Java Assessment – SE_CEG_JAV-0004

Project: Task Automation and Scheduling System

Objective:

Develop a task automation and scheduling system that allows users to automate repetitive tasks and schedule them for execution. The system should focus on modularity, concurrency, and integration with external APIs.

Features:

1. **User Authentication**: Implement secure user login and registration.
2. **Task Management**: Create, edit, and delete tasks with specific actions.
3. **Scheduling**: Schedule tasks to run at specified times or intervals.
4. **Integration**: Connect with external APIs to perform actions (e.g., send emails, fetch data).
5. **Notifications**: Alert users upon task completion or failure.

Technical Requirements:

- **Java 17**: Utilize new features and enhancements in Java 17.
- **Spring Boot**: Use Spring Boot for building the application.
- **Quartz Scheduler**: Implement task scheduling.
- **RESTful APIs**: Create APIs for task management and user interactions.
- **Concurrency**: Use Java's concurrency utilities for handling multiple tasks.
- **Database**: Use a relational database like PostgreSQL for data persistence.
- **Docker**: Containerize the application for easy deployment.

Steps to Implement:

1. **Set Up the Project**:
    - Create a Spring Boot project using Maven or Gradle.
2. **Develop Core Features**:
    - **User Authentication**: Implement using Spring Security.
    - **Task Management**: Create APIs to manage tasks and their schedules.
    - **Scheduling**: Use Quartz Scheduler to automate task execution.
3. **Integration with External APIs**:
    - Implement connectors to interact with external services (e.g., email, data APIs).
4. **Implement Concurrency**:
    - Use Java's concurrency features to handle multiple task executions efficiently.
5. **Database Integration**:
    - Use JPA/Hibernate for ORM with PostgreSQL.

    - Ensure proper indexing and query optimization.
6. **Containerization**:
    - Create a Dockerfile to containerize the application.
7. **Testing**:
    - Write unit and integration tests for key components.
8. **Documentation**:
    - Provide comprehensive documentation, including setup instructions and API documentation.

Evaluation Criteria

1. **Functionality – 50%**
    - Verify the implementation of all specified features.
    - Test the integration with scheduling and external APIs.
2. **Code Quality – 10%**
    - Review code readability, modularity, and adherence to best practices.
    - Check for proper use of Java 17 features.
3. **Performance and Concurrency – 20%**
    - Assess the system's ability to handle concurrent task executions.
    - Evaluate the use of concurrency utilities.
4. **Security – 10%**
    - Check the implementation of authentication and data protection.
5. **Documentation – 10%**
    - Assess the completeness and clarity of the documentation provided.