

Evently

Context

Large-scale events often see thousands of people rushing to book tickets at the same time. Behind the scenes, this requires a backend system that can handle **concurrency**, **prevent overselling**, **scale during traffic spikes**, and **provide analytics** for organizers.

Your task is to design and build the backend for **Evently**, a platform where users can browse events, book tickets, and track their bookings — while admins manage events and view insights.

Problem Statement

Design and build a scalable backend system that supports the following:

1. User Features

- Browse a list of upcoming events with details (name, venue, time, capacity).
- Book and cancel tickets, ensuring seat availability is updated correctly.
- View booking history.

2. Admin Features

- Create, update, and manage events.
- View booking analytics (total bookings, most popular events, capacity utilization).

System Design Requirements

1. Concurrency & Race Conditions

- Handle simultaneous booking requests without overselling tickets.
- Discuss techniques like optimistic locking, transactions, or queues.

2. Database Design

- Model users, events, and bookings effectively.
- Ensure integrity between event capacity and active bookings.

3. Scalability

- Design APIs that can handle peak traffic (e.g., thousands of concurrent booking requests).
- Explain choices around caching, database indexing, or sharding.

4. APIs

- Expose clean, RESTful endpoints for all core features.
- Include proper error handling and status codes.

Optional Enhancements (Stretch Goals)

- **Waitlist system:** Users can join a waitlist when an event is full.
- **Seat-level booking:** Allow users to pick specific seats within a venue.
- **Notifications:** Inform users if a waitlist spot opens.
- **Advanced analytics:** Endpoints for most-booked events, cancellation rates, and daily booking stats.
- **Creative features welcome:** Any out-of-the-box ideas that improve the user or admin experience will earn brownie points.

Deliverables

1. **Working Backend Application**
 - A deployed backend service that demonstrates Evently's core features and any enhancements.
 - Must be deployed on a platform like Render, Railway, Heroku, or similar. Share the live API base URL.
2. **Code:** Upload your project to a GitHub repository and share the link. Use a **non-descriptive project name** to avoid plagiarism.
3. **High-Level Architecture Diagram** - A diagram showing the main system components, such as APIs, databases, caching, and concurrency handling mechanisms.
4. **Entity-Relationship (ER) Diagram** - A clear representation of the relationships between key entities (users, events, bookings, etc.).
5. **Documentation** - A short explanation (1–2 pages) covering:
 - Major design decisions and trade-offs (e.g., concurrency handling, database schema).
 - How you approached scalability and fault tolerance.
 - Any creative features or optimizations you added.
 - API documentation (can be as simple as an OpenAPI/Swagger spec or a README with endpoints).
6. **Video Submission** - A 5–7 minute walkthrough showing:
 - The system in action (demo of APIs).
 - Key architectural and design decisions.
 - Challenges you faced and how you solved them.

Evaluation Criteria

1. **System Design & Scalability**
 - How well concurrency, database modeling, and load handling are addressed.

2. **API Quality**

- RESTful structure, clarity, error handling, and documentation.

3. **Code Quality & Maintainability**

- Readability, modularity, and best practices.

4. **Performance**

- Ability to handle load (simulated or discussed in trade-offs).

5. **Creativity & Innovation**

- Unique features, optimizations, or enhancements.

6. **Communication**

- Clarity of explanations in documentation and video.

Eventually is your chance to showcase how you'd design and implement a backend system that balances correctness, performance, and scalability. We're excited to see your solution! ✨