Nama : Brian Cahya Purnama
NIM : H1D022009
Shift Lama : C
Shift Baru : D
Paket : 1
Jenis Aplikasi : Aplikasi Manajemen Kesehatan
Tabel : Rekam Medis Pasien

PENJELASAN KODE DAN TAMPILAN

1. Register
   Kode di bawah ini untuk membuat input nama, email, password, dan konfirmasi password dengan validasi masing-masing:

```
Widget _namaTextField() {
  return _buildTextField(
    label: "Nama",
    controller: _namaTextboxController,
    prefixIcon: Icons.person,
    validator: (value) {
      if (value!.length < 3) {
        return "Nama harus diisi minimal 3 karakter";
      }
      return null;
    },
  );
}

Widget _emailTextField() {
  return _buildTextField(
    label: "Email",
    controller: _emailTextboxController,
    prefixIcon: Icons.email,
    keyboardType: TextInputType.emailAddress,
    validator: (value) {
      if (value!.isEmpty) {
        return 'Email harus diisi';
      }
      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;
    },
  );
}

Widget _passwordTextField() {
  return _buildTextField(
    label: "Password",
    controller: _passwordTextboxController,
    prefixIcon: Icons.lock,
    isPassword: true,
    validator: (value) {
      if (value!.length < 6) {
```

```
                return "Password harus diisi minimal 6 karakter";
              }
              return null;
            },
          );
        }

        Widget _passwordKonfirmasiTextField() {
          return _buildTextField(
            label: "Konfirmasi Password",
            controller: null,
            prefixIcon: Icons.lock_outline,
            isPassword: true,
            validator: (value) {
              if (value != _passwordTextboxController.text) {
                return "Konfirmasi Password tidak sama";
              }
              return null;
            },
          );
}
```

Kode di bawah ini untuk memvalidasi form dan memulai proses registrasi:

```
        Widget _buttonRegistrasi() {
          return Container(
            decoration: BoxDecoration(
              gradient: LinearGradient(
                colors: [primaryGreen, accentYellow],
                begin: Alignment.centerLeft,
                end: Alignment.centerRight,
              ),
              borderRadius: BorderRadius.circular(12),
            ),
            child: ElevatedButton(
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.transparent,
                padding: const EdgeInsets.symmetric(vertical: 15),
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(12),
                ),
              ),
              child: const Text(
                "REGISTRASI",
                style: TextStyle(
                  fontSize: 16,
                  fontWeight: FontWeight.bold,
                  color: Colors.white,
                ),
              ),
              onPressed: () {
                var validate = _formKey.currentState!.validate();
                if (validate) {
                  if (!_isLoading) _submit();
                }
              },
            ),
          );
}
```

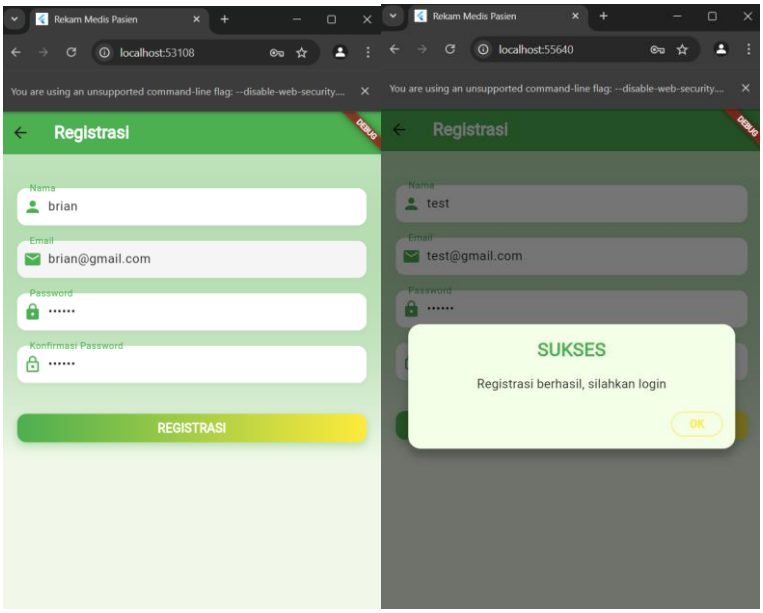
Kode di bawah ini mengirim data registrasi dan menangani hasilnya:

```
        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;
          });
}
```

Penjelasan singkat keseluruhan kode:
Potongan kode di atas untuk mengatur input data pengguna, tombol registrasi yang memvalidasi input, dan proses untuk mengirim data registrasi ke server atau backend. Hasil dari registrasi (berhasil atau gagal) ditampilkan menggunakan dialog.

Screenshoot:



2. Login
Kode di bawah ini membuat input untuk email dan password dengan validasi masing-masing:

```
        Widget _emailTextField() {
          return Container(
            decoration: BoxDecoration(
              color: Colors.white,
              borderRadius: BorderRadius.circular(12),
              boxShadow: [
                BoxShadow(
                  color: Colors.grey.withOpacity(0.1),
                  spreadRadius: 1,
```

```
                  blurRadius: 3,
                  offset: const Offset(0, 2),
                ),
              ],
            ),
            child: TextFormField(
              decoration: InputDecoration(
                labelText: "Email",
                prefixIcon: Icon(Icons.email, color: primaryGreen),
              ),
              keyboardType: TextInputType.emailAddress,
              controller: _emailTextboxController,
              validator: (value) {
                if (value!.isEmpty) {
                  return 'Email harus diisi';
                }
                return null;
              },
            ),
          );
        }

        Widget _passwordTextField() {
          return Container(
            decoration: BoxDecoration(
              color: Colors.white,
              borderRadius: BorderRadius.circular(12),
            ),
            child: TextFormField(
              decoration: InputDecoration(
                labelText: "Password",
                prefixIcon: Icon(Icons.lock, color: primaryGreen),
              ),
              obscureText: true,
              controller: _passwordTextboxController,
              validator: (value) {
                if (value!.isEmpty) {
                  return "Password harus diisi";
                }
                return null;
              },
            ),
          );
}
```

Kode ini merupakan sebuah tombol untuk memvalidasi form dan memulai proses login:

```
        Widget _buttonLogin() {
          return Container(
            decoration: BoxDecoration(
              gradient: LinearGradient(
                colors: [primaryGreen, accentYellow],
              ),
              borderRadius: BorderRadius.circular(12),
            ),
            child: ElevatedButton(
              child: const Text(
                "LOGIN",
                style: TextStyle(
                  fontSize: 16,
                  fontWeight: FontWeight.bold,
                  color: Colors.white,
                ),
              ),
              onPressed: () {
                var validate = _formKey.currentState!.validate();
                if (validate) {
                  if (!_isLoading) _submit();
                }
              },
            ),
          );
}
```

Kode ini digunakan untuk mengirim data login dan menangani hasilnya:

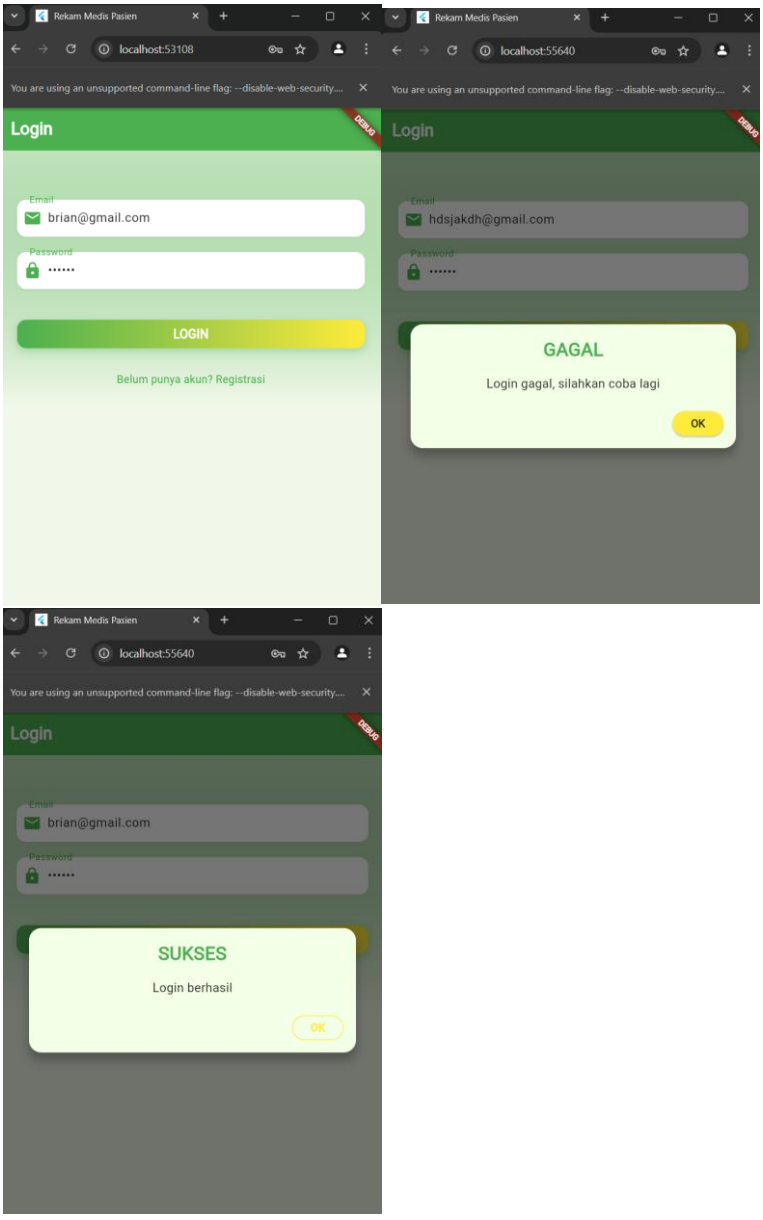
```
        void _submit() {
          _formKey.currentState!.save();
          setState(() {
            _isLoading = true;
          });

          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);

              showDialog(
                context: context,
                barrierDismissible: false,
                builder: (BuildContext context) => SuccessDialog(
                  description: "Login berhasil",
                  okClick: () {
                    Navigator.pushReplacement(
                      context,
                      MaterialPageRoute(builder: (context) => const PasienPage()),
                    );
                  },
                ),
              );
            } else {
              showDialog(
                context: context,
                barrierDismissible: false,
                builder: (BuildContext context) => const WarningDialog(
                  description: "Login gagal, silahkan coba lagi",
                ),
              );
            }
          }, onError: (error) {
            showDialog(
              context: context,
              barrierDismissible: false,
              builder: (BuildContext context) => const WarningDialog(
                description: "Login gagal, silahkan coba lagi",
              ),
            );
          });

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

Kode di bawah ini menyediakan opsi untuk pengguna yang belum memiliki akun:

```
        Widget _menuRegistrasi() {
          return Center(
            child: InkWell(
              child: Text(
                "Belum punya akun? Registrasi",
                style: TextStyle(
                  color: primaryGreen,
                  fontSize: 14,
                  fontWeight: FontWeight.w500,
                ),
              ),
              onTap: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(builder: (context) => const RegistrasiPage()),
                );
              },
            ),
          );
}
```

Penjelasan singkat keseluruhan kode:
Kode ini membuat halaman login dengan input email dan password. Tombol login memvalidasi input dan mengirim data ke server. Jika login berhasil, pengguna diarahkan ke halaman berikutnya (PasienPage) setelah menyimpan token dan ID pengguna. Jika login gagal, ditampilkan pesan peringatan. Ada juga opsi untuk menuju halaman registrasi bagi pengguna baru.

Screenshoot:



3. Tampil List Data
Kode ini menggunakan FutureBuilder untuk menampilkan daftar pasien yang diambil secara asinkron menggunakan PasienBloc.getPasiens(). Jika data berhasil diambil (snapshot.hasData), maka data tersebut akan dikirim ke widget ListPasien. Jika data belum tersedia atau sedang diproses, akan ditampilkan CircularProgressIndicator sebagai indikator loading:

```
        body: Container(
          decoration: BoxDecoration(
```

```
            gradient: LinearGradient(
              begin: Alignment.topCenter,
              end: Alignment.bottomCenter,
              colors: [
                primaryGreen.withOpacity(0.3),
                backgroundColor,
              ],
            ),
          ),
          child: FutureBuilder<List>(
            future: PasienBloc.getPasiens(),
            builder: (context, snapshot) {
              if (snapshot.hasError) print(snapshot.error);
              return snapshot.hasData
                  ? ListPasien(
                      list: snapshot.data,
                    )
                  : Center(
                      child: CircularProgressIndicator(
                        color: primaryGreen,
                      ),
                    );
            },
          ),
)
```

Pada kode di bawah ini ListView.builder digunakan untuk menampilkan data pasien dalam bentuk list. itemCount menentukan jumlah item yang ditampilkan, dan itemBuilder menghasilkan widget untuk setiap item. Untuk setiap item dalam list, widget ItemPasien digunakan untuk menampilkan data pasien tertentu:

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

          @override
          Widget build(BuildContext context) {
            return ListView.builder(
              padding: const EdgeInsets.all(16),
              itemCount: list == null ? 0 : list!.length,
              itemBuilder: (context, i) {
                return ItemPasien(
                  pasien: list![i],
                );
              },
            );
          }
}
```

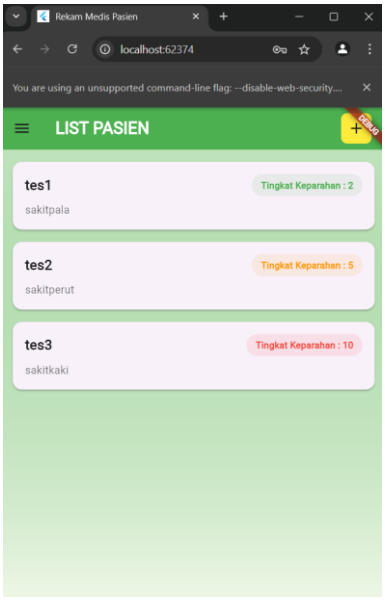
Selanjutkan, kode di bawah ini menggunkan ItemPasien merupakan widget untuk menampilkan informasi detail setiap pasien dalam bentuk Card. Data yang ditampilkan termasuk nama pasien (namapasien), tingkat keparahan (severity), dan gejala (symptom). _getSeverityColor(): Fungsi ini digunakan untuk menentukan warna berdasarkan tingkat keparahan pasien (merah untuk tinggi, oranye untuk sedang, dan hijau untuk rendah). InkWell memungkinkan setiap item pasien bisa diklik untuk navigasi ke halaman detail pasien (PasienDetail):

```
        class ItemPasien extends StatelessWidget {
          final Pasien pasien;
          const ItemPasien({Key? key, required this.pasien}) : super(key: key);

          Color _getSeverityColor(int severity) {
            if (severity >= 8) return Colors.red;
            if (severity >= 5) return Colors.orange;
            return Colors.green;
          }

          @override
          Widget build(BuildContext context) {
            return Card(
```

```
              elevation: 2,
              margin: const EdgeInsets.only(bottom: 16),
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(12),
              ),
              child: InkWell(
                borderRadius: BorderRadius.circular(12),
                onTap: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) => PasienDetail(
                        pasien: pasien,
                      ),
                    ),
                  );
                },
                child: Container(
                  padding: const EdgeInsets.all(16),
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      Row(
                        mainAxisAlignment: MainAxisAlignment.spaceBetween,
                        children: [
                          Expanded(
                            child: Text(
                              pasien.namapasien!,
                              style: const TextStyle(
                                fontSize: 18,
                                fontWeight: FontWeight.bold,
                                fontFamily: 'Sans-serif',
                              ),
                            ),
                          ),
                          Container(
                            padding: const EdgeInsets.symmetric(
                              horizontal: 12,
                              vertical: 6,
                            ),
                            decoration: BoxDecoration(
                              color:
        _getSeverityColor(pasien.severity!).withOpacity(0.1),
                              borderRadius: BorderRadius.circular(20),
                            ),
                            child: Text(
                              'Tingkat Keparahan : ${pasien.severity}',
                              style: TextStyle(
                                color: _getSeverityColor(pasien.severity!),
                                fontWeight: FontWeight.bold,
                                fontSize: 12,
                                fontFamily: 'Sans-serif',
                              ),
                            ),
                          ),
                        ],
                      ),
                      const SizedBox(height: 8),
                      Text(
                        pasien.symptom!,
                        style: const TextStyle(
                          fontSize: 14,
                          color: Colors.black54,
                          fontFamily: 'Sans-serif',
                        ),
                      ),
                    ],
                  ),
                ),
              ),
            );
          }
}
```

Penjelasan singkat keseluruhan kode:
Kode ini menampilkan halaman daftar pasien dengan memuat data pasien menggunakan FutureBuilder dan menampilkan daftar tersebut dalam bentuk ListView. Setiap pasien ditampilkan dalam Card dengan informasi detail seperti nama, gejala, dan tingkat keparahan. Pengguna bisa menambahkan pasien baru melalui tombol di AppBar dan dapat melihat detail pasien dengan mengetuk salah satu item di list.

Screenshoot:



4. Tampil Detail
Kode di bawah ini membuat tampilan detail informasi pasien seperti nama, gejala, dan tingkat keparahan, kode ini menampilkan informasi pasien dalam Card yang memberikan informasi nama, gejala, dan tingkat keparahan. Tingkat keparahan memiliki warna berbeda sesuai nilainya: merah untuk keparahan tinggi, oranye untuk sedang, dan hijau untuk rendah:

```
        body: Container(
          decoration: BoxDecoration(
            gradient: LinearGradient(
              begin: Alignment.topCenter,
              end: Alignment.bottomCenter,
              colors: [
                primaryGreen.withOpacity(0.3),
                backgroundColor,
              ],
            ),
          ),
          child: Center(
            child: Padding(
              padding: const EdgeInsets.all(16.0),
              child: Card(
                elevation: 4,
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(16),
                ),
                child: Padding(
                  padding: const EdgeInsets.all(24.0),
                  child: Column(
                    mainAxisSize: MainAxisSize.min,
                    crossAxisAlignment: CrossAxisAlignment.start,
```

```
                    children: [
                      Text(
                        "Nama    Pasien:    ${widget.pasien?.namapasien    ??    'Tidak
        diketahui'}",
                        style: const TextStyle(
                          fontSize: 20.0,
                          fontWeight: FontWeight.bold,
                          fontFamily: 'Sans-serif',
                        ),
                      ),
                      const SizedBox(height: 12),
                      Text(
                        "Gejala: ${widget.pasien?.symptom ?? 'Tidak ada data'}",
                        style: const TextStyle(
                          fontSize: 18.0,
                          color: Colors.black87,
                          fontFamily: 'Sans-serif',
                        ),
                      ),
                      const SizedBox(height: 12),
                      Text(
                        "Tingkat  Keparahan:  ${widget.pasien?.severity  ??  'Tidak  ada
        data'}",
                        style: TextStyle(
                          fontSize: 18.0,
                          color:      widget.pasien?.severity      !=      null      &&
        widget.pasien!.severity! >= 8
                              ? Colors.red
                              :      (widget.pasien?.severity      !=      null      &&
        widget.pasien!.severity! >= 5
                                  ? Colors.orange
                                  : Colors.green),
                          fontWeight: FontWeight.bold,
                          fontFamily: 'Sans-serif',
                        ),
                      ),
                      const SizedBox(height: 24),
                      _tombolHapusEdit(),
                    ],
                  ),
                ),
              ),
            ),
          ),
        ),
)
```

Kode bawah ini membuat tombol untuk mengedit data pasien dan menghapus data pasien setelah konfirmasi. Tombol "EDIT" membawa pengguna ke halaman formulir untuk mengubah data pasien, sedangkan tombol "DELETE" akan meminta konfirmasi untuk menghapus data pasien:

```
Widget _tombolHapusEdit() {
  return Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [
      // Tombol Edit
      ElevatedButton(
        style: ElevatedButton.styleFrom(
          backgroundColor: primaryGreen,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(8),
          ),
          padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 12),
        ),
        child: const Text(
          "EDIT",
          style: TextStyle(
            fontFamily: 'Sans-serif',
            color: Colors.white,
            fontWeight: FontWeight.bold,
          ),
        ),
```

```
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => PasienForm(
                pasien: widget.pasien!,
              ),
            ),
          );
        },
      ),
      // Tombol Hapus
      ElevatedButton(
        style: ElevatedButton.styleFrom(
          backgroundColor: Colors.redAccent,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(8),
          ),
          padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 12),
        ),
        child: const Text(
          "DELETE",
          style: TextStyle(
            fontFamily: 'Sans-serif',
            color: Colors.white,
            fontWeight: FontWeight.bold,
          ),
        ),
        onPressed: () => confirmHapus(),
      ),
    ],
  );
}
```

Kode bawah ini menampilkan dialog konfirmasi untuk memastikan tindakan penghapusan data, Kode ini memastikan bahwa sebelum data pasien dihapus, pengguna diberi pilihan untuk membatalkan atau melanjutkan proses. Jika pengguna memilih "Ya", data akan dihapus melalui PasienBloc. Jika berhasil, pengguna akan diberi notifikasi dan diarahkan kembali ke halaman PasienPage:

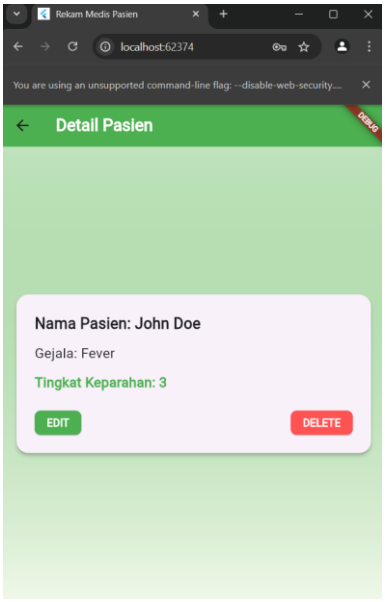
```
void confirmHapus() {
  if (widget.pasien?.id == null) {
    showDialog(
      context: context,
      builder: (BuildContext context) => const WarningDialog(
        description: "ID Pasien tidak ditemukan, tidak bisa menghapus.",
      ),
    );
    return;
  }

  AlertDialog alertDialog = AlertDialog(
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(16),
    ),
    backgroundColor: backgroundColor,
    content: const Text(
      "Yakin ingin menghapus data ini?",
      style: TextStyle(
        fontSize: 16.0,
        fontFamily: 'Sans-serif',
        fontWeight: FontWeight.w500,
        color: Colors.black87,
      ),
    ),
    actions: [
      ElevatedButton(
        style: ElevatedButton.styleFrom(
          backgroundColor: primaryGreen,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(8),
          ),
```

```
          padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 8),
        ),
        child: const Text(
          "Ya",
          style: TextStyle(
            fontFamily: 'Sans-serif',
            color: Colors.white,
            fontWeight: FontWeight.bold,
          ),
        ),
        onPressed: () async {
          bool success = await PasienBloc.deletePasien(
            id: widget.pasien!.id!,
          );
          if (success) {
            showDialog(
              context: context,
              barrierDismissible: false,
              builder: (BuildContext context) => SuccessDialog(
                description: "Pasien berhasil dihapus",
                okClick: () {
                  Navigator.of(context).pushReplacement(
                    MaterialPageRoute(
                      builder: (context) => const PasienPage(),
                    ),
                  );
                },
              ),
            );
          } else {
            showDialog(
              context: context,
              builder: (BuildContext context) => const WarningDialog(
                description: "Hapus gagal, silahkan coba lagi",
              ),
            );
          }
        },
      ),
      OutlinedButton(
        style: OutlinedButton.styleFrom(
          side: BorderSide(color: primaryGreen),
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(8),
          ),
          padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 8),
        ),
        child: Text(
          "Batal",
          style: TextStyle(
            fontFamily: 'Sans-serif',
            color: primaryGreen,
            fontWeight: FontWeight.bold,
          ),
        ),
        onPressed: () => Navigator.pop(context),
      ),
    ],
  );
  showDialog(builder: (context) => alertDialog, context: context);
}
```

Penjelasan singkat keseluruhan kode:

Kode ini membuat halaman detail untuk melihat data pasien. Halaman ini menampilkan nama pasien, gejala, dan tingkat keparahan dengan warna yang berbeda sesuai nilai keparahan. Terdapat dua tombol: "EDIT" untuk mengubah data pasien dan "DELETE" untuk menghapus data. Sebelum menghapus, pengguna diminta konfirmasi, dan jika data berhasil dihapus, pengguna akan menerima notifikasi dan kembali ke halaman daftar pasien. Jika terjadi kesalahan saat menghapus, pesan peringatan ditampilkan.

Screenshoot:



5. Tambah Data

Kode bawah ini memungkinkan pengguna untuk menambahkan data pasien baru dengan mengisi formulir yang berisi nama pasien, gejala, dan tingkat keparahan. Tombol "SIMPAN" akan muncul ketika pengguna berada di mode tambah data, dan setelah pengisian selesai, data akan disimpan ke dalam sistem. Nama Pasien: Untuk memasukkan nama pasien, dengan validasi bahwa field ini harus diisi. Gejala: Untuk memasukkan gejala yang dialami pasien, juga dengan validasi bahwa field ini tidak boleh kosong. Tingkat Keparahan: Input untuk memasukkan tingkat keparahan dalam bentuk angka, yang juga wajib diisi:

```
        body: Container(
          decoration: BoxDecoration(
            gradient: LinearGradient(
              begin: Alignment.topCenter,
              end: Alignment.bottomCenter,
              colors: [
                primaryGreen.withOpacity(0.3),
                backgroundColor,
              ],
            ),
          ),
          child: SingleChildScrollView(
            child: Padding(
              padding: const EdgeInsets.all(24.0),
              child: Form(
                key: _formKey,
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.stretch,
                  children: [
                    const SizedBox(height: 20),
                    _buildTextField(
                      label: "Nama Pasien",
                      controller: _patientNameTextboxController,
                      prefixIcon: Icons.person,
                      validator: (value) {
                        if (value!.isEmpty) {
                          return "Nama Pasien harus diisi";
                        }
```

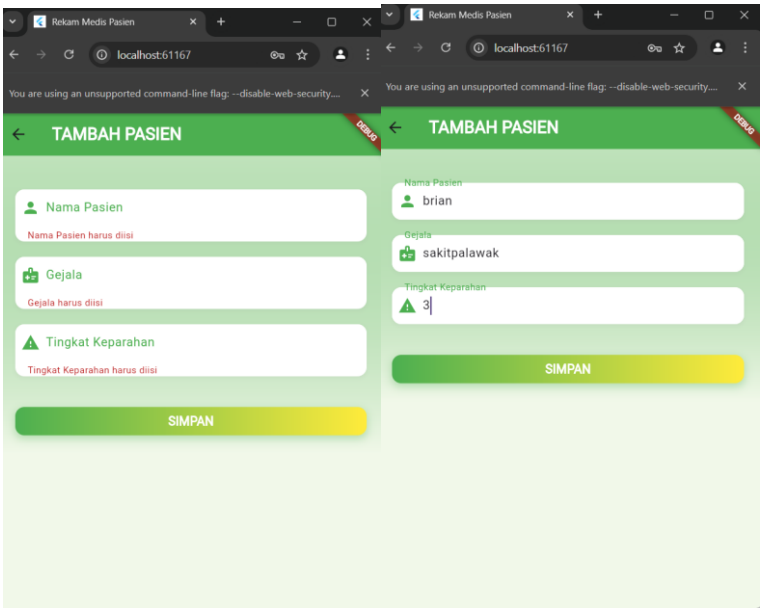
```
                        return null;
                      },
                    ),
                    const SizedBox(height: 20),
                    _buildTextField(
                      label: "Gejala",
                      controller: _symptomTextboxController,
                      prefixIcon: Icons.medical_information,
                      validator: (value) {
                        if (value!.isEmpty) {
                          return "Gejala harus diisi";
                        }
                        return null;
                      },
                    ),
                    const SizedBox(height: 20),
                    _buildTextField(
                      label: "Tingkat Keparahan",
                      controller: _severityTextboxController,
                      prefixIcon: Icons.warning_outlined,
                      keyboardType: TextInputType.number,
                      validator: (value) {
                        if (value!.isEmpty) {
                          return "Tingkat Keparahan harus diisi";
                        }
                        return null;
                      },
                    ),
                    const SizedBox(height: 40),
                    _buttonSubmit(),
                  ],
                ),
              ),
            ),
          ),
        ),
),
```

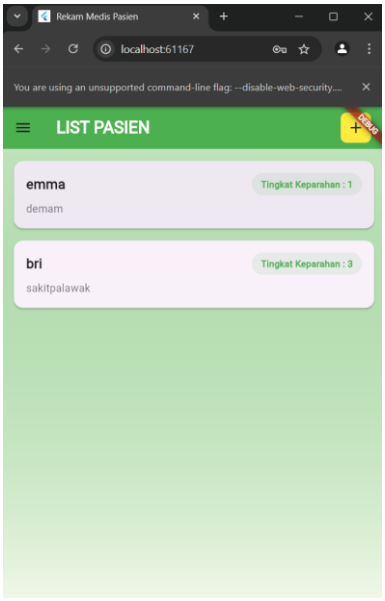
Setelah semua field valid dan tombol "SIMPAN" ditekan, kode berikut ini berisi fungsi simpan() dipanggil untuk menyimpan data baru ke sistem. Pada fungsi ini, data pasien baru (seperti nama, gejala, dan tingkat keparahan) diambil dari form, kemudian data tersebut dikirim ke PasienBloc.addPasien untuk ditambahkan ke dalam sistem. Jika berhasil, pengguna akan diarahkan kembali ke halaman daftar pasien (PasienPage). Jika terjadi kesalahan, dialog peringatan akan muncul, memberitahu bahwa proses simpan gagal:

```
        void simpan() {
          setState(() {
            _isLoading = true;
          });
          Pasien createPasien = Pasien(id: null);
          createPasien.namapasien = _patientNameTextboxController.text;
          createPasien.symptom = _symptomTextboxController.text;
          createPasien.severity = int.parse(_severityTextboxController.text);
          PasienBloc.addPasien(pasien: createPasien).then((value) {
            Navigator.of(context).push(MaterialPageRoute(
                builder: (BuildContext context) => const PasienPage()));
          }, onError: (error) {
            showDialog(
                context: context,
                builder: (BuildContext context) => const WarningDialog(
                      description: "Simpan gagal, silahkan coba lagi",
                    ));
          });
          setState(() {
            _isLoading = false;
          });
}
```

Penjelasan singkat keseluruhan kode:
Kode ini digunakan untuk menambahkan pasien baru. Terdapat form dengan tiga input: nama pasien, gejala, dan tingkat keparahan. Setelah form diisi, tombol "SIMPAN" memungkinkan data untuk dikirim ke backend. Jika berhasil, data pasien disimpan, dan pengguna diarahkan ke halaman daftar pasien. Jika gagal, dialog peringatan muncul untuk memberi tahu bahwa penyimpanan data tidak berhasil.

Screenshoot:





6. Ubah Data
Kode berikut ini memungkinkan pengguna untuk mengubah data pasien yang sudah ada. Saat mode ubah diaktifkan, formulir yang sama seperti pada mode tambah akan ditampilkan, tetapi dengan data pasien yang sudah diisi sebelumnya. Tombol "UBAH" akan muncul sebagai pengganti tombol "SIMPAN". Kode ini berfungsi untuk menentukan apakah halaman yang dibuka merupakan mode tambah atau ubah. Jika pasien sudah ada (melalui parameter widget.pasien), maka field pada formulir akan diisi

dengan data pasien tersebut. Judul halaman berubah menjadi "UBAH PASIEN", dan tombol akan berubah menjadi "UBAH". Jika tidak ada data pasien yang diberikan, mode akan menjadi tambah data:

```
        void isUpdate() {
          if (widget.pasien != null) {
            setState(() {
              judul = "UBAH PASIEN";
              tombolSubmit = "UBAH";
              _patientNameTextboxController.text = widget.pasien!.namapasien!;
              _symptomTextboxController.text = widget.pasien!.symptom!;
              _severityTextboxController.text = widget.pasien!.severity.toString();
            });
          } else {
            judul = "TAMBAH PASIEN";
            tombolSubmit = "SIMPAN";
          }
}
```

Kode bawah ini adalah tampilan formulir untuk mengubah data. Formulir ini berfungsi sama seperti pada mode tambah, namun dengan data pasien yang sudah ada, sehingga pengguna bisa mengubah informasi yang diperlukan:

```
body: Container(
  decoration: BoxDecoration(
    gradient: LinearGradient(
      begin: Alignment.topCenter,
      end: Alignment.bottomCenter,
      colors: [
        primaryGreen.withOpacity(0.3),
        backgroundColor,
      ],
    ),
  ),
  child: SingleChildScrollView(
    child: Padding(
      padding: const EdgeInsets.all(24.0),
      child: Form(
        key: _formKey,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            const SizedBox(height: 20),
            _buildTextField(
              label: "Nama Pasien",
              controller: _patientNameTextboxController,
              prefixIcon: Icons.person,
              validator: (value) {
                if (value!.isEmpty) {
                  return "Nama Pasien harus diisi";
                }
                return null;
              },
            ),
            const SizedBox(height: 20),
            _buildTextField(
              label: "Gejala",
              controller: _symptomTextboxController,
              prefixIcon: Icons.medical_information,
              validator: (value) {
                if (value!.isEmpty) {
                  return "Gejala harus diisi";
                }
                return null;
              },
            ),
            const SizedBox(height: 20),
            _buildTextField(
              label: "Tingkat Keparahan",
              controller: _severityTextboxController,
              prefixIcon: Icons.warning_outlined,
              keyboardType: TextInputType.number,
```

```
              validator: (value) {
                if (value!.isEmpty) {
                  return "Tingkat Keparahan harus diisi";
                }
                return null;
              },
            ),
            const SizedBox(height: 40),
            _buttonSubmit(),
          ],
        ),
      ),
    ),
  ),
),
```

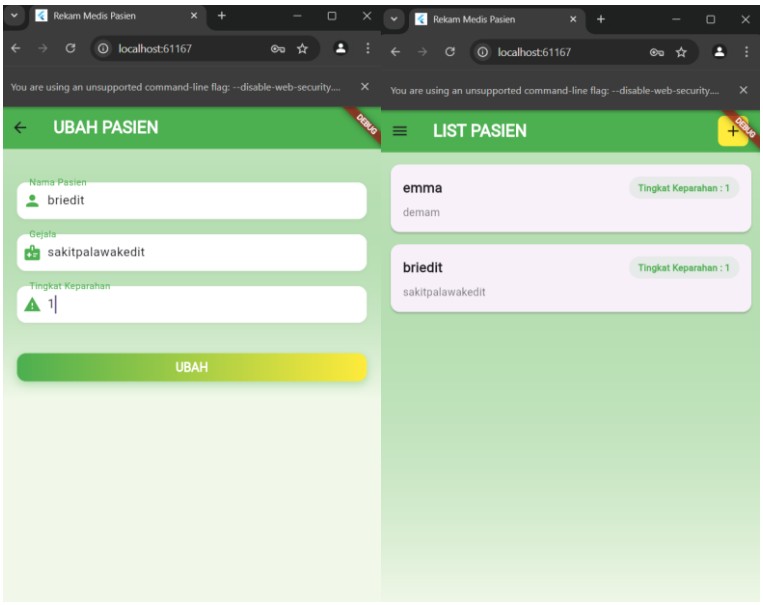
KodeBerikut bawah ini adalah fungsi yang menangani proses ubah data. Pada fungsi ubah(), data pasien yang baru diisi dalam form diambil dan digunakan untuk memperbarui data yang ada melalui PasienBloc.updatePasien. Proses pembaruan ini menggunakan id pasien yang ada untuk memastikan data yang diubah adalah data pasien yang benar. Jika berhasil, pengguna akan diarahkan kembali ke halaman daftar pasien (PasienPage). Jika terjadi kesalahan, dialog peringatan akan muncul untuk memberi tahu bahwa proses ubah data gagal:

```
void ubah() {
  setState(() {
    _isLoading = true;
  });
  Pasien updatePasien = Pasien(id: widget.pasien!.id!);
  updatePasien.namapasien = _patientNameTextboxController.text;
  updatePasien.symptom = _symptomTextboxController.text;
  updatePasien.severity = int.parse(_severityTextboxController.text);
  PasienBloc.updatePasien(pasien: updatePasien).then((value) {
    Navigator.of(context).push(MaterialPageRoute(
        builder: (BuildContext context) => const PasienPage()));
  }, onError: (error) {
    showDialog(
        context: context,
        builder: (BuildContext context) => const WarningDialog(
              description: "Permintaan ubah data gagal, silahkan coba lagi",
            ));
  });
  setState(() {
    _isLoading = false;
  });
}
```

Penjelasan singkat keseluruhan kode:

Kode ini digunakan untuk mengubah data pasien yang sudah ada. Halaman formulir diisi dengan data pasien yang akan diubah. Setelah pengguna mengedit data dan menekan tombol "UBAH", data pasien akan diperbarui di sistem. Jika berhasil, pengguna akan diarahkan kembali ke halaman daftar pasien. Jika gagal, dialog peringatan akan muncul untuk memberi tahu bahwa proses pengubahan data tidak berhasil.

Screenshoot:



7. Hapus Data

Kode di bawah ini digunakan untuk melakukan konfirmasi penghapusan data pasien. Kode melalui beberapa tahap yaitu, pertama Pengecekan ID Pasien: Kode memulai dengan memeriksa apakah widget.pasien?.id valid. Jika ID pasien tidak ditemukan (null), pesan peringatan ditampilkan menggunakan WarningDialog dan proses penghapusan dihentikan. Kedua, Konfirmasi Penghapusan: Jika ID pasien valid, sebuah dialog (AlertDialog) akan muncul, menanyakan apakah pengguna yakin ingin menghapus data pasien. Terdapat dua tombol aksi: Tombol "Ya": Jika ditekan, akan memanggil fungsi PasienBloc.deletePasien untuk menghapus pasien dari database berdasarkan ID. Jika berhasil, sebuah SuccessDialog muncul dengan pesan sukses, dan setelah itu pengguna diarahkan kembali ke halaman PasienPage. Tombol "Batal": Menutup dialog dan membatalkan penghapusan. Ketiga, Penghapusan Gagal: Jika penghapusan pasien gagal, pesan peringatan ditampilkan menggunakan WarningDialog:

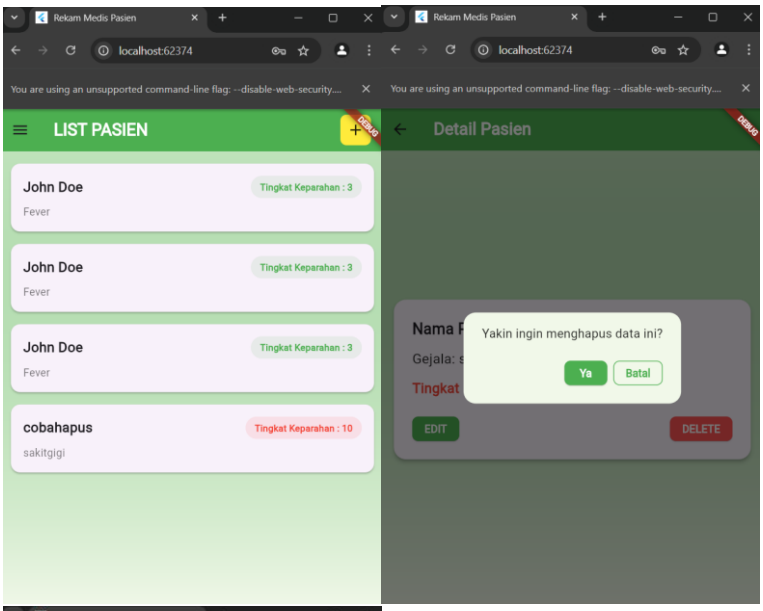
```
        void confirmHapus() {
          if (widget.pasien?.id == null) {
            showDialog(
              context: context,
              builder: (BuildContext context) => const WarningDialog(
                description: "ID Pasien tidak ditemukan, tidak bisa menghapus.",
              ),
            );
            return;
          }

          AlertDialog alertDialog = AlertDialog(
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(16),
            ),
            backgroundColor: backgroundColor,
            content: const Text(
              "Yakin ingin menghapus data ini?",
              style: TextStyle(
                fontSize: 16.0,
                fontFamily: 'Sans-serif',
                fontWeight: FontWeight.w500,
                color: Colors.black87,
              ),
            ),
```

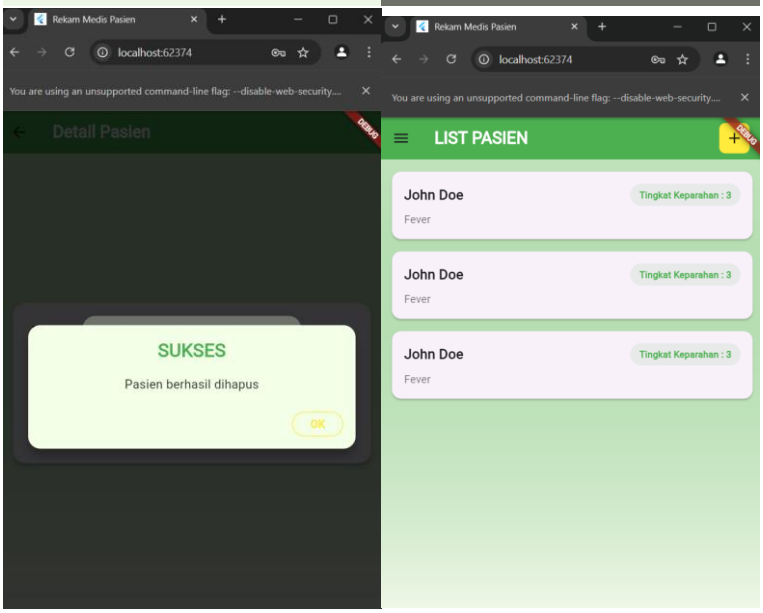
```
            actions: [
              ElevatedButton(
                style: ElevatedButton.styleFrom(
                  backgroundColor: primaryGreen,
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(8),
                  ),
                  padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 8),
                ),
                child: const Text(
                  "Ya",
                  style: TextStyle(
                    fontFamily: 'Sans-serif',
                    color: Colors.white,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                onPressed: () async {
                  bool success = await PasienBloc.deletePasien(
                    id: widget.pasien!.id!,
                  );
                  if (success) {
                    showDialog(
                      context: context,
                      barrierDismissible: false,
                      builder: (BuildContext context) => SuccessDialog(
                        description: "Pasien berhasil dihapus",
                        okClick: () {
                          Navigator.of(context).pushReplacement(
                            MaterialPageRoute(
                              builder: (context) => const PasienPage(),
                            ),
                          );
                        },
                      ),
                    );
                  } else {
                    showDialog(
                      context: context,
                      builder: (BuildContext context) => const WarningDialog(
                        description: "Hapus gagal, silahkan coba lagi",
                      ),
                    );
                  }
                },
              ),
              OutlinedButton(
                style: OutlinedButton.styleFrom(
                  side: BorderSide(color: primaryGreen),
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(8),
                  ),
                  padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 8),
                ),
                child: Text(
                  "Batal",
                  style: TextStyle(
                    fontFamily: 'Sans-serif',
                    color: primaryGreen,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                onPressed: () => Navigator.pop(context),
              ),
            ],
          );
          showDialog(builder: (context) => alertDialog, context: context);
}
```

Penjelasan singkat keseluruhan kode:
Kode ini menangani proses penghapusan data pasien. Ketika pengguna menekan tombol "DELETE", dialog konfirmasi muncul. Jika pasien berhasil dihapus, pengguna

diberi notifikasi, dan diarahkan kembali ke halaman utama. Jika penghapusan gagal, pesan peringatan ditampilkan.

Screenshoot:





8. Logout

Kode di bawah ini mendefinisikan tombol Logout di dalam widget ListTile, yang akan ditampilkan di dalam Drawer (menu samping). Icon dan Label: Bagian ini menggunakan Icon(Icons.logout) dan menampilkan teks "Logout". Fungsi onTap: Saat tombol ini ditekan, fungsi onTap akan dipanggil:

```
        ListTile(
          leading: Icon(Icons.logout, color: primaryGreen),
          title: Text(
            'Logout',
            style: TextStyle(
              color: primaryGreen,
              fontFamily: 'Sans-serif',
              fontWeight: FontWeight.w500,
            ),
          ),
```

```
          onTap: () async {
            await LogoutBloc.logout().then((value) => {
              Navigator.of(context).pushAndRemoveUntil(
                MaterialPageRoute(builder: (context) => LoginPage()),
                (route) => false
              )
            });
          },
)
```

Kode di bawah ini mempunyai onTap yang melakukan proses logout dengan langkah-langkah berikut. Pertama, Memanggil LogoutBloc.logout(): Fungsi logout() dari LogoutBloc dipanggil untuk menangani proses logout dari server atau menghapus data pengguna yang tersimpan secara lokal. Kedua, Menangani Hasil Logout: Setelah fungsi logout() selesai dieksekusi (mengembalikan hasil melalui then), pengguna akan diarahkan ke halaman login (LoginPage) menggunakan Navigator.pushAndRemoveUntil. Terakhir, pushAndRemoveUntil: Menghapus semua halaman sebelumnya dari tumpukan halaman dan mengganti dengan halaman login. Ini memastikan pengguna tidak bisa kembali ke halaman sebelumnya setelah logout.

```
        await LogoutBloc.logout().then((value) => {
          Navigator.of(context).pushAndRemoveUntil(
            MaterialPageRoute(builder: (context) => LoginPage()),
            (route) => false
          )
});
```

Setelah proses logout selesai, pengguna diarahkan ke halaman login, kode bawah ini menutup semua halaman yang terbuka dan menggantinya dengan LoginPage, mencegah pengguna kembali ke halaman sebelumnya dengan tombol "back".

```
        Navigator.of(context).pushAndRemoveUntil(
          MaterialPageRoute(builder: (context) => LoginPage()),
          (route) => false
)
```

Penjelasan singkat keseluruhan kode:
Tombol Logout yang ada di dalam Drawer menampilkan ikon dan teks logout. Ketika tombol ditekan, fungsi LogoutBloc.logout() dipanggil untuk melakukan logout dari sistem. Setelah logout berhasil, pengguna diarahkan ke halaman login (LoginPage), dan semua halaman sebelumnya dihapus dari tumpukan navigasi, sehingga pengguna tidak bisa kembali ke halaman sebelumnya setelah keluar.

Screenshoot: