POLITEKNIK NEGERI BANYUWANGI JURUSAN TEKNOLOGI
INFORMASI PROGRAM STUDI TEKNIK REKAYASA PERANGKAT LUNAK

Laporan Final Project: CineBooking App

Pemrograman Perangkat Bergerak
Semester 3 (Tiga)

KELOMPOK 03

Moh. Jevon Attaillah	362458302035	*Backend Engineer & Data Seeder*
Adelia Fioren Zety	362458302124	*UI Engineer - Home Module*
Intan Rahma Safira	362458302099	*UI Engineer - Seat Matrix*
Salam Rizqi Mulia	362458302041	*Logic Controller*
Achmad Alfarizy Satriya G	362458302147	*QA Lead, Auth & Profile*

Dosen Pengampu: Tri Hadiah Mulia Bunda, S.Kom., M.T. Desember 2024

BAB 1
Pendahuluan & Pembagian Kerja

1.1 Deskripsi Aplikasi

Aplikasi CineBooking adalah aplikasi pemesanan tiket bioskop real-time berbasis Flutter dengan Firebase. Aplikasi ini memungkinkan pengguna untuk melihat film yang tersedia, memilih kursi, dan melakukan booking dengan perhitungan harga dinamis.

1.2 Pembagian Tanggung Jawab Kelompok 3

No	Nama	NIM	Peran	Kode Watermark
1	Moh. Jevon Attaillah	362458302035	Backend Engineer & Data Seeder	`MovieModelJevon`, `BookingModelJevon`, `MovieSeederJevon`
2	Adelia Fioren Zety	362458302124	UI Engineer - Home Module	`HomePageAdel`, `MovieCardAdel`, `MovieDetailPageAdel`
3	Intan Rahma Safira	362458302099	UI Engineer - Seat Matrix	`SeatPageIntan`, `SeatItemIntan`
4	Salam Rizqi Mulia	362458302041	Logic Controller	`SeatControllerSalam`
5	Achmad Alfarizy Satriya G	362458302147	QA Lead, Auth & Profile	`LoginPageBioskopFariz`, `RegisterPageBioskopFariz`, `ProfilePageFariz`

BAB 2
Bukti Keaslian Kode (Strict Mode)

2.1 Watermark Code

Setiap kode memiliki inisial pembuat di nama class, fungsi, dan variabel:

```
// Contoh watermark dari kode program:
class MovieModelJevon {}          // Inisial "Jevon"
class HomePageAdel {}             // Inisial "Adel"
class SeatPageIntan {}            // Inisial "Intan"
class SeatControllerSalam {}      // Inisial "Salam"
class LoginPageBioskopFariz {}    // Inisial "Fariz"
```

2.2 Logic Trap Implementasi Manual

```
// File: controllers/seat_controller.dart
int calculateTotalPrice() {
  int total = basePriceSalam * _selectedSeatsSalam.length;

  // Trap 1: Jika judul film > 10 karakter, tambah Rp 2.500 per kursi
  if (movieTitleSalam.length > 10) {
    total = total + (2500 * _selectedSeatsSalam.length);
  }

  // Trap 2: Kursi genap dapat diskon 10%
  for (var seat in _selectedSeatsSalam) {
    final number = int.tryParse(seat.substring(1)) ?? 0;
    if (number % 2 == 0) {
      total -= (basePriceSalam * 0.10).toInt();
    }
  }
  return total;
}
```

BAB 3
Arsitektur Backend

Dikerjakan oleh Backend Architect:

3.1 Struktur Database (Firestore)

Database menggunakan Cloud Firestore dengan 3 Collection utama. Penamaan field (`snake_case`) disesuaikan untuk fungsi `toMapJevon` dan `fromMapJevon`.

3.2 Collection: users

Menyimpan data profil pengguna dan saldo.

1. uid (String): Primary Key (dari Firebase Auth)
2. email (String): Email mahasiswa
3. username (String): Nama tampilan pengguna
4. balance (Number (Int)): Saldo awal (Default: 0)
5. created_at (Timestamp): Waktu registrasi akun

3.3 Collection: movies

Menyimpan data film untuk ditampilkan pada Grid Home.

1. movie_id (String): Primary Key (Manual ID: movie1 - movie10)
2. title (String): Judul Film (Logic Trap Title Tax)
3. poster_url (String): URL Gambar Poster (TMDB)
4. base_price (Number (Int)): Harga dasar tiket
5. rating (Number (Double)): Rating film (1.0 - 5.0)
6. duration (Number (Int)): Durasi film dalam menit

3.4 Collection: bookings

Menyimpan riwayat transaksi tiket.

1. booking_id (String): Primary Key (Auto Generated UUID)
2. user_id (String): Foreign Key ke collection Users
3. movie_title (String): Disimpan statis untuk riwayat
4. seats (Array (String)): List kursi (Contoh: [“A1”, “A2”])
5. total_price (Number (Int)): Harga final setelah Logic Trap
6. booking_date (Timestamp): Waktu transaksi dilakukan

3.5 Implementasi Class Model (Protokol Integritas)

Semua Model Data dibuat manual dengan akhiran Jevon.

User: UserModelJevon (File: user_model_jevon.dart) Movie: MovieModelJevon (File: movie_model_jevon.dart) Booking: BookingModelJevon (File: booking_model_jevon.dart)

Setiap class memiliki fungsi `fromMapJevon` dan `toMapJevon`.

3.6 Data Seeding

Proses **seeding** dilakukan melalui class MovieSeederJevon di halaman SeedPageJevon. Data dummy mencakup variasi panjang judul untuk **Logic Trap** (“The Long Title Tax”).

Daftar 10 Produk Dummy (Film):

1. Avatar (Judul Pendek - 6 huruf)
2. Titanic (Judul Pendek - 7 huruf)
3. Spider-Man: No Way Home (Judul Panjang > 10 huruf - Kena Pajak)
4. The Batman (Judul Pas 10 huruf)
5. Avengers: Endgame (Judul Panjang - Kena Pajak)
6. Dune (Judul Pendek)
7. No Time To Die (Judul Panjang - Kena Pajak)
8. The Matrix Resurrections (Judul Panjang - Kena Pajak)
9. Eternals (Judul Pendek)
10. Shang-Chi (Judul Pendek)

BAB 4
Implementasi Halaman Home

4.1 Implementasi Halaman Home

Halaman Home merupakan halaman utama pada aplikasi CineBooking yang menampilkan daftar film yang sedang tersedia. Halaman ini menjadi titik awal interaksi pengguna setelah berhasil login. Tampilan halaman Home dibuat menggunakan widget `Scaffold` yang terdiri dari `AppBar` dan bagian body utama.

```
return Scaffold(
  backgroundColor: const Color(0xFF0F1419),
  appBar: AppBar(
    backgroundColor: const Color(0xFF1A1F29),
  ),
```

Pada bagian `AppBar`, ditambahkan ikon profil di sebelah kiri yang mengarahkan pengguna ke halaman profil, serta tombol logout di sebelah kanan. Proses logout dilengkapi dengan dialog konfirmasi agar pengguna tidak keluar aplikasi secara tidak sengaja.

4.2 Penampilan Data Film pada Halaman Home

Daftar film pada halaman Home ditampilkan menggunakan `StreamBuilder` yang terhubung dengan Firebase Firestore.

```
StreamBuilder(
  stream: FirebaseFirestore.instance.collection("movies").snapshots(),
  builder: (context, snapshot) {
    // implementasi builder
  }
)
```

Penggunaan `StreamBuilder` bertujuan agar tampilan film dapat diperbarui secara otomatis ketika terjadi perubahan data pada database, seperti penambahan atau penghapusan film. Jika data masih dimuat, aplikasi akan menampilkan indikator loading. Apabila tidak terdapat data film, maka akan ditampilkan pesan “No Movies Available” agar pengguna mengetahui kondisi tersebut.

4.3 Tampilan Daftar Film Menggunakan Grid

Film ditampilkan dalam bentuk grid agar tampilannya lebih rapi dan mudah dilihat. Implementasi grid dilakukan menggunakan `GridView.builder`.

```
GridView.builder(
  gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
    crossAxisCount: gridCount,
  ),
  itemBuilder: (context, i) {
    return MovieCardAdel(movie: movies[i], onTap: () {});
  },
)
```

Jumlah kolom grid disesuaikan dengan ukuran layar menggunakan `MediaQuery`. Jika layar berukuran besar, jumlah kolom akan bertambah sehingga tampilan tetap responsif di berbagai perangkat.

4.4 Implementasi Komponen Movie Card

Setiap film ditampilkan menggunakan komponen `MovieCardAdel`. Komponen ini menampilkan poster film, judul, rating, harga tiket, dan durasi film.

```
InkWell(
  onTap: onTap,
  child: Container(
    decoration: BoxDecoration(
      borderRadius: BorderRadius.circular(16),
    ),
  ),
)
```

Widget `InkWell` digunakan agar kartu film dapat ditekan dan memberikan respon visual kepada pengguna. Poster film ditampilkan menggunakan `Image.network` dan dilengkapi dengan error handler jika gambar gagal dimuat. Selain itu, ditambahkan efek bayangan (shadow) dan gradasi warna agar tampilan kartu film terlihat lebih menarik dan modern.

4.5 Halaman Detail Film

Halaman detail film digunakan untuk menampilkan informasi lengkap dari film yang dipilih. Tampilan halaman ini dibuat menggunakan `CustomScrollView` dan `SliverAppBar`.

```
SliverAppBar(
  expandedHeight: 450,
  pinned: true,
  flexibleSpace: FlexibleSpaceBar(
    background: Image.network(movie.poster_url),
  ),
)
```

Penggunaan `SliverAppBar` memberikan efek animasi saat halaman di-scroll, sehingga tampilan terlihat lebih dinamis. Transisi antar halaman juga menggunakan `Hero` agar perpindahan dari poster film ke halaman detail terlihat lebih halus. Informasi film seperti judul, rating, durasi, dan harga tiket ditampilkan dalam bentuk kartu informasi agar mudah dibaca oleh pengguna.

4.6 Navigasi ke Halaman Pemilihan Kursi

Pada bagian bawah halaman detail film terdapat tombol `Select Seats` yang digunakan untuk melanjutkan proses pemesanan tiket.

```
FloatingActionButton.extended(
  onPressed: () {
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (_) => SeatPageIntan(
          movieTitleIntan: movie.title,
```

```
          basePriceIntan: movie.base_price,
        ),
      ),
    );
  },
)
```

Tombol ini memudahkan pengguna untuk langsung menuju halaman pemilihan kursi setelah melihat detail film yang diinginkan.

4.7 Kesimpulan

Berdasarkan implementasi yang telah dilakukan, antarmuka pengguna pada aplikasi CineBooking telah dirancang dengan tampilan yang sederhana, responsif, dan mudah digunakan. Setiap halaman saling terhubung dengan baik melalui navigasi yang jelas, sehingga pengguna dapat melakukan proses pemesanan tiket dengan nyaman. Pengelolaan data dan logika aplikasi ditangani pada modul lain, sedangkan pada bab ini difokuskan pada implementasi tampilan dan interaksi pengguna.

BAB 5
Implementasi Halaman Seat Selection

5.1 Implementasi Halaman Seat Selection

Halaman Seat Selection merupakan halaman untuk memilih kursi bioskop sebelum melakukan pembayaran tiket. Halaman ini dikerjakan oleh Intan sebagai Frontend Engineer pada modul Seat Matrix. Tampilan halaman dibuat menggunakan widget Scaffold dengan bagian body utama menampilkan grid kursi.

```
return Scaffold(
  backgroundColor: const Color(0xFF0F1419),
  appBar: AppBar(
    backgroundColor: const Color(0xFF1A1F29),
    title: Text("Pilih Kursi"),
  ),
```

Pada bagian `AppBar`, ditambahkan tombol kembali (Back) di sisi kiri agar pengguna dapat kembali ke halaman detail film tanpa kehilangan data kursi yang telah dipilih.

5.2 Grid Kursi Dinamis

Kursi ditampilkan dalam bentuk grid menggunakan `GridView.builder` dengan konfigurasi dinamis, misalnya 6×8, agar tampilan kursi lebih rapi dan responsif di berbagai ukuran layar.

```
GridView.builder(
  gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
    crossAxisCount: 6,
  ),
  itemBuilder: (context, index) {
    return SeatItemIntan(
      seatNameIntan: seats[index].name,
      isSoldIntan: seats[index].isSold,
      isSelectedIntan: seats[index].isSelected,
      onTapIntan: () => controller.toggleSeat(seats[index].name),
    );
  },
)
```

Widget kursi dinamai `SeatItemIntan` sesuai ketentuan. Status kursi diperbarui secara real-time menggunakan Provider `SeatControllerSalam`.

5.3 Status Visual Kursi

Setiap kursi memiliki tiga status yang ditandai dengan warna berbeda:

```
if (isSoldIntan) {
  seatColorIntan = Color(0xFFFF0000);
} else if (isSelectedIntan) {
  seatColorIntan = Color(0xFF1A2634);
} else {
```

```
  seatColorIntan = Color(0xFFAAAAAA);
}
```

Abu-abu: kursi tersedia, Biru: kursi dipilih (user dapat toggle select/unselect), Merah: kursi terjual (tidak bisa diklik)

5.4 Perhitungan Total Harga

Total harga tiket dihitung otomatis berdasarkan jumlah kursi yang dipilih dan harga dasar per kursi. Nilai ini ditampilkan di bagian bawah halaman agar pengguna dapat melihat total pembayaran sebelum melanjutkan.

```
 double totalPrice = controller.calculateTotalPrice();
```

5.5 Navigasi ke pembayaran

Setelah memilih kursi, pengguna dapat menekan tombol Proceed to Payment untuk melanjutkan proses pemesanan tiket.

```
FloatingActionButton.extended(
  onPressed: () {
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (_) => PaymentPage(
          selectedSeats: controller.selectedSeats,
          totalPrice: controller.calculateTotalPrice(),
        ),
      ),
    );
  },
  label: Text("Proceed to Payment"),
)
```

5.6 Kesimpulan

Modul Seat Selection yang dikerjakan oleh Intan berhasil menampilkan grid kursi dinamis dengan status kursi yang jelas, toggle select/unselect interaktif, dan perhitungan harga otomatis. Antarmuka ini responsif dan intuitif sehingga memudahkan pengguna memilih kursi dengan nyaman sebelum melakukan pembayaran tiket.

BAB 6
State Management & Logic

Dikerjakan oleh Transaction Logic Specialist[cite: 37].

6.1 State Booking Seat

State management menggunakan (Provider/Bloc/GetX) untuk fitur **Book Now**[cite: 38, 39]. Pembuatan logika pemesanan kursi dilakukan dengan menggunakan Provider, pada baris pertama terdapat class dengan nama SeatControllerSalam sebagai controller state yang mengatur logika pemesanan kursi pada aplikasi. Class tersebut menggunakan ChangeNotifier sebagai pembaruan UI ketika ada perubahan data.

6.2 Inisialisasi dan Deklarasi Variabel

```
final List<String> _selectedSeatsSalam = [];
List<String> get selectedSeats => _selectedSeatsSalam;
List<String> _soldSeatsSalam = [];
List<String> get soldSeats => _soldSeatsSalam;

int basePriceSalam = 0;
String movieTitleSalam = "";
StreamSubscription? _bookingSubscriptionSalam;
```

`_selectedSeatsSalam` variabel yang berfungsi untuk menyimpan kursi yang dipilih oleh pengguna melalui state lokal. `_soldSeatsSalam` variabel yang berfungsi untuk meyimpan kursi yang sudah dibeli, data diambil dari real-time Firestore. `basePrice` harga dasar untuk setiap kursi. `movieTitle` untuk menyimpan judul film yang dipilih. `_bookingSubscription` untuk menyimpan hasil monitoring perubahan data dari Firestore secara realtime.

6.3 InitData() - Mengatur data awal

```
void initData({
  required int basePriceInput,
  required String movieTitleInput,
}) {
  basePriceSalam = basePriceInput;
  movieTitleSalam = movieTitleInput;
  _listenSoldSeatsSalam(movieTitleSalam);
}
```

6.4 Logika Toggle dan Checkout

6.4.1 Toggle Seat

```
void toggleSeat(String seat) {
  if (_selectedSeatsSalam.contains(seat)) {
    _selectedSeatsSalam.remove(seat);
  } else {
    _selectedSeatsSalam.add(seat);
  }
  notifyListeners();
}
```

`toggleSeat(String seat)` memungkinkan pengguna memilih atau membatalkan pilihan kursi, kursi ditambahkan ke `_selectedSeatsSalam` jika belum dipilih, atau dihapus jika sudah ada, setiap perubahan memanggil `notifyListeners()` agar UI update.

6.4.2 Perhitungan Total Harga

```
int calculateTotalPrice() {
  int total = basePriceSalam * _selectedSeatsSalam.length;
  if (movieTitleSalam.length > 10) {
    total = total + (2500 * _selectedSeatsSalam.length);
  }
  for (var seat in _selectedSeatsSalam) {
    final number = int.tryParse(seat.substring(1)) ?? 0;
    if (number % 2 == 0) {
      total -= (basePriceSalam * 0.10).toInt();
    }
  }
  return total;
}
```

Logika diatas menghitung total berdasarkan jumlah kursi x harga dasar yang didapat dari `basePriceSalam`, tambahan biaya 2500 per kursi jika judul film lebih dari 10 karakter, Diskon 10% untuk kursi genap diambil dari nomor kursi. Total harga akan otomatis digunakan saat checkout.

6.4.3 Checkout

```
  Future<String?> checkout(String userId) async {
    try {
      final result = await InternetAddress.lookup('google.com');
      if (result.isEmpty || result[0].rawAddress.isEmpty) {
        return "Tidak ada koneksi internet. Silahkan cek jaringan Anda";
      }
    } catch (_) {
      return "Tidak ada koneksi internet. Silahkan cek jaringan Anda";
    }

    if (_selectedSeatsSalam.isEmpty || basePriceSalam == 0 ||
movieTitleSalam.isEmpty) {
      return "Data booking tidak lengkap";
    }

    final bookingCollection =
FirebaseFirestore.instance.collection('bookings');
    final docRef = bookingCollection.doc();
    final bookingId = docRef.id;
    final data = {
      'booking_id': bookingId,
      'user_id': userId,
      'movie_title': movieTitleSalam,
      'seats': _selectedSeatsSalam,
      'total_price': calculateTotalPrice(),
      'booking_date': Timestamp.now(),
```

```
    };

    await docRef.set(data);
    await
FirebaseFirestore.instance.collection("bookings").where("movie_title",
isEqualTo: movieTitleSalam).snapshots().first;
    _selectedSeatsSalam.clear();
    notifyListeners();
    return null;
  }
```

Melakukan checkout dengan memeriksa koneksi internet terlebih dahulu. Jika tidak ada, akan mengembalikan pesan error dan validasi data booking. Untuk validasi data booking, jika kursi belum dipilih, harga belum diatur, atau judul film kosong, maka akan mengembalikan pesan error. Penyimpanan data booking ke Firestore di koleksi `bookings`, field yang disimpan `booking_id`, `user_id`, `movie_title`, `seats`, `total_price`, `booking_date`. Setelah booking berhasil, `_selectedSeatsSalam` akan dikosongkan untuk memulai transaksi baru.

BAB 7
Implementasi Antarmuka & Auth

7.1 Halaman Login & Register (Fariz)

Halaman Login dan Register digunakan untuk proses autentikasi pengguna sebelum mengakses aplikasi CineBooking. Modul ini mengelola input nama, email, dan password menggunakan TextEditingController untuk memudahkan validasi data.

```
class _RegisterPageBioskopFarizState extends State<RegisterPageBioskopFariz> {
  final nameControllerFariz = TextEditingController();
  final emailControllerFariz = TextEditingController();
  final passControllerFariz = TextEditingController();
  bool obscureFariz = true;
  bool isLoadingFariz = false;
  bool nameErrorFariz = false;
  bool emailErrorFariz = false;
  bool passErrorFariz = false;

  final regexEmailFariz = RegExp(r'^[a-zA-Z0-9._%+-]+@poliwangi\.ac\.id$');
```

Validasi email diterapkan dengan RegExp untuk memastikan pengguna menggunakan domain resmi kampus, sehingga akses aplikasi dibatasi hanya untuk pengguna yang valid. Selain itu, password divalidasi agar memenuhi ketentuan keamanan yang telah ditentukan.

7.1.1 Implementasi `SharedPreferences` untuk remember me

Fitur remember me diimplementasikan menggunakan SharedPreferences untuk menyimpan status login pengguna secara lokal pada perangkat. Status login disimpan dalam bentuk nilai boolean dengan key isLoggedIn.

```
Future<void> farizCheckLoginStatus() async {
    final prefs = await SharedPreferences.getInstance();
    bool isLoggedIn = prefs.getBool('isLoggedIn') ?? false;

    if (isLoggedIn) {
      if (!mounted) return;
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (_) => const HomePageAdel()),
      );
    }
  }
```

Saat aplikasi dijalankan, fungsi farizCheckLoginStatus() akan mengecek status login. Jika isLoggedIn bernilai true, pengguna langsung diarahkan ke halaman Home menggunakan Navigator.pushReplacement, sehingga halaman login tidak dapat diakses kembali melalui tombol back. Selain itu, fitur remember me juga menyimpan email dan password pengguna jika opsi diaktifkan. Data tersebut akan dimuat kembali secara otomatis ketika aplikasi dibuka.

```
Future<void> farizLoadLoginData() async {
    final prefs = await SharedPreferences.getInstance();
    setState(() {
      farizRememberMe = prefs.getBool('rememberMe') ?? false;
      if (farizRememberMe) {
        farizEmailController.text = prefs.getString('savedEmail') ?? "";
        farizPasswordController.text = prefs.getString('savedPassword') ?? "";
      }
    });
  }
```

Data login disimpan atau dihapus berdasarkan kondisi remember me yang dipilih pengguna.

```
Future<void> farizSaveLoginData(String email, String password) async {
    final prefs = await SharedPreferences.getInstance();
    if (farizRememberMe) {
      await prefs.setString('savedEmail', email);
      await prefs.setString('savedPassword', password);
      await prefs.setBool('rememberMe', true);
    } else {
      await prefs.remove('savedEmail');
      await prefs.remove('savedPassword');
      await prefs.setBool('rememberMe', false);
    }
  }
```

7.2 Error handling dengan snackbar

Penanganan kesalahan pada halaman Login dan Register diimplementasikan menggunakan SnackBar untuk memberikan umpan balik langsung kepada pengguna. Pendekatan ini memastikan pengguna mengetahui kesalahan input atau kegagalan proses autentikasi tanpa keluar dari halaman. Sistem terlebih dahulu melakukan validasi input dasar. Jika kolom email atau password belum diisi, maka aplikasi akan menampilkan pesan kesalahan.

```
if (emailErrorFariz || passErrorFariz) {
      farizShowErrorSnackbar("Please fill in all fields");
      return;
    }
```

Selanjutnya, email divalidasi menggunakan regular expression. Jika email tidak sesuai dengan domain kampus, maka pesan kesalahan akan ditampilkan.

```
if (!regexEmail.hasMatch(email)) {
      farizShowErrorSnackbar("Email must use @poliwangi.ac.id");
      return;
    }
```

Pada proses autentikasi dengan Firebase, aplikasi menangani berbagai kemungkinan error, seperti email tidak terdaftar, password salah, atau format email tidak valid. Setiap kesalahan ditampilkan dengan pesan yang sesuai agar mudah dipahami oleh pengguna.

```
} on FirebaseAuthException catch (e) {
      String message = "Login failed";
```

```
      if (e.code == 'user-not-found') message = "No user found for that
email";
      if (e.code == 'wrong-password') message = "Wrong password provided";
      if (e.code == 'invalid-email') message = "Invalid email format";
      farizShowErrorSnackbar(message);
    } finally {
      if (mounted) setState(() => farizIsLoading = false);
    }
  }
```

Dengan penggunaan SnackBar, proses error handling menjadi lebih informatif dan meningkatkan pengalaman pengguna tanpa mengganggu alur aplikasi.

BAB 8
Penutup dan Pengujian

Berdasarkan hasil pengujian yang telah dilakukan sesuai dengan skenario yang ditetapkan, dapat disimpulkan bahwa aplikasi CineBooking telah memenuhi seluruh kriteria fungsional dan logika bisnis yang diwajibkan.

8.1 Pengujian Trap Judul

Pada pengujian Trap Judul, pemilihan film dengan judul pendek “Avengers” (8 karakter) dan judul panjang “Spider-Man: No Way Home” (21 karakter) menunjukkan perbedaan harga tiket. Film dengan judul lebih dari 10 karakter dikenakan tambahan Long Title Tax, sehingga harga dasar per kursi meningkat sesuai ketentuan. Hal ini membuktikan bahwa logika penambahan pajak berdasarkan panjang judul film telah diimplementasikan dengan benar.

8.2 Pengujian Trap kursi

Pada pengujian Trap Kursi, pemilihan kursi A1 (ganjil) dan A2 (genap) menghasilkan total harga yang tidak dihitung secara linear. Kursi ganjil dikenakan harga normal, sedangkan kursi genap mendapatkan diskon sebesar 10%, sehingga total harga sesuai dengan rumus HargaDasar + (HargaDasar − 10%). Hasil ini memastikan bahwa aturan Odd/Even Seat Rule diterapkan secara manual dan akurat.

8.3 Pengujian Crash Handling

Pada pengujian Crash Handling, ketika koneksi internet dimatikan saat tombol “Bayar” ditekan, aplikasi tidak mengalami force close. Sistem menampilkan pesan kesalahan kepada pengguna melalui notifikasi, sehingga pengguna tetap berada di dalam aplikasi. Hal ini menunjukkan bahwa mekanisme error handling telah berjalan dengan baik dan aplikasi memiliki stabilitas yang memadai.

8.4 Video Demo Kode

https://drive.google.com/drive/folders/1QkQTW50Yjv8zw_441B2LT8Kr62dnattT?usp=sharing