

**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 :

```
1  import 'package:flutter/material.dart';
2  import 'package:prak_10/view/my_db_view.dart';
3
4  void main() {
5    runApp(const MyApp());
6  }
7
8  class MyApp extends StatelessWidget {
9    const MyApp({super.key});
10
11    // This widget is the root of your application.
12    @override
13    Widget build(BuildContext context) {
14      return MaterialApp(
15        title: 'Flutter Demo',
16        theme: ThemeData(
17          colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
18          useMaterial3: true,
19        ),
20        home: MyDatabaseView(),
21      );
22    }
23  }
24
```

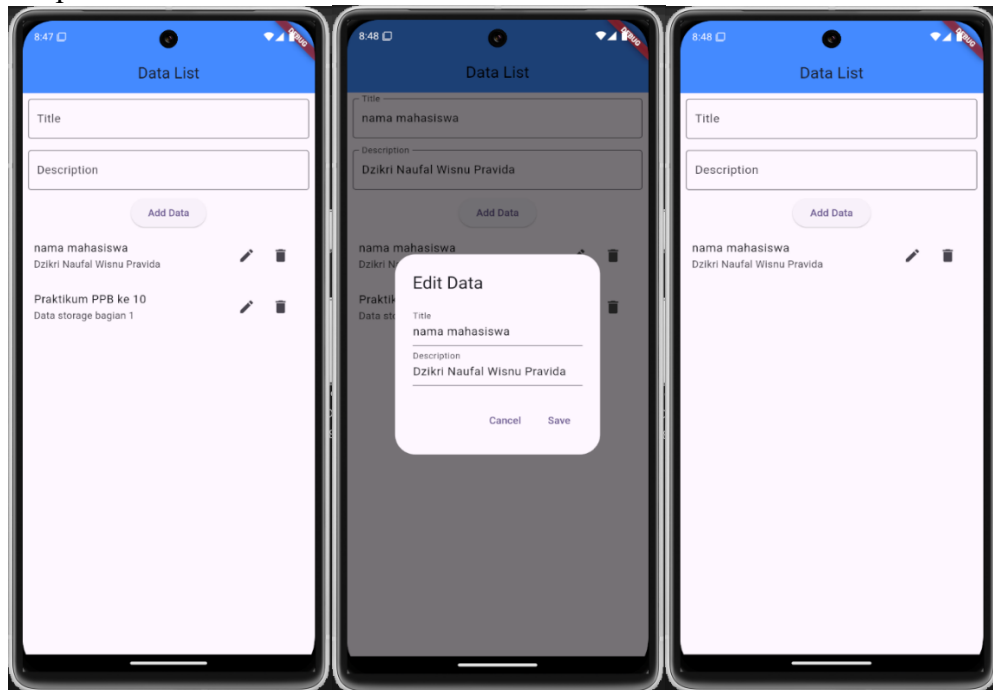
Db helper

```
1  import 'package:sqflite/sqflite.dart';
2  import 'package:path/path.dart';
3
4  class DatabaseHelper {
5    static final DatabaseHelper _instance = DatabaseHelper._internal();
6    static Database? _database;
7
8    factory DatabaseHelper() {
9      return _instance;
10   }
11
12   //private constructor
13   DatabaseHelper._internal();
14
15   // getter untuk database
16   Future<Database> get database async {
17     if (_database != null) return _database!;
18     {
19       _database = await _initDatabase();
20       return _database!;
21     }
22   }
23
24   //inisialisasi database
25   Future<Database> _initDatabase() async {
26     // mendapatkan path untuk database
27     String path = join(await getDatabasesPath(), 'my_prakdatabase.db');
28     // membuka database
29     return await openDatabase(
30       path,
31       version: 1,
32       onCreate: _onCreate,
33     );
34   }
35
36   Future<void> _onCreate(Database db, int version) async {
37     await db.execute('''
38 CREATE TABLE my_table(
39 id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
40 title TEXT,
41 description TEXT,
42 createdAt TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP)
43 ''');
44   }
45
46   // method untuk memasukan data ke tabel
47   Future<int> insert(Map<String, dynamic> row) async {
48     Database db = await database;
49     return await db.insert('my_table', row);
50   }
51
52   //method mengambil semua data dari tabel
53   Future<List<Map<String, dynamic>>> queryAllRows() async {
54     Database db = await database;
55     return await db.query('my_table');
56   }
57
58   //method untuk memperbarui data dalam tabel
59   Future<int> update(Map<String, dynamic> row) async {
60     Database db = await database;
61     int id = row['id'];
62     return await db.update('my_table', row, where: 'id = ?', whereArgs: [id]);
63   }
64
65   // method untuk menghapus data
66   Future<int> delete(int id) async {
67     Database db = await database;
68     return await db.delete('my_table', where: 'id = ?', whereArgs: [id]);
69   }
70 }
71
```

Db view

```
1 import 'package:flutter/material.dart';
2 import 'package:prak_10/helper/db_helper.dart';
3
4 class MyDatabaseView extends StatefulWidget {
5   const MyDatabaseView({super.key});
6
7   @override
8   State<MyDatabaseView> createState() => _MyDatabaseViewState();
9 }
10
11 class _MyDatabaseViewState extends State<MyDatabaseView> {
12   final DatabaseHelper dbHelper = DatabaseHelper();
13   List<Map<String, dynamic>> _dbData = [];
14   final TextEditingController _titleController = TextEditingController();
15   final TextEditingController _descriptionController = TextEditingController();
16
17   @override
18   void initState() {
19     _refreshData();
20     super.initState();
21   }
22
23   @override
24   void dispose() {
25     _titleController.dispose();
26     _descriptionController.dispose();
27     super.dispose();
28   }
29
30   // method to refresh data from the database
31   void _refreshData() async {
32     final data = await dbHelper.queryAllRows();
33     setState(() {
34       _dbData = data;
35     });
36   }
37
38   // method to add data to the database
39   void _addData() async {
40     await dbHelper.insert({
41       'title': _titleController.text,
42       'description': _descriptionController.text,
43     });
44     _titleController.clear();
45     _descriptionController.clear();
46     _refreshData();
47   }
48
49   // method to update data in the database
50   void _updateData(int id) async {
51     await dbHelper.update({
52       'id': id,
53       'title': _titleController.text,
54       'description': _descriptionController.text,
55     });
56     _titleController.clear();
57     _descriptionController.clear();
58     _refreshData();
59   }
60
61   // method to delete data from the database
62   void _deleteData(int id) async {
63     await dbHelper.delete(id);
64     _refreshData();
65   }
66
67   // show dialog to edit data
68   void _showEditDialog(Map<String, dynamic> item) {
69     _titleController.text = item['title'];
70     _descriptionController.text = item['description'];
71     showDialog(
72       context: context,
73       builder: (BuildContext context) {
74         return AlertDialog(
75           title: const Text('Edit Data'),
76           content: Column(
77             mainAxisAlignment: MainAxisAlignment.min,
78             children: [
79               TextField(
80                 controller: _titleController,
81                 decoration: const InputDecoration(labelText: 'Title'),
82               ),
83               TextField(
84                 controller: _descriptionController,
85                 decoration: const InputDecoration(labelText: 'Description'),
86               ),
87             ],
88           ),
89           actions: [
90             TextButton(
91               child: const Text('Cancel'),
92               onPressed: () {
93                 Navigator.of(context).pop();
94               },
95             ),
96             TextButton(
97               child: const Text('Save'),
98               onPressed: () {
99                 _updateData(item['id']);
100                 Navigator.of(context).pop();
101               },
102             ),
103           ],
104         );
105       },
106     );
107   }
108
109   @override
110   Widget build(BuildContext context) {
111     return Scaffold(
112       appBar: AppBar(
113         title: const Text('Data List'),
114         backgroundColor: Colors.blueAccent,
115         centerTitle: true,
116       ),
117       body: Column(
118         children: [
119           Padding(
120             padding: const EdgeInsets.all(8.0),
121             child: TextField(
122               controller: _titleController,
123               decoration: const InputDecoration(
124                 labelText: 'Title',
125                 border: OutlineInputBorder(),
126               ),
127             ),
128           ),
129           Padding(
130             padding: const EdgeInsets.all(8.0),
131             child: TextField(
132               controller: _descriptionController,
133               decoration: const InputDecoration(
134                 labelText: 'Description',
135                 border: OutlineInputBorder(),
136               ),
137             ),
138           ),
139           ElevatedButton(
140             onPressed: _addData,
141             child: const Text('Add Data'),
142           ),
143           Expanded(
144             child: ListView.builder(
145               itemCount: _dbData.length,
146               itemBuilder: (context, index) {
147                 final item = _dbData[index];
148                 return ListTile(
149                   title: Text(item['title']),
150                   subtitle: Text(item['description']),
151                   trailing: Row(
152                     mainAxisAlignment: MainAxisAlignment.min,
153                     children: [
154                       IconButton(
155                         icon: const Icon(Icons.edit),
156                         onPressed: () => _showEditDialog(item),
157                       ),
158                       IconButton(
159                         icon: const Icon(Icons.delete),
160                         onPressed: () => _deleteData(item['id']),
161                       ),
162                     ],
163                   ),
164                 );
165               },
166             ),
167           ),
168         ],
169       ),
170     );
171   }
172 }
173
```

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:

- Form Input: Buat form input untuk menambahkan biodata mahasiswa, dengan kolom:
 - Nama
 - Nim
 - Alamat
 - Hobi
- Tampilkan Daftar Mahasiswa: Setelah data berhasil ditambahkan, tampilkan daftar semua data mahasiswa yang sudah disimpan di halaman utama.
- Implementasikan fitur Create (untuk menyimpan data mahasiswa) dan Read (untuk menampilkan daftar mahasiswa yang sudah disimpan).

Sourcecode

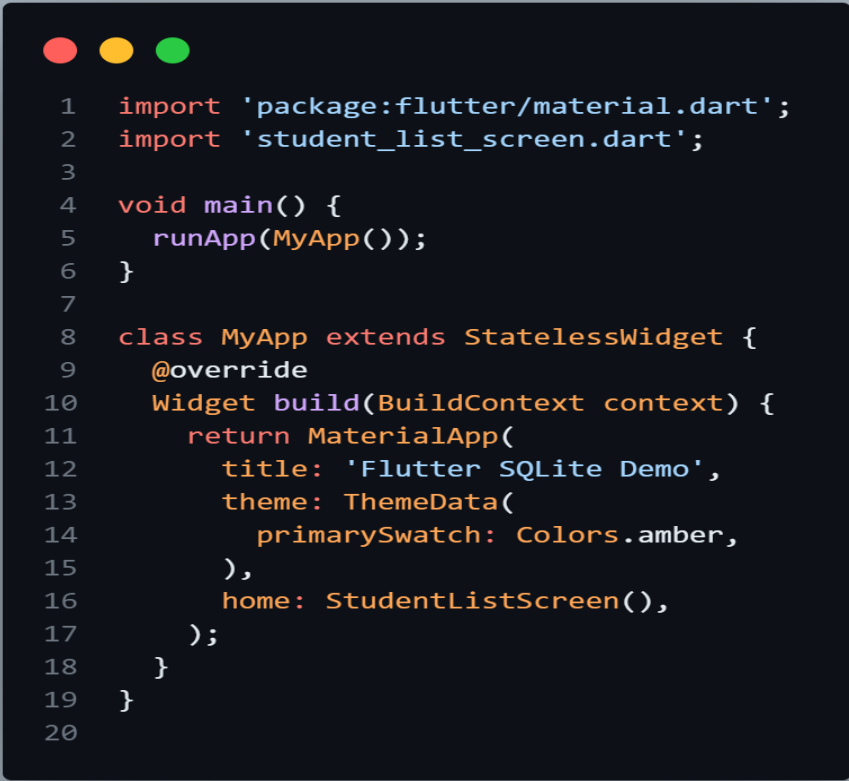
Instalasi package ke dalam pubspec.yaml

dependencies:

 sqlite: ^2.0.0+3

 path: ^1.8.0

main.dart



```
1  import 'package:flutter/material.dart';
2  import 'student_list_screen.dart';
3
4  void main() {
5    runApp(MyApp());
6  }
7
8  class MyApp extends StatelessWidget {
9    @override
10   Widget build(BuildContext context) {
11     return MaterialApp(
12       title: 'Flutter SQLite Demo',
13       theme: ThemeData(
14         primarySwatch: Colors.amber,
15       ),
16       home: StudentListScreen(),
17     );
18   }
19 }
20
```

Db helper

```
1 import 'package:sqflite/sqflite.dart';
2 import 'package:path/path.dart';
3
4 class DatabaseHelper {
5   static final DatabaseHelper _instance = DatabaseHelper._internal();
6   static Database? _database;
7
8   factory DatabaseHelper() => _instance;
9
10  DatabaseHelper._internal();
11
12  Future<Database> get database async {
13    if (_database != null) return _database!;
14    _database = await _initDatabase();
15    return _database!;
16  }
17
18  Future<Database> _initDatabase() async {
19    String path = join(await getDatabasesPath(), 'students.db');
20    return await openDatabase(
21      path,
22      version: 1,
23      onCreate: (db, version) async {
24        await db.execute('''
25          CREATE TABLE students (
26            id INTEGER PRIMARY KEY AUTOINCREMENT,
27            name TEXT,
28            nim TEXT,
29            address TEXT,
30            hobby TEXT
31          )
32        ''');
33      },
34    );
35  }
36
37  Future<int> insertStudent(Map<String, dynamic> row) async {
38    Database db = await database;
39    return await db.insert('students', row);
40  }
41
42  Future<List<Map<String, dynamic>>> getStudents() async {
43    Database db = await database;
44    return await db.query('students');
45  }
46
47  Future<int> updateStudent(Map<String, dynamic> row) async {
48    Database db = await database;
49    int id = row['id'];
50    return await db.update('students', row, where: 'id = ?', whereArgs: [id]);
51  }
52 }
53
```

Student list

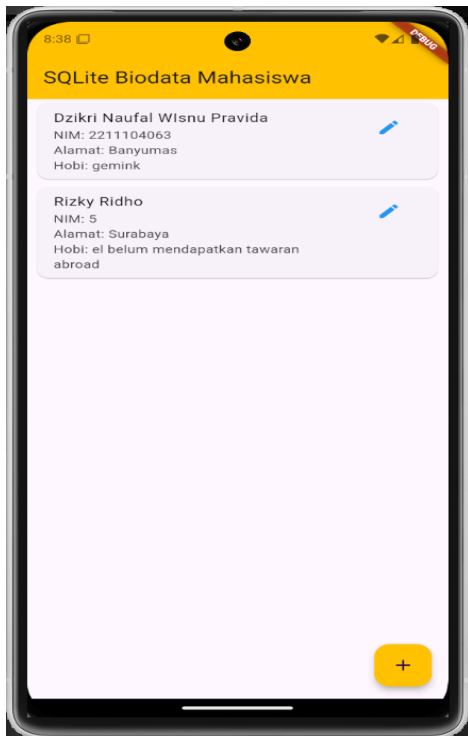
```
1 import 'package:flutter/material.dart';
2 import 'db_helper.dart';
3 import 'form_screen.dart';
4
5 class StudentListScreen extends StatefulWidget {
6   @override
7   _StudentListScreenState createState() => _StudentListScreenState();
8 }
9
10 class _StudentListScreenState extends State<StudentListScreen> {
11   final DatabaseHelper _dbHelper = DatabaseHelper();
12   List<Map<String, dynamic>> _students = [];
13
14   @override
15   void initState() {
16     super.initState();
17     _loadStudents();
18   }
19
20   void _loadStudents() async {
21     final data = await _dbHelper.getStudents();
22     setState(() {
23       _students = data;
24     });
25   }
26
27   void _navigateToAddStudent({Map<String, dynamic>? student}) async {
28     await Navigator.push(
29       context,
30       MaterialPageRoute(
31         builder: (context) => AddStudentFormScreen(student: student),
32       ),
33     );
34     _loadStudents();
35   }
36
37   @override
38   Widget build(BuildContext context) {
39     return Scaffold(
40       appBar: AppBar(
41         title: Text('SQLite Biodata Mahasiswa'),
42         backgroundColor: Colors.amber,
43       ),
44       body: ListView.builder(
45         itemCount: _students.length,
46         itemBuilder: (context, index) {
47           final student = _students[index];
48           return Card(
49             margin: const EdgeInsets.symmetric(vertical: 5, horizontal: 10),
50             child: ListTile(
51               title: Text(student['name']),
52               subtitle: Text(
53                 'NIM: ${student['nim']}\nAlamat: ${student['address']}\nHobi: ${student['hobby']}',
54               ),
55               isThreeLine: true,
56               trailing: Row(
57                 mainAxisAlignment: MainAxisAlignment.min,
58                 children: [
59                   IconButton(
60                     icon: Icon(Icons.edit, color: Colors.blue),
61                     onPressed: () => _navigateToAddStudent(student: student),
62                   ),
63                 ],
64               ),
65             ),
66           );
67         },
68       ),
69       floatingActionButton: FloatingActionButton(
70         onPressed: () => _navigateToAddStudent(),
71         child: Icon(Icons.add),
72         backgroundColor: Colors.amber,
73       ),
74     );
75   }
76 }
77
```


Form screen

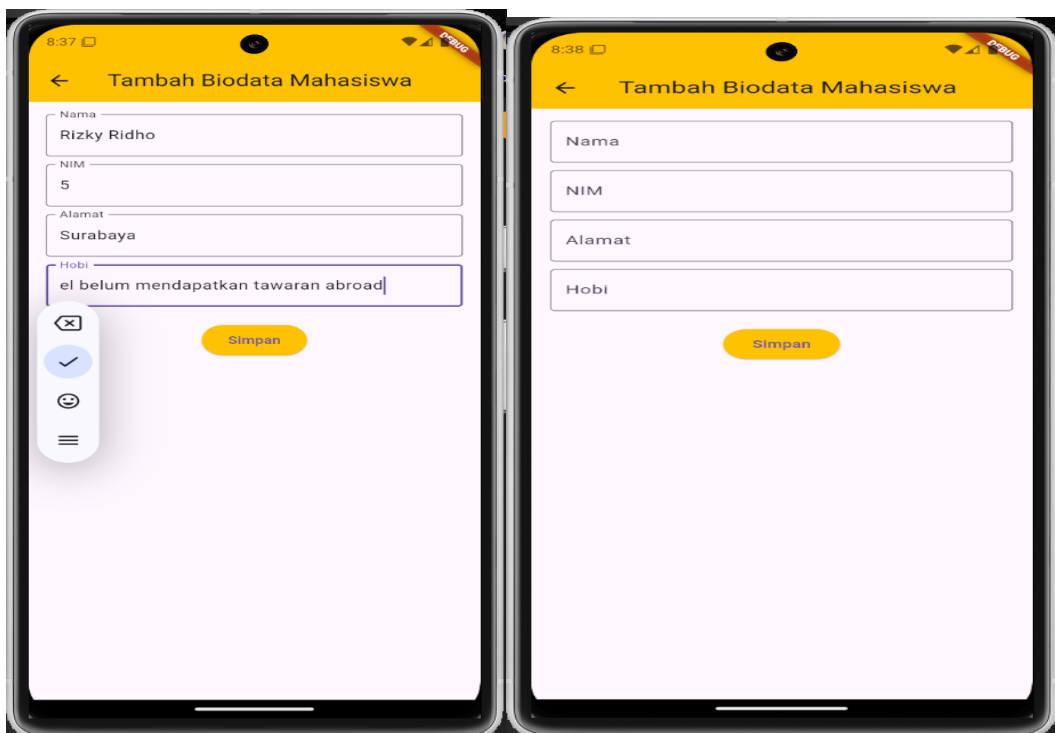
```
1 import 'package:flutter/material.dart';
2 import 'db_helper.dart';
3
4 class AddStudentFormScreen extends StatefulWidget {
5   final Map<String, dynamic>? student; // Optional parameter for editing
6
7   const AddStudentFormScreen({Key? key, this.student}) : super(key: key);
8
9   @override
10  _AddStudentFormScreenState createState() => _AddStudentFormScreenState();
11 }
12
13 class _AddStudentFormScreenState extends State<AddStudentFormScreen> {
14   final DatabaseHelper _dbHelper = DatabaseHelper();
15   final _nameController = TextEditingController();
16   final _nimController = TextEditingController();
17   final _addressController = TextEditingController();
18   final _hobbyController = TextEditingController();
19
20   @override
21   void initState() {
22     super.initState();
23     // If editing, pre-fill the fields
24     if (widget.student != null) {
25       _nameController.text = widget.student!['name'];
26       _nimController.text = widget.student!['nim'];
27       _addressController.text = widget.student!['address'];
28       _hobbyController.text = widget.student!['hobby'];
29     }
30   }
31
32   void _saveStudent() async {
33     if (widget.student == null) {
34       // Insert new student
35       await _dbHelper.insertStudent({
36         'name': _nameController.text,
37         'nim': _nimController.text,
38         'address': _addressController.text,
39         'hobby': _hobbyController.text,
40       });
41     } else {
42       // Update existing student
43       await _dbHelper.updateStudent({
44         'id': widget.student!['id'],
45         'name': _nameController.text,
46         'nim': _nimController.text,
47         'address': _addressController.text,
48         'hobby': _hobbyController.text,
49       });
50     }
51     Navigator.pop(context);
52   }
53
54   @override
55   Widget build(BuildContext context) {
56     return Scaffold(
57       appBar: AppBar(
58         title: Text(widget.student == null
59           ? 'Tambah Biodata Mahasiswa'
60           : 'Edit Biodata Mahasiswa'),
61         backgroundColor: Colors.amber,
62       ),
63       body: Padding(
64         padding: const EdgeInsets.all(16.0),
65         child: Column(
66           children: [
67             TextField(
68               controller: _nameController,
69               decoration: InputDecoration(
70                 labelText: 'Nama',
71                 border: OutlineInputBorder(),
72               ),
73             ),
74             SizedBox(height: 10),
75             TextField(
76               controller: _nimController,
77               decoration: InputDecoration(
78                 labelText: 'NIM',
79                 border: OutlineInputBorder(),
80               ),
81             ),
82             SizedBox(height: 10),
83             TextField(
84               controller: _addressController,
85               decoration: InputDecoration(
86                 labelText: 'Alamat',
87                 border: OutlineInputBorder(),
88               ),
89             ),
90             SizedBox(height: 10),
91             TextField(
92               controller: _hobbyController,
93               decoration: InputDecoration(
94                 labelText: 'Hobi',
95                 border: OutlineInputBorder(),
96               ),
97             ),
98             SizedBox(height: 20),
99             ElevatedButton(
100               onPressed: _saveStudent,
101               child: Text(widget.student == null ? 'Simpan' : 'Update'),
102               style: ElevatedButton.styleFrom(backgroundColor: Colors.amber),
103             ),
104           ],
105         ),
106       ),
107     );
108   }
109 }
110
```

Screenshoot Output

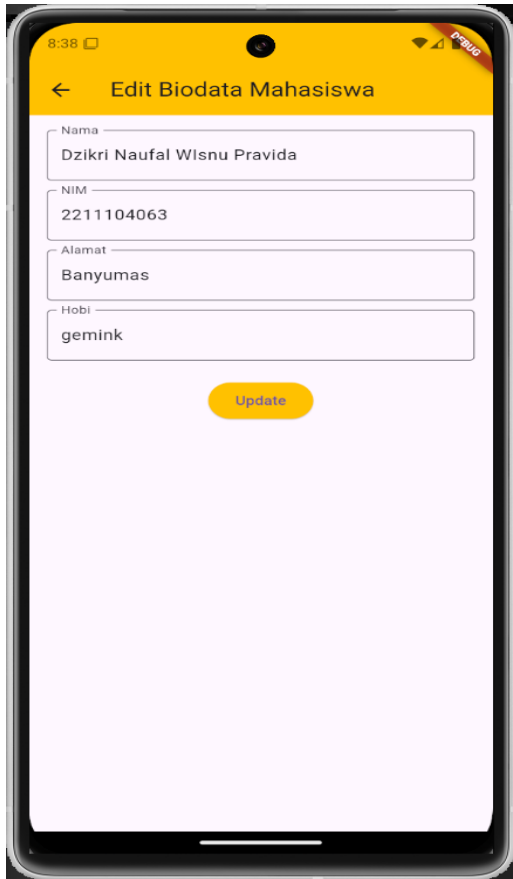
Tampilan utama



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.