

**LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK**

**MODUL X
DATA STORAGE (BAGIAN I)**



Disusun Oleh :

Aditya Prabu Mukti / 2211104037

SE-06-02

Asisten Praktikum :

Muhammad Faza Zulian Gesit Al Barru

Aisyah Hasna Aulia

Dosen Pengampu :

Yudha Islami Sulistya

**PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK
FAKULTAS INFORMATIKA**

2024

GUIDED

A. PUBSPEC.YAML

Sourcecode:

```

name: pertemuan10
description: "A new Flutter project."
# The following line prevents the package from being accidentally
published to
# pub.dev using `flutter pub publish`. This is preferred for private
packages.
publish_to: 'none' # Remove this line if you wish to publish to
pub.dev

# The following defines the version and build number for your
application.
# A version number is three numbers separated by dots, like 1.2.43
# followed by an optional build number separated by a +.
# Both the version and the builder number may be overridden in
flutter
# build by specifying --build-name and --build-number, respectively.
# In Android, build-name is used as versionName while build-number
used as versionCode.
# Read more about Android versioning at
https://developer.android.com/studio/publish/versioning
# In iOS, build-name is used as CFBundleShortVersionString while
build-number is used as CFBundleVersion.
# Read more about iOS versioning at
#
https://developer.apple.com/library/archive/documentation/General/Ref
erence/InfoPlistKeyReference/Articles/CoreFoundationKeys.html
# In Windows, build-name is used as the major, minor, and patch parts
# of the product and file versions while build-number is used as the
build suffix.
version: 1.0.0+1

environment:
  sdk: ^3.5.4

# Dependencies specify other packages that your package needs in
order to work.
# To automatically upgrade your package dependencies to the latest
versions
# consider running `flutter pub upgrade --major-versions`.
Alternatively,
# dependencies can be manually updated by changing the version
numbers below to
# the latest version available on pub.dev. To see which dependencies
have newer
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
    sdk: flutter

# The following adds the Cupertino Icons font to your application.
# Use with the CupertinoIcons class for iOS style icons.
cupertino_icons: ^1.0.8
sqflite: ^2.4.1
path: ^1.9.0

dev_dependencies:
  flutter_test:
    sdk: flutter

# The "flutter_lints" package below contains a set of recommended
lints to
# encourage good coding practices. The lint set provided by the
package is
# activated in the `analysis_options.yaml` file located at the root
of your
# package. See that file for information about deactivating
specific lint
# rules and activating additional ones.
flutter_lints: ^4.0.0

# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter packages.
flutter:

  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the icons in
  # the material Icons class.
  uses-material-design: true

  # To add assets to your application, add an assets section, like
  this:
  # assets:
  #   - images/a_dot_burr.jpeg
  #   - images/a_dot_ham.jpeg

  # An image asset can refer to one or more resolution-specific
  "variants", see
  # https://flutter.dev/to/resolution-aware-images

  # For details regarding adding assets from package dependencies,
  see
  # https://flutter.dev/to/asset-from-package

  # To add custom fonts to your application, add a fonts section
  here,
  # in this "flutter" section. Each entry in this list should have a
  # "family" key with the font family name, and a "fonts" key with a
  # list giving the asset and other descriptors for the font. For
  # example:
  # fonts:
  #   - family: Schyler
  #     fonts:
  #       - asset: fonts/Schyler-Regular.ttf
  #       - asset: fonts/Schyler-Italic.ttf
  #         style: italic
  #   - family: Trajan Pro
  #     fonts:
  #       - asset: fonts/TrajanPro.ttf
  #       - asset: fonts/TrajanPro_Bold.ttf
  #         weight: 700
  #
  # For details regarding fonts from package dependencies,
  # see https://flutter.dev/to/font-from-package

```

Deskripsi Program

File di atas adalah konfigurasi proyek Flutter bernama pertemuan10 yang menggunakan format pubspec.yaml untuk mengatur metadata proyek, dependensi, dan pengaturan lainnya. Dalam file ini, ditentukan nama proyek, deskripsi, dan versi aplikasi pertama (1.0.0+1). File ini juga menetapkan versi SDK Dart (^3.5.3) untuk memastikan kesesuaian dengan lingkungan pengembangan. Dependensi utama yang digunakan antara lain flutter untuk pengembangan antarmuka pengguna, cupertino_icons untuk ikon bergaya iOS, sqflite untuk pengelolaan database lokal, dan path untuk manajemen jalur file. Selain itu, ada juga dependensi pengembangan seperti flutter_test untuk pengujian dan flutter_lints untuk memastikan praktik pengkodean yang baik. Terdapat pula placeholder untuk aset seperti gambar, yang dapat diaktifkan bila diperlukan dalam proyek.

B. DBHELPER

Sourcecode:

```

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 {
    final dbPath = join(await getDatabasesPath(), 'my_database.db');
    return await openDatabase(dbPath, 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
      )
    ''');
  }

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

  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]);
  }
}

```

Deskripsi Program

Program di atas adalah kelas `DatabaseHelper` untuk mengelola database SQLite di Flutter menggunakan pola singleton. Database bernama `dimascahyo.db` dibuat dengan tabel `my_table`, berisi kolom `id`, `title`, `description`, dan `createdAt`. Kelas ini menyediakan metode CRUD: `insert` untuk menambah data, `queryAllRows` untuk membaca semua data, `update` untuk memperbarui data berdasarkan `id`, dan `delete` untuk menghapus data berdasarkan `id`. Dengan ini, pengelolaan database menjadi sederhana dan terorganisir.

B. MYDBVIEW

Sourcecode:

```

import 'package:flutter/material.dart';
import 'package:pertemuan18/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() {
    _refreshData();
    super.initState();
  }

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

  // Metode untuk memunculkan data dari database
  void _refreshData() async {
    final data = await dbHelper.queryAllRows();
    setState(() {
      _dbData = data;
    });
  }

  // Metode untuk menambahkan data ke database
  void _addData() async {
    await dbHelper.insert({
      'title': _titleController.text,
      'description': _descriptionController.text,
    });
    _titleController.clear();
    _descriptionController.clear();
    _refreshData();
  }

  // Metode untuk memperbarui data dalam database
  void _updateData(int id) async {
    await dbHelper.update({
      'id': id,
      'title': _titleController.text,
      'description': _descriptionController.text,
    });
    _titleController.clear();
    _descriptionController.clear();
    _refreshData();
  }

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Database View'),
      ),
      body: Column(
        children: [
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: TextField(
              controller: _titleController,
              decoration: const InputDecoration(
                border: OutlineInputBorder(),
                labelText: 'Title',
              ),
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: TextField(
              controller: _descriptionController,
              decoration: const InputDecoration(
                border: OutlineInputBorder(),
                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(
                    mainAxisAlignment: MainAxisAlignment.min,
                    children: [
                      IconButton(
                        icon: const Icon(Icons.edit),
                        onPressed: () => _showEditDialog(item),
                      ),
                      IconButton(
                        icon: const Icon(Icons.delete),
                        onPressed: () => _deleteData(item['id']),
                      ),
                    ],
                  ),
                );
              },
            ),
          ),
        ],
      ),
    );
  }
}

```

Deskripsi Program

Kode di atas adalah widget Flutter untuk mengelola data SQLite dengan operasi CRUD (Create, Read, Update, Delete). Menggunakan `DatabaseHelper`, data ditampilkan dalam ListView dan dapat ditambah, diperbarui, atau dihapus. Input dilakukan melalui TextField, dan dialog digunakan untuk pengeditan. Metode `_refreshData` memastikan data selalu diperbarui secara real-time di tampilan. Widget ini mempermudah pengelolaan database langsung dari aplikasi.

D. MAIN PROGRAM

Sourcecode:


```

import 'package:flutter/material.dart';
import 'package:pertemuan18/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() {
    _refreshData();
    super.initState();
  }

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

  // Metode untuk memunculkan data dari database
  void _refreshData() async {
    final data = await dbHelper.queryAllRows();
    setState(() {
      _dbData = data;
    });
  }

  // Metode untuk menambahkan data ke database
  void _addData() async {
    await dbHelper.insert({
      'title': _titleController.text,
      'description': _descriptionController.text,
    });
    _titleController.clear();
    _descriptionController.clear();
    _refreshData();
  }

  // Metode untuk memperbarui data dalam database
  void _updateData(int id) async {
    await dbHelper.update({
      'id': id,
      'title': _titleController.text,
      'description': _descriptionController.text,
    });
    _titleController.clear();
    _descriptionController.clear();
    _refreshData();
  }

  // Metode untuk menghapus data dari database
  void _deleteData(int id) async {
    await dbHelper.delete(id);
    _refreshData();
  }

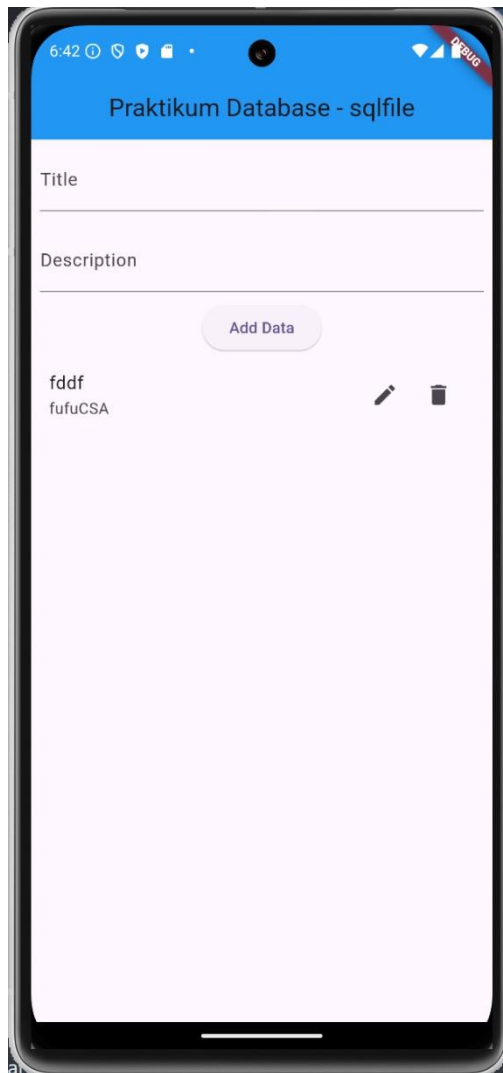
  // Menampilkan dialog untuk mengedit data
  void _showEditDialog(Map<String, dynamic> item) {
    _titleController.text = item['title'];
    _descriptionController.text = item['description'];

    showDialog(
      context: context,
      builder: (context) {
        return AlertDialog(
          title: const Text('Edit Item'),
          content: Column(
            mainAxisAlignment: MainAxisAlignment.min,
            children: [
              TextField(
                controller: _titleController,
                decoration: const InputDecoration(
                  labelText: 'Title',
                ),
              ),
              TextField(
                controller: _descriptionController,
                decoration: const InputDecoration(
                  labelText: 'Description',
                ),
              ),
            ],
          ),
          actions: [
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: const Text('Cancel'),
            ),
            TextButton(
              onPressed: () {
                _updateData(item['id']);
                Navigator.of(context).pop();
              },
              child: const Text('Save'),
            ),
          ],
        );
      },
    );
  }

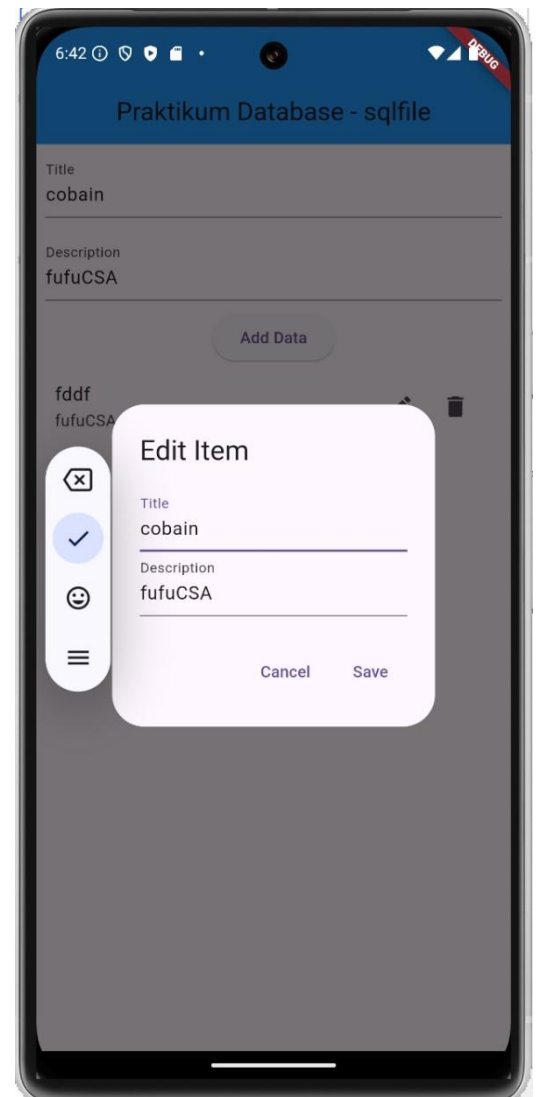
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Database View'),
      ),
      body: Column(
        children: [
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: TextField(
              controller: _titleController,
              decoration: const InputDecoration(
                border: OutlineInputBorder(),
                labelText: 'Title',
              ),
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: TextField(
              controller: _descriptionController,
              decoration: const InputDecoration(
                border: OutlineInputBorder(),
                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(
                    mainAxisAlignment: MainAxisAlignment.min,
                    children: [
                      IconButton(
                        icon: const Icon(Icons.edit),
                        onPressed: () => _showEditDialog(item),
                      ),
                      IconButton(
                        icon: const Icon(Icons.delete),
                        onPressed: () => _deleteData(item['id']),
                      ),
                    ],
                  ),
                );
              },
            ),
          ),
        ],
      ),
    );
  }
}

```

Screenshoot Output



Tampilan aplikasi



Form edit data

Deskripsi Program

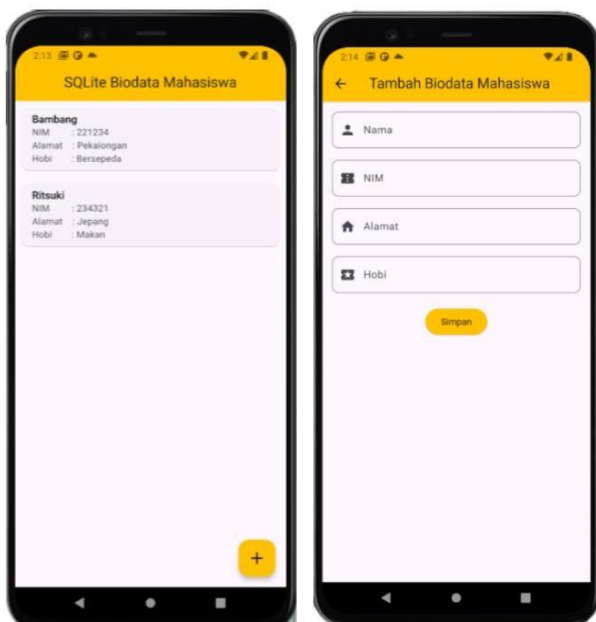
Aplikasi ini adalah Flutter CRUD sederhana menggunakan SQLite, memungkinkan pengguna menambahkan, melihat, mengedit, dan menghapus data melalui antarmuka dengan input Title dan Description. Data ditampilkan dalam ListView dan diperbarui secara real-time, memudahkan praktik pengelolaan database lokal.

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:



Sourcecode:

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

class DatabaseHelper {
  static final DatabaseHelper _instance = DatabaseHelper._internal();
  factory DatabaseHelper() => _instance;

  static Database? _database;

  DatabaseHelper._internal();

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

  Future<Database> _initDatabase() async {
    final dbPath = await getDatabasesPath();
    return openDatabase(
      join(dbPath, 'mahasiswa.db'),
      onCreate: (db, version) async {
        await db.execute('''
          CREATE TABLE mahasiswa (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            nama TEXT,
            nim TEXT,
            alamat TEXT,
            hobi TEXT
          )
        ''');
      },
      version: 1,
    );
  }

  Future<int> insertMahasiswa(Mahasiswa mahasiswa) async {
    final db = await database;
    return await db.insert('mahasiswa', mahasiswa.toMap());
  }

  Future<List<Mahasiswa>> getMahasiswaList() async {
    final db = await database;
    final List<Map<String, dynamic>> maps = await
    db.query('mahasiswa');
    return maps.map((map) => Mahasiswa.fromMap(map)).toList();
  }
}
```

B. MODELS/MAHASISWA.DART

Sourcecode:

```
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,
  });

  // Convert from Map
  factory Mahasiswa.fromMap(Map<String, dynamic> map) {
    return Mahasiswa(
      id: map['id'],
      nama: map['nama'],
      nim: map['nim'],
      alamat: map['alamat'],
      hobi: map['hobi'],
    );
  }

  // Convert to Map
  Map<String, dynamic> toMap() {
    return {
      'id': id,
      'nama': nama,
      'nim': nim,
      'alamat': alamat,
      'hobi': hobi,
    };
  }
}
```

C. SCREEN/FORM_SCREEN.DART

Sourcecode:

```

import 'package:flutter/material.dart';
import '../db/database_helper.dart';
import '../models/mahasiswa.dart';

class FormScreen extends StatefulWidget {
  @override
  _FormScreenState createState() => _FormScreenState();
}

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

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

  @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,
          children: [
            TextFormField(
              controller: _namaController,
              decoration: InputDecoration(
                labelText: 'Nama',
                prefixIcon: Icon(Icons.person),
              ),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Nama tidak boleh kosong';
                }
                return null;
              },
            ),
            TextFormField(
              controller: _nimController,
              decoration: InputDecoration(
                labelText: 'NIM',
                prefixIcon: Icon(Icons.school),
              ),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'NIM tidak boleh kosong';
                }
                return null;
              },
            ),
            TextFormField(
              controller: _alamatController,
              decoration: InputDecoration(
                labelText: 'Alamat',
                prefixIcon: Icon(Icons.home),
              ),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Alamat tidak boleh kosong';
                }
                return null;
              },
            ),
            TextFormField(
              controller: _hobiController,
              decoration: InputDecoration(
                labelText: 'Hobi',
                prefixIcon: Icon(Icons.sports),
              ),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Hobi tidak boleh kosong';
                }
                return null;
              },
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: _saveData,
              child: Text('Simpan'),
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.orange,
              ),
            ),
          ],
        ),
      ),
    );
  }
}

```

D. SCREEN/HOME_SCREEN.DART

Sourcecode:

```
import 'package:flutter/material.dart';
import '../db/database_helper.dart';
import '../models/mahasiswa.dart';
import 'form_screen.dart';

class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  List<Mahasiswa> mahasiswaList = [];

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

  Future<void> fetchMahasiswa() async {
    final data = await DatabaseHelper().getMahasiswaList();
    setState(() {
      mahasiswaList = data;
    });
  }

  @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.symmetric(horizontal: 16, vertical:
8),
            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: () async {
          final result = await Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => FormScreen()),
          );
          if (result == true) {
            fetchMahasiswa();
          }
        },
        child: Icon(Icons.add),
        backgroundColor: Colors.orange,
      ),
    );
  }
}
```

E. MAIN.DART

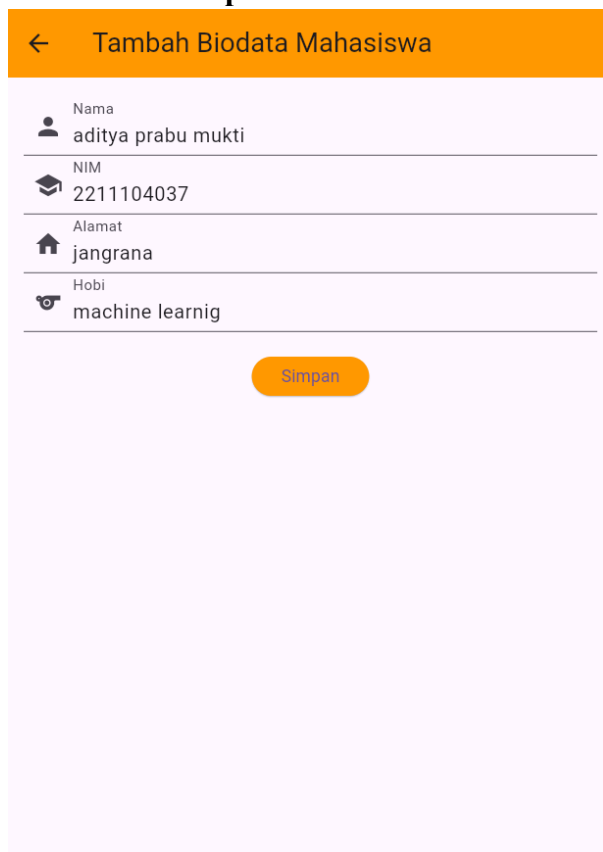
Sourcecode:

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'SQLite Mahasiswa',
      theme: ThemeData(primarySwatch: Colors.orange),
      home: HomeScreen(),
    );
  }
}
```

Screenshoot Output



← Tambah Biodata Mahasiswa

| | |
|--------|--------------------|
| Nama | aditya prabu mukti |
| NIM | 2211104037 |
| Alamat | jangrana |
| Hobi | machine learnig |

Simpan

Penjelasan:

Aplikasi ini merupakan Flutter CRUD untuk mengelola data mahasiswa dengan menggunakan SQLite. Halaman utama aplikasi menampilkan daftar mahasiswa yang dilengkapi dengan tombol untuk menambahkan,

mengedit, dan menghapus data. Struktur folder aplikasi ini terorganisasi dengan baik, terdiri dari db/database_helper.dart yang berfungsi untuk pengelolaan logika database, models/mahasiswa.dart sebagai model data mahasiswa, serta screen/form_screen.dart dan screen/home_screen.dart yang digunakan untuk form input dan tampilan daftar mahasiswa. Aplikasi ini dirancang dengan pendekatan yang memisahkan setiap komponen agar lebih mudah dipahami dan dikelola.