Integration Project

Order Management Service

In this project, you will develop a microservice in Java using the Spring Boot framework to manage orders for an eCommerce platform. The project should be structured so it can be completed in 1-3 days and aims to evaluate your skills in creating well-structured microservices and writing clean, maintainable code.

Project Goals:

- Develop a microservice for order management within an eCommerce environment.
- Demonstrate effective use of the Spring Boot framework.
- Showcase your ability to work with relational data, implement scheduled tasks, and, optionally, handle inter-service communication and non-relational data.

Evaluation Criteria:

- Microservice architecture and structure.
- Code quality and organization.
- Appropriate use of Spring Boot and associated frameworks.
- Handling of relational data.
- (Optional) Implementation of inter-service communication and handling of non-relational data.

Required Task: 1st Microservice: Order Management Service

1. **Purpose:** This service manages orders and their status updates.
2. **Functionality:**
    - Define an Order model that has a one-to-many relationship with an OrderLine model. The Order model should include fields such as:
        - orderId, customerName, status (default: "unprocessed"), orderDate
    - The OrderLine model should include fields such as:
        - productId, quantity, price

- Implement CRUD operations for Order and OrderLine models, stored in a PostgreSQL database.
- Expose a REST controller (/orders) with standard CRUD endpoints to manage orders.
- Implement a scheduler that runs every minute to automatically update the status field of "unprocessed" orders to "processed".

3. **Database:** Configure and connect to a PostgreSQL database to manage order data.
4. **Endpoints:**
   - POST /orders: Create a new order with associated order lines (returns id).
   - GET /orders/{id}: Retrieve an order by ID.
   - PUT /orders/{id}: Update an order.
   - DELETE /orders/{id}: Delete an order.

Optional Task: 2nd Microservice - Logging Service

1. **Purpose:** This optional service will log processed order data to MongoDB.
2. **Functionality:**
   - Expose a REST endpoint (/logs) that accepts a request to save log details such as:
     - orderId, amount (sum of price * quantity for each OrderLine), itemsCount, date
   - The scheduler in Microservice 1 should call this endpoint after successfully updating an order's status to "processed".
3. **Database:** Connect to MongoDB to store log records.
4. **Endpoint:**
   - POST /logs: Accept log data and save it to MongoDB.

Technical Requirements:

- Use Java and Spring Boot for both services.
- Configure basic error handling, logging, and scheduling in Order Management **Microservice**.
- Provide documentation for setup instructions C API descriptions.

Docker compose instructions

We provide a Docker Compose configuration that sets up the necessary infrastructure for implementing and testing the project. The databases are initialized as empty, with no predefined tables or collections, allowing you to create schemas based on your project's requirement.