LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK

MODUL X

DATA STORAGE (BAGIAN I)

Disusun Oleh :
Dwi Candra Pratama / 2211104035

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

GUIDED
DATA STORAGE (BAGIAN I)

A. TUJUAN PRAKTIKUM

- Mahasiswa mampu memahami konsep layout pada Flutter.
- Mahasiswa dapat mengimplementasikan desain user interface pada Flutter.

1. Pengenalan Sqlite

SQLite adalah sistem basis data relasional yang digunakan untuk menyimpan data secara **offline** dalam aplikasi mobile (di *local storage* atau lebih tepatnya di *cache memory* aplikasi). SQLite mendukung operasi **CRUD** (*Create, Read, Update, Delete*) yang memungkinkan pengelolaan data dengan mudah. Struktur dan perintah database SQLite mirip dengan SQL pada umumnya, seperti penggunaan tabel, tipe data, dan query. SQLite merupakan pilihan populer untuk aplikasi mobile karena ringan, cepat, dan tidak memerlukan server eksternal. Untuk memahami konsep dasar SQL lebih lanjut, Anda dapat mengacu pada berbagai sumber atau dokumentasi resmi SQLite.

2. SQL Helper Dasar

Dalam Flutter, SQL Helper mengacu pada pendekatan atau kelas pembantu yang digunakan untuk mengelola database SQLite dengan lebih mudah, biasanya melalui penggunaan paket seperti sqflite. SQL Helper dirancang untuk menyederhanakan operasi CRUD melalui metode terstruktur, seperti:

- Create: Menambahkan data ke dalam database.
- Read: Membaca data dari tabel.
- Update: Memperbarui data yang sudah ada.
- Delete: Menghapus data dari database.

Paket sqflite memberikan API sederhana untuk menangani operasi SQLite di Flutter dan sering digunakan bersama dengan SQL Helper untuk menjaga kode tetap modular dan terorganisasi.

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

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

```
37      cupertino_icons: ^1.0.8
38      sqflite: ^2.4.1
39      path: ^1.9.0
```

2. Inisialisasi Database Helper

   Kode ini membuat kelas `DatabaseHelper` sebagai *singleton* untuk mengelola database:

```
class DatabaseHelper {
```

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

    factory DatabaseHelper() {
      return _instance;
    }

    DatabaseHelper._internal();
}
```

Penjelasan:

- Singleton Pattern: Menggunakan instance tunggal dari `DatabaseHelper` agar database tidak dibuat berkali-kali.
- Factory constructor `(factory DatabaseHelper)` memastikan semua pemanggilan kelas ini mengembalikan instance yang sama.

3. Inisialisasi Database

Database hanya diinisialisasi sekali menggunakan metode `get database`:

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

Penjelasan:

- Metode ini memastikan database diinisialisasi hanya sekali dan di-cache di variabel `_database` untuk mengurangi overhead.

Metode _initDatabase digunakan untuk menentukan nama database dan lokasinya:

```
Future<Database> _initDatabase() async {
  String path = join(await getDatabasesPath(), 'my_prakdatabase.db');
  return await openDatabase(
    path,
    version: 1,
    onCreate: _onCreate,
  );
}
```

Penjelasan:

- `getDatabasesPath():` Mengambil lokasi direktori database.
- `openDatabase`: Membuka atau membuat database baru.
- `onCreate`: Callback yang akan dieksekusi saat database pertama kali dibuat.

4. Membuat Struktur Tabel

Metode `_onCreate` digunakan untuk mendefinisikan struktur tabel:

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

Penjelasan:

- Tabel `my_table` memiliki kolom:
  - `id`: Primary key yang bertipe integer dan otomatis bertambah.
  - `title`: Kolom teks.
  - `description`: Kolom teks.
  - `createdAt`: Kolom waktu dengan nilai default saat ini.

5. Operasi CRUD

   metode CRUD (Create, Read, Update, Delete):

   a) Create (Insert)

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

   Penjelasan:

   - `db.insert`: Memasukkan data ke dalam tabel `my_table`. Data berupa `Map<String, dynamic>.`

   b) Read (Query All Rows)

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

   Penjelasan:

   - `db.query`: Mengambil semua baris dari tabel `my_table`.

   c) Update

```
        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]);
```

```
}
```

Penjelasan:

- `db.update`: Memperbarui data berdasarkan `id`.

d) Delete

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

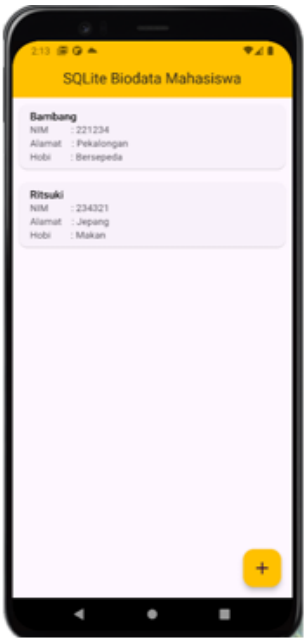
Penjelasan:

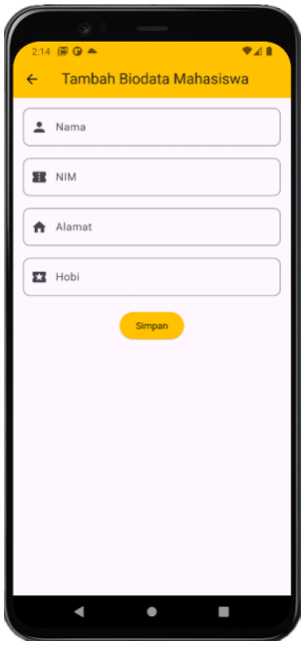
- `db.delete`: Menghapus data berdasarkan `id`.

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).

Contoh output:





Note: Jangan lupa sertakan source code, screenshoot output, dan deskripsi program. Kreatifitas menjadi nilai tambah.

SOURCE CODE:

```
30  dependencies:
31    flutter:
32      sdk: flutter
33
34
35    # The following adds the Cupertino Icons font to your application.
36    # Use with the CupertinoIcons class for iOS style icons.
37    cupertino_icons: ^1.0.8
38    sqflite: ^2.4.1
39    path: ^1.9.0
```

FOLDER `database` FILE `db_helper.dart.`

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

class DatabaseHelper {
  static final _databaseName = "mahasiswa.db";
  static final _databaseVersion = 1;

  static final table = 'mahasiswa';

  static final columnId = 'id';
  static final columnName = 'nama';
  static final columnNim = 'nim';
  static final columnAlamat = 'alamat';
  static final columnHobi = 'hobi';

  // Singleton
  DatabaseHelper._privateConstructor();
  static final DatabaseHelper instance = DatabaseHelper._privateConstructor();

  static Database? _database;

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

  _initDatabase() async {
    String path = join(await getDatabasesPath(), _databaseName);
```

```
    return await openDatabase(path, version: _databaseVersion, onCreate:
_onCreate);
  }

  Future _onCreate(Database db, int version) async {
    await db.execute('''
      CREATE TABLE $table (
        $columnId INTEGER PRIMARY KEY AUTOINCREMENT,
        $columnName TEXT NOT NULL,
        $columnNim TEXT NOT NULL,
        $columnAlamat TEXT NOT NULL,
        $columnHobi TEXT NOT NULL
      )
    ''');
  }

  Future<int> insertMahasiswa(Map<String, dynamic> row) async {
    Database? db = await instance.database;
    return await db!.insert(table, row);
  }

  Future<List<Map<String, dynamic>>> getMahasiswa() async {
    Database? db = await instance.database;
    return await db!.query(table);
  }
}
```

FOLDER `models` FILE `mahasiswa.dart.`

```
class Mahasiswa {
  final int? id;
  final String nama;
  final String nim;
  final String alamat;
  final String hobi;

  Mahasiswa({this.id, required this.nama, required this.nim, required this.alamat,
required this.hobi});

  Map<String, dynamic> toMap() {
    return {
      'id': id,
      'nama': nama,
      'nim': nim,
      'alamat': alamat,
```

```
      'hobi': hobi,
    };
  }

  factory Mahasiswa.fromMap(Map<String, dynamic> map) {
    return Mahasiswa(
      id: map['id'],
      nama: map['nama'],
      nim: map['nim'],
      alamat: map['alamat'],
      hobi: map['hobi'],
    );
  }
}
```

FOLDER `screens` FILE `add_mahasiswa_screen.dart.`

```
import 'package:flutter/material.dart';
import 'package:praktikum_10/database/db_helper.dart';
import 'package:praktikum_10/models/mahasiswa.dart';

class AddMahasiswaScreen extends StatelessWidget {
  final TextEditingController _namaController = TextEditingController();
  final TextEditingController _nimController = TextEditingController();
  final TextEditingController _alamatController = TextEditingController();
  final TextEditingController _hobiController = TextEditingController();

  void _saveMahasiswa(BuildContext context) async {
    final mahasiswa = Mahasiswa(
      nama: _namaController.text,
      nim: _nimController.text,
      alamat: _alamatController.text,
      hobi: _hobiController.text,
    );
    await DatabaseHelper.instance.insertMahasiswa(mahasiswa.toMap());
    Navigator.pop(context);
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Tambah Biodata Mahasiswa'),
        backgroundColor: Colors.pinkAccent[400], // Warna latar belakang AppBar
      ),
```

```
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            TextField(
              controller: _namaController,
              decoration: InputDecoration(labelText: 'Nama'),
            ),
            TextField(
              controller: _nimController,
              decoration: InputDecoration(labelText: 'NIM'),
            ),
            TextField(
              controller: _alamatController,
              decoration: InputDecoration(labelText: 'Alamat'),
            ),
            TextField(
              controller: _hobiController,
              decoration: InputDecoration(labelText: 'Hobi'),
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: () => _saveMahasiswa(context),
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.pinkAccent[400], // Warna latar tombol
              ),
              child: Text(
                'Simpan',
                style: TextStyle(
                  color: Colors.white, // Warna teks agar terlihat jelas
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

FOLDER `screens` FILE `home_screen.dart.`

```
import 'package:flutter/material.dart';
import 'package:praktikum_10/database/db_helper.dart';
import 'package:praktikum_10/models/mahasiswa.dart';
```

```
class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  List<Mahasiswa> mahasiswaList = [];

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

  Future<void> _loadMahasiswa() async {
    final data = await DatabaseHelper.instance.getMahasiswa();
    setState(() {
      mahasiswaList = data.map((e) => Mahasiswa.fromMap(e)).toList();
    });
  }

  void _navigateToAddMahasiswa() async {
    await Navigator.pushNamed(context, '/add');
    _loadMahasiswa();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        centerTitle: true,
        title: Text('SQLite Biodata Mahasiswa'),
        backgroundColor: Colors.pinkAccent[400],
      ),
      body: ListView.builder(
        itemCount: mahasiswaList.length,
        itemBuilder: (context, index) {
          final mahasiswa = mahasiswaList[index];
          return Padding(
            padding: const EdgeInsets.symmetric(horizontal: 16.0, vertical: 8.0),
            child: Card(
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(10.0),
              ),
              elevation: 4,
```

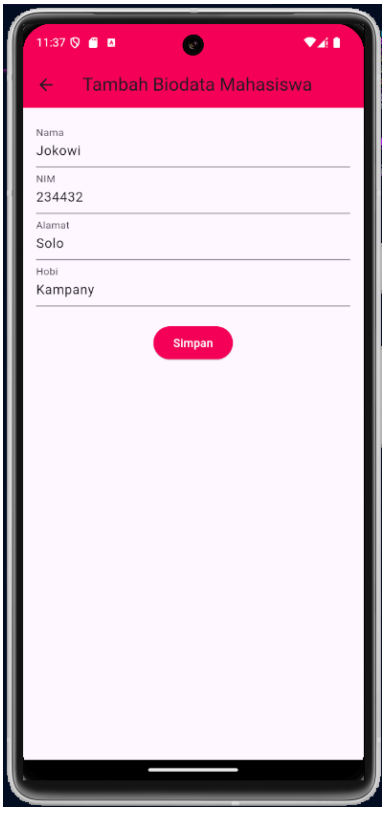
```
            child: Padding(
              padding: const EdgeInsets.all(16.0),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    mahasiswa.nama,
                    style: TextStyle(
                      fontWeight: FontWeight.bold,
                      fontSize: 18,
                    ),
                  ),
                  SizedBox(height: 4),
                  Text('NIM     : ${mahasiswa.nim}'),
                  Text('Alamat  : ${mahasiswa.alamat}'),
                  Text('Hobi    : ${mahasiswa.hobi}'),
                ],
              ),
            ),
          ),
        );
      },
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: _navigateToAddMahasiswa,
      backgroundColor: Colors.pinkAccent[400],
      child: Icon(Icons.add),
    ),
  );
 }
}
```

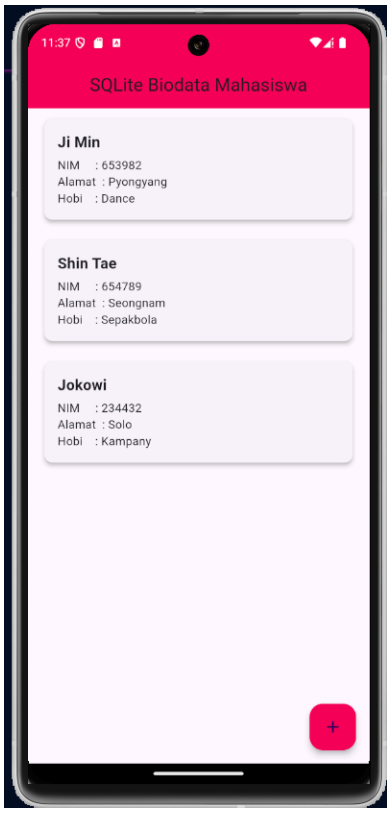
FILE `main.dart.`

```
import 'package:flutter/material.dart';
import 'package:praktikum_10/screens/add_mahasiswa_screen.dart';
import 'screens/home_screen.dart';

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

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
```

```
    return MaterialApp(
      title: 'SQLite Biodata Mahasiswa',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(primarySwatch: Colors.amber),
      initialRoute: '/',
      routes: {
        '/': (context) => HomeScreen(),
        '/add': (context) => AddMahasiswaScreen(),
      },
    );
  }
}
```

OutPut





Deskripsi Program:

Program ini merupakan aplikasi Flutter berbasis SQLite yang dirancang untuk menyimpan dan menampilkan biodata mahasiswa. Pengguna dapat menambahkan data mahasiswa yang meliputi nama, NIM, alamat, dan hobi melalui sebuah form input di halaman "Tambah Biodata Mahasiswa." Data yang berhasil disimpan akan ditampilkan dalam bentuk daftar di halaman utama menggunakan elemen Card untuk mempercantik tampilan. Aplikasi ini menerapkan fitur Create untuk menyimpan data dan Read untuk membaca serta menampilkan data yang tersimpan di dalam database lokal SQLite. Navigasi antarhalaman menggunakan Navigator, sementara desain antarmuka menonjolkan warna-warna cerah seperti kuning dan merah muda (pink) untuk memberikan kesan modern dan menarik.