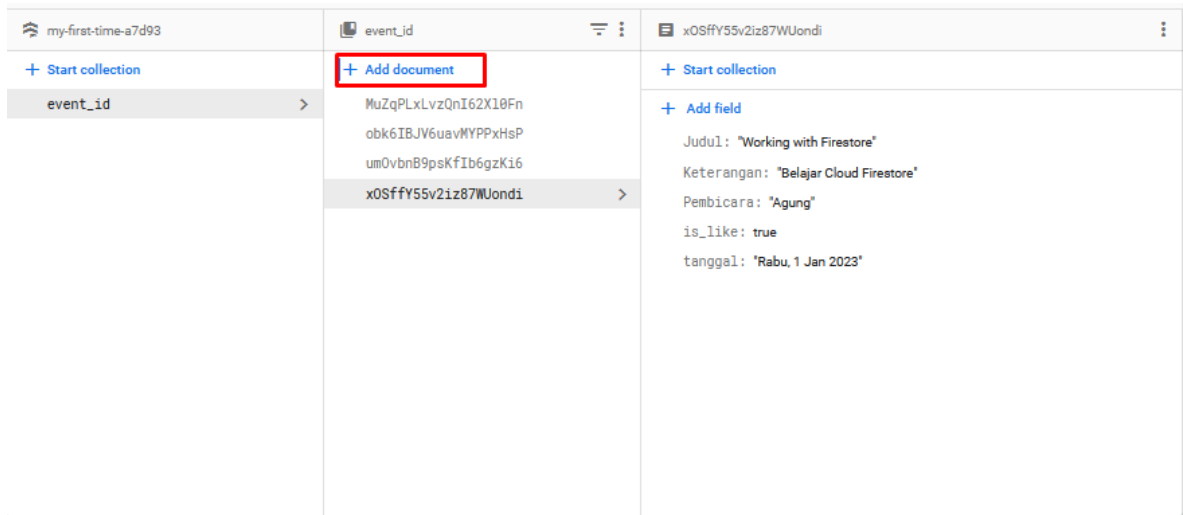
Parent path
/event_id

Document ID ⓘ
x0SffY55v2iz87WUondi

Field	Type	Value
Judul	string	ing with Firestore
tanggal	string	Rabu, 1 Jan 2023
Pembicara	string	Agung
Keterangan	string	Belajar Cloud Fire
is_like	boolean	true

Cancel Save

4. Lakukan hal penambahan dokumen dengan langkah yang sama, sampai anda berhasil memasukkan beberapa nilai yang berbeda untuk field dan tipe data yang sama



Mengatur Project Flutter dengan Cloud Firestore

Setelah membuat proyek Cloud Firebase, anda akan membuat sebuah tampilan aplikasi, dan menghubungkan aplikasi tersebut dengan Firestore anda, dengan cara:

1. Buat Sebuah Class untuk menampung data dari Cloud Firestore

```

1 import 'package:cloud_firestore/cloud_firestore.dart';
2
3 class EventModel {
4   String? id;
5   String judul;
6   String keterangan;
7   String tanggal;
8   bool is_like;
9   String pembicara;
10
11   EventModel(
12     {this.id,
13     required this.judul,
14     required this.keterangan,
15     required this.tanggal,
16     required this.is_like,
17     required this.pembicara});
18
19   Map<String, dynamic> toMap() {
20     return {
21       'Judul': judul,
22       'Keterangan': keterangan,
23       'Pembicara': pembicara,
24       'is_like': is_like,
25       'tanggal': tanggal
26     };
27   }
28 }

```

```

29 EventModel.fromDocSnapshot(DocumentSnapshot<Map<String, dynamic>> doc)
30 : id = doc.id,
31   judul = doc.data()?['Judul'],
32   keterangan = doc.data()?['Keterangan'],
33   pembicara = doc.data()?['Pembicara'],
34   is_like = doc.data()?['is_like'],
35   tanggal = doc.data()?['tanggal'];
36 }
37

```

2. Buat Proyek Flutter baru anda dan buat tampilan aplikasi Frond-End Anda sebagai berikut :

```

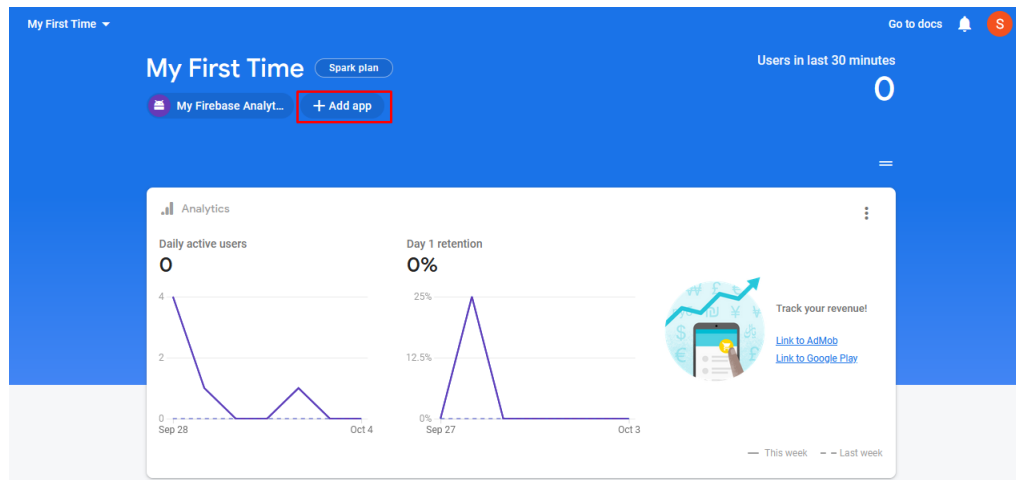
1 import 'package:cloud_firestore/cloud_firestore.dart';
2 import 'package:fab_circular_menu/fab_circular_menu.dart';
3 import 'package:firebase_core/firebase_core.dart';
4 import 'package:flutter/material.dart';
5 import 'package:flutter_my_firestore/Praktek/EventModel.dart';
6
7 class MyHome extends StatefulWidget {
8   const MyHome({Key? key}) : super(key: key);
9
10  @override
11  State<MyHome> createState() => _MyHomeState();
12 }
13
14 class _MyHomeState extends State<MyHome> {
15   List<EventModel> details = [];
16   @override
17   Widget build(BuildContext context) {
18     return Scaffold(
19       appBar: AppBar(title: Text("Cloud Firestore")),
20       body: ListView.builder(
21         itemCount: details.length,
22         itemBuilder: (context, position) {
23           return CheckboxListTile(
24             onChanged: (bool? value) {},
25             value: null,
26           );
27         },
28       ),
29       floatingActionButton: FabCircularMenu(children: <Widget>[
30         IconButton(icon: Icon(Icons.add), onPressed: () {}),
31         IconButton(icon: Icon(Icons.minimize), onPressed: () {}),
32       ]),
33     );
34   }
35 }
36

```

3. Untuk memasukkan Fab dengan membuat menu melingkar yang bagus, anda dapat menggunakan

```
fab_circular_menu: ^1.0.2
```

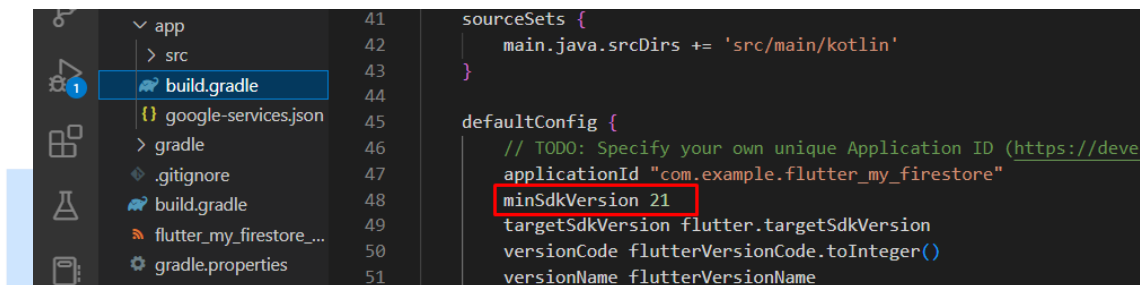
4. Langkah selanjutnya, hubungkan proyek flutter anda dengan firebase dengan mengikuti langkah-langkah yang diberikan. **(Pertemuan 6)**



- Masukkan pada pubspec.xml anda, 2 library yang akan kita gunakan untuk bekerja dengan firestore, yaitu:

```
cloud_firestore: ^3.5.0
firebase_core: ^1.24.0
```

- Pastikan, minimumSDK anda pada build.gradle pada level app merupakan minimumSDK 21, dikarenakan Firestore dapat bekerja dengan baik pada SDK terkecil adalah 21.



- Untuk mencoba, apakah koneksi ke dalam Cloud Firestore anda berhasil, anda dapat mencoba melakukan koneksi ke dalam firebase, dan mengambil data dokument yang anda simpan dengan menginisialisasi Firebase, dan membuat instance dari Firestore dalam sebuah fungsi Future asinkron.

```
36 Future testData() async {
37   await Firebase.initializeApp();
38   print('init Done');
39   FirebaseFirestore db = await FirebaseFirestore.instance;
40   print('init Firestore Done');
```


8. Untuk membaca dokumen yang anda buat, anda dapat menggunakan instance dari Firestore, untuk membaca koleksi dokumen anda dengan memasukkan nama koleksi anda dalam firestore yaitu **event_id**, dan menggunakan perintah **get()**, untuk mendapatkan semua isi dokument pada **event_id** anda.

```

41   var data = await db.collection('event_id').get().then((event) {
42     for (var doc in event.docs) {
43       print("${doc.id} => ${doc.data()}");
44     }
45   });
46 }

```

9. Terakhir, jalankan fungsi **testData()** pada Widget Build anda

```

17   Widget build(BuildContext context) {
18     testData();
19     return Scaffold(
20       appBar: AppBar(title: Text("Cloud Firestore")),
21       body: ListView.builder(
22         itemCount: (details != null) ? details.length : 0,

```

10. Jalankan aplikasi, dan perhatikan, pada bagian **debug console** anda, semua data yang telah anda isikan, telah terbaca.

```

I/flutter ( 9442): MuZqPLxLvzQnI62Xl0Fn => {Keterangan: Pengenalan Firebase pada Flutter, is_like: false, Judul: F
irebase Intro, tanggal: Timestamp(seconds=1673024400, nanoseconds=233000000), Pembicara: Fandi}
I/flutter ( 9442): obk6IBJV6uavMYPPxHsP => {Keterangan: Pengembangan Aplikasi Mobile Menggunakan Flutter, is_like:
true, Judul: Flutter, tanggal: Timestamp(seconds=1667109600, nanoseconds=536000000), Pembicara: Andi}
I/flutter ( 9442): umOvbnB9psKfIb6gzKi6 => {Keterangan: Pengenalan Cloud Firestore, is_like: true, Judul: Cloud Fi
restore, tanggal: Timestamp(seconds=1674630000, nanoseconds=103000000), Pembicara: Fandi}

```

Menampilam data Cloud Firestore ke dalam Aplikasi

Setelah berhasil melakukan koneksi ke dalam Cloud Firebase, anda dapat menampilkan hasil bacaan anda, kedalam aplikasi yang anda bangun, dengan memanfaatkan Model data yang telah kita buat, yaitu dengan cara

1. Buat fungsi untuk membaca data untuk disimpan pada list detail anda.

```

48 Future readData() async {
49   await Firebase.initializeApp();
50   FirebaseFirestore db = await FirebaseFirestore.instance;
51   var data = await db.collection('event_id').get();
52   setState(() {
53     details =
54       data.docs.map((doc) => EventModel.fromDocSnapshot(doc)).toList();
55   });
56 }

```

Dengan bantuan kelas Model kita, yaitu fungsi **fromDocSnapshot()**, anda dapat mengembalikan semua nilai yang ada pada format Json menjadi objek data class

EventModel

2. Jalankan fungsi ini pada lifecycle **initstate()**

```

17 @override
18 void initState() {
19   readData();
20   super.initState();
21 }
22

```

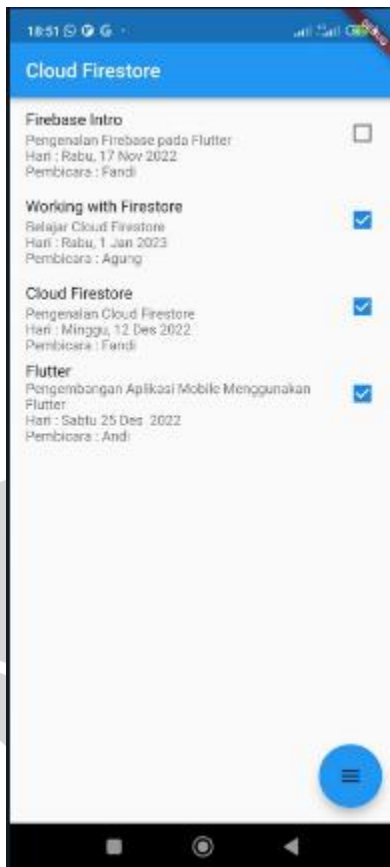
3. Rancanglah tampilan dari ListViewBuilder anda dengan memanfaatkan objek dari kelas data **EventModel** anda, seperti berikut ini

```

27 body: ListView.builder(
28   itemCount: details.length,
29   itemBuilder: (context, position) {
30     return CheckboxListTile(
31       onChanged: (bool? value) {},
32       value: details[position].is_like,
33       title: Text(details[position].judul),
34       subtitle: Text("${details[position].keterangan}" +
35         "\nHari : ${details[position].tanggal}" +
36         "\nPembicara : ${details[position].pembicara}"),
37       isThreeLine: true,
38     );
39   },
40 ),

```

Setelah itu, jalankan aplikasi anda, maka anda telah berhasil menampilkan data dari Cloud Firestore anda ke dalam aplikasi anda.



Operasi Insert Cloud Firestore

Untuk operasi insert buatlah fungsi berikut kedalam **main.dart** anda.

1. Untuk proses tambah data anda akan menggunakan fungsi **add**.

```

74  addRand() async {
75    FirebaseFirestore db = await FirebaseFirestore.instance;
76    EventModel InsertData = EventModel(
77      judul: getRandString(5),
78      keterangan: getRandString(30),
79      tanggal: getRandString(10),
80      is_like: Random().nextBool(),
81      pembicara: getRandString(20));
82    await db.collection("event_id").add(InsertData.toMap());
83    setState(() {
84      details.add(InsertData);
85    });
86  }

```

```

88 String getRandString(int len) {
89     var random = Random.secure();
90     var values = List<int>.generate(len, (i) => random.nextInt(255));
91     return base64UrlEncode(values);
92 }

```

Proses **add()** pada firestore, mewajibkan anda menentukan letak koleksi yang anda ingin tambahkan data. Penambahan data dapat dilakukan dengan membuat sebuah objek dari **EventModel**, pada variable insertData, dan dengan bantuan kelas Model Data EventModel yaitu **toMap()**, untuk mengubah objek EventModel menjadi Map yang dapat digunakan untuk memerintahkan memasukkan data baru ke dalam Cloud Firestore.

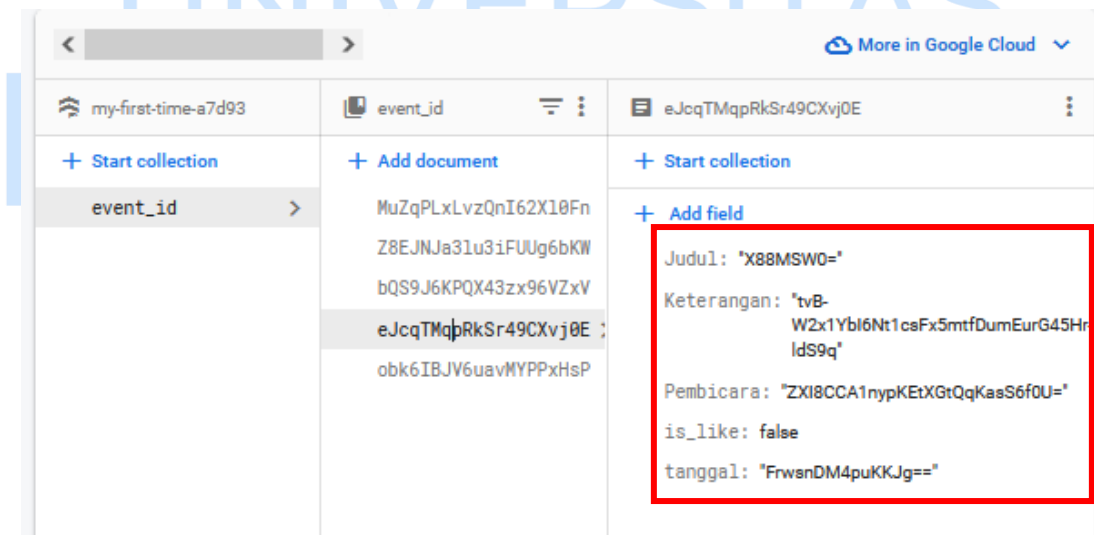
2. Catatan : Fungsi **getRandString()** digunakan untuk membuat string acak sebagai inisialisasi nilai untuk objek EventModel
3. Letakkan fungsi ini pada FAB tombol add anda.

```

47 floatingActionButton: FabCircularMenu(children: <Widget>[
48     IconButton(
49         icon: Icon(Icons.add),
50         onPressed: () {
51             addRand();
52         },

```

4. Jalankan aplikasi, dan klik tombol add pada Fab. Maka anda akan melihat data anda telah bertambah baik di Firestore maupun di aplikasi anda.





Operasi Delete Cloud Firestore

Untuk operasi hapus, kita akan mencoba menghapus data terakhir yang masuk tersimpan di dalam Cloud Firestore. Untuk hapus, buatlah fungsi berikut kedalam **main.dart** anda.

1. Untuk proses tambah data anda akan menggunakan fungsi **delete**.

```

94  deleteLast(String documentId) async {
95    FirebaseFirestore db = await FirebaseFirestore.instance;
96    await db.collection("event_id").doc(documentId).delete();
97    setState(() {
98      details.removeLast();
99    });
100 }

```

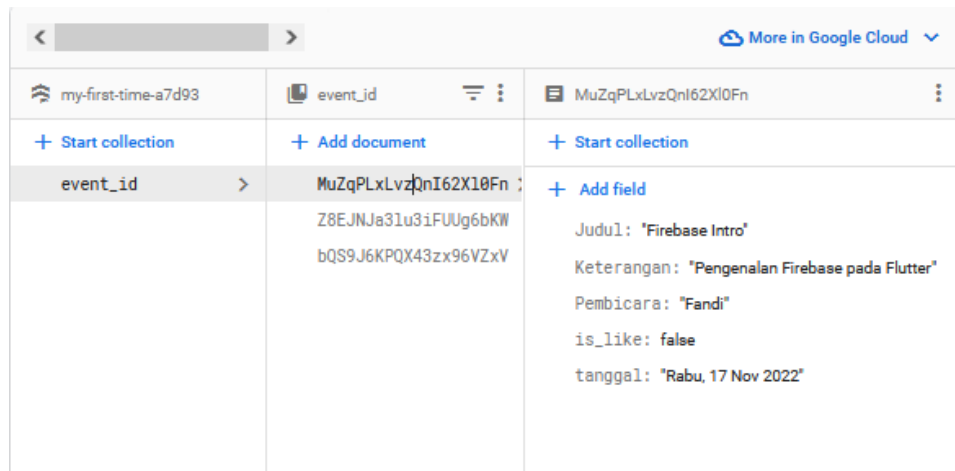
2. Proses delete akan berdasarkan dokument id yang dikirim melalui pemanggil. Oleh karean itu, gunakan fungsi deleteLast() dengan mengirimkan data terakhir yang ada dalam detail anda, pada fungsi onPressed fab minimize anda.

```

53     IconButton(
54       icon: Icon(Icons.minimize),
55       onPressed: () {
56         if (details.last.id != null) {
57           deleteLast(details.last.id!);
58         }
59       })

```

3. Jalankan aplikasi, dan klik tombol minimize pada Fab. Maka anda akan melihat data anda telah berkurang baik di Firestore maupun di aplikasi anda.



Operasi Update Cloud Firestore

Untuk operasi update buatlah fungsi berikut kedalam **main.dart** anda.

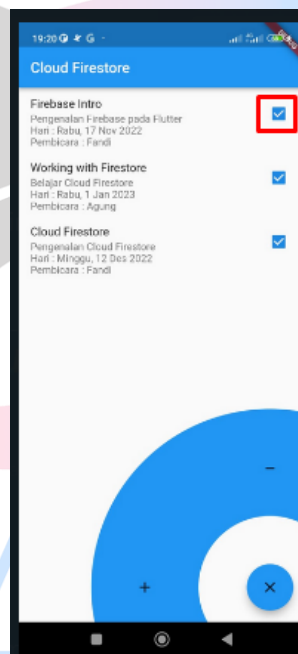
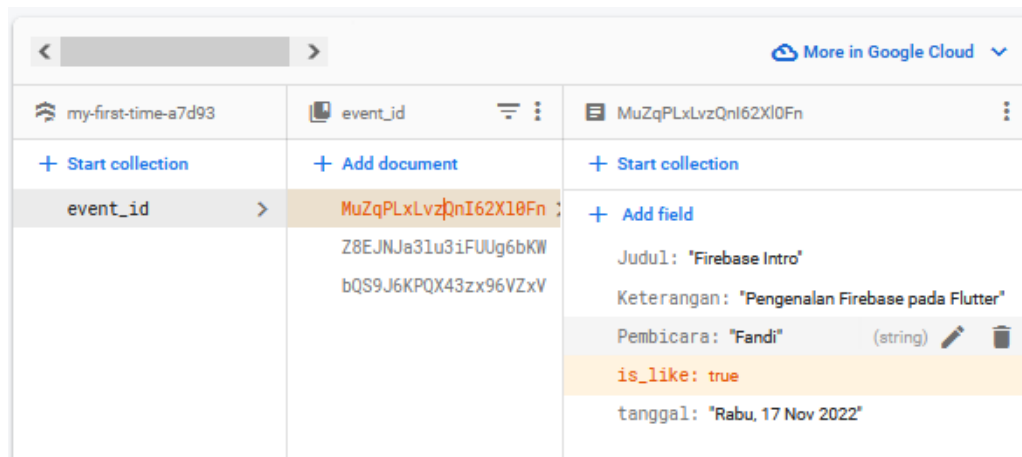
1. Untuk proses tambah data anda akan menggunakan fungsi **update**.

```
102 updateEvent(int pos) async {  
103   FirebaseFirestore db = await FirebaseFirestore.instance;  
104   await db  
105     .collection("event_id")  
106     .doc(details[pos].id)  
107     .update({'is_like': !details[pos].is_like});  
108   setState(() {  
109     details[pos].is_like = !details[pos].is_like;  
110   });  
111 }
```

2. Proses update dilakukan dengan menentukan id dokument, dan mengguakan perintah **update()**. Adapun parameter di dalam update() merupakan objek Map, yang dapat anda isikan sesuai dengan kebutuhan anda, menentukan data yang ingin anda ubah atau tambahkan kedalam dokument id. Untuk menjalankan fungsi tersebut, anda dapat berikan pada fungsi **onChange** pada CheckboxListTile anda.

```
34   return CheckboxListTile(  
35     onChanged: (bool? value) {  
36       updateEvent(position);  
37     },  
38     value: details[position].is_like,  
39     title: Text(details[position].judul),  
40     subtitle: Text("${details[position].keterangan}" +  
41       "\nHari : ${details[position].tanggal}" +  
42       "\nPembicara : ${details[position].pembicara}"),  
43     isThreeLine: true,  
44   );
```

3. Jalankan aplikasi, dan klik **Checkbox List Tile** anda. Maka anda akan melihat tanda centang pada Checkbox akan berubah sesuai dengan pilihan anda baik di Firestore maupun di aplikasi anda.



Latihan

1. Setelah menyelesaikan proyek di atas, rancanglah UI untuk melakukan proses penghapusan, update data dan delete dengan cara anda, yang berbeda dari contoh yang diberikan. Anda boleh menggunakan custom dialog, ataupun dengan ui tambahan lainnya (Menambahkan button di dalam ListView). Pastikan data yang telah anda tambahkan atau hapus dapat muncul pada Aplikasi anda.