

# Laporan UTS

## Mobile Application & SQLite

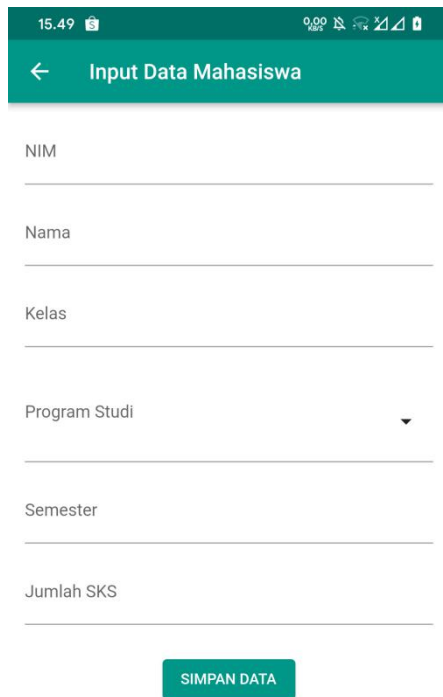


Dibuat Oleh :

Nama	: Muhamad Faisal Halim
NIM	: 19.240.0163
Kelas	: 5P43
Mata Kuliah	: Mobile Application

Sekolah Tinggi Manajemen Informatika Dan Komputer  
Widya Pratama Pekalongan  
Jl. Patriot No. 25 Pekalongan

## Hasil Akhir



15.49

← Input Data Mahasiswa

NIM

Nama

Kelas

Program Studi

Semester

Jumlah SKS

SIMPAN DATA

membuat tampilan input seperti pada gambar disamping dan dengan beberapa ketentuan berikut.

1. terhubung ke SQLite
2. input nim, sks, dan semester keyboard hanya menampilkan nomor
3. menampilkan daftar mahasiswa yang tersimpan didatabase
4. bisa menampilkan detail dan CRUD

## Langkah Pengerjaan

Pertama membuat project flutter, dan buka pada texteditor VSCode. lalu edit file pubspec.yaml dan tambahkan kode berikut untuk menginstall dan menambahkan library pada flutter.

```
dependencies:
  ...

  sqflite: ^2.0.0+4
  path_provider: ^2.0.5
  flutter_masked_text2: ^0.9.1
  dropdown_search: ^2.0.1

  ...
```

selanjutnya, buat folder baru didalam folder lib, dengan nama config. lalu membuat file config.dart file ini berisi semua config yang dibutuhkan pada proyek ini.

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

class MyConfig {

  MyConfig._privateConstructor();
  static final MyConfig instance = MyConfig._privateConstructor();

  // Config App
  String get app_name => "Data Mahasiswa";
  String get insert_page => "Input Data Mahasiswa";
  String get edit_page => "Edit Data Mahasiswa";

  // Database
  String get db_name => "Mahasiswa.db";
  String get tb_name => "Mahasiswa";

  // Color
  MaterialColor appColor = Colors.teal;

  Color bgColor = Colors.white;
  Color iconColor = Colors.teal.shade300;
  Color btnDelColor = Colors.red.shade300;
  Color listBgColor = Colors.white70;

}

```

membuat folder dan file model, untuk kode seperti dibawah ini. model akan membantu kita mapping data saat melakukan insert ke dalam database, get data dari database, dll.

```
class MahasiswaModel {
  final int? id;
  final String nim;
  final String nama;
  final String kelas;
  final String progdi;
  final String smt;
  final String sks;

  MahasiswaModel({
    this.id, required this.nim, required this.nama, required this.kelas,
    required this.progdi, required this.smt, required this.sks
  });

  factory MahasiswaModel.fromMap(Map<String, dynamic> json) => MahasiswaModel(
    id: json['id'],
    nim: json['nim'],
    nama: json['nama'],
    kelas: json['kelas'],
    progdi: json['progdi'],
    smt: json['smt'],
    sks: json['sks']
  );

  Map<String, dynamic> toMap() {
    return {
      'nim': nim,
      'nama': nama,
      'kelas': kelas,
      'progdi': progdi,
      'smt': smt,
      'sks': sks
    };
  }
}

```

sebelum melangkah ke file-file lain kita saya membuat file database helper, file ini akan melakukan eksekusi setiap tindakan yang diambil jika berhubungan dengan database. misal input, hapus, dan update.

kode dari file database helper terlihat seperti berikut.

```

import 'package:sqflite/sqflite.dart';
import 'package:faisal_0163/config/config.dart';

import 'package:faisal_0163/models/mahasiswa_model.dart';

class DatabaseHelper {
  DatabaseHelper._privateConstructor();
  static final DatabaseHelper instance = DatabaseHelper._privateConstructor();

  static Database? _database;
  Future<Database> get database async => _database = await _initDatabase();

  Future<Database> _initDatabase() async {
    return await openDatabase(MyConfig.instance.tb_name, version: 1, onCreate: _onCreate);
  }

  Future _onCreate(Database db, int version) async{
    await db.execute('''
      CREATE TABLE '''+ MyConfig.instance.tb_name +'''(
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        nim TEXT,
        nama TEXT,
        kelas TEXT,
        progdi TEXT,
        smt TEXT,
        sks TEXT
      )
    ''');
  }

  Future<List<MahasiswaModel>> getAllItems() async {
    Database db = await instance.database;
    var tv = await db.query(MyConfig.instance.tb_name);

    List<MahasiswaModel> tvlist = tv.isNotEmpty
      ? tv.map((e) => MahasiswaModel.fromMap(e)).toList()
      : [];

    return tvlist;
  }

  Future<int> insert(MahasiswaModel mhs) async {
    Database db = await instance.database;
    return db.insert(MyConfig.instance.tb_name, mhs.toMap());
  }

  Future<int> remove(int id) async {
    Database db = await instance.database;
    return await db.delete(MyConfig.instance.tb_name, where: 'id = ?', whereArgs: [id]);
  }

  Future<int> update(MahasiswaModel mhs) async {
    Database db = await instance.database;
    return await db.update(MyConfig.instance.tb_name, mhs.toMap(),
      where: "id=?", whereArgs: [mhs.id]
    );
  }
}

```

selanjutnya edit file main.dart agar terlihat seperti berikut, dalam file main ini akan menampilkan semua konten dari file home.dart yang akan kita buat.

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:faisal_0163/config/config.dart';
import 'package:faisal_0163/pages/home.dart';

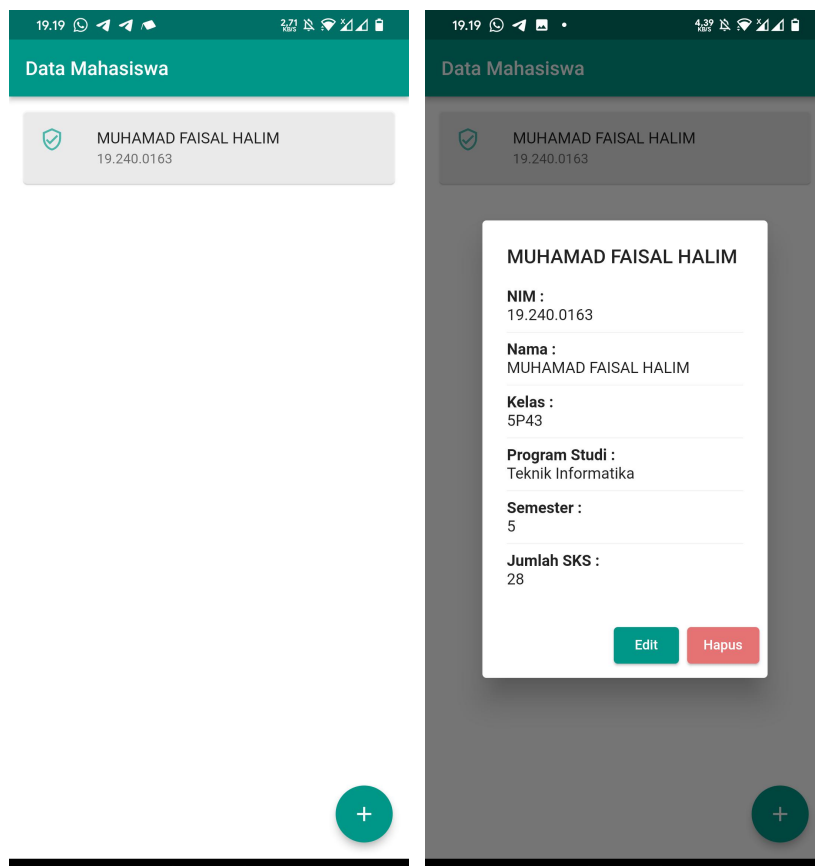
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

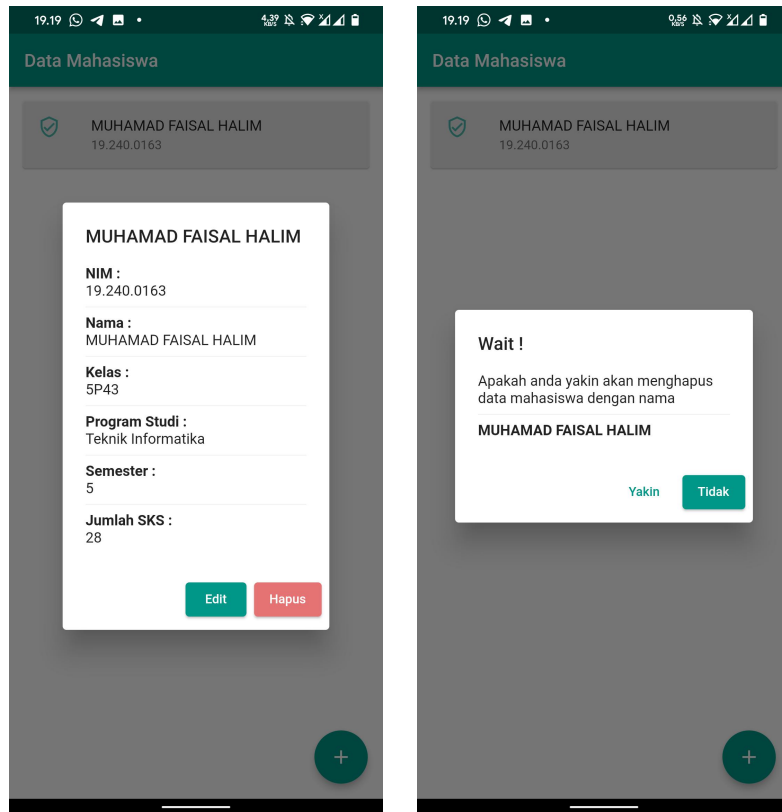
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: MyConfig.instance.app_name,
      home: HomePages(title: MyConfig.instance.app_name),
      theme: ThemeData(
        primarySwatch: MyConfig.instance.appColor,
        scaffoldBackgroundColor: MyConfig.instance.bgColor
      ),
      debugShowCheckedModeBanner: false
    );
  }
}
```

dan membuat folder pages untuk tampilan user-nya. didalam folder pages pertama kita buat file home.dart, file ini akan menjadi tampilan utama dan tempat menampung semua list mahasiswa yang diambil dari database.

untuk file home.dart ini bisa diatur sedemikian rupa agar nampak seperti berikut. dengan menggunakan FutureBuilder dan ListView, dan ketika data di pencet akan menampilkan popup yang dibuat dengan AlertDialog, popup ini berisi detail mahasiswa, dan beberapa tombol.



tolbol edit akan membawa kita kedalam halaman edit dan tombol delete akan menampilkan popup konfirmasi hapus, jika kita pilih ya data akan terhapus dan jika pilih tidak data tidak akan terhapus.



selanjutnya membuat file input.dart, pada file ini kita bisa melakukan insert data kedalam database. didalam file ini kita diharuskan mengimportkan beberapa file dan library yang sudah kita tambahkan. dan untuk tataletak bisa diantur sedemikian rupa.

sebagai pembeda proyek saya dengan orang lain dikelas, pada bagian input nim saya memmbuatnya agar nim bisa memformat otomatis (menambahkan titik otomatis), dengan bantuan libraru flutter\_masked\_text2. untuk controllernya akan terlihat seperti berikut.

```
...
final nim = MaskedTextController(mask: '00.000.0000');
final nama = TextEditingController();
final kelas = TextEditingController();
final progdi = TextEditingController();
final smt = TextEditingController();
final sks = TextEditingController();
...
```

dan pada agian textfieldnya seperti berikut.

```
...
TextField(
  controller: nim,
  keyboardType: TextInputType.number,
  decoration: InputDecoration(
    labelText: "NIM",
    errorText: nim.validate ? "NIM Harus Diisi" : null,
  ),
),
...
```

dalam ketentuan kita diharuskan membuat dropdown, untuk itu saya menggunakan library tambahan yaitu dropdown\_search dan koden seperti berikut.

```
...
DropdownSearch<String>(  
  mode: Mode.MENU,  
  showSelectedItem: true,  
  items: const [  
    "Pilih Program Studi",  
    "Teknik Informatika",  
    "Sistem Informasi",  
    "Komputerisasi Akuntansi",  
    "Manajemen Informatika"  
  ],  
  dropdownSearchDecoration: InputDecoration(  
    labelText: "Program Studi",  
    errorText: progdi.validate ? "Program Studi Harus Dipilih" : null,  
  ),  
  popupItemDisabled: (String s) => s.contains('Pilih Program Studi'),  
  onChanged: (data) {  
    progdi.text = data!  
  },  
),  
...
```

dan untuk keyboard hanya menampilkan nomor, bisa menambahkan kode dibawah ini pada setiap TextField yang ada.

```
...  
keyboardType: TextInputType.number,  
...
```

dan untuk pembeda lain, saya menambahkan validasi pada saat input data, ketika textfield kosong, maka data tidak dapat ditambahkan, dan akan menampilkan error seperti berikut.

Untuk membuatnya, pertama kita tambahkan kode berikut.

```
...  
bool nimvalidate = false;  
bool namavalidate = false;  
bool kelasvalidate = false;  
bool progdivalidate = false;  
bool smtvalidate = false;  
bool sksvalidate = false;  
  
@override  
void dispose() {  
  nim.dispose();  
  nama.dispose();  
  kelas.dispose();  
  progdi.dispose();  
  smt.dispose();  
  sks.dispose();  
  
  super.dispose();  
}  
...
```



selanjutnya tambahkan kode ini pada setiap input decoration disetiap textfieldnya.

```
...
TextField(
  ...
  decoration: InputDecoration(
    ...
    hintText: "NIM Harus Diisi" : null,
  ),
),
...
```

dan melakukan beberapa pengkondisian saat tombol simpan dipencet. dan kodenya terlihat seperti dibawah ini.

```
...
ElevatedButton(
  child: const Text("SIMPAN DATA"),
  onPressed: () async {
    setState(() {
      nim.text.isEmpty ? nimvalidate = true : nimvalidate = false;
      nama.text.isEmpty ? namavalidate = true : namavalidate = false;
      kelas.text.isEmpty ? kelasvalidate = true : kelasvalidate = false;
      progdi.text.isEmpty ? progdivalidate = true : progdivalidate = false;
      smt.text.isEmpty ? smtvalidate = true : smtvalidate = false;
      sks.text.isEmpty ? sksvalidate = true : sksvalidate = false;
    });

    if ((nimvalidate == false) &&
        (namavalidate == false) &&
        (kelasvalidate == false) &&
        (progdivalidate == false) &&
        (smtvalidate == false) &&
        (sksvalidate == false)) {

      await DatabaseHelper.instance.insert(MahasiswaModel(
        nim: nim.text,
        nama: nama.text,
        kelas: kelas.text,
        progdi: progdi.text,
        smt: smt.text,
        sks: sks.text,
      ));

      Navigator.pushAndRemoveUntil(
        context, MaterialPageRoute(builder: (context) => MyApp()),
        (Route-dynamic> route) => false,
      );
    } else {
      AlertDialog(
        title: const Text("Woops !"),
        content: const Text("Sepertinya ada kesalahan"),
        actions: [
          TextButton(
            child: const Text("OK"),
            onPressed: () => Navigator.pop(context),
          ),
        ],
      );
    }
  },
),
...
```

dari kode diatas bisa diambil alur sebagai berikut, ketika tombol simpan di pencet controller akan di cek apakah datanya kosong atau tidak, jika kosong maka pesan errpr akan tampil dengan bantuan set state, dan jika tidak kosong maka akan langsung diinputkan kedalam database dan kita langsung diarahkan ke home.dart.

selanjutnya dan terakhir adalah file edit.dart, kita bisa menduplikat file input.dart dan menambahkan beberapa perubahan.



yang pertama dengan tentunya merubah nama class dan nama filenya, setelah itu dengan menambahkan beberapa atribut didalam class EditMahasiswa.

```
...
class EditMahasiswa extends StatefulWidget {
  final int? id;
  final String? nim;
  final String? nama;
  final String? kelas;
  final String? progdi;
  final String? smt;
  final String? sks;

  const EditMahasiswa({
    Key? key, this.id, this.nim, this.nama, this.kelas,
    this.progdi, this.smt, this.sks }) : super(key: key);

  @override
  _EditMahasiswaState createState() => _EditMahasiswaState();
}
...
```

dan berikut kode saat mengirimkan data dari home.dart ke edit.dart

```
// kode ini ada didalam file Home.dart
// dan dieksekusi saat kita menekan tombol edit
...
ElevatedButton(
  onPressed: () {
    Navigator.push(
      context, MaterialPageRoute(
        builder: (context) => (
          EditMahasiswa(
            id: mhs.id,
            nim: mhs.nim,
            nama: mhs.nama,
            kelas: mhs.kelas,
            progdi: mhs.progdi,
            smt: mhs.smt,
            sks: mhs.sks,
          )
        )
      );
    },
  child: const Text("Edit")
),
...
```

selanjutnya dengan menambahkan 1 controller baru, yaitu controller untuk id. dan buat sebuah init state yang mengisi data yang dikirim dari home.dart kedalam setiap controller yang kita buat.

```
...
final id = TextEditingController();
...
...
@override
void initState() {
  super.initState();
  id.text = widget.id!.toString();
  nim.text = widget.nim!;
  nama.text = widget.nama!;
  kelas.text = widget.kelas!;
  progdi.text = widget.progdi!;
  smt.text = widget.smt!;
  sks.text = widget.sks!;
}
...
```

dan merubah dibagian saat tombol simpan dipencet, dari yang sebelumnya insert, diubah ke update.

```

...
await DatabaseHelper.instance.update(MahasiswaModel(
    id: int.parse(id.text),
    nim: nim.text,
    nama: nama.text,
    kelas: kelas.text,
    progdi: progdi.text,
    smt: smt.text,
    sks: sks.text
));
...

```

jika dirasa kode sudah benar, sesuai dan tidak ada error, bisa langsung diuji coba.

untuk melakukannya kita memerlukan android, karena jika di tes dengan web broser saya mencoba dan terjadi error saat menyimpan datanya.

sekian laporan UTS ini. mohon maaf jika salah-salah kata atau yang menyinggung.

## Show Off

