LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK

MODUL X

DATA STORAGE (BAGIAN 1)

Disusun Oleh :

Dewi Atika Muthi / 2211104042

SE-06-02

Asisten Praktikum :

Muhammad Faza Zulian 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

BAB I
PENDAHULUAN

A. DASAR TEORI

SQLite merupakan sebuah sistem manajemen basis data relasional yang bersifat self-contained, serverless, dan zero-configuration. Berbeda dengan sistem database konvensional yang memerlukan server terpisah, SQLite terintegrasi langsung ke dalam aplikasi sebagai bagian dari programnya. Dalam konteks pengembangan aplikasi mobile, SQLite menjadi pilihan populer karena karakteristiknya yang ringan dan efisien.

Beberapa konsep penting dalam SQLite:

1. Local Storage: SQLite menyimpan seluruh database dalam satu file di local storage perangkat, tepatnya pada cache memory aplikasi. Hal ini membuat akses data menjadi sangat cepat dan efisien.
2. CRUD Operations: SQLite mendukung operasi dasar database yaitu:
   - Create: Membuat record baru
   - Read: Membaca atau mengambil data
   - Update: Memperbarui data yang ada
   - Delete: Menghapus data
3. SQL Helper: Dalam Flutter, SQL Helper adalah class yang berisi kumpulan method untuk melakukan operasi database. Package sqflite digunakan sebagai jembatan antara Flutter dan SQLite [3].

Arsitektur penyimpanan data SQLite dalam Flutter dapat digambarkan sebagai berikut:

```
[Flutter App] <-> [SQL Helper] <-> [SQLite Database] <-> [Local Storage]
```

B. MAKSUD DAN TUJUAN

Maksud dan tujuan praktikum kali ini adalah:

- Memahami konsep dan implementasi penyimpanan data lokal menggunakan SQLite dalam pengembangan aplikasi Flutter
- Menguasai operasi CRUD (Create, Read, Update, Delete) pada database SQLite
- Mampu mengimplementasikan SQL Helper untuk manajemen database
- Dapat membuat aplikasi Flutter yang memanfaatkan penyimpanan data lokal
- Mengembangkan kemampuan dalam mengelola data persistent pada aplikasi mobile

BAB II
IMPLEMENTASI

A. PRAKTIKUM (GUIDED)

Soal Studi Case

Membuat form sederhana dan menyimpannya di database

1. Menambahkan plugin ke pubspec.yaml

```
 sqflite: ^2.4.1
path: ^1.9.0
```

2. Buat file db_helper.dart untuk mengelola database dan import package sqflite dan path.

Sourcecode helper/db_helper.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();

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

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

  // Membuat tabel saat db pertama kali dibuat
  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)
''');
  }

  // Metode untuk memasukkan data kedalam tabel
  Future<int> insert(Map<String, dynamic> row) async {
    Database db = await database;
    return await db.insert('my_table', row);
```

```
  }

  // Metode untuk mengambil semua data dari tabel
  Future<List<Map<String, dynamic>>> queryAllRows() async {
    Database db = await database;
    return await db.query('my_table');
  }

  // 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]);
  }

  // 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]);
  }
}
```

3. Membuat file my_db_view.dart untuk membuat antarmuka pengguna (UI) untuk mengelola data dalam database.

Sourcecode view/my_db_view.dart

```
import 'package:flutter/material.dart';
import 'package:md7/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();
  }

  // Metode untuk memperbarui data dari database
  void _refreshData() async {
    final data = await dbHelper.queryAllRows();
    setState(() {
      _dbData = data;
    });
  }

  // Metode untuk menambahkan data ke database
  void _addData() async {
```

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

  // Metode untuk memperbarui data dalam database
  void _updateData(int id) async {
    await dbHelper.update({
      'id': id,
      'title': _titleController.text,
      'description': _descriptionController.text,
    });
    _titleController.clear();
    _descriptionController.clear();
    _refreshData();
  }

  // Metode untuk menghapus data dari database
  void _deleteData(int id) async {
    await dbHelper.delete(id);
    _refreshData();
  }

  // Menampilkan dialog untuk mengedit data
  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(
        backgroundColor: Colors.pink[200],
        centerTitle: true,
        title: const Text('Praktikum Database - sqlfile'),
      ),
      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']),
                      ),
                    ],
                  ),
                );
              },
            ),
          ),
        ],
      ),
    );
  }
}
```

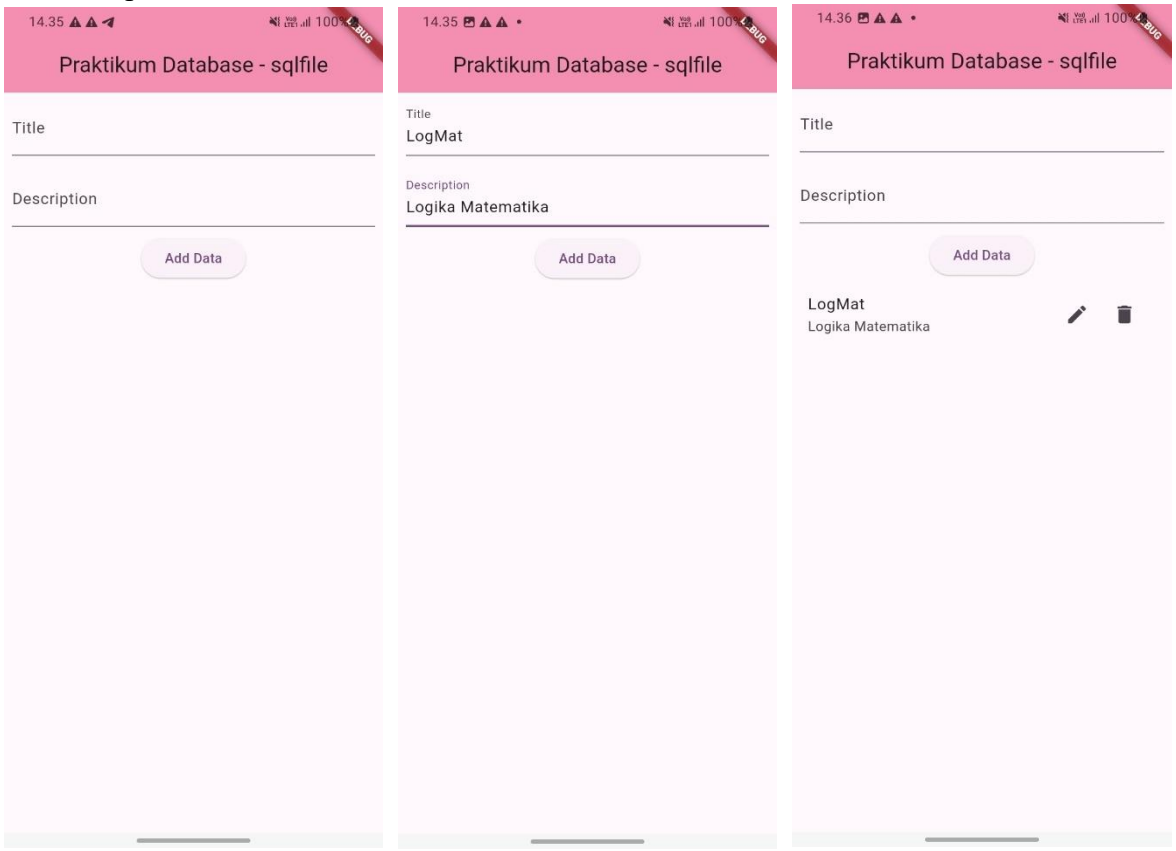
4. Membuat main.dart untuk menjalankan aplikasi

Sourcecode helper/db_helper.dart

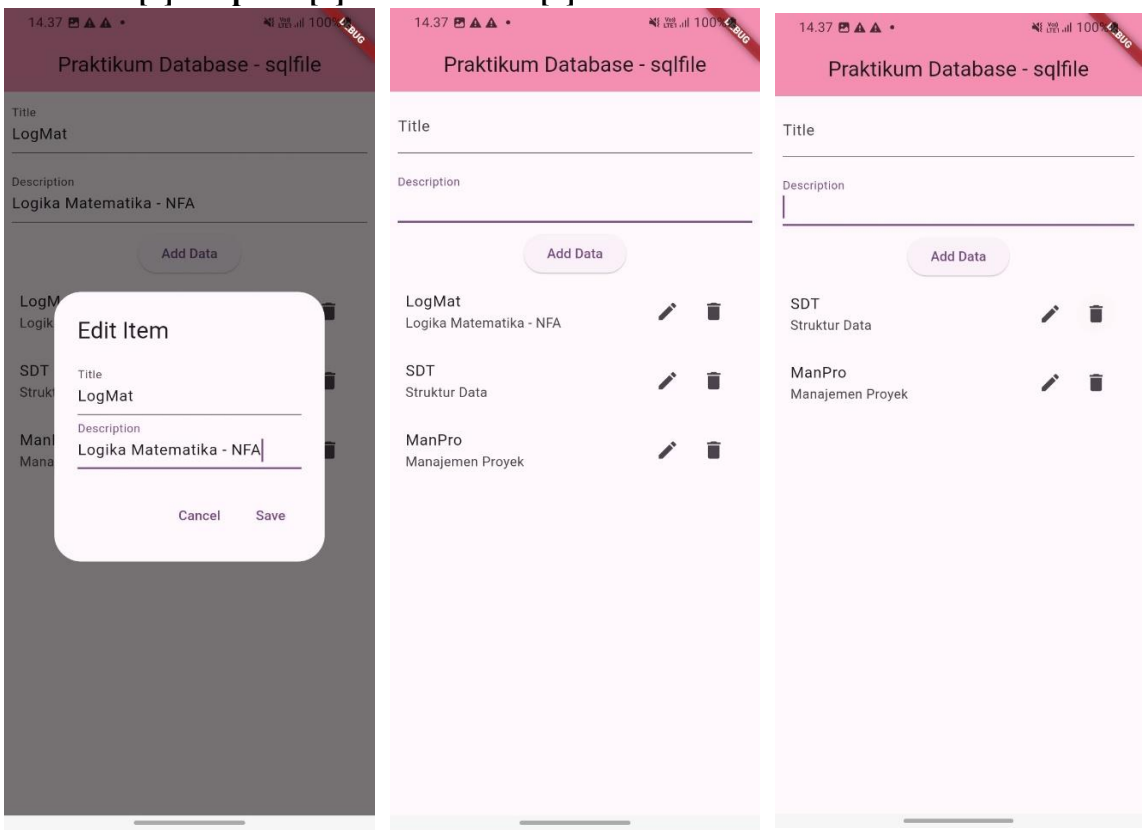
```
import 'package:flutter/material.dart';
import 'package:md7/view/my_db_view.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor:
Colors.purple.shade100),
        useMaterial3: true,
      ),
      home: const MyDatabaseView(), // Arahkan ke MyDatabaseView
    );
  }
}
```

Screenshoot Output

Start > Input Elemen > Add data



Edit Data[0] > Update[0] > Delete Data[0]



Deskripsi Program

Dalam program olah data ini terdapat form dengan isi field title dan description, yang dimana program ini menggunakan 3 file yaitu db_helper.dart, my_db_view.dart, dan main.dart.

Dalam db_helper.dart sendiri berisikan kelas DatabaseHelper, berisikan metode untuk mempermudah manipulasi data seperti insert, queryAllRows, update, dan delete. Selanjutnya ada my_db_view.dart untuk membuat UI sehingga pengguna bisa menambahkan, mengedit, dan menghapus data melalui elemen UI seperti text fields, tombol, dan list.

Jika dilihat dari gambar output, dengan program ini, pengguna dapat menambah data berupa judul dan deskripsi melalui form input, melihat data dalam bentuk daftar, memperbarui data dengan membuka dialog edit, menghapus data dengan tombol delete.

B. Tugas Mandiri (UNGUIDED)

Soal Studi Case

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

Sourcecode helper/database_helper.dart:

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

class DatabaseHelper {
```

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

  factory DatabaseHelper() {
    return _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: _onCreate,
    );
  }

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

  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', orderBy: 'id ASC');
  }
}
```

Sourcecode view/biodata_view.dart:

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

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

  @override
  State<BiodataMahasiswaView> createState() => _BiodataMahasiswaViewState();
}

class _BiodataMahasiswaViewState extends State<BiodataMahasiswaView> {
  final DatabaseHelper dbHelper = DatabaseHelper();
  List<Map<String, dynamic>> _mahasiswaList = [];

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

```
  }

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.pink,
        title: const Text('SQLite Biodata Mahasiswa'),
        titleTextStyle: TextStyle(color: Colors.white, fontSize: 22),
      ),
      body: Column(
        children: [
          Expanded(
            child: _mahasiswaList.isEmpty
                ? const Center(
                    child: Column(
                      mainAxisAlignment: MainAxisAlignment.center,
                      children: [
                        Icon(
                          Icons.person_search,
                          size: 64,
                          color: Colors.grey,
                        ),
                        SizedBox(height: 16),
                        Text(
                          'Belum ada data mahasiswa',
                          style: TextStyle(
                            color: Colors.grey,
                            fontSize: 16,
                          ),
                        ),
                      ],
                    ),
                  )
                : ListView.builder(
                    itemCount: _mahasiswaList.length,
                    itemBuilder: (context, index) {
                      final mahasiswa = _mahasiswaList[index];
                      return Card(
                        margin: const EdgeInsets.all(8),
                        child: Stack(
                          children: [
                            Padding(
                              padding: const EdgeInsets.all(16),
                              child: Column(
                                crossAxisAlignment: CrossAxisAlignment.start,
                                children: [
                                  Text(
                                    mahasiswa['nama'] ?? '',
                                    style: const TextStyle(
                                      fontSize: 18,
                                      fontWeight: FontWeight.bold,
                                    ),
                                  ),
                                  const SizedBox(height: 8),
                                  Text('NIM: ${mahasiswa['nim']}'),
                                  Text('Alamat: ${mahasiswa['alamat']}'),
                                  Text('Hobi: ${mahasiswa['hobi']}'),
                                ],
                              ),
                            ),
```

```
                          ],
                        ),
                      );
                    },
                  ),
            ),
          ],
        ),
        floatingActionButton: FloatingActionButton(
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => _BuildInputForm(
                  refreshData: _refreshData,
                ),
              ),
            );
          },
          child: const Icon(Icons.add),
        ),
      );
  }
}

class _BuildInputForm extends StatelessWidget {
  final Function refreshData;
  final Map<String, dynamic>? existingData;
  final DatabaseHelper dbHelper = DatabaseHelper();
  final TextEditingController _namaController = TextEditingController();
  final TextEditingController _nimController = TextEditingController();
  final TextEditingController _alamatController = TextEditingController();
  final TextEditingController _hobiController = TextEditingController();

  _BuildInputForm({required this.refreshData, this.existingData}) {
    if (existingData != null) {
      _namaController.text = existingData!['nama'] ?? '';
      _nimController.text = existingData!['nim'] ?? '';
      _alamatController.text = existingData!['alamat'] ?? '';
      _hobiController.text = existingData!['hobi'] ?? '';
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(
          'Tambah Biodata Mahasiswa',
          style: TextStyle(color: Colors.white),
        ),
        backgroundColor: Colors.pink,
        leading: IconButton(
          icon: const Icon(Icons.arrow_back, color: Colors.white),
          onPressed: () => Navigator.pop(context),
        ),
      ),
      body: SingleChildScrollView(
        padding: const EdgeInsets.all(16),
        child: Column(
          children: [
            TextField(
              controller: _namaController,
              decoration: const InputDecoration(
                labelText: 'Nama',
                prefixIcon: Icon(Icons.person, color: Colors.pink),
                border: OutlineInputBorder(),
                focusedBorder: OutlineInputBorder(
```

```
                borderSide: BorderSide(color: Colors.pink),
              ),
            ),
          ),
          const SizedBox(height: 16),
          TextField(
            controller: _nimController,
            decoration: const InputDecoration(
              labelText: 'NIM',
              prefixIcon: Icon(Icons.badge, color: Colors.pink),
              border: OutlineInputBorder(),
              focusedBorder: OutlineInputBorder(
                borderSide: BorderSide(color: Colors.pink),
              ),
            ),
          ),
          const SizedBox(height: 16),
          TextField(
            controller: _alamatController,
            decoration: const InputDecoration(
              labelText: 'Alamat',
              prefixIcon: Icon(Icons.home, color: Colors.pink),
              border: OutlineInputBorder(),
              focusedBorder: OutlineInputBorder(
                borderSide: BorderSide(color: Colors.pink),
              ),
            ),
          ),
          const SizedBox(height: 16),
          TextField(
            controller: _hobiController,
            decoration: const InputDecoration(
              labelText: 'Hobi',
              prefixIcon: Icon(Icons.favorite, color: Colors.pink),
              border: OutlineInputBorder(),
              focusedBorder: OutlineInputBorder(
                borderSide: BorderSide(color: Colors.pink),
              ),
            ),
          ),
          const SizedBox(height: 24),
          ElevatedButton.icon(
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.pink,
              minimumSize: const Size.fromHeight(50),
              foregroundColor: Colors.white,
            ),
            onPressed: () async {
              if (_namaController.text.isEmpty ||
                  _nimController.text.isEmpty ||
                  _alamatController.text.isEmpty ||
                  _hobiController.text.isEmpty) {
                ScaffoldMessenger.of(context).showSnackBar(
                  const SnackBar(
                    content: Text('Semua field harus diisi!'),
                    backgroundColor: Colors.red,
                  ),
                );
                return;
              }

              await dbHelper.insert({
                'nama': _namaController.text,
                'nim': _nimController.text,
                'alamat': _alamatController.text,
                'hobi': _hobiController.text,
              });
              refreshData();
```

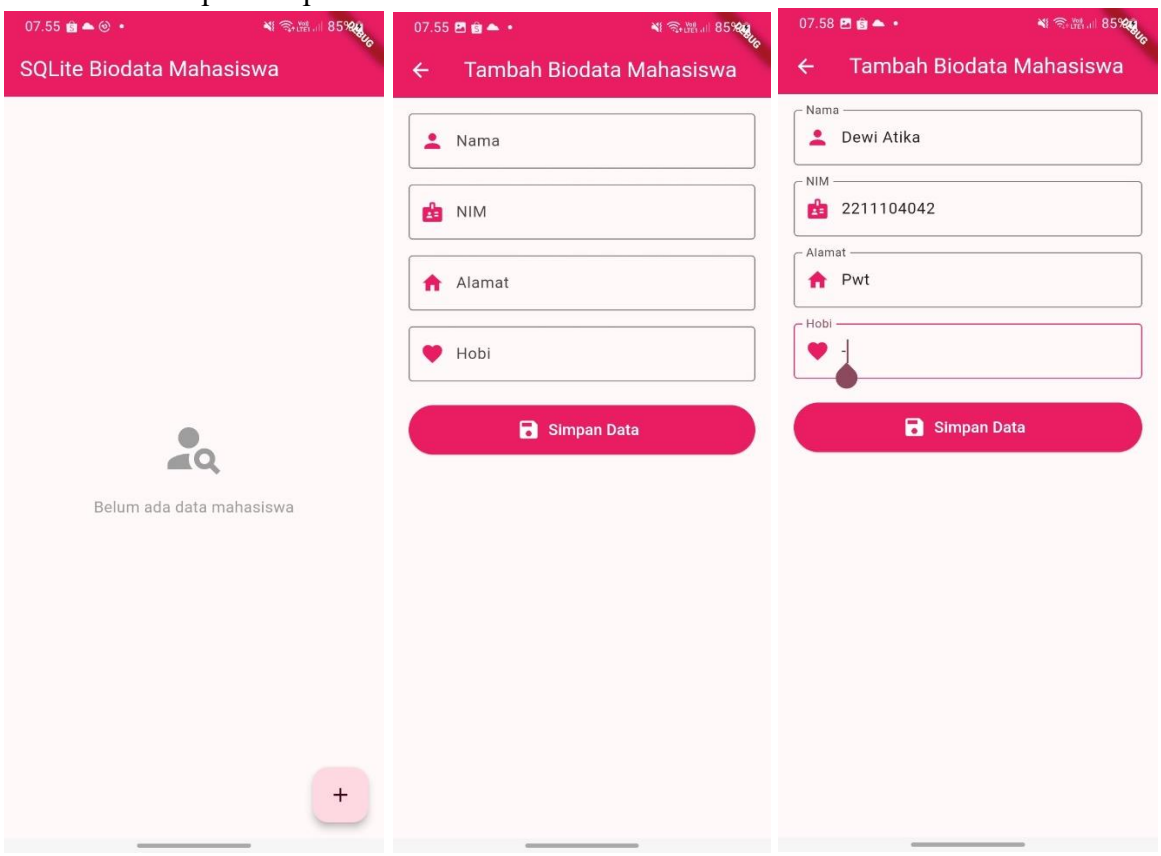
```
                Navigator.pop(context);

                ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(
                    content: const Text('Data berhasil disimpan!'),
                    backgroundColor: Colors.green,
                  ),
                );
              },
              icon: const Icon(Icons.save),
              label: const Text(
                'Simpan Data',
                style: TextStyle(fontSize: 16),
              ),
            ),
          ),
        ],
      ),
    ),
  );
  }
}
```

Sourcecode main.dart:

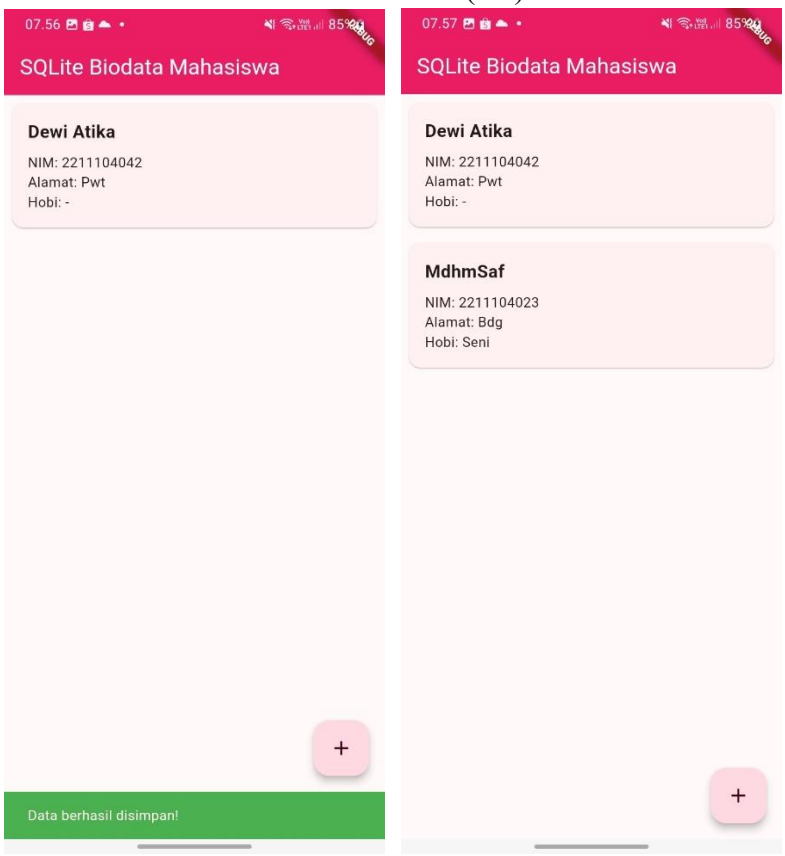
```
import 'package:flutter/material.dart';
import 'view/biodata_view.dart';

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(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.pink.shade300),
        useMaterial3: true,
      ),
      home: const BiodataMahasiswaView(),
    );
  }
}
```

Screenshoot Output

Start > Form input > Input Data



Submit data > Submit Another Data(list)



Deskripsi Program

database_helper.dart berfungsi mengelola database SQLite, seperti membuat tabel mahasiswa dengan kolom seperti id, nama, nim, alamat, dan hobi. Memasukkan data ke dalam tabel (insert), lalu mengambil semua data yang tersimpan di tabel (queryAllRows).

biodata_view.dart berfungi untuk membuat tampilan utama aplikasi, berisi daftar mahasiswa yang diambil dari database dalam bentuk ListView. Lalu ada tombol tambah data menggunakan FloatingActionButton yang mengarahkan pengguna ke form input.

Form Input (_BuildInputForm) untuk menambahkan biodata baru ke database. Menggunakan ScaffoldMessenger untuk menampilkan notifikasi ketika data berhasil disimpan atau validasi gagal.

BAB III
KESIMPULAN DAN SARAN

A. KESIMPULAN

Pembelajaran pada modul 10 tentang Data Storage menggunakan SQLite dalam Flutter telah memberikan pemahaman mendalam tentang manajemen data lokal dalam pengembangan aplikasi mobile. Melalui praktikum ini, kita memahami bahwa SQLite itu untuk menyimpan data secara persisten di perangkat mobile tanpa memerlukan koneksi server. Implementasi CRUD operations menggunakan SQL Helper memudahkan dalam mengelola data aplikasi secara terstruktur. Penggunaan database lokal juga meningkatkan performa aplikasi karena data dapat diakses dengan cepat tanpa bergantung pada koneksi internet. Praktikum ini juga menunjukkan pentingnya arsitektur yang baik dalam pengelolaan data, dimana pemisahan antara logic database dan UI membuat code lebih maintainable.

B. REFERENSI

SQLite. "About SQLite". SQLite.org, 2024.

Vaish, Gaurav. "Getting Started with SQLite Database in Flutter". Medium, 2023.

Flutter Team. "SQLite Plugin Documentation". flutter.dev, 2024.