**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)

**1. 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 |

**2. 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

**(2) 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

**(3) 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

**(4) 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)

**3. 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 |

**(2) 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 |

**(3) 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 |

**(4) 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 |

**(5) 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 |

**4. 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.

**5. 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.

**1. 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.

**2. 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).

**3. 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).

**4. 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.

**(2) Product Service**

* CRUD sản phẩm, danh mục.
* Lưu thông tin sản phẩm vào MongoDB.
* Dùng Elasticsearch để tìm kiếm sản phẩm.

**(3) Order Service**

* Người dùng đặt hàng.
* Lưu đơn hàng vào PostgreSQL.
* Gửi event qua Kafka/RabbitMQ để xử lý thanh toán.

**(4) 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.

**(5) Cart Service**

* Lưu giỏ hàng tạm thời vào Redis (key: user\_id).
* Khi user thanh toán, dữ liệu giỏ hàng gửi sang Order Service.

**5. 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.

**6. 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.

**7. 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.

**8. 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**.