Full-Stack Senior Engineer Case Study

You are tasked with building a high-level prototype of **EventHub: Reviews & RFP Lite**, a simplified event-venue marketplace. You have **24 hours** to deliver a working vertical slice that demonstrates your ability to design, code, and communicate clearly.

Core Use Cases

- Venue Discovery: Search venues by city, capacity, and tags. Sort by rating and reviews.
- RFP Submission: Authenticated buyers can submit RFPs with event details. Venue owners can Accept/Counter/Decline.
- Reviews & Ratings: Buyers can review venues after completed events. Show aggregated ratings.
- Auth: Roles include buyer, venueOwner, and admin.

Data Model (suggested)

- User(id, email, hashed_password, role, created_at)
- Venue(id, name, city, capacity_min, capacity_max, tags[], owner_user_id, created_at)
- Rfp(id, venue_id, buyer_user_id, event_date, headcount, budget_min, budget_max, notes, status, created_at, updated_at)
- Review(id, venue_id, buyer_user_id, rating, text, created_at)
- RatingAggregate(venue_id, avg_rating, count_total, counts_json)

API Surface (minimum)

- POST /auth/signup, POST /auth/login
- GET /venues?filters
- POST /rfps (idempotent), PATCH /rfps/:id
- POST /reviews (server validates eligibility)
- GET /venues/:id/reviews (paginated)

Architecture Expectations

- Backend: Java (Spring Boot/Dropwizard) or Node.js/Express.
- Database: Postgres/MySQL with indexes.
- Async: Message queue for review aggregate updates.
- Caching: Redis for top venues by city/rating.
- Security: JWT sessions, input validation, rate limits.
- Observability: request logging, metrics, error tracing.
- CI/CD: GitHub Actions, Dockerfile, docker-compose.

Frontend Expectations

- Use React or Next.js with TypeScript.
- Pages: / (search), /venues/[id], /dashboard (buyer/venueOwner views).
- Justify CSR vs SSR/ISR for search results.
- Handle loading/error states, ensure accessibility basics.

Deliverables (within 24 hours)

- README with architecture diagram, run instructions, and trade-offs.
- Backend repo: APIs, migrations, tests for review eligibility, RFP idempotency.
- Frontend repo: basic UI flows, typed API client, accessibility.
- OpenAPI/Postman collection.
- Sample seed data (venues, reviews, RFPs).

Evaluation Rubric

- System Design & Trade-offs (20 pts)
- Backend Quality (25 pts)
- Frontend Quality (20 pts)
- Security & Reliability (15 pts)
- Performance (10 pts)
- Developer Experience & Docs (10 pts)

You are expected to deliver a **high-level functional prototype** in 24 hours. Focus on correctness, clarity, and demonstrating senior-level judgment in design and implementation choices.