

**TUGAS GUIDED
PEMROGRAMAN PERANGKAT BERGERAK**

**MODUL X
DATA STORAGE BAGIAN 1**



Disusun Oleh :
Althafia Defiyandrea Laskanadya Wibowo
2211104011 / SE06-01

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

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. SPL 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.

Main.dart

```
1 import 'package:flutter/material.dart';
2 import 'package:praktikum_10/view/my_db_view.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: 'Praktikum Data Storage',
13      theme: ThemeData(
14        primarySwatch: Colors.orange,
15      ),
16      home: MyDatabaseView(),
17    );
18  }
19 }
```

Db_helper.dart

```
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 constructor untuk mengembalikan instance singleton
9   factory DatabaseHelper() {
10     return _instance;
11   }
12
13   // Private constructor
14   DatabaseHelper._internal();
15 }
```

```

16 // Getter untuk database
17 Future<Database> get database async {
18   if (_database != null) return _database!;
19   {
20     _database = await _initDatabase();
21     return _database!;
22   }
23 }
24
25 // inisiasi 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: 1,
33     onCreate: _onCreate,
34   );
35 }
36
37 //membuat tabel saat db pertama kali dibuat
38 Future<void> _onCreate(Database db, int version) async {
39   await db.execute('''
40 CREATE TABLE my_table(
41 id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
42 title TEXT,
43 description TEXT,
44 createdAt TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP)
45 ''');
46 }
47
48 // metode memasukan data ke dalam tabel
49 Future<int> insert(Map<String, dynamic> row) async {
50   Database db = await database;
51   int result = await db.insert('my_table', row);
52   print('Inserted row: $result'); // Log untuk memeriksa hasil insert
53   return result;
54 }
55
56 // metode mengambil semua data dari tabel
57 Future<List<Map<String, dynamic>>> queryAllRows() async {
58   Database db = await database;
59   List<Map<String, dynamic>> result = await db.query('my_table');
60   print('Fetched data: $result'); // Log untuk memeriksa data yang diambil
61   return result;
62 }
63
64 // metode untuk memperbarui data dalam tabel
65 Future<int> update(Map<String, dynamic> row) async {
66   Database db = await database;
67   int id = row['id'];
68   return await db.update('my_table', row, where: 'id = ?', whereArgs: [id]);
69 }
70
71 // metode menghapus data dari tabel
72 Future<int> delete(int id) async {
73   Database db = await database;
74   return await db.delete('my_table', where: 'id = ?', whereArgs: [id]);
75 }
76 }

```

My_db_view

```
1 import 'package:flutter/material.dart';
2 import 'package:praktikum_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     super.initState();
20     _refreshData();
21   }
22
23   @override
24   void dispose() {
25     _titleController.dispose();
26     _descriptionController.dispose();
27     super.dispose();
28   }
29
30   // Ubah menjadi Future<void> dan beri async
31   Future<void> _refreshData() async {
32     final data = await dbHelper.queryAllRows();
33     print(data); // Tambahkan log ini
34     setState(() {
35       _dbData = data;
36     });
37   }
38
39   // Ubah menjadi Future<void> dan beri async
40   Future<void> _addData() async {
41     await dbHelper.insert({
42       'title': _titleController.text,
43       'description': _descriptionController.text,
44     });
45     _titleController.clear();
46     _descriptionController.clear();
47     await _refreshData(); // Pastikan menunggu refresh data setelah insert
48   }
49
50   // Ubah menjadi Future<void> dan beri async
51   Future<void> _updateData(int id) async {
52     await dbHelper.update({
53       'id': id,
54       'title': _titleController.text,
55       'description': _descriptionController.text,
56     });
57     _titleController.clear();
58     _descriptionController.clear();
59     await _refreshData(); // Pastikan menunggu refresh data setelah update
60   }
61
62   // Ubah menjadi Future<void> dan beri async
63   Future<void> _deleteData(int id) async {
64     await dbHelper.delete(id);
65     await _refreshData(); // Pastikan menunggu refresh data setelah delete
66   }
67
68   // Menampilkan dialog untuk mengedit data
69   void _showEditDialog(Map<String, dynamic> item) {
70     _titleController.text = item['title'];
71     _descriptionController.text = item['description'];
72   }
```

```

72
73     showDialog(
74       context: context,
75       builder: (context) {
76         return AlertDialog(
77           title: const Text('Edit Item'),
78           content: Column(
79             mainAxisAlignment: MainAxisAlignment.min,
80             children: [
81               TextField(
82                 controller: _titleController,
83                 decoration: InputDecoration(labelText: 'Title'),
84               ),
85               TextField(
86                 controller: _descriptionController,
87                 decoration: InputDecoration(labelText: 'Description'),
88               ),
89             ],
90           ),
91           actions: [
92             TextButton(
93               onPressed: () {
94                 Navigator.of(context).pop();
95               },
96               child: Text('Cancel'),
97             ),
98             TextButton(
99               onPressed: () {
100                 _updateData(item['id']);
101                 Navigator.of(context).pop();
102               },
103               child: const Text('Save'),
104             ),
105           ],
106         );
107       },
108     );
109   }
110
111   // Menampilkan dialog untuk menambahkan data
112   void _showAddDialog() {
113     _titleController.clear();
114     _descriptionController.clear();
115
116     showDialog(
117       context: context,
118       builder: (context) {
119         return AlertDialog(
120           title: Text('Add New Item'),
121           content: Column(
122             mainAxisAlignment: MainAxisAlignment.min,
123             children: [
124               TextField(
125                 controller: _titleController,
126                 decoration: InputDecoration(labelText: 'Title'),
127               ),
128               TextField(
129                 controller: _descriptionController,
130                 decoration: InputDecoration(labelText: 'Description'),
131               ),
132             ],
133           ),
134           actions: [
135             TextButton(
136               onPressed: () {
137                 Navigator.of(context).pop();
138               },
139               child: Text('Cancel'),
140             ),
141             TextButton(
142               onPressed: () {
143                 _addData();
144                 Navigator.of(context).pop();
145               },
146               child: const Text('Add'),
147             ),
148           ],
149         );
150       },
151     );
152   }
153

```

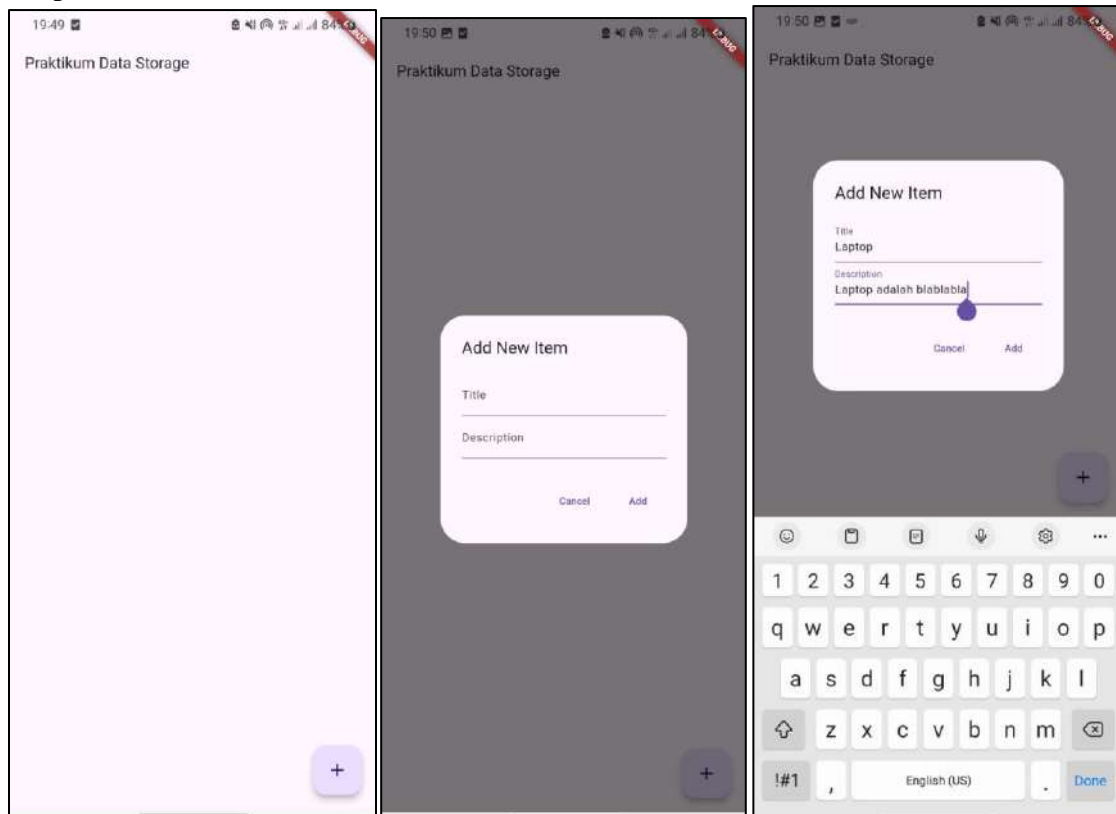


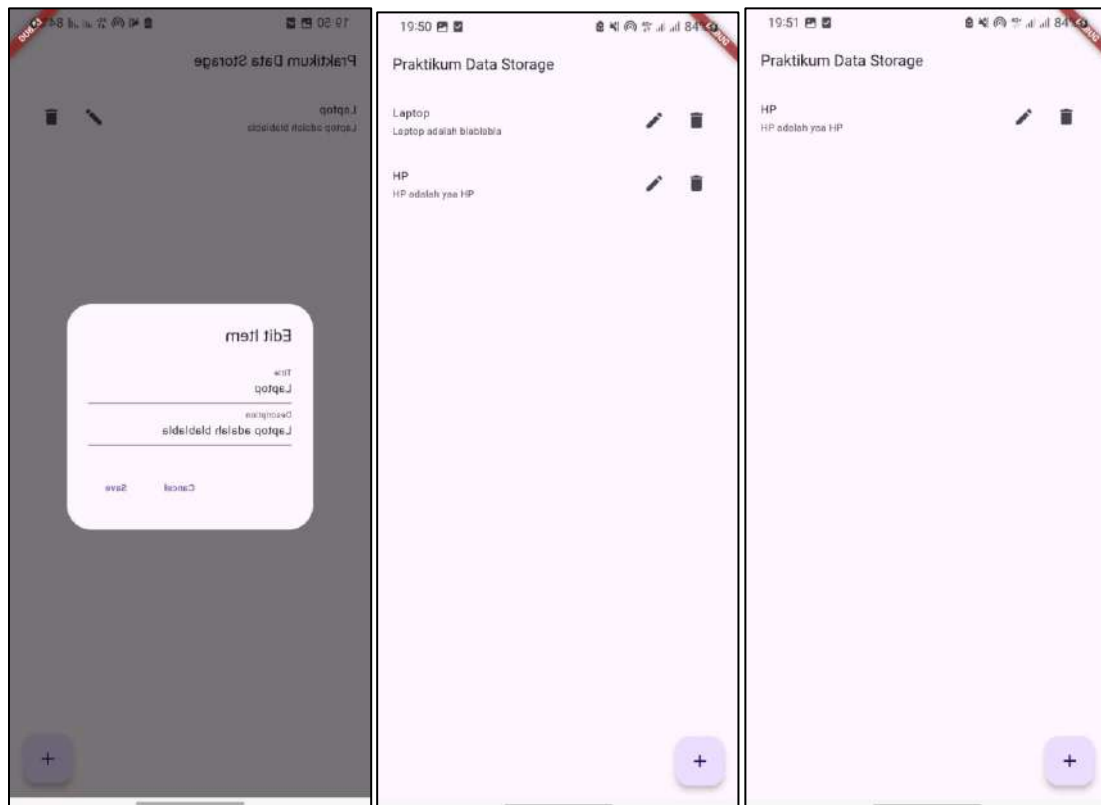
```

154 @override
155 Widget build(BuildContext context) {
156   return Scaffold(
157     appBar: AppBar(
158       title: Text('Praktikum Data Storage'),
159     ),
160     body: ListView.builder(
161       itemCount: _dbData.length,
162       itemBuilder: (context, index) {
163         final item = _dbData[index];
164         print('Displaying item: ${item['title']}');
165         return ListTile(
166           title: Text(item['title'] ?? 'No Title'),
167           subtitle: Text(item['description'] ?? 'No Description'),
168           trailing: Row(
169             mainAxisAlignment: MainAxisAlignment.min,
170             children: [
171               IconButton(
172                 icon: Icon(Icons.edit),
173                 onPressed: () -> _showEditDialog(item),
174               ),
175               IconButton(
176                 icon: Icon(Icons.delete),
177                 onPressed: () -> _deleteData(item['id']),
178               ),
179             ],
180           ),
181         );
182       },
183     ),
184     floatingActionButton: FloatingActionButton(
185       onPressed: _showAddDialog,
186       child: Icon(Icons.add),
187     ),
188   );
189 }
190 }

```

Output





Program ini adalah sebuah aplikasi Flutter yang menggunakan SQLite sebagai database lokal untuk mengelola data sederhana. Aplikasi ini memungkinkan pengguna untuk menambahkan, mengedit, melihat, dan menghapus data berupa judul (title) dan deskripsi (description) melalui antarmuka yang interaktif. Database SQLite diimplementasikan melalui DatabaseHelper, yang menyediakan fungsi seperti insert, update, queryAllRows, dan delete untuk mengelola tabel bernama my_table. Pada bagian UI, pengguna dapat menggunakan dialog untuk memasukkan atau memperbarui data, serta tombol aksi untuk menghapus data. Aplikasi ini sangat cocok untuk pemahaman dasar tentang integrasi Flutter dengan SQLite dalam membangun aplikasi mobile berbasis data.