Specno

Specno Tech Assessment

_

Introduction

This assessment is designed to evaluate your expertise in building scalable, maintainable, and performant software solutions. It will assess your ability to design a modular architecture, implement best practices, and follow coding standards that align with enterprise-level software development.

Objective

Create a microservices-based application that manages project tasks across multiple teams. The system must include the following functionalities:

1. Project Management
    - Users can create, update, and delete projects.
    - Each project has a title, description, and deadline.
2. Task Management
    - Users can create, assign, update, and delete tasks within projects.
    - Each task has a title, description, status (To Do, In Progress, Done), and assignee.
3. User Management
    - Implement authentication (JWT-based or OAuth2).
    - Users should be able to register, log in, and manage their profiles.
    - User roles: Admin, Project Manager, Developer.

Constraints

- The application must be built using a modern backend framework (Node.js with NestJS, Spring Boot, Django, etc.).
- The frontend must be implemented using a frontend framework (React, Vue, or Angular).
- The system must expose a RESTful API (or GraphQL if preferred) for communication.
- Use a database (PostgreSQL, MongoDB, or MySQL) for persistence.
- Implement unit and integration tests to ensure code quality.
- The code must follow version control best practices (branching, commit messages, PR reviews, etc.).
- Document how to set up and run the application in a README.md file.
- The assignee is free to use an existing dashboard-based template for the UI.

Bonus Points

(Optional, but recommended)

- Track Project Completion Status
    - Track whether all tasks in a project were completed before the deadline.
    - Provide a UI component that visually indicates project completion status.
- Implement a Notification Service (Event-Driven Architecture)
    - When a task is assigned or its status changes, a notification should be sent to the assignee.

    - Use an event bus (e.g., Kafka, RabbitMQ, or Redis Pub/Sub) or an internal queuing system.
- Implement a CI/CD pipeline to automate testing and deployment.
- Containerize services using Docker.
- Orchestrate services using Kubernetes.
- Provide a Swagger/OpenAPI documentation for the backend API.
- Deploy the application to a cloud provider (AWS, GCP, or Azure).

Resources

Include any relevant documentation, API specifications, or UI designs in the provided resource links.

Criteria

You will be assessed on the following:

- Code Quality: Clean, readable, and maintainable code.
- Architecture: Modular and scalable design.
- Performance: Efficient use of resources and optimized queries.
- Security: Authentication, authorization, and data protection.
- Testing: Presence of meaningful unit and integration tests.
- Deployment & DevOps: Effective CI/CD, containerization, and cloud readiness.

Good luck! We look forward to seeing your innovative solutions!