NestJS Backend Hiring Assignment (2 Hours)

Project Title: Multi-Tenant Issue & Activity Management API

Time Limit: 2 Hours (Strict)

This assignment evaluates backend engineering skills using NestJS, including module design, controllers and services, guards, database modeling, multi-tenancy, and authorization. Completing every feature is not mandatory. Correct architectural decisions and backend enforcement are valued more than feature completeness.

Technology Requirements (Mandatory)

1. Node.js (runtime)
2. NestJS (latest stable version)
3. TypeScript
4. Database: PostgreSQL / SQLite / MongoDB
5. ORM recommended: Prisma or TypeORM
6. REST APIs (JSON)
7. Frontend or UI is NOT required

Problem Statement

Build a multi-tenant backend API using NestJS for an internal Issue Management system used by multiple organizations. Each organization (tenant) must have strictly isolated data. Users belong to exactly one organization and have one role: ADMIN or MEMBER.

Authentication is NOT required. You may mock user context (userId, organizationId, role) using request headers, a custom middleware, or a Guard. Authorization and data isolation must be enforced entirely on the backend.

Core Backend Requirements

1. Multi-Tenancy: All data must be scoped by organizationId and enforced in services or guards.
2. Issue Management: Create, read, update, and delete issues.
3. Role-Based Authorization: Only ADMIN users may update status, assign, or delete issues.
4. Activity Log: Status or assignee changes must generate persistent activity records.

Required API Endpoints

1. POST /issues
2. GET /issues
3. GET /issues/:id
4. PATCH /issues/:id
5. DELETE /issues/:id

Out of Scope (Explicit)

1. Authentication or login systems
2. Frontend or UI development
3. Deployment, Docker, or hosting
4. Automated testing (unit, integration, or e2e)

Mandatory Architecture Questions (README.md)

1. How did you implement multi-tenancy in NestJS?
2. Where does authorization logic live (Guard vs Service) and why?
3. How would you prevent cross-organization data leaks in production?
4. What parts of this system would break or need redesign at scale (100,000 organizations)?
5. What features did you intentionally skip due to time constraints and why?

Submission Rules (Strict)

1. Submit only a public GitHub repository link.
2. Repository must include README.md with setup instructions and architecture answers.
3. Include a .env.example file (no secrets).
4. Application must start successfully using npm run start.
5. Late or incomplete submissions will be rejected.

Evaluation Criteria

1. NestJS Architecture (Modules, Guards, DTOs) – 30%
2. Multi-Tenancy & Authorization Enforcement – 30%
3. API Design & Validation – 20%
4. Code Quality & Project Structure – 10%
5. README & Architectural Reasoning – 10%