DAY-15 ASSIGNMENT

EurekaApplication.java

```
package com.example.eureka;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication
@EnableEurekaServer
public class EurekaApplication {

	public static void main(String[] args) {
		SpringApplication.run(EurekaApplication.class, args);
	}

}
```

Application.properties

```
spring.application.name=eureka

server.port=8761

eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

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>3.5.4</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.wipro</groupId>
	<artifactId>Assignment1_Day15_ECommerceMicroservices</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>Assignment1_Day15_ECommerceMicroservices</name>
	<description>Demo Project for Spring Boot Microservices</description>
	<url/>
```

```
<licenses>
	<license/>
</licenses>
<developers>
	<developer/>
</developers>
<scm>
	<connection/>
	<developerConnection/>
	<tag/>
	<url/>
</scm>
<properties>
	<java.version>17</java.version>
	<spring-cloud.version>2025.0.0</spring-cloud.version>
</properties>
<dependencies>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-data-jpa</artifactId>
	</dependency>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-web</artifactId>
	</dependency>
	<dependency>
		<groupId>org.springframework.cloud</groupId>
		<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
	</dependency>
	<dependency>
		<groupId>org.springframework.cloud</groupId>
		<artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
	</dependency>
	<dependency>
		<groupId>org.springframework.cloud</groupId>
		<artifactId>spring-cloud-starter-openfeign</artifactId>
	</dependency>

	<dependency>
		<groupId>com.h2database</groupId>
		<artifactId>h2</artifactId>
		<scope>runtime</scope>
	</dependency>
	<dependency>
		<groupId>org.projectlombok</groupId>
		<artifactId>lombok</artifactId>
		<optional>true</optional>
	</dependency>
	<dependency>
		<groupId>org.springframework.boot</groupId>
```

```
                <artifactId>spring-boot-starter-test</artifactId>
                <scope>test</scope>
            </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.apache.maven.plugins</groupId>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <configuration>
                        <annotationProcessorPaths>
                            <path>
                                <groupId>org.projectlombok</groupId>
                                <artifactId>lombok</artifactId>
                            </path>
                        </annotationProcessorPaths>
                    </configuration>
                </plugin>
                <plugin>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-maven-plugin</artifactId>
                    <configuration>
                        <excludes>
                            <exclude>
                                <groupId>org.projectlombok</groupId>
                                <artifactId>lombok</artifactId>
                            </exclude>
                        </excludes>
                    </configuration>
                </plugin>
            </plugins>
        </build>

</project>
```

ProductServiceApplication.java

```
package com.example.productservice;
```

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@EnableDiscoveryClient
@SpringBootApplication
public class ProductserviceApplication {

	public static void main(String[] args) {
		SpringApplication.run(ProductserviceApplication.class, args);
	}

}
```

ProductController.java

```
package com.example.productservice.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.example.productservice.entity.Product;
import com.example.productservice.repository.ProductRepository;

@RestController
@RequestMapping("/products")
public class ProductController {

   @Autowired
   private ProductRepository repo;

   @PostMapping
   public Product addProduct(@RequestBody Product p) {
      return repo.save(p);
   }

   @GetMapping("/{id}")
   public Product getProduct(@PathVariable String id) {
      return repo.findById(id).orElse(null);
   }
```

```
    @GetMapping
    public List<Product> getAll() {
        return repo.findAll();
    }

    @PutMapping("/{id}/reduceStock")
    public ResponseEntity<String> reduceStock(@PathVariable String id, @RequestParam
int qty) {
        Product product = repo.findById(id).orElse(null);
        if (product != null && product.getStock() >= qty) {
            product.setStock(product.getStock() - qty);
            repo.save(product);
            return ResponseEntity.ok("Stock reduced.");
        }
        return ResponseEntity.badRequest().body("Insufficient stock.");
    }
}
```

Product.java

```
package com.example.productservice.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;

@Entity
public class Product {
    @Id
    private String id;
    private String name;
    private double price;
    private int stock;
        public String getId() {
                return id;
        }
        public void setId(String id) {
                this.id = id;
        }
        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
        public double getPrice() {
                return price;
        }
        public void setPrice(double price) {
                this.price = price;
        }
        public int getStock() {
```

```
		return stock;
	}
	public void setStock(int stock) {
		this.stock = stock;
	}

}
```

ProductRepository.java

```
package com.example.productservice.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.example.productservice.entity.Product;

@Repository
public interface ProductRepository extends JpaRepository<Product, String> {
}
```

Application.properties

```
spring.application.name=product-service
server.port=8081
spring.datasource.url=jdbc:h2:mem:productdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.h2.console.enabled=true
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
management.endpoints.web.exposure.include=*
management.endpoint.health.show-details=always
info.app.name=Product Service
info.app.version=1.0.0
```

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>3.5.4</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.example</groupId>
```

```
	<artifactId>productservice1</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>product-service1</name>
	<description>Demo project for Spring Boot</description>
	<url/>
	<licenses>
		<license/>
	</licenses>
	<developers>
		<developer/>
	</developers>
	<scm>
		<connection/>
		<developerConnection/>
		<tag/>
		<url/>
	</scm>
	<properties>
		<java.version>17</java.version>
		<spring-cloud.version>2025.0.0</spring-cloud.version>
	</properties>
	<dependencies>
	<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-devtools</artifactId>
			<scope>runtime</scope>
			<optional>true</optional>
		</dependency>
		<dependency>
			<groupId>com.h2database</groupId>
			<artifactId>h2</artifactId>
			<scope>runtime</scope>
```

```
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</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>
```

OrderServiceApplication.java

```
package com.example.orderservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.openfeign.EnableFeignClients;

@SpringBootApplication
@EnableDiscoveryClient
@EnableFeignClients
public class OrderserviceApplication {

	public static void main(String[] args) {
		SpringApplication.run(OrderserviceApplication.class, args);
	}

}
```

OrderController.java

```
package com.example.orderservice.controller;
import java.util.Optional;
import java.util.UUID;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import com.example.orderservice.DTO.OrderRequest;
import com.example.orderservice.DTO.PaymentRequest;
import com.example.orderservice.DTO.PaymentResponse;
import com.example.orderservice.DTO.ProductResponse;
import com.example.orderservice.entity.Order;
import com.example.orderservice.feign.PaymentClient;
import com.example.orderservice.feign.ProductClient;
import com.example.orderservice.repository.OrderRepository;
@RestController
@RequestMapping("/orders")
public class OrderController {
   @Autowired
   private OrderRepository orderRepo;
   @Autowired
   private ProductClient productClient;
   @Autowired
   private PaymentClient paymentClient;
   @PostMapping
   public ResponseEntity<?> placeOrder(@RequestBody OrderRequest request) {
      ProductResponse product = productClient.getProductById(request.getProductId());
      if (product == null || product.getStock() < request.getQuantity()) {
         return ResponseEntity.badRequest().body("Product unavailable or insufficient
stock.");
      }
      double total = product.getPrice() * request.getQuantity();
      String orderId = UUID.randomUUID().toString();
      Order order = new Order();
      order.setId(orderId);
      order.setProductId(request.getProductId());
      order.setQuantity(request.getQuantity());
      order.setTotalAmount(total);
      order.setStatus("PENDING_PAYMENT");
      orderRepo.save(order);
      PaymentRequest paymentReq = new PaymentRequest();
      paymentReq.setOrderId(orderId);
      paymentReq.setAmount(total);
      paymentReq.setPaymentMethod(request.getPaymentMethod());
      PaymentResponse paymentResp = paymentClient.processPayment(paymentReq);
      if ("SUCCESS".equalsIgnoreCase(paymentResp.getStatus())) {
         order.setStatus("CONFIRMED");
         productClient.reduceStock(product.getId(), request.getQuantity());
      } else {
         order.setStatus("FAILED");
      }
```

```
        orderRepo.save(order);
        return ResponseEntity.ok(order);
    }
    @GetMapping("/{orderId}")
    public ResponseEntity<?> getOrderById(@PathVariable String orderId) {
        Optional<Order> order = orderRepo.findById(orderId);
        return order.map(ResponseEntity::ok)
                .orElseGet(() -> ResponseEntity.notFound().build());
    }
    @PutMapping("/{orderId}")
    public ResponseEntity<?> updateOrder(
            @PathVariable String orderId,
            @RequestBody OrderRequest request) {

        Optional<Order> existingOrderOpt = orderRepo.findById(orderId);
        if (existingOrderOpt.isEmpty()) {
            return ResponseEntity.notFound().build();
        }
        Order order = existingOrderOpt.get();
        ProductResponse product = productClient.getProductById(request.getProductId());
        if (product == null || product.getStock() < request.getQuantity()) {
            return ResponseEntity.badRequest().body("Product unavailable or insufficient stock
for update.");
        }
        order.setProductId(request.getProductId());
        order.setQuantity(request.getQuantity());
        order.setTotalAmount(product.getPrice() * request.getQuantity());
        order.setStatus("UPDATED");
        orderRepo.save(order);
       return ResponseEntity.ok(order);
    }
    @DeleteMapping("/{orderId}")
    public ResponseEntity<?> deleteOrder(@PathVariable String orderId) {
        Optional<Order> order = orderRepo.findById(orderId);
        if (order.isEmpty()) {
            return ResponseEntity.notFound().build();
        }
        orderRepo.deleteById(orderId);
        return ResponseEntity.ok("Order with ID " + orderId + " has been cancelled and
deleted.");
    }
}
```

OrderRequest.java

```
package com.example.orderservice.DTO;

public class OrderRequest {
    private String productId;
    private int quantity;
    private String paymentMethod;
```

```
    public String getProductId() {
        return productId;
    }
    public void setProductId(String productId) {
        this.productId = productId;
    }
    public int getQuantity() {
        return quantity;
    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
    public String getPaymentMethod() {
        return paymentMethod;
    }
    public void setPaymentMethod(String paymentMethod) {
        this.paymentMethod = paymentMethod;
    }
}
```

PaymentRequest.java

```
package com.example.orderservice.DTO;

public class PaymentRequest {
    private String orderId;
    private double amount;
    private String paymentMethod;

    public String getOrderId() {
        return orderId;
    }
    public void setOrderId(String orderId) {
        this.orderId = orderId;
    }
    public double getAmount() {
        return amount;
    }
    public void setAmount(double amount) {
        this.amount = amount;
    }
    public String getPaymentMethod() {
        return paymentMethod;
    }
    public void setPaymentMethod(String paymentMethod) {
        this.paymentMethod = paymentMethod;
    }
}
```

PaymentResponse.java

```
package com.example.orderservice.DTO;

public class PaymentResponse {
    private String status;

    public String getStatus() {
        return status;
    }
    public void setStatus(String status) {
        this.status = status;
    }
}
```

ProductResponse.java

```
package com.example.orderservice.DTO;

public class ProductResponse {
    private String id;
    private String name;
    private double price;
    private int stock;

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {
        this.price = price;
    }
    public int getStock() {
        return stock;
    }
    public void setStock(int stock) {
        this.stock = stock;
    }
}
```

Order.java

```
package com.example.orderservice.entity;
```

```java
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name = "orders")
public class Order {
    @Id
    private String id;
    private String productId;
    private int quantity;
    private double totalAmount;
    private String status;

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getProductId() {
        return productId;
    }
    public void setProductId(String productId) {
        this.productId = productId;
    }
    public int getQuantity() {
        return quantity;
    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
    public double getTotalAmount() {
        return totalAmount;
    }
    public void setTotalAmount(double totalAmount) {
        this.totalAmount = totalAmount;
    }
    public String getStatus() {
        return status;
    }
    public void setStatus(String status) {
        this.status = status;
    }
}
```

PaymentClient.java

```java
package com.example.orderservice.feign;
```

```
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;

import com.example.orderservice.DTO.PaymentRequest;
import com.example.orderservice.DTO.PaymentResponse;

@FeignClient(name = "payment-service")
public interface PaymentClient {
    @PostMapping("/payments")
    PaymentResponse processPayment(@RequestBody PaymentRequest paymentRequest);
}
```

ProductClient.java

```
package com.example.orderservice.feign;

import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestParam;

import com.example.orderservice.DTO.ProductResponse;

@FeignClient(name = "product-service")
public interface ProductClient {
    @GetMapping("/products/{id}")
    ProductResponse getProductById(@PathVariable String id);

    @PutMapping("/products/{id}/reduceStock")
    ResponseEntity<String> reduceStock(@PathVariable String id, @RequestParam int qty);
}
```

OrderRepository.java

```
package com.example.orderservice.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.example.orderservice.entity.Order;

@Repository
public interface OrderRepository extends JpaRepository<Order, String> {
}
```

OrderService.java

```
package com.example.orderservice.service;

import org.springframework.stereotype.Service;
```

```
import com.example.orderservice.entity.Order;

import java.util.*;
import java.util.concurrent.ConcurrentHashMap;

@Service
public class OrderService {

   private final Map<String, Order> orderMap = new ConcurrentHashMap<>();

   public Order saveOrder(Order order) {
      orderMap.put(order.getId(), order);
      return order;
   }

   public List<Order> getAllOrders() {
      return new ArrayList<>(orderMap.values());
   }

   public Optional<Order> getOrderById(String id) {
      return Optional.ofNullable(orderMap.get(id));
   }
}
```

Application.properties

```
server.port=8082
spring.application.name=order-service
spring.datasource.url=jdbc:h2:mem:orderdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.h2.console.enabled=true
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
management.endpoints.web.exposure.include=*
management.endpoint.health.show-details=always
info.app.name=Order Service
info.app.version=1.0.0
```

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>3.5.4</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.example</groupId>
	<artifactId>orderservice2</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>order-service2</name>
	<description>Demo project for Spring Boot</description>
	<url/>
	<licenses>
		<license/>
	</licenses>
	<developers>
		<developer/>
	</developers>
	<scm>
		<connection/>
		<developerConnection/>
		<tag/>
		<url/>
	</scm>
	<properties>
		<java.version>17</java.version>
		<spring-cloud.version>2025.0.0</spring-cloud.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-actuator</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-openfeign</artifactId>
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-devtools</artifactId>
```

```
                <scope>runtime</scope>
                <optional>true</optional>
            </dependency>
            <dependency>
                <groupId>com.h2database</groupId>
                <artifactId>h2</artifactId>
                <scope>runtime</scope>
            </dependency>
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-test</artifactId>
                <scope>test</scope>
            </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>
```

PaymentServiceApplication.java

```
package com.example.paymentservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
@SpringBootApplication
@EnableDiscoveryClient
public class PaymentServiceApplication {
   public static void main(String[] args) {
      SpringApplication.run(PaymentServiceApplication.class, args);
   }
}
```

PaymentController.java

```
package com.example.paymentservice.controller;

import com.example.paymentservice.dto.PaymentRequest;
import com.example.paymentservice.dto.PaymentResponse;
import com.example.paymentservice.entity.Payment;
import com.example.paymentservice.repository.PaymentRepository;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.UUID;

@RestController
@RequestMapping("/payments")
public class PaymentController {

    @Autowired
    private PaymentRepository paymentRepository;

    @PostMapping
    public PaymentResponse processPayment(@RequestBody PaymentRequest request) {
        Payment payment = new Payment();
        payment.setId(UUID.randomUUID().toString());
        payment.setOrderId(request.getOrderId());
        payment.setAmount(request.getAmount());
        payment.setPaymentMethod(request.getPaymentMethod());

        String status = request.getAmount() > 0 ? "SUCCESS" : "FAILED";
        payment.setStatus(status);

        paymentRepository.save(payment);

        PaymentResponse response = new PaymentResponse();
        response.setStatus(status);
        return response;
    }
}
```

PaymentRequest.java

```
package com.example.paymentservice.dto;

public class PaymentRequest {
    private String orderId;
    private double amount;
    private String paymentMethod;

    public String getOrderId() { return orderId; }
    public void setOrderId(String orderId) { this.orderId = orderId; }
```

```
    public double getAmount() { return amount; }
    public void setAmount(double amount) { this.amount = amount; }

    public String getPaymentMethod() { return paymentMethod; }
    public void setPaymentMethod(String paymentMethod) { this.paymentMethod =
paymentMethod; }
}
```

PaymentResponse.java

```
package com.example.paymentservice.dto;

public class PaymentResponse {
    private String status;

    public String getStatus() { return status; }
    public void setStatus(String status) { this.status = status; }
}
```

Payment.java

```
package com.example.paymentservice.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;

@Entity
public class Payment {
    @Id
    private String id;
    private String orderId;
    private double amount;
    private String paymentMethod;
    private String status;

    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getOrderId() { return orderId; }
    public void setOrderId(String orderId) { this.orderId = orderId; }

    public double getAmount() { return amount; }
    public void setAmount(double amount) { this.amount = amount; }

    public String getPaymentMethod() { return paymentMethod; }
    public void setPaymentMethod(String paymentMethod) { this.paymentMethod =
paymentMethod; }

    public String getStatus() { return status; }
    public void setStatus(String status) { this.status = status; }
}
```

PaymentRepository.java

```
package com.example.paymentservice.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.example.paymentservice.entity.Payment;

@Repository
public interface PaymentRepository extends JpaRepository<Payment, String> {
}
```

Application.properties

```
spring.application.name=payment-service
server.port=8083
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
management.endpoints.web.exposure.include=*
management.endpoint.health.show-details=always
info.app.name=Payment Service
info.app.version=1.0.0
```

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>3.5.4</version>
                <relativePath/> <!-- lookup parent from repository -->
        </parent>
        <groupId>com.example</groupId>
        <artifactId>paymentservice</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>paymentservice</name>
        <description>Payment Service for E-commerce</description>
        <url/>
        <licenses>
                <license/>
        </licenses>
        <developers>
                <developer/>
        </developers>
        <scm>
                <connection/>
```

```
        <developerConnection/>
        <tag/>
        <url/>
    </scm>
    <properties>
        <java.version>17</java.version>
        <spring-cloud.version>2025.0.0</spring-cloud.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </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>
```