LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK

MODUL 10

DATA STORAGE (BAGIAN 1)

Disusun Oleh :

Dzikri Naufal Wisnu Pravida/2211104063

SE06-02

Asisten Praktikum :

Muhammad Faza Zulian

Aisyah Hasna Aulia

Dosen Pengampu :

Yudha Islami Sulistya

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

PRAKTIKUM

A. GUIDED

1. PENGENALAN SQLite

SQLite adalah database relasional yang digunakan untuk penyimpanan data secara offline dalam aplikasi mobile, tepatnya pada cache memory aplikasi. SQLite mendukung operasi CRUD (create, read, update, dan delete) dan memiliki struktur yang serupa dengan SQL pada umumnya, dengan variabel dan tipe data yang tidak jauh berbeda.

2. SQL Helper Dasar

SQL Helper dalam konteks Flutter merujuk pada penggunaan paket seperti sqflite untuk mengelola database SQLite. Ini adalah sebuah class yang membantu membuat metode-metode terkait perubahan data, memungkinkan pengembang melakukan operasi CRUD pada database SQLite dengan mudah melalui method-method seperti insert, query, update, dan delete.

3. Hasil Praktikum

Source Code :

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

```

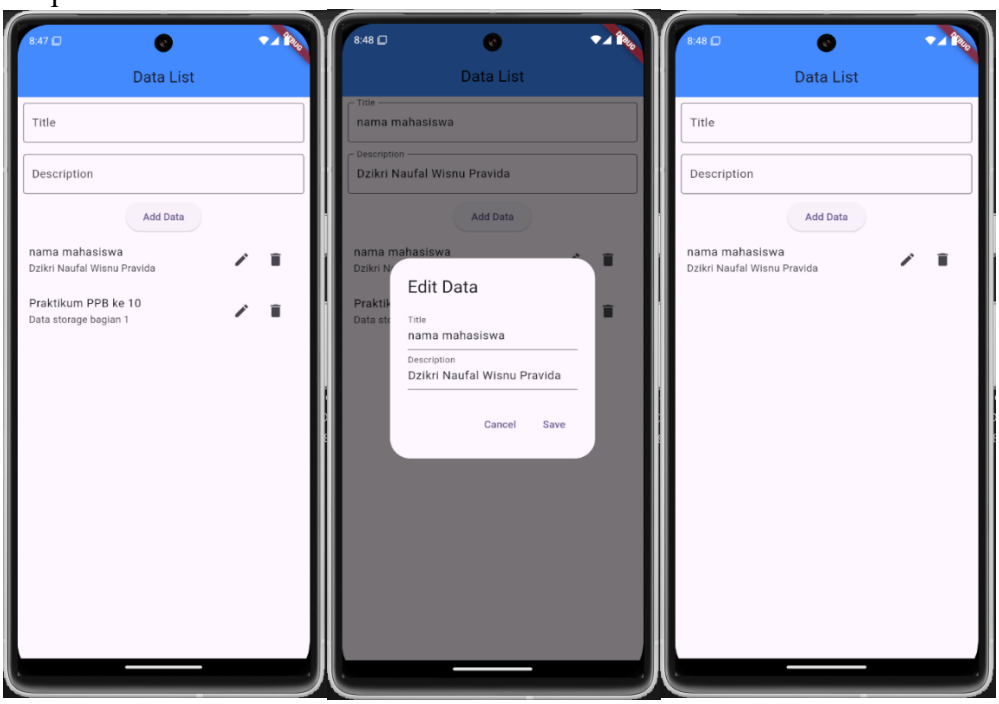
Db helper

```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;
  }

  //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(), '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)
''');
  }

  // method untuk memasukan data ke tabel
  Future<int> insert(Map<String, dynamic> row) async {
    Database db = await database;
    return await db.insert('my_table', row);
  }

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

  //method 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]);
  }

  // method untuk menghapus data
  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:prak_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();
  }

  // method to refresh data from the database
  void  refreshData() async {
    final data = await dbHelper.queryAllRows();
    setState(() {
      _dbData = data;
    });
  }

  // method to add data to the database
  void  addData() async {
    await dbHelper.insert({
      'title': _titleController.text,
      'description': _descriptionController.text,
    });
     titleController.clear();
    _descriptionController.clear();
    _refreshData();
  }

  // method to update data in the database
  void _updateData(int id) async {
    await dbHelper.update({
      'id': id,
      'title': _titleController.text,
      'description': _descriptionController.text,
    });
    _titleController.clear();
    _descriptionController.clear();
    _refreshData();
  }

  // method to delete data from the database
  void _deleteData(int id) async {
    await dbHelper.delete(id);
    _refreshData();
  }

  // show dialog to edit data
  void _showEditDialog(Map<String, dynamic> item) {
    _titleController.text = item['title'];
    _descriptionController.text = item['description'];
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: const Text('Edit Data'),
          content: Column(
            mainAxisSize: MainAxisSize.min,
            children: [
              TextField(
                controller: _titleController,
                decoration: const InputDecoration(labelText: 'Title'),
              ),
              TextField(
                controller: _descriptionController,
                decoration: const InputDecoration(labelText: 'Description'),
              ),
            ],
          ),
          actions: [
            TextButton(
              child: const Text('Cancel'),
              onPressed: () {
                Navigator.of(context).pop();
              },
            ),
            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('Data List'),
        backgroundColor: Colors.blueAccent,
        centerTitle: true,
      ),
      body: Column(
        children: [
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: TextField(
              controller:  titleController,
              decoration: const InputDecoration(
                labelText: 'Title',
                border: OutlineInputBorder(),
              ),
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: TextField(
              controller: _descriptionController,
              decoration: const InputDecoration(
                labelText: 'Description',
                border: OutlineInputBorder(),
              ),
            ),
          ),
          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']),
                      ),
                    ],
                  ),
                );
              },
            ),
          ],
        ],
      ),
    );
  }
}
```

Output :



B. UNGUIDED

1. 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 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).

Sourcecode

Instalasi package ke dalam pubspec.yaml

dependencies:

  sqflite: ^2.0.0+3

  path: ^1.8.0

main.dart

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter SQLite Demo',
      theme: ThemeData(
        primarySwatch: Colors.amber,
      ),
      home: StudentListScreen(),
    );
  }
}

```

Db 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() => _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(), 'students.db');
    return await openDatabase(
      path,
      version: 1,
      onCreate: (db, version) async {
        await db.execute('''
          CREATE TABLE students (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT,
            nim TEXT,
            address TEXT,
            hobby TEXT
          )
        ''');
      },
    );
  }

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

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

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

Student list

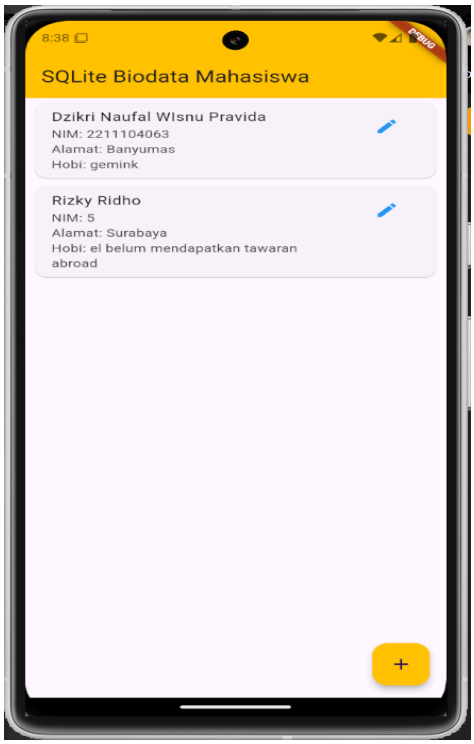
```dart
import 'package:flutter/material.dart';
import 'db_helper.dart';
import 'form_screen.dart';

class StudentListScreen extends StatefulWidget {
  @override
  _StudentListScreenState createState() => _StudentListScreenState();
}

class _StudentListScreenState extends State<StudentListScreen> {
  final DatabaseHelper _dbHelper = DatabaseHelper();
  List<Map<String, dynamic>> _students = [];

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

  void _loadStudents() async {
    final data = await _dbHelper.getStudents();
    setState(() {
      _students = data;
    });
  }

  void _navigateToAddStudent({Map<String, dynamic>? student}) async {
    await Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => AddStudentFormScreen(student: student),
      ),
    );
    _loadStudents();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('SQLite Biodata Mahasiswa'),
        backgroundColor: Colors.amber,
      ),
      body: ListView.builder(
        itemCount: _students.length,
        itemBuilder: (context, index) {
          final student = _students[index];
          return Card(
            margin: const EdgeInsets.symmetric(vertical: 5, horizontal: 10),
            child: ListTile(
              title: Text(student['name']),
              subtitle: Text(
                'NIM: ${student['nim']}\nAlamat: ${student['address']}\nHobi: ${student['hobby']}',
              ),
              isThreeLine: true,
              trailing: Row(
                mainAxisSize: MainAxisSize.min,
                children: [
                  IconButton(
                    icon: Icon(Icons.edit, color: Colors.blue),
                    onPressed: () => _navigateToAddStudent(student: student),
                  ),
                ],
              ),
            ),
          );
        },
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () => _navigateToAddStudent(),
        child: Icon(Icons.add),
        backgroundColor: Colors.amber,
      ),
    );
  }
}
```

Form screen

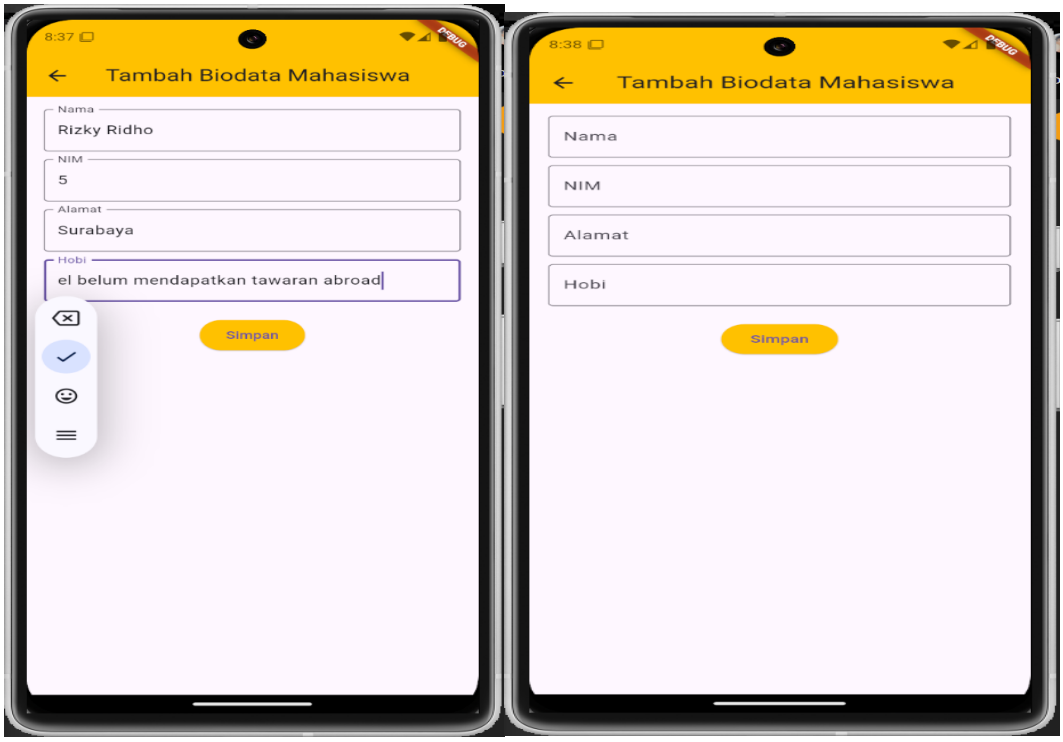
```dart
import 'package:flutter/material.dart';
import 'db_helper.dart';

class AddStudentFormScreen extends StatefulWidget {
  final Map<String, dynamic>? student; // Optional parameter for editing

  const AddStudentFormScreen({Key? key, this.student}) : super(key: key);

  @override
  _AddStudentFormScreenState createState() => _AddStudentFormScreenState();
}

class _AddStudentFormScreenState extends State<AddStudentFormScreen> {
  final DatabaseHelper _dbHelper = DatabaseHelper();
  final _nameController = TextEditingController();
  final _nimController = TextEditingController();
  final _addressController = TextEditingController();
  final _hobbyController = TextEditingController();

  @override
  void initState() {
    super.initState();
    // If editing, pre-fill the fields
    if (widget.student != null) {
      _nameController.text = widget.student!['name'];
      _nimController.text = widget.student!['nim'];
      _addressController.text = widget.student!['address'];
      _hobbyController.text = widget.student!['hobby'];
    }
  }

  void _saveStudent() async {
    if (widget.student == null) {
      // Insert new student
      await _dbHelper.insertStudent({
        'name': _nameController.text,
        'nim': _nimController.text,
        'address': _addressController.text,
        'hobby': _hobbyController.text,
      });
    } else {
      // Update existing student
      await _dbHelper.updateStudent({
        'id': widget.student!['id'],
        'name': _nameController.text,
        'nim': _nimController.text,
        'address': _addressController.text,
        'hobby': _hobbyController.text,
      });
    }
    Navigator.pop(context);
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.student == null
            ? 'Tambah Biodata Mahasiswa'
            : 'Edit Biodata Mahasiswa'),
        backgroundColor: Colors.amber,
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            TextField(
              controller: _nameController,
              decoration: InputDecoration(
                labelText: 'Nama',
                border: OutlineInputBorder(),
              ),
            ),
            SizedBox(height: 10),
            TextField(
              controller: _nimController,
              decoration: InputDecoration(
                labelText: 'NIM',
                border: OutlineInputBorder(),
              ),
            ),
            SizedBox(height: 10),
            TextField(
              controller: _addressController,
              decoration: InputDecoration(
                labelText: 'Alamat',
                border: OutlineInputBorder(),
              ),
            ),
            SizedBox(height: 10),
            TextField(
              controller: _hobbyController,
              decoration: InputDecoration(
                labelText: 'Hobi',
                border: OutlineInputBorder(),
              ),
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: _saveStudent,
              child: Text(widget.student == null ? 'Simpan' : 'Update'),
              style: ElevatedButton.styleFrom(backgroundColor: Colors.amber),
            ),
          ],
        ),
      ),
    );
  }
}
```

Screenshoot Output

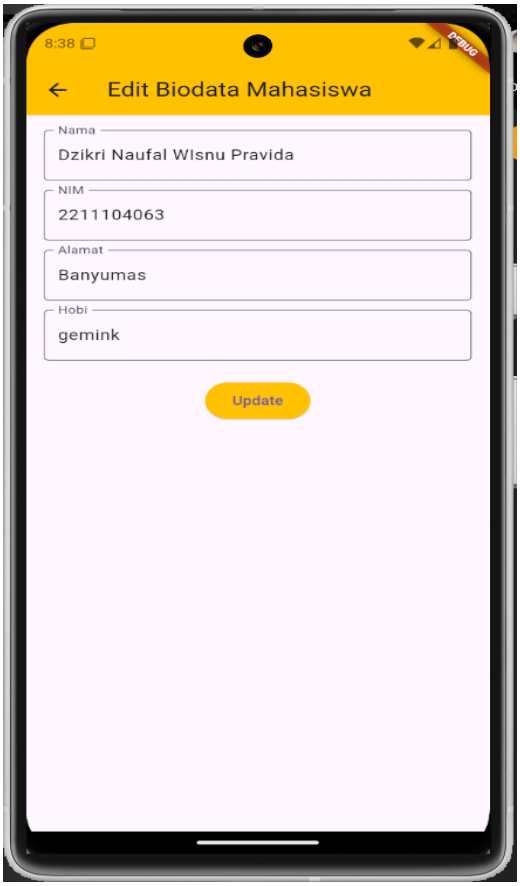
Tampilan utama



Tampilan menambah data



Tampilan update data



Deskripsi Program

Program ini mengimplementasikan dari data storage SQLite untuk menyimpan dan mengedit. Data yang disimpan mencakup nama, NIM, alamat, dan hobi mahasiswa, yang dapat dimasukkan melalui form input yang dilengkapi dengan Outline Border untuk tampilan yang lebih bersih. Setelah data ditambahkan, daftar mahasiswa akan ditampilkan di halaman utama menggunakan ListView, di mana setiap item daftar menampilkan informasi lengkap tentang mahasiswa tersebut. Pengguna dapat membuka form input dengan mengklik tombol tambah atau mengedit data yang ada dengan memilih tombol edit pada masing-masing item dalam daftar.