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

```dart
import 'package:flutter/material.dart';
import 'package:praktikum_10/view/my_db_view.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Praktikum Data Storage',
      theme: ThemeData(
        primarySwatch: Colors.orange,
      ),
      home: MyDatabaseView(),
    );
  }
}
```

   Db_helper.dart

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

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

  // factory constructor untuk mengembalikan instance singleton
  factory DatabaseHelper() {
    return _instance;
  }

  // Private constructor
  DatabaseHelper._internal();
```

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

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

  //membuat tabel saat db pertama kali dibuat
  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)
  ''');
  }

  // metode memasukan data ke dalam tabel
  Future<int> insert(Map<String, dynamic> row) async {
    Database db = await database;
    int result = await db.insert('my_table', row);
    print('Inserted row: $result');  // Log untuk memeriksa hasil insert
    return result;
  }

  // metode mengambil semua data dari tabel
  Future<List<Map<String, dynamic>>> queryAllRows() async {
    Database db = await database;
    List<Map<String, dynamic>> result = await db.query('my_table');
    print('Fetched data: $result');  // Log untuk memeriksa data yang diambil
    return result;
  }

  // metode untuk memperbarui data dalam tabel
  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]);
  }

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

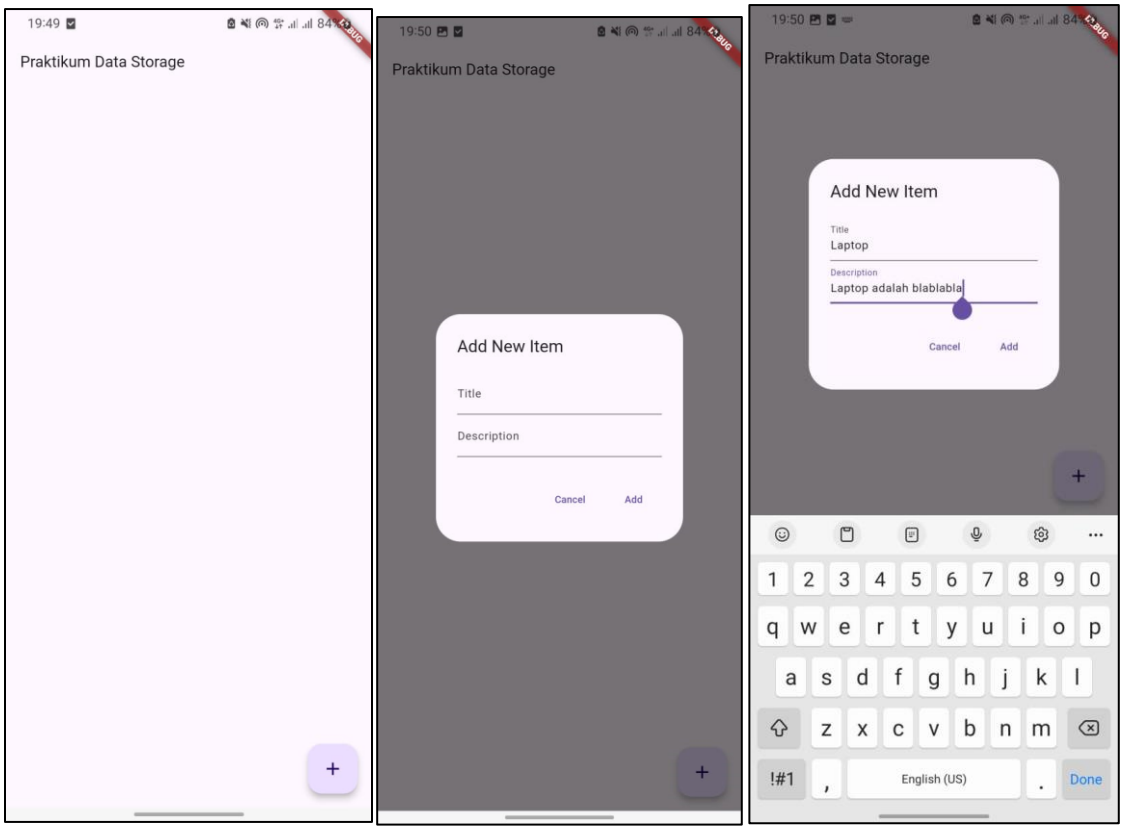
My_db_view

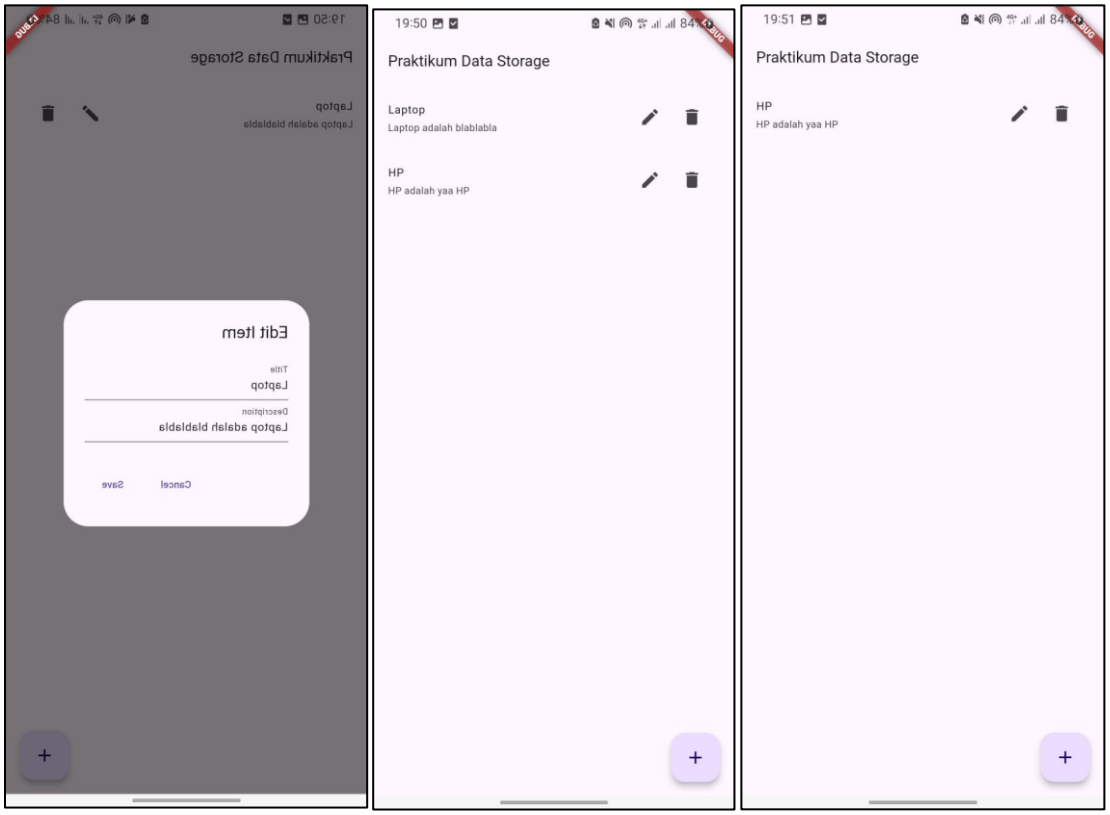
```dart
import 'package:flutter/material.dart';
import 'package:praktikum_10/helper/db_helper.dart';

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

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

class _MyDatabaseViewState extends State<MyDatabaseView> {
  final DatabaseHelper dbHelper = DatabaseHelper();
  List<Map<String, dynamic>> _dbData = [];
  final TextEditingController _titleController = TextEditingController();
  final TextEditingController _descriptionController = TextEditingController();

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

  @override
  void dispose() {
    _titleController.dispose();
    _descriptionController.dispose();
    super.dispose();
  }

  // Ubah menjadi Future<void> dan beri async
  Future<void> _refreshData() async {
    final data = await dbHelper.queryAllRows();
    print(data); // Tambahkan log ini
    setState(() {
      _dbData = data;
    });
  }

  // Ubah menjadi Future<void> dan beri async
  Future<void> _addData() async {
    await dbHelper.insert({
      'title': _titleController.text,
      'description': _descriptionController.text,
    });
    _titleController.clear();
    _descriptionController.clear();
    await _refreshData(); // Pastikan menunggu refresh data setelah insert
  }

  // Ubah menjadi Future<void> dan beri async
  Future<void> _updateData(int id) async {
    await dbHelper.update({
      'id': id,
      'title': _titleController.text,
      'description': _descriptionController.text,
    });
    _titleController.clear();
    _descriptionController.clear();
    await _refreshData(); // Pastikan menunggu refresh data setelah update
  }

  // Ubah menjadi Future<void> dan beri async
  Future<void> _deleteData(int id) async {
    await dbHelper.delete(id);
    await _refreshData(); // Pastikan menunggu refresh data setelah delete
  }

  // 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 Item'),
          content: Column(
            mainAxisSize: MainAxisSize.min,
            children: [
              TextField(
                controller: _titleController,
                decoration: InputDecoration(labelText: 'Title'),
              ),
              TextField(
                controller: _descriptionController,
                decoration: InputDecoration(labelText: 'Description'),
              ),
            ],
          ),
          actions: [
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: Text('Cancel'),
            ),
            TextButton(
              onPressed: () {
                _updateData(item['id']);
                Navigator.of(context).pop();
              },
              child: const Text('Save'),
            ),
          ],
        );
      },
    );
  }

  // Menampilkan dialog untuk menambahkan data
  void _showAddDialog() {
    _titleController.clear();
    _descriptionController.clear();

    showDialog(
      context: context,
      builder: (context) {
        return AlertDialog(
          title: Text('Add New Item'),
          content: Column(
            mainAxisSize: MainAxisSize.min,
            children: [
              TextField(
                controller: _titleController,
                decoration: InputDecoration(labelText: 'Title'),
              ),
              TextField(
                controller: _descriptionController,
                decoration: InputDecoration(labelText: 'Description'),
              ),
            ],
          ),
          actions: [
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: Text('Cancel'),
            ),
            TextButton(
              onPressed: () {
                _addData();
                Navigator.of(context).pop();
              },
              child: const Text('Add'),
            ),
          ],
        );
      },
    );
  }
```

```dart
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Praktikum Data Storage'),
      ),
      body: ListView.builder(
        itemCount: _dbData.length,
        itemBuilder: (context, index) {
          final item = _dbData[index];
          print('Displaying item: ${item['title']}');
          return ListTile(
            title: Text(item['title'] ?? 'No Title'),
            subtitle: Text(item['description'] ?? 'No Description'),
            trailing: Row(
              mainAxisSize: MainAxisSize.min,
              children: [
                IconButton(
                  icon: Icon(Icons.edit),
                  onPressed: () => _showEditDialog(item),
                ),
                IconButton(
                  icon: Icon(Icons.delete),
                  onPressed: () => _deleteData(item['id']),
                ),
              ],
            ),
          );
        },
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _showAddDialog,
        child: Icon(Icons.add),
      ),
    );
  }
}
```

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.