Grocery Ordering Platform - Microservices Assessment

Objective

Design and implement a microservices-based grocery ordering platform where authenticated users (end consumers) can browse grocery items, add them to a cart, and place orders. The system will use Firebase Authentication and PostgreSQL, with clear service boundaries and inter-service communication.

Functional Requirements

1. User Service

- Authenticate users using Firebase Authentication.

- Secure endpoints using Firebase JWT validation.

- Provide endpoints to:

  • View authenticated user profile

  • Maintain role (e.g., consumer; optional: retailer)

2. Product Catalog Service

- Manage a catalog of grocery items with attributes like:

  • Name, Description, Unit (e.g., 1kg), Price, Shelf Life, Stock Quantity

- Provide endpoints to:

  • List all items

  • Get item details

  • (Optional) Search/filter items

3. Cart Service

- Each user maintains a shopping cart with items from the catalog.

- Cart includes product ID, quantity, total price.

- Provide endpoints to:

  • Add/update/remove items in the cart

- View current cart
- Clear cart

4. Order Service

- Manage orders based on cart contents.

- Order details include: ID, user ID, item list, total cost, status.

- Provide endpoints to:

  - Place new order
  - View past orders
  - Update order status (admin-only)

Non-Functional Requirements

- Firebase authentication and Spring Security for protecting endpoints.

- REST (HTTP/Feign) for service communication.

- Optional: Kafka/RabbitMQ for future asynchronous handling.

- PostgreSQL for each service's persistence.

- Use Spring Data JPA for ORM.

- Use Testcontainers for integration tests.

- Dockerize services and manage using Docker Compose.

- Use Spring Cloud Gateway for API routing.

- Use Spring Cloud Config Server for centralized config.

Expected Deliverables

- Clean, modular codebase with separate Spring Boot services.

- Docker + Docker Compose setup.

- Swagger/Postman collection for testing APIs.

- README.md with instructions.

- Integration tests using Testcontainers.

Assessment Criteria

Criteria	Weightage
Microservice design & separation	⭐⭐⭐⭐
Firebase integration & security	⭐⭐⭐⭐
API structure & documentation	⭐⭐⭐
PostgreSQL modeling & JPA use	⭐⭐⭐
Dockerization and config management	⭐⭐⭐⭐
Testing (integration/unit)	⭐⭐⭐
Code quality & readability	⭐⭐⭐⭐