Jasmine Adzra Fakhirah

H1D022071

Responsi UTS Prak Pemob

Shift D

Pariwisata - Tabel Penginapan

Hasil Modulo NIM : 3

3 digit terakhir : 071

Segment Satu

1. Paket: 3
2. Jenis Aplikasi: Aplikasi Manajemen Pariwisata

Segment Dua

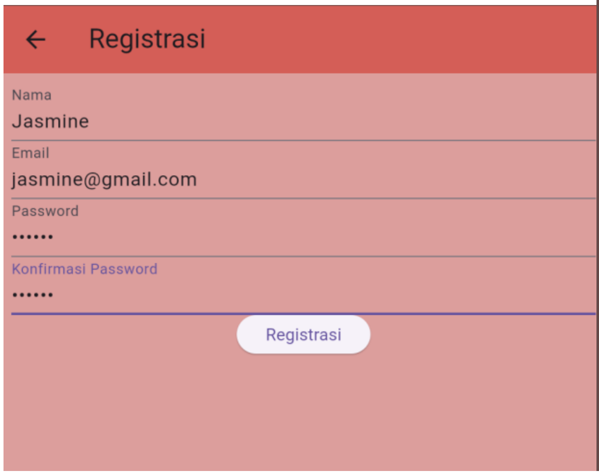
1. Digit Puluhan: 7
2. Nama Tabel: penginapan
3. Kolom 1: id (int, PK, increment)
4. Kolom 2: accommodation (String)
5. Kolom 3: room (String)
6. Kolom 4: rate (Integer)

Segment Tiga

1. Digit Satuan: 1
2. Kustomisasi Tampilan UI: Tema Terang Merah, Font Arial

Penjelasan Apps

Registrasi

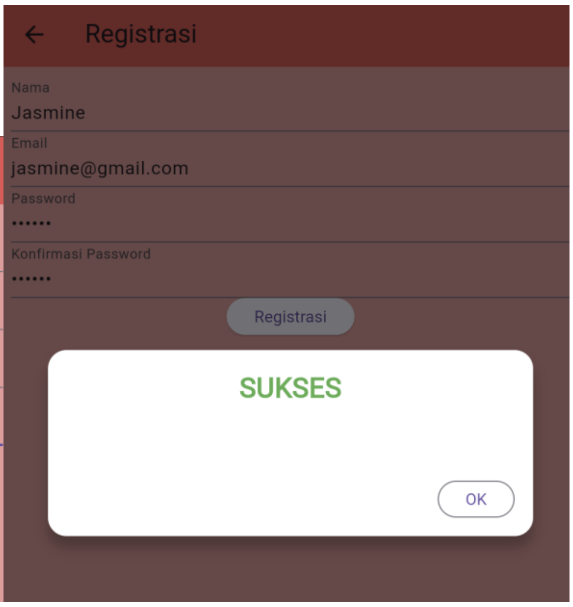

Registrasi

Nama
Jasmine

Email
jasmine@gmail.com

Password
••••••

Konfirmasi Password
••••••


Registrasi

Nama
Jasmine

Email
jasmine@gmail.com

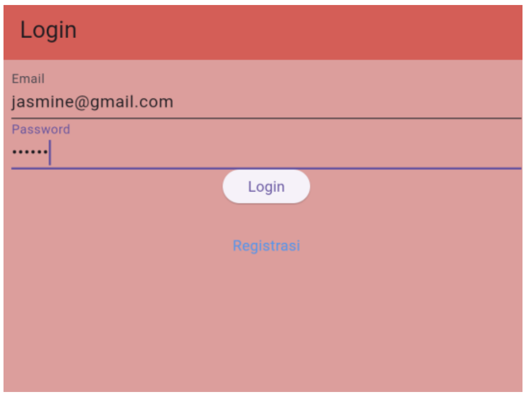
Password
••••••

Konfirmasi Password
••••••

SUKSES

Proses registrasi dimulai dengan pengguna mengisi formulir yang berisi kolom untuk nama, email, password, dan konfirmasi password. Ketika tombol "Registrasi" diklik, formulir akan divalidasi untuk memastikan semua kolom diisi dengan benar—nama harus minimal 3 karakter, email harus valid, password harus minimal 6 karakter, dan konfirmasi password harus sama dengan password. Jika semua validasi berhasil, permintaan registrasi dikirim ke backend menggunakan fungsi RegistrasiBloc.registrasi, yang melakukan panggilan API untuk mengirimkan data. Jika responsnya berhasil, dialog sukses akan muncul yang memberi tahu pengguna bahwa registrasi berhasil dan memintanya untuk login. Jika terjadi kesalahan, dialog peringatan akan muncul, menunjukkan bahwa registrasi gagal.

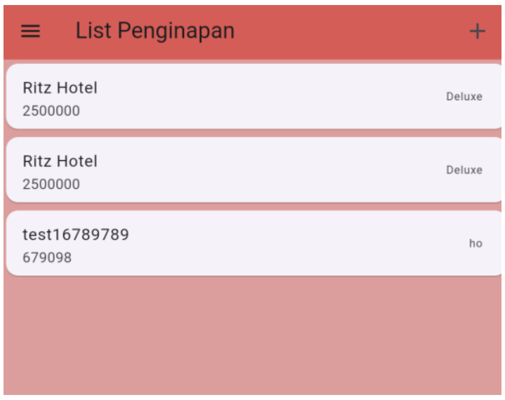
Login



Proses login dimulai ketika pengguna mengisi email dan password di dalam form yang telah disediakan pada halaman login. Setelah pengguna menekan tombol "Login", aplikasi akan memvalidasi input, memastikan bahwa email dan password tidak kosong. Jika validasi berhasil, data yang diisi pengguna dikirim ke backend melalui fungsi LoginBloc.login, yang melakukan panggilan API untuk otentikasi.

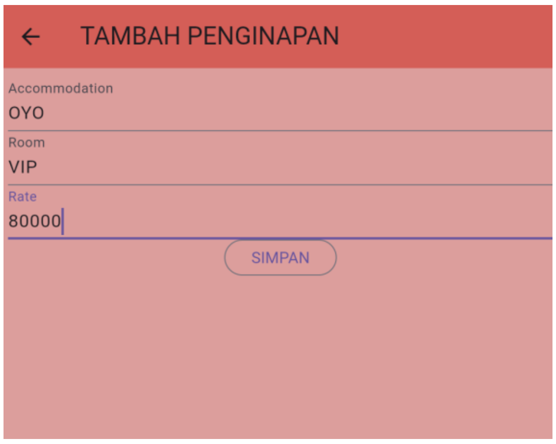
Jika login berhasil (kode status API adalah 200), aplikasi akan menyimpan token pengguna dan ID pengguna di penyimpanan lokal menggunakan UserInfo().setToken dan UserInfo().setUserID. Selanjutnya, pengguna akan diarahkan ke halaman PenginapanPage sebagai tanda bahwa login berhasil. Jika login gagal, aplikasi akan menampilkan dialog peringatan yang memberitahukan bahwa login tidak berhasil dan meminta pengguna untuk mencoba lagi. Untuk pengguna yang belum registrasi dapat menekan tombol registrasi yang ada di bawah tombol login.

View



Proses view pada halaman PenginapanPage dimulai dengan menampilkan daftar penginapan menggunakan FutureBuilder yang memanggil data dari PenginapanBloc.getPenginapan. Jika data berhasil diambil, daftar penginapan ditampilkan dalam bentuk ListView. Setiap item penginapan ditampilkan menggunakan widget ItemPenginapan, yang menampilkan nama penginapan, harga (rate), dan jumlah kamar (room). Saat pengguna menekan salah satu item penginapan, mereka diarahkan ke halaman detail (PenginapanDetail) melalui navigasi Navigator.push. Selain itu, pengguna dapat menambahkan penginapan baru melalui tombol "+" di AppBar, yang akan membuka form penambahan penginapan.

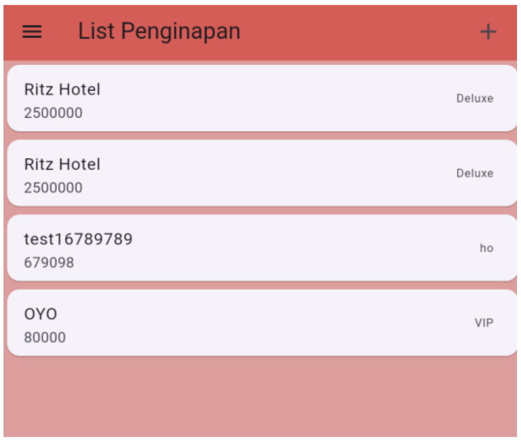
Create


TAMBAH PENGINAPAN

Accommodation
OYO

Room
VIP

Rate
80000


List Penginapan

Ritz Hotel
2500000
Deluxe

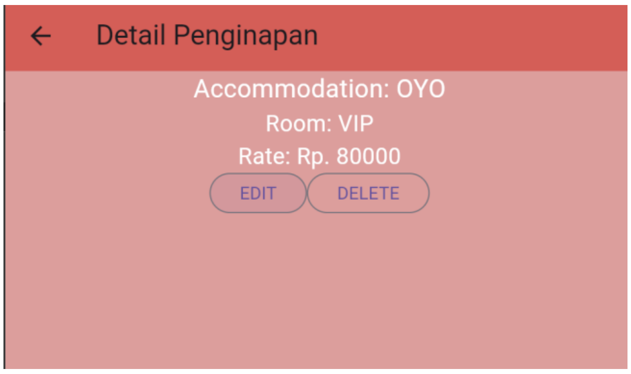
Ritz Hotel
2500000
Deluxe

test16789789
679098
ho

OYO
80000
VIP

Proses create penginapan dimulai ketika pengguna menekan tombol "+" di halaman PenginapanPage, yang akan membawa mereka ke halaman PenginapanForm melalui Navigator.push. Di halaman ini, pengguna mengisi form dengan detail penginapan baru, seperti accomodation, room, dan rate. Setelah form diisi dan disubmit, data tersebut akan dikirim ke API menggunakan fungsi createPenginapan yang ada di PenginapanBloc. Jika proses berhasil, penginapan baru akan ditambahkan ke database, dan pengguna akan diarahkan kembali ke halaman daftar penginapan untuk melihat data terbaru yang telah ditambahkan.

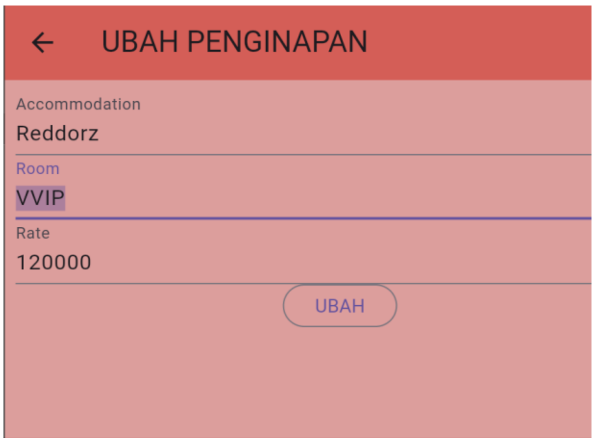
Detail


Detail Penginapan

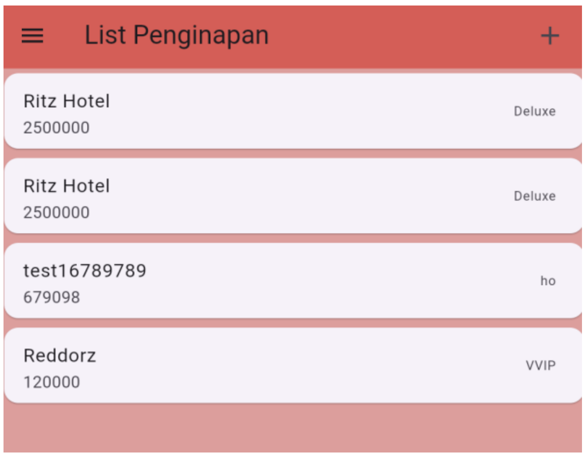
Accommodation: OYO
Room: VIP
Rate: Rp. 80000

Proses **detail view** penginapan dimulai ketika pengguna memilih salah satu penginapan dari daftar di halaman utama. Pengguna akan diarahkan ke halaman **PenginapanDetail**, yang menampilkan informasi lengkap dari penginapan yang dipilih, seperti nama penginapan (accommodation), jumlah kamar (room), dan harga (rate). Di halaman ini, terdapat dua tombol: **Edit** dan **Delete**. Tombol **Edit** memungkinkan pengguna untuk memperbarui data penginapan, sementara tombol **Delete** akan meminta konfirmasi sebelum menghapus penginapan tersebut dari database melalui API. Setelah penghapusan berhasil, pengguna akan diarahkan kembali ke halaman daftar penginapan.

Edit



Accommodation
Reddorz
Room
VVIP
Rate
120000



Ritz Hotel
2500000
Deluxe

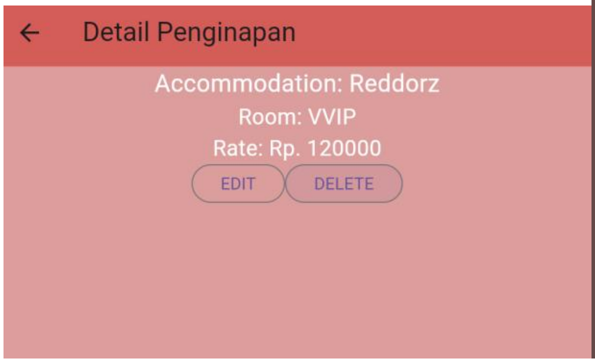
Ritz Hotel
2500000
Deluxe

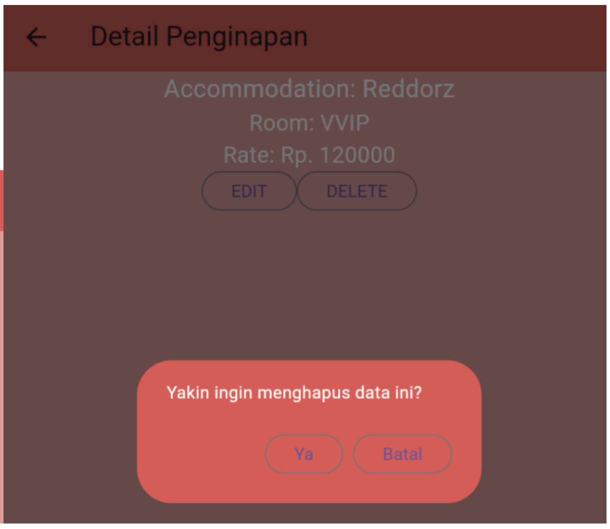
test16789789
679098
ho

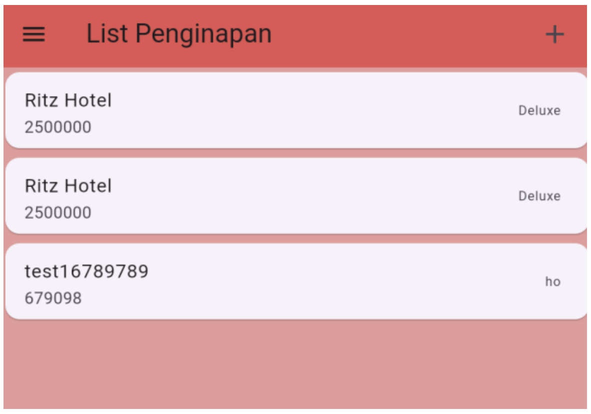
Reddorz
120000
VVIP

Proses **edit** dimulai ketika pengguna menekan tombol **Edit** pada halaman detail penginapan. Pengguna akan diarahkan ke halaman **PenginapanForm**, di mana form isian sudah terisi dengan data penginapan yang ingin diedit. Pengguna bisa mengubah informasi seperti nama penginapan (accommodation), jumlah kamar (room), atau harga (rate). Setelah perubahan dilakukan, pengguna menekan tombol **Simpan** untuk mengirim data yang telah diperbarui ke API menggunakan metode yang sesuai. Setelah pembaruan berhasil, pengguna akan diarahkan kembali ke halaman daftar penginapan dengan data yang telah diperbarui.

Delete

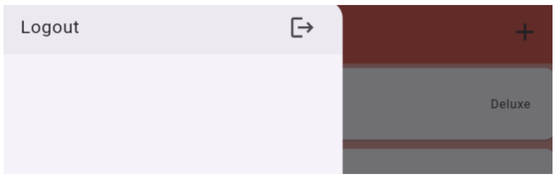


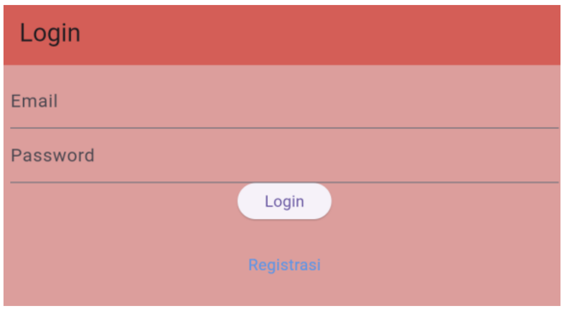




Proses **delete** dimulai ketika pengguna menekan tombol **Delete** pada halaman detail penginapan. Hal ini akan memunculkan dialog konfirmasi yang menanyakan apakah pengguna yakin ingin menghapus data tersebut. Jika pengguna memilih **Ya**, maka sistem akan memanggil metode untuk menghapus penginapan tersebut melalui API dengan ID penginapan yang relevan. Setelah penghapusan berhasil, pengguna akan diarahkan kembali ke halaman daftar penginapan, dan daftar tersebut akan diperbarui untuk mencerminkan bahwa penginapan yang dihapus tidak lagi ditampilkan. Jika terjadi kesalahan saat penghapusan, dialog peringatan akan muncul, memberi tahu pengguna bahwa proses penghapusan gagal.

Logout





Proses **logout** dimulai ketika pengguna memilih opsi **Logout** dari sidebar di halaman penginapan. Setelah itu, sistem akan memanggil metode **logout** yang terdapat dalam **LogoutBloc**. Metode ini akan menghapus informasi pengguna yang tersimpan, seperti token otentikasi dan ID pengguna, melalui metode **logout** yang ada di kelas **UserInfo**. Setelah proses logout selesai, pengguna akan diarahkan kembali ke halaman login, sehingga mereka harus memasukkan kredensial mereka lagi untuk mengakses aplikasi. Hal ini memastikan bahwa sesi pengguna diakhiri dengan aman.

Kode

[main.dart]

```
import 'package:flutter/material.dart';
import '/helpers/user_info.dart';
import '/ui/login_page.dart';
import '/ui/penginapan_page.dart'; // Updated to use penginapan_page

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 PenginapanPage(); // Updated to use PenginapanPage
   });
  } else {
   setState(() {
    page = const LoginPage();
   });
  }
 }

 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   title: 'Aplikasi Manajemen Pariwisata',
   debugShowCheckedModeBanner: false,
   theme: ThemeData(
    fontFamily: 'Arial', // Set the font family to Arial
    primarySwatch: Colors.red, // Primary color
    brightness: Brightness.light, // Set brightness to light
    appBarTheme: AppBarTheme(
     color: Colors.red[400], // Light red color for the app bar
    ),
    scaffoldBackgroundColor: Colors.red[200],
    textTheme: TextTheme(
     bodyMedium: TextStyle(color: Colors.white), // Default text color for body
     bodySmall: TextStyle(color: Colors.white), // Default text color for small body text
    ),
   ),
   home: page, // Use the login state to determine the home page
  );
 }
}
```

[ui/login_page.dart]

```
import 'package:flutter/material.dart';
import 'package:responsi_uts_pemob/bloc/login_bloc.dart';
import 'package:responsi_uts_pemob/helpers/user_info.dart';
import 'package:responsi_uts_pemob/ui/penginapan_page.dart';
import 'package:responsi_uts_pemob/ui/registrasi_page.dart';
import 'package:responsi_uts_pemob/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'),
    ),
    body: SingleChildScrollView(
     child: Padding(
      padding: const EdgeInsets.all(8.0),
      child: Form(
       key: _formKey,
       child: Column(
        children: [
         _emailTextField(),
         _passwordTextField(),
         _buttonLogin(),
         const SizedBox(
          height: 30,
         ),
         _menuRegistrasi()
        ],
       ),
      ),
     ),
    ),
   );
  }
//Membuat Textbox email
 Widget _emailTextField() {
  return TextFormField(
   decoration: const InputDecoration(labelText: "Email"),
   keyboardType: TextInputType.emailAddress,
   controller: _emailTextboxController,
   validator: (value) {
//validasi harus diisi
    if (value!.isEmpty) {
     return 'Email harus diisi';
    }
    return null;
   },
  );
 }
//Membuat Textbox password
 Widget _passwordTextField() {
  return TextFormField(
   decoration: const InputDecoration(labelText: "Password"),
   keyboardType: TextInputType.text,
   obscureText: true,
   controller: _passwordTextboxController,
   validator: (value) {
//jika karakter yang dimasukkan kurang dari 6 karakter
    if (value!.isEmpty) {
     return "Password harus diisi";
    }
    return null;
   },
   );
  }
//Membuat Tombol Login
//Membuat Tombol Login
  Widget _buttonLogin() {
    return ElevatedButton(
        child: const Text("Login"),
        onPressed: () {
          var validate = _formKey.currentState!.validate();
          if (validate) {
            if (!_isLoading) _submit();
          }
        });
  }
  void _submit() {
    _formKey.currentState!.save();
    setState(() {
      _isLoading = true;
    });
    LoginBloc.login(
        email: _emailTextboxController.text,
        password: _passwordTextboxController.text)
        .then((value) async {
      if (value.code == 200) {
        await UserInfo().setToken(value.token.toString());
        await UserInfo().setUserID(int.parse(value.userID.toString()));
        Navigator.pushReplacement(context,
            MaterialPageRoute(builder: (context) => const PenginapanPage()));
      } else {
        showDialog(
            context: context,
            barrierDismissible: false,
            builder: (BuildContext context) => const WarningDialog(
              description: "Login gagal, silahkan coba lagi",
            ));
      }
    }, onError: (error) {
      print(error);
      showDialog(
          context: context,
          barrierDismissible: false,
          builder: (BuildContext context) => const WarningDialog(
            description: "Login gagal, silahkan coba lagi",
          ));
    });
    setState(() {
      _isLoading = false;
   });
  }
// Membuat menu untuk membuka halaman registrasi
  Widget _menuRegistrasi() {
   return Center(
      child: InkWell(
      child: const Text(
      "Registrasi",
       style: TextStyle(color: Colors.blue),
      ),
       onTap: () {
        Navigator.push(context,
            MaterialPageRoute(builder: (context) => const RegistrasiPage()));
       },
      ),
   );
  }
}
```

[ui/registrasi_page.dart]

```
import 'package:flutter/material.dart';
import 'package:responsi_uts_pemob/bloc/registrasi_bloc.dart';
import 'package:responsi_uts_pemob/widget/success_dialog.dart';
import 'package:responsi_uts_pemob/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"),
     ),
    body: SingleChildScrollView(
      child: Padding(
        padding: const EdgeInsets.all(8.0),
        child: Form(
          key: _formKey,
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              _namaTextField(),
              _emailTextField(),
              _passwordTextField(),
              _passwordKonfirmasiTextField(),
              _buttonRegistrasi(),
            ],
          ),
        ),
      ),
    ),
  );
}

// Membuat Textbox Nama
Widget _namaTextField() {
  return TextFormField(
    decoration: const InputDecoration(labelText: "Nama"),
    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"),
    keyboardType: TextInputType.emailAddress,
    controller: _emailTextboxController,
    validator: (value) {
      // validasi harus diisi
      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"),
    keyboardType: TextInputType.text,
    obscureText: true,
    controller: _passwordTextboxController,
    validator: (value) {
      // jika karakter yang dimasukkan kurang dari 6 karakter
      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"),
    keyboardType: TextInputType.text,
    obscureText: true,
    validator: (value) {
      // jika inputan tidak sama dengan password
      if (value != _passwordTextboxController.text) {
        return "Konfirmasi Password tidak sama";
      }
      return null;
    },
  );
}

// Membuat Tombol Registrasi
```

```
  Widget _buttonRegistrasi() {
    return ElevatedButton(
      child: const Text("Registrasi"),
      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",
          okClick: () {
            Navigator.pop(context);
          },
        ),
      );
    }, onError: (error) {
      showDialog(
        context: context,
        barrierDismissible: false,
        builder: (BuildContext context) => const WarningDialog(
          description: "Registrasi gagal, silahkan coba lagi",
        ),
      );
    });
    setState(() {
      _isLoading = false;
    });
  }
}
```

[ui/penginapan_page.dart]

```
import 'package:flutter/material.dart';
import 'package:responsi_uts_pemob/bloc/logout_bloc.dart';
import 'package:responsi_uts_pemob/bloc/penginapan_bloc.dart';
import 'package:responsi_uts_pemob/model/penginapan.dart';
import 'package:responsi_uts_pemob/ui/login_page.dart';
import 'package:responsi_uts_pemob/ui/detail_penginapan.dart';
import 'package:responsi_uts_pemob/ui/penginapan_form.dart';

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

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

class _PenginapanPageState extends State<PenginapanPage> {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: const Text('List Penginapan'),
    actions: [
     Padding(
      padding: const EdgeInsets.only(right: 20.0),
      child: GestureDetector(
       child: const Icon(Icons.add, size: 26.0),
       onTap: () async {
        Navigator.push(
         context,
         MaterialPageRoute(builder: (context) => PenginapanForm()), // Removed
'const'
        );
       },
      ),
     ),
    ],
   ),
   drawer: Drawer(
    child: ListView(
     children: [
      ListTile(
       title: const Text('Logout'),
       trailing: const Icon(Icons.logout),
       onTap: () async {
          await LogoutBloc.logout().then((value) {
            Navigator.of(context).pushAndRemoveUntil(
              MaterialPageRoute(builder: (context) => const LoginPage()),
                  (route) => false,
            );
          });
        },
      ),
    ],
  ),
),
body: FutureBuilder<List<Penginapan>>(
  future: PenginapanBloc.getPenginapan(),
  builder: (context, snapshot) {
    if (snapshot.hasError) {
      print(snapshot.error);
    }
    return snapshot.hasData
        ? ListPenginapan(list: snapshot.data)
        : const Center(child: CircularProgressIndicator());
  },
),
    );
  }
}

class ListPenginapan extends StatelessWidget {
  final List<Penginapan>? list;

  const ListPenginapan({Key? key, this.list}) : super(key: key);

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

class ItemPenginapan extends StatelessWidget {
  final Penginapan penginapan;

  const ItemPenginapan({Key? key, required this.penginapan}) : super(key: key);
```

```
 @override
 Widget build(BuildContext context) {
  return GestureDetector(
   onTap: () {
    Navigator.push(
     context,
     MaterialPageRoute(
      builder: (context) => PenginapanDetail(penginapan: penginapan),
     ),
    );
   },
   child: Card(
    child: ListTile(
     title: Text(penginapan.accommodation!), // Corrected property
     subtitle: Text(penginapan.rate.toString()), // Corrected property
     trailing: Text(penginapan.room!), // Corrected property
    ),
   ),
  );
 }
}
```

[ui/penginapan_form.dart]

```
import 'package:flutter/material.dart';
import 'package:responsi_uts_pemob/bloc/penginapan_bloc.dart'; // Ensure you
have this import
import 'package:responsi_uts_pemob/model/penginapan.dart';
import 'package:responsi_uts_pemob/ui/penginapan_page.dart';
import 'package:responsi_uts_pemob/widget/warning_dialog.dart'; // Import your
warning dialog

// ignore: must_be_immutable
class PenginapanForm extends StatefulWidget {
 Penginapan? penginapan;

 PenginapanForm({Key? key, this.penginapan}) : super(key: key);

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

class _PenginapanFormState extends State<PenginapanForm> {
 final _formKey = GlobalKey<FormState>();
 bool _isLoading = false;
 String judul = "TAMBAH PENGINAPAN";
String tombolSubmit = "SIMPAN";

final _accommodationTextboxController = TextEditingController();
final _roomTextboxController = TextEditingController();
final _rateTextboxController = TextEditingController();

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

isUpdate() {
  if (widget.penginapan != null) {
    setState(() {
      judul = "UBAH PENGINAPAN";
      tombolSubmit = "UBAH";
      _accommodationTextboxController.text = widget.penginapan!.accommodation!;
      _roomTextboxController.text = widget.penginapan!.room!;
      _rateTextboxController.text = widget.penginapan!.rate.toString();
    });
  } else {
    judul = "TAMBAH PENGINAPAN";
    tombolSubmit = "SIMPAN";
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text(judul)),
    body: SingleChildScrollView(
      child: Padding(
        padding: const EdgeInsets.all(8.0),
        child: Form(
          key: _formKey,
          child: Column(
            children: [
              _accommodationTextField(),
              _roomTextField(),
              _rateTextField(),
              _buttonSubmit()
            ],
          ),
        ),
      ),
    ),
  );
}

// Creating Textbox for Accommodation
Widget _accommodationTextField() {
  return TextFormField(
    decoration: const InputDecoration(labelText: "Accommodation"),
    keyboardType: TextInputType.text,
    controller: _accommodationTextboxController,
    validator: (value) {
      if (value!.isEmpty) {
        return "Accommodation harus diisi";
      }
      return null;
    },
  );
}

// Creating Textbox for Room
Widget _roomTextField() {
  return TextFormField(
    decoration: const InputDecoration(labelText: "Room"),
    keyboardType: TextInputType.text,
    controller: _roomTextboxController,
    validator: (value) {
      if (value!.isEmpty) {
        return "Room harus diisi";
      }
      return null;
    },
  );
}

// Creating Textbox for Rate
Widget _rateTextField() {
  return TextFormField(
    decoration: const InputDecoration(labelText: "Rate"),
    keyboardType: TextInputType.number,
    controller: _rateTextboxController,
    validator: (value) {
      if (value!.isEmpty) {
        return "Rate harus diisi";
      }
      return null;
    },
  );
}
```

```
// Creating Save/Update Button
Widget _buttonSubmit() {
 return OutlinedButton(
  child: Text(tombolSubmit),
  onPressed: () {
   var validate = _formKey.currentState!.validate();
   if (validate) {
    if (!_isLoading) {
     if (widget.penginapan != null) {
      // Update penginapan logic
      ubah();
     } else {
      // Add new penginapan logic
      simpan();
     }
    }
   }
  },
 );
}

// Method for saving new penginapan
simpan() {
 setState(() {
  _isLoading = true;
 });

 Penginapan createPenginapan = Penginapan(id: null);
 createPenginapan.accommodation = _accommodationTextboxController.text;
 createPenginapan.room = _roomTextboxController.text;
 createPenginapan.rate = int.parse(_rateTextboxController.text);

 PenginapanBloc.addPenginapan(penginapan: createPenginapan).then((value) {
  Navigator.of(context).push(MaterialPageRoute(
    builder: (BuildContext context) => const PenginapanPage()));
 }, onError: (error) {
  showDialog(
    context: context,
    builder: (BuildContext context) => const WarningDialog(
     description: "Simpan gagal, silahkan coba lagi",
    ));
 });

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

 // Method for updating existing penginapan
 ubah() {
  setState(() {
   _isLoading = true;
  });

  // Create an instance of Penginapan for updating
  Penginapan updatePenginapan = Penginapan(id: widget.penginapan!.id!);
  updatePenginapan.accommodation = _accommodationTextboxController.text;
  updatePenginapan.room = _roomTextboxController.text;
  updatePenginapan.rate = int.parse(_rateTextboxController.text); // Ensure this is
safely parsed

  // Call the update function in your Bloc
  PenginapanBloc.updatePenginapan(penginapan: updatePenginapan).then((value) {
   Navigator.of(context).push(MaterialPageRoute(
    builder: (BuildContext context) => const PenginapanPage(), // Replace with your
destination page
   ));
  }, onError: (error) {
   showDialog(
     context: context,
     builder: (BuildContext context) => const WarningDialog(
      description: "Permintaan ubah data gagal, silahkan coba lagi",
     ));
  });

  setState(() {
   _isLoading = false;
  });
 }
}
```

[ui/detail_penginapan.dart]

```
import 'package:flutter/material.dart';
import '../bloc/penginapan_bloc.dart'; // Import your bloc for managing penginapan
import '../widget/warning_dialog.dart'; // Import your warning dialog
import '/model/penginapan.dart';
import '/ui/penginapan_form.dart';
import 'penginapan_page.dart'; // Import your main penginapan page

// ignore: must_be_immutable
class PenginapanDetail extends StatefulWidget {
  Penginapan? penginapan;
  PenginapanDetail({Key? key, this.penginapan}) : super(key: key);

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

class _PenginapanDetailState extends State<PenginapanDetail> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Detail Penginapan'),
      ),
      body: Center(
        child: Column(
          children: [
            Text(
              "Accommodation: ${widget.penginapan!.accommodation}",
              style: const TextStyle(fontSize: 20.0),
            ),
            Text(
              "Room: ${widget.penginapan!.room}",
              style: const TextStyle(fontSize: 18.0),
            ),
            Text(
              "Rate: Rp. ${widget.penginapan!.rate}",
              style: const TextStyle(fontSize: 18.0),
            ),
            _tombolHapusEdit(),
          ],
        ),
      ),
    );
  }

  Widget _tombolHapusEdit() {
    return Row(
      mainAxisSize: MainAxisSize.min,
      children: [
        // Tombol Edit
        OutlinedButton(
          child: const Text("EDIT"),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => PenginapanForm(
                  penginapan: widget.penginapan!,
                ),
              ),
            );
          },
        ),
        // Tombol Hapus
        OutlinedButton(
          child: const Text("DELETE"),
          onPressed: () => confirmHapus(),
        ),
      ],
    );
  }

  void confirmHapus() {
    AlertDialog alertDialog = AlertDialog(
      backgroundColor: Colors.red[400],
      content: const Text("Yakin ingin menghapus data ini?"),
      actions: [
        // tombol hapus
        OutlinedButton(
          child: const Text("Ya"),
          onPressed: () {
            PenginapanBloc.deletePenginapan(id: widget.penginapan!.id!).then(
                (value) {
              Navigator.of(context).pushReplacement(
                MaterialPageRoute(builder: (context) => const PenginapanPage()),
              );
              },
              onError: (error) {
              showDialog(
                context: context,
                builder: (BuildContext context) => const WarningDialog(
                  description: "Hapus gagal, silahkan coba lagi",
                ),
              );
              },
            );
          },
        ),
        // tombol batal
        OutlinedButton(
          child: const Text("Batal"),
          onPressed: () => Navigator.pop(context),
        ),
    ],
   );
   showDialog(builder: (context) => alertDialog, context: context);
  }
}
```

[model/login.dart]

```
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: obj['data']['user']['id'],
          userEmail: obj['data']['user']['email']);
    } else {
      return Login(
        code: obj['code'],
        status: obj['status'],
      );
    }
  }
}
```

[model/registrasi.dart]

```
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']);
  }
}
```

[model/penginapan.dart]

```
class Penginapan {
  int? id;
  String? accommodation;
  String? room;
  int? rate;

  Penginapan({this.id, this.accommodation, this.room, this.rate});

  factory Penginapan.fromJson(Map<String, dynamic> obj) {
    return Penginapan(
      id: obj['id'],
      accommodation: obj['accommodation'],
      room: obj['room'],
      rate: obj['rate'] is String
          ? int.tryParse(obj['rate']) // Convert String to int safely
          : obj['rate'], // Keep it as int if it's already int
    );
  }
}
```

[helpers/user_info.dart]

```
import 'package:shared_preferences/shared_preferences.dart';
class UserInfo {
  Future setToken(String value) async {
    final SharedPreferences pref = await SharedPreferences.getInstance();
    return pref.setString("token", value);
  }
  Future<String?> getToken() async {
    final SharedPreferences pref = await SharedPreferences.getInstance();
    return pref.getString("token");
  }
  Future setUserID(int value) async {
    final SharedPreferences pref = await SharedPreferences.getInstance();
    return pref.setInt("userID", value);
  }
  Future<int?> getUserID() async {
    final SharedPreferences pref = await SharedPreferences.getInstance();
    return pref.getInt("userID");
  }
  Future logout() async {
    final SharedPreferences pref = await SharedPreferences.getInstance();
    pref.clear();
  }
}
```

[helpers/app_exception.dart]

```
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: ");
}
```

[helpers/api.dart]

```
import 'dart:io';
import 'package:http/http.dart' as http;
import '/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 the server with StatusCode:
  ${response.statusCode}',
      );
    }
  }
  }
```

[helpers/api_url.dart]

```
class ApiUrl {
 static const String baseUrl = 'http://responsi.webwizards.my.id/api/pariwisata';
 static const String listPenginapan = baseUrl + '/penginapan';
 static const String createPenginapan = baseUrl + '/penginapan';

 static const String registrasi = 'http://responsi.webwizards.my.id/api/registrasi';
 static const String login = 'http://responsi.webwizards.my.id/api/login';


 static String showPenginapan(int id) {
  return baseUrl + '/penginapan/' + id.toString();
 }

 static String updatePenginapan(int id) {
  return baseUrl + '/penginapan/' + id.toString() + '/update';
 }
```

```
  static String deletePenginapan(int id) {
    return baseUrl + '/penginapan/' + id.toString() + '/delete';
  }
}
```

[bloc/login_bloc.dart]

```
import 'dart:convert';
import '/helpers/api.dart';
import '/helpers/api_url.dart';
import '/model/login.dart';
class LoginBloc {
  static Future<Login> login({String? email, String? password}) async {
    String apiUrl = ApiUrl.login;
    var body = {"email": email, "password": password};
    var response = await Api().post(apiUrl, body);
    var jsonObj = json.decode(response.body);
    return Login.fromJson(jsonObj);
  }
}
```

[bloc/logout_bloc.dart]

```
import '/helpers/user_info.dart';
class LogoutBloc {
  static Future logout() async {
    await UserInfo().logout();
  }
}
```

[bloc/registrasi_bloc.dart]

```
import 'dart:convert';
import '/helpers/api.dart';
import '/helpers/api_url.dart';
import '/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);
  }
}
```

[bloc/penginapan_bloc.dart]

```
import 'dart:convert';
import '/helpers/api.dart';
import '/helpers/api_url.dart';
import '/model/penginapan.dart';  // Update the import to use the Penginapan model

class PenginapanBloc {
  static Future<List<Penginapan>> getPenginapan() async {
    String apiUrl = ApiUrl.listPenginapan; // Update to use the new penginapan API
    var response = await Api().get(apiUrl);
    var jsonObj = json.decode(response.body);
    List<dynamic> listPenginapan = (jsonObj as Map<String, dynamic>)['data'];
    List<Penginapan> penginapans = [];

    for (int i = 0; i < listPenginapan.length; i++) {
      penginapans.add(Penginapan.fromJson(listPenginapan[i])); // Use the Penginapan
model
    }

    return penginapans;
  }

  static Future<bool> addPenginapan({required Penginapan penginapan}) async {
    String apiUrl = ApiUrl.createPenginapan; // Update to use the new penginapan API
    var body = {
      "accommodation": penginapan.accommodation,
      "room": penginapan.room,
      "rate": penginapan.rate.toString(), // Ensure data is formatted correctly
    };

    var response = await Api().post(apiUrl, body);
    var jsonObj = json.decode(response.body);
    return jsonObj['status'];
  }

  static Future<bool> updatePenginapan({required Penginapan penginapan}) async {
    String apiUrl = ApiUrl.updatePenginapan(penginapan.id!); // Update to use the new
penginapan API
    var body = {
      "accommodation": penginapan.accommodation,
      "room": penginapan.room,
    "rate": penginapan.rate,
   };

   var response = await Api().put(apiUrl, jsonEncode(body));
   var jsonObj = json.decode(response.body);
   return jsonObj['status'];
  }

  static Future<bool> deletePenginapan({required int id}) async {
   String apiUrl = ApiUrl.deletePenginapan(id); // Update to use the new penginapan
API
   var response = await Api().delete(apiUrl);
   var jsonObj = json.decode(response.body);
   return (jsonObj as Map<String, dynamic>)['status']; // Ensure the response structure
matches
  }
}
```

[widget/success_dialog.dart]

```
import 'package:flutter/material.dart';
class Consts {
  Consts._();
  static const double padding = 16.0;
  static const double avatarRadius = 66.0;
}
class SuccessDialog extends StatelessWidget {
  final String? description;
  final VoidCallback? okClick;
  const SuccessDialog({Key? key, this.description, this.okClick})
     : super(key: key);
  @override
  Widget build(BuildContext context) {
   return Dialog(
     shape: RoundedRectangleBorder(
       borderRadius: BorderRadius.circular(Consts.padding)),
     elevation: 0.0,
     backgroundColor: Colors.transparent,
     child: dialogContent(context),
   );
  }

  dialogContent(BuildContext context) {
   return Container(
      padding: const EdgeInsets.only(
      top: Consts.padding,
      bottom: Consts.padding,
      left: Consts.padding,
      right: Consts.padding,
   ),
   margin: const EdgeInsets.only(top: Consts.avatarRadius),
   decoration: BoxDecoration(
   color: Colors.white,
   shape: BoxShape.rectangle,
   borderRadius: BorderRadius.circular(Consts.padding),
   boxShadow: const [
   BoxShadow(
   color: Colors.black26,
   blurRadius: 10.0,
   offset: Offset(0.0, 10.0),
   ),
   ],
   ),
   child: Column(
   mainAxisSize: MainAxisSize.min,
   children: [
   const Text(
   "SUKSES",
   style: TextStyle(
   fontSize: 24.0,
   fontWeight: FontWeight.w700,
      color: Colors.green),
   ),
     const SizedBox(height: 16.0),
     Text(
      description!,
      textAlign: TextAlign.center,
      style: const TextStyle(
       fontSize: 16.0,
      ),
     ),
     const SizedBox(height: 24.0),
     Align(
      alignment: Alignment.bottomRight,
      child: OutlinedButton(
       onPressed: () {
        Navigator.of(context).pop(); // To close the dialog
        okClick!();
       },
       child: const Text("OK"),
      ),
     )
   ],
     ),
     );
   }
}
```

[widget/warning_dialog.dart]

```
import 'package:flutter/material.dart';
class Consts {
  Consts._();
  static const double padding = 16.0;
  static const double avatarRadius = 66.0;
}
class WarningDialog extends StatelessWidget {
  final String? description;
  final VoidCallback? okClick;
  const WarningDialog({Key? key, this.description, this.okClick})
      : super(key: key);
  @override
  Widget build(BuildContext context) {
   return Dialog(
    shape: RoundedRectangleBorder(
       borderRadius: BorderRadius.circular(Consts.padding)),
    elevation: 0.0,
    backgroundColor: Colors.transparent,
    child: dialogContent(context),
   );
  }
  dialogContent(BuildContext context) {
   return Container(
      padding: const EdgeInsets.only(
      top: Consts.padding,
      bottom: Consts.padding,
      left: Consts.padding,
      right: Consts.padding,
   ),
   margin: const EdgeInsets.only(top: Consts.avatarRadius),
   decoration: BoxDecoration(
   color: Colors.white,
   shape: BoxShape.rectangle,
   borderRadius: BorderRadius.circular(Consts.padding),
   boxShadow: const [
   BoxShadow(
   color: Colors.black26,
   blurRadius: 10.0,
   offset: Offset(0.0, 10.0),
  ),
  ],
  ),
  child: Column(
  mainAxisSize: MainAxisSize.min,
  children: [
  const Text(
  "GAGAL",
  style: TextStyle(
  fontSize: 24.0, fontWeight: FontWeight.w700, color: Colors.red),
  ),
  const SizedBox(height: 16.0),
  Text(
  description!,
  textAlign: TextAlign.center,
  style: const TextStyle(
   fontSize: 16.0,
  ),
  ),
   const SizedBox(height: 24.0),
   Align(
    alignment: Alignment.bottomRight,
    child: ElevatedButton(
     onPressed: () {
      Navigator.of(context).pop(); // To close the dialog
     },
     child: const Text("OK"),
    ),
   )
  ],
  ),
  );
 }
}
```