Java (Backend) - HanseCom Coding@Home Challenge

Java Code Challenge Description

Title: Development an simple Online Service Monitoring Tool with Spring Boot

Overview:

The Hansecom Codeing Challenge is organized as follows:

1. Do the task below, spend as much time as you think, but we absolutely do note expect a fully developed ready-to-go solution. This is more for you to demonstrate and explain your ideas in the step 3.
2. Checkin you solution to gitlab.com (not github.com!) invite pavel.reich@hansecom.com, thoralf.rickert-wendt@hansecom.com and kay.paetzold@hansecom.com .
3. Present you solution to the team normally in an online meeting. The team will ask you questions, why you did thing like you did and what other options you see.

Task:

In this challenge, you will to first drop of an monitoring tool using Java and Spring Boot that allows users to configure between 1 to 5 monitoring jobs. Each job will periodically test a specified URL and record the outcome of each test (e.g., success, failure, response time). The system must store these results and expose them via a REST API, which will be consumed by a separate React frontend application (the development of the frontend is not included in this challenge).

Requirements:

1. **Spring Boot Application:**
   - Use Java and Spring Boot to create the backend application.
   - Implement REST APIs to manage monitoring jobs and retrieve their results.
2. **Monitoring Jobs Configuration:**
   - Allow configuration of 1-5 monitoring jobs through an API *or choose a simplier approach.*
   - Each job should include:
     - A unique identifier.
     - ….
3. **Execution of Monitoring Jobs:**
   - Implement the logic to test the specified URLs at the configured intervals.
   - Record the results of each test, including success or failure, response time, and any error messages.
4. **Data Storage:**
   - Store the monitoring results in a database. You can choose any relational or NoSQL database.
5. **API for Retrieving Results:**
   - Develop an API to query and retrieve the monitoring results for each job.
   - Include filtering capabilities to retrieve results based on time range, status, etc.
6. **Error Handling and Validation:**
   - Implement proper error handling and validate the configurations for monitoring jobs.
7. **Testing**
   a. the microservice should be automatically tested for successful and not so successful scenarios. Please cover the most important scenarios.
8. **Documentation:**
   - Provide a README file detailing how to set up and run your application, including configuring monitoring jobs and accessing the results.

Evaluation Criteria:

- Functionality: The tool correctly monitors URLs and records results as specified.
- Code Quality: The code is clean, well-organized, and follows best practices.
- Error Handling: The application gracefully handles and reports errors.
- Documentation: The provided documentation is clear and helpful.
- Testing: microservice is automatically tested and the most important scenarios are covered.

How to work and present the result of this Challenge ?

- Use an existing or create a gitlab account
- Invite kay.paetzold@hansecom.com, pavel.reich@hansecom.com, thoralf.rickert-wendt@hansecom.com
- Push your Code
- Present and explain you solution to the team over Teams

Bonus: Think of some possible additions (only brainstorming),

Provide a short list or present verbally your ideas in the demo