1

Backend Exercise: **Task Management System**

What is the Task Management System?

The Task Management System backend is a standalone application designed to handle the logic and data management for managing tasks. It provides a set of APIs to perform CRUD operations on tasks and includes authentication, authorization, validation, and other features.

Technology constraints

- We currently have a microservices architecture with services mostly written in Spring Boot.
- We would like you to develop a single service using Spring Boot with either Java or Kotlin that exposes several REST endpoints (see next page)
- You can use whatever database and data access method; they create for the required database structure.
- You should include a README file that has instructions for us to get the solution running on our machines.

What are we looking to test?

- The overall software architecture of the application
- The structure and **quality of the code itself**
- The use of well-known patterns for REST and Spring development
- Your ability to model the problem domain (data models and APIs)
- Bonus points if you include unit tests in your solution.

What are we *not* looking to test?

- We don't expect you to implement any front-end interfaces

External Resources

- Building a RESTful service in Spring Boot
- Testing in Spring Boot

2

Backend Exercise: Task Management System

Your APIs should provide the following functionality to downstream consumers of the API (in a **RESTful** way):

Task Management APIs

- Implement CRUD operations for managing tasks (create, read, update, delete).
- Each task should have a title, description, status (e.g., todo, in progress, done), priority, and due date.

User Authentication and Authorization

- Implement JWT-based authentication with Spring Security.
- Define user roles (e.g., admin, regular user) and enforce authorization rules for accessing certain endpoints.

Validation and Error Handling

- Input data is validated to ensure integrity and consistency.
- Proper error-handling mechanisms are implemented to provide meaningful error messages in case of invalid requests or server errors.

Search and filtering

- Endpoints are provided to search for tasks based on various criteria such as title, description, status, and due date.
- Filtering options allow users to narrow down task lists based on specific attributes or conditions.

Email Notifications

- If desired, the backend can include functionality to send email notifications for upcoming task deadlines or important updates.
- Email integration can be implemented using third-party services or custom SMTP configurations.

TIP #1: define a clear data model for the Tasks, Users, History, and Notifications. You should include the SQL create table statements etc. in your code base.

TIP #2: invest time in building some 'seed data' that covers all the use cases you want to demonstrate.

BONUS POINTS:
- APIs support pagination to manage large datasets efficiently, returning a limited number of tasks per request.