PRAKTIKUM PEMROGRAMAN PERANGKAT BERGERAK
TUGAS GUIDED & UNGUIDED

MODUL X
DATA STORAGE (BAGIAN I)

Disusun Oleh :

Rizky Hanifa Afania / 2211104017

SE-06-01

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 offline untuk aplikasi mobile, khususnya pada local storage seperti cache memory aplikasi. SQLite mendukung operasi CRUD (create, read, update, dan delete), yang merupakan komponen penting dalam manajemen data. Struktur database di SQLite mirip dengan SQL pada umumnya, termasuk dalam hal variabel dan tipe data yang digunakan. Informasi dasar mengenai SQL dapat diakses melalui tautan berikut.

2. SQL Helper Dasar

Dalam Flutter, SQL Helper biasanya mengacu pada pemanfaatan paket seperti sqflite untuk mengelola database SQLite. SQL Helper berperan sebagai kelas yang berisi metode-metode yang digunakan untuk memproses perubahan data. Sqflite sendiri adalah plugin Flutter yang mendukung pelaksanaan operasi CRUD (Create, Read, Update, Delete) pada database SQLite.

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

1. Tambahkan plugin sqflite dan path ke file pubspec.yaml.

```
30  dependencies:
31    flutter:
32      sdk: flutter
33    cupertino_icons: ^1.0.8
34    sqflite: ^2.4.1
35    path: ^1.9.0
```

2. Buat class baru bernama DatabseHelper untk mengelola database dan import package sqflite dan path di file db_helper.dart

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

//Kelas databse untuk mengelola database
class DatabaseHelper {
  static final DatabaseHelper _instance =
DatabaseHelper._internal();
  static Database? _database;
}
```

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

```
// factory constructor untuk mengembalikan instance singletonce
  factory DatabaseHelper() {
    return _instance;
  }

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

4. Buat Getter untuk database

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

5. Inisialisasi database dengan nama database yang kita mau

```
  //Inisialisasi database
  Future<Database> _initDatabase() async {
    // mendapatkan path untuk database
    String path = join(await getDatabasesPath(), 'unguided.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 db dengan record dan value id, title,
description
  Future<void> _onCreate(Database db, int version) async {
```

```
    await db.execute('''
CREATE TABLE my_table(
id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
title TEXT,
description TEXT,
createdAt TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP)
''');
  }
```

7. Buat metode untuk memasukkan data ke dalam tabel.

```
// metode untuk mengambil semua data dari tabel
  Future<int> insert(Map<String, dynamic> row) async {
    Database db = await database;
    return await db.insert('my_table', row);
  }
```

8. Lalu, metode untuk mengambil semua data dari tabel

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

9. Buat metode untuk memperbarui data dalam tabel.

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

```
whereArgs: [id]);
  }
```

a. Read

Pada package sqflite, kita dapat menggunakan metode query() untuk membaca data dari database. Dengan sqflite di Flutter, berbagai perintah seperti where, groupBy, orderBy, dan having dapat digunakan dalam membuat query. Selain itu, kita memiliki fleksibilitas untuk membaca satu data maupun banyak data sekaligus. Berikut adalah contoh kode untuk operasi read menggunakan sqflite:

Membaca semua data

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

Membaca satu data melalui id

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

Source Code Praktikum:

main.dart

```
import 'package:flutter/material.dart';
import 'package:pert_10/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: 'Database Storage',
      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';

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

// factory constructor untuk mengembalikan instance singletonce
  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(), 'unguided.db');
```

```
    // membuka database
    return await openDatabase(
      path,
      version: 1,
      onCreate: _onCreate,
    );
  }

  //Membuat tabel db dengan record dan value id, title, 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 mengambil semua data dari tabel
  Future<int> insert(Map<String, dynamic> row) async {
    Database db = await database;
    return await db.insert('my_table', row);
  }

//  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]);
  }

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

```
  }
}
```

my_view.dart

```
import 'package:flutter/material.dart';
import 'package:pert_10/helper/db_helper.dart';

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

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

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

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

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

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

  void _addData() async {
```

```
    if (_titleController.text.isEmpty || _descriptionController.text.isEmpty)
{
      _showSnackbar('Title and Description cannot be empty!');
      return;
    }
    await dbHelper.insert({
      'title': _titleController.text,
      'description': _descriptionController.text,
    });
    _titleController.clear();
    _descriptionController.clear();
    _refreshData();
  }

  void _updateData(int id) async {
    if (_titleController.text.isEmpty || _descriptionController.text.isEmpty)
{
      _showSnackbar('Title and Description cannot be empty!');
      return;
    }
    await dbHelper.update({
      'id': id,
      'title': _titleController.text,
      'description': _descriptionController.text,
    });
    _titleController.clear();
    _descriptionController.clear();
    _refreshData();
  }

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

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

    showDialog(
      context: context,
      builder: (context) {
        return AlertDialog(
```

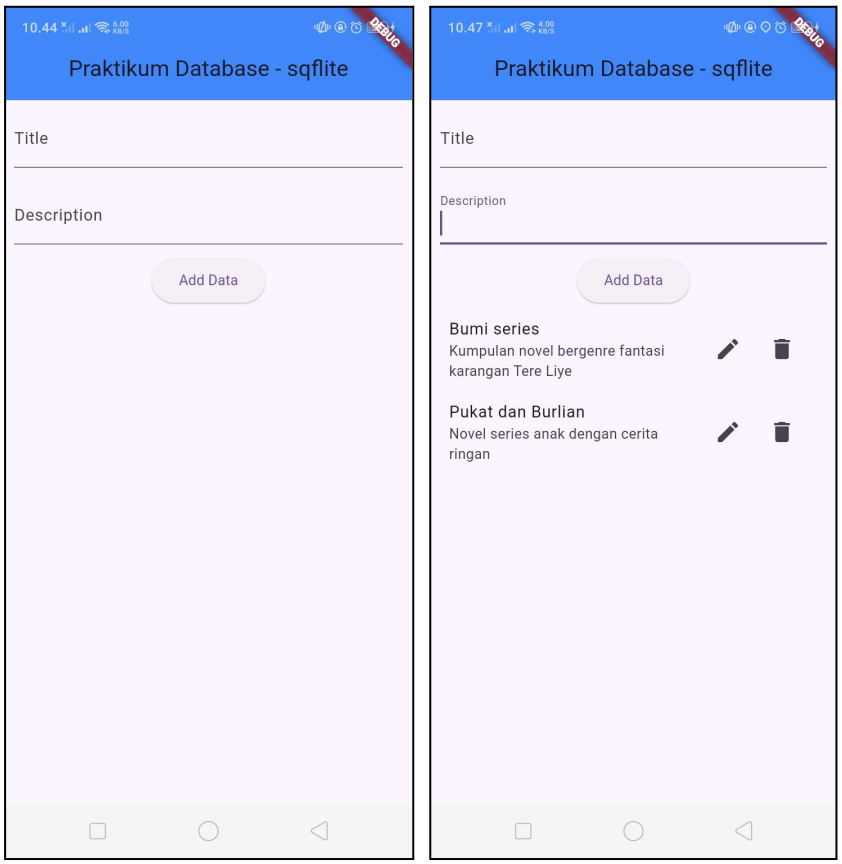
```
        title: const Text('Edit Item'),
        content: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            TextField(
              controller: _titleController,
              decoration: const InputDecoration(labelText: 'Title'),
            ),
            TextField(
              controller: _descriptionController,
              decoration: const InputDecoration(labelText: 'Description'),
            ),
          ],
        ),
        actions: [
          TextButton(
            onPressed: () {
              Navigator.of(context).pop();
            },
            child: const Text('Cancel'),
          ),
          TextButton(
            onPressed: () {
              _updateData(item['id']);
              Navigator.of(context).pop();
            },
            child: const Text('Save'),
          ),
        ],
      );
    },
  );
}

void _showSnackbar(String message) {
  ScaffoldMessenger.of(context)
      .showSnackBar(SnackBar(content: Text(message)));
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
```

```
      title: const Text('Praktikum Database - sqflite'),
      backgroundColor: Colors.blueAccent,
      centerTitle: true,
    ),
    body: Column(
      children: [
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: TextField(
            controller: _titleController,
            decoration: const InputDecoration(labelText: 'Title'),
          ),
        ),
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: TextField(
            controller: _descriptionController,
            decoration: const InputDecoration(labelText: 'Description'),
          ),
        ),
        ElevatedButton(
          onPressed: _addData,
          child: const Text('Add Data'),
        ),
        Expanded(
          child: ListView.builder(
            itemCount: _dbData.length,
            itemBuilder: (context, index) {
              final item = _dbData[index];
              return ListTile(
                title: Text(item['title']),
                subtitle: Text(item['description']),
                trailing: Row(
                  mainAxisSize: MainAxisSize.min,
                  children: [
                    IconButton(
                      icon: const Icon(Icons.edit),
                      onPressed: () {
                        _showEditDialog(item);
                      },
                    ),
                    IconButton(
                      icon: const Icon(Icons.delete),
```

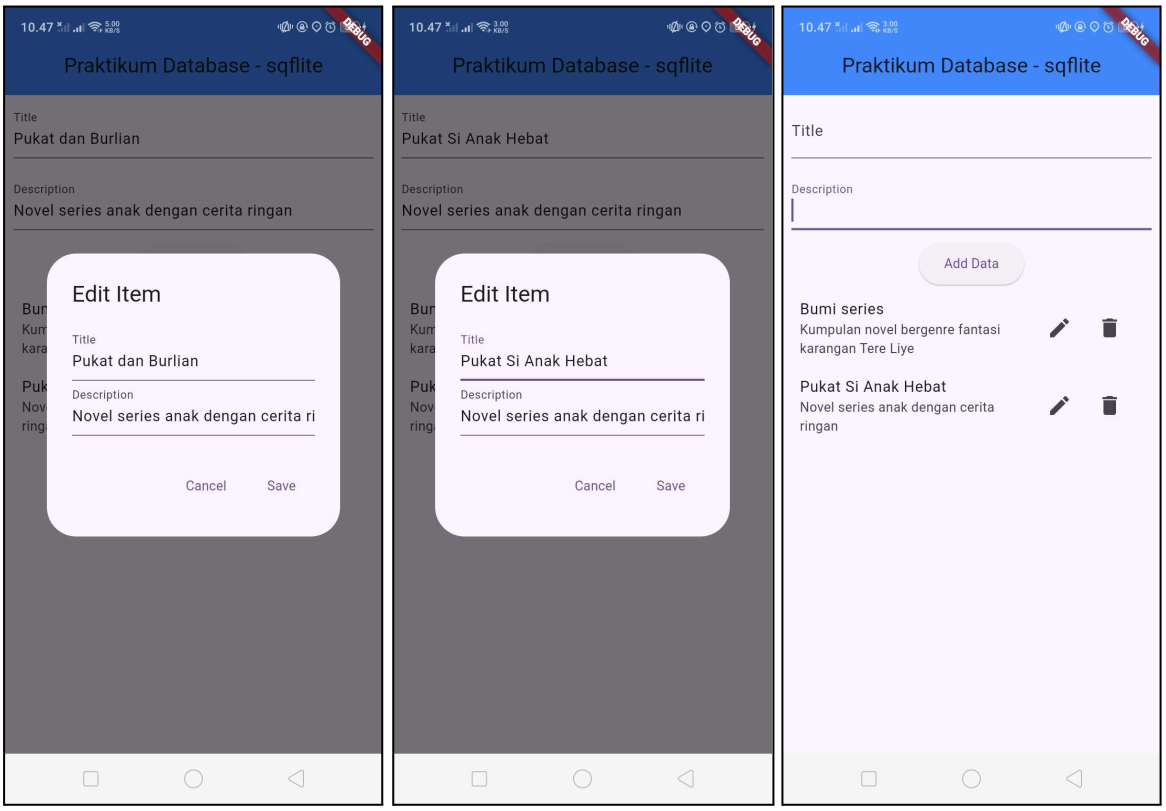
```
                        onPressed: () => _deleteData(item['id']),
                      ),
                    ],
                  ),
                );
              },
            ),
          ),
        ],
      ),
    );
  }
}
```

Output:

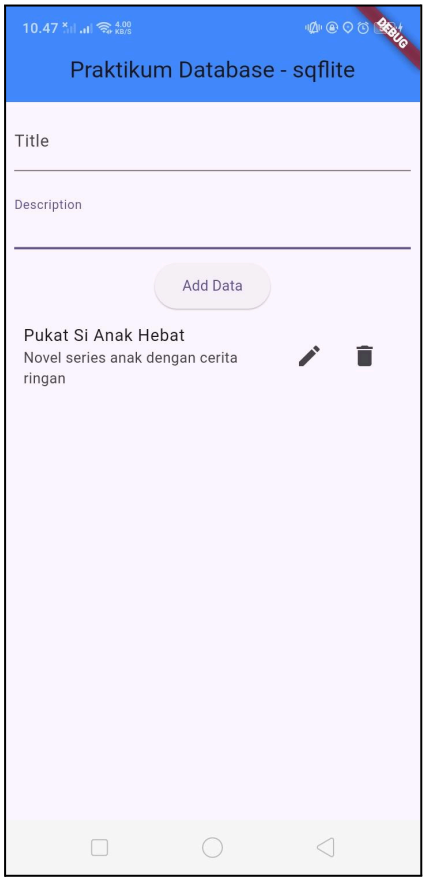
Berikut adalah alaman utama, lalu user bisa mengisi data pada “Title” dan “Description”.



Jika ingin mengedit user bisa klik icon “pensil” lalu bisa merubah data, misalkan akan merubah title, dari “Pukat dan Burlian” diubah menjadi “Pukat si Anak Hebat” lalu klik tombol Save untuk menyimpan perubahan.



Jika ingin menghapus data bisa dengan menekan ikon sampah, lalu data akan terhapus, misalkan akan menghapus “Bumi Serie” berikut adalah tampilan setelah data dihapus.



UNGUIDED

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

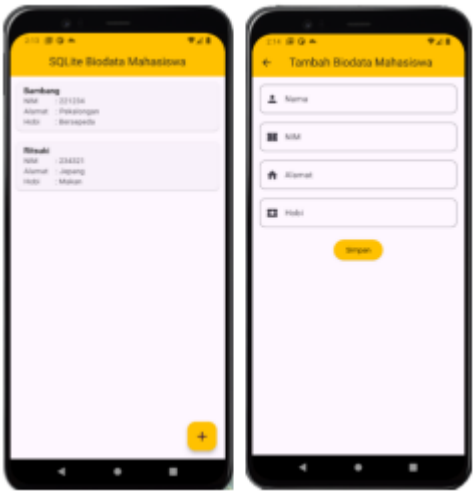
Alur Aplikasi:

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

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

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

d. Contoh output:



Note: Jangan lupa sertakan source code, screenshoot output, dan deskripsi program.

Kreatifitas menjadi nilai tambah

Source Code:

a. Main.dart

```dart
import 'package:belajar_sqlite/view/main_view.dart';
import 'package:belajar_sqlite/view/my_db_view.dart';
import 'package:flutter/material.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: 'Unguided Pertemuan 10',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: MainView(),
    );
  }
}

```

b. db_helper.dart

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

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

// factory constructor untuk mengembalikan instance singletonce
  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!;
    }
  }
```

```dart
//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: 2,
    onCreate: _onCreate,
    // Tambahkan ini untuk menangani upgrade
    onUpgrade: _onUpgrade,
  );
}

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

Future<void> _onUpgrade(Database db, int oldVersion, int newVersion) async {
  if (oldVersion < 2) {
    // Tambahkan kolom alamat dan hobi jika belum ada
    await db.execute('ALTER TABLE my_table ADD COLUMN alamat TEXT');
    await db.execute('ALTER TABLE my_table ADD COLUMN hobi TEXT');
  }
}

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

//  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]);
}

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

c. my_db_view.dart

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

class MyDatabaseView extends StatefulWidget {
  final Map<String, dynamic>? item; // Menambahkan parameter item untuk
edit

  const MyDatabaseView({super.key, this.item});

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

class _MyDatabaseViewState extends State<MyDatabaseView> {
  final DatabaseHelper dbHelper = DatabaseHelper();
  final TextEditingController _namaController =
TextEditingController();
  final TextEditingController _nimController = TextEditingController();
  final TextEditingController _alamatController =
TextEditingController();
  final TextEditingController _hobiController =
TextEditingController();

  @override
  void initState() {
    super.initState();
    if (widget.item != null) {
      // Jika data ada (edit mode), masukkan data ke dalam controller
      _namaController.text = widget.item?['nama'] ?? '';
      _nimController.text = widget.item?['nim'] ?? '';
      _alamatController.text = widget.item?['alamat'] ?? '';
      _hobiController.text = widget.item?['hobi'] ?? '';
    }
  }

  @override
  void dispose() {
    _namaController.dispose();
    _nimController.dispose();
    _alamatController.dispose();
    _hobiController.dispose();
    super.dispose();
```

```
}

void _saveData() async {
  if (_namaController.text.isEmpty ||
      _nimController.text.isEmpty ||
      _alamatController.text.isEmpty ||
      _hobiController.text.isEmpty) {
    _showSnackbar('Tidak boleh kosong!');
    return;
  }

  if (widget.item == null) {
    // Jika item null berarti tambah data
    await dbHelper.insert({
      'nama': _namaController.text,
      'nim': _nimController.text,
      'alamat': _alamatController.text,
      'hobi': _hobiController.text,
    });
  } else {
    // Jika item tidak null berarti edit data
    await dbHelper.update({
      'id': widget.item?['id'],
      'nama': _namaController.text,
      'nim': _nimController.text,
      'alamat': _alamatController.text,
      'hobi': _hobiController.text,
    });
  }

  Navigator.pop(context);
}

void _showSnackbar(String message) {
  ScaffoldMessenger.of(context)
      .showSnackBar(SnackBar(content: Text(message)));
}

//Mendefinisikan fungsi buildText
Widget buildTextField({
  required TextEditingController controller,
  required String label,
  required IconData icon,
```

```
}) {
  return TextField(
    controller: controller,
    decoration: InputDecoration(
      labelText: label,
      prefixIcon: Icon(icon),
      border: OutlineInputBorder(),
    ),
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(widget.item == null
          ? 'Tambah Biodata Mahasiswa'
          : 'Edit Biodata Mahasiswa'),
      backgroundColor: Colors.amber,
      centerTitle: true,
    ),
    body: Padding(
      padding: const EdgeInsets.all(8.0),
      child: Column(
        children: [
          buildTextField(
            controller: _namaController,
            label: 'Nama',
            icon: Icons.person,
          ),
          const SizedBox(height: 16),
          buildTextField(
            controller: _nimController,
            label: 'NIM',
            icon: Icons.confirmation_num,
          ),
          const SizedBox(height: 16),
          buildTextField(
            controller: _alamatController,
            label: 'Alamat',
            icon: Icons.home,
          ),
          const SizedBox(height: 16),
```

```
            buildTextField(
              controller: _hobiController,
              label: 'Hobi',
              icon: Icons.local_activity,
            ),
            const SizedBox(height: 16),
            ElevatedButton(
              onPressed: _saveData,
              child: const Text(
                'Simpan',
                style: TextStyle(fontSize: 16),
              ),
              style: ElevatedButton.styleFrom(backgroundColor:
Colors.amber),
            ),
          ],
        ),
      ),
    );
  }
}
```

d. main_view.dart

```
import 'package:flutter/material.dart';
import 'package:belajar_sqlite/helper/db_helper.dart';
import 'package:belajar_sqlite/view/my_db_view.dart';

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

  @override
  State<MainView> createState() => _MainViewState();
}

class _MainViewState extends State<MainView> {
  final DatabaseHelper dbHelper = DatabaseHelper();
  List<Map<String, dynamic>> _dbData = [];

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

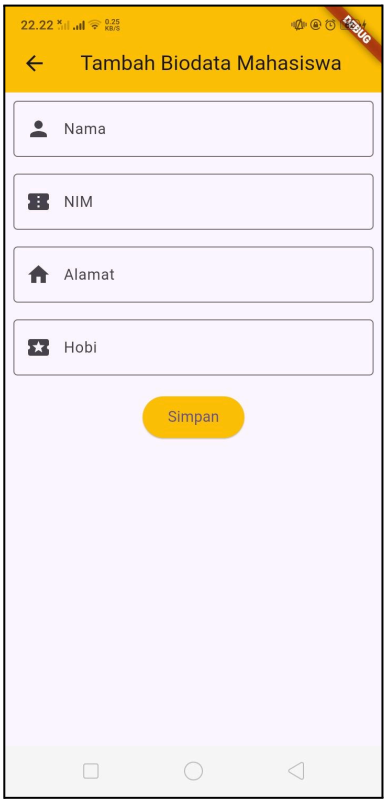
```
  }

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

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

  //untuk mengedit data
  void _editData(Map<String, dynamic> item) {
    // Panggil MyDatabaseView dengan mode edit
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) =>
            MyDatabaseView(item: item), // Kirim data untuk diedit
      ),
    ).then((_) {
      _refreshData(); // Refresh data setelah edit
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('SQLite Biodata Mahasiswa'),
        backgroundColor: Colors.amber,
        centerTitle: true,
      ),
      body: _dbData.isEmpty
          ? Center(
              child: Text(
                'Belum ada data mahasiswa',
                style: TextStyle(
                  fontSize: 15,
```

```
                color: Colors.grey,
              ),
            ),
          )
        : ListView.builder(
            padding: const EdgeInsets.all(8.0),
            itemCount: _dbData.length,
            itemBuilder: (context, index) {
              final item = _dbData[index];
              return Card(
                margin: const EdgeInsets.symmetric(vertical: 8),
                child: ListTile(
                  title: Text(
                    item['nama'],
                    style: const TextStyle(
                        fontWeight: FontWeight.bold, fontSize: 18),
                  ),
                  subtitle: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      Text('NIM: ${item['nim']}'),
                      Text('Alamat: ${item['alamat']}'),
                      Text('Hobi: ${item['hobi']}'),
                    ],
                  ),
                  trailing: Row(
                    mainAxisSize: MainAxisSize.min,
                    children: [
                      // Ikon pensil untuk edit
                      IconButton(
                        icon: const Icon(Icons.edit),
                        onPressed: () =>
                            _editData(item), // Panggil fungsi edit
                      ),
                      // Ikon tong sampah untuk delete
                      IconButton(
                        icon: const Icon(Icons.delete),
                        onPressed: () => _deleteData(item['id']),
                      ),
                    ],
                  ),
                ),
              );
```

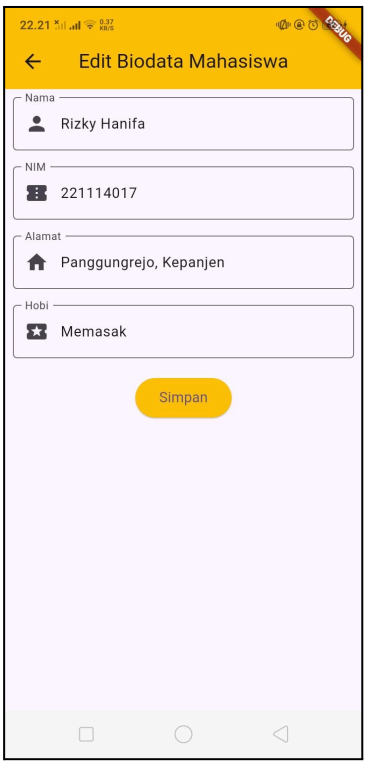
```
            },
          ),
      floatingActionButton: FloatingActionButton(
        onPressed: () async {
          await Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => MyDatabaseView(),
            ),
          );
          _refreshData();
        },
        backgroundColor: Colors.amber,
        child: const Icon(Icons.add),
      ),
    );
  }
}
```

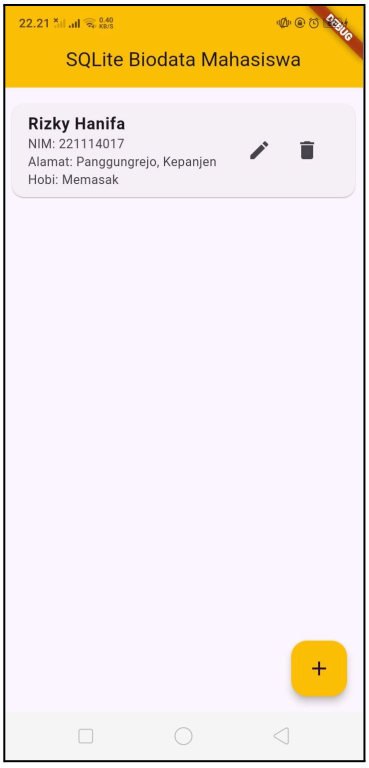
Output:

a. Berikut ini adalah tampilan utama dari program, ketika belum ada data yang diinputkan. Ketika user ingin menginputkan data mahasiswa, maka menekan tombol plus di pojok kanan bawah, selanjutnya akan ditampilkan form “Tambah Biodata Mahasiswa’.

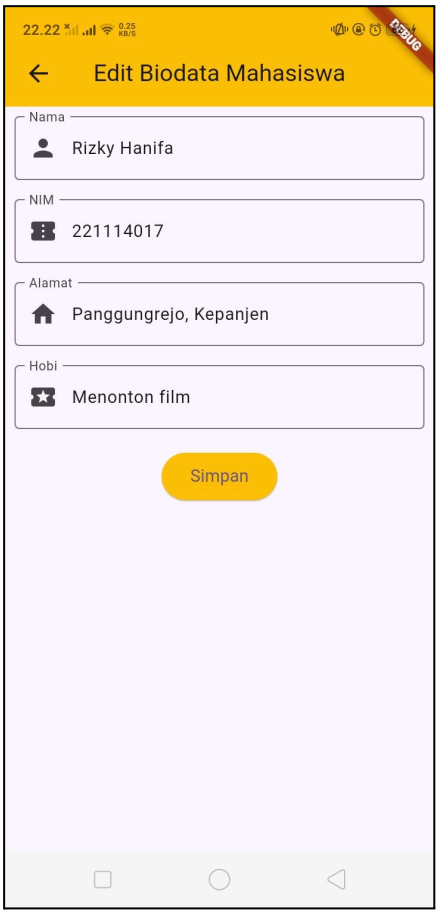


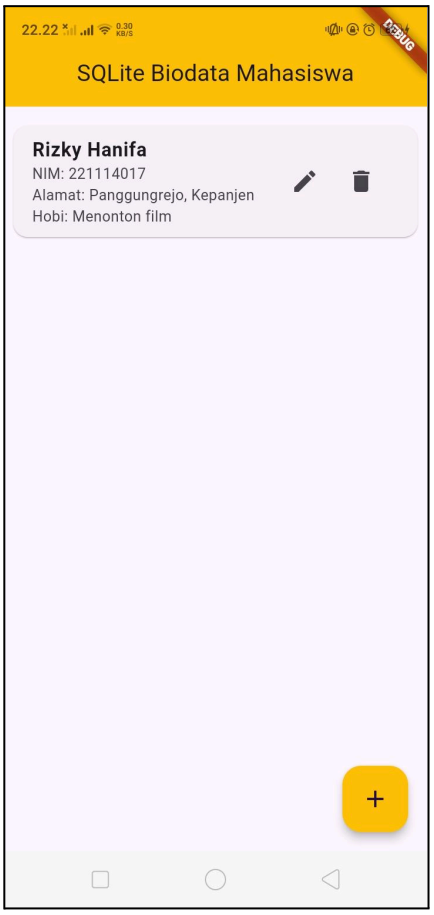
b. Ini adalah tampilan ketika user mengisi form. Setelah user klik tombol simpan, data ada akan ditampilkan di halaman utama





c. Ketika ingin mengedit, user bisa klik icon “pensil” lalu bisa merubah data, misalkan akan merubah hobi yang awalnya memasak menjadi menonton film





d. Jika ingin menghapus data bisa dengan menekan ikon sampah, lalu data akan terhapus.

Deskripsi Program:

Di dalam program ini menggunakan SQLite untuk mengelola data biodata mahasiswa. Pengguna bisa menambahkan, mengedit, dan menghapus data mahasiswa, lalu data akan tersimpan secara offline menggunakan database SQLite. Berikut adalah penjelasan tentang struktur program:

a. **File main.dart:**

   Digunakan untuk menjalankan aplikasi dengan homepage “mainView” yang menampilkan daftar data mahasiswa yang tersimpan di database.

b. **File db_helper.dart:**

   Digunakan untuk mengelola database SQLite. File ini berisi semua fungsi untuk menginisialisasi database, membuat tabel, dan mengelola data seperti menyisipkan (insert), memperbarui (update), dan menghapus (delete) data. Terdapat jua sebuah tabel untuk menyimpan data yaitu ID, nama, NIM, alamat, dan hobi.

c. **Antarmuka Utama (MainView):**

   Digunakan untuk menampilkan daftar data mahasiswa yang disimpan di database. Jika data belum ada, aplikasi akan menunjukkan pesan "Belum ada data mahasiswa". Pengguna dapat menambahkan data baru dengan menekan tombol tambah (FloatingActionButton) atau mengedit dan menghapus data yang ada melalui ikon yang tersedia di setiap item daftar.

d. **Form Input Data (MyDatabaseView):**

   Digunakan untuk menambah atau mengedit data mahasiswa. Di halaman ini memiliki beberapa kolom input seperti nama, NIM, alamat, dan hobi. Jika data sedang diedit, nilai yang ada akan dimasukkan ke kolom input secara otomatis. Lalu terdapat tombol "Simpan" yang digunakan untuk menyimpan data ke database.