

**TUGAS PENDAHULUAN / TUGAS UNGUIDED  
PEMROGRAMAN PERANGKAT BERGERAK**

**MODUL X**

**Data Storage Bagian 1**



**Disusun Oleh :**

**Fahmi hasan asagaf 2311104074**

**SE 07 02**

**Asisten Praktikum :**

**Zulfa Mustafa Akhyar Iswahyudi  
Yoga Eka Pratama**

**Dosen Pengampu :**

**Yudha Islami Sulisty, S.Kom., M.Cs.**

**PROGRAM STUDI S1 SOFTWARE ENGINEERING**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2025**

## TUGAS UNGUIDED

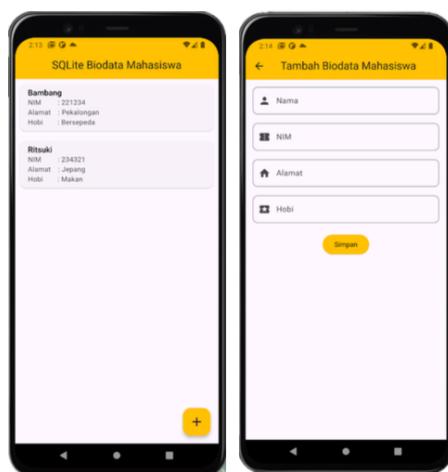
### A. SOAL

#### **Tugas Mandiri (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.*

## B. JAWABAN

### Main.dart

```
import 'package:flutter/material.dart';
import 'main_page.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Biodata Mahasiswa',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        useMaterial3: true,
      ),
      home: const MainPage(),
      debugShowCheckedModeBanner: false,
    );
  }
}
```

### Pubspec.yaml

```
name: biodata_mahasiswa
description: Aplikasi biodata mahasiswa dengan SQLite

environment:
  sdk: '>=3.0.0 <4.0.0'

dependencies:
  flutter:
    sdk: flutter
  sqflite: ^2.3.0
  path: ^1.8.3

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^3.0.0
```

## Database\_helper.dart

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

class DatabaseHelper {
  static Database? _database;
  static const String tableName = 'mahasiswa';
  static const String columnId = 'id';
  static const String columnNama = 'nama';
  static const String columnNim = 'nim';
  static const String columnAlamat = 'alamat';
  static const String columnHobi = 'hobi';

  // Tambahkan getter untuk database
  static Future<Database> get db async {
    return await database;
  }

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

  static Future<Database> _initDatabase() async {
    String path = join(await getDatabasesPath(), 'biodata_mahasiswa.db');

    return await openDatabase(
      path,
      version: 1,
      onCreate: _createTable,
    );
  }

  static Future<void> _createTable(Database db, int version) async {
    await db.execute('''
      CREATE TABLE $tableName (
        $columnId INTEGER PRIMARY KEY AUTOINCREMENT,
        $columnNama TEXT NOT NULL,
        $columnNim TEXT NOT NULL,
        $columnAlamat TEXT NOT NULL,
        $columnHobi TEXT NOT NULL
      )
    ''');
    print('Table $tableName created successfully');
  }

  // CREATE – Menyimpan data mahasiswa
  static Future<int> insertMahasiswa(Map<String, dynamic> mahasiswa) async {
    try {
      final db = await database;
      final result = await db.insert(
        tableName,
        mahasiswa,
      );
      return result;
    } catch (e) {
      print('Error inserting data: $e');
      return -1;
    }
  }

  // READ – Mengambil data mahasiswa
  static Future<List<Map<String, dynamic>> > getAllMahasiswa() async {
    final db = await database;
    final List<Map<String, dynamic>> result = await db.query(tableName);
    return result;
  }

  // UPDATE – Mengupdate data mahasiswa
  static Future<int> updateMahasiswa(Map<String, dynamic> mahasiswa) async {
    try {
      final db = await database;
      final result = await db.update(
        tableName,
        mahasiswa,
        where: 'id = ?',
        whereArgs: [mahasiswa['id']],
      );
      return result;
    } catch (e) {
      print('Error updating data: $e');
      return -1;
    }
  }

  // DELETE – Menghapus data mahasiswa
  static Future<int> deleteMahasiswa(int id) async {
    try {
      final db = await database;
      final result = await db.delete(
        tableName,
        where: 'id = ?',
        whereArgs: [id],
      );
      return result;
    } catch (e) {
      print('Error deleting data: $e');
      return -1;
    }
  }
}
```

```
        conflictAlgorithm: ConflictAlgorithm.replace,
    );
    print('Data inserted successfully with ID: $result');
    return result;
} catch (e) {
    print('Error inserting data: $e');
    return -1;
}
}

// READ – Membaca semua data mahasiswa
static Future<List<Map<String, dynamic>>> getMahasiswa() async {
try {
    final db = await database;
    final result = await db.query(tableName);
    print('Data fetched successfully: ${result.length} records');
    return result;
} catch (e) {
    print('Error fetching data: $e');
    return [];
}
}

// UPDATE – Memperbarui data mahasiswa
static Future<int> updateMahasiswa(Map<String, dynamic> mahasiswa) async {
try {
    final db = await database;
    final result = await db.update(
        tableName,
        mahasiswa,
        where: '$columnId = ?',
        whereArgs: [mahasiswa[columnId]],
    );
    print('Data updated successfully');
    return result;
} catch (e) {
    print('Error updating data: $e');
    return -1;
}
}

// DELETE – Menghapus data mahasiswa
static Future<int> deleteMahasiswa(int id) async {
try {
    final db = await database;
    final result = await db.delete(
        tableName,
        where: '$columnId = ?',
        whereArgs: [id],
    );
    print('Data deleted successfully');
    return result;
} catch (e) {
    print('Error deleting data: $e');
}
```

```
        return -1;
    }
}
```

### Mahasiswa\_model.dart

```
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,
    });

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

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

## Input\_mahasiswa\_page.dart

```
import 'package:flutter/material.dart';
import 'database_helper.dart';

class InputMahasiswaPage extends StatefulWidget {
  final Map<String, dynamic>? mahasiswa;

  const InputMahasiswaPage({Key? key, this.mahasiswa}) : super(key: key);

  @override
  _InputMahasiswaPageState createState() => _InputMahasiswaPageState();
}

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

  bool _isLoading = false;

  @override
  void initState() {
    super.initState();
    // Jika dalam mode edit, isi form dengan data yang ada
    if (widget.mahasiswa != null) {
      _namaController.text = widget.mahasiswa!['nama'] ?? '';
      _nimController.text = widget.mahasiswa!['nim'] ?? '';
      _alamatController.text = widget.mahasiswa!['alamat'] ?? '';
      _hobiController.text = widget.mahasiswa!['hobi'] ?? '';
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.mahasiswa == null
            ? 'Tambah Mahasiswa'
            : 'Edit Mahasiswa'),
        backgroundColor: Colors.blue[700],
      ),
      body: _isLoading
          ? Center(child: CircularProgressIndicator())
          : Padding(
              padding: const EdgeInsets.all(16.0),
              child: Form(
                key: _formKey,
                child: Column(
                  children: [
                    TextFormField(
                      controller: _namaController,
                      decoration: InputDecoration(

```

```
        labelText: 'Nama',
        border: OutlineInputBorder(),
        prefixIcon: Icon(Icons.person),
      ),
      validator: (value) {
        if (value == null || value.isEmpty) {
          return 'Nama harus diisi';
        }
        return null;
      },
    ),
    SizedBox(height: 16),
    TextFormField(
      controller: _nimController,
      decoration: InputDecoration(
        labelText: 'NIM',
        border: OutlineInputBorder(),
        prefixIcon: Icon(Icons.confirmation_number),
      ),
      validator: (value) {
        if (value == null || value.isEmpty) {
          return 'NIM harus diisi';
        }
        return null;
      },
    ),
    SizedBox(height: 16),
    TextFormField(
      controller: _alamatController,
      maxLines: 3,
      decoration: InputDecoration(
        labelText: 'Alamat',
        border: OutlineInputBorder(),
        prefixIcon: Icon(Icons.home),
      ),
      validator: (value) {
        if (value == null || value.isEmpty) {
          return 'Alamat harus diisi';
        }
        return null;
      },
    ),
    SizedBox(height: 16),
    TextFormField(
      controller: _hobiController,
      decoration: InputDecoration(
        labelText: 'Hobi',
        border: OutlineInputBorder(),
        prefixIcon: Icon(Icons.sports_esports),
      ),
      validator: (value) {
        if (value == null || value.isEmpty) {
          return 'Hobi harus diisi';
        }
      }
    )
  ],
)
```

```
        return null;
    },
),
SizedBox(height: 24),
ElevatedButton(
    onPressed: _isLoading ? null : _simpanData,
    style: ElevatedButton.styleFrom(
        backgroundColor: Colors.blue[700],
        minimumSize: Size(double.infinity, 50),
    ),
    child: _isLoading
        ? CircularProgressIndicator(color: Colors.white)
        : Text(
            widget.mahasiswa == null ? 'Simpan' : 'Update',
            style: TextStyle(fontSize: 16, color: Colors.white),
        ),
),
],
),
),
),
),
),
),
),
);
}
}

Future<void> _simpanData() async {
if (_formKey.currentState!.validate()) {
    setState(() {
        _isLoading = true;
    });
}

try {
    Map<String, dynamic> mahasiswa = {
        'nama': _namaController.text,
        'nim': _nimController.text,
        'alamat': _alamatController.text,
        'hobi': _hobiController.text,
    };
}

if (widget.mahasiswa != null && widget.mahasiswa!['id'] != null) {
    // Update data yang sudah ada
    mahasiswa['id'] = widget.mahasiswa!['id'];
    await DatabaseHelper.updateMahasiswa(mahasiswa);
    ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Data berhasil diperbarui!')),
    );
} else {
    // Tambah data baru
    await DatabaseHelper.insertMahasiswa(mahasiswa);
    ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Data berhasil disimpan!')),
    );
}

Navigator.pop(context, true);
}
```

```

        } catch (e) {
            ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(content: Text('Error: Gagal menyimpan data')),
            );
            print('Error saving data: $e');
        } finally {
            setState(() {
                _isLoading = false;
            });
        }
    }
}

@Override
void dispose() {
    _namaController.dispose();
    _nimController.dispose();
    _alamatController.dispose();
    _hobiController.dispose();
    super.dispose();
}
}
}

```

## Main\_page.dart

```

import 'package:flutter/material.dart';
import 'database_helper.dart';
import 'input_mahasiswa_page.dart';

class MainPage extends StatefulWidget {
    const MainPage({Key? key}) : super(key: key);

    @override
    _MainPageState createState() => _MainPageState();
}

class _MainPageState extends State<MainPage> {
    List<Map<String, dynamic>> _daftarMahasiswa = [];
    bool _isLoading = true;

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

    Future<void> _loadMahasiswa() async {
        setState(() {
            _isLoading = true;
        });

```

```
try {
    final data = await DatabaseHelper.getMahasiswa();
    setState(() {
        _daftarMahasiswa = data;
        _isLoading = false;
    });
} catch (e) {
    print('Error loading data: $e');
    setState(() {
        _isLoading = false;
    });
}
}

void _tambahMahasiswa() async {
    final result = await Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => InputMahasiswaPage()),
    );

    if (result == true) {
        await _loadMahasiswa();
    }
}

void _editMahasiswa(Map<String, dynamic> mahasiswa) async {
    final result = await Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => InputMahasiswaPage(mahasiswa: mahasiswa),
        ),
    );

    if (result == true) {
        await _loadMahasiswa();
    }
}

void _hapusMahasiswa(int id) async {
    showDialog(
        context: context,
        builder: (context) => AlertDialog(
            title: Text('Hapus Data'),
            content: Text('Apakah Anda yakin ingin menghapus data ini?'),
            actions: [
                TextButton(
                    onPressed: () => Navigator.pop(context),
                    child: Text('Batal'),
                ),
                TextButton(
                    onPressed: () async {
                        Navigator.pop(context);
                        await DatabaseHelper.deleteMahasiswa(id);
                        ScaffoldMessenger.of(context).showSnackBar(

```

```
        SnackBar(content: Text('Data berhasil dihapus!')),
    );
    await _loadMahasiswa();
},
child: Text('Hapus', style: TextStyle(color: Colors.red)),
),
],
),
);
}

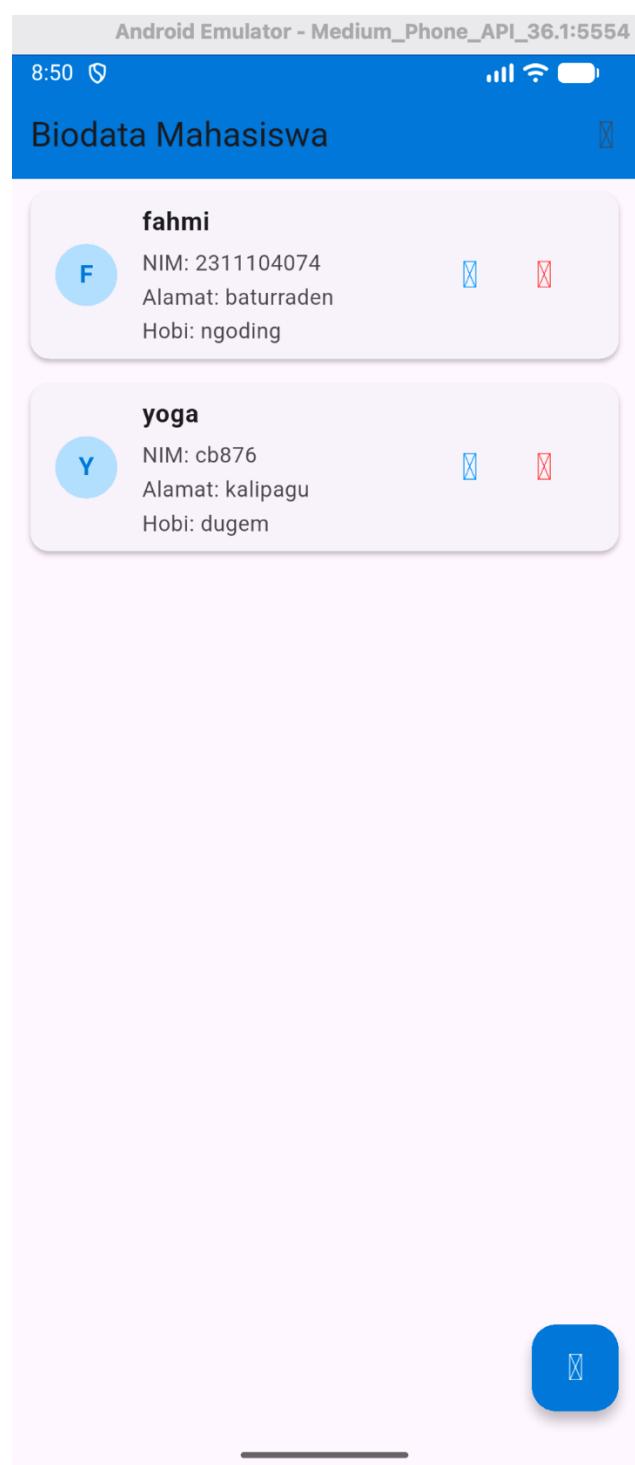
@Override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: Text('Biodata Mahasiswa'),
backgroundColor: Colors.blue[700],
elevation: 0,
actions: [
IconButton(
icon: Icon(Icons.refresh),
onPressed: _loadMahasiswa,
),
],
),
body: _isLoading
? Center(child: CircularProgressIndicator())
: _daftarMahasiswa.isEmpty
? Center(
child: Column(
mainAxisAlignment: MainAxisAlignment.center,
children: [
Icon(Icons.people_alt_outlined, size: 80, color: Colors.grey),
SizedBox(height: 16),
Text(
'Belum ada data mahasiswa',
style: TextStyle(fontSize: 18, color: Colors.grey),
),
SizedBox(height: 8),
Text(
'Tap + untuk menambah data',
style: TextStyle(color: Colors.grey),
),
SizedBox(height: 16),
ElevatedButton(
 onPressed: _tambahMahasiswa,
child: Text('Tambah Data Pertama'),
),
],
),
)
: RefreshIndicator(
onRefresh: _loadMahasiswa,
child: ListView.builder(

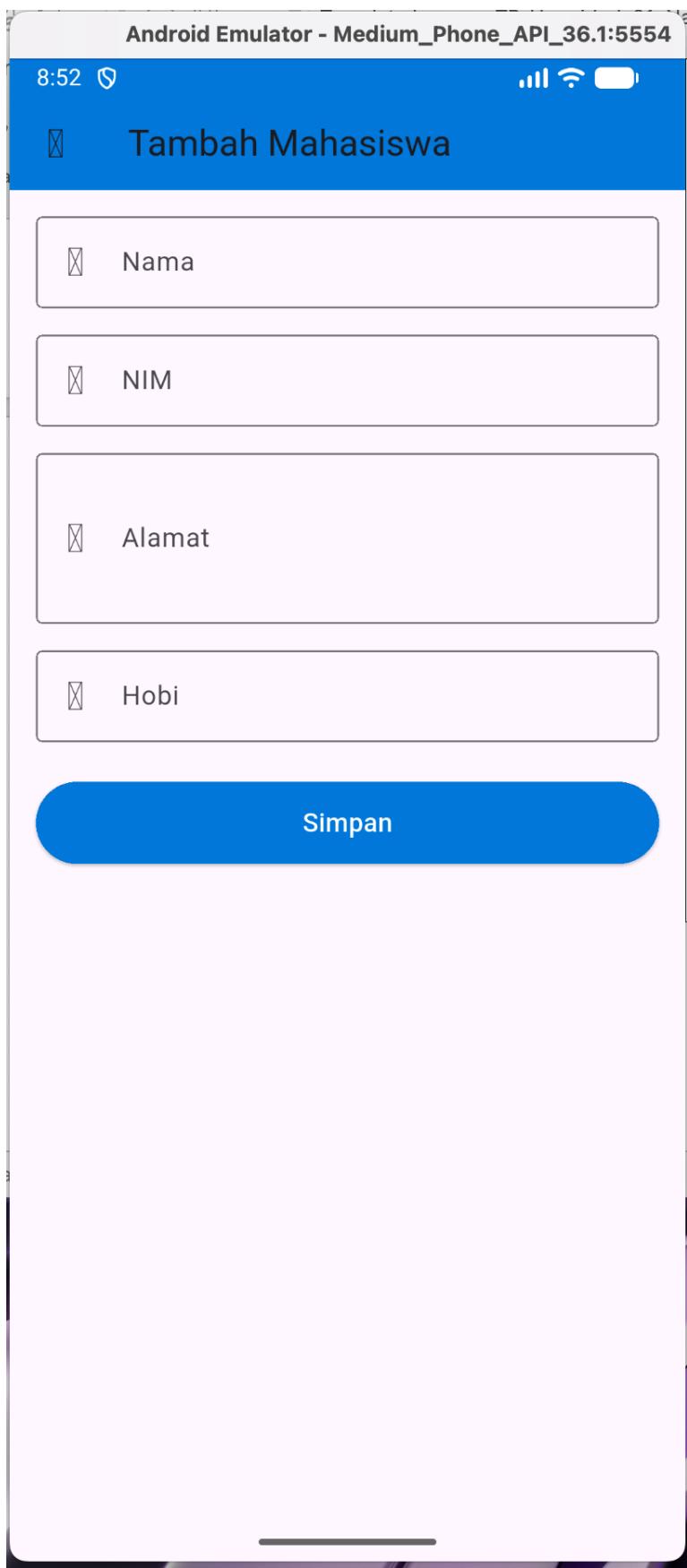
```

```
        itemCount: _daftarMahasiswa.length,
        itemBuilder: (context, index) {
            final mahasiswa = _daftarMahasiswa[index];
            return Card(
                margin: EdgeInsets.symmetric(horizontal: 16, vertical: 8),
                elevation: 2,
                child: ListTile(
                    leading: CircleAvatar(
                        backgroundColor: Colors.blue[100],
                        child: Text(
                            mahasiswa['nama'].toString().substring(0,
1).toUpperCase(),
                            style: TextStyle(color: Colors.blue[700], fontWeight:
FontWeight.bold),
                        ),
                    ),
                    title: Text(
                        mahasiswa['nama'],
                        style: TextStyle(
                            fontWeight: FontWeight.bold,
                            fontSize: 16,
                        ),
                    ),
                    subtitle: Column(
                        crossAxisAlignment: CrossAxisAlignment.start,
                        children: [
                            SizedBox(height: 4),
                            Text('NIM: ${mahasiswa['nim']}'),
                            SizedBox(height: 2),
                            Text('Alamat: ${mahasiswa['alamat']}'),
                            SizedBox(height: 2),
                            Text('Hobi: ${mahasiswa['hobi']}'),
                        ],
                    ),
                    trailing: Row(
                        mainAxisAlignment: MainAxisAlignment.end,
                        mainAxisSize: MainAxisSize.min,
                        children: [
                            IconButton(
                                icon: Icon(Icons.edit, color: Colors.blue),
                                onPressed: () => _editMahasiswa(mahasiswa),
                            ),
                            IconButton(
                                icon: Icon(Icons.delete, color: Colors.red),
                                onPressed: () => _hapusMahasiswa(mahasiswa['id']),
                            ),
                        ],
                    ),
                );
            },
        ),
        floatingActionButton: FloatingActionButton(
            onPressed: _tambahMahasiswa,
```

```
        backgroundColor: Colors.blue[700],  
        child: Icon(Icons.add, color: Colors.white),  
      ),  
    );  
  }  
}
```

## Output





## Deskripsi Program

Aplikasi Biodata Mahasiswa ini merupakan sebuah aplikasi mobile yang dikembangkan menggunakan framework Flutter dengan memanfaatkan SQLite sebagai database lokal untuk menyimpan data secara persisten. Aplikasi ini dirancang khusus untuk mengelola data biodata mahasiswa dengan antarmuka yang user-friendly dan fungsionalitas yang lengkap. Pengguna dapat dengan mudah menambahkan data mahasiswa baru melalui form input yang terstruktur, yang mencakup field nama, NIM, alamat domisili, dan hobi. Setelah data berhasil disimpan, aplikasi akan menampilkan daftar semua mahasiswa yang telah terdaftar dalam bentuk card yang informatif di halaman utama.

Fitur utama aplikasi ini mencakup operasi CRUD (Create, Read, Update, Delete) yang memungkinkan pengguna tidak hanya menambah dan melihat data, tetapi juga mengedit informasi mahasiswa yang sudah ada serta menghapus data yang tidak diperlukan lagi. Setiap operasi dilengkapi dengan feedback visual yang jelas, seperti snackbars untuk konfirmasi keberhasilan operasi dan dialog konfirmasi untuk tindakan penghapusan guna mencegah kesalahan. Aplikasi ini juga memiliki validasi form yang memastikan semua field wajib diisi sebelum data dapat disimpan, sehingga menjaga integritas data.

Dari segi arsitektur, aplikasi ini menerapkan pemisahan concern yang baik dengan pembagian kode menjadi database helper untuk menangani operasi SQLite, model untuk merepresentasikan objek data, dan UI components untuk antarmuka pengguna. Desainnya yang responsif dan modern menggunakan Material Design principles memastikan pengalaman pengguna yang optimal baik di perangkat Android maupun iOS. Aplikasi ini sangat cocok untuk keperluan edukasi dalam mempelajari pengembangan aplikasi mobile dengan Flutter dan konsep database lokal, sekaligus dapat dikembangkan lebih lanjut dengan fitur-fitur tambahan seperti pencarian, sorting, atau integrasi dengan cloud storage.