Exercises on Microservices with Spring Boot 3.0

1. User and Order Management System

Problem: Create two microservices:

- User Service to manage users.
- Order Service to manage orders placed by users.

Requirements:

- Use REST APIs for CRUD operations.
- Communication between microservices should be done using WebClient (Spring WebFlux) or OpenFeign.
- Store data in MySQL or PostgreSQL.

Solution Approach:

1. Create two Spring Boot microservices using Spring Initializr with dependencies: Spring Web, Spring Data JPA, MySQL/PostgreSQL Driver, Lombok, and Spring Cloud OpenFeign/WebFlux.
2. Implement UserService with endpoints for creating and retrieving users.
3. Implement OrderService with endpoints for placing and retrieving orders.
4. Enable service-to-service communication from OrderService to UserService using WebClient or FeignClient to fetch user details.
5. Configure database connection properties in application.yml or application.properties.
6. Use @Entity classes and JPA Repositories for User and Order persistence.
7. Test each API independently and in combination.

2. Inventory Management System with Service Discovery

Problem: Create:

- Product Service: Manage products and stock.
- Inventory Service: Track stock levels for each product.

Requirements:

- Use Spring Cloud Netflix Eureka for service discovery.
- Implement centralized configuration using Spring Cloud Config Server.

Exercises on Microservices with Spring Boot 3.0

Solution Approach:

1. Set up Eureka Server as a separate Spring Boot application using the Eureka Server dependency.
2. Register Product Service and Inventory Service as Eureka clients.
3. Implement REST APIs in ProductService for adding and retrieving products.
4. Implement InventoryService with APIs for checking and updating stock levels.
5. Use FeignClient or WebClient in InventoryService to call ProductService.
6. Set up Spring Cloud Config Server and use Git or local file system for storing externalized configurations.
7. Ensure each microservice fetches configuration from the config server and registers itself with Eureka.
8. Test service discovery and inter-service communication using Eureka Dashboard and Postman.