LAPORAN PRAKTIKUM
Modul 10
"DATA STORAGE (BAGIAN I)"

Disusun Oleh:
Ganesha Rahman Gibran -2211104058
Kelas S1SE-06-02

Dosen:
Yudha Islami Sulistya, S.Kom., M.Cs.

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO

2024

Tujuan

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

Landasan Teori

1. Pengenalan SQLite

SQLite adalah sistem database relasional yang sering digunakan untuk menyimpan data secara lokal di perangkat, khususnya dalam aplikasi mobile. SQLite memungkinkan aplikasi menyimpan data secara offline dengan memanfaatkan local storage (biasanya di direktori cache aplikasi). Seperti sistem database lainnya, SQLite mendukung operasi CRUD (Create, Read, Update, Delete) yang penting untuk pengelolaan data. Struktur database SQLite mirip dengan SQL pada umumnya, termasuk variabel dan tipe data. SQLite menjadi solusi ideal untuk aplikasi ringan yang membutuhkan pengelolaan data sederhana tanpa server.

2. SQL Helper Dasar di Flutter

Di Flutter, pengelolaan SQLite sering dilakukan menggunakan plugin sqflite. Sqflite memungkinkan integrasi SQLite ke dalam aplikasi Flutter, termasuk operasi CRUD. Untuk mempermudah manajemen database, biasanya dibuat sebuah class helper, misalnya DatabaseHelper. Berikut adalah langkah-langkah implementasi dasar:

2.1. Menambahkan Dependency sqflite

Tambahkan dependensi **sqflite** dan **path** ke dalam file pubspec.yaml. Kedua plugin ini digunakan untuk mengakses SQLite dan mendapatkan lokasi penyimpanan database di perangkat.

```
dependencies:
  sqflite: ^x.x.x
 path: ^x.x.x
```

2.2. Membuat Class DatabaseHelper

Buat file baru bernama db_helper.dart. Class DatabaseHelper akan berfungsi untuk mengelola koneksi dan operasi pada database.

```
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();
}
```

2.3. Menambahkan Getter Database

Tambahkan getter untuk mengakses database secara singleton.

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

```
}
```

2.4.Inisiasi Database

Metode _initDatabase bertugas untuk membuat atau membuka database dengan nama yang diinginkan.

```
Future<Database> _initDatabase() async {
  String path = join(await getDatabasesPath(), 'prak_database.db');
  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,
      title TEXT,
      description TEXT,
      createdAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    )
  ''');
}
```

2.5.Operasi CRUD

- Create

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

- Read

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

Future<List<Map<String, dynamic>>> getItem(int id) async {
  Database db = await database;
  return await db.query('my_table', where: "id = ?",
    whereArgs: [id], limit: 1);
}
```

- 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]);
}
```

- Delete

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

Guided

Input :

- main.dart

```
import 'package:flutter/material.dart';
import 'package:guided1/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 Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: MyDatabaseView(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});
  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        title: Text(widget.title),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            const Text(
```

```
              'You have pushed the button this many times:',
            ),
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headlineMedium,
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ),
    );
  }
}
```

- Lib/helper/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;
  }

  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(), 'my_prakdatabase.db');
    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)
      ''');
    }

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

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

    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]);
    }

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

- Lib/view/my_db_view.dart

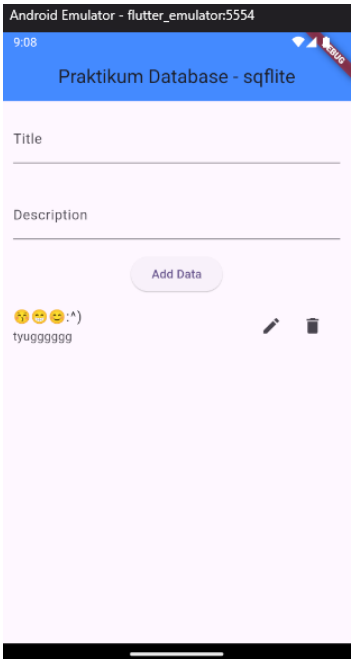
```
import 'package:flutter/material.dart';
import 'package:guided1/helper/db_helper.dart';

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

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

class _MyDatabaseViewState extends State<MyDatabaseView> {
  final DatabaseHelper dbHelper = DatabaseHelper(); // Inisialisasi
   DatabaseHelper
  List<Map<String, dynamic>> _dbData = []; // Menyimpan data dari database
  final TextEditingController _titleController = TextEditingController(); //
   Controller untuk title
  final TextEditingController _descriptionController = TextEditingController();
   // Controller untuk deskripsi

  @override
  void initState() {
    super.initState();
    _refreshData(); // Mengambil data saat widget diinisialisasi
  }

  @override
  void dispose() {
    _titleController.dispose(); // Membersihkan controller saat widget
```

```
    dihancurkan
    _descriptionController.dispose();
    super.dispose();
  }

  // Mengambil semua data dari database dan memperbarui state
  void _refreshData() async {
    final data = await dbHelper.queryAllRows();
    setState(() {
      _dbData = data;
    });
  }

  // Menambahkan data baru ke database
  void _addData() async {
    await dbHelper.insert({
      'title': _titleController.text,
      'description': _descriptionController.text,
    });
    _titleController.clear(); // Mengosongkan input
    _descriptionController.clear();
    _refreshData();
  }

  // Memperbarui data yang ada di database
  void _updateData(int id) async {
    await dbHelper.update({
      'id': id,
      'title': _titleController.text,
      'description': _descriptionController.text,
    });
    _titleController.clear();
    _descriptionController.clear();
    _refreshData();
  }

  // Menghapus data dari database berdasarkan id
  void _deleteData(int id) async {
    await dbHelper.delete(id);
    _refreshData();
  }

  // Menampilkan dialog untuk mengedit data
  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 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(
              onPressed: () {
                _updateData(item['id']);
                Navigator.of(context).pop();
              },
              child: const Text('Update'),
            ),
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: const Text('Cancel'),
            ),
          ],
        );
      },
    );
  }

  @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(16.0),
            child: TextField(
              controller: _titleController,
              decoration: const InputDecoration(
                labelText: 'Title',
              ),
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(16.0),
```

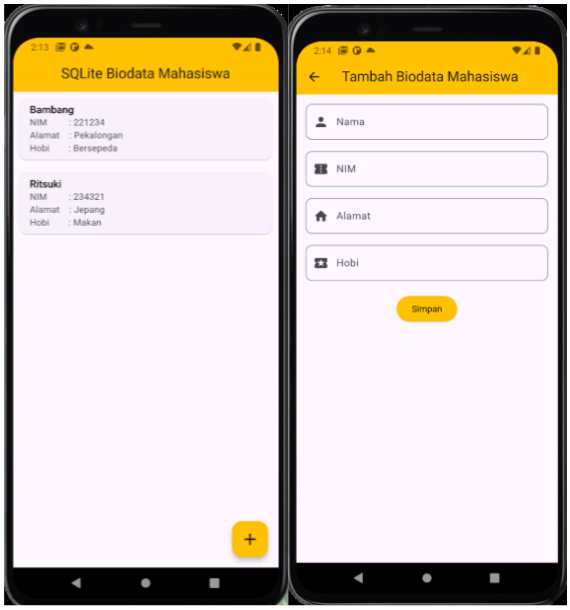
```
              child: TextField(
                controller: _descriptionController,
                decoration: const InputDecoration(
                  labelText: 'Description',
                ),
              ),
            ),
            ElevatedButton(
              onPressed: _addData,
              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 :



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



Input :

- Lib/main.dart

```
import 'package:flutter/material.dart';
import 'screens/home_page.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: HomePage(),
    );
  }
}
```

- Lib/helpers/db_helper.dart

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

class DBHelper {
  static Database? _database;

  static Future<Database> _getDatabase() async {
    if (_database != null) return _database!;
    _database = await _initDatabase();
    return _database!;
  }

  static Future<Database> _initDatabase() async {
    final databasePath = await getDatabasesPath();
    final path = join(databasePath, 'mahasiswa.db');

    return openDatabase(
      path,
      version: 1,
      onCreate: (db, version) {
```

```
        return db.execute(
          '''
          CREATE TABLE mahasiswa (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            nama TEXT,
            nim TEXT,
            alamat TEXT,
            hobi TEXT
          )
          ''',
        );
      },
    );
  }

  static Future<void> insertMahasiswa(Mahasiswa mahasiswa) async {
    final db = await _getDatabase();
    await db.insert('mahasiswa', mahasiswa.toMap());
  }

  static Future<List<Mahasiswa>> getAllMahasiswa() async {
    final db = await _getDatabase();
    final List<Map<String, dynamic>> maps = await db.query('mahasiswa');

    return List.generate(
      maps.length,
      (i) => Mahasiswa.fromMap(maps[i]),
    );
  }
}
```

- Lib/models/mahasiswa.dart

```
class Mahasiswa {
  int? id;
  String nama;
  String nim;
  String alamat;
  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'],
    );
  }
}
```

- Lib/screens/home_page.dart

```
import 'package:flutter/material.dart';
import '../helpers/db_helper.dart';
import '../models/mahasiswa.dart';
import 'tambah_mahasiswa_page.dart';

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

class _HomePageState extends State<HomePage> {
  List<Mahasiswa> mahasiswaList = [];

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

  Future<void> fetchData() async {
    mahasiswaList = await DBHelper.getAllMahasiswa();
    setState(() {});
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('SQLite Biodata Mahasiswa'),
        backgroundColor: Colors.orange,
      ),
      body: ListView.builder(
        itemCount: mahasiswaList.length,
        itemBuilder: (context, index) {
          final mahasiswa = mahasiswaList[index];
          return Card(
            margin: EdgeInsets.all(10),
            child: ListTile(
              title: Text(mahasiswa.nama),
              subtitle: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text('NIM: ${mahasiswa.nim}'),
                  Text('Alamat: ${mahasiswa.alamat}'),
                  Text('Hobi: ${mahasiswa.hobi}'),
                ],
              ),
            ),
          );
        },
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(builder: (context) =>
TambahMahasiswaPage(fetchData)),
```

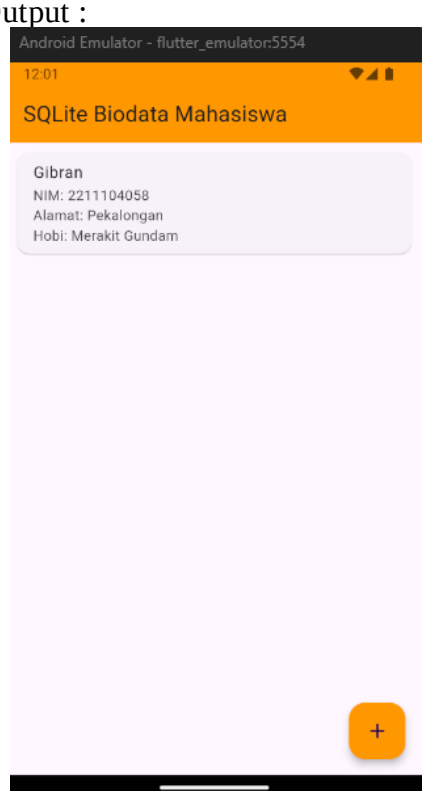
```
          );
        },
        child: Icon(Icons.add),
        backgroundColor: Colors.orange,
      ),
    );
  }
}
```

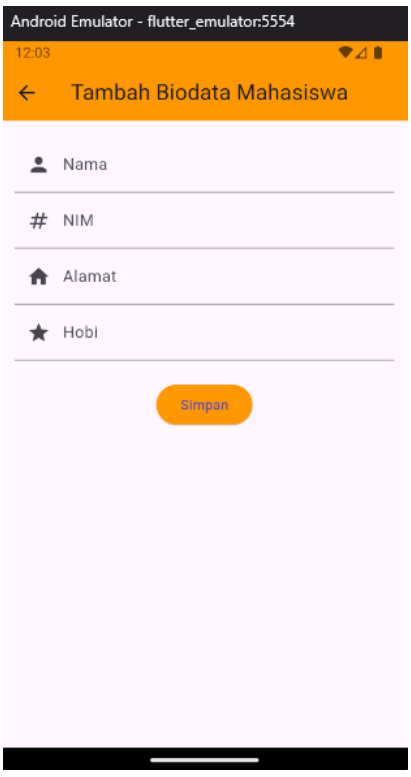
- Lib/screens/tambah_mahasiswa_page.dart

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

class TambahMahasiswaPage extends StatefulWidget {
  final Function onSaved;
  TambahMahasiswaPage(this.onSaved);

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

class _TambahMahasiswaPageState extends State<TambahMahasiswaPage> {
  final _formKey = GlobalKey<FormState>();
  final _namaController = TextEditingController();
  final _nimController = TextEditingController();
  final _alamatController = TextEditingController();
  final _hobiController = TextEditingController();

  Future<void> saveData() async {
    if (_formKey.currentState!.validate()) {
      Mahasiswa mahasiswa = Mahasiswa(
        nama: _namaController.text,
        nim: _nimController.text,
        alamat: _alamatController.text,
        hobi: _hobiController.text,
      );
      await DBHelper.insertMahasiswa(mahasiswa);
      widget.onSaved();
      Navigator.pop(context);
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Tambah Biodata Mahasiswa'),
        backgroundColor: Colors.orange,
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Form(
          key: _formKey,
          child: Column(
            children: [
              TextFormField(
                controller: _namaController,
                decoration: InputDecoration(labelText: 'Nama', prefixIcon:
Icon(Icons.person)),
                validator: (value) => value!.isEmpty ? 'Nama wajib diisi' :
null,
              ),
```

```
              TextFormField(
                controller: _nimController,
                decoration: InputDecoration(labelText: 'NIM', prefixIcon:
Icon(Icons.numbers)),
                validator: (value) => value!.isEmpty ? 'NIM wajib diisi' :
null,
              ),
              TextFormField(
                controller: _alamatController,
                decoration: InputDecoration(labelText: 'Alamat',
prefixIcon: Icon(Icons.home)),
                validator: (value) => value!.isEmpty ? 'Alamat wajib diisi'
: null,
              ),
              TextFormField(
                controller: _hobiController,
                decoration: InputDecoration(labelText: 'Hobi', prefixIcon:
Icon(Icons.star)),
                validator: (value) => value!.isEmpty ? 'Hobi wajib diisi' :
null,
              ),
              SizedBox(height: 20),
              ElevatedButton(
                onPressed: saveData,
                child: Text('Simpan'),
                style: ElevatedButton.styleFrom(backgroundColor:
Colors.orange),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

Output :





Deskripsi kode :

Implementasi halaman untuk menambahkan data biodata mahasiswa dalam aplikasi Flutter dengan menggunakan SQLite. Halaman ini berfungsi sebagai formulir input yang memungkinkan pengguna untuk memasukkan data seperti nama, NIM, alamat, dan hobi. Formulir ini dilengkapi dengan validasi untuk memastikan setiap kolom diisi sebelum data dapat disimpan. Jika validasi berhasil, data yang diinput akan dibuat menjadi sebuah objek Mahasiswa dan disimpan ke dalam database SQLite melalui fungsi DBHelper.insertMahasiswa. Setelah data berhasil disimpan, fungsi callback onSaved akan dipanggil untuk memperbarui daftar data di halaman utama, dan pengguna akan diarahkan kembali ke halaman sebelumnya.

Antarmuka halaman ini dirancang menggunakan widget Flutter seperti Scaffold, dengan elemen-elemen seperti AppBar untuk judul halaman, serta TextFormField untuk kolom input data. Setiap kolom input memiliki ikon yang relevan untuk membantu pengguna memahami fungsi masing-masing kolom, misalnya ikon orang untuk nama dan ikon rumah untuk alamat. Tombol "Simpan" yang berwarna oranye disediakan sebagai aksi utama, yang mengeksekusi logika penyimpanan data saat ditekan. Dengan desain ini, halaman tersebut mendukung pengalaman pengguna yang sederhana dan intuitif dalam menambahkan data mahasiswa ke aplikasi.

Kesimpulan

Sqflite mempermudah pengelolaan database SQLite di Flutter, mendukung operasi CRUD yang dapat disesuaikan dengan kebutuhan aplikasi. Dengan bantuan class helper seperti DatabaseHelper, pengelolaan database menjadi lebih terstruktur dan efisien.