PRAKTIKUM PEMROGRAMAN PERANGKAT BERGERAK

MODUL X

DATA STORAGE BAGIAN I

Disusun Oleh:

Aorinka Anendya Chazanah / 2211104013

S1 SE-06-01

Asisten Praktikum:

MuhammadFazaZulian Gesit Al Barru

Aisyah Hasna Aulia

Dosen Pengampu:

Yudha Islami Sulistya, S.Kom., M.Cs

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

GUIDED

1. **Pengenalan SQLite**

   SQLite adalah sebuah database relasional yang digunakan untuk penyimpanan data secara offline pada aplikasi mobile (disimpan dalam local storage, khususnya pada cache memory aplikasi). SQLite mendukung operasi CRUD (create, read, update, dan delete), yang merupakan empat operasi dasar dalam pengelolaan data. Struktur database SQLite mirip dengan SQL pada umumnya, dengan variabel dan tipe data yang serupa dengan yang ada di SQL.

2. **SQL Helper Dasar**

   Dalam Flutter, SQL Helper biasanya merujuk pada penggunaan paket seperti sqflite untuk mengelola database SQLite. SQL Helper adalah kelas yang berfungsi untuk membuat berbagai metode yang berkaitan dengan manipulasi data. Sqflite sendiri adalah plugin Flutter yang memungkinkan pengguna untuk melakukan operasi CRUD (Create, Read, Update, Delete) pada database SQLite. Berikut adalah langkah-langkah dasar untuk menggunakan sqflite sebagai SQL Helper di Flutter:

   1. Tambahkan plugin sqflite dan path ke dalam file pubspec.yaml. Plugin dapat diunduh di sini.
   2. Buat kelas baru bernama DatabaseHelper untuk mengelola database, serta impor paket sqflite dan path dalam file db_helper.dart.

```dart
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';

// kelas DatabaseHelper untuk mengelola database
class DatabaseHelper {

static final DatabaseHelper _instance =
DatabaseHelper._internal();
static Database? _database;
```

3. Buat factory constructor untuk mengembalikan instance singleton dan private singleton.

```
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';

// kelas DatabaseHelper untuk mengelola database
class DatabaseHelper {
  static final DatabaseHelper _instance =
DatabaseHelper._internal();
  static Database? _database;

//Factory constructor untuk mengembalikan instance
singleton
  factory DatabaseHelper() {
    return _instance;
  }

 // Private constructor
  DatabaseHelper._internal();
```

4. Buat Getter untuk database.

```
Future<Database> get database async {
    if (_database != null) return _database!;
    {
      _database = await _initDatabase();
      return _database!;
    }
  }
```

5. Inisialisasi database dengan nama database yang kita mau.

```
Future<Database> _initDatabase() async {
// mendapatkan path untuk database
    String path = join(await getDatabasesPath(),
'my_prakdatabase.db');
// membuka database
    return await openDatabase(
      path,
      version: 1,
      onCreate: _onCreate,
    );
  }
```

6. Kemudian buat tabel untuk database-nya dengan record atau value id, title, dan description.

```
Future<void> _onCreate(Database db, int version) async
{
    await db.execute('''
CREATE TABLE my_table(
id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
title TEXT,
description TEXT,
createdAt TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP)
''');
  }
```

7. Buat metode untuk memasukkan data ke dalam tabel.

```
Future<int> insert(Map<String, dynamic> row) async {
    Database db = await database;
    return await db.insert('my_table', row);
  }
```

8. Lalu, metode untuk mengambil semua data dari tabel

```
Future<List<Map<String, dynamic>>> queryAllRows()
async {
    Database db = await database;
    return await db.query('my_table');
  }
```

9. Buat metode untuk memperbarui data dalam tabel.

```
Future<int> update(Map<String, dynamic> row) async {
    Database db = await database;
    int id = row['id'];
    return await db.update('my_table', row, where: 'id
= ?', whereArgs: [id]);
  }
```

10. Diakhiri dengan metode untuk menghapus data dari tabel.

```
Future<int> delete(int id) async {
    Database db = await database;
```

```
    return await db.delete('my_table', where: 'id =
?', whereArgs: [id]);
  }
}
```

a. Read

Pada paket sqflite, kita dapat menggunakan query() untuk mengakses data yang ada dalam database. Dengan sqflite di Flutter, kita bisa membuat query menggunakan berbagai perintah, seperti where, groupBy, orderBy, dan having. Selain itu, kita juga dapat membaca satu atau beberapa data sekaligus. Berikut adalah contoh kode untuk operasi pembacaan (read) pada sqflite.

Membaca semua data

```
  Future<List<Map<String, dynamic>>> queryAllRows()
async {
    Database db = await database;
    return await db.query('my_table');
  }
```

Membaca satu data melalui id

```
 Future<List<Map<String, dynamic>>> getItem(int id)
async {
Database db = await database;
Return await db.query('my_table', row, where: "id =
?",
whereArgs: [id], limit: 1);
}
```

Source Code Praktikum

File main.dart

```dart
import 'package:flutter/material.dart';
import 'package:pertemuan_10/view/my_db_view.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor:
Colors.deepPurple),
        useMaterial3: true,
      ),
      home: MyDatabaseView(),
    );
  }
}
```

File db_helper.dart

```dart
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';

// kelas DatabaseHelper untuk mengelola database
class DatabaseHelper {
  static final DatabaseHelper _instance =
DatabaseHelper._internal();
  static Database? _database;

  //Factory constructor untuk mengembalikan instance singleton
  factory DatabaseHelper() {
    return _instance;
  }

  // Private constructor
  DatabaseHelper._internal();
```

```
  Future<Database> get database async {
    if (_database != null) return _database!;
    {
      _database = await _initDatabase();
      return _database!;
    }
  }

  Future<Database> _initDatabase() async {
// mendapatkan path untuk database
    String path = join(await getDatabasesPath(),
'my_prakdatabase.db');
// membuka database
    return await openDatabase(
      path,
      version: 1,
      onCreate: _onCreate,
    );
  }

  Future<void> _onCreate(Database db, int version) async {
    await db.execute('''
CREATE TABLE my_table(
id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
title TEXT,
description TEXT,
createdAt TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP)
''');
  }

  Future<int> insert(Map<String, dynamic> row) async {
    Database db = await database;
    return await db.insert('my_table', row);
  }

  Future<List<Map<String, dynamic>>> queryAllRows() async {
    Database db = await database;
    return await db.query('my_table');
  }

  Future<int> update(Map<String, dynamic> row) async {
    Database db = await database;
    int id = row['id'];
    return await db.update('my_table', row, where: 'id = ?',
whereArgs: [id]);
  }
```

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

File my_db_view.dart

```
import 'package:flutter/material.dart';
import 'package:pertemuan_10/helper/db_helper.dart';

class MyDatabaseView extends StatefulWidget {
  const MyDatabaseView({super.key});

  @override
  State<MyDatabaseView> createState() => _MyDatabaseViewState();
}

class _MyDatabaseViewState extends State<MyDatabaseView> {
  final DatabaseHelper dbHelper = DatabaseHelper();
  List<Map<String, dynamic>> _dbdata = [];
  final TextEditingController _titleController =
TextEditingController();
  final TextEditingController _descriptionController =
TextEditingController();

  @override
  void initState() {
    _refreshData();
    super.initState();
  }

  @override
  void dispose() {
    _titleController.dispose();
    _descriptionController.dispose();
    super.dispose();
  }

  void _refreshData() async {
    final data = await dbHelper.queryAllRows();
    setState(() {
      _dbdata = data;
    });
```

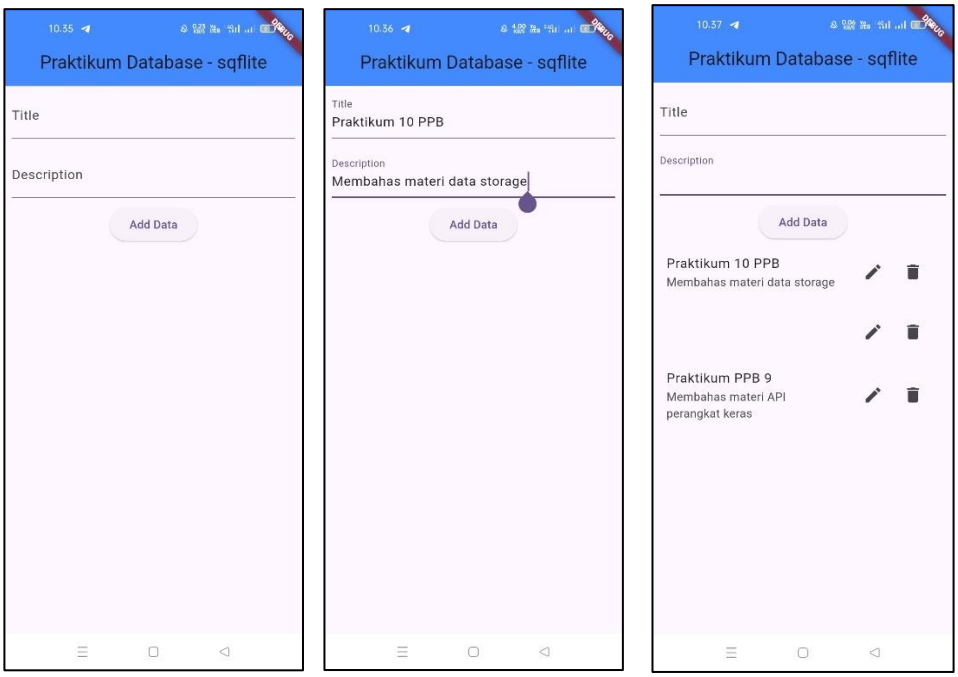
```
  }

  void _addData() async {
    await dbHelper.insert({
      'title': _titleController.text,
      'description': _descriptionController.text,
    });
    _titleController.clear();
    _descriptionController.clear();
    _refreshData();
  }

  void _updateData(int id) async {
    await dbHelper.update({
      'id': id,
      'title': _titleController.text,
      'description': _descriptionController.text
    });
    _titleController.clear();
    _descriptionController.clear();
    _refreshData();
  }

  void _deleteData(int id) async {
    await dbHelper.delete(id);
    _refreshData();
  }

  void _showEditDialog(Map<String, dynamic> item) {
    _titleController.text = item['title'];
    _descriptionController.text = item['description'];

    showDialog(
        context: context,
        builder: (context) {
          return AlertDialog(
            title: const Text('Edit Item'),
            content: Column(
              mainAxisSize: MainAxisSize.min,
              children: [
                TextField(
                  controller: _titleController,
                  decoration: const InputDecoration(labelText:
'Title'),
                ),
                TextField(
                  controller: _descriptionController,
```

```
                decoration: const InputDecoration(labelText:
'Description'),
              ),
            ],
          ),
          actions: [
            TextButton(
              onPressed: () => Navigator.of(context).pop(),
              child: const Text('Cancel'),
            ),
            TextButton(
              onPressed: () {
                _updateData(item['id']);
                Navigator.of(context).pop();
              },
              child: const Text('Save'),
            ),
          ],
        );
      });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Praktikum Database - sqflite'),
        backgroundColor: Colors.blueAccent,
        centerTitle: true,
      ),
      body: Column(
        children: [
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: TextField(
              controller: _titleController,
              decoration: const InputDecoration(labelText:
'Title'),
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: TextField(
              controller: _descriptionController,
              decoration: const InputDecoration(labelText:
'Description'),
            ),
```

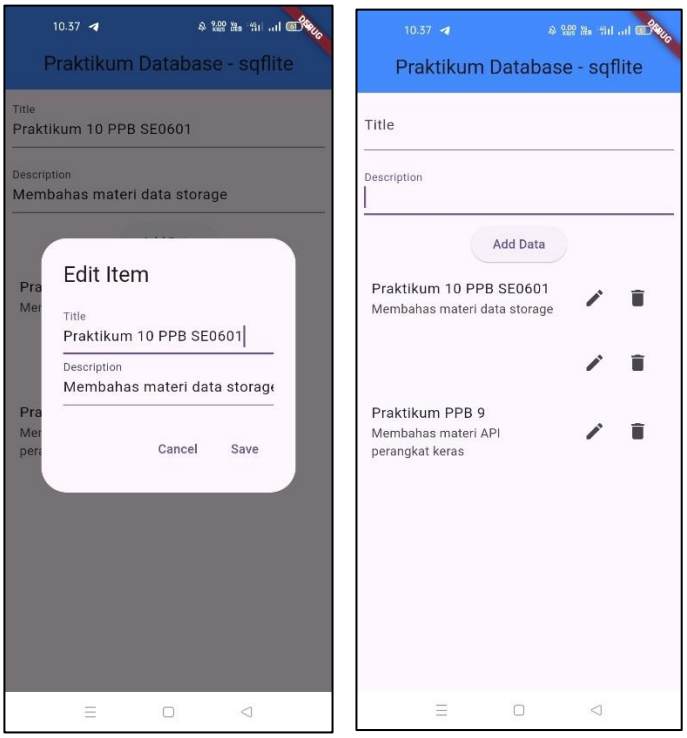
```
          ),
          ElevatedButton(
            onPressed: _addData,
            child: const Text('Add Data'),
          ),
          Expanded(
            child: ListView.builder(
              itemCount: _dbdata.length,
              itemBuilder: (context, index) {
                final item = _dbdata[index];
                return ListTile(
                  title: Text(item['title']),
                  subtitle: Text(item['description']),
                  trailing: Row(
                    mainAxisSize: MainAxisSize.min,
                    children: [
                      IconButton(
                        icon: const Icon(Icons.edit),
                        onPressed: () => _showEditDialog(item),
                      ),
                      IconButton(
                        icon: const Icon(Icons.delete),
                        onPressed: () => _deleteData(item['id']),
                      ),
                    ],
                  ),
                );
              },
            ),
          ),
        ],
      ),
    );
  }
}
```

Hasil Output

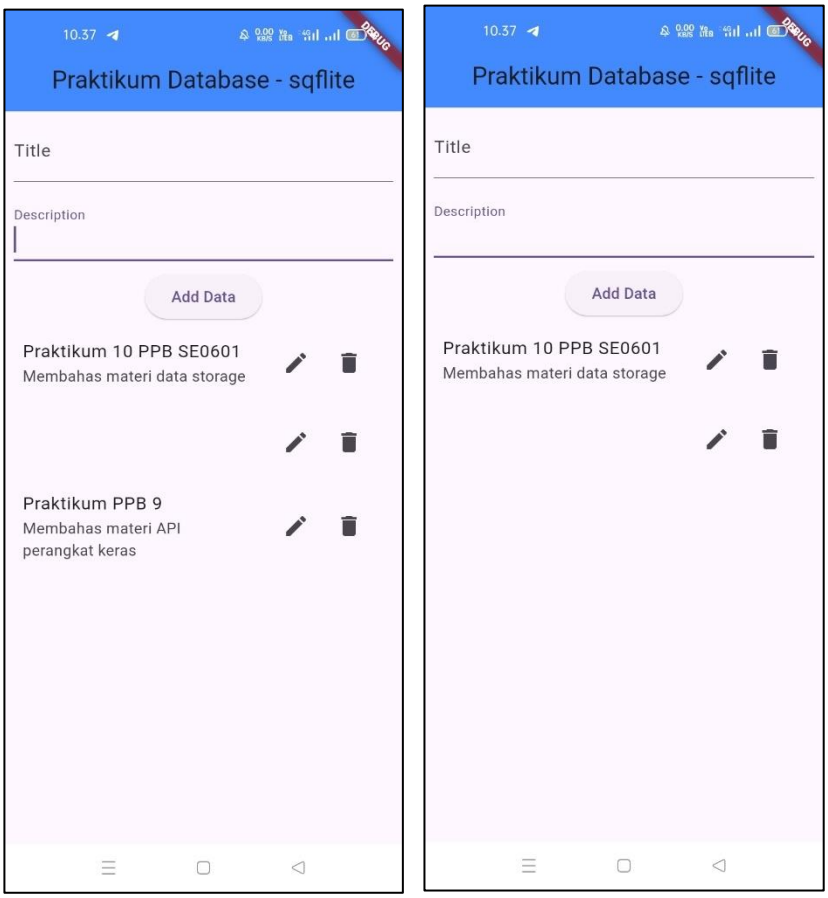
Tampilan halaman utama yang masih kosong, pengguna dapat menginputkan data dengan memasukkan **Title** dan **Description.**



Data dapat diedit dengan memilih ikon edit.



Selain itu, data juga dapat dihapus dengan memilih ikon hapus.Berikut tampilan sebelum dan sesudah dihapus.



UNGUIDED

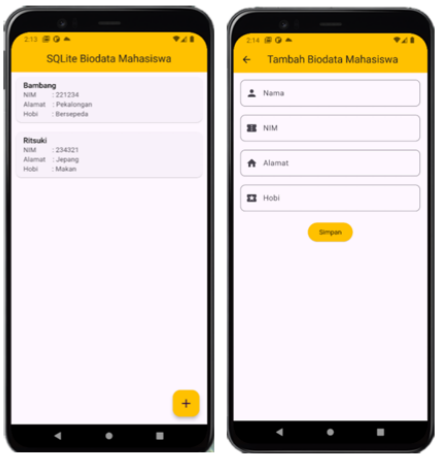
(Soal) Buatlah sebuah project aplikasi Flutter dengan SQLite untuk menyimpan data biodata mahasiswa yang terdiri dari nama, NIM, domisili, dan hobi. Data yang dimasukkan melalui form akan ditampilkan dalam daftar di halaman utama. Alur Aplikasi:

a) Form Input: Buat form input untuk menambahkan biodata mahasiswa, dengan kolom: - - - - Nama Nim Alamat Hobi

b) Tampilkan Daftar Mahasiswa: Setelah data berhasil ditambahkan, tampilkan daftar semua data mahasiswa yang sudah disimpan di halaman utama.

c) Implementasikan fitur Create (untuk menyimpan data mahasiswa) dan Read (untuk menampilkan daftar mahasiswa yang sudah disimpan).

d) Contoh output



Note: Jangan lupa sertakan source code, screenshoot output, dan deskripsi program. Kreatifitas menjadi nilai tambah.

Source Code Unguided

File main.dart

```dart
import 'package:flutter/material.dart';
import 'package:unguided_modul_10/view/my_view.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Biodata Mahasiswa',
      theme: ThemeData(
        primarySwatch: Colors.amber,
      ),
      home: MyDatabaseView(),
    );
  }
}
```

File db_helper.dart

```dart
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';

class DatabaseHelper {
  static final DatabaseHelper _instance =
DatabaseHelper._internal();
  static Database? _database;

  factory DatabaseHelper() => _instance;

  DatabaseHelper._internal();

  Future<Database> get database async {
    if (_database != null) return _database!;
    _database = await _initDatabase();
    return _database!;
  }

  Future<Database> _initDatabase() async {
    String path = join(await getDatabasesPath(),
'biodata_mahasiswa.db');
```

```
    return await openDatabase(
      path,
      version: 1,
      onCreate: (db, version) async {
        await db.execute('''
          CREATE TABLE mahasiswa(
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            nama TEXT,
            nim TEXT,
            alamat TEXT,
            hobi TEXT
          )
        ''');
      },
    );
  }

  Future<int> insert(Map<String, dynamic> row) async {
    Database db = await database;
    return await db.insert('mahasiswa', row);
  }

  Future<List<Map<String, dynamic>>> queryAllRows() async {
    Database db = await database;
    return await db.query('mahasiswa');
  }

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

  // Fungsi Update
  Future<int> update(Map<String, dynamic> row) async {
    Database db = await database;
    // Pastikan id ada dalam map, untuk mengetahui baris mana yang
akan diupdate
    return await db.update(
      'mahasiswa',
      row,
      where: 'id = ?',
      whereArgs: [row['id']],
    );
  }
}
```

File my_view.dart

```dart
import 'package:flutter/material.dart';
import 'package:unguided_modul_10/helper/db_helper.dart';
import 'package:unguided_modul_10/screens/add_data_page.dart';
import 'package:unguided_modul_10/screens/edit_data_page.dart';

class MyDatabaseView extends StatefulWidget {
  const MyDatabaseView({super.key});

  @override
  State<MyDatabaseView> createState() => _MyDatabaseViewState();
}

class _MyDatabaseViewState extends State<MyDatabaseView> {
  final DatabaseHelper dbHelper = DatabaseHelper();
  List<Map<String, dynamic>> _dataMahasiswa = [];

  @override
  void initState() {
    super.initState();
    _refreshData();
  }

  void _refreshData() async {
    final data = await dbHelper.queryAllRows();
    setState(() {
      _dataMahasiswa = data;
    });
  }

  void _navigateToAddDataPage() async {
    final result = await Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => AddDataPage(dbHelper: dbHelper),
      ),
    );
    if (result == true) {
      _refreshData();
    }
  }

  void _navigateToEditDataPage(Map<String, dynamic> mahasiswa)
async {
    final result = await Navigator.push(
      context,
```

```
      MaterialPageRoute(
        builder: (context) => EditDataPage(
          dbHelper: dbHelper,
          mahasiswa: mahasiswa,
        ),
      ),
    );
    if (result == true) {
      _refreshData();
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('SQLite Biodata Mahasiswa'),
        backgroundColor: Colors.amber,
      ),
      body: _dataMahasiswa.isEmpty
          ? const SizedBox() // Tidak ada teks atau widget lain
          : ListView(
              children: _dataMahasiswa.map((mahasiswa) {
                return Card(
                  margin:
                      const EdgeInsets.symmetric(horizontal: 16,
vertical: 8),
                  child: ListTile(
                    title: Text(mahasiswa['nama']),
                    subtitle: Text(
                        'NIM: ${mahasiswa['nim']}\nAlamat:
${mahasiswa['alamat']}\nHobi: ${mahasiswa['hobi']}'),
                    onTap: () =>
_navigateToEditDataPage(mahasiswa),
                  ),
                );
              }).toList(),
            ),
      floatingActionButton: FloatingActionButton(
        onPressed: _navigateToAddDataPage,
        child: const Icon(Icons.add),
        backgroundColor: Colors.amber,
      ),
    );
  }
}
```

File add_data_page.dart

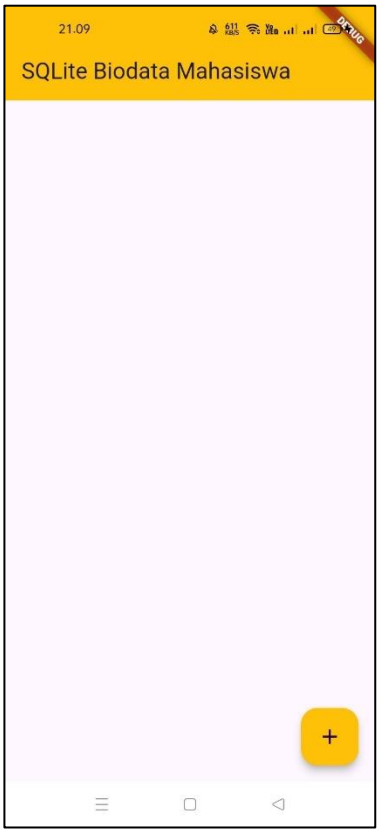
```dart
import 'package:flutter/material.dart';
import 'package:unguided_modul_10/helper/db_helper.dart';

class AddDataPage extends StatefulWidget {
  final DatabaseHelper dbHelper;

  const AddDataPage({super.key, required this.dbHelper});

  @override
  State<AddDataPage> createState() => _AddDataPageState();
}

class _AddDataPageState extends State<AddDataPage> {
  final TextEditingController _namaController =
TextEditingController();
  final TextEditingController _nimController =
TextEditingController();
  final TextEditingController _alamatController =
TextEditingController();
  final TextEditingController _hobiController =
TextEditingController();

  void _saveData() async {
    if (_namaController.text.isNotEmpty &&
        _nimController.text.isNotEmpty &&
        _alamatController.text.isNotEmpty &&
        _hobiController.text.isNotEmpty) {
      // Simpan data ke database
      await widget.dbHelper.insert({
        'nama': _namaController.text,
        'nim': _nimController.text,
        'alamat': _alamatController.text,
        'hobi': _hobiController.text,
      });

      // Bersihkan input form
      _namaController.clear();
      _nimController.clear();
      _alamatController.clear();
      _hobiController.clear();

      // Kembalikan ke halaman sebelumnya dengan indikator sukses
      Navigator.pop(context, true);
    } else {
      // Tampilkan peringatan jika form tidak lengkap
```

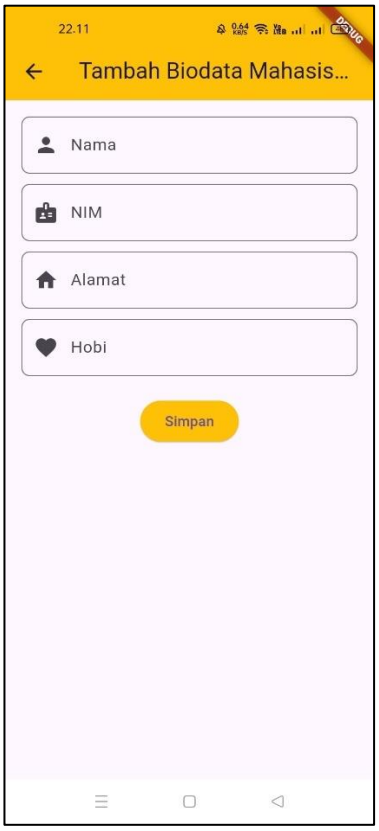
```
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Mohon lengkapi semua
kolom!')),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Tambah Biodata Mahasiswa'),
        backgroundColor: Colors.amber,
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            TextField(
              controller: _namaController,
              decoration: InputDecoration(
                labelText: 'Nama',
                prefixIcon: const Icon(Icons.person),
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(8.0),
                ),
              ),
            ),
            const SizedBox(height: 10),
            TextField(
              controller: _nimController,
              decoration: InputDecoration(
                labelText: 'NIM',
                prefixIcon: const Icon(Icons.badge),
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(8.0),
                ),
              ),
            ),
            const SizedBox(height: 10),
            TextField(
              controller: _alamatController,
              decoration: InputDecoration(
                labelText: 'Alamat',
                prefixIcon: const Icon(Icons.home),
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(8.0),
```

```
              ),
            ),
          ),
          const SizedBox(height: 10),
          TextField(
            controller: _hobiController,
            decoration: InputDecoration(
              labelText: 'Hobi',
              prefixIcon: const Icon(Icons.favorite),
              border: OutlineInputBorder(
                borderRadius: BorderRadius.circular(8.0),
              ),
            ),
          ),
          const SizedBox(height: 20),
          ElevatedButton(
            onPressed: _saveData,
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.amber,
            ),
            child: const Text('Simpan'),
          ),
        ],
      ),
    ),
  );
 }
}
```

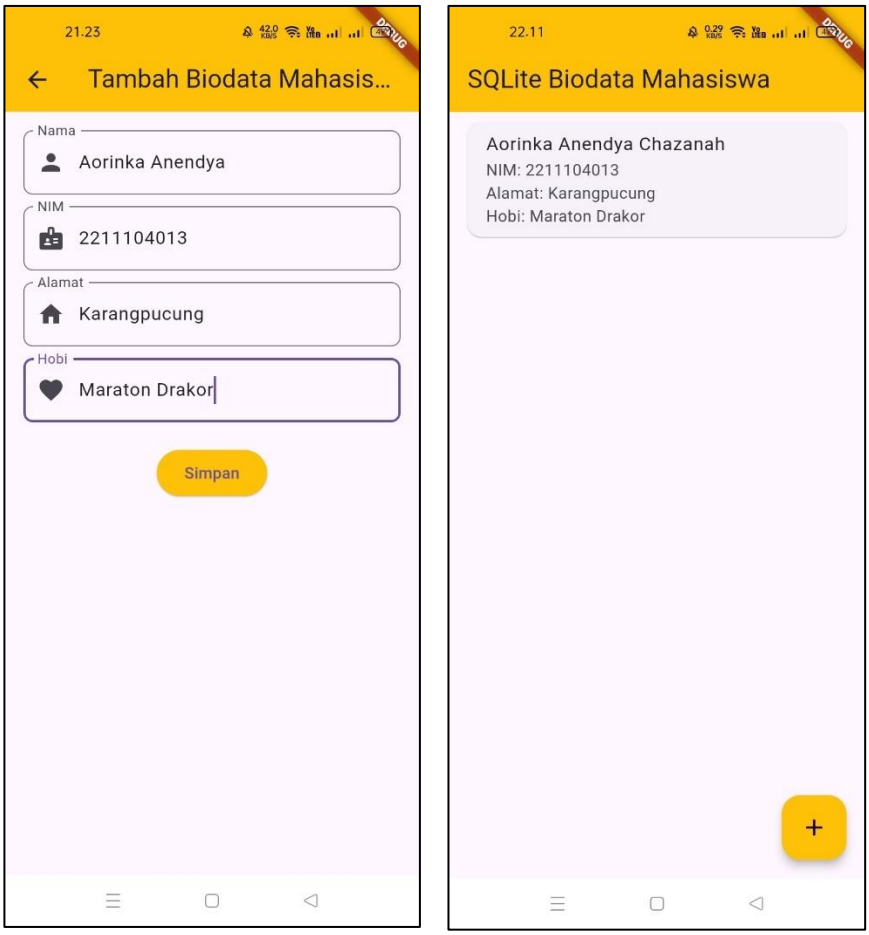
Hasil Output

Berikut ini adalah halaman awal ketika aplikasi dibuka dan belum ada data yang ditambahkan





Berikut adalah proses insert data dan data akan tampil di halaman utama.



Deskripsi Program

- **Fungsi Main.dart**: Pada main.dart, aplikasi dimulai dengan widget MyDatabaseView, yang menampilkan daftar mahasiswa dari database SQLite. Jika tidak ada data, hanya tombol floating button. Tombol ini mengarahkan pengguna ke halaman AddDataPage untuk menambah data mahasiswa baru.
- **Fungsi db_helper.dart**: File db_helper.dart mengelola operasi database SQLite, seperti membuat, memasukkan, mengambil, dan menghapus data. Fungsi _initDatabase() menginisialisasi database, insert() menambah data, queryAllRows() mengambil data, dan delete() menghapus data berdasarkan ID.
- **Fungsi my_view.dart**: File my_view.dart menampilkan data mahasiswa dalam ListView dan menggunakan ListTile untuk tiap item. Selain menampilkan data, halaman sini juga memiliki tombol FAB untuk menambah data mahasiswa baru, serta memastikan tampilan selalu diperbarui saat data berubah.

- **Fungsi add_data_page.dart**: menyediakan formulir untuk menambah data mahasiswa baru. Pengguna mengisi kolom input, dan setelah menekan "Simpan", data disimpan ke database SQLite melalui fungsi insert() di DatabaseHelper, lalu kembali ke halaman utama dengan daftar mahasiswa yang diperbarui.