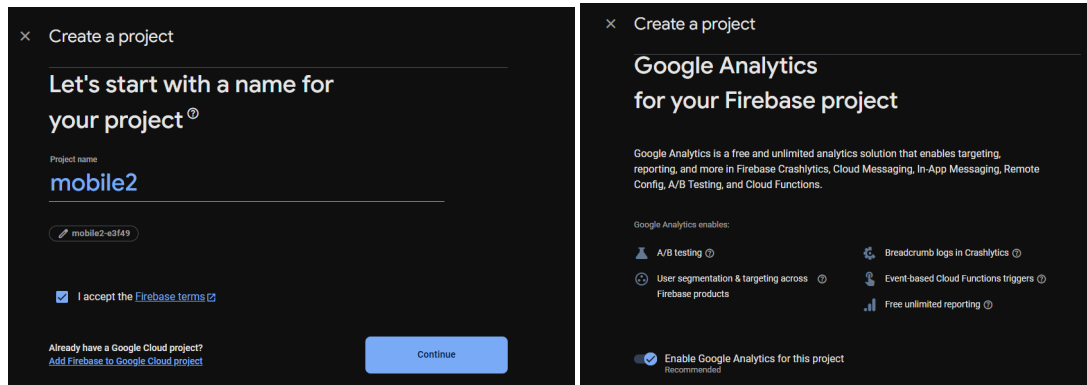


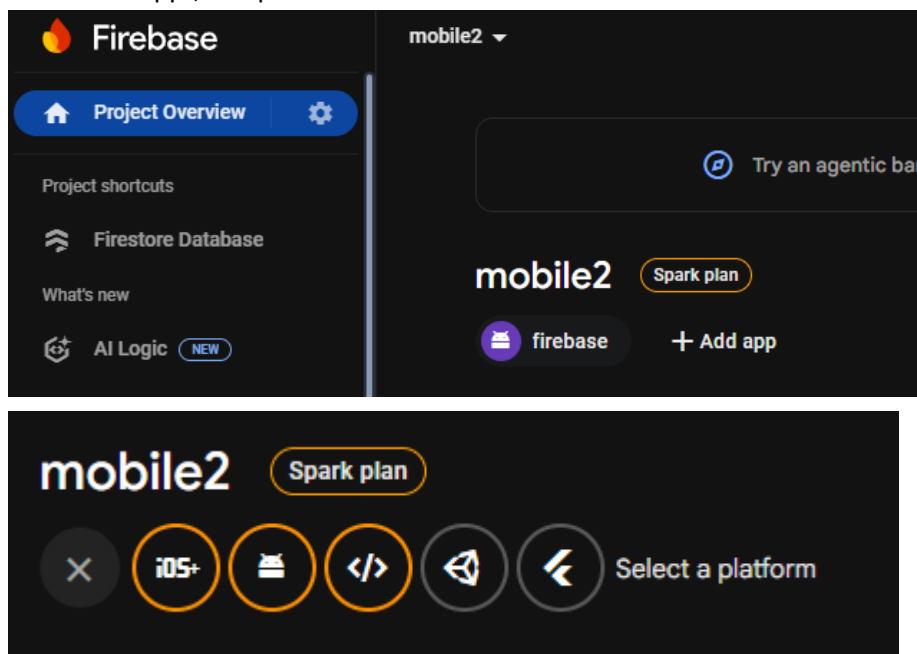
TUGAS 5

NIM : 22.230.0001
NAMA : Fadhli Hilman Saputra
KELAS : 7P51

1. Membuat akun firebase pada <https://console.firebase.google.com/u/0/mfa>
2. Buat project baru dengan firebase

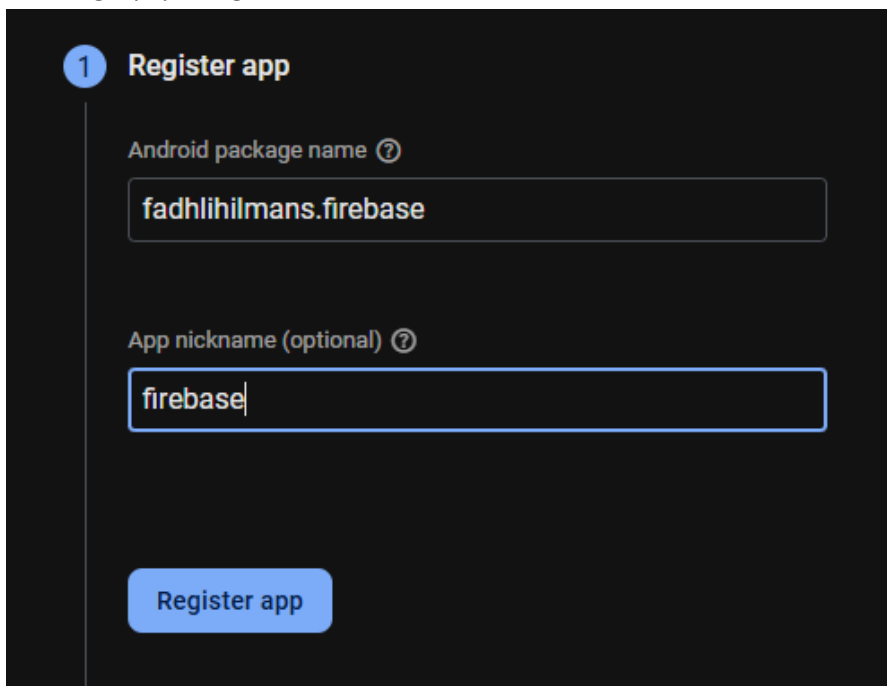


3. Pilih "+add app", lalu pilih "icon android"



4. Ikuti step berikut:

- Lengkapi package name,



1 Register app

Android package name ?

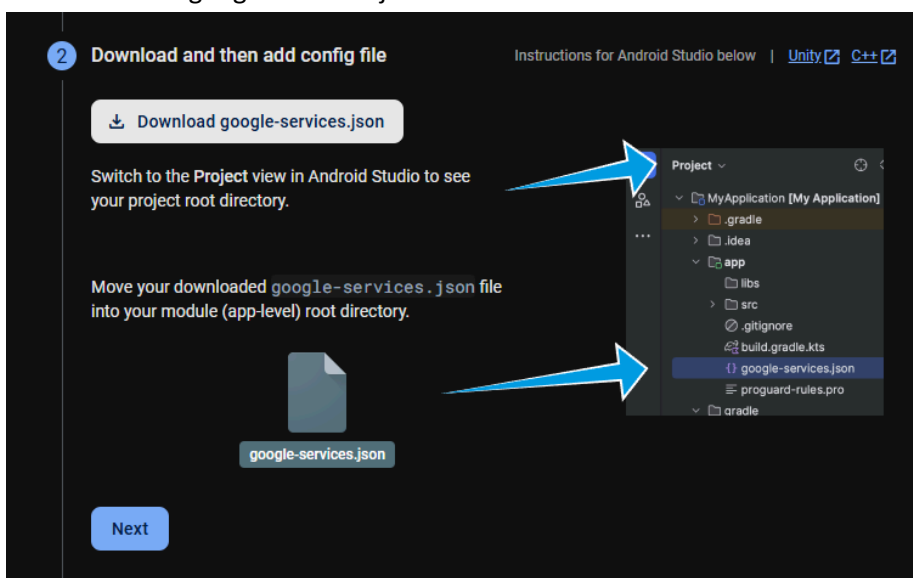
fadhlihilmans.firebase

App nickname (optional) ?

firebase

Register app

- download google-services.json



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.

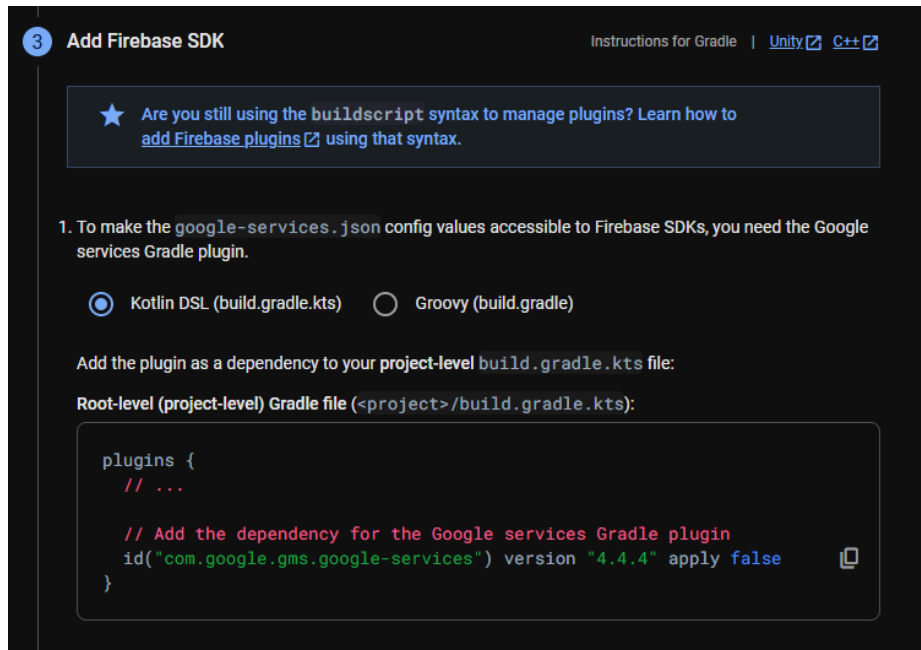
google-services.json

Next

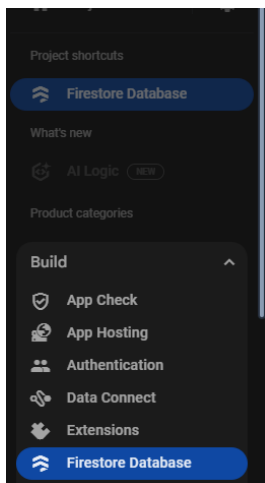
Project view in Android Studio showing the file structure:

- MyApplication [My Application]
- gradle
- .idea
- app
 - libs
 - src
 - .gitignore
 - build.gradle.kts
 - google-services.json
 - proguard-rules.pro
- gradle

- selanjutnya pilih kotlin dsl (build.gradle.kts)

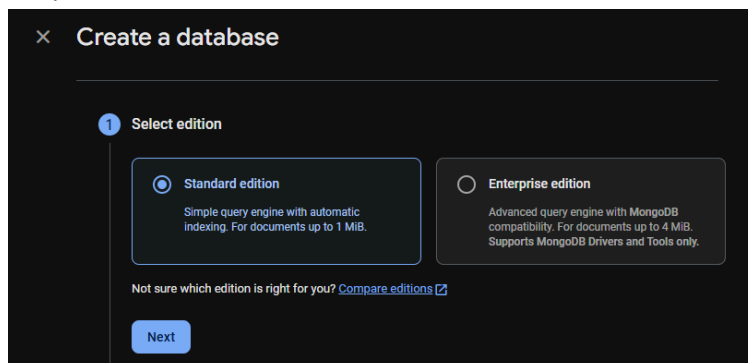


5. Buat database baru dengan cara pilih build ☐ firebase database pada sidebar



6. Selanjutnya create database dan ikuti step berikut:

- pilih standart edition



- ID location biarkan default

× Create a database

2 Database ID & location

Database ID

(default)

Location

nam5 (United States)

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

⚠ After you set this location, you cannot change it later. [Learn more](#)

Next

- Pilih Start in test mode

× Create a database

3 Configure

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(2025, 11, 26);
    }
  }
}
```

⚠ 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

Cancel Create

7. Selanjutnya pilih Start Collection

- Collection id isi “mahasiswa” sebagai nama table

Start a collection

1 Give the collection an ID — 2 Add its first document

Parent path

/

Collection ID ⓘ

mahasiswa

Cancel Next

- Inputkan nim, nama, prodi, kelas serta tipe data dan value

Start a collection

✓ Give the collection an ID — 2 Add its first document

Document parent path
/mahasiswa

Document ID ②
1

Field Type

nim = string

String
22.230.0001

Field Type

nama = string

String
fadhli hilman saputra

Field Type

prodi = string

String
Sistem Informasi

Field Type

kelas = string

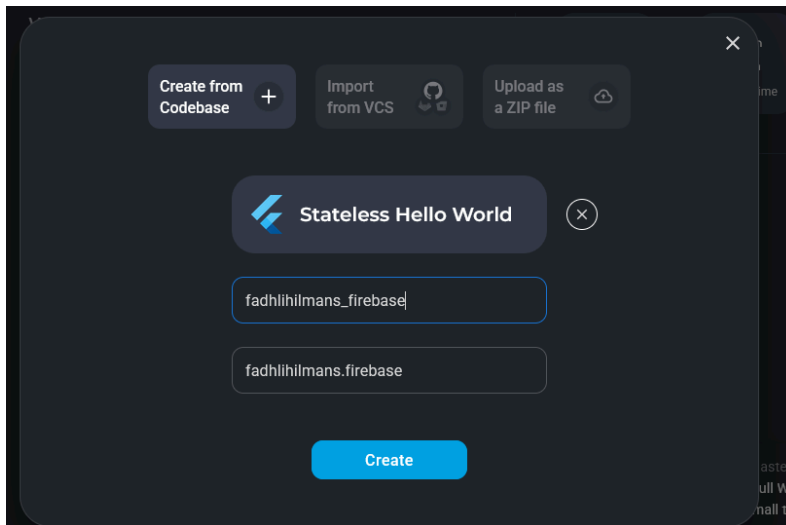
String
7P51

... Add field

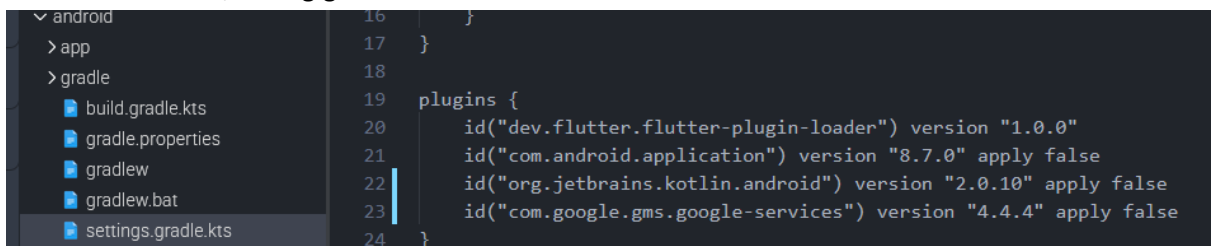
Cancel Save

🏠 > mahasiswa > 1 ✎	More in Google Cloud ▾	
(default)	mahasiswa	1
+ Start collection	+ Add document	+ Start collection
mahasiswa >	1 >	+ Add field
		kelas: "7P51"
		nama: "fadhli hilman saputra"
		nim: "22.230.0001"
		prodi: "Sistem Informasi"

8. Buat project pada <https://flutlab.io/>
9. Create project, sesuaikan packagename dengan penamaan android packagename pada firebase sebelumnya, disini saya menggunakan nama "fadhlilhilmans.firebase"



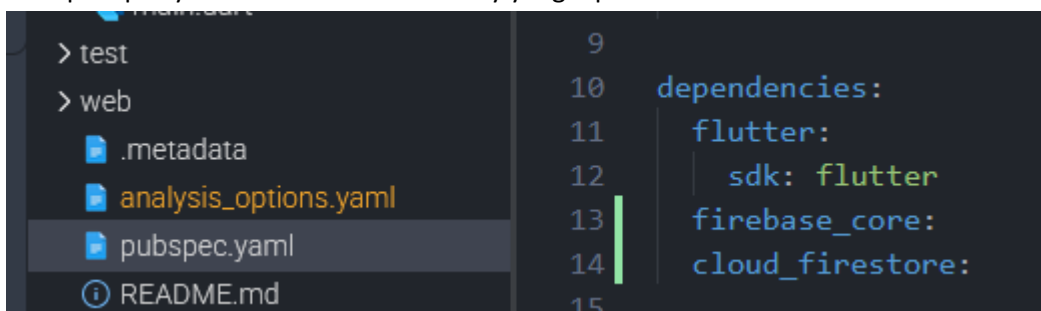
10. Pada file : android/setting.gradle.kts



Penjelasan :

- Ubah Versi Kotlin Menjadi: 2.0.10 agar compatible dengan versi terbaru
- Tambahkan **id("com.google.gms.google-services") version "4.4.4"** apply false
Ini adalah plugin Google Services. Ia digunakan untuk memproses file konfigurasi Google (seperti google-services.json) dan secara otomatis menambahkan dependensi yang diperlukan untuk layanan seperti Firebase

11. Buka pubspec.yaml lalu tambahkan library yang diperlukan:



Penjelasan :

- **firebase_core** : Diperlukan sebagai inisialisasi layanan Firebase. wajib menyertakan package ini agar aplikasi Flutter dapat terhubung dengan firebase
- **cloud_firestore** : Diperlukan secara spesifik untuk menggunakan layanan basis data dari Google, yaitu Cloud Firestore

12. Buka file : android/app/build.gradle.kts

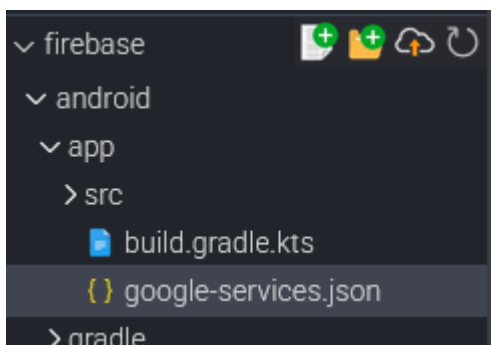
- Tambahkan id("com.google.gms.google-services")

```
1  plugins {  
2      id("com.android.application")  
3      id("kotlin-android")  
4      // The Flutter Gradle Plugin must be applied  
5      id("dev.flutter.flutter-gradle-plugin")  
6      id("com.google.gms.google-services")  
7  }  
8
```

- Tambahkan dependensi berikut:

```
dependencies {  
    implementation(platform("com.google.firebase:firebase-bom:34.4.0"))  
    implementation("com.google.firebase:firebase-analytics")  
}
```

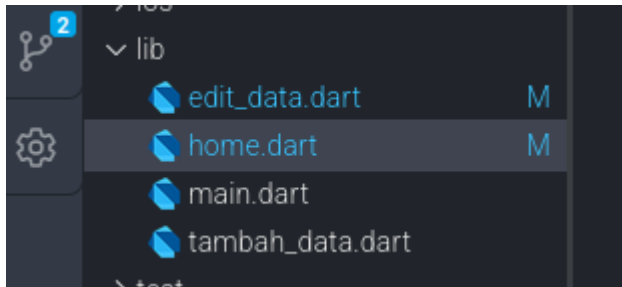
13. Letakkan (upload) file google-services.json yang sudah didownload pada android/app/



14. Ganti minSdkVersion di file android/app/build.gradle.kts menjadi 23 agar dapat menggunakan fitur Firebase

```
defaultConfig {  
    // TODO: Specify your own unique Application ID (https://developer.android.com/studio/run/application-id)  
    applicationId = "fadhlihilman.firebase"  
    // You can update the following values to match your application's requirements:  
    // For more information, see: https://flutter.dev/to/review-gradle-config  
    // minSdk = flutter.minSdkVersion  
    minSdk = 23  
    targetSdk = flutter.targetSdkVersion  
    versionCode = flutter.versionCode  
    versionName = flutter.versionName  
}
```


15. pada folder "lib/" siapkan halaman "tambah_data.dart", "edit_data.dart", "home.dart", "main.dart"



16. Menyiapkan halaman tambah_data.dart

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

class TambahData extends StatefulWidget {
  @override
  _TambahDataState createState() => _TambahDataState();
}

class _TambahDataState extends State<TambahData> {
  TextEditingController nimController = TextEditingController();
  TextEditingController namaController = TextEditingController();
  TextEditingController prodiController = TextEditingController();
  TextEditingController kelasController = TextEditingController();

  void addData() {
    DocumentReference documentReference = FirebaseFirestore.instance
      .collection('mahasiswa')
      .doc(nimController.text);

    Map<String, dynamic> mhs = ({
      "nim": nimController.text,
      "nama": namaController.text,
      "prodi": prodiController.text,
      "kelas": kelasController.text,
    });

    documentReference
      .set(mhs)
      .whenComplete(() => print('${nimController.text} created'));

    Navigator.pop(context);
  }
}
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("ADD DATA"),
    ),
    body: Padding(
      padding: EdgeInsets.all(10.0),
      child: ListView(
        children: <Widget>[
          Text(
            "Input Data Mahasiswa",
            style: TextStyle(
              color: Colors.red,
              fontStyle: FontStyle.italic,
              fontWeight: FontWeight.bold,
              fontSize: 25,
            ),
          ),
          SizedBox(
            height: 40,
          ),
          TextFormField(
            controller: nimController,
            decoration: InputDecoration(
              labelText: "NIM",
            ),
          ),
          TextFormField(
            controller: namaController,
            decoration: InputDecoration(labelText: "Nama"),
          ),
          TextFormField(
            controller: prodiController,
            decoration: InputDecoration(labelText: "Prodi"),
          ),
          TextFormField(
            controller: kelasController,
            decoration: InputDecoration(labelText: "Kelas"),
          ),
          SizedBox(
            height: 50,
          ),
        ],
      ),
    ),
  );
}
```

```

    ),
    ElevatedButton(
      onPressed: () {
        addData();

        Navigator.pushAndRemoveUntil(
          context,
          MaterialPageRoute(builder: (context) => Home()),
          (Route<dynamic> route) => false,
        );
      },
      child: Text("Simpan"),
    ),
  ],
),
);
}
}

```

Penjelasan :

```

Windsurf: Refactor | Explain
class TambahData extends StatefulWidget {
  @override
  _TambahDataState createState() => _TambahDataState();
}

```

- Mendeklarasikan kelas TambahData sebagai StatefulWidget,

```

Windsurf: Refactor | Explain
class _TambahDataState extends State<TambahData> {
  TextEditingController nimController = TextEditingController();
  TextEditingController namaController = TextEditingController();
  TextEditingController prodiController = TextEditingController();
  TextEditingController kelasController = TextEditingController();
}

```

- Membuat sebuah objek controller untuk TextFormField, yaitu kolom input untuk nim, nama, prodi, kelas. Objek ini berfungsi sebagai jembatan untuk teks yang diinput oleh pengguna pada kolom tersebut.

```

void addData() {
  DocumentReference documentReference = FirebaseFirestore.instance
    .collection('mahasiswa')
    .doc(nimController.text);

  Map<String, dynamic> mhs = ({
    "nim": nimController.text,
    "nama": namaController.text,
    "prodi": prodiController.text,
    "kelas": kelasController.text,
  });

  documentReference
    .set(mhs)
    .whenComplete(() => print('${nimController.text} created'));

  Navigator.pop(context);
}

```

- "void addData()" Mendefinisikan sebuah metode (fungsi) bernama addData. Metode ini berisi logika utama untuk mengambil data dari semua controller dan memproses penyimpanannya ke database Cloud Firestore.
- "DocumentReference documentReference = FirebaseFirestore.instance.collection('mahasiswa').doc(nimController.text);" membuat sebuah deklarasi spesifik di dalam database Cloud Firestore. Ini memberitahu sistem untuk Mengakses instance (layanan) FirebaseFirestore, Melihat ke dalam collection (nama table) bernama mahasiswa, Menargetkan sebuah dokumen yang ID-nya diambil dari teks yang ada di nimController.
- "Map<String, dynamic> mhs = ({ ... });" Membuat sebuah objek Map (struktur data dengan format key-value) yang berisi data mahasiswa. "nim", "nama", "prodi", "kelas" akan menjadi nama field di dalam Firestore.

```

documentReference
  .set(mhs)
  .whenComplete(() => print('${nimController.text} created'));

Navigator.pop(context);
}

```

- perintah eksekusi untuk mengirim data. Metode .set(mhs) akan mengirim data yang ada di dalam Map mhs ke lokasi yang sudah ditentukan oleh documentReference. Jika data dengan ID (NIM) tersebut belum ada, data baru akan dibuat. Jika sudah ada, datanya akan ditimpa/update.

```

    TextFormField(
      controller: nimController,
      decoration: InputDecoration(
        labelText: "NIM",
      ),
    ),
    TextFormField(
      controller: namaController,
      decoration: InputDecoration(labelText: "Nama"),
    ),
    TextFormField(
      controller: prodiController,
      decoration: InputDecoration(labelText: "Prodi"),
    ),
    TextFormField(
      controller: kelasController,
      decoration: InputDecoration(labelText: "Kelas")
    ),

```

- Widget antarmuka tampilan yang menampilkan kolom input teks. Properti controller pada setiap TextFormField (seperti controller: nimController, controller: namaController, controller: prodiController, dan controller: kelasController) merupakan bagian yang menghubungkan dengan variabel controller-nya masing-masing, sehingga data yang diinput pada setiap kolom (NIM, Nama, Prodi, dan Kelas) dapat dibaca dan dikelola

```

),
ElevatedButton(
  onPressed: () {
    addData();

    Navigator.pushAndRemoveUntil(
      context,
      MaterialPageRoute(builder: (context) => Home()),
      (Route<dynamic> route) => false,
    );
  },
  child: Text("Simpan"),
),

```

- onPressed: () { addData(); ... }, fungsi yang mengeksekusi ketika ElevatedButton ("Simpan") diklik. Di dalam fungsi ini, memanggil metode addData() untuk memulai proses penyimpanan data ke Firebase.
- "Navigator.pushAndRemoveUntil(context, ... (Route<dynamic> route) => false);" Perintah yang dieksekusi setelah memanggil addData(). Perintah ini akan mengarahkan ke halaman Home.

17. Menyiapkan halaman edit_data.dart

```

import 'package:flutter/material.dart';

import 'package:cloud_firestore/cloud_firestore.dart';

import 'home.dart';

class EditData extends StatefulWidget {
  final String? nim;

```

```

final String? nama;
final String? prodi;
final String? kelas;

EditData({this.nim, this.nama, this.prodi, this.kelas});

@override
_EditDataState createState() => _EditDataState();
}

class _EditDataState extends State<EditData> {
  TextEditingController nimController = new TextEditingController();
  TextEditingController namaController = new TextEditingController();
  TextEditingController prodiController = new TextEditingController();
  TextEditingController kelasController = new TextEditingController();

  void editData() {
    DocumentReference documentReference =
FirebaseFirestore.instance.collection('mahasiswa').doc(widget.nim);

    Map<String, dynamic> mhs = ({
      "nim": widget.nim,
      "nama": namaController.text,
      "prodi": prodiController.text,
      "kelas": kelasController.text
    });

    // update data to Firebase

    documentReference
      .update(mhs)
      .whenComplete(() => print('${widget.nim} updated'));
  }

  void deleteData() {
    DocumentReference documentReference =
FirebaseFirestore.instance.collection('mahasiswa').doc(widget.nim);

    // delete data from Firebase

    documentReference

```

```

        .delete()
        .whenComplete(() => print('${widget.nim} deleted'));
    }

    void konfirmasi() {
        AlertDialog alertDialog = AlertDialog(
            content: Text("Apakah anda yakin akan menghapus data
'$ {widget.nama}'"),
            actions: <Widget>[
                ElevatedButton(
                    style: ElevatedButton.styleFrom(
                        foregroundColor: Colors.red,
                    ),
                    child: Text(
                        "OK DELETE!",
                        style: TextStyle(color: Colors.black),
                    ),
                    onPressed: () {
                        deleteData();

                        Navigator.pushAndRemoveUntil(
                            context,
                            MaterialPageRoute(builder: (context) => Home()),
                            (Route<dynamic> route) => false,
                        );
                    },
                ),
                ElevatedButton(
                    style: ElevatedButton.styleFrom(
                        foregroundColor: Colors.green,
                    ),
                    child: Text("CANCEL", style: TextStyle(color: Colors.black)),
                    onPressed: () => Navigator.pop(context),
                ),
            ],
        );

        showDialog(
            context: context,
            builder: (BuildContext context) {
                return alertDialog;
            },
        );
    }

```

```

}

@override
void initState() {
  nimController = TextEditingController(text: widget.nim);
  namaController = TextEditingController(text: widget.nama);
  prodiController = TextEditingController(text: widget.prodi);
  kelasController = TextEditingController(text: widget.kelas);

  super.initState();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("EDIT DATA"),
    ),
    body: Padding(
      padding: EdgeInsets.all(10.0),
      child: ListView(
        children: <Widget>[
          Text(
            "Ubah Data Mahasiswa",
            style: TextStyle(
              color: Colors.red,
              fontStyle: FontStyle.italic,
              fontWeight: FontWeight.bold,
              fontSize: 25,
            ),
          ),
          SizedBox(
            height: 40,
          ),
          TextFormField(
            controller: nimController,
            decoration: InputDecoration(
              labelText: "NIM",
            ),
          ),
          TextFormField(
            controller: namaController,
            decoration: InputDecoration(labelText: "Nama"),

```



```

    ),
    TextFormField(
      controller: prodiController,
      decoration: InputDecoration(labelText: "Prodi"),
    ),
    TextFormField(
      controller: kelasController,
      decoration: InputDecoration(labelText: "Kelas"),
    ),
    SizedBox(
      height: 50,
    ),
    Row(
      mainAxisAlignment: MainAxisAlignment.spaceAround,
      children: [
        ElevatedButton(
          style: ElevatedButton.styleFrom(
            foregroundColor: Colors.orange,
          ),
          onPressed: () {
            editData();

            Navigator.pushAndRemoveUntil(
              context,
              MaterialPageRoute(builder: (context) => Home()),
              (Route<dynamic> route) => false,
            );
          },
          child: Text("Ubah"),
        ),
        ElevatedButton(
          style: ElevatedButton.styleFrom(
            foregroundColor: Colors.red,
          ),
          onPressed: () {
            konfirmasi();
          },
          child: Text("Hapus"),
        ),
      ],
    ),
  ],
),

```

```

    ),
  );
}
}

```

Penjelasan :

```

Windsurf: Refactor | Explain
class EditData extends StatefulWidget {
  final String? nim;
  final String? nama;
  final String? prodi;
  final String? kelas;

  EditData({this.nim, this.nama, this.prodi, this.kelas});

  @override
  _EditDataState createState() => _EditDataState();
}

```

- Mendeklarasikan kelas EditData sebagai StatefulWidget
- "final String? nim; ... EditData({this.nim, this.nama, this.prodi, this.kelas});" kode yang berfungsi menerima data mahasiswa (NIM, Nama, Prodi, Kelas) dari halaman sebelumnya. Saat memilih salah satu data di halaman daftar Home, kode inilah yang "menangkap" data tersebut agar bisa ditampilkan di halaman EditData.

```

TextEditingController nimController = new TextEditingController();
TextEditingController namaController = new TextEditingController();
TextEditingController prodiController = new TextEditingController();
TextEditingController kelasController = new TextEditingController();

```

- Kode ini memasukkan data lama (seperti widget.nama) ke dalam kotak input "Nama". Hal yang sama berlaku untuk NIM, Prodi, dan Kelas. Jadi, saat halaman "Edit" terbuka, semua kolom sudah terisi.

```

Windsurf: Refactor | Explain | Generate Function Comment | X
void editData() {
    DocumentReference documentReference =
        FirebaseFirestore.instance.collection('mahasiswa').doc(widget.nim);

    Map<String, dynamic> mhs = ({
        "nim": widget.nim,
        "nama": namaController.text,
        "prodi": prodiController.text,
        "kelas": kelasController.text
    });

    // update data to Firebase

    documentReference
        .update(mhs)
        .whenComplete(() => print('${widget.nim} updated'));
}

```

- "void editData()" Ini adalah metode / fungsi yang sebagai logika utama untuk memperbarui (meng-update) data di database Firebase.
- "DocumentReference documentReference = FirebaseFirestore.instance.collection('mahasiswa').doc(widget.nim);" Kode ini mencari data yang ingin diubah. Ia masuk ke collection mahasiswa dan mencari data yang NIM-nya (widget.nim) sesuai dengan yang sedang diedit.
- "documentReference.update(mhs)", untuk menyimpan perubahan ke Firebase

```

Windsurf: Refactor | Explain | Generate Function Comment | X
void deleteData() {
    DocumentReference documentReference =
        FirebaseFirestore.instance.collection('mahasiswa').doc(widget.nim);

    // delete data from Firebase

    documentReference
        .delete()
        .whenComplete(() => print('${widget.nim} deleted'));
}

```

- "void deleteData()" Ini adalah fungsi yang berisi perintah untuk menghapus data dari database.
- "documentReference.delete()", Ini adalah perintah untuk menghapus permanen dokumen yang alamatnya sudah ditemukan sebelumnya (berdasarkan widget.nim).

```

Windsurf: Refactor | Explain | Generate Function Comment | ×
void konfirmasi() {
  AlertDialog alertDialog = AlertDialog(
    content: Text("Apakah anda yakin akan menghapus data '${widget.nama}'),
    actions: <Widget>[
      ElevatedButton(
        style: ElevatedButton.styleFrom(
          foregroundColor: Colors.red,
        ),
        child: Text(
          "OK DELETE!",
          style: TextStyle(color: Colors.black),
        ),
        onPressed: () {
          deleteData();

          Navigator.pushAndRemoveUntil(
            context,
            MaterialPageRoute(builder: (context) => Home()),
            (Route<dynamic> route) => false,
          );
        },
      ),
      ElevatedButton(
        style: ElevatedButton.styleFrom(
          foregroundColor: Colors.green,
        ),
        child: Text("CANCEL", style: TextStyle(color: Colors.black)),
        onPressed: () => Navigator.pop(context),
      ),
    ],
  );

  showDialog(
    context: context,
    builder: (BuildContext context) {
      return alertDialog;
    },
  );
}

```

- "void konfirmasi()", Ini adalah fungsi yang berisi untuk memunculkan modal/dialog peringatan (pop-up) "Apakah Anda yakin?"
- "AlertDialog alertDialog = AlertDialog(...)", tombol pilihan ("OK DELETE!" dan "CANCEL").
- "onPressed: () { deleteData(); ... }", kode yang dijalankan kalau tombol "OK DELETE!" ditekan. Fungsinya memanggil deleteData() (untuk menghapus data) lalu pindah kembali ke halaman Home.
- "showDialog(context: context, ...)", kode tersebut berfungsi menampilkan modal dialog.

```

TextFormField(
  controller: nimController,
  decoration: InputDecoration(
    labelText: "NIM",
  ),
),
TextFormField(
  controller: namaController,
  decoration: InputDecoration(labelText: "Nama"),
),
TextFormField(
  controller: prodiController,
  decoration: InputDecoration(labelText: "Prodi"),
),
TextFormField(
  controller: kelasController,
  decoration: InputDecoration(labelText: "Kelas"),
),

```

- Widget antarmuka tampilan yang menampilkan kolom input teks. Properti controller pada setiap TextFormField (seperti controller: nimController, controller: namaController, controller: prodiController, dan controller: kelasController)

```

Row(
  mainAxisAlignment: MainAxisAlignment.spaceAround,
  children: [
    ElevatedButton(
      style: ElevatedButton.styleFrom(
        foregroundColor: Colors.orange,
      ),
      onPressed: () {
        editData();

        Navigator.pushAndRemoveUntil(
          context,
          MaterialPageRoute(builder: (context) => Home()),
          (Route<dynamic> route) => false,
        );
      },
      child: Text("Ubah"),
    ),
    ElevatedButton(
      style: ElevatedButton.styleFrom(
        foregroundColor: Colors.red,
      ),
      onPressed: () {
        konfirmasi();
      },
      child: Text("Hapus"),
    ),
  ],
),

```

- "ElevatedButton(onPressed: () { editData(); ... })", Ini adalah tombol "Ubah". Saat ditekan, ia memanggil fungsi editData() (untuk menyimpan perubahan) lalu kembali ke halaman Home.
- "ElevatedButton(onPressed: () { konfirmasi(); })", Ini adalah tombol "Hapus". Saat ditekan, ia tidak langsung menghapus, tapi memanggil fungsi konfirmasi() untuk memunculkan kotak peringatan dulu.

18. Menyiapkan halaman home:

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

import 'package:flutter/material.dart';

import 'tambah_data.dart';

import 'edit_data.dart';

class Home extends StatefulWidget {

  @override

  HomeState createState() => HomeState();

}

class HomeState extends State<Home> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Database Firebase'),

      ),

      floatingActionButton: FloatingActionButton(

        onPressed: () {

          Navigator.push(

            context,

            MaterialPageRoute(

              builder: (context) => TambahData(),

            ),

          );

        },

        child: Icon(Icons.add),

      ),

    ),
```

```

body: StreamBuilder(
  stream:
FirebaseFirestore.instance.collection('mahasiswa').snapshots(),
  builder: (context, snapshot) {
    if (snapshot.hasData) {
      return Container(
        padding: EdgeInsets.fromLTRB(5, 10, 5, 5),
        child: Card(
          child: ListView.builder(
            shrinkWrap: true,
            itemCount: snapshot.data!.docs.length,
            itemBuilder: (context, index) {
              DocumentSnapshot documentSnapshot =
                snapshot.data!.docs[index];
              return Column(
                children: [
                  GestureDetector(
                    child: ListTile(
                      leading: Icon(Icons.person),
                      title: Text(documentSnapshot["nim"]),
                      // subtitle:
Text(documentSnapshot["nama"]),
                      subtitle: Column(
                        crossAxisAlignment: CrossAxisAlignment
                          .start, // Agar teks rata kiri
                        children: [
                          Text(documentSnapshot["nama"]),
                          Text(
                            'Prodi:
${documentSnapshot["prodi"]}',

```

```

                style: TextStyle(fontSize: 12),
            ),
            Text(
                'Kelas:
${documentSnapshot["kelas"]}',
                style: TextStyle(fontSize: 12),
            ),
        ],
    ),
    trailing:
Icon(Icons.navigate_next_rounded),
),
onTap: () => Navigator.push(
    context,
    MaterialPageRoute(
        builder: (context) => EditData(
            nim: documentSnapshot["nim"],
            nama: documentSnapshot["nama"],
            prodi: documentSnapshot["prodi"],
            kelas: documentSnapshot["kelas"],
        ),
    ),
),
),
Divider(
    color: Colors.black,
    indent: 10,
    endIndent: 10,
)
],

```



```
        );  
    },  
    ),  
    ),  
);  
} else {  
    return Center(  
        child: CircularProgressIndicator(  
            backgroundColor: Colors.black,  
        ),  
    );  
}  
},  
),  
);  
}  
}
```

Penjelasan singkat :

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'tambah_data.dart';
import 'edit_data.dart';
```

- Berfungsi untuk menghubungkan halaman Home dengan dua halaman lain, yaitu TambahData dan EditData. Ini diperlukan agar aplikasi bisa berpindah halaman (navigasi) ke halaman tambah atau edit.

```
floatingActionButton: FloatingActionButton(
  onPressed: () {
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => TambahData(),
      ),
    );
  },
  child: Icon(Icons.add),
),
```

- widget yang membuat tombol dengan ikon tambah (Icons.add).

```
body: StreamBuilder(
  stream: FirebaseFirestore.instance.collection('mahasiswa').snapshots(),
  builder: (context, snapshot) {
```

- StreamBuilder dan "stream: FirebaseFirestore.instance.collection('mahasiswa').snapshots()," berfungsi untuk mengelola data dari Firebase secara real-time. Ia akan otomatis memperbarui tampilan (ListView) setiap kali ada perubahan data (tambah, edit, atau hapus) di database Firebase pada collection 'mahasiswa'.

```
if (snapshot.hasData) {  
    return Container(  
        child: Text(  
            snapshot.data.toString(),  
            style: TextStyle(  
                color: Colors.white,  
                fontSize: 24,  
            ),  
        ),  
    );  
}
```

- Ini adalah pengecekan kondisi di dalam StreamBuilder. Kode ini memeriksa apakah snapshot (kiriman data dari Firebase) sudah berhasil diterima dan berisi data. Jika YA, maka tampilkan datanya.

```
} else {  
    return Center(  
        child: CircularProgressIndicator(  
            backgroundColor: Colors.black,  
        ),  
    );  
}
```

- Jika snapshot.hasData adalah TIDAK (artinya data belum siap atau sedang dimuat), maka tampilkan ikon indikator loading di tengah layar.
-

```

return Container(
  padding: EdgeInsets.fromLTRB(5, 10, 5, 5),
  child: Card(
    child: ListView.builder(
      shrinkWrap: true,
      itemCount: snapshot.data!.docs.length,
      itemBuilder: (context, index) {
        DocumentSnapshot documentSnapshot =
          snapshot.data!.docs[index];
        return Column(

```

```

return Column(
  children: [
    GestureDetector(
      child: ListTile(
        leading: Icon(Icons.person),
        title: Text(documentSnapshot["nim"]),
        // subtitle: Text(documentSnapshot["nama"]),
        subtitle: Column(
          crossAxisAlignment: CrossAxisAlignment
            .start, // Agar teks rata kiri
          children: [
            Text(documentSnapshot["nama"]),
            Text(
              'Prodi: ${documentSnapshot["prodi"]}',
              style: TextStyle(fontSize: 12),
            ),
            Text(
              'Kelas: ${documentSnapshot["kelas"]}',
              style: TextStyle(fontSize: 12),
            ),
          ],
        ),
        trailing: Icon(Icons.navigate_next_rounded),
      ),
      onTap: () => Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => EditData(
            nim: documentSnapshot["nim"],
            nama: documentSnapshot["nama"],
            prodi: documentSnapshot["prodi"],
            kelas: documentSnapshot["kelas"],
          ),
        ),
      ),
    ),
  ],
).

```

- "ListView.builder(...)", Ini adalah widget yang membangun tampilan daftar (list) secara otomatis. Ia akan membuat satu baris item untuk setiap data mahasiswa yang diterima dari snapshot.
- "itemCount: snapshot.data!.docs.length", Kode ini memberitahu ListView.builder jumlah total item yang harus dibuat, berdasarkan berapa banyak dokumen (data) yang ada di dalam snapshot.

- "DocumentSnapshot documentSnapshot = snapshot.data!.docs[index];", Di dalam ListView.builder, kode ini berfungsi untuk mengambil satu per satu data mahasiswa dari daftar dan menyimpannya ke dalam variabel documentSnapshot.
- "child: ListTile(...)", widget yang mengatur tampilan untuk satu baris data mahasiswa, lengkap dengan ikon (leading), judul (title: Text"(documentSnapshot["nim"]))", dan subjudul (yang berisi Nama, Prodi, dan Kelas).
- "onTap: () => Navigator.push(context, MaterialPageRoute(builder: (context) => EditData(...)),);", Ini adalah aksi yang terjadi saat salah satu item di daftar ditekan. Fungsinya adalah untuk membuka halaman EditData.

19. Menyiapkan halaman main.dart

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

import 'package:firebase_core/firebase_core.dart';

import 'home.dart';

void main() async {

  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp();

  runApp(MyApp());
}

class MyApp extends StatelessWidget {

  // This widget is the root of your application.

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      home: Home(),

      debugShowCheckedModeBanner: false,

    );

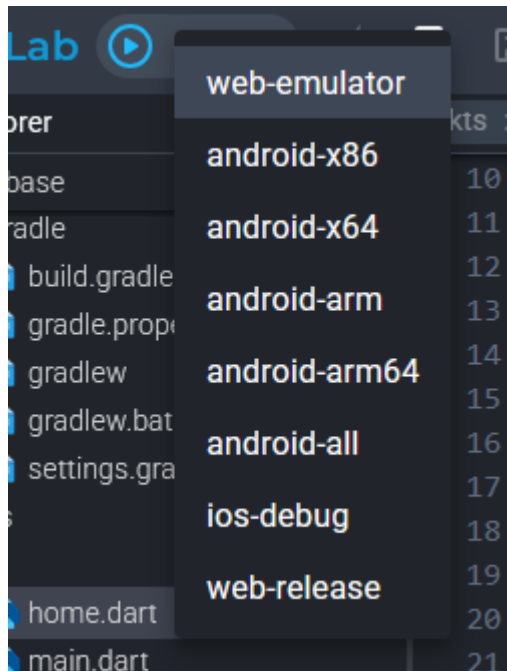
  }

}
```

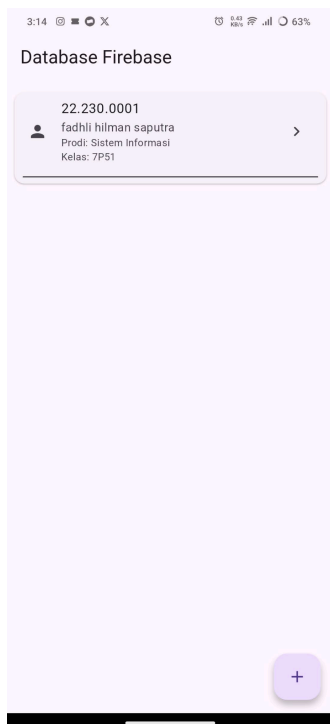
Penjelasan singkat :

- import 'package:flutter/material.dart'; Mengimpor semua widget dan tema dasar Material Design Flutter
- import 'package:firebase_core/firebase_core.dart'; Mengimpor package Firebase Core yang diperlukan untuk menginisialisasi (menghubungkan) aplikasi dengan proyek Firebase Anda.
- import 'home.dart'; Mengimpor file home.dart yang berisi kelas Home, yaitu halaman utama (tampilan daftar data) yang akan ditampilkan pertama kali.
- Mengatur widget Home() (yang berisi tampilan data Firestore) sebagai halaman awal aplikasi.

20. Jalankan proyek :



- Pilih android-arm64
- Lalu download pada hp
- Buka aplikasi hasil download tersebut
- Test Tampilan:



- Test Tambah Data

3:14 0.45 KB/s 63%

← ADD DATA

Input Data Mahasiswa

NIM
22.230.1111

Nama
Inputtt nama

Prodi
input prodii

Kelas
7M51

Simpan

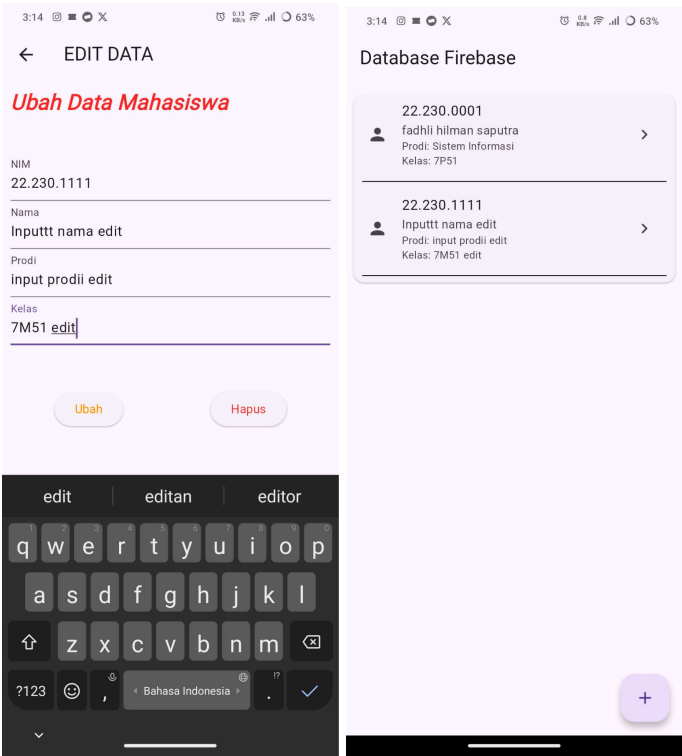
3:14 4.11 KB/s 63%

Database Firebase

22.230.0001	fadhli hilman saputra	>
Prodi: Sistem Informasi Kelas: 7P51		
22.230.1111	Inputtt nama	>
Prodi: input prodii Kelas: 7M51		

+

- Test Edit Data



- Test hapus data

