

Nama : Nailia Farah Isnaeni

NIM : H1D022065

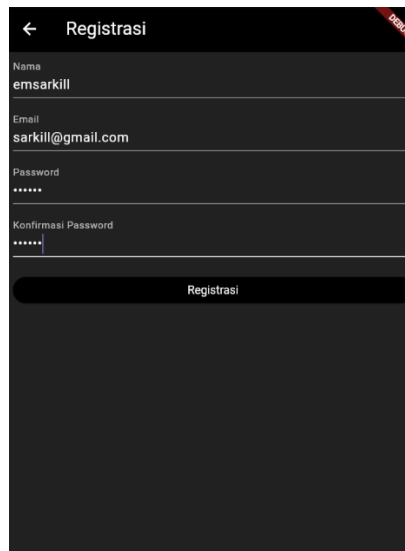
Shift : C

Responsi 1 Praktikum Pemrograman Web

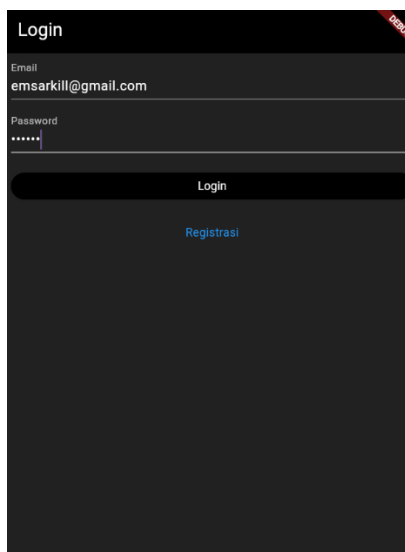
Kesehatan

Kesehatan Mental

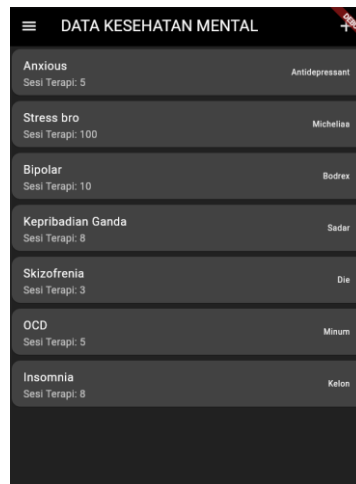
1. Registrasi



2. Login

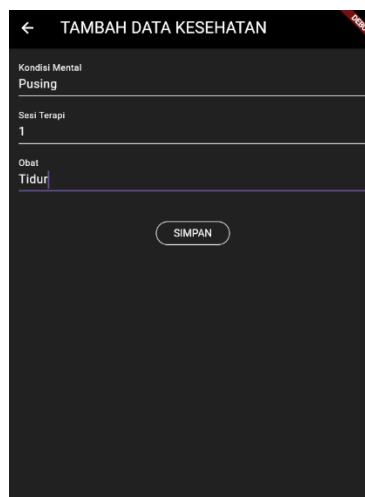


3. Page



DATA KESEHATAN MENTAL	
Anxious Sesi Terapi: 5	Antidepressant
Stress bro Sesi Terapi: 100	Michellaa
Bipolar Sesi Terapi: 10	Bodrex
Kepribadian Ganda Sesi Terapi: 8	Sadar
Skizofrenia Sesi Terapi: 3	Die
OCD Sesi Terapi: 5	Minum
Insomnia Sesi Terapi: 8	Kelon

4. Tambah



← TAMBAH DATA KESEHATAN

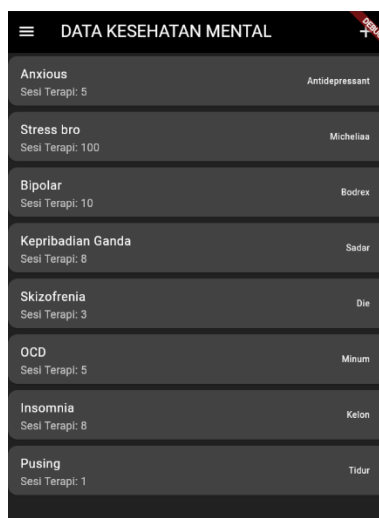
Kondisi Mental
Pusing

Sesi Terapi
1

Obat
Tidur

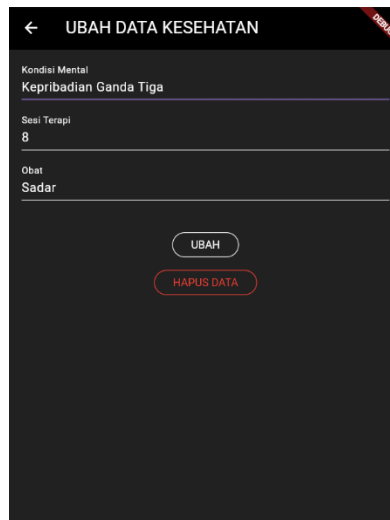
SIMPAN

After

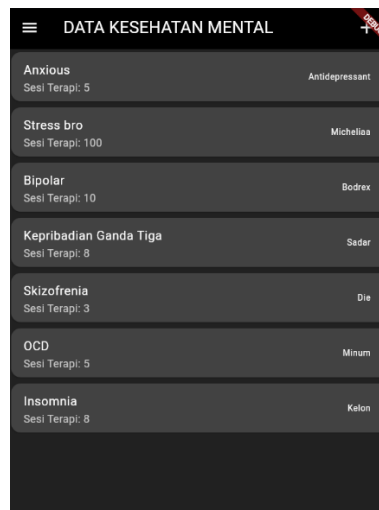


DATA KESEHATAN MENTAL	
Anxious Sesi Terapi: 5	Antidepressant
Stress bro Sesi Terapi: 100	Michellaa
Bipolar Sesi Terapi: 10	Bodrex
Kepribadian Ganda Sesi Terapi: 8	Sadar
Skizofrenia Sesi Terapi: 3	Die
OCD Sesi Terapi: 5	Minum
Insomnia Sesi Terapi: 8	Kelon
Pusing Sesi Terapi: 1	Tidur

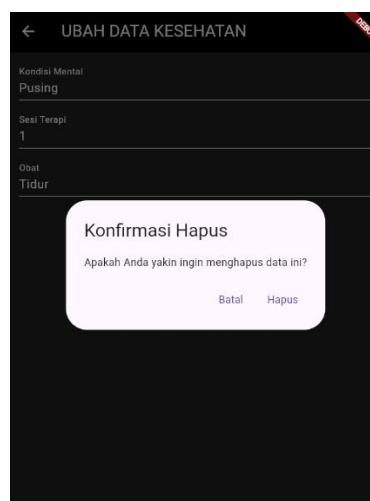
5. Edit



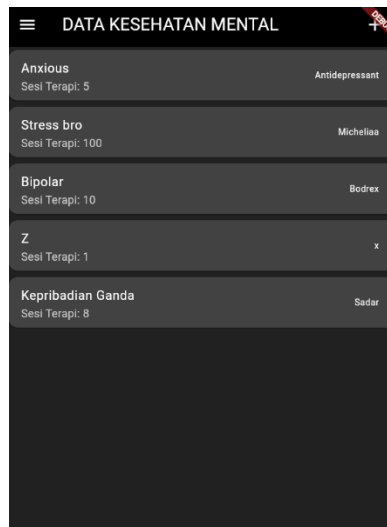
After



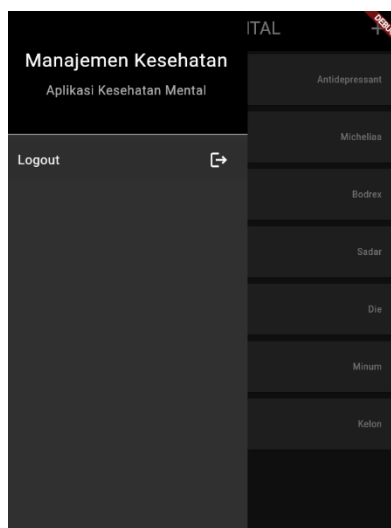
6. Hapus



After



7. Sidemenu



PENJELASAN KODE :

1. Registrasi page

Class `_RegistrasiPageState` : Kelas ini merupakan bagian dari `RegistrasiPage` yang mengelola status dan tampilan halaman. Di dalamnya, terdapat variabel untuk menyimpan status pemuatan (`_isLoading`) dan kontroler untuk mengambil input dari pengguna (`_namaTextboxController`, `_emailTextboxController`, `_passwordTextboxController`).

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

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

Widget `_namaTextField` : Widget ini membuat kolom input untuk nama. Kolom ini menggunakan `TextFormField` dan memiliki validator yang memeriksa apakah input memiliki minimal 3 karakter. Desain kolom ini juga mengikuti tema warna yang konsisten dengan halaman.

```
Widget _namaTextField() {
  return TextFormField(
    decoration: const InputDecoration(
      labelText: "Nama",
      labelStyle: TextStyle(fontFamily: 'Georgia', color: Colors.white70),
      enabledBorder: UnderlineInputBorder(
        borderSide: BorderSide(color: Colors.white54),
      ), // UnderlineInputBorder
      focusedBorder: UnderlineInputBorder(
        borderSide: BorderSide(color: Colors.white),
      ), // UnderlineInputBorder
    ), // InputDecoration
    style: const TextStyle(fontFamily: 'Georgia', color: Colors.white),
    keyboardType: TextInputType.text,
    controller: _namaTextboxController,
    validator: (value) {
      if (value!.length < 3) {
        return "Nama harus diisi minimal 3 karakter";
      }
      return null;
    },
  ); // TextFormField
}
```

Widget `_emailTextField` : Mirip dengan kolom nama, widget ini untuk memasukkan email. Validatornya memeriksa apakah email diisi dan formatnya valid menggunakan ekspresi reguler.

```
Widget _emailTextField() {
  return TextFormField(
    decoration: const InputDecoration(
      labelText: "Email",
      labelStyle: TextStyle(fontFamily: 'Georgia', color: Colors.white70),
      enabledBorder: UnderlineInputBorder(
        borderSide: BorderSide(color: Colors.white54),
      ), // UnderlineInputBorder
      focusedBorder: UnderlineInputBorder(
        borderSide: BorderSide(color: Colors.white),
      ), // UnderlineInputBorder
    ), // InputDecoration
    style: const TextStyle(fontFamily: 'Georgia', color: Colors.white),
    keyboardType: TextInputType.emailAddress,
    controller: _emailTextboxController,
    validator: (value) {
      if (value!.isEmpty) {
        return 'Email harus diisi';
      }

      Pattern pattern =
        r'^([*>()[]\.\.,;\s@"]+)(\.[*>()[]\.\.,;\s@"]+)*(\.[*>()[]\.\.,;\s@"]+)*$';
      RegExp regex = RegExp(pattern.toString());
      if (!regex.hasMatch(value)) {
        return "Email tidak valid";
      }
      return null;
    },
  ); // TextFormField
}
```

Widget `_passwordTextField` : Widget ini digunakan untuk menginput password. Ini juga memiliki validator yang memastikan password terdiri dari minimal 6 karakter.

```

Widget _passwordTextField() {
  return TextFormField(
    decoration: const InputDecoration(
      labelText: "Password",
      labelStyle: TextStyle(fontFamily: 'Georgia', color: Colors.white70),
      enabledBorder: UnderlineInputBorder(
        borderSide: BorderSide(color: Colors.white54),
      ), // UnderlineInputBorder
      focusedBorder: UnderlineInputBorder(
        borderSide: BorderSide(color: Colors.white),
      ), // UnderlineInputBorder
    ), // InputDecoration
    style: const TextStyle(fontFamily: 'Georgia', color: Colors.white),
    obscureText: true,
    controller: _passwordTextboxController,
    validator: (value) {
      if (value!.length < 6) {
        return "Password harus diisi minimal 6 karakter";
      }
      return null;
    },
  ); // TextFormField
}

```

Widget _passwordKonfirmasiTextField : Widget ini digunakan untuk mengkonfirmasi password. Validatornya memastikan bahwa nilai yang dimasukkan sama dengan password yang telah diinput sebelumnya.

```

Widget _passwordKonfirmasiTextField() {
  return TextFormField(
    decoration: const InputDecoration(
      labelText: "Konfirmasi Password",
      labelStyle: TextStyle(fontFamily: 'Georgia', color: Colors.white70),
      enabledBorder: UnderlineInputBorder(
        borderSide: BorderSide(color: Colors.white54),
      ), // UnderlineInputBorder
      focusedBorder: UnderlineInputBorder(
        borderSide: BorderSide(color: Colors.white),
      ), // UnderlineInputBorder
    ), // InputDecoration
    style: const TextStyle(fontFamily: 'Georgia', color: Colors.white),
    obscureText: true,
    validator: (value) {
      if (value != _passwordTextboxController.text) {
        return "Konfirmasi Password tidak sama";
      }
      return null;
    },
  ); // TextFormField
}

```

Widget _buttonRegistrasi : Widget ini menciptakan tombol untuk melakukan registrasi. Jika sedang memuat (loading), tombol akan menampilkan indikator pemuatan, jika tidak, tombol akan menampilkan teks "Registrasi". Saat tombol ditekan, fungsi _submit akan dipanggil.

```

Widget _buttonRegistrasi() {
  return SizedBox(
    width: double.infinity, // Tombol memenuhi lebar
    child: ElevatedButton(
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.black,
        foregroundColor: Colors.white,
        textStyle: const TextStyle(fontFamily: 'Georgia'),
        padding: const EdgeInsets.symmetric(vertical: 15), // Tinggi tombol
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(30), // Membulatkan sudut
        ), // RoundedRectangleBorder
      ),
      child: _isLoading
        ? const SizedBox(
            height: 18,
            width: 18,
            child: CircularProgressIndicator(
              color: Colors.white,
              strokeWidth: 2,
            ), // CircularProgressIndicator
          ) // SizedBox
        : const Text("Registrasi"),
      onPressed: () {
        var validate = _formKey.currentState!.validate();
        if (validate && !_isLoading) {
          submit();
        }
      },
    ),
  );
}

```

2. Login page

LoginPageState : Kelas ini mengelola status dan tampilan halaman login. Di dalamnya, terdapat variabel untuk status pemuatan (_isLoading) dan kontroler untuk menangkap input pengguna (_emailTextboxController dan _passwordTextboxController).

```

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

```

Widget _emailTextField:Widget ini digunakan untuk input email. Menggunakan TextFormField, ia memiliki validator untuk memastikan bahwa kolom tidak kosong sebelum melanjutkan.


```

Widget _emailTextField() {
  return TextFormField(
    decoration: const InputDecoration(
      labelText: "Email",
      labelStyle: TextStyle(fontFamily: 'Georgia', color: Colors.white70),
      enabledBorder: UnderlineInputBorder(
        borderSide: BorderSide(color: Colors.white54),
      ), // UnderlineInputBorder
      focusedBorder: UnderlineInputBorder(
        borderSide: BorderSide(color: Colors.white),
      ), // UnderlineInputBorder
    ), // InputDecoration
    style: const TextStyle(fontFamily: 'Georgia', color: Colors.white),
    keyboardType: TextInputType.emailAddress,
    controller: _emailTextboxController,
    validator: (value) {
      if (value!.isEmpty) {
        return 'Email harus diisi';
      }
      return null;
    },
  ); // TextFormField
}

```

Widget _passwordTextField: Mirip dengan kolom email, widget ini untuk memasukkan password dengan pengaturan agar karakter yang dimasukkan tersembunyi. Validator memeriksa agar kolom tidak kosong.

```

Widget _passwordTextField() {
  return TextFormField(
    decoration: const InputDecoration(
      labelText: "Password",
      labelStyle: TextStyle(fontFamily: 'Georgia', color: Colors.white70),
      enabledBorder: UnderlineInputBorder(
        borderSide: BorderSide(color: Colors.white54),
      ), // UnderlineInputBorder
      focusedBorder: UnderlineInputBorder(
        borderSide: BorderSide(color: Colors.white),
      ), // UnderlineInputBorder
    ), // InputDecoration
    style: const TextStyle(fontFamily: 'Georgia', color: Colors.white),
    obscureText: true,
    controller: _passwordTextboxController,
    validator: (value) {
      if (value!.isEmpty) {
        return "Password harus diisi";
      }
      return null;
    },
  ); // TextFormField
}

```

Widget _buttonLogin: Widget ini membuat tombol untuk login. Jika sedang memuat, tombol akan menampilkan indikator pemuatan; jika tidak, tombol akan menampilkan teks "Login". Ketika tombol ditekan, fungsi _submit akan dipanggil untuk memproses login.

```

Widget _buttonLogin() {
  return SizedBox(
    width: double.infinity, // Tombol memenuhi lebar
    child: ElevatedButton(
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.black,
        foregroundColor: Colors.white,
        textStyle: const TextStyle(fontFamily: 'Georgia'),
        padding: const EdgeInsets.symmetric(vertical: 15), // Tinggi tombol
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(30), // Membulatkan sudut
        ), // RoundedRectangleBorder
      ),
      child: _isLoading
        ? const SizedBox(
            height: 18,
            width: 18,
            child: CircularProgressIndicator(
              color: Colors.white,
              strokeWidth: 2,
            ), // CircularProgressIndicator
          ) // SizedBox
        : const Text("Login"),
      onPressed: () {
        var validate = _formKey.currentState!.validate();
        if (validate && !_isLoading) {
          _submit();
        }
      }
    )
  );
}

```

Widget _menuRegistrasi: Widget ini menyediakan opsi bagi pengguna untuk melakukan registrasi jika mereka belum memiliki akun. Menggunakan InkWell, pengguna dapat mengetuk teks "Registrasi" untuk diarahkan ke halaman registrasi.

```

Widget _menuRegistrasi() {
  return Center(
    child: InkWell(
      child: const Text(
        "Registrasi",
        style: TextStyle(
          color: Colors.blue,
          fontFamily: 'Georgia',
        ), // TextStyle
      ), // Text
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => const RegistrasiPage()),
        );
      },
    ), // InkWell
  ); // Center
}

```

Lalu ada Fungsi `_submit`: Fungsi ini menangani proses login. Setelah memvalidasi form, ia mengubah status pemuatan menjadi true dan memanggil metode login dari `LoginBloc`, mengirimkan email dan password. Jika login berhasil (kode 200), pengguna akan diarahkan ke halaman kesehatan mental. Jika gagal, dialog peringatan akan ditampilkan. Fungsi `_showErrorDialog`: Fungsi ini menampilkan dialog peringatan jika terjadi kesalahan saat login, memberikan umpan balik kepada pengguna bahwa login gagal.

3. Kesehatan mental page

Widget `Drawer`: `Drawer` berisi header dan opsi logout. Ketika pengguna mengetuk opsi logout, `LogoutBloc` akan dipanggil untuk mengeluarkan pengguna dari aplikasi dan mengarahkan kembali ke halaman login.

```

drawer: Drawer(
  backgroundColor: Colors.grey[850],
  child: ListView(
    padding: EdgeInsets.zero,
    children: [
      DrawerHeader(

```

Kelas `'KesehatanMentalPage'` adalah turunan dari `'StatefulWidget'` yang menyediakan antarmuka untuk menampilkan data kesehatan mental.

```

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

```

Di dalamnya, kelas `_KesehatanMentalPageState` mengelola status dan tampilan halaman dengan menggunakan `Scaffold`, yang mencakup `AppBar` berjudul "DATA KESEHATAN MENTAL" dan drawer untuk logout.

```
class _KesehatanMentalPageState extends State<KesehatanMentalPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.black,
        iconTheme: const IconThemeData(color: Colors.white),
        title: const Text(
          'DATA KESEHATAN MENTAL',
          style: TextStyle(fontFamily: 'Georgia', color: Colors.white),
        ), // Text
        actions: [
          Padding(
            padding: const EdgeInsets.only(right: 20.0),
            child: GestureDetector(
              child: const Icon(Icons.add, size: 26.0, color: Colors.white),
              onTap: () {
                Navigator.push(
                  context,
```

Metode `build` membangun tampilan halaman dengan `FutureBuilder`, yang digunakan untuk mengambil data kesehatan mental secara asinkron menggunakan `KesehatanMentalBloc`. Jika data berhasil diambil, akan ditampilkan melalui widget `ListKesehatanMental`, sementara indikator pemuatan muncul jika data belum tersedia.

```
class ListKesehatanMental extends StatelessWidget {
  final List? list;
  const ListKesehatanMental({Key? key, this.list}) : super(key: key);

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

class ItemKesehatanMental extends StatelessWidget {
  final KesehatanMental data;
  const ItemKesehatanMental({Key? key, required this.data}) : super(key: key);
```

Widget `ListKesehatanMental` menggunakan `ListView.builder` untuk menampilkan daftar item, sedangkan widget `ItemKesehatanMental` menampilkan setiap item dalam bentuk `ListTile` yang dibungkus dalam `Card`. Ketika item diketuk, pengguna diarahkan ke halaman formulir untuk mengedit data, menampilkan atribut seperti `mentalState`, jumlah sesi terapi (`therapySessions`), dan pengobatan (`medication`) dengan teks alternatif jika nilai

tidak ada. Halaman ini memberikan antarmuka yang responsif untuk pengguna dalam melihat dan mengelola data kesehatan mental mereka.

4. Kesehatan mental form

Kelas `KesehatanMentalForm` adalah turunan dari `StatefulWidget` yang digunakan untuk menampilkan formulir untuk menambah atau mengubah data kesehatan mental. Kelas ini menerima parameter opsional `dataKesehatanMental` yang berisi data yang akan diedit, jika ada. Dalam kelas `_KesehatanMentalFormState`, terdapat beberapa atribut, termasuk `TextEditingController` untuk setiap field input dan `GlobalKey<FormState>` untuk mengelola validasi formulir.

```
class KesehatanMentalForm extends StatefulWidget {
  final KesehatanMental? dataKesehatanMental;

  const KesehatanMentalForm({Key? key, this.dataKesehatanMental})
    : super(key: key);

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

class _KesehatanMentalFormState extends State<KesehatanMentalForm> {
  final _formKey = GlobalKey<FormState>();
  final _mentalStateController = TextEditingController();
  final _therapySessionsController = TextEditingController();
  final _medicationController = TextEditingController();
  bool _isLoading = false;
}
```

Metode `initState` digunakan untuk menginisialisasi controller dengan data yang ada, jika terdapat data yang dioper. Metode `build` membangun tampilan halaman dengan `AppBar`, yang menampilkan judul yang berbeda tergantung pada apakah data ada atau tidak. Tampilan formulir dikelilingi oleh `SingleChildScrollView` agar pengguna dapat menggulir jika layar tidak cukup besar. Terdapat tiga field input untuk kondisi mental, sesi terapi, dan obat, yang menggunakan metode `_buildTextField`.

```
void initState() {
  super.initState();
  // Inisialisasi controller dengan data jika ada
  if (widget.dataKesehatanMental != null) {
    _mentalStateController.text =
      widget.dataKesehatanMental!.mentalState ?? "";
    _therapySessionsController.text =
      widget.dataKesehatanMental!.therapySessions?.toString() ?? "";
    _medicationController.text = widget.dataKesehatanMental!.medication ?? "";
  }
}
```

Tombol submit dan tombol hapus juga disediakan, dengan masing-masing memanggil fungsi `_onSubmit` dan `_onDelete` saat ditekan. Fungsi `_onSubmit` menangani validasi dan

pengiriman data ke backend menggunakan `KesehatanMentalBloc`, baik untuk menyimpan data baru maupun untuk memperbarui data yang ada. Jika penghapusan dipilih, fungsi `_onDelete` menampilkan dialog konfirmasi sebelum menghapus data. Jika berhasil, pengguna diarahkan kembali ke halaman `KesehatanMentalPage`. Kesalahan selama operasi akan ditangani dengan menampilkan dialog kesalahan menggunakan metode `_showErrorDialog`. Halaman ini memberikan antarmuka yang intuitif bagi pengguna untuk mengelola data kesehatan mental mereka.

```
Widget _buildSubmitButton() {
  return OutlinedButton(
    style: OutlinedButton.styleFrom(
      foregroundColor: Colors.white,
      side: const BorderSide(color: Colors.white),
      textStyle: const TextStyle(fontFamily: 'Georgia'),
    ),
    child: Text(widget.dataKesehatanMental != null ? "UBAH" : "SIMPAN"),
    onPressed: _onSubmit,
  ); // OutlinedButton
}

// Widget untuk tombol hapus
Widget _buildDeleteButton() {
  return OutlinedButton(
    style: OutlinedButton.styleFrom(
      foregroundColor: Colors.red,
      side: const BorderSide(color: Colors.red),
      textStyle: const TextStyle(fontFamily: 'Georgia'),
    ),
    child: const Text("HAPUS DATA"),
    onPressed: _onDelete,
  ); // OutlinedButton
}
```

5. Kesehatan mental detail

Kelas KesehatanMentalDetail adalah turunan dari StatefulWidget yang digunakan untuk menampilkan detail dari data kesehatan mental yang telah dipilih. Kelas ini menerima parameter data, yang merupakan objek KesehatanMental, dan menampilkan atribut-atributnya seperti ID, kondisi mental, sesi terapi, dan obat. Dalam kelas `_KesehatanMentalDetailState`, metode `build` digunakan untuk membangun tampilan halaman, yang terdiri dari AppBar dengan judul "Detail Kesehatan Mental" dan area konten yang menunjukkan detail informasi kesehatan mental.

```

class KesehatanMentalDetail extends StatefulWidget {
  final KesehatanMental? data;

  const KesehatanMentalDetail({Key? key, this.data}) : super(key: key);

  @override
  KesehatanMentalDetailState createState() => _KesehatanMentalDetailState();
}

```

Halaman ini menggunakan Column untuk menyusun informasi dengan rapi, di mana setiap detail ditampilkan menggunakan metode `_buildDetailText`, yang mengatur gaya teks dengan ukuran dan warna tertentu. Di bagian bawah halaman, terdapat dua tombol: satu untuk mengedit data dan satu lagi untuk menghapus data. Tombol edit akan mengarahkan pengguna ke halaman `KesehatanMentalForm` dengan data yang relevan untuk diedit.

Ketika tombol hapus ditekan, metode `_confirmHapus` akan dipanggil untuk menampilkan dialog konfirmasi. Jika pengguna memilih untuk menghapus, maka data akan dihapus menggunakan metode `deleteDataKesehatanMental` dari `KesehatanMentalBloc`. Jika penghapusan berhasil, pengguna akan diarahkan kembali ke halaman `KesehatanMentalPage`. Jika gagal, dialog kesalahan akan ditampilkan menggunakan `WarningDialog`.

```

Widget _tombolHapusEdit() {
  return Row(
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    children: [
      // Tombol Edit
      OutlinedButton(
        style: OutlinedButton.styleFrom(
          backgroundColor: const Color.fromARGB(255, 126, 113, 205),
        ),
        child: const Text(
          "EDIT",
          style: TextStyle(color: Colors.white, fontFamily: 'Courier New'),
        ), // Text
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) =>
                KesehatanMentalForm(dataKesehatanMental: widget.data!),
            ), // MaterialPageRoute
          );
        },
      ), // OutlinedButton
    ],
  );
}

```



```

void _confirmHapus() {
  AlertDialog alertDialog = AlertDialog(
    backgroundColor: Colors.grey[800],
    content: const Text(
      "Yakin ingin menghapus data ini?",
      style: TextStyle(fontFamily: 'Courier New', color: Colors.white),
    ), // Text
    actions: [
      // Tombol Hapus
      OutlinedButton(
        child: const Text(
          "Ya",
          style: TextStyle(
            color: Color.fromARGB(255, 232, 125, 227),
            fontFamily: 'Courier New',
          ), // TextStyle
        ), // Text
        onPressed: () {
          final id = widget.data?.id;
          if (id != null) {
            KesehatanMentalBloc.deleteDataKesehatanMental(id: id).then(
              (value) {
                if (value) {
                  Navigator.of(context).pushReplacement(
                    MaterialPageRoute(
                      builder: (context) => const KesehatanMentalPage(),
                    ), // MaterialPageRoute

```