**Exercises on Microservices with Spring Boot 3.0**

**1. Build a User and Order Management System**

**Problem: Create two microservices:**

1. User Service to manage users.
2. Order Service to manage orders placed by users.

**Requirements:**

1. Use REST APIs.
2. Communicate between services using WebClient (Spring WebFlux) or OpenFeign.
3. Store data in MySQL or PostgreSQL.

**Project: user-service**

**User.java**

package com.example.user\_service.model;  
  
import jakarta.persistence.\*;  
import lombok.\*;  
  
@Entity  
public class User {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 public User(Long id, String name, String email) {  
 this.id = id;  
 this.name = name;  
 this.email = email;  
 }  
  
 public User() {  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 private String name;  
 private String email;  
}

**UserController.java**

package com.example.user\_service.controller;  
  
import com.example.user\_service.model.User;  
import com.example.user\_service.service.UserService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
import java.util.Optional;  
  
@RestController  
@RequestMapping("/users")  
public class UserController {  
  
 @Autowired  
 private UserService userService;  
  
 @PostMapping  
 public User createUser(@RequestBody User user) {  
 return userService.save(user);  
 }  
  
 @GetMapping("")  
 public List<User> getAllUsers(){  
 return userService.getAllUsers();  
 }  
  
 @GetMapping("/{id}")  
 public Optional<User> getUser(@PathVariable Long id) {  
 return userService.getById(id);  
 }  
}

**UserService.java**

package com.example.user\_service.service;  
  
import com.example.user\_service.model.User;  
import com.example.user\_service.repository.UserRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class UserService {  
  
 @Autowired  
 private UserRepository userRepository;  
  
 public User save(User user) {  
 return userRepository.save(user);  
 }  
  
 public Optional<User> getById(Long id) {  
 return userRepository.findById(id);  
 }  
  
 public List<User> getAllUsers() {  
 return userRepository.findAll();  
 }  
}

**UserRepository.java**

package com.example.user\_service.repository;  
  
import com.example.user\_service.model.User;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface UserRepository extends JpaRepository<User, Long> {  
}

**UserServiceApplication.java**

package com.example.user\_service;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class UserServiceApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(UserServiceApplication.class, args);  
 }  
  
}

**Application.properties**

spring.application.name=user-service  
  
spring.datasource.url=jdbc:mysql://localhost:3306/userdb  
spring.datasource.username=root  
spring.datasource.password=password  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect  
  
server.port=8081

**Project: order-service**

**Application.properties**

spring.application.name=order-service  
  
spring.datasource.url=jdbc:mysql://localhost:3306/orderdb  
spring.datasource.username=root  
spring.datasource.password=password  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect  
  
server.port=8082

**Order.java**

package com.example.order\_service.model;  
  
import jakarta.persistence.\*;  
import lombok.\*;  
  
@Entity  
@Table(name = "orders")  
public class Order {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private String product;  
  
 public Order() {  
 }  
  
 public Order(Long id, String product, Double amount, Long userId) {  
 this.id = id;  
 this.product = product;  
 this.amount = amount;  
 this.userId = userId;  
 }  
  
 public Long getUserId() {  
 return userId;  
 }  
  
 public void setUserId(Long userId) {  
 this.userId = userId;  
 }  
  
 public Double getAmount() {  
 return amount;  
 }  
  
 public void setAmount(Double amount) {  
 this.amount = amount;  
 }  
  
 public String getProduct() {  
 return product;  
 }  
  
 public void setProduct(String product) {  
 this.product = product;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 private Double amount;  
  
 private Long userId; // External reference to user-service  
}

**OrderRepository.java**

package com.example.order\_service.repository;  
  
import com.example.order\_service.model.Order;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface OrderRepository extends JpaRepository<Order, Long> {  
}

**UserClient.java**

package com.example.order\_service.client;  
  
import com.example.order\_service.dto.UserDTO;  
import org.springframework.cloud.openfeign.FeignClient;  
import org.springframework.web.bind.annotation.\*;  
  
@FeignClient(name = "user-service", url = "http://localhost:8081")  
public interface UserClient {  
 @GetMapping("/users/{id}")  
 UserDTO getUserById(@PathVariable("id") Long id);  
}

**OrderController.java**

package com.example.order\_service.controller;  
  
import com.example.order\_service.client.UserClient;  
import com.example.order\_service.dto.UserDTO;  
import com.example.order\_service.model.Order;  
import com.example.order\_service.service.OrderService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.\*;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
import java.util.Optional;  
  
@RestController  
@RequestMapping("/orders")  
public class OrderController {  
  
 @Autowired  
 private OrderService orderService;  
  
 @Autowired  
 private UserClient userClient;  
  
 @PostMapping  
 public ResponseEntity<?> createOrder(@RequestBody Order order) {  
 try {  
 UserDTO user = userClient.getUserById(order.getUserId());  
 return ResponseEntity.*ok*(orderService.save(order));  
 } catch (Exception e) {  
 return ResponseEntity.*status*(HttpStatus.*BAD\_REQUEST*).body("User does not exist");  
 }  
 }  
  
 @GetMapping("/{id}")  
 public Optional<Order> getOrder(@PathVariable Long id) {  
 return orderService.getById(id);  
 }  
  
 @GetMapping("")  
 public List<Order> getAllOrders(){  
 return orderService.getAllOrders();  
 }  
}

**UserDTO.java**

package com.example.order\_service.dto;  
  
import lombok.\*;  
  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
public class UserDTO {  
 private Long id;  
 private String name;  
 private String email;  
}

**OrderService.java**

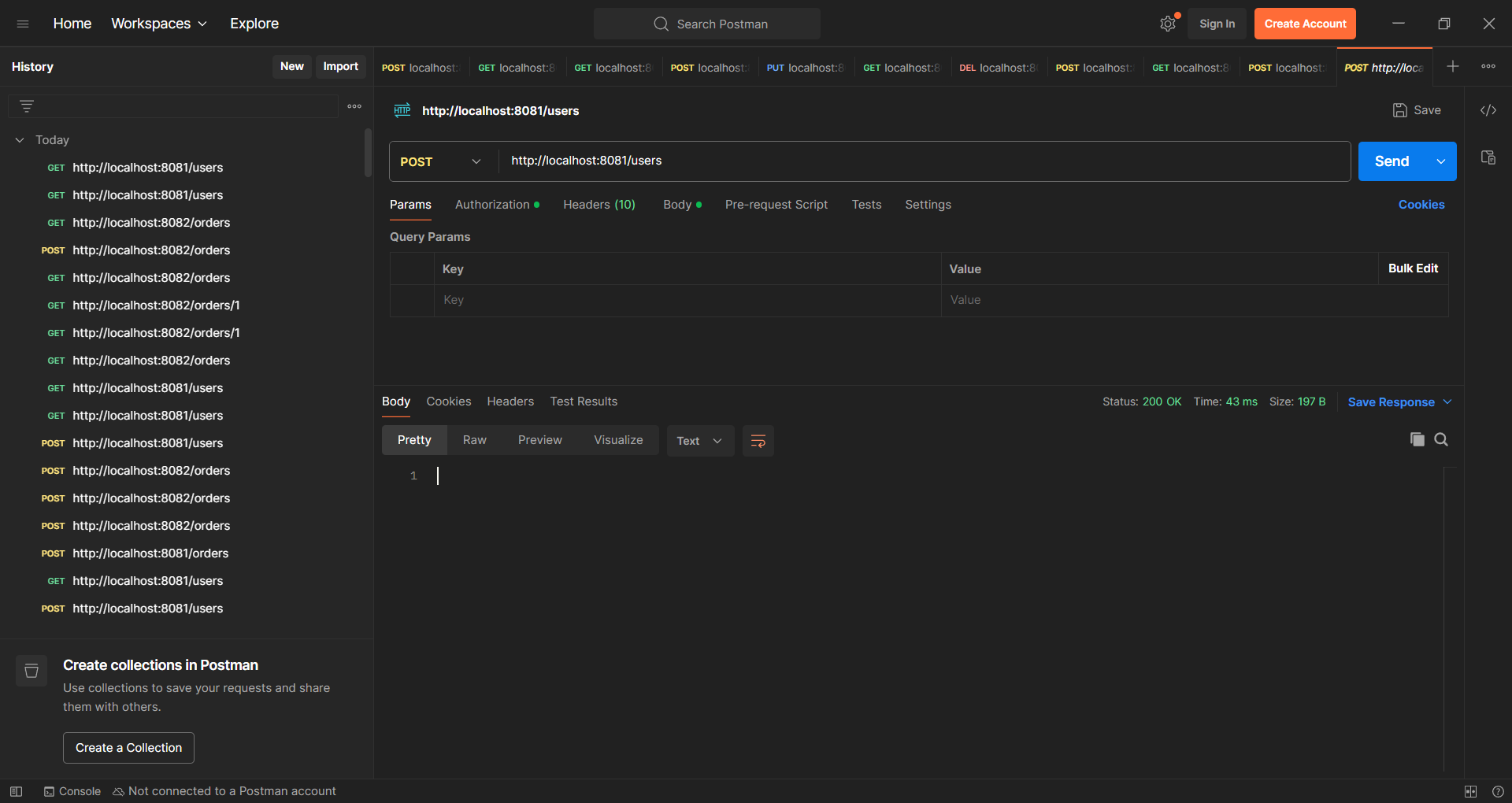
package com.example.order\_service.service;  
  
import com.example.order\_service.model.Order;  
import com.example.order\_service.repository.OrderRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class OrderService {  
  
 @Autowired  
 private OrderRepository orderRepository;  
  
 public Order save(Order order) {  
 return orderRepository.save(order);  
 }  
  
 public Optional<Order> getById(Long id) {  
 return orderRepository.findById(id);  
 }  
  
 public List<Order> getAllOrders() {  
 return orderRepository.findAll();  
 }  
}

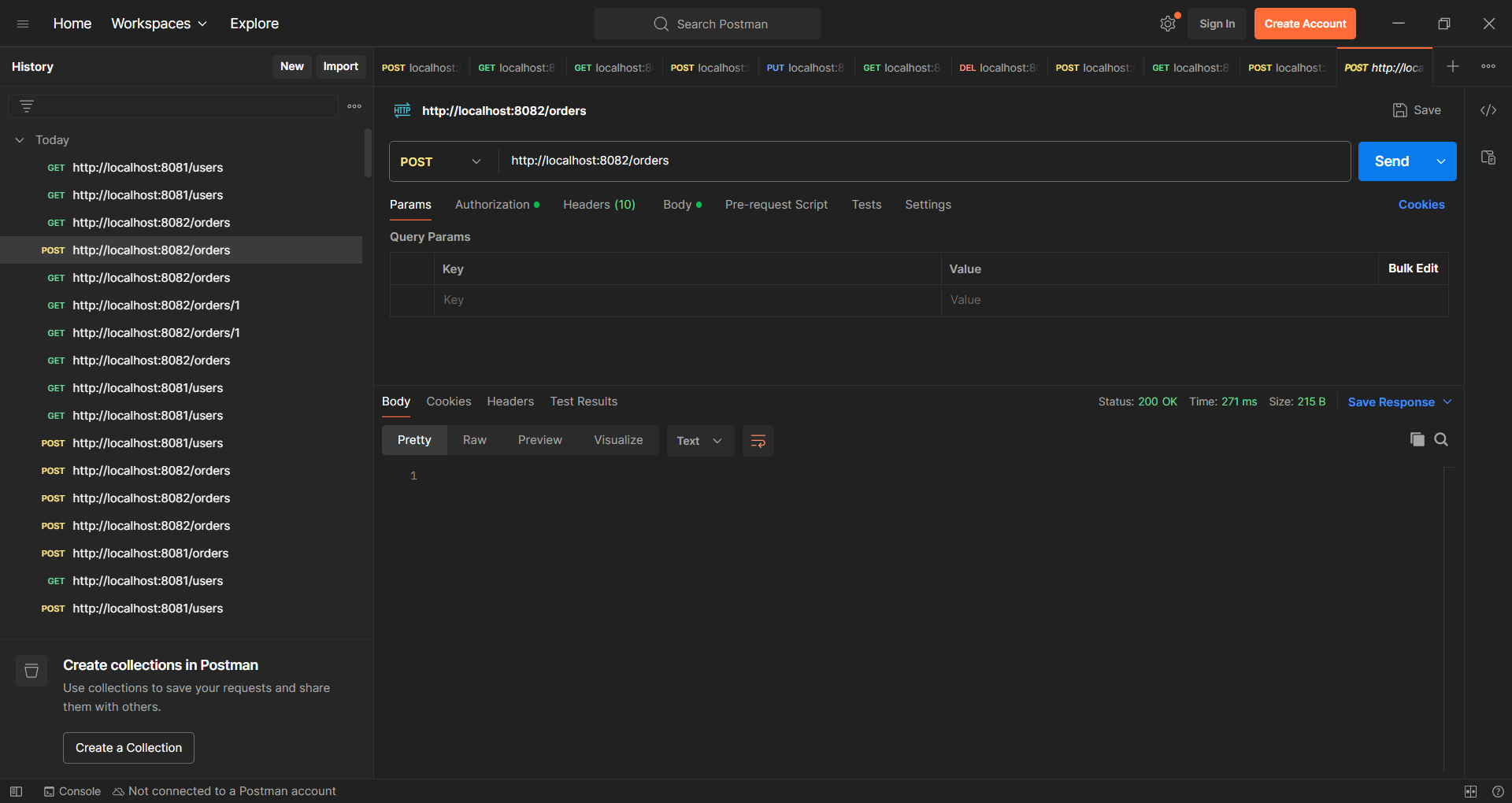
**OrderServiceApplication.java**

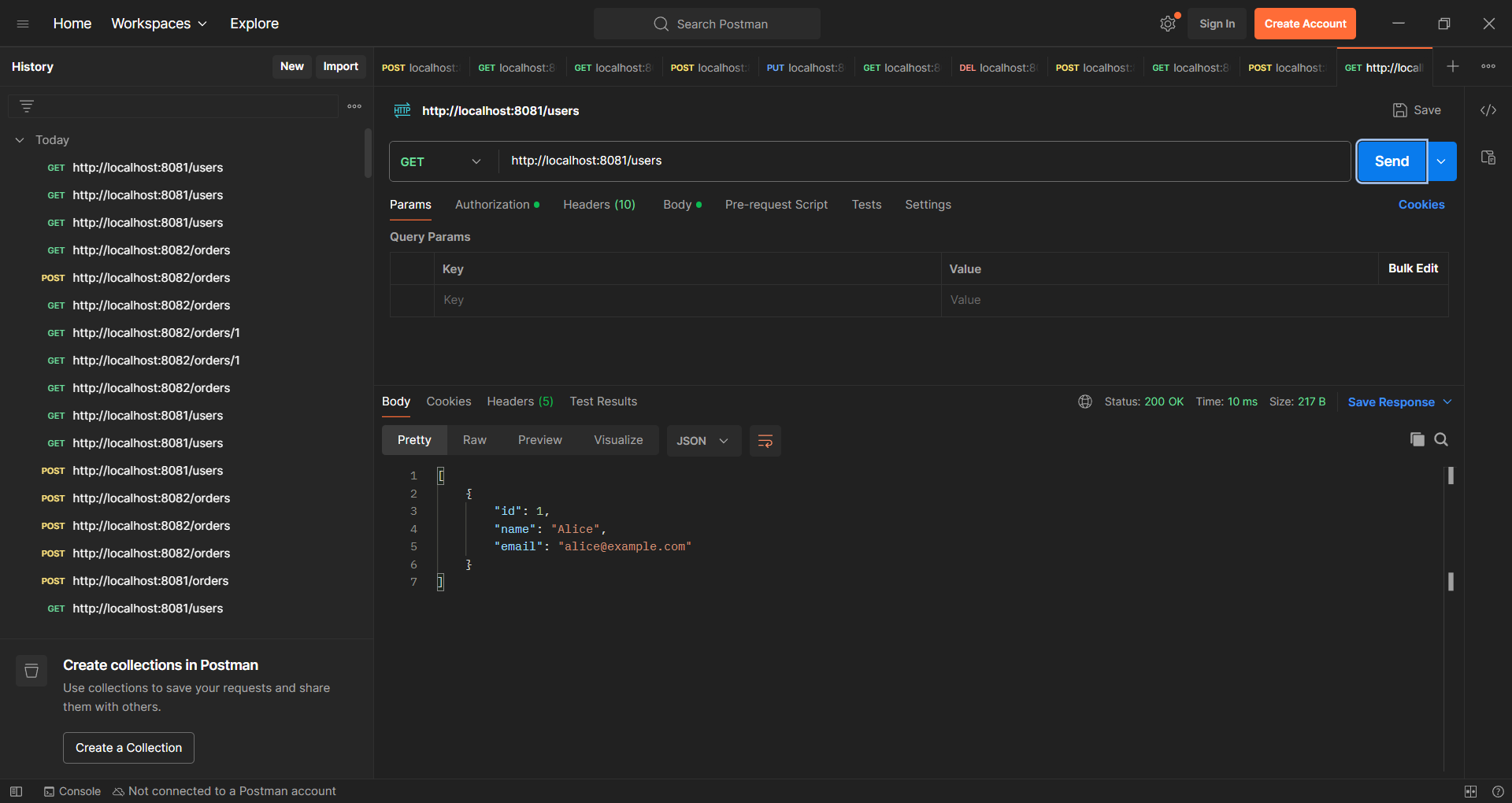
package com.example.order\_service;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.openfeign.EnableFeignClients;  
  
@SpringBootApplication  
@EnableFeignClients  
public class OrderServiceApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(OrderServiceApplication.class, args);  
 }  
}

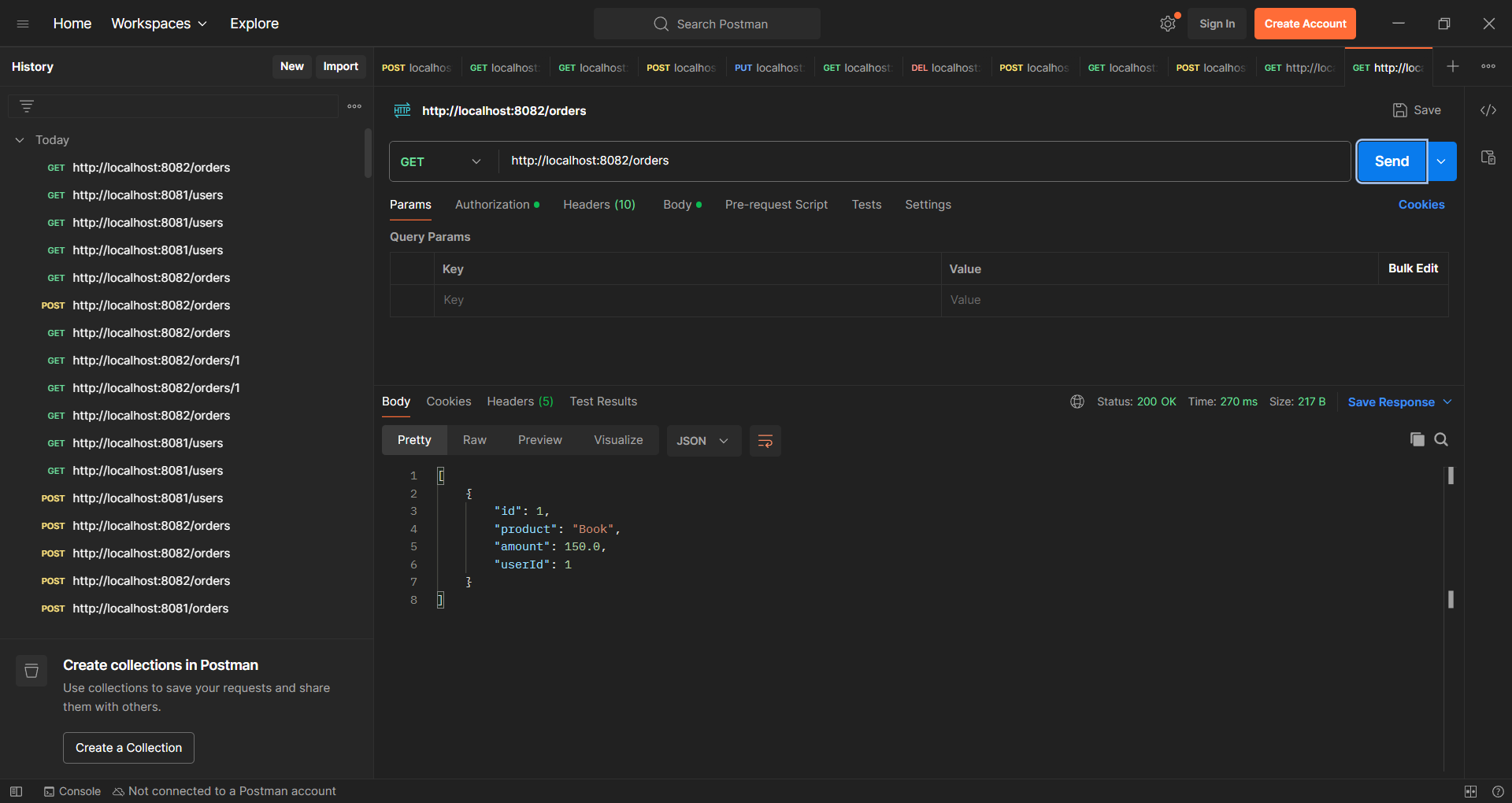
**Output:**

**Post Users:**



**PostOrders:**

**Get Users:**

**Get Orders:**

**2. Inventory Management System with Service Discovery**

**Problem: Create:**

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

**Requirements:**

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

**Project: config-server**

**ConfigServerApplication.java**

package com.library.configserver;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.config.server.EnableConfigServer;  
  
@SpringBootApplication  
@EnableConfigServer  
public class ConfigServerApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(ConfigServerApplication.class, args);  
 }  
  
}

**Application.yml**

spring:  
 profiles:  
 active: native  
 application:  
 name: config-server  
 cloud:  
 config:  
 server:  
 native:  
 search-locations: classpath:/configurations  
  
server:  
 port: 8888

**Project: eureka-server**

**EurekaServerApplication.java**

package com.library.eurekaserver;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;  
  
@SpringBootApplication  
@EnableEurekaServer  
public class EurekaServerApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(EurekaServerApplication.class, args);  
 }  
  
}

**Application.yml**

spring:  
 application:  
 name: eureka-server  
 config:  
 import: optional:configserver:http://localhost:8888

**Project: inventory-service**

**InventoryServiceApplication.java**

package com.library.inventoryservice;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class InventoryServiceApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(InventoryServiceApplication.class, args);  
 }  
  
}

**Application.yml**

spring:  
 application:  
 name: inventory-service  
  
 config:  
 import: optional:configserver:http://localhost:8888

**Project: product-service**

**ProductController.java**

package com.library.productservice.controller;  
  
  
import com.library.productservice.model.Product;  
import com.library.productservice.service.ProductService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
import java.util.Optional;  
  
@RestController  
@RequestMapping("/api/products")  
public class ProductController {  
  
 @Autowired  
 private ProductService productService;  
  
 @GetMapping  
 public List<Product> getAllProducts() {  
 return productService.getAllProducts();  
 }  
  
 @GetMapping("/{id}")  
 public Optional<Product> getProductById(@PathVariable Long id) {  
 return productService.getProductById(id);  
 }  
  
 @PostMapping  
 public Product createProduct(@RequestBody Product product) {  
 return productService.createProduct(product);  
 }  
  
 @DeleteMapping("/{id}")  
 public void deleteProduct(@PathVariable Long id) {  
 productService.deleteProduct(id);  
 }  
}

**Product.java**

package com.library.productservice.model;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
  
@Entity  
public class Product {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private String name;  
  
  
  
 private String description;  
 private double price;  
 public Product() {  
 }  
 public Product(Long id, String name, String description, double price) {  
 this.id = id;  
 this.name = name;  
 this.description = description;  
 this.price = price;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public void setDescription(String description) {  
 this.description = description;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double price) {  
 this.price = price;  
 }   
}

**ProductRepository.java**

package com.library.productservice.repository;  
  
  
import com.library.productservice.model.Product;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface ProductRepository extends JpaRepository<Product, Long> {  
}

**ProductService.java**

package com.library.productservice.service;  
  
  
import com.library.productservice.model.Product;  
import com.library.productservice.repository.ProductRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class ProductService {  
  
 @Autowired  
 private ProductRepository productRepository;  
  
 public List<Product> getAllProducts() {  
 return productRepository.findAll();  
 }  
  
 public Optional<Product> getProductById(Long id) {  
 return productRepository.findById(id);  
 }  
  
 public Product createProduct(Product product) {  
 return productRepository.save(product);  
 }  
  
 public void deleteProduct(Long id) {  
 productRepository.deleteById(id);  
 }  
}

**ProductServiceApplication.java**

package com.library.productservice;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class ProductServiceApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(ProductServiceApplication.class, args);  
 }  
  
}

**Application.yml**

spring:  
 application:  
 name: product-service  
  
 config:  
 import: optional:configserver:http://localhost:8888

Configuration Files:

Application.yml

eureka:  
 instance:  
 hostname: localhost  
 client:  
 service-url:  
 defaultZone: http://localhost:7777/eureka/

eureka-server.yml

server:  
 port: 7777  
  
eureka:  
 server:  
 enable-self-preservation: false  
 instance:  
 hostname: localhost  
 client:  
 register-with-eureka: false  
 fetch-registry: false  
 service-url:  
 defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/

inventory-service.yml

server:  
 port: 8085  
  
spring:  
 datasource:  
 url: jdbc:mysql://localhost:3306/inventorydb  
 username: root  
 password: password  
 driver-class-name: com.mysql.cj.jdbc.Driver  
  
 jpa:  
 hibernate:  
 ddl-auto: update  
 show-sql: true  
 database-platform: org.hibernate.dialect.MySQL8Dialect

order-service.yml

server:  
 port: 8086  
  
spring:  
 datasource:  
 url: jdbc:mysql://localhost:3306/orderdb  
 username: root  
 password: password  
 driver-class-name: com.mysql.cj.jdbc.Driver  
  
 jpa:  
 hibernate:  
 ddl-auto: update  
 show-sql: true  
 database-platform: org.hibernate.dialect.MySQL8Dialect

product-service.yml

server:  
 port: 8087  
  
spring:  
 datasource:  
 url: jdbc:mysql://localhost:3306/productdb  
 username: root  
 password: password  
 driver-class-name: com.mysql.cj.jdbc.Driver  
  
 jpa:  
 hibernate:  
 ddl-auto: update  
 show-sql: true  
 database-platform: org.hibernate.dialect.MySQL8Dialect

user-service.yml

server:  
 port: 8088  
  
spring:  
 datasource:  
 url: jdbc:mysql://localhost:3306/userdb  
 username: root  
 password: password  
 jpa:  
 hibernate:  
 ddl-auto: update  
 show-sql: true

Project: eureka-server

EurekaServerApplication.java

package com.library.eurekaserver;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;  
  
@SpringBootApplication  
@EnableEurekaServer  
public class EurekaServerApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(EurekaServerApplication.class, args);  
 }  
  
}

application.yml

spring:  
 application:  
 name: eureka-server  
 config:  
 import: optional:configserver:http://localhost:8888

pom.xml

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.library</groupId>  
 <artifactId>eureka-server</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>eureka-server</name>  
 <description>eureka-server</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>21</java.version>  
 <spring-cloud.version>2025.0.0</spring-cloud.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-config</artifactId>  
 </dependency>  
 </dependencies>  
 <dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-dependencies</artifactId>  
 <version>${spring-cloud.version}</version>  
 <type>pom</type>  
 <scope>import</scope>  
 </dependency>  
 </dependencies>  
 </dependencyManagement>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

Project: inventory-service

InventoryController.java

package com.library.inventoryservice.controller;  
  
import com.library.inventoryservice.model.Inventory;  
import com.library.inventoryservice.repository.InventoryRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/inventory")  
public class InventoryController {  
  
 @Autowired  
 private InventoryRepository repository;  
  
 @GetMapping  
 public List<Inventory> getAll() {  
 return repository.findAll();  
 }  
  
 @PostMapping  
 public Inventory create(@RequestBody Inventory inventory) {  
 return repository.save(inventory);  
 }  
  
 @GetMapping("/product/{productId}")  
 public Inventory getByProductId(@PathVariable Long productId) {  
 return repository.findByProductId(productId);  
 }  
  
 @PutMapping("/product/{productId}")  
 public Inventory updateStock(@PathVariable Long productId, @RequestBody Inventory newInventory) {  
 Inventory inventory = repository.findByProductId(productId);  
 if (inventory != null) {  
 inventory.setStock(newInventory.getStock());  
 return repository.save(inventory);  
 }  
 return null;  
 }  
}

Inventory.java

package com.library.inventoryservice.model;  
  
import jakarta.persistence.\*;  
  
@Entity  
@Table(name = "inventory")  
public class Inventory {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private Long productId;  
 private int stock;  
 private int quantity;  
  
 public Inventory() {  
 }  
  
 public Inventory(int quantity, int stock, Long productId, Long id) {  
 this.quantity = quantity;  
 this.stock = stock;  
 this.productId = productId;  
 this.id = id;  
 }  
  
 public Inventory(Long productId, int stock) {  
 this.productId = productId;  
 this.stock = stock;  
 }  
  
 // Getters and setters  
 public Long getId() {  
 return id;  
 }  
  
 public Long getProductId() {  
 return productId;  
 }  
  
 public void setProductId(Long productId) {  
 this.productId = productId;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public int getQuantity() {  
 return quantity;  
 }  
  
 public void setQuantity(int quantity) {  
 this.quantity = quantity;  
 }  
  
 public int getStock() {  
 return stock;  
 }  
  
 public void setStock(int stock) {  
 this.stock = stock;  
 }  
  
  
}

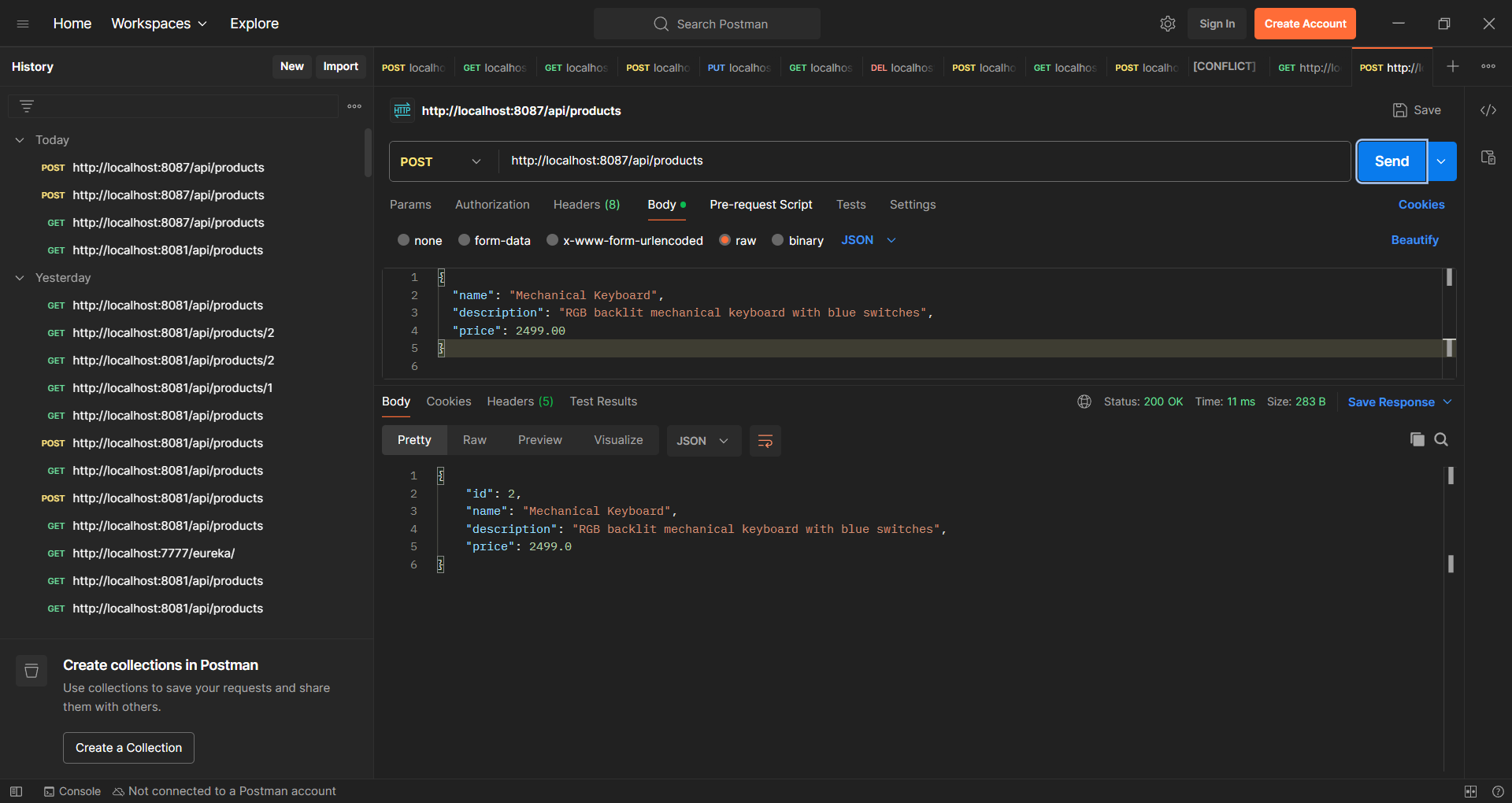
InventoryRepository.java

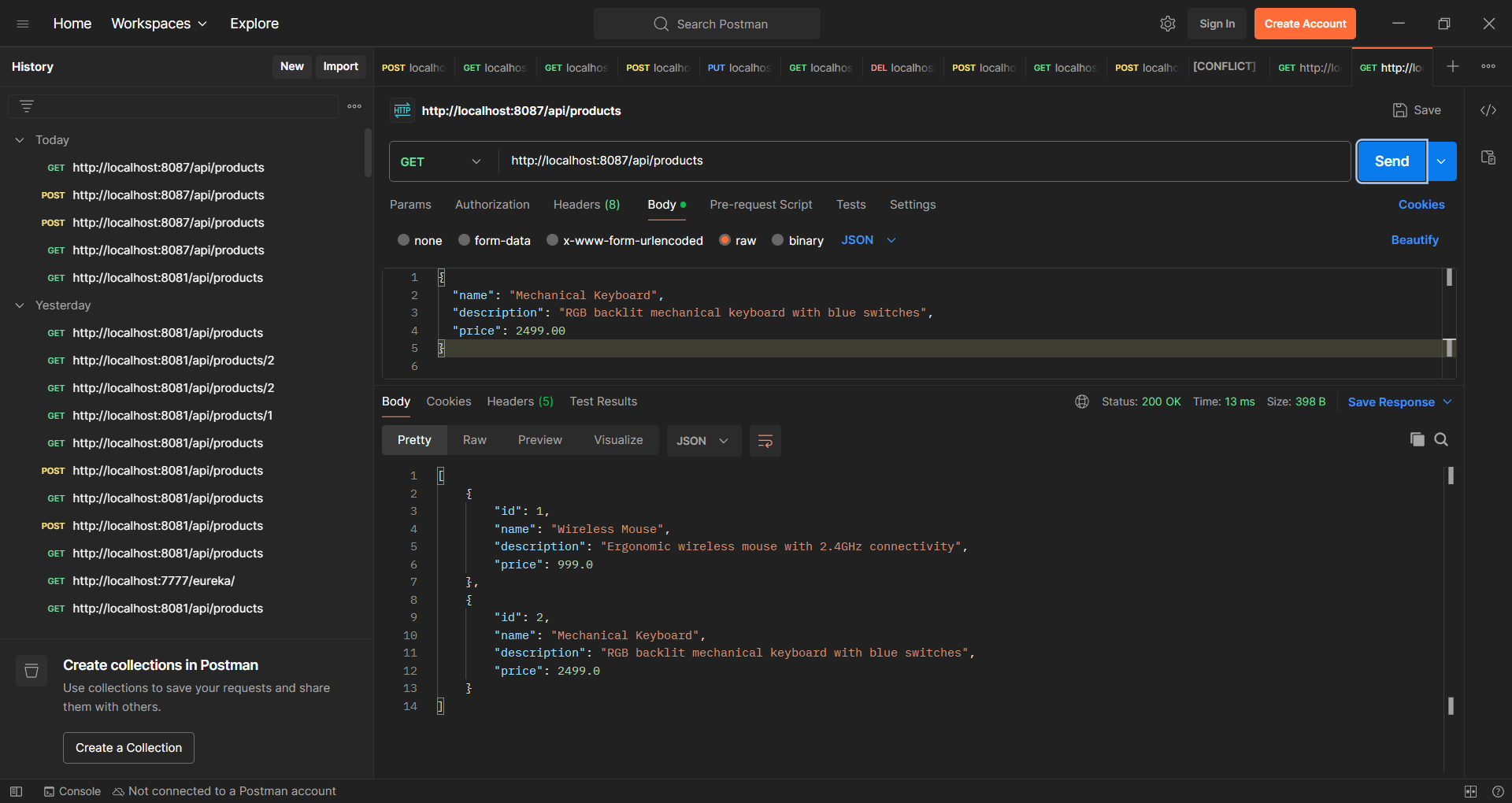
package com.library.inventoryservice.repository;  
  
import com.library.inventoryservice.model.Inventory;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface InventoryRepository extends JpaRepository<Inventory, Long> {  
 Inventory findByProductId(Long productId);  
}

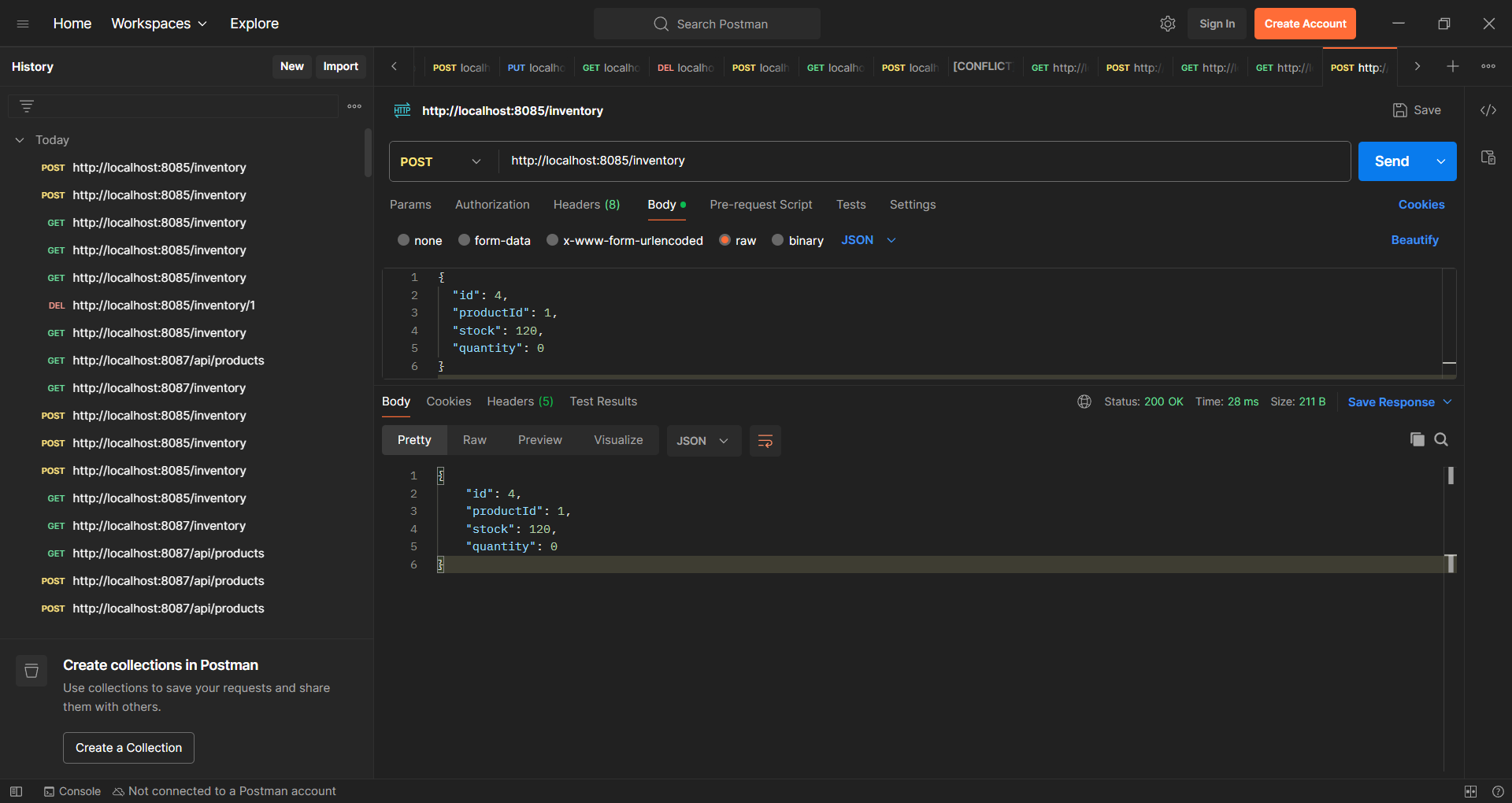
InventoryService.java

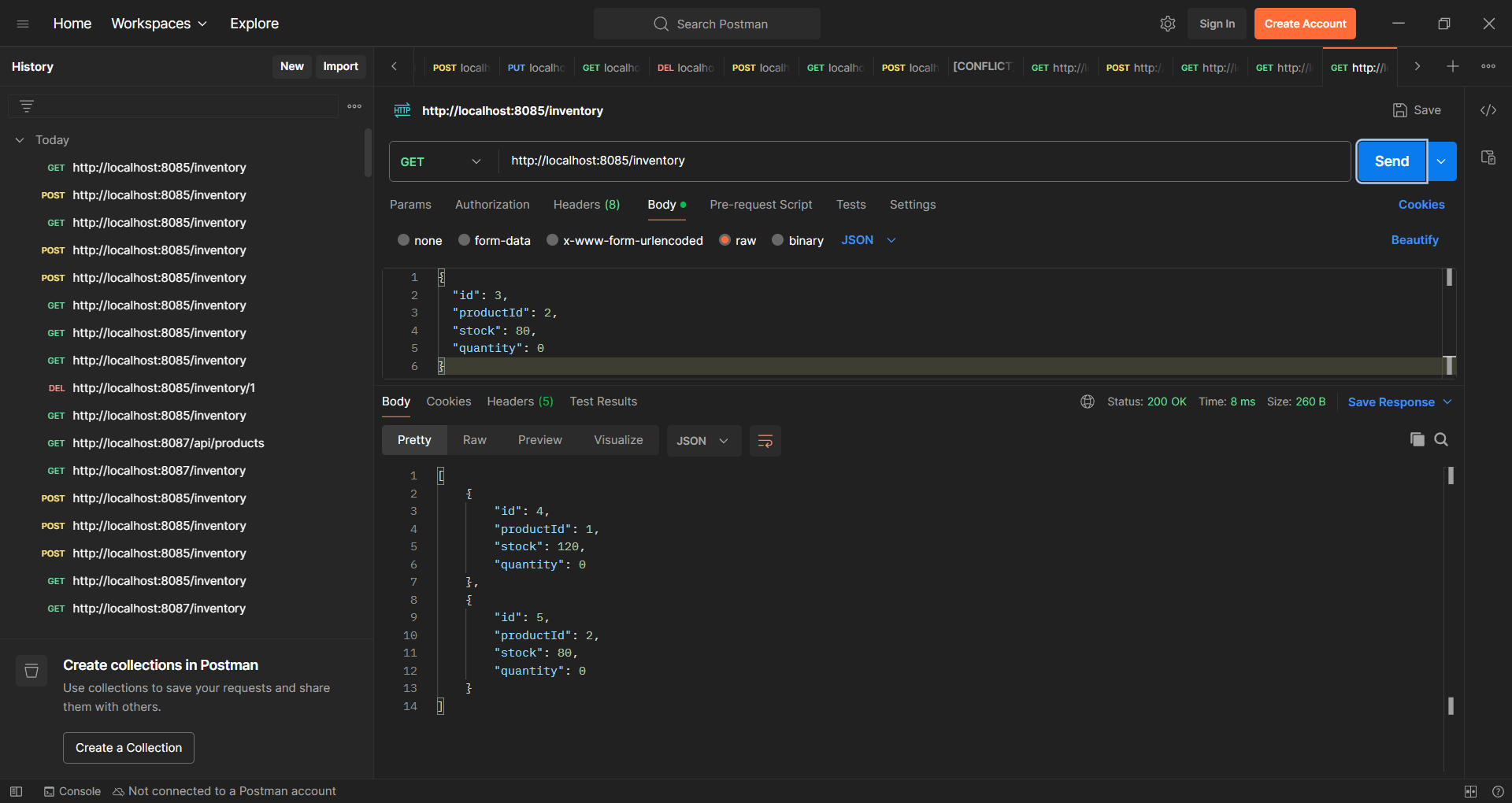
package com.library.inventoryservice.service;  
  
import com.library.inventoryservice.model.Inventory;  
import com.library.inventoryservice.repository.InventoryRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class InventoryService {  
  
 private final InventoryRepository inventoryRepository;  
  
 @Autowired  
 public InventoryService(InventoryRepository inventoryRepository) {  
 this.inventoryRepository = inventoryRepository;  
 }  
  
 public List<Inventory> getAllInventory() {  
 return inventoryRepository.findAll();  
 }  
  
 public Inventory getInventoryByProductId(Long productId) {  
 return inventoryRepository.findByProductId(productId);  
 }  
  
 public Inventory saveInventory(Inventory inventory) {  
 return inventoryRepository.save(inventory);  
 }  
  
 public Inventory updateStock(Long productId, int quantity) {  
 Inventory inventory = inventoryRepository.findByProductId(productId);  
 if (inventory != null) {  
 inventory.setQuantity(quantity);  
 return inventoryRepository.save(inventory);  
 }  
 return null;  
 }  
  
 public void deleteInventory(Long productId) {  
 Inventory inventory = inventoryRepository.findByProductId(productId);  
 if (inventory != null) {  
 inventoryRepository.delete(inventory);  
 }  
 }  
}

Output:









3. Implement an API Gateway Problem:

Create an API Gateway to route requests to:

• Customer Service

• Billing Service

**Requirements:**

• Use Spring Cloud Gateway.

• Implement rate limiting, caching, and path rewriting.

**Step 1: Create Customer Service Microservice**

**1.1 Generate project**

Go to <https://start.spring.io/>

* Project: Maven Project
* Language: Java
* Spring Boot: 2.7.13 (or latest stable 2.x)
* Group: com.example
* Artifact: customer-service
* Dependencies:
  + Spring Web (for REST API)

1.2 Create a simple REST Controller

package com.example.customerservice;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class CustomerServiceApplication {

public static void main(String[] args) {

SpringApplication.run(CustomerServiceApplication.class, args);

}

}

**Create a controller class:**

package com.example.customerservice.controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class CustomerController {

@GetMapping("/hello")

public String sayHello() {

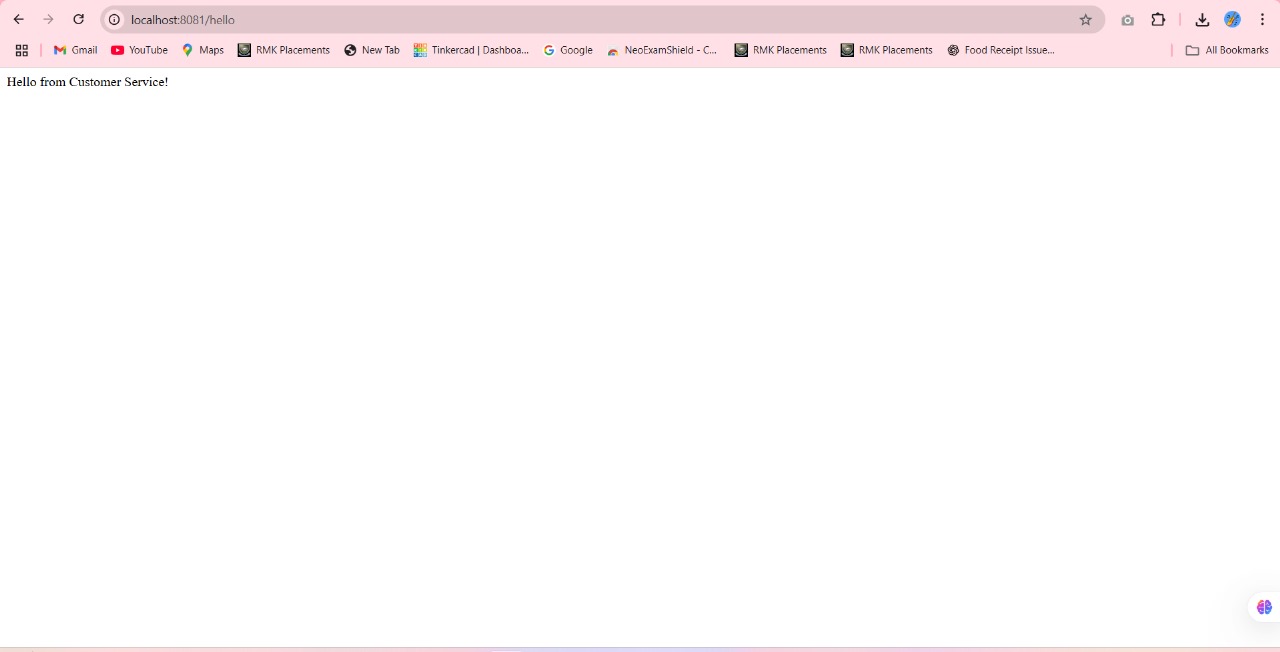
return "Hello from Customer Service!";

}

}

**1.4 Test Customer Service**

<http://localhost:8081/hello>



**Step 2: Create Billing Service Microservice**

**2.1 Generate project from Spring Initializr**

* Artifact: billing-service
* Add Spring Web

**2.2 Create main app class**

package com.example.billingservice;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class BillingServiceApplication {

public static void main(String[] args) {

SpringApplication.run(BillingServiceApplication.class, args);

}

}

**2.3 Create controller**

package com.example.billingservice.controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class BillingController {

@GetMapping("/hello")

public String sayHello() {

return "Hello from Billing Service!";

}

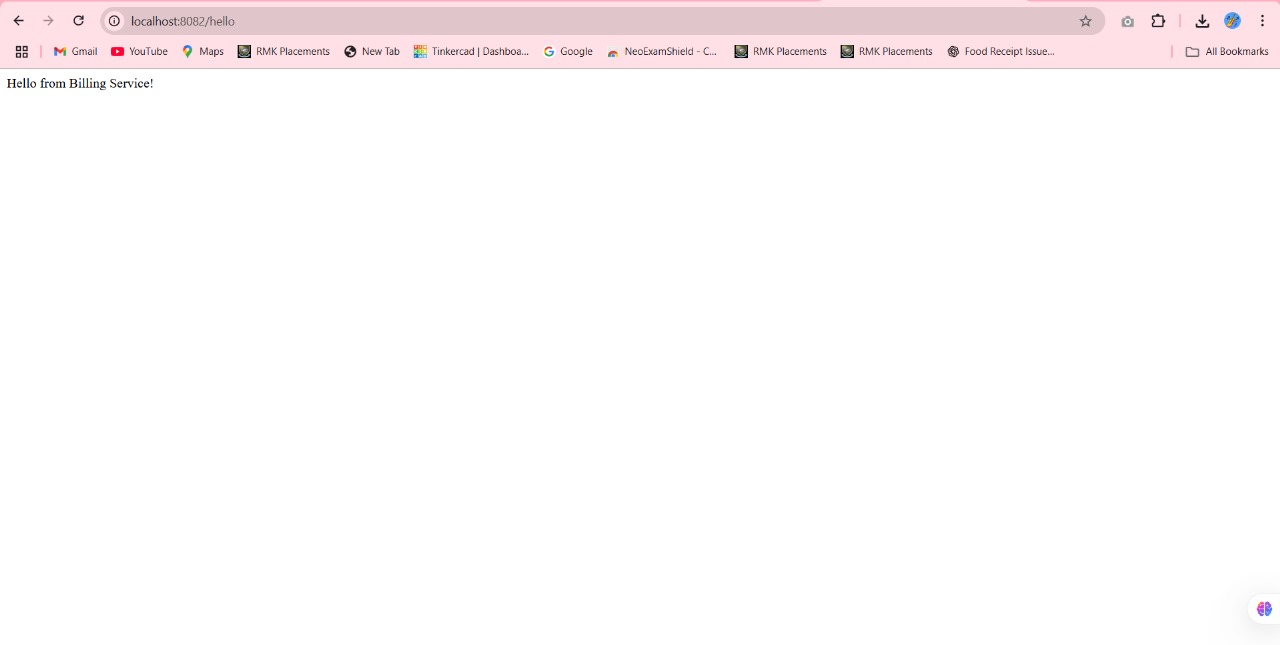
}

**2.4 Change port to 8082**

server.port=8082

**2.5 Run Billing Service**

<http://localhost:8082/hello>



**Step 3: Create API Gateway**

**3.1 Generate project**

Use Spring Initializr:

* Artifact: api-gateway
* Dependencies:
  + Spring Cloud Gateway
  + Spring Boot Actuator (optional)

**3.2 Create main app**

package com.example.apigateway;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class ApiGatewayApplication {

public static void main(String[] args) {

SpringApplication.run(ApiGatewayApplication.class, args);

}

}

**3.3 Configure routing in application.yml**

server:

port: 8080

spring:

cloud:

gateway:

routes:

- id: customer-service

uri: http://localhost:8081

predicates:

- Path=/customer/\*\*

filters:

- RewritePath=/customer/(?<segment>.\*), /$\{segment}

- id: billing-service

uri: http://localhost:8082

predicates:

- Path=/billing/\*\*

filters:

- RewritePath=/billing/(?<segment>.\*), /$\{segment}

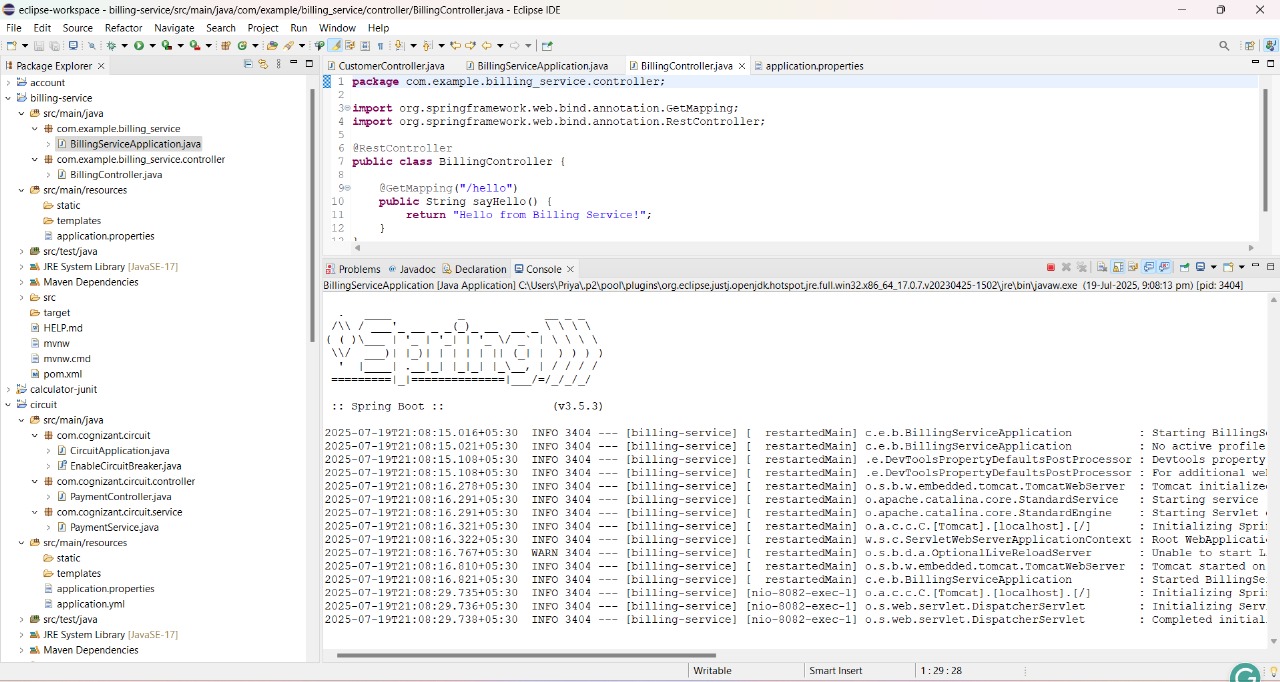
**3.4 Run API Gateway**

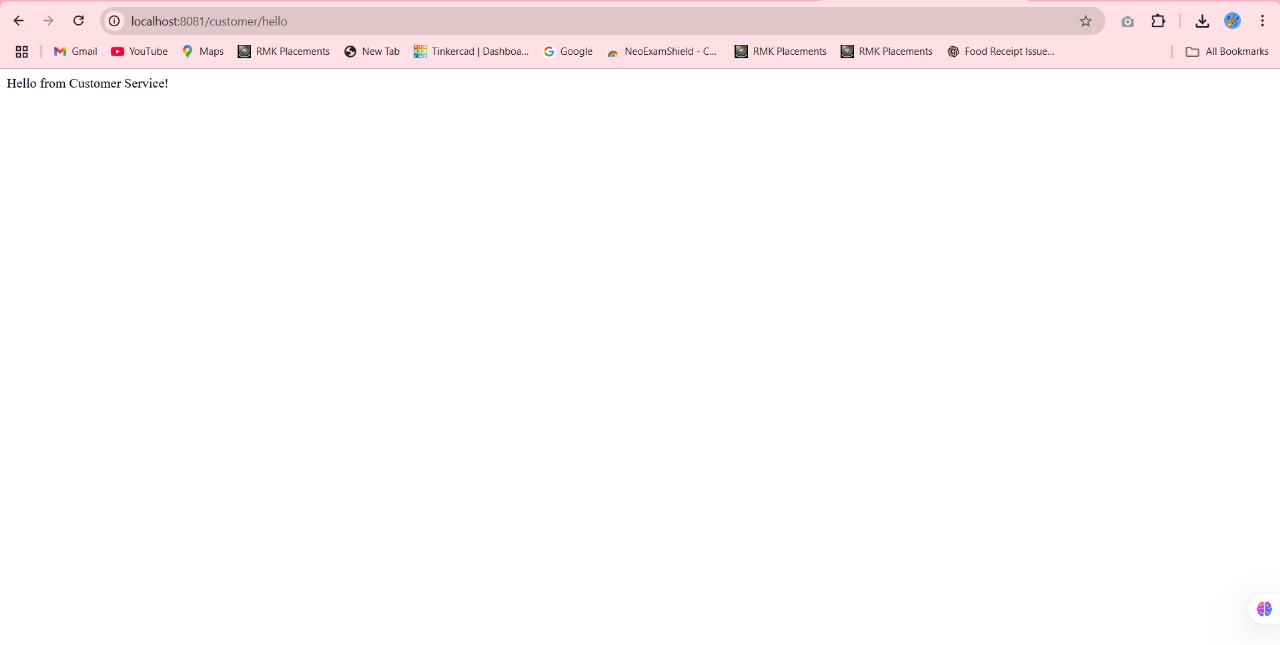
Run your ApiGatewayApplication.

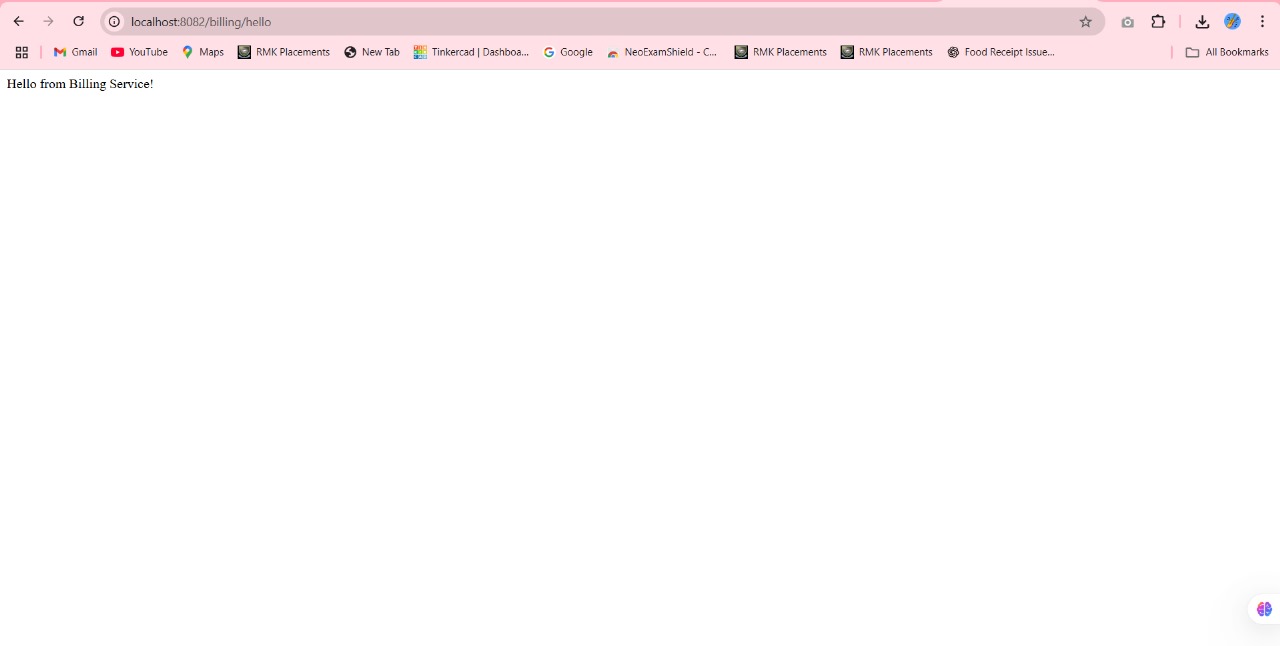
**3.5 Test Gateway Routing**

Test these URLs:

* http://localhost:8080/customer/hello → Should return **Hello from Customer Service!**
* http://localhost:8080/billing/hello → Should return **Hello from Billing Service!**







4. Resilient Microservices with Circuit Breaker

Problem: A Payment Service calls a slow third-party API.

Requirements:

• Implement Circuit Breaker and fallback logic using Resilience4j.

• Log and monitor fallback events.

**Step 1: Create a Spring Boot Project:**

You can use [Spring Initializr](https://start.spring.io/) or your IDE.

**Add these dependencies**:

* Spring Web
* Spring Boot Actuator
* Spring Boot DevTools
* Resilience4j
* (Optional) Lombok

**pom.xml**

<dependency>

<groupId>io.github.resilience4j</groupId>

<artifactId>resilience4j-spring-boot3</artifactId>

</dependency>

<dependency>

<groupId>io.github.resilience4j</groupId>

<artifactId>resilience4j-timelimiter</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

**Step 2: Create the Payment Service:**

Let’s simulate a **slow third-party API** call from your PaymentService.

@Service

public class PaymentService {

@CircuitBreaker(name = "paymentService", fallbackMethod = "fallbackPayment")

public String processPayment() {

// Simulate a slow third-party API (delay)

try {

Thread.sleep(5000); // 5 seconds

} catch (InterruptedException e) {

e.printStackTrace();

}

return "Payment processed successfully!";

}

public String fallbackPayment(Throwable t) {

return "Payment service is currently unavailable. Please try again later.";

}

}

**Step 3: Create Controller:**

@RestController

@RequestMapping("/api/payment")

public class PaymentController {

@Autowired

private PaymentService paymentService;

@GetMapping

public String makePayment() {

return paymentService.processPayment();

}

}

**Step 4: Configure application.yml**

resilience4j:

circuitbreaker:

instances:

paymentService:

registerHealthIndicator: true

slidingWindowSize: 5

failureRateThreshold: 50

waitDurationInOpenState: 10s

permittedNumberOfCallsInHalfOpenState: 2

minimumNumberOfCalls: 5

automaticTransitionFromOpenToHalfOpenEnabled: true

management:

endpoints:

web:

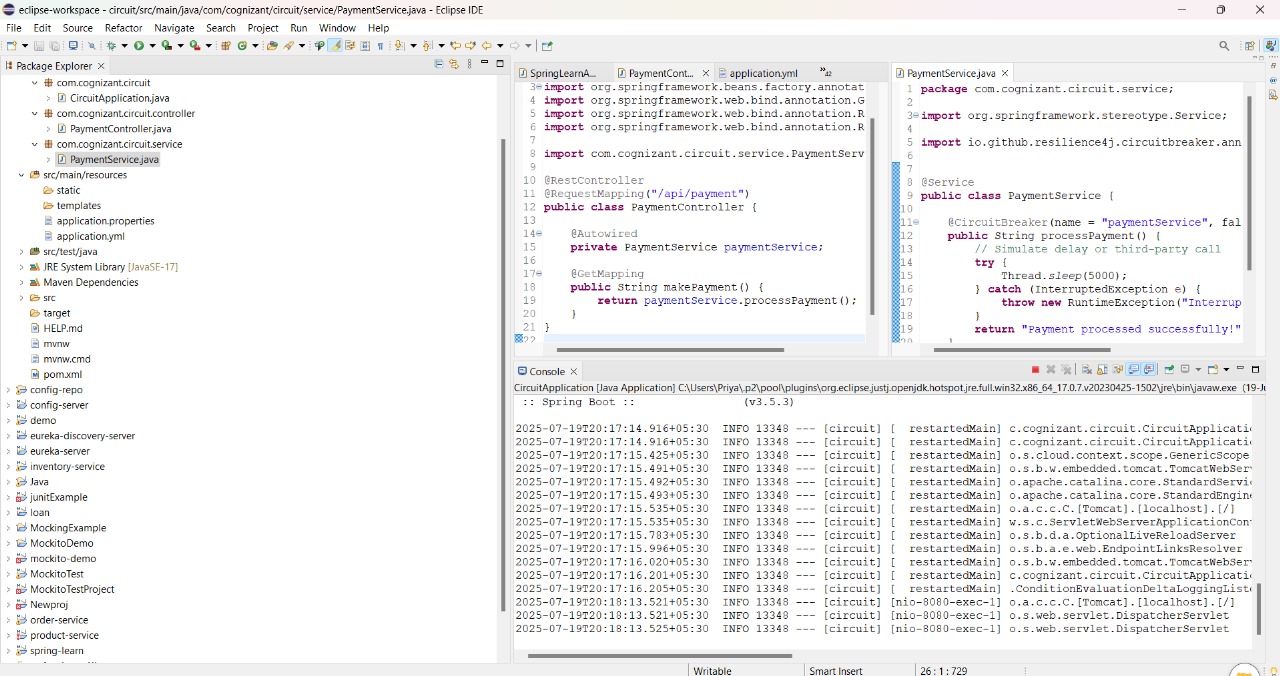
exposure:

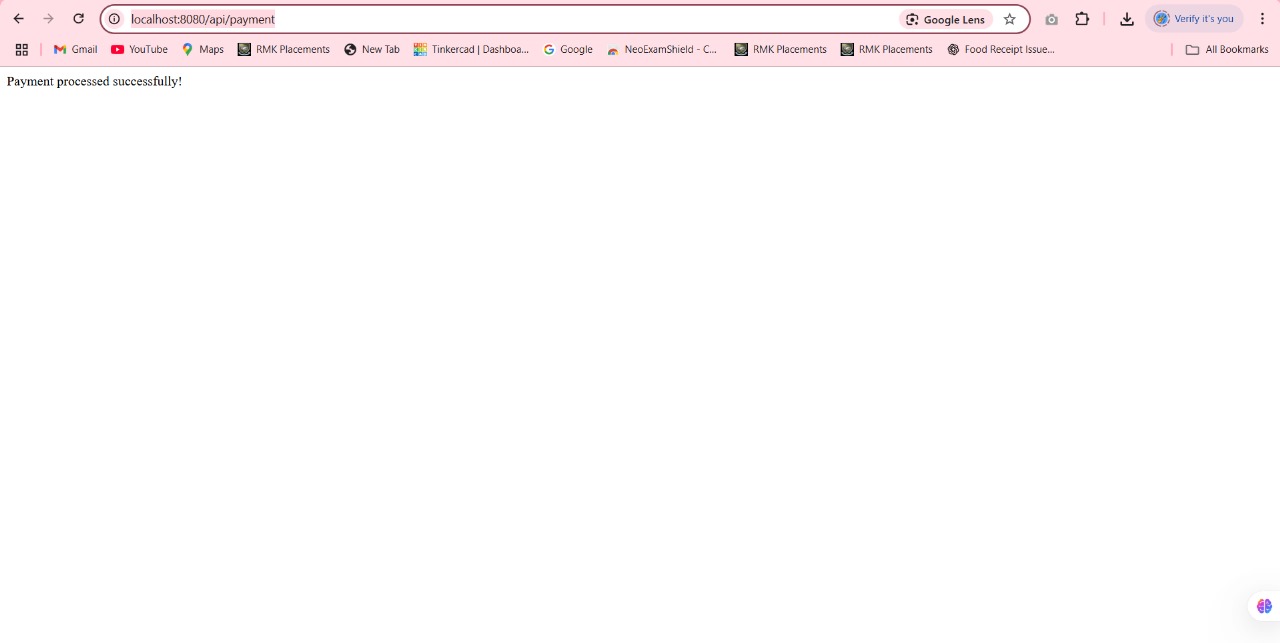
include: "\*"

**Step 5: Test It**

1. Run the app.
2. Call http://localhost:8080/api/payment.
3. It’ll delay for 5 seconds.
4. After 50% of the calls fail (e.g., timeouts or exceptions), circuit opens.
5. You’ll get the **fallback response** instantly.

**OUTPUT:**





**Force a Failure to Trigger the Fallback:**

