**Yêu cầu User Service từ Khách Hàng**

**Dự án:** Shopee Clone Mini  
**Chức năng:** Quản lý người dùng, phân quyền theo vai trò (role-based access control - RBAC)

# Vai trò người dùng (Roles)

Hệ thống sẽ có **4 vai trò chính**, mỗi vai trò có quyền hạn khác nhau:

| **Role** | **Mô tả** |
| --- | --- |
| **User** (Người mua) | Người dùng thông thường, có thể mua hàng |
| **Seller** (Người bán) | Đăng sản phẩm, quản lý đơn hàng của shop |
| **Admin** | Quản trị viên hệ thống, có quyền kiểm soát toàn bộ hệ thống |
| **Super Admin** | Quản lý tất cả tài khoản, chỉnh sửa dữ liệu quan trọng |

# Quyền hạn theo từng Role

Mỗi vai trò sẽ có các quyền hạn cụ thể:

* 1. **User (Người mua hàng)**

**🔹 Đăng ký / đăng nhập**

🔹 Xem danh sách sản phẩm

🔹 Thêm sản phẩm vào giỏ hàng

🔹 Đặt hàng (checkout)

🔹 Xem lịch sử mua hàng

🔹 Viết đánh giá sản phẩm

🔹 Cập nhật thông tin cá nhân

❌ Không thể đăng sản phẩm

❌ Không thể chỉnh sửa đơn hàng của người khác

* 1. **Seller (Người bán hàng)**

✅ Có toàn bộ quyền của **User**

🔹 Đăng sản phẩm lên hệ thống

🔹 Quản lý đơn hàng của shop mình

🔹 Cập nhật số lượng, giá sản phẩm của shop

🔹 Xem thống kê doanh thu cá nhân

❌ Không thể truy cập thông tin của seller khác

❌ Không thể quản lý user

* 1. **Admin (Quản trị viên)**

✅ Có toàn bộ quyền của **Seller**

🔹 Duyệt / Xóa sản phẩm vi phạm

🔹 Quản lý tài khoản người dùng (khóa / mở khóa tài khoản)

🔹 Kiểm tra báo cáo khiếu nại của người dùng

🔹 Theo dõi giao dịch, doanh thu toàn hệ thống

❌ Không thể chỉnh sửa thông tin Super Admin

❌ Không thể xóa dữ liệu nhạy cảm

* 1. **Super Admin (Cao nhất)**

✅ Có toàn bộ quyền của **Admin**

🔹 Chỉnh sửa thông tin admin khác

🔹 Reset hệ thống nếu có lỗi nghiêm trọng

🔹 Truy cập database hệ thống

❌ Không có quyền mua hàng hoặc bán hàng (tài khoản chỉ để quản lý)

# API Endpoint Cho User Service

Các API chính cần thiết cho User Service:

* 1. **Xác thực (Auth)**

| **Method** | **Endpoint** | **Mô tả** | **Role** |
| --- | --- | --- | --- |
| POST | /api/auth/register | Đăng ký tài khoản | Public |
| POST | /api/auth/login | Đăng nhập và nhận JWT | Public |
| POST | /api/auth/logout | Đăng xuất | User, Seller, Admin |
| POST | /api/auth/refresh | Refresh JWT Token | All |

* 1. **Quản lý tài khoản**

| **Method** | **Endpoint** | **Mô tả** | **Role** |
| --- | --- | --- | --- |
| GET | /api/user/profile | Lấy thông tin cá nhân | User, Seller |
| PUT | /api/user/profile | Cập nhật thông tin cá nhân | User, Seller |
| GET | /api/user/orders | Lịch sử đơn hàng | User |
| GET | /api/user/addresses | Danh sách địa chỉ nhận hàng | User |
| POST | /api/user/addresses | Thêm địa chỉ nhận hàng | User |
| DELETE | /api/user/addresses/:id | Xóa địa chỉ nhận hàng | User |

* 1. **Quản lý Seller**

| **Method** | **Endpoint** | **Mô tả** | **Role** |
| --- | --- | --- | --- |
| POST | /api/seller/register | Đăng ký trở thành Seller | User |
| GET | /api/seller/orders | Danh sách đơn hàng của shop | Seller |
| PUT | /api/seller/orders/:id/status | Cập nhật trạng thái đơn hàng | Seller |
| POST | /api/seller/products | Đăng sản phẩm mới | Seller |
| PUT | /api/seller/products/:id | Chỉnh sửa sản phẩm | Seller |
| DELETE | /api/seller/products/:id | Xóa sản phẩm | Seller |

* 1. **Quản lý Admin**

| **Method** | **Endpoint** | **Mô tả** | **Role** |
| --- | --- | --- | --- |
| GET | /api/admin/users | Danh sách tất cả user | Admin |
| PUT | /api/admin/users/:id/ban | Khóa tài khoản user | Admin |
| GET | /api/admin/reports | Xem danh sách báo cáo vi phạm | Admin |
| DELETE | /api/admin/reports/:id | Xóa báo cáo đã xử lý | Admin |
| GET | /api/admin/sales | Thống kê doanh thu hệ thống | Admin |

* 1. **Quản lý Super Admin**

| **Method** | **Endpoint** | **Mô tả** | **Role** |
| --- | --- | --- | --- |
| GET | /api/super-admin/admins | Danh sách admin | Super Admin |
| POST | /api/super-admin/admins | Thêm admin mới | Super Admin |
| DELETE | /api/super-admin/admins/:id | Xóa admin | Super Admin |
| PUT | /api/super-admin/system-reset | Reset hệ thống | Super Admin |

# Các yêu cầu bảo mật

✅ **JWT Authentication** để bảo vệ API.

✅ **Role-based Authorization** để kiểm soát quyền truy cập.

✅ **Rate Limiting** để chống spam request (tối đa 5 lần đăng nhập sai/phút).

✅ **CORS Policy** để ngăn truy cập trái phép từ frontend khác.

✅ **Hash mật khẩu với BCrypt** trước khi lưu vào database.

✅ **Token Expiration & Refresh Token** để tăng cường bảo mật.

# **Lộ trình triển khai**

Bạn có thể làm theo thứ tự này:  
1. Xây dựng hệ thống **đăng ký, đăng nhập với JWT**.  
2️. Tạo database **MySQL** với bảng users, roles, permissions.  
3️. Viết middleware kiểm tra **role-based authorization**.  
4️. Xây dựng **API quản lý tài khoản User, Seller**.  
5️. Xây dựng **API quản lý Admin & Super Admin**.  
6️. Viết test với **JUnit (Spring Boot) hoặc Jest (NodeJS)**.  
7️. Tối ưu bảo mật & kiểm tra performance.

# Lên kiến trúc hệ thống

Trước khi code, bạn nên vẽ sơ đồ tổng quan về **microservices** mà bạn sẽ xây dựng.

**Các microservices chính:**

* **User Service** → Quản lý tài khoản, xác thực người dùng.
* **Shop Service** → Quản lý gian hàng của seller, danh sách sản phẩm
* **Product Service** → Quản lý sản phẩm, danh mục.
* **Order Service** → Xử lý đơn hàng.
* **Cart Service** → Quản lý giỏ hàng.
* **Payment Service** → Xử lý thanh toán.
* **Review Service** → Đánh giá sản phẩm.
* **Notification Service** → Gửi thông báo khi có đơn hàng mới.
* **Search Service** → Tìm kiếm sản phẩm bằng Elasticsearch.

# Chọn công nghệ phù hợp

Bạn có thể chọn ngôn ngữ, framework và database như sau:

**✔ Backend**

* **Spring Boot (Java)** (ổn định, mạnh về microservices).
* **NestJS (Node.js)** (dễ phát triển nhanh).
* **Golang** (hiệu suất cao, nhưng cần hiểu concurrency).

→ Nếu bạn quen với **Spring Boot**, thì đây là lựa chọn tốt nhất.

**✔ Database**

* **MySQL** (quản lý dữ liệu có quan hệ: User, Shop, Order, Payment).
* **MongoDB** (lưu dữ liệu linh hoạt như Product, Review).
* **Redis** (cache giỏ hàng, session user).
* **Elasticsearch** (tìm kiếm sản phẩm nhanh hơn).

# Cài đặt môi trường làm việc

Bạn cần chuẩn bị các công cụ sau:

**✔ Công cụ phát triển**

* **IntelliJ IDEA** (nếu dùng Java) hoặc **VS Code** (cho NestJS, Golang).
* **Postman** (test API).
* **Docker** (chạy database và microservices).
* **Kafka / RabbitMQ** (xử lý message queue).

**✔ Database & Caching**

* **PostgreSQL** (docker run postgres).
* **MongoDB** (docker run mongo).
* **Redis** (docker run redis).

**✔ API Gateway**

* **Spring Cloud Gateway** (cho Spring Boot).
* **Kong API Gateway** (nếu muốn performance cao).

# Xây dựng từng microservice

Bạn có thể bắt đầu từ những service quan trọng nhất:

* 1. **User Service (Xác thực, JWT)**
* Đăng ký, đăng nhập (email, password).
* OAuth2 (Google, Facebook login).
* Lưu thông tin người dùng vào PostgreSQL.
* JWT authentication & refresh token.
  1. **Product Service**
* CRUD sản phẩm, danh mục.
* Lưu thông tin sản phẩm vào MongoDB.
* Dùng Elasticsearch để tìm kiếm sản phẩm.
  1. **Order Service**
* Người dùng đặt hàng.
* Lưu đơn hàng vào MySQL.
* Gửi event qua Kafka/RabbitMQ để xử lý thanh toán.

**Order Service (Quản lý đơn hàng - MySQL)**

📌 **Mục tiêu:**

* Xử lý đơn hàng, trạng thái đơn hàng, xác nhận & hủy đơn.
* Kết nối với **Cart Service** để lấy sản phẩm trong giỏ hàng.
* Lưu thông tin đơn hàng trong **MySQL**.

📌 **API Endpoints**

| **Method** | **Endpoint** | **Mô tả** | **Role** |
| --- | --- | --- | --- |
| POST | /api/orders | Tạo đơn hàng mới từ giỏ hàng | User |
| GET | /api/orders | Lấy danh sách đơn hàng của user | User |
| GET | /api/orders/{id} | Lấy chi tiết đơn hàng | User |
| PUT | /api/orders/{id}/status | Cập nhật trạng thái đơn hàng (đã giao, đã hủy) | Seller |
| DELETE | /api/orders/{id} | Hủy đơn hàng | User |

📌 **Database Tables (MySQL)**

* orders: chứa thông tin đơn hàng.
* order\_items: danh sách sản phẩm trong đơn hàng.
  1. **Payment Service**
* Xử lý thanh toán với VNPay, MoMo, Stripe.
* Kiểm tra trạng thái thanh toán (thành công/thất bại).
* Gửi thông báo qua Notification Service.

📌 **Mục tiêu:**

* Xử lý thanh toán đơn hàng (VNPay, Momo, Stripe, PayPal,...)
* Lưu lịch sử thanh toán vào MySQL.
* Tích hợp với **Order Service** để xác nhận thanh toán.

📌 **API Endpoints**

| **Method** | **Endpoint** | **Mô tả** | **Role** |
| --- | --- | --- | --- |
| POST | /api/payments | Thanh toán đơn hàng | User |
| GET | /api/payments/{orderId} | Xem trạng thái thanh toán | User |
| POST | /api/payments/webhook | Nhận callback từ cổng thanh toán | Hệ thống |

📌 **Database Tables (MySQL)**

* payments: chứa lịch sử thanh toán.
* transactions: chi tiết giao dịch.
  1. **Cart Service**
* Lưu giỏ hàng tạm thời vào Redis (key: user\_id).
* Khi user thanh toán, dữ liệu giỏ hàng gửi sang Order Service.

📌 **Mục tiêu:**

* Lưu tạm thời giỏ hàng của user.
* Hỗ trợ thêm/xóa/sửa số lượng sản phẩm trong giỏ.
* Kết nối với **Order Service** khi đặt hàng.

📌 **API Endpoints**

| **Method** | **Endpoint** | **Mô tả** | **Role** |
| --- | --- | --- | --- |
| POST | /api/cart | Thêm sản phẩm vào giỏ | User |
| GET | /api/cart | Xem giỏ hàng | User |
| PUT | /api/cart/{id} | Cập nhật số lượng sản phẩm | User |
| DELETE | /api/cart/{id} | Xóa sản phẩm khỏi giỏ | User |
| DELETE | /api/cart/clear | Xóa toàn bộ giỏ hàng | User |

📌 **Database (Redis)**

* **Key:** cart:userId
* **Value:** Danh sách sản phẩm, số lượng, giá.
  1. **Review Service (Đánh giá sản phẩm - MongoDB)**

📌 **Mục tiêu:**

* Cho phép user đánh giá sản phẩm.
* Lưu trữ đánh giá trong **MongoDB**.
* Cập nhật điểm rating cho **Product Service**.

📌 **API Endpoints**

| **Method** | **Endpoint** | **Mô tả** | **Role** |
| --- | --- | --- | --- |
| POST | /api/reviews | Viết đánh giá sản phẩm | User |
| GET | /api/reviews/{productId} | Xem đánh giá của sản phẩm | User |
| DELETE | /api/reviews/{id} | Xóa đánh giá | User |

📌 **Database (MongoDB)**

* **Collection:** reviews
* **Schema:** { userId, productId, rating, comment, createdAt }
  1. **Notification Service (Gửi thông báo - Kafka, Firebase)**

📌 **Mục tiêu:**

Gửi thông báo khi có đơn hàng mới, thanh toán thành công,...

Hỗ trợ **Kafka** hoặc **Firebase Cloud Messaging (FCM)**.

📌 **API Endpoints**

| **Method** | * 1. **Endpoint** | * 1. **Mô tả** | * 1. **Role** |
| --- | --- | --- | --- |
| GET | * 1. /api/notifications | * 1. Lấy danh sách thông báo | * 1. User |
| POST | * 1. /api/notifications/send | * 1. Gửi thông báo thủ công | * 1. Admin |

📌 **Công nghệ:**

Brevo API Key: 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

**Kafka** → Gửi thông báo real-time.

**FCM** → Đẩy thông báo lên mobile app.

* 1. **Search Service (Tìm kiếm sản phẩm - Elasticsearch)**

📌 **Mục tiêu:**

* Cung cấp tìm kiếm sản phẩm nhanh chóng.
* Tích hợp với **Product Service** để cập nhật dữ liệu.

📌 **API Endpoints**

| **Method** | **Endpoint** | **Mô tả** | **Role** |
| --- | --- | --- | --- |
| GET | /api/search?query=... | Tìm kiếm sản phẩm | User |

📌 **Database (Elasticsearch)**

* **Index:** products
* **Schema:** { id, name, description, price, category }

# Sử dụng message queue để xử lý sự kiện

* Khi có đơn hàng mới, **Order Service** gửi event **"Order Created"** đến Kafka.
* **Payment Service** nhận event, kiểm tra thanh toán.
* Nếu thành công, gửi event **"Payment Completed"** để cập nhật trạng thái đơn hàng.

Dùng **Kafka hoặc RabbitMQ** để tránh bị nghẽn hệ thống.

# Bảo mật API

* **Spring Security / OAuth2** cho xác thực.
* **Rate Limiting** trên API Gateway (ngăn spam request).
* **CORS, Helmet.js** để bảo vệ API khỏi tấn công XSS, CSRF.

# Viết test & kiểm tra performance

* Dùng **JUnit / Jest** để viết test case cho API.
* **JMeter / k6** để kiểm tra tải hệ thống.
* **Redis Cache** để tối ưu tốc độ query.

# CI/CD & Deployment

Bạn có thể:

* Dùng **Docker Compose** để chạy local.
* Deploy lên **AWS, GCP, DigitalOcean**.
* Dùng **Kubernetes (K8s)** nếu muốn mở rộng.
* Monitor với **Prometheus + Grafana**.

# Lộ trình triển khai

1. **Cấu hình môi trường** (cài đặt Docker, database).
2. **Viết API cho User Service (đăng nhập, JWT)**.
3. **Xây dựng Product Service (MongoDB, Elasticsearch)**.
4. **Tạo Order Service và Payment Service (Kafka event-driven)**.
5. **Thêm Cart Service (Redis)**.
6. **Tích hợp API Gateway và bảo mật hệ thống**.
7. **Tối ưu hiệu suất và viết test**.