



POLITEKNIK NEGERI BANYUWANGI

Laporan Final Project: CinemaPro

Mobile Application Development
Semester Ganjil 2024/2025

Disusun Oleh Kelompok 07:

Cheryl Aurellya Bangun Jaya	NIM001	<i>Backend Engineer & Data Seeder</i>
Dino Febiyan	362458302043	<i>Frontend Engineer (Home Module)</i>
Rusydi Jabir Al Awfa	362458302044	<i>Frontend Engineer (Seat Matrix Module)</i>
Mohammad Faisal	NIM004	<i>Logic Controller</i>
Semua		<i>QA Lead, Auth & Profile</i>

BAB 1

Pendahuluan & Pembagian Kerja

1.1 Deskripsi Aplikasi

CinemaPro adalah aplikasi pemesanan tiket bioskop berbasis Flutter dengan integrasi Firebase, mendukung multi-platform, dan memiliki sistem autentikasi yang aman.

1.2 Tabel Pembagian Tanggung Jawab

Berikut adalah detail pembagian tugas sesuai instruksi ujian:

1.3 Tabel Pembagian Tanggung Jawab (sementara)

Berikut adalah ringkasan pembagian tugas (sementara) yang dapat diperbaiki nanti:

Anggota 1: Backend Engineer & Data Seeder — Cheryl Aurellya Bangun Jaya

Anggota 2: Frontend Engineer (Home Module) — Dino Febiyan

Anggota 3: Frontend Engineer (Seat Matrix Module) — Rusydi Jabir Al Awfa

Anggota 4: Logic Controller (The Brain) — Mohammad Faisal

Anggota 5: QA Lead, Auth & Profile — Semua anggota

BAB 2

Bukti Keaslian Kode (Strict Mode)

Sesuai instruksi ujian untuk mencegah penggunaan **code generator** otomatis

2.1 Watermark Code

Berikut adalah bukti penggunaan suffix inisial pada variabel Widget dan Fungsi:

```
String _formatPrice_dino(int price) {
    final priceStr = price.toString();
    final buffer = StringBuffer('Rp ');
    for (int i = 0; i < priceStr.length; i++) {
        if (i > 0 && (priceStr.length - i) % 3 == 0) {
            buffer.write('.');
        }
        buffer.write(priceStr[i]);
    }
    return buffer.toString();
}
```

Penjelasan: Fungsi `_formatPrice_dino` menunjukkan kode dibuat oleh Dino (UI Engineer)

2.2 Logic Trap

Implementasi logika bisnis “The ”Long Title” Tax”

```
// Contoh snippet kode (Logic Trap)
void hitungDiskon_dani(String nim) {
    int lastDigit = int.parse(nim.characters.last);
    if (lastDigit % 2 != 0) {
        // Logika Ganjil
        print("Diskon 5%");
    } else {
        // Logika Genap
        print("Gratis Ongkir");
    }
}
```

TEMPEL KODEMU NDEK KENE @ISAL

Implementasi logika bisnis unik “Odd/Even Seat Rule:”

```
// Contoh snippet kode (Logic Trap)
void hitungDiskon_dani(String nim) {
    int lastDigit = int.parse(nim.characters.last);
    if (lastDigit % 2 != 0) {
        // Logika Ganjil
        print("Diskon 5%");
    } else {
```

```
// Logika Genap  
print("Gratis Ongkir");  
}  
}
```

TEMPEL KODEMU NDEK KENE @ISAL

BAB 3

Arsitektur Backend

Dikerjakan oleh Backend Engineer & Data Seeder[cite: 23].

3.1 Struktur Database (Firestore)

Menggunakan 3 Collection utama: Users, Products, dan Transactions[cite: 13, 15, 17].

3.2 Data Seeding

Bukti seeding minimal 10 produk dummy ke Firebase[cite: 27].

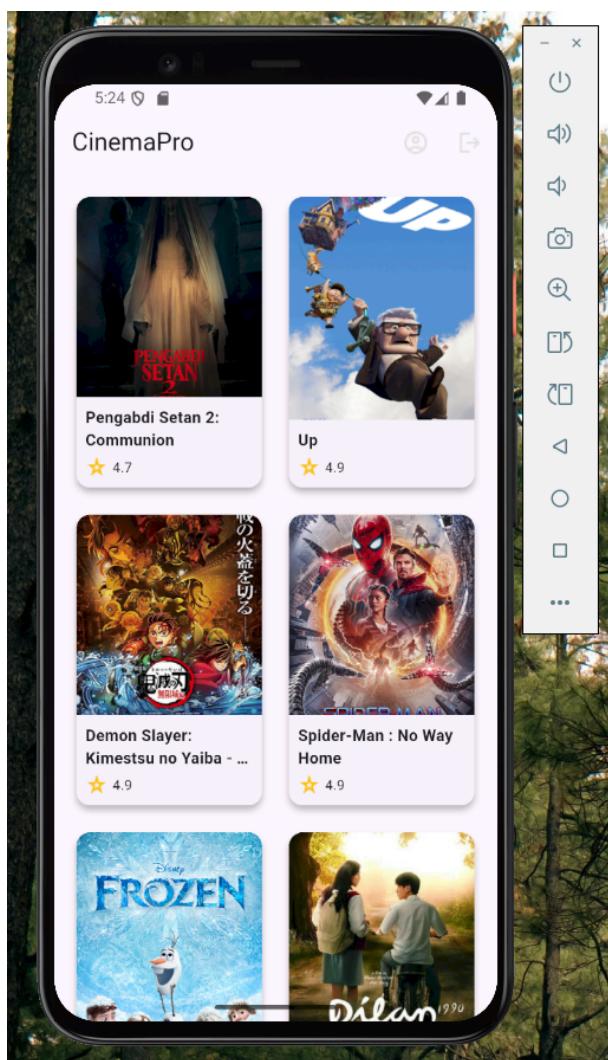
BAB 4

Implementasi UI Home & Detail

Dikerjakan oleh Frontend Engineer (Home Module)

4.1 UI Home

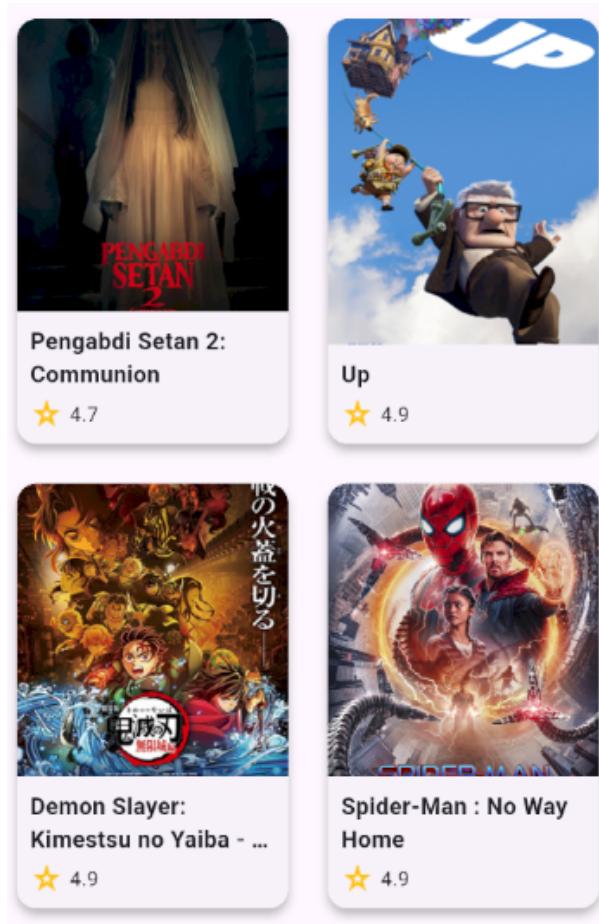
Halaman UI Home ini dirancang untuk menampilkan koleksi film dalam format grid beserta dengan judulnya. Dengan halaman ini, diharapkan pengguna akan bisa melihat berbagai film secara bersamaan dengan susunan yang rapi dan mudah untuk diakses. Ketika sebuah poster dipilih, aplikasi akan menggunakan konsep Hero widget agar muncul animasi transisi ke detail page, sehingga perpindahan menuju halaman detail film diharapkan akan lebih bagus dan menarik.



4.1.1 Membuat tampilan Grid Film (SliverGridDelegate).

Pada bagian ini saya membuat tampilan daftar film dalam bentuk grid dengan memanfaatkan `GridView.builder`. Untuk mengatur susunan grid, saya menggunakan `SliverGridDelegateWithMaxCrossAxisExtent`, di mana saya menentukan lebar maksimal setiap item sebesar 200 pixel. Dengan cara ini, tampilan home akan menjadi lebih bagus

karena jumlah kolom akan menyesuaikan ukuran layar. Selain itu, saya juga menambahkan jarak antar item menggunakan crossAxisSpacing dan mainAxisSpacing, serta mengatur rasio tinggi-lebar dengan childAspectRatio agar poster film lebih rapi. Data film saya ambil dari Firebase melalui snapshot.data!.docs kemudian saya taruh ke model MovieModelCheryl. Setiap item grid saya bungkus dengan GestureDetector sehingga ketika pengguna menekan sebuah poster, aplikasi akan diarahkan ke halaman detail film (DetailPage_dino).



```
return GridView.builder(  
    padding: const EdgeInsets.all(16.0),  
    gridDelegate: SliverGridDelegateWithMaxCrossAxisExtent(  
        maxCrossAxisExtent: 200,  
        crossAxisSpacing: 16.0,  
        mainAxisSpacing: 16.0,  
        childAspectRatio: 0.65,  
    ),  
    itemCount: snapshot.data!.docs.length,  
    itemBuilder: (context, index) {  
        final doc = snapshot.data!.docs[index];  
        final movieData = doc.data() as Map<String, dynamic>;  
        final movie = MovieModelCheryl.fromMap_Cheryl(movieData, doc.id);  
  
        return GestureDetector(  
            onTap: () {  
                Navigator.push(  
                    context,  
                    MaterialPageRoute(builder: (context) =>  
                        DetailPage_dino(  
                            movie: movie,  
                            id: doc.id,  
                        ),  
                );  
            },  
            child: Container(  
                width: 180,  
                height: 270,  
                decoration: BoxDecoration(  
                    image: DecorationImage(  
                        fit: BoxFit.cover,  
                        image: NetworkImage(movieData['poster']),  
                    ),  
                ),  
            ),  
        );  
    },  
);
```

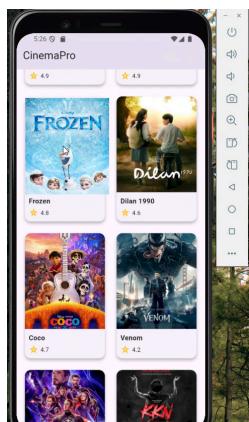
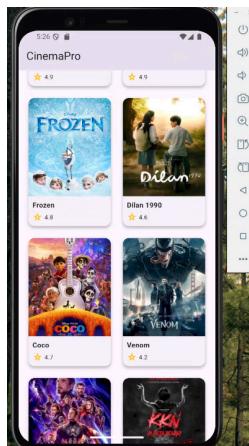
```

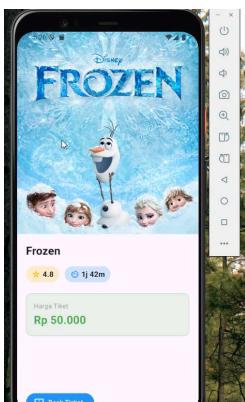
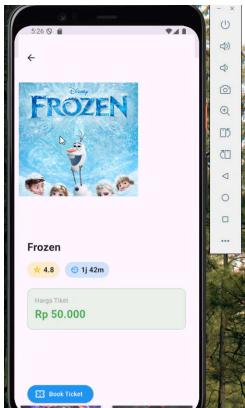
        context,
        MaterialPageRoute(
            builder: (context) => DetailPage_dino(movie: movie),
        ),
    );
},
);
},
);
);

```

4.1.2 Menggunakan widget Hero pada gambar poster agar ada animasi saat pindah ke detail.

Untuk menampilkan poster film, saya menggunakan Image.network agar gambar bisa langsung diambil dari link URL. Poster tersebut saya bungkus dengan widget Hero yang memiliki tag unik berdasarkan ID film. Dengan cara ini, ketika pengguna menekan sebuah poster, Flutter otomatis menampilkan animasi transisi dari poster di grid menuju poster di halaman detail. Efek animasi ini akan membuat perpindahan menjadi lebih bagus. Selain itu, saya menambahkan ClipRRect agar memberikan efek sudut melengkung pada gambar, serta errorBuilder dan loadingBuilder ketika ada kondisi gambar rusak atau masih dalam proses loading.





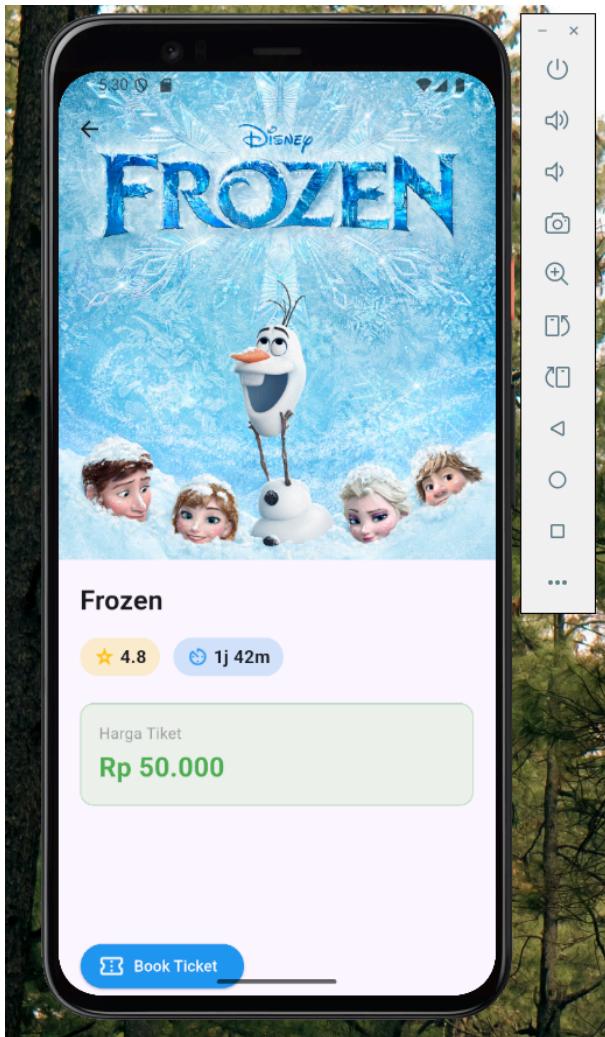
```
child: ClipRRect(
  borderRadius: const BorderRadius.vertical(
    top: Radius.circular(12),
  ),
  child: Hero(
    tag: movie.movieID,
    child: Image.network(
      movie.posterUrl,
      fit: BoxFit.cover,
      errorBuilder: (context, error, stackTrace) {
        return Container(
          color: Colors.grey[300],
          child: Center(
            child: Image.asset(
              'assets/icons/gambarRusak.png',

```

```
        width: 50,
        height: 50,
        fit: BoxFit.contain,
        color: Colors.grey,
      ),
    ),
  );
},
loadingBuilder: (context, child, loadingProgress) {
  if (loadingProgress == null) return child;
  return Center(
    child: CircularProgressIndicator(),
  );
},
),
),
);
};
```

4.2 UI Detail

Pada bagian UI Detail ini, saya membuat tampilan khusus yang berfungsi untuk menampilkan informasi lengkap mengenai film yang dipilih oleh pengguna. Halaman ini menjadi lanjutan dari UI Home, sehingga setelah pengguna menekan salah satu poster film, mereka akan diarahkan ke halaman detail. Tujuan utama dari UI Detail adalah memberikan informasi dari film, di mana pengguna bisa melihat detail film sekaligus melakukan tindakan pemesanan tiket dengan mudah.



4.2.1 Menampilkan info film.

Untuk menampilkan informasi film, saya memanfaatkan data yang sudah diambil dari Firebase dan kemudian ditampilkan dalam bentuk teks maupun gambar. Informasi yang saya tampilkan yaitu judul film, poster, serta harga tiket yang diformat menggunakan fungsi formatPrice_dino. Fungsi ini saya buat agar harga tiket ditampilkan dalam format rupiah dengan penulisan yang rapi, misalnya “Rp 25.000”. Poster film ditampilkan menggunakan Image.network sehingga gambar bisa langsung diambil dari URL, dan saya tambahkan ClipRRect untuk memberikan efek sudut melengkung. Selain itu, saya juga menambahkan errorBuilder dan loadingBuilder untuk menangani kondisi ketika gambar rusak atau masih dalam proses loading.



Frozen

★ 4.8

⌚ 1j 42m

Harga Tiket

Rp 50.000

```
String _formatPrice_dino(int price) {
    final priceStr = price.toString();
    final buffer = StringBuffer('Rp ');
    for (int i = 0; i < priceStr.length; i++) {
        if (i > 0 && (priceStr.length - i) % 3 == 0) {
            buffer.write('.');
        }
        buffer.write(priceStr[i]);
    }
    return buffer.toString();
}

Widget buildMovieInfo_dino(MovieModelCheryl movie) {
    return SingleChildScrollView(
        child: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            children: [
                ClipRRect(
                    borderRadius: const BorderRadius.vertical(top: Radius.circular(16)),
                    child: Image.network(
```

```

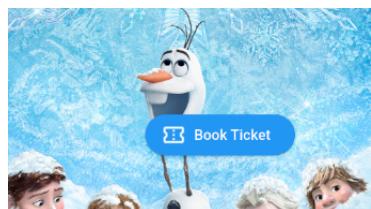
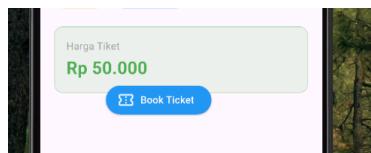
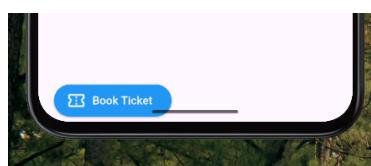
        movie.posterUrl,
        fit: BoxFit.cover,
        width: double.infinity,
        errorBuilder: (context, error, stackTrace) {
            return Container(
                height: 220,
                color: Colors.grey[300],
                child: Center(
                    child: Image.asset(
                        'assets/icons/gambarRusak.png',
                        width: 50,
                        height: 50,
                        color: Colors.grey,
                    ),
                ),
            );
        },
        loadingBuilder: (context, child, loadingProgress) {
            if (loadingProgress == null) return child;
            return const SizedBox(
                height: 220,
                child: Center(child: CircularProgressIndicator()),
            );
        },
    ),
),
const SizedBox(height: 16),
Text(
    movie.title,
    style: const TextStyle(
        fontSize: 22,
        fontWeight: FontWeight.bold,
    ),
),
const SizedBox(height: 8),
Text(
    _formatPrice_dino(movie.basePrice),
    style: const TextStyle(
        fontSize: 18,
        color: Colors.blueAccent,
        fontWeight: FontWeight.w600,
    ),
),
const SizedBox(height: 12),
if (movie.overview != null && movie.overview!.isNotEmpty)
Text(
    movie.overview!,
    style: const TextStyle(fontSize: 14, color: Colors.black87),
),
],
),

```

```
);  
}
```

4.2.2 Tombol "Book Ticket" melayang (Floating Action Button) di bawah.

Di bagian bawah halaman, saya menambahkan tombol Book Ticket yang melayang menggunakan Positioned dan ElevatedButton.icon. Tombol ini saya bungkus dengan GestureDetector sehingga posisinya bisa digeser oleh pengguna, sehingga bisa digeser sesuai kebutuhan. Saat tombol ditekan, sistem akan memeriksa apakah pengguna sudah login melalui Firebase Authentication. Jika belum login, akan muncul pesan peringatan menggunakan Snackbar. Namun jika sudah login, aplikasi akan menampilkan pesan "Sedang memproses pemesanan" dan langsung mengarahkan pengguna ke halaman pemilihan kursi (SeatMatrixJabir). Dengan adanya tombol ini, proses pemesanan tiket menjadi lebih praktis karena pengguna bisa langsung melakukan booking dari halaman detail film.



```
double posX = 16;  
double posY = 0;  
  
@override  
void initState() {  
    super.initState();  
    WidgetsBinding.instance.addPostFrameCallback((_) {  
        final screenHeight = MediaQuery.of(context).size.height;  
        setState(() {  
            posY = screenHeight - 50;  
        });  
    });  
}  
  
Positioned(  
    left: posX,  
    top: posY,  
    child: GestureDetector(  
        onPanUpdate: (details) {
```

```

        setState(() {
            posX += details.delta.dx;
            posY += details.delta.dy;
        });
    },
    child: ElevatedButton.icon(
        onPressed: () async {
            final user = FirebaseAuth.instance.currentUser;
            if (user == null) {
                ScaffoldMessenger.of(context).showSnackBar(
                    const SnackBar(content: Text('Harap Login Terlebih Dahulu!')),
                );
                return;
            }
            ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(content: Text('Sedang memproses pemesanan')),
            );
            try {
                Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder: (context) => SeatMatrixJabir(
                            movieTitle: movie.title,
                            userId: user.uid,
                            totalPrice: movie.basePrice,
                        ),
                    ),
                );
            } catch (e) {
                if (context.mounted) {
                    ScaffoldMessenger.of(context).hideCurrentSnackBar();
                    ScaffoldMessenger.of(context).showSnackBar(
                        SnackBar(
                            content: Text('Gagal: $e'),
                            backgroundColor: Colors.red,
                        ),
                    );
                }
            }
        },
        icon: Image.asset(
            'assets/icons/ticket.png',
            width: 24,
            height: 24,
            color: Colors.white,
        ),
        label: const Text(
            'Book Ticket',
            style: TextStyle(color: Colors.white),
        ),
        style: ElevatedButton.styleFrom(
            backgroundColor: Colors.blue,

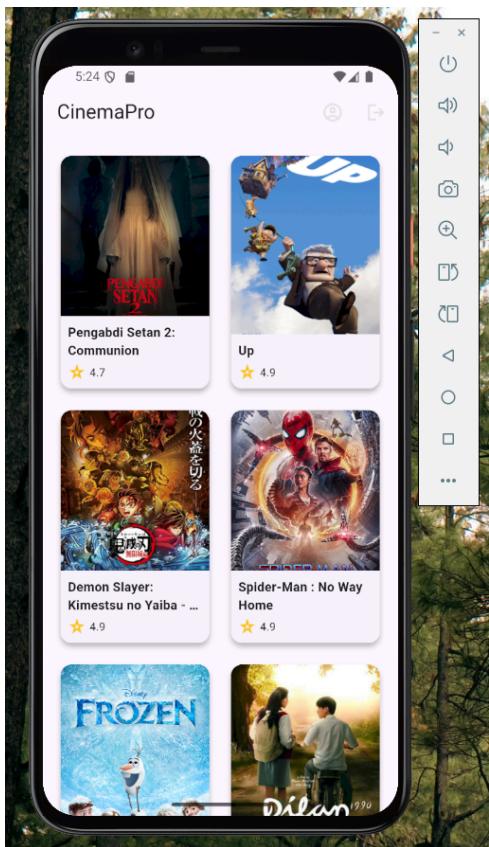
```

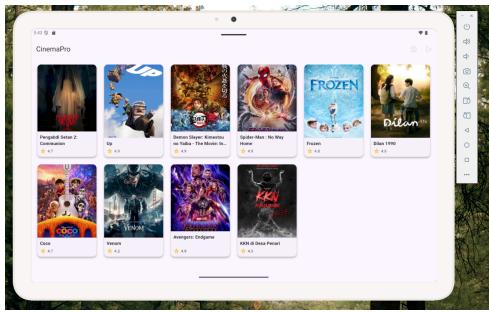
```
        elevation: 6,
        padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 12),
        shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(12),
        ),
    ),
),
),
),
),
),
),
)
```

4.3 Constraint

4.3.1 Layout harus responsif di HP kecil maupun besar.

Pada bagian constraint ini, saya akan memberikan bukti bahwa tampilan aplikasi benar benar responsif, artinya bisa menyesuaikan dengan berbagai ukuran layar ponsel. Sebelumnya saya menggunakan SliverGridDelegateWithMaxCrossAxisExtent untuk menampilkan daftar film dalam bentuk grid. Dengan cara ini, setiap item grid akan memiliki lebar maksimal tertentu, sehingga jumlah kolom akan otomatis menyesuaikan ukuran layar.





Ketika saya mencoba aplikasi di emulator dengan layar kecil, grid hanya menampilkan dua kolom agar poster tetap terlihat jelas dan tidak terlalu sempit. Sebaliknya, saat saya jalankan di layar yang lebih besar, jumlah kolom bertambah, sehingga ruang kosong bisa dimanfaatkan dengan baik. Hal ini menunjukkan bahwa penggunaan sliver membuat layout lebih fleksibel dan tetap rapi di berbagai ukuran layar.

Dengan cara ini, constraint “layout harus responsif” terbukti sudah terpenuhi, karena tampilan aplikasi tidak pecah atau berantakan meskipun dicoba di perangkat dengan resolusi berbeda. Bukti implementasi dapat dilihat langsung saat aplikasi dijalankan di emulator atau ponsel dengan ukuran layar berbeda, grid film tetap rapi serta poster tidak terpotong.

BAB 5

State Management & Logic

Dikerjakan oleh Logic Controller (Seat Matrix Module)[cite: 37].

5.1 State Keranjang

State management menggunakan (Provider/Bloc/GetX) untuk fitur **Add to Cart** dan **Remove from Cart**[cite: 38, 39].

5.2 Perhitungan Checkout

Total harga dihitung berdasarkan logika diskon NIM dan stok di Firebase berkurang otomatis (**Transaction Write**)[cite: 41].

BAB 6

Auth & Profile

Dikerjakan oleh Backend Engineer & Frontend Engineer[cite: 42].

6.1 Main

Pada bagian main ini, pertama tama yang saya lakukan adalah mengaktifkan Firebase agar aplikasi bisa terhubung dengan layanan login dan database. setelah itu saya menjalankannya dengan memanggil myapp

```
void main() async {
    WidgetsFlutterBinding.ensureInitialized();
    await Firebase.initializeApp(options:
DefaultFirebaseOptions.currentPlatform);
    runApp(const MyApp_dino());
}
```

Di dalam MyApp_dino, saya pakai MaterialApp untuk mengatur tema aplikasi, seperti warna biru sebagai warna utama, font Poppins, dan menghilangkan tulisan “debug” di pojok. Halaman awalnya saya arahkan ke initialScreen_dino.

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

    @override
    Widget build(BuildContext context) {
        return MaterialApp(
            title: 'CinemaPro',
            theme: ThemeData(
                primarySwatch: Colors.blue,
                fontFamily: 'Poppins',
                textTheme: Theme.of(context).textTheme,
            ),
            debugShowCheckedModeBanner: false,
            home: const initialScreen_dino(),
        );
    }
}
```

Nah, di initialScreen_dino saya membuat logika untuk menentukan halaman pertama yang muncul. Saya menggunakan StreamBuilder untuk mengecek status login pengguna. Kalau masih loading, tampil indikator bulat. Kalau pengguna sudah login, saya melakukan pengecekan data lagi di Firestore. Kalau datanya ada, langsung masuk ke HomePage_dino. Kalau datanya gagal dimuat, muncul pesan error. Sedangkan kalau pengguna belum login, otomatis diarahkan ke LoginPage_dino.

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

    @override
```

```

Widget build(BuildContext context) {
  return StreamBuilder<User?>(
    stream: FirebaseAuth.instance.authStateChanges(),
    builder: (context, snapshot) {

      if (snapshot.connectionState == ConnectionState.waiting) {
        return const Scaffold(
          body: Center(child: CircularProgressIndicator()),
        );
      }

      if (snapshot.hasData && snapshot.data != null) {
        return FutureBuilder<DocumentSnapshot>(
          future: FirebaseFirestore.instance
            .collection('users')
            .doc(snapshot.data!.uid)
            .get(),

          builder: (context, userSnapshot) {

            if (userSnapshot.connectionState == ConnectionState.waiting) {
              return const Scaffold(
                body: Center(child: CircularProgressIndicator()),
              );
            }

            if (userSnapshot.hasData && userSnapshot.data!.exists) {
              return HomePage_dino();
            } else {
              return const Scaffold(
                body: Center(child: Text('Gagal memuat data user')),
              );
            }
          },
        );
      }
      return const LoginPage_dino();
    },
  );
}

```

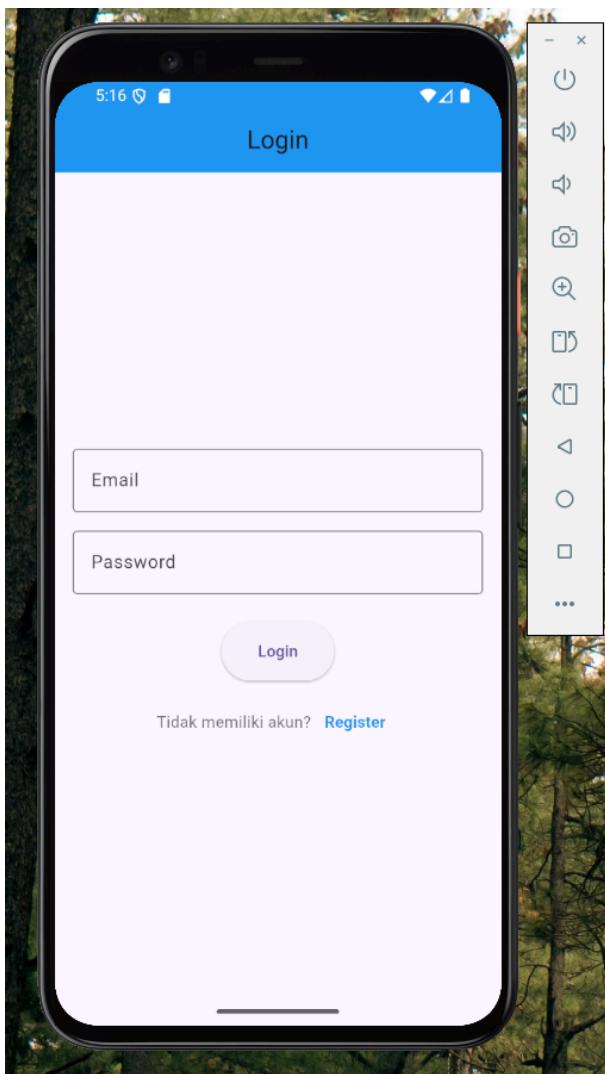
6.2 Auth Register

Struktur branch yang digunakan sesuai instruksi[cite: 49, 50, 51, 52, 53]:

- feature/backend-setup
- feature/ui-widgets
- feature/auth-nav
- feature/cart-state
- feature/testing-docs

6.3 Auth Login

Pada bagian login ini saya membuat sebuah halaman yang memungkinkan pengguna masuk ke aplikasi menggunakan akun yang sudah terdaftar di Firebase. Tampilan login ini terdiri dari dua input utama, yaitu email dan password, yang masing-masing dilengkapi dengan validasi. Untuk email, saya menambahkan aturan bahwa alamat harus menggunakan domain kampus @student.univ.ac.id, sehingga hanya mahasiswa yang bisa mendaftar dan masuk. Jika pengguna salah mengetik atau tidak mengisi, maka sistem langsung menampilkan pesan error dan border input berubah menjadi merah sebagai tanda visual. Hal yang sama juga berlaku pada password, di mana sistem memastikan password minimal enam karakter, dan jika tidak sesuai maka field akan ditandai dengan border merah.



```
class _LoginPageState_dino extends State<LoginPage_dino> {
    final _formKey = GlobalKey<FormState>();

    final TextEditingController emailController = TextEditingController();
    final TextEditingController passwordController = TextEditingController();

    bool _isLoading = false;

    @override
```

```

Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Login'),
      centerTitle: true,
      backgroundColor: Colors.blue,
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Form(
        key: _formKey,
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            TextFormField(
              controller: emailController,
              decoration: const InputDecoration(
                labelText: 'Email',
                border: OutlineInputBorder(),
                errorBorder: OutlineInputBorder(
                  borderSide: BorderSide(color: Colors.red),
                ),
                focusedErrorBorder: OutlineInputBorder(
                  borderSide: BorderSide(color: Colors.red, width: 2),
                ),
              ),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Email wajib diisi';
                }
                if (!EmailValidator.isValidStudentEmail(value)) {
                  return 'Email harus @student.univ.ac.id';
                }
                return null;
              },
            ),
            const SizedBox(height: 16),
            TextFormField(
              controller: passwordController,
              obscureText: true,
              decoration: const InputDecoration(
                labelText: 'Password',
                border: OutlineInputBorder(),
                errorBorder: OutlineInputBorder(
                  borderSide: BorderSide(color: Colors.red),
                ),
                focusedErrorBorder: OutlineInputBorder(
                  borderSide: BorderSide(color: Colors.red, width: 2),
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
}

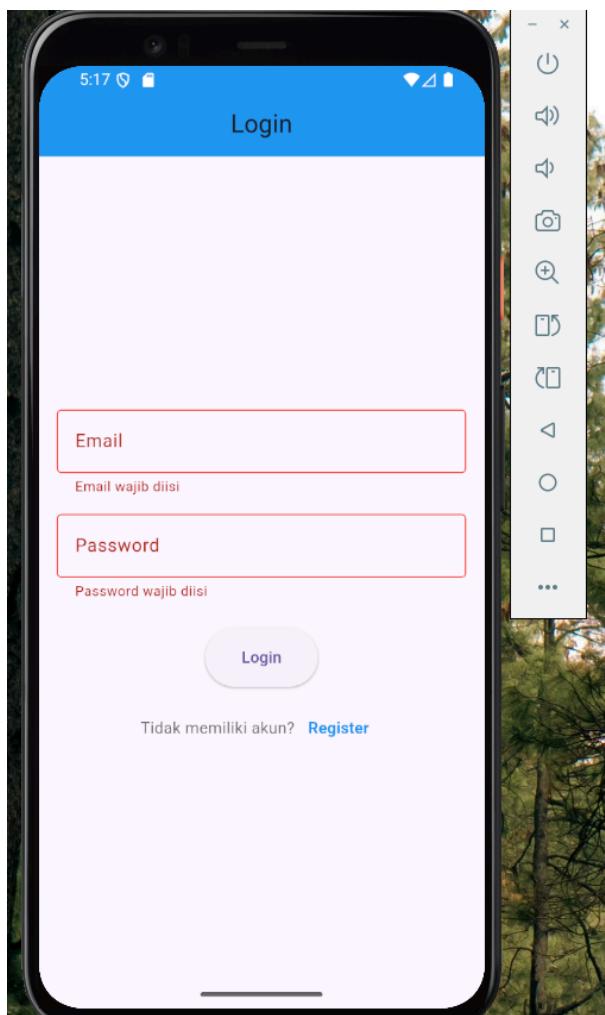
```

```

),
validator: (value) {
  if (value == null || value.isEmpty) {
    return 'Password wajib diisi';
  }
  if (value.length < 6) {
    return 'Password minimal 6 karakter';
  }
  return null;
},
),

```

Selain itu, saya menambahkan logika agar ketika tombol **Login** ditekan, aplikasi terlebih dahulu memvalidasi form. Jika valid, sistem akan mencoba masuk menggunakan `FirebaseAuth.signInWithEmailAndPassword`. Selama proses ini berlangsung, tombol login diganti dengan indikator loading agar pengguna tahu bahwa proses sedang berjalan. Jika login berhasil, muncul pesan **Login berhasil** dengan warna hijau, lalu pengguna diarahkan ke halaman utama `HomePage_dino`. Sebaliknya, jika login gagal, sistem menampilkan pesan error berwarna merah sesuai dengan alasan kegagalan dari Firebase.



```

_isLoading
? const CircularProgressIndicator()

```

```

        : ElevatedButton(
            onPressed: _login_dino,
            style: ElevatedButton.styleFrom(
                padding: const EdgeInsets.symmetric(
                    horizontal: 32,
                    vertical: 16,
                ),
            ),
            child: const Text('Login'),
        ),
    ),
    const SizedBox(height: 12),
}

Future<void> _login_dino() async {
    if (!_formKey.currentState!.validate()) {
        return;
    }

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

    try {
        await FirebaseAuth.instance.signInWithEmailAndPassword(
            email: emailController.text.trim(),
            password: passwordController.text,
        );

        if (mounted) {
            ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(
                    content: Text('Login berhasil'),
                    backgroundColor: Colors.green,
                ),
            );
        }

        Navigator.pushReplacement(
            context,
            MaterialPageRoute(builder: (context) => HomePage_dino()),
        );
    }
}

} on FirebaseAuthException catch (e) {
    if (mounted) {
        ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(
                content: Text('Login gagal: ${e.message}'),
                backgroundColor: Colors.red,
            ),
        );
    }
} finally {
}

```

```

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

```

Di bagian bawah halaman, saya juga menyediakan opsi bagi pengguna yang belum memiliki akun. Terdapat teks **Tidak memiliki akun?** dengan tombol **Register** yang akan mengarahkan pengguna ke halaman pendaftaran. Dengan cara ini, halaman login tidak hanya berfungsi untuk login saja, tetapi juga menjadi navigator bagi pengguna baru untuk membuat akun.

Tidak memiliki akun? [Register](#)

```

Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
        const Text(
            "Tidak memiliki akun?",
            style: TextStyle(color: Colors.black54),
        ),
        TextButton(
            onPressed: () {
                Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder: (context) => RegisterScreen(),
                    ),
                );
            },
            child: const Text(
                'Register',
                style: TextStyle(
                    color: Colors.blue,
                    fontWeight: FontWeight.bold,
                ),
            ),
        ),
    ],
),

```

6.4 Profile

BAB 7

Pengujian & Penutup

7.1 Handling Data

Bukti fitur `ListView.builder` tidak error saat data kosong dan implementasi `LoadingIndicator`[cite: 44, 45].

7.2 Link Video Demo

Berikut adalah link video demo aplikasi yang diunggah ke Google Drive/YouTube:

- **Link:** [Masukkan Link Di Sini]