LAPORAN TUGAS BESAR PEMROGRAMAN IV

APLIKASI FLUTTER "PSIKOFARMAKA"

Disusun guna memenuhi tugas besar mata kuliah Pemograman IV

Dosen Pengampu :

Indra Riksa Herlambang,. S.Tr.Kom., M.Kom., SFPC

Universitas Logistik & Bisnis Internasional

Disusun Oleh:

Dirga Febrian	NPM. 1.21.4.039
Juwita Stefany Hutapea	NPM. 1.21.4.026

UNIVERSITAS LOGISTIK BISNIS INTERNASIONAL

PROGRAM DIPLOMA IV TEKNIK INFORMATIKA

BANDUNG

2024

PEMBAHASAN

- **Source code dan penjelasan**

1) MODEL

SOURCE CODE login_model.dart

```
//DIGUNAKAN UNTUK FORM LOGIN
class LoginInput {
  final String username;
  final String password;

  LoginInput({
    required this.username,
    required this.password,
  });

  Map<String, dynamic> toJson() => {
        "username": username,
        "password": password,
      };
}

//DIGUNAKAN UNTUK RESPONSE
class LoginResponse {
  final int status;
  final String message;

  LoginResponse({
    required this.status,
    required this.message,
  });
  factory LoginResponse.fromJson(Map<String, dynamic> json) => LoginResponse(
        status: json["status"],
        message: json["message"],
      );
}
```

Penjelasan :	1. LoginInput Class: memiliki 2 properti yaitu username dan password, keduanya memiliki tipe string. 2. Kata kunci required pada constructor menunjukkan bahwa property harus diisi saat membuat objek pada LoginInput. 3. Map<String,dynamic>: metode ini digunakan untuk mengonversi objek menjadi forman JSON. 4. LoginResponse Class: merepresentasikan respons dari operasi login, memiliki dua property yaitu status dan message dengan tipe data int dan string. 5. Metode factory fromJson: digunakan saat menguraikan respons dari server.

SOURCE CODE obat_model.dart

```
//DIGUNAKAN UNTUK GET ALL DATA
class ObatModel {
  final String id;
  final String jenisObat;
  final String namaObat;
  final String deskripsi;

  ObatModel({
    required this.id,
    required this.jenisObat,
    required this.namaObat,
    required this.deskripsi,
  });

  factory ObatModel.fromJson(Map<String, dynamic> json) => ObatModel(
        id: json["_id"],
        jenisObat: json["jenis_obat"],
        namaObat: json["nama_obat"],
        deskripsi: json["deskripsi"],
      );

  Map<String, dynamic> toJson() => {
        "_id": id,
        "jenis_obat": jenisObat,
        "nama_obat": namaObat,
        "deskripsi": deskripsi,
      };
}

//DIGUNAKAN UNTUK FORM INPUT
class ObatInput {
  final String jenisObat;
  final String namaObat;
  final String deskripsi;

  ObatInput({
    required this.jenisObat,
    required this.namaObat,
    required this.deskripsi,
  });

  Map<String, dynamic> toJson() => {
        "jenis_obat": jenisObat,
        "nama_obat": namaObat,
        "deskripsi": deskripsi,
      };
}
```

```dart
//DIGUNAKAN UNTUK RESPONSE
class ObatResponse {
  final String? insertedId;
  final String message;
  final int status;

  ObatResponse({
    this.insertedId,
    required this.message,
    required this.status,
  });

  factory ObatResponse.fromJson(Map<String, dynamic> json) => ObatResponse(
        insertedId: json["inserted_id"],
        message: json["message"],
        status: json["status"],
      );
}
```

Penjelasan :	1. Class Obatmodel: memiliki 4 properti yaitu id, jenisObat, namaObat, dan deskripsi. 2. Metode factory fromJson: digunakan untuk membuat objek obatmodel dari data JSON dan digunakan saat mengonversi respons dari server ke dalam objek dart. 3. Class Obatinput: merepresentasikan input yang dibutuhkan untuk menambahkan data pada form input yaitu jenisoBat, namaObat, dan deskripsi 4. class Obatresponse: merepresentasikan respons dari operasi terkait obat. Memiliki 3 properti yaitu insertedID (ID yang dimasukkan setelah operasi tambah), message (pesan dari server), dan status (status operasi). 5. Menggunakan constructor untuk mendapatkan nilai pada message dan status, sementara insertedID dapat bernilai null.

SOURCE CODE penyakit_model.dart

```dart
import 'package:dio/dio.dart';

//DIGUNAKAN UNTUK GET ALL DATA
class PenyakitModel {
  final String id;
  final String jenisPenyakit;
  final String namaPenyakit;
  final String deskripsi;
  final String namaObat;

  PenyakitModel({
    required this.id,
    required this.jenisPenyakit,
    required this.namaPenyakit,
```

```
    required this.deskripsi,
    required this.namaObat,
  });

  factory PenyakitModel.fromJson(Map<String, dynamic> json) => PenyakitModel(
        id: json["_id"],
        jenisPenyakit: json["jenis_penyakit"],
        namaPenyakit: json["nama_penyakit"],
        deskripsi: json["deskripsi"],
        namaObat: json["obat"]["nama_obat"],
      );

  Map<String, dynamic> toJson() => {
        "_id": id,
        "jenis_penyakit": jenisPenyakit,
        "nama_penyakit": namaPenyakit,
        "deskripsi": deskripsi,
        "nama_obat": namaObat,
      };
}

//DIGUNAKAN UNTUK FORM INPUT
class PenyakitInput {
  final String jenisPenyakit;
  final String namaPenyakit;
  final String deskripsi;
  final String namaObat;

  PenyakitInput({
    required this.jenisPenyakit,
    required this.namaPenyakit,
    required this.deskripsi,
    required this.namaObat,
  });

  FormData formData() => FormData.fromMap({
        "jenis_penyakit": jenisPenyakit,
        "nama_penyakit": namaPenyakit,
        "deskripsi": deskripsi,
        "nama_obat": namaObat,
      });
}

//DIGUNAKAN UNTUK RESPONSE
class PenyakitResponse {
  final String? insertedId;
  final String message;
  final int status;
```

```
  PenyakitResponse({
    this.insertedId,
    required this.message,
    required this.status,
  });

  factory PenyakitResponse.fromJson(Map<String, dynamic> json) =>
      PenyakitResponse(
        insertedId: json["inserted_id"],
        message: json["message"],
        status: json["status"],
      );
}
```

Penjelasan :	1. Class Penyakitmodel: memliki 5 properti yaitu id (identifier unik), jenisPenyakit, namaPenyakit, deskripsi, dan objek dari Obatmodel dengan nama obat. 2. Metode fromJson digunakan untuk mengubah data JSON yang didapatkan dari server menjadi objek PenyakitModel. Sedangkan metode toJson digunakan untuk mengubah objek PenyakitModel menjadi data JSON yang dapat dikirim ke serverClass Penyakitinput: merepresentasikan input form yang dibutuhkan untuk menambahkan data penyakit. 3. Class Penyakitresponse: merepresentasikan respons dari operasi terkait obat. Memiliki 3 properti yaitu insertedID (ID yang dimasukkan setelah operasi tambah), message (pesan dari server), dan status (status operasi). 4. Menggunakan constructor untuk mendapatkan nilai pada message dan status, sementara insertedID dapat bernilai null.

SOURCE CODE rs_model.dart

```
//DIGUNAKAN UNTUK GET ALL DATA
class RSModel {
  final String id;
  final String namaRS;
  final String noTelp;
  final String alamat;
  final String latitude;
  final String longitude;
  RSModel({
    required this.id,
    required this.namaRS,
    required this.noTelp,
    required this.alamat,
    required this.latitude,
    required this.longitude,
  });

  factory RSModel.fromJson(Map<String, dynamic> json) => RSModel(
        id: json["_id"],
```

```
        namaRS: json["nama_rs"],
        noTelp: json["no_telp"],
        alamat: json["alamat"],
        latitude: json["latitude"],
        longitude: json["longitude"],
      );

  Map<String, dynamic> toJson() => {
        "_id": id,
        "nama_rs": namaRS,
        "no_telp": noTelp,
        "alamat": alamat,
        "latitude": latitude,
        "longitude": longitude,
      };
}

//DIGUNAKAN UNTUK FORM INPUT
class RSInput {
  final String namaRS;
  final String noTelp;
  final String alamat;
  final String latitude;
  final String longitude;

  RSInput({
    required this.namaRS,
    required this.noTelp,
    required this.alamat,
    required this.latitude,
    required this.longitude,
  });

  Map<String, dynamic> toJson() => {
        "nama_rs": namaRS,
        "no_telp": noTelp,
        "alamat": alamat,
        "latitude": latitude,
        "longitude": longitude,
      };
}

//DIGUNAKAN UNTUK RESPONSE
class RSResponse {
  final String? insertedId;
  final String message;
  final int status;
```

```
  RSResponse({
    this.insertedId,
    required this.message,
    required this.status,
  });

  factory RSResponse.fromJson(Map<String, dynamic> json) => RSResponse(
        insertedId: json["inserted_id"],
        message: json["message"],
        status: json["status"],
      );
}
```

Penjelasan :	1. Class RSmodel: memiliki 6 properti yaitu id, namaRS, noTelp, Alamat, latitude, longitude. 2. Menggunakan constructor agar keenam property memiliki nilai saat objek dibuat. 3. Class RSinput: merepresentasikan input form yang dibutuhkan untuk menambahkan data rumah sakit. 4. Menggunakan Map<String,dynamic> untuk mengonversi objek ke format JSON. 5. Class Rsresponse: memiliki 3 properti yaitu insertedID (ID yang dimasukkan setelah operasi tambah), message (pesan dari server), dan status (status operasi). 6. Menggunakan constructor untuk mendapatkan nilai pada message dan status, sementara insertedID dapat bernilai null.

2) SERVICE

SOURCE CODE api_login.dart

```
import 'package:flutter/material.dart';
import 'package:dio/dio.dart';
import 'package:flutter_obat/model/login_model.dart';
import 'dart:convert';

class ApiLogin {
  final Dio dio = Dio();
  final String _baseUrl =
      'https://asia-southeast2-obat-409909.cloudfunctions.net';

  //login
  Future<LoginResponse?> login(LoginInput login) async {
    try {
      final response = await dio.post(
        '$_baseUrl/user',
        data: login.toJson(),
      );

      if (response.statusCode == 200) {
        debugPrint('Response Data: ${response.data}');
```

```
        return LoginResponse.fromJson(json.decode(response.data));
      }
      return null;
    } on DioException catch (e) {
      if (e.response != null && e.response!.statusCode != 200) {
        debugPrint('Client error - the request cannot be fulfilled');
        return LoginResponse.fromJson(e.response!.data);
      }
      rethrow;
    } catch (e) {
      rethrow;
    }
  }
}
```

Penjelasan :	1. Dio/dio.dart: klien HTTP untuk dart yang biasa digunakan untuk membuat permintaan HTTP. 2. Login_model.dart: mengimpor model khusus (logininput dan loginresponse) untuk mewakili struktur data yang terkait dengan proses login. 3. Dart:convert: Pustaka bawaan dari dart untuk menyandikan dan mendekode data JSON. 4. Class API login: bertanggung jawab untuk menangani permintaan API terkait login. Memiliki instance kelas dio dan baseUrl yang menyimpan url dasar untuk API. 5. Metode login: mengambil objek dari logininput yang bernama login sebagai parameter. 6. $_baseUrl.user: url target untuk permintaan HTTP POST. 7. Await: digunakan untuk menunggu penyelesaian permintaan Dio POST yang bersifat asinkron. Hasilnya disimpan dalam variabel response. 8. debugPrint('Data Respons: ${response.data}'): mencetak data respons ke konsol debug. Berguna untuk debugging agar dapat melihat respons dari server. 9. return LoginResponse.fromJson(json.decode(response.data)): jika kode status respons adalah 200, metode ini akan mencetak data respons mengembalikan objek loginresponse. 10. Jika DioException tertangkap, maka akan memeriksa apakah ada respons dan apakah kode statusnya bukan 200. Jika ya, maka akan mencetak pesan kesalahan dan mengembalikan objek LoginResponse yang dibuat dari data respons kesalahan.

SOURCE CODE api_obat.dart

```
import 'package:flutter/material.dart';
import 'package:dio/dio.dart';
import 'package:flutter_obat/model/obat_model.dart';
import 'dart:convert';

class ApiObat {
  final Dio dio = Dio();
```

```dart
final String _baseUrl =
    'https://asia-southeast2-obat-409909.cloudfunctions.net';

Future<Iterable<ObatModel>?> getAllObat() async {
  try {
    var response = await dio.get('$_baseUrl/obat');
    if (response.statusCode == 200) {
      debugPrint(response.data.toString());

      var datas = json.decode(response.data);

      final obatList = (datas['data'] as List)
          .map((obat) => ObatModel.fromJson(obat))
          .toList();
      return obatList;
    }
    return null;
  } on DioException catch (e) {
    if (e.response != null && e.response!.statusCode != 200) {
      debugPrint('Client error - the request cannot be fulfilled');
      return null;
    }
    rethrow;
  } catch (e) {
    rethrow;
  }
}

Future<ObatModel?> getSingleObat(String id) async {
  try {
    var response = await dio.get('$_baseUrl/obat?_id=$id');
    if (response.statusCode == 200) {
      debugPrint('respon: ${response.data}');
      final data = json.decode(response.data);
      return ObatModel.fromJson(data['data']);
    }
    return null;
  } on DioException catch (e) {
    if (e.response != null && e.response!.statusCode != 200) {
      debugPrint('Client error - the request cannot be fulfilled');
      return null;
    }
    rethrow;
  } catch (e) {
    rethrow;
  }
}
```

```
Future<ObatResponse?> postObat(ObatInput obat) async {
  try {
    final response = await dio.post(
      '$_baseUrl/obat',
      data: obat.formData(),
    );
    if (response.statusCode == 200) {
      debugPrint(response.data.toString());
      return ObatResponse.fromJson(json.decode(response.data));
    }
    return null;
  } catch (e) {
    rethrow;
  }
}

Future<ObatResponse?> putObat(String id, ObatInput obat) async {
  try {
    final response = await Dio().put(
      '$_baseUrl/obat?_id=$id',
      data: obat.formData(),
    );
    if (response.statusCode == 200) {
      return ObatResponse.fromJson(json.decode(response.data));
    }
    return null;
  } catch (e) {
    rethrow;
  }
}

Future deleteObat(String id) async {
  try {
    final response = await Dio().delete('$_baseUrl/obat?_id=$id');
    if (response.statusCode == 200) {
      return ObatResponse.fromJson(json.decode(response.data));
    }
    return null;
  } catch (e) {
    rethrow;
  }
}

Future<List<String>?> getAllObatNames() async {
  try {
    var response = await dio.get('$_baseUrl/obat');
    if (response.statusCode == 200) {
      debugPrint(response.data.toString());
```

```
        var datas = json.decode(response.data);
        if (datas['data'] is List) {
          final obatList = (datas['data'] as List)
              .map((obat) => ObatModel.fromJson(obat).namaObat)
              .toList();
          return obatList;
        } else {
          debugPrint('Unexpected data format in the response');
          return null;
        }
      }
      return null;
    } on DioException catch (e) {
      if (e.response != null && e.response!.statusCode != 200) {
        debugPrint('Client error - the request cannot be fulfilled');
        return null;
      }
      rethrow;
    } catch (e) {
      rethrow;
    }
  }
}
```

Penjelasan :	1. final Dio dio = Dio(); dan final String _baseUrl = 'https://asia-southeast2-obat-409909.cloudfunctions.net'; Variabel ini digunakan untuk mengatur instance Dio dan URL dasar dari API. 2. Future<Iterable<ObatModel>?> getAllObat(): metode ini digunakan untuk mendapatkan semua data dari obat server. Jika respon dari server memiliki status kode 200, maka data tersebut akan didekode dari format JSON ke dalam model Obatmodel. 3. Future<ObatModel?> getSingleObat(String id): metode ini digunakan untuk mendapatkan satu data obat berdasarkan ID yang diberikan. 4. Future<ObatResponse?> postObat(ObatInput obat): metode ini digunakan untuk membuat data baru obat di server. Data obat yang akan dibuat dikirimkan dalam bentuk JSON. 5. Future<ObatResponse?> putObat(String id, ObatInput obat): metode ini digunakan untuk memperbarui data obat yang sudah ada di server. 6. Future deleteObat(String id): metode ini digunakan untuk menghapus data obat dari server berdasarkan ID yang diberikan. 7. Future<List<String>?> getAllObatNames(): metode ini digunakan untuk mendapatkan semua nama obat dari server.

SOURCE CODE api_penyakit.dart

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

```dart
import 'package:flutter_obat/model/penyakit_model.dart';
import 'dart:convert';

class ApiPenyakit {
  final Dio dio = Dio();
  final String _baseUrl =
      'https://asia-southeast2-obat-409909.cloudfunctions.net';

  Future<Iterable<PenyakitModel>?> getAllPenyakit() async {
    try {
      var response = await dio.get('$_baseUrl/penyakit');
      if (response.statusCode == 200) {
        debugPrint(response.data.toString());

        var datas = json.decode(response.data);

        final penyakitList = (datas['data'] as List)
            .map((penyakit) => PenyakitModel.fromJson(penyakit))
            .toList();
        return penyakitList;
      }
      return null;
    } on DioException catch (e) {
      if (e.response != null && e.response!.statusCode != 200) {
        debugPrint('Client error - the request cannot be fulfilled');
        return null;
      }
      rethrow;
      // } catch (e) {
      //   rethrow;
    }
  }

  Future<PenyakitModel?> getSinglePenyakit(String id) async {
    try {
      var response = await dio.get('$_baseUrl/penyakit?_id=$id');
      if (response.statusCode == 200) {
        debugPrint('respon: ${response.data}');
        final data = json.decode(response.data);
        return PenyakitModel.fromJson(data['data']);
      }
      return null;
    } on DioException catch (e) {
      if (e.response != null && e.response!.statusCode != 200) {
        debugPrint('Client error - the request cannot be fulfilled');
        return null;
      }
      rethrow;
```

```
  } catch (e) {
    rethrow;
  }
}

Future<PenyakitResponse?> postPenyakit(PenyakitInput penyakit) async {
  try {
    final response = await dio.post(
      '$_baseUrl/penyakit',
      data: penyakit.formData(),
      // data: penyakit.toJson(),
    );
    if (response.statusCode == 200) {
      return PenyakitResponse.fromJson(json.decode(response.data));
    }
    return null;
  } catch (e) {
    rethrow;
  }
}

Future<PenyakitResponse?> putPenyakit(
    String id, PenyakitInput penyakit) async {
  try {
    final response = await Dio().put(
      '$_baseUrl/penyakit?_id=$id',
      data: penyakit.formData(),
    );
    if (response.statusCode == 200) {
      return PenyakitResponse.fromJson(json.decode(response.data));
    }
    return null;
  } catch (e) {
    rethrow;
  }
}

Future deletePenyakit(String id) async {
  try {
    final response = await Dio().delete('$_baseUrl/penyakit?_id=$id');
    if (response.statusCode == 200) {
      return PenyakitResponse.fromJson(json.decode(response.data));
    }
    return null;
  } catch (e) {
    rethrow;
  }
}
```

Penjelasan :	1. final Dio dio = Dio(); dan final String _baseUrl = 'https://asia-southeast2-obat-409909.cloudfunctions.net': membuat instance dari dio dan menentukan url. 2. Future<Iterable<PenyakitModel>?> getAllPenyakit() async {}: Fungsi ini digunakan untuk mendapatkan semua data penyakit dari API. Fungsi ini mengirimkan permintaan GET ke endpoint '/penyakit' dan mengembalikan list dari objek PenyakitModel. 3. Future<PenyakitModel?> getSinglePenyakit(String id) async {}: Fungsi ini digunakan untuk mendapatkan satu data penyakit dari API berdasarkan ID. Fungsi ini mengirimkan permintaan GET ke endpoint '/penyakit' dengan parameter '_id' . 4. Future<PenyakitResponse?> postPenyakit(PenyakitInput penyakit) async {}: Fungsi ini digunakan untuk mengirim data penyakit baru ke API. Fungsi ini mengirimkan permintaan POST ke endpoint '/penyakit' dengan data penyakit sebagai payload dan mengembalikan objek PenyakitResponse. 5. Future<PenyakitResponse?> putPenyakit(String id, PenyakitInput penyakit) async {}: Fungsi ini digunakan untuk memperbarui data penyakit di API. Fungsi ini mengirimkan permintaan PUT ke endpoint '/penyakit' dengan ID dan data penyakit sebagai payload 6. Future deletePenyakit(String id) async {}: Fungsi ini digunakan untuk menghapus data penyakit di API. Fungsi ini mengirimkan permintaan DELETE ke endpoint '/penyakit' dengan ID sebagai parameter.

SOURCE CODE api_rs.dart

```
import 'package:flutter/material.dart';
import 'package:dio/dio.dart';
import 'package:flutter_obat/model/rs_model.dart';
import 'dart:convert';

class ApiRS {
  final Dio dio = Dio();
  final String _baseUrl =
      'https://asia-southeast2-obat-409909.cloudfunctions.net';

  Future<Iterable<RSModel>?> getAllRS() async {
    try {
      var response = await dio.get('$_baseUrl/rs');
      if (response.statusCode == 200) {
        debugPrint(response.data.toString());

        var datas = json.decode(response.data);

        final rsList =
            (datas['data'] as List).map((rs) =>
RSModel.fromJson(rs)).toList();
        return rsList;
      }
```

```
    return null;
  } on DioException catch (e) {
    if (e.response != null && e.response!.statusCode != 200) {
      debugPrint('Client error - the request cannot be fulfilled');
      return null;
    }
    rethrow;
  } catch (e) {
    rethrow;
  }
}

Future<RSModel?> getSingleRS(String id) async {
  try {
    var response = await dio.get('$_baseUrl/rs?_id=$id');
    if (response.statusCode == 200) {
      debugPrint('respon: ${response.data}');
      final data = json.decode(response.data);
      return RSModel.fromJson(data['data']);
    }
    return null;
  } on DioException catch (e) {
    if (e.response != null && e.response!.statusCode != 200) {
      debugPrint('Client error - the request cannot be fulfilled');
      return null;
    }
    rethrow;
  } catch (e) {
    rethrow;
  }
}

Future<RSResponse?> postRS(RSInput rs) async {
  try {
    final response = await dio.post(
      '$_baseUrl/rs',
      data: rs.formData(),
    );
    if (response.statusCode == 200) {
      return RSResponse.fromJson(json.decode(response.data));
    }
    return null;
  } catch (e) {
    rethrow;
  }
}

Future<RSResponse?> putRS(String id, RSInput rs) async {
```

```
    try {
      final response = await Dio().put(
        '$_baseUrl/rs?_id=$id',
        data: rs.formData(),
      );
      if (response.statusCode == 200) {
        return RSResponse.fromJson(json.decode(response.data));
      }
      return null;
    } catch (e) {
      rethrow;
    }
  }

  Future deleteRS(String id) async {
    try {
      final response = await Dio().delete('$_baseUrl/rs?_id=$id');
      if (response.statusCode == 200) {
        return RSResponse.fromJson(json.decode(response.data));
      }
      return null;
    } catch (e) {
      rethrow;
    }
  }
}
```

Penjelasan :	1. final Dio dio = Dio(); dan final String _baseUrl = 'https://asia-southeast2-obat-409909.cloudfunctions.net': membuat instance dari dio dan menentukan url. 2. Future<Iterable<RSModel>?> getAllRS() async {}: Fungsi ini digunakan untuk mendapatkan semua data rumah sakit dari API. Fungsi ini mengirimkan permintaan GET ke endpoint '/rs' dan mengembalikan list dari objek RSModel 3. Future<RSModel?> getSingleRS(String id) async {}: Fungsi ini digunakan untuk mendapatkan satu data rumah sakit dari API berdasarkan ID. Fungsi ini mengirimkan permintaan GET ke endpoint '/rs' dengan parameter '_id'. 4. Future<RSResponse?> postRS(RSInput rs) async {}: Fungsi ini digunakan untuk mengirim data rumah sakit baru ke API. Fungsi ini mengirimkan permintaan POST ke endpoint '/rs' dengan data rumah sakit sebagai payload dan mengembalikan objek RSResponse. 5. Future<RSResponse?> putRS(String id, RSInput rs) async {}: Fungsi ini digunakan untuk memperbarui data rumah sakit di API. Fungsi ini mengirimkan permintaan PUT ke endpoint '/rs' dengan ID dan data rumah sakit sebagai payload dan mengembalikan objek RSResponse. 6. Future deleteRS(String id) async {}: Fungsi ini digunakan untuk menghapus data rumah sakit di API. Fungsi ini mengirimkan permintaan DELETE ke endpoint '/rs' dengan ID sebagai parameter.

SOURCE CODE auth_manager.dart

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

class AuthManager {
  static const String loginStatusKey = '';
  static const String loginTimeKey = '';

  static Future<bool> isLoggedIn() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    bool isLoggedIn = prefs.getBool('loginStatusKey') ?? false;
    String? loginTimeString = prefs.getString('loginTimeKey');
    if (isLoggedIn && loginTimeString != null) {
      try {
        DateTime loginTime = DateTime.parse(loginTimeString);
        final Duration timeDifference = DateTime.now().difference(loginTime);
// Set maximum durasi untuk validasi login di bawah ini
        const Duration maxDuration = Duration(hours: 4);
        if (timeDifference > maxDuration) {
          await logout();
          return false;
        }
        return true;
      } catch (e) {
        debugPrint('Error parsing DateTime: $e');
        await logout();
        return false;
      }
    }
    return false;
  }

  static Future<bool> isAdmin() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    String? username = prefs.getString('username');
    return username == 'admin';
  }

  static Future<void> login(String username) async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    prefs.setBool('loginStatusKey', true);
    prefs.setString('loginTimeKey', DateTime.now().toString());
    prefs.setString('username', username);
  }

  static Future<void> getUser() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    prefs.getString('username');
```

```
    prefs.getString('phone_number');
  }

  static Future<void> logout() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    prefs.remove('loginStatusKey');
    prefs.remove('loginTimeKey');
    prefs.remove('username');
  }
}
```

Penjelasan :	1. Class auth manager digunakan untuk mengelola autentikasi pengguna di aplikasi Flutter. 2. static const String loginStatusKey = ''; dan static const String loginTimeKey = '';: Variabel ini digunakan untuk menyimpan key dari status login dan waktu login. 3. static Future<bool> isLoggedIn() async {}: Fungsi ini digunakan untuk memeriksa apakah pengguna sudah login. Fungsi ini mendapatkan status login dan waktu login dari shared preferences dan memeriksa apakah pengguna sudah login. 4. static Future<bool> isAdmin() async {}: Fungsi ini digunakan untuk memeriksa apakah pengguna adalah admin. Fungsi ini mendapatkan username dari shared preferences dan memeriksa apakah username tersebut adalah 'admin'. 5. static Future<void> login(String username) async {}: Fungsi ini digunakan untuk melakukan login. Fungsi ini menyimpan status login, waktu login, dan username ke shared preferences. 6. static Future<void> getUser() async {}: Fungsi ini digunakan untuk mendapatkan user. Fungsi ini mendapatkan username dan nomor telepon dari shared preferences. 7. static Future<void> logout() async {}: Fungsi ini digunakan untuk melakukan logout. Fungsi ini menghapus status login, waktu login, dan username dari shared preferences.

3) View

➢ Screen/Obat

SOURCE CODE obat_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_obat/model/obat_model.dart';
import 'package:flutter_obat/service/api_obat.dart';
import 'package:flutter_obat/view/widget/obat_card.dart';
import 'package:file_picker/file_picker.dart';
import 'package:cached_network_image/cached_network_image.dart';

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

  @override
```

```
  State<ObatScreen> createState() => _ObatScreenState();
}

class _ObatScreenState extends State<ObatScreen> {
  final _formKey = GlobalKey<FormState>();
  final _namaObat = TextEditingController();
  final _jenisObat = TextEditingController();
  final _deskripsi = TextEditingController();
  String _result = '-';

  PlatformFile? file;
  String? _namaFile;
  String? _pathFile;

  final ApiObat _dataService = ApiObat();
  List<ObatModel> _obatModel = [];

  ObatResponse? obatRes;

  bool isEdit = false;
  String idObat = '';

  @override
  void dispose() {
    _namaObat.dispose();
    _jenisObat.dispose();
    _deskripsi.dispose();
    super.dispose();
  }

  bool _validateFile(String? pathFile) {
    if (pathFile == null) {
      return true;
    }
    return false;
  }

  void _pickFile() async {
    final result = await FilePicker.platform.pickFiles();
    if (result == null) return;

    file = result.files.single;

    setState(() {
      _namaFile = file!.name;
      _pathFile = file!.path;
    });
  }
```

```dart
Future<void> refreshObatList() async {
  final users = await _dataService.getAllObat();
  setState(() {
    if (_obatModel.isNotEmpty) _obatModel.clear();
    if (users != null) _obatModel.addAll(users);
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text(
        "Psikofarmaka",
        style: TextStyle(
          fontWeight: FontWeight.bold,
          color: Colors.white,
        ),
      ),
      iconTheme: const IconThemeData(color: Colors.white),
      // centerTitle: true,
      backgroundColor: Colors.blue.shade600,
    ),
    body: Container(
      width: double.infinity,
      padding: const EdgeInsets.all(20),
      child: Form(
        key: _formKey,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            const SizedBox(height: 20.0),
            TextField(
              controller: _jenisObat,
              onChanged: (value) {
                setState(() {});
              },
              decoration: InputDecoration(
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(10),
                ),
                contentPadding: const EdgeInsets.symmetric(
                  horizontal: 16,
                  vertical: 12,
                ),
                labelText: 'Jenis Obat',
                hintText: 'Masukkan jenis obat',
```

```
                suffixIcon: _jenisObat.text.isNotEmpty
                    ? IconButton(
                        onPressed: () {
                          setState(() {
                            _jenisObat.clear();
                          });
                        },
                        icon: const Icon(Icons.clear),
                      )
                    : null,
              ),
            ),
            const SizedBox(height: 8.0),
            TextField(
              controller: _namaObat,
              onChanged: (value) {
                setState(() {});
              },
              decoration: InputDecoration(
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(10),
                ),
                contentPadding: const EdgeInsets.symmetric(
                  horizontal: 16,
                  vertical: 12,
                ),
                labelText: 'Nama Obat',
                hintText: 'Masukkan nama obat',
                suffixIcon: _namaObat.text.isNotEmpty
                    ? IconButton(
                        onPressed: () {
                          setState(() {
                            _namaObat.clear();
                          });
                        },
                        icon: const Icon(Icons.clear),
                      )
                    : null,
              ),
            ),
            const SizedBox(height: 8.0),
            TextField(
              controller: _deskripsi,
              maxLines: 3,
              onChanged: (value) {
                setState(() {});
              },
              decoration: InputDecoration(
```

```
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(10),
                ),
                contentPadding: const EdgeInsets.symmetric(
                  horizontal: 16,
                  vertical: 12,
                ),
                labelText: 'Deskripsi',
                hintText: 'Masukkan deskripsi obat',
                suffixIcon: _deskripsi.text.isNotEmpty
                    ? IconButton(
                        onPressed: () {
                          setState(() {
                            _deskripsi.clear();
                          });
                        },
                        icon: const Icon(Icons.clear),
                      )
                    : null,
              ),
            ),
            const SizedBox(height: 8.0),
            buildFilePicker(context),
            const SizedBox(height: 8.0),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Wrap(
                  alignment: WrapAlignment.center,
                  spacing: 8.0,
                  children: [
                    ElevatedButton(
                      onPressed: () async {
                        if (_namaObat.text.isEmpty ||
                            _jenisObat.text.isEmpty ||
                            _deskripsi.text.isEmpty) {
                          displaySnackbar('Semua field harus diisi');
                          return;
                        }
                        final postModel = ObatInput(
                          jenisObat: _jenisObat.text,
                          namaObat: _namaObat.text,
                          deskripsi: _deskripsi.text,
                          imagePath: _pathFile!,
                          imageName: _namaFile!,
                        );
                        ObatResponse? res;
                        if (isEdit) {
```

```
                            res = await _dataService.putObat(idObat,
postModel);
                          } else {
                            res = await _dataService.postObat(postModel);
                          }

                          setState(() {
                            obatRes = res;
                            isEdit = false;
                          });
                          _namaObat.clear();
                          _jenisObat.clear();
                          _deskripsi.clear();
                          await refreshObatList();
                        },
                        child: Text(isEdit ? 'UPDATE' : 'POST'),
                      ),
                      if (isEdit)
                        ElevatedButton(
                          onPressed: () {
                            _namaObat.clear();
                            _jenisObat.clear();
                            _deskripsi.clear();
                            setState(() {
                              isEdit = false;
                            });
                          },
                          child: const Text('Cancel Update',
                              style: TextStyle(color: Colors.red)),
                        ),
                    ],
                  )
                ],
              ),
              hasilCard(context),
              Flexible(
                child: Row(
                  children: [
                    Expanded(
                      child: ElevatedButton(
                        onPressed: () async {
                          await refreshObatList();
                          setState(() {});
                        },
                        child: const Text('Refresh Data'),
                      ),
                    ),
                    const SizedBox(width: 8.0),
```

```
                  ElevatedButton(
                    onPressed: () {
                      setState(() {
                        _result = '-';
                        _obatModel.clear();
                        obatRes = null;
                      });
                    },
                    child: const Text('Reset'),
                  ),
                ],
              ),
            ),
            const SizedBox(height: 8.0),
            const Text(
              'List Obat',
              style: TextStyle(
                fontWeight: FontWeight.bold,
                fontSize: 20.0,
              ),
            ),
            const SizedBox(height: 8.0),
            Expanded(
              child: _obatModel.isEmpty ? Text(_result) : _buildListObat(),
            ),
            const SizedBox(
              height: 20,
            ),
          ],
        ),
      ),
    ),
  );
}

dynamic displaySnackbar(String msg) {
  return ScaffoldMessenger.of(context)
      .showSnackBar(SnackBar(content: Text(msg)));
}

Widget buildFilePicker(BuildContext context) {
  return TextField(
    controller: TextEditingController(text: _namaFile),
    readOnly: true,
    onChanged: (value) {
      setState(() {});
    },
    decoration: InputDecoration(
```

```
      border: OutlineInputBorder(
        borderRadius: BorderRadius.circular(10),
      ),
      contentPadding: const EdgeInsets.symmetric(
        horizontal: 16,
        vertical: 12,
      ),
      labelText: 'Gambar Obat',
      hintText: 'Masukkan gambar obat',
      suffixIcon: _namaFile != null
          ? IconButton(
              icon: const Icon(Icons.clear),
              onPressed: () {
                setState(() {
                  _namaFile = null;
                });
              },
            )
          : IconButton(
              icon: const Icon(Icons.attach_file),
              onPressed: () async {
                _pickFile();
              },
            ),
    ),
  );
}

Widget _buildListObat() {
  return ListView.separated(
      itemBuilder: (context, index) {
        final obatList = _obatModel[index];
        return Card(
          child: ListTile(
            leading: CachedNetworkImage(
              imageUrl: obatList.gambar,
              width: 50,
              height: 50,
              fit: BoxFit.cover,
            ),
            title: Text(obatList.namaObat),
            trailing: Row(
              mainAxisSize: MainAxisSize.min,
              children: [
                IconButton(
                  onPressed: () async {
                    final obats =
                        await _dataService.getSingleObat(obatList.id);
```

```
                    setState(() {
                      if (obats != null) {
                        _namaObat.text = obats.namaObat;
                        _jenisObat.text = obats.jenisObat;
                        _deskripsi.text = obats.deskripsi;
                        isEdit = true;
                        idObat = obats.id;
                      }
                    });
                  },
                  icon: const Icon(Icons.edit),
                ),
                IconButton(
                  onPressed: () {
                    _showDeleteConfirmationDialog(
                        obatList.id, obatList.namaObat);
                  },
                  icon: const Icon(Icons.delete),
                ),
              ],
            ),
          ),
        );
      },
      separatorBuilder: (context, index) => const SizedBox(height: 10.0),
      itemCount: _obatModel.length);
}

Widget hasilCard(BuildContext context) {
  return Column(children: [
    if (obatRes != null)
      ObatCard(
        obatRes: obatRes!,
        onDismissed: () {
          setState(() {
            obatRes = null;
          });
        },
      )
    else
      const Text(''),
  ]);
}

void _showDeleteConfirmationDialog(String id, String nama) {
  showDialog(
    context: context,
    builder: (BuildContext context) {
```

```
        return AlertDialog(
          title: const Text('Konfirmasi Hapus'),
          content: Text('Apakah Anda yakin ingin menghapus data $nama ?'),
          actions: <Widget>[
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: const Text('CANCEL'),
            ),
            TextButton(
              onPressed: () async {
                ObatResponse? res = await _dataService.deleteObat(id);
                setState(() {
                  obatRes = res;
                });
                Navigator.of(context).pop();
                await refreshObatList();
              },
              child: const Text('DELETE'),
            ),
          ],
        );
      },
    );
  }
}
```

Penjelasan :	1. final _formKey = GlobalKey<FormState>(); dan final _namaObat = TextEditingController(); hingga _deskripsi = TextEditingController(); variabel ini digunakan untuk menyimpan state dari form dan teks input. 2. PlatformFile? file; dan String? _namaFile; hingga _pathFile; variabel ini digunakan untuk menyimpan informasi tentang file yang dipilih oleh pengguna. 3. final ApiObat _dataService = ApiObat(); dan List<ObatModel> _obatModel = []; variabel ini digunakan untuk berinteraksi dengan API dan menyimpan data obat. 4. bool isEdit = false; dan String idObat = ''; variabel ini digunakan untuk menentukan apakah aplikasi sedang dalam mode edit atau bukan dan untuk menyimpan ID obat yang sedang diedit. 5. void _pickFile() async {}: fungsi ini digunakan untuk memilih file dari sistem file pengguna. 6. Future<void> refreshObatList() async {}: fungsi ini digunakan untuk mengambil semua data obat dari server dan disimpan di _obatModel. 7. Widget build(BuildContext context) {}: fungsi ini digunakan untuk membangun UI dari halaman ObatScreen yang terdapat form untuk input data obat, tombol untuk memilih file, dan daftar obat yang ditampilkan. 8. dynamic displaySnackbar(String msg) {}: fungsi ini digunakan

	untuk menampilkan snackbar dengan pesan tertentu. 9. Widget buildFilePicker(BuildContext context) {}: fungsi ini digunakan untuk membangun widget untuk memilih file. 10. Widget _buildListObat() {}: fungsi ini digunakan untuk membangun daftar obat. 11. void _showDeleteConfirmationDialog(String id, String nama) {}: fungsi ini digunakan untuk menampilkan dialog konfirmasi hapus obat.

SOURCE CODE list_obat.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_obat/model/obat_model.dart';
import 'package:flutter_obat/service/api_obat.dart';
import 'package:cached_network_image/cached_network_image.dart';

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

  @override
  State<ListObat> createState() => _ListObatState();
}

class _ListObatState extends State<ListObat> {
  final ApiObat _dataService = ApiObat();
  List<ObatModel> obat = [];

  @override
  void initState() {
    super.initState();
    _dataService.getAllObat().then((value) {
      setState(() {
        obat = value?.toList() ?? [];
      });
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          "Psikofarmaka",
          style: TextStyle(
            fontWeight: FontWeight.bold,
            color: Colors.white,
          ),
        ),
        iconTheme: const IconThemeData(color: Colors.white),
```

```
        backgroundColor: Colors.blue.shade600,
      ),
      body: Container(
        width: double.infinity,
        padding: const EdgeInsets.all(20),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            const SizedBox(height: 20),
            const Center(
              child: Text(
                "List Obat",
                style: TextStyle(
                  fontSize: 20,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ),
            const SizedBox(height: 20),
            Expanded(
              child: ListView.builder(
                itemCount: obat.length,
                itemBuilder: (context, index) {
                  return Card(
                    child: ListTile(
                      leading: CachedNetworkImage(
                        imageUrl: obat[index].gambar,
                        width: 50,
                        height: 50,
                        fit: BoxFit.cover,
                      ),
                      title: Text(obat[index].namaObat),
                      trailing: ElevatedButton(
                        onPressed: () {
                          _showDetailsBottomSheet(context, obat[index]);
                        },
                        child: const Text('Details'),
                      ),
                    ),
                  );
                },
              ),
            ),
          ],
        ),
      ),
    );
  }
```

```
// Function to show details in a BottomSheet
void _showDetailsBottomSheet(BuildContext context, ObatModel obatModel) {
  showModalBottomSheet(
    context: context,
    builder: (BuildContext context) {
      return Container(
        width: double.infinity,
        padding: const EdgeInsets.all(16),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            const Center(
              child: Text(
                "Details Obat",
                style: TextStyle(
                  fontWeight: FontWeight.bold,
                  fontSize: 20,
                ),
              ),
            ),
            const SizedBox(height: 15),
            const Text(
              "Jenis Obat: ",
              style: TextStyle(
                fontWeight: FontWeight.bold,
              ),
            ),
            Container(
              padding: const EdgeInsets.all(7),
              decoration: BoxDecoration(
                color: Colors.blue.shade200,
                borderRadius: BorderRadius.circular(15),
                boxShadow: [
                  BoxShadow(
                    color: Colors.grey.withOpacity(0.3),
                    spreadRadius: 2,
                    blurRadius: 5,
                    offset: const Offset(0, 3),
                  ),
                ],
              ),
              child: Text(
                obatModel.jenisObat,
                style: const TextStyle(
                  fontWeight: FontWeight.normal,
                ),
              ),
```

```
            ),
            const SizedBox(height: 15),
            const Text(
              "Nama Obat: ",
              style: TextStyle(
                fontWeight: FontWeight.bold,
              ),
            ),
            Container(
              padding: const EdgeInsets.all(7),
              decoration: BoxDecoration(
                color: Colors.blue.shade200,
                borderRadius: BorderRadius.circular(15),
                boxShadow: [
                  BoxShadow(
                    color: Colors.grey.withOpacity(0.3),
                    spreadRadius: 2,
                    blurRadius: 5,
                    offset: const Offset(0, 3),
                  ),
                ],
              ),
              child: Text(
                obatModel.namaObat,
                style: const TextStyle(
                  fontWeight: FontWeight.normal,
                ),
              ),
            ),
            const SizedBox(height: 15),
            const Text(
              "Deskripsi Obat: ",
              style: TextStyle(
                fontWeight: FontWeight.bold,
              ),
            ),
            Container(
              padding: const EdgeInsets.all(7),
              decoration: BoxDecoration(
                color: Colors.blue.shade200,
                borderRadius: BorderRadius.circular(15),
                boxShadow: [
                  BoxShadow(
                    color: Colors.grey.withOpacity(0.3),
                    spreadRadius: 2,
                    blurRadius: 5,
                    offset: const Offset(0, 3),
                  ),
```

```
                ],
              ),
              child: Text(
                obatModel.deskripsi,
                style: const TextStyle(
                  fontWeight: FontWeight.normal,
                ),
                textAlign: TextAlign.justify,
              ),
            ),
            const SizedBox(height: 20),
            ElevatedButton(
              onPressed: () {
                Navigator.pop(context); // Close the BottomSheet
              },
              child: const Text('Close'),
            ),
          ],
        ),
      );
    },
  );
 }
}
```

Penjelasan :	1. final ApiObat _dataService = ApiObat(); dan List<ObatModel> obat = [] variabel ini digunakan untuk berinteraksi dengan API dan menyimpan data obat. 2. void initState() {}: fungsi ini dipanggil saat widget pertama kali dimasukkan ke dalam tree widget. Semua data obat diambil dari server dan disimpan di variabel obat. 3. Widget build(BuildContext context) {}: fungsi ini digunakan untuk membangun UI dari halaman ListObat yang didalamnya terdapat judul dan daftar obat yang ditampilkan. 4. void _showDetailsBottomSheet(BuildContext context, ObatModel obatModel) {}: fungsi ini digunakan untuk menampilkan detail obat dalam modal bottom sheet.

➢ Screen/User

SOURCE CODE dashboard_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_obat/service/auth_manager.dart';
import 'package:flutter_obat/view/screen/login_screen.dart';
import 'package:flutter_obat/view/screen/user/data_obat.dart';
import 'package:flutter_obat/view/screen/user/data_penyakit.dart';
import 'package:flutter_obat/view/screen/user/data_rs.dart';
import 'package:flutter_obat/view/widget/dashboard.dart';
import 'package:shared_preferences/shared_preferences.dart';
```

```dart
class DashboardScreen extends StatefulWidget {
  const DashboardScreen({Key? key}) : super(key: key);

  @override
  State<DashboardScreen> createState() => _DashboardScreenState();
}

class _DashboardScreenState extends State<DashboardScreen> {
  List<Map<String, dynamic>> categories = [
    {
      "image": "assets/images/medicine.png",
      "text": "Obat",
      "screen": const DataObat(),
    },
    {
      "image": "assets/images/disease.png",
      "text": "Penyakit",
      "screen": const DataPenyakit(),
    },
    {
      "image": "assets/images/hospital.png",
      "text": "Rumah Sakit",
      "screen": const DataRS(),
    }
  ];

  late SharedPreferences loginData;
  String username = '';

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

  void inital() async {
    loginData = await SharedPreferences.getInstance();
    setState(() {
      username = loginData.getString('username').toString();
    });
  }

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () {
        FocusScope.of(context).unfocus();
```

```
    },
    child: Scaffold(
      appBar: AppBar(
        title: Text(
          "Psikofarmaka",
          style: TextStyle(
            fontWeight: FontWeight.bold,
            color: Colors.white,
            fontFamily: 'YourCustomFont', // Replace with your custom font
          ),
        ),
        iconTheme: const IconThemeData(color: Colors.white),
        backgroundColor: Colors.orange.shade800,
        flexibleSpace: Container(
          decoration: BoxDecoration(
            gradient: LinearGradient(
              colors: [Colors.orange.shade800, Colors.orange.shade600],
              begin: Alignment.topLeft,
              end: Alignment.bottomRight,
            ),
          ),
        ),
        actions: [
          IconButton(
            onPressed: () {
              _showLogoutConfirmationDialog(context);
            },
            icon: const Icon(Icons.logout),
            color: Colors.white,
          ),
        ],
      ),
      body: Container(
        width: double.infinity,
        padding: const EdgeInsets.all(20),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Container(
              padding: const EdgeInsets.all(12),
              decoration: BoxDecoration(
                color: Colors.red.shade300,
                borderRadius: BorderRadius.circular(20),
                boxShadow: [
                  BoxShadow(
                    color: Colors.black.withOpacity(0.2),
                    blurRadius: 5,
                    offset: const Offset(0, 3),
```

```
                    ),
                  ],
                ),
                child: Row(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    const CircleAvatar(
                      radius: 24,
                      backgroundImage:
AssetImage("assets/images/avatar.png"),
                    ),
                    const SizedBox(width: 16.0),
                    Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: [
                        const Text(
                          'Hai, Welcome',
                          style: TextStyle(
                            fontSize: 14,
                            color: Colors.white70,
                          ),
                        ),
                        const SizedBox(height: 4),
                        Text(
                          username,
                          style: const TextStyle(
                            fontWeight: FontWeight.bold,
                            fontSize: 18,
                            color: Colors.white,
                          ),
                        ),
                      ],
                    ),
                  ],
                ),
              ),
              const SizedBox(height: 25.0),
              Column(
                children: [
                  const Padding(
                    padding: EdgeInsets.symmetric(horizontal: 10),
                    child: Row(
                      mainAxisAlignment: MainAxisAlignment.spaceBetween,
                      children: [
                        Text(
                          "Kategori",
                          style: TextStyle(
                            fontWeight: FontWeight.bold,
```

```
                          fontSize: 16,
                        ),
                      ),
                    ],
                  ),
                ),
                const SizedBox(height: 15),
                Container(
                  padding: const EdgeInsets.all(12),
                  decoration: BoxDecoration(
                    color: Colors.grey.shade200,
                    borderRadius: BorderRadius.circular(20),
                    boxShadow: [
                      BoxShadow(
                        color: Colors.grey.withOpacity(0.3),
                        spreadRadius: 2,
                        blurRadius: 5,
                        offset: const Offset(0, 3),
                      ),
                    ],
                  ),
                  child: SingleChildScrollView(
                    scrollDirection: Axis.horizontal,
                    child: Row(
                      children: List.generate(
                        categories.length,
                        (index) => Padding(
                          padding: const EdgeInsets.symmetric(horizontal:
8),
                          child: DashboardCard(
                            image: categories[index]['image'],
                            text: categories[index]['text'],
                            press: () {
                              Navigator.push(
                                context,
                                MaterialPageRoute(
                                  builder: (context) =>
                                      categories[index]['screen'],
                                ),
                              );
                            },
                          ),
                        ),
                      ),
                    ),
                  ),
                ),
              ],
```

```
              ),
            ],
          ),
        ),
      ),
    );
  }

  void _showLogoutConfirmationDialog(BuildContext context) {
    showDialog(
      context: context,
      builder: (BuildContext dialogContext) {
        return AlertDialog(
          title: const Text('Konfirmasi Logout'),
          content: const Text('Anda yakin ingin logout?'),
          actions: <Widget>[
            TextButton.icon(
              onPressed: () {
                Navigator.of(dialogContext).pop();
              },
              icon: Icon(Icons.cancel, color: Colors.red),
              label: const Text('Tidak', style: TextStyle(color:
Colors.red)),
            ),
            TextButton.icon(
              onPressed: () async {
                await AuthManager.logout();
                Navigator.pushAndRemoveUntil(
                  dialogContext,
                  MaterialPageRoute(
                    builder: (context) => const LoginScreen(),
                  ),
                  (Route<dynamic> route) => false,
                );
              },
              icon: Icon(Icons.check, color: Colors.green),
              label: const Text('Ya', style: TextStyle(color: Colors.green)),
            ),
          ],
        );
      },
    );
  }
}
```

Penjelasan :	1. Class dashboardscreen: kelas yang mewakili layar dashboard. Kelas ini stateful widget. 2. Class _DashboardScreenState: menyimpan data-data dan logika terkait dengan layar dashboard.

	3. Variabel dan state initialization: variabel kategori berisi daftar kategori dengan informasi gambar, teks, dan layar yang terakit. LoginData digunakan untuk mengelola data login pengguna menggunakan SharedPreferences. Username menyimpan nama pengguna yang akan ditampilkan di layar. 4. initState dan initial : initState adalah metode bawaan dari stateful widget yang dipanggil ketika state pertama kali dibuat. Initial digunakan untuk menginisialisasi data login dari sharedpreferences.

SOURCE CODE data_obat.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_obat/model/obat_model.dart';
import 'package:flutter_obat/service/api_obat.dart';
import 'package:cached_network_image/cached_network_image.dart';

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

  @override
  State<DataObat> createState() => _DataObatState();
}

class _DataObatState extends State<DataObat> {
  final ApiObat _dataService = ApiObat();
  List<ObatModel> obat = [];

  @override
  void initState() {
    super.initState();
    _dataService.getAllObat().then((value) {
      setState(() {
        obat = value?.toList() ?? [];
      });
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          "Psikofarmaka",
          style: TextStyle(
            fontWeight: FontWeight.bold,
            color: Colors.white,
          ),
        ),
        iconTheme: const IconThemeData(color: Colors.white),
```

```
      backgroundColor: Colors.blue.shade600,
    ),
    body: Container(
      width: double.infinity,
      padding: const EdgeInsets.all(20),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          const SizedBox(height: 20),
          Expanded(
            child: ListView.builder(
              itemCount: obat.length,
              itemBuilder: (context, index) {
                return Card(
                  child: ListTile(
                    leading: CachedNetworkImage(
                      imageUrl: obat[index].gambar,
                      width: 50,
                      height: 50,
                      fit: BoxFit.cover,
                    ),
                    title: Text(obat[index].namaObat),
                    trailing: ElevatedButton(
                      onPressed: () {
                        _showDetailsBottomSheet(context, obat[index]);
                      },
                      child: const Text('Details'),
                    ),
                  ),
                );
              },
            ),
          ),
        ],
      ),
    ),
  );
}

// Function to show details in a BottomSheet
void _showDetailsBottomSheet(BuildContext context, ObatModel obatModel) {
  showModalBottomSheet(
    context: context,
    builder: (BuildContext context) {
      return Container(
        width: double.infinity,
        padding: const EdgeInsets.all(16),
        child: Column(
```

```
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              const Center(
                child: Text(
                  "Details Obat",
                  style: TextStyle(
                    fontWeight: FontWeight.bold,
                    fontSize: 20,
                  ),
                ),
              ),
              const SizedBox(height: 15),
              const Text(
                "Jenis Obat: ",
                style: TextStyle(
                  fontWeight: FontWeight.bold,
                ),
              ),
              Container(
                padding: const EdgeInsets.all(7),
                decoration: BoxDecoration(
                  color: Colors.blue.shade200,
                  borderRadius: BorderRadius.circular(15),
                  boxShadow: [
                    BoxShadow(
                      color: Colors.grey.withOpacity(0.3),
                      spreadRadius: 2,
                      blurRadius: 5,
                      offset: const Offset(0, 3),
                    ),
                  ],
                ),
                child: Text(
                  obatModel.jenisObat,
                  style: const TextStyle(
                    fontWeight: FontWeight.normal,
                  ),
                ),
              ),
              const SizedBox(height: 15),
              const Text(
                "Nama Obat: ",
                style: TextStyle(
                  fontWeight: FontWeight.bold,
                ),
              ),
              Container(
                padding: const EdgeInsets.all(7),
```

```
            decoration: BoxDecoration(
              color: Colors.blue.shade200,
              borderRadius: BorderRadius.circular(15),
              boxShadow: [
                BoxShadow(
                  color: Colors.grey.withOpacity(0.3),
                  spreadRadius: 2,
                  blurRadius: 5,
                  offset: const Offset(0, 3),
                ),
              ],
            ),
            child: Text(
              obatModel.namaObat,
              style: const TextStyle(
                fontWeight: FontWeight.normal,
              ),
            ),
          ),
        ),
        const SizedBox(height: 15),
        const Text(
          "Deskripsi Obat: ",
          style: TextStyle(
            fontWeight: FontWeight.bold,
          ),
        ),
        Container(
          padding: const EdgeInsets.all(7),
          decoration: BoxDecoration(
            color: Colors.blue.shade200,
            borderRadius: BorderRadius.circular(15),
            boxShadow: [
              BoxShadow(
                color: Colors.grey.withOpacity(0.3),
                spreadRadius: 2,
                blurRadius: 5,
                offset: const Offset(0, 3),
              ),
            ],
          ),
          child: Text(
            obatModel.deskripsi,
            style: const TextStyle(
              fontWeight: FontWeight.normal,
            ),
            textAlign: TextAlign.justify,
          ),
        ),
```

```
              const SizedBox(height: 20),
              ElevatedButton(
                onPressed: () {
                  Navigator.pop(context); // Close the BottomSheet
                },
                child: const Text('Close'),
              ),
            ],
          ),
        );
      },
    );
  }
}
```

Penjelasan :	1. final ApiObat _dataService = ApiObat(); dan List<ObatModel> obat = []; variabel ini digunakn untuk berinteraksi dengan API dan menyimpan data obat. 2. void initState() {}: pada fungsi ini, semua data obat diambil dari server dan disimpan di variabel. 3. void _showDetailsBottomSheet(BuildContext context, ObatModel obatModel) {}: fungsi ini digunakan untuk menampilkan detail obat dalam modal bottom sheet.

- **Screen/login**

SOURCE CODE login_screen.dart

```
import 'package:flutter_obat/service/api_user.dart';
import 'package:flutter/material.dart';
import 'package:flutter_obat/service/auth_manager.dart';
import 'package:flutter_obat/view/screen/register_screen.dart';
import 'package:flutter_obat/view/widget/bottom.dart';
import 'package:flutter_obat/model/user_model.dart';
import 'package:flutter_obat/view/widget/bottom_user.dart';

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

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

class _LoginScreenState extends State<LoginScreen> {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

  final _usernameController = TextEditingController();
  final _passwordController = TextEditingController();

  final String admin = 'admin';
```

```dart
final ApiUser _dataService = ApiUser();

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

void checkLogin() async {
  bool isLoggedIn = await AuthManager.isLoggedIn();
  if (isLoggedIn) {
    if (await AuthManager.isAdmin()) {
      // ignore: use_build_context_synchronously
      Navigator.pushAndRemoveUntil(
        context,
        MaterialPageRoute(
          builder: (context) => const DynamicBottomNavBar(),
        ),
        (route) => false,
      );
    } else {
      // ignore: use_build_context_synchronously
      Navigator.pushAndRemoveUntil(
        context,
        MaterialPageRoute(
          builder: (context) => const DynamicBottomNavBarUser(),
        ),
        (route) => false,
      );
    }
  }
}

@override
void dispose() {
  _usernameController.dispose();
  _passwordController.dispose();
  super.dispose();
}

String? _validateUsername(String? value) {
  if (value == null || value.isEmpty) {
    return 'Username tidak boleh kosong';
  }
  if (value.length < 5) {
    return 'Masukkan minimal 5 karakter';
  }
```

```
  return null;
}

String? _validatePassword(String? value) {
  if (value == null || value.isEmpty) {
    return 'Password tidak boleh kosong';
  }
  if (value.length < 5) {
    return 'Masukkan minimal 5 karakter';
  }
  return null;
}

@override
Widget build(BuildContext context) {
  return GestureDetector(
    onTap: () {
      FocusScope.of(context).unfocus();
    },
    child: Scaffold(
      appBar: AppBar(
        title: const Text(
          "Psikofarmaka",
          style: TextStyle(
            fontWeight: FontWeight.bold,
          ),
        ),
        centerTitle: true,
      ),
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Form(
            key: _formKey,
            child: Column(
              children: [
                const Padding(
                  padding: EdgeInsets.all(8.0),
                  child: Column(
                    children: [
                      Text(
                        'Halo, Selamat Datang',
                        style: TextStyle(
                          fontSize: 20.0,
                          fontWeight: FontWeight.bold,
                        ),
                      ),
                      Text(
```

```
                    'Silahkan Login!',
                    style: TextStyle(
                      fontSize: 15.0,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                ],
              ),
            ),
            const SizedBox(height: 40),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: TextFormField(
                validator: _validateUsername,
                controller: _usernameController,
                decoration: InputDecoration(
                  prefixIcon: const Icon(Icons.account_circle_rounded),
                  hintText: 'Write username here...',
                  labelText: 'Username',
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(10),
                  ),
                  contentPadding: const EdgeInsets.symmetric(
                    horizontal: 16,
                    vertical: 12,
                  ),
                  fillColor: const Color.fromARGB(255, 242, 254, 255),
                  filled: true,
                ),
              ),
            ),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: TextFormField(
                obscureText: true,
                controller: _passwordController,
                validator: _validatePassword,
                decoration: InputDecoration(
                  prefixIcon: const Icon(Icons.password_rounded),
                  hintText: 'Write password here...',
                  labelText: 'Password',
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(10),
                  ),
                  contentPadding: const EdgeInsets.symmetric(
                    horizontal: 16,
                    vertical: 12,
                  ),
```

```
                        fillColor: const Color.fromARGB(255, 242, 254, 255),
                        filled: true,
                      ),
                    ),
                  ),
                  const SizedBox(height: 20),
                  SizedBox(
                    width: double.infinity,
                    child: ElevatedButton(
                      onPressed: () async {
                        final isValidForm =
_formKey.currentState!.validate();
                        if (isValidForm) {
                          final postModel = LoginInput(
                            username: _usernameController.text,
                            password: _passwordController.text,
                          );
                          LoginResponse? res =
                              await _dataService.login(postModel);
                          if (res!.status == 200) {
                            await
AuthManager.login(_usernameController.text);
                            if (_usernameController.text.toLowerCase() ==
                                admin) {
                              // ignore: use_build_context_synchronously
                              Navigator.pushAndRemoveUntil(
                                context,
                                MaterialPageRoute(
                                  builder: (context) =>
                                      const DynamicBottomNavBar(),
                                ),
                                (route) => false,
                              );
                            } else {
                              // ignore: use_build_context_synchronously
                              Navigator.pushAndRemoveUntil(
                                context,
                                MaterialPageRoute(
                                  builder: (context) =>
                                      const DynamicBottomNavBarUser(),
                                ),
                                (route) => false,
                              );
                            }
                          } else {
                            displaySnackbar(res.message);
                          }
                        }
```

```
                    },
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.orange.shade900,
                    ),
                    child: const Text(
                      "Login",
                      style: TextStyle(
                        color: Colors.white,
                      ),
                    ),
                  ),
                ),
                const SizedBox(height: 20),
                Row(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: [
                    const Text(
                      'Belum punya akun? ',
                      style: TextStyle(fontSize: 15),
                    ),
                    GestureDetector(
                      onTap: () {
                        Navigator.pushReplacement(
                          context,
                          MaterialPageRoute(
                            builder: (context) => const RegisterScreen(),
                          ),
                        );
                      },
                      child: const Text(
                        'Register',
                        style: TextStyle(
                          fontSize: 15.0,
                          fontWeight: FontWeight.bold,
                          color: Colors.orange,
                        ),
                      ),
                    ),
                  ],
                )
              ],
            ),
          ),
        ),
      ),
    );
  }
```

```
  dynamic displaySnackbar(String msg) {
    return ScaffoldMessenger.of(context)
        .showSnackBar(SnackBar(content: Text(msg)));
  }
}
```

Penjelasan :	1. final GlobalKey<FormState> _formKey = GlobalKey<FormState>(); dan final _usernameController = TextEditingController(); hingga _passwordController; variabel ini digunakan untuk menyimpan state dari form dan teks input. 2. final String admin = 'admin'; dan final ApiUser _dataService = ApiUser(); variabel ini digunakan untuk menyimpan username admin dan berinteraksi dengan API. 3. void initState() {}: dalam fungsi ini terdapat fungsi checkLogin() yang dipanggil untuk memeriksa apakah pengguna sudah masuk atau belum. 4. void checkLogin() async {}: fungsi ini digunakan untuk memeriksa apakah pengguna sudah masuk atau belum. Jika sudah masuk, maka pengguna akan diarahkan ke halaman DynamicBottomNavbar. 5. String? _validateUsername(String? value) {} dan String? _validatePassword(String? value) {}: fungsi ini digunakan untuk memvalidasi input username dan password. 6. dynamic displaySnackbar(String msg) {}: fungsi ini digunakan untuk menampilkan snackbar dengan pesan tertentu.

➢ Screen/register

SOURCE CODE

```dart
import 'package:flutter_obat/service/api_user.dart';
import 'package:flutter/material.dart';
import 'package:flutter_obat/view/screen/login_screen.dart';
import 'package:flutter_obat/model/user_model.dart';

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

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

class _RegisterScreenState extends State<RegisterScreen> {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

  final _usernameController = TextEditingController();
  final _phoneNumberController = TextEditingController();
  final _passwordController = TextEditingController();
  final _confirmPasswordController = TextEditingController();

  final ApiUser _dataService = ApiUser();
```

```dart
@override
void dispose() {
  _usernameController.dispose();
  _passwordController.dispose();
  super.dispose();
}

String? _validateUsername(String? value) {
  if (value == null || value.isEmpty) {
    return 'Username tidak boleh kosong';
  }
  if (value.length < 5) {
    return 'Masukkan minimal 5 karakter';
  }
  return null;
}

String? _validatePhoneNumber(String? value) {
  if (value == null || value.isEmpty) {
    return 'Nomor telepon tidak boleh kosong';
  }
  if (!RegExp(r'^[0-9]+$').hasMatch(value)) {
    return 'Nomor telepon hanya boleh mengandung karakter angka';
  }

  String cleanedValue = value.replaceAll(RegExp(r'\D'), '');

  if (!cleanedValue.startsWith('0')) {
    return 'Nomor telepon harus diawali dengan 0';
  }
  if (cleanedValue.length < 8) {
    return 'Nomor telepon minimal 8 karakter';
  }
  if (cleanedValue.length > 13) {
    return 'Nomor telepon maksimal 13 karakter';
  }
  return null;
}

String? _validatePassword(String? value) {
  if (value == null || value.isEmpty) {
    return 'Password tidak boleh kosong';
  }
  if (value.length < 5) {
    return 'Masukkan minimal 5 karakter';
  }
  return null;
```

```
}

String? _validateConfirmPassword(String? value) {
  if (value == null || value.isEmpty) {
    return 'Confirm Password tidak boleh kosong';
  }
  if (value != _passwordController.text) {
    return 'Password tidak sama';
  }
  return null;
}

@override
Widget build(BuildContext context) {
  return GestureDetector(
    onTap: () {
      FocusScope.of(context).unfocus();
    },
    child: Scaffold(
      appBar: AppBar(
        title: const Text(
          "Psikofarmaka",
          style: TextStyle(
            fontWeight: FontWeight.bold,
          ),
        ),
        centerTitle: true,
      ),
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Form(
            key: _formKey,
            child: Column(
              children: [
                const Padding(
                  padding: EdgeInsets.all(8.0),
                  child: Column(
                    children: [
                      Text(
                        'Halo, Selamat Datang',
                        style: TextStyle(
                          fontSize: 20.0,
                          fontWeight: FontWeight.bold,
                        ),
                      ),
                      Text(
                        'Silahkan Register!',
```

```
                        style: TextStyle(
                          fontSize: 15.0,
                          fontWeight: FontWeight.bold,
                        ),
                      ),
                    ],
                  ),
                ),
                const SizedBox(height: 40),
                Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: TextFormField(
                    validator: _validateUsername,
                    controller: _usernameController,
                    decoration: InputDecoration(
                      prefixIcon: const Icon(Icons.account_circle_rounded),
                      hintText: 'Write username here...',
                      labelText: 'Username',
                      border: OutlineInputBorder(
                        borderRadius: BorderRadius.circular(10),
                      ),
                      contentPadding: const EdgeInsets.symmetric(
                        horizontal: 16,
                        vertical: 12,
                      ),
                      fillColor: const Color.fromARGB(255, 242, 254, 255),
                      filled: true,
                    ),
                  ),
                ),
                Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: TextFormField(
                    validator: _validatePhoneNumber,
                    controller: _phoneNumberController,
                    decoration: InputDecoration(
                      prefixIcon: const Icon(Icons.account_circle_rounded),
                      hintText: 'Write phone number here...',
                      labelText: 'Phone Number',
                      border: OutlineInputBorder(
                        borderRadius: BorderRadius.circular(10),
                      ),
                      contentPadding: const EdgeInsets.symmetric(
                        horizontal: 16,
                        vertical: 12,
                      ),
                      fillColor: const Color.fromARGB(255, 242, 254, 255),
                      filled: true,
```

```
                    ),
                  ),
                ),
                Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: TextFormField(
                    obscureText: true,
                    controller: _passwordController,
                    validator: _validatePassword,
                    decoration: InputDecoration(
                      prefixIcon: const Icon(Icons.password_rounded),
                      hintText: 'Write password here...',
                      labelText: 'Password',
                      border: OutlineInputBorder(
                        borderRadius: BorderRadius.circular(10),
                      ),
                      contentPadding: const EdgeInsets.symmetric(
                        horizontal: 16,
                        vertical: 12,
                      ),
                      fillColor: const Color.fromARGB(255, 242, 254, 255),
                      filled: true,
                    ),
                  ),
                ),
                Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: TextFormField(
                    obscureText: true,
                    controller: _confirmPasswordController,
                    validator: _validateConfirmPassword,
                    decoration: InputDecoration(
                      prefixIcon: const Icon(Icons.password_rounded),
                      hintText: 'Write confirm password here...',
                      labelText: 'Confirm Password',
                      border: OutlineInputBorder(
                        borderRadius: BorderRadius.circular(10),
                      ),
                      contentPadding: const EdgeInsets.symmetric(
                        horizontal: 16,
                        vertical: 12,
                      ),
                      fillColor: const Color.fromARGB(255, 242, 254, 255),
                      filled: true,
                    ),
                  ),
                ),
                const SizedBox(height: 20),
```

```
                  SizedBox(
                    width: double.infinity,
                    child: ElevatedButton(
                      onPressed: () async {
                        final isValidForm =
_formKey.currentState!.validate();
                        if (isValidForm) {
                          final postModel = RegisterInput(
                            username: _usernameController.text,
                            phoneNumber: _phoneNumberController.text,
                            password: _passwordController.text,
                            confirmPassword: _confirmPasswordController.text,
                          );
                          RegisterResponse? res =
                              await _dataService.register(postModel);
                          if (res != null) {
                            if (res.status == 200) {
                              displaySnackbar(res.message);
                              // ignore: use_build_context_synchronously
                              Navigator.pushReplacement(
                                context,
                                MaterialPageRoute(
                                  builder: (context) => const LoginScreen(),
                                ),
                              );
                            } else {
                              displaySnackbar(res.message);
                            }
                          }
                        }
                      },
                      style: ElevatedButton.styleFrom(
                        backgroundColor: Colors.orange.shade900,
                      ),
                      child: const Text(
                        "Register",
                        style: TextStyle(
                          color: Colors.white,
                        ),
                      ),
                    ),
                  ),
                  const SizedBox(height: 20),
                  Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                      const Text(
                        'Sudah punya akun? ',
```

```
                    style: TextStyle(fontSize: 15),
                  ),
                  GestureDetector(
                    onTap: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const LoginScreen(),
                        ),
                      );
                    },
                    child: const Text(
                      'Login',
                      style: TextStyle(
                        fontSize: 15.0,
                        fontWeight: FontWeight.bold,
                        color: Colors.orange,
                      ),
                    ),
                  ),
                ),
              ],
            )
          ],
        ),
      ),
    ),
  );
 }

 dynamic displaySnackbar(String msg) {
   return ScaffoldMessenger.of(context)
       .showSnackBar(SnackBar(content: Text(msg)));
 }
}
```

Penjelasan :	1. final GlobalKey<FormState> _formKey = GlobalKey<FormState>(); dan final _usernameController = TextEditingController(); hingga _confirmPasswordController; variabel ini digunakan untuk menyimpan state dari form dan teks input. 2. final ApiUser _dataService = ApiUser(); variabel ini digunakan untuk berinteraksi dengan API. 3. void dispose() {}: fungsi ini digunakan untuk melakukan pembersihan atau pelepasan sumber daya yang telah dialokasikan pada saat kelas State tersebut dihancurkan atau tidak lagi diperlukan. 4. String? _validateUsername(String? value) {}, String? _validatePhoneNumber(String? value) {}, String? _validatePassword(String? value) {}, dan String?

	_validateConfirmPassword(String? value) {}: fungsi ini digunakan untuk memvalidasi input username, nomor telepon , password, dan konfirmasi password. 5. dynamic displaySnackbar(String msg) {}: fungsi ini digunakan untuk menampilkan snackbar dengan pesan tertentu.

- **Screen/index**

SOURCE CODE index_screen.dart

```
import 'package:flutter/material.dart';
// import 'package:flutter_obat/view/screen/login_screen.dart';
import 'package:flutter_obat/view/widget/bottom.dart';
import 'package:flutter_obat/view/widget/index.dart';

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

  @override
  State<IndexScreen> createState() => _IndexScreenState();
}

class _IndexScreenState extends State<IndexScreen> {
  int currentPage = 0;
  List<Map<String, String>> indexData = [
    {
      "text": "Welcome to Psikofarmaka, Let’s find out!",
      "image": "assets/images/splash_obat1.png"
    },
    {
      "text":
          "We help people to know about various kinds of drugs, \nespecially
psychopharmaceuticals",
      "image": "assets/images/splash_obat2.png"
    },
    {
      "text":
          "We show the types of mental illnesses and \nhospitals that provide
mental health services",
      "image": "assets/images/splash_obat3.png"
    },
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // appBar: AppBar(),
      body: SizedBox(
        width: double.infinity,
```

```
        child: SafeArea(
          child: Column(
            children: <Widget>[
              Expanded(
                flex: 3,
                child: PageView.builder(
                  onPageChanged: (value) {
                    setState(() {
                      currentPage = value;
                    });
                  },
                  itemCount: indexData.length,
                  itemBuilder: (context, index) => IndexContent(
                    image: indexData[index]["image"],
                    text: indexData[index]['text'],
                  ),
                ),
              ),
              Expanded(
                flex: 2,
                child: Padding(
                  padding: const EdgeInsets.symmetric(horizontal: 20),
                  child: Column(
                    children: <Widget>[
                      const Spacer(),
                      Row(
                        mainAxisAlignment: MainAxisAlignment.center,
                        children: List.generate(
                          indexData.length,
                          (index) => AnimatedContainer(
                            duration: const Duration(milliseconds: 200),
                            margin: const EdgeInsets.only(right: 5),
                            height: 6,
                            width: currentPage == index ? 20 : 6,
                            decoration: BoxDecoration(
                              color: currentPage == index
                                  ? Colors.blue
                                  : Colors.grey,
                              borderRadius: BorderRadius.circular(3),
                            ),
                          ),
                        ),
                      ),
                      const Spacer(flex: 3),
                      SizedBox(
                        width: double
                            .infinity, // Set lebar tombol menjadi sepanjang
layar
```

```
                      child: ElevatedButton(
                        onPressed: () {
                          Navigator.push(
                            context,
                            MaterialPageRoute(
                              // builder: (context) => const LoginScreen(),
                              builder: (context) =>
                                  const DynamicBottomNavBar(),
                            ),
                          );
                        },
                        child: const Text("Login"),
                      ),
                    ),
                    const Spacer(),
                  ],
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
}

Widget buildPage(String text, String image) {
  return Container(
    padding: const EdgeInsets.all(16.0),
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Text(
          text,
          style: const TextStyle(fontSize: 15.0),
          textAlign: TextAlign.center,
        ),
        const SizedBox(height: 20.0),
        Image.asset(
          image,
          height: 250.0,
          width: 250.0,
        ),
      ],
    ),
  );
```

Penjelasan :	1. Pernyataan Impor : kode untuk mengimpor paket dan modul yang diperlukan. 2. Deklarasi kelas indexscreen : sebagai StatefulWidget yang artinya dapat berubah seiring waktu. 3. Class _indexscreen : berisi variabel dan metode yang menentukan perilaku widget. 4. Variabel : currentPage untuk melacak halaman yang sedang ditampilkan, dan indexData untuk menyimpan data setiap halaman termasuk teks dan gambar yang akan ditampilkan. 5. Metode build : menggunakan Scaffold yang menyediakan kerangka lalu didalamnya menggunakan column untuk mengatur anak-anaknya. 6. PageView.builder : membuat daftar yang dapat digulir halaman demi halaman. onPageChanged balik diperbarui currentPage setiap kali halaman berubah. Setiap halaman menampilkan IndexContent dengan teks dan gambar sesuai dari indexdata. 7. AnimatedContainer : widget yang mewakili titik-titik di bagian bawah layar dan propertinya akan berubah sesuai dengan halaman yang sedang ditampilkan. 8. ElevatedButton : tombol navigasi untuk ke layar lain (dynamicbottomnavbar) saat ditekan.

➢ Screen/category

SOURCE CODE

```dart
import 'package:flutter/material.dart';
import 'package:flutter_obat/view/widget/category.dart';
import 'package:flutter_obat/view/screen/admin/list_user.dart';
import 'package:flutter_obat/service/auth_manager.dart';
import 'package:flutter_obat/view/screen/login_screen.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:flutter_obat/view/widget/bottom_obat.dart';
import 'package:flutter_obat/view/widget/bottom_penyakit.dart';
import 'package:flutter_obat/view/widget/bottom_rs.dart';

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

  @override
  State<CategoriesScreen> createState() => _CategoriesScreenState();
}

class _CategoriesScreenState extends State<CategoriesScreen> {
  List<Map<String, dynamic>> categories = [
    {
      "icon": "assets/icons/obat.svg",
      "text": "Obat",
      "data": "10",
      "screen": const DynamicBottomNavBarObat()
```

```
  },
  {
    "icon": "assets/icons/penyakit.svg",
    "text": "Penyakit",
    "data": "10",
    "screen": const DynamicBottomNavBarPenyakit()
  },
  {
    "icon": "assets/icons/rs.svg",
    "text": "Rumah Sakit",
    "data": "10",
    "screen": const DynamicBottomNavBarRS()
  },
  {
    "icon": "assets/icons/profile.svg",
    "text": "User",
    "data": "10",
    "screen": const ListUser()
  },
];

late SharedPreferences loginData;
String username = '';

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

void inital() async {
  loginData = await SharedPreferences.getInstance();
  setState(() {
    username = loginData.getString('username').toString();
  });
}

@override
Widget build(BuildContext context) {
  return GestureDetector(
    onTap: () {
      FocusScope.of(context).unfocus();
    },
    child: Scaffold(
      appBar: AppBar(
        title: const Text(
          "Psikofarmaka",
          style: TextStyle(
```

```
              fontWeight: FontWeight.bold,
              color: Colors.white,
            ),
          ),
          // centerTitle: true,
          iconTheme: const IconThemeData(color: Colors.white),
          backgroundColor: Colors.orange.shade800,
          actions: [
            IconButton(
              onPressed: () {
                _showLogoutConfirmationDialog(context);
              },
              icon: const Icon(Icons.logout),
              color: Colors.white,
            ),
          ],
        ),
        body: Container(
          width: double.infinity,
          padding: const EdgeInsets.all(20),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Container(
                padding: const EdgeInsets.all(12),
                decoration: BoxDecoration(
                  color: Colors.red.shade300,
                  borderRadius: BorderRadius.circular(20),
                  boxShadow: [
                    BoxShadow(
                      color: Colors.grey.withOpacity(0.3),
                      spreadRadius: 2,
                      blurRadius: 5,
                      offset: const Offset(0, 3),
                    ),
                  ],
                ),
                child: Row(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    const CircleAvatar(
                      radius: 24,
                      backgroundImage:
AssetImage("assets/images/avatar.png"),
                    ),
                    const SizedBox(width: 16.0),
                    Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
```

```
                    children: [
                      const Text(
                        'Hai, Welcome',
                        style: TextStyle(
                          fontSize: 14,
                          color: Colors.white70,
                        ),
                      ),
                      const SizedBox(height: 4),
                      Text(
                        username,
                        style: const TextStyle(
                          fontWeight: FontWeight.bold,
                          fontSize: 18,
                          color: Colors.white,
                        ),
                      ),
                    ],
                  ),
                ],
              ),
            ),
            const SizedBox(height: 25.0),
            Container(
              padding: const EdgeInsets.all(16),
              decoration: BoxDecoration(
                color: Colors.grey.shade100,
                borderRadius: BorderRadius.circular(20),
                boxShadow: [
                  BoxShadow(
                    color: Colors.grey.withOpacity(0.3),
                    spreadRadius: 2,
                    blurRadius: 5,
                    offset: const Offset(0, 3),
                  ),
                ],
              ),
              child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                crossAxisAlignment: CrossAxisAlignment.start,
                children: List.generate(
                  categories.length,
                  (index) => InkWell(
                    child: DataCard(
                      text: categories[index]["text"],
                      data: categories[index]["data"],
                    ),
                  ),
```

```
                ),
              ),
            ),
            const SizedBox(height: 25.0),
            Container(
              padding: const EdgeInsets.all(16),
              decoration: BoxDecoration(
                color: Colors.grey.shade100,
                borderRadius: BorderRadius.circular(20),
                boxShadow: [
                  BoxShadow(
                    color: Colors.grey.withOpacity(0.3),
                    spreadRadius: 2,
                    blurRadius: 5,
                    offset: const Offset(0, 3),
                  ),
                ],
              ),
              child: Column(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                crossAxisAlignment: CrossAxisAlignment.start,
                children: List.generate(
                  categories.length,
                  (index) => InkWell(
                    child: CategoryCard(
                      icon: categories[index]["icon"],
                      text: categories[index]["text"],
                      press: () {
                        Navigator.push(
                          context,
                          MaterialPageRoute(
                            builder: (context) =>
categories[index]["screen"],
                          ),
                        );
                      },
                    ),
                  ),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
```

```
  void _showLogoutConfirmationDialog(BuildContext context) {
    showDialog(
      context: context,
      builder: (BuildContext dialogContext) {
        return AlertDialog(
          title: const Text('Konfirmasi Logout'),
          content: const Text('Anda yakin ingin logout?'),
          actions: <Widget>[
            TextButton.icon(
              onPressed: () {
                Navigator.of(dialogContext).pop();
              },
              icon: Icon(Icons.cancel, color: Colors.red),
              label: const Text('Tidak', style: TextStyle(color:
Colors.red)),
            ),
            TextButton.icon(
              onPressed: () async {
                await AuthManager.logout();
                Navigator.pushAndRemoveUntil(
                  dialogContext,
                  MaterialPageRoute(
                    builder: (context) => const LoginScreen(),
                  ),
                  (Route<dynamic> route) => false,
                );
              },
              icon: Icon(Icons.check, color: Colors.green),
              label: const Text('Ya', style: TextStyle(color: Colors.green)),
            ),
          ],
        );
      },
    );
  }
}
```

Penjelasan :	1. List<Map<String, dynamic>> categories = [...] : variabel ini digunakan untuk menyimpan daftar kategori yang tersedia. 2. late SharedPreferences loginData; dan String username = ''; variabel ini digunakan untuk menyimpan data login dan username. 3. void initState() {} dan void inital() async {}: dalam fungsi ini data login diambil dari shared preferences dan disimpan di variabel username. 4. void _showLogoutConfirmationDialog(BuildContext context) {}: fungsi ini digunakan untuk menampilkan dialog konfirmasi logout. Jika pengguna memilih YA maka pengguna akan logout dan diarahkan ke halaman LoginScreen.

- **Screen/about**

SOURCE CODE

```
import 'package:flutter/material.dart';
import 'package:flutter_obat/view/screen/admin/profile_screen.dart';
import 'package:shared_preferences/shared_preferences.dart';

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

  @override
  State<MyShared> createState() => _MySharedState();
}

class _MySharedState extends State<MyShared> {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  final TextEditingController textController = TextEditingController();
  final TextEditingController textController2 = TextEditingController();
  late SharedPreferences loginData;

  @override
  void dispose() {
    textController.dispose();
    textController2.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          "Psikofarmaka",
          style: TextStyle(
            fontWeight: FontWeight.bold,
            color: Colors.white,
          ),
        ),
        iconTheme: const IconThemeData(color: Colors.white),
        backgroundColor: Colors.orange.shade800,
      ),
      body: Container(
        margin: const EdgeInsets.all(16),
        child: SingleChildScrollView(
          child: Form(
            key: _formKey,
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
```

```
                const SizedBox(height: 16),
                _buildCard(
                  'About Application',
                  'Aplikasi Psikofarmaka adalah platform yang menyediakan
solusi komprehensif untuk administrasi data obat, penyakit, dan rumah sakit.
Aplikasi ini memberikan pengalaman pengguna yang baik, mendukung tugas
administratif, dan menyediakan akses yang mudah ke data kesehatan.',
                  Icons.info_outline,
                  Colors.indigo.shade300,
                ),
                const SizedBox(height: 16),
                _buildCard(
                  'Purpose of Application',
                  '1. Memberikan solusi untuk admin dalam melakukan tugas
CRUD terkait obat, penyakit, dan rumah sakit, sehingga memperoleh manajemen
data yang efisien.\n'
                      '2. Menyediakan akses yang mudah bagi pengguna untuk
melihat informasi terkait obat, penyakit, dan rumah sakit, mendukung
pemahaman yang lebih baik tentang kesehatan.\n'
                      '3. Memungkinkan admin untuk dengan mudah menambah,
memperbarui, dan menghapus data obat, penyakit, dan rumah sakit, meningkatkan
produktivitas dalam administrasi informasi kesehatan.\n'
                      '4. Menyelenggarakan sistem keamanan yang terintegrasi
untuk memastikan bahwa hanya admin yang memiliki hak akses penuh terhadap
fungsi CRUD, sementara pengguna hanya dapat melihat data yang tersedia.',
                  Icons.assignment,
                  Colors.teal.shade300,
                ),
                const SizedBox(height: 16),
                _buildProfileCard(context),
              ],
            ),
          ),
        ),
      ),
    );
  }

  Widget _buildCard(
      String title, String content, IconData iconData, Color cardColor) {
    return Container(
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(16),
        gradient: LinearGradient(
          begin: Alignment.topLeft,
          end: Alignment.bottomRight,
          colors: [cardColor, cardColor.withOpacity(0.8)],
        ),
```

```
      boxShadow: [
        BoxShadow(
          color: Colors.grey.shade300,
          blurRadius: 10,
          offset: Offset(0, 5),
        ),
      ],
    ),
    margin: const EdgeInsets.only(bottom: 16),
    child: Padding(
      padding: const EdgeInsets.all(16),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Row(
            children: [
              Icon(
                iconData,
                color: Colors.black,
              ),
              const SizedBox(width: 8),
              Text(
                title,
                style: TextStyle(
                  fontSize: 20.0,
                  fontWeight: FontWeight.bold,
                  color: Colors.black,
                ),
              ),
            ],
          ),
          const SizedBox(height: 8),
          Text(
            content,
            style: TextStyle(
              fontSize: 16.0,
              color: Colors.white,
            ),
          ),
        ],
      ),
    ),
  );
}

Widget _buildProfileCard(BuildContext context) {
  return Hero(
    tag: 'profileHero',
```

```
      child: Material(
        color: Colors.transparent,
        child: Card(
          elevation: 5,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(16),
          ),
          margin: const EdgeInsets.only(bottom: 16),
          child: InkWell(
            onTap: () {
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => ProfileScreen(),
                ),
              );
            },
            child: Padding(
              padding: const EdgeInsets.all(16),
              child: Row(
                children: [
                  Icon(
                    Icons.person,
                    color: Colors.indigo.shade400,
                  ),
                  const SizedBox(width: 8),
                  Text(
                    'Developer Profile',
                    style: TextStyle(
                      fontSize: 16.0,
                      fontWeight: FontWeight.bold,
                      color: Colors.indigo.shade400,
                    ),
                  ),
                ],
              ),
            ),
          ),
        ),
      ),
    );
  }
```

Penjelasan :	1. Class _MySharedState : menyimpan data dan logika terkait dengan layar tersebut, termasuk formulir (_formKey) dan controller untuk dua buah teks. 2. Metode dispose() : digunakan untuk membersihkan atau membebaskan sumber daya yang digunakan, khususnya controller untuk teks, ketika layar ini dihapus atau tidak lagi digunakan. 3. Metode build() : membangun UI dari layar MyShared menggunakan

	berbagai komponen Flutter, seperti AppBar, Container, Column, Row, Form, dan lainnya. 4. Metode _buildCard() : mengembalikan widget yang merepresentasikan kartu dengan judul, konten, ikon, dan warna latar belakang tertentu.

- **Screen/profile**

SOURCE CODE

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

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

  @override
  State<ProfileScreen> createState() => _ProfileScreenState();
}

class _ProfileScreenState extends State<ProfileScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          "Psikofarmaka",
          style: TextStyle(
            fontWeight: FontWeight.bold,
            color: Colors.white,
          ),
        ),
        iconTheme: const IconThemeData(color: Colors.white),
        backgroundColor: Colors.orange.shade800,
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: <Widget>[
            const SizedBox(height: 20),
            const Center(
              child: Text(
                "Profile",
                style: TextStyle(
                  fontSize: 20,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ),
```

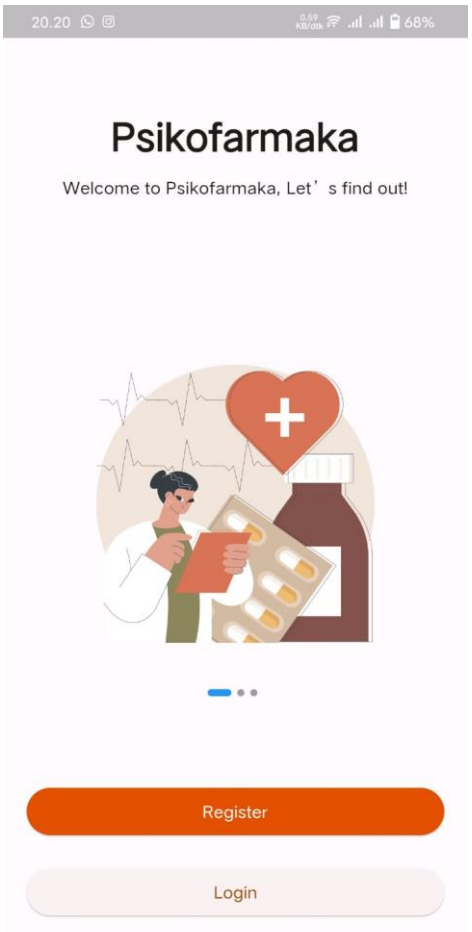
```
            const SizedBox(height: 20),
            _buildCreatorProfile(
                "Creator 1",
                "Dirga Febrian",
                "1214039",
                "@febriand_1",
                "Live life to the fullest",
                "assets/images/dirga.png"),
            const SizedBox(height: 20),
            _buildCreatorProfile(
                "Creator 2",
                "Juwita Stefany",
                "1214026",
                "@wops1e_",
                "Be yourself; everyone else is already taken",
                "assets/images/juwitas.png"),
          ],
        ),
      ),
    );
}

Widget _buildCreatorProfile(String title, String fullName, String npm,
    String socialMedia, String motto, String imagePath) {
  return Card(
    elevation: 10,
    child: ListTile(
      leading: CircleAvatar(
        radius: 30,
        backgroundImage: AssetImage(imagePath),
      ),
      title: Text(
        title,
        style: const TextStyle(fontWeight: FontWeight.bold),
      ),
      subtitle: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text("Nama: $fullName"),
          Text("NPM: $npm"),
          Text("Instagram: $socialMedia"),
          Text("Motto: $motto"),
        ],
      ),
    ),
  );
}
```

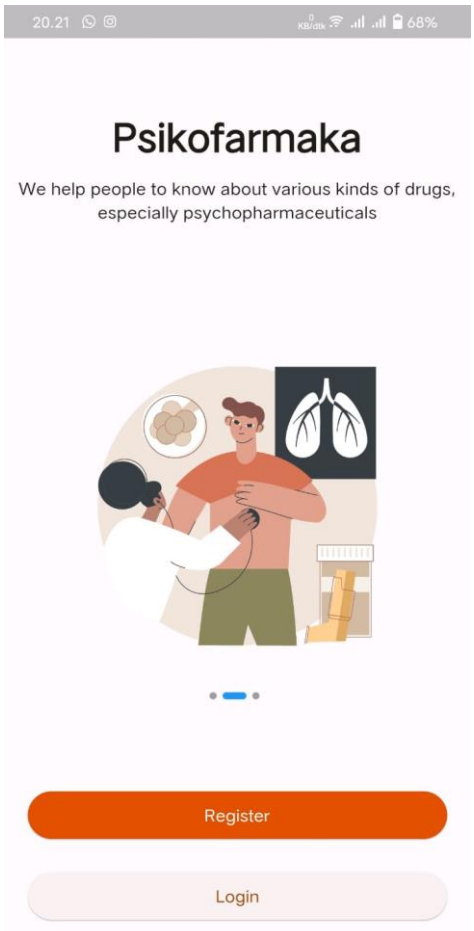
Penjelasan :	1. Class _ProfileScreenState : digunakan untuk menyimpan data dan logika terkait dengan layar tersebut. 2. Metode build() : membangun UI dari layar ProfileScreen menggunakan berbagai komponen Flutter, seperti AppBar, Column, Padding, dan lainnya. 3. Widget _buildCreatorProfile() : untuk membuat widget yang merepresentasikan profil kreator. Metode ini mengembalikan sebuah Card yang berisi ListTile dengan informasi tentang kreator. 4. Navigasi ke ProfileScreen : memiliki struktur dasar dengan app bar dan body yang berisi daftar profil kreator yang dibuat menggunakan _buildCreatorProfile().

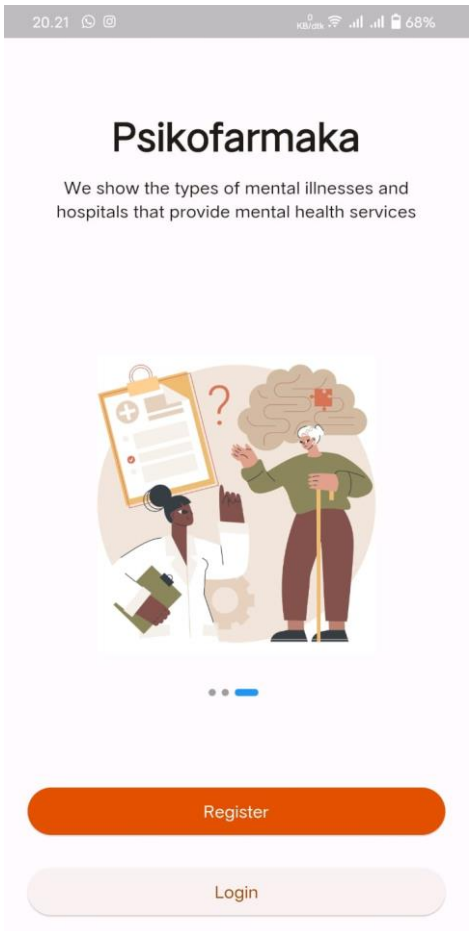
- **Output Program**

A. Splash Screen

Pada saat membuka aplikasi, maka layar yang tampil pertama kali dimuat ialah splash screen ini. Pada screen ini terdapat nama aplikasi, pesan singkat, dan gambar animasi yang dapat digeser sehingga membuat layar terlihat lebih menarik. Di screen ini juga terdapat dua button yaitu register dan login.

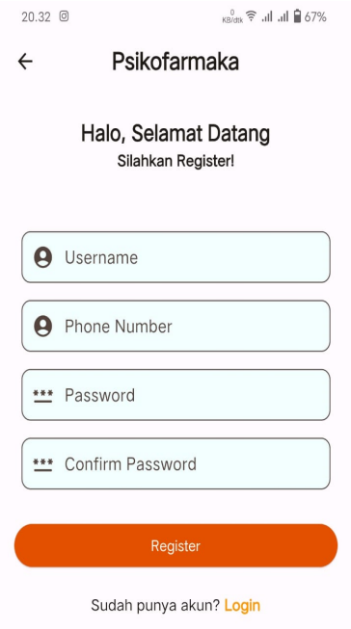


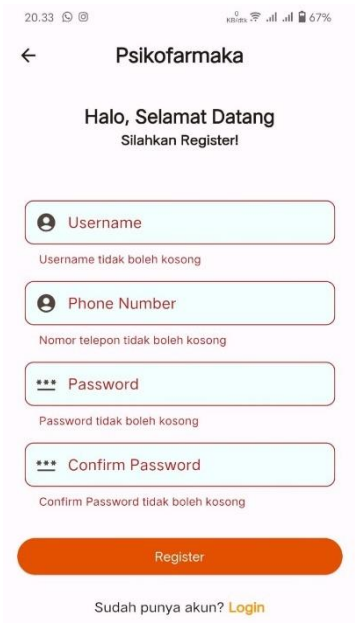




B. Register

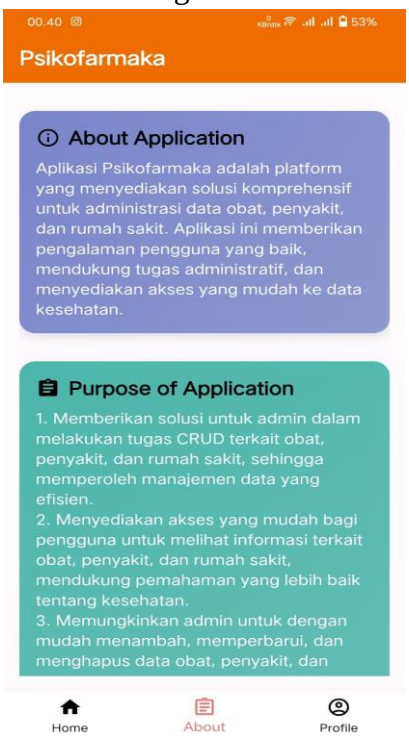
Pada halaman ini, pengguna diminta untuk melakukan registrasi akun dengan mengisi username, phone number, password, dan confirm password. Dan semua data harus wajib diisi karena jika tidak diisi namun sudah menekan button register maka akan menampilkan validasi bahwa data tidak boleh kosong. Ketika data sudah diisi, pengguna diarahkan ke halaman login dan terdapat alert bahwa registrasi berhasil.

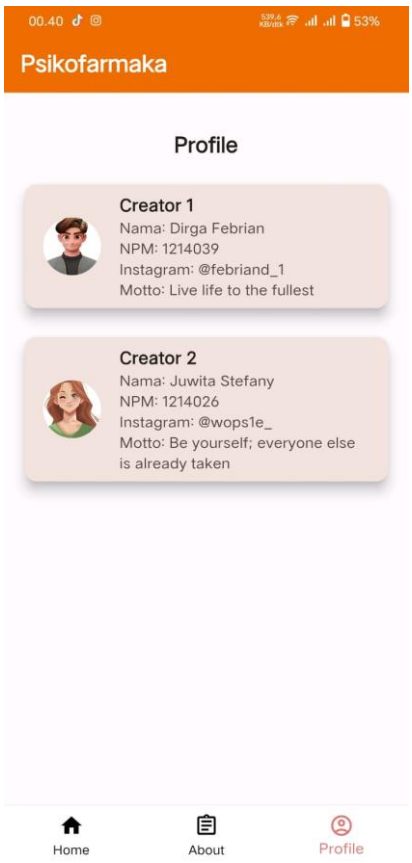


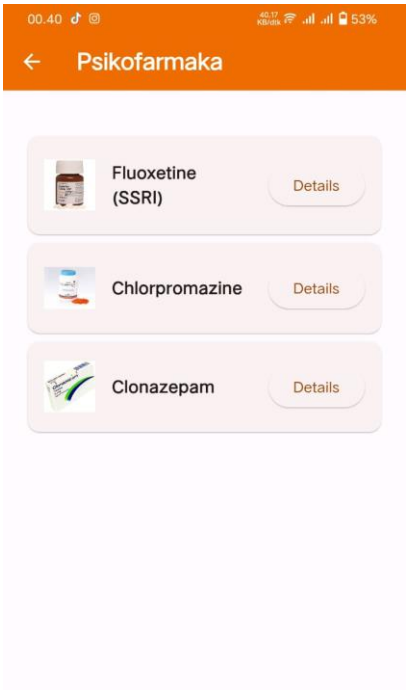


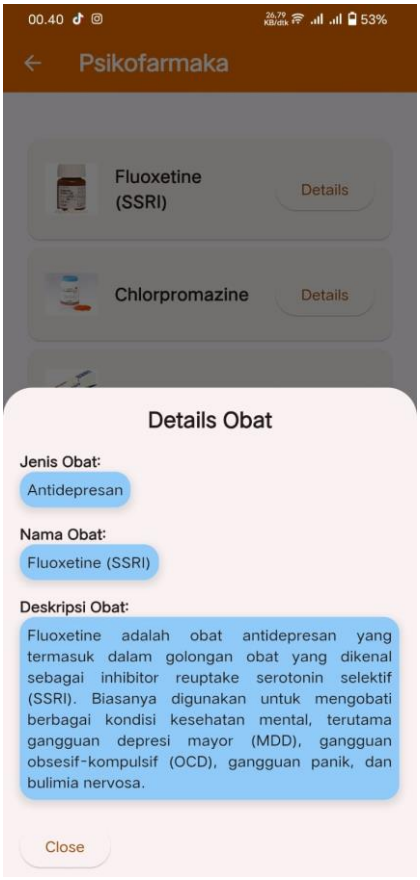
C. User

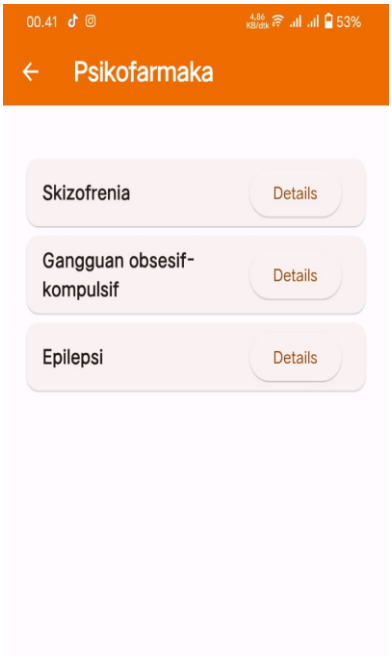
Setelah melakukan register, user dapat langsung masuk ke aplikasi dengan login terlebih dahulu. Kemudian akan tampil halaman dashboard dimana terdapat beberapa kategori data yang dapat dilihat oleh user. Pada setiap halaman kategorinya, user dapat melihat detail data. Selain itu, di halaman ini juga terdapat bottomnavbar about dan profile, dimana user dapat melihat deskripsi aplikasi dan tujuan aplikasi ini dibuat, serta user dapat melihat profile dari developer aplikasi. Jika tidak ada lagi yang ingin dilakukan user, maka user dapat melakukan logout dengan pilihan ya dan tidak, jika ya user akan diarahkan kembali pada halaman login.

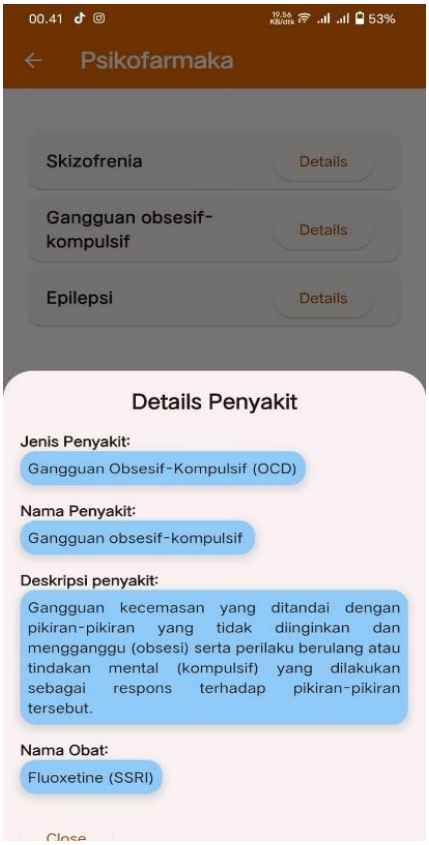


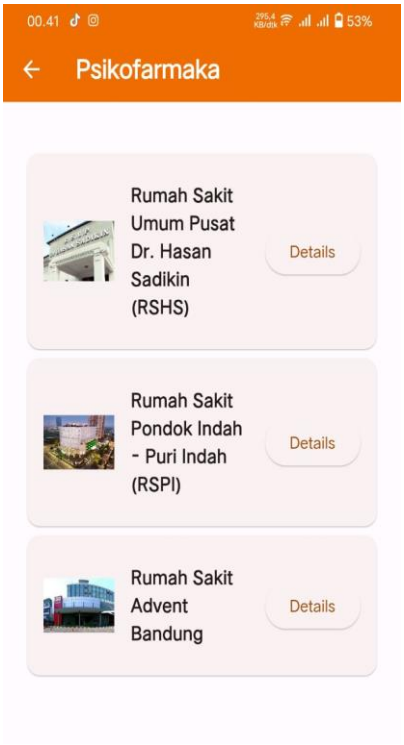

Psikofarmaka
Profile
Creator 1
Nama: Dirga Febrian
NPM: 1214039
Instagram: @febriand_1
Motto: Live life to the fullest
Creator 2
Nama: Juwita Stefany
NPM: 1214026
Instagram: @wops1e_
Motto: Be yourself; everyone else is already taken
Home
About
Profile

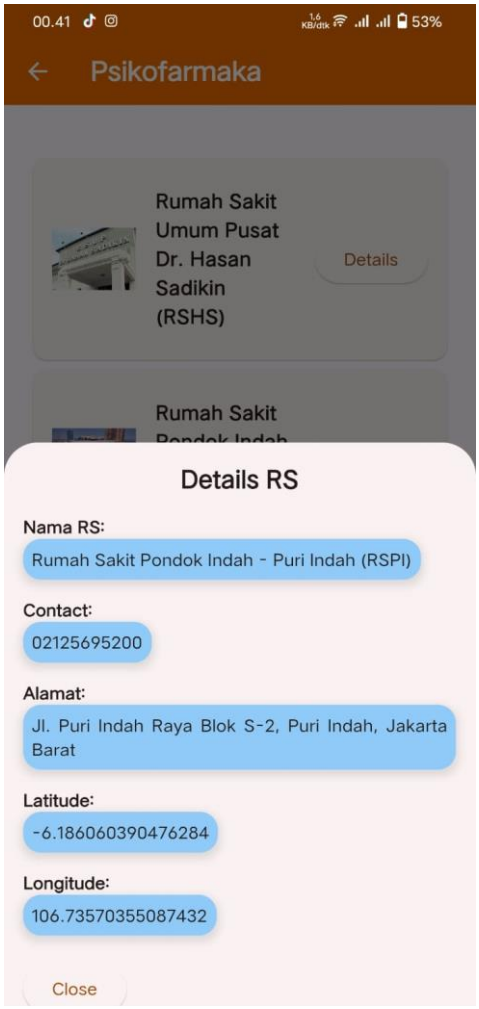

Psikofarmaka
Fluoxetine (SSRI)
Details
Chlorpromazine
Details
Clonazepam
Details


Psikofarmaka
Fluoxetine (SSRI)
Details
Chlorpromazine
Details
Details Obat
Jenis Obat:
Antidepresan
Nama Obat:
Fluoxetine (SSRI)
Deskripsi Obat:
Fluoxetine adalah obat antidepresan yang termasuk dalam golongan obat yang dikenal sebagai inhibitor reuptake serotonin selektif (SSRI). Biasanya digunakan untuk mengobati berbagai kondisi kesehatan mental, terutama gangguan depresi mayor (MDD), gangguan obsesif-kompulsif (OCD), gangguan panik, dan bulimia nervosa.
Close


Psikofarmaka
Skizofrenia
Details
Gangguan obsesif-kompulsif
Details
Epilepsi
Details

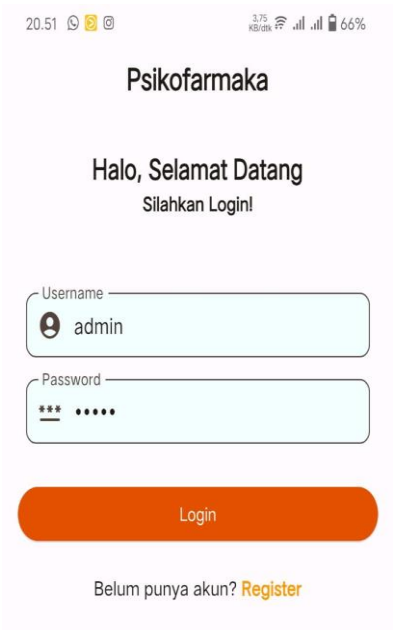

Psikofarmaka
Skizofrenia
Details
Gangguan obsesif-kompulsif
Details
Epilepsi
Details
Details Penyakit
Jenis Penyakit:
Gangguan Obsesif-Kompulsif (OCD)
Nama Penyakit:
Gangguan obsesif-kompulsif
Deskripsi penyakit:
Gangguan kecemasan yang ditandai dengan pikiran-pikiran yang tidak diinginkan dan mengganggu (obsesi) serta perilaku berulang atau tindakan mental (kompulsif) yang dilakukan sebagai respons terhadap pikiran-pikiran tersebut.
Nama Obat:
Fluoxetine (SSRI)
Close

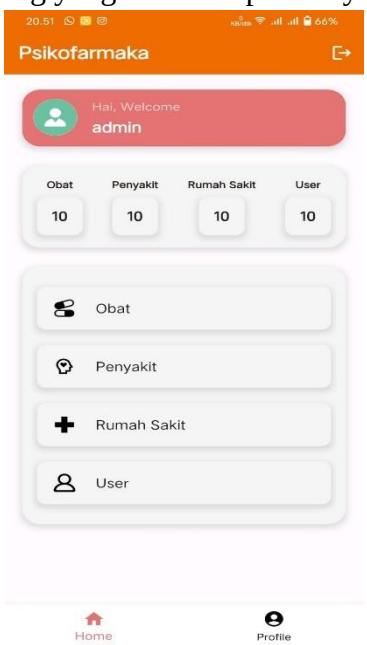

Psikofarmaka
Rumah Sakit Umum Pusat Dr. Hasan Sadikin (RSHS)
Details
Rumah Sakit Pondok Indah - Puri Indah (RSPI)
Details
Rumah Sakit Advent Bandung
Details

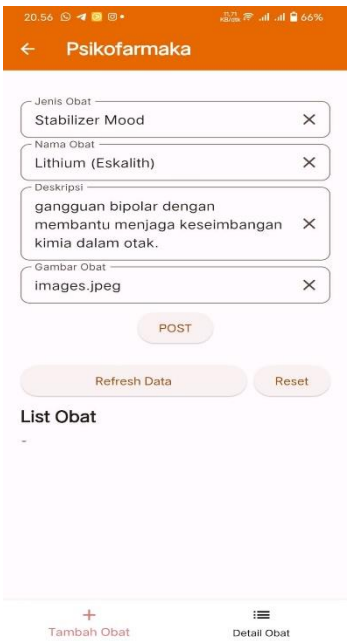


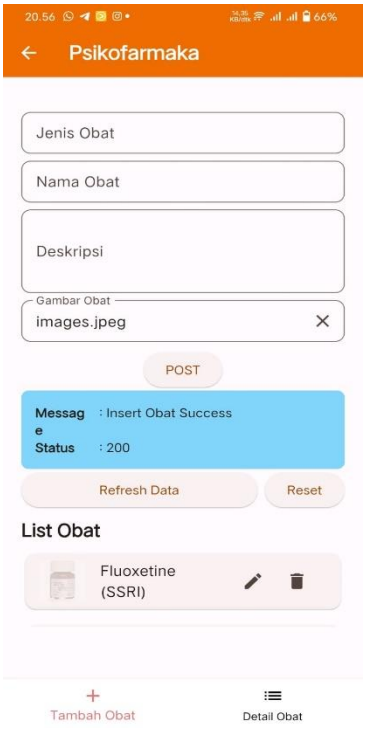
D. Admin

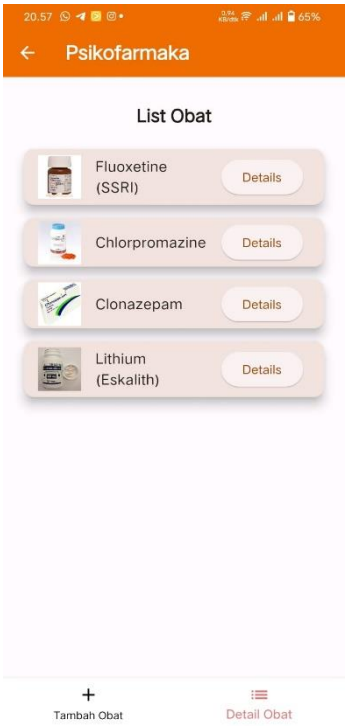
Admin dapat melakukan login ke dalam aplikasi dan akan diarahkan ke dashboard admin dimana terdapat beberapa category seperti obat, penyakit, rumah sakit, dan user. Pada halaman dashboard ini terdapat 2 bottomnavbar yaitu home dan profile. Di halaman ini, admin dapat melakukan CRUD terhadap beberapa kategori tersebut dan dapat melihat list dari data-data yang ada. Pada gambar dibawah ini hanya menampilkan CRUD dari data obat sebagai contohnya. Jika admin tidak memiliki kegiatan lagi pada halaman ini maka admin dapat melakukan logout account. Dan akan muncul LogOut Dialog yang memberi pilihan ya dan tidak.

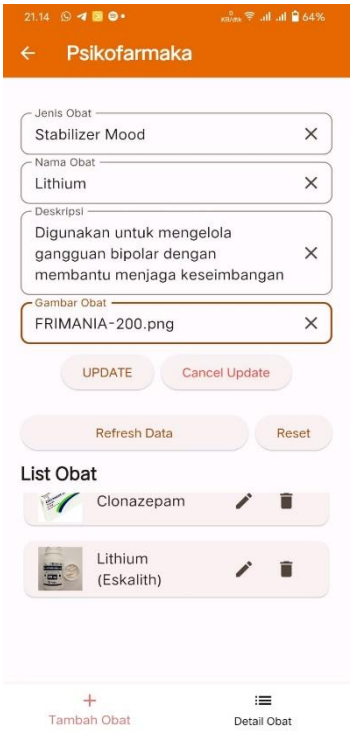




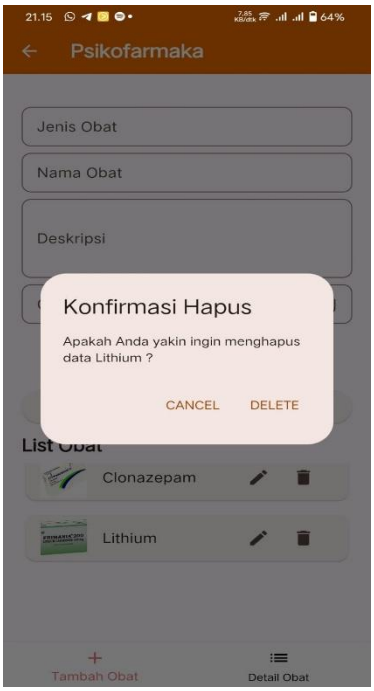


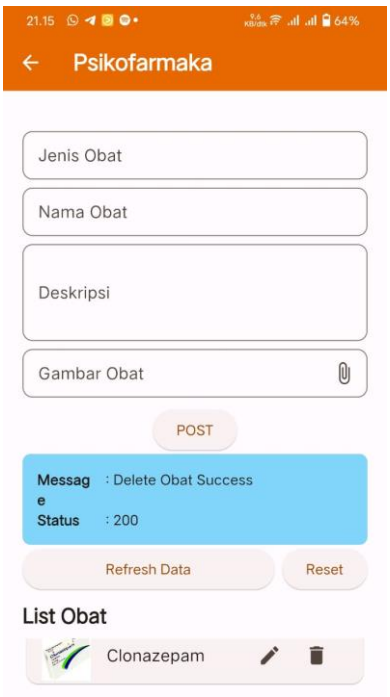

Psikofarmaka
Jenis Obat
Nama Obat
Deskripsi
Gambar Obat
images.jpeg
POST
Message : Insert Obat Success
Status : 200
Refresh Data
Reset
List Obat
Fluoxetine (SSRI)
Tambah Obat
Detail Obat


Psikofarmaka
List Obat
Fluoxetine (SSRI)
Details
Chlorpromazine
Details
Clonazepam
Details
Lithium (Eskalith)
Details
Tambah Obat
Detail Obat


Psikofarmaka
Jenis Obat
Stabilizer Mood
Nama Obat
Lithium
Deskripsi
Digunakan untuk mengelola gangguan bipolar dengan membantu menjaga keseimbangan
Gambar Obat
FRIMANIA-200.png
UPDATE
Cancel Update
Refresh Data
Reset
List Obat
Clonazepam
Lithium (Eskalith)
Tambah Obat
Detail Obat


Psikofarmaka
Jenis Obat
Nama Obat
Deskripsi
Gambar Obat
Masukkan gambar obat
POST
Message : Update Obat Success
Status : 200
Refresh Data
Reset
List Obat
Clonazepam
Lithium
Tambah Obat
Detail Obat


Psikofarmaka
Jenis Obat
Nama Obat
Deskripsi
Konfirmasi Hapus
Apakah Anda yakin ingin menghapus data Lithium ?
CANCEL
DELETE
List Obat
Clonazepam
Lithium
Tambah Obat
Detail Obat


Psikofarmaka
Jenis Obat
Nama Obat
Deskripsi
Gambar Obat
POST
Message : Delete Obat Success
Status : 200
Refresh Data
Reset
List Obat
Clonazepam


Psikofarmaka
Hai, Welcome
admin
Obat
Penyakit
Rumah Sakit
User
10
10
10
10
Konfirmasi Logout
Anda yakin ingin logout?
Tidak
Ya
Rumah Sakit
User
Home
Profile

PENUTUP

Aplikasi Flutter Obat merupakan aplikasi yang dirancang untuk memberikan Solusi dalam administrasi data obat, penyakit, dan rumah sakit. Aplikasi ini memungkinkan admin untuk melakukan operasi CRUD (Create,Read,Update,Delete) terkait obat,penyakit, dan rumah sakit secara efisien. Selain itu, aplikasi ini juga menyediakan akses yang mudah bagi pengguna untuk melihat informasi terkait obat,penyakit, dan rumah sakit.

Berikut adalah beberapa fitur utama dari aplikasi ini :

1. Autentikasi Pengguna
   Aplikasi ini memiliki fitur login dan register untuk memastikan bahwa hanya pengguna yang sah yang dapat mengakses aplikasi.
2. Administrasi Data
   Admin dapat menambah, memperbarui, dan menghapus data obat, penyakit, dan rumah sakit.
3. Akses Informasi
   Pengguna dapat melihat detail terkait obat, penyakit, dan rumah sakit.
4. Keamanan
   Aplikasi ini menghasilkan sistem keamanan yang baik untuk memastikan bahwa hanya admin yang memiliki hak akses penuh terhadap fungsi CRUD, sementara pengguna hanya dapat melihat data yang tersedia.