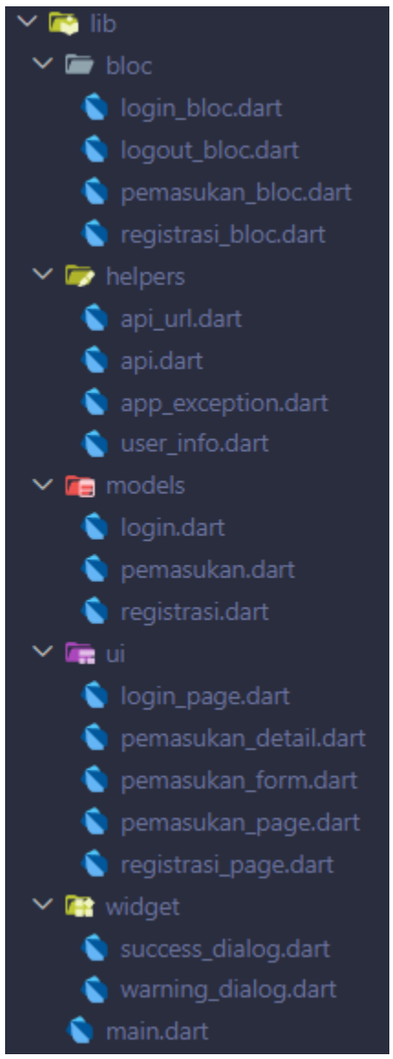
Nama : Jehian Athaya Tsani Az Zuhry
NIM : H1D022006
Praktikum : Mobile
Shift KRS/Baru : C/D

RESPONSI 1 APLIKASI MANAJEMEN KEUANGAN PEMASUKAN PELANGI

1. Struktur Aplikasi
    a. helpers: Menyimpan file user_info.dart yang mengelola penyimpanan data pengguna menggunakan SharedPreferences.
    b. ui: Menyimpan antarmuka pengguna untuk setiap halaman (seperti login_page.dart, pemasukan_page.dart).
    c. bloc: Mengelola logika bisnis dan interaksi dengan API (seperti login_bloc.dart, logout_bloc.dart, pemasukan_bloc.dart).
    d. models: Menyimpan model data, seperti pemasukan.dart, yang berfungsi untuk merepresentasikan data DTO (Data Transfer Object).



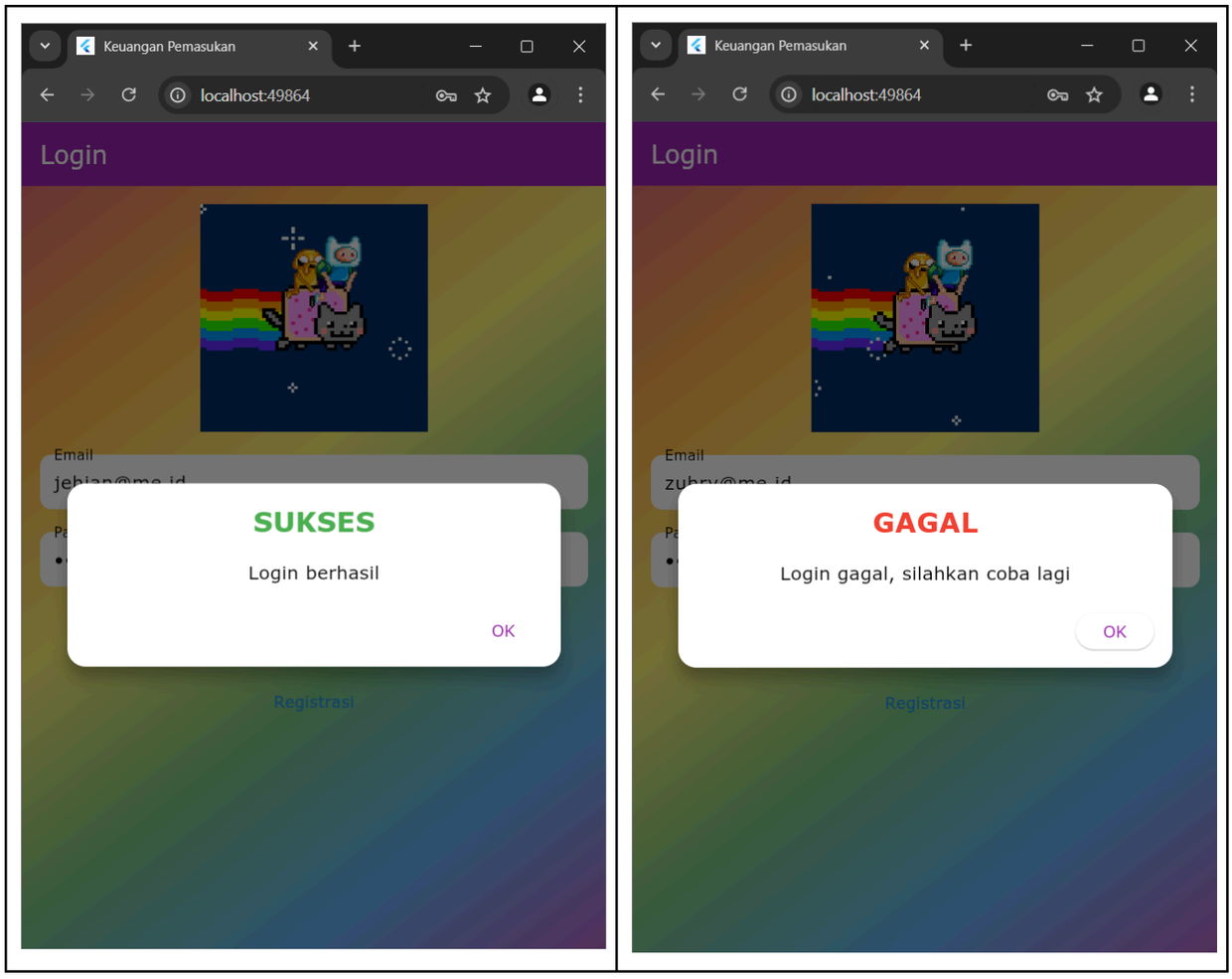
2. File main.dart
   Merupakan titik awal aplikasi, di mana:
    a. Aplikasi dijalankan dengan runApp().
    b. Metode isLogin() digunakan untuk memeriksa apakah pengguna telah login dengan mengecek token yang disimpan. Jika token ada, pengguna diarahkan ke halaman PemasukanPage, jika tidak, diarahkan ke LoginPage.

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

3. Halaman Login (login_page.dart)
   Halaman ini berfungsi untuk login pengguna. Terdapat beberapa komponen utama:
   a. Formulir Login: Terdiri dari email dan password yang harus diisi. Validasi dilakukan untuk memastikan tidak ada field yang kosong.
   b. Metode _submit(): Mengirim permintaan login ke API. Jika berhasil, token dan user ID disimpan, dan pengguna diarahkan ke PemasukanPage.

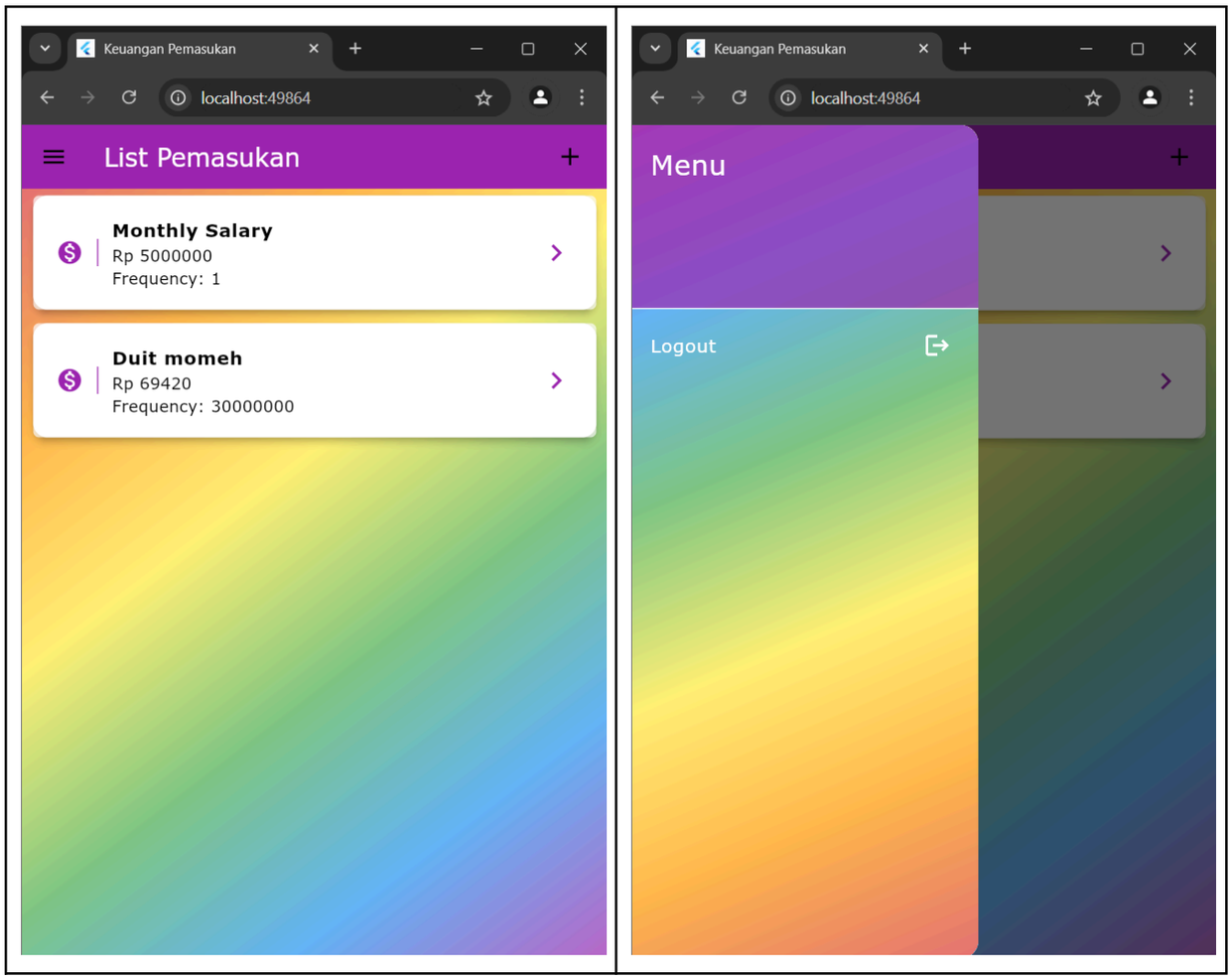
```
void _submit() {
  _formKey.currentState!.save();
  setState(() {
    _isLoading = true;
  });

  LoginBloc.login(
          email: _emailTextboxController.text,
          password: _passwordTextboxController.text)
      .then((value) async {
    if (value.code == 200) {
      await UserInfo().setToken(value.token ?? "");
      await UserInfo().setUserID(int.tryParse(value.userID.toString()) ?? 0);
      // Navigasi ke PemasukanPage
    } else {
      // Tampilkan dialog kesalahan
    }
  }, onError: (error) {
    // Tampilkan dialog kesalahan
  });
}
```



4. Halaman Pemasukan (pemasukan_page.dart)
   Halaman yang menampilkan daftar pemasukan:
   a. Drawer: Menyediakan menu logout. Ketika dipilih, pengguna diarahkan kembali ke halaman login.
   b. ListPemasukan: Menampilkan daftar pemasukan dengan menggunakan widget ListView yang memanggil API untuk mengambil data pemasukan.

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('List Pemasukan'),
      actions: [
        GestureDetector(
          child: const Icon(Icons.add),
          onTap: () {
            Navigator.push(context, MaterialPageRoute(builder: (context) => const PemasukanForm()));
          },
        )
      ],
    ),
    body: FutureBuilder<List<Pemasukan>>(
      future: PemasukanBloc.getPemasukans(),
      builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
          return const Center(child: CircularProgressIndicator());
        } else if (snapshot.hasError) {
          return Center(child: Text('Error: ${snapshot.error}'));
        } else if (!snapshot.hasData || snapshot.data!.isEmpty) {
          return const Center(child: Text('Tidak ada data pemasukan.'));
        } else {
          return ListPemasukan(list: snapshot.data);
        }
      },
    ),
  );
}
```



5. API Interaction

   Penggunaan API adalah kunci pada fungsi aplikasi ini. Berikut beberapa endpoint utama yang digunakan:

   a. Registrasi: POST /registrasi untuk mendaftar pengguna baru.
   b. Login: POST /login untuk autentikasi pengguna.
   c. CRUD Pemasukan:
      i. Mendapatkan daftar pemasukan: GET /keuangan/pemasukan
      ii. Menambah pemasukan: POST /keuangan/pemasukan
      iii. Mengedit pemasukan: PUT /keuangan/pemasukan/{{id}}/update
      iv. Menghapus pemasukan: DELETE /keuangan/pemasukan/{{id}}/delete

```dart
// Contoh request untuk menambahkan pemasukan:
Future<void> addPemasukan(String source, int amount, int frequency) async {
  final response = await http.post(
    Uri.parse('http://103.196.155.42/api/keuangan/pemasukan'),
    headers: {'Content-Type': 'application/json'},
    body: jsonEncode({'source': source, 'amount': amount, 'frequency': frequency}),
  );
}
```

6. Halaman Registrasi (registrasi_page.dart)

   Halaman ini berfungsi untuk registrasi pengguna baru. Berikut adalah komponen utama yang terdapat dalam halaman registrasi:

   a. Struktur Halaman
      i. Formulir Registrasi: Pengguna diminta untuk mengisi nama, email, password, dan konfirmasi password. Validasi dilakukan pada setiap field.
      ii. Metode _buildTextField(): Membuat textbox untuk input pengguna dengan validasi yang sesuai untuk setiap field.
   b. Validasi Input
      i. Nama harus diisi minimal 3 karakter.
      ii. Email harus valid sesuai format email.
      iii. Password harus diisi minimal 6 karakter.
      iv. Konfirmasi password harus sama dengan password.

```dart
validator: (value) {
  if (value!.isEmpty) {
```

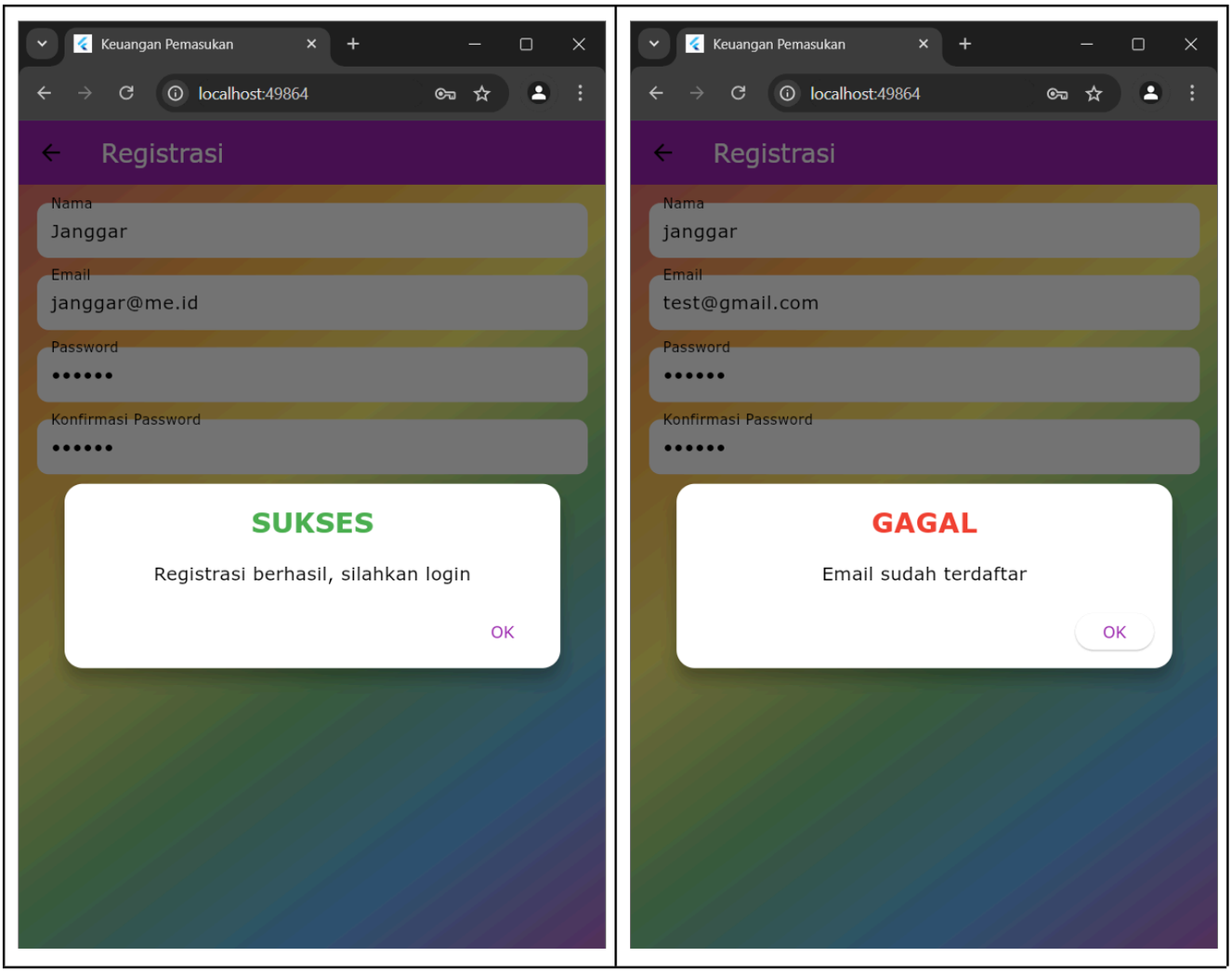
```
    return "$label harus diisi";
  }
  if (label == "Nama" && value.length < 3) {
    return "Nama harus diisi minimal 3 karakter";
  }
  if (label == "Email") {
                                                    Pattern             pattern             =
r'^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\[[0-9]{1,3}\.[0-9]{1,3}
\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$';
    RegExp regex = RegExp(pattern.toString());
    if (!regex.hasMatch(value)) {
      return "Email tidak valid";
    }
  }
  if (label == "Password" && value.length < 6) {
    return "Password harus diisi minimal 6 karakter";
  }
  if (label == "Konfirmasi Password" && value != _passwordTextboxController.text) {
    return "Konfirmasi Password tidak sama";
  }
  return null;
}
```

c. Pengolahan Registrasi

i. Metode _checkEmailAndSubmit(): Memeriksa apakah email sudah terdaftar sebelum mengirim permintaan registrasi.

ii. Metode _submit(): Mengirimkan data registrasi ke API menggunakan RegistrasiBloc dan menampilkan dialog sukses atau kesalahan berdasarkan respon dari API.

```
void _submit() {
  // ...
  RegistrasiBloc.registrasi(
    nama: _namaTextboxController.text,
    email: _emailTextboxController.text,
    password: _passwordTextboxController.text,
  ).then((value) {
    if (value['status']) {
      // Tampilkan dialog sukses
    } else {
      // Tampilkan dialog kesalahan
    }
  // ...
  });
}
```



7. Bloc Registrasi (registrasi_bloc.dart)
   File ini mengatur komunikasi dengan API untuk proses registrasi.
   a. Metode registrasi(): Mengambil data dari API untuk mendaftar pengguna baru.

```dart
static Future<Map<String, dynamic>> registrasi({
  String? nama,
  String? email,
  String? password,
}) async {
  String apiUrl = ApiUrl.registrasi;
  var body = {"nama": nama, "email": email, "password": password};
  try {
    var response = await Api().post(apiUrl, body);
    var jsonObj = json.decode(response.body);
    return {
      'status': jsonObj['status'],
      'message': jsonObj['message'] ?? 'Email sudah terdaftar',
    };
  } catch (e) {
    return {
      'status': false,
      'message': 'Terjadi kesalahan: ${e.toString()}',
    };
  }
}
```

8. Kelas API (api.dart)
   File ini menangani semua interaksi dengan API menggunakan metode HTTP: POST, GET, PUT, dan DELETE.
   a. Metode post(): Mengirim permintaan untuk mendaftar pengguna baru.
   b. Metode _returnResponse(): Mengolah respons dari server dan menangani status kode HTTP yang berbeda.

```dart
Future<dynamic> post(dynamic url, dynamic data) async {
  var token = await UserInfo().getToken();
  //...
  final response = await http.post(Uri.parse(url), body: data,
```

```
    headers: {HttpHeaders.authorizationHeader: "Bearer $token"});
  responseJson = _returnResponse(response);
}
```

9. Kelas URL API (api_url.dart)
   Menentukan semua URL endpoint untuk API. Contoh URL untuk registrasi pengguna:

```
static const String registrasi = baseUrl + '/registrasi';
```

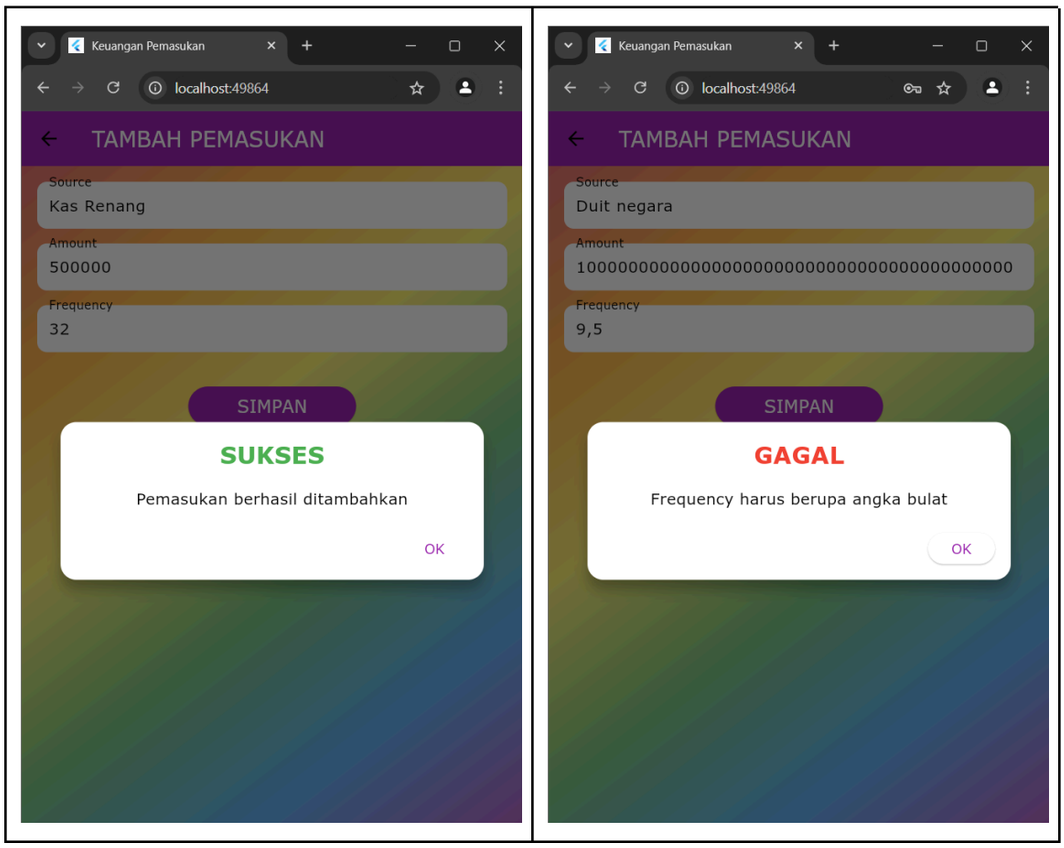
10. Halaman Form Pemasukan (pemasukan_form.dart)
    a. Struktur Halaman
       Halaman ini digunakan untuk menambah atau mengedit data pemasukan. Berikut adalah komponen utama yang terdapat di dalamnya:
       i. Formulir Pemasukan: Memungkinkan pengguna untuk mengisi Source, Amount, dan Frequency untuk setiap pemasukan.
       ii. Metode _isUpdate(): Memeriksa apakah halaman ini digunakan untuk mengedit (update) data. Jika ada data pemasukan yang sudah ada, field akan diisi dengan data tersebut.
    b. Implementasi
       i. Widget Form

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
              title: Text(_judul, style: const TextStyle(fontFamily: 'Verdana', color:
Colors.white)),
      backgroundColor: Colors.purple,
    ),
    body: Container(
      height: double.infinity,
      decoration: BoxDecoration( // Gradient background
        gradient: LinearGradient(
          begin: Alignment.topLeft,
          end: Alignment.bottomRight,
          colors: [
            Colors.red[300]!,
            Colors.orange[300]!,
            Colors.yellow[300]!,
            Colors.green[300]!,
            Colors.blue[300]!,
            Colors.purple[300]!,
          ],
        ),
      ),
      child: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Form(
            key: _formKey,
            child: Column(
              children: [
                _buildTextField(_sourceTextboxController, "Source"),
                _buildTextField(_amountTextboxController, "Amount", TextInputType.number),
                                    _buildTextField(_frequencyTextboxController, "Frequency",
TextInputType.number),
                const SizedBox(height: 20),
                _buttonSubmit(), // Tombol untuk simpan atau ubah
              ],
            ),
          ),
        ),
      ),
    ),
  );
}
```

       ii. Simpan Data
       Metode _simpan(): Mengirim data ke API dan menunjukkan dialog berhasil atau gagal.

```
_simpan() async {
  setState(() {
```

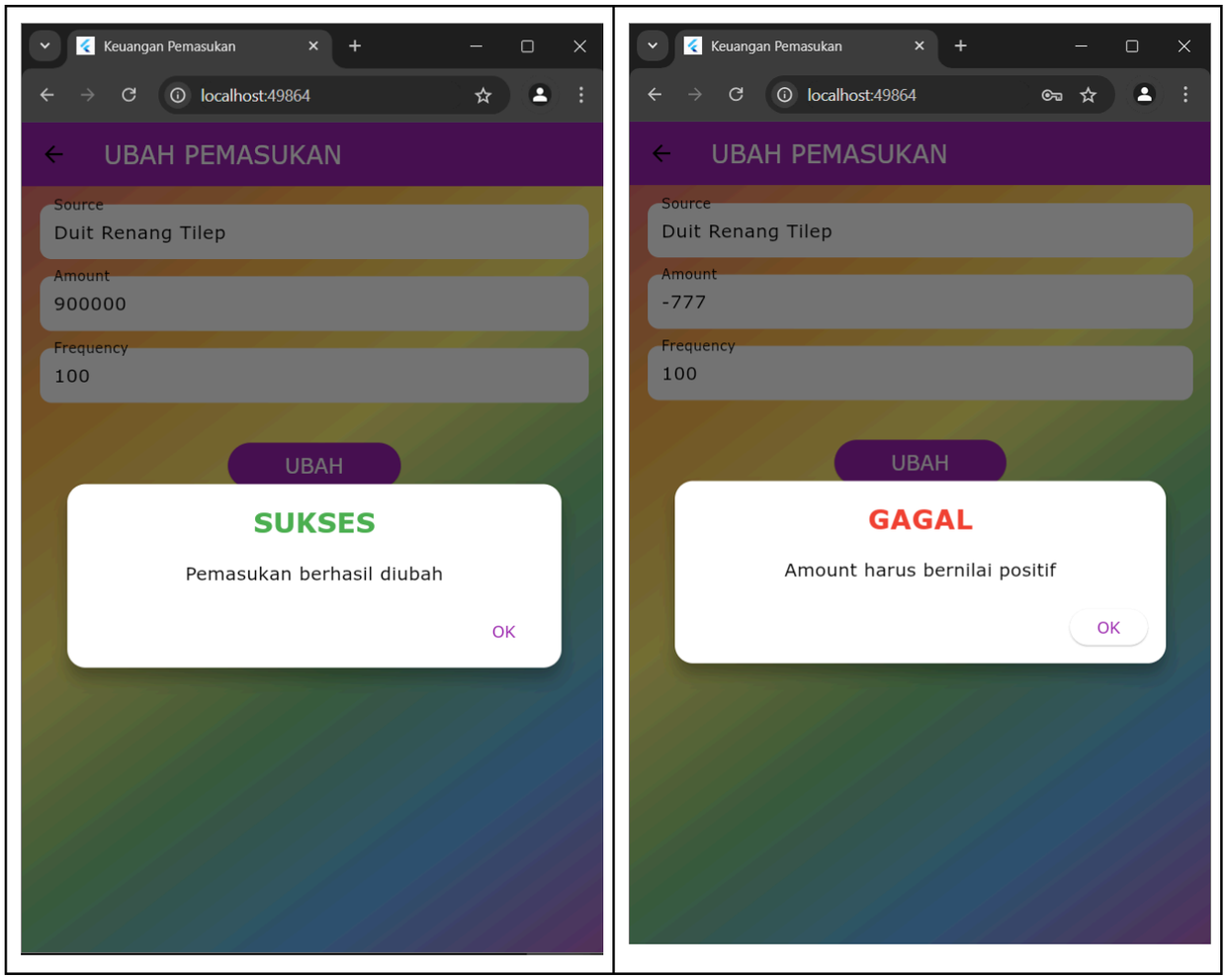
```
    _isLoading = true;
  });
  Pemasukan createPemasukan = Pemasukan(
    source: _sourceTextboxController.text,
    amount: int.parse(_amountTextboxController.text),
    frequency: int.parse(_frequencyTextboxController.text),
  );
  try {
    var result = await PemasukanBloc.addPemasukan(pemasukan: createPemasukan);
    if (result['success']) {
      // Tampilkan dialog sukses
    }
  } catch (error) {
    // Tampilkan dialog gagal
  } finally {
    setState(() {
      _isLoading = false;
    });
  }
}
```



iii. Ubah Data

Metode _ubah(): Memproses pengeditan data yang sudah ada.

```
_ubah() async {
  setState(() {
    _isLoading = true;
  });
  Pemasukan updatePemasukan = Pemasukan(
    id: widget.pemasukan!.id,
    source: _sourceTextboxController.text,
    amount: int.parse(_amountTextboxController.text),
    frequency: int.parse(_frequencyTextboxController.text),
  );
  PemasukanBloc.updatePemasukan(pemasukan: updatePemasukan).then((value) {
    // Tampilkan dialog sukses atau gagal
  });
  setState(() {
    _isLoading = false;
  });
}
```



11. Bloc Pemasukan (pemasukan_bloc.dart)
    a. Mendapatkan Daftar Pemasukan

```dart
static Future<List<Pemasukan>> getPemasukans() async {
  String apiUrl = ApiUrl.listPemasukan;
  var response = await Api().get(apiUrl);
  // Parsing data dari respons
}
```

    b. Menambah Pemasukan

```dart
static Future<Map<String, dynamic>> addPemasukan({Pemasukan? pemasukan}) async {
  String apiUrl = ApiUrl.createPemasukan;
  var body = {
    "source": pemasukan!.source,
    "amount": pemasukan.amount.toString(),
    "frequency": pemasukan.frequency.toString()
  };
  try {
    var response = await Api().post(apiUrl, body);
    // Parsing respons
  } catch (e) {
    return {
      'success': false,
      'message': 'Terjadi kesalahan: ${e.toString()}',
    };
  }
}
```

    c. Memperbarui Pemasukan

```dart
static Future updatePemasukan({required Pemasukan pemasukan}) async {
  String apiUrl = ApiUrl.updatePemasukan(pemasukan.id!);
  var body = {
    "source": pemasukan.source,
    "amount": pemasukan.amount.toString(),
    "frequency": pemasukan.frequency.toString()
  };
  var response = await Api().put(apiUrl, jsonEncode(body));
  return jsonObj['status'];
}
```

d. Menghapus Pemasukan

```
static Future<bool> deletePemasukan({int? id}) async {
  String apiUrl = ApiUrl.deletePemasukan(id!);
  var response = await Api().delete(apiUrl);
  if (response != null && response.body.isNotEmpty) {
    var jsonObj = json.decode(response.body);
    return jsonObj['status'] == true;
  } else {
    return false;
  }
}
```

12. Halaman Detail Pemasukan (pemasukan_detail.dart)
    a. Memperlihatkan Detail Pemasukan

    Halaman ini menampilkan detail pemasukan yang dipilih sebelumnya.

    i. Detail Card untuk menampilkan informasi pemasukan.
    ii. Tombol Aksi: Tombol untuk Edit dan Delete.

    Implementasi

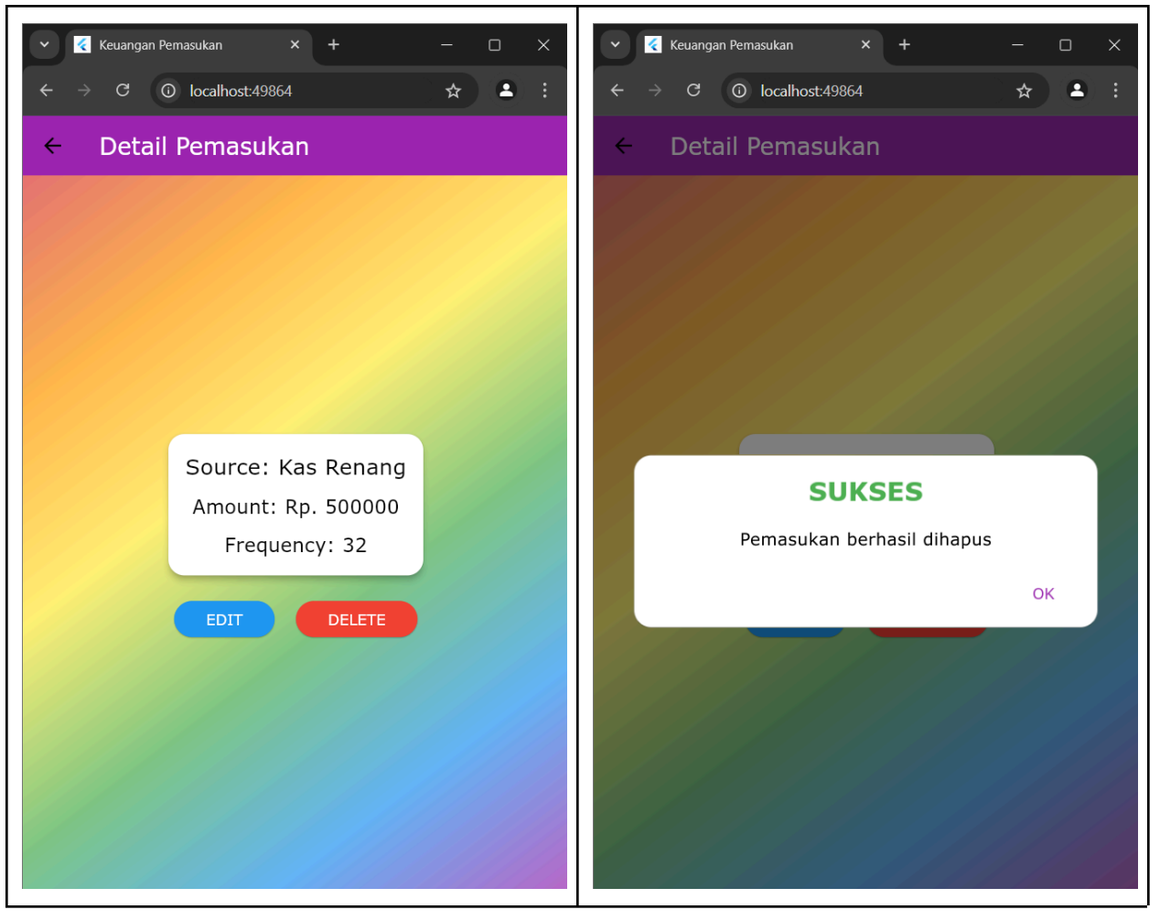
```
Widget _buildDetailCard() {
  return Card(
    elevation: 5,
    child: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        children: [
              Text("Source: ${widget.pemasukan!.source}", style: const TextStyle(fontSize:
20.0)),
          // Menampilkan amount dan frequency
        ],
      ),
    ),
  );
}
```

    b. Aksi Edit dan Delete

    i. Edit: Mengarah ke halaman formulir untuk mengedit data pemasukan.
    ii. Delete: Menampilkan dialog konfirmasi sebelum menghapus pemasukan.

    Konfirmasi Hapus

```
void _confirmDelete(BuildContext context) {
  AlertDialog alertDialog = AlertDialog(
    content: const Text("Yakin ingin menghapus data ini?"),
    actions: [
      OutlinedButton(
        child: const Text("Ya"),
        onPressed: () async {
          bool success = await PemasukanBloc.deletePemasukan(id: widget.pemasukan!.id!);
          // Tampilkan dialog sukses atau gagal
        },
      ),
      OutlinedButton(
        child: const Text("Batal"),
        onPressed: () => Navigator.pop(context),
      ),
    ],
  );
  showDialog(builder: (context) => alertDialog, context: context);
}
```


Keuangan Pemasukan
localhost:49864
Detail Pemasukan
Source: Kas Renang
Amount: Rp. 500000
Frequency: 32
EDIT
DELETE
Keuangan Pemasukan
localhost:49864
Detail Pemasukan
SUKSES
Pemasukan berhasil dihapus
OK