LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK

MODUL 10

DATA STORAGE (BAGIAN 1)

Disusun Oleh :

YOGI HAFIDH MAULANA / 2211104061

SE06-02

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

A. GUIDED

- **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 dan private 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!;
    }
  }

  // Method untuk menginisialisasi 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,
    );
  }

  // Method untuk membuat 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)
''');
  }

  // 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);
  }

  // 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]);
  }
}
```

- **Db View**

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

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

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

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

  // Method untuk menambahkan data baru ke database
  void _addData() async {
    await dbHelper.insert({
      "title": _titleController.text,
      "description": _descriptionController.text,
    });

    _titleController.clear();
    _descriptionController.clear();
    _refreshData();
  }

  // Method untuk mengupdate data
  void _updateData(int id) async {
    await dbHelper.update({
      "id": id,
      "title": _titleController.text,
      "description": _descriptionController.text,
    });
    _titleController.clear();
    _descriptionController.clear();
    _refreshData();
  }

  // Method untuk menghapus data
  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("Simpan"))
            ],
          );
        });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Praktikum Database - sqdlite"),
        backgroundColor: Colors.blueAccent,
        centerTitle: true,
      ),
      body: Column(
        children: [
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: TextField(
              controller: _titleController,
              decoration: const InputDecoration(labelText: "Judul"),
            ),
          ),
          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(
                          onPressed: () {
                            _showEditDialog(item);
                          },
                          icon: const Icon(Icons.edit)),
                      IconButton(
                          onPressed: () {
                            _deleteData(item["id"]);
                          },
                          icon: const Icon(Icons.delete))
                    ],
                  ),
                );
              },
            ),
          )
        ],
      ),
    );
  }
}
```

Code detail

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

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

  void dispose() {
    _titleController.dispose();
    _descriptionController.dispose();
    super.dispose();
  }
```

```dart
  // Method untuk memperbarui data dari database
  void _refreshData() async {
    final data = await dbHelper.queryAllRows();
    setState(() {
      _dbdata = data;
    });
  }
```

```dart
  // Method untuk menambahkan data baru ke database
  void _addData() async {
    await dbHelper.insert({
      "title": _titleController.text,
      "description": _descriptionController.text,
    });

    _titleController.clear();
    _descriptionController.clear();
    _refreshData();
  }
```

```dart
// Method untuk mengupdate data
void _updateData(int id) async {
  await dbHelper.update({
    "id": id,
    "title": _titleController.text,
    "description": _descriptionController.text,
  });
  _titleController.clear();
  _descriptionController.clear();
  _refreshData();
}
```

```dart
// Method untuk menghapus data
void _deleteData(int id) async {
  await dbHelper.delete(id);
  _refreshData();
}
```

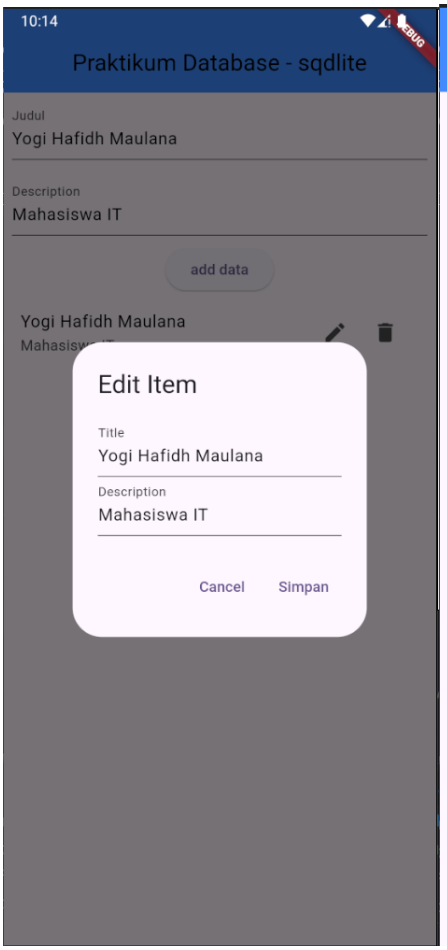
```dart
void _showEditDialog(Map<String, dynamic> item) {
   _titleController.text = item["title"];
   _descriptionController.text = item["description"];

   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("Simpan"))
           ],
         );
       });
 }
```

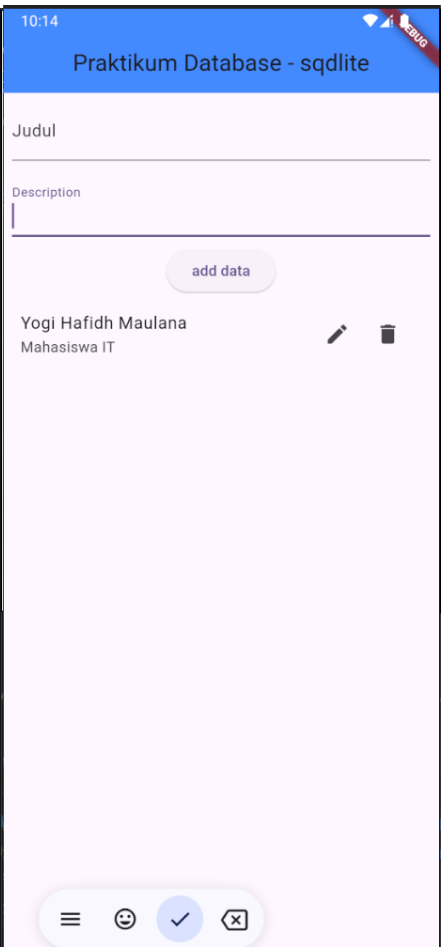
```dart
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Praktikum Database - sqdlite"),
        backgroundColor: Colors.blueAccent,
        centerTitle: true,
      ),
      body: Column(
        children: [
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: TextField(
              controller: _titleController,
              decoration: const InputDecoration(labelText: "Judul"),
            ),
          ),
          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(
                          onPressed: () {
                            _showEditDialog(item);
                          },
                          icon: const Icon(Icons.edit)),
                      IconButton(
                          onPressed: () {
                            _deleteData(item["id"]);
                          },
                          icon: const Icon(Icons.delete))
                    ],
                  ),
                );
              },
            ),
          )
        ],
      ),
    );
  }
}
```

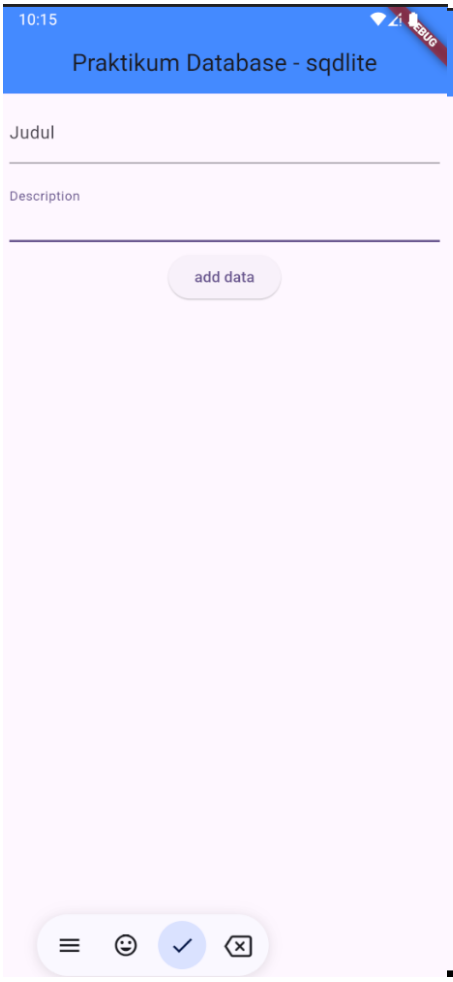
- **Main**

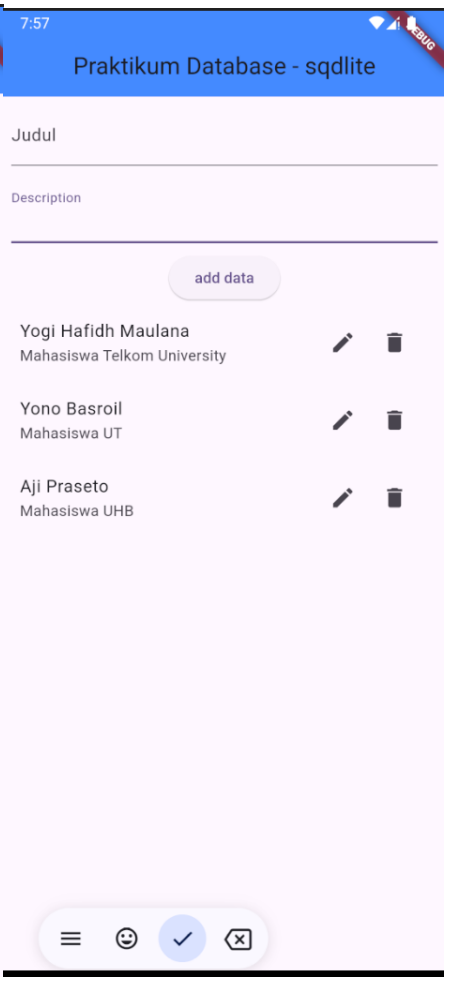
```dart
import 'package:flutter/material.dart';
import 'package:praktikum/view/my_db_view.dart';

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

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

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

Output


10:14
Praktikum Database - sqdlite
Judul
Yogi Hafidh Maulana
Description
Mahasiswa IT
add data
Yogi Hafidh Maulana
Edit Item
Title
Yogi Hafidh Maulana
Description
Mahasiswa IT
Cancel
Simpan


10:14
Praktikum Database - sqdlite
Judul
Description
add data
Yogi Hafidh Maulana
Mahasiswa IT


10:15
Praktikum Database - sqdlite
Judul
Description
add data


7:57
Praktikum Database - sqdlite
Judul
Description
add data
Yogi Hafidh Maulana
Mahasiswa Telkom University
Yono Basroil
Mahasiswa UT
Aji Praseto
Mahasiswa UHB

Deskripsi Program

Pada pertemuan ini saya menggunakan SQLite untuk menyimpan, menampilkan, memperbarui, dan menghapus data berupa judul dan deskripsi. DatabaseHelper berfungsi sebagai pengelola database SQLite dengan metode untuk membuat tabel (_onCreate), menambahkan data (insert), membaca semua data (queryAllRows), memperbarui data (update), dan menghapus data (delete). Ketika aplikasi pertama kali dijalankan, tabel my_table dibuat di database jika belum ada. Data yang disimpan di database diambil menggunakan metode queryAllRows dan ditampilkan di antarmuka pengguna menggunakan widget ListView.builder, yang memungkinkan daftar data diperbarui secara dinamis setiap kali ada perubahan.

Komponen utama aplikasi ini adalah widget MyDatabaseView, yang mengelola antarmuka pengguna untuk memasukkan, mengedit, atau menghapus data. Pengguna dapat mengetik judul dan deskripsi di dalam TextField yang tersedia, lalu menekan tombol "Add Data" untuk menyimpan data ke database menggunakan metode insert. Data yang ditampilkan di daftar dapat diedit dengan menekan tombol edit (ikon pensil), yang membuka dialog untuk memperbarui data menggunakan metode update. Untuk menghapus data, pengguna dapat menekan tombol hapus (ikon tempat sampah), yang memanggil metode delete. Semua perubahan data di database secara otomatis diperbarui di layar dengan memanggil metode _refreshData, yang mengambil ulang data dari database dan memperbarui variabel _dbdata yang digunakan oleh ListView.builder. Antarmuka sederhana ini memungkinkan pengelolaan data secara efisien melalui database lokal SQLite.

B. Unguided

1. Soal 1

Code Db Helper

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

class DBHelper {
  static final DBHelper _instance = DBHelper._init();
  static Database? _database;

  DBHelper._init();

  factory DBHelper() => _instance;

  Future<Database> get database async {
    if (_database != null) return _database!;
    _database = await _initDB('students.db');
    return _database!;
  }

  Future<Database> _initDB(String fileName) async {
    final dbDir =
        Directory.systemTemp;
    final dbPath = path.join(dbDir.path, fileName);

    return await openDatabase(dbPath, version: 1, onCreate: _createDB);
  }

  Future _createDB(Database db, int version) async {
    const sql = '''
      CREATE TABLE students (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        name TEXT,
        nim TEXT,
        address TEXT,
        hobby TEXT
      )
    ''';
    await db.execute(sql);
  }

  Future<int> insertStudent(Map<String, dynamic> data) async {
    final db = await database;
    return await db.insert('students', data);
  }

  Future<List<Map<String, dynamic>>> getStudents() async {
    final db = await database;
    return await db.query('students');
  }
}
```

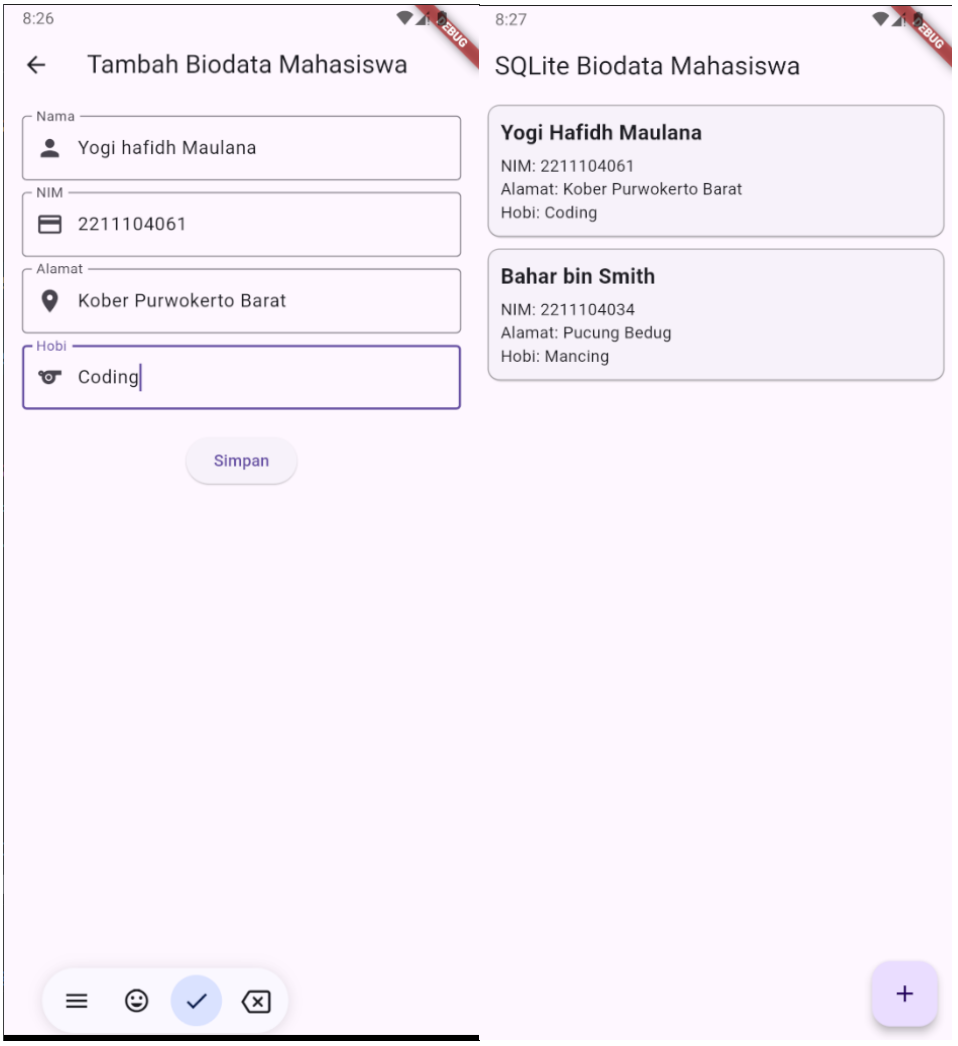
Code Student Form

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

class StudentForm extends StatefulWidget {
  @override
  _StudentFormState createState() => _StudentFormState();
}

class _StudentFormState extends State<StudentForm> {
  final _formKey = GlobalKey<FormState>();
  final _nameController = TextEditingController();
  final _nimController = TextEditingController();
  final _addressController = TextEditingController();
  final _hobbyController = TextEditingController();
  final DBHelper _dbHelper = DBHelper();

  void _saveStudent() async {
    if (_formKey.currentState!.validate()) {
      await _dbHelper.insertStudent({
        'name': _nameController.text,
        'nim': _nimController.text,
        'address': _addressController.text,
        'hobby': _hobbyController.text,
      });
      Navigator.pop(context, true);
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Tambah Biodata Mahasiswa')),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Form(
          key: _formKey,
          child: Column(
            children: [
              TextFormField(
                controller: _nameController,
                decoration: InputDecoration(
                  prefixIcon: Icon(Icons.person),
                  labelText: 'Nama',
                  border: OutlineInputBorder(),
                ),
                validator: (value) =>
                    value!.isEmpty ? 'Nama tidak boleh kosong' : null,
              ),
              SizedBox(height: 10),
              TextFormField(
                controller: _nimController,
                decoration: InputDecoration(
                  prefixIcon: Icon(Icons.credit_card),
                  labelText: 'NIM',
                  border: OutlineInputBorder(),
                ),
                validator: (value) =>
                    value!.isEmpty ? 'NIM tidak boleh kosong' : null,
              ),
              SizedBox(height: 10),
              TextFormField(
                controller: _addressController,
                decoration: InputDecoration(
                  prefixIcon: Icon(Icons.location_on),
                  labelText: 'Alamat',
                  border: OutlineInputBorder(),
                ),
                validator: (value) =>
                    value!.isEmpty ? 'Alamat tidak boleh kosong' : null,
              ),
              SizedBox(height: 10),
              TextFormField(
                controller: _hobbyController,
                decoration: InputDecoration(
                  prefixIcon: Icon(Icons.sports),
                  labelText: 'Hobi',
                  border: OutlineInputBorder(),
                ),
                validator: (value) =>
                    value!.isEmpty ? 'Hobi tidak boleh kosong' : null,
              ),
              SizedBox(height: 20),
              ElevatedButton(
                onPressed: _saveStudent,
                child: Text('Simpan'),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

Code Main

```dart
import 'package:flutter/material.dart';
import 'package:praktikum/helper/db_helper.dart';
import 'package:praktikum/view/student_form.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'SQLite Biodata Mahasiswa',
      theme: ThemeData(primarySwatch: Colors.orange),
      home: HomePage(),
    );
  }
}

class HomePage extends StatefulWidget {
  @override
  _HomePageState createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  final DBHelper _dbHelper = DBHelper();
  List<Map<String, dynamic>> _students = [];

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

  Future<void> _fetchStudents() async {
    final data = await _dbHelper.getStudents();
    setState(() {
      _students = data;
    });
  }

  void _navigateToForm() async {
    final result = await Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => StudentForm()),
    );
    if (result == true) {
      _fetchStudents();
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('SQLite Biodata Mahasiswa')),
      body: ListView.builder(
        itemCount: _students.length,
        itemBuilder: (context, index) {
          final student = _students[index];
          return Card(
            margin: const EdgeInsets.symmetric(horizontal: 10, vertical: 5),
            child: Container(
              padding: const EdgeInsets.all(10),
              decoration: BoxDecoration(
                  border: Border.all(
                    width: 1.0,
                    color: const Color.fromRGBO(0, 0, 0, 0.301),
                  ),
                  borderRadius: BorderRadius.circular(10)),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    student['name'],
                    style: const TextStyle(
                        fontSize: 18, fontWeight: FontWeight.bold),
                  ),
                  const SizedBox(height: 5),
                  Text('NIM: ${student['nim']}'),
                  Text('Alamat: ${student['address']}'),
                  Text('Hobi: ${student['hobby']}'),
                ],
              ),
            ),
          );
        },
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _navigateToForm,
        child: const Icon(Icons.add),
      ),
    );
  }
}
```

Output



Deskripsi Program

Program ini bekerja dengan menggunakan SQLite sebagai database lokal untuk menyimpan data mahasiswa, yang terdiri dari kolom name, nim, address, dan hobby. Saat aplikasi pertama kali dijalankan, database diperiksa apakah sudah ada; jika belum, tabel students akan dibuat menggunakan perintah SQL melalui fungsi _createDB. Data dari tabel ini kemudian diambil melalui fungsi getStudents() dan ditampilkan dalam daftar di halaman utama menggunakan widget ListView.builder. Setiap data mahasiswa ditampilkan menggunakan widget Card, lengkap dengan informasi yang diambil dari database.

Ketika tombol tambah (+) ditekan, aplikasi membawa pengguna ke halaman formulir input menggunakan navigasi. Di halaman formulir, pengguna dapat memasukkan data mahasiswa, seperti nama, NIM, alamat, dan hobi. Saat tombol simpan ditekan, aplikasi memvalidasi data input, kemudian data disimpan ke tabel students menggunakan fungsi insertStudent(). Setelah berhasil disimpan, pengguna kembali diarahkan ke halaman utama, dan daftar mahasiswa otomatis diperbarui dengan memuat ulang data dari database melalui fungsi _fetchStudents(). Proses ini memastikan data selalu sinkron antara database dan tampilan aplikasi. Flutter menangani antarmuka dengan widget seperti Card, Container, dan TextFormField untuk memberikan pengalaman pengguna yang responsif dan menarik.