LAPORAN UJIAN AKHIR SEMESTER

MATA KULIAH PEMROGRAMAN BACKEND LANJUTAN

“Sistem Pemesanan Tiket: Bioskop”

DISUSUN OLEH:

Faza Ulul Ilma **434231030**

Nahdah Zafira Br Tb **434231024**

Kamilatus Sa’adah **434231035**

PROGRAM STUDI D-IV TEKNIK INFORMATIKA

FAKULTAS VOKASI

UNIVERSITAS AIRLANGGA

SURABAYA

2025

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring meningkatnya jumlah penonton bioskop, terutama untuk film populer, pemesanan tiket secara manual atau melalui sistem lama menimbulkan beberapa permasalahan utama:

1. Double Booking:

   Dua atau lebih pengguna dapat memesan kursi yang sama secara bersamaan. Hal ini menyebabkan kursi tercatat lebih dari satu kali sebagai booked sehingga data menjadi tidak konsisten.
2. Race Condition:

   Terjadi ketika dua proses bersaing untuk mengubah status kursi secara hampir bersamaan. Contoh: User A dan User B melihat kursi A1 available hampir bersamaan, lalu keduanya mengirim request booking → tanpa mekanisme penguncian atau versioning, kursi bisa double booked.
3. High Traffic:

   Saat film populer ditayangkan, banyak user yang mengakses sistem bersamaan. Sistem harus mampu menangani konkurensi tinggi agar tetap responsif dan akurat.

Karena permasalahan tersebut, diperlukan sistem backend yang aman, cepat, dan dapat mencegah konflik data saat pemesanan tiket. Sistem ini juga harus mampu menahan kursi sementara (on-hold) sebelum booking selesai untuk mengurangi risiko double booking.

1.2 Tujuan Sistem

Sistem ini dirancang untuk mempermudah pengguna memesan tiket bioskop secara online. Tujuan utama:

- Pengguna dapat memilih film, jadwal, dan kursi.
- Memesan satu atau beberapa kursi sekaligus dengan aman.
- Membatalkan booking jika diperlukan.
- Menahan kursi sementara untuk mencegah double booking dan race condition.

1.3 Ruang Lingkup

a. Fitur utama:

   - Melihat seat map per studio/jadwal.

- Melakukan booking kursi.
- Membatalkan booking.
- Menahan kursi sementara (hold seat).

b. Tidak termasuk pembayaran atau integrasi pihak ketiga.

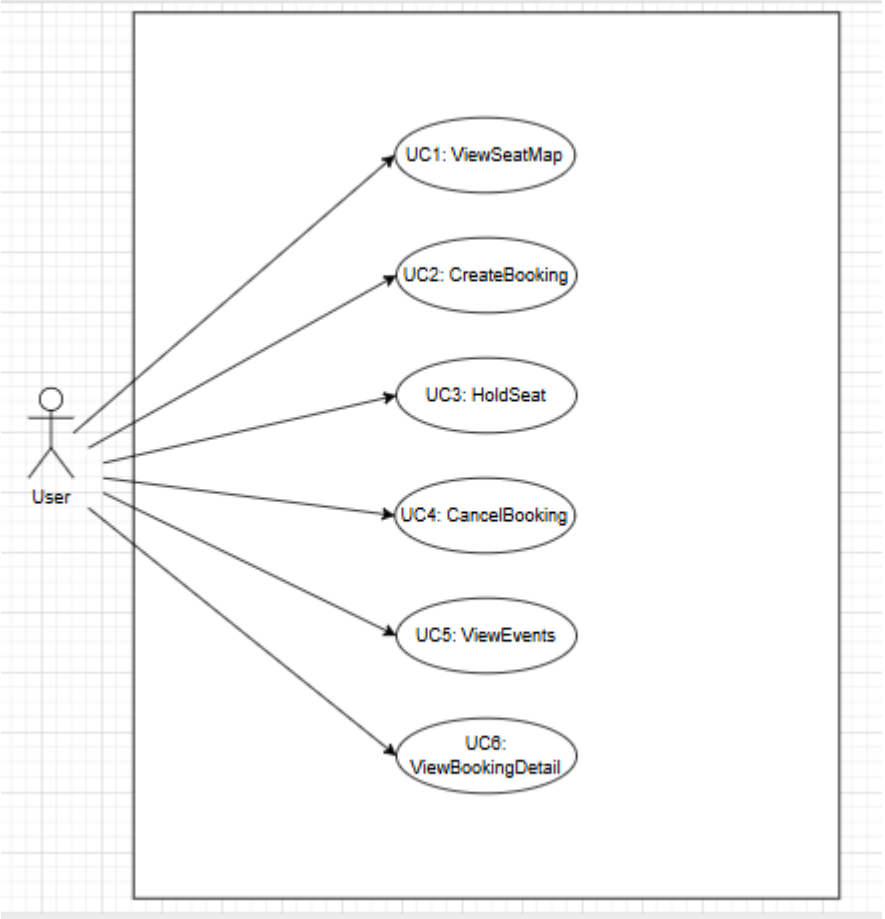
BAB II
Analisa Domain dan Use Case

2.1 Deskripsi Domain

Entitas	Deskripsi
Film/Event	Film yang ditayangkan di bioskop dengan jadwal tertentu
Studio	Ruangan tempat film diputar
Seat/Kursi	Tempat duduk di studio, status: available/booked/on-hold
Booking	Pemesanan kursi oleh user, info film, studio, kursi, user
User	Orang yang menggunakan sistem untuk memesan tiket

2.2 Use Case

Pada sistem informasi pemesanan bioskop ini, aktor dan use case yang berinteraksi menggambarkan kebutuhan fungsional sistem. Aktornya merupakan user, yaitu pengguna yang melakukan pemesanan tiket, melihat jadwal, mengecek status kursi, dan mengelola booking yang telah dibuat. Use Case Diagram dibawah dibuat untuk menunjukkan fitur-fitur yang dapat dilakukan oleh pengguna:



UC1 - ViewSeatMap

Use Case ini memungkinkan pengguna untuk melihat status seluruh kursi pada suatu jadwal (event). Sistem akan menampilkan status kursi (available, on_hold, atau taken)

UC2 - CreateBooking

Use Case digunakan ketika pengguna melakukan pemesanan kursi untuk sebuah event. kursi yang tersedia, sistem akan memverifikasi ketersediaan kursi tersebut dan kemudian membuat booking baru. Kursi yang berhasil dipesan akan berubah status menjadi *taken* dan dicatat dalam data booking pengguna.

UC3 - HoldSeat

Use case ini berfungsi untuk menahan satu atau lebih kursi selama proses checkout. Ketika pengguna mulai melakukan proses pemesanan, sistem akan mengubah status kursi menjadi *on_hold* selama periode waktu tertentu untuk mencegah pengguna lain mengambil kursi yang sama. Jika pengguna tidak menyelesaikan transaksi dalam waktu yang ditentukan, status kursi akan dikembalikan menjadi *available*.

UC4 - CancelBooking

Pengguna dapat membatalkan pemesanan yang telah dibuat melalui use case ini. Ketika pembatalan dilakukan, sistem akan memperbarui status booking menjadi *cancelled* dan seluruh kursi yang terkait dikembalikan ke status *available* sehingga dapat dipesan kembali oleh pengguna lain.

UC5 - ViewEvents

Use case ini memungkinkan pengguna untuk melihat daftar event atau jadwal film yang tersedia. Informasi yang ditampilkan meliputi judul film, studio, tanggal, dan waktu penayangan. Pengguna dapat memilih salah satu event untuk melihat seat map ataupun memulai proses pemesanan.

UC6 - ViewBookingDetail

Use case ini digunakan untuk melihat detail pesanan tiket yang sudah dibuat oleh pengguna. Sistem akan menampilkan informasi seperti kursi yang dipesan, jadwal film, status booking, dan waktu pemesanan. Fitur ini membantu pengguna memastikan bahwa pesanan telah tercatat dengan benar.

2.3 Use Case Spesification

1. Melihat status kursi pada event

Nama Use Case	UC1- ViewSeatMap
Aktor	User
Deskripsi	User melihat denah kursi (seat map) untuk film dan jadwal tertentu. Sistem menampilkan status setiap kursi (available, booked, atau on-hold).
Prioritas	Tinggi
Status	Aktif
Pre-Condition	Film dan jadwal tersedia.
Post-Condition	User dapat melihat seat map dengan status kursi yang up-to-date.
Basic Path	1. User memilih film. 2. User memilih jadwal. 3. Sistem menampilkan seat map lengkap dengan status kursi.

Alternative Paths	1. User memilih film dan jadwal. 2. Sistem memeriksa ketersediaan film dan jadwal di database. 3. Jika film atau jadwal tidak tersedia, sistem menampilkan pesan error: "Film atau jadwal tidak tersedia".

2. Memesan kursi pada event

Nama Use Case	**UC2 - CreateBooking**
Aktor	User
Deskripsi	User memesan satu atau beberapa kursi untuk film tertentu pada jadwal tertentu. Sistem akan menahan kursi sementara dan mengubah status kursi menjadi booked setelah booking berhasil.
Prioritas	Tinggi
Status	Aktif
Pre-Condition	1. User telah masuk dalam sistem 2. Film dan Jadwal tersedia
Post-Condition	Kursi yang awalnya memiliki stastu available berubah menjadi booked; booking tercatat di database; user mendapatkan konfirmasi booking
Basic Path	1. User memilih film dan jadwal. 2. User memilih kursi. 3. Sistem menahan kursi sementara (on-hold). 4. User menyelesaikan booking. 5. Sistem mengubah status kursi menjadi booked dan menyimpan booking.

Alternative Paths	1. Kursi sudah dibooking oleh user lain → Sistem menampilkan pesan “Kursi sudah dibooking” dan meminta user memilih kursi lain. 2. Timeout hold seat habis sebelum booking selesai → Sistem membatalkan hold seat, kursi kembali available, user diberi notifikasi untuk memilih kursi kembali.

3. Membatalkan pesanan

Nama Use Case	**UC4 - CancelBooking**
Aktor	User
Deskripsi	User dapat membatalkan pemesanan. Kursi yang dibatalkan akan kembali menjadi available.
Prioritas	Sedang
Status	Aktif
Pre-Condition	1. Booking harus ada dalam sistem. 2. Booking belum kadaluarsa atau belum dikonfirmasi.
Post-Condition	1. Booking dibatalkan. 2. Kursi yang terkait dikembalikan menjadi available.
Basic Path	1. User memilih booking yang ingin dibatalkan. 2. Sistem mengecek apakah booking valid dan bisa dibatalkan. 3. Sistem menghapus atau mengubah status booking menjadi canceled. 4. Sistem mengembalikan kursi menjadi available.
Alternative Paths	1. Jika booking tidak ditemukan maka sistem akan memunculkan pesan *“Booking not found”*

	2. Jika booking sudah paid sistem akan menampilkan pesan *"Paid booking cannot be canceled"*

4. Menahan kursi pesanan

Nama Use Case	**UC3 - HoldSeat**
Aktor	User
Deskripsi	User menahan satu atau beberapa kursi agar tidak diambil orang lain selama proses pemesanan berlangsung. Kursi yang ditahan bersifat sementara (durasi 10 menit).
Prioritas	Sangat Tinggi (terkait race condition)
Status	
Pre-Condition	1. Kursi yang dipilih harus dalam status available. 2. User sudah memilih event/jadwal.
Post-Condition	1. Kursi berubah status menjadi hold. 2. Sistem menyimpan waktu kadaluarsa hold.
Basic Path	User memilih kursi. Sistem mengecek status kursi. Sistem melakukan atomic update Jika kursi masih available maka ubah menjadi hold. Sistem mencatat user yang meng-hold kursi tersebut. Sistem mengembalikan informasi bahwa kursi berhasil di-hold.
Alternative Paths	Jika kursi sudah hold/booked maka sistem akan menolak permintaan dan menampilkan pesan "Kursi tidak tersedia."

5. Melihat event

Nama Use Case	**UC5 - ViewEvents**
Aktor	User
Deskripsi	User dapat melihat daftar event atau jadwal film yang tersedia. Informasi yang ditampilkan meliputi judul film, studio, tanggal, dan waktu penayangan. User dapat memilih salah satu event untuk melihat seat map atau memulai proses pemesanan.
Prioritas	Tinggi
Status	Aktif
Pre-Condition	1. Terdapat film dan jadwal yang aktif di sistem. 2. User sudah masuk dalam sistem
Post-Condition	Sistem menampilkan daftar event yang tersedia (judul film dan jadwal tayang)
Basic Path	1. User membuka halaman daftar event/jadwal film. 2. Sistem menampilkan daftar film beserta studio, tanggal, dan waktu penayangan. 3. User memilih event tertentu untuk melihat seat map atau memulai booking.
Alternative Paths	Jika kursi sudah hold/booked maka sistem akan menolak permintaan dan menampilkan pesan “Kursi tidak tersedia.”

6. Melihat detail booking

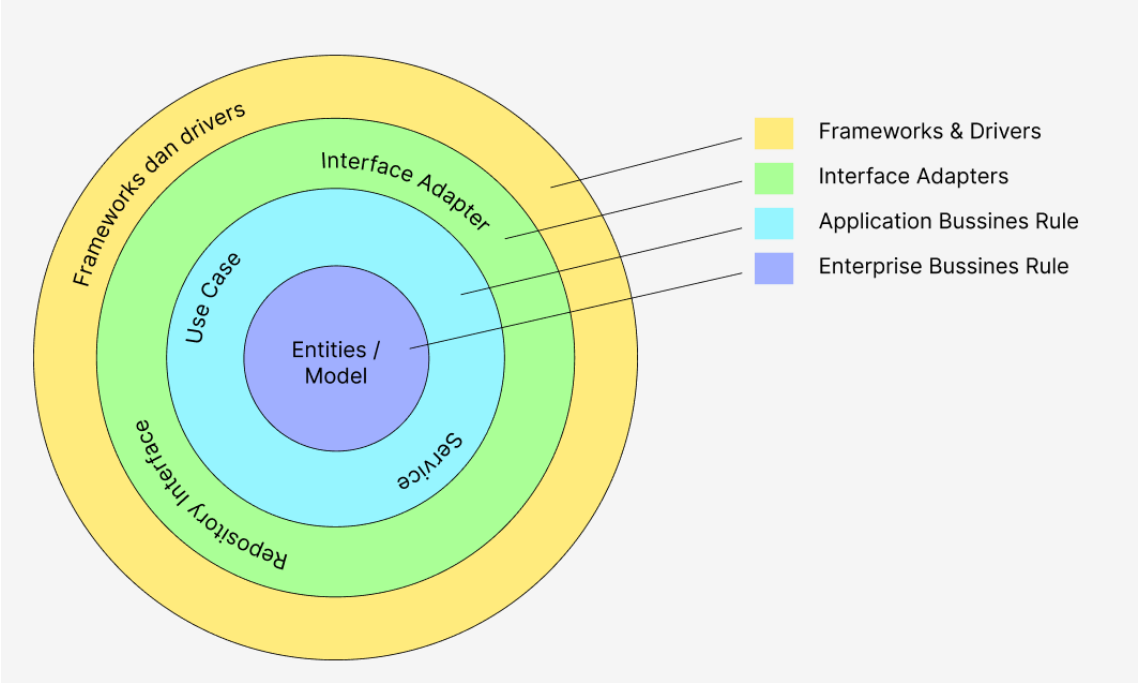
Nama Use Case	**UC6 - ViewBookingDetail**
Aktor	User
Deskripsi	User dapat melihat detail pesanan tiket yang sudah dibuat. Sistem menampilkan informasi kursi yang dipesan, jadwal film, status booking, dan waktu pemesanan. Fitur ini membantu memastikan pesanan telah tercatat dengan benar.

Prioritas	Tinggi
Status	Aktif
Pre-Condition	User telah melakukan booking sebelumnya
Post-Condition	Detail booking ditampilkan oleh sistem
Basic Path	1. User membuka halaman “My Bookings” atau halaman riwayat booking. 2. Sistem menampilkan daftar booking user. 3. User memilih salah satu booking. 4. Sistem menampilkan detail booking: kursi, jadwal film, status, dan waktu pemesanan.
Alternative Paths	1. User membuka halaman “My Bookings” atau halaman riwayat booking. 2. Sistem menampilkan pesan “belum ada tiket yang sedang dibooking”

BAB III
Design Sistem

3.1 Arsitektur Sistem

Arsitektur sistem pada sistem pemesanan tiket akan dibuat menjadi clean Architecture ini dirancang agar proyek tetap terstruktur, mudah dipahami, dan mudah dikembangkan. Arsitektur tidak dibuat terlalu rumit, namun tetap mengikuti prinsip pemisahan tanggung jawab (*separation of concerns*).



Gambar di atas menunjukkan penerapan Clean Architecture pada sistem pemesanan tiket bioskop yang saya rancang. Clean Architecture digunakan agar sistem lebih mudah di-maintain, scalable, testable, dan tidak bergantung pada framework maupun database tertentu.

Arsitektur ini dibagi menjadi empat lapisan konsentris, dari yang paling inti hingga yang paling luar. Setiap lapisan memiliki tanggung jawab dan aturan dependensi yang jelas: ketergantungan kode hanya boleh mengarah dari luar ke dalam, tetapi aturan bisnis di dalam tidak boleh mengetahui apa pun tentang lapisan luar.

1. Enterprise Business Rule (Lapisan Paling Dalam — Entities / Models)

   Lapisan ini berisi definisi **domain model** dan aturan bisnis paling inti dari aplikasi. Pada sistem pemesanan tiket, lapisan ini diisi oleh:

- **Event** (film/jadwal)
- **Seat** (kursi di studio)
- **Booking** (pemesanan)
- **User**

Karakteristik lapisan ini adalah:

- Tidak bergantung pada framework, database, maupun implementasi teknis lainnya.
- Representasi murni dari konsep domain bisnis.
- Stabil (jarang berubah meskipun teknologi luar berubah).

2. Application Business Rule (Use Case / Service Layer)

   Lapisan ini menggambarkan **aturan bisnis aplikasi** atau **use case** yang berjalan di sistem.
   Di struktur proyek, lapisan ini diisi oleh *service-service*, contoh:

   - event_service.go
   - seat_service.go
   - booking_service.go

   Fungsi lapisan ini :

   - Menentukan alur logis sebuah use case.
     Contoh: proses booking kursi → cek kursi available → hold kursi → simpan booking.
   - Mengatur interaksi antara Entities dan Repository Interface.
   - Tidak bergantung pada database atau web framework.
   - Lapisan hanya mengetahui Entities (domain) dan Interface Repository (abstraksi penyimpanan data)

3. Interface Adapters (Controller, Repository Interface, Presenter)

   Lapisan ini berfungsi sebagai **adapter** antara dunia luar dengan use case.

   Isinya meliputi:

   - Repository Interface

     Bertugas untuk mendefinisikan kontrak penyimpanan data dan Use Case memanggil interface ini, bukan MongoDB langsung.

- Routes

  Bertugas untuk Menerima HTTP request dari Fiber, Memparsing input (params, body), Memanggil **Service (Use Case)** dan Mengembalikan response dengan standar JSON.

- Helper

  Bertugas untuk Membentuk format response, Validasi input dasar dan Format waktu.

- Middleware

  Bertugas untuk Otentikasi dan Filter request sebelum masuk ke handler.

4. Frameworks & Drivers (Lapisan Paling Luar)

   Lapisan ini adalah bagian yang berhubungan langsung dengan sistem eksternal dan infrastruktur. Isinya meliputi :
   - Fiber Web Framework (Main.go, routes)
     Bertugas untuk Menyediakan server HTTP, Routing dan Middleware handling
   - Database Driver - MongoDB (database.go)
     Bertugas untuk Mengatur koneksi dan Menyediakan implementasi repository.
   - Config (config.go, env)
     Bertugas untuk Menyimpan konfigurasi aplikasi.
   - Swagger Documentation (docs/swagger.yaml)
     Digunakan oleh framework untuk dokumentasi API.
   - Main Entry Point (Main.go)
     Digunakan untuk Memulai aplikasi, Load config, Connect database, Register routes dan Menjalankan server Fiber.

Berikut meruapakan alur ketergantungan
Controller / Routes → Use Case / Service → Repository Interface → Repository Implementation → MongoDB

1. **Routes (dengan handler)** menerima request dari user dan memanggil **Use Case / Service** sesuai fungsionalitas, misal booking kursi atau melihat seat map.

2. **Use Case / Service** memproses logika aplikasi dan aturan bisnis (Entities), lalu memanggil **Repository Interface** untuk operasi data.

3. **Repository Interface** mendefinisikan kontrak method untuk menyimpan, mengambil, atau memperbarui data, tanpa tergantung ke database spesifik.

4. **Repository Implementation** mengimplementasikan interface tersebut menggunakan **MongoDB**, menjalankan query atau update collection.

5. **MongoDB** menyimpan semua data Event, Seat, Booking, dan User.

3.2 Struktur File / Modul

```
/Ticket bioskop
│
├── app
│   │
│   ├── models
│   │   ├── event.go
│   │   ├── seat.go
│   │   ├── booking.go
│   │   └── user.go
│   │
│   ├── repository
│   │   ├── event_repository.go
│   │   ├── seat_repository.go
│   │   └── booking_repository.go
│   │
│   └── service
│       ├── event_service.go
│       ├── seat_service.go
│       └── booking_service.go
│
├── config
│   └── config.go
│
├── database
│   └── mongodb.go
│
├── routes
│   ├── event_routes.go
│   ├── seat_routes.go
│   └── booking_routes.go
│
├── helper
│   ├── response.go
│   ├── time.go
│   └── validator.go
│
```

```
├── middleware
│   └── auth.go   (opsional)
│
├── docs
│   └── swagger.yaml
│
├── .env
├── go.mod
└── main.go
```

3.3 Desain Database (MongoDB)

Pada studi kasus pemesanan tiket bioskop, MongoDB digunakan karena sifatnya yang fleksibel, cepat dalam membaca data, serta cocok untuk data yang berubah-ubah seperti status kursi (available, hold, booked). Desain skema yang digunakan mengikuti struktur domain: Event, Seat, Booking, dan User. Setiap koleksi disusun untuk mendukung proses melihat seat map, melakukan booking, membatalkan pemesanan, serta melakukan mekanisme hold.

1. Koleksi : Events
   Menyimpan data jadwal pemutaran film atau suatu event tertentu.

Field	**Tipe**	**Deskripsi**
_id	ObjectId	ID unik event
title	string	Judul film
studio	string	Studio pemutaran
show_date	string	Tanggal pemutaran
show_time	time	Jam mulai film
total_seat	int	Total kursi di studio

Contoh struktur dokumen :

```
{
  "_id": ObjectId("64f1c1234abc56789def5555"),
  "title": "Agak Lain 2",
  "studio": "Studio 2",
  "show_date": "2025-12-20",
  "show_time": "18:30",
  "total_seat": 104
}
```

2. Koleksi : Seats
   Menyimpan informasi tiap kursi per event.

Field	Tipe	Deskripsi
_id	ObjectId	ID unik kursi
event_id	ObjectId	referensi ke event
seat_number	string	Nomor kursi
status	string	status kursi : available, on_hold, booked
hold_expire	date	Waktu kadaluarsa hold (jika status on_hold)
user_id	ObjectId	ID user yang menahan atau memesan kursi

   Contoh struktur dokumen :

```
{
  "_id": ObjectId("64f1a2345abc56789def0456"),
  "event_id": ObjectId("64f1a1234abc56789def0123"),
  "seat_number": "A1",
  "status": "available",
  "hold_expire": null,
  "user_id": null
}
```

3. Koleksi : Bookings
   Menyimpan pemesanan kursi oleh user

Field	Tipe	Deskripsi
_id	ObjectId	ID unik booking
user_id	ObjectId	referensi ke user
evet_id	ObjectId	Referensi ke event
seats	array	List kursi yang dibooking
status	string	Status booking: pending, booked, cancelled
booking_time	date	Waktu booking dibuat

total_price	double	Total harga tiket

Contoh struktur dokumen :

```
{
  "_id": ObjectId("64f1a3456abc56789def0789"),
  "user_id": ObjectId("64f1a5678abc56789def0990"),
  "event_id": ObjectId("64f1a1234abc56789def0123"),
  "seats": ["A1", "A2", "A3"],
  "status": "booked",
  "booking_time": "2025-12-07T15:30:00Z",
  "total_price": 150000
}
```

4. Koleksi : users
   Menyimpan data pengguna.

Field	**Tipe**	**Deskripsi**
_id	ObjectId	ID unik user
name	string	Nama user
email	string	email
password	string	password_hashed
phone	string	nomor hp

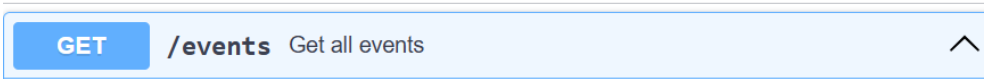
contoh struktur dokumen :

```
{
  "_id": ObjectId("64f1a5678abc56789def0990"),
  "name": "Faza Ilma",
  "email": "faza@example.com",
  "password": "hashed_password",
  "phone": "081234567890"
}
```

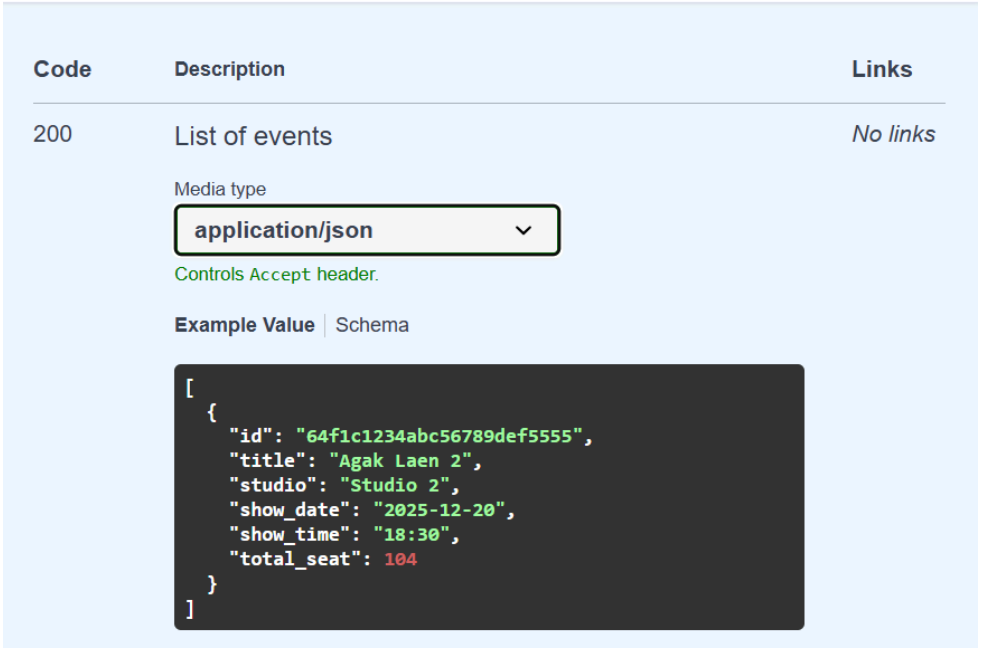
3.4 API

1. Menampilkan daftar film yang tersedia

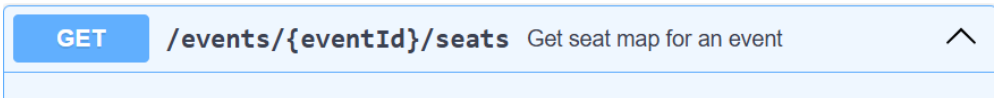
Path :



Response :

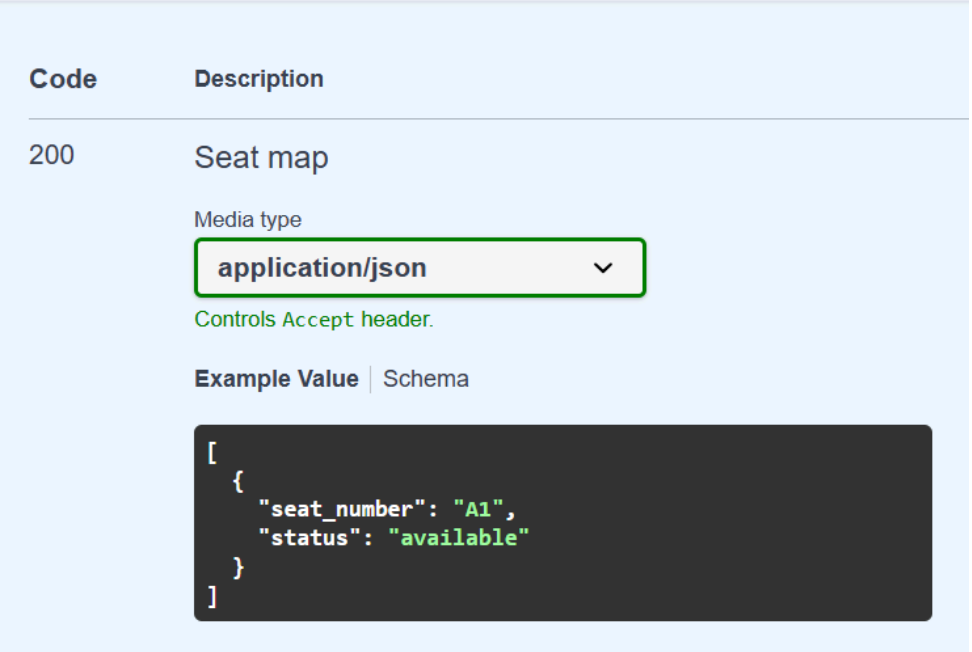


2. Menampilkan status semua kursi pada event tertentu
   Path :

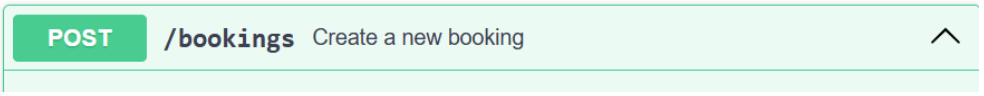


Parameter :

Response :



3. Memesan satu atau beberapa kursi pada event tertentu
   Path :



Request body :

Response :

201 Booking berhasil

Media type

application/json

Controls `Accept` header.

Example Value | Schema

```
{
  "booking_id": "64f1c3456abc56789def8888",
  "status": "booked",
  "seats": [
    "A1",
    "A2",
    "A3"
  ],
  "message": "Booking berhasil"
}
```

400 Kursi sudah dibooking

Media type

application/json

Example Value | Schema

```
{
  "error": "Kursi sudah dibooking",
  "seats_unavailable": [
    "A2"
  ]
```

4. Membatalkan booking yang sudah dibuat
   path :

DELETE /bookings/{bookingId} Cancel a booking

Parameter :

Name	Description
bookingId * required string *(path)*	ID booking bookingId

Response :

Code	Description
200	Booking dibatalkan Media type: application/json Controls Accept header. Example Value | Schema { "booking_id": "64f1c3456abc56789def8888", "status": "cancelled", "message": "Booking berhasil dibatalkan" }
404	Booking tidak ditemukan Media type: application/json Example Value | Schema { "error": "Booking not found" }

5. Menahan kursi sementara agar tidak di booking orang
path :

POST /events/{eventId}/hold Hold seats temporarily

Parameter :

Name	Description
eventId * required string (path)	ID event eventId

Request body :

Example Value | Schema

```
{
  "user_id": "64f1c5678abc56789def7777",
  "seats": [
    "A4",
    "A5"
  ]
}
```

Response :

200 Kursi berhasil di-hold

Media type

application/json

Controls Accept header.

Example Value | Schema

```
{
  "held_seats": [
    "A4",
    "A5"
  ],
  "expire_at": "2025-12-07T15:40:00Z",
  "message": "Kursi berhasil di-hold selama 10 menit"
}
```

400 Kursi tidak tersedia

Media type

application/json

Example Value | Schema

```
{
  "error": "Kursi tidak tersedia",
  "seats_unavailable": [
    "A4"
  ]
}
```

3.5 Analisis Race Condition

1. Permasalahan

Race condition terjadi ketika dua pengguna secara bersamaan memilih kursi yang sama dan mengirimkan permintaan booking hampir dalam waktu bersamaan.
Jika sistem tidak memiliki mekanisme pengaman, kedua permintaan dapat terbaca bahwa kursi masih “available”, sehingga keduanya berhasil memesan kursi yang sama (*double booking*), menyebabkan inkonsistensi data dan kerugian operasional.

2. Solusi
   Untuk mencegah double booking, terdapat dua pendekatan utama yang dapat diterapkan pada MongoDB.
   Solusi 1: MongoDB Transaction (Atomic Multi-Document Operation)
   Transaksi digunakan untuk mengelompokkan beberapa operasi (cek ketersediaan kursi dan update status kursi) ke dalam satu unit kerja yang bersifat atomic. Apabila salah satu operasi gagal—misalnya kursi sudah dibooking oleh pengguna lain—maka seluruh transaksi otomatis dibatalkan (rollback).
   Cara Kerja:
   1. Sistem membuka transaksi
   2. Mengecek kursi dengan kondisi status = “available”
   3. Mengubah status menjadi “booked”
   4. Menyimpan data booking
   5. Jika proses lain mengakses kursi yang sama, transaksinya akan gagal

   Kelebihan
   - Konsistensi sangat tinggi.
   - Aman untuk operasi yang melibatkan banyak dokumen (kursi, event, booking).
   - Cocok untuk kasus pemesanan tiket dengan traffic padat.

   Kekurangan
   - Performa lebih lambat dibanding tanpa transaksi, karena membutuhkan locking proses database.
   - Mengharuskan MongoDB menggunakan Replica Set untuk mendukung transaksi.

   Solusi 2: Optimistic Locking dengan Versioning
   Optimistic locking menggunakan versi atau timestamp untuk memastikan update hanya terjadi jika data belum berubah sejak terakhir dibaca. Sistem tidak mengunci kursi saat dicek, tapi mengecek versi data saat update. Jika versi berbeda, update gagal.
   Cara Kerja Singkat

1. Sistem membaca kursi: { status: "available", version: 1 }
2. Sistem mencoba update dengan kondisi versi yang sama
3. Jika versi cocok → update berhasil, version → 2
4. Jika versi tidak cocok → update gagal → kursi dianggap telah dibooking pengguna lain

Kelebihan

- Tidak menggunakan locking → lebih cepat dan ringan.
- Skalabilitas tinggi.
- Cocok ketika konflik jarang terjadi.

Kekurangan

- Jika traffic sangat tinggi, konflik akan sering, sehingga update gagal berkali-kali.
- Memerlukan mekanisme retry di sisi server agar user tidak terus gagal.

Solusi Terbaik

Berdasarkan kebutuhan sistem pemesanan kursi bioskop yang menuntut konsistensi tinggi, kemampuan pemrosesan multi-seat, serta risiko konflik yang besar pada situasi high traffic, maka MongoDB **Transaction** merupakan solusi terbaik untuk mencegah double booking. Pendekatan ini memberikan tingkat keamanan data yang lebih tinggi, mekanisme atomic yang jelas, serta kontrol penuh terhadap pemrosesan paralel, sehingga lebih sesuai untuk sistem reservasi real-time seperti pemesanan tiket bioskop.

BAB IV
Testing

No	Use Case	Test Case	Kondisi Awal	Input	Expected Output / Behaviour
1	ViewSeatMap	Berhasil menampilkan seat map	Event & jadwal valid; kursi memiliki status available/on_hold /booked	User pilih film → pilih jadwal/event	Sistem menampilkan seluruh kursi dengan status
2	ViewSeatMap	Event tidak ditemukan	Event tidak ada	User pilih event yang tidak valid	Error: “film dengan jadwal tersebut tidak ditemukan”
3	HoldSeat	Hold berhasil	Kursi available	User X hold seat A1 (timeout 10 menit)	Kursi berubah menjadi on_hold oleh user X dengan timestamp
4	HoldSeat	Hold gagal	Kursi on_hold atau booked	User X hold seat A1	Error: “Kursi tidak tersedia untuk ditahan”
5	CreateBooking	Booking berhasil (seat on_hold milik user yang sama)	Kursi A1 on_hold oleh user X	User X melakukan confirm booking	Booking status: booked, kursi berubah booked
6	CreateBooking	Booking gagal – kursi tidak on_hold user	Kursi A1 on_hold oleh user lain	User X confirm booking	Error: “Kursi sedang ditahan user lain”
7	CreateBooking	Booking gagal – kursi sudah booked	Kursi A1 booked	User X confirm booking	Error: “Kursi sudah dibooking”

8	CancelBooking	Cancel berhasil (booking pending)	Booking status pending, kursi on_hold	User X klik cancel pada pending booking	Booking → cancelled, kursi kembali available
9	CancelBooking	Cancel gagal – booking tidak ditemukan	Tidak ada booking dengan ID tersebut	User X cancel booking ID salah	Error: “Booking tidak ditemukan”
10	CancelBooking	Cancel gagal – booking sudah booked	Status booking = booked	User klik cancel	Error: “Booking sudah terkonfirmasi dan tidak dapat dibatalkan”
11	ViewEvents	Daftar event berhasil ditampilkan	Ada film & jadwal	User meminta list event	Sistem menampilkan judul film, studio, tanggal, jam
12	ViewEvents	Event kosong	Tidak ada jadwal	User meminta list event	Sistem menampilkan pesan: “Tidak ada event tersedia”
13	ViewBooking Detail	Lihat detail berhasil	Booking valid	User X melihat Booking ID	Sistem menampilkan detail booking (kursi, jadwal, status, waktu pesan)
14	ViewBooking Detail	Detail gagal – booking tidak ditemukan	Booking tidak ada	User X buka Booking ID salah	Error: “Booking tidak ditemukan”