**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

🙢 🕮 🙠



**BÁO CÁO CUỐI KỲ**

**MÔN HỌC:** **LẬP TRÌNH WEB**

XÂY DỰNG WEBSITE THƯƠNG MẠI ĐIỆN TỬ "BÁN ĐẶC SẢN" (BANDACSAN)

|  |  |  |
| --- | --- | --- |
| **GVHD:** | ThS. Nguyễn Hữu Trung |  |
| **SVTH:** | Đặng Gia Huy | 23110101 |
|  | Cao Thọ Phú Thịnh | 21144449 |
|  |

**TP.HCM, tháng 01 năm 2026**

**MỤC LỤC**

[CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI 1](#_Toc218604904)

[1.1. Lý do chọn đề tài 1](#_Toc218604905)

[1.2. Mục tiêu đồ án 1](#_Toc218604906)

[1.3. Công nghệ và Kiến trúc hệ thống 1](#_Toc218604907)

[CHƯƠNG 2: PHÂN TÍCH & THIẾT KẾ HỆ THỐNG 1](#_Toc218604908)

[2.1. Phân tích tác nhân và Use Case 1](#_Toc218604909)

[2.2. Thiết kế Cơ sở dữ liệu (ERD) 2](#_Toc218604910)

[CHƯƠNG 3: HIỆN THỰC CÁC CHỨC NĂNG CHI TIẾT 3](#_Toc218604911)

[3.1. PHÂN HỆ CHUNG (AUTHENTICATION & SECURITY) 3](#_Toc218604912)

[3.1.1. Quy trình Xác thực (Authentication Flow) 3](#_Toc218604913)

[3.1.2. Hiện thực Code (Security Config) 5](#_Toc218604914)

[3.2. PHÂN HỆ VENDOR (NGƯỜI BÁN) 6](#_Toc218604915)

[3.2.1. Quản lý Sản phẩm (Product Management) 6](#_Toc218604916)

[3.2.2. Quản lý Đơn hàng & Doanh thu 9](#_Toc218604917)

[3.3. PHÂN HỆ CUSTOMER (NGƯỜI MUA) 9](#_Toc218604918)

[3.3.1. Hiển thị trang chủ (UserPageController) 9](#_Toc218604919)

[3.3.2. Giỏ hàng & Thanh toán (Checkout Process) 10](#_Toc218604920)

[3.3.3. Hệ thống Chat Real-time 12](#_Toc218604921)

[3.4. PHÂN HỆ ADMIN (QUẢN TRỊ VIÊN) 13](#_Toc218604922)

[3.4.1. Quản lý Người dùng (User Management) 13](#_Toc218604923)

[3.4.2. Quản lý Danh mục (Category Management) 14](#_Toc218604924)

[3.4.3. Quản lý Sản phẩm Hệ thống 14](#_Toc218604925)

[CHƯƠNG 4: KẾT LUẬN & HƯỚNG PHÁT TRIỂN 15](#_Toc218604926)

[4.1. Kết luận 15](#_Toc218604927)

[4.2. Hạn chế 15](#_Toc218604928)

[4.3. Hướng phát triển trong tương lai 15](#_Toc218604929)

# 

# **CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI**

## **1.1. Lý do chọn đề tài**

Trong kỷ nguyên số hóa, nhu cầu tiêu thụ các sản phẩm đặc sản vùng miền ngày càng tăng cao. Tuy nhiên, việc kết nối giữa người sản xuất địa phương và người tiêu dùng trên cả nước vẫn còn nhiều rào cản về địa lý và kênh phân phối.

Đề tài **"Xây dựng Website Thương mại điện tử Bán Đặc Sản"** ra đời nhằm giải quyết bài toán trên, cung cấp một nền tảng trực tuyến chuyên biệt, giúp người bán quảng bá đặc sản và người mua dễ dàng tiếp cận sản phẩm chất lượng.

## **1.2. Mục tiêu đồ án**

* Xây dựng hệ thống sàn thương mại điện tử đa người dùng (Multi-vendor).
* Đảm bảo quy trình mua sắm trọn vẹn: Tìm kiếm -> Giỏ hàng -> Đặt hàng -> Thanh toán -> Theo dõi đơn hàng.
* Cung cấp công cụ quản lý doanh thu và sản phẩm hiệu quả cho người bán.
* Tích hợp các tiện ích tương tác thời gian thực (Chat) để nâng cao trải nghiệm người dùng.

## **1.3. Công nghệ và Kiến trúc hệ thống**

Hệ thống được phát triển dựa trên mô hình MVC (Model-View-Controller) cổ điển kết hợp với kiến trúc Layered Architecture của Spring Boot.

* **Ngôn ngữ & Framework:** Java, Spring Boot 3.4.12 (Cung cấp container DI, Auto-configuration).
* **Database:** MySQL 8.0 (Lưu trữ dữ liệu quan hệ, sử dụng Hibernate/JPA để ánh xạ ORM).
* **Template Engine:** Thymeleaf (Render giao diện phía Server).
* **Frontend:** HTML5, CSS3, JavaScript (Bootstrap/Tailwind).
* **Security:** Spring Security (Quản lý xác thực và phân quyền theo Role: ADMIN, VENDOR, CUSTOMER).

# **CHƯƠNG 2: PHÂN TÍCH & THIẾT KẾ HỆ THỐNG**

## **2.1. Phân tích tác nhân và Use Case**

Hệ thống được thiết kế phục vụ 4 nhóm đối tượng chính với các quyền hạn cụ thể:

1. **Guest (Khách vãng lai):** Xem sản phẩm, tìm kiếm, đăng ký, đăng nhập.
2. **Customer (Người mua):** Quản lý giỏ hàng, đặt hàng, thanh toán, chat với người bán, xem lịch sử đơn hàng.
3. **Vendor (Người bán):** Đăng bán sản phẩm, quản lý kho, xem đơn hàng liên quan đến mình, xem báo cáo doanh thu.
4. **Admin (Quản trị viên):** Quản lý người dùng, quản lý danh mục sản phẩm, kiểm duyệt hệ thống.

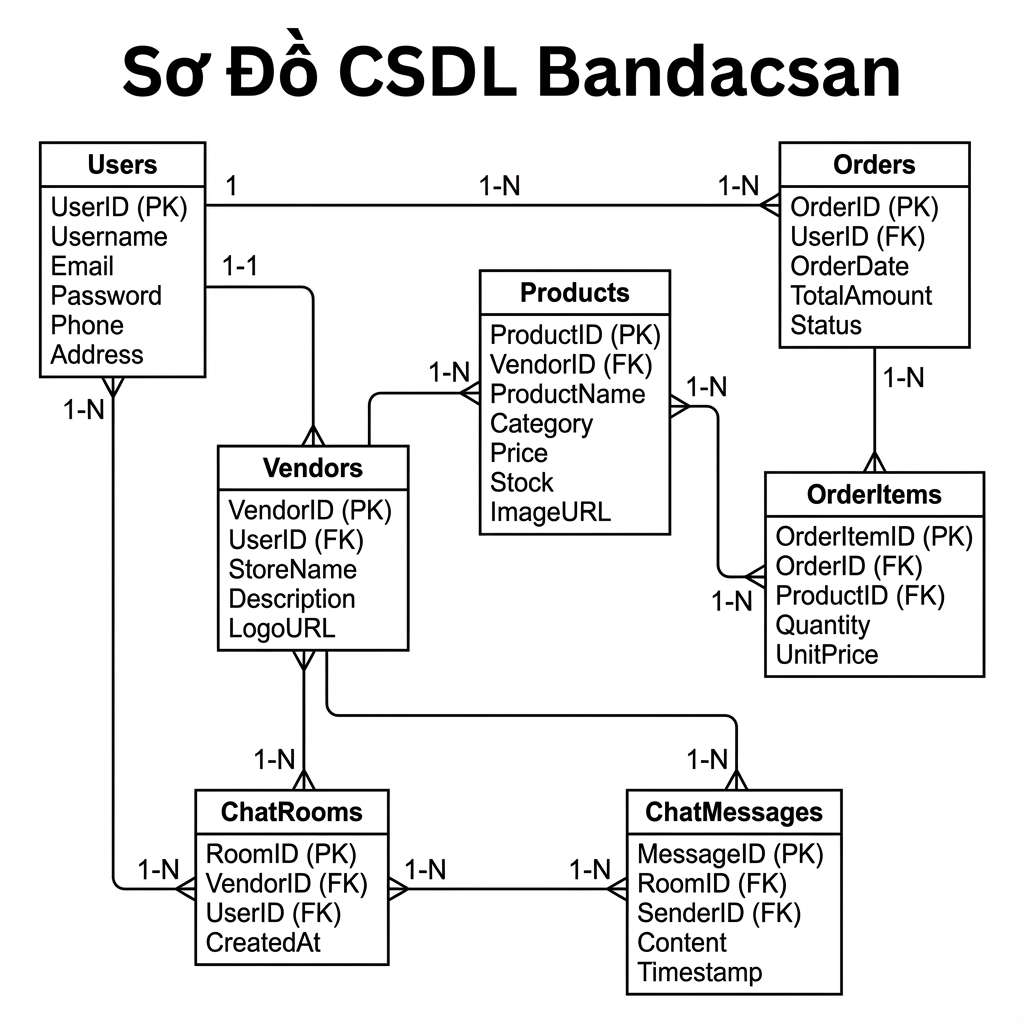
**

*Hình 2.1: Lược đồ Use Case tổng quát của hệ thống Bản Đắc Sản*

## **2.2. Thiết kế Cơ sở dữ liệu (ERD)**

Cơ sở dữ liệu bao gồm các thực thể chính:

* users: Lưu thông tin tài khoản và phân quyền.
* products: Lưu thông tin sản phẩm, liên kết với Vendor và Category.
* orders & order\_items: Lưu trữ thông tin đơn hàng và chi tiết từng món hàng.
* chat\_rooms & chat\_messages: Phục vụ chức năng nhắn tin.

**

*Hình 2.2: Sơ đồ quan hệ thực thể (ERD)*

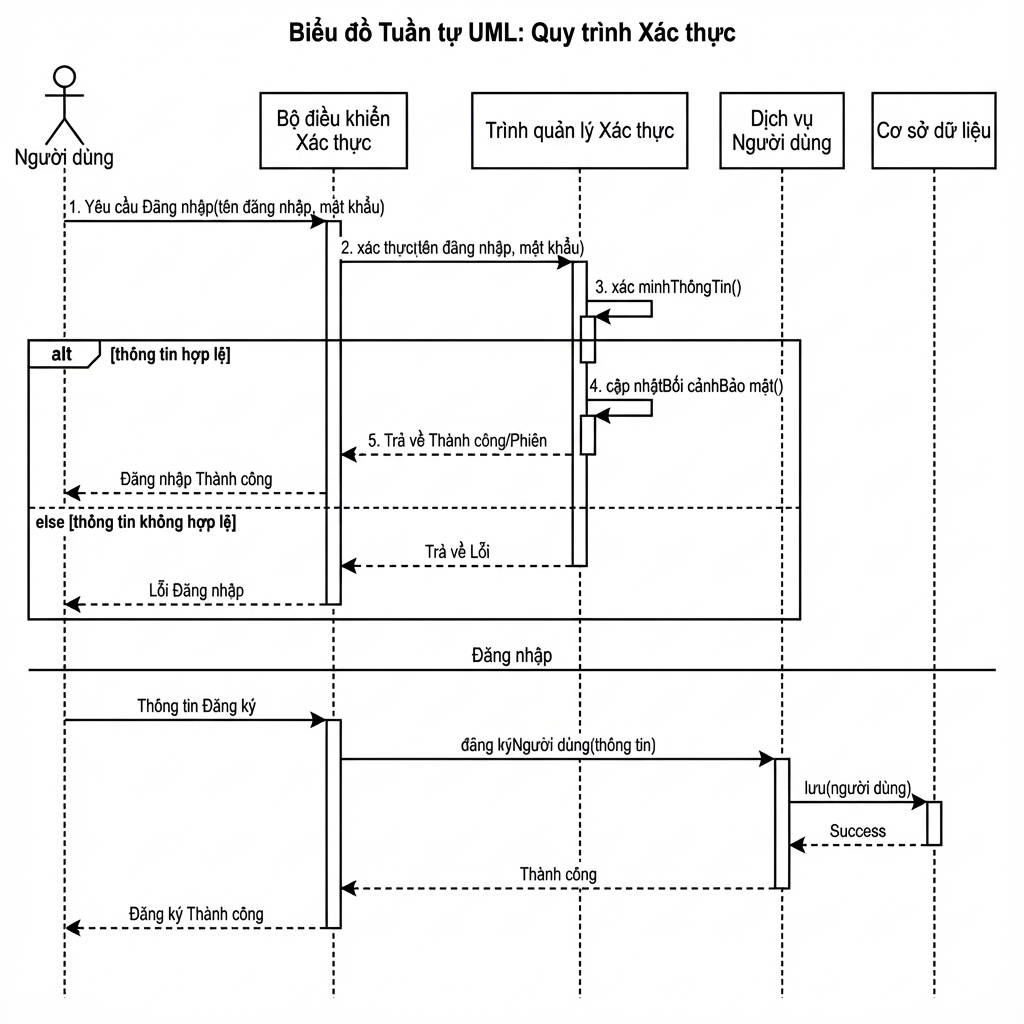
# **CHƯƠNG 3: HIỆN THỰC CÁC CHỨC NĂNG CHI TIẾT**

## **3.1. PHÂN HỆ CHUNG (AUTHENTICATION & SECURITY)**

Đây là phân hệ nền tảng, đảm bảo tính bảo mật cho ứng dụng.

### **3.1.1. Quy trình Xác thực (Authentication Flow)**

Khi người dùng đăng nhập, hệ thống sẽ xác thực qua DaoAuthenticationProvider. Mật khẩu được mã hóa bằng thuật toán BCrypt trước khi lưu xuống DB.

**

Đoạn mã

sequenceDiagram

actor User as Người Dùng

participant Auth as AuthController

participant Manager as AuthManager

participant Service as UserService

participant DB as Database

User->>Auth: 1. Đăng nhập (username, password)

Auth->>Manager: 2. authenticate(user, pass)

Manager->>Service: 3. verifyCredentials()

Service->>DB: Truy vấn User

DB-->>Service: Thông tin User (Hashed Pass)

alt Thông tin hợp lệ

Manager-->>Auth: Authentication (Success)

Auth-->>User: Cấp Session & Redirect Home

else Sai thông tin

Manager-->>Auth: Ném lỗi BadCredentials

Auth-->>User: Hiển thị lỗi đăng nhập

end

### **3.1.2. Hiện thực Code (Security Config)**

Cấu hình SecurityFilterChain để phân quyền truy cập dựa trên URL pattern:

Java

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf(AbstractHttpConfigurer::disable)

.authorizeHttpRequests(auth -> auth

.requestMatchers("/api/auth/\*\*", "/login", "/register", "/", "/home", "/css/\*\*", "/js/\*\*").permitAll()

.requestMatchers("/admin/\*\*").hasRole("ADMIN")

.requestMatchers("/vendor/\*\*").hasRole("VENDOR")

.requestMatchers("/user/\*\*").hasRole("CUSTOMER")

.anyRequest().authenticated()

)

.formLogin(form -> form

.loginPage("/login")

.defaultSuccessUrl("/home", true) // Chuyển hướng thông minh sau khi login

.permitAll()

)

.logout(logout -> logout.logoutSuccessUrl("/login?logout"));

return http.build();

}

}

**Controller xử lý Đăng ký (AuthController.java):**

Java

@RestController

@RequestMapping("/api/auth")

@RequiredArgsConstructor

public class AuthController {

private final IUserService userService;

@PostMapping("/register")

public ResponseEntity<?> register(@RequestBody UserRequestDTO dto) {

try {

// Gọi service để hash password và lưu user

return ResponseEntity.ok(userService.register(dto));

} catch (Exception e) {

return ResponseEntity.badRequest().body("Lỗi đăng ký: " + e.getMessage());

}

}

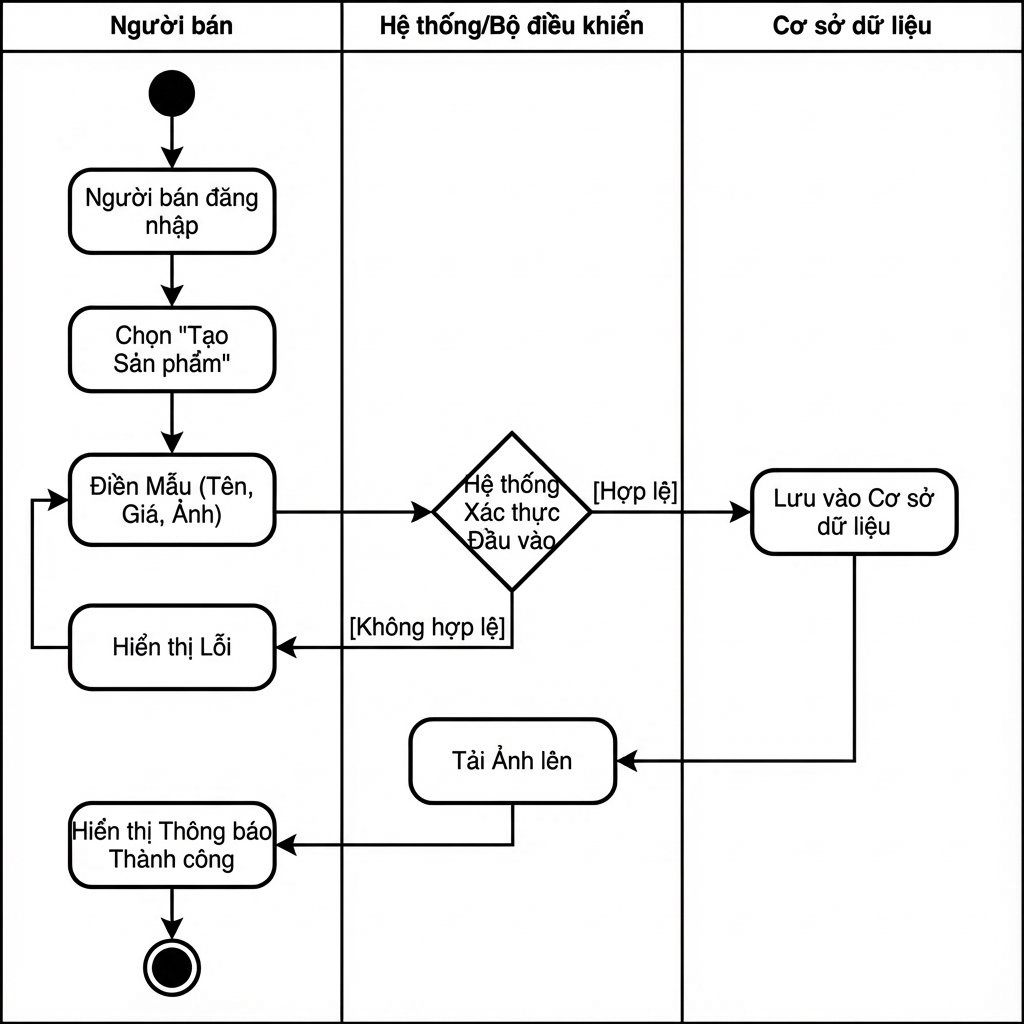
}

## **3.2. PHÂN HỆ VENDOR (NGƯỜI BÁN)**

Phân hệ này tập trung vào nghiệp vụ quản lý kho hàng và đơn hàng.

### **3.2.1. Quản lý Sản phẩm (Product Management)**

Người bán thực hiện quy trình CRUD (Thêm, Xem, Sửa, Xóa) sản phẩm. Khi tạo mới, sản phẩm sẽ tự động được gắn vendor\_id của người đang đăng nhập.

*(Mô tả bằng Lược đồ hoạt động - Activity Diagram):*

Đoạn mã

flowchart TD

Vendor([Người Bán]) -->|Đăng nhập| Start((Bắt đầu))

Start --> Create[Chọn 'Tạo Sản Phẩm']

Create --> Form[Nhập: Tên, Giá, Mô tả, Ảnh...]

Form --> Validate{Hệ thống Validate}

Validate -->|Hợp lệ| Save[Lưu vào Database]

Save --> Upload[Upload File Ảnh lên Server]

Upload --> Success[Thông báo Thành công]

Success --> End(((Kết thúc)))

Validate -->|Lỗi| Error[Hiển thị lỗi Validation]

Error --> Form

**Hiện thực Code (VendorContentController.java):**

Java

@Controller

@RequestMapping("/vendor")

public class VendorContentController {

@Autowired

private IProductService productService;

@Autowired

private IVendorRepository vendorRepository;

// Xem danh sách sản phẩm của chính Vendor đó

@GetMapping("/products")

public String products(Model model, HttpSession session) {

User user = (User) session.getAttribute("user");

var vendor = vendorRepository.findVendorByUser\_Id(user.getId()).orElseThrow();

List<Product> products = productService.getAll().stream()

.filter(p -> p.getVendorId().equals(vendor.getId()))

.collect(Collectors.toList());

model.addAttribute("products", products);

return "vendor/product/list";

}

// Xử lý tạo sản phẩm mới

@PostMapping("/products/create")

public String createProduct(ProductRequestDTO request, HttpSession session) {

User user = (User) session.getAttribute("user");

Vendor vendor = vendorRepository.findVendorByUser\_Id(user.getId()).get();

request.setVendorId(vendor.getId()); // Gán chủ sở hữu

productService.create(request, request.getImageFile());

return "redirect:/vendor/products";

}

}

### **3.2.2. Quản lý Đơn hàng & Doanh thu**

Hệ thống tính toán doanh thu bằng cách lọc các OrderItem thuộc về Vendor trong các Order có trạng thái DELIVERED (Đã giao hàng).

Java

@GetMapping("/orders")

public String orders(Model model, HttpSession session) {

User user = (User) session.getAttribute("user");

Vendor vendor = vendorRepository.findVendorByUser\_Id(user.getId()).get();

// Logic nghiệp vụ: Chỉ lấy đơn hàng có chứa sản phẩm của Vendor

List<OrderDTO> vendorOrders = orderService.getOrdersByVendorId(vendor.getId());

// Tính tổng doanh thu

double totalRevenue = vendorOrders.stream()

.filter(o -> o.getStatus() == OrderStatus.DELIVERED)

.mapToDouble(OrderDTO::getTotalAmount)

.sum();

model.addAttribute("orders", vendorOrders);

model.addAttribute("totalRevenue", totalRevenue);

return "vendor/orders";

}

## **3.3. PHÂN HỆ CUSTOMER (NGƯỜI MUA)**

Đây là phân hệ có tần suất sử dụng cao nhất, yêu cầu giao diện thân thiện và xử lý nhanh.

### **3.3.1. Hiển thị trang chủ (UserPageController)**

Sử dụng Pageable để phân trang sản phẩm, tối ưu hiệu suất tải trang.

Java

@GetMapping({"/user", "/user/home"})

public String home(Model model) {

// Lấy 8 sản phẩm mới nhất

Pageable pageable = PageRequest.of(0, 8, Sort.by("id").descending());

Page<ProductResponseDTO> products = productService.getAll(pageable);

model.addAttribute("latestProducts", products.getContent());

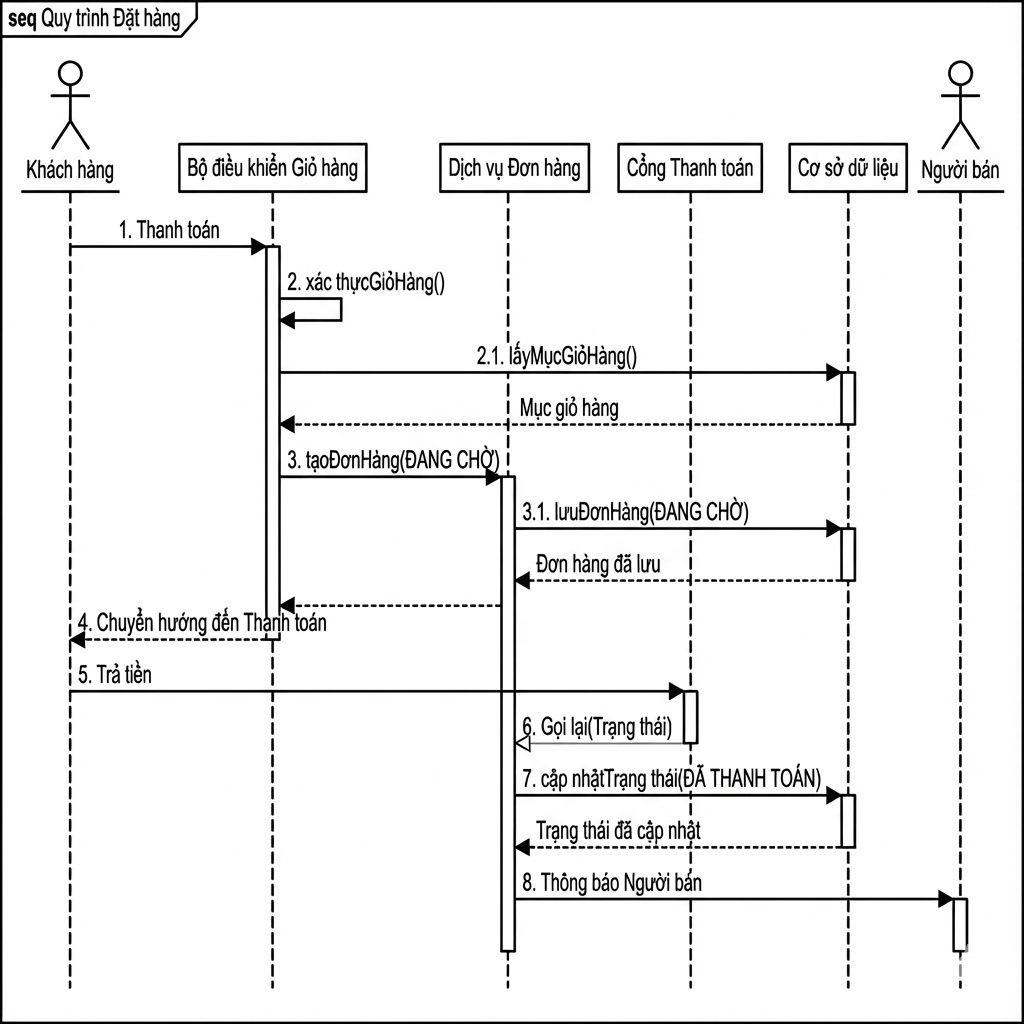
model.addAttribute("categories", categoryService.getAll());

return "user/home";

}

### **3.3.2. Giỏ hàng & Thanh toán (Checkout Process)**

Quy trình thanh toán hỗ trợ cả phương thức COD (Cash on Delivery) và ví điện tử (Momo - Giả lập hoặc tích hợp API).

*(Mô tả bằng Lược đồ tuần tự - Sequence Diagram):*

Đoạn mã

sequenceDiagram

actor Customer as Khách Hàng

participant Cart as CartController

participant Order as OrderService

participant Gateway as Cổng Thanh Toán (Momo)

participant DB as Database

actor Vendor as Người Bán

Customer->>Cart: 1. Checkout (Xác nhận giỏ hàng)

Cart->>Order: 2. createOrder (Status: PENDING)

Order->>DB: Lưu đơn hàng & Chi tiết

Order-->>Cart: Trả về Order ID

alt Thanh toán Online (Momo)

Cart->>Gateway: 3. Lấy URL thanh toán

Cart-->>Customer: 4. Redirect sang Momo

Customer->>Gateway: 5. Thanh toán thành công

Gateway->>Order: 6. Callback/IPN

Order->>DB: 7. Update Status = PAID

else Thanh toán COD

Cart-->>Customer: 4. Thông báo đặt hàng thành công

end

Order->>Vendor: 8. Notification (Có đơn mới)

**Code xử lý thanh toán (CheckoutController.java):**

Java

@PostMapping(consumes = MediaType.APPLICATION\_JSON\_VALUE)

public ResponseEntity<?> processCheckout(@RequestBody CheckoutRequestDTO request, HttpSession session) {

User user = (User) session.getAttribute("user");

// Bước 1: Tạo đơn hàng ở trạng thái PENDING

OrderDTO orderDTO = orderService.createOrder(user.getId(), request);

// Bước 2: Kiểm tra phương thức thanh toán

if ("MOMO".equalsIgnoreCase(request.getPaymentMethod())) {

// Tích hợp logic lấy Payment URL từ Momo Service

String payUrl = momoService.getMomoPayUrl(orderDTO.getId());

return ResponseEntity.ok(Map.of("payUrl", payUrl));

}

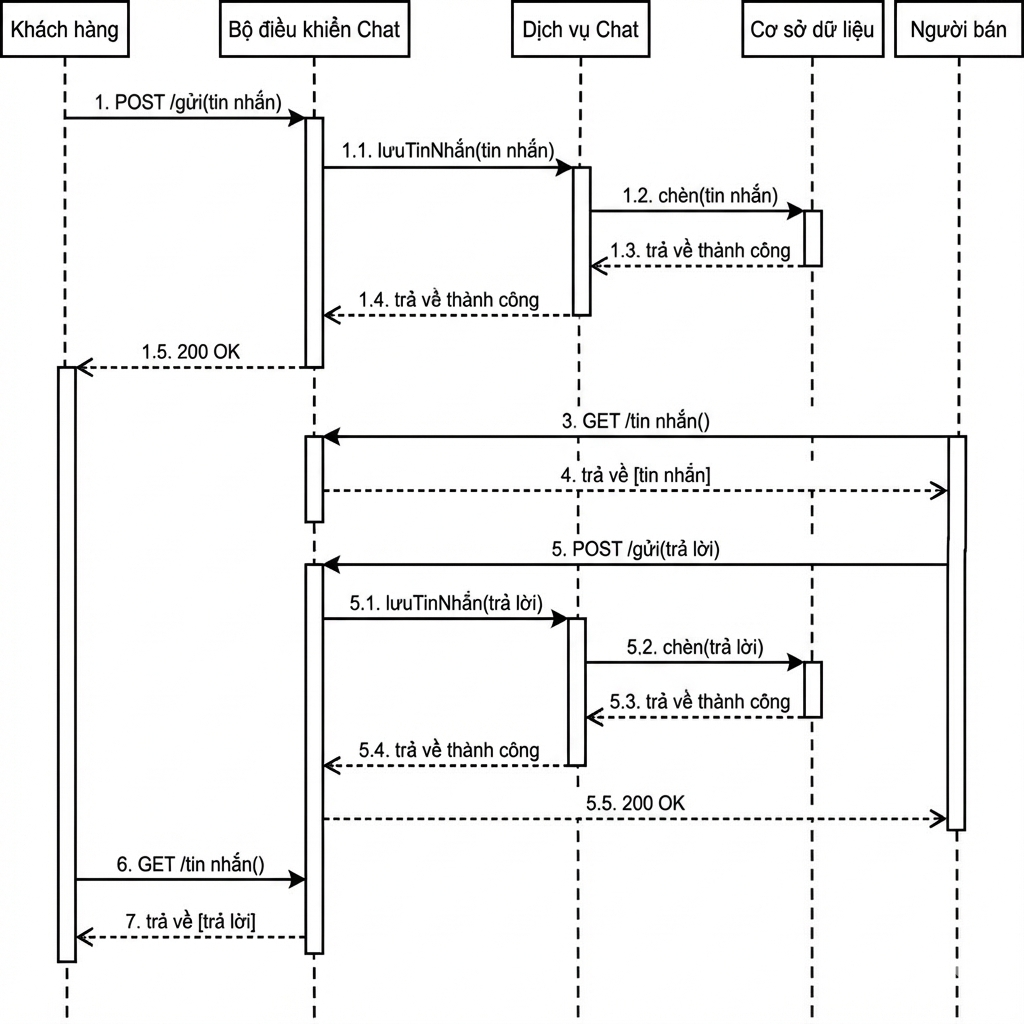
// Bước 3: Trả về thành công cho COD

return ResponseEntity.ok(orderDTO);

}

### **3.3.3. Hệ thống Chat Real-time**

Sử dụng kỹ thuật **Short Polling** (Client gửi request liên tục mỗi 2-3 giây) để cập nhật tin nhắn mới từ phía đối tác.



Java

@Controller

@RequestMapping("/user/chat")

public class ChatController {

// Gửi tin nhắn

@PostMapping("/send")

@ResponseBody

public ChatMessageDTO sendMessage(@RequestParam Long roomId, @RequestParam String message, HttpSession session) {

User user = (User) session.getAttribute("user");

return chatService.sendMessage(roomId, user.getId(), message);

}

// Polling tin nhắn mới

@GetMapping("/room/{roomId}/messages")

@ResponseBody

public List<ChatMessageDTO> getRoomMessages(@PathVariable Long roomId) {

return chatService.getRoomMessages(roomId);

}

}

## **3.4. PHÂN HỆ ADMIN (QUẢN TRỊ VIÊN)**

Admin đóng vai trò giám sát và vận hành toàn bộ hệ thống.

### **3.4.1. Quản lý Người dùng (User Management)**

Admin có quyền tạo tài khoản với vai trò cụ thể và xóa tài khoản vi phạm.

Java

@Controller

@RequestMapping("/admin/users")

public class AdminUserPageController {

// Tạo user mới (Set Role trực tiếp: ADMIN, VENDOR, CUSTOMER)

@PostMapping("/create")

public String createSubmit(HttpServletRequest request, RedirectAttributes redirectAttributes) {

UserRequestDTO dto = new UserRequestDTO();

dto.setUsername(request.getParameter("username"));

dto.setPassword(request.getParameter("password"));

dto.setRole(request.getParameter("role"));

userService.register(dto);

redirectAttributes.addFlashAttribute("successMessage", "Tạo người dùng thành công!");

return "redirect:/admin/users";

}

// Xóa user

@PostMapping("/delete")

public String deleteUser(@RequestParam Long id) {

userService.deleteUser(id);

return "redirect:/admin/users";

}

}

### **3.4.2. Quản lý Danh mục (Category Management)**

Admin xây dựng cây danh mục sản phẩm (Ví dụ: Đặc sản miền Bắc, Đồ khô, Rượu...) để Vendor có thể gán sản phẩm vào đúng loại.

Java

@Controller

@RequestMapping("/admin/categories")

public class AdminCategoryPageController {

@PostMapping("/create")

public String createSubmit(CategoryRequestDTO dto, RedirectAttributes redirectAttributes) {

categoryService.create(dto);

return "redirect:/admin/categories";

}

@PostMapping("/delete")

public String deleteCategory(@RequestParam Long id) {

categoryService.delete(id);

return "redirect:/admin/categories";

}

}

### **3.4.3. Quản lý Sản phẩm Hệ thống**

Admin có quyền xem danh sách tất cả sản phẩm của mọi Vendor và can thiệp (xóa/khóa) nếu sản phẩm vi phạm chính sách của sàn.

Java

@Controller

@RequestMapping("/admin/products")

public class ProductPageController {

// View danh sách toàn bộ sản phẩm

@GetMapping

public String listPage(Model model) {

model.addAttribute("products", productService.getAll());

return "admin/product/list";

}

// Chức năng Edit/Delete dành cho Admin...

}

# **CHƯƠNG 4: KẾT LUẬN & HƯỚNG PHÁT TRIỂN**

## **4.1. Kết luận**

Đồ án "Bán Đặc Sản" đã hoàn thành các mục tiêu đề ra ban đầu:

1. **Kiến trúc:** Xây dựng thành công hệ thống trên nền tảng Spring Boot ổn định, phân chia module rõ ràng.
2. **Chức năng:** Hoàn thiện các luồng nghiệp vụ cốt lõi: Đăng ký/Đăng nhập, Quản lý sản phẩm (Vendor), Mua hàng (Customer), Quản trị (Admin).
3. **Trải nghiệm:** Giao diện Thymeleaf thân thiện, tích hợp Chat giúp tăng tương tác.

## **4.2. Hạn chế**

* Chức năng Chat sử dụng Polling gây tốn tài nguyên server nếu lượng user lớn.
* Chưa tích hợp thanh toán thực tế (mới dừng lại ở môi trường Sandbox/Test).

## **4.3. Hướng phát triển trong tương lai**

* **Công nghệ:** Chuyển đổi module Chat sang sử dụng **WebSocket (STOMP)** để giao tiếp thời gian thực hiệu quả hơn.
* **Nền tảng:** Phát triển ứng dụng di động (Mobile App) sử dụng React Native hoặc Flutter, kết nối qua RESTful API hiện có.
* **Tính năng:** Tích hợp AI để gợi ý sản phẩm dựa trên lịch sử mua hàng và vị trí địa lý của người dùng.