Title: Ticket Booking API with Concurrency-Safe Locking

Time Limit: 48 hours

Tech (recommended): Node.js, Express, Database(SQL/NoSQL)

Focus Areas:

- Asynchronous programming
- Understanding of systems and not merely CRUD operations
- Clean API design
- Code quality & structure
- Ability to explain their approach

Core Problem

Context: Why This Problem Exists (Read Carefully)

You are building the backend for an event ticketing system, similar to Onlybees.
When a popular event goes live, hundreds or thousands of users may try to purchase the same limited number of tickets at the exact same time.

Here's an example:

- A concert has 5 VIP tickets left.
- At 9:00:00 AM, 20 users click "Book Now" simultaneously.
- All of them send a request to your backend at almost the same instant.

If your backend is not designed correctly, here is what will happen:

Typical Overselling Scenario (Not the outcome we desire)

1. All 20 requests hit the server.
2. Each request reads: remaining = 5.
3. Each request thinks: *"Oh perfect, tickets are available."*
4. Each request subtracts from that outdated remaining.
5. Total tickets sold may become:
   5 available → 20 sold → overselling by 15 tickets.

This is called a race condition, and it is one of the most common failures in ticketing systems, flight booking, hotel booking, and flash-sale platforms.

Your job as a backend engineer is to design the booking flow so that overselling becomes impossible, no matter how many requests arrive at the same time.

In Summary,

You're not simply building CRUD APIS.
You are solving a real backend engineering problem that matters in production:

"How do we guarantee no overselling under high traffic?"

Data Models
(The following are examples for NoSQL. Feel free to use SQL)

Design at least these collections:

1. **Event**

```
Event {
  _id,
  name: string,
  sections: [
    {
      _id,
      name: string,
      price: number,
      capacity: number,
      remaining: number
    }
  ]
}
```

2. **Booking**

```
Booking {
  _id,
  eventId,
  sectionId,
  qty: number,
  createdAt
}
```

You're free to extend this.

Required Endpoints

A. Create Event

POST /events/create

Create an event with sections (including capacity & initial remaining).

B. Get Event

GET /events/:id

Returns event + sections with remaining seats.

C. Create Booking (CRITICAL PART)

POST /book/…

Body:

```
{
  "eventId": "…",
  "sectionId": "…",
  "qty": 3
  ...
}
```

Requirements:

1. **Check availability safely under concurrency.**
2. **Prevent overselling, even if multiple requests for the same section arrive at the same time.**
3. **If enough seats:**
   - Deduct from remaining
   - Create a booking document
   - Return success response
4. **If not enough seats:**
   - Return a clear error response

You must implement some form of locking

D. List Bookings

GET /bookings

Returns all bookings with event & section info.

Locking / Concurrency Task (Mandatory)

You must achieve a locking solution.

You must do ALL of this:

1. **Implement a concurrency-safe booking flow. Only then create a booking.**

2. **Write a small script / test to simulate race condition**
   e.g., A Node script or simple JS that sends 10 parallel booking requests for the same section when only, say, 5 seats remain.
   Then show:
   - How many bookings succeeded
   - That remaining never goes below 0
   - That total qty booked ≤ capacity

   This is the real-world validation of your locking design.

3. **Write a short explanation (in README)**
   - What was the overselling problem?
   - What exact mechanism did they implement?
   - Why is it safe (or safe enough) in this setup?
   - What would they improve in a real production system?

Submission Requirements
You must submit:

1. GitHub repo link
2. Instructions to run (env vars, npm commands)
3. A test or script showing concurrent booking
4. README with a section titled:
   Locking Strategy & Concurrency Handling

For Doubts/Questions:

If you have any questions, reply to this email. If you don't receive a response within 2 hours, you may WhatsApp at +91 8415031939.

You will also be evaluated on how you ask questions and your ability to work independently while still unblocking yourself when needed.