Nest Solutions Online Assessment

Overview

You have been asked to design the API for a new real-time employee recognition system. This system will enable employees to send kudos to one another and provide insights across the organization. There is no finalized specification only core requirements and expectations from stakeholders. You will need to analyze, interpret and make architectural decisions based on this evolving situation just as you would in real-world software development.

Core requirements

The company wants to support the following functionality:

- Employees can recognize coworkers with brief messages and emojis
- Recognitions can be public, private, or anonymous
- Employees get notified when they are recognized (real-time delivery is desired)

The leadership team has expressed interest in:

- Analytics features (e.g., recognitions by team, by keyword, by engagement level)
- Role-based access control (especially for managers, HR, and cross-functional leads).
- Extensibility to future features (badges, likes, reactions, comments).

Some additional contexts from stakeholders:

- Real-time updates are a must… unless they're hard. Then batch everything every 10 minutes.
- We'll need to integrate this with the company's internal communication platform such as Slack or Microsoft Teams.
- We're planning to build team-level and keyword-based analytics. Just store the data in a way that'll support it.

Implementation Guide

Design and implement a graphql based API that supports the core functionality while anticipating future needs. You are required to:

- Making sense of evolving and conflicting requirements
- Consider and document user workflows or usage patterns that your API should support
- Thinking through user roles, visibility rules, and integration points
- Modeling a clean, extensible GraphQL schema
- Designing for real-time updates and/or fallback strategies
- Planning data structures that support analytics and future feature growth

- Explaining trade-offs and assumptions clearly, as you would in a real product

The focus is on your thought process, architectural reasoning, and practical decision-making in a realistic engineering scenario.

Deliverables (Submit as a [full_name].zip file)

Your zip file must include:

1. API Schema
    a. Types, queries, mutations, and at least one subscription
    b. Descriptions for all fields and operations
    c. Access control or role-based visibility model
2. API Implementation
    a. A working GraphQL server (TypeScript is preferred but not required)
    b. Sample mock data or simple in-memory storage
    c. Examples of API usage: queries, mutations, and subscription test
    d. Security and performance considerations
    e. Code should be modular, follow modern best practices, and be organized for maintainability and clarity
3. API Documentation
    a. Clear developer documentation (README or equivalent)
    b. Explanation of your decisions, assumptions, and reasoning
    c. Highlight how your API handles ambiguous or conflicting requirements

Integrity Notice

- This assessment is specifically designed to evaluate your ability to think critically, design and document well-structured API, and navigate ambiguous requirements. The goal is not simply to produce working code but to understand how you approach realistic engineering scenarios.
- You are strictly prohibited from using any AI tools or large language models such as ChatGPT, GitHub Copilot, or similar services to generate your solution. Any use of such tools will be considered a violation of the assessment's integrity guidelines.
- Submissions that lack original reasoning, demonstrate insufficient understanding of the problem or appear to be primarily AI-generated are subject to rejection.
- We are not seeking a perfect solution. We are evaluating your own thought process, decision-making, and architectural clarity. By submitting this assessment, you confirm that the work is your own and has not been outsourced or generated by automated tools.