

# Proyek 4 - Modul 2

## Authentication & Navigation



[https://github.com/dzhax499/PY4\\_2C\\_D3\\_2024\\_Modul\\_071/tree/main/MODUL\\_2/Logbook\\_app\\_071](https://github.com/dzhax499/PY4_2C_D3_2024_Modul_071/tree/main/MODUL_2/Logbook_app_071)

071 – Dzakir Tsabit – 17/02/26

## Daftar Isi

MODUL 2: Authentication & Navigation.....	4
Tujuan Pembelajaran.....	4
Materi Esensial .....	4
Target Milestones .....	4
Materi untuk Praktikum .....	5
2.1 Persiapan: Clean Code & Arsitektur Berkas (Modular).....	5
Langkah: Membuat Struktur Folder Fitur .....	5
Struktur Folder Akhir: .....	5
2.2 Memahami Alamat Berkas (Path).....	5
Cara Membaca Alamat (Package Path).....	6
Organisasi Import .....	6
2.3 Prinsip SOLID dalam Modul Ini .....	6
2.4 Langkah-Langkah Detail Praktikum .....	6
Langkah 1: Onboarding .....	6
Langkah 2: Menyiapkan "Peta" Navigasi di main.dart .....	6
Langkah 3: Membuat Si "Otak" Keamanan (login_controller.dart) .....	7
Langkah 4: Membuat Si "Wajah" Gerbang (login_view.dart) .....	8
Langkah 5: Passing Data (Login to Counter).....	9
Langkah 6: The Logout Feature (Returning to Gate).....	10
Troubleshooting Lab.....	13
Tugas Praktikum.....	13
Task 1: Tugas Pendahuluan .....	13
Task 2: The Login Portal (LOTS) .....	13
Task 3: Persistent History Logger (HOTS).....	14
Hints: Memahami Shared Preferences .....	14
Snippets: Implementasi Data Persistence.....	14
Catatan Penting .....	15
Homework Cosmetic & UX Enhancement (30%).....	15
Penilaian .....	15
📝 Kuesioner Self-Assessment.....	15
🤝 Peer Assessment (Apresiasi Rekan Sejawat 1).....	17
🤝 Peer Assessment (Apresiasi Rekan Sejawat 2).....	18
🤝 Peer Assessment (Apresiasi Rekan Sejawat 3).....	19
🧠 Template Lesson Learnt (Refleksi Akhir) .....	20

 Log LLM: The Fact Check & Twist (Homework) .....	21
 Contoh Cara Bertanya yang Benar (Prompting) .....	21
Rubrik Penilaian Dosen Manajer .....	22
 Rubrik Penilaian Log LLM (Integritas).....	22
Referensi .....	23

# MODUL 2: Authentication & Navigation

Selamat datang di tahap selanjutnya! Jika pada modul pertama kita telah berhasil membangun fondasi aplikasi, di modul kedua ini kita akan bertindak sebagai Gatekeeper (Penjaga Gerbang). Anda akan belajar bagaimana mengamankan aplikasi melalui fitur Authentication dan mengatur lalu lintas antar halaman menggunakan Navigation.

Kita tidak hanya sekadar membuat fitur login, tetapi juga akan mulai menerapkan Arsitektur Modular dan prinsip Clean Code untuk memastikan kode kita rapi, terorganisir, dan profesional. Kita juga akan memperluas cakrawala aplikasi dengan fitur Onboarding, Multi-Page Navigation, dan Session Management. Anda akan belajar bagaimana mengalirkan data (Username) antar halaman dan bagaimana mengelola pintu keluar (Logout) secara aman.

## Tujuan Pembelajaran

Setelah menyelesaikan modul ini, diharapkan Anda mampu:

1. Mahasiswa mampu menghubungkan modul baru (Login) dengan modul yang sudah ada (Counter).
2. Mahasiswa memahami pentingnya BuildContext dalam navigasi.
3. Mahasiswa mampu mengimplementasikan sistem keamanan pintu masuk (Authentication) sebelum mengakses fitur utama.

## Materi Esensial

Sebelum kita mulai menulis kode, pahami dahulu dua pilar utama hari ini:

1. The Concept of BuildContext: Mengapa navigasi butuh "peta" lokasi.
2. Navigation Stack: Memahami konsep tumpukan halaman (Push vs Pop).
3. Code Reusability: Memanfaatkan CounterView dari Minggu ke-1 sebagai halaman tujuan (Home).

## Target Milestones

Sebelum pulang dari lab hari ini, pastikan kalian sudah mencentang semua daftar di bawah ini:

TABEL 2.1: Daftar Target Milestones Modul 2 dan Indikator Keberhasilan.

Milestone	Indikator Keberhasilan
Integration	Proyek Minggu 1 (Counter) berhasil dipanggil oleh Modul Login.
Auth Guard	Berhasil memvalidasi akun sebelum mengizinkan masuk ke Logbook.
Context Master	Mampu menjelaskan dan memperbaiki error context pada navigasi.
UX Flow	Menggunakan pushReplacement agar alur kembali (back) terkunci.

# Materi untuk Praktikum

Pada bagian ini, kita tidak akan membuat proyek baru. Kita akan memodifikasi proyek logbook\_app dari Minggu 1. Tujuannya adalah menambahkan "Pintu Gerbang" berupa halaman Login sebelum masuk ke halaman Counter (LogBook).

## 2.1 Persiapan: Clean Code & Arsitektur Berkas (Modular)

Sebelum mulai menulis kode, kita akan melakukan "bedah rumah". Dalam proyek skala besar, menyimpan semua file di satu folder lib adalah bencana. Kita akan menerapkan Modular Folder Structure.

### Langkah: Membuat Struktur Folder Fitur

Buka VS Code, lalu buatlah struktur folder di dalam lib seperti berikut:

1. Buat folder features di dalam lib.
2. Di dalam features, buat folder auth (untuk semua hal terkait Login).
3. Di dalam features, buat folder logbook (untuk fitur utama kita dari Minggu 1).
4. Pindahkan file Minggu 1 (counter\_controller.dart dan counter\_view.dart) ke dalam folder logbook.
5. Buat file baru di dalam folder auth: login\_view.dart dan login\_controller.dart.

### Struktur Folder Akhir:

```
lib/
├── features/
│   ├── onboarding/
│   │   └── onboarding_view.dart
│   ├── auth/
│   │   ├── login_controller.dart
│   │   └── login_view.dart
│   └── logbook/
│       ├── counter_controller.dart
│       └── counter_view.dart
└── main.dart
```

## 2.2 Memahami Alamat Berkas (Path)

Setelah Anda memindahkan file ke dalam struktur folder features/, Anda akan melihat beberapa baris kode menjadi merah (error). Hal ini terjadi karena "alamat" yang lama sudah tidak berlaku. Di Flutter (Dart), ada dua cara untuk memanggil file lain:

1. Relative Path (Alamat Relatif)

Cara ini menggunakan titik sebagai acuan posisi file Anda saat ini.

- ./ : Berarti di folder yang sama.
- ../ : Berarti naik satu tingkat ke folder di atasnya.

Contoh Kasus: Jika Anda berada di login\_view.dart dan ingin memanggil counter\_view.dart: import '../features/logbook/counter\_view.dart'; (Artinya: Naik dua kali ke folder lib, lalu masuk ke features/logbook/)

## 2. Package Path (Alamat Paket/Absolut)

Cara ini adalah Best Practice yang direkomendasikan. Kita memanggil file mulai dari nama proyek kita, tidak peduli di mana posisi file kita sekarang.

Contoh Kasus: import 'package:logbook\_app/features/logbook/counter\_view.dart';

## Cara Membaca Alamat (Package Path)

Rumus Menulis Import: import 'package:nama\_proyek/folder/subfolder/nama\_file.dart';

1. package: : Protokol wajib untuk memanggil paket.
2. logbook\_app : Ini adalah nama proyek Bapak (sesuaikan dengan nama yang ada di file pubspec.yaml).
3. features/auth/ : Jalur folder tempat file disimpan.
4. login\_view.dart : Nama file tujuan.

## Organisasi Import

Selalu rapihkan import dengan urutan:

- Library Flutter/Dart: (Contoh: package:flutter/material.dart)
- Package Pihak Ketiga: (Jika ada)
- File Internal Proyek: (Contoh: package:logbook\_app/features/auth/login\_controller.dart)

## 2.3 Prinsip SOLID dalam Modul Ini

1. S - Single Responsibility Principle (SRP): \* LoginView: Hanya bertanggung jawab menampilkan form dan menerima input.
  - LoginController: Hanya bertanggung jawab memvalidasi kredensial (username/password).
2. Encapsulation (Modular): Setiap fitur (Auth, Logbook) terisolasi di foldernya masing-masing. Jika ada error di bagian Login, kita tahu persis harus mencari di folder features/auth.
3. O - Open-Closed Principle (OCP): \* Struktur Controller kita buat siap untuk dikembangkan (misal: menambah daftar user) tanpa harus mengubah fungsi utama navigasi di View.

## 2.4 Langkah-Langkah Detail Praktikum

### Langkah 1: Onboarding

- Konsep: Mahasiswa membuat OnboardingView yang memiliki sebuah variabel int step = 1.
- Logika: Jika tombol "Next" ditekan, step++. Jika step > 3, maka pindah ke LoginView menggunakan Navigator.pushReplacement.
- Tujuan: Memahami state management sederhana di satu halaman sebelum pindah halaman.

### Langkah 2: Menyiapkan "Peta" Navigasi di main.dart

Buka file lib/main.dart. Kita perlu mengubah halaman yang pertama kali muncul (initial route).

```

// main.dart
import 'package:flutter/material.dart';
// Sesuaikan path import dengan struktur folder baru
import 'package:logbook_app_001/features/onboarding/onboarding_view.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'LogBook App',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.indigo),
      ),
      home: const OnboardingView(),
    );
  }
}

```

### Langkah 3: Membuat Si "Otak" Keamanan (login\_controller.dart)

Buat file baru bernama lib/login\_controller.dart. Controller ini fokus pada pengecekan data.

```

// login_controller.dart
class LoginController {
  // Database sederhana (Hardcoded)
  final String _validUsername = "admin";
  final String _validPassword = "123";

  // Fungsi pengecekan (Logic-Only)
  // Fungsi ini mengembalikan true jika cocok, false jika salah.
  bool login(String username, String password) {
    if (username == _validUsername && password == _validPassword) {
      return true;
    }
    return false;
  }
}

```

## Langkah 4: Membuat Si "Wajah" Gerbang (login\_view.dart)

Buat file baru bernama lib/features/auth/login\_view.dart. Perhatikan penggunaan TextEditingController untuk mengambil input.

```
// login_view.dart
import 'package:flutter/material.dart';
// Import Controller milik sendiri (masih satu folder)
import 'package:logbook_app_001/features/auth/login_controller.dart';
// Import View dari fitur lain (Logbook) untuk navigasi
import 'package:logbook_app_001/features/logbook/counter_view.dart';

class LoginView extends StatefulWidget {
    const LoginView({super.key});
    @override
    State<LoginView> createState() => _LoginViewState();
}

class _LoginViewState extends State<LoginView> {
    // Inisialisasi Otak dan Controller Input
    final LoginController _controller = LoginController();
    final TextEditingController _userController = TextEditingController();
    final TextEditingController _passController = TextEditingController();

    void _handleLogin() {
        String user = _userController.text;
        String pass = _passController.text;

        bool isSuccess = _controller.login(user, pass);

        if (isSuccess) {
            Navigator.pushReplacement(
                context,
                MaterialPageRoute(
                    // Di sini kita kirimkan variabel 'user' ke parameter 'username'
                    builder: (context) => CounterView(
                        username: user),
                ),
            );
        } else {
            ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(content: Text("Login Gagal! Gunakan admin/123")),
            );
        }
    }

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

```

    appBar: AppBar(title: const Text("Login Gatekeeper")),
    body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
            children: [
                TextField(
                    controller: _userController,
                    decoration: const InputDecoration(labelText: "Username"),
                ),
                TextField(
                    controller: _passController,
                    obscureText: true, // Menyembunyikan teks password
                    decoration: const InputDecoration(labelText: "Password"),
                ),
                const SizedBox(height: 20),
                ElevatedButton(onPressed: _handleLogin, child: const
Text("Masuk")),
            ],
        ),
    );
}
}

```

## Langkah 5: Passing Data (Login to Counter)

- Update: Saat Login berhasil, LoginView akan mengirimkan String username ke CounterView.
- Syntax: MaterialPageRoute(builder: (context) => CounterView(username: user))
- Update file counter\_view.dart

```

import 'package:flutter/material.dart';
import 'package:logbook_app_001/features/logbook/counter_controller.dart';

class CounterView extends StatefulWidget {
    // Tambahkan variabel final untuk menampung nama
    final String username;

    // Update Constructor agar mewajibkan (required) kiriman nama
    const CounterView({super.key, required this.username});

    @override
    State<CounterView> createState() => _CounterViewState();
}

class _CounterViewState extends State<CounterView> {
    final CounterController _controller = CounterController();

    @override
    Widget build(BuildContext context) {

```

```

        return Scaffold(
            appBar: AppBar(
                // Gunakan widget.username untuk menampilkan data dari kelas utama
                title: Text("Logbook: ${widget.username}"),
                backgroundColor: Theme.of(context).colorScheme.inversePrimary,
                actions: [
                    // Kita siapkan tombol logout di sini untuk Fase 3 nanti
                    IconButton(
                        icon: const Icon(Icons.logout),
                        onPressed: () {
                            // Logika logout nanti di Fase 3
                        },
                    ),
                ],
            ),
            body: Center(
                child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                        Text("Selamat Datang, ${widget.username}!"),
                        const SizedBox(height: 10),
                        const Text("Total Hitungan Anda:"),
                        Text(
                            '${_controller.value}',
                            style: Theme.of(context).textTheme.headlineLarge,
                        ),
                    ],
                ),
            ),
            floatingActionButton: FloatingActionButton(
                onPressed: () => setState(() => _controller.increment()),
                child: const Icon(Icons.add),
            ),
        );
    );
}

```

## Langkah 6: The Logout Feature (Returning to Gate)

- Update di CounterView: Menambahkan tombol "Logout" di AppBar (menggunakan properti actions).
- Logika: Menggunakan Navigator.pushAndRemoveUntil. Ini materi penting agar mahasiswa tahu cara menghapus semua tumpukan halaman (stack) sehingga user tidak bisa memencet tombol "Back" HP untuk masuk lagi tanpa login.

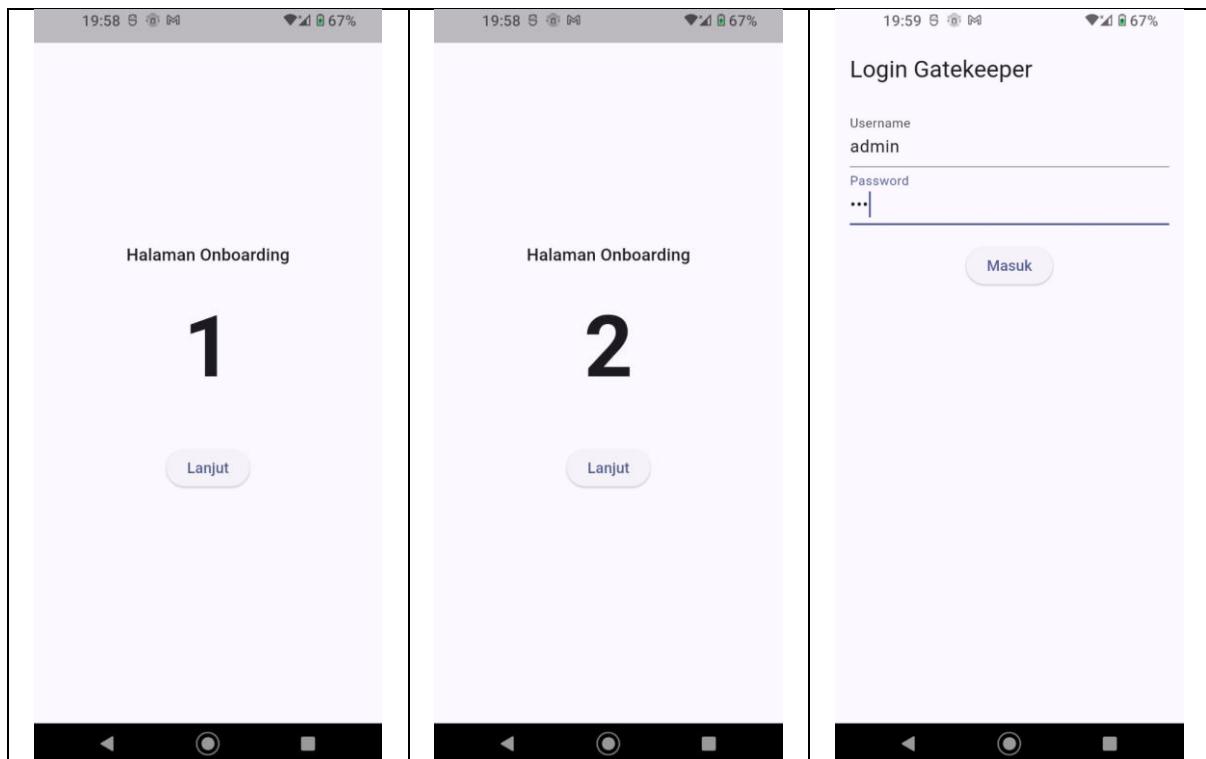
```

// Di dalam AppBar -> actions: [...]
IconButton(
    icon: const Icon(Icons.logout),

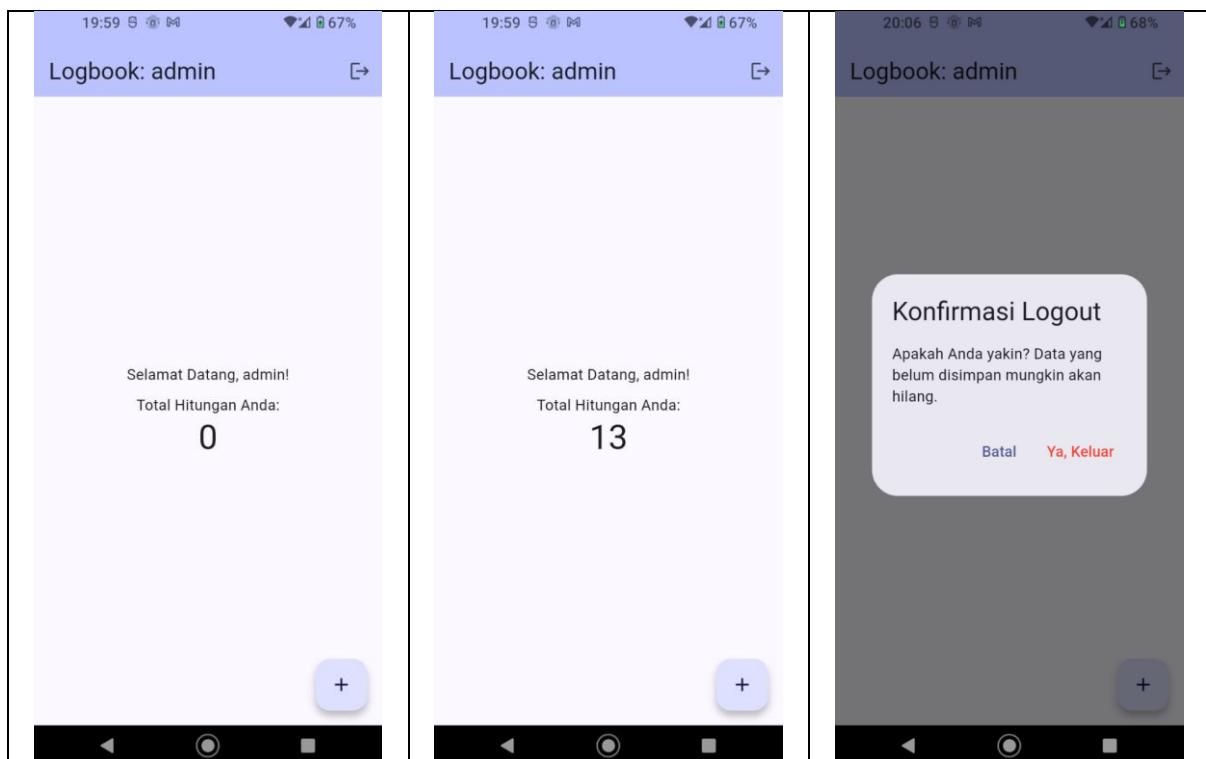
```

```
 onPressed: () {
    // 1. Munculkan Dialog Konfirmasi
    showDialog(
        context: context,
        builder: (BuildContext context) {
            return AlertDialog(
                title: const Text("Konfirmasi Logout"),
                content: const Text("Apakah Anda yakin? Data yang belum disimpan mungkin akan hilang."),
                actions: [
                    // Tombol Batal
                    TextButton(
                        onPressed: () => Navigator.pop(context), // Menutup dialog saja
                        child: const Text("Batal"),
                    ),
                    // Tombol Ya, Logout
                    TextButton(
                        onPressed: () {
                            // Menutup dialog
                            Navigator.pop(context);

                            // 2. Navigasi kembali ke Onboarding (Membersihkan Stack)
                            Navigator.pushAndRemoveUntil(
                                context,
                                MaterialPageRoute(builder: (context) => const OnboardingView()),
                                (route) => false,
                            );
                        },
                        child: const Text("Ya, Keluar", style: TextStyle(color: Colors.red)),
                    ),
                ],
            );
        },
    );
},
```



**GAMBAR 1.1:** Penampakan aplikasi Logbook dengan Auth dan Onboarding pada perangkat fisik Android.



**GAMBAR 1.2:** Penampakan aplikasi Logbook dengan Counter, Logout dan Dialog pada perangkat fisik Android.

## Troubleshooting Lab

Gejala	Penyebab	Solusi
<b>Error: "Undefined name 'CounterView'"</b>	-	Pastikan sudah melakukan import 'counter_view.dart'; di bagian atas file login_view.dart.
<b>Lupa TextEditingController:</b>	Klik login tidak ada respon atau teks tidak terbaca.	Pastikan controller: _userController sudah dipasang di dalam widget TextField.
<b>Masalah Navigasi di Controller</b>	-	Jika mahasiswa mencoba memindahkan Navigator ke dalam file LoginController, akan muncul error. Controller tidak punya Context. Navigasi harus dilakukan di View.

## Tugas Praktikum

Untuk menguji sejauh mana pemahaman kalian tentang SRP dan Dart Dasar, selesaikan dua tantangan berikut. Ingat aturan mainnya: 70% harus selesai di Lab (hingga fungsi berjalan), dan 30% sisanya (perapian UI/UX) bisa kalian jadikan homework untuk mempercantik portofolio kalian.

[https://github.com/dzhax499/PY4\\_2C\\_D3\\_2024\\_Modul\\_071/tree/main/MODUL\\_2/logbook\\_app\\_071](https://github.com/dzhax499/PY4_2C_D3_2024_Modul_071/tree/main/MODUL_2/logbook_app_071)

### Task 1: Tugas Pendahuluan

**Dikerjakan secara mandiri sebelum sesi praktikum dimulai.**

1. Siapkan asset gambar (minimal 3 buah) yang akan digunakan untuk menggantikan angka pada fitur Onboarding nanti.
2. Pelajari dokumentasi Flutter tentang Shared Preferences atau Path Provider sebagai cara menyimpan data ringan di memori HP.
3. Jawablah singkat: Apa perbedaan mendasar antara Navigator.push() dan Navigator.pushAndRemoveUntil()?

### Task 2: The Login Portal (LOTS)

**Fokus: Mengimplementasikan keamanan dasar dan validasi input..**

Spesifikasi Tugas:

1. Controller: Modifikasi LoginController agar mendukung sistem Multiple Users menggunakan tipe data Map<String, String>.
2. Security Logic: \* Implementasikan validasi agar field Username dan Password tidak boleh kosong (tampilkan pesan error).

- Berikan batas percobaan login. Jika salah 3 kali, tombol login menjadi tidak aktif (disabled) selama 10 detik.
- View: Tambahkan fitur Show/Hide Password menggunakan ikon mata pada TextField password.

Kriteria Selesai di Lab (70%):

- [x] Berhasil login dengan minimal 2 akun berbeda dari Map.
- [x] Pesan Snackbar muncul jika login gagal.
- [x] Fitur Show/Hide Password berfungsi dengan lancar.

## Task 3: Persistent History Logger (HOTS)

**Fokus:** Membuat aplikasi mampu "mengingat" data meskipun aplikasi ditutup.

Spesifikasi Tugas:

- Logic: Tambahkan fungsi di CounterController untuk menyimpan nilai hitungan terakhir ke dalam penyimpanan lokal (bisa menggunakan Shared Preferences atau menulis ke file .txt sederhana).
- History Log: Simpan riwayat aktivitas (Contoh: "User admin menambah +5 pada jam 10:00") ke dalam List.
- Data Persistence: Saat aplikasi dibuka kembali dan user berhasil login, aplikasi harus menampilkan "Angka Terakhir" yang tersimpan sebelumnya, bukan kembali ke angka 0.

Kriteria Selesai di Lab (70%):

- [x] Data angka terakhir tidak hilang saat aplikasi di-restart (Hot Restart).
- [x] Riwayat aktivitas tersimpan rapi dan dapat ditampilkan kembali.
- [x] Logika penyimpanan tetap terisolasi di dalam CounterController (SRP).

## Hints: Memahami Shared Preferences

Bayangkan Shared Preferences sebagai buku catatan kecil yang disimpan oleh sistem operasi Android/iOS khusus untuk aplikasi Anda.

- Format Data: Menggunakan sistem Key-Value (seperti Kamus). Contoh: Kunci "last\_number" berisi nilai 42.
- Sifat: Data tetap ada meskipun aplikasi ditutup paksa (Force Close) atau HP dimatikan.

## Snippets: Implementasi Data Persistence

- Instalasi (Pubspec.yaml) Setelah itu jalankan flutter pub get

```
dependencies:
  shared_preferences: ^2.2.2
```

- Menulis Data (Save)

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

class CounterController {
  // Fungsi untuk menyimpan angka terakhir
  Future<void> saveLastValue(int value) async {
    final prefs = await SharedPreferences.getInstance();
```

```
        await prefs.setInt('last_counter', value);
        // 'last_counter' adalah Kunci (Key) untuk memanggil data nanti
    }
}
```

### 3. Membaca Data (Load)

```
Future<int> loadLastValue() async {
    final prefs = await SharedPreferences.getInstance();
    // Ambil nilai berdasarkan Key, jika kosong (null) berikan nilai
    default 0
    return prefs.getInt('last_counter') ?? 0;
}
```

## Catatan Penting

1. Asynchronous (Future/Await): SharedPreferences.getInstance() bersifat async. Jadi diharuskan kata kunci await agar aplikasi menunggu hingga "buku catatan" terbuka sebelum membaca/menulis.
2. Hot Reload vs Hot Restart: Hot Reload terkadang tidak menjalankan ulang initState. Jika data belum muncul, lakukan Hot Restart (tombol petir hijau/ikon melingkar di VS Code) agar fungsi loadLastValue dipanggil kembali.
3. Key yang Konsisten: nama Key (misal: 'last\_counter') saat menyimpan dan membaca harus sama persis. Jika beda satu huruf saja, data tidak akan ditemukan.

## Homework Cosmetic & UX Enhancement (30%)

Setelah logika di atas berjalan lancar di lab, gunakan waktu di rumah untuk:

1. Visual Onboarding: Ganti angka 1, 2, dan 3 pada OnboardingView dengan gambar/ilustrasi yang sudah Anda siapkan di tugas pendahuluan. Tambahkan teks deskripsi yang menarik di bawah gambar tersebut.
2. Indicator: Tambahkan titik indikator (Page Indicator) di bawah gambar untuk menunjukkan posisi halaman onboarding (misal: Bulatan biru untuk halaman aktif).
3. The Welcome Banner: Modifikasi CounterView agar menampilkan pesan unik berdasarkan waktu login (Contoh: "Selamat Pagi, [Nama]" jika login jam 06.00-11.00).
4. Tantangan: Jika saya login sebagai 'admin', angka terakhir saya adalah 10. Jika saya logout dan masuk sebagai 'budi', angka terakhir saya harusnya berbeda atau mulai dari 0.

## Penilaian

### Kuesioner Self-Assessment

Jawablah dengan jujur sesuai dengan apa yang kalian rasakan setelah menyelesaikan Modul 1. Berikan tanda centang (✓) pada kolom angka yang paling sesuai:

Keterangan Skala: 1: Sangat Tidak Setuju (STS) | 2: Tidak Setuju (TS) | 3: Netral (N) | 4: Setuju (S) | 5: Sangat Setuju (SS)

No	Pernyataan Refleksi	1	2	3	4	5
1	1	2	3	4	5	

<b>1</b>	Saya mampu menjelaskan alasan teknis penerapan Folder Structure (Features) dalam mengelola proyek aplikasi yang mulai kompleks.					✓
<b>2</b>	Saya memahami peran BuildContext sebagai "peta" dan alasan mengapa navigasi tidak boleh diletakkan di dalam file Controller.					✓
<b>3</b>	Saya mampu menganalisis perbedaan penggunaan pushReplacement dan pushAndRemoveUntil untuk menjaga keamanan alur aplikasi (Navigation Stack).					✓
<b>4</b>	Saya memahami konsep Asynchronous (async, await, Future) saat melakukan proses penyimpanan data ke memori lokal (Shared Preferences).					✓
<b>5</b>	Saya merasa percaya diri untuk melakukan Passing Data antar halaman (mengirim data Username dari Login ke Counter) secara mandiri.					✓

Apa bagian yang paling menantang atau membingungkan bagi kalian di modul ini?

## Peer Assessment (Apresiasi Rekan Sejawat 1)

Setelah selesai, tukarkan HP kalian dengan teman sebangku. Cobalah aplikasi mereka dan berikan penilaian sejajar mungkin. Ingat, tujuannya bukan menjatuhkan, tapi saling memperbaiki!

**Nama Penilai:** Dzakir Tsabit Asy Syafiq **Nama Pemilik Aplikasi:** Muh

No	Kriteria Pengalaman Pengguna (UX)	Skor (1-5)	Catatan "Hal Keren/Saran"
1	Navigasi Alur: Apakah alur dari Onboarding → Login → Counter terasa mulus dan tidak membingungkan?		
2	Feedback (Pesan): Apakah Snackbar (pesan error) muncul dengan jelas saat Anda memasukkan username/password yang salah?		
3	Konfirmasi Logout: Apakah dialog konfirmasi muncul saat tombol Logout ditekan, dan apakah tombol "Batal" bekerja dengan benar?		
4	Ketahanan (Security): Setelah Logout, coba tekan tombol "Back" pada perangkat. Apakah aplikasi tetap di halaman Login (tidak kembali ke Counter)?		
5	Kejutan & Estetika: Apakah gambar/ilustrasi pada halaman Onboarding dan sapaan nama di halaman Counter terlihat menarik dan personal?		

Print Screen Aplikasi Teman Anda:

## Peer Assessment (Apresiasi Rekan Sejawat 2)

Setelah selesai, tukarkan HP kalian dengan teman sebangku. Cobalah aplikasi mereka dan berikan penilaian sejajar mungkin. Ingat, tujuannya bukan menjatuhkan, tapi saling memperbaiki!

**Nama Penilai:** \_\_\_\_\_ **Nama Pemilik Aplikasi:** \_\_\_\_\_

No	Kriteria Pengalaman Pengguna (UX)	Skor (1-5)	Catatan "Hal Keren/Saran"
1	Navigasi Alur: Apakah alur dari Onboarding → Login → Counter terasa mulus dan tidak membingungkan?		
2	Feedback (Pesan): Apakah Snackbar (pesan error) muncul dengan jelas saat Anda memasukkan username/password yang salah?		
3	Konfirmasi Logout: Apakah dialog konfirmasi muncul saat tombol Logout ditekan, dan apakah tombol "Batal" bekerja dengan benar?		
4	Ketahanan (Security): Setelah Logout, coba tekan tombol "Back" pada perangkat. Apakah aplikasi tetap di halaman Login (tidak kembali ke Counter)?		
5	Kejutan & Estetika: Apakah gambar/ilustrasi pada halaman Onboarding dan sapaan nama di halaman Counter terlihat menarik dan personal?		

Print Screen Aplikasi Teman Anda:

## Peer Assessment (Apresiasi Rekan Sejawat 3)

Setelah selesai, tukarkan HP kalian dengan teman sebangku. Cobalah aplikasi mereka dan berikan penilaian sejajar mungkin. Ingat, tujuannya bukan menjatuhkan, tapi saling memperbaiki!

**Nama Penilai:** \_\_\_\_\_ **Nama Pemilik Aplikasi:** \_\_\_\_\_

No	Kriteria Pengalaman Pengguna (UX)	Skor (1-5)	Catatan "Hal Keren/Saran"
1	Navigasi Alur: Apakah alur dari Onboarding → Login → Counter terasa mulus dan tidak membingungkan?		
2	Feedback (Pesan): Apakah Snackbar (pesan error) muncul dengan jelas saat Anda memasukkan username/password yang salah?		
3	Konfirmasi Logout: Apakah dialog konfirmasi muncul saat tombol Logout ditekan, dan apakah tombol "Batal" bekerja dengan benar?		
4	Ketahanan (Security): Setelah Logout, coba tekan tombol "Back" pada perangkat. Apakah aplikasi tetap di halaman Login (tidak kembali ke Counter)?		
5	Kejutan & Estetika: Apakah gambar/ilustrasi pada halaman Onboarding dan sapaan nama di halaman Counter terlihat menarik dan personal?		

Print Screen Aplikasi Teman Anda:



## Template Lesson Learnt (Refleksi Akhir)

Jangan biarkan ilmu hari ini menguap begitu saja. Tuliskan 3 poin utama yang baru kalian sadari atau pahami hari ini.

1. Konsep Baru: (Contoh: Baru tahu kalau \_ itu buat bikin variabel jadi rahasia/private).
2. Kemenangan Kecil: (Contoh: Berhasil benerin error Path yang bikin pusing selama 1 jam).
3. Target Berikutnya: (Contoh: Pengen tahu gimana cara bikin tampilan yang lebih berwarna di bab selanjutnya).

## Log LLM: The Fact Check & Twist (Homework)

Menggunakan AI itu boleh, yang tidak boleh adalah menjadi "Zombi AI" (Copy-Paste tanpa mikir). Gunakan template ini setiap kali kalian meminta bantuan ChatGPT/Gemini/Claude.

### Contoh Cara Bertanya yang Benar (Prompting)

-  **Salah:** "Buatin kode list di Flutter." (Terlalu umum, kalian akan bingung bacanya).
-  **Benar:** "Saya punya class CounterController di Dart. Saya ingin menambah `List<String>` untuk simpan riwayat tiap tombol diklik. Tolong berikan logika murni Dart-nya saja tanpa UI."

Komponen	Isian Mahasiswa
Pertanyaan (Prompt)	"Gimana cara nambahin data ke List tapi dibatasi cuma 5 data terbaru aja di Dart?"
Jawaban AI (Intisari)	AI menyarankan pakai fungsi <code>insert(0, data)</code> lalu pakai <code>removeLast()</code> kalau panjang list > 5.
The Fact Check	Saya coba di DartPad, ternyata <code>removeLast()</code> bakal error kalau List-nya masih kosong. Jadi saya tambahin pengecekan if ( <code>list.isNotEmpty</code> ).
The Twist (Modifikasi)	Saya nggak cuma simpan teks biasa, tapi saya tambahin jam otomatis pakai <code>DateTime.now()</code> supaya lebih informatif seperti aplikasi logbook asli.

## Rubrik Penilaian Dosen Manajer

Tabel 2.2: Rubrik Penilaian Praktikum Modul 1.

Kriteria	Skor 0-50 (Kurang)	Skor 51-80 (Cukup)	Skor 81-100 (Sangat Baik)	Bobot
<b>Arsitektur Modular</b>	File masih menumpuk di folder lib atau struktur folder berantakan.	Sudah menggunakan folder features, tapi penempatan file Controller/View masih tertukar.	Struktur folder features/auth, features/onboarding, dan features/logbook rapi dan konsisten.	25%
<b>Task 1 (Auth &amp; Logic)</b>	Login gagal atau logika validasi (admin/123) ditulis di dalam file View.	Login berhasil, tapi tidak ada validasi input kosong atau <i>Show/Hide password</i> tidak jalan.	Logika login di Controller sempurna, mendukung <i>multiple users</i> , dan validasi UI sangat informatif.	25%
<b>Task 2 (Persistence)</b>	Data angka terakhir hilang saat aplikasi di-restart (Hot Restart).	Menggunakan <i>Shared Preferences</i> tapi terjadi error saat membaca data awal (null).	Data tersimpan permanen dan muncul otomatis saat login kembali (Data Persistence sukses).	25%
<b>Task 3 (Nav &amp; UX)</b>	Alur navigasi berantakan (bisa 'back' setelah login/logout) atau aplikasi crash.	Alur navigasi jalan, tapi dialog konfirmasi logout tidak muncul atau tidak kembali ke Onboarding.	Alur navigasi aman menggunakan pushAndRemoveUntil, dialog konfirmasi bekerja, dan UX mulus.	25%

## Rubrik Penilaian Log LLM (Integritas)

Tabel 2.3: Rubrik Penilaian Integritas Log LLM.

Kriteria	Skor 1	Skor 3	Skor 5
<b>Kejujuran</b>	Mengaku tidak pakai AI padahal kode sangat kompleks.	Mencantumkan prompt tapi jawaban AI disalin bulat-bulat.	Terbuka soal bantuan AI dan mencantumkan prompt dengan jelas.
<b>Analisa (Fact Check)</b>	Tidak ada pengecekan ulang.	Mengecek apakah kode jalan atau tidak saja.	Mampu menemukan celah atau kelemahan dari saran AI.
<b>Kreativitas (Twist)</b>	Kode 100% sama dengan saran AI.	Mengubah nama variabel saja.	Mengintegrasikan saran AI ke dalam struktur SRP yang sudah dibuat.

## Referensi