

**LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK**

**MODUL X
DATA STORAGE (BAGIAN I)**



Disusun Oleh :

Devrin Anggun Saputri / 2211104001

SE0601

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

GUIDED

1. Pengenalan SQLite

SQLite adalah database relasional yang berfungsi sebagai penyimpanan data secara offline dalam aplikasi mobile, khususnya pada local storage seperti cache memory aplikasi. SQLite mendukung empat operasi dasar, yaitu CRUD (create, read, update, dan delete), yang menjadi bagian penting dalam pengelolaan data. Struktur database pada SQLite serupa dengan SQL secara umum, termasuk dalam hal variabel dan tipe data yang digunakan.

2. SQL Helper Dasar

Dalam Flutter, SQL Helper biasanya mengacu pada pemanfaatan paket seperti **sqflite** untuk mengelola database SQLite. SQL Helper merupakan sebuah class yang digunakan untuk mendefinisikan berbagai metode yang berkaitan dengan manipulasi data. Plugin **sqflite** memungkinkan pengembang melakukan operasi CRUD (Create, Read, Update, Delete) pada database SQLite secara efisien.

Berikut adalah langkah-langkah dasar untuk menggunakan sqflite sebagai SQL Helper di Flutter :

1. Tambahkan plugin sqflite dan path ke file pubspec.yaml.
2. Buat class baru bernama DatabaseHelper untuk mengelola database dan import package sqflite dan path di file db_helper.dart.

```
1 import 'package:sqflite/sqflite.dart';
2 import 'package:path/path.dart';
3
4 class DatabaseHelper {
5   static final DatabaseHelper _instance = DatabaseHelper._internal();
6   static Database? _database;
```

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

```
// 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!;
  }
}
```

4. Buat Getter untuk database.
5. Inisialisasi database dengan nama database yang kita mau.

```
// inisiasi 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,
  );
}
```

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

```
//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)
''');
}
```

7. Buat metode untuk memasukkan data ke dalam tabel.

```
Future<int> insert(Map<String, dynamic> row) async {
  Database db = await database;
  int result = await db.insert('my_table', row);
  print('Inserted row: $result'); // Log untuk memeriksa hasil insert
  return result;
}
```

8. Lalu, metode untuk mengambil semua data dari tabel.

```
// metode mengambil semua data dari tabel
Future<List<Map<String, dynamic>>> queryAllRows() async {
  Database db = await database;
  List<Map<String, dynamic>> result = await db.query('my_table');
  print('Fetched data: $result'); // Log untuk memeriksa data yang diambil
  return result;
}
```

9. Buat metode untuk memperbarui data dalam tabel.

```
// 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.

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

Praktikum

1. Source Code:

- main.dart

```
1 import 'package:flutter/material.dart';
2 import 'package:praktikum_10/view/my_db_view.dart';
3
4 void main() {
5   runApp(MyApp());
6 }
7
8 class MyApp extends StatelessWidget {
9   @override
10  Widget build(BuildContext context) {
11    return MaterialApp(
12      title: 'Praktikum Data Storage',
13      theme: ThemeData(
14        primarySwatch: Colors.orange,
15      ),
16      home: MyDatabaseView(),
17    );
18  }
19 }
```

- db_helper.dart

```
1 import 'package:sqflite/sqflite.dart';
2 import 'package:path/path.dart';
3
4 class DatabaseHelper {
5   static final DatabaseHelper _instance = DatabaseHelper._internal();
6   static Database? _database;
7
8   // factory constructor untuk mengembalikan instance singleton
9   factory DatabaseHelper() {
10     return _instance;
11   }
12
13   // Private constructor
14   DatabaseHelper._internal();
15
16   // Getter untuk database
17   Future<Database> get database async {
18     if (_database != null) return _database!;
19     {
20       _database = await _initDatabase();
21       return _database!;
22     }
23   }
24
25   // inisiasi database
26   Future<Database> _initDatabase() async {
27     // mendapatkan path untuk database
28     String path = join(await getDatabasesPath(), 'my_prakdatabase.db');
29     // membuka database
30     return await openDatabase(
```

```

31     path,
32     version: 1,
33     onCreate: _onCreate,
34   );
35 }
36
37 //membuat tabel saat db pertama kali dibuat
38 Future<void> _onCreate(Database db, int version) async {
39   await db.execute('''
40     CREATE TABLE my_table(
41       id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
42       title TEXT,
43       description TEXT,
44       createdAt TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP)
45   ''');
46 }
47
48 // metode memasukan data ke dalam tabel
49 Future<int> insert(Map<String, dynamic> row) async {
50   Database db = await database;
51   int result = await db.insert('my_table', row);
52   print('Inserted row: $result'); // Log untuk memeriksa hasil insert
53   return result;
54 }
55
56 // metode mengambil semua data dari tabel
57 Future<List<Map<String, dynamic>>> queryAllRows() async {
58   Database db = await database;
59   List<Map<String, dynamic>> result = await db.query('my_table');
60   print('Fetched data: $result'); // Log untuk memeriksa data yang diambil
61   return result;
62 }
63
64 // metode untuk memperbarui data dalam tabel
65 Future<int> update(Map<String, dynamic> row) async {
66   Database db = await database;
67   int id = row['id'];
68   return await db.update('my_table', row, where: 'id = ?', whereArgs: [id]);
69 }
70
71 // metode menghapus data dari tabel
72 Future<int> delete(int id) async {
73   Database db = await database;
74   return await db.delete('my_table', where: 'id = ?', whereArgs: [id]);
75 }
76 }

```

- my_db_view.dart

```

1  import 'package:flutter/material.dart';
2  import 'package:praktikum_18/helper/db_helper.dart';
3
4  class MyDatabaseView extends StatefulWidget {
5    const MyDatabaseView({super.key});
6
7    @override
8    State<MyDatabaseView> createState() => _MyDatabaseViewState();
9  }
10
11  class _MyDatabaseViewState extends State<MyDatabaseView> {
12    final DatabaseHelper dbHelper = DatabaseHelper();
13    List<Map<String, dynamic>> _dbData = [];
14    final TextEditingController _titleController = TextEditingController();
15    final TextEditingController _descriptionController = TextEditingController();
16
17    @override
18    void initState() {
19      super.initState();
20      _refreshData();
21    }
22
23    @override
24    void dispose() {
25      _titleController.dispose();
26      _descriptionController.dispose();
27      super.dispose();
28    }
29
30    // Ubah menjadi Future<void> dan beri async
31    Future<void> _refreshData() async {
32      final data = await dbHelper.queryAllRows();
33      print(data); // tambahkan log ini
34      setState(() {
35        _dbData = data;
36      });
37    }

```

```

38
39 // Ubah menjadi Future<void> dan beri async
40 Future<void> _addData() async {
41   await dbHelper.insert({
42     'title': _titleController.text,
43     'description': _descriptionController.text,
44   });
45   _titleController.clear();
46   _descriptionController.clear();
47   await _refreshData(); // Pastikan rerunggu refresh data setelah insert
48 }
49
50 // Ubah menjadi Future<void> dan beri async
51 Future<void> _updateData(int id) async {
52   await dbHelper.update({
53     'id': id,
54     'title': _titleController.text,
55     'description': _descriptionController.text,
56   });
57   _titleController.clear();
58   _descriptionController.clear();
59   await _refreshData(); // Pastikan rerunggu refresh data setelah update
60 }
61
62 // Ubah menjadi Future<void> dan beri async
63 Future<void> _deleteData(int id) async {
64   await dbHelper.delete(id);
65   await _refreshData(); // Pastikan rerunggu refresh data setelah delete
66 }
67
68 // Menampilkan dialog untuk mengedit data
69 void _showEditDialog(Map<String, dynamic> item) {
70   _titleController.text = item['title'];
71   _descriptionController.text = item['description'];
72
73   showDialog(
74     context: context,
75     builder: (context) {
76       return AlertDialog(
77         title: const Text('Edit Item'),
78         content: Column(
79           mainAxisAlignment: MainAxisAlignment.min,
80           children: [
81             TextField(
82               controller: _titleController,
83               decoration: InputDecoration(labelText: 'Title'),
84             ),
85             TextField(
86               controller: _descriptionController,
87               decoration: InputDecoration(labelText: 'Description'),
88             ),
89           ],
90         ),
91         actions: [
92           TextButton(
93             onPressed: () {
94               Navigator.of(context).pop();
95             },
96             child: Text('Cancel'),
97           ),
98           TextButton(
99             onPressed: () {
100               updateData(item['id']);
101               Navigator.of(context).pop();
102             },
103             child: const Text('Save'),
104           ),
105         ],
106       );
107     },
108   );
109 }
110
111 // Menampilkan dialog untuk menambahkan data
112 void _showAddDialog() {
113   _titleController.clear();
114   _descriptionController.clear();
115
116   showDialog(
117     context: context,
118     builder: (context) {
119       return AlertDialog(
120         title: Text('Add New Item'),
121         content: Column(
122           mainAxisAlignment: MainAxisAlignment.min,
123           children: [
124             TextField(
125               controller: _titleController,
126               decoration: InputDecoration(labelText: 'Title'),
127             ),
128             TextField(
129               controller: _descriptionController,
130               decoration: InputDecoration(labelText: 'Description'),
131             ),
132           ],
133         ),
134         actions: [

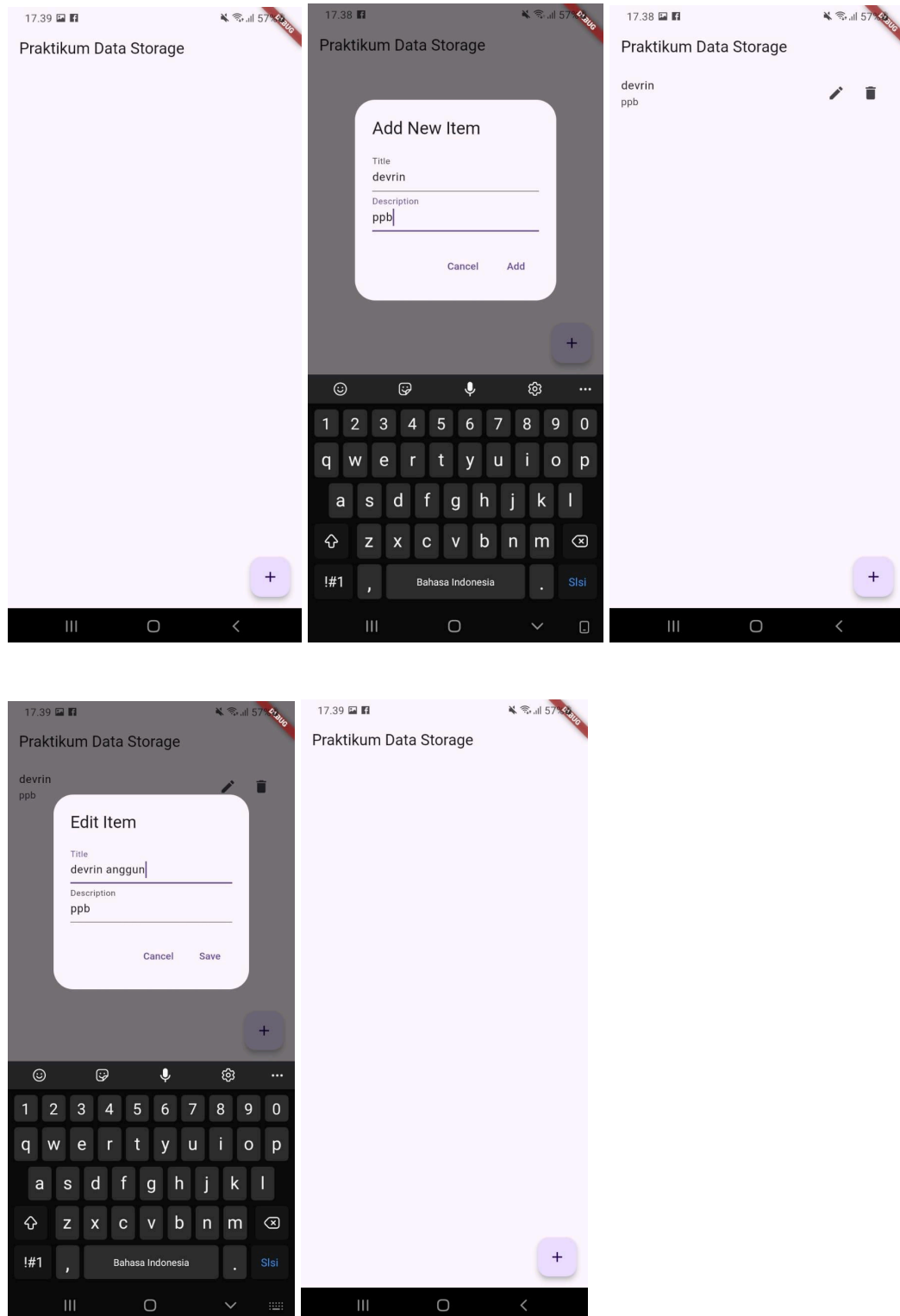
```

```

135         TextButton(
136             onPressed: () {
137                 Navigator.of(context).pop();
138             },
139             child: Text('Cancel'),
140         ),
141         TextButton(
142             onPressed: () {
143                 _addData();
144                 Navigator.of(context).pop();
145             },
146             child: const Text('Add'),
147         ),
148     ],
149 );
150 },
151 );
152 }
153
154 @override
155 Widget build(BuildContext context) {
156     return Scaffold(
157         appBar: AppBar(
158             title: Text('Praktikum Data Storage'),
159         ),
160         body: ListView.builder(
161             itemCount: _dbData.length,
162             itemBuilder: (context, index) {
163                 final item = _dbData[index];
164                 print('Displaying item: ${item['title']}');
165                 return ListTile(
166                     title: Text(item['title'] ?? 'No title'),
167                     subtitle: Text(item['description'] ?? 'No Description'),
168                     trailing: Row(
169                         mainAxisAlignment: MainAxisAlignment.min,
170                         children: [
171                             IconButton(
172                                 icon: Icon(Icons.edit),
173                                 onPressed: () => _showEditDialog(item),
174                             ),
175                             IconButton(
176                                 icon: Icon(Icons.delete),
177                                 onPressed: () => _deleteData(item['id']),
178                             ),
179                         ],
180                     ),
181                 );
182             },
183         ),
184         floatingActionButton: FloatingActionButton(
185             onPressed: _showAddDialog,
186             child: Icon(Icons.add),
187         ),
188     );
189 }
190 }

```

2. Output Hasil



Halaman awal, add data, kolom data, edit/hapus data.

3. Penjelasan Program

Kode di atas adalah implementasi aplikasi Flutter untuk menyimpan, mengelola, dan menampilkan data menggunakan SQLite melalui plugin **sqflite**.

1. DatabaseHelper Class

DatabaseHelper adalah class yang mengelola koneksi database SQLite. Class ini menggunakan pola singleton untuk memastikan hanya ada satu instance dari database yang digunakan di seluruh aplikasi. Di dalamnya terdapat metode untuk:

- **Inisialisasi database (`_initDatabase`)** dengan membuat file database bernama `my_prakdatabase.db`.
- **Pembuatan tabel (`_onCreate`)** saat database pertama kali dibuat. Tabel bernama `my_table` memiliki kolom `id`, `title`, `description`, dan `createdAt`.
- **Operasi CRUD:**
 1. insert: Menambahkan data baru ke tabel.
 2. queryAllRows: Mengambil semua data dari tabel.
 3. update: Memperbarui data yang ada berdasarkan `id`.
 4. delete: Menghapus data berdasarkan `id`.

2. My Database View Class

Class ini bertindak sebagai tampilan utama aplikasi. Menggunakan **StatefulWidget**, class ini berfungsi untuk menampilkan data dan memungkinkan interaksi dengan database. Beberapa fitur utama:

1. **Inisialisasi dan Pembaruan Data:** Metode `_refreshData` mengambil data dari database dan memperbarui tampilan.
2. **Menambahkan Data:** Metode `_addData` menambahkan entri baru ke database dan memperbarui daftar data yang ditampilkan.
3. **Mengedit dan Menghapus Data:**
 - a. `updateData` digunakan untuk memperbarui entri yang dipilih.
 - b. `_deleteData` untuk menghapus entri dari database.
4. **Dialog Interaktif:**
 - a. `_showEditDialog`: Menampilkan dialog untuk mengedit data.
 - b. `_showAddDialog`: Menampilkan dialog untuk menambahkan data baru.

3. Fitur Tambahan

- a. **Log Debugging:** Beberapa log (seperti hasil query atau data yang diambil) ditambahkan untuk membantu debugging.
- b. **Desain UI:** Menggunakan `ListView` untuk menampilkan daftar data, dengan fitur edit dan hapus yang ditambahkan pada setiap item.

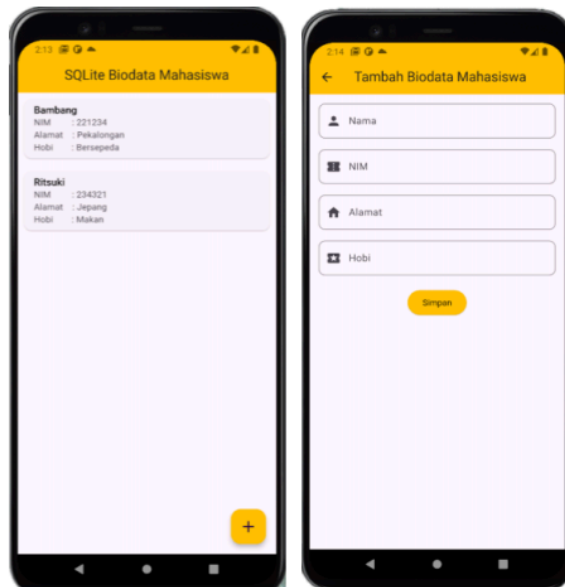
Aplikasi ini merupakan contoh sederhana dari integrasi Flutter dengan SQLite untuk pengelolaan data lokal, yang cocok digunakan dalam aplikasi mobile berskala kecil hingga menengah.

UNGUIDED

1. (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, screenshot output, dan deskripsi program. Kreatifitas menjadi nilai tambah.

1. Source Code

- main.dart

```
1  import 'package:flutter/material.dart';
2  import 'view/main_view.dart';
3
4  void main() {
5    runApp(const MainApp());
6  }
7
8  class MainApp extends StatelessWidget {
9    const MainApp({Key? key}) : super(key: key);
10
11    @override
12    Widget build(BuildContext context) {
13      return MaterialApp(
14        debugShowCheckedModeBanner: false,
15        home: MainView(),
16      );
17    }
18  }
```

- db_helper.dart

```
1  import 'package:path/path.dart';
2  import 'package:sqflite/sqflite.dart';
3
4  class DatabaseHelper {
5    static final DatabaseHelper _instance = DatabaseHelper._internal();
6    static Database? _database;
7
8    factory DatabaseHelper() => _instance;
9
10    DatabaseHelper._internal();
11
12    Future<Database> get database async {
13      if (_database != null) return _database!;
14      _database = await _initDatabase();
15      return _database!;
16    }
17
18    Future<Database> _initDatabase() async {
19      String path = join(await getDatabasesPath(), 'my_database.db');
20      return await openDatabase(
21        path,
22        version: 1,
23        onCreate: _onCreate,
24      );
25    }
26
27    Future<void> _onCreate(Database db, int version) async {
28      await db.execute('''
29        CREATE TABLE mahasiswa(
30          id INTEGER PRIMARY KEY AUTOINCREMENT,
31          nama TEXT,
32          nim TEXT,
33          alamat TEXT,
34          hobi TEXT
35        )
36      ''');
37    }
38  }
```

```

38
39 Future<int> insert(Map<String, dynamic> row) async {
40     Database db = await database;
41     return await db.insert('mahasiswa', row);
42 }
43
44 Future<List<Map<String, dynamic>>> queryAllRows() async {
45     Database db = await database;
46     return await db.query('mahasiswa');
47 }
48
49 Future<int> update(Map<String, dynamic> row) async {
50     Database db = await database;
51     return await db.update(
52         'mahasiswa',
53         row,
54         where: 'id = ?',
55         whereArgs: [row['id']],
56     );
57 }
58
59 Future<int> delete(int id) async {
60     Database db = await database;
61     return await db.delete(
62         'mahasiswa',
63         where: 'id = ?',
64         whereArgs: [id],
65     );
66 }
67 }
68

```

- mahasiswa_model.dart

```

1 class Mahasiswa {
2     int? id;
3     String nama;
4     String nim;
5     String alamat;
6     String hobi;
7
8     Mahasiswa(
9         {this.id,
10         required this.nama,
11         required this.nim,
12         required this.alamat,
13         required this.hobi});
14
15     Map<String, dynamic> toMap() {
16         return {
17             'id': id,
18             'nama': nama,
19             'nim': nim,
20             'alamat': alamat,
21             'hobi': hobi,
22         };
23     }
24 }
25

```

- add_mahasiswa.dart

```
1 import 'package:flutter/material.dart';
2 import '../helper/db_helper.dart';
3
4 class AddMahasiswaView extends StatefulWidget {
5   final Function refreshData;
6
7   const AddMahasiswaView({Key? key, required this.refreshData})
8     : super(key: key);
9
10  @override
11  _AddMahasiswaViewState createState() => _AddMahasiswaViewState();
12 }
13
14 class _AddMahasiswaViewState extends State<AddMahasiswaView> {
15   final _namaController = TextEditingController();
16   final _nimController = TextEditingController();
17   final _alamatController = TextEditingController();
18   final _hobiController = TextEditingController();
19   final dbHelper = DatabaseHelper();
20
21   void addMahasiswa() async {
22     await dbHelper.insert({
23       'nama': _namaController.text,
24       'nim': _nimController.text,
25       'alamat': _alamatController.text,
26       'hobi': _hobiController.text,
27     });
28     widget.refreshData();
29     Navigator.pop(context);
30   }
31
32   // Method to build a text field with icon
33   Widget buildTextField({
34     required TextEditingController controller,
35     required String label,
36     required IconData icon,
37   }) {
38     return TextField(
39       controller: controller,
40       decoration: InputDecoration(
41         labelText: label,
42         prefixIcon: Icon(icon),
43         border: OutlineInputBorder(
44           borderRadius: BorderRadius.circular(8),
45         ),
46       ),
47     );
48   }
49
50   @override
51   Widget build(BuildContext context) {
52     return Scaffold(
53       appBar: AppBar(
54         title: const Text(
55           'Tambah Mahasiswa',
56           style: TextStyle(fontWeight: FontWeight.bold, color: Colors.white),
57         ),
58         centerTitle: true,
59         backgroundColor: const Color.fromARGB(255, 212, 148, 228),
60       ),
61       body: Padding(
62         padding: const EdgeInsets.all(16.0),
63         child: Column(
64           children: [
65             buildTextField(
66               controller: _namaController,
67               label: 'Nama',
68               icon: Icons.person,
69             ),
70             const SizedBox(height: 16),
71             buildTextField(
72               controller: _nimController,
73               label: 'NIM',
74               icon: Icons.book_outlined,
75             ),
76           ],
77         ),
78       ),
79     );
80   }
81 }
```

```

76         const SizedBox(height: 16),
77         buildTextField(
78             controller: _alamatController,
79             label: 'Alamat',
80             icon: Icons.location_on,
81         ),
82         const SizedBox(height: 16),
83         buildTextField(
84             controller: _hobiController,
85             label: 'Hobi',
86             icon: Icons.try_sms_star,
87         ),
88         const SizedBox(height: 20),
89         ElevatedButton(
90             onPressed: addMahasiswa,
91             style: ElevatedButton.styleFrom(
92                 backgroundColor: const Color.fromARGB(255, 212, 148, 228),
93                 padding:
94                     const EdgeInsets.symmetric(vertical: 12, horizontal: 20),
95                 shape: RoundedRectangleBorder(
96                     borderRadius: BorderRadius.circular(8),
97                 ),
98             ),
99             child: const Text(
100                 'Simpan',
101                 style: TextStyle(
102                     fontSize: 16, color: Color.fromARGB(255, 255, 255, 255)),
103             ),
104         ),
105     ],
106 ),
107 ),
108 );
109 }
110 }
111

```

- main.view.dart

```

1 import 'package:biodata_mahasiswa/helper/db_helper.dart';
2 import 'package:biodata_mahasiswa/view/add_mahasiswa.dart';
3 import 'package:flutter/material.dart';
4
5 class MainView extends StatefulWidget {
6     @override
7     _MainViewState createState() => _MainViewState();
8 }
9
10 class _MainViewState extends State<MainView> {
11     final dbHelper = DatabaseHelper();
12     List<Map<String, dynamic>> mahasiswaList = [];
13
14     void refreshData() async {
15         final data = await dbHelper.queryAllRows();
16         setState(() {
17             mahasiswaList = data;
18         });
19     }
20
21     @override
22     void initState() {
23         super.initState();
24         refreshData();
25     }
26
27     @override
28     Widget build(BuildContext context) {
29         return Scaffold(
30             appBar: AppBar(
31                 title: const Text(
32                     'Biodata Mahasiswa',
33                     style: TextStyle(
34                         fontWeight: FontWeight.bold,
35                         color: Color.fromARGB(255, 255, 255, 255)),
36                 ),
37             centerTitle: true,
38             backgroundColor: const Color.fromARGB(255, 212, 148, 228),
39         ),

```

```

40     body: mahasiswalist.isEmpty
41     ? const Center(
42       child: Text(
43         'Belum ada data mahasiswa.',
44         style: TextStyle(fontSize: 16, color: Colors.purple),
45       ),
46     )
47     : ListView.builder(
48       itemCount: mahasiswalist.length,
49       padding: const EdgeInsets.symmetric(vertical: 10),
50       itemBuilder: (context, index) {
51         final item = mahasiswalist[index];
52         return Card(
53           margin:
54             const EdgeInsets.symmetric(vertical: 8, horizontal: 16),
55           shape: RoundedRectangleBorder(
56             borderRadius: BorderRadius.circular(12),
57           ),
58           elevation: 3,
59           child: ListTile(
60             contentPadding: const EdgeInsets.all(16),
61             title: Text(
62               item['nama'],
63               style: const TextStyle(
64                 fontWeight: FontWeight.bold, fontSize: 18),
65             ),
66             subtitle: Column(
67               crossAxisAlignment: CrossAxisAlignment.start,
68               children: [
69                 const SizedBox(height: 8),
70                 Text('NIM: ${item['nim']}'),
71                 Text('Alamat: ${item['alamat']}'),
72                 Text('Hobi: ${item['hobi']}'),
73               ],
74             ),
75           ),
76         );
77       },
78     ),
79     floatingActionButton: FloatingActionButton(
80       backgroundColor: Colors.blueGrey,
81       child: const Icon(Icons.add),
82       onPressed: () {
83         Navigator.push(
84           context,
85           MaterialPageRoute(
86             builder: (context) => AddMahasiswaView(refreshData: refreshData),
87           ),
88         );
89       },
90     ),
91   );
92 }
93 }
94

```

2. Output Hasil

- input data

14:06 94%

← Tambah Mahasiswa

Nama
aorin

NIM
2211104013

Alamat
cilacap

Hobi
mengetik

Simpan

- lihat data

14:05 94%

Biodata Mahasiswa

devrin
NIM: 2211104001
Alamat: banjarnegara
Hobi: menonton film

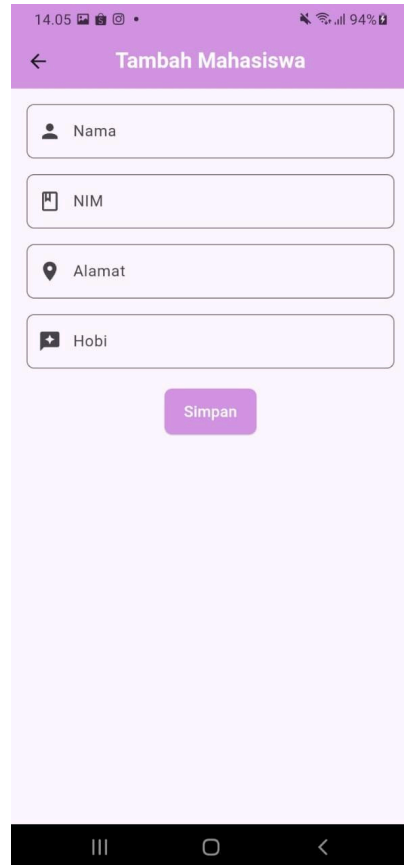
aziz
NIM: 2211104026
Alamat: banjarnegara
Hobi: game

hanifa
NIM: 2211104017
Alamat: malang
Hobi: drakor

atika
NIM: 2211103003
Alamat: majenang
Hobi: makan

+

- bagian input data



3. Deskripsi Program

Program di atas adalah implementasi class `'DatabaseHelper'` dalam Flutter untuk mengelola database SQLite. Class ini menggunakan pola singleton untuk memastikan hanya satu instance dari database yang aktif selama aplikasi berjalan. Database yang dikelola bernama `'my_database.db'`, dan di dalamnya terdapat sebuah tabel bernama `'mahasiswa'`. Tabel ini memiliki lima kolom: `'id'` sebagai primary key yang akan diisi secara otomatis, serta kolom lainnya seperti `'nama'`, `'nim'`, `'alamat'`, dan `'hobi'` untuk menyimpan informasi mahasiswa.

Class ini menyediakan berbagai metode untuk melakukan operasi CRUD (Create, Read, Update, Delete). Metode `'insert'` digunakan untuk menambahkan data baru ke tabel mahasiswa, sementara `'queryAllRows'` memungkinkan pengambilan semua data yang ada di tabel tersebut. Untuk memperbarui data yang sudah ada, disediakan metode `'update'`, yang mengubah entri berdasarkan kolom `'id'`. Selain itu, metode `'delete'` memungkinkan penghapusan entri tertentu dari tabel berdasarkan `'id'` juga.

Dengan desain ini, `'DatabaseHelper'` mempermudah pengelolaan data mahasiswa dalam aplikasi Flutter, cocok untuk aplikasi sederhana yang membutuhkan penyimpanan data lokal.