**UỶ BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**Đề tài: Kế hoạch kiểm thử**

**Website bán thức ăn nhanh UMADAKIMASU**

Môn học: Kiểm thử phần mềm

Năm học: 2025-2026

Giảng viên: TS. Đỗ Như Tài

|  |  |  |
| --- | --- | --- |
| Thành viên: | | |
| Hồ Nguyên Minh | 3122411123 | honguyenminh123@gmail.com |
| Thông Trung Quyền | 3122411175 | trungquyen286@gmail.com |
| Lục Trọng Khoa | 3122411096 | Luctrongkhoa@gmail.com |

**Thành phố Hồ Chí Minh, năm 2025**

Mục lục

[Mục lục 1](#_75wzm59c2kq6)

[1: Giới thiệu 1](#_ayvo8m2r2q1t)

[1.1:Tổng quan 1](#_px8a1khzc7o4)

[1.2:Business Context 2](#_rvm4mcgf90yt)

[2: Phân tích thiết kế hệ thống và thiết kế kiến trúc 4](#_rpgvsyce57)

[2.1: Thiết kế hệ thống 4](#_ipmim28elpdf)

[2.1.1: Yêu cầu chức năng(BR) 4](#_sv00g7vyksth)

[2.1.2: Business Actor 6](#_e66blcawww1s)

[2.1.3: Chức năng - Business process ( quy trình) 7](#_wiz4u3lhoe7v)

[2.1.4: Dữ liệu - Conceptual model 34](#_tehnftkcxvc4)

[2.1.5: Giao diện - Wire flow 36](#_kjqvqrnn2q77)

[2.2: Thiết kế kiến trúc 37](#_9pnd4ydcccay)

[2.2.1:Sơ đồ tổng quan 37](#_l85h692oc7ql)

[3: Test Plan 42](#_dpfdneboyatw)

[**3.1: Purpose** 42](#_770y4xmq3v5h)

[3.2: Background 42](#_418ipmbx8od)

[3.3: Scope 42](#_hu1wp19hae2l)

[3.4: Project Identification 42](#_w16tkvfohz52)

[3.5: References 43](#_55kle1frvlad)

[3.6: Features to be Tested 44](#_26nn3ecrvmhd)

[3.6.1: Functional Testing 44](#_xqemxp5q539e)

[3.6.2: Usability Testing 45](#_t3v2eams6au5)

[3.6.3: Design Constraints 46](#_sh0tktb0n25l)

[3.6.4: Interfaces 46](#_inj3vr934ev2)

[3.7: Test Strategy 47](#_5hsm8wtljkev)

[3.7.1: Testing Types 47](#_chpkrd30gfmx)

[3.7.2: Tools 48](#_f1k4co565lwl)

[3.8: Nguồn nhân lực (Human Resources) 48](#_2icxishhkfiy)

[3.9: Các mốc tiến độ dự án (Project Milestones) 49](#_7iset0dfamdc)

[3.10: Deliverables 50](#_tp93lh66937q)

Từ điển thuật ngữ

|  |  |  |
| --- | --- | --- |
| **Từ viết tắt** | **Định nghĩa** | **Ghi chú** |
| Content Type | Kiểu nội dung được quản lý trong hệ thống |  |
| Workflow | Dòng công việc được sử dụng trong quá trình biên tập và xuất bản thông tin trong hệ thống |  |
| Sitemap | Sơ đồ website |  |
| System log | Ghi lại nhật ký hệ thống |  |
| User | Người sử dụng |  |
| SGU | Trường Đại học Sài Gòn |  |
| Ban ĐBCT  TTGD | Ban đảm bảo chất lượng và thanh tra giáo dục |  |

1: Giới thiệu

1.1:Tổng quan

UMADAKIMASU là một **ứng dụng web đặt đồ ăn (food delivery / food ordering)** hướng tới người dùng cuối (khách đặt hàng), quản trị viên (quản lý menu, đơn hàng) và đôi khi cả tài xế/giao hàng. Mục tiêu là tạo trải nghiệm đặt đồ nhanh gọn, có giỏ hàng, thanh toán và quản lý đơn hàng qua admin panel.

Các tính năng trong ứng dụng bao gồm:

**Danh mục món (Menu Management):** hiển thị món ăn theo danh mục, hỗ trợ hình ảnh, mô tả, đánh giá.

**Đặt hàng trực tuyến (Online Ordering):** khách hàng có thể chọn món, thêm vào giỏ, áp dụng khuyến mãi và tiến hành thanh toán.

**Thanh toán & Giao hàng (Payment & Delivery):** hỗ trợ nhiều phương thức thanh toán, theo dõi trạng thái giao hàng.

**Quản lý người dùng & phân quyền (Access Control):** hệ thống đăng nhập, phân quyền Buyer / SysAdmin để kiểm soát truy cập.

Đây là nền tảng giúp số hóa hoạt động nhà hàng, tăng trải nghiệm khách hàng, giảm tải quản lý và nâng cao hiệu quả vận hành.

1.2:Business Context

**Đối tượng chính:**

*Khách hàng (Buyer):* sử dụng website để duyệt món, đặt hàng, thanh toán

*Quản trị viên (SysAdmin):* quản lý menu, kho, đơn hàng, tài khoản người dùng, xử lý thanh toán & đơn giao hàng.

*Cổng thanh toán (Payment Gateway):* tích hợp để xử lý giao dịch trực tuyến.

**Hoạt động nghiệp vụ chính:**

Duyệt & chọn món: khách hàng truy cập website, xem danh mục, search món ăn.

Đặt hàng qua giỏ: khách hàng thêm món vào giỏ, cập nhật số lượng, xem tổng tiền, checkout.

Thanh toán: hệ thống kiểm tra tồn kho, xử lý giao dịch với Payment Gateway,

Giao hàng: hệ thống quản lý đơn giao, cập nhật trạng thái giao hàng.

Quản lý truy cập: người dùng đăng ký/đăng nhập, phân quyền Buyer hoặc SysAdmin để giới hạn chức năng.

**Kết quả mong đợi:**

Khách hàng có trải nghiệm liền mạch từ duyệt món → đặt món → thanh toán → nhận món/giao hàng.

Quản trị viên kiểm soát toàn bộ hoạt động từ quản lý menu, đơn hàng, đến phân quyền người dùng.

Nhà hàng tối ưu hóa vận hành, giảm sai sót thủ công, tăng doanh thu và sự hài lòng của khách hàng.

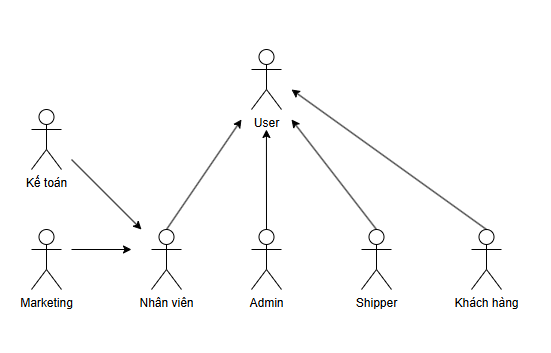
2: Phân tích thiết kế hệ thống và thiết kế kiến trúc

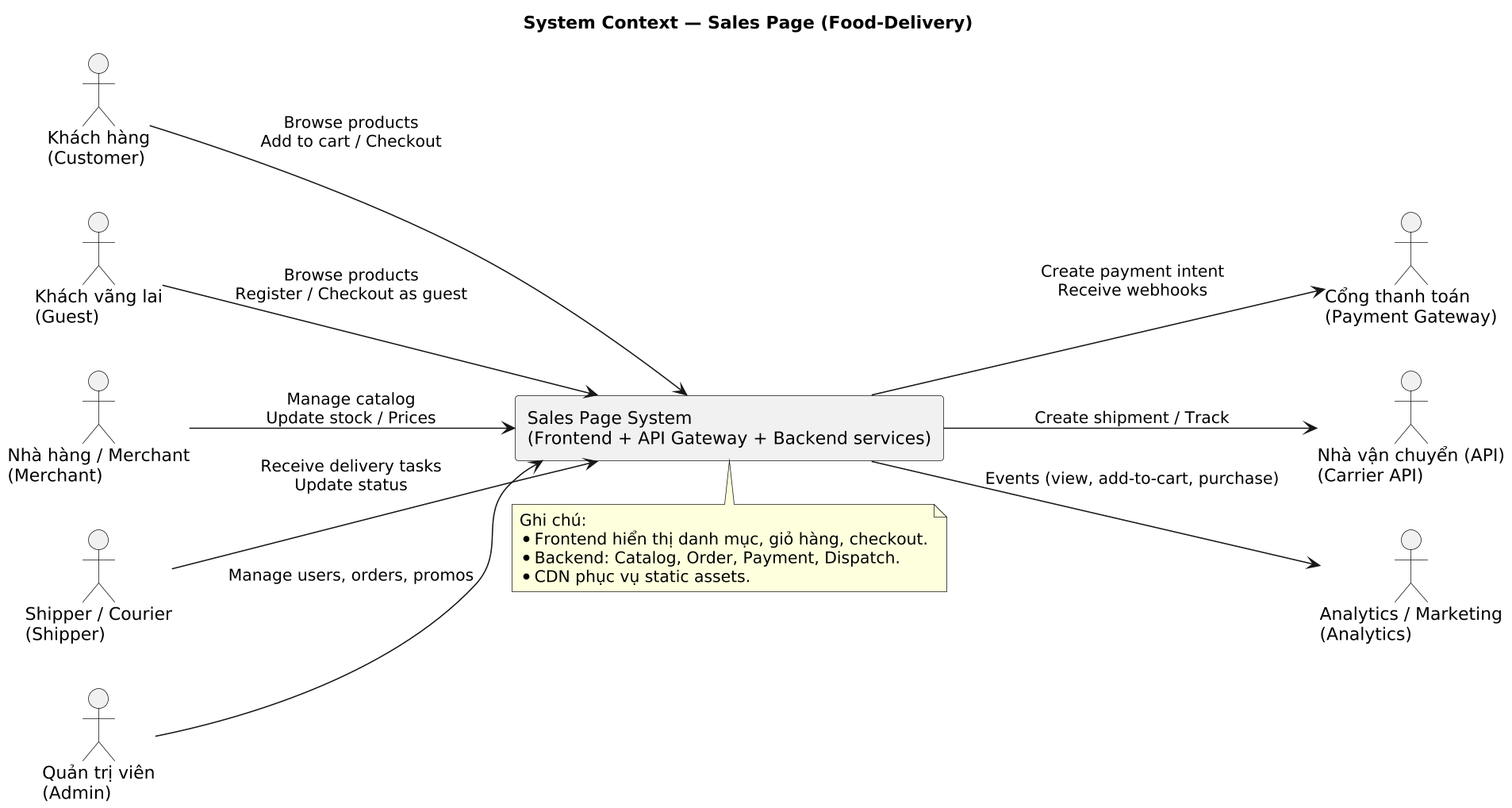
2.1: Thiết kế hệ thống

2.1.1: Yêu cầu chức năng(BR)

| Yêu cầu chức năng (ID) | Vai trò (Role/Actor) | Nội dung yêu cầu (Requirement Description) | Mức thiết yếu (Priority) |
| --- | --- | --- | --- |
| **BR1** | **QUẢN LÝ TÀI KHOẢN KHÁCH HÀNG** | | |
| BR1.1 | Khách hàng (KH) | Cho phép **Đăng ký** tài khoản mới (Tên, Email, Mật khẩu, Số điện thoại). | High |
| BR1.2 | Khách hàng (KH) | Thực hiện **Đăng nhập/Đăng xuất** an toàn (**JWT Authentication**). | High |
| BR1.3 | Khách hàng (KH) | Xem và **Cập nhật** thông tin cá nhân và quản lý nhiều địa chỉ giao hàng. | Medium |
| **BR2** | **QUẢN LÝ SẢN PHẨM & ĐẶT HÀNG** | | |
| BR2.1 | Khách hàng (KH) | **Duyệt Menu**: Xem danh sách món ăn/thực đơn với hình ảnh, mô tả và giá cả. | High |
| BR2.2 | Khách hàng (KH) | Thực hiện chức năng **Tìm kiếm** và **Lọc** món ăn theo tên hoặc danh mục. | Medium |
| BR2.3 | Khách hàng (KH) | Quản lý **Giỏ hàng**: Thêm/Xóa món, điều chỉnh số lượng. | High |
| BR2.4 | Khách hàng (KH) | Xem **Chi tiết Đơn hàng** (tổng tiền, phí giao hàng) trước khi xác nhận. | High |
| **BR3** | **THANH TOÁN & XỬ LÝ ĐƠN HÀNG** | | |
| BR3.1 | Khách hàng (KH) | **Thanh toán trực tuyến** an toàn thông qua cổng thanh toán tích hợp (**Stripe**). | High |
| BR3.2 | Khách hàng (KH) | Lựa chọn phương thức **Thanh toán khi nhận hàng (COD)**. | High |
| BR3.3 | Khách hàng (KH) | **Hoàn tất Đặt hàng** và nhận thông báo/xác nhận từ hệ thống. | High |
| BR3.4 | Khách hàng (KH) | **Theo dõi Trạng thái Đơn hàng** (đang xử lý, đang chuẩn bị, đang giao). | Medium |
| BR3.5 | Khách hàng (KH) | Xem lại **Lịch sử Đơn hàng** và chi tiết các đơn hàng đã hoàn tất. | High |
| **BR4** | **CHỨC NĂNG QUẢN TRỊ (ADMIN PANEL)** | | |
| BR4.1 | Quản trị viên (Admin) | **Quản lý Đơn hàng**: Xem tổng quan và chi tiết tất cả đơn hàng. | High |
| BR4.2 | Quản trị viên (Admin) | **Cập nhật Trạng thái** đơn hàng (xác nhận, từ chối, hoàn thành, v.v.). | High |
| BR4.3 | Quản trị viên (Admin) | **Quản lý Menu**: Thêm, sửa, xóa, và cập nhật giá, hình ảnh món ăn. | High |
| BR4.4 | Quản trị viên (Admin) | Quản lý **Danh mục** (thêm/sửa/xóa). | Medium |
| BR4.5 | Quản trị viên (Admin) | **Quản lý Người dùng**: Xem danh sách và thông tin cơ bản của tất cả Khách hàng. | Medium |
| BR4.6 | Quản trị viên (Admin) | **Báo cáo Thống kê** về doanh thu và số lượng đơn hàng trong khoảng thời gian. | Medium |
| **BR5** | **CHỨC NĂNG HẬU MÃI** | | |
| BR5.1 | Khách hàng (KH) | **Đánh giá và viết nhận xét** (Rating & Review) cho món ăn và dịch vụ. | Medium |
| BR5.2 | Quản trị viên (Admin) | **Quản lý Đánh giá**: Xem và ẩn/hiển thị các đánh giá từ khách hàng. | Low |

2.1.2: Business Actor





2.1.3: Chức năng - Business process ( quy trình)

**Workflow**

- Quy trình nghiệp vụ của một website bán hàng thức ăn trực tuyến xoay quanh việc quản lý ba luồng hoạt động chính: Khách hàng đặt hàng, Hệ thống xử lý và Quản trị vận hành. - Quy trình nghiệp vụ cơ bản được mô tả như sau:

**Quy trình Đặt hàng của Khách hàng (User Flow)**

- Các bước mà người dùng thực hiện để mua món ăn trên website:

| STT | Bước nghiệp vụ | Chi tiết hoạt động |
| --- | --- | --- |
| **1.** | **Truy cập & Xác thực** | Khách hàng truy cập website. Thực hiện **Đăng nhập** hoặc **Đăng ký** tài khoản (sử dụng **JWT Authentication**). |
| **2.** | **Duyệt Menu & Tìm kiếm** | Khách hàng xem danh sách các món ăn, nhà hàng, hoặc sử dụng chức năng tìm kiếm/lọc để chọn món. |
| **3.** | **Chọn món & Giỏ hàng** | Khách hàng chọn món, tùy chỉnh nếu cần, và thêm vào **Giỏ hàng (Add to Cart)**. |
| **4.** | **Kiểm tra & Thanh toán** | Khách hàng kiểm tra lại đơn hàng, nhập thông tin giao hàng, và lựa chọn phương thức **Thanh toán** an toàn (ví dụ: qua **Stripe** hoặc COD). |
| **5.** | **Hoàn tất Đặt hàng** | Khách hàng xác nhận và đặt hàng. Hệ thống gửi thông báo xác nhận đơn hàng (email/push notification). |

**Quy trình Xử lý Đơn hàng (System & Restaurant Flow)**

- Các bước diễn ra sau khi đơn hàng được đặt, chủ yếu được thực hiện qua Bảng điều khiển quản trị (Admin Panel) và tại Nhà hàng:

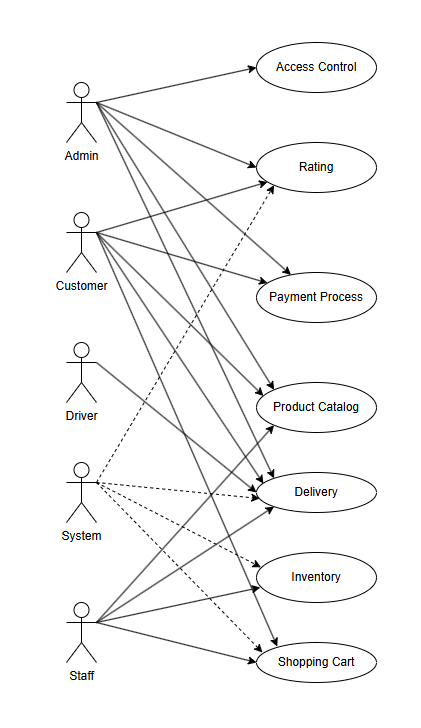
| STT | Bước nghiệp vụ | Chi tiết hoạt động |
| --- | --- | --- |
| **1.** | **Tiếp nhận Đơn hàng** | **Hệ thống quản trị (Admin)** hoặc **Nhà hàng** nhận thông báo về đơn hàng mới. |
| **2.** | **Xác nhận & Chuẩn bị** | Nhà hàng kiểm tra tính khả dụng của món ăn và **Xác nhận/Từ chối** đơn hàng. Nếu xác nhận, bắt đầu quá trình chuẩn bị món ăn. |
| **3.** | **Quản lý Đơn hàng (Order Management)** | Admin cập nhật trạng thái đơn hàng (ví dụ: đang chuẩn bị, đã sẵn sàng giao). |
| **4.** | **Bàn giao Vận chuyển** | Món ăn sau khi hoàn tất được đóng gói và bàn giao cho bộ phận giao hàng. |

**Quy trình Giao hàng & Hậu mãi (Fulfillment & After-Sale Flow)**

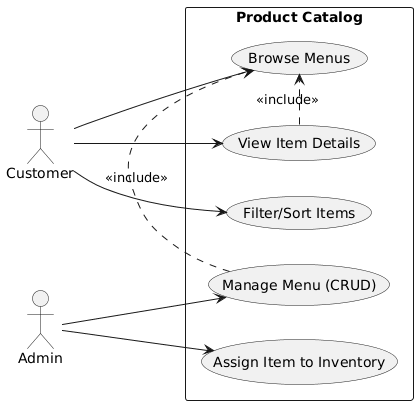
- Các bước cuối cùng để hoàn tất giao dịch và thu thập phản hồi:

| STT | Bước nghiệp vụ | Chi tiết hoạt động |
| --- | --- | --- |
| **1.** | **Vận chuyển & Theo dõi** | Khách hàng có thể **Theo dõi đơn hàng theo thời gian thực (Real-Time Tracking)**. |
| **2.** | **Hoàn tất Giao dịch** | Đơn giao hàng thành công. Khách hàng thanh toán nếu chọn COD. Hệ thống ghi nhận đơn hàng đã hoàn tất. |

2.1.3.2: Use case

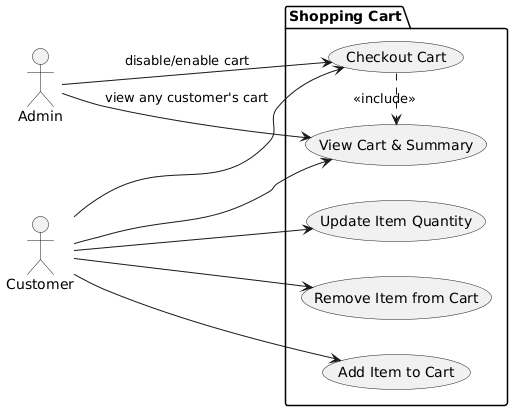


**Xem và quản lý danh mục món ăn**

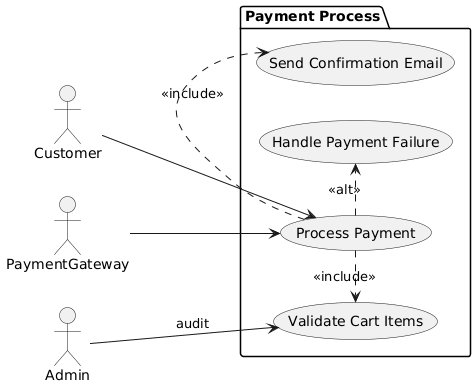


|  |  |  |
| --- | --- | --- |
| Tên use case | Xem và quản lý danh mục món ăn | |
| Tác nhân chính | Buyer (Khách hàng) và SysAdmin (Quản trị viên hệ thống) | |
| Mục tiêu | Khách hàng có thể xem các món ăn đang được bán trên website, tìm kiếm và sắp xếp theo nhu cầu. Quản trị viên có thể thêm, chỉnh sửa hoặc xóa món ăn để cập nhật thông tin thực tế của nhà hàng. | |
| Tiền điều kiện | Hệ thống có ít nhất 1 món ăn có sẵn  Người dùng đã đăng nhập vào hệ thống. | |
| Luồng chính | Actor action | System Respone |
| 1. Người dùng truy cập vào trang “Danh mục món ăn” |  |
|  | 2. Hệ thống hiển thị danh sách các món có sẵn, bao gồm tên, hình ảnh, giá, mô tả ngắn và trạng thái còn hàng. |
| 3. Người dùng có thể tìm kiếm món theo tên hoặc danh mục, hoặc lọc theo giá, món nổi bật, đánh giá,… |  |
|  | 4. Hệ thống cập nhật kết quả hiển thị theo các tiêu chí mà khách hàng chọn.. **A1** |
| 5. Khách hàng nhấn vào một món cụ thể để xem chi tiết. |  |
|  | 6. Hệ thống hiển thị đầy đủ thông tin chi tiết món ăn: tên, mô tả, giá, hình ảnh và đánh giá. |
| 7. Quản trị viên truy cập trang quản trị và thêm, chỉnh sửa hoặc xóa món ăn. |  |
|  | 8. Hệ thống lưu các thay đổi, cập nhật dữ liệu và hiển thị lại danh mục món ăn mới cho khách hàng.**A2** |
| Luồng thay thế | **A1**. Khách hàng nhập điều kiện lọc nhưng không có món phù hợp. Hệ thống hiển thị thông báo: “Không tìm thấy món phù hợp.”  **A2**. Quản trị viên nhập dữ liệu không hợp lệ. Hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại thông tin hợp lệ. | |
| Hậu điều kiện: | Khách hàng xem được thông tin món ăn đầy đủ, chính xác.  Danh mục món ăn hiển thị được cập nhật theo dữ liệu thực tế. | |
| Activity Diagram | | |
| Screen Design | | |
|  | | |

**Quản lý Giỏ hàng**

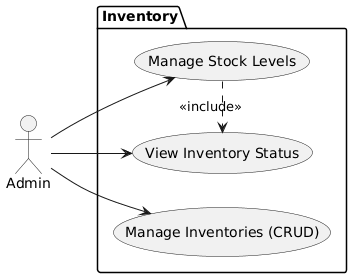


|  |  |  |
| --- | --- | --- |
| Tên use case | Quản lý Giỏ hàng | |
| Tác nhân chính | Khách hàng và Quản trị viên hệ thống | |
| Mục tiêu | Khách hàng có thể thêm, chỉnh sửa, xóa món trong giỏ và tiến hành thanh toán. Quản trị viên có thể theo dõi, quản lý hoặc vô hiệu hóa giỏ hàng không hợp lệ. | |
| Tiền điều kiện | Khách hàng đã đăng nhập hợp lệ. | |
| Luồng chính | Actor action | System Respone |
| 1. Khách hàng chọn món và nhấn nút “Thêm vào giỏ hàng.”**A1** |  |
|  | 2. Hệ thống kiểm tra tồn kho. Nếu hợp lệ, món được thêm vào giỏ hàng. |
| 3. Khách hàng truy cập trang “Giỏ hàng.” |  |
|  | 4. Hệ thống hiển thị danh sách các món trong giỏ, gồm tên, hình ảnh, giá, số lượng, khuyến mãi, phí giao hàng và tổng cộng. |
| 5. Khách hàng thay đổi số lượng món ăn trong giỏ. **A2** |  |
|  | 6. Hệ thống tự động cập nhật tổng tiền theo số lượng mới. |
| 7. Khách hàng xóa một món khỏi giỏ nếu không muốn mua. |  |
|  | 8. Hệ thống loại bỏ món đó khỏi danh sách và tính lại tổng tiền. |
| 9. Khách hàng nhấn nút **“Thanh toán.”** **A3** |  |
|  | 10. Hệ thống kiểm tra điều kiện. Nếu hợp lệ, chuyển sang quy trình thanh toán. |
| 11. Quản trị viên truy cập trang quản trị để xem danh sách các giỏ hàng đang tồn tại. |  |
|  | 12. Hệ thống hiển thị danh sách giỏ hàng, tổng giá trị, khuyến mãi và trạng thái, đồng thời cho phép vô hiệu hóa giỏ hàng khi cần. |
| Luồng thay thế | **A1** Nếu món đã hết hàng, hệ thống thông báo: “Món này đã hết hàng, không thể thêm vào giỏ.”  **A2** Nếu số lượng nhập vượt quá tồn kho, hệ thống cảnh báo: “Số lượng vượt quá tồn kho, vui lòng giảm lại.”  **A3**. Nếu giỏ hàng rỗng khi nhấn “Thanh toán”. Hệ thống hiển thị: “Không thể thanh toán vì giỏ hàng trống.” | |
| Hậu điều kiện: | Khách hàng xem được thông tin món ăn đầy đủ, chính xác.  Danh mục món ăn hiển thị được cập nhật theo dữ liệu thực tế. | |
| Activity Diagram | | |
| Screen Design | | |
|  | | |

**Quy trình Thanh toán** 

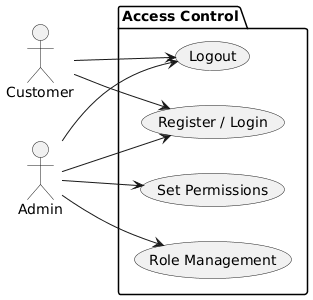
|  |  |  |
| --- | --- | --- |
| Tên use case | Quy trình Thanh toán | |
| Tác nhân chính | Khách hàng và Quản trị viên hệ thống, Cổng thanh toán (Payment Gateway) | |
| Mục tiêu | Khách hàng có thể thanh toán đơn hàng sau khi đặt món thành công. Hệ thống xử lý thanh toán, gửi xác nhận và cập nhật trạng thái đơn hàng. | |
| Tiền điều kiện | Khách hàng đã có giỏ hàng hợp lệ.  Các món trong giỏ còn hàng trong kho. | |
| Luồng chính | Actor action | System Respone |
| 1. Khách hàng nhấn nút “Thanh toán” trên trang giỏ hàng. |  |
|  | 2. Hệ thống kiểm tra lại thông tin đơn hàng, bao gồm danh sách món, số lượng, tồn kho và tổng tiền.**A1** |
|  | 3. Hệ thống hiển thị trang **“Thông tin thanh toán”**, yêu cầu khách hàng nhập địa chỉ giao hàng, phương thức thanh toán và xác nhận đơn. |
| 4. Khách hàng nhập đầy đủ thông tin và nhấn **“Xác nhận thanh toán.”** |  |
|  | 5. Hệ thống gửi yêu cầu thanh toán đến Cổng thanh toán, xử lý thông tin thanh toán |
|  | 6. Nếu thành công, hệ thống ghi nhận thanh toán, tạo mã đơn hàng và gửi email xác nhận cho khách hàng.**A2** |
|  | 7. Hệ thống cập nhật trạng thái đơn hàng là “Đã thanh toán.” |
| 9. Quản trị viên truy cập trang “Quản lý đơn hàng” để kiểm tra các giao dịch mới. |  |
| Luồng thay thế | **A1** Nếu giỏ hàng trống hoặc món đã hết hàng, hệ thống hiển thị: “Không thể thanh toán vì đơn hàng không hợp lệ.”  **A2** Nếu thông tin đơn hàng không hợp lệ, hệ thống sẽ hiển thị: “Không thể thanh toán vì thông tin không hợp lệ” | |
| Hậu điều kiện: | Thanh toán hoàn tất, hệ thống tạo đơn hàng và gửi email xác nhận.  Khách hàng xem được trạng thái đơn hàng trong lịch sử mua hàng.  Quản trị viên có thể theo dõi và quản lý tất cả giao dịch trong hệ thống. | |
| Activity Diagram | | |
| Screen Design | | |
|  | | |

**Quản lý Món ăn**



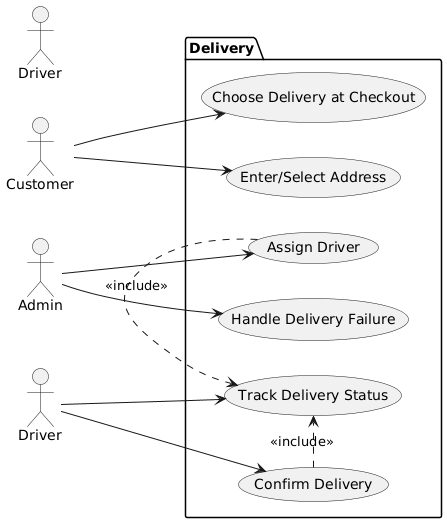
|  |  |  |
| --- | --- | --- |
| Tên use case | Quản lý Món ăn | |
| Tác nhân chính | Quản trị viên hệ thống | |
| Mục tiêu | Quản trị viên thêm, sửa, xóa và theo dõi số lượng tồn kho của từng món ăn. Hệ thống tự động trừ kho khi có đơn hàng mới. | |
| Tiền điều kiện | Quản trị viên đã đăng nhập hợp lệ.  Cơ sở dữ liệu đã có danh mục món ăn. | |
| Luồng chính | Actor action | System Respone |
| 1. Quản trị viên truy cập vào trang “Quản lý Món ăn.” |  |
|  | 2. Hệ thống hiển thị danh sách món ăn |
| 3. Quản trị viên chọn “Thêm mới món” |  |
| 4. Quản trị viên nhập thông tin và nhấn “Lưu.” |  |
|  | 5. Hệ thống kiểm tra thông tin, lưu vào cơ sở dữ liệu và cập nhật danh sách tồn kho.**A1** |
|  | 6. Khi có đơn hàng mới, hệ thống tự động trừ số lượng món trong kho tương ứng. |
| 7. Quản trị viên có thể ẩn món khỏi kho nếu món đó ngừng kinh doanh. |  |
|  | 8. Hệ thống xác nhận thao tác, xóa dữ liệu và cập nhật danh sách kho.**A2** |
| Luồng thay thế | **A1** Nếu thông tin nhập bị thiếu, hệ thống sẽ thông báo lỗi  **A2** Nếu món đang thuộc một đơn hàng chưa hoàn tất, hệ thống hiển thị cảnh báo: Không thể xóa món vì đang có đơn hàng đang xử lý | |
| Hậu điều kiện: | Thông tin món ăn, tồn kho được cập nhật chính xác sau mỗi đơn hàng hoặc thao tác quản trị.  Quản trị viên có thể dễ dàng theo dõi, thêm mới hoặc điều chỉnh số lượng tồn. | |
| Activity Diagram | | |
| Screen Design | | |
|  | | |

**Quản lý Đăng nhập**



|  |  |  |
| --- | --- | --- |
| Tên use case | Quản lý Đăng nhập | |
| Tác nhân chính | Khách hàng và Quản trị viên hệ thống | |
| Mục tiêu | Người dùng đăng nhập vào hệ thống bằng tài khoản hợp lệ.  Hệ thống phân quyền và điều hướng người dùng đến đúng giao diện theo vai trò. | |
| Tiền điều kiện | Người dùng đã có tài khoản hợp lệ trong hệ thống.  Hệ thống đang hoạt động bình thường. | |
| Luồng chính | Actor action | System Respone |
| 1. Người dùng truy cập trang “Đăng nhập”. |  |
|  | 2. Hệ thống hiển thị biểu mẫu nhập tên đăng nhập và mật khẩu..**A1** |
| 3. Người dùng nhập thông tin đăng nhập và nhấn “Đăng nhập.” |  |
|  | 4. Hệ thống kiểm tra thông tin, nếu thông tin hợp lệ, hệ thống điều hướng người dùng đến trang tương ứng theo phân quyền hệ thống. **A2, A3** |
| Luồng thay thế | **A1** Nếu tên đăng nhập hoặc mật khẩu không chính xác. Hệ thống hiển thị: “Tên đăng nhập hoặc mật khẩu không đúng.”  **A2** Nếu vai trò của người dùng là Khách hàng, hệ thống giới hạn quyền truy cập chỉ trong khu vực mua hàng (xem món, giỏ hàng, thanh toán).  **A3** Nếu vai trò là Quản trị viên, hệ thống cấp quyền truy cập toàn bộ chức năng quản lý: món ăn, đơn hàng, kho, người dùng. | |
| Hậu điều kiện: | Người dùng đăng nhập thành công và được điều hướng đúng giao diện theo vai trò. | |
| Activity Diagram | | |
| Screen Design | | |
|  | | |

**Quản lý Giao hàng**



|  |  |  |
| --- | --- | --- |
| Tên use case | Quản lý Giao hàng | |
| Tác nhân chính | Khách hàng và Quản trị viên hệ thống, Nhân viên giao hàng | |
| Mục tiêu | Quản trị viên gán đơn cho tài xế và theo dõi tiến trình giao hàng. Nhân viên giao hàng cập nhật trạng thái đơn hàng sau mỗi bước. | |
| Tiền điều kiện | Đơn hàng đã được tạo và thanh toán thành công. | |
| Luồng chính | Actor action | System Respone |
| 1.Quản trị viên truy cập mục “Quản lý giao hàng.” |  |
|  | 2. Hệ thống hiển thị danh sách đơn hàng. |
| 3. Quản trị viên chọn đơn hàng chưa giao và gán tài xế giao hàng. |  |
|  | 4. Hệ thống cập nhật trạng thái đơn là “Đang giao hàng.”. |
| 5. Nhân viên giao hàng cập nhật trạng thái đơn theo tiến trình: Đang giao hàng và Đã giao thành công |  |
|  | 6. Khi giao hàng thành công, hệ thống ghi nhận trạng thái đơn “Hoàn tất.” |
| 7. Khách hàng nhận được thông báo xác nhận đơn hàng đã giao. |  |
| Luồng thay thế |  | |
| Hậu điều kiện: | Trạng thái giao hàng được theo dõi và cập nhật chính xác trong hệ thống.  Khách hàng nhận thông tin đơn hàng theo thời gian thực.  Quản trị viên giám sát toàn bộ tiến trình và xử lý sự cố nhanh chóng. | |
| Activity Diagram | | |
| Screen Design | | |
|  | | |

2.1.4 User Story

User Stories — Product Catalog (Danh mục món)

**Là một Khách hàng (Buyer),** tôi muốn xem danh sách món trên trang chủ với khả năng lọc và sắp xếp (tên, giá, đánh giá, flag món hot, ảnh, mô tả ngắn) để tôi có thể nhanh tìm món cần mua.

**Khi** lọc theo tên hoặc khoảng giá, **thì** danh sách cần thu hẹp chỉ hiển thị những món phù hợp.

**Khi** sắp xếp theo giá hoặc theo tên (tăng/giảm), **thì** danh sách phải phản ánh thứ tự tương ứng.

**Khi** vừa lọc vừa sắp xếp, **thì** hai thao tác phải kết hợp chính xác.

**Là một Khách hàng,** tôi muốn xem trang chi tiết món (tên, mô tả đầy đủ, tồn kho có sẵn, thông tin kho lưu trữ như địa chỉ, flag hot, đánh giá) để quyết định mua.

**Là Quản trị viên hệ thống (SysAdmin),** tôi muốn quản lý món (CRUD) và gán món vào kho tồn (assign inventory) để cập nhật dữ liệu hiển thị và tồn kho.

User Stories — Shopping Cart (Giỏ hàng)

**Là một Khách hàng,** tôi muốn thêm món từ trang danh mục vào giỏ (mặc định thêm 1 đơn vị) để giữ món định mua.

**Là một Khách hàng,** tôi muốn thêm món từ trang chi tiết vào giỏ (mặc định 1) nếu tôi quyết định mua khi xem chi tiết.

**Là một Khách hàng,** tôi muốn xem danh sách món trong giỏ và bảng tóm tắt (cart total, promotion item saving, subtotal, shipping cost, promotion shipping saving, total order amount) để biết chi phí hiện tại.

**Là một Khách hàng,** tôi muốn cập nhật số lượng món trong giỏ; **khi** cập nhật, **thì** bảng tóm tắt phải cập nhật tương ứng.

**Là một Khách hàng,** tôi muốn xóa món khỏi giỏ; **khi** xóa, **thì** bảng tóm tắt phải cập nhật tương ứng.

**Là một Khách hàng,** tôi muốn thực hiện checkout giỏ hàng để đặt hàng.

**Nếu** số lượng sản phẩm trong giỏ = 0, **thì** không cho phép checkout.

**Khi** giỏ hàng được checkout, **thì** quy trình thanh toán sẽ bắt đầu.

**Là Quản trị viên (SysAdmin),** tôi muốn xem giỏ hàng của mọi khách, hiển thị các thông tin tổng hợp như cart total, promotion savings, subtotal, shipping, total order amount.

**Là Quản trị viên,** tôi muốn enable/disable bất kỳ giỏ hàng nào (vô hiệu / kích hoạt) để quản lý giỏ hàng không hợp lệ.

User Stories — Payment Process (Thanh toán)

**Bất kỳ Khách hàng nào** có thể thanh toán (thực hiện payment).

**Khi** quy trình thanh toán bắt đầu (sau checkout), hệ thống sẽ:

xác thực thông tin sản phẩm (tồn kho, giá, mã khuyến mãi...),

xử lý thanh toán (mô phỏng/hoặc tích hợp gateway),

gửi email xác nhận/trạng thái cho Khách hàng.

**Nếu** có thông tin sản phẩm không hợp lệ trong quá trình thanh toán, **thì** thanh toán bị hủy và gửi email thông báo cho Khách hàng.

**Khi** thanh toán hoàn tất, **thì** trạng thái thanh toán được đánh dấu là processed và Khách hàng nhận email xác nhận.

User Stories — Inventory (Tồn kho)

**Là Quản trị viên (SysAdmin),** tôi muốn quản lý danh sách món ăn(CRUD) — thêm/xóa món ra danh sách món ăn

User Stories — Access Control (Đăng nhập & Phân quyền)

**Mỗi Khách hàng / Quản trị viên** là một User trong hệ thống.

**Là một Người dùng (Buyer hoặc SysAdmin),** tôi muốn đăng nhập vào hệ thống.

**Khi** người dùng có role = Buyer đăng nhập, **thì** chuyển đến trang danh mục sản phẩm.

**Khi** người dùng có role = SysAdmin đăng nhập, **thì** chuyển đến trang quản trị.

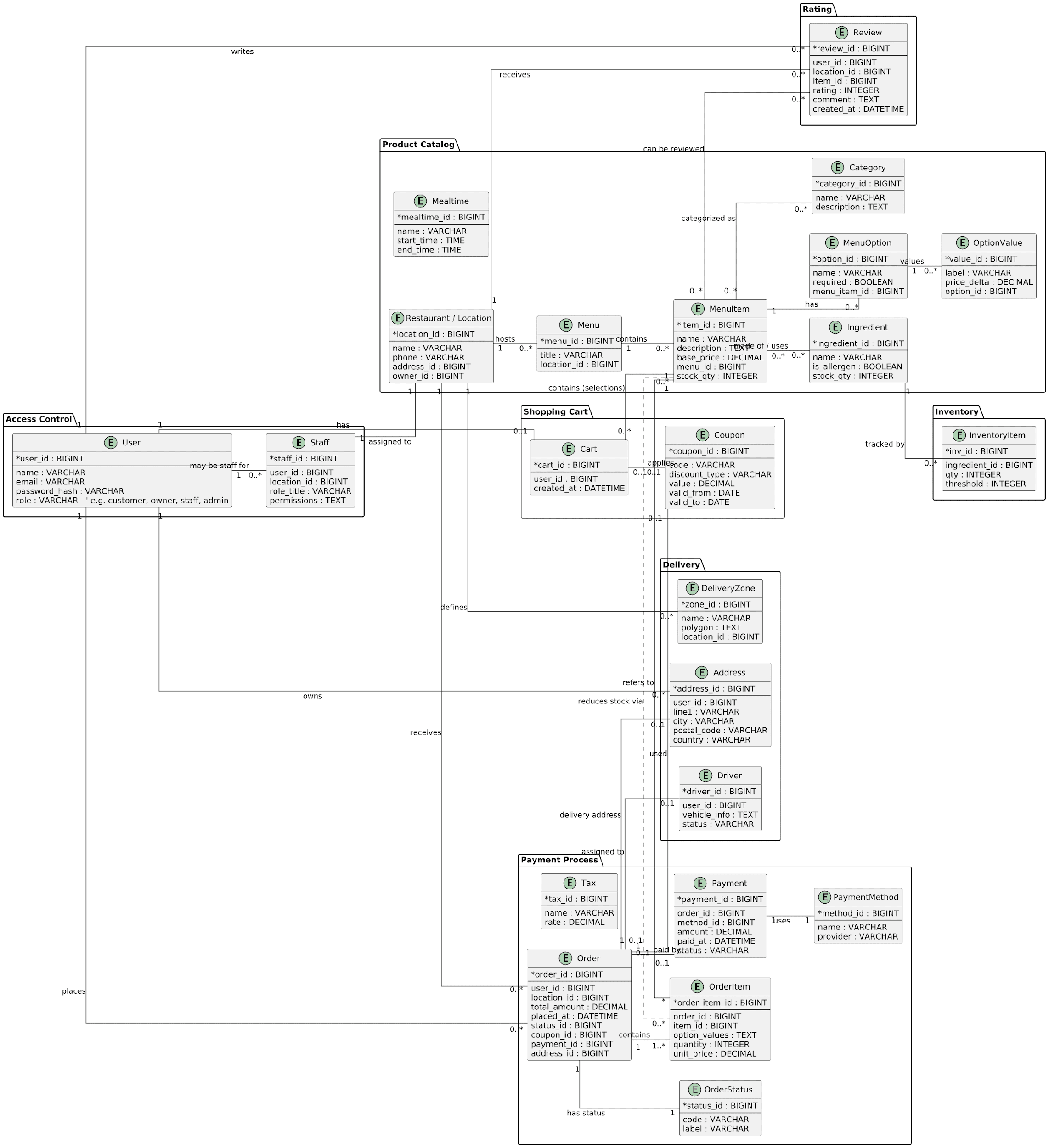
**Là một Người dùng,** tôi muốn đăng xuất khỏi hệ thống.

User Stories — Delivery (Giao hàng)

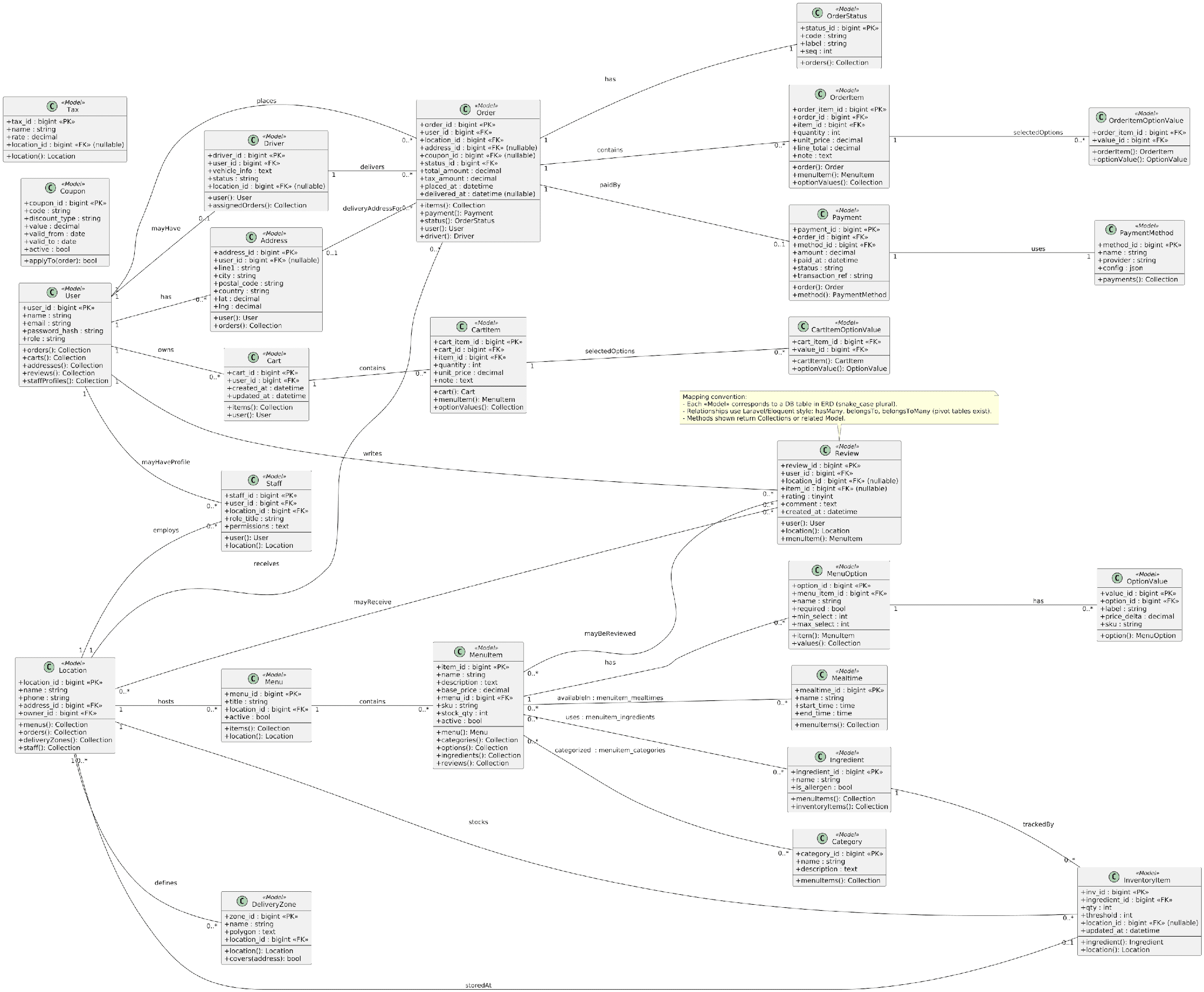
**Là một Khách hàng,** tôi muốn chọn giao hàng (địa chỉ, phí) khi checkout.

**Là Quản trị viên,** tôi muốn gán đơn cho tài xế, theo dõi trạng thái giao hàng (assigned, delivering, delivered).

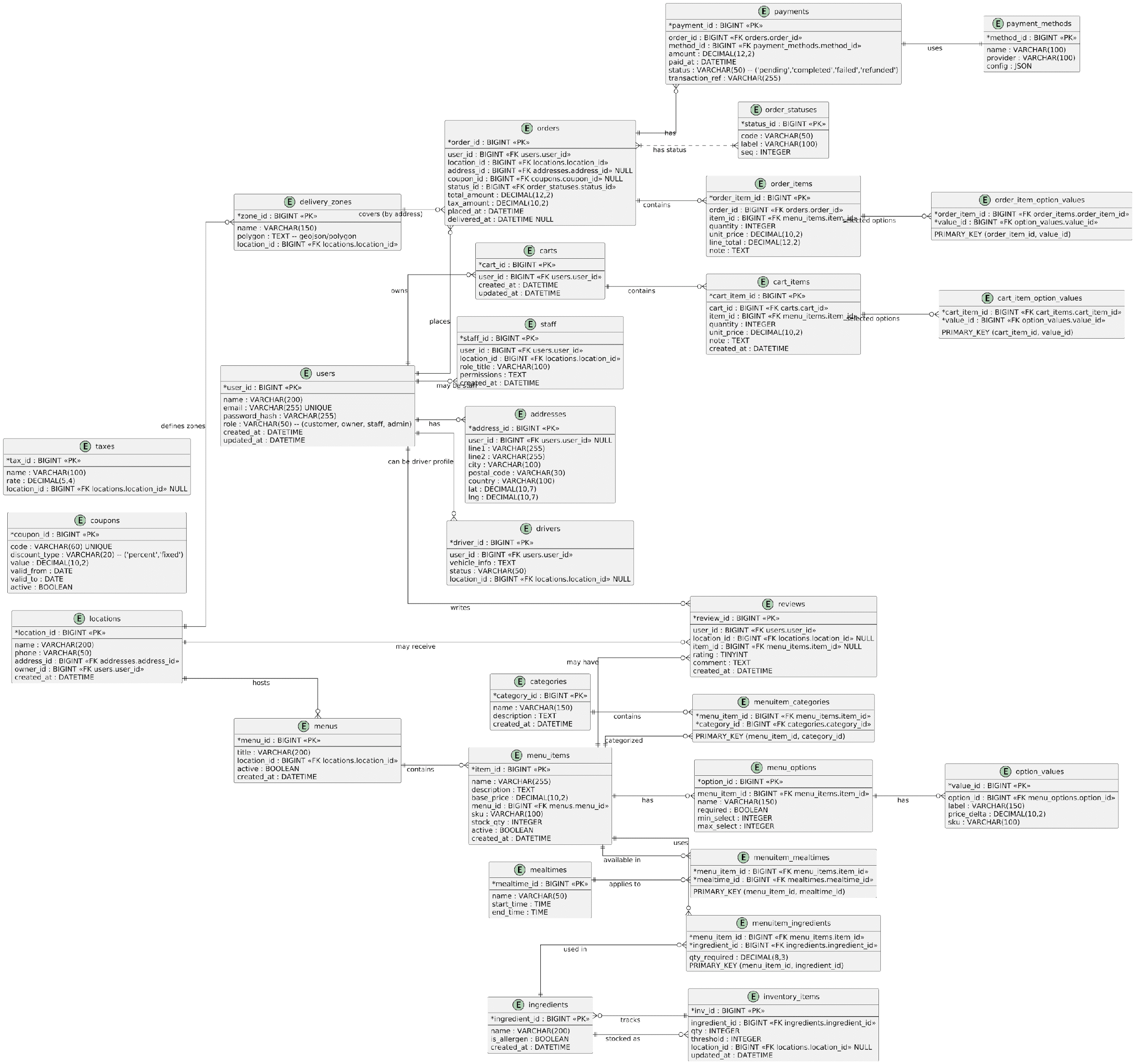
2.1.4: Dữ liệu - Conceptual model



**Object Model (Class Diagram)**



**Data model - ERD**



2.1.5: Giao diện - Wire flow

Nhánh chính:

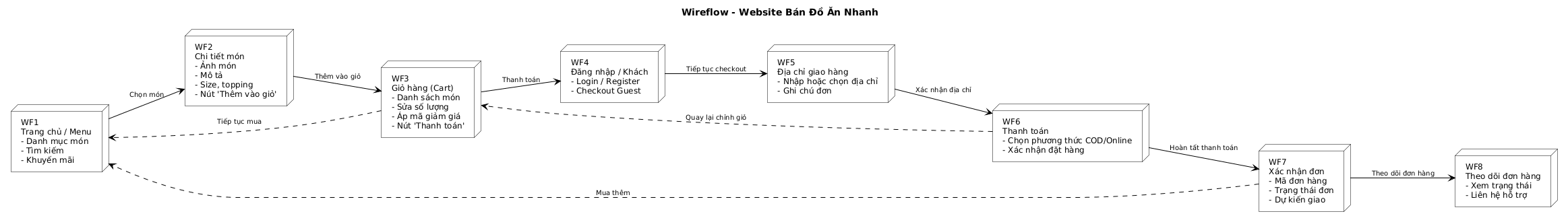
Trang chủ → Chi tiết món → Giỏ hàng → (Đăng nhập / Khách) → Địa chỉ → Thanh toán → Xác nhận

Nhánh phụ:

Từ Giỏ hàng → quay lại Trang chủ

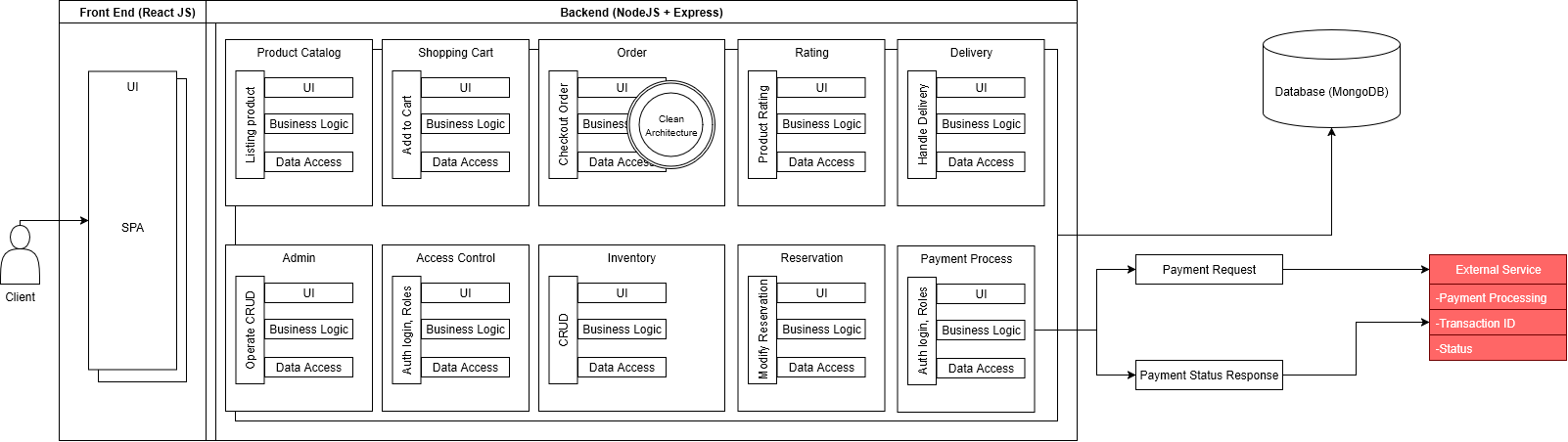
Từ Thanh toán → quay lại Giỏ hàng

Từ Xác nhận → theo dõi đơn (Order Tracking)



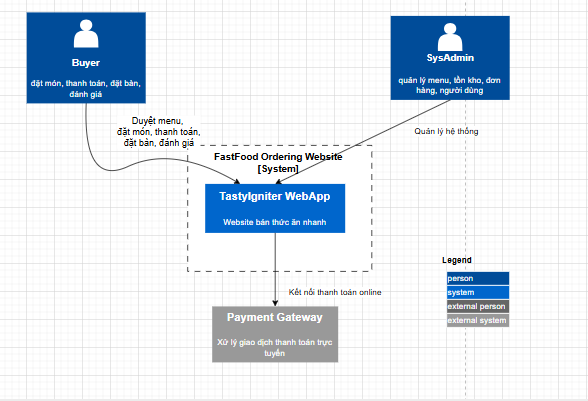
2.2: Thiết kế kiến trúc

**Sơ đồ khối**



**Khung nhìn thành phần**

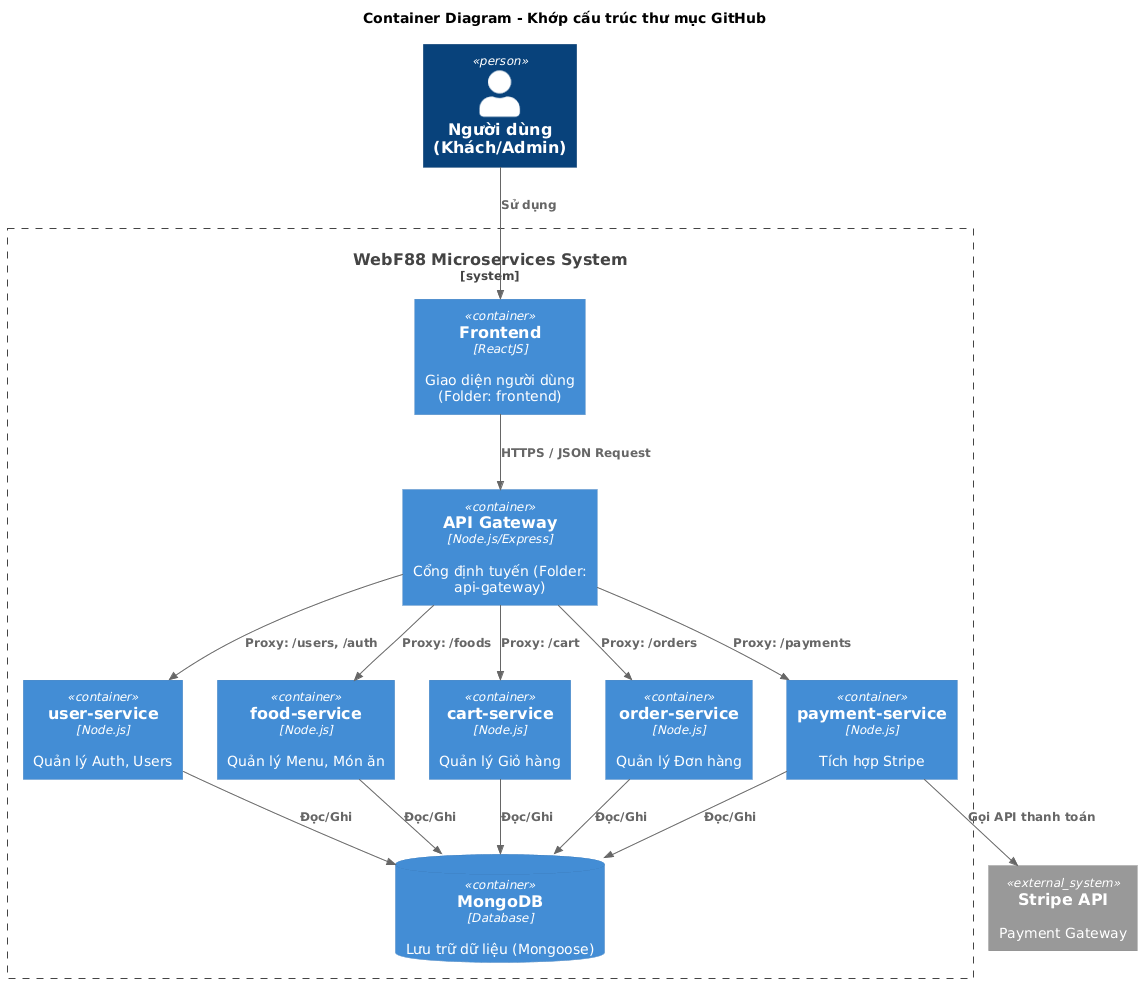
C1



**Mô tả chi tiết:**

* **Tác nhân nội bộ:**
  + **Khách hàng (Buyer):** Người dùng truy cập hệ thống để tìm kiếm món ăn, tạo đơn hàng và thực hiện thanh toán trực tuyến.
  + **Quản trị viên (SysAdmin):** Chịu trách nhiệm quản lý toàn bộ hệ thống (CRUD món ăn, quản lý người dùng) và cập nhật trạng thái đơn hàng (ví dụ: từ "Chờ xử lý" sang "Đang giao" hoặc "Hoàn tất").
* **Hệ thống bên ngoài:**
  + **Stripe Payment:** Hệ thống tích hợp với cổng thanh toán Stripe để xử lý giao dịch và trả về kết quả thanh toán.

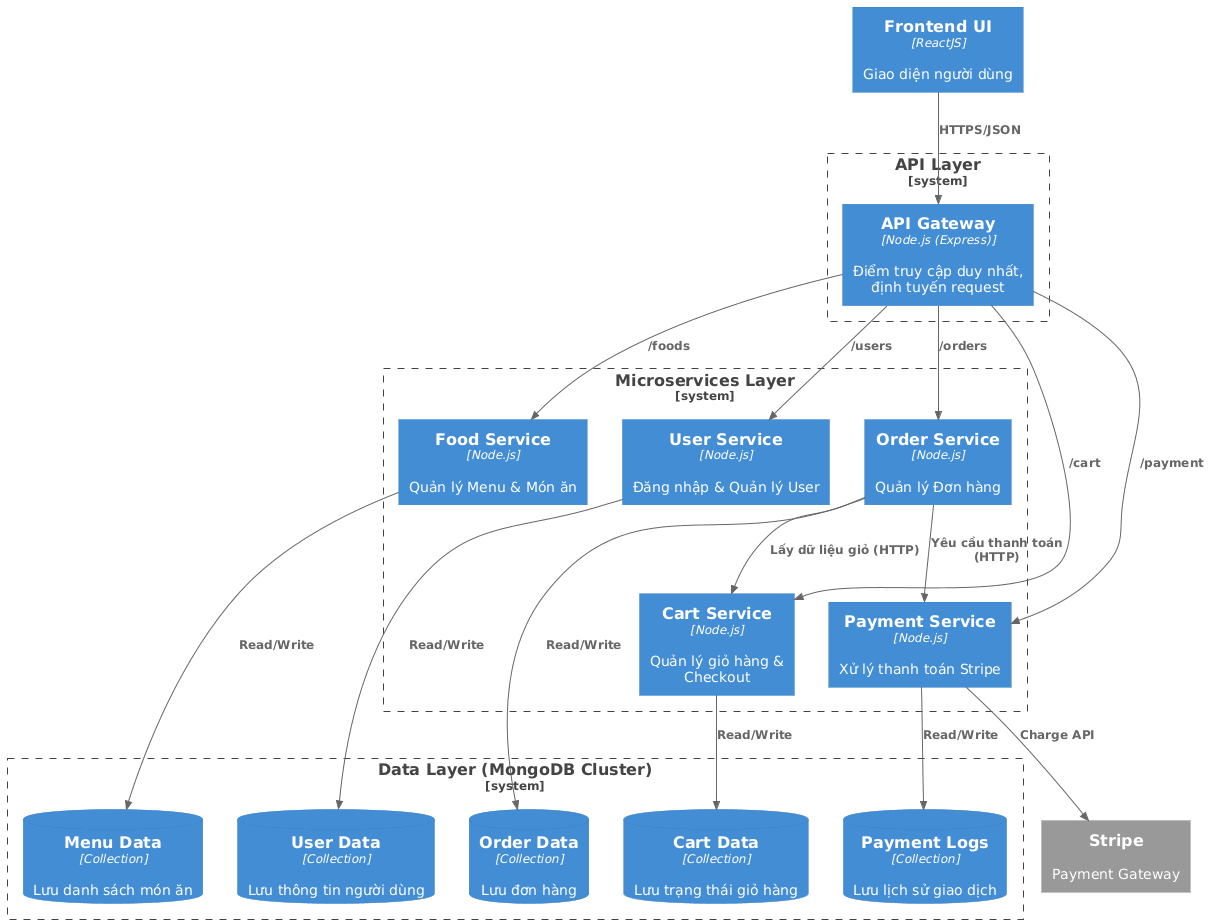
C2



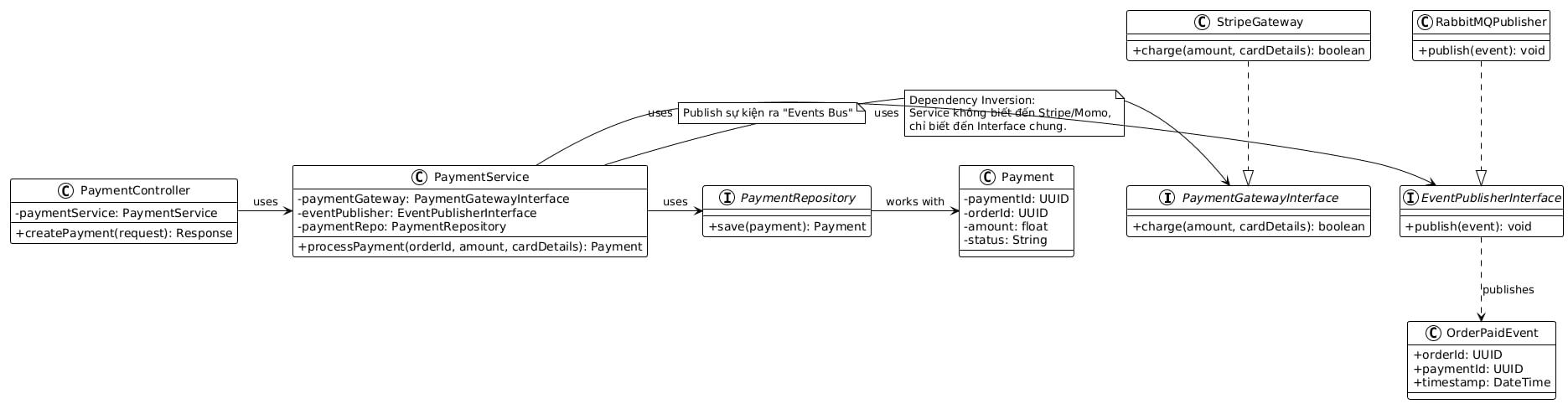
**Mô tả chi tiết:**

1. **Frontend (Web Application):** Được xây dựng bằng **ReactJS**, cung cấp giao diện tương tác duy nhất. Khách hàng sử dụng để đặt món, trong khi Quản trị viên sử dụng để quản lý dữ liệu thông qua các trang Dashboard được phân quyền.
2. **API Gateway:** Đóng vai trò là cổng vào duy nhất (Entry Point) của hệ thống Backend. Nó chịu trách nhiệm nhận request từ Frontend và định tuyến (Routing) đến đúng service đích (ví dụ: request /foods sẽ về food-service, /orders sẽ về order-service).
3. **Microservices Layer:** Hệ thống bao gồm 5 dịch vụ nghiệp vụ cốt lõi được phát triển bằng **Node.js (Express)**:
   * **User Service:** Quản lý xác thực (Auth), đăng ký/đăng nhập, phân quyền (Role-based) và thông tin hồ sơ người dùng.
   * **Food Service:** Quản lý danh mục món ăn (Menu), giá cả, thông tin chi tiết và hình ảnh sản phẩm.
   * **Cart Service:** Quản lý trạng thái giỏ hàng tạm thời của người dùng, tính toán tổng tiền trước khi tạo đơn.
   * **Order Service:** Tiếp nhận yêu cầu đặt hàng, lưu trữ thông tin đơn hàng và cho phép Admin cập nhật trạng thái xử lý (Ví dụ: Từ "Đã thanh toán" sang "Đang chuẩn bị").
   * **Payment Service:** Xử lý logic thanh toán, tạo phiên giao dịch và giao tiếp với Stripe API để xác nhận kết quả.
4. **Data Layer:** Sử dụng **MongoDB** (NoSQL), mỗi service tương tác với các collection dữ liệu riêng biệt để đảm bảo tính độc lập và toàn vẹn dữ liệu.

C3



**Phân tích luồng xử lý:**

* **Tầng API Layer (Gateway):**
  + Sử dụng **Express.js** làm lớp trung gian nhận các HTTP Request.
  + Thực hiện chức năng Reverse Proxy, giúp ẩn giấu cấu trúc mạng lưới service bên trong, tăng cường bảo mật cho hệ thống.
* **Tầng Microservices Layer:** Các service (Food, Order, User...) đều tuân thủ mô hình thiết kế **MVC (Model-View-Controller)**:
  + **Routes:** Định nghĩa các API Endpoint (RESTful).
  + **Controllers:** Chứa logic nghiệp vụ chính (Business Logic). Ví dụ: OrderController sẽ nhận yêu cầu tạo đơn, kiểm tra tồn kho (nếu có), và gọi sang Payment Service.
  + **Models (Mongoose):** Định nghĩa Schema dữ liệu và thực hiện các truy vấn trực tiếp xuống MongoDB.
  + **Giao tiếp liên service:** Các service trao đổi dữ liệu thông qua HTTP Request nội bộ (REST API). Ví dụ: Khi người dùng đặt hàng, Order Service sẽ gọi API của Cart Service để lấy danh sách món ăn hiện tại.
* **Tầng Data Layer (MongoDB Cluster):** Dữ liệu được tổ chức thành các Collection:
  + Menu Data: Chứa thông tin món ăn.
  + User Data: Chứa thông tin tài khoản Khách và Admin.
  + Order Data: Lưu trữ lịch sử đơn hàng và trạng thái.
  + Payment Logs: Lưu vết các giao dịch thanh toán.

**Mô tả chi tiết:**

Sơ đồ thể hiện cấu trúc thiết kế hướng đối tượng (Object-Oriented Design) của dịch vụ thanh toán, áp dụng các nguyên lý kỹ thuật phần mềm nâng cao để đảm bảo tính linh hoạt và dễ bảo trì:

* Luồng xử lý chính:
  + PaymentController: Đóng vai trò là lớp giao diện (Interface Layer), tiếp nhận yêu cầu thanh toán createPayment từ API và chuyển tiếp dữ liệu hợp lệ xuống lớp nghiệp vụ.
  + PaymentService: Là lớp trung tâm chứa logic nghiệp vụ (Business Layer). Tại đây, các quy tắc thanh toán được thực thi (ví dụ: kiểm tra số tiền, trạng thái đơn hàng).
* Áp dụng Dependency Inversion Principle (Nguyên lý đảo ngược sự phụ thuộc):
  + Thay vì PaymentService gọi trực tiếp đến StripeGateway (lớp cụ thể), nó chỉ phụ thuộc vào một giao diện trừu tượng là PaymentGatewayInterface.
  + Lợi ích: Giúp hệ thống không bị phụ thuộc cứng vào một nhà cung cấp thanh toán duy nhất. Nếu sau này cần chuyển từ Stripe sang PayPal hoặc Momo, lập trình viên chỉ cần tạo một class mới (ví dụ: MomoGateway) thực thi Interface này mà không cần sửa đổi code cốt lõi của PaymentService.
* Mô hình Event-Driven (Hướng sự kiện):
  + Sơ đồ thể hiện việc sử dụng EventPublisherInterface để phát đi sự kiện OrderPaidEvent (Sự kiện đơn hàng đã thanh toán) sau khi giao dịch thành công.
  + Điều này cho phép các service khác (như Order Service, Notification Service) "lắng nghe" và tự động cập nhật trạng thái hoặc gửi email mà không cần Payment Service phải gọi trực tiếp, giúp giảm sự phụ thuộc chéo (Decoupling) giữa các microservices.
* Lớp dữ liệu (Persistence):
  + PaymentRepository: Chịu trách nhiệm lưu trữ đối tượng Payment (Entity) vào cơ sở dữ liệu, tách biệt logic lưu trữ khỏi logic nghiệp vụ.

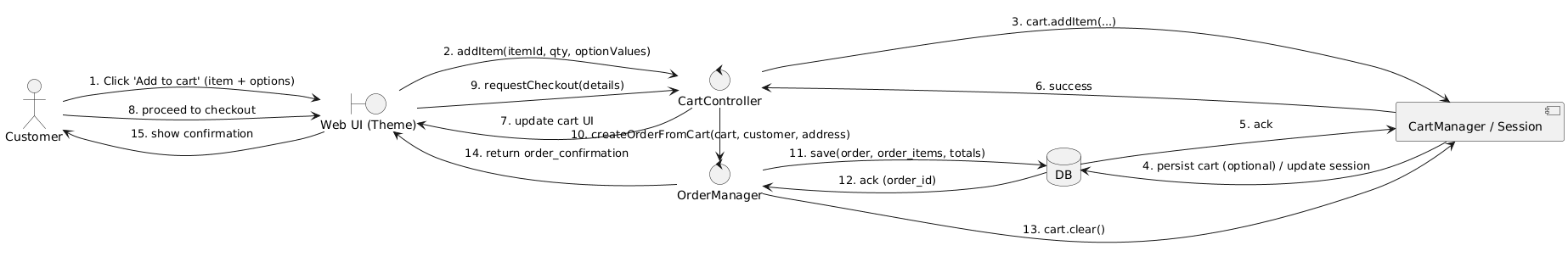
**Khung nhìn giao tiếp - communication view**

1) Product Catalog — Browse menu



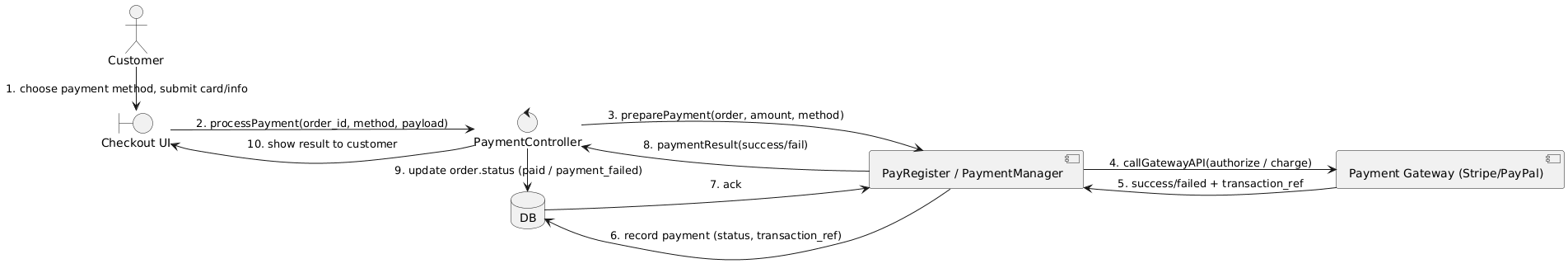
Quy trình hiển thị thực đơn bắt đầu khi khách hàng truy cập trang danh mục, lúc này giao diện Web UI sẽ gửi yêu cầu requestMenu (kèm các tham số lọc nếu có) đến MenuController. Tại đây, Controller sẽ gọi xuống MenuService để thực hiện logic nghiệp vụ, tiến hành truy vấn dữ liệu từ Database (hoặc Cache) để lấy về danh sách món ăn, danh mục và các tùy chọn đi kèm. Dữ liệu thô sau khi lấy được sẽ đóng gói thành các đối tượng DTO (Data Transfer Object) và trả ngược về cho Web UI để hiển thị trực quan lên màn hình cho người dùng.

1. Shopping Cart — Add item → Checkout



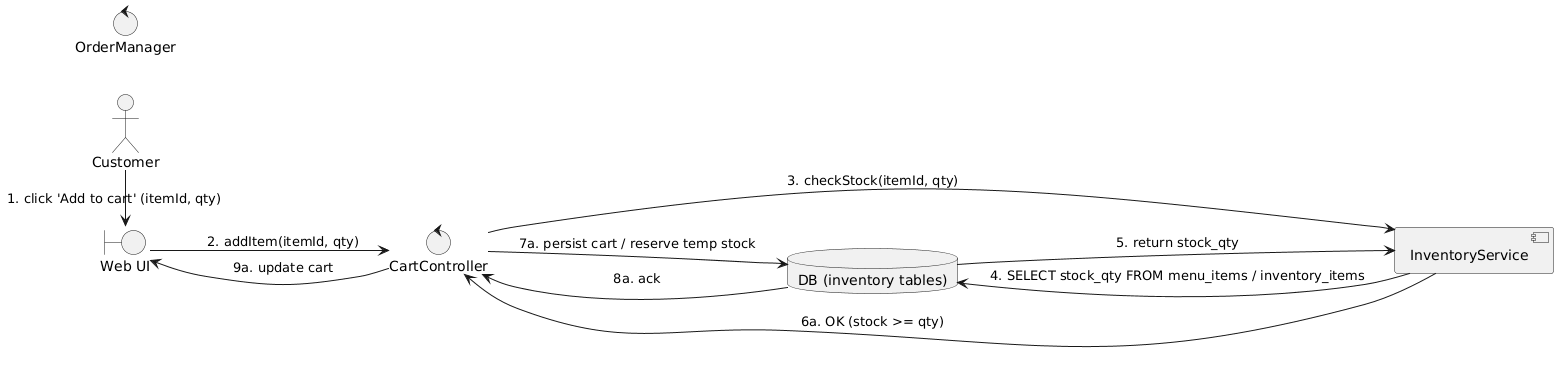
Quy trình bắt đầu khi khách hàng chọn món và nhấn "Thêm vào giỏ", hệ thống sẽ gửi yêu cầu addItem đến CartController để lưu trữ món ăn cùng các tùy chọn (Size, Topping) vào cơ sở dữ liệu giỏ hàng, đồng thời tính toán lại tổng tiền để cập nhật giao diện. Khi khách hàng quyết định nhấn "Thanh toán", CartController sẽ chuyển toàn bộ dữ liệu sang OrderManager. Thành phần này chịu trách nhiệm khởi tạo một bản ghi đơn hàng (Order) mới trong Database với trạng thái ban đầu là "Chờ xử lý", sau đó thực hiện lệnh xóa sạch dữ liệu trong giỏ hàng (cart.clear) để hoàn tất phiên mua sắm.

1. Payment Process — Checkout → Gateway



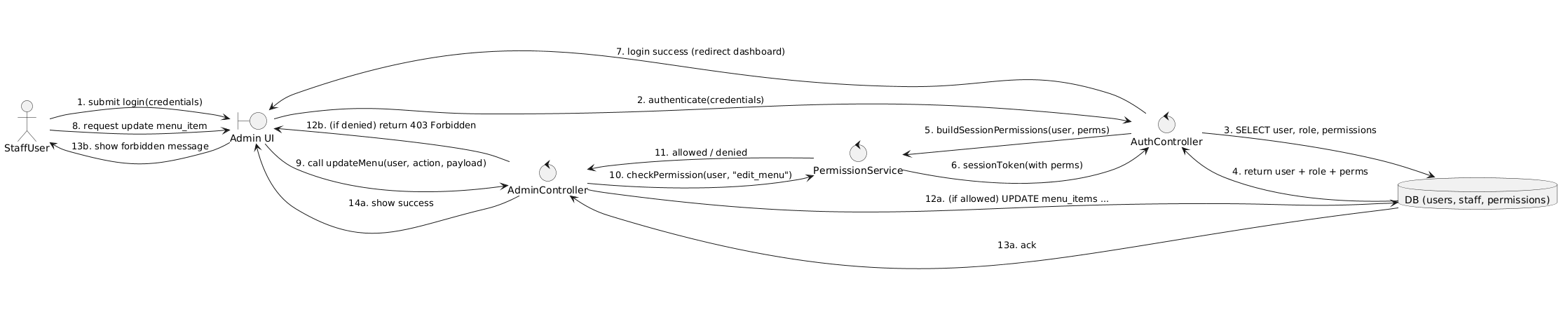
Tại bước Checkout, sau khi khách hàng nhập thông tin và xác nhận, PaymentController sẽ tiếp nhận yêu cầu và điều phối PaymentManager để xử lý giao dịch. Hệ thống sẽ thực hiện gọi API sang Cổng thanh toán bên thứ 3 (Payment Gateway như Stripe hoặc PayPal) để trừ tiền tài khoản khách hàng. Dựa trên kết quả trả về từ Gateway, nếu giao dịch thành công, hệ thống sẽ cập nhật trạng thái đơn hàng thành "Đã thanh toán" và ghi lại mã tham chiếu; ngược lại, nếu thất bại, hệ thống sẽ thông báo lỗi cho người dùng và giữ nguyên trạng thái đơn hàng là "Chờ thanh toán".

1. Inventory — Stock check & reduce on order



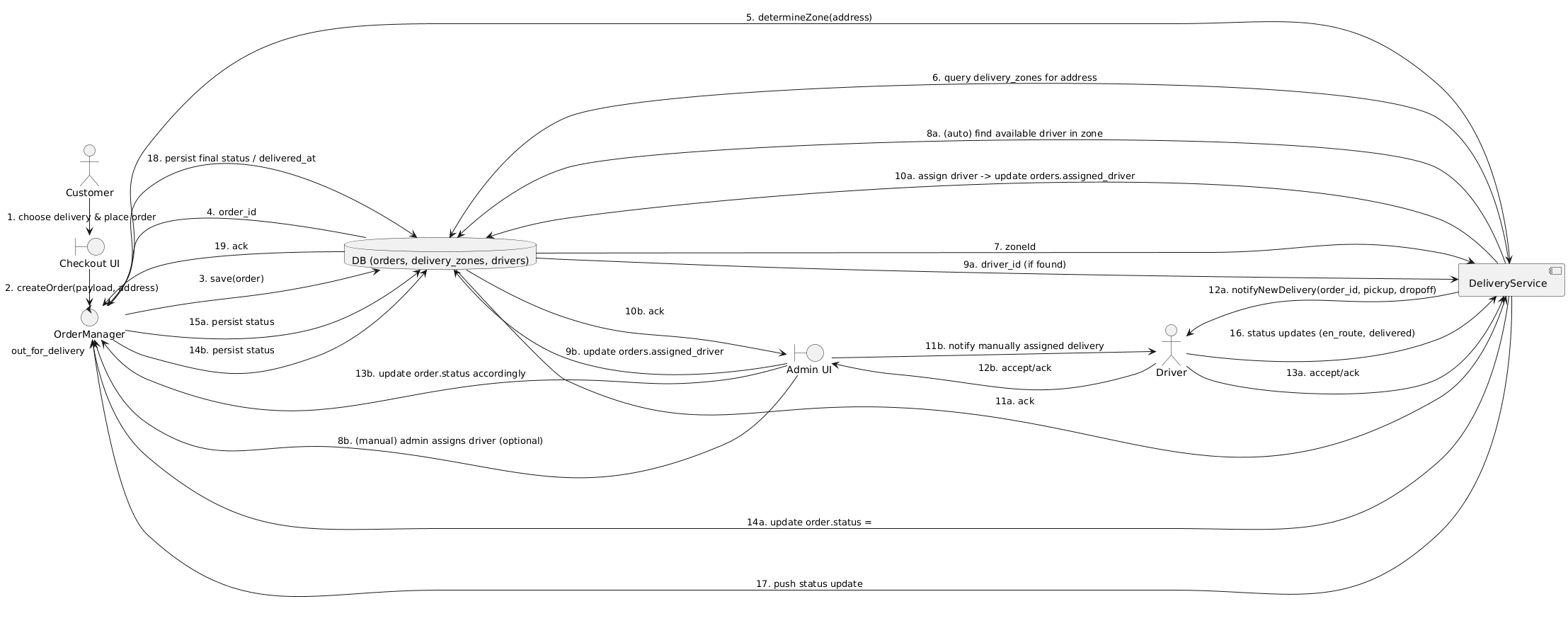
Để đảm bảo tính toàn vẹn dữ liệu kho hàng, ngay khi khách hàng thực hiện thao tác thêm món vào giỏ, CartController sẽ gọi sang InventoryService để kiểm tra số lượng tồn kho (checkStock). Hệ thống chỉ cho phép thêm món nếu số lượng tồn kho đáp ứng đủ (stock >= qty), ngược lại sẽ trả về thông báo lỗi. Sau khi đơn hàng được tạo thành công, hệ thống sẽ tự động thực hiện trừ số lượng tồn kho thực tế trong bảng dữ liệu (inventory tables) để đồng bộ hóa dữ liệu cho các lần mua tiếp theo.

1. Access Control — Roles & Staff actions



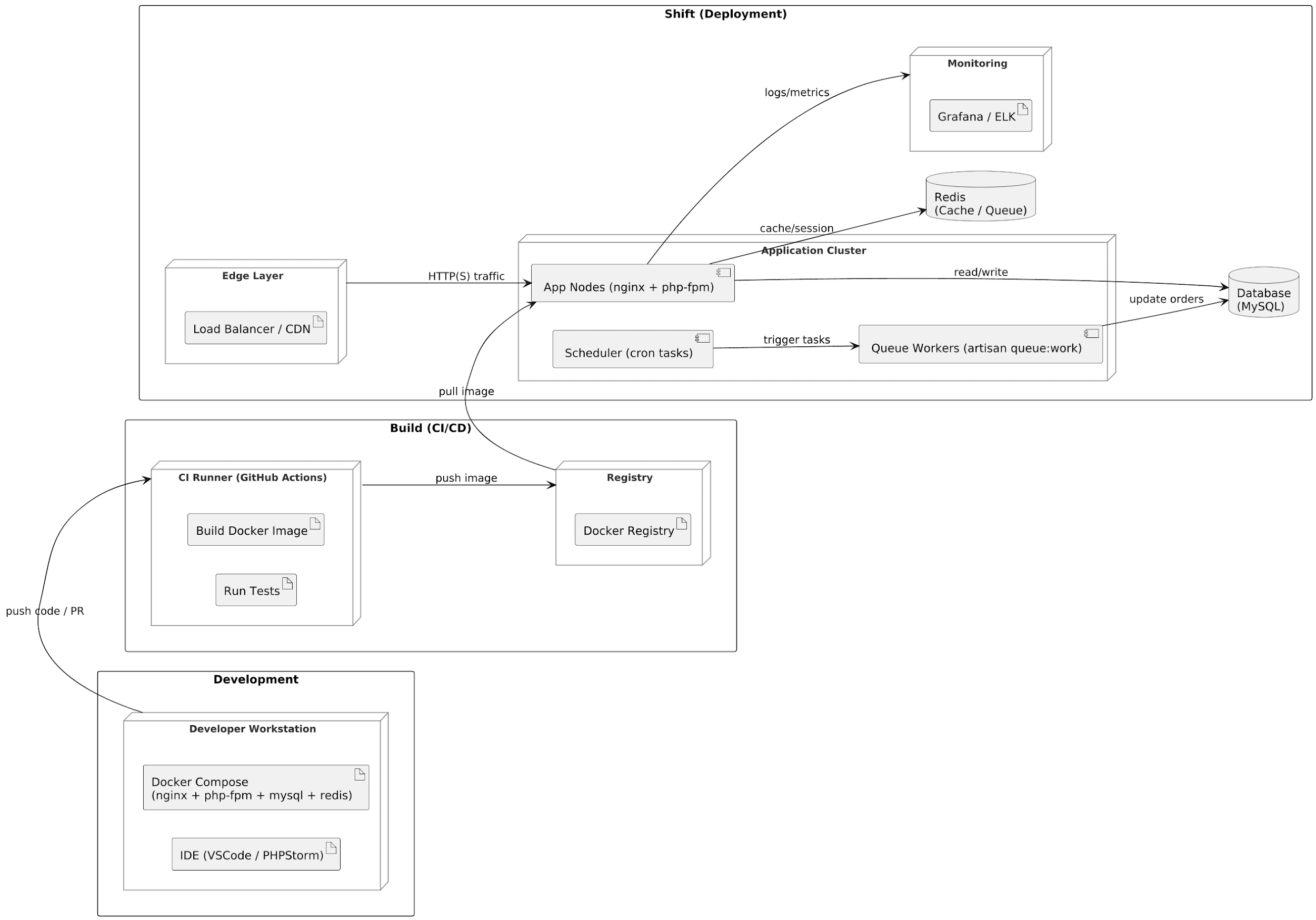
Khi người dùng thực hiện đăng nhập, AuthController sẽ tiếp nhận và đối chiếu thông tin xác thực (credentials) với dữ liệu trong Database. Nếu thông tin chính xác, PermissionService sẽ tiến hành tải danh sách quyền hạn tương ứng với Vai trò (Role) của người dùng đó (ví dụ: Staff, Admin hay Customer). Trong quá trình sử dụng, mỗi khi người dùng thực hiện một hành động nhạy cảm (như Chỉnh sửa Menu), hệ thống sẽ kích hoạt kiểm tra quyền (checkPermission); nếu tài khoản không đủ thẩm quyền, hệ thống sẽ chặn yêu cầu và trả về lỗi 403 Forbidden để bảo mật.

1. Delivery — Assign order → driver → update tracking



Khi người dùng thực hiện đăng nhập, AuthController sẽ tiếp nhận và đối chiếu thông tin xác thực (credentials) với dữ liệu trong Database. Nếu thông tin chính xác, PermissionService sẽ tiến hành tải danh sách quyền hạn tương ứng với Vai trò (Role) của người dùng đó (ví dụ: Staff, Admin hay Customer). Trong quá trình sử dụng, mỗi khi người dùng thực hiện một hành động nhạy cảm (như Chỉnh sửa Menu), hệ thống sẽ kích hoạt kiểm tra quyền (checkPermission); nếu tài khoản không đủ thẩm quyền, hệ thống sẽ chặn yêu cầu và trả về lỗi 403 Forbidden để bảo mật.

**Khung nhìn triển khai - deployment view**



3: Test Plan

3.1: Purpose

Tài liệu Test Plan này được xây dựng nhằm xác định phạm vi, chiến lược và kế hoạch kiểm thử cho Website Bán Thức Ăn Nhanh phát triển dựa trên kiến trúc **Microservices** sử dụng công nghệ **Node.js và ReactJS** nhằm hỗ trợ các mục tiêu sau:

* Xác định thông tin dự án, các thành phần phần mềm và module cần được kiểm thử.
* Liệt kê các yêu cầu kiểm thử chính (Requirements for Test).
* Đề xuất và mô tả các chiến lược, kỹ thuật kiểm thử được sử dụng.
* Xác định nguồn lực, môi trường, và công cụ kiểm thử cần thiết.
* Cung cấp danh sách các sản phẩm bàn giao (Deliverables) liên quan đến hoạt động kiểm thử.

## 3.2: Background

Hệ thống được phát triển là một website bán thức ăn nhanh trực tuyến, cho phép khách hàng xem danh mục món ăn, thêm vào giỏ hàng, thanh toán trực tuyến và theo dõi trạng thái giao hàng.

Hệ thống được xây dựng theo kiến trúc Microservices, sử dụng Node.js (Express) cho Backend, ReactJS cho Frontend và cơ sở dữ liệu MongoDB, đảm bảo khả năng mở rộng và hiệu năng cao

## 3.3: Scope

Tài liệu này mô tả kế hoạch kiểm thử áp dụng cho toàn bộ quá trình phát triển của hệ thống, bao gồm các bản dựng (Build), các bản thử nghiệm (Release Candidate), và bản phát hành chính thức (Release).

Phạm vi kiểm thử bao gồm tất cả các chức năng chính được mô tả trong tài liệu yêu cầu phần mềm (SRS)

**3.4: Project Identification**

Bảng dưới đây liệt kê các tài liệu và nguồn thông tin liên quan được sử dụng trong quá trình xây dựng kế hoạch kiểm thử:

| **Tài liệu** | **Đã tạo hay có sẵn** | **Đã nhận hoặc xem xét** | **Tác giả hoặc Nguồn** | **Ghi chú** |
| --- | --- | --- | --- | --- |
| Phân tích Yêu cầu Nghiệp vụ |  Yes  No |  Yes  No |  | Tài liệu chính mô tả bối cảnh , các đối tượng , hoạt động nghiệp vụ , và kết quả mong đợi. |
| Mô hình Khái niệm (Conceptual Model) |  Yes  No |  Yes  No |  | Cung cấp thiết kế mô hình dữ liệu và mối quan hệ giữa các thực thể trong hệ thống. |
| Biểu đồ Use Case |  Yes  No |  Yes  No |  | Mô tả trực quan các tương tác cấp cao giữa tác nhân (Customer, Admin) và các phân hệ chính của website. |
| Câu chuyện Người dùng (User Stories) |  Yes  No |  Yes  No |  | Định nghĩa chi tiết các yêu cầu chức năng từ góc nhìn người dùng, làm cơ sở để viết các trường hợp kiểm thử. |
| Kế hoạch Dự án |  Yes  No |  Yes  No |  | Cung cấp thông tin về lịch trình, nguồn lực và các mốc thời gian của dự án. |
| Kịch bản & Trường hợp kiểm thử (Test Scenario & Case) | ** Yes  No** | ** Yes  No** |  | Các bước chi tiết để kiểm tra một chức năng hoặc một luồng nghiệp vụ. |
| Checklist Đánh giá Test Case | ** Yes  No** | ** Yes  No** |  | Công cụ để đảm bảo chất lượng và độ bao phủ của các trường hợp kiểm thử. |

## 3.5: References

Bảng dưới đây liệt kê các tài liệu tham khảo được sử dụng trong quá trình xây dựng kế hoạch kiểm thử cho dự án Website Bán Thức Ăn Nhanh.  
Những tài liệu này cung cấp thông tin nền tảng về yêu cầu hệ thống, thiết kế kỹ thuật và tài liệu hướng dẫn của nền tảng được sử dụng làm cơ sở phát triển.

| **No.** | **Tài liệu tham chiếu** | **Mô tả chi tiết** |
| --- | --- | --- |
| **1** | **Documentation** | Tài liệu hướng dẫn chính thức của nền tảng, bao gồm quy trình cài đặt, cấu hình hệ thống, quản lý menu, giỏ hàng, thanh toán và giao hàng. Đây là nguồn tham khảo kỹ thuật quan trọng giúp nhóm phát triển và kiểm thử hiểu rõ luồng xử lý chuẩn của framework, từ đó định nghĩa và thiết kế các kịch bản test phù hợp. |
| **2** | **Source Code** | Kho mã nguồn mở chính của nền tảng, được sử dụng làm cơ sở cho hệ thống bán thức ăn nhanh. Mã nguồn cung cấp cấu trúc thư mục, module, và API nội bộ, giúp nhóm kiểm thử xác định logic hoạt động, dependencies và các điểm cần kiểm tra trong quá trình integration testing. |
| **3** | **Phân tích Yêu cầu Nghiệp vụ Website Bán Thức Ăn Nhanh** | Tài liệu phân tích nghiệp vụ gốc của nhóm sinh viên, mô tả chi tiết bối cảnh hệ thống, đối tượng sử dụng (Buyer, SysAdmin, Delivery Driver), và các luồng nghiệp vụ chính như: duyệt món, giỏ hàng, thanh toán, giao hàng và quản lý kho. Đây là tài liệu cơ sở quan trọng dùng để xác định phạm vi kiểm thử, use case chính và kịch bản test cho toàn dự án. |

## 3.6: Features to be Tested

Danh sách các chức năng cần kiểm thử được xác định dựa trên yêu cầu nghiệp vụ (Business Requirements), tài liệu phân tích hệ thống, và cấu trúc của nền tảng.  
Các chức năng này bao gồm các phần **chính của quy trình mua hàng trực tuyến**, từ khi người dùng truy cập trang web đến khi đơn hàng được giao thành công.

### 3.6.1: Functional Testing

**Product Catalog (Danh mục món ăn)**

* Xác minh rằng người dùng có thể xem danh sách món ăn theo danh mục.
* Kiểm tra khả năng lọc món theo tên, giá, hoặc danh mục cụ thể.
* Kiểm tra chức năng sắp xếp món theo giá tăng/giảm hoặc theo tên.
* Xác minh thông tin chi tiết món ăn được hiển thị đầy đủ (tên, mô tả, hình ảnh, tình trạng tồn kho).
* Kiểm tra hiển thị tồn kho chính xác theo dữ liệu thực tế trong hệ thống.

**Shopping Cart (Giỏ hàng)**

* Xác minh khả năng thêm món vào giỏ hàng từ danh mục hoặc trang chi tiết món.
* Kiểm tra cập nhật số lượng món (tăng/giảm).
* Kiểm tra xóa món khỏi giỏ hàng và cập nhật tổng giá trị đơn hàng tương ứng.
* Xác minh tính toán tổng tiền, phí giao hàng, khuyến mãi, và tổng cộng chính xác.
* Kiểm tra điều kiện không cho phép checkout khi giỏ hàng trống.

**Payment Process (Thanh toán)**

* Xác minh quy trình thanh toán diễn ra đúng khi có dữ liệu hợp lệ.
* Kiểm tra xử lý khi sản phẩm hết hàng hoặc không hợp lệ.
* Xác minh trạng thái thanh toán thay đổi chính xác (pending → processed → failed).
* Kiểm tra email xác nhận được gửi đến khách hàng khi giao dịch thành công.
* Kiểm tra xử lý lỗi và thông báo khi thanh toán thất bại.

**Access Control (Đăng nhập & Phân quyền)**

* Xác minh người dùng có thể đăng nhập bằng tài khoản hợp lệ.
* Kiểm tra phân quyền theo vai trò (Buyer, SysAdmin).
* Kiểm tra chức năng đăng xuất và chuyển hướng đúng trang.
* Xác minh Buyer không truy cập được trang quản trị, và ngược lại.

### 3.6.2: Usability Testing

* Kiểm tra giao diện người dùng (UI) hiển thị rõ ràng, dễ thao tác.
* Kiểm tra khả năng phản hồi (responsive design) trên các thiết bị khác nhau.
* Xác minh luồng người dùng hợp lý (duyệt món → giỏ hàng → thanh toán → giao hàng).
* Kiểm tra thông báo lỗi và xác nhận hiển thị đúng ngữ cảnh.

### 3.6.3: Design Constraints

**Server / Hạ tầng (Environment Constraints)**

* Hệ thống yêu cầu Node.js phiên bản 18.x trở lên (LTS).
* Cơ sở dữ liệu sử dụng MongoDB (hỗ trợ Mongoose ODM) phiên bản 6.0 trở lên.
* Sử dụng NPM hoặc Yarn để quản lý các gói thư viện (dependencies) cho cả Backend và Frontend.
* Máy chủ cần hỗ trợ Containerization (Docker) để vận hành các service độc lập (User, Food, Order...) và API Gateway.

• **Giới hạn dữ liệu & Ký tự (Data / Character Constraints)**

* Một số trường dữ liệu như mô tả món, tên món ăn cần có giới hạn độ dài hợp lý để tránh vượt quá giới hạn database hoặc gây lỗi UI.

**Khả năng tương thích & Hiệu năng (Compatibility / Performance Constraints)**

* Giao diện phải tương thích với các trình duyệt phổ biến: Chrome, Firefox, Edge.
* Khi hệ thống có số lượng người dùng / đơn hàng lớn, cần sử dụng caching, queue hoặc tối ưu backend để tránh nghẽn (Laravel queue, vô hiệu debug mode khi đưa vào môi trường sản xuất)

### 3.6.4: Interfaces

**User Interface (Giao diện người dùng)**

* Giao diện Buyer: các trang chính như danh mục món, chi tiết món, giỏ hàng, thanh toán, theo dõi đơn hàng.
* Giao diện Admin / SysAdmin: quản lý menu, kho, đơn hàng, người dùng, địa chỉ giao hàng.
* Các nút, form, menu, điều hướng phải hoạt động đúng theo thiết kế UI/UX.

**Payment Gateway Interface (Giao tiếp cổng thanh toán)**

* Hệ thống giao tiếp với cổng thanh toán trong chế độ *sandbox / thử nghiệm* để kiểm thử trả về trạng thái thanh toán (success, fail, pending).
* Xác minh request / response giữa hệ thống và gateway (ví dụ: xác thực chữ ký, xử lý webhook) hoạt động đúng.

**Database Interface (Giao tiếp với cơ sở dữ liệu)**

* Các thao tác CRUD thông qua Mongoose Model cho các Collection: Users, Foods, Orders, Carts.
* Kiểm tra tính toàn vẹn dữ liệu giữa các Collection (Document References) khi thực hiện các thao tác xóa/cập nhật.
* Khi cập nhật trạng thái đơn hàng hoặc tồn kho, các trigger phải cập nhật đồng bộ cơ sở dữ liệu.

3.7: Test Strategy

3.7.1: Testing Types

**Functional and Usability Testing**

| **Mục** | **Nội dung chi tiết** |
| --- | --- |
| **Test Objective** | Đảm bảo các chức năng cốt lõi của hệ thống — bao gồm *Danh mục món ăn, Giỏ hàng, Thanh toán và Phân quyền người dùng* — hoạt động chính xác theo yêu cầu và mang lại trải nghiệm người dùng thân thiện. |
| **Technique** | • Thực thi các *use case* từ tài liệu yêu cầu nghiệp vụ. • Sử dụng dữ liệu đầu vào hợp lệ và không hợp lệ để xác minh phản hồi hệ thống. • Quan sát hành vi giao diện (UI) khi người dùng thao tác để đánh giá khả dụng (usability). Ghi nhận và theo dõi lỗi thông qua hệ thống báo cáo tự động của Katalon Studio và các bộ sưu tập kiểm thử của Postman |
| **Completion Criteria** | • Tất cả test case chức năng và khả dụng được thực thi. • Không còn lỗi nghiêm trọng (Critical/Major) tồn tại chưa sửa. • Tỷ lệ pass ≥ 90% trên tổng số test case. |
| **Special Considerations** | • Một số lỗi giao diện nhỏ (UI alignment) có thể chấp nhận tạm thời nếu không ảnh hưởng đến luồng chức năng. Cần môi trường test ổn định (Node.js Runtime, MongoDB Cluster). |

**User Interface Testing**

| **Mục** | **Nội dung chi tiết** |
| --- | --- |
| **Test Objective** | Đảm bảo giao diện người dùng hiển thị đúng thiết kế, căn chỉnh chính xác, hoạt động ổn định và tương thích với nhiều thiết bị, trình duyệt. |
| **Technique** | • So sánh giao diện thực tế với prototype / mockup UI. • Kiểm thử thủ công trên các trình duyệt (Chrome, Firefox, Edge) và thiết bị (PC, mobile). • Kiểm tra khả năng phản hồi (responsive design). |
| **Completion Criteria** | • 100% các thành phần UI hiển thị đúng, không lỗi căn chỉnh hoặc tràn khung. • Các phần tử tương tác (button, input, dropdown, link) hoạt động đúng. • Không có lỗi rendering trên các độ phân giải phổ biến. |
| **Special Considerations** | • Yêu cầu truy cập bản thiết kế UI cuối cùng từ nhóm Frontend. • Cần danh sách chính thức các độ phân giải và trình duyệt được hỗ trợ để xác định phạm vi test. |

3.7.2: Tools

| **Loại công cụ** | **Tên công cụ** | **Mục đích sử dụng** |
| --- | --- | --- |
| **UI & Automation** | **Katalon Studio, Cypress** | Kiểm thử tự động luồng chức năng và E2E (End-to-End) trên giao diện Web. |
| **API & Interface** | **Postman, Swagger** | Kiểm thử API, đo lường hiệu năng và xác thực cấu trúc dữ liệu giao tiếp tầng dịch vụ. |
| **Database** | **MongoDB Compass** | Truy cập trực tiếp cơ sở dữ liệu để xác thực tính toàn vẹn sau kiểm thử (Grey-box). |
| **Security** | **OWASP ZAP, Burp Suite** | Quét lỗ hổng bảo mật ứng dụng và kiểm soát quyền truy cập hệ thống. |
| **Analysis** | **Chrome DevTools, npm audit** | Gỡ lỗi giao diện theo thời gian thực và kiểm tra tính an toàn của các thư viện Backend. |

3.8: Nguồn nhân lực (Human Resources)

Bảng dưới đây mô tả các nhân sự cần thiết tham gia vào hoạt động kiểm thử của dự án.

| **Vị trí** | **Số lượng đề xuất tối thiểu *(tương đương toàn thời gian – FTE)*** | **Nhiệm vụ / Mô tả công việc** |
| --- | --- | --- |
| **Trưởng nhóm kiểm thử (Test Lead / QA Coordinator)** | 1 FTE | Chịu trách nhiệm lập kế hoạch và điều phối toàn bộ hoạt động kiểm thử. **Nhiệm vụ:** • Xây dựng và duy trì Kế hoạch kiểm thử (Test Plan). • Xác định phạm vi, chiến lược, và mức độ ưu tiên kiểm thử. • Rà soát test case và giám sát tiến độ thực hiện. • Tổng hợp kết quả, lập báo cáo kiểm thử cuối cùng. • Là cầu nối giữa nhóm kiểm thử và nhóm phát triển. |
| **Nhân viên kiểm thử (Tester)** | 1 FTE | Thực hiện các trường hợp kiểm thử và ghi nhận kết quả. **Nhiệm vụ:** • Thiết kế và thực thi test case thủ công cho các module: Danh mục món, Giỏ hàng, Thanh toán. • Ghi nhận lỗi vào báo cáo kiểm thử của Katalon Studio và Postman, sau đó phối hợp với lập trình viên để xác minh sau khi sửa. • Thực hiện kiểm thử hồi quy (Regression) sau mỗi lần cập nhật mã nguồn. • Kiểm tra giao diện người dùng (UI) và phản hồi của hệ thống. |
| **Lập trình viên / Hỗ trợ kỹ thuật (Developer / Technical Support)** | 1 FTE | Hỗ trợ kỹ thuật và thiết lập môi trường kiểm thử. **Nhiệm vụ:** Cài đặt và cấu hình môi trường Microservices (Node.js, Express, MongoDB, Docker). • Hỗ trợ nhóm kiểm thử xử lý lỗi kỹ thuật hoặc backend. • Thực hiện sửa lỗi và triển khai các bản build mới để kiểm thử. |

3.9: Các mốc tiến độ dự án (Project Milestones)

Phần này mô tả tiến độ và khối lượng công việc dự kiến cho hoạt động kiểm thử của dự án.  
Kế hoạch kiểm thử được thực hiện trong **8 tuần**, chia thành 5 giai đoạn (Build).  
Mỗi giai đoạn bao gồm bốn bước chính: **Lập kế hoạch (Plan Test) → Thiết kế (Design Test) → Thực thi (Execute Test) → Đánh giá (Evaluate Test)**.

| **Công việc / Giai đoạn (Milestone Task)** | **Nỗ lực (Effort – ngày)** | **Ngày bắt đầu (Start Date)** | **Ngày kết thúc (End Date)** |
| --- | --- | --- | --- |
| **Build 1 – Lập kế hoạch & chuẩn bị môi trường**  • Plan Test  • Design Test  • Execute Test  • Evaluate Test | 10 | 06/10/2025 | 17/10/2025 |
| **Build 2 – Kiểm thử chức năng & giao diện (Functional & UI Test)**  • Plan Test  • Design Test  • Execute Test  • Evaluate Test | 12 | 20/10/2025 | 31/10/2025 |
| **Build 3 – Kiểm thử tích hợp & hồi quy (Integration & Regression Test)**  • Plan Test  • Design Test  • Execute Test  • Evaluate Test | 12 | 03/11/2025 | 14/11/2025 |
| **Release Candidate – Phiên bản kiểm thử hoàn thiện**  • Plan Test  • Design Test  • Execute Test  • Evaluate Test | 11 | 17/11/2025 | 28/11/2025 |
| **Release Version – Phiên bản phát hành cuối cùng**  • Plan Test  • Design Test  • Execute Test  • Evaluate Test | 11 | 01/12/2025 | 12/12/2025 |

3.10: Deliverables

Liệt kê các sản phẩm đầu ra chính của quá trình kiểm thử, được xây dựng và cập nhật xuyên suốt vòng đời dự án:

* **Test Plan:**  
  Kế hoạch kiểm thử mô tả phạm vi, mục tiêu, chiến lược, tiến độ, và công cụ kiểm thử được sử dụng trong dự án.
* **Test Cases (TC):**  
  Bộ trường hợp kiểm thử chi tiết được thiết kế dựa trên các User Story, bao gồm các module chính: Danh mục món, Giỏ hàng, Thanh toán, Giao hàng, và Tồn kho.
* **Test Logs:**  
  Bản ghi kết quả thực thi kiểm thử, bao gồm dữ liệu đầu vào, đầu ra mong đợi, kết quả thực tế và trạng thái kiểm thử (Pass/Fail).  
  Kết quả được ghi nhận thủ công trong Excel/Word và tự động lưu trong GitHub Actions đối với kiểm thử tự động.
* **Bug Reports:**  
  Danh sách lỗi được phát hiện trong quá trình kiểm thử, được ghi nhận và theo dõi qua **MantisBT** hoặc **GitHub Issues**, kèm thông tin mức độ nghiêm trọng, bước tái hiện và người phụ trách khắc phục.
* **Test Report (Test Summary Report):**  
  Báo cáo tổng kết toàn bộ hoạt động kiểm thử, thống kê số lượng test case đã chạy, tỷ lệ Pass/Fail, mức độ bao phủ và đánh giá chất lượng hệ thống trước khi phát hành.

4. Thiết kế kiểm thử

4.1. Chiến lược và Phương pháp luận Thiết kế Kiểm thử

Tiến trình thiết kế kiểm thử cho hệ thống Website Fast Food Delivery (UMA) được xây dựng dựa trên sự đồng bộ hóa giữa quy trình phát triển và quy trình bảo soát chất lượng. Nhóm đã quyết định áp dụng mô hình **V-Model (Mô hình chữ V)** làm khung tham chiếu cốt lõi. Trong mô hình này, các hoạt động kiểm thử được lập kế hoạch và thiết kế song song với các giai đoạn phân tích đặc tả và thiết kế hệ thống. Cụ thể, các tiêu chí chấp nhận (Acceptance Criteria) được xác định ngay từ giai đoạn thu thập User Story, giúp thu hẹp khoảng cách giữa kỳ vọng của người dùng và sản phẩm thực tế.

Về mặt kỹ thuật, dự án triển khai đa dạng các phương pháp luận kiểm thử để tối ưu hóa khả năng phát hiện lỗi ở nhiều tầng kiến trúc khác nhau:

**Phương pháp Kiểm thử hộp đen (Black-box Testing):** Đây là kỹ thuật trọng tâm được sử dụng để xác thực các yêu cầu chức năng và phi chức năng của hệ thống. Nhóm thiết kế các kịch bản kiểm thử dựa trên hành vi người dùng cuối (End-user behavior) và các đặc tả nghiệp vụ. Kiểm thử viên tập trung vào việc xử lý các giá trị đầu vào (Input) và đối chiếu kết quả trả về (Output) mà không can thiệp vào cấu trúc mã nguồn, qua đó đánh giá được tính toàn vẹn của các luồng nghiệp vụ như đặt hàng và thanh toán.

**Phương pháp Kiểm thử hộp trắng (White-box Testing):** Phương pháp này được triển khai nhằm kiểm soát độ bao phủ của mã nguồn (Code coverage). Nhóm tập trung phân tích các cấu trúc điều khiển, các nhánh logic và các câu lệnh điều kiện trong Backend nhằm đảm bảo mọi luồng xử lý bên trong đều vận hành chính xác. Điều này giúp loại bỏ các đoạn mã dư thừa và đảm bảo tính tối ưu trong quá trình xử lý dữ liệu của các Controller và Model.

**Phương pháp Kiểm thử hộp xám (Grey-box Testing):** Nhóm vận dụng kiểm thử hộp xám để tối ưu hóa quá trình xác thực dữ liệu tại tầng trung gian. Phương pháp này kết hợp giữa việc thao tác trên giao diện người dùng (Hộp đen) và sự hiểu biết về cấu trúc lược đồ (Schema) của cơ sở dữ liệu MongoDB (Hộp trắng). Kỹ thuật này đóng vai trò quyết định trong việc đảm bảo rằng các tương tác từ người dùng được ánh xạ chính xác vào cơ sở dữ liệu, đặc biệt là các thông tin nhạy cảm như lịch sử giao dịch và dữ liệu người dùng.

4.2. Phân loại và Phạm vi Kiểm thử áp dụng

Dựa trên các phương pháp luận đã thiết lập, quy trình kiểm thử hệ thống được phân rã thành năm loại hình chính, mỗi loại hình tập trung vào một khía cạnh chất lượng cụ thể của hệ thống Website:

4.2.1. Kiểm thử chức năng (Functional Testing)

Kiểm thử chức năng đóng vai trò xác thực tính đúng đắn của các quy tắc nghiệp vụ (Business Rules). Nhóm áp dụng phương pháp hộp đen để kiểm tra toàn bộ vòng đời của một đơn hàng, từ giai đoạn khởi tạo tài khoản, lựa chọn món ăn theo danh mục, quản lý giỏ hàng cho đến khi hoàn tất thanh toán. Các kỹ thuật như Phân tích giá trị biên và Phân vùng tương đương được sử dụng để thiết kế các kịch bản cho các trường dữ liệu như giá sản phẩm, số lượng món và định dạng email người dùng.

4.2.2. Kiểm thử độ khả dụng (Usability Testing)

Mục tiêu của kiểm thử độ khả dụng là đánh giá sự tương tác giữa người dùng và hệ thống. Nhóm tập trung vào việc đo lường mức độ dễ dàng trong thao tác, tính trực quan của giao diện UI và sự mạch lạc của trải nghiệm UX. Các kịch bản kiểm thử được xây dựng xung quanh khả năng tìm kiếm món ăn, tốc độ thực hiện đơn hàng và sự hiệu quả của các phản hồi (Feedback) từ hệ thống như thông báo lỗi hoặc thông báo đặt hàng thành công.

4.2.3. Kiểm thử tương thích (Compatibility Testing)

Do tính chất đa dạng của môi trường người dùng, kiểm thử tương thích được thực hiện nhằm đảm bảo Website UMA vận hành ổn định trên nhiều cấu hình trình duyệt và hệ điều hành. Nhóm thực hiện kiểm thử trên các trình duyệt tiêu chuẩn như Google Chrome, Microsoft Edge và trên hai hệ điều hành chính là Windows và macOS. Trọng tâm là xác thực tính phản hồi (Responsiveness) của giao diện và sự tương thích của các mã kịch bản (Scripting) trên các nền tảng khác nhau.

4.2.4. Kiểm thử cơ sở dữ liệu (Database Testing)

A screenshot of a computer

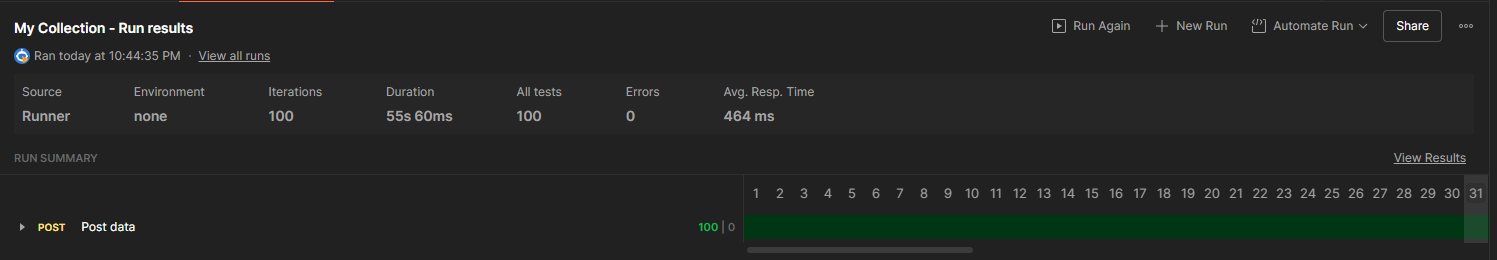
AI-generated content may be incorrect.Kiểm thử cơ sở dữ liệu được triển khai theo phương pháp hộp xám để xác thực tính toàn vẹn và nhất quán của dữ liệu. Nhóm thực hiện các kịch bản kiểm thử liên quan đến việc lưu trữ Order ID, đối chiếu các tham chiếu (References) giữa bảng User và bảng Order, cũng như kiểm tra tính chính xác của dữ liệu trong các trường lưu trữ giá trị đơn hàng. Việc kiểm tra này đảm bảo cơ sở

dữ liệu MongoDB luôn phản ánh đúng trạng thái giao dịch thực tế trên website.

Nhiều user load menu món ăn, break ở khoảng 100

4.2.5. Kiểm thử hiệu năng (Performance Testing)

Sử dụng phương pháp kiểm thử hộp đen ở tầng API, nhóm tiến hành đo lường khả năng xử lý của hệ thống dưới áp lực tải. Các kịch bản được thiết kế để xác định thời gian phản hồi trung bình của máy chủ, khả năng xử lý các yêu cầu đồng thời (Concurrency) và xác định ngưỡng chịu tải tối đa (Stress point) của hạ tầng Render. Điều này giúp dự báo các vấn đề tiềm ẩn khi số lượng người dùng tăng đột biến trong thực tế.

Gửi Post request đăng ký/đăng nhập đến API để kiểm tra database

4.2.6. Kiểm thử giao tiếp API (Interface Testing)

Sử dụng phương pháp **Hộp xám** để kiểm tra luồng trao đổi dữ liệu giữa các thành phần trong hệ thống. Nhóm tập trung xác thực dữ liệu gửi đi từ Frontend đến Backend và ngược lại thông qua các API như /admin/add-food, /user/login. Kỹ thuật này giúp phát hiện lỗi sai lệch định dạng JSON hoặc thiếu sót dữ liệu trong quá trình truyền tải.

4.2.7. Kiểm thử bảo mật (Security Testing)

Áp dụng kết hợp phương pháp **Hộp đen** và **Hộp xám** để tìm kiếm các lỗ hổng tiềm ẩn. Các kịch bản tập trung vào việc ngăn chặn tấn công SQL Injection, XSS, kiểm tra cơ chế phân quyền JWT và chính sách hết hạn của Token để bảo vệ thông tin cá nhân khách hàng.

4.3. Công cụ kiểm thử và Vai trò trong Hệ thống

Để tối ưu hóa quy trình kiểm soát chất lượng theo mô hình **V-Model**, dự án đã lựa chọn một bộ công cụ đa dạng, hỗ trợ từ kiểm thử thủ công đến tự động hóa trên nhiều tầng kiến trúc khác nhau.

4.3.1. Nhóm công cụ Kiểm thử Giao diện (UI Testing - Black-box)

Nhóm công cụ này tập trung vào việc xác thực trải nghiệm người dùng cuối mà không cần can thiệp vào logic xử lý nội bộ của hệ thống.

**Katalon Studio:** Đóng vai trò là nền tảng chính cho kiểm thử tự động (Automation Testing), giúp xây dựng các bộ kịch bản kiểm thử hồi quy (Regression Testing) cho các tính năng cốt lõi như Đặt hàng và Giỏ hàng.

**Cypress:** Được sử dụng để thực hiện kiểm thử E2E (End-to-End) cho Frontend ReactJS, giúp mô phỏng chính xác hành vi người dùng trên trình duyệt với tốc độ thực thi cao.

**Chrome DevTools:** Hỗ trợ kiểm thử viên rà soát cấu trúc DOM, tinh chỉnh CSS trực tiếp và theo dõi tab Network để kiểm tra các tài nguyên được tải lên giao diện.

4.3.2. Nhóm công cụ Kiểm thử Giao tiếp (Interface & Database Testing - Grey-box)

Nhóm công cụ này hỗ trợ phương pháp kiểm thử hộp xám, cho phép kiểm thử viên vừa thao tác trên bề mặt, vừa quan sát được luồng dữ liệu bên dưới.

**Postman:** Công cụ chủ đạo để kiểm tra các Endpoint API, xác thực mã trạng thái (Status Code), cấu trúc phản hồi JSON và thực hiện các bài kiểm tra hiệu năng (Performance) giả lập 500+ người dùng.

**Swagger:** Cung cấp tài liệu API trực quan, giúp nhóm kiểm thử hiểu rõ các tham số đầu vào và đầu ra cần thiết để thiết kế kịch bản Interface Testing chính xác.

**MongoDB Compass:** Công cụ trực quan hóa cơ sở dữ liệu, cho phép kiểm thử viên truy cập trực tiếp vào các Collection như Users hay Orders để xác minh dữ liệu đã được lưu trữ đúng sau khi thực hiện các hành động trên website.

4.3.3. Nhóm công cụ Kiểm thử Bảo mật (Security Testing - Black-box & Grey-box)

Mục tiêu của nhóm này là phát hiện các lỗ hổng hệ thống và đảm bảo an toàn cho thông tin khách hàng.

**OWASP ZAP & Burp Suite:** Hai công cụ chuyên dụng để quét lỗ hổng bảo mật tự động và can thiệp proxy. Chúng giúp phát hiện các nguy cơ tấn công phổ biến như SQL Injection, Cross-Site Scripting (XSS) và kiểm tra tính hợp lệ của cơ chế phân quyền JWT.

**npm audit:** Được sử dụng để quét các thư viện bên thứ ba (Dependencies) trong Node.js, giúp phát hiện sớm các rủi ro bảo mật từ các mã nguồn mở được tích hợp vào dự án.

5. Kết luận

5.1. Kết luận tổng thể

Sau quá trình nghiên cứu, xây dựng kế hoạch và triển khai thực thi kiểm thử cho hệ thống Website Fast Food Delivery (UMA), nhóm đã hoàn thành xuất sắc các mục tiêu chiến lược đề ra. Quy trình kiểm thử được vận hành bài bản dựa trên mô hình **V-Model**, tạo ra sự kết nối chặt chẽ giữa đặc tả yêu cầu, thiết kế hệ thống và các cấp độ kiểm thử tương ứng.

Các kết quả đạt được cụ thể bao gồm:

**Về quy trình và phương pháp luận:** Nhóm đã triển khai toàn diện các phương pháp kiểm thử Hộp đen, Hộp trắng và Hộp xám. Đặc biệt, việc áp dụng kiểm thử hộp xám đã giúp kiểm soát tốt luồng dữ liệu giữa giao diện và hệ quản trị cơ sở dữ liệu MongoDB.

**Về độ bao phủ kiểm thử:** Dự án đã xác thực chất lượng trên toàn bộ các khía cạnh trọng yếu bao gồm: Chức năng, Độ khả dụng, Tương thích, Cơ sở dữ liệu, Hiệu năng, Giao tiếp API (Interface) và Bảo mật hệ thống. Các nghiệp vụ cốt lõi từ quản lý tài khoản, danh mục sản phẩm đến quy trình thanh toán Stripe đều đạt tiêu chuẩn vận hành.

**Về mặt công cụ:** Sự phối hợp nhịp nhàng giữa các công cụ hiện đại như **Katalon Studio, Postman, Cypress** và **Swagger** đã giúp tối ưu hóa hiệu suất làm việc, tăng độ bao phủ kịch bản và giảm thiểu tối đa sai sót so với phương pháp kiểm thử thủ công truyền thống.

5.2. Ưu điểm và Hạn chế

5.2.1. Ưu điểm

**Tính thực tế:** Bộ kịch bản kiểm thử (Test Cases) được xây dựng chi tiết, bám sát các câu chuyện người dùng (User Stories) và các tình huống nghiệp vụ thực tế phát sinh trong dự án.

**Tính đồng bộ:** Quy trình kiểm thử tương thích (Compatibility Testing) đảm bảo website hiển thị và hoạt động ổn định, đồng nhất trên các trình duyệt phổ biến (Chrome, Edge) và các hệ điều hành khác nhau (Windows, macOS).

**Tính kiểm soát:** Việc sử dụng các công cụ quét bảo mật và gỡ lỗi (OWASP ZAP, Chrome DevTools) giúp phát hiện sớm các lỗ hổng tầng giao diện và tầng dịch vụ.

5.2.2. Hạn chế

**Kiểm thử áp lực:** Do giới hạn về thời gian và hạ tầng máy chủ, các kịch bản kiểm thử hiệu năng chưa bao quát được hết các kịch bản tải cực đại trong thời gian dài (Stress Testing) để xác định ngưỡng chịu lỗi cuối cùng.

**Trải nghiệm người dùng:** Một số thông báo lỗi hệ thống trả về cho người dùng cuối còn mang tính tổng quát, cần được chi tiết hóa hơn để tăng tính thân thiện và hỗ trợ người dùng xử lý sự cố tốt hơn.

5.3. Hướng phát triển

Dựa trên những kết quả đã đạt được và các hạn chế còn tồn tại, nhóm đề xuất các hướng phát triển trọng tâm cho giai đoạn tiếp theo của dự án:

**Tự động hóa toàn diện:** Tiếp tục mở rộng và hoàn thiện các kịch bản tự động hóa trên **Katalon Studio** và **Cypress** để tích hợp sâu vào quy trình **CI/CD**. Điều này giúp thực hiện kiểm thử hồi quy (Regression Testing) tự động ngay khi có thay đổi mã nguồn, đảm bảo tính ổn định liên tục.

**Nâng cao bảo mật chuyên sâu:** Triển khai các kịch bản kiểm thử bảo mật nâng cao bằng **Burp Suite**, tập trung vào việc kiểm tra tính toàn vẹn của Token (JWT) và ngăn chặn các kỹ thuật tấn công phức tạp như SQL Injection hay Cross-Site Scripting (XSS).

**Tối ưu hóa hiệu năng và khả năng mở rộng:** Sử dụng các công cụ chuyên sâu hơn để thực hiện Load Testing với quy mô hàng ngàn người dùng ảo đồng thời, từ đó đưa ra các giải pháp tối ưu hóa mã nguồn Backend và cấu hình máy chủ.

**Kiểm thử giao diện lập trình (API Testing):** Tận dụng **Swagger** để chuẩn hóa tài liệu API và mở rộng các bộ kiểm thử tích hợp (Integration Testing) giữa các dịch vụ trong hệ thống Microservices.