

GUIDED & UNGUIDED
PRAKTIKUM PEMROGRAMAN PERANGKAT BERGERAK
MODUL X
DATA STORAGE BAGIAN 1



Disusun Oleh :

Maria Nathasya Desfera Pangestu / 2211104008

SE0601

Asisten Praktikum :

Muhammad Faza Zulian Gesit Al Barru

Aisyah Hasna Aulia

Dosen Pengampu :

Yudha Islami Sulistyia, S.Kom., M.Cs.

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

GUIDED

1. Pengenalan SQLite

SQLite adalah database relasional yang merupakan penyimpanan data secara offline untuk sebuah mobile app (pada local storage, lebih tepatnya pada cache memory aplikasi). SQLite memiliki CRUD (create, read, update dan delete). Empat operasi tersebut penting dalam sebuah penyimpanan. Untuk struktur database pada SQLite, sama seperti SQL pada umumnya, variabel dan tipe data yang dimiliki tidak jauh berbeda dengan SQL.

2. SQL Helper Dasar

Dalam Flutter, SQL Helper biasanya merujuk pada penggunaan paket seperti sqflite untuk mengelola database SQLite. SQL Helper merupakan class untuk membuat beberapa method yang berkaitan dengan perubahan data. sqflite adalah plugin Flutter yang memungkinkan untuk melakukan operasi CRUD (Create, Read, Update, Delete) pada database SQLite.

3. Source code db_helper.dart

```
 1 import 'package:sqflite/sqflite.dart';
 2 import 'package:path/path.dart';
 3
 4 //Kelas database untuk mengelola database
 5 class DatabaseHelper {
 6   static final DatabaseHelper _instance = DatabaseHelper._internal();
 7   static Database? _database;
 8
 9   // factory constructor untuk mengembalikan instance singleton
10   factory DatabaseHelper() {
11     return _instance;
12   }
13
14   // Private constructor
15   DatabaseHelper._internal();
16
17   //Getter untuk database
18   Future<Database> get database async {
19     if (_database != null) return _database!;
20     {
21       _database = await _initDatabase();
22       return _database!;
23     }
24   }
25
26   //Inisialisasi database
27   Future<Database> _initDatabase() async {
28     // mendapatkan path untuk database
29     String path = join(await getDatabasesPath(), 'my_prakdatabase.db');
30     // membuka database
31     return await openDatabase(
32       path,
33       version: 1,
34       onCreate: _onCreate,
35     );
36   }
37
38   //Membuat tabel db dengan record dan value id, title, description
39   Future<void> _onCreate(Database db, int version) async {
40     await db.execute('''
41 CREATE TABLE my_table(
42   id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
43   title TEXT,
44   description TEXT,
45   createdAt TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP)
46 ''');
47   }
48
49   // metode untuk mengambil semua data dari tabel
50   Future<int> insert(Map<String, dynamic> row) async {
51     Database db = await database;
52     return await db.insert('my_table', row);
53   }
54
55   // metode untuk memperbarui data dalam tabel
56   Future<int> update(Map<String, dynamic> row) async {
57     Database db = await database;
58     int id = row['id'];
59     return await db.update('my_table', row, where: 'id = ?', whereArgs: [id]);
60   }
61
62   // metode untuk menghapus data dari tabel
63   Future<int> delete(int id) async {
64     Database db = await database;
65     return await db.delete('my_table', where: 'id = ?', whereArgs: [id]);
66   }
67
68   //Membaca semua data
69   Future<List<Map<String, dynamic>>> queryAllRows() async {
70     Database db = await database;
71     return await db.query('my_table');
72   }
73 }
```

4. Source code main.dart



```
1 import 'package:flutter/material.dart';
2 import 'package:guided/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   @override
12   Widget build(BuildContext context) {
13     return MaterialApp(
14       title: 'Database Storage',
15       theme: ThemeData(
16         colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
17         useMaterial3: true,
18       ),
19       home: MyDatabaseView(),
20     );
21   }
22 }
```

5. Source code my_db_view.dart

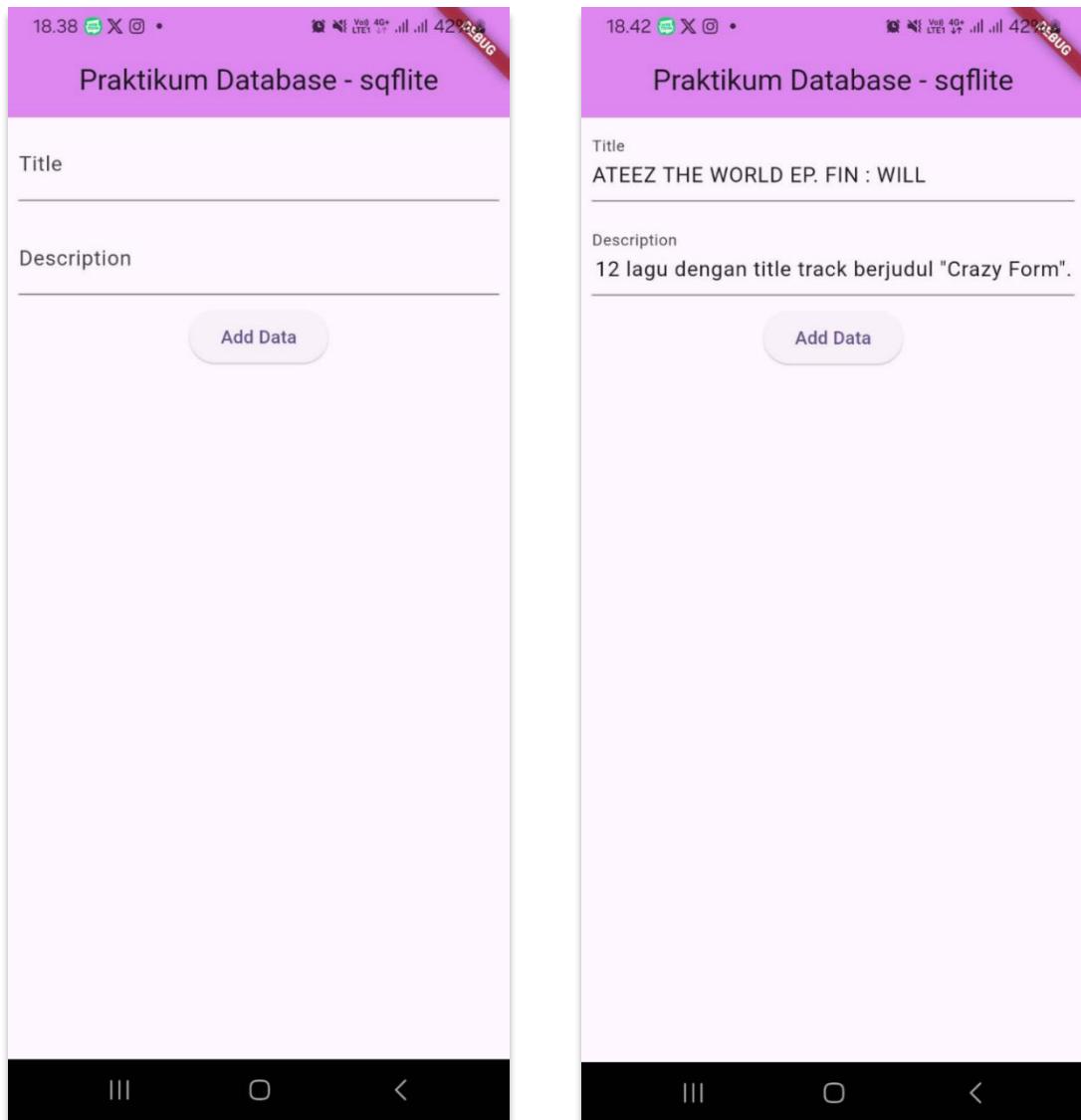
```
 1 import 'package:flutter/material.dart';
 2 import 'package:guided/helper/db_helper.dart';
 3
 4 class MyDatabaseView extends StatefulWidget {
 5   const MyDatabaseView({super.key});
 6
 7   @override
 8   State<MyDatabaseView> createState() => _MyDatabaseViewState();
 9 }
10
11 class _MyDatabaseViewState 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   void _refreshData() async {
31     final data = await dbHelper.queryAllRows();
32     setState(() {
33       _dbData = data;
34     });
35   }
36
37   void _addData() async {
38     if (_titleController.text.isEmpty || _descriptionController.text.isEmpty) {
39       _showSnackbar('Title and Description cannot be empty!');
40       return;
41     }
42     await dbHelper.insert({
43       'title': _titleController.text,
44       'description': _descriptionController.text,
45     });
46     _titleController.clear();
47     _descriptionController.clear();
48     _refreshData();
49   }
50
51   void _updateData(int id) async {
52     if (_titleController.text.isEmpty || _descriptionController.text.isEmpty) {
53       _showSnackbar('Title and Description cannot be empty!');
54       return;
55     }
56     await dbHelper.update({
57       'id': id,
58       'title': _titleController.text,
59       'description': _descriptionController.text,
60     });
61     _titleController.clear();
62     _descriptionController.clear();
63     _refreshData();
64   }
65
66   void _deleteData(int id) async {
67     await dbHelper.delete(id);
68     _refreshData();
69   }
70
71   void _showEditDialog(Map<String, dynamic> item) {
72     _titleController.text = item['title'];
73     _descriptionController.text = item['description'];
74 }
```

```
74
75     showDialog(
76         context: context,
77         builder: (context) {
78             return AlertDialog(
79                 title: const Text('Edit Item'),
80                 content: Column(
81                     mainAxisSize: MainAxisSize.min,
82                     children: [
83                         TextField(
84                             controller: _titleController,
85                             decoration: const InputDecoration(labelText: 'Title'),
86                         ),
87                         TextField(
88                             controller: _descriptionController,
89                             decoration: const InputDecoration(labelText: 'Description'),
90                         ),
91                     ],
92                 ),
93                 actions: [
94                     TextButton(
95                         onPressed: () {
96                             Navigator.of(context).pop();
97                         },
98                         child: const Text('Cancel'),
99                     ),
100                    TextButton(
101                        onPressed: () {
102                            _updateData(item['id']);
103                            Navigator.of(context).pop();
104                        },
105                        child: const Text('Save'),
106                    ),
107                ],
108            );
109        },
110    );
111 }
112
113 void _showSnackbar(String message) {
114     ScaffoldMessenger.of(context)
115         .showSnackBar(SnackBar(content: Text(message)));
116 }
117
```

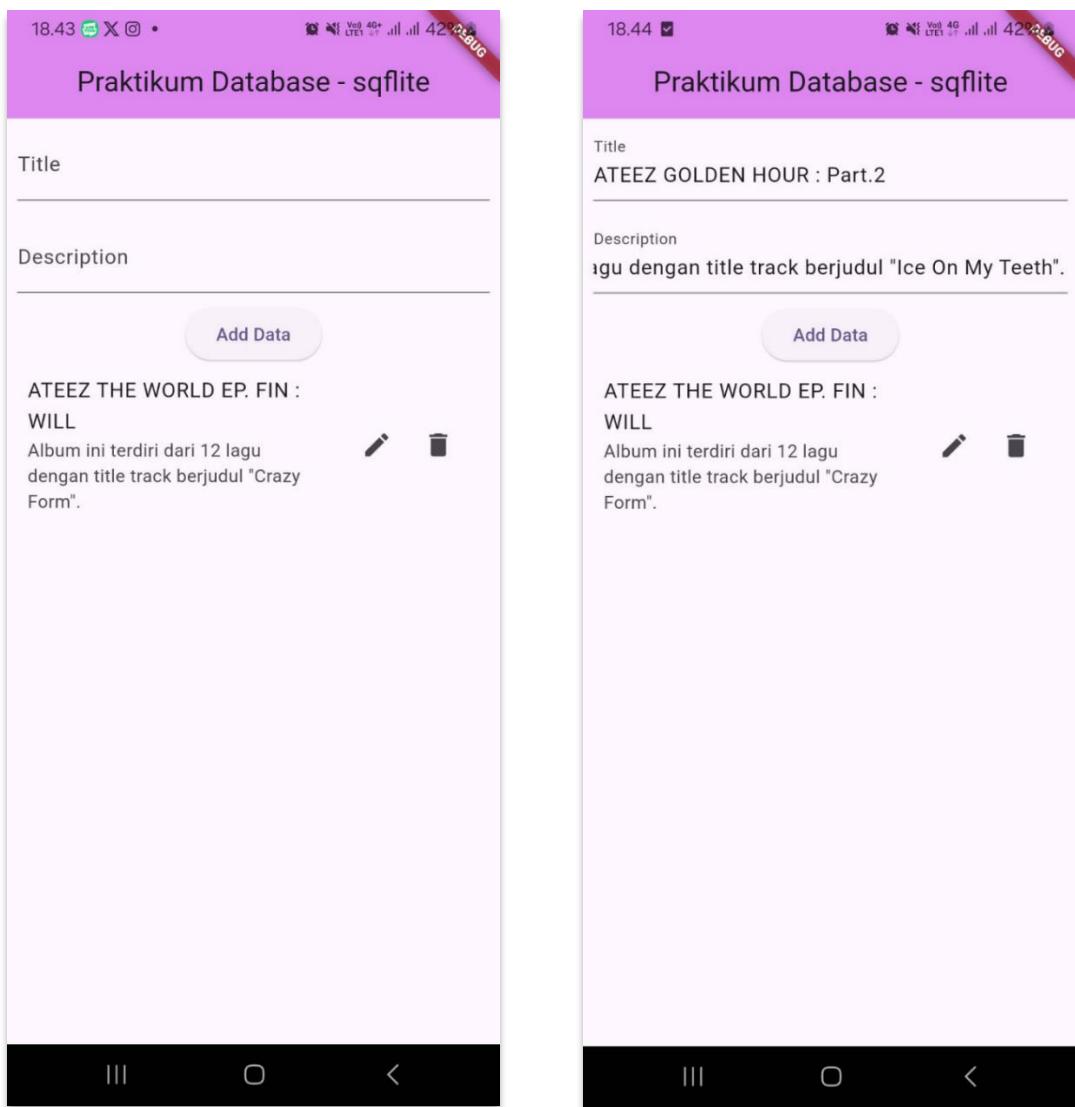
```
117
118 @override
119 Widget build(BuildContext context) {
120   return Scaffold(
121     appBar: AppBar(
122       title: const Text('Praktikum Database - sqflite'),
123       backgroundColor: const Color.fromARGB(255, 222, 135, 240),
124       centerTitle: true,
125     ),
126     body: Column(
127       children: [
128         Padding(
129           padding: const EdgeInsets.all(8.0),
130           child: TextField(
131             controller: _titleController,
132             decoration: const InputDecoration(labelText: 'Title'),
133           ),
134         ),
135         Padding(
136           padding: const EdgeInsets.all(8.0),
137           child: TextField(
138             controller: _descriptionController,
139             decoration: const InputDecoration(labelText: 'Description'),
140           ),
141         ),
142         ElevatedButton(
143           onPressed: _addData,
144           child: const Text('Add Data'),
145         ),
146         Expanded(
147           child: ListView.builder(
148             itemCount: _dbData.length,
149             itemBuilder: (context, index) {
150               final item = _dbData[index];
151               return ListTile(
152                 title: Text(item['title']),
153                 subtitle: Text(item['description']),
154                 trailing: Row(
155                   mainAxisAlignment: MainAxisAlignment.end,
156                   children: [
157                     IconButton(
158                       icon: const Icon(Icons.edit),
159                       onPressed: () {
160                         _showEditDialog(item);
161                       },
162                     ),
163                     IconButton(
164                       icon: const Icon(Icons.delete),
165                       onPressed: () => _deleteData(item['id']),
166                     ),
167                   ],
168                 ),
169               );
170             },
171           ),
172         ),
173       ],
174     ),
175   );
176 }
177 }
```

6. Hasil Output

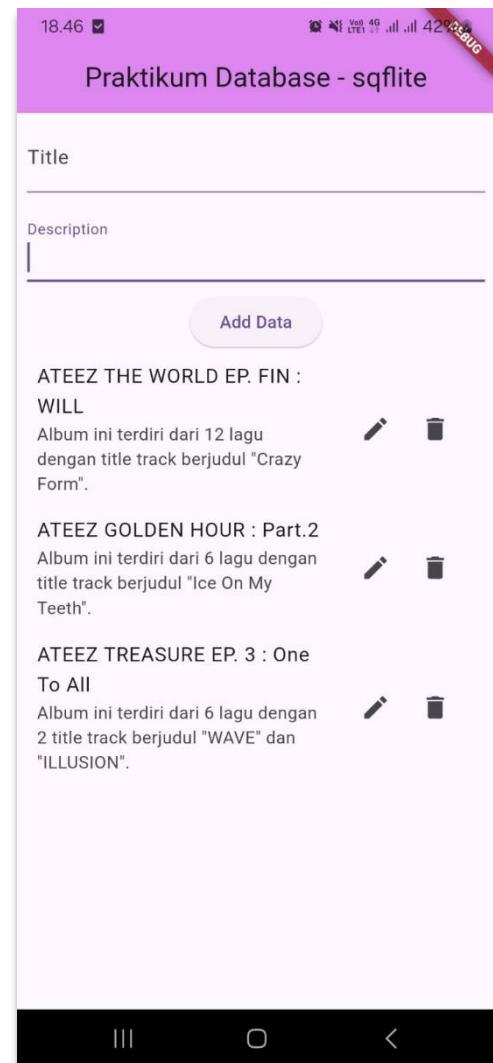
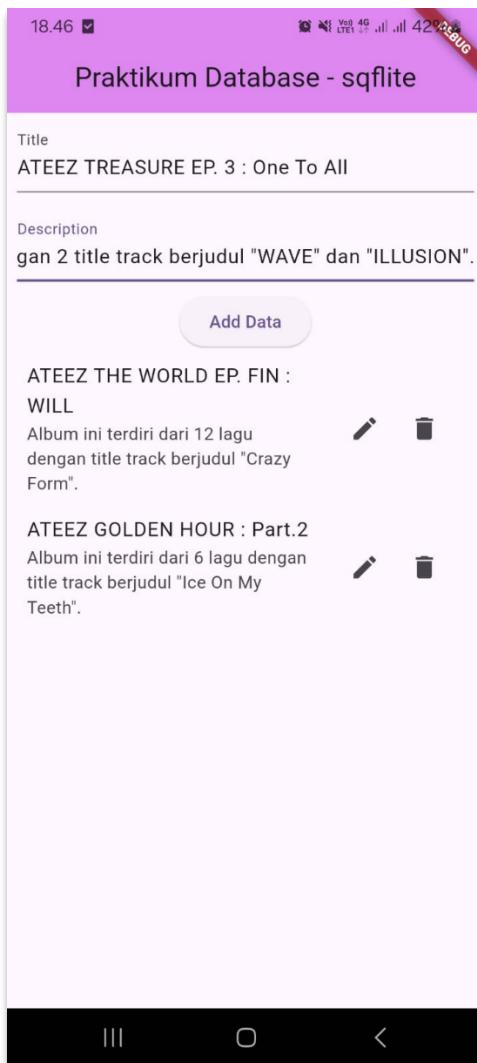
Tampilan awal dan saat menambahkan data



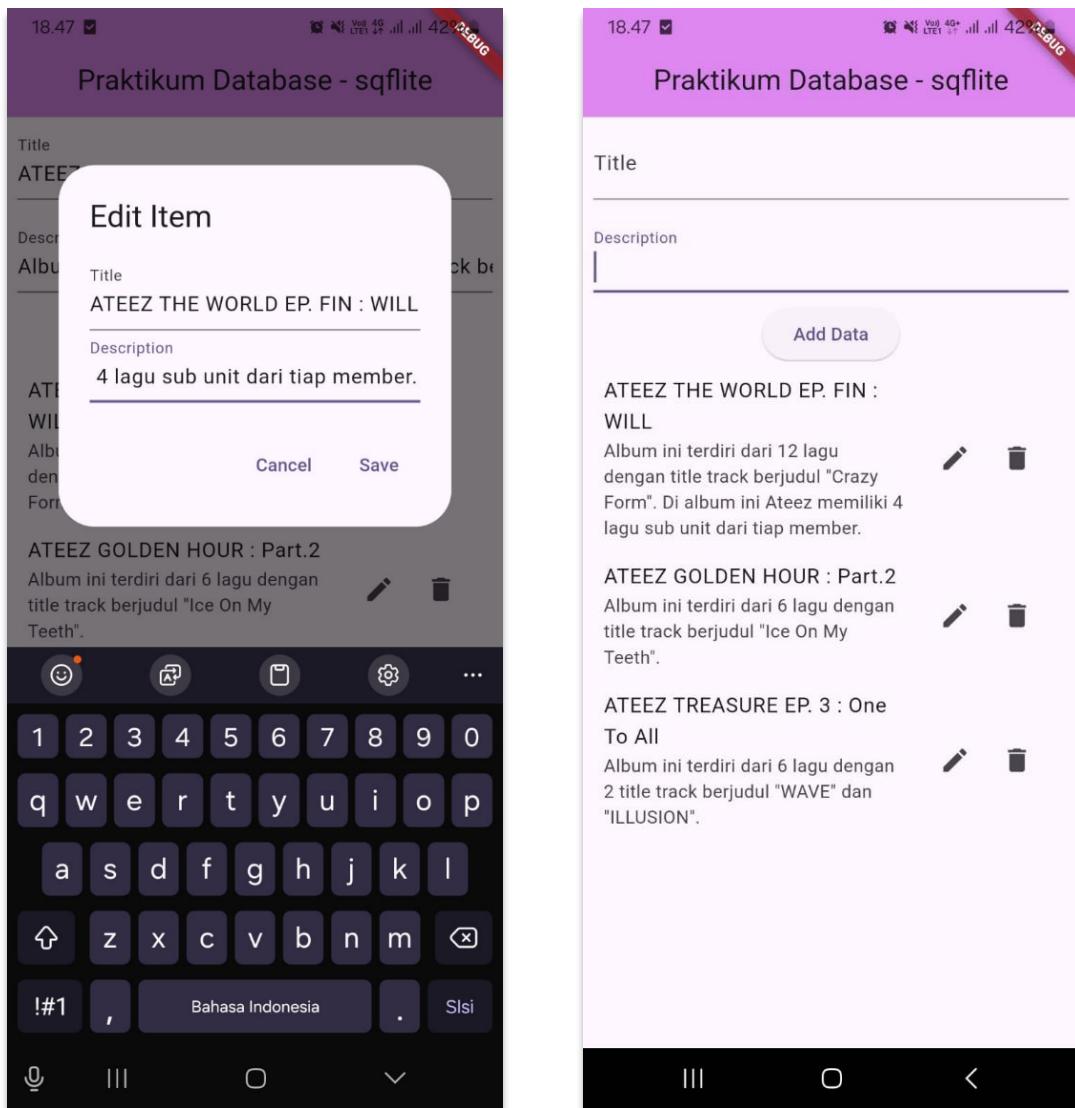
Tampilan sesudah mengklik “Add Data” dan menambahkan data baru



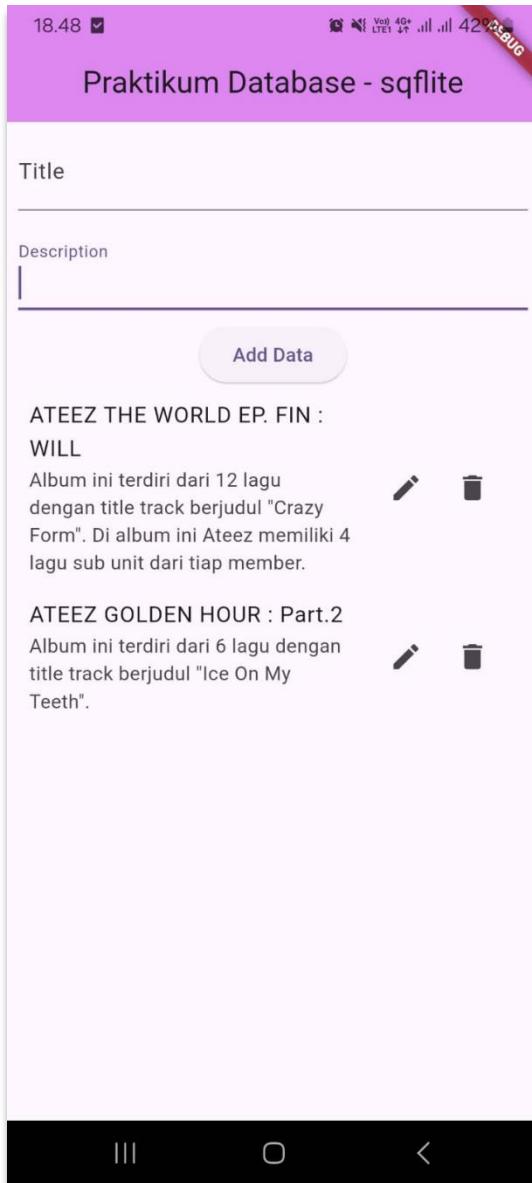
Tampilan sesudah menambahkan data kedua dan ketiga



Tampilan saat mengedit data dan setelah data diedit



Tampilan akhir setelah salah satu data dihapus



7. Penjelasan

Kode Flutter ini berfungsi untuk menampilkan data dari sebuah database. Dua library penting yang diimpor pada awal kode adalah flutter/material.dart menyediakan berbagai komponen untuk membuat antarmuka pengguna, lalu guided/view/my_db_view.dart sebagai tampilan utama aplikasi. Proses eksekusi dimulai dari fungsi main(), di mana kita menjalankan widget MyApp dan kelas MyApp mewakili seluruh aplikasi. Dalam metode build MaterialApp digunakan untuk membangun tampilan utama yang mengatur tema dasar aplikasi, seperti warna, font, dan juga menetapkan MyDatabaseView sebagai halaman utama. Kode programnya

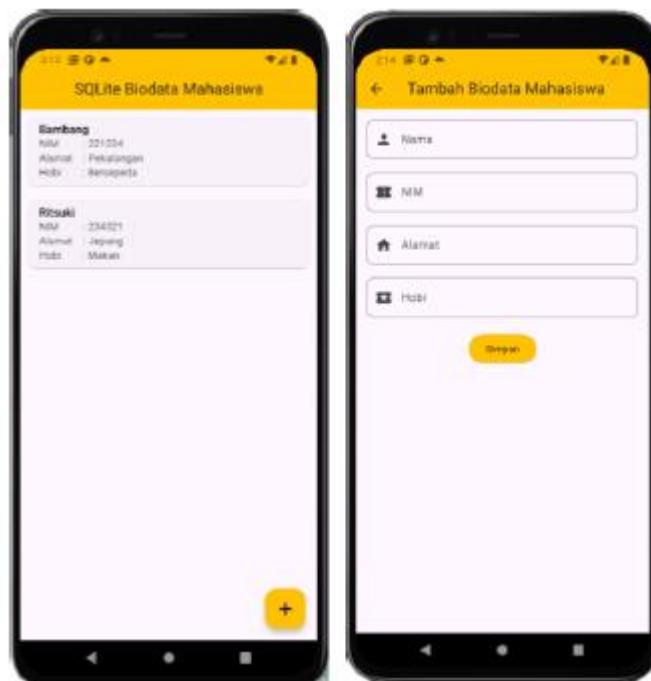
memungkinkan pengguna untuk bisa mengambil data(membaca data dari database dan menyimpannya untuk dilihat), menampilkan data (menampilkan data dalam format yang mudah dibaca oleh pengguna, seperti daftar, tabel, atau grafik), memungkinkan interaksi (seperti menambahkan, mengedit, atau menghapus data dari database).

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:

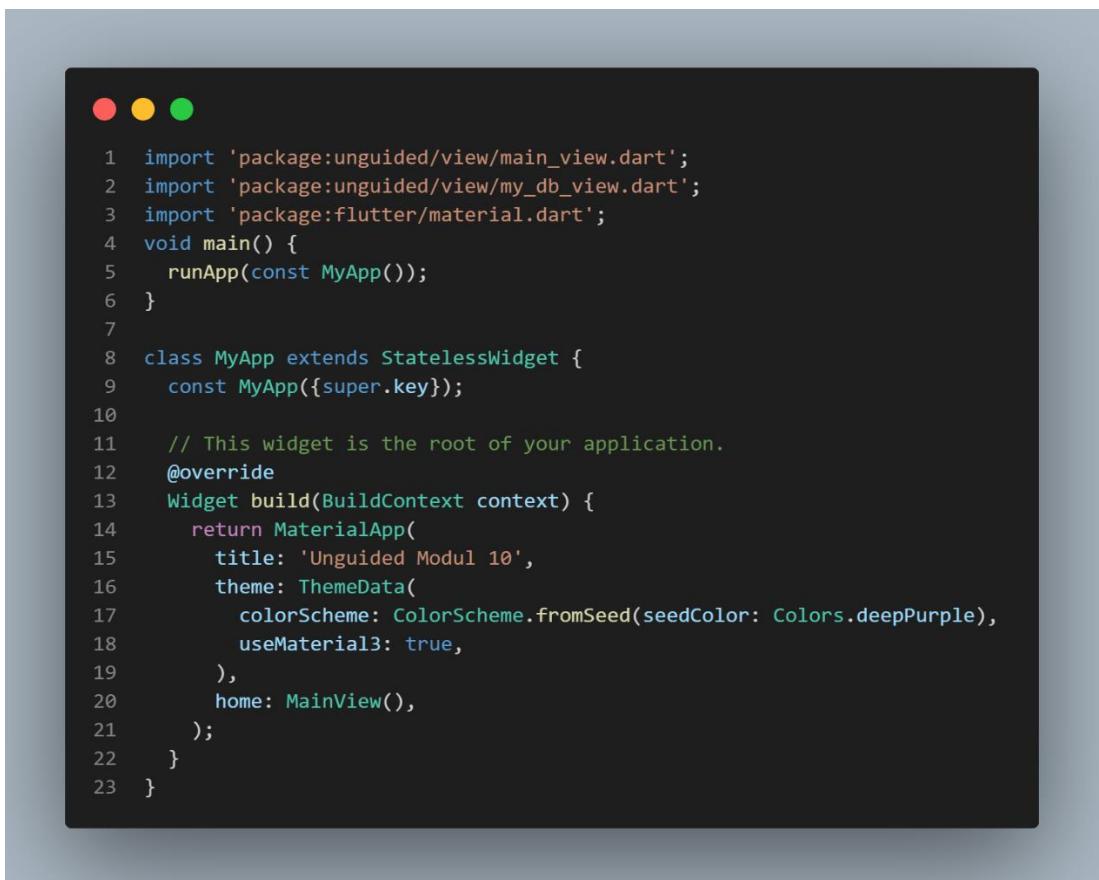


Note: Jangan lupa sertakan source code, screenshot output, dan deskripsi program. Kreatifitas menjadi nilai tambah.

Jawab

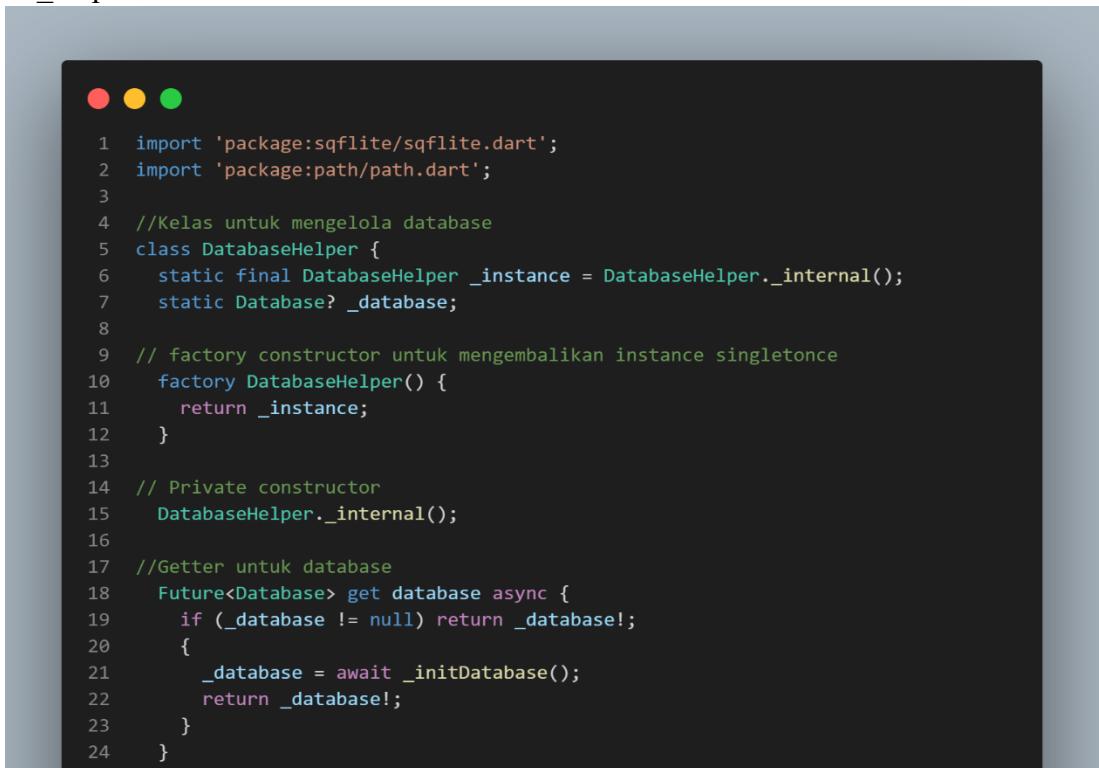
Source code

➤ Main.dart



```
1 import 'package:unguided/view/main_view.dart';
2 import 'package:unguided/view/my_db_view.dart';
3 import 'package:flutter/material.dart';
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: 'Unguided Modul 10',
16      theme: ThemeData(
17        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
18        useMaterial3: true,
19      ),
20      home: MainView(),
21    );
22  }
23 }
```

➤ db_helper.dart



```
1 import 'package:sqflite/sqflite.dart';
2 import 'package:path/path.dart';
3
4 //Kelas untuk mengelola database
5 class DatabaseHelper {
6   static final DatabaseHelper _instance = DatabaseHelper._internal();
7   static Database? _database;
8
9   // factory constructor untuk mengembalikan instance singleton
10  factory DatabaseHelper() {
11    return _instance;
12  }
13
14  // Private constructor
15  DatabaseHelper._internal();
16
17  //Getter untuk database
18  Future<Database> get database async {
19    if (_database != null) return _database!;
20    {
21      _database = await _initDatabase();
22      return _database!;
23    }
24  }
25 }
```

```
24
25     //Inisialisasi database
26     Future<Database> _initDatabase() async {
27         // mendapatkan path untuk database
28         String path = join(await getDatabasesPath(), 'my_prakdatabase.db');
29         // membuka database
30         return await openDatabase(
31             path,
32             version: 2,
33             onCreate: _onCreate,
34             // Ini untuk menangani upgrade
35             onUpgrade: _onUpgrade,
36             );
37         }
38     }
39
40     //Membuat tabel db dengan record dan value id, nama, nim
41     Future<void> _onCreate(Database db, int version) async {
42         await db.execute('''
43             CREATE TABLE my_table(
44                 id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
45                 nama TEXT,
46                 nim TEXT,
47                 alamat TEXT,
48                 hobi TEXT,
49                 createdAt TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP)
50             ''');
51     }
52 }
```

```
52
53     Future<void> _onUpgrade(Database db, int oldVersion, int newVersion) async {
54         if (oldVersion < 2) {
55             // Menambahkan kolom alamat dan hobi jika belum ada
56             await db.execute('ALTER TABLE my_table ADD COLUMN alamat TEXT');
57             await db.execute('ALTER TABLE my_table ADD COLUMN hobi TEXT');
58         }
59     }
60
61     // metode untuk mengambil semua data dari tabel
62     Future<int> insert(Map<String, dynamic> row) async {
63         Database db = await database;
64         return await db.insert('my_table', row);
65     }
66
67     // metode untuk memperbarui data dalam tabel
68     Future<int> update(Map<String, dynamic> row) async {
69         Database db = await database;
70         int id = row['id'];
71         return await db.update('my_table', row, where: 'id = ?', whereArgs: [id]);
72     }
73
74     // metode untuk menghapus data dari tabel
75     Future<int> delete(int id) async {
76         Database db = await database;
77         return await db.delete('my_table', where: 'id = ?', whereArgs: [id]);
78     }
79
80     //Membaca semua data
81     Future<List<Map<String, dynamic>>> queryAllRows() async {
82         Database db = await database;
83         return await db.query('my_table');
84     }
85 }
```

➤ Main_view.dart

```
 1 import 'package:flutter/material.dart';
 2 import 'package:unguided/helper/db_helper.dart';
 3 import 'package:unguided/view/my_db_view.dart';
 4
 5 class MainView extends StatefulWidget {
 6   const MainView({super.key});
 7
 8   @override
 9   State<MainView> createState() => _MainViewState();
10 }
11
12 class _MainViewState extends State<MainView> {
13   final DatabaseHelper dbHelper = DatabaseHelper();
14   List<Map<String, dynamic>> _dbData = [];
15
16   @override
17   void initState() {
18     _refreshData();
19     super.initState();
20   }
21
22   void _refreshData() async {
23     final data = await dbHelper.queryAllRows();
24     setState(() {
25       _dbData = data;
26     });
27   }
28
29 //untuk menghapus data
30 void _deleteData(int id) async {
31   await dbHelper.delete(id);
32   _refreshData();
33 }
34
```

```
34
35     //untuk mengedit data
36     void _editData(Map<String, dynamic> item) {
37         // Panggil MyDatabaseView dengan mode edit
38         Navigator.push(
39             context,
40             MaterialPageRoute(
41                 builder: (context) =>
42                     MyDatabaseView(item: item), // Mengirim data untuk diedit
43             ),
44         ).then((_) {
45             _refreshData(); // Merefresh data setelah edit
46         });
47     }
48
49     @override
50     Widget build(BuildContext context) {
51         return Scaffold(
52             appBar: AppBar(
53                 title: const Text('SQLite Biodata Mahasiswa'),
54                 backgroundColor: const Color.fromARGB(255, 222, 135, 240),
55                 centerTitle: true,
56             ),
57             body: _dbData.isEmpty
58                 ? Center(
59                     child: Text(
60                         'Belum ada data mahasiswa',
61                         style: TextStyle(
62                             fontSize: 15,
63                             color: Colors.grey,
64                         ),
65                     ),
66                 )
67                 : ListView.builder(
68                     padding: const EdgeInsets.all(8.0),
69                     itemCount: _dbData.length,
70                     itemBuilder: (context, index) {
71                         final item = _dbData[index];
72                         return Card(
73                             margin: const EdgeInsets.symmetric(vertical: 8),
74                             child: ListTile(
75                                 title: Text(
76                                     item['nama'],
77                                     style: const TextStyle(
78                                         fontWeight: FontWeight.bold, fontSize: 18),
79                                 ),
80                                 subtitle: Column(
81                                     crossAxisAlignment: CrossAxisAlignment.start,
82                                     children: [
83                                         Text('NIM: ${item['nim']}'),
84                                         Text('Alamat: ${item['alamat']}'),
85                                         Text('Hobi: ${item['hobi']}'),
86                                     ],
87                                 ),
88                                 trailing: Row(
89                                     mainAxisAlignment: MainAxisAlignment.end,
90                                     children: [
91                                         // Ikon pensil untuk edit
92                                         IconButton(
93                                             icon: const Icon(Icons.edit),
94                                             onPressed: () =>
95                                                 _editData(item), // Panggil fungsi edit
96                                         ),
97                                     ],
98                                 ),
99                             ),
100                         );
101                     });
102                 );
103             );
104         );
105     }
106 }
```

```
    // 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: const Color.fromARGB(255, 222, 135, 240),
  child: const Icon(Icons.add),
),
);
}
}
```

➤ My_db_view.dart

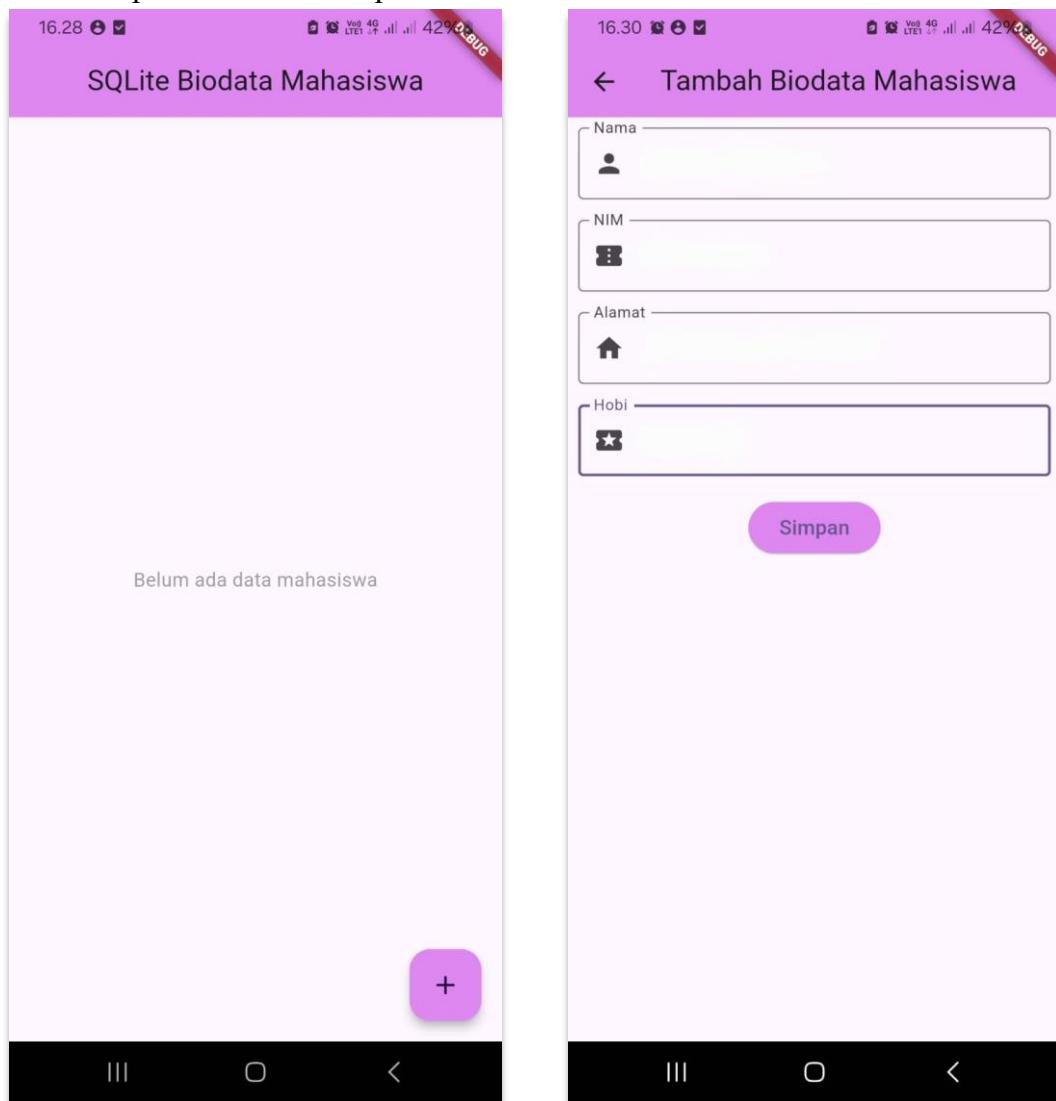
```
1 import 'package:flutter/material.dart';
2 import 'package:unguided/helper/db_helper.dart';
3
4 class MyDatabaseView extends StatefulWidget {
5   final Map<String, dynamic>? item; // Menambahkan parameter item untuk edit
6
7   const MyDatabaseView({super.key, this.item});
8
9   @override
10  State<MyDatabaseView> createState() => _MyDatabaseViewState();
11 }
12
13 class _MyDatabaseViewState extends State<MyDatabaseView> {
14   final DatabaseHelper dbHelper = DatabaseHelper();
15   final TextEditingController _namaController = TextEditingController();
16   final TextEditingController _nimController = TextEditingController();
17   final TextEditingController _alamatController = TextEditingController();
18   final TextEditingController _hobiController = TextEditingController();
19
20   @override
21   void initState() {
22     super.initState();
23     if (widget.item != null) {
24       // Jika data ada (edit mode), masukkan data ke dalam controller
25       _namaController.text = widget.item?['nama'] ?? '';
26       _nimController.text = widget.item?['nim'] ?? '';
27       _alamatController.text = widget.item?['alamat'] ?? '';
28       _hobiController.text = widget.item?['hobi'] ?? '';
29     }
30   }
31
32   @override
33   void dispose() {
34     _namaController.dispose();
35     _nimController.dispose();
36     _alamatController.dispose();
37     _hobiController.dispose();
38     super.dispose();
39   }
}
```

```
40
41     void _saveData() async {
42         if (_namaController.text.isEmpty ||
43             _nimController.text.isEmpty ||
44             _alamatController.text.isEmpty ||
45             _hobiController.text.isEmpty) {
46             _showSnackbar('Tidak boleh kosong!');
47             return;
48         }
49
50         if (widget.item == null) {
51             // Jika item null berarti tambah data
52             await dbHelper.insert({
53                 'nama': _namaController.text,
54                 'nim': _nimController.text,
55                 'alamat': _alamatController.text,
56                 'hobi': _hobiController.text,
57             });
58         } else {
59             // Jika item tidak null berarti edit data
60             await dbHelper.update({
61                 'id': widget.item?['id'],
62                 'nama': _namaController.text,
63                 'nim': _nimController.text,
64                 'alamat': _alamatController.text,
65                 'hobi': _hobiController.text,
66             });
67         }
68
69         Navigator.pop(context);
70     }
71
72     void _showSnackbar(String message) {
73         ScaffoldMessenger.of(context)
74             .showSnackBar(SnackBar(content: Text(message)));
75     }
76
77     //Mendefinisikan fungsi buildText
78     Widget buildTextField({
79         required TextEditingController controller,
80         required String label,
81         required IconData icon,
82     }) {
83         return TextField(
84             controller: controller,
85             decoration: InputDecoration(
86                 labelText: label,
87                 prefixIcon: Icon(icon),
88                 border: OutlineInputBorder(),
89             ),
90         );
91     }
92 }
```

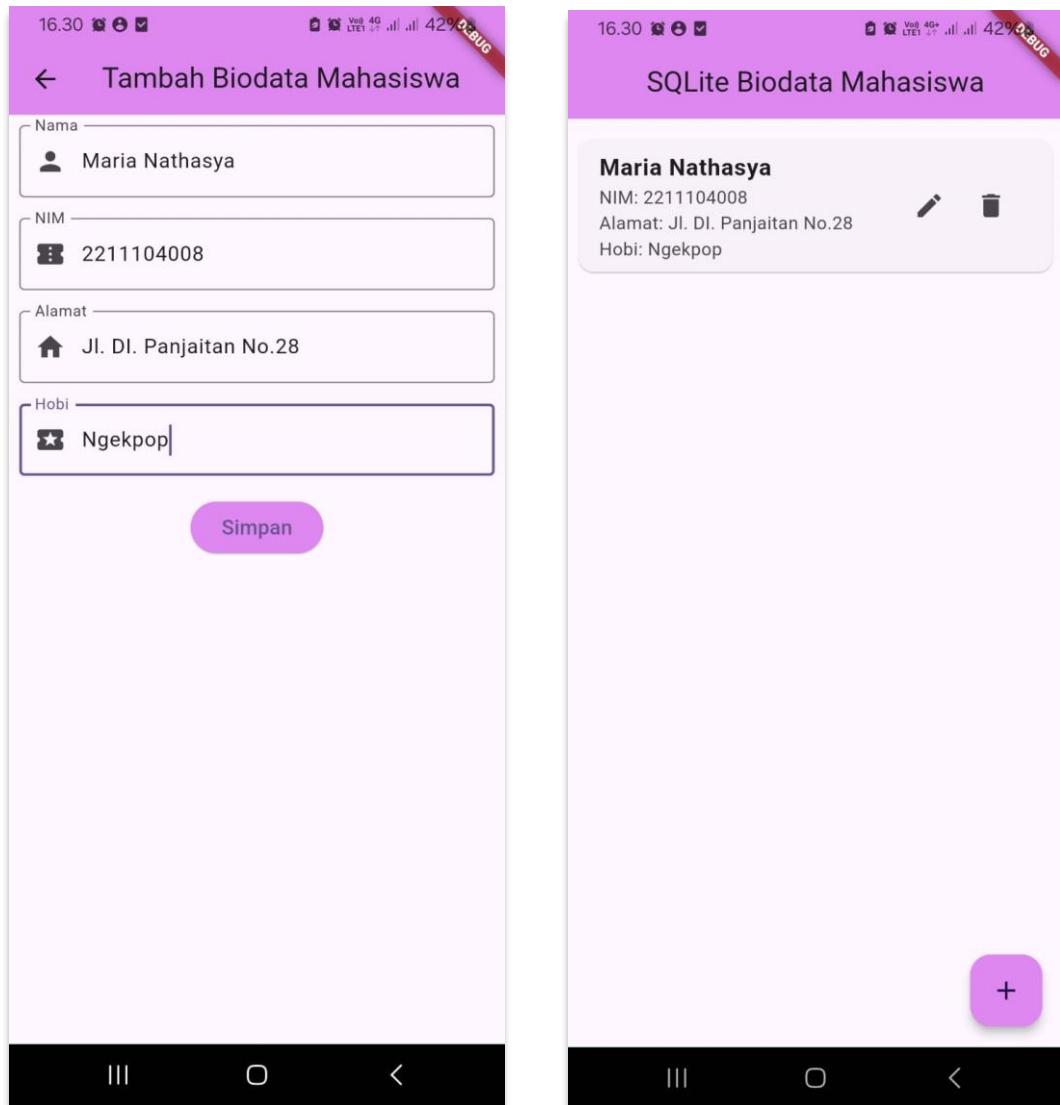
```
93     @override
94     Widget build(BuildContext context) {
95       return Scaffold(
96         appBar: AppBar(
97           title: Text(widget.item == null
98             ? 'Tambah Biodata Mahasiswa'
99             : 'Edit Biodata Mahasiswa'),
100          backgroundColor: const Color.fromARGB(255, 222, 135, 240),
101          centerTitle: true,
102        ),
103        body: Padding(
104          padding: const EdgeInsets.all(8.0),
105          child: Column(
106            children: [
107              buildTextField(
108                controller: _namaController,
109                label: 'Nama',
110                icon: Icons.person,
111              ),
112              const SizedBox(height: 16),
113              buildTextField(
114                controller: _nimController,
115                label: 'NIM',
116                icon: Icons.confirmation_num,
117              ),
118              const SizedBox(height: 16),
119              buildTextField(
120                controller: _alamatController,
121                label: 'Alamat',
122                icon: Icons.home,
123              ),
124              const SizedBox(height: 16),
125              buildTextField(
126                controller: _hobiController,
127                label: 'Hobi',
128                icon: Icons.local_activity,
129              ),
130              const SizedBox(height: 16),
131              ElevatedButton(
132                onPressed: _saveData,
133                child: const Text(
134                  'Simpan',
135                  style: TextStyle(fontSize: 16),
136                ),
137                style: ElevatedButton.styleFrom(backgroundColor: const Color.fromARGB(255, 222, 135, 240)),
138              ),
139            ],
140          ),
141        ),
142      );
143    }
144 }
```

Hasil Outputnya

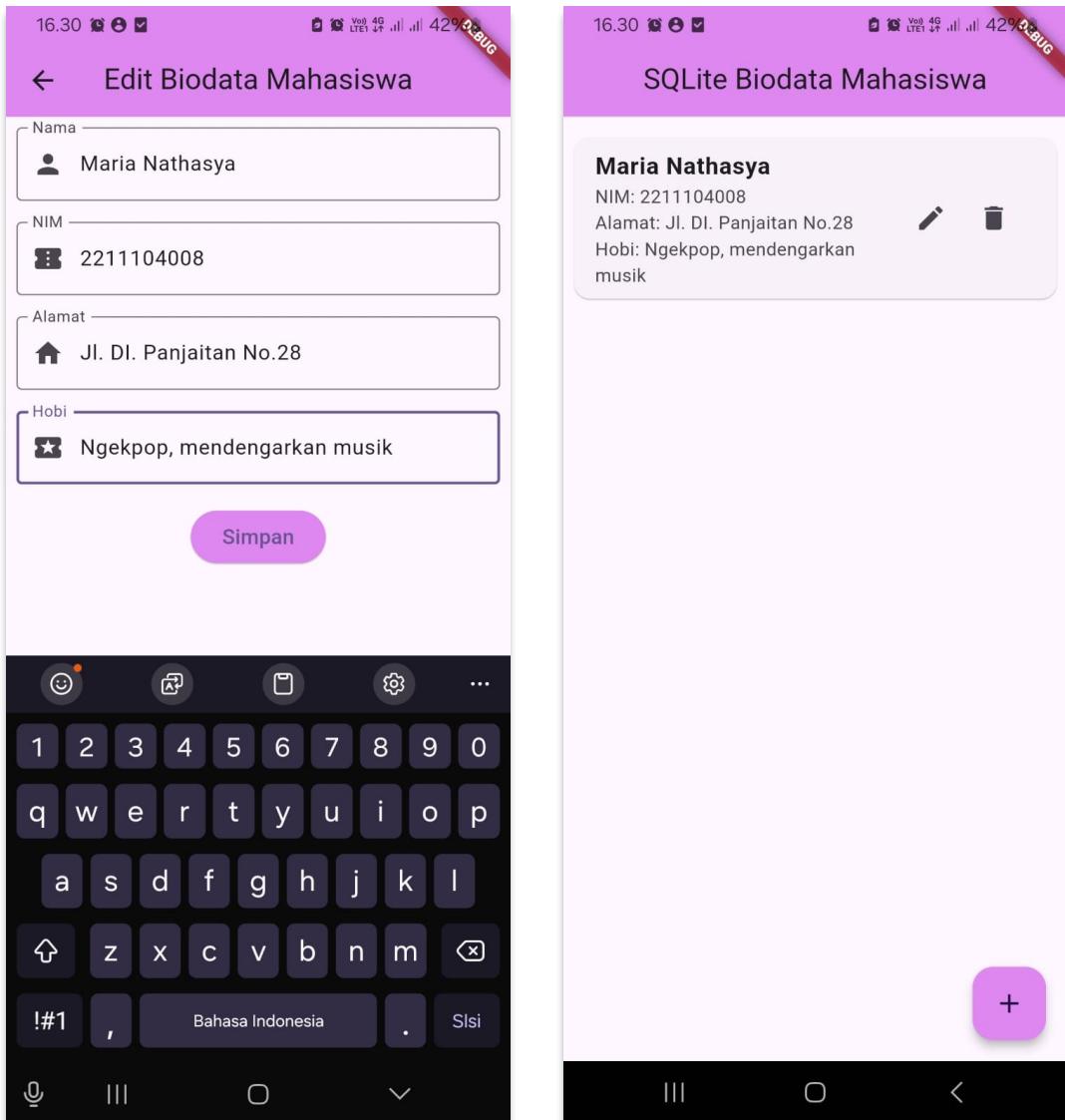
- Tampilan awal dan tampilan sebelum menambahkan biodata mahasiswa



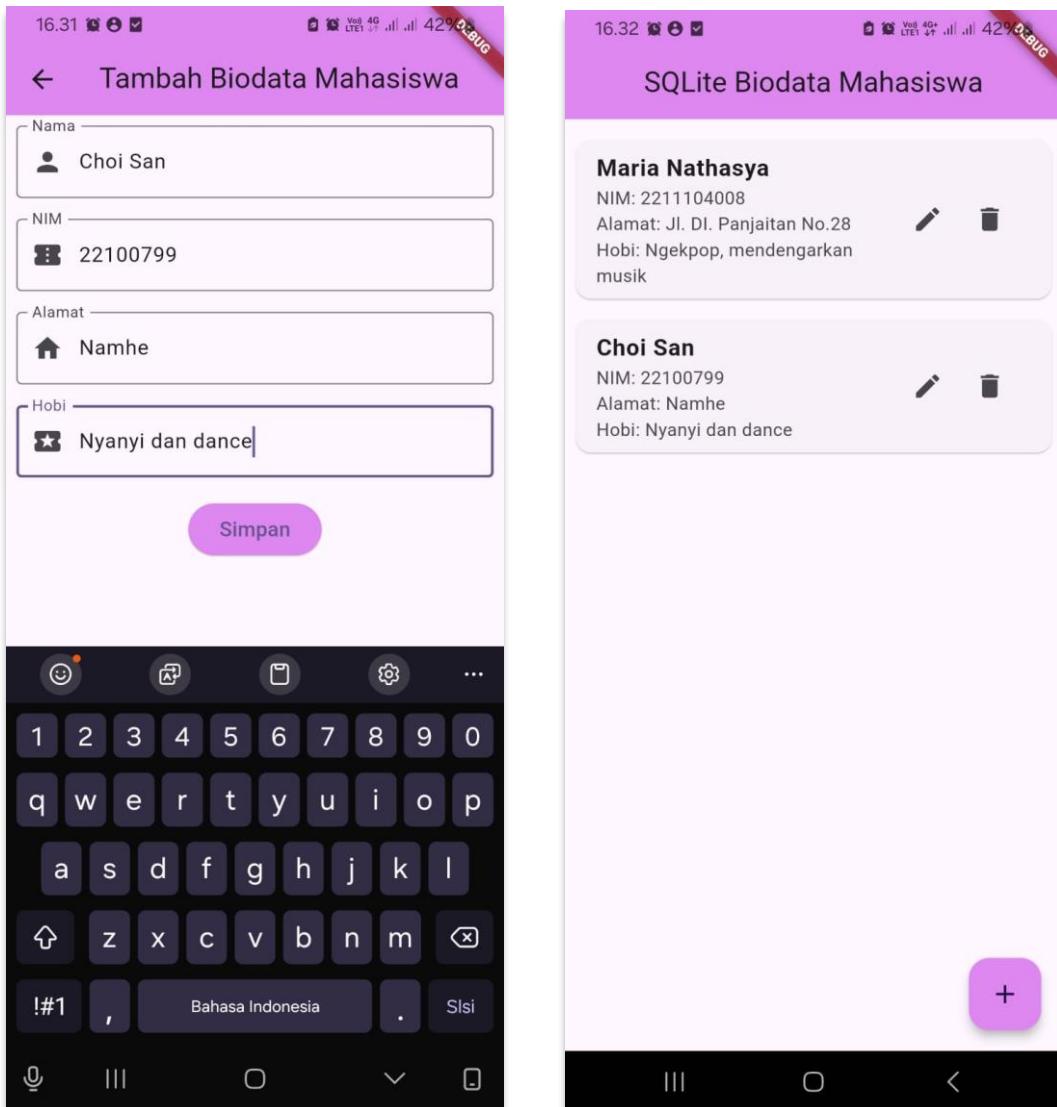
- Tampilan saat pengisian biodata mahasiswa dan tampilan setelah mengisi biodata mahasiswa



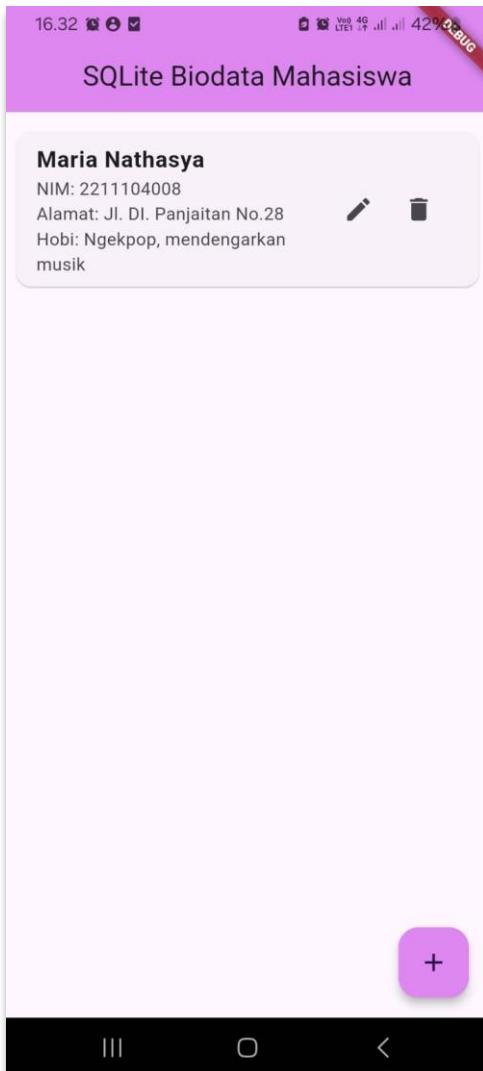
➤ Tampilan saat mengedit biodata mahasiswa dan hasil setelah biodata diedit



- Tampilan ketika menambah biodata mahasiswa dan tampilan setelah biodata baru ditambahkan



- Tampilan akhir setelah salah satu biodata mahasiswa dihapus



Penjelasan

Aplikasi Flutter ini menggunakan SQLite sebagai database lokal untuk mengelola data siswa. MainView menggunakan widget ListView untuk menampilkan daftar siswa dalam bentuk kartu. Fungsi `_refreshData` digunakan untuk mengambil data dari database, sedangkan `_deleteData` digunakan untuk menghapus data secara langsung. Setiap kartu memiliki tombol Edit untuk menuju ke halaman edit data (MyDatabaseView), dan tombol Delete untuk menghapus data secara langsung. Selain itu, ada tombol Action Button Floating yang dapat digunakan untuk menambah data baru, ini akan membuka halaman MyDatabaseView tanpa parameter.

Halaman MyDatabaseView memungkinkan kita menambah atau mengubah data siswa. Nama, NIM, alamat, dan hobi diatur menggunakan beberapa TextEditingController. Fungsi `initState` yaitu memasukkan data saat ini ke dalam form jika halaman dibuka dalam mode edit. Fungsi `_saveData` melakukan proses penyimpanan dan memeriksa apakah semua input terisi sebelum menyimpan data baru atau memperbarui data yang ada di database. Form input, tombol Simpan, dan judul dinamis yang berubah sesuai mode merupakan tampilan halaman.