LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK

MODUL 10: Data Storage Bagian 1

Disusun Oleh :

Ade Fatkhul Anam / 22111040451

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

TELKOM UNIVERSITY PURWOKERTO

2024

MODUL 10
DATA STORAGE BAGIAN 1

A. DB Helper

Source code:

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


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

  // Factory constructor
  factory DatabaseHelper() {
    return _instance;
  }

  // Private constructor
  DatabaseHelper._internal();

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


  // Inisialisasi 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
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)
''');

}

// Menambahkan data
Future<int> insert(Map<String, dynamic> row) async {
Database db = await database;
return await db.insert('my_table', row);
}

// Mengambil semua data
Future<List<Map<String, dynamic>>> queryAllRows() async {
Database db = await database;
return await db.query('my_table');
}

// Buat metode memperbarui data
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]);
}

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


}
```

Deskripsi program:
Program ini adalah kelas helper bernama DatabaseHelper untuk mengelola database SQLite di aplikasi Flutter. Dengan pola singleton, program memastikan hanya satu instance database yang aktif.
Fitur utama:

1. Inisialisasi Database: Membuat database my_prakdatabase.db dengan tabel my_table yang berisi kolom id, title, description, dan createdAt.
2. Operasi CRUD:
   - Create: Menambahkan data dengan insert().
   - Read: Mengambil semua data dengan queryAllRows().
   - Update: Memperbarui data berdasarkan id dengan update().
   - Delete: Menghapus data berdasarkan id dengan delete().

Program ini mendukung pengelolaan data secara efisien untuk aplikasi lokal.

B. My DB View

Source code:

```
import 'package:flutter/material.dart';
import 'package:prak_db/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();
  }

  void _refreshData() async {
    final data = await dbHelper.queryAllRows();
    setState(() {
      dbData = data;
    });
  }

  void _addData() async {
    if (_titleController.text.isNotEmpty && _descriptionController.text.isNotEmpty) {
      await dbHelper.insert({
        'title': _titleController.text,
        'description': _descriptionController.text,
```

```
      });
      _titleController.clear();
      _descriptionController.clear();
      _refreshData();
    } else {
      _showErrorDialog('Fields cannot be empty.');
    }
  }

  void _updateData(int id) async {
    if (_titleController.text.isNotEmpty && _descriptionController.text.isNotEmpty) {
      await dbHelper.update({
        'id': id,
        'title': _titleController.text,
        'description': _descriptionController.text,
      });
      _titleController.clear();
      _descriptionController.clear();
      _refreshData();
    } else {
      _showErrorDialog('Fields cannot be empty.');
    }
  }

  void _deleteData(int id) async {
    await dbHelper.delete(id);
    _refreshData();
  }

  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: 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'),
            ),
          ],
        );
      },
    );
  }

  void _showErrorDialog(String message) {
    showDialog(
      context: context,
      builder: (context) {
        return AlertDialog(
          title: const Text('Error'),
          content: Text(message),
          actions: [
```

```
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: const Text('OK'),
            ),
          ],
        );
      },
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Praktikum Database - sqflite'),
        backgroundColor: Colors.blueGrey,
        centerTitle: true,
      ),
      body: Column(
        children: [
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: TextField(
              controller: _titleController,
              decoration: const InputDecoration(labelText: 'Title'),
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: TextField(
              controller: _descriptionController,
              decoration: const InputDecoration(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(
                    mainAxisSize: MainAxisSize.min,
                    children: [
                      IconButton(
                        icon: const Icon(Icons.edit),
                        onPressed: () => _showEditDialog(item),
                      ),
                      IconButton(
                        icon: const Icon(Icons.delete),
                        onPressed: () => _deleteData(item['id']),
                      ),
                    ],
                  ),
                );
              },
            ),
          ),
        ],
      ),
    );
  }
}
```

Deskripsi progam:

Program ini adalah aplikasi Flutter berbasis SQLite untuk mengelola data sederhana (CRUD: Create, Read, Update, Delete). Berikut adalah fitur utama:

1. Fungsi CRUD:
   - Tambah data: Menggunakan form input untuk menambahkan data baru ke database.
   - Baca data: Menampilkan daftar data dari database dalam bentuk ListView.
   - Edit data: Mengubah data dengan dialog pengeditan.
   - Hapus data: Menghapus data dari database.
2. UI dan Pengalaman Pengguna:
   - Input data dilakukan melalui TextField untuk judul dan deskripsi.
   - Daftar data ditampilkan dengan opsi edit dan hapus melalui tombol IconButton.
   - Dialog validasi muncul jika input kosong atau saat mengedit data.
3. Database Management:
   - Database SQLite diakses menggunakan DatabaseHelper.
   - Data otomatis diperbarui saat ada perubahan (tambah, edit, atau hapus).

   Program ini adalah contoh implementasi CRUD untuk aplikasi lokal dengan tampilan yang sederhana namun fungsional.

C. Main

Source code:

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Praktikum Database',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // TRY THIS: Try running your application with "flutter run". You'll see
        // the application has a purple toolbar. Then, without quitting the app,
        // try changing the seedColor in the colorScheme below to Colors.green
        // and then invoke "hot reload" (save your changes or press the "hot
        // reload" button in a Flutter-supported IDE, or press "r" if you used
        // the command line to start the app).
        //
        // Notice that the counter didn't reset back to zero; the application
        // state is not lost during the reload. To reset the state, use hot
        // restart instead.
        //
        // This works for code too, not just values: Most code changes can be
        // tested with just a hot reload.
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const MyDatabaseView(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});

  // This widget is the home page of your application. It is stateful, meaning
  // that it has a State object (defined below) that contains fields that affect
  // how it looks.

  // This class is the configuration for the state. It holds the values (in this
  // case the title) provided by the parent (in this case the App widget) and
  // used by the build method of the State. Fields in a Widget subclass are
  // always marked "final".

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      // This call to setState tells the Flutter framework that something has
      // changed in this State, which causes it to rerun the build method below
      // so that the display can reflect the updated values. If we changed
      // _counter without calling setState(), then the build method would not be
      // called again, and so nothing would appear to happen.
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    // This method is rerun every time setState is called, for instance as done
    // by the _incrementCounter method above.
```

```
    //
    // The Flutter framework has been optimized to make rerunning build methods
    // fast, so that you can just rebuild anything that needs updating rather
    // than having to individually change instances of widgets.
    return Scaffold(
      appBar: AppBar(
        // TRY THIS: Try changing the color here to a specific color (to
        // Colors.amber, perhaps?) and trigger a hot reload to see the AppBar
        // change color while the other colors stay the same.
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        // Here we take the value from the MyHomePage object that was created by
        // the App.build method, and use it to set our appbar title.
        title: Text(widget.title),
      ),
      body: Center(
        // Center is a layout widget. It takes a single child and positions it
        // in the middle of the parent.
        child: Column(
          // Column is also a layout widget. It takes a list of children and
          // arranges them vertically. By default, it sizes itself to fit its
          // children horizontally, and tries to be as tall as its parent.
          //
          // Column has various properties to control how it sizes itself and
          // how it positions its children. Here we use mainAxisAlignment to
          // center the children vertically; the main axis here is the vertical
          // axis because Columns are vertical (the cross axis would be
          // horizontal).
          //
          // TRY THIS: Invoke "debug painting" (choose the "Toggle Debug Paint"
          // action in the IDE, or press "p" in the console), to see the
          // wireframe for each widget.
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            const Text(
              'You have pushed the button this many times:',
            ),
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headlineMedium,
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ), // This trailing comma makes auto-formatting nicer for build methods.
    );
  }
}
```

Deskripsi program:

Program ini adalah aplikasi Flutter yang menampilkan dua fitur utama: fungsi penghitung sederhana dan tampilan manajemen database menggunakan SQLite. Berikut adalah deskripsi singkatnya:

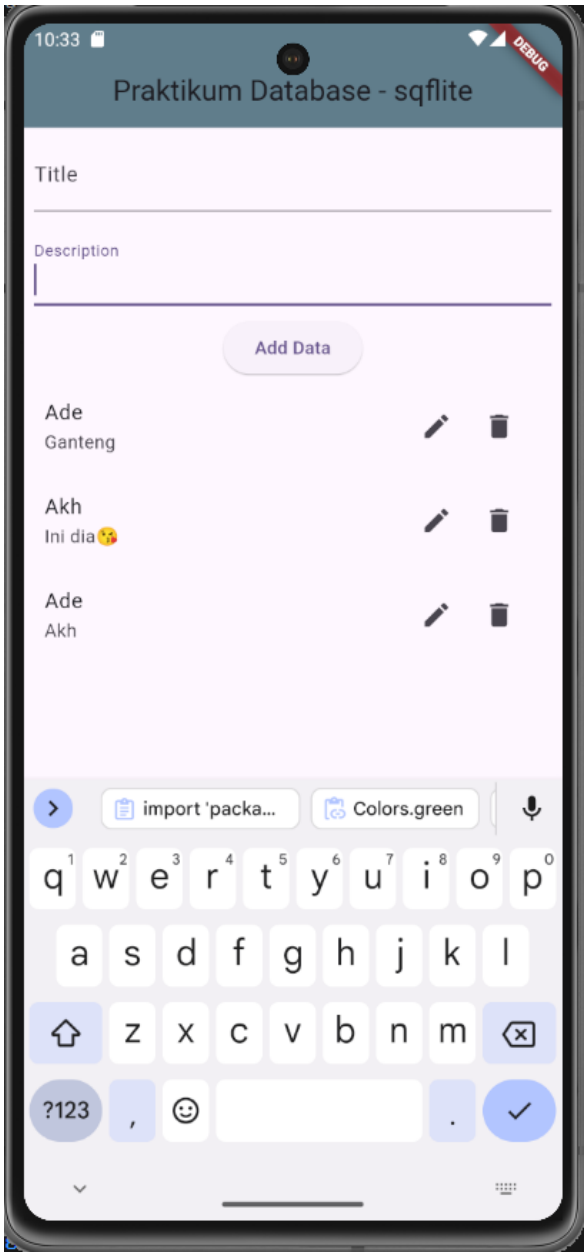
1. Fitur Penghitung (Counter):
    - Bagian MyHomePage menunjukkan antarmuka sederhana dengan penghitung angka yang bertambah setiap kali tombol dengan ikon plus (+) ditekan.
    - Fitur ini hanya sebagai contoh penggunaan *state management* dengan setState di Flutter.
2. Manajemen Database (CRUD):
    - Komponen MyDatabaseView (diatur sebagai halaman utama aplikasi) digunakan untuk menampilkan dan mengelola data berbasis SQLite.

- Menampilkan, menambah, memperbarui, dan menghapus data di dalam database lokal menggunakan class DatabaseHelper.

3. Tampilan Utama (Main):
   - MyApp adalah entri utama aplikasi yang menentukan tema dan tampilan awal menggunakan MaterialApp.
   - Aplikasi ini mendukung tema *Material Design 3*, dengan tampilan modern dan responsif.

   Fokus aplikasi ini adalah pembelajaran dasar pengelolaan database lokal di Flutter dengan tambahan contoh fitur penghitung untuk memperkenalkan *stateful widgets*.

Output:

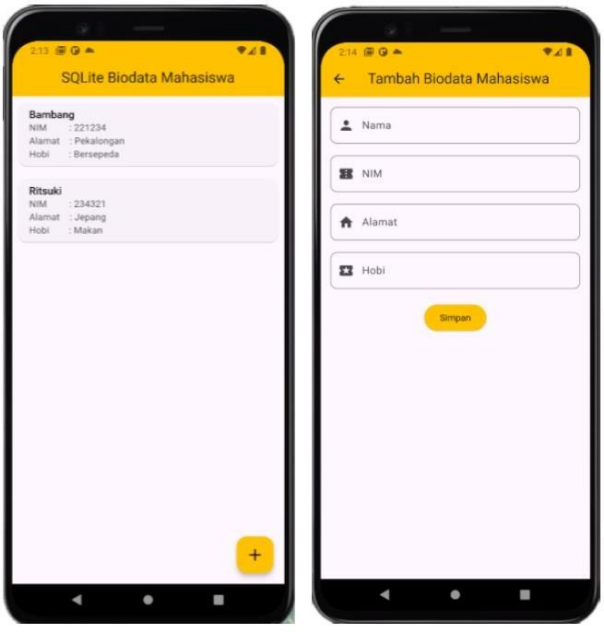


UNGUIDED

1. (Soal) Buatlah sebuah project aplikasi Flutter dengan SQLite untuk menyimpan data biodata mahasiswa yang terdiridari nama, NIM, domisili, dan hobi. Data yang dimasukkan melalui form akan ditampilkan dalam daftar di halamanutama.

   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 sudahdisimpan di halaman utama.
c. Implementasikan fitur Create (untuk menyimpan data mahasiswa) dan Read (untuk menampilkan daftar mahasiswayang sudah disimpan).
d. Contoh output:



Source Code Unguided

1. db_helper.dart

```
// ignore: depend_on_referenced_packages
import 'package:sqflite/sqflite.dart';
// ignore: depend_on_referenced_packages
import 'package:path/path.dart';
import 'student_model.dart';

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

  static Database? _database;

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

  Future<Database> _initDB() async {
    String path = join(await getDatabasesPath(), 'students.db');
    return await openDatabase(path, version: 1, onCreate: _onCreate);
  }

  Future<void> _onCreate(Database db, int version) async {
    await db.execute('''
      CREATE TABLE students(
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        name TEXT,
        nim TEXT,
        address TEXT,
        hobby TEXT
      )
    ''');
  }

  Future<int> insertStudent(Student student) async {
    final db = await database;
    return await db.insert('students', student.toMap());
  }

  Future<List<Student>> getAllStudents() async {
    final db = await database;
    final List<Map<String, dynamic>> result = await db.query('students');
    return result.map((data) => Student(
      id: data['id'],
      name: data['name'],
      nim: data['nim'],
      address: data['address'],
      hobby: data['hobby'],
    )).toList();
  }

  openDatabase(String path, {required int version, required Future<void>
Function(Database db, int version) onCreate}) {}
}

class Database {
  execute(String s) {}
```

```
  insert(String s, Map<String, dynamic> map) {}

  query(String s) {}
}
```

2. student_model.dart

```
class Student {
  final int? id;
  final String name;
  final String nim;
  final String address;
  final String hobby;

  Student({this.id, required this.name, required this.nim, required this.address,
   required this.hobby});

  Map<String, dynamic> toMap() {
    return {
      'id': id,
      'name': name,
      'nim': nim,
      'address': address,
      'hobby': hobby,
    };
  }
}
```

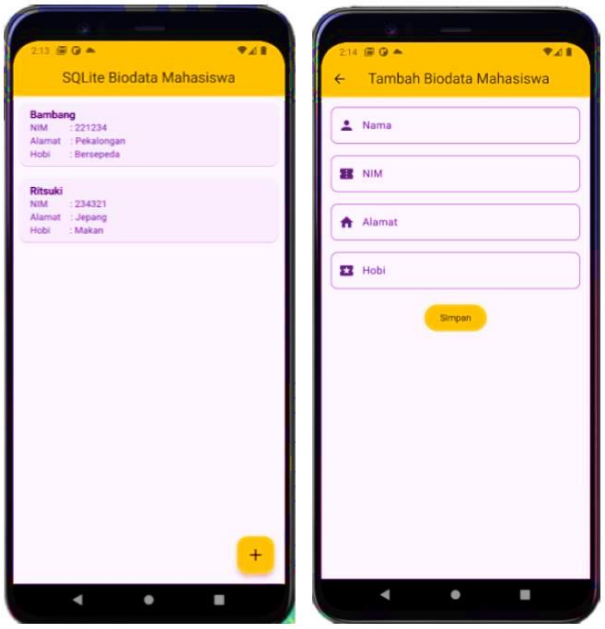
3. main.dart

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'SQLite Biodata Mahasiswa',
      theme: ThemeData(primarySwatch: Colors.orange),
      home: const StudentListPage(),
    );
  }
}

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

  @override
  State<StudentListPage> createState() => _StudentListPageState();
}

class _StudentListPageState extends State<StudentListPage> {
  final DBHelper dbHelper = DBHelper();
```

```
List<Student> students = [];

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

void _loadStudents() async {
  final data = await dbHelper.getAllStudents();
  setState(() {
    students = data;
  });
}

void _addStudent(Student student) async {
  await dbHelper.insertStudent(student);
  _loadStudents();
}

void _showAddStudentDialog() {
  final nameController = TextEditingController();
  final nimController = TextEditingController();
  final addressController = TextEditingController();
  final hobbyController = TextEditingController();

  showDialog(
    context: context,
    builder: (context) {
      return AlertDialog(
        title: const Text('Tambah Biodata Mahasiswa'),
        content: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            TextField(
              controller: nameController,
              decoration: const InputDecoration(labelText: 'Nama'),
            ),
            TextField(
              controller: nimController,
              decoration: const InputDecoration(labelText: 'NIM'),
            ),
            TextField(
              controller: addressController,
              decoration: const InputDecoration(labelText: 'Alamat'),
            ),
            TextField(
              controller: hobbyController,
              decoration: const InputDecoration(labelText: 'Hobi'),
            ),
          ],
        ),
        actions: [
          TextButton(
            onPressed: () {
              Navigator.of(context).pop();
            },
            child: const Text('Batal'),
          ),
          TextButton(
            onPressed: () {
              final newStudent = Student(
                name: nameController.text,
```

```
                    nim: nimController.text,
                    address: addressController.text,
                    hobby: hobbyController.text,
                  );
                  _addStudent(newStudent);
                  Navigator.of(context).pop();
                },
                child: const Text('Simpan'),
              ),
            ],
          );
        },
      );
    }

    @override
    Widget build(BuildContext context) {
      return Scaffold(
        appBar: AppBar(
          title: const Text('SQLite Biodata Mahasiswa'),
          centerTitle: true,
        ),
        body: ListView.builder(
          itemCount: students.length,
          itemBuilder: (context, index) {
            final student = students[index];
            return ListTile(
              title: Text(student.name),
              subtitle: Text('NIM: ${student.nim}\nAlamat:
     ${student.address}\nHobi: ${student.hobby}'),
            );
          },
        ),
        floatingActionButton: FloatingActionButton(
          onPressed: _showAddStudentDialog,
          child: const Icon(Icons.add),
        ),
      );
    }
}
```

Ouput:



D. Kesimpulan

Melalui praktikum ini, kita berhasil membangun aplikasi Flutter yang mampu menyimpan dan menampilkan data biodata mahasiswa menggunakan SQLite. Hal ini memperlihatkan bagaimana mengintegrasikan database lokal dengan aplikasi Flutter, serta mengelola data dalam aplikasi berbasis mobile secara efisien. Praktikum ini juga meningkatkan pemahaman tentang konsep dasar pengelolaan database dalam pengembangan aplikasi mobile.