

Nama : Argo Iqbal Suranata

NIM : H1D022057

APLIKASI MANAJEMEN KESEHATAN PEMANTAUAN TIDUR

User Interface

1. Source Code Login_Page

```
import 'package:flutter/material.dart';
import 'package:health/bloc/login_bloc.dart';
import 'package:health/helpers/user_info.dart';
import 'package:health/ui/pemantauan_page.dart';
import 'package:health/ui/registrasi_page.dart';
import 'package:health/widget/warning_dialog.dart';

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

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

class _LoginPageState extends State<LoginPage> {
  final _formKey = GlobalKey<FormState>();
  bool _isLoading = false;
  final _emailTextboxController = TextEditingController();
  final _passwordTextboxController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Login', style: TextStyle(fontFamily: 'Calibri', fontSize: 20,
fontWeight: FontWeight.bold, color: Colors.white)),
        centerTitle: true,
        backgroundColor: const Color(0xFFFF0B0B), // Pastel pink color
        elevation: 5, // Add shadow effect
      ),
      body: Container(
        width: double.infinity,
        height: double.infinity,
        decoration: BoxDecoration(
          gradient: LinearGradient(
```

```

        colors: [
            const Color(0xFFFF0E68C), // Light pastel yellow at the bottom
            const Color(0xFFE7F0F4), // Light pastel blue at the top
        ],
        begin: Alignment.bottomCenter, // Start from the bottom
        end: Alignment.topCenter, // Gradient moves to the top
    ),
),
child: SingleChildScrollView(
    child: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
                const SizedBox(height: 100), // Spacer for centering
                _logo(),
                const SizedBox(height: 20),
                _loginForm(), // Form is now within a card with better frame
                const SizedBox(height: 20),
                _buttonRegistrasiText(), // Text button for registration
            ],
        ),
    ),
),
),
);
}

// Menampilkan Logo
Widget _logo() {
    return Column(
        children: [
            const Icon(
                Icons.health_and_safety,
                size: 80,
                color: Color(0xFFFF0B0B0), // Pastel pink color
            ),
            const SizedBox(height: 10),
            const Text(
                "Snooze Right, Live Bright!",
                style: TextStyle(
                    fontSize: 24,
                    fontFamily: 'Calibri',
                ),
            ),
        ],
    );
}

```

```

        fontWeight: FontWeight.bold,
        color: Color(0xFFFF0B0B0),
    ),
),
],
);
}

// Form login yang diletakkan di dalam Card dengan frame yang diperbagus
Widget _loginForm() {
return Card(
    shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(20), // Sudut lebih halus untuk Card
        side: BorderSide(
            color: Colors.grey.shade300, // Border warna abu-abu terang
            width: 1.5, // Ukuran border lebih tebal
        ),
    ),
    elevation: 8, // Tambahkan shadow untuk efek depth
    child: Padding(
        padding: const EdgeInsets.all(20.0), // Padding lebih besar
        child: Form(
            key: _formKey,
            child: Column(
                children: [
                    _emailTextField(),
                    const SizedBox(height: 10),
                    _passwordTextField(),
                    const SizedBox(height: 20),
                    _buttonLogin(), // Tombol login dengan desain kotak yang sedikit
                    rounded
                ],
            ),
        ),
    );
}

// Membuat Textbox Email
Widget _emailTextField() {
return TextFormField(
    decoration: const InputDecoration(
        labelText: "Email",

```

```

        border: OutlineInputBorder(),
        filled: true,
        fillColor: Colors.white,
    ),
    keyboardType: TextInputType.emailAddress,
    controller: _emailTextboxController,
    validator: (value) {
        if (value!.isEmpty) {
            return 'Email harus diisi';
        }
        return null;
    },
    style: const TextStyle(fontFamily: 'Calibri'),
);
}

// Membuat Textbox Password
Widget _passwordTextField() {
    return TextFormField(
        decoration: const InputDecoration(
            labelText: "Password",
            border: OutlineInputBorder(),
            filled: true,
            fillColor: Colors.white,
        ),
        keyboardType: TextInputType.text,
        obscureText: true,
        controller: _passwordTextboxController,
        validator: (value) {
            if (value!.isEmpty) {
                return "Password harus diisi";
            }
            return null;
        },
        style: const TextStyle(fontFamily: 'Calibri'),
    );
}

// Membuat Tombol Login dengan bentuk kotak sedikit rounded
Widget _buttonLogin() {
    return ElevatedButton(
        style: ElevatedButton.styleFrom(
            backgroundColor: const Color(0xFFFF0B0B), // Pastel pink color

```

```

padding: const EdgeInsets.symmetric(horizontal: 40, vertical: 15),
shape: RoundedRectangleBorder(
  borderRadius: BorderRadius.circular(8), // Rounded lebih kecil
),
),
child: const Text(
  "Login",
  style: TextStyle(fontFamily: 'Calibri', fontSize: 18, color: Colors.white),
),
onPressed: () {
  var validate = _formKey.currentState!.validate();
  if (validate) {
    if (!_isLoading) _submit();
  }
},
);
}

// Membuat Tombol Registrasi sebagai teks underline
Widget _buttonRegistrasiText() {
  return TextButton(
    onPressed: () {
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => const RegistrasiPage()),
      );
    },
    child: const Text(
      "Belum punya akun? Registrasi",
      style: TextStyle(
        fontFamily: 'Calibri',
        fontSize: 16,
        color: Colors.blue,
        decoration: TextDecoration.underline, // Underline the text
      ),
    ),
  );
}

void _submit() {
  _formKey.currentState!.save();
  setState(() {
    _isLoading = true;
  });
}

```

```

});

LoginBloc.login(
  email: _emailTextboxController.text,
  password: _passwordTextboxController.text,
).then((value) async {
  print("Response: ${value.toString()}");
  if (value.code == 200) {
    if (value.token != null) {
      await UserInfo().setToken(value.token!);
    } else {
      _showWarningDialog("Token is null");
    }

    if (value.userID != null) {
      await UserInfo().setUserID(value.userID!);
    } else {
      _showWarningDialog("User ID is null");
    }

    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (context) => const
PemantauanTidurPage()),
    );
  } else {
    _showWarningDialog("Login gagal, silahkan coba lagi");
  }
}, onError: (error) {
  print("Error: $error");
  _showWarningDialog("Login gagal, silahkan coba lagi");
}).whenComplete(() {
  setState(() {
    _isLoading = false;
  });
});
}

void _showWarningDialog(String message) {
  showDialog(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) => WarningDialog(

```

<pre> description: message,),); } } </pre>	
<p>Halaman ini terdiri dari sebuah form untuk input email dan password, serta tombol untuk melakukan login dan navigasi ke halaman registrasi. Komponen visual seperti teks, ikon, dan tombol didesain dengan warna-warna pastel dan sudut yang lebih halus untuk tampilan yang lebih menarik. Saat tombol login ditekan, validasi form dijalankan, dan jika validasi berhasil, proses login dilakukan menggunakan LoginBloc. Jika login berhasil, token dan user ID disimpan, kemudian pengguna diarahkan ke halaman pemantauan tidur. Jika login gagal, muncul dialog peringatan yang menampilkan pesan kesalahan.</p>	
Input Valid	Input Invalid
ibal@gmail.com	i@gmail.com
123456	654321
	

2. Source Code Registrasi_Page

```

import 'package:flutter/material.dart';
import 'package:health/bloc/registrasi_bloc.dart';
import 'package:health/widget/success_dialog.dart';
import 'package:health/widget/warning_dialog.dart';

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

```

```

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

class _RegistrasiPageState extends State<RegistrasiPage> {
  final _formKey = GlobalKey<FormState>();
  bool _isLoading = false;

  final _namaTextboxController = TextEditingController();
  final _emailTextboxController = TextEditingController();
  final _passwordTextboxController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Registrasi"),
        centerTitle: true,
        backgroundColor: const Color(0xFFFF0B0B0), // Warna pastel pink yang
seragam
      ),
      body: Container(
        decoration: BoxDecoration(
          gradient: LinearGradient(
            colors: [
              const Color(0xFFFF0E68C), // Warna pastel kuning
              const Color(0xFFE7F0F4), // Warna pastel biru
            ],
            begin: Alignment.topLeft,
            end: Alignment.bottomRight,
          ),
        ),
        child: SingleChildScrollView(
          child: Padding(
            padding: const EdgeInsets.all(16.0), // Padding yang lebih besar
            child: Form(
              key: _formKey,
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  _namaTextField(),
                  const SizedBox(height: 16), // Ruang antar widget

```



```

        _emailTextField(),
        const SizedBox(height: 16), // Ruang antar widget
        _passwordTextField(),
        const SizedBox(height: 16), // Ruang antar widget
        _passwordKonfirmasiTextField(),
        const SizedBox(height: 20), // Ruang antar widget
        _buttonRegistrasi(),
      ],
    ),
  ),
),
),
);
}

```

// Membuat Textbox Nama

```

Widget _namaTextField() {
  return TextFormField(
    decoration: const InputDecoration(
      labelText: "Nama",
      border: OutlineInputBorder(), // Menambahkan border
    ),
    keyboardType: TextInputType.text,
    controller: _namaTextboxController,
    validator: (value) {
      if (value!.length < 3) {
        return "Nama harus diisi minimal 3 karakter";
      }
      return null;
    },
  );
}

```

// Membuat Textbox Email

```

Widget _emailTextField() {
  return TextFormField(
    decoration: const InputDecoration(
      labelText: "Email",
      border: OutlineInputBorder(), // Menambahkan border
    ),
    keyboardType: TextInputType.emailAddress,
    controller: _emailTextboxController,
  );
}

```

```

validator: (value) {
  if (value!.isEmpty) {
    return 'Email harus diisi';
  }

  // Validasi email
  Pattern pattern =
r'^((([^<>()[]\.,;:\s@\"']+(\.[^<>()[]\.,;:\s@\"']+)*)|(\".+\"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\)|((\[[a-zA-Z-0-9]+\.[a-zA-Z]{2,}))$);
  RegExp regex = RegExp(pattern.toString());
  if (!regex.hasMatch(value)) {
    return "Email tidak valid";
  }
  return null;
},
);
}

// Membuat Textbox Password
Widget _passwordTextField() {
  return TextFormField(
    decoration: const InputDecoration(
      labelText: "Password",
      border: OutlineInputBorder(), // Menambahkan border
    ),
    keyboardType: TextInputType.text,
    obscureText: true,
    controller: _passwordTextboxController,
    validator: (value) {
      if (value!.length < 6) {
        return "Password harus diisi minimal 6 karakter";
      }
      return null;
    },
  );
}

// Membuat Textbox Konfirmasi Password
Widget _passwordKonfirmasiTextField() {
  return TextFormField(
    decoration: const InputDecoration(
      labelText: "Konfirmasi Password",

```

```

        border: OutlineInputBorder(), // Menambahkan border
    ),
    keyboardType: TextInputType.text,
    obscureText: true,
    validator: (value) {
        if (value != _passwordTextboxController.text) {
            return "Konfirmasi Password tidak sama";
        }
        return null;
    },
);
}

// Membuat Tombol Registrasi
Widget _buttonRegistrasi() {
    return ElevatedButton(
        child: const Text("Registrasi"),
        style: ElevatedButton.styleFrom(
            backgroundColor: const Color(0xFFFF0B0B0), // Warna pastel pink
            padding: const EdgeInsets.symmetric(vertical: 16.0, horizontal: 32.0), //
        ),
        onPressed: () {
            var validate = _formKey.currentState!.validate();
            if (validate) {
                if (!_isLoading) _submit();
            }
        },
    );
}

void _submit() {
    _formKey.currentState!.save();
    setState(() {
        _isLoading = true;
    });

    RegistrasiBloc.registrasi(
        nama: _namaTextboxController.text,
        email: _emailTextboxController.text,
        password: _passwordTextboxController.text,
    );
}

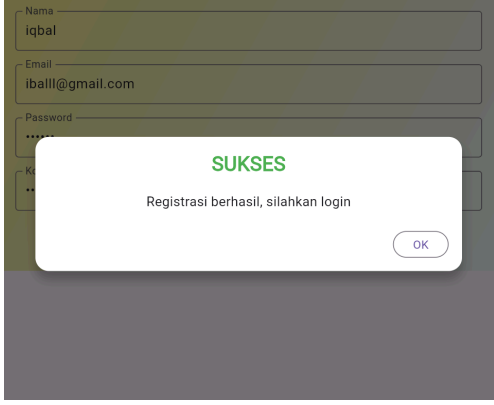
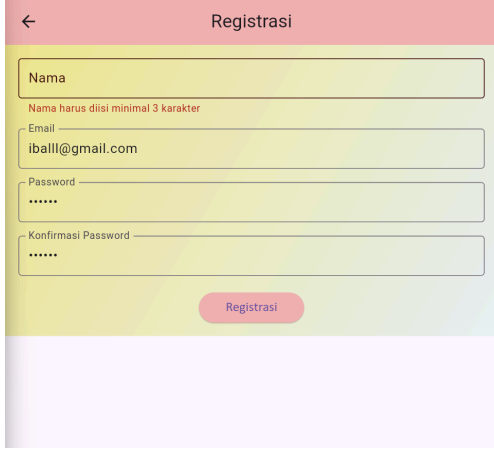
```

```

).then((value) {
  showDialog(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) => SuccessDialog(
      description: "Registrasi berhasil, silahkan login",
      onClick: () {
        Navigator.pop(context);
      },
    ),
  );
}, onError: (error) {
  showDialog(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) => const WarningDialog(
      description: "Registrasi gagal, silahkan coba lagi",
    ),
  );
}).whenComplete(() {
  setState(() {
    _isLoading = false;
  });
});
}
}

```

Kode di atas adalah implementasi halaman registrasi di Flutter yang memungkinkan pengguna untuk mendaftar akun baru. Pengguna harus mengisi formulir dengan nama, email, password, dan konfirmasi password. Setiap bagian dari form diperiksa: nama harus minimal 3 karakter, email harus valid, password harus setidaknya 6 karakter, dan konfirmasi password harus sama dengan password. Kalau semuanya sudah benar, pengguna bisa menekan tombol "Registrasi" dan data akan dikirim ke `RegistrasiBloc` untuk diproses. Jika berhasil, akan muncul pesan sukses, kalau gagal, akan muncul pesan peringatan. Desain halaman ini dibuat dengan warna-warna pastel yang lembut, dan tombol registrasi dibuat besar dan mudah terlihat untuk kenyamanan pengguna.

Input Valid	Input Invalid
iqbal	'kosong'
iballl@gmail.com	iballl@gmail.com
123456	123456
	

3. Source Code Pemantauan_Page

```
import 'package:flutter/material.dart';
import 'package:health/bloc/logout_bloc.dart';
import 'package:health/bloc/pemantauan_tidur_bloc.dart';
import 'package:health/model/pemantauan_tidur.dart';
import 'package:health/ui/health_detail.dart';
import 'package:health/ui/health_form.dart';
import 'package:health/ui/login_page.dart';

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

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

class _PemantauanTidurPageState extends State<PemantauanTidurPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
```

```

title: Text(
  'Sleep Monitoring',
  style: const TextStyle(fontFamily: 'Calibri', fontSize: 20),
),
centerTitle: true,
backgroundColor: const Color(0xFFFF0B0B0), // Pastel pink color
elevation: 5, // Add shadow effect
actions: [
  Padding(
    padding: const EdgeInsets.only(right: 20.0),
    child: GestureDetector(
      child: const Icon(Icons.add, size: 26.0),
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => PemantauanTidurForm(),
          ),
        );
      },
    ),
  ),
],
),
drawer: Drawer(
  child: ListView(
    children: [
      ListTile(
        title: Text('Logout', style: const TextStyle(fontFamily: 'Calibri')),
        trailing: const Icon(Icons.logout),
        onTap: () async {
          await LogoutBloc.logout().then((value) => {
            Navigator.of(context).pushAndRemoveUntil(
              MaterialPageRoute(builder: (context) => const LoginPage()),
              (route) => false,
            )
          });
        },
      ),
    ],
  ),
),
body: Container(

```

```

decoration: BoxDecoration(
  gradient: LinearGradient(
    colors: [
      const Color(0xFFFF0E68C), // Light pastel yellow
      const Color(0xFFE7F0F4), // Light pastel blue
    ],
    begin: Alignment.topLeft,
    end: Alignment.bottomRight,
  ),
),
child: FutureBuilder<List<PemantauanTidur>>(
  future: PemantauanTidurBloc.getData(),
  builder: (context, snapshot) {
    if (snapshot.hasError) print(snapshot.error);
    return snapshot.hasData
      ? ListPemantauanTidur(list: snapshot.data)
      : const Center(
        child: CircularProgressIndicator(),
      );
  },
),
);
}
}

class ListPemantauanTidur extends StatelessWidget {
  final List<PemantauanTidur>? list;
  const ListPemantauanTidur({Key? key, this.list}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      itemCount: list == null ? 0 : list!.length,
      itemBuilder: (context, i) {
        return ItemPemantauanTidur(pemantauanTidur: list![i]);
      },
    );
  }
}

class ItemPemantauanTidur extends StatelessWidget {
  final PemantauanTidur pemantauanTidur;

```

```

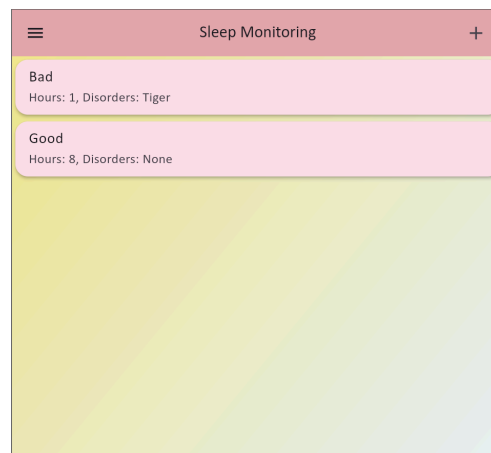
const ItemPemantauanTidur({Key? key, required this.pemantauanTidur})
: super(key: key);

@override
Widget build(BuildContext context) {
  return GestureDetector(
    onTap: () {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => PemantauanTidurDetail(
            pemantauanTidur: pemantauanTidur,
          ),
        ),
      );
    },
    child: Card(
      color: const Color(0xFFFFDDDE6), // Light pastel pink color
      elevation: 3, // Add shadow effect
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(15), // Rounded corners
      ),
      child: ListTile(
        title: Text(
          pemantauanTidur.sleepQuality!,
          style: const TextStyle(fontFamily: 'Calibri', fontSize: 18),
        ),
        subtitle: Text(
          'Hours: ${pemantauanTidur.sleepHours}, Disorders:
          ${pemantauanTidur.sleepDisorders}',
          style: const TextStyle(fontFamily: 'Calibri', fontSize: 16),
        ),
      ),
    ),
  );
}

```

Kode di atas adalah implementasi halaman pemantauan tidur dalam aplikasi Flutter yang menggunakan arsitektur BLoC (Business Logic Component). Halaman ini dimulai dengan kelas `PemantauanTidurPage`, yang mengatur tampilan dasar dengan `AppBar` berjudul "Sleep Monitoring" dan tombol

untuk menambahkan data pemantauan tidur baru. Terdapat juga `Drawer` yang memungkinkan pengguna untuk logout dan kembali ke halaman login. Di bagian tubuh halaman, data pemantauan tidur diambil menggunakan `FutureBuilder`, yang memanggil metode `getData()` dari `PemantauanTidurBloc`. Jika data berhasil diambil, data tersebut ditampilkan dalam daftar menggunakan `ListPemantauanTidur`. Setiap item dalam daftar diwakili oleh kelas `ItemPemantauanTidur`, yang menampilkan informasi tentang kualitas tidur, jam tidur, dan gangguan tidur. Ketika pengguna menekan item tersebut, aplikasi akan navigasi ke halaman detail pemantauan tidur menggunakan data yang dipilih. Dengan demikian, alur kode ini mengelola pengambilan dan tampilan data pemantauan tidur, serta navigasi antar halaman dengan menggunakan arsitektur yang terstruktur.



4. Source Code Health_Detail

```
import 'package:flutter/material.dart';
import 'package:health/bloc/pemantauan_tidur_bloc.dart';
import 'package:health/model/pemantauan_tidur.dart';
import 'package:health/ui/health_form.dart';
import 'package:health/ui/pemantauan_page.dart';
import 'package:health/widget/warning_dialog.dart';

class PemantauanTidurDetail extends StatefulWidget {
  final PemantauanTidur? pemantauanTidur;

  PemantauanTidurDetail({Key? key, this.pemantauanTidur}) : super(key: key);

  @override
  _PemantauanTidurDetailState createState() =>
    _PemantauanTidurDetailState();
}
```

```
}
```

```
class _PemantauanTidurDetailState extends State<PemantauanTidurDetail> {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: const Text(  
          'Sleep Monitoring Detail',  
          style: TextStyle(fontFamily: 'Calibri', fontSize: 24, fontWeight:  
FontWeight.bold),  
        ),  
        centerTitle: true,  
        backgroundColor: const Color(0xFFFF0B0B0), // Pastel pink color  
      ),  
      body: Container(  
        decoration: BoxDecoration(  
          gradient: LinearGradient(  
            colors: [  
              const Color(0xFFFF0E68C), // Light pastel yellow  
              const Color(0xFFE7F0F4), // Light pastel blue  
            ],  
            begin: Alignment.topLeft,  
            end: Alignment.bottomRight,  
          ),  
        ),  
        child: Center(  
          child: Padding(  
            padding: const EdgeInsets.all(20.0),  
            child: Column(  
              mainAxisAlignment: MainAxisAlignment.center,  
              children: [  
                Text(  
                  "Kualitas Tidur : ${widget.pemantauanTidur!.sleepQuality}",  
                  style: const TextStyle(  
                    fontSize: 22.0,  
                    fontFamily: 'Calibri',  
                    fontWeight: FontWeight.bold,  
                  ),  
                ),  
                const SizedBox(height: 15),  
                Text(  
                  "Jumlah Jam Tidur : ${widget.pemantauanTidur!.sleepHours} jam",
```

```

        style: const TextStyle(
          fontSize: 20.0,
          fontFamily: 'Calibri',
        ),
      ),
      const SizedBox(height: 15),
      Text(
        "Gangguan Tidur : ${widget.pemantauanTidur!.sleepDisorders}",
        style: const TextStyle(
          fontSize: 20.0,
          fontFamily: 'Calibri',
        ),
      ),
      const SizedBox(height: 30),
      _tombolHapusEdit(),
    ],
  ),
),
),
);
}

```

```

Widget _tombolHapusEdit() {
  return Row(
    mainAxisAlignment: MainAxisAlignment.min,
    children: [
      // Tombol Edit
      ElevatedButton(
        style: ElevatedButton.styleFrom(
          backgroundColor: const Color(0xFFFF0B0B0), // Pastel pink color
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(25), // Rounded corners
          ),
          padding: const EdgeInsets.symmetric(horizontal: 24, vertical: 12), //
Custom padding
        ),
        child: const Text(
          "EDIT",
          style: TextStyle(
            fontFamily: 'Calibri',
            fontSize: 16,
            fontWeight: FontWeight.bold,

```

```

    ),
  ),
  onPressed: () {
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => PemantauanTidurForm(
          pemantauanTidur: widget.pemantauanTidur!,
        ),
      ),
    );
  },
),
const SizedBox(width: 10), // Spacing between buttons
// Tombol Hapus
ElevatedButton(
  style: ElevatedButton.styleFrom(
    backgroundColor: const Color(0xFFE7F0F4), // Light pastel blue
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(25), // Rounded corners
    ),
    padding: const EdgeInsets.symmetric(horizontal: 24, vertical: 12), //
Custom padding
  ),
  child: const Text(
    "DELETE",
    style: TextStyle(
      fontFamily: 'Calibri',
      fontSize: 16,
      fontWeight: FontWeight.bold,
    ),
  ),
  onPressed: () => confirmHapus(),
),
],
);
}

void confirmHapus() {
  AlertDialog alertDialog = AlertDialog(
    content: const Text(
      "Yakin ingin menghapus data ini?",
      style: TextStyle(

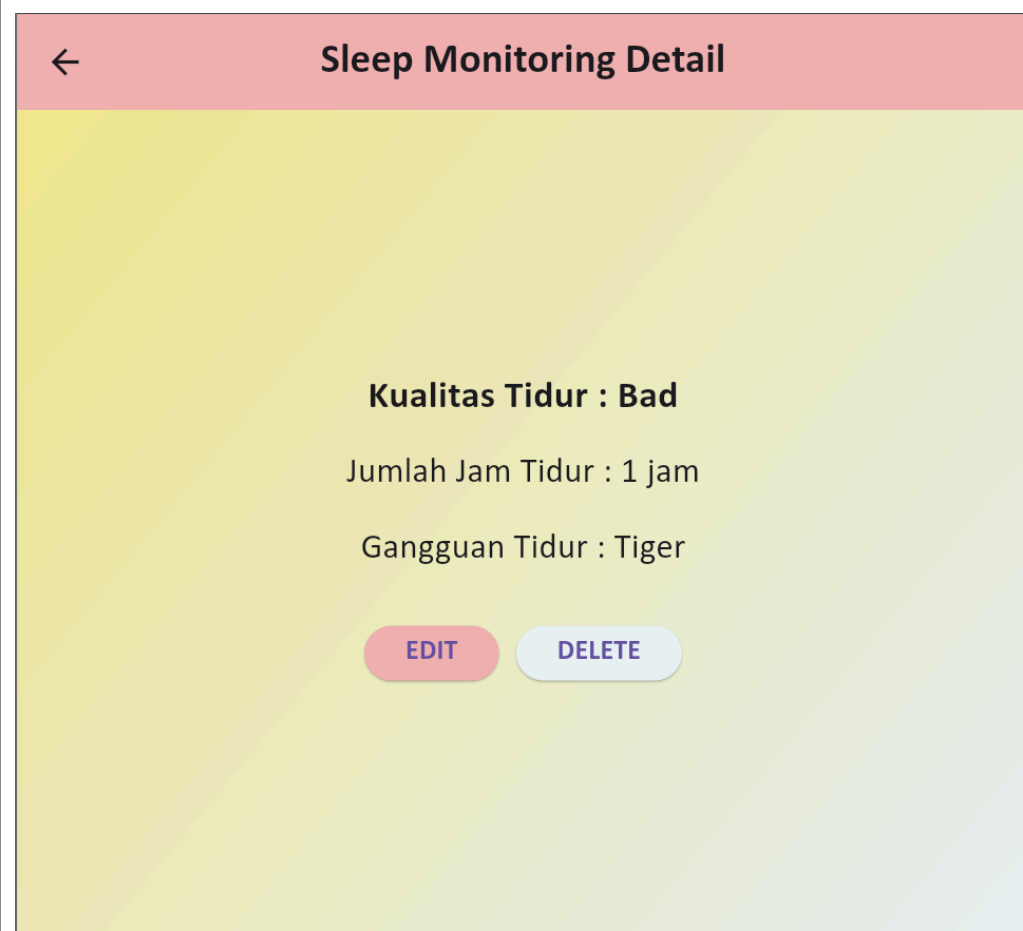
```

```

        fontFamily: 'Calibri',
        fontSize: 18,
      ),
    ),
    actions: [
      // Tombol Hapus
      OutlinedButton(
        child: const Text(
          "Ya",
          style: TextStyle(fontFamily: 'Calibri'),
        ),
        onPressed: () {
          PemantauanTidurBloc.deletePemantauanTidur(id:
widget.pemantauanTidur!.id!).then(
            (value) {
              Navigator.of(context).push(MaterialPageRoute(
                builder: (context) => const PemantauanTidurPage(),
              ));
            },
            onError: (error) {
              showDialog(
                context: context,
                builder: (BuildContext context) => const WarningDialog(
                  description: "Hapus gagal, silahkan coba lagi",
                ),
              );
            },
          );
        },
      ),
      // Tombol Batal
      OutlinedButton(
        child: const Text(
          "Batal",
          style: TextStyle(fontFamily: 'Calibri'),
        ),
        onPressed: () => Navigator.pop(context),
      ),
    ],
  );
  showDialog(builder: (context) => alertDialog, context: context);
}
}

```

Kode di atas merupakan implementasi halaman detail pemantauan tidur dalam aplikasi Flutter, yang berfungsi untuk menampilkan informasi rinci tentang data pemantauan tidur yang dipilih, seperti kualitas tidur, jumlah jam tidur, dan gangguan tidur. Halaman ini diwakili oleh kelas `PemantauanTidurDetail`, yang menerima objek `PemantauanTidur` sebagai parameter. Dalam metode `build`, tampilan dibuat dengan `AppBar` berjudul "Sleep Monitoring Detail" dan latar belakang bergradasi. Informasi dari objek `pemantauanTidur` ditampilkan di tengah halaman menggunakan widget teks. Selain itu, terdapat dua tombol: "EDIT" untuk mengarahkan pengguna ke halaman formulir pemantauan tidur untuk mengedit data dan "DELETE" yang memunculkan dialog konfirmasi sebelum menghapus data. Jika pengguna mengkonfirmasi penghapusan, data akan dihapus melalui metode `deletePemantauanTidur` di `PemantauanTidurBloc`, dan jika berhasil, pengguna akan kembali ke halaman pemantauan tidur. Jika terjadi kesalahan, dialog peringatan akan ditampilkan.



Edit	Delete

5. Source Code Health_Form

```
import 'package:flutter/material.dart';
import 'package:health/bloc/pemantauan_tidur_bloc.dart';
import 'package:health/model/pemantauan_tidur.dart';
import 'package:health/ui/pemantauan_page.dart';
import 'package:health/widget/warning_dialog.dart';

class PemantauanTidurForm extends StatefulWidget {
  PemantauanTidur? pemantauanTidur;

  PemantauanTidurForm({Key? key, this.pemantauanTidur}) : super(key: key);

  @override
  _PemantauanTidurFormState createState() =>
    _PemantauanTidurFormState();
}
```

```

}

class _PemantauanTidurFormState extends State<PemantauanTidurForm> {
  final _formKey = GlobalKey<FormState>();
  bool _isLoading = false;
  String judul = "TAMBAH DATA PEMANTAUAN TIDUR";
  String tombolSubmit = "SIMPAN";

  final _sleepQualityController = TextEditingController();
  final _sleepHoursController = TextEditingController();
  final _sleepDisordersController = TextEditingController();

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

  void isUpdate() {
    if (widget.pemantauanTidur != null) {
      setState(() {
        judul = "UBAH DATA PEMANTAUAN TIDUR";
        tombolSubmit = "UBAH";
        _sleepQualityController.text = widget.pemantauanTidur!.sleepQuality!;
        _sleepHoursController.text =
            widget.pemantauanTidur!.sleepHours.toString();
        _sleepDisordersController.text =
            widget.pemantauanTidur!.sleepDisorders!;
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(judul, style: const TextStyle(fontFamily: 'Calibri', fontSize: 20,
            fontWeight: FontWeight.bold)),
        centerTitle: true,
        backgroundColor: const Color(0xFFFF0B0B), // Pastel pink color
        elevation: 5,
      ),
      body: Container(

```



```

decoration: BoxDecoration(
  gradient: LinearGradient(
    colors: [
      const Color(0xFFFF0E68C), // Light pastel yellow
      const Color(0xFFE7F0F4), // Light pastel blue
    ],
    begin: Alignment.topLeft,
    end: Alignment.bottomRight,
  ),
),
child: SingleChildScrollView(
  child: Padding(
    padding: const EdgeInsets.all(16.0),
    child: Form(
      key: _formKey,
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          _sleepQualityTextField(),
          const SizedBox(height: 10),
          _sleepHoursTextField(),
          const SizedBox(height: 10),
          _sleepDisordersTextField(),
          const SizedBox(height: 20),
          _buttonSubmit(),
        ],
      ),
    ),
  ),
),
);
}

Widget _sleepQualityTextField() {
  return TextFormField(
    decoration: const InputDecoration(labelText: "Kualitas Tidur", border:
OutlineInputBorder(), filled: true, fillColor: Colors.white),
    keyboardType: TextInputType.text,
    controller: _sleepQualityController,
    validator: (value) {
      if (value!.isEmpty) {
        return "Kualitas Tidur harus diisi";
      }
    },
  );
}

```

```

    }
    return null;
  },
  style: const TextStyle(fontFamily: 'Calibri'),
);
}

Widget _sleepHoursTextField() {
  return TextFormField(
    decoration: const InputDecoration(labelText: "Jam Tidur", border:
OutlineInputBorder(), filled: true, fillColor: Colors.white),
    keyboardType: TextInputType.number,
    controller: _sleepHoursController,
    validator: (value) {
      if (value!.isEmpty) {
        return "Jam Tidur harus diisi";
      }
    },
    return null;
  },
  style: const TextStyle(fontFamily: 'Calibri'),
);
}

Widget _sleepDisordersTextField() {
  return TextFormField(
    decoration: const InputDecoration(labelText: "Gangguan Tidur", border:
OutlineInputBorder(), filled: true, fillColor: Colors.white),
    keyboardType: TextInputType.text,
    controller: _sleepDisordersController,
    validator: (value) {
      if (value!.isEmpty) {
        return "Gangguan Tidur harus diisi";
      }
    },
    return null;
  },
  style: const TextStyle(fontFamily: 'Calibri'),
);
}

Widget _buttonSubmit() {
  return OutlinedButton(
    style: OutlinedButton.styleFrom(
      backgroundColor: const Color(0xFFFF0B0B), // Pastel pink color

```

```

padding: const EdgeInsets.symmetric(horizontal: 40, vertical: 15),
shape: RoundedRectangleBorder(
  borderRadius: BorderRadius.circular(30), // Rounded corners
),
),
child: Text(tombolSubmit, style: const TextStyle(fontFamily: 'Calibri',
fontSize: 18)),
onPressed: () {
  var validate = _formKey.currentState!.validate();
  if (validate) {
    if (!_isLoading) {
      if (widget.pemantauanTidur != null) {
        ubah();
      } else {
        simpan();
      }
    }
  }
},
);
}

void simpan() {
  setState(() {
    _isLoading = true;
  });

  PemantauanTidur createPemantauan = PemantauanTidur(id: null);
  createPemantauan.sleepQuality = _sleepQualityController.text;
  createPemantauan.sleepHours = int.parse(_sleepHoursController.text);
  createPemantauan.sleepDisorders = _sleepDisordersController.text;

  PemantauanTidurBloc.addPemantauanTidur(pemantauanTidur:
createPemantauan).then((value) {
    Navigator.of(context).pushReplacement(MaterialPageRoute(builder:
(BuildContext context) => const PemantauanTidurPage()));
  }, onError: (error) {
    showDialog(
      context: context,
      builder: (BuildContext context) => const WarningDialog(
        description: "Simpan gagal, silahkan coba lagi",
      ),
    );
  });
}

```

```

    }).whenComplete(() {
      setState(() {
        _isLoading = false;
      });
    });
  }

  void ubah() {
    setState(() {
      _isLoading = true;
    });

    PemantauanTidur updatePemantauan = PemantauanTidur(id:
    widget.pemantauanTidur!.id!);
    updatePemantauan.sleepQuality = _sleepQualityController.text;
    updatePemantauan.sleepHours = int.parse(_sleepHoursController.text);
    updatePemantauan.sleepDisorders = _sleepDisordersController.text;

    PemantauanTidurBloc.updatePemantauanTidur(pemantauanTidur:
    updatePemantauan).then((value) {
      Navigator.of(context).pushReplacement(MaterialPageRoute(builder:
      (BuildContext context) => const PemantauanTidurPage()));
    }, onError: (error) {
      showDialog(
        context: context,
        builder: (BuildContext context) => const WarningDialog(
          description: "Permintaan ubah data gagal, silahkan coba lagi",
        ),
      );
    });
    }).whenComplete(() {
      setState(() {
        _isLoading = false;
      });
    });
  }
}

```

Kode di atas adalah implementasi halaman formulir pemantauan tidur dalam aplikasi Flutter yang memungkinkan pengguna untuk menambah atau mengubah data pemantauan tidur. Kelas `PemantauanTidurForm` menerima objek `PemantauanTidur` yang dapat digunakan untuk mengedit data yang ada. Pada `initState`, metode `isUpdate` digunakan untuk memeriksa apakah

data pemantauan tidur sudah ada, dan jika ada, judul halaman serta tombol disesuaikan dengan konteks update. Formulir ini terdiri dari tiga input: kualitas tidur, jumlah jam tidur, dan gangguan tidur, yang menggunakan widget `TextFormField` dan memiliki validasi untuk memastikan input tidak kosong. Pengguna dapat menyimpan data dengan menekan tombol "SIMPAN" atau "UBAH" (tergantung konteks) yang memicu metode `simpan` atau `ubah`. Metode ini mengatur status loading dan menggunakan `PemantauanTidurBloc` untuk menambahkan atau memperbarui data ke database. Jika operasi berhasil, pengguna akan dialihkan kembali ke halaman pemantauan tidur; jika gagal, dialog peringatan akan muncul. Selain itu, terdapat penggunaan `SingleChildScrollView` untuk memastikan tampilan formulir tetap dapat di-scroll jika diperlukan.

←

TAMBAH DATA PEMANTAUAN TIDUR

Kualitas Tidur

So Bad

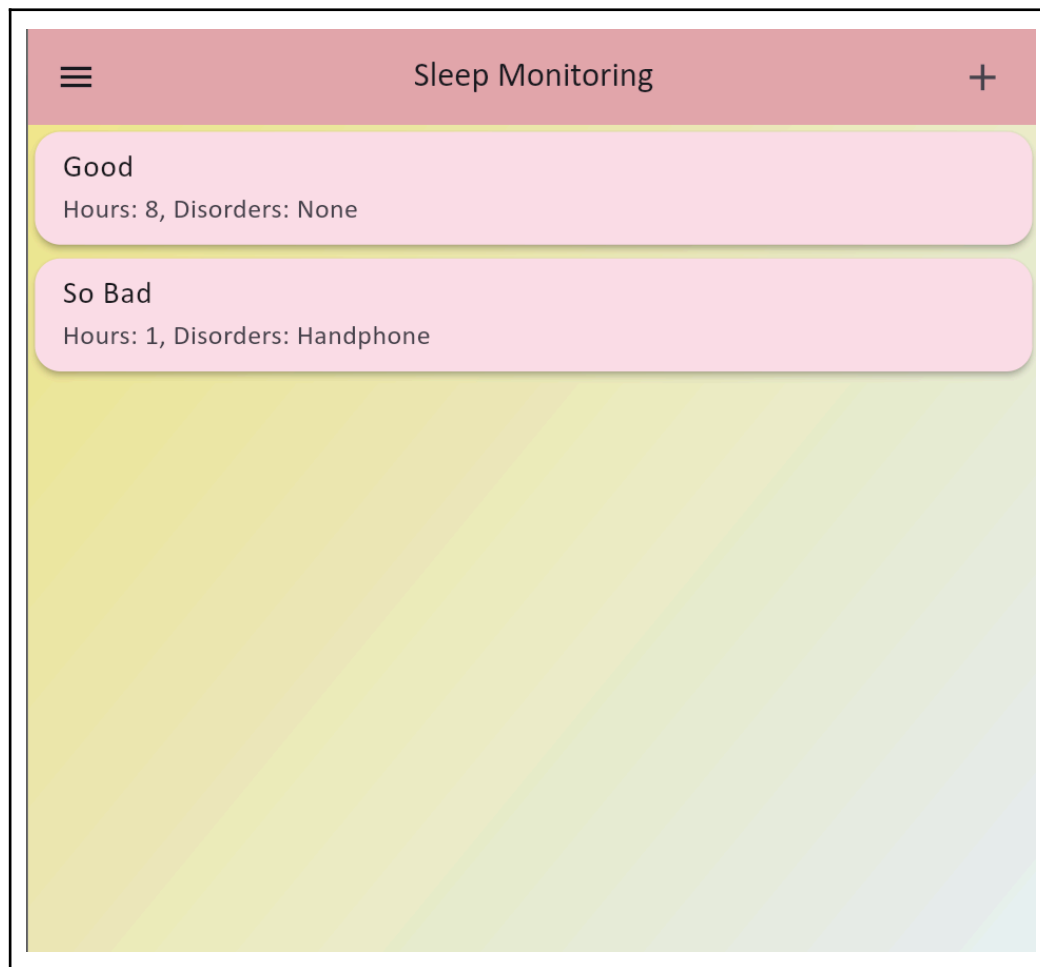
Jam Tidur

1

Gangguan Tidur

Handphone

SIMPAN



Bloc

1. Source Code Login_Bloc

```
import 'dart:convert';
import 'package:health/helpers/api.dart';
import 'package:health/helpers/api_url.dart';
import 'package:health/model/login.dart';

class LoginBloc {
  static Future<Login> login({String? email, String? password}) async {
    String apiUrl = ApiUrl.login;
    var body = {"email": email, "password": password};

    // Debugging: Cetak URL dan body
    print("URL: $apiUrl");
    print("Body: $body");
```

```

var response = await Api().post(apiUrl, body);

// Pastikan response status code
print("Response status code: ${response.statusCode}");

if (response.statusCode == 200) {
  var jsonObj = json.decode(response.body);
  return Login.fromJson(jsonObj);
} else {
  throw Exception("Failed to load login data");
}
}
}

```

Kode di atas adalah implementasi dari `LoginBloc`, sebuah kelas yang bertanggung jawab untuk melakukan proses login pengguna dalam aplikasi Flutter menggunakan API. Fungsi statis `login` menerima dua parameter opsional, yaitu `email` dan `password`, yang merupakan informasi yang diperlukan untuk proses autentikasi. Di dalam fungsi, URL API untuk login diambil dari `ApiUrl.login`, dan kemudian sebuah objek `body` dibuat dalam format JSON yang berisi email dan password. Selanjutnya, kode mencetak URL dan body ke konsol untuk keperluan debugging, sehingga pengembang dapat melihat apa yang dikirim ke server. Proses login dilakukan dengan memanggil metode `post` dari kelas `Api`, yang mengirimkan permintaan ke URL API dengan data `body`. Setelah menerima respons, kode mencetak status kode respons untuk memverifikasi apakah permintaan berhasil atau tidak. Jika status kode adalah 200, yang menandakan permintaan berhasil, maka respons JSON di-decode dan diubah menjadi objek `Login` menggunakan metode `fromJson`. Jika status kode bukan 200, maka fungsi akan melemparkan pengecualian dengan pesan "Failed to load login data", yang menunjukkan bahwa ada kesalahan dalam proses login.

2. Source Code Logout_Bloc

```

import 'package:health/helpers/user_info.dart';

class LogoutBloc {
  static Future logout() async {
    await UserInfo().logout();
  }
}

```

Kode di atas mendefinisikan kelas `LogoutBloc`, yang bertanggung jawab untuk mengelola proses logout pengguna dalam aplikasi Flutter. Kelas ini memiliki metode statis `logout`, yang melakukan logout dengan memanggil metode `logout` dari kelas `UserInfo`. Ketika `logout` dipanggil, metode ini akan mengeksekusi logika yang telah ditentukan dalam kelas `UserInfo`, seperti menghapus informasi pengguna yang tersimpan (misalnya, token autentikasi atau data sesi), sehingga pengguna dianggap telah keluar dari aplikasi. Kode ini memberikan cara yang terorganisir untuk menangani fungsi logout, memisahkan logika dari bagian antarmuka pengguna (UI) dan memberikan kemudahan dalam pengelolaan status autentikasi pengguna.

3. Source Code Pemantauan_Tidur_Bloc

```
import 'dart:convert';
import 'package:health/helpers/api.dart';
import 'package:health/helpers/api_url.dart';
import 'package:health/model/pemantauan_tidur.dart';

class PemantauanTidurBloc {
  // Mengambil data sleep monitoring
  static Future<List<PemantauanTidur>> getData() async {
    String apiUrl = ApiUrl.listPemantauanTidur; // URL untuk API list
    pemantauan tidur
    var response = await Api().get(apiUrl);

    if (response.statusCode == 200) {
      var jsonObj = json.decode(response.body);
      List<dynamic> listPemantauan = (jsonObj as Map<String,
dynamic>)['data'];
      return listPemantauan.map((data) =>
PemantauanTidur.fromJson(data)).toList();
    } else {
      throw Exception('Gagal memuat data: ${response.reasonPhrase}');
    }
  }

  // Menambah data sleep monitoring
  static Future<bool> addData({required PemantauanTidur
pemantauanTidur}) async {
    String apiUrl = ApiUrl.createPemantauanTidur; // URL untuk API create
    pemantauan tidur
```



```

var body = {
    "sleep_quality": pemantauanTidur.sleepQuality,
    "sleep_hours": pemantauanTidur.sleepHours.toString(),
    "sleep_disorders": pemantauanTidur.sleepDisorders,
};
var response = await Api().post(apiUrl, body);

if (response.statusCode == 200) {
    var jsonObj = json.decode(response.body);
    return jsonObj['status'] == true; // Kembalikan status boolean
} else {
    throw Exception('Gagal menambah data: ${response.reasonPhrase}');
}
}

// Mengupdate data sleep monitoring
static Future<bool> updateData({required PemantauanTidur
pemantauanTidur}) async {
    String apiUrl = ApiUrl.updatePemantauanTidur(pemantauanTidur.id!); //
URL untuk update
    var body = {
        "sleep_quality": pemantauanTidur.sleepQuality,
        "sleep_hours": pemantauanTidur.sleepHours.toString(),
        "sleep_disorders": pemantauanTidur.sleepDisorders,
    };
    var response = await Api().put(apiUrl, jsonEncode(body));

    if (response.statusCode == 200) {
        var jsonObj = json.decode(response.body);
        return jsonObj['status'] == true; // Kembalikan status boolean
    } else {
        throw Exception('Gagal mengupdate data: ${response.reasonPhrase}');
    }
}

// Menghapus data sleep monitoring
static Future<bool> deletePemantauanTidur({required int id}) async {
    String apiUrl = ApiUrl.deletePemantauanTidur(id); // URL untuk delete
pemantauan tidur
    var response = await Api().delete(apiUrl);

    if (response.statusCode == 200) {
        var jsonObj = json.decode(response.body);

```

```

        return jsonObj['status'] == true; // Kembalikan status boolean
    } else {
        throw Exception('Gagal menghapus data: ${response.reasonPhrase}');
    }
}

// Tambahkan data pemantauan tidur
static Future<bool> addPemantauanTidur({required PemantauanTidur
pemantauanTidur}) async {
    return await addData(pemantauanTidur: pemantauanTidur);
}

// Update data pemantauan tidur
static Future<bool> updatePemantauanTidur({required PemantauanTidur
pemantauanTidur}) async {
    return await updateData(pemantauanTidur: pemantauanTidur);
}
}

```

Kode di atas mendefinisikan kelas `PemantauanTidurBloc`, yang mengelola operasi CRUD (Create, Read, Update, Delete) untuk data pemantauan tidur dalam aplikasi Flutter. Kelas ini berisi metode statis untuk mengambil data (`getData`), menambah data (`addData`), memperbarui data (`updateData`), dan menghapus data (`deletePemantauanTidur`). Setiap metode melakukan panggilan HTTP ke API yang ditentukan oleh URL yang diambil dari kelas `ApiUrl`. Setelah mendapatkan respons, metode memeriksa kode status untuk memastikan permintaan berhasil. Jika berhasil, data yang diterima akan diproses dan dikembalikan; jika tidak, akan dilemparkan pengecualian dengan pesan kesalahan. Metode tambahan `addPemantauanTidur` dan `updatePemantauanTidur` disediakan untuk memudahkan penggunaan dan meningkatkan keterbacaan kode, mengarahkan pemanggilan langsung ke metode yang relevan.

4. Source Code Registrasi_Bloc

```

import 'dart:convert';
import 'package:health/helpers/api.dart';
import 'package:health/helpers/api_url.dart';
import 'package:health/model/registrasi.dart';

class RegistrasiBloc {
    static Future<Registrasi> registrasi(

```

```

    {String? nama, String? email, String? password}) async {
    String apiUrl = ApiUrl.registrasi;
    var body = {"nama": nama, "email": email, "password": password};
    var response = await Api().post(apiUrl, body);
    var jsonObj = json.decode(response.body);
    return Registrasi.fromJson(jsonObj);
  }
}

```

Kode di atas mendefinisikan kelas `RegistrasiBloc`, yang bertanggung jawab untuk menangani proses registrasi pengguna dalam aplikasi Flutter. Kelas ini memiliki metode statis `registrasi`, yang menerima parameter nama, email, dan password untuk membuat akun baru. Di dalam metode, URL untuk API registrasi diambil dari kelas `ApiUrl`, dan data yang diperlukan dikemas ke dalam format JSON. Metode ini kemudian melakukan permintaan HTTP POST ke API menggunakan kelas `Api`, dan setelah mendapatkan respons, data yang diterima dalam format JSON akan diurai menggunakan `json.decode()`. Akhirnya, respons tersebut dipetakan ke dalam objek `Registrasi` dengan memanggil metode `fromJson`, sehingga pengguna dapat dengan mudah mendapatkan informasi yang diperlukan dari hasil registrasi.

Model

1. Source Code Login

```

class Login {
  int? code;
  bool? status;
  String? token;
  int? userID;
  String? userEmail;
  Login({this.code, this.status, this.token, this.userID, this.userEmail});
  factory Login.fromJson(Map<String, dynamic> obj) {
    if (obj['code'] == 200) {
      return Login(
        code: obj['code'],
        status: obj['status'],
        token: obj['data']['token'],
        userID: int.parse(obj['data']['user']['id']),
        userEmail: obj['data']['user']['email']);
    } else {
      return Login(
        code: obj['code'],

```

```

        status: obj['status'],
    );
}
}
}

```

Kelas `Login` di atas merupakan model data yang digunakan untuk merepresentasikan hasil proses login pengguna dalam aplikasi Flutter. Kelas ini memiliki beberapa atribut opsional, seperti `code`, `status`, `token`, `userID`, dan `userEmail`, yang masing-masing berfungsi untuk menyimpan informasi terkait respons login. Metode `factory Login.fromJson` berfungsi untuk mengonversi data JSON yang diterima dari server menjadi objek `Login`. Jika kode status dalam respons adalah 200 (menandakan keberhasilan), maka objek `Login` akan diisi dengan informasi seperti token autentikasi dan ID serta email pengguna. Jika tidak, objek `Login` hanya akan diisi dengan kode dan status tanpa informasi tambahan. Ini memungkinkan aplikasi untuk dengan mudah mengelola dan memanfaatkan hasil dari proses login.

2. Source Code Pemantauan_Tidur

```

class PemantauanTidur {
  int? id;
  String? sleepQuality;
  int? sleepHours;
  String? sleepDisorders;

  PemantauanTidur({
    this.id,
    this.sleepQuality,
    this.sleepHours,
    this.sleepDisorders,
  });

  factory PemantauanTidur.fromJson(Map<String, dynamic> json) {
    return PemantauanTidur(
      id: json['id'],
      sleepQuality: json['sleep_quality'],
      sleepHours: json['sleep_hours'],
      sleepDisorders: json['sleep_disorders'],
    );
  }
}

```

```

Map<String, dynamic> toJson() {
  return {
    'id': id,
    'sleep_quality': sleepQuality,
    'sleep_hours': sleepHours,
    'sleep_disorders': sleepDisorders,
  };
}
}

```

Kelas `PemantauanTidur` merupakan model data yang digunakan untuk merepresentasikan informasi terkait pemantauan tidur dalam aplikasi Flutter. Kelas ini memiliki empat atribut opsional: `id` (tipe `int`), `sleepQuality` (tipe `String`), `sleepHours` (tipe `int`), dan `sleepDisorders` (tipe `String`). Atribut-atribut ini menyimpan informasi mengenai kualitas tidur, jam tidur, dan gangguan tidur yang dialami pengguna. Metode `factory PemantauanTidur.fromJson` digunakan untuk membuat objek `PemantauanTidur` dari data JSON yang diterima, mengonversi nilai-nilai dari kunci yang sesuai dalam JSON ke dalam atribut kelas. Sebaliknya, metode `toJson` berfungsi untuk mengonversi objek `PemantauanTidur` kembali menjadi format JSON, memudahkan proses pengiriman data ke server atau penyimpanan. Dengan cara ini, kelas ini memfasilitasi interoperabilitas antara aplikasi dan API, memastikan bahwa data terkait pemantauan tidur dapat dikelola dengan baik.

3. Source Code Registrasi

```

class Registrasi {
  int? code;
  bool? status;
  String? data;
  Registrasi({this.code, this.status, this.data});
  factory Registrasi.fromJson(Map<String, dynamic> obj) {
    return Registrasi(
      code: obj['code'], status: obj['status'], data: obj['data']);
  }
}

```

Kelas `Registrasi` adalah model data yang digunakan untuk merepresentasikan informasi yang berkaitan dengan proses registrasi

pengguna dalam aplikasi Flutter. Kelas ini memiliki tiga atribut opsional: ``code`` (tipe ``int``), ``status`` (tipe ``bool``), dan ``data`` (tipe ``String``). Atribut ``code`` biasanya digunakan untuk menyimpan kode status dari respons server, ``status`` menunjukkan apakah registrasi berhasil atau tidak, dan ``data`` menyimpan informasi tambahan yang mungkin dikirimkan oleh server sebagai hasil dari proses registrasi. Metode ``factory Registrasi.fromJson`` berfungsi untuk membuat objek ``Registrasi`` dari data dalam format JSON yang diterima. Metode ini mengambil peta (map) ``Map<String, dynamic>`` yang berisi data dari server dan mengonversinya menjadi objek ``Registrasi``, dengan mengisi atribut-atributnya sesuai dengan nilai yang terdapat dalam peta tersebut. Dengan cara ini, kelas ``Registrasi`` memungkinkan pengelolaan dan akses yang mudah terhadap data yang diterima dari proses registrasi, serta memudahkan integrasi dengan API yang digunakan dalam aplikasi.

Helpers

1. Source Code Api_Url

```
class ApiUrl {
  static const String baseUrl = 'http://103.196.155.42/api';

  // URL untuk Registrasi dan Login
  static const String registrasi = baseUrl + '/registrasi';
  static const String login = baseUrl + '/login';

  // URL untuk Pemantauan Tidur
  static const String listPemantauanTidur =
    baseUrl + '/kesehatan' + '/pemantauan_tidur';
  static const String createPemantauanTidur = baseUrl + '/kesehatan' +
    '/pemantauan_tidur';

  static String updatePemantauanTidur(int id) {
    return baseUrl + '/kesehatan' + '/pemantauan_tidur/' + id.toString() +
    '/update';
  }

  static String deletePemantauanTidur(int id) {
    return baseUrl + '/kesehatan' + '/pemantauan_tidur/' + id.toString() +
    '/delete';
  }

  static String showPemantauanTidur(int id) {
    return baseUrl + '/kesehatan' + '/pemantauan_tidur/' + id.toString();
  }
}
```

```
}  
}
```

Kelas `ApiUrl` adalah kelas yang berfungsi sebagai pengelola URL untuk melakukan komunikasi dengan API dalam aplikasi Flutter. Kelas ini mendefinisikan berbagai endpoint API yang digunakan untuk operasi seperti registrasi, login, dan pengelolaan data pemantauan tidur. Kelas ini memiliki satu atribut statis, yaitu `baseUrl`, yang menyimpan URL dasar API. Kemudian, kelas ini mendeklarasikan beberapa URL statis yang berkaitan dengan registrasi dan login, seperti `registrasi` dan `login`. Untuk pemantauan tidur, terdapat beberapa URL seperti `listPemantauanTidur`, `createPemantauanTidur`, dan fungsi-fungsi yang mengembalikan URL untuk operasi spesifik, seperti `updatePemantauanTidur`, `deletePemantauanTidur`, dan `showPemantauanTidur`, yang membutuhkan parameter `id`. Fungsi-fungsi ini memungkinkan pengembang untuk membangun URL yang diperlukan untuk melakukan permintaan HTTP ke server dengan mudah. Dengan cara ini, kelas `ApiUrl` memudahkan pengelolaan dan penggunaan endpoint API di seluruh aplikasi, serta menjaga agar kode tetap terorganisir dan mudah dipahami.

2. Source Code Api

```
import 'dart:io';  
import 'package:http/http.dart' as http;  
import 'package:health/helpers/user_info.dart';  
import 'app_exception.dart';  
  
class Api {  
  Future<dynamic> post(dynamic url, dynamic data) async {  
    var token = await UserInfo().getToken();  
    var responseJson;  
  
    try {  
      final response = await http.post(  
        Uri.parse(url),  
        body: data,  
        headers: {HttpHeaders.authorizationHeader: "Bearer $token"},  
      );  
      responseJson = _returnResponse(response);  
    } on SocketException {  
      throw FetchDataException('No Internet connection');  
    }  
  }  
}
```

```

    return responseJson;
}

Future<dynamic> get(dynamic url) async {
    var token = await UserInfo().getToken();
    var responseJson;

    try {
        final response = await http.get(
            Uri.parse(url),
            headers: {HttpHeaders.authorizationHeader: "Bearer $token"},
        );
        responseJson = _returnResponse(response);
    } on SocketException {
        throw FetchDataException('No Internet connection');
    }

    return responseJson;
}

Future<dynamic> put(dynamic url, dynamic data) async {
    var token = await UserInfo().getToken();
    var responseJson;

    try {
        final response = await http.put(
            Uri.parse(url),
            body: data,
            headers: {
                HttpHeaders.authorizationHeader: "Bearer $token",
                HttpHeaders.contentTypeHeader: "application/json",
            },
        );
        responseJson = _returnResponse(response);
    } on SocketException {
        throw FetchDataException('No Internet connection');
    }

    return responseJson;
}

Future<dynamic> delete(dynamic url) async {

```



```

var token = await UserInfo().getToken();
var responseJson;

try {
    final response = await http.delete(
        Uri.parse(url),
        headers: {HttpHeaders.authorizationHeader: "Bearer $token"},
    );
    responseJson = _returnResponse(response);
} on SocketException {
    throw FetchDataException('No Internet connection');
}

return responseJson;
}

dynamic _returnResponse(http.Response response) {
    switch (response.statusCode) {
        case 200:
            return response;
        case 400:
            throw BadRequestException(response.body.toString());
        case 401:
        case 403:
            throw UnauthorisedException(response.body.toString());
        case 422:
            throw InvalidInputException(response.body.toString());
        case 500:
        default:
            throw FetchDataException(
                'Error occurred while communicating with server. StatusCode:
                ${response.statusCode}');
    }
}

```

Kelas `Api` dalam kode ini berfungsi sebagai lapisan untuk melakukan permintaan HTTP (POST, GET, PUT, DELETE) ke API dengan pengelolaan token otorisasi untuk autentikasi. Setiap metode (post, get, put, delete) secara asinkron mengambil URL dan data (jika diperlukan), lalu menambahkan header otorisasi yang menyertakan token yang diambil dari kelas `UserInfo`. Jika permintaan berhasil, hasilnya akan diproses melalui metode `_returnResponse`, yang mengevaluasi status kode respons dan

mengembalikan respons atau melempar pengecualian yang sesuai untuk berbagai kondisi kesalahan, seperti kesalahan permintaan (400), otorisasi (401, 403), input tidak valid (422), atau kesalahan server (500). Kelas ini juga menangani pengecualian jaringan, seperti tidak adanya koneksi internet, dengan melempar `FetchDataException`. Dengan desain ini, kelas `Api` memastikan interaksi yang efisien dan terstruktur dengan server API, serta menangani kesalahan dengan baik.

3. Source Code App_Exception

```
class AppException implements Exception {
    final _message;
    final _prefix;
    AppException([this._message, this._prefix]);
    @override
    String toString() {
        return "$_prefix$_message";
    }
}

class FetchDataException extends AppException {
    FetchDataException([String? message])
        : super(message, "Error During Communication: ");
}

class BadRequestException extends AppException {
    BadRequestException([message]) : super(message, "Invalid Request: ");
}

class UnauthorisedException extends AppException {
    UnauthorisedException([message]) : super(message, "Unauthorised: ");
}

class UnprocessableEntityException extends AppException {
    UnprocessableEntityException([message])
        : super(message, "Unprocessable Entity: ");
}

class InvalidInputException extends AppException {
    InvalidInputException([String? message]) : super(message, "Invalid Input: ");
}
```

Kelas `AppException` dan kelas-kelas turunannya dalam kode ini digunakan untuk menangani pengecualian kustom yang berkaitan dengan komunikasi dengan API. `AppException` adalah kelas dasar yang mengimplementasikan antarmuka `Exception`, dengan dua atribut: `_message` untuk menyimpan pesan kesalahan dan `_prefix` untuk menyimpan awalan yang memberikan konteks pada jenis kesalahan. Kelas-kelas yang diturunkan dari `AppException`, seperti `FetchDataException`, `BadRequestException`, `UnauthorisedException`, `UnprocessableEntityException`, dan `InvalidInputException`, masing-masing menyajikan jenis kesalahan spesifik yang dapat terjadi selama interaksi dengan API. Setiap kelas turunan memanggil konstruktor `AppException` dengan pesan yang sesuai dan awalan yang mendeskripsikan jenis kesalahan. Dengan desain ini, kode menyediakan cara yang terstruktur untuk menangani berbagai situasi kesalahan, memungkinkan pengembang untuk menangkap dan merespons kesalahan dengan lebih jelas dan informatif.

4. Source Code User_Info

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

class UserInfo {
  // Set the user token
  Future<void> setToken(String token) async {
    final prefs = await SharedPreferences.getInstance();
    await prefs.setString('user_token', token);
  }

  // Get the user token
  Future<String?> getToken() async {
    final prefs = await SharedPreferences.getInstance();
    return prefs.getString('user_token');
  }

  // Set the user ID
  Future<void> setUserID(int userID) async {
    final prefs = await SharedPreferences.getInstance();
    await prefs.setInt('user_id', userID);
  }

  // Get the user ID
  Future<int?> getUserID() async {
    final prefs = await SharedPreferences.getInstance();
```

```

    return prefs.getInt('user_id');
  }

  // Logout method
  Future<void> logout() async {
    final prefs = await SharedPreferences.getInstance();
    await prefs.remove('user_token');
    await prefs.remove('user_id');
  }

  // Clear user data (could be a general method for any additional cleanup)
  Future<void> clearUserData() async {
    final prefs = await SharedPreferences.getInstance();
    await prefs.remove('user_token');
    await prefs.remove('user_id');
  }
}

```

Kelas `UserInfo` berfungsi untuk mengelola penyimpanan dan pengambilan informasi pengguna, seperti token dan ID pengguna, menggunakan paket `shared_preferences` di Flutter. Kelas ini menyediakan metode untuk menyimpan (`setToken`, `setUserID`) dan mengambil (`getToken`, `getUserID`) data pengguna yang disimpan secara lokal. Selain itu, terdapat metode `logout` yang menghapus token dan ID pengguna saat pengguna keluar dari aplikasi, serta metode `clearUserData` yang juga dapat digunakan untuk membersihkan data pengguna tambahan jika diperlukan. Dengan pendekatan ini, `UserInfo` memastikan bahwa informasi sensitif pengguna dikelola dengan cara yang terstruktur dan efisien, mendukung pengalaman pengguna yang lebih baik dan keamanan data.

Main

1. Source Code Main

```

import 'package:flutter/material.dart';
import 'package:health/helpers/user_info.dart';
import 'package:health/ui/login_page.dart';
import 'package:health/ui/pemantauan_page.dart';

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

```

```

class MyApp extends StatefulWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  Widget page = const CircularProgressIndicator();
  @override
  void initState() {
    super.initState();
    isLogin();
  }

  void isLogin() async {
    var token = await UserInfo().getToken();
    if (token != null) {
      setState(() {
        page = const PemantauanTidurPage();
      });
    } else {
      setState(() {
        page = const LoginPage();
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Toko Kita',
      debugShowCheckedModeBanner: false,
      home: page,
    );
  }
}

```

Kode di atas merupakan implementasi dasar dari aplikasi Flutter yang memutuskan tampilan awal berdasarkan status login pengguna. Kelas `MyApp`, yang merupakan turunan dari `StatefulWidget`, menginisialisasi aplikasi dengan tampilan `CircularProgressIndicator` saat memuat. Pada

metode `initState`, fungsi `isLogin` dipanggil untuk memeriksa apakah pengguna sudah login dengan mengambil token pengguna dari `UserInfo`. Jika token tidak null, aplikasi akan menampilkan halaman pemantauan tidur (`PemantauanTidurPage`), sedangkan jika token null, halaman login (`LoginPage`) akan ditampilkan. Di dalam metode `build`, aplikasi menggunakan `MaterialApp` dengan judul "Toko Kita" dan menyembunyikan banner debug, menetapkan `home` dengan halaman yang ditentukan sebelumnya. Dengan cara ini, aplikasi memberikan pengalaman pengguna yang sesuai berdasarkan status autentikasi.