Employee Management System

Introduction

Welcome to our Employee Management System challenge! In this assessment, you'll be tasked with creating a REST-based API platform using SpringBoot to manage employees and departments. Please read below problem statement and build a Java based API service. You should submit the code base as a git repository within the give deadline. After code review you will be asked to join for a technical & code review session where you will be asked to explain the code base you submitted.

Problem Statement

You are required to create a Java SpringBoot application for managing employees and departments. Here are the specific requirements:

- [ ] **Create Employee:** Implement an API endpoint to create a new employee with all required details.
- [ ] **Update Employee:** Develop an API endpoint to update an existing employee's information.
- [ ] **Add Department:** Create an API endpoint to add a new department to the system.
- [ ] **Delete Department:** Implement an API endpoint to delete a department. This operation should fail if there are employees assigned to the department.
- [ ] **Update Department:** Develop an API endpoint to update department details.
- [ ] **Update Employee's Department:** Implement an API endpoint to move an employee from one department to another
- [ ] **Fetch All Employees and Departments:** Implement API endpoints to fetch all employees and all departments.

- ☐ **Expand Employees under Departments:** Create an API endpoint for departments that, when provided with a parameter **expand=employee**, returns the department along with a list of all employees under that department.
- ☐ **List Employee Name and ID:** Develop an API endpoint to list employee names and IDs. This should be triggered by passing **lookup=true** as a parameter.
- ☐ **Employee Information:** Ensure that the API provides employee details including name, date of birth, salary, department, address, role/title, joining date, yearly bonus percentage and reporting manager (another employee in the same table, all employees except the top level employee should have a reporting manager).
- ☐ **Department Information:** Ensure that departments have a name, creation date, and department head (an employee who exists in the employee table).
- ☐ **Minimum Requirements:** Create at least 3 departments and 25 employees.
- ☐ **Pagination**: All GET APIs should be by default paginated with 20 items in a single page. The API response should give which page and total number of pages.

Deliverables

Your submission should include:

- ☐ **JSON Schema:** Define the request and response JSON schemas for the APIs.
- ☐ **Database Structure:** Design the tables required to store the employee and department information.
- ☐ **API Logic:** Write code to implement POST, PUT/PATCH, GET, and DELETE APIs with proper error handling.
- ☐ **Complex Logic Handling:** Implement logic to handle complex scenarios like generating analytics data and reporting chains.

- ☐ **Code Logic:** Loops and conditional statements, usage of collections and appropriate data types to store data.
- ☐ **Code Flow:** Controller → Service → Database logic
- ☐ **JPA Implementation:** Use JPA to interact with the database using entity classes.
- ☐ **Git Management:** Create a GitHub repository containing your codebase.
- ☐ **DB Script:** Submit the database script to create tables

We look forward to seeing your solution! If you have any questions, feel free to ask.

You are expected come for the technical interview with the laptop loaded with the code base in any code editor you wish (InteliJ or VS Code etc). You will be asked to make modifications in the code during the interview. You may be asked to write more APIs or logical/alogorithmic changes.

Good luck!