# 📅 Event Booking API

A backend-only event management system built with Node.js and MongoDB.  
This project is designed to help you practise backend skills for interviews — no AWS required.

## Core Features

### 1. User Authentication

- Register

- Login (JWT)

- Role-based access (`user`, `admin`)

### 2. Event Management (Admin only)

- Create an event (title, description, date, location, capacity, price)

- Update event details

- Delete event

- List all events (with search & filter)

### 3. Event Booking (User)

- Book a seat for an event (capacity decreases automatically)

- Cancel booking (capacity increases automatically)

- View all bookings by the logged-in user

### 4. Event Listing (Public)

- View upcoming events

- Search events by name, date, or location

- Pagination support

## Data Model Requirements

User:  
- name  
- email (unique)  
- password (hashed)  
- role (`user`, `admin`)  
- createdAt

Event:  
- title  
- description  
- date (must be in the future when creating)  
- location  
- capacity (positive integer)  
- price  
- createdBy (admin ID)  
- createdAt

Booking:  
- userId  
- eventId  
- bookingDate  
- status (`booked`, `cancelled`)

## Validation Rules

- Email must be unique

- Event date must be in the future

- Capacity must be a positive integer

- Users can’t book the same event twice

- Cancelled bookings free up event capacity

## API Endpoints

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Endpoint | Access | Description |
| POST | /auth/register | Public | Create new user |
| POST | /auth/login | Public | Login & get JWT |
| POST | /events | Admin | Create event |
| GET | /events | Public | List events with filters |
| GET | /events/:id | Public | Get event details |
| PUT | /events/:id | Admin | Update event |
| DELETE | /events/:id | Admin | Delete event |
| POST | /bookings/:eventId | User | Book event |
| GET | /bookings | User | View my bookings |
| DELETE | /bookings/:id | User | Cancel booking |

## Extra Challenges

- Add search & filter by location, price range, and date

- Add rate limiting for booking API

- Implement transaction-like logic to handle race conditions when booking

- Add unit tests for booking logic

- Implement soft delete for events instead of hard delete

## Skills Practised

- REST API design

- MongoDB schema relationships (User ↔ Booking ↔ Event)

- JWT authentication & role-based access

- Validation & error handling

- Pagination, filtering, sorting

- Handling concurrency (race conditions)