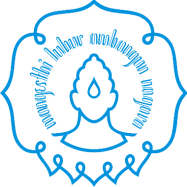
**PRATIK APLIKASI MOBILE**

***“Saving Employee”***

Modul Praktikum 10

****

**Disusun oleh :**

Diah Munica Nawang

V3922015 / TI D

**Dosen :**

Trisna Ari Roshinta, S.S.T., M.T

**PS D-III TEKNIK INFORMATIKA**

**SEKOLAH VOKASI**

**UNIVERSITAS SEBELAS MARET**

**2023**

**Laporan Proyek: Aplikasi "SAVING EMPLOYEE" dengan Flutter**

**I. Pendahuluan**

***1.1 Tujuan***

Laporan ini dibuat untuk memberikan pandangan menyeluruh tentang pengembangan aplikasi **"Saving Employee"** menggunakan Flutter, dengan tujuan menciptakan daftar kontak telepon yang dapat dikelola.

***1.2 Lingkup***

Laporan ini mencakup detail implementasi dasar aplikasi, termasuk struktur folder, komponen utama, dan langkah-langkah pengembangan selanjutnya.

**II. Implementasi Dasar**

***2.1 Struktur Folder***

Struktur folder proyek diatur sebagai berikut:

```

lib/

|-- main.dart

|-- employeelist.dart

|-- database/

| |-- dbhelper.dart

|-- model/

| |-- employee.dart

```

 🡺Implementasi

***2.2 Deskripsi File Utama***

***2.2.1 `main.dart`***

|  |
| --- |
| **import** 'package:flutter/material.dart'; *// Import pustaka Flutter untuk UI berbasis material design.*  **import** 'package:pegawai/database/dbhelper.dart'; *// Import file dbhelper.dart untuk interaksi dengan database.*  **import** 'package:pegawai/model/employee.dart'; *// Import model Employee dari file employee.dart.*  **import** 'package:pegawai/employeelist.dart'; *// Import file employeelist.dart yang berisi implementasi tampilan daftar karyawan.*    void main() => runApp(MyApp());    **class** MyApp extends StatelessWidget {  @**override**  Widget build(BuildContext context) {  *// Mendefinisikan tema aplikasi dan halaman utama.*  **return** MaterialApp(  title: 'SQFLite DataBase Demo',  theme: ThemeData(  primarySwatch: Colors.blue,  ),  home: MyHomePage(title: 'Flutter Demo Home Page'),  );  }  }    **class** MyHomePage extends StatefulWidget {  MyHomePage({Key? key, **this**.title}) : **super**(key: key);  **final** String? title;    @**override**  \_MyHomePageState createState() => \_MyHomePageState();  }    **class** \_MyHomePageState extends State<MyHomePage> {  Employee employee = Employee(firstName: '', lastName: '', mobileNo: '');  late String firstname, lastname, mobileno;    **final** scaffoldKey = GlobalKey<ScaffoldState>();  **final** formKey = GlobalKey<FormState>();    @**override**  Widget build(BuildContext context) {  *// Membuat tampilan Scaffold dengan Appbar dan Form.*  **return** Scaffold(  key: scaffoldKey,  appBar: AppBar(  title: Text('Saving Employee'),  actions: <Widget>[  *// Tombol navigasi ke halaman daftar karyawan.*  IconButton(  icon: **const** Icon(Icons.view\_list),  tooltip: 'Next choice',  onPressed: navigateToEmployeeList,  ),  ],  ),  body: Padding(  padding: **const** EdgeInsets.all(16.0),  child: Form(  key: formKey,  child: ListView(  children: [  *// Input teks untuk memasukkan First Name.*  TextFormField(  keyboardType: TextInputType.text,  decoration: InputDecoration(labelText: 'First Name'),  validator: (**val**) => **val**?.length == 0 ? "Enter FirstName" : **null**,  onSaved: (**val**) => firstname = **val**!,  ),  *// Input teks untuk memasukkan Last Name.*  TextFormField(  keyboardType: TextInputType.text,  decoration: InputDecoration(labelText: 'Last Name'),  validator: (**val**) => **val**?.length == 0 ? 'Enter LastName' : **null**,  onSaved: (**val**) => lastname = **val**!,  ),  *// Input teks untuk memasukkan Mobile No.*  TextFormField(  keyboardType: TextInputType.phone,  decoration: InputDecoration(labelText: 'Mobile No'),  validator: (**val**) => **val**?.length == 0 ? 'Enter Mobile No' : **null**,  onSaved: (**val**) => mobileno = **val**!,  ),  *// Tombol untuk menyimpan data karyawan.*  Container(  margin: **const** EdgeInsets.only(top: 10.0),  child: ElevatedButton(  onPressed: \_submit,  child: Text('Save Employee'),  ),  )  ],  ),  ),  ),  );  }    *// Metode yang dipanggil saat tombol "Save Employee" ditekan.*  void \_submit() async {  *// Memvalidasi form sebelum menyimpan data.*  **if** (formKey.currentState!.validate()) {  formKey.currentState!.save();    *// Membuat objek DBHelper untuk berinteraksi dengan database.*  **var** dbHelper = DBHelper();    *// Menyimpan data karyawan ke dalam database.*  await dbHelper.saveEmployee(Employee(  firstName: firstname,  lastName: lastname,  mobileNo: mobileno,  ));    *// Menampilkan snackbar untuk memberi tahu pengguna bahwa data berhasil disimpan.*  \_showSnackBar("Data saved successfully");  }  }    *// Metode untuk menampilkan snackbar.*  void \_showSnackBar(String text) {  ScaffoldMessenger.of(context).showSnackBar(  SnackBar(content: Text(text)),  );  }    *// Metode untuk navigasi ke halaman daftar karyawan.*  void navigateToEmployeeList() {  Navigator.push(  context,  MaterialPageRoute(builder: (context) => MyEmployeeList()),  );  }  } |

***2.2.2 employeelist.dart***

|  |
| --- |
| **import** 'package:flutter/material.dart'; *// Mendeklarasikan ketergantungan pada pustaka Flutter untuk pengembangan antarmuka pengguna (UI) berbasis material design.*  **import** 'package:pegawai/model/employee.dart'; *// Mengimpor definisi model Employee dari file employee.dart*  **import** 'dart:async'; *// Mengimpor pustaka dart:async untuk menggunakan fitur-fitur seperti Future dan Stream, yang sering digunakan untuk menangani operasi asynchronous.*  **import** 'package:pegawai/database/dbhelper.dart'; *// Mengimpor implementasi DBHelper dari file dbhelper.dart*    *// Widget utama yang menampilkan daftar employee.*  **class** MyEmployeeList extends StatefulWidget {  @**override**  MyEmployeeListPageState createState() => MyEmployeeListPageState();  }    **class** MyEmployeeListPageState extends State<MyEmployeeList> {  *// Inisialisasi objek DBHelper yang akan digunakan untuk berinteraksi dengan database.*  late DBHelper dbHelper;    @**override**  void initState() {  *// Method yang dipanggil ketika state dari widget ini dibuat.*  **super**.initState();  dbHelper = DBHelper();  *// Inisialisasi DBHelper saat widget dibuat.*  }    Future<List<Employee>> fetchEmployeesFromDatabase() async {  *// Method untuk mengambil daftar employee dari database menggunakan DBHelper.*  **return** dbHelper.getEmployees();  }    @**override**  Widget build(BuildContext context) {  *// Membuat tampilan utama widget menggunakan Scaffold.*  **return** Scaffold(  appBar: AppBar(  title: Text('Employee List'),  ),  body: Container(  padding: EdgeInsets.all(16.0),  *// Menggunakan FutureBuilder untuk menampilkan data ketika sudah selesai diambil.*  child: FutureBuilder<List<Employee>>(  future: fetchEmployeesFromDatabase(),  builder: (context, snapshot) {  **if** (snapshot.connectionState == ConnectionState.waiting) {  *// Menampilkan indikator loading jika data masih diambil.*  **return** Center(child: CircularProgressIndicator());  } **else** **if** (snapshot.hasError) {  *// Menampilkan pesan error jika terjadi kesalahan.*  **return** Center(child: Text("Error: ${snapshot.error}"));  } **else** **if** (!snapshot.hasData || snapshot.**data**!.isEmpty) {  *// Menampilkan pesan jika tidak ada data employee.*  **return** Center(child: Text("No employees found."));  } **else** {  *// Menampilkan daftar employee menggunakan ListView.builder.*  **return** ListView.builder(  itemCount: snapshot.**data**!.length,  itemBuilder: (context, index) {  **return** Column(  crossAxisAlignment: CrossAxisAlignment.start,  children: <Widget>[  Row(  mainAxisAlignment: MainAxisAlignment.spaceBetween,  children: [  Text(  'No. ${snapshot.data![index].id}. ${snapshot.data![index].firstName} ${snapshot.data![index].lastName}',  style: TextStyle(  fontWeight: FontWeight.bold,  fontSize: 18.0,  ),  ),  Row(  children: [  IconButton(  icon: Icon(Icons.edit),  onPressed: () {  \_editEmployee(snapshot.**data**![index]);  },  ),  IconButton(  icon: Icon(Icons.delete),  onPressed: () {  \_deleteEmployee(snapshot.**data**![index]);  },  ),  ],  ),  ],  ),  Text(  'Mobile No: ${snapshot.data![index].mobileNo}',  style: TextStyle(  fontWeight: FontWeight.bold,  fontSize: 14.0,  ),  ),  Divider(),  ],  );  },  );  }  },  ),  ),  );  }    void \_editEmployee(Employee employee) {  *// Pindah ke halaman edit employee saat tombol edit ditekan.*  Navigator.push(  context,  MaterialPageRoute(  builder: (context) => EditEmployeeScreen(  employee: employee,  onEmployeeUpdated: \_updateEmployeeList,  ),  ),  );  }    void \_deleteEmployee(Employee employee) {  *// Menampilkan dialog konfirmasi penghapusan employee.*  showDialog(  context: context,  builder: (BuildContext context) {  **return** AlertDialog(  title: Text("Confirm Delete"),  content: Text("Are you sure you want to delete this employee?"),  actions: [  TextButton(  onPressed: () {  Navigator.pop(context);  },  child: Text("Cancel"),  ),  TextButton(  onPressed: () async {  *// Menghapus employee dari database dan memperbarui tampilan.*  await dbHelper.deleteEmployee(employee.id!);  Navigator.pop(context);  \_updateEmployeeList();  },  child: Text("Delete"),  ),  ],  );  },  );  }    void \_updateEmployeeList() {  *// Memperbarui tampilan daftar employee.*  setState(() {});  }  }    *// Halaman untuk mengedit data employee.*  **class** EditEmployeeScreen extends StatefulWidget {  **final** Employee employee;  **final** Function() onEmployeeUpdated;    EditEmployeeScreen({required **this**.employee, required **this**.onEmployeeUpdated});    @**override**  \_EditEmployeeScreenState createState() => \_EditEmployeeScreenState();  }    **class** \_EditEmployeeScreenState extends State<EditEmployeeScreen> {  *// Controller untuk mengelola input teks pada form.*  late TextEditingController firstNameController;  late TextEditingController lastNameController;  late TextEditingController mobileNoController;    @**override**  void initState() {  *// Method yang dipanggil ketika state dari widget ini dibuat.*  **super**.initState();  *// Mengisi controller dengan data employee yang akan diubah.*  firstNameController = TextEditingController(text: widget.employee.firstName);  lastNameController = TextEditingController(text: widget.employee.lastName);  mobileNoController = TextEditingController(text: widget.employee.mobileNo);  }    @**override**  Widget build(BuildContext context) {  *// Membuat tampilan halaman edit employee.*  **return** Scaffold(  appBar: AppBar(  title: Text('Edit Employee'),  ),  body: Padding(  padding: **const** EdgeInsets.all(16.0),  child: Column(  crossAxisAlignment: CrossAxisAlignment.start,  children: [  *// Input teks untuk mengedit first name.*  TextField(  controller: firstNameController,  decoration: InputDecoration(labelText: 'First Name'),  ),  *// Input teks untuk mengedit last name.*  TextField(  controller: lastNameController,  decoration: InputDecoration(labelText: 'Last Name'),  ),  *// Input teks untuk mengedit nomor telepon.*  TextField(  controller: mobileNoController,  decoration: InputDecoration(labelText: 'Mobile No'),  ),  SizedBox(height: 16.0),  *// Tombol untuk menyimpan perubahan pada data employee.*  ElevatedButton(  onPressed: () {  \_updateEmployee();  },  child: Text('Save Changes'),  ),  ],  ),  ),  );  }    *// Method untuk menyimpan perubahan data employee ke database.*  Future<void> \_updateEmployee() async {  *// Membuat objek Employee baru dengan data yang telah diubah.*  Employee updatedEmployee = Employee(  id: widget.employee.id,  firstName: firstNameController.text,  lastName: lastNameController.text,  mobileNo: mobileNoController.text,  );    *// Memanggil method updateEmployee dari DBHelper untuk menyimpan perubahan ke database.*  await DBHelper().updateEmployee(updatedEmployee);  widget.onEmployeeUpdated(); *// Memanggil callback untuk merefresh tampilan daftar employee.*  Navigator.pop(context);  }  } |

### ***2.3 Implementasi Fungsionalitas Dasar***

#### *2.3.1 Model - employee.dart*

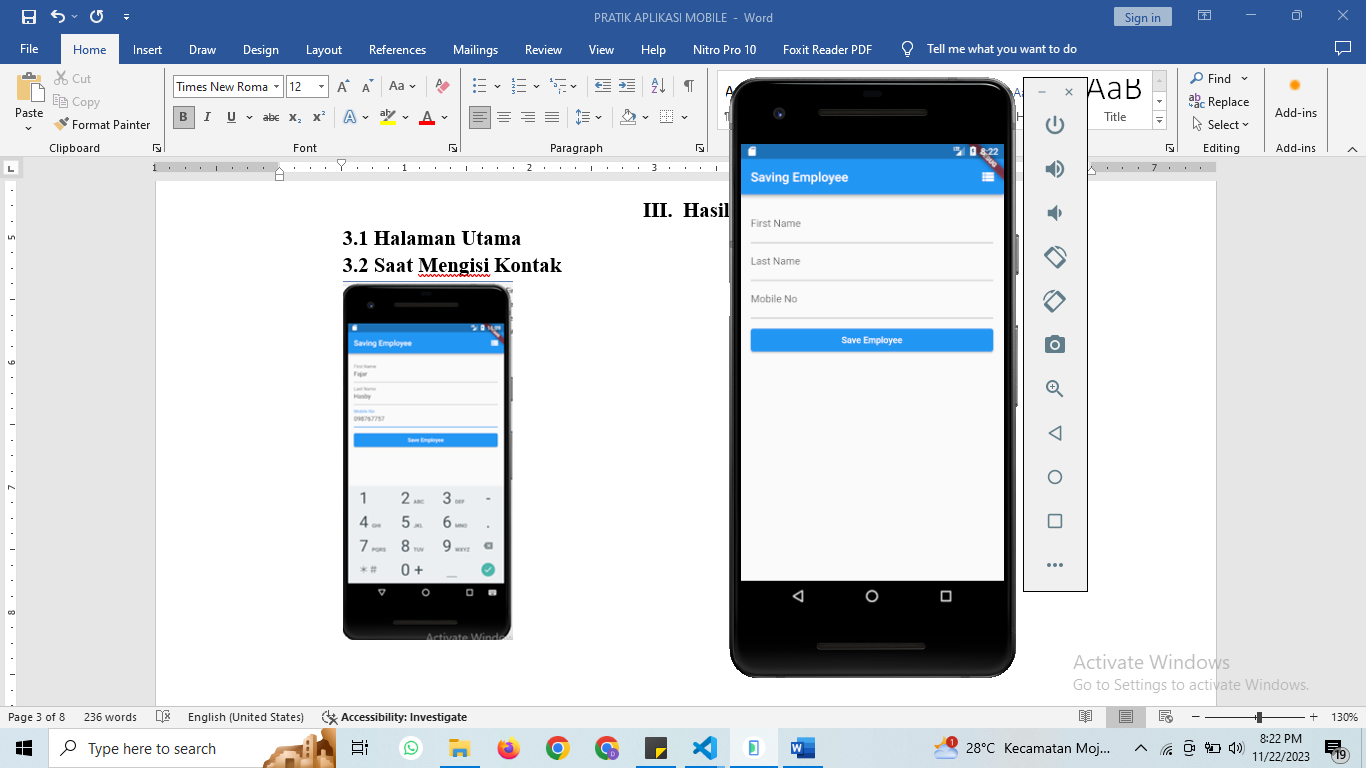
|  |
| --- |
| **class** Employee {  int? id;  **final** String firstName;  **final** String lastName;  **final** String mobileNo;    *// Konstruktor untuk membuat instance dari kelas Employee.*  Employee({  **this**.id,  required **this**.firstName,  required **this**.lastName,  required **this**.mobileNo,  });    *// Metode setter untuk id.*  **set** setId(int? value) {  id = value;  }    *// Metode statis untuk membuat objek Employee dari peta.*  static Employee fromMap(Map<String, Object?> map) => Employee(  id: map["\_id"] **as** int?,  firstName: map["firstname"] **as** String,  lastName: map["lastname"] **as** String,  mobileNo: map["mobileno"] **as** String,  );  } |

#### 2.3.2 Database - dbhelper.dart

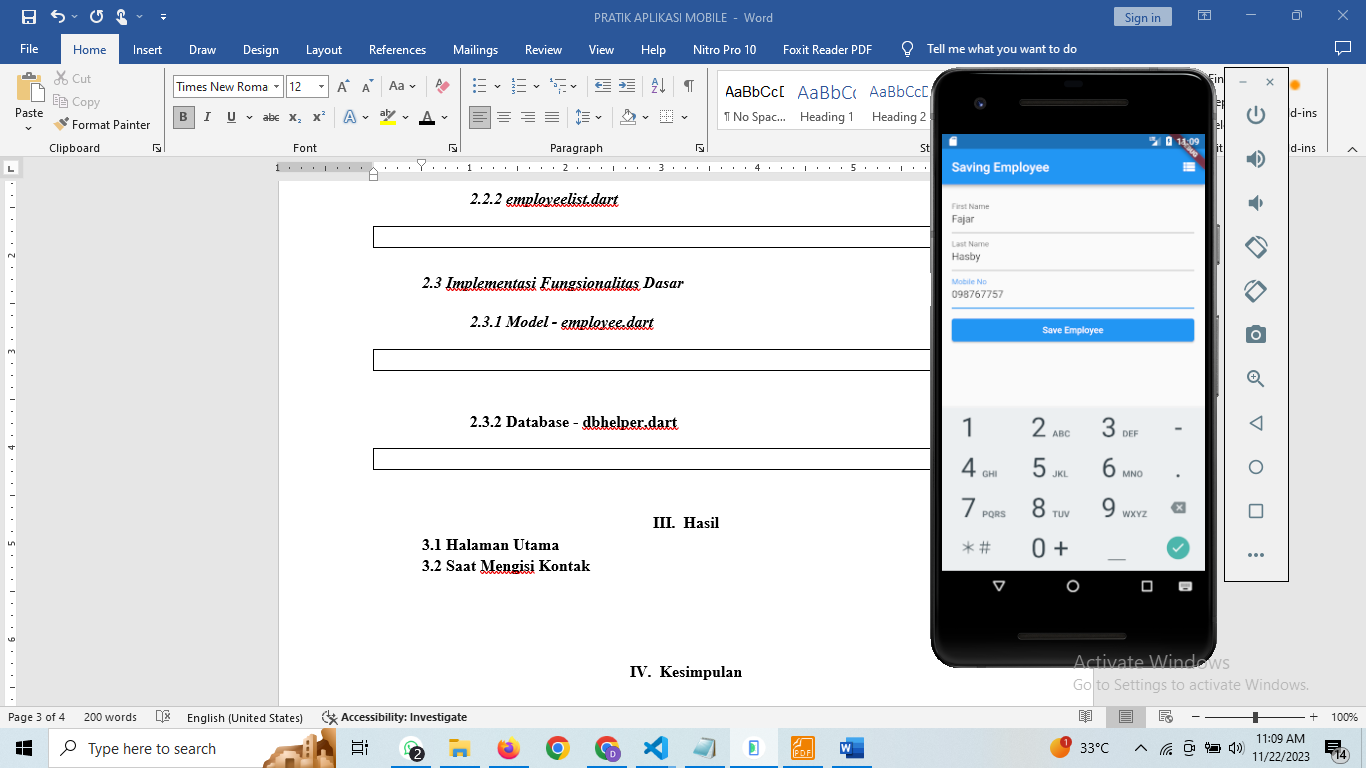
|  |
| --- |
| **import** 'dart:async'; *// Mengimpor pustaka untuk mendukung asynchronous programming dengan menggunakan Future dan Stream.*  **import** 'dart:io' **as** io; *// Mengimpor pustaka I/O untuk mengakses sistem file.*  **import** 'package:path/path.dart'; *// Mengimpor fungsi join dari pustaka path untuk menggabungkan path direktori.*  **import** 'package:sqflite/sqflite.dart'; *// Mengimpor pustaka sqflite untuk berinteraksi dengan database SQLite.*  **import** 'package:path\_provider/path\_provider.dart'; *// Mengimpor pustaka untuk mendapatkan path dari direktori aplikasi.*  **import** 'package:pegawai/model/employee.dart'; *// Mengimpor definisi atau model dari karyawan (employee) yang kemungkinan didefinisikan dalam file employee.dart.*    **class** DBHelper {  *// Deklarasi variabel statik \_db untuk menyimpan instance Database.*  static Database? \_db;    *// Fungsi getter async untuk mendapatkan instance Database.*  Future<Database> **get** db async {  *// Jika \_db sudah diinisialisasi, kembalikan nilai \_db.*  **if** (\_db != **null**) **return** \_db!;    *// Jika belum diinisialisasi, panggil fungsi initDb untuk membuat dan membuka database.*  \_db = await initDb();  **return** \_db!;  }    *// Fungsi async untuk menginisialisasi dan membuka database.*  Future<Database> initDb() async {  *// Mendapatkan direktori dokumen aplikasi.*  io.Directory documentsDirectory = await getApplicationDocumentsDirectory();    *// Menyusun path lengkap untuk database.*  **final** String path = join(documentsDirectory.path, "employee.db");    *// Membuka database atau membuatnya jika belum ada.*  **var** theDb = await openDatabase(path, version: 1, onCreate: \_onCreate);  **return** theDb;  }    *// Metode \_onCreate yang dipanggil saat database dibuat untuk pertama kali.*  void \_onCreate(Database db, int version) async {  *// Saat membuat database, buat tabel Employee.*  await db.execute(  "CREATE TABLE Employee(\_id INTEGER PRIMARY KEY AUTOINCREMENT, firstname TEXT NOT NULL, lastname TEXT NOT NULL, mobileno TEXT NOT NULL)");  print("Created tables");  }    *// Metode untuk menyimpan data karyawan ke dalam database.*  Future<void> saveEmployee(Employee employee) async {  *// Mendapatkan instance Database.*  **var** dbClient = await db;    **try** {  *// Melakukan transaksi untuk memastikan keamanan operasi database.*  await dbClient.transaction((txn) async {  *// Menjalankan query SQL untuk menyimpan data karyawan.*  **return** await txn.rawInsert(  'INSERT INTO Employee(firstname, lastname, mobileno) VALUES(?, ?, ?)',  [employee.firstName, employee.lastName, employee.mobileNo],  );  });  } **catch** (e) {  *// Menangani kesalahan, misalnya dengan mencetak pesan atau melempar kembali error.*  print("Error inserting employee: $e");  }  }    *// Metode untuk mendapatkan daftar semua karyawan dari database.*  Future<List<Employee>> getEmployees() async {  *// Mendapatkan instance Database.*  **var** dbClient = await db;    *// Menjalankan query SQL untuk mengambil semua data karyawan.*  **final** list = await dbClient.rawQuery('SELECT \* FROM Employee');    *// Mengonversi hasil query ke dalam objek Employee dan mengembalikannya dalam bentuk list.*  **return** list.map((json) => Employee.fromMap(json)).toList();  }    *// Metode untuk memperbarui data karyawan dalam database.*  Future<void> updateEmployee(Employee employee) async {  *// Mendapatkan instance Database.*  **var** dbClient = await db;    **try** {  *// Melakukan transaksi untuk memastikan keamanan operasi database.*  await dbClient.transaction((txn) async {  *// Menjalankan query SQL untuk memperbarui data karyawan.*  await txn.rawUpdate(  'UPDATE Employee SET firstname = ?, lastname = ?, mobileno = ? WHERE \_id = ?',  [employee.firstName, employee.lastName, employee.mobileNo, employee.id],  );  });  } **catch** (e) {  *// Menangani kesalahan, misalnya dengan mencetak pesan atau melempar kembali error.*  print("Error updating employee: $e");  }  }    *// Metode untuk menghapus data karyawan dari database berdasarkan ID.*  Future<void> deleteEmployee(int id) async {  *// Mendapatkan instance Database.*  **var** dbClient = await db;    **try** {  *// Melakukan transaksi untuk memastikan keamanan operasi database.*  await dbClient.transaction((txn) async {  *// Menjalankan query SQL untuk menghapus data karyawan berdasarkan ID.*  await txn.rawDelete('DELETE FROM Employee WHERE \_id = ?', [id]);  });  } **catch** (e) {  *// Menangani kesalahan, misalnya dengan mencetak pesan atau melempar kembali error.*  print("Error deleting employee: $e");  }  }    *// Metode untuk menutup koneksi ke database.*  Future<void> close() async {  **var** dbClient = await db;  dbClient.close();  }  } |

**III. Hasil**

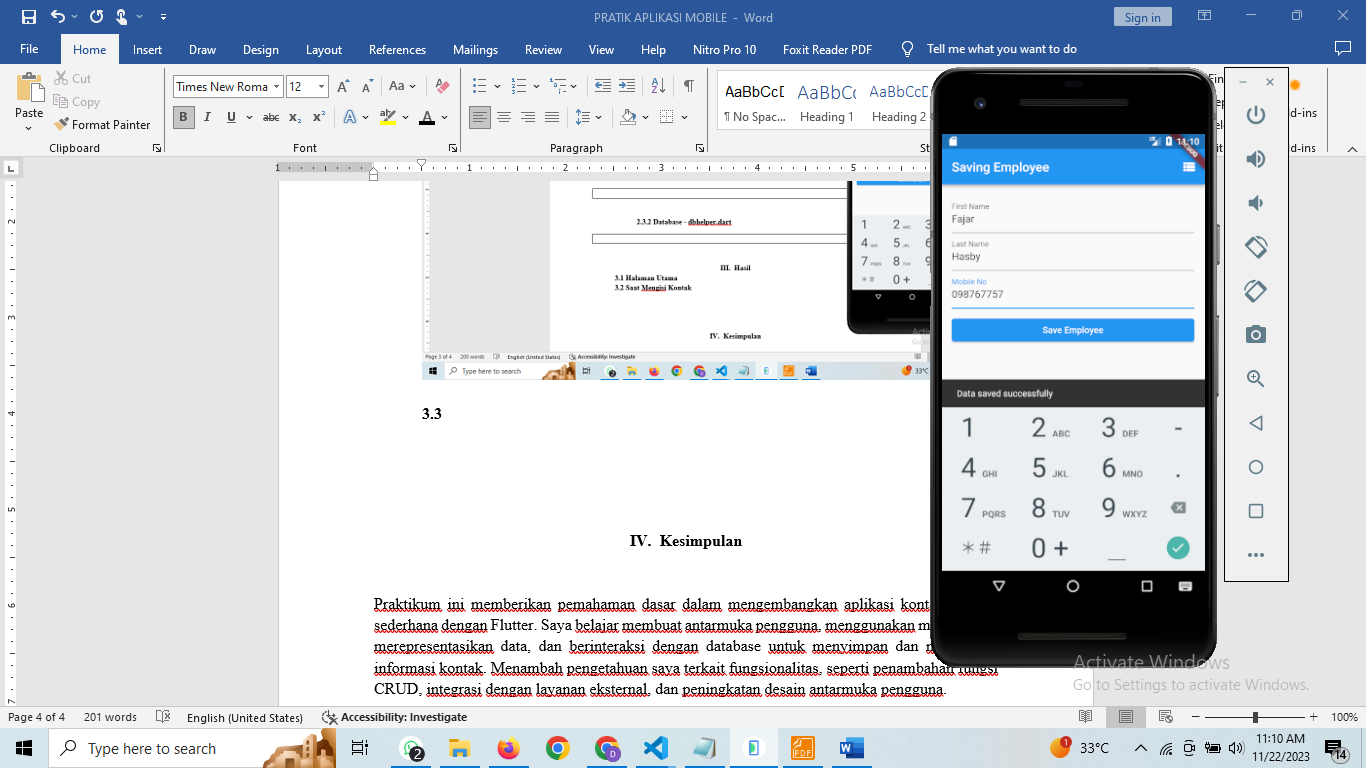
**3.1 Halaman Utama**



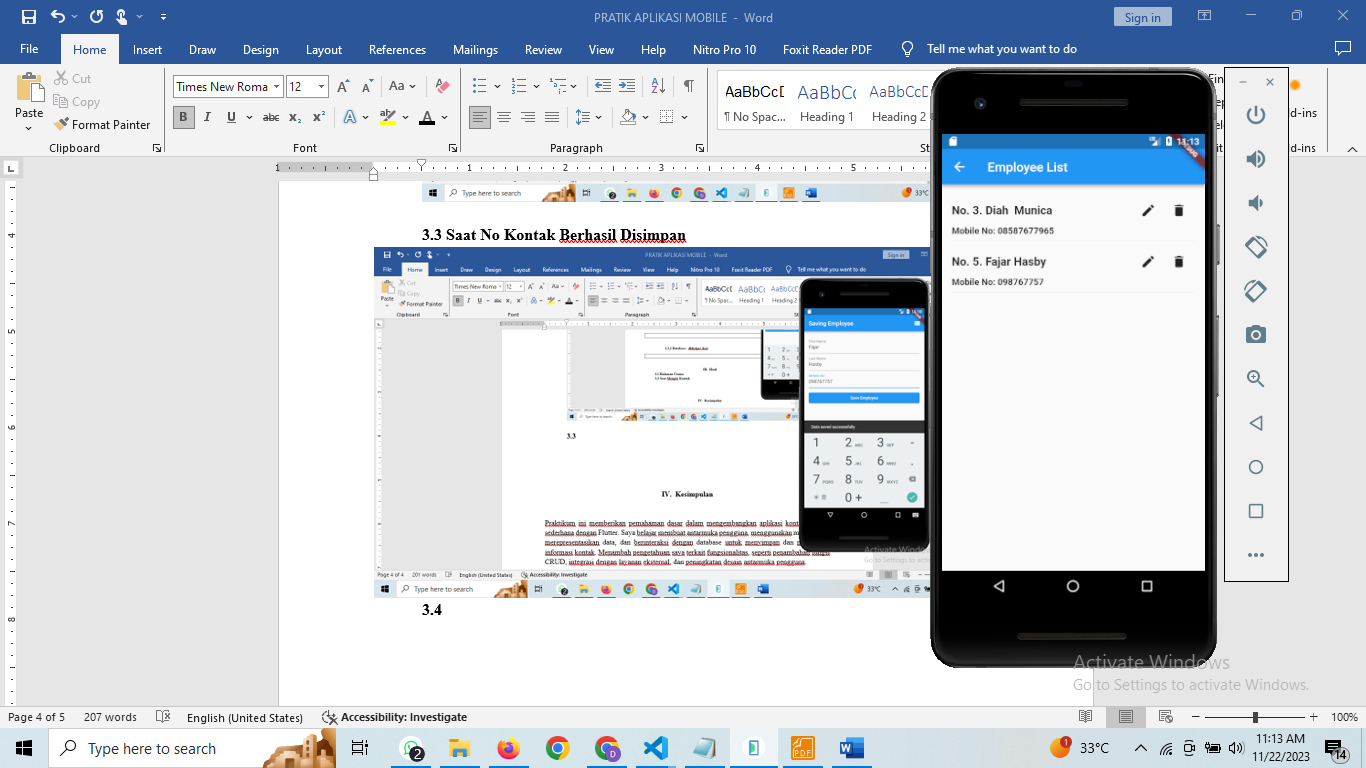
**3.2 Saat Mengisi Kontak**



**3.3 Saat No Kontak Berhasil Disimpan**

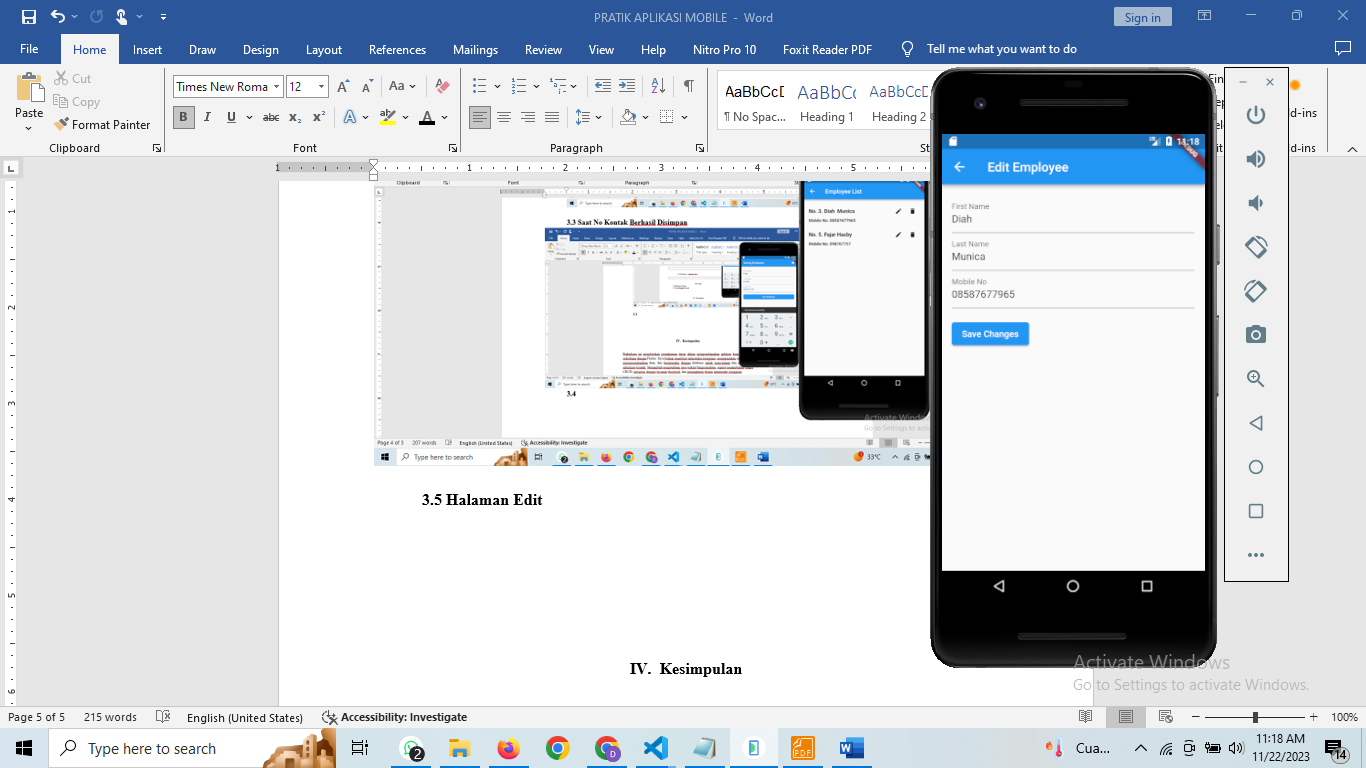


**3.4 List Kontak Yang Sudah Tersimpan**

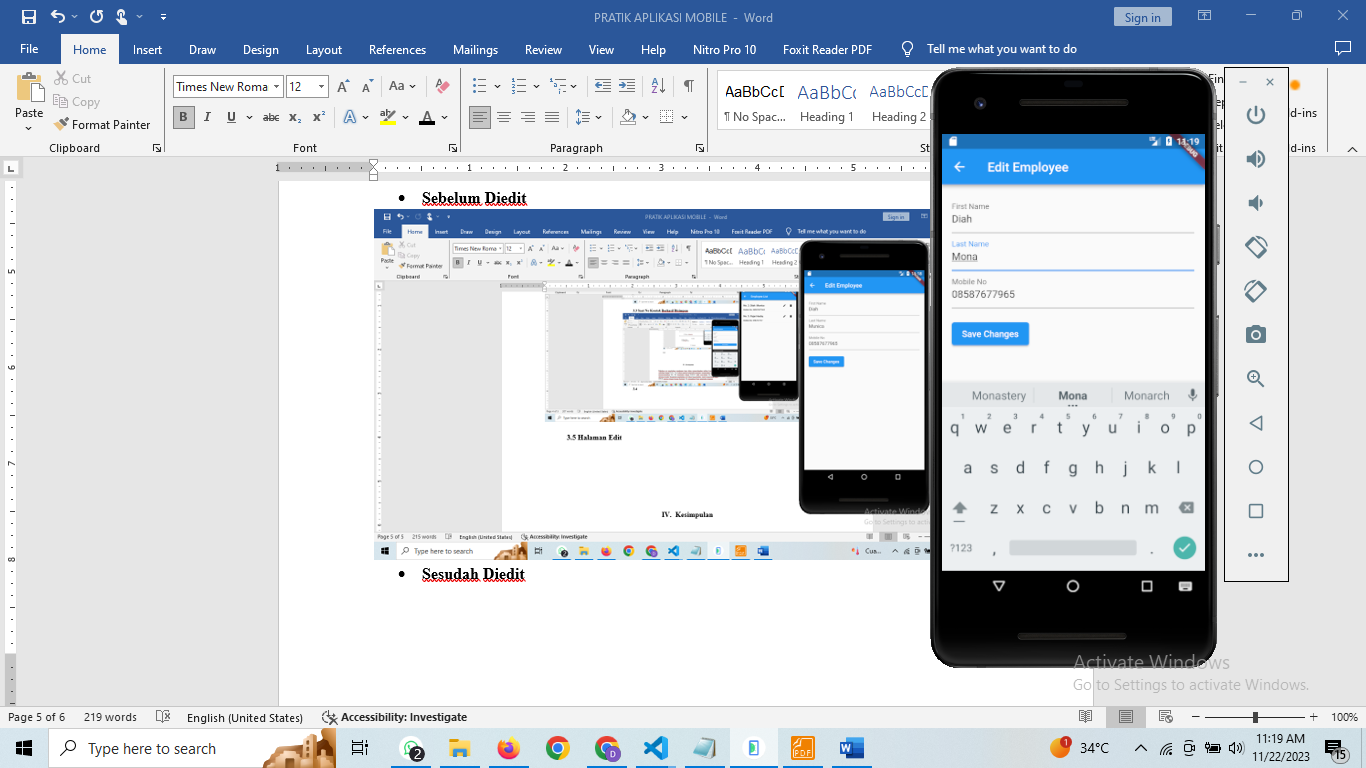


**3.5 Halaman Edit**

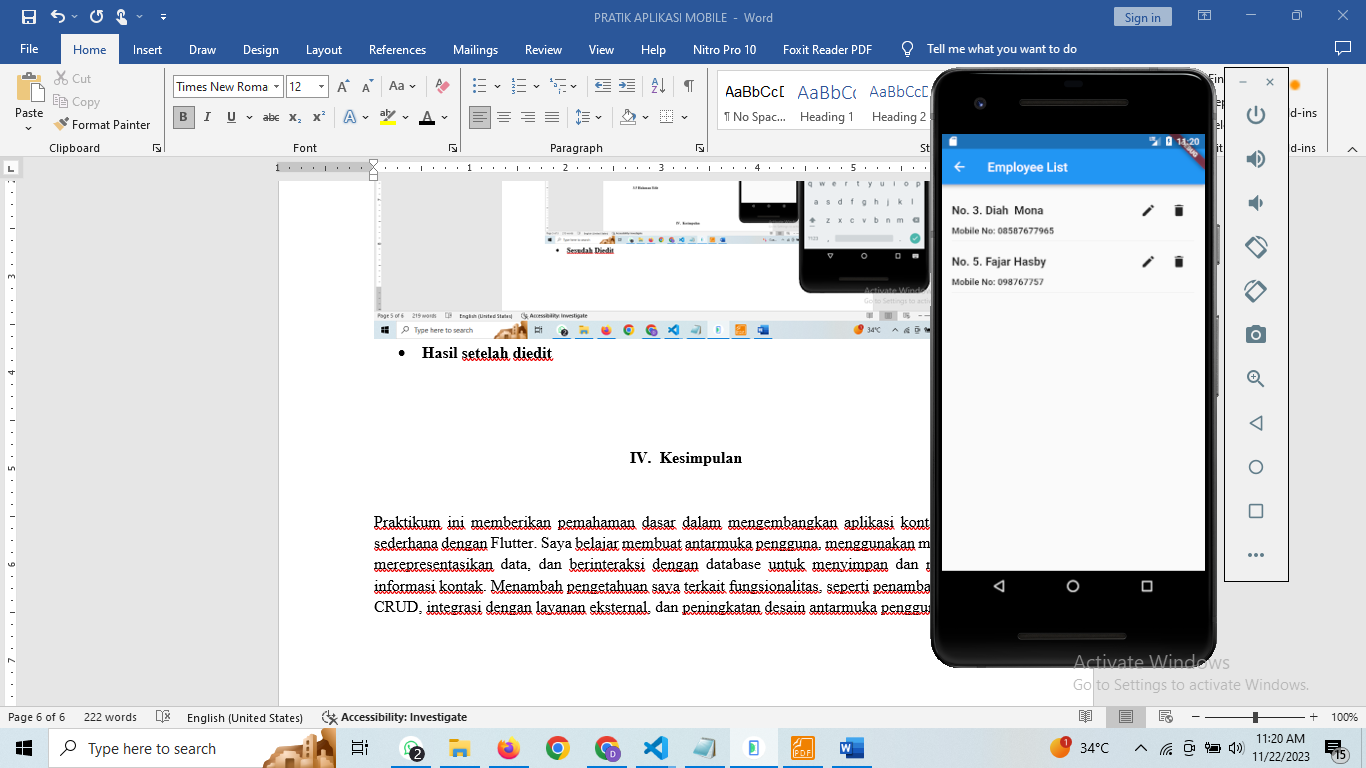
* **Sebelum Diedit**



* **Sesudah Diedit**

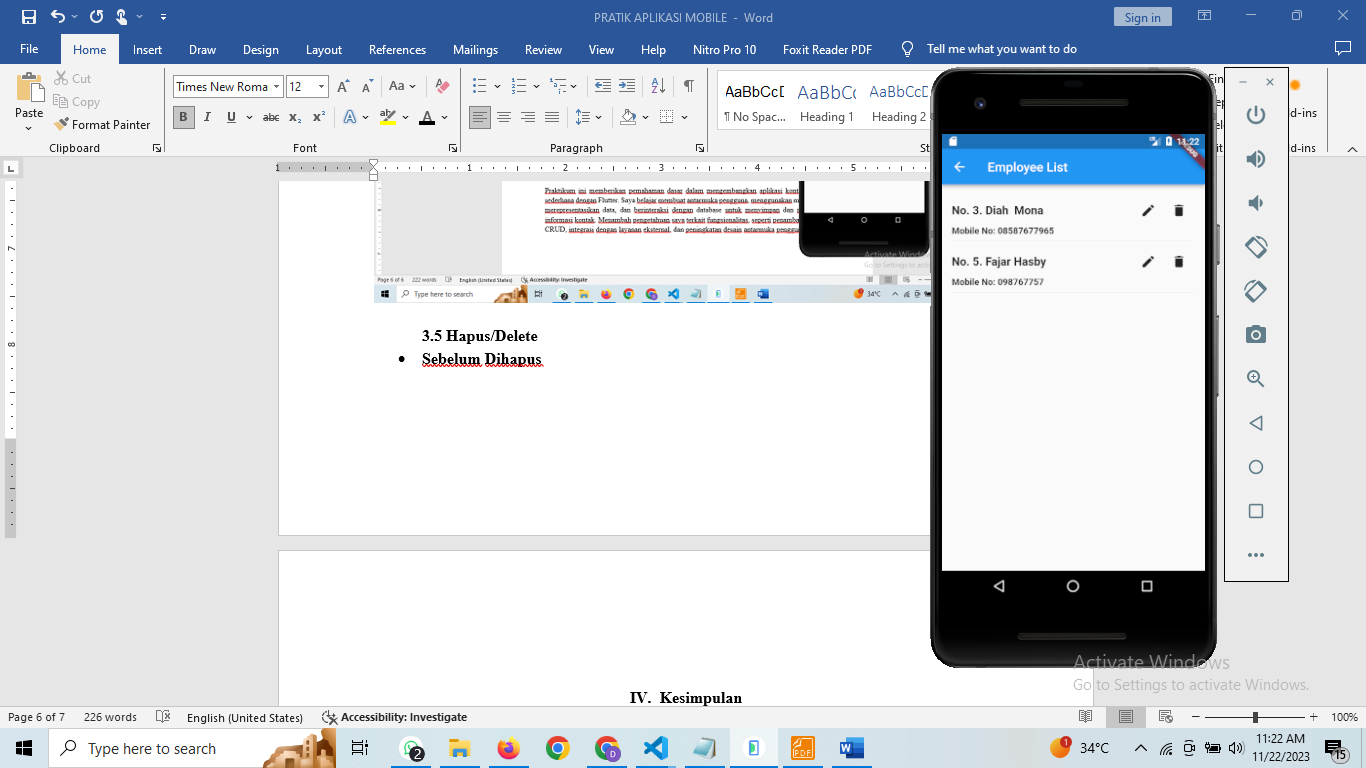


* **Hasil setelah diedit**

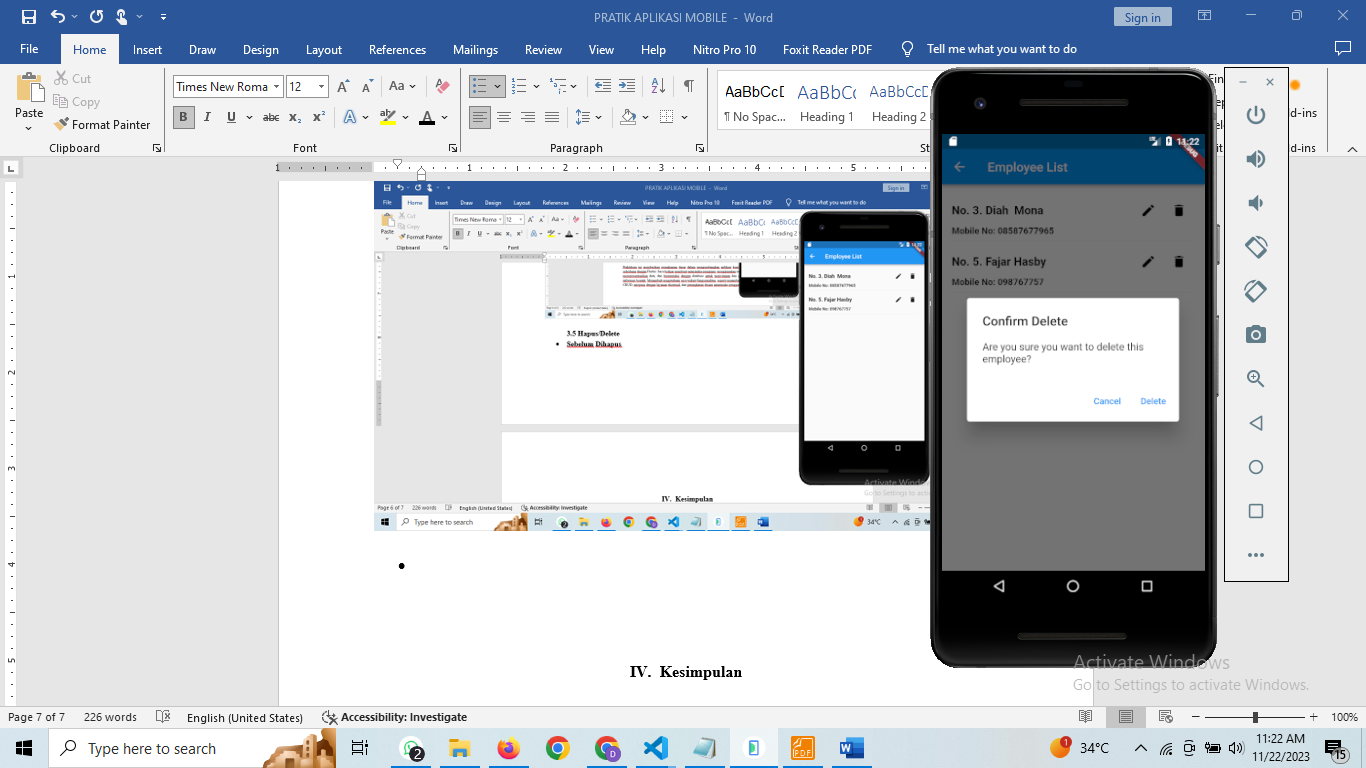


**3.5 Hapus/Delete**

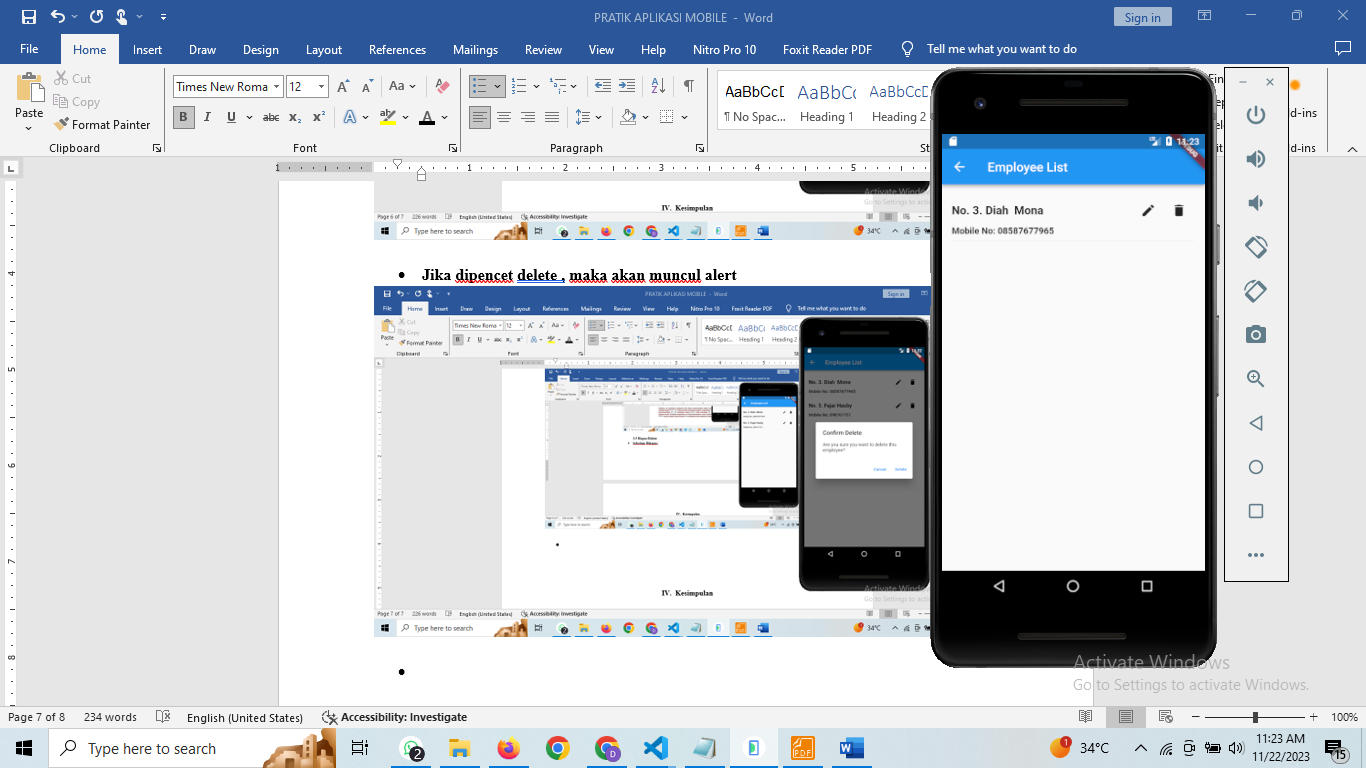
* **Sebelum Dihapus**



* **Jika dipencet delete , maka akan muncul alert**



* **Sesudah di delete**



**IV. Kesimpulan**

Praktikum ini memberikan pemahaman dasar dalam mengembangkan aplikasi kontak telepon sederhana dengan Flutter. Saya belajar membuat antarmuka pengguna, menggunakan model untuk merepresentasikan data, dan berinteraksi dengan database untuk menyimpan dan mengambil informasi kontak. Menambah pengetahuan saya terkait fungsionalitas, seperti penambahan fungsi CRUD, integrasi dengan layanan eksternal, dan peningkatan desain antarmuka pengguna.