Spring Boot Technical Task: Employee Management System

Overview

Build a simple Employee Management System using Spring Boot. The system should support:

- Basic CRUD operations for Employees and Departments
- Role-based authentication and authorization
- Database design using JPA (you'll provide an ERD)
- A cron job that runs daily to generate a summary report

1. CRUD Operations (Using REST APIs)

Employee Entity:

- id: Long (PK)
- name: String
- email: String (unique)
- salary: Double
- hireDate: Date
- department: Many-to-One with Department

Endpoints:

- POST /employees
- GET /employees
- GET /employees/{id}
- PUT /employees/{id}
- DELETE /employees/{id}

Department Entity:

- id: Long (PK)
- name: String (unique)

Endpoints:

- POST /departments
- GET /departments

Spring Boot Technical Task: Employee Management System

- GET /departments/{id}
- PUT /departments/{id}
- DELETE /departments/{id}

2. Spring Security

- Add in-memory authentication with 2 roles:
  - ADMIN: Can access all endpoints
  - USER: Can only access GET /employees and GET /departments
- Secure all other endpoints based on the user's role
- Use HTTP Basic Auth for simplicity

3. Database Design (ERD)

- Draw the ERD diagram showing:
  - One department can have many employees
  - Each employee belongs to one department
- Submit as image or use dbdiagram.io/draw.io

4. Cron Job (Scheduled Task)

- Runs every day at 9:00 AM
- Logs the total number of employees in each department
- Optionally stores the log in a daily_summary table (Bonus)

Technical Requirements

- Java 17+
- Spring Boot 3.x
- Spring Data JPA
- Spring Security
- H2 or PostgreSQL
- @Scheduled annotation for cron job

Spring Boot Technical Task: Employee Management System

Deliverables

1. Complete Spring Boot project as a GitHub repository with meaningful commit messages
2. REST API documentation (Postman collection must be submitted)
3. ERD diagram (JPG, PNG, or PDF)
4. Proper exception handling (this is a must)
5. Unit tests for business logic (this is a must)
6. Brief explanation of design decisions (README)

Evaluation Criteria

Area	Criteria
CRUD Logic	Correctness, RESTful API standards, error handling
Security	Proper use of Spring Security, role-based access
Database Design	Normalized structure, correct relationships and ERD
Cron Job	Correct cron syntax, clean implementation
Code Quality	Clean code, layered architecture
Exception Handling	Well-structured and consistent exception responses (must)
Unit Testing	Tests covering service layer or business logic (must)
Version Control	GitHub repo with clean commit history and messages
Documentation	Postman collection and clear README