# Smart Order Management System - Microservices Project

## 📘 Project Overview

The Smart Order Management System is a microservices-based application built using Spring Boot, Spring Data JPA, PostgreSQL, Spring Cloud Gateway, and Eureka. This project demonstrates key microservices concepts such as authentication, service-to-service communication, API Gateway, and database per service.

## 🧩 Microservices Overview

### User Service

* • Handles user registration and authentication
* • Manages user roles (USER, ADMIN)
* • JWT authentication

### Product Service

* • Manages product catalog
* • CRUD operations for products

### Order Service

* • Handles order placement and retrieval
* • Communicates with User and Product Services

### Payment Service

* • Processes mock payments for orders
* • Simulates payment gateway integration

### Notification Service (Optional)

* • Sends notifications when orders are placed or payments succeed
* • Can use RabbitMQ for async communication

## 🚪 API Gateway

Spring Cloud Gateway is used to route requests to appropriate services. It handles authentication, security, and rate limiting if configured.

## 🧰 Tools & Technologies

Spring Boot, Spring Data JPA, PostgreSQL, Spring Cloud Gateway, Spring Security, JWT, Eureka Server, FeignClient, Lombok, MapStruct/ModelMapper, Swagger, RabbitMQ (Optional)

## 📦 Entity Structures

### User

* - id
* - name
* - email
* - password
* - role

### Product

* - id
* - name
* - description
* - price
* - quantity

### Order

* - id
* - userId
* - List<OrderItem>
* - totalAmount
* - status

### OrderItem

* - id
* - productId
* - quantity
* - price

### Payment

* - id
* - orderId
* - amount
* - status

### Notification (Optional)

* - id
* - userId
* - message
* - type
* - status

## 🧠 Architecture Overview

Client → API Gateway → Eureka Server → Microservices  
- User Service  
- Product Service  
- Order Service  
- Payment Service  
- Notification Service (optional)  
  
Each service has its own PostgreSQL schema or database. Services communicate via REST or FeignClients.

## 📘 Learning Path

1. 1. Start with User Service - JWT login/registration
2. 2. Create Product Service with CRUD operations
3. 3. Add Order Service and connect with Product & User
4. 4. Setup API Gateway and route traffic
5. 5. Secure Gateway using JWT
6. 6. Integrate Payment Service
7. 7. (Optional) Add Notification and Distributed Logging