beyond gravity

24H Backend case study

Context

Your team is building a global monitoring system for natural disasters which ingests real-time earthquake data, stores it efficiently, and makes it available for visualization and downstream analytics.

Your mission

Build a secure, scalable backend for a real-time earthquake monitoring system. Your responsibilities include ingesting data from external sources, storing it efficiently, exposing secure APIs, and simulating real-time updates.

Requirements

- Use Python (FastAPI preferred) or Node.js
- Ingest earthquake data from USGS
- Store data in PostgreSQL with a scalable schema
- Implement secure RESTful API endpoints:
    - Authenticated access (e.g., OAuth2)
    - GET /earthquakes: list with filters
    - GET /earthquakes/{id}: detailed view
- Simulate real-time updates using SSE or WebSocket
- Add logging and error handling
- Containerize the service using Docker
- Write a README with setup instructions

Deliverables

GitHub repo with:

- Source code
- Docker setup (compose and file)
- README with Setup instructions and design decisions
- Sample data or ingestion script
- API documentation (e.g., inline or OpenAPI)
- Authentication setup (mock or real)
- Brief notes on design decisions
- Short document with your assumptions and any limitations

Bonus (Optional)

- Use PostGIS or pgvector for geospatial/vector support

- Add RBAC
- Integrate basic monitoring (e.g., Prometheus)
- Include unit and integration tests
- Simulate on-prem deployment (e.g., config separation)

Evaluation Criteria

- clear separation of concerns is proven, scalable design and structure
- Clean, maintainable code with unit/integration testing
- OAuth2 implementation, secure endpoints
- RESTful design, filtering, pagination
- Basic logging, error tracking, performance awareness
- Docker compose and PostGIS or RBAC (if applicable)
- Documentation and ease of setup

We understand this case study requires a meaningful time investment, especially during a busy week. While we don't expect perfection, the goal is to get a deeper sense of how you approach problems, structure your work, and make technical decisions under realistic constraints. Do your best and prioritize on the elements you think are most important in the given time – we then discuss them together.

Happy coding,
Your Beyond Gravity Downstream team!