Answers to Microservices Exercises

1. Centralized Authentication with OAuth2/OIDC

Implement OAuth2/OIDC using Spring Boot by including Spring Security and OAuth2 client dependencies. Configure your client credentials in `application.yml`, and setup security with `oauth2Login()` in your `SecurityConfig.java`. Use a REST controller to return authenticated user info.

2. Authorization & Resource Server Configuration

Use Spring Security and OAuth2 Resource Server dependencies. Configure `application.yml` with issuer URI. Set up `ResourceServerConfig.java` to secure endpoints using JWT authentication via `.oauth2ResourceServer().jwt()`.

3. Secure Communication with JWT

Add Spring Security and JJWT dependencies. Use `JwtTokenProvider` to generate tokens and `JwtTokenFilter` to validate them. Configure the security filter chain to check JWTs before processing any request.

4. Routing and Filtering with API Gateway

Add `spring-cloud-starter-gateway` dependency. Define routes in `application.properties`. Implement a `GlobalFilter` class to log requests passing through the gateway.

5. Load Balancing in API Gateway

Include `spring-cloud-starter-loadbalancer`. Define a service URI like `lb://example-service` in routes. Provide a `LoadBalancerConfiguration` bean using `RandomLoadBalancer`.

6. Resilience Patterns (Circuit Breaker)

Use `resilience4j-spring-boot2`. Configure circuit breaker settings in `application.properties`. Provide a `Customizer<ReactiveResilience4JCircuitBreakerFactory>` bean to define default fallback

behavior.

7. User & Order Management Microservices

Create two services: User and Order. Use REST APIs with WebClient or OpenFeign for communication. Persist data in a relational database like MySQL or PostgreSQL.

8. Inventory Management with Eureka & Config Server

Use Eureka for service discovery. Build Product and Inventory services. Centralize configuration with Spring Cloud Config Server.

9. API Gateway with Enhancements

Create a gateway to route to Customer and Billing services. Add rate limiting, caching, and URL rewriting in the gateway configuration.

10. Circuit Breaker in Payment Service

Wrap slow API calls with Resilience4j Circuit Breaker. Provide fallback logic and monitor events for analysis.