TurbineOps Lite — Full■Stack Engineering Case Study

Duration: 48 – 72 hours

Objective

Implement a minimal but production■ready **Turbine → Inspection → Findings → Repair Plan** workflow. Demonstrate end■to■end capability in architecture, coding, testing, documentation, and deployability. Candidates may optionally replace MongoDB with AWS DynamoDB for the NoSQL component.

Functional Scope

1. CRUD Turbine (name, location lat/lng, manufacturer, mw_rating).
2. CRUD Inspection linked to Turbine (date, inspector_name, data_source, raw_package_url).
3. Prevent overlapping inspections on same turbine/date.
4. Findings with category, severity, estimated_cost, notes. Apply rule: if BLADE_DAMAGE + 'crack' → severity ≥ 4.
5. Generate Repair Plan summarizing findings, total cost, and priority (HIGH ≥ 5, MEDIUM 3–4, LOW else).
6. Realtime UI update via WebSocket/SSE when a Repair Plan is generated.
7. Filter inspections by date range, turbine, and data source. Text search on Findings notes.
8. Three roles: ADMIN, ENGINEER, VIEWER with proper authorization.

Technical Requirements

1. Frontend — React / Angular / Vue (TypeScript preferred).
2. Backend — Node.js + Express + Apollo GraphQL + REST.
3. Database — PostgreSQL via Prisma (Primary). Optional NoSQL component (MongoDB or AWS DynamoDB).
4. Containerization — Docker (+ docker■compose for Postgres & NoSQL).
5. Auth — JWT with seeded users (admin/engineer/viewer).
6. Testing — unit + integration for core flows + frontend component tests.
7. Documentation — OpenAPI / Swagger for REST, GraphQL SDL, INSTALL & ARCHITECTURE docs.
8. CI/CD awareness — simple GitHub Actions workflow included.

Deliverables

1. Working app reachable via docker■compose up.

2. REST and GraphQL APIs with documentation.
3. SQL schema and migrations included.
4. Optional NoSQL (DynamoDB/Mongo) usage for inspection logs.
5. Frontend with Turbines, Inspections, Findings and Repair Plan UI.
6. Unit + integration tests run green in one command.
7. Comprehensive docs: INSTALL, API, ARCHITECTURE, DB_SCHEMA, TESTING.

Evaluation Criteria (100 pts + bonus)

1. Architecture & Code Quality — 20 pts.
2. Backend Correctness — 20 pts.
3. Data Design (SQL + NoSQL) — 15 pts.
4. Frontend UX & Components — 15 pts.
5. Testing Depth — 15 pts.
6. DevEx & Deployability — 10 pts.
7. Documentation — 5 pts.
8. Stretch features (Background jobs, metrics, multi■tenant, etc.) — bonus 10 pts.

Submission Guidelines

Fork the starter repo or extract the provided ZIP. Push to your own public GitHub repository. Include clear commit history, instructions, and a short README summarizing your design decisions.