ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

Đồ án môn học Kiểm thử phần mềm

KIỂM THỬ WEBSITE CAKE FANTASY

GVHD: Đỗ Như Tài

Sinh viên thực hiện:

3122411218 - Võ Duy Toàn

3122411059 - Lê Thanh Hùng

Lớp: DCT122C4

TPHCM, ngày 23 tháng 9 năm 2025

MỤC LỤC

MỤC LỤC HÌNH ẢNH

LỜI MỞ ĐẦU

Trong bối cảnh kỷ nguyên số bùng nổ, sự chuyển đổi từ mô hình kinh doanh truyền thống sang nền tảng trực tuyến là yêu cầu tất yếu để tồn tại và phát triển. Dự án "Xây dựng và phát triển Website Tiệm Bánh XYZ" không chỉ là một ứng dụng thương mại điện tử đơn thuần, mà còn là một nghiên cứu chuyên sâu về quy trình phát triển phần mềm toàn diện. Từ giai đoạn phân tích yêu cầu nghiệp vụ, thiết kế hệ thống, lập trình, cho đến kiểm thử và triển khai, mỗi bước đi đều được thực hiện một cách có hệ thống và khoa học. Mục tiêu của chúng tôi là tạo ra một sản phẩm không chỉ hoạt động hiệu quả mà còn mang lại trải nghiệm người dùng tối ưu, mở ra một mô hình kinh doanh số bền vững cho ngành bánh ngọt.

Trong quá trình làm báo cáo gặp nhiều khó khắn với thiếu sót kiến thức cũ và bổ sung kiến thức mới. Thời gian thực hiện thì khá ngắn nhưng cũng đủ để hoàn thành dự án. Bài báo cáo này là minh chứng cho quá trình làm, tổng hợp toàn bộ kết quả.

CHƯƠNG 1: TỔNG QUAN DỰ ÁN

1.1. Introduction

Dự án này mô tả toàn bộ quá trình phát triển và hoàn thiện một website thương mại điện tử chuyên biệt mang tên Cake Fantasy. Nền tảng được thiết kế để kết nối trực tiếp các sản phẩm bánh chất lượng với khách hàng, mang lại trải nghiệm mua sắm nhanh chóng và tiện lợi thông qua môi trường trực tuyến.

Bằng cách áp dụng các phương pháp phân tích và thiết kế và kiểm thử phần mềm chuyên nghiệp, Cake Fantasy hướng đến việc cung cấp một giải pháp thương mại điện tử ổn định, hiệu quả và thân thiện với người dùng. Mục tiêu cuối cùng là tối ưu hóa hoạt động kinh doanh, đảm bảo mọi đơn hàng được xử lý trơn tru từ khâu chọn sản phẩm đến khi giao hàng hoàn tất, qua đó nâng cao trải nghiệm mua sắm trực tuyến cho khách hàng yêu thích các sản phẩm bánh ngọt.

1.2. Hiện trạng khách hàng

1.2.1. Giới thiệu về công ty

Là công ty có chuỗi cửa hàng chuyên bán lẻ các loại bánh đóng gói, dụng cụ làm bánh, nguyên liệu làm bánh cũng như, trang trí tiệc nho nhỏ có thể tự lắp đặt nhanh chóng.

Bộ phận nhân sự và giám sát:

- Thêm quyền cho nhân viên
- Thêm quản trị viên
- Thêm nhà cung cấp và sản phẩm mới

Bộ phân chăm sóc và xử lý đơn hàng:

- Xác nhận đóng gói đơn hàng.
- Theo dõi từng giai đoạn đơn hàng

Bộ phận quản lý kho:

- Tồn kho cuối ngày.
- Tạo phiếu nhập hàng.

1.2.2 Quy trình nghiệp vụ hiện tại

Bộ phận Nhân sự và Giám sát:

Thêm quyền cho nhân viên: Định rõ và gán các vai trò, quyền hạn truy cập (ví dụ: xem, chỉnh sửa, xóa dữ liệu) trên hệ thống cho một nhân viên cụ thể dựa trên vị trí công việc và nhu cầu thực tế.

Thêm nhà cung cấp và sản phẩm mới:

1. Nhà cung cấp: Thu thập thông tin (tên, mã số thuế, địa chỉ, điều khoản thanh toán) và nhập vào hệ thống quản lý.
2. Sản phẩm mới: Nhập thông tin chi tiết (tên, mã SKU, mô tả, giá vốn, giá bán lẻ, đơn vị tính) và phân loại vào danh mục sản phẩm của công ty.

Bộ phận Xử lý Đơn hàng:

Xác nhận đóng gói đơn hàng: Sau khi hàng hóa được tập hợp và đóng gói hoàn chỉnh, nhân viên kiểm tra đối chiếu (ví dụ: scan mã vạch sản phẩm/đơn hàng) để xác nhận số lượng, chủng loại chính xác theo yêu cầu khách hàng, sau đó cập nhật trạng thái đơn hàng trên hệ thống là **"Đã đóng gói/Sẵn sàng giao"**.

Theo dõi từng giai đoạn đơn hàng: Cập nhật và theo dõi trạng thái từ khi đơn hàng được tạo **Đã xác nhận/Chuẩn bị hàng → Đang vận chuyển → Giao thành công → Hoàn tất/Hoàn trả**. Phối hợp với bộ phận vận chuyển để đảm bảo tiến độ.

Bộ phận Quản lý Kho

Tồn kho cuối ngày: Tiến hành kiểm kê, đối chiếu số lượng hàng hóa thực tế còn lại trong kho với số liệu ghi nhận trên hệ thống (sổ sách hoặc phần mềm) vào cuối mỗi ngày làm việc. Báo cáo chênh lệch (nếu có) và xác định giá trị hàng tồn kho.

Tạo phiếu nhập hàng: Lập một chứng từ ghi nhận việc hàng hóa (từ nhà cung cấp, hoặc hàng hoàn trả) được đưa vào kho. Phiếu bao gồm thông tin: số lượng, chủng loại, đơn vị cung cấp, ngày nhập, và người kiểm đếm. Sau đó, cập nhật tăng số lượng hàng hóa tương ứng vào hệ thống tồn kho.

1.3. Yêu cầu nghiệp vụ

1.3.1. Yêu cầu chức năng

Người dùng Khách hàng:

Quản lý Sản phẩm và duyệt: Khách hàng có thể duyệt, tìm kiếm, lọc sản phẩm theo danh mục/giá. Xem chi tiết sản phẩm (ảnh, mô tả, giá, tồn kho khả dụng).

Quản lý giỏ hàng: Cho phép thêm/xóa sản phẩm, cập nhật số lượng trong giỏ hàng trước khi đặt.

Quản lý Đặt hàng: Hoàn tất quy trình Check-out: Nhập thông tin giao hàng, xác nhận đơn hàng.

Thanh toán trực tuyến: Tích hợp cổng thanh toán MoMo để xử lý giao dịch an toàn. Cung cấp tùy chọn Thanh toán kiểm thử cho mục đích phát triển.

Theo dõi Đơn hàng: Xem lịch sử đơn hàng và theo dõi trạng thái cập nhật theo thời gian thực (Đang xử lý, Đã giao, Đã hủy...).

Đánh giá và nhận xét: Cho phép khách hàng đánh giá và viết nhận xét về sản phẩm đã mua để hiển thị trên trang chi tiết sản phẩm.

Người dùng Quản trị/Nhân viên:

Dashboard & Thống kê: Cung cấp Bảng điều khiển hiển thị chỉ số chính (Doanh thu, Đơn hàng, Tồn kho sắp hết, Giá trị đơn TB) và biểu đồ xu hướng doanh thu theo thời gian.

Quản lý Sản phẩm: Thêm, sửa, xóa sản phẩm, quản lý thông tin chi tiết (tên, giá vốn, giá bán, mô tả, danh mục, đơn vị tính, khối lượng đóng gói). Tích hợp Upload hình ảnh (Cloudinary).

Quản lý đơn hàng: Xem, lọc, và cập nhật trạng thái của tất cả đơn hàng. Phân biệt trạng thái đơn hàng và trạng thái thanh toán.

Quản lý tồn kho và GRN: Quản lý số lượng tồn kho thực tế. Xử lý nhập hàng thông qua Phiếu nhập hàng, theo dõi trạng thái duyệt của từng phiếu nhập.

Quản lý nhà cung cấp: Thêm, sửa, xóa hồ sơ nhà cung cấp (tên, thông tin liên hệ, địa chỉ) để phục vụ cho việc quản lý GRN.

Quản lý người dùng và phân quyền: Quản lý tài khoản Admin và Nhân viên. Thiết lập và kiểm soát Vai trò để giới hạn quyền truy cập vào các module khác nhau.

Báo cáo và xuất dữ liệu: Tạo Báo cáo Doanh thu theo khoảng thời gian và danh mục. Hỗ trợ tùy chọn Xuất dữ liệu dưới các định dạng: CSV và PDF.

In ấn: Hỗ trợ in ấn các tài liệu liên quan đến nghiệp vụ, bao gồm in mã vạch (barcode) cho sản phẩm.

1.3.2. Yêu cầu phi chức năng

Yêu cầu về Hiệu năng

- Thời gian Tải trang: Trang chủ và trang danh sách sản phẩm phải tải hoàn toàn trong vòng dưới 3 giây trên mạng 4G tiêu chuẩn.
- Tốc độ Xử lý API: Các API quan trọng (Ví dụ: Đặt hàng, Thêm sản phẩm) phải phản hồi nhanh.
- Khả năng chịu tải: Hệ thống phải duy trì hiệu suất ổn định khi có 50+ người dùng đồng thời.

Yêu cầu về Bảo mật

- Bảo mật Giao dịch: Toàn bộ giao tiếp sử dụng HTTPS/SSL. Tích hợp MoMo phải tuân thủ các quy tắc bảo mật về xử lý thanh toán.
- Xác thực & Phân quyền: Sử dụng JWT (JSON Web Token) cho xác thực phiên người dùng. Phải có kiểm soát truy cập nghiêm ngặt để đảm bảo người dùng chỉ truy cập được các tính năng theo Vai trò của họ (Admin, Employee).
- Bảo mật Dữ liệu: Mã hóa mật khẩu người dùng trong cơ sở dữ liệu MySQL.

Yêu cầu về Công nghệ & Khả dụng

- Công nghệ Khung: Hệ thống phải được xây dựng hoàn toàn bằng React 19, Node.js, Express, và MySQL như đã định nghĩa.
- Thiết kế đáp ứng: Giao diện Khách hàng (Frontend) phải hoạt động tốt trên cả máy tính để bàn và thiết bị di động.
- Dịch vụ Bên ngoài: Phải tích hợp thành công Cloudinary để lưu trữ hình ảnh sản phẩm và MoMo cho thanh toán.

1.4. Kế hoạch dự án

1.4.1. Phạm vi và mục tiêu

Mục tiêu dự án:

- Xây dựng hoàn thiện hệ thống thương mại điện tử "Cake Fantasy" trên nền tảng Web App.
- Cung cấp công cụ quản lý toàn diện cho doanh nghiệp (từ nhập kho, quản lý sản phẩm, đơn hàng đến báo cáo doanh thu).
- Mang lại trải nghiệm mua sắm trực tuyến mượt mà, nhanh chóng cho khách hàng với tính năng thanh toán điện tử tích hợp.

Phạm vi trong dự án (In-scope):

- **Phân hệ Khách hàng (Front-end):** Trang chủ, Tìm kiếm/Lọc sản phẩm, Giỏ hàng, Đặt hàng (Checkout), Thanh toán MoMo, Theo dõi đơn hàng, Đánh giá sản phẩm.
- **Phân hệ Quản trị (Back-end/Admin Portal):** Dashboard thống kê, Quản lý sản phẩm (CRUD), Quản lý đơn hàng (Order status), Quản lý kho (GRN, Tồn kho), Quản lý nhà cung cấp, Phân quyền (Admin/Employee), Báo cáo (Doanh thu, Xuất CSV/PDF), In mã vạch.
- **Tích hợp hệ thống bên thứ ba:** Cloudinary (lưu trữ ảnh), MoMo (cổng thanh toán), Thư viện tạo Barcode/PDF.

Phạm vi ngoài dự án (Out-scope):

- Ứng dụng di động (Native Mobile App) trên iOS/Android (chỉ dừng lại ở Web Responsive).
- Hệ thống Logistics phức tạp (tự động tính phí ship theo API của đơn vị vận chuyển thực tế - sử dụng phí ship cố định hoặc tự cấu hình).
- Hệ thống AI gợi ý sản phẩm phức tạp (Machine Learning).

1.4.2. Cơ sở hạ tầng

Để đảm bảo đáp ứng các yêu cầu phi chức năng về hiệu năng và công nghệ (React 19, Node.js, MySQL), dự án yêu cầu cơ sở hạ tầng như sau:

Môi trường phát triển (Development):

- **Phần cứng:** Máy tính cá nhân để chạy Localhost và các máy ảo/Docker container.
- **IDE/Editor:** Visual Studio Code (với các Extension hỗ trợ JS, React, MySQL).
- **Quản lý mã nguồn:** GitHub.

Môi trường công nghệ (Tech Stack):

- **Frontend:** React 19, CSS Framework TailwindCSS.
- **Backend:** Node.js, Express Framework.
- **Cơ sở dữ liệu:** MySQL
- **Lưu trữ (Storage):** Cloudinary
- **Thanh toán:** MoMo Developer API.

Môi trường triển khai (Deployment - Dự kiến):

- Frontend Hosting: Vercel.
- Backend Hosting: Railway.
- Database Hosting: Railway.

1.4.3. Lịch trình thực hiện

Dự án được chia thành các giai đoạn chính theo mô hình phát triển phần mềm (SDLC) với các mốc thời gian và kết quả bàn giao cụ thể:

Giai đoạn	Công việc chính	Kết quả bàn giao
Giai đoạn 1: Khởi tạo & Phân tích	- Thu thập yêu cầu chi tiết. - Phân tích quy trình nghiệp vụ (AS-IS, TO-BE). - Xác định yêu cầu chức năng/phi chức năng.	- Tài liệu đặc tả yêu cầu (SRS). - Kế hoạch dự án (Project Plan).
Giai đoạn 2: Thiết kế hệ thống	- Thiết kế cơ sở dữ. - Thiết kế kiến trúc hệ thống. - Thiết kế UI/UX.	- Bản thiết kế CSDL. - File thiết kế giao diện. - Tài liệu thiết kế kiến trúc.
Giai đoạn 3: Phát triển	- **Sprint 1:** Dựng khung dự án (Skeleton), Database Migration, Auth (JWT). - **Sprint 2:** Module Quản lý Sản phẩm, Danh mục, Nhà cung cấp, Cloudinary. - **Sprint 3:** Module Mua hàng (Frontend), Giỏ hàng, Thanh toán MoMo. - **Sprint 4:** Module Quản lý Đơn hàng, Kho (GRN), Báo cáo, In ấn.	- Source code hoàn chỉnh. - API Documentation.
Giai đoạn 4: Kiểm thử	- Viết Test Case. - Thực hiện Unit Test, Integration Test.	- Bộ Test Case & Test Plan. - Báo cáo lỗi (Bug Report).

	- Kiểm thử luồng thanh toán MoMo. - UAT (User Acceptance Testing).	- Kết quả kiểm thử (Test Report).
Giai đoạn 5: Triển khai & Đóng gói	- Triển khai lên server thực tế. - Hướng dẫn sử dụng. - Tổng kết dự án.	- Website hoạt động (Live URL). - Tài liệu hướng dẫn sử dụng (User Guide). - Báo cáo tổng kết.

1.4.4. Quản lý cấu hình

Để đảm bảo tính nhất quán và dễ dàng bảo trì trong quá trình phát triển, dự án áp dụng các quy tắc quản lý cấu hình sau:

Quản lý phiên bản:

Sử dụng **Github** để quản lý mã nguồn.

Mô hình rẽ nhánh (Branching Model):

- main: Mã nguồn ổn định, sẵn sàng triển khai (Production).
- develop: Mã nguồn đang phát triển, tích hợp các tính năng mới.
- feature/ten-tinh-nang: Nhánh riêng cho từng chức năng (ví dụ: feature/payment-momo, feature/inventory-grn).

Quản lý biến môi trường:

Sử dụng tệp .env để lưu trữ các thông tin nhạy cảm và cấu hình động, không commit trực tiếp lên Git.

Các biến quan trọng:

- DB_HOST, DB_USER, DB_PASS: Kết nối MySQL.
- JWT_SECRET: Khóa bí mật cho xác thực token.
- CLOUDINARY_URL: API key cho dịch vụ ảnh.
- MOMO_PARTNER_CODE, MOMO_ACCESS_KEY: Cấu hình thanh toán.

Quản lý thư viện phụ thuộc:

Sử dụng npm hoặc yarn để quản lý các gói thư viện.

Phiên bản các thư viện được khóa trong package-lock.json để đảm bảo mọi thành viên trong nhóm dev chạy trên cùng một môi trường phiên bản.

1.5. Các ràng buộc hệ thống

Ràng buộc về Kỹ thuật và Công nghệ:

- Bắt buộc sử dụng Stack MEAN: Dự án bắt buộc phải sử dụng bộ công nghệ đã được định nghĩa: Frontend (React 19, Vite, Chart.js), Backend (Node.js, Express, JWT), Database (MySQL). Không được phép thay thế bằng các công nghệ tương đương (ví dụ: Vue, Java Spring, MongoDB).
- Tích hợp Cổng thanh toán: Dự án bắt buộc phải tích hợp và vận hành thành công với cổng thanh toán MoMo để xử lý các giao dịch thực tế. Đây là yêu cầu tích hợp bên ngoài không thể thay đổi.
- Quản lý hình ảnh: Việc lưu trữ và phân phối hình ảnh sản phẩm bắt buộc phải sử dụng dịch vụ đám mây Cloudinary. Hệ thống không được lưu trữ hình ảnh trực tiếp trên máy chủ backend.
- Cơ chế Xác thực: Toàn bộ cơ chế xác thực phiên và ủy quyền truy cập API bắt buộc phải sử dụng chuẩn JSON Web Token (JWT).
- Ràng buộc về Nghiệp vụ và Quản trị
- Quản lý tồn kho bằng GRN: Việc cập nhật số lượng tồn kho bắt buộc phải được thực hiện thông qua quy trình Phiếu nhập hàng (GRN). Không được phép cập nhật tồn kho thủ công hoặc trực tiếp ngoài quy trình GRN.
- Cấu trúc phân quyền cứng: Hệ thống phải duy trì cấu trúc phân quyền cố định với hai vai trò chính: Admin (Toàn quyền) và Employee (Quyền giới hạn). Việc tạo thêm các vai trò phức tạp khác nằm ngoài phạm vi của dự án hiện tại.

Ràng buộc về Khả năng Tương thích

- Tương thích Trình duyệt: Hệ thống phải đảm bảo hoạt động tương thích và ổn định trên ba trình duyệt phổ biến nhất (Chrome, Firefox, Edge) với các phiên bản mới nhất.
- Tiêu chuẩn hiển thị: Giao diện Khách hàng (Frontend) phải hoạt động tốt trên các kích thước màn hình phổ biến (Responsive Design), bắt buộc hỗ trợ thiết bị di động.

CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1. Business context

Website Tiệm Bánh Cake Fantasy bao gồm các kịch bản nghiệp vụ chính: Danh mục sản phẩm, Giỏ hàng, Quy trình thanh toán, Quản lý tồn kho, Hệ thống đánh giá, Quản lý đơn hàng, Báo cáo & Phân tích, và Phân quyền truy cập. Danh mục sản phẩm

Khách hàng có thể duyệt danh sách sản phẩm với chức năng lọc và sắp xếp theo tên, loại, hoặc giá. Khi chọn một sản phẩm, họ có thể xem chi tiết sản phẩm gồm tên, mô tả, số lượng tồn kho, giá bán, hình ảnh (lưu trữ trên Cloudinary), điểm đánh giá trung bình, chi tiết nhận xét của người mua khác, và thông tin loại sản phẩm.

Admin có thể thêm, chỉnh sửa, hoặc xóa sản phẩm, quản lý hình ảnh qua Cloudinary, cập nhật tồn kho, tạo và in mã vạch (barcode) cho sản phẩm, và quản lý danh mục sản phẩm. Hệ thống hỗ trợ nhiều hình ảnh cho một sản phẩm để tăng tính trực quan.

Giỏ hàng:

Khách hàng có thể thêm sản phẩm vào giỏ hàng qua nút “Add to Cart” trên mỗi thẻ sản phẩm hoặc trực tiếp từ trang chi tiết với tùy chọn số lượng. Sau khi thêm sản phẩm, họ có thể xem giỏ hàng với bảng tóm tắt hiển thị thông tin: tổng phụ, phí giao hàng, và tổng giá trị đơn hàng. Giỏ hàng sẽ cập nhật tự động theo thời gian thực khi khách hàng thay đổi số lượng hoặc xóa sản phẩm. Khi xác nhận mua hàng, họ nhấn “Checkout” để tiến hành thanh toán. Hệ thống sẽ kiểm tra tính hợp lệ của sản phẩm và tồn kho trước khi cho phép thanh toán.

Quy trình thanh toán:

Sau khi bấm “Checkout” và điền thông tin giao hàng (họ tên, email, địa chỉ, thành phố, mã vùng, quốc gia, số điện thoại), hệ thống sẽ xác thực thông tin và xử lý thanh toán qua cổng MoMo (ví điện tử Việt Nam) hoặc Mock Payment Service (dành cho môi trường phát triển và kiểm thử).

Sau khi thanh toán thành công, hệ thống tạo đơn hàng trong cơ sở dữ liệu, cập nhật trạng thái thanh toán, và chuyển hướng đến trang xác nhận giao dịch.

Cổng thanh toán được tích hợp bảo mật nhằm đảm bảo xử lý giao dịch an toàn và quản lý lỗi hiệu quả.

Quản lý tồn kho:

Admin có thể quản lý tồn kho thông qua nhiều tính năng:

1. Cập nhật số lượng hàng
2. Tạo và quản lý phiếu nhập hàng (GRN)
3. Quản lý nhà cung cấp (CRUD: thêm, sửa, xóa, xem thông tin như tên, địa chỉ, liên hệ, email, số điện thoại)
4. Tự động cập nhật số lượng tồn kho khi hàng nhập về
5. In mã vạch cho từng sản phẩm hoặc in hàng loạt
6. Sử dụng POS System cho bán hàng tại quầy

Mỗi GRN bao gồm thông tin nhà cung cấp, ngày nhập, tổng tiền, và chi tiết từng mặt hàng (số lượng, đơn giá). Hệ thống hỗ trợ xem danh sách và chi tiết từng GRN.

Hệ thống đánh giá:

Sau khi mua hàng, khách hàng có thể đánh giá sản phẩm (1–5 sao) và viết nhận xét chi tiết về trải nghiệm.

Hệ thống hiển thị điểm trung bình trên thẻ sản phẩm và trang chi tiết, cho phép xem toàn bộ đánh giá từ khách khác, hiển thị tên người đánh giá, thời gian đăng, và thống kê phân bố sao.Tính năng này giúp người mua khác tham khảo và giúp cửa hàng cải thiện chất lượng dịch vụ.

Quản lý đơn hàng:

Khách hàng có thể xem toàn bộ lịch sử đơn hàng, kiểm tra trạng thái theo thời gian thực, và xem chi tiết từng đơn (sản phẩm, số lượng, giá, địa chỉ giao hàng, trạng thái thanh toán).

Admin có thể xem tất cả đơn hàng, lọc, tìm kiếm, và cập nhật trạng thái (Item Processing → Food Processing → Out for Delivery → Delivered).

Nhân viên có thể hỗ trợ xử lý đơn và cập nhật trạng thái.

Hệ thống duy trì nhật ký thay đổi để phục vụ mục đích quản lý và chăm sóc khách hàng.

Báo cáo & Phân tích

Admin có thể xem và xuất báo cáo tổng hợp gồm:

- Doanh thu, số lượng đơn hàng, số sản phẩm, giá trị trung bình mỗi đơn
- Biểu đồ xu hướng doanh thu theo tuần, tháng, năm
- Thống kê sản phẩm bán chạy
- Cảnh báo hàng tồn thấp
- Xuất báo cáo ra PDF hoặc Excel để lưu trữ và phân tích

Hệ thống báo cáo cung cấp dữ liệu phân tích hỗ trợ ra quyết định và đánh giá hiệu quả kinh doanh.

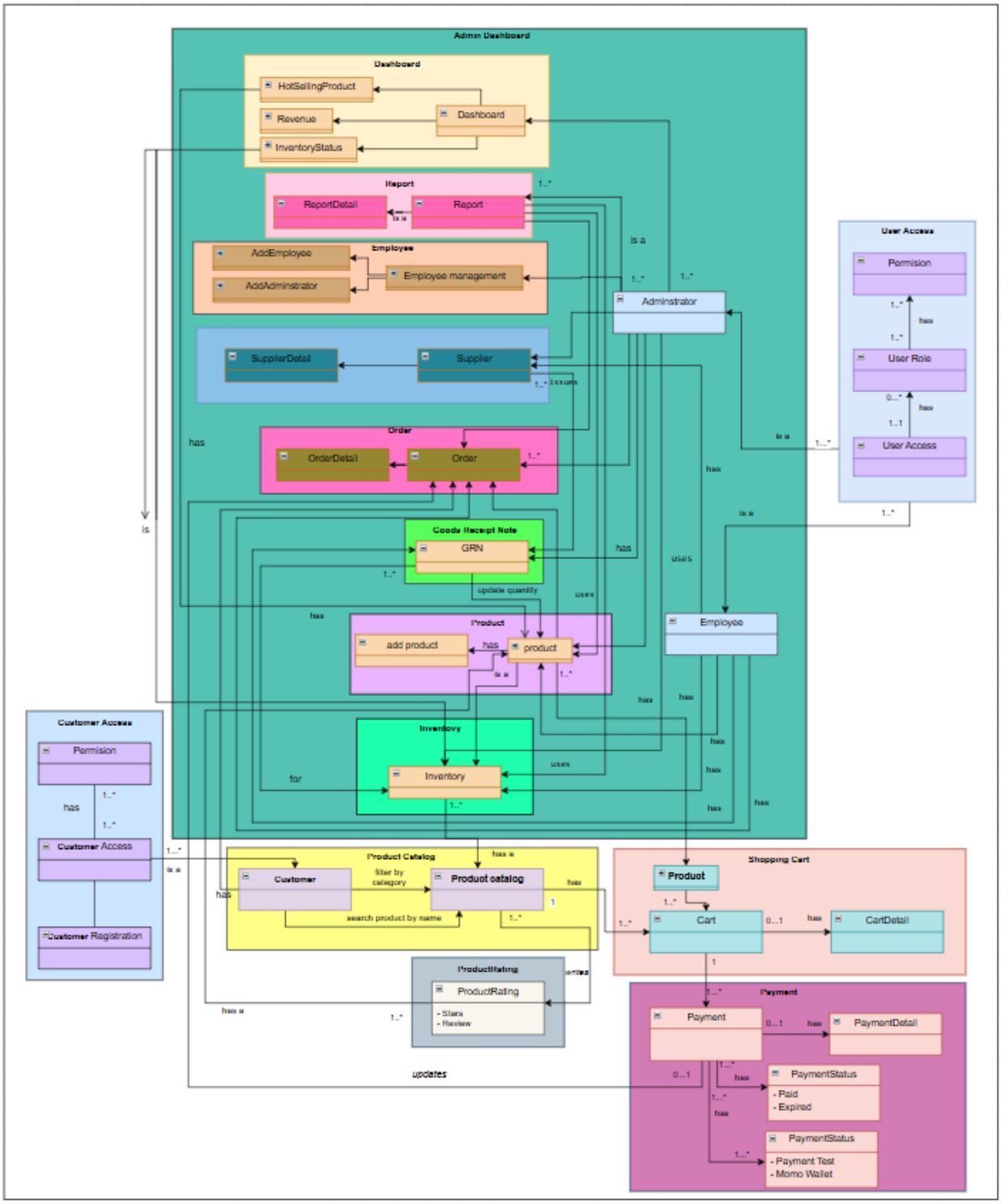
Phân quyền truy cập:

Khách hàng, Nhân viên và Quản trị viên có thể đăng ký, đăng nhập/đăng xuất qua hệ thống xác thực JWT (JSON Web Token).

- Khách hàng: sau khi đăng nhập thành công sẽ được chuyển đến trang danh mục sản phẩm, có thể xem giỏ hàng, đơn hàng và hồ sơ cá nhân.
- Nhân viên: có quyền quản lý sản phẩm, đơn hàng, GRN.
- Quản trị viên: có toàn quyền quản lý người dùng, sản phẩm, nhà cung cấp, đơn hàng, GRN, báo cáo, hệ thống POS, và cấu hình hệ thống.

Hệ thống áp dụng middleware xác thực & phân quyền để ngăn chặn truy cập trái phép vào các khu vực nhạy cảm.

2.2. Conceptual Model

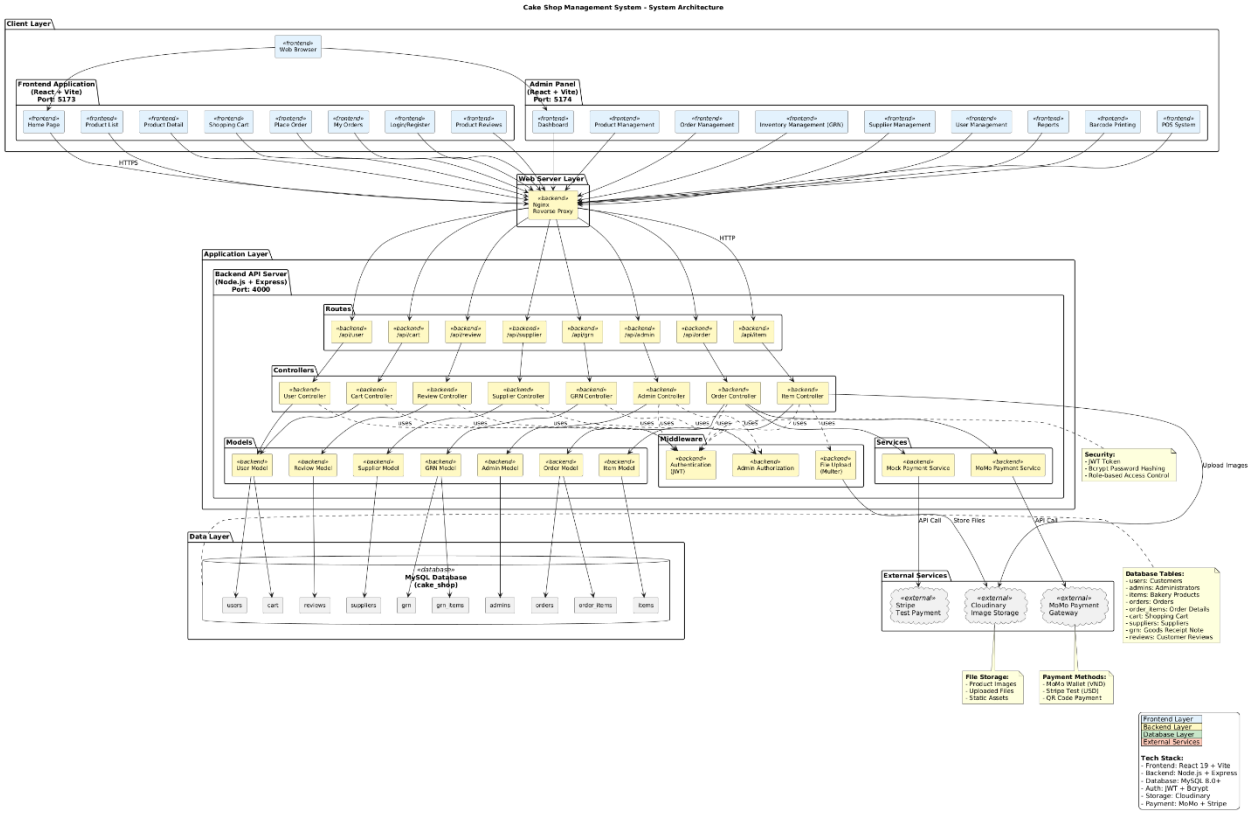


Hình 1. Conceptual Model

Hệ thống **Website Tiệm Bánh (Cake Fantasy)** được thiết kế dựa trên mô hình ý niệm tập trung vào việc quản lý luồng hàng hóa và quy trình kinh doanh khép kín. Hệ thống phân chia quyền truy cập thành hai phân hệ chính:

Administrator (dành cho quản trị viên/nhân viên) và **Customer** (dành cho khách hàng mua sắm), đảm bảo tính bảo mật và chuyên biệt hóa chức năng.

2.2.1 Kiến trúc thống thể



Hình 2. System Architecture

Hệ thống "Website Tiệm Bánh Cake Fantasy" được thiết kế và xây dựng dựa trên mô hình kiến trúc **Client-Server 3 tầng** đảm bảo sự tách biệt rõ ràng giữa giao diện người dùng, logic nghiệp vụ và lưu trữ dữ liệu. Hệ thống giao tiếp thông qua chuẩn RESTful API và được bảo mật bằng cơ chế xác thực JWT.

Tầng Giao diện: Phía Client bao gồm hai ứng dụng Single Page Application (SPA) riêng biệt được phát triển bằng **React** và **Vite**, chạy trên trình duyệt web:

- **Customer Application (Port 5173):** Giao diện dành cho khách hàng với các chức năng tìm kiếm sản phẩm, quản lý giỏ hàng và thực hiện thanh toán.
- **Admin Panel (Port 5174):** Giao diện quản trị viên dùng để xem Dashboard thống kê, quản lý vòng đời sản phẩm, xử lý đơn hàng và theo dõi nhập/xuất kho.

Tầng Ứng dụng Backend là trung tâm xử lý logic của hệ thống, được xây dựng trên nền tảng **Node.js** với **Express Framework** (Port 4000). Kiến trúc Backend tuân thủ mô hình Controller-Service-Model:

- **API Routes & Controllers:** Tiếp nhận các yêu cầu HTTPS từ Client, xử lý logic nghiệp vụ cho các luồng chính như: Auth, Product, Cart, Order, Supplier và GRN.
- **Services & Middleware:** Tích hợp các dịch vụ bên thứ ba và đảm nhiệm các tác vụ trung gian như xác thực (Authentication), phân quyền (Role-based Authorization), kiểm tra dữ liệu đầu vào (Validation) và ghi log lỗi.

Tầng Dữ liệu và Tích hợp

- **Database:** Hệ thống sử dụng **MySQL** (schema cake_shop) làm cơ sở dữ liệu chính, lưu trữ các bảng thực thể cốt lõi như Users, Products, Orders, Inventory, Suppliers. Việc truy xuất dữ liệu được thực hiện thông qua **Sequelize ORM** giúp đảm bảo tính toàn vẹn và dễ dàng bảo trì.
- **Dịch vụ bên thứ 3:** Hệ thống tích hợp **Cloudinary** để lưu trữ và tối ưu hóa hình ảnh sản phẩm. Đối với thanh toán, hệ thống kết nối với cổng thanh toán **MoMo** và hỗ trợ Mock Payment Service để phục vụ quá trình kiểm thử tự động.

Hạ tầng và Vận hành Hệ thống được thiết kế để dễ dàng triển khai và mở rộng. Backend và Frontend được đóng gói (Containerization) sử dụng **Docker**. Môi trường kiểm thử được hỗ trợ bởi **Jest** cho Unit Test, cùng với hệ thống logging tập trung để giám sát lỗi và hiệu năng vận hành.

2.2.2 Mô hình các phân hệ chức năng

User Story

Danh mục sản phẩm

Khách hàng muốn xem danh sách các sản phẩm bánh với chức năng lọc và sắp xếp trên trang chủ (tên, ảnh, mô tả, giá, danh mục và đánh giá).

Khi lọc theo bất kỳ danh mục nào (ví dụ: Bánh Sinh Nhật, Bánh Cưới, Bánh Cupcake) và tên sản phẩm, danh sách sản phẩm cần thu hẹp với các sản phẩm phù hợp.

Khi lọc theo khoảng giá, danh sách sản phẩm chỉ hiển thị những sản phẩm trong khoảng giá đó.

Khi sắp xếp theo thứ tự giảm dần hoặc tăng dần theo giá hoặc tên sản phẩm, danh sách sản phẩm cần tuân theo thứ tự sắp xếp này.

Khi cả lọc và sắp xếp được thực hiện cùng lúc, danh sách sản phẩm sẽ bị ảnh hưởng bởi cả hai thao tác như mô tả ở trên.

Khách hàng muốn xem chi tiết một sản phẩm với đầy đủ các thuộc tính như tên, mô tả chi tiết, số lượng còn trong kho, giá, nhiều ảnh sản phẩm, danh mục, điểm đánh giá trung bình, tổng số lượt đánh giá và danh sách đánh giá từ khách hàng khác với xếp hạng và bình luận của họ.

Khách hàng muốn xem sản phẩm được hiển thị với hình ảnh lưu trữ trên Cloudinary để tải nhanh và chất lượng cao.

Quản trị viên muốn quản lý sản phẩm (các thao tác CRUD) bao gồm tạo sản phẩm mới, chỉnh sửa thông tin sản phẩm hiện có, xóa sản phẩm, tải lên nhiều ảnh sản phẩm thông qua tích hợp Cloudinary và quản lý danh mục sản phẩm.

Quản trị viên muốn cập nhật mức tồn kho sản phẩm khi nhận hàng mới hoặc thực hiện điều chỉnh thủ công.

Quản trị viên muốn tạo và in mã vạch cho từng sản phẩm để sử dụng cho việc theo dõi hàng tồn kho tại cửa hàng.

Quản trị viên muốn in mã vạch hàng loạt cho nhiều sản phẩm cùng lúc để tiết kiệm thời gian trong quá trình thiết lập hàng tồn kho.

Giỏ hàng

Khách hàng muốn thêm bất kỳ sản phẩm nào từ trang danh mục sản phẩm vào giỏ hàng bằng cách nhấp vào nút "Thêm vào Giỏ" (một sản phẩm sẽ được thêm theo mặc định).

Khách hàng muốn xem trang chi tiết sản phẩm và thêm sản phẩm này vào giỏ hàng với số lượng chỉ định nếu thích.

Khách hàng muốn xem danh sách các sản phẩm đã thêm vào giỏ hàng với tên sản phẩm, ảnh, giá và số lượng cho từng mặt hàng.

Khách hàng muốn xem bảng thông tin tóm tắt cho giỏ hàng hiện tại hiển thị tổng phụ giỏ hàng, phí giao hàng và tổng số tiền đơn hàng trên trang giỏ hàng.

Khách hàng muốn cập nhật số lượng của bất kỳ sản phẩm nào trong giỏ hàng bằng cách tăng hoặc giảm số lượng.

Khi cập nhật số lượng sản phẩm, tổng phụ giỏ hàng và tổng số tiền đơn hàng trong bảng tóm tắt cần được cập nhật tự động và ngay lập tức.

Khách hàng muốn xóa bất kỳ sản phẩm nào khỏi giỏ hàng mà không muốn mua nữa.

Khi xóa sản phẩm khỏi giỏ hàng, bảng thông tin tóm tắt cần được cập nhật tương ứng để phản ánh các thay đổi.

Khách hàng muốn tiến hành thanh toán giỏ hàng để đặt hàng.

Khi số lượng sản phẩm trong giỏ hàng bằng không, nút thanh toán sẽ bị vô hiệu hóa hoặc quá trình thanh toán không được tiếp tục.

Khi giỏ hàng được thanh toán, khách hàng được đưa đến trang đặt hàng để nhập thông tin giao hàng trước khi quá trình thanh toán bắt đầu.

Đặt hàng & quy trình thanh toán

Khách hàng muốn nhập thông tin giao hàng bao gồm họ, tên, email, địa chỉ đường phố, thành phố, tỉnh/thành, mã bưu điện, quốc gia và số điện thoại trước khi hoàn tất đơn hàng.

Khách hàng muốn chọn phương thức thanh toán để hoàn tất giao dịch mua hàng (Thanh toán MoMo cho ví điện tử Việt Nam hoặc các phương thức thanh toán được hỗ trợ khác).

Khi chọn Thanh toán MoMo, hệ thống sẽ chuyển hướng tôi đến cổng thanh toán MoMo để quét mã QR và hoàn tất thanh toán.

Trong thời điểm quá trình thanh toán diễn ra, hệ thống xác thực thông tin sản phẩm (tình trạng tồn kho, giá cả), xử lý thanh toán thông qua cổng thanh toán đã chọn, tạo bản ghi đơn hàng với tất cả các mặt hàng và chuyển hướng đến trang xác minh.

Khi bất kỳ thông tin sản phẩm nào không hợp lệ (hết hàng, giá thay đổi, sản phẩm không còn), quá trình thanh toán sẽ bị hủy và thông báo lỗi sẽ được hiển thị cho Khách hàng.

Khi quá trình thanh toán thành công, hệ thống đánh dấu đơn hàng là đã thanh toán, cập nhật số lượng tồn kho sản phẩm, xóa giỏ hàng và hiển thị xác nhận đơn hàng với chi tiết đơn hàng và mã đơn hàng.

Khi quá trình thanh toán thất bại hoặc bị hủy, hệ thống hiển thị thông báo lỗi và cho phép Khách hàng thử lại hoặc chọn phương thức thanh toán khác.

Khách hàng muốn nhận xác nhận đơn hàng với chi tiết đơn hàng sau khi thanh toán thành công để biết đơn hàng của mình đã được đặt thành công.

Quản lý đơn hàng

Khách hàng muốn xem lịch sử đơn hàng đầy đủ hiển thị tất cả các đơn hàng trong quá khứ và hiện tại với ngày đặt hàng, tổng đơn hàng, trạng thái thanh toán và trạng thái giao hàng.

Khách hàng muốn xem thông tin chi tiết cho từng đơn hàng bao gồm danh sách các mặt hàng đã đặt, số lượng, giá từng món, tổng số tiền, địa chỉ giao hàng và trạng thái đơn hàng hiện tại.

Khách hàng muốn theo dõi trạng thái đơn hàng theo thời gian thực để biết khi nào đơn hàng đang được xử lý, chuẩn bị, đang giao hay đã giao.

Quản trị viên muốn xem tất cả đơn hàng từ tất cả khách hàng trong hệ thống với khả năng lọc và tìm kiếm theo ngày đặt hàng, tên khách hàng, trạng thái đơn hàng hoặc mã đơn hàng.

Quản trị viên muốn cập nhật trạng thái đơn hàng qua nhiều giai đoạn: "Đang xử lý đơn hàng", "Đang chế biến", "Đang giao hàng" và "Đã giao".

Khi trạng thái đơn hàng được cập nhật, khách hàng có thể xem trạng thái đã cập nhật trong lịch sử đơn hàng của họ.

Nhân viên muốn xem đơn hàng và cập nhật trạng thái đơn hàng dựa trên cấp độ quyền hạn được gán để hỗ trợ hoàn thành đơn hàng.

Quản trị viên muốn xem chi tiết đơn hàng bao gồm thông tin khách hàng, địa chỉ giao hàng, các mặt hàng đã đặt với số lượng và giá, phương thức thanh toán, trạng thái thanh toán và thời gian đơn hàng.

Quản lý hàng tồn kho & phiếu nhập hàng

Quản trị viên muốn quản lý hàng tồn kho sản phẩm bằng cách theo dõi số lượng tồn kho cho tất cả sản phẩm trong hệ thống.

Quản trị viên muốn tạo Phiếu Nhập Kho mới khi nhận hàng từ nhà cung cấp, bao gồm chọn nhà cung cấp, ngày nhận, và danh sách các mặt hàng đã nhận với số lượng và đơn giá.

Quản trị viên muốn xem danh sách tất cả Phiếu Nhập Kho trong hệ thống với số phiếu, tên nhà cung cấp, ngày nhận, tổng số tiền và thông tin tạo phiếu.

Quản trị viên muốn xem thông tin chi tiết của một Phiếu Nhập Kho cụ thể bao gồm tất cả các mặt hàng đã nhận, số lượng, đơn giá và tổng số tiền.

Khi Phiếu Nhập Kho được tạo và lưu, số lượng tồn kho sản phẩm sẽ được cập nhật tự động bằng cách cộng thêm số lượng đã nhận vào mức tồn kho hiện có.

Quản trị viên muốn nhận cảnh báo hoặc xem chỉ báo khi sản phẩm sắp hết hàng hoặc hết hàng.

Nhân viên muốn tạo và quản lý Phiếu Nhập Kho dựa trên cấp độ quyền hạn được gán để giúp đỡ trong quy trình nhập kho.

Quản lý nhà cung cấp

Quản trị viên muốn quản lý nhà cung cấp (các thao tác CRUD) bao gồm tạo nhà cung cấp mới, chỉnh sửa thông tin nhà cung cấp hiện có và xóa nhà cung cấp.

Quản trị viên muốn lưu trữ thông tin nhà cung cấp bao gồm tên nhà cung cấp, tên người liên hệ, địa chỉ, số điện thoại và địa chỉ email.

Quản trị viên muốn xem danh sách tất cả nhà cung cấp trong hệ thống với thông tin cơ bản của họ.

Quản trị viên muốn chọn nhà cung cấp khi tạo Phiếu Nhập Kho để theo dõi nhà cung cấp nào đã cung cấp hàng hóa.

Hệ Thống Đánh Giá & Nhận Xét

Khách hàng muốn đánh giá bất kỳ sản phẩm nào mà tôi đã mua hoặc thử (từ 1 đến 5 sao) để chia sẻ trải nghiệm của mình.

Khách hàng muốn viết nhận xét chi tiết bằng văn bản cho một sản phẩm giải thích trải nghiệm và ý kiến của mình.

Khách hàng muốn xem điểm đánh giá trung bình cho mỗi sản phẩm được hiển thị rõ ràng trên thẻ sản phẩm và trang chi tiết sản phẩm.

Khách hàng muốn xem tổng số lượt đánh giá cho mỗi sản phẩm để biết có bao nhiêu người đã đánh giá nó.

Khách hàng muốn xem tất cả các đánh giá từ khách hàng khác cho một sản phẩm bao gồm xếp hạng sao, nội dung đánh giá, tên người đánh giá và ngày đánh giá.

Khách hàng muốn xem thống kê phân bố đánh giá cho biết có bao nhiêu khách hàng đã cho 1 sao, 2 sao, 3 sao, 4 sao và 5 sao.

Khi có đánh giá mới được gửi, điểm đánh giá trung bình và tổng số lượt đánh giá của sản phẩm sẽ được cập nhật tự động.

Báo cáo thống kê

Quản trị viên muốn xem bảng điều khiển với thống kê tổng quan bao gồm tổng doanh thu, tổng số đơn hàng, tổng số sản phẩm và các chỉ số hiệu suất chính khác.

Quản trị viên muốn xem biểu đồ trực quan hiển thị xu hướng doanh thu theo tuần, tháng và năm để hiểu hiệu suất kinh doanh theo thời gian.

Quản trị viên muốn xem danh sách các sản phẩm bán chạy nhất với số lượng bán và đóng góp doanh thu để xác định các sản phẩm có hiệu suất tốt nhất.

Quản trị viên muốn xem cảnh báo tồn kho thấp cho các sản phẩm sắp hết hoặc đã hết hàng để lên kế hoạch nhập hàng.

Quản trị viên muốn tải xuống và xuất báo cáo chi tiết ở định dạng PDF hoặc Excel để phân tích ngoại tuyến, lưu trữ hồ sơ và chia sẻ với các bên liên quan.

Quản trì viên muốn xem báo cáo hiệu suất sản phẩm cho biết sản phẩm nào được xem nhiều nhất, thêm vào giỏ hàng nhiều nhất và mua nhiều nhất.

Tạo mã vạch và hệ thống POS

Quản trị viên muốn tạo mã vạch cho một sản phẩm cụ thể để sử dụng cho việc theo dõi hàng tồn kho tại cửa hàng và giao dịch POS.

Quản trị viên muốn in mã vạch cho một sản phẩm để gắn vào các mặt hàng tồn kho thực tế.

Quản trị viên muốn tạo và in mã vạch hàng loạt cho nhiều sản phẩm cùng lúc để tăng tốc độ quy trình gắn nhãn hàng tồn kho.

Quản trị viên hoặc Nhân viên muốn sử dụng hệ thống POS (Điểm Bán Hàng) cho các giao dịch mua bán tại cửa hàng bằng cách quét mã vạch sản phẩm hoặc chọn sản phẩm thủ công.

Quản trị viên hoặc Nhân viên muốn xử lý các giao dịch tại cửa hàng thông qua hệ thống POS và cập nhật hàng tồn kho tự động khi hoàn tất giao dịch bán hàng.

Quản lý người dùng

Mỗi Khách hàng, Nhân viên và Quản trị viên là một Người dùng trong hệ thống.

Khách hàng muốn đăng ký tài khoản mới với email, mật khẩu, tên và số điện thoại để truy cập hệ thống.

Khách hàng muốn đăng nhập vào hệ thống bằng email và mật khẩu của mình.

Khi người dùng với vai trò Khách hàng đăng nhập thành công, họ sẽ được đưa đến trang danh mục sản phẩm (trang chủ) và có thể duyệt sản phẩm, quản lý giỏ hàng và đặt hàng.

Khách hàng muốn đăng xuất khỏi hệ thống để kết thúc phiên làm việc một cách an toàn.

Quản trị viên muốn đăng nhập vào trang quản trị bằng thông tin đăng nhập quản trị (email và mật khẩu).

Khi người dùng với vai trò Quản trị viên đăng nhập thành công, họ sẽ được đưa đến bảng điều khiển quản trị với quyền truy cập đầy đủ vào tất cả các tính năng bao gồm quản lý người dùng, quản lý sản phẩm, quản lý đơn hàng, quản lý nhà cung cấp, quản lý Phiếu Nhập Kho, báo cáo & phân tích, tạo mã vạch và cài đặt hệ thống.

Nhân viên muốn đăng nhập vào trang quản trị bằng thông tin đăng nhập nhân viên của mình.

Khi người dùng với vai trò Nhân viên đăng nhập thành công, họ sẽ được đưa đến bảng điều khiển quản trị với quyền truy cập hạn chế vào các tính năng bao gồm quản lý sản phẩm, quản lý đơn hàng và quản lý Phiếu Nhập Kho, nhưng không có quyền truy cập vào quản lý người dùng hoặc báo cáo nhạy cảm.

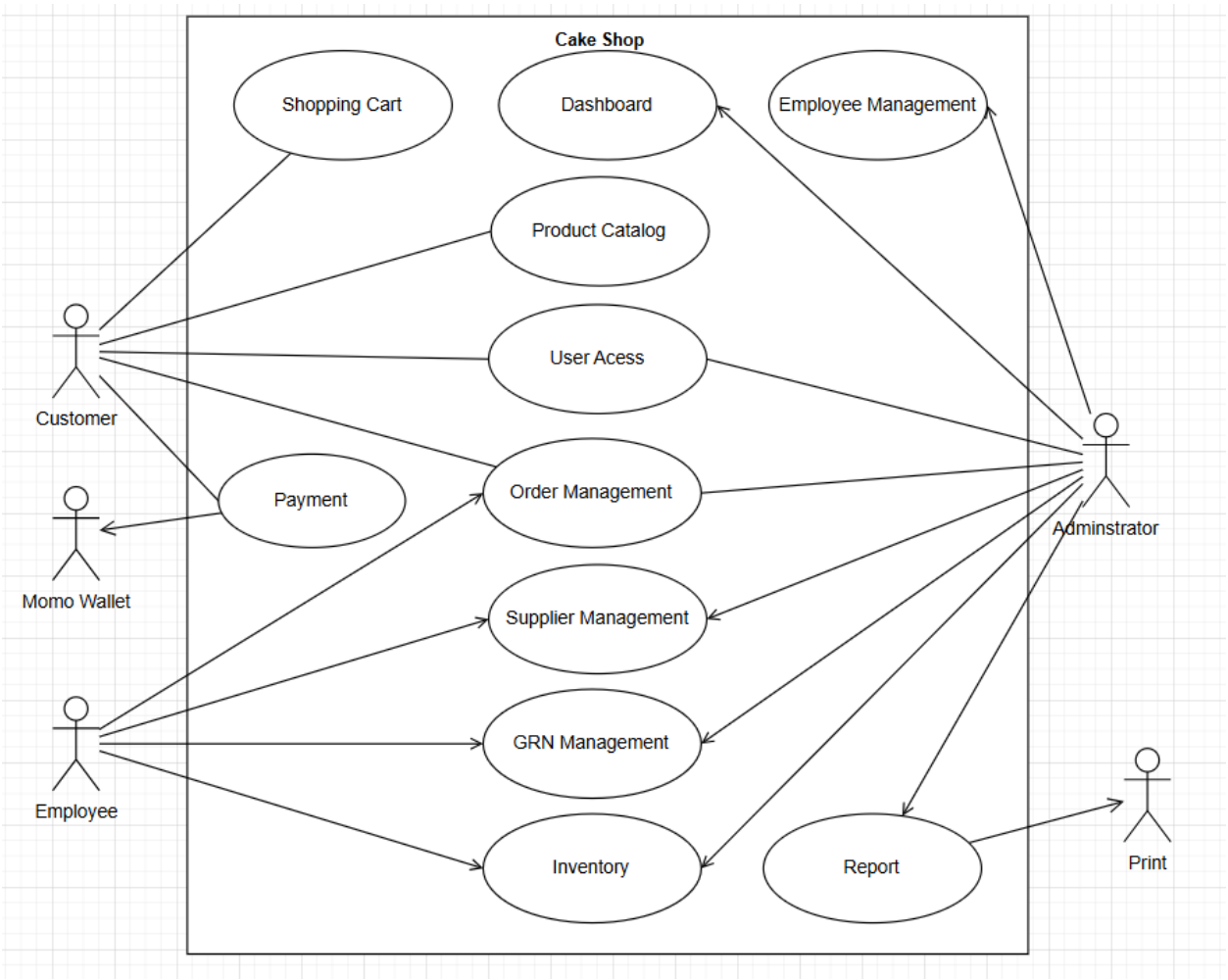
Quản trị viên muốn quản lý tài khoản quản trị viên và nhân viên (các thao tác CRUD) bao gồm tạo quản trị viên/nhân viên mới, chỉnh sửa thông tin người dùng hiện có, cập nhật cấp độ quyền hạn và xóa người dùng.

Quản trị viên muốn gán cấp độ quyền hạn cho người dùng: Quản trị viên cho quyền truy cập hệ thống đầy đủ, và Nhân viên cho quyền truy cập hạn chế.

Khi người dùng cố gắng truy cập một tính năng mà họ không có quyền, hệ thống sẽ từ chối truy cập và hiển thị thông báo "Truy cập bị từ chối".

Khách hàng, Quản trị viên hoặc Nhân viên muốn hệ thống sử dụng xác thực JWT (JSON Web Token) bảo mật để bảo vệ phiên làm việc và dữ liệu của tôi.

Mô hình Use - case



Hình 3. Use case tổn quát của hệ thống

Trong sơ đồ có 4 tác nhân:

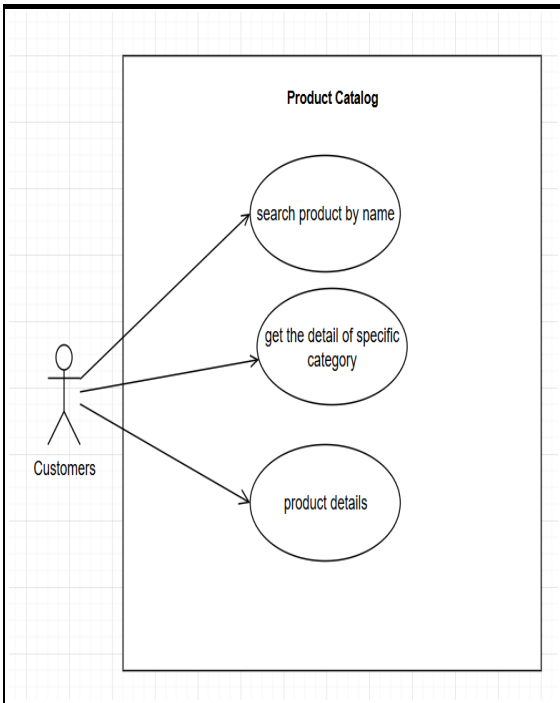
Tác nhân	**Vai trò**	**Mô tả**
Customer	Người mua bánh	Thực hiện các hoạt động như xem sản phẩm, đặt hàng, thanh toán.
Employee	Quản lý kho, đơn hàng, hàng nhập	Phụ trách các công việc vận hành nội bộ như kiểm tra hàng hóa, nhập hàng, cập nhật tồn kho.
Administrator	Người quản lý toàn hệ thống	Có quyền cao nhất: quản lý người dùng, sản phẩm, báo cáo, thống kê, phân quyền.

Momo Wallet	Hệ thống thanh toán ngoài	Được kết nối để thực hiện giao dịch thanh toán điện tử cho khách hàng.

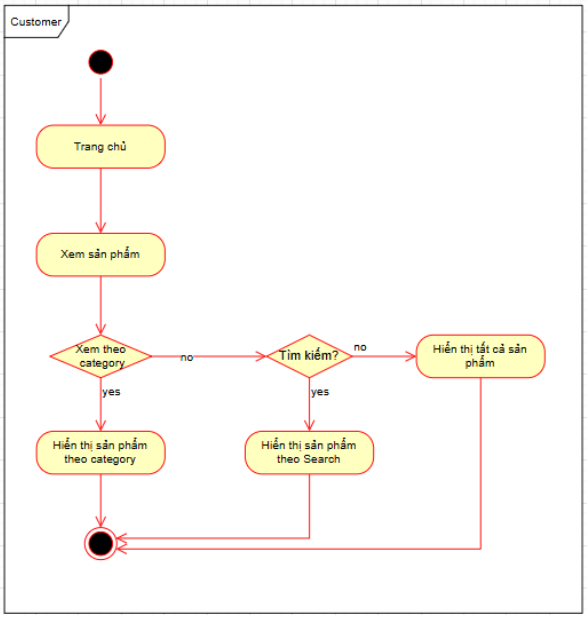
Các Use Case chính trong hệ thống Cake Shop

Use Case	**Mô tả chức năng**	**Liên quan tới tác nhân**
Shopping Cart	Khách hàng thêm/xóa bánh, xem giỏ hàng.	Customer
Payment	Thực hiện thanh toán đơn hàng qua Momo.	Customer ⇆ Momo Wallet
Order	Quản lý quy trình đặt hàng: tạo, xác nhận, theo dõi.	Customer, Employee, Administrator
Product Management	Quản lý sản phẩm: thêm, sửa, xóa bánh, giá, hình ảnh.	Administrator
User Access	Quản lý đăng nhập, phân quyền người dùng.	Customer, Employee, Administrator
Dashboard	Giao diện tổng quan: hiển thị doanh số, thống kê.	Administrator
Supplier	Quản lý thông tin nhà cung cấp nguyên liệu.	Employee, Administrator
GRN	Phiếu nhập hàng – khi nhận hàng từ nhà cung cấp.	Employee, Administrator
Inventory	Quản lý tồn kho, cập nhật khi nhập/xuất hàng.	Employee, Administrator
Employee	Quản lý thông tin nhân viên, vai trò, lương.	Administrator
Report	Xuất các báo cáo: doanh thu, tồn kho, sản phẩm bán chạy.	Administrator

Use case Sản phẩm



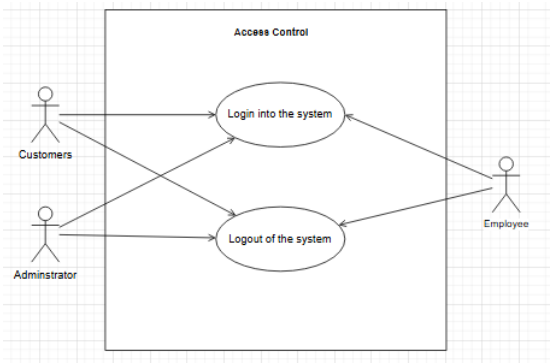
Hình 4. Use-case của chức năng Product Catalog



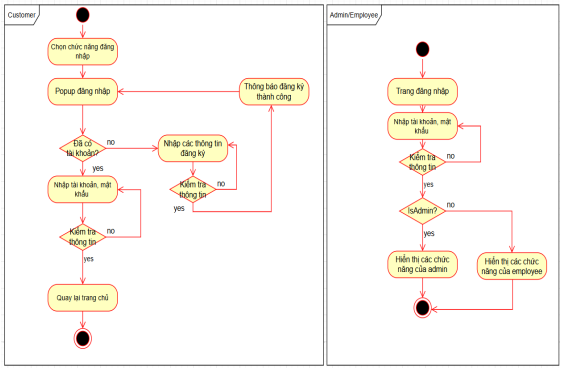
Hình 5. Activity của chức năng Product Catalog

Người dùng	Hành động/Nghiệp vụ	Mô tả
Khách hàng	Tìm kiếm sản phẩm theo tên	Cho phép khách hàng tìm kiếm nhanh sản phẩm họ quan tâm.
Khách hàng	Xem chi tiết danh mục cụ thể	Cho phép khách hàng duyệt và xem các sản phẩm thuộc một danh mục nhất định.
Khách hàng	Product detais	Cho phép khách hàng xem chi tiết thông tin của sản phẩm

Phân hệ Kiểm soát truy cập



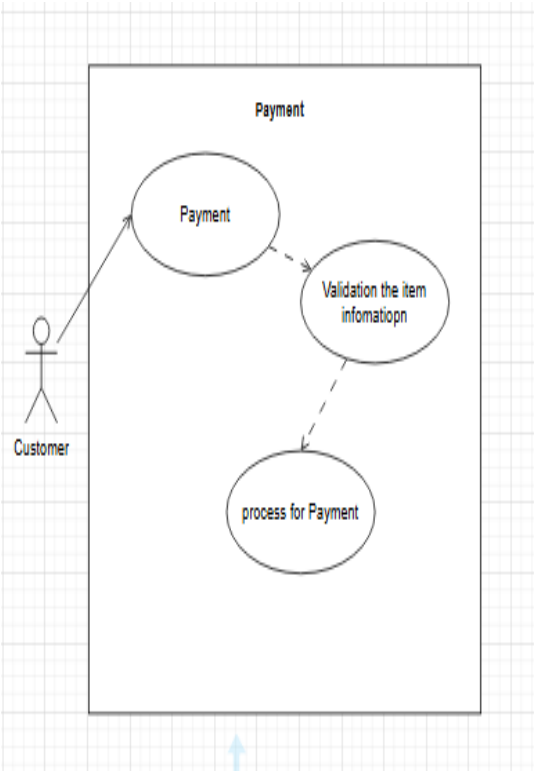
Hình 6. Use-case của chức năng đăng nhập



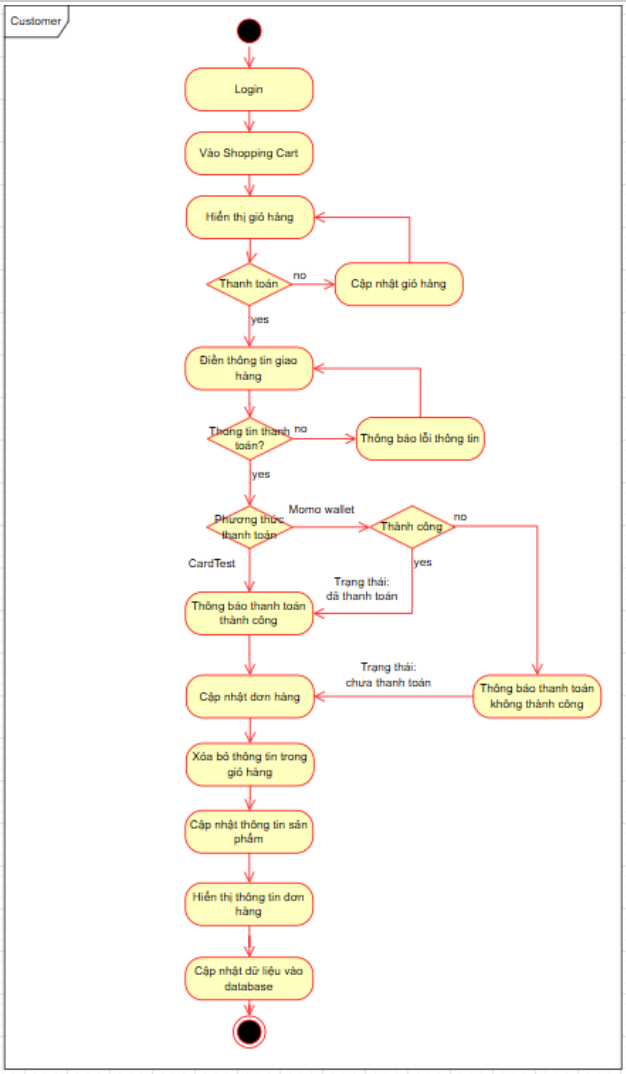
Hình 7. Activity của chức năng đăng nhập

Người dùng	Hành động/Nghiệp vụ	Mô tả
Khách hàng, Quản trị viên, Nhân viên	Đăng nhập vào hệ thống	Yêu cầu tất cả các loại người dùng hợp lệ phải nhập thông tin đăng nhập (ví dụ: tên người dùng và mật khẩu) để được cấp quyền truy cập các chức năng của hệ thống.
Khách hàng, Quản trị viên, Nhân viên	Đăng xuất khỏi hệ thống	Cho phép người dùng kết thúc phiên làm việc và bảo mật tài khoản bằng cách thoát khỏi hệ thống.

Use case Thanh toán



Hình 8. Use-case của chức năng thanh toán

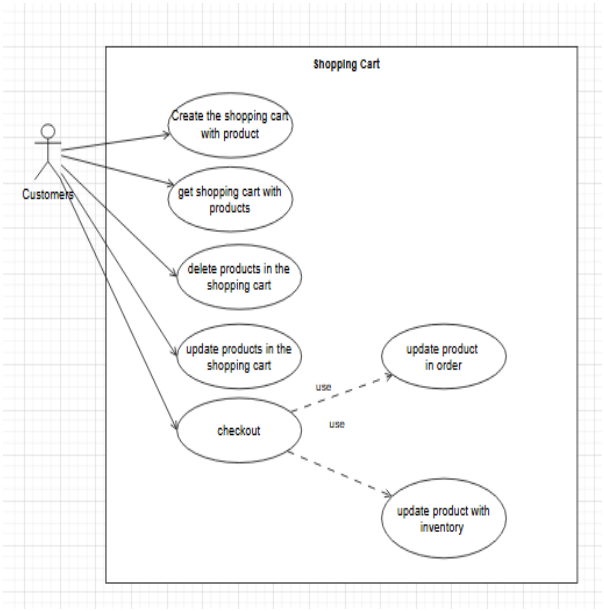


Hình 9. Activity của chức năng Payment

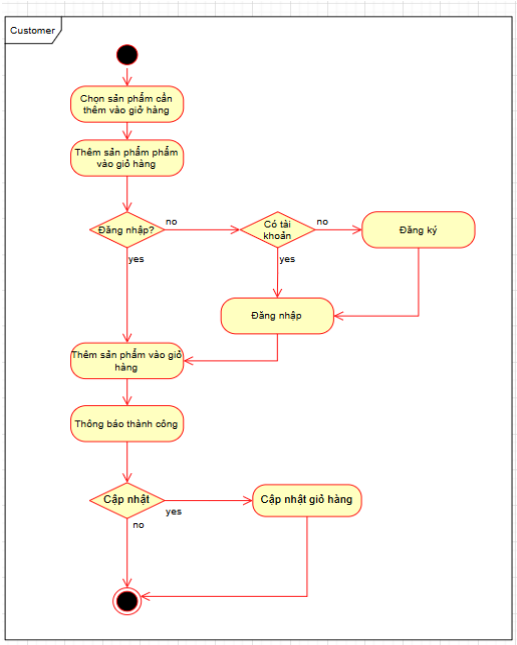
Người dùng	Hành động/Nghiệp vụ	Mô tả
Khách hàng	Thanh toán	Use Case tổng quát, đại diện cho toàn bộ quy trình mua hàng và trả tiền.
System	Xác thực thông tin mặt hàng	Bao gồm trong Use Case Thanh toán. Hệ thống phải kiểm tra lại các thông tin như số lượng, giá cả, tình trạng còn hàng của các mặt hàng trong đơn hàng trước khi cho phép tiếp tục thanh toán.

System	Xử lý thanh toán	Mở rộng từ Use Case Xác thực thông tin mặt hàng. Sau khi thông tin mặt hàng đã được xác thực, hệ thống sẽ thực hiện các bước xử lý thanh toán thực tế (ví dụ: kết nối với cổng thanh toán, trừ tiền, ghi nhận giao dịch).

Use case Giỏ hàng



Hình 10. Use-case của chức năng giỏ hàng

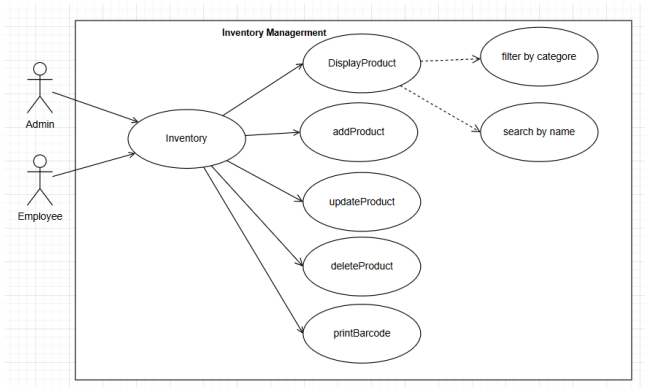


Hình 11. Activity của chức năng giỏ hàng

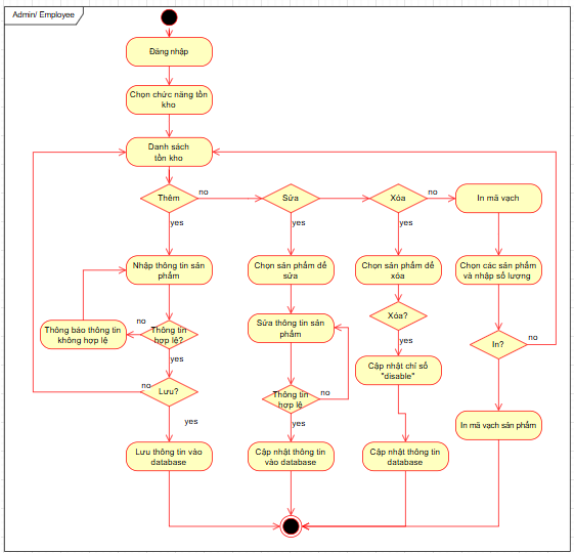
Hành động/Nghiệp vụ	Mô tả
Tạo giỏ hàng với sản phẩm	Thêm sản phẩm đầu tiên vào giỏ hàng, tạo giỏ hàng nếu chưa có.
Xem giỏ hàng với các sản phẩm	Xem lại danh sách các mặt hàng đã chọn trong giỏ hàng.
Xóa sản phẩm trong giỏ hàng	Loại bỏ một hoặc nhiều sản phẩm khỏi giỏ hàng.
Cập nhật sản phẩm trong giỏ hàng	Thay đổi số lượng của một sản phẩm đã có trong giỏ hàng.

Thanh toán/Kiểm tra	Bắt đầu quy trình mua hàng, chuyển từ Giỏ hàng sang Đơn hàng.
Cập nhật sản phẩm trong đơn hàng	Là một bước bắt buộc trong quá trình Checkout. Hệ thống sử dụng chức năng này để ghi nhận thông tin sản phẩm từ giỏ hàng vào đơn hàng mới.
Cập nhật tồn kho	Là một bước bắt buộc trong quá trình Checkout. Hệ thống thực hiện trừ số lượng sản phẩm đã bán khỏi kho hàng ngay khi quá trình thanh toán hoàn tất.

Use case Tồn kho



Hình 12. Use-case của chức năng quản lý tồn kho

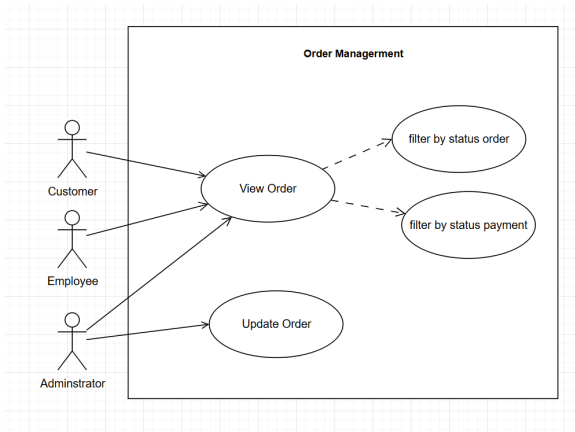


Hình 13. Activity của chức năng quản lý tồn kho

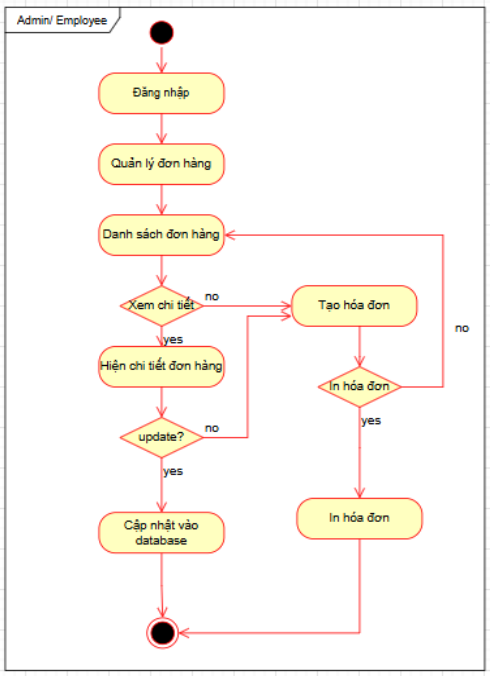
Người dùng	Hành động/Nghiệp vụ	Mô tả
Quản trị viên, Nhân viên	Tồn kho	Đại diện cho các tương tác cơ bản với hệ thống kho hàng.
Quản trị viên, Nhân viên	Hiển thị sản phẩm	Cho phép xem danh sách, thông tin và số lượng tồn kho hiện tại của các sản

		phẩm. Có thể lọc theo danh mục sản phẩm hoặc tìm theo tên sản phẩm
Quản trị viên, Nhân viên	Thêm sản phẩm	Cho phép thêm sản phẩm vào kho
Quản trị viên, Nhân viên	Thêm sản phẩm	Cho phép sửa thông tin sản phẩm
Quản trị viên, Nhân viên	Xóa sản phẩm	Cho phép xóa sản phẩm khỏi kho
Quản trị viên, Nhân viên	In mã vạch	Cho phép tạo và in mã vạch cho các sản phẩm mới hoặc sản phẩm cần dán nhãn lại.

Use case Đặt hàng



Hình 14. Use-case của chức năng quản lý đơn hàng

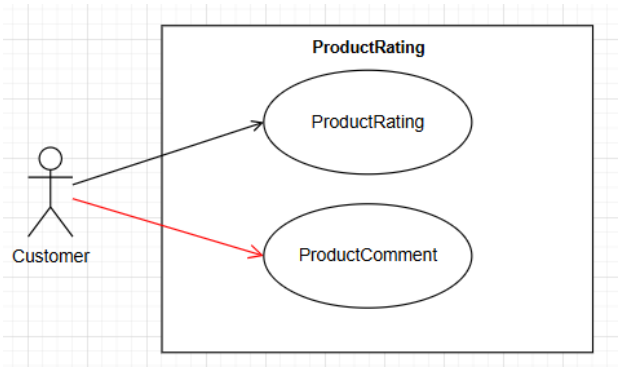


Hình 15. Activity của chức năng quản lý đơn hàng

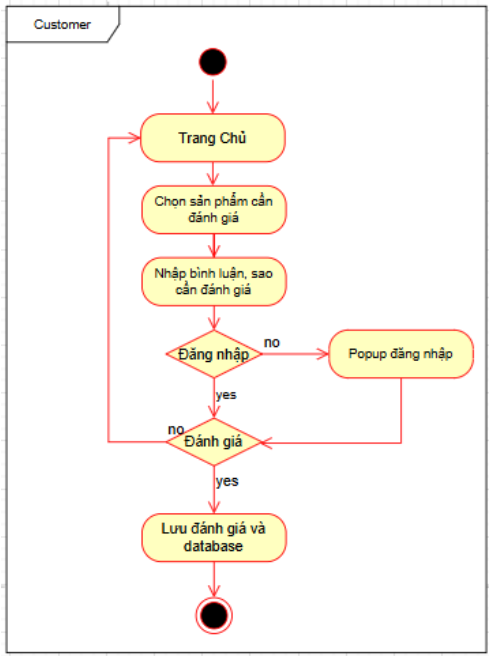
Người dùng	Hành động/Nghiệp vụ	Mô tả

Khách hàng Quản trị viên Nhân viên	Xem đơn hàng	Là một bước bắt buộc của Use Case Order dành cho khách hàng. Cho phép xem lại các đơn hàng đã thực hiện trước đó. Có thể xem theo các bộ lộc theo trạng thái đơn hàng, trạng thái thanh toán
Quản trị viên	Cập nhật đơn hàng	Cho phép quản trị viên thay đổi trạng thái, thông tin giao hàng hoặc các chi tiết khác của một đơn hàng.

Use case Đánh giá sản phẩm



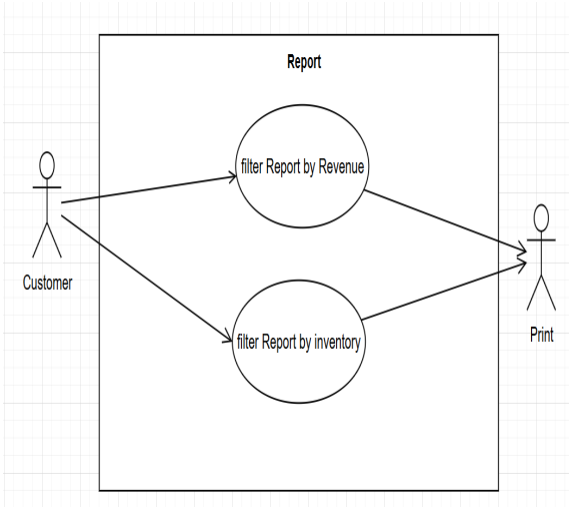
Hình 16. Use-case của chức năng đánh giá



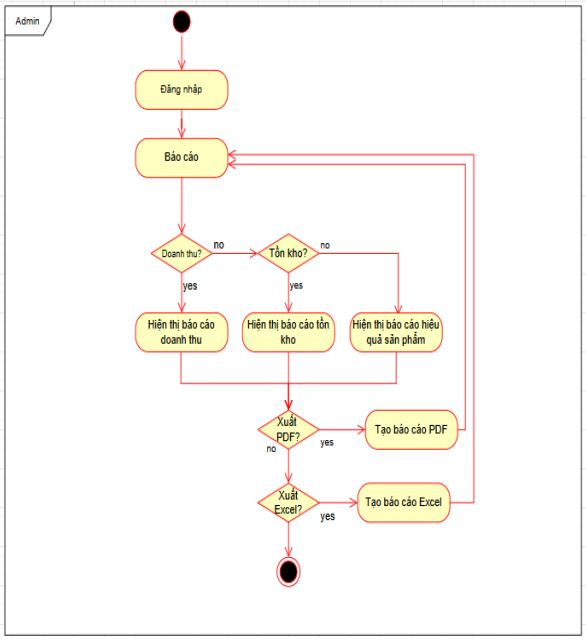
Hình 17. Activity của chức năng đánh giá

Người dùng	Hành động/Nghiệp vụ	Mô tả
Khách hàng	Đánh giá sản phẩm	Cho phép khách hàng đã mua và sử dụng sản phẩm gửi đánh giá 1-5 sao
Khách hàng	Đánh giá sản phẩm	Cho phép khách hàng đã mua và sử dụng sản phẩm gửi đánh giá nhận xét về sản phẩm đó.

Phân hệ Báo cáo



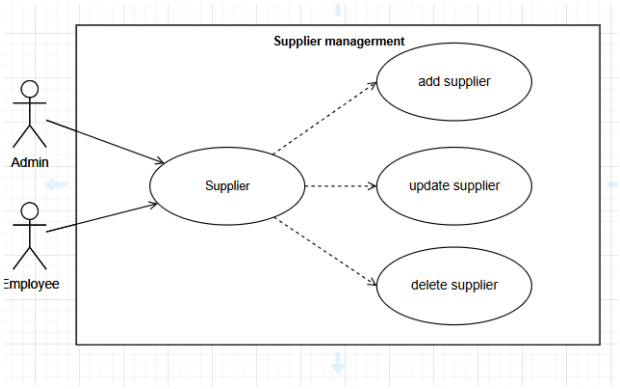
Hình 18. Use-case của chức năng báo cáo



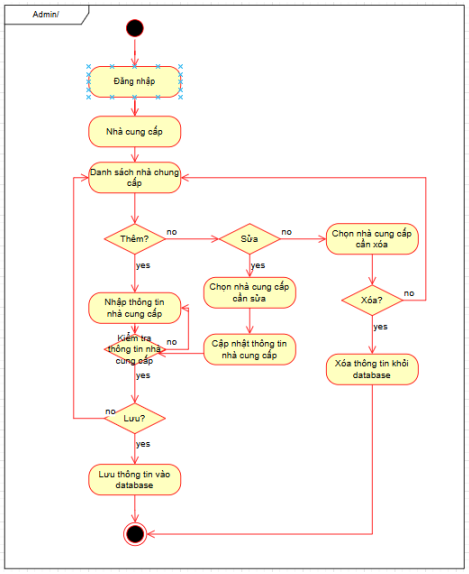
Hình 19. Activity của chức năng báo cáo

Người dùng	Hành động/Nghiệp vụ	Mô tả
Admin	Filter report by revenue	Cho phép người dùng tạo báo cáo tùy chỉnh bằng cách chọn lọc dữ liệu dựa doanh thu và khoảng thời gian cụ thể. Sau đó có thể in ra báo báo này
Admin	Filter report by inventory	Cho phép người dùng tạo báo cáo tùy chỉnh bằng cách chọn lọc dữ liệu dựa trên tồn kho và khoảng thời gian cụ thể. Sau đó có thể in ra báo báo này

Use case Nhà cung cấp



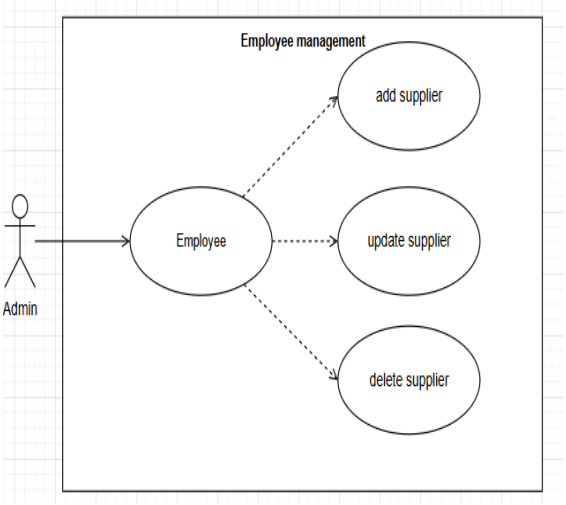
Hình 20. Use-case của chức năng quản lý nhà cung cấp



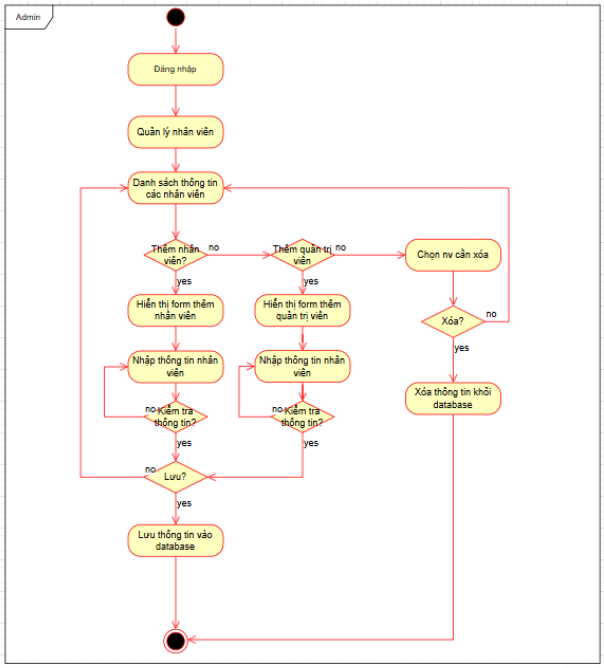
Hình 21. Activity của chức năng quản lý nhà cung cấp

Người dùng	Hành động/Nghiệp vụ	Mô tả
Quản trị viên, Nhân viên	Nhà cung cấp	Đại diện cho việc truy cập thông tin nhà cung cấp.
Quản trị viên, Nhân viên	Thêm nhà cung cấp	Cho phép nhập thông tin nhà cung cấp mới vào hệ thống (tên, địa chỉ, liên hệ...).
Quản trị viên, Nhân viên	Cập nhật nhà cung cấp	Cho phép thay đổi thông tin của nhà cung cấp đã có.
Quản trị viên, Nhân viên	Xóa nhà cung cấp	Cho phép loại bỏ thông tin của một nhà cung cấp khỏi hệ thống.

Use case Nhân viên



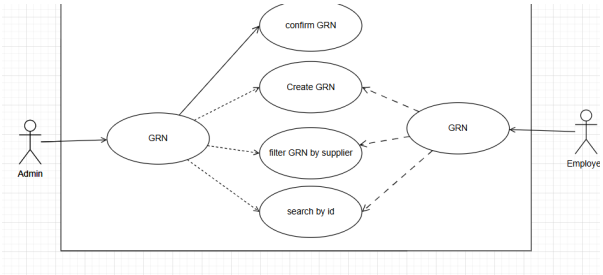
Hình 22. Use-case của chức năng quản lý nhân viên



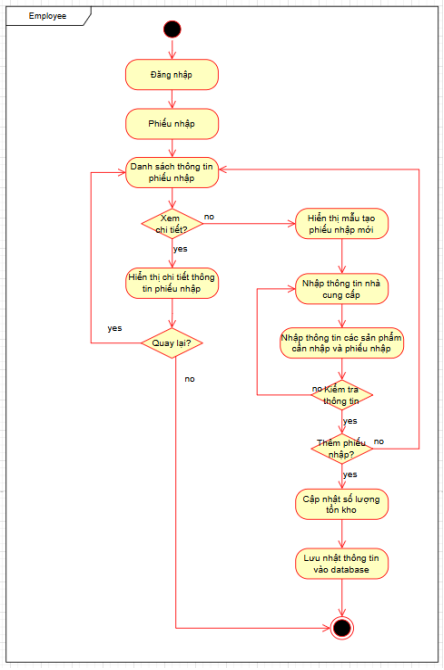
Hình 23. Activity của chức năng quản lý nhân viên

Người dùng	Hành động/Nghiệp vụ	Mô tả
Quản trị viên	Nhân viên	Đại diện cho việc quản lý tài khoản nhân viên.
Quản trị viên	Thêm nhân viên	Cho phép tạo tài khoản và hồ sơ nhân viên mới. Lưu ý: Tên Use Case này có thể bị gõ nhầm, thường sẽ là "add employee".
Quản trị viên	Cập nhật nhân viên	Cho phép thay đổi thông tin hoặc quyền hạn của nhân viên. Lưu ý: Tên Use Case này có thể bị gõ nhầm, thường sẽ là "update employee".
Quản trị viên	Xóa nhân viên	Cho phép vô hiệu hóa hoặc xóa tài khoản nhân viên. Lưu ý: Tên Use Case này có thể bị gõ nhầm, thường sẽ là "delete employee".

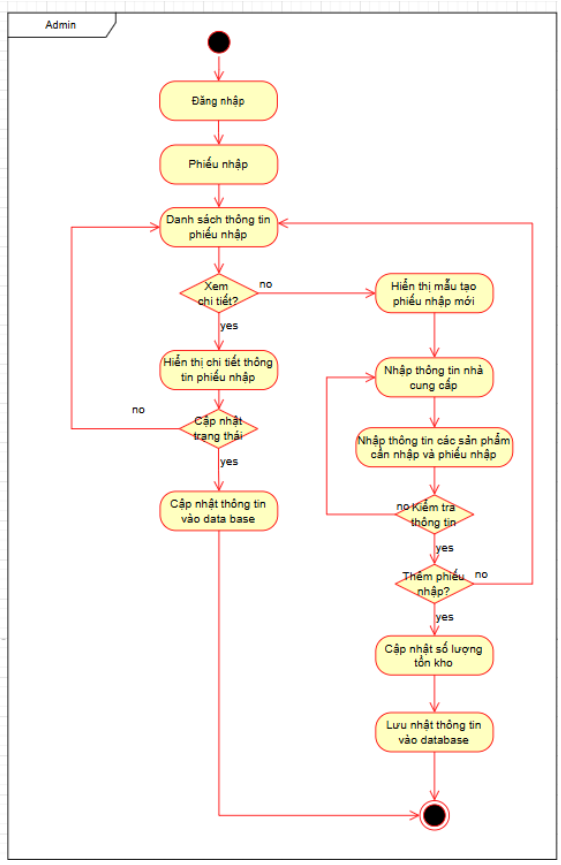
Use case Phiếu nhập hàng



Hình 24. Use-case của chức năng quản lý phiếu nhập



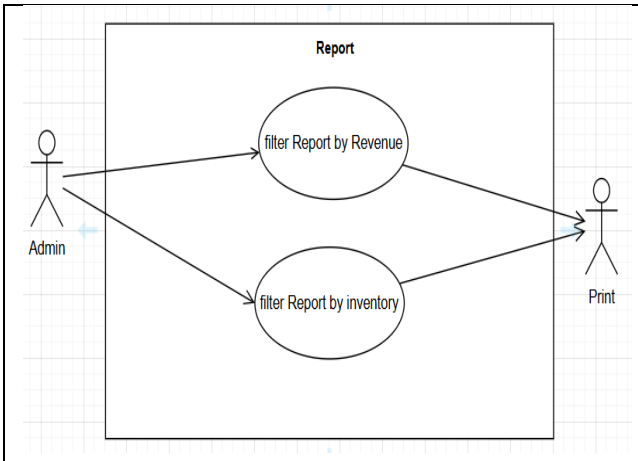
Hình 25. Activity của chức năng quản lý phiếu nhập dành cho nhân viên



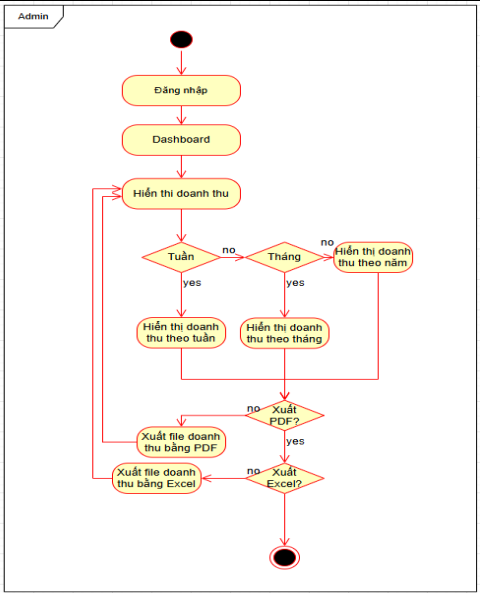
Hình 26. Activity của chức năng quản lý phiếu nhập dành cho admin

Người dùng	Hành động/Nghiệp vụ	Mô tả
Quản trị viên, Nhân viên	GRN	Đại diện cho việc tương tác với các phiếu nhập hàng.
Quản trị viên, Nhân viên	Tạo GRN	Lập phiếu khi hàng hóa từ nhà cung cấp được giao đến và xác nhận nhập kho. Việc này thường dẫn đến cập nhật tồn kho (liên kết với phân hệ Inventory).
Quản trị viên, Nhân viên	Lọc GRN theo nhà cung cấp	Cho phép tìm kiếm và xem các phiếu nhập hàng chỉ liên quan đến một nhà cung cấp cụ thể.
Quản trị viên, Nhân viên	Tìm kiếm theo ID	Cho phép tìm kiếm một phiếu nhập hàng cụ thể dựa trên số ID (mã định danh) của nó.
Quản trị viên	Confirm GRN	Cho phép quản trị viên xác nhận phiếu nhập hàng để cập nhật số lượng sản phẩm vào tồn kho

Use case thống kê doanh thu



Hình 27. Use case của chức năng thống kê



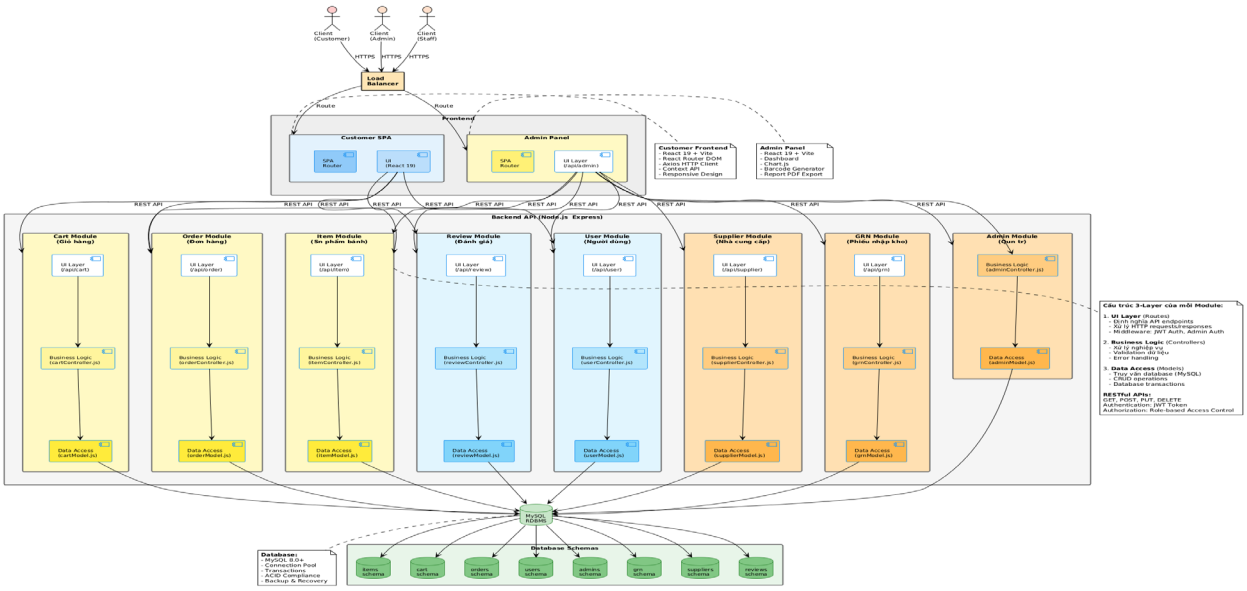
Hình 28. Activity của chức năng thống kê

Mục	Nội dung mô tả
Tên Use Case	Xem và xuất báo cáo doanh thu
Actor	Admin
Mục tiêu	Cho phép Admin xem doanh thu theo tuần/tháng/năm và xuất báo cáo dạng PDF hoặc Excel.
Điều kiện tiên quyết	Admin đã đăng nhập vào hệ thống.

Điều kiện kết thúc	Báo cáo được hiển thị hoặc file báo cáo được xuất thành công.
Mô tả luồng sự kiện	1. Admin đăng nhập vào hệ thống. 2. Admin truy cập Dashboard. 3. Hệ thống hiển thị chức năng xem doanh thu. 4. Admin chọn loại báo cáo muốn xem: Tuần, Tháng, hoặc Năm. – Nếu chọn Tuần → Hệ thống hiển thị doanh thu theo tuần. – Nếu chọn Tháng → Hệ thống hiển thị doanh thu theo tháng. – Nếu không chọn gì → Hệ thống hiển thị doanh thu theo năm. 5. Admin chọn xuất báo cáo (tùy chọn). – Nếu chọn Xuất PDF → Hệ thống xuất file báo cáo dạng PDF. – Nếu chọn Xuất Excel → Hệ thống xuất file báo cáo dạng Excel. 6. Use case kết thúc.
Ngoại lệ	- Lỗi tải dữ liệu doanh thu. - Không tạo được file PDF hoặc Excel.

2.3. Kiến trúc hệ thống & thiết kế kỹ thuật

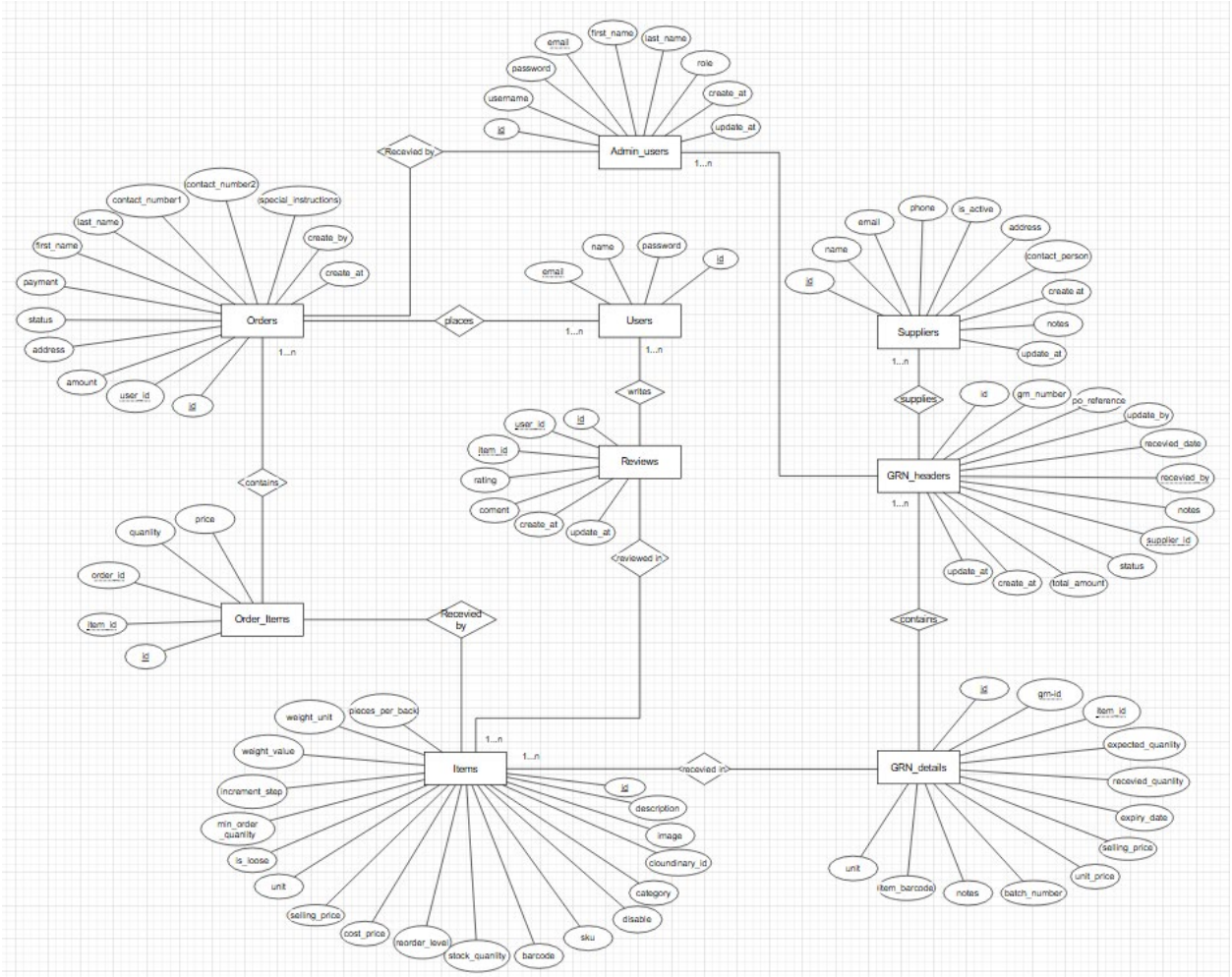
2.3.2. Sơ đồ kiến trúc SPA



Hình 29. Sơ đồ kiến trúc SPA (Singer page application)

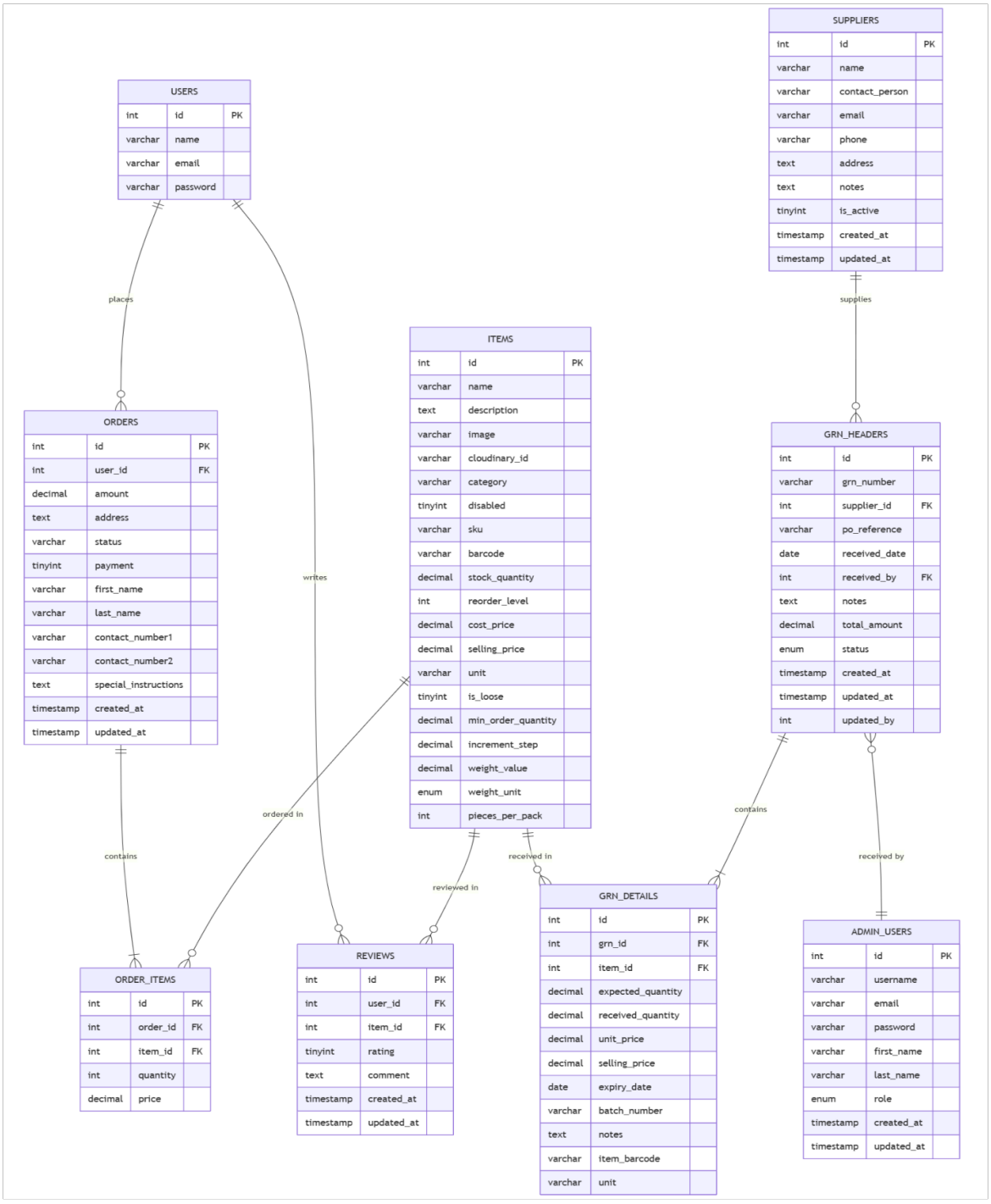
2.3.3 Sơ đồ ERD

ERD Mức khái niệm



Hình 30. Sơ đồ ERD ở mức khái niệm của hệ thống

ERD Mức vật lý



Hình 31. Sơ đồ ERD ở mức vật lý của hệ thống

Quản lý Sản phẩm và Kho Trung tâm của hệ thống là thực thể **Items** Để đáp ứng đặc thù kinh doanh thực phẩm, bảng này không chỉ lưu trữ thông tin định danh (Tên, SKU, Barcode, Hình ảnh) mà còn tích hợp chặt chẽ các thông số quản lý:

- **Quản lý tài chính:** Tách biệt *Giá vốn* và *Giá bán* để phục vụ tính toán lợi nhuận.
- **Kiểm soát tồn kho:** Theo dõi *Số lượng tồn* theo thời gian thực và hỗ trợ cảnh báo nhập hàng tự động thông qua *Mức đặt hàng tối* và *Điểm tái đặt hàng.*
- **Quy cách hàng hóa:** Quản lý đơn vị tính, trọng lượng và quy cách đóng gói.

Nhập hàng Quy trình nhập kho được thiết kế theo mô hình Master-Detail để đảm bảo khả năng truy xuất nguồn gốc:

- **Suppliers:** Quản lý thông tin đối tác cung ứng.
- **GRN (Goods Received Note):** Phiếu nhập kho bao gồm **GRN_headers** (thông tin tổng quan, người phụ trách nhập, thời gian) và **GRN_details** (chi tiết từng mặt hàng).
- *Điểm đặc biệt:* Hệ thống lưu trữ **Số lô** và **Hạn sử dụng** tại bảng chi tiết nhập. Điều này cho phép kiểm soát chất lượng hàng hóa (FIFO/FEFO) và quản lý hàng hết hạn – yếu tố quan trọng nhất trong quản lý thực phẩm.

Bán hàng Dữ liệu đơn hàng được tổ chức nhằm đảm bảo tính lịch sử và chính xác về tài chính:

- **Orders:** Lưu trữ thông tin giao dịch (Khách hàng, Trạng thái đơn, Phương thức thanh toán, Địa chỉ giao hàng).
- **Order_Items:** Lưu chi tiết sản phẩm trong đơn. Đặc biệt, bảng này lưu trữ giá bán tại thời điểm giao dịch, đảm bảo doanh thu không bị sai lệch khi giá gốc của sản phẩm thay đổi trong tương lai.

Người dùng Hệ thống phân tách người dùng thành hai thực thể riêng biệt:

- **Users:** Khách hàng mua sắm, tích hợp tính năng Đánh giá để tăng tính tương tác.
- **Admin_users:** Nhân viên/Quản trị viên hệ thống. Quyền hạn thao tác (nhập kho, duyệt đơn) được kiểm soát thông qua thuộc tính phân quyền.

Các mối quan hệ chính

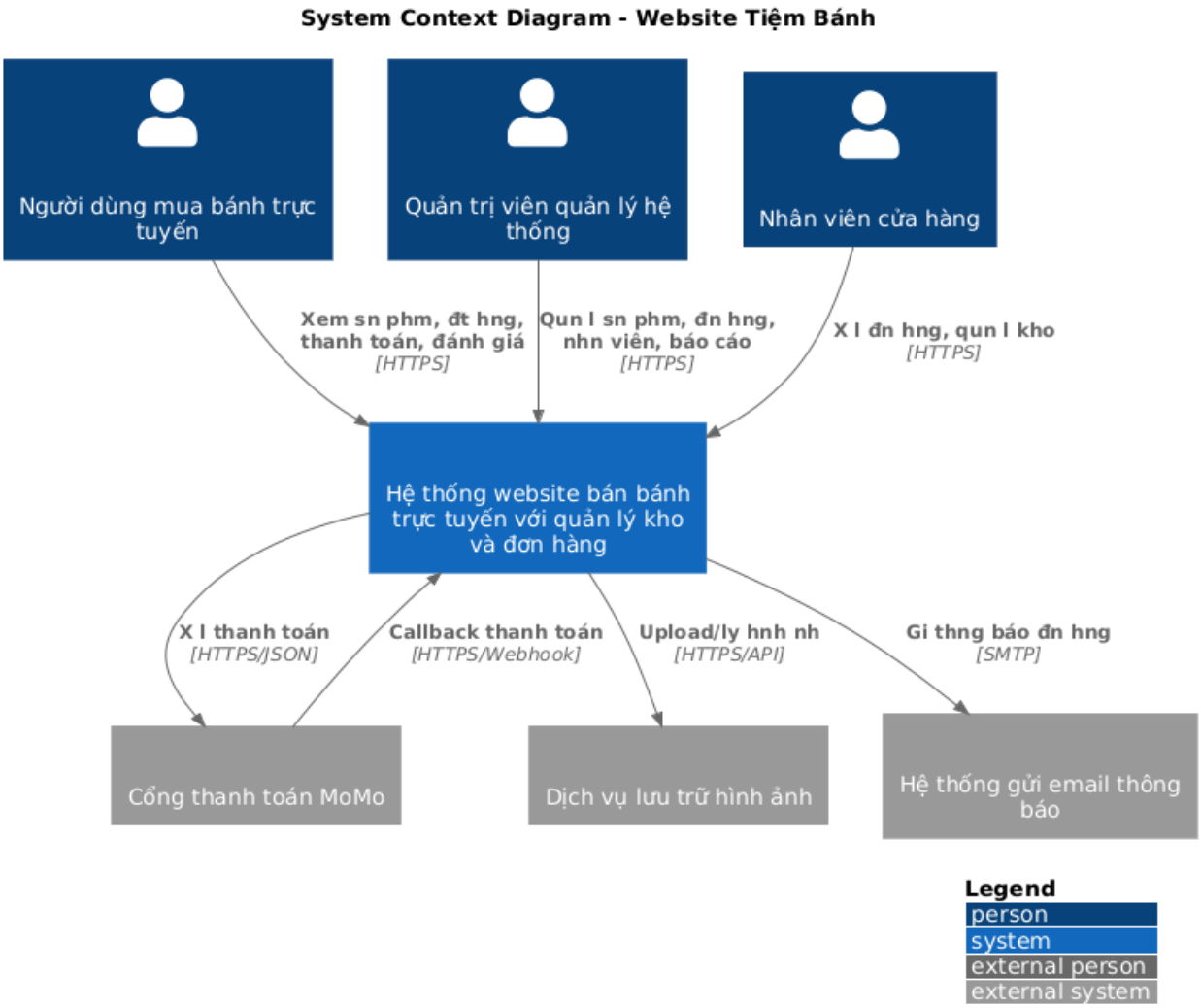
Mô hình tuân thủ các quy tắc chuẩn hóa dữ liệu với các mối quan hệ 1-N (One-to-Many) chủ đạo:

- **Supplier – GRN – Items:** Một nhà cung cấp cung ứng nhiều lô hàng; mỗi lô hàng cập nhật số lượng cho kho sản phẩm.

- **User – Orders – Items:** Một khách hàng thực hiện nhiều đơn hàng; mỗi đơn hàng bao gồm nhiều sản phẩm.
- **Admin – Operations:** Quản trị viên có mối quan hệ tham chiếu đến các nghiệp vụ quan trọng như "Người nhận hàng" (Received by trong GRN) và "Người xử lý đơn" (trong Orders) để phục vụ công tác audit (kiểm tra).

2.3.4 Mô hình C4

Sơ đồ C1 - System Context



Hình 32. Sơ đồ C1 - System Context

Web Application

- **Chức năng:** Là giao diện tương tác trực tiếp với người dùng cuối (Khách hàng) và đội ngũ vận hành (Admin, Nhân viên).
- **Trách nhiệm:**

- Hiển thị danh mục bánh, chi tiết sản phẩm và giỏ hàng cho khách mua.
- Cung cấp Dashboard quản lý kho, nhập đơn GRN, và duyệt đơn hàng cho Admin/Staff.
- Gửi yêu cầu (Requests) về phía Backend thông qua giao thức HTTPS.

– **Người dùng:** Khách hàng, Quản trị viên, Nhân viên cửa hàng.

API Application

– **Chức năng:** Là trung tâm xử lý logic nghiệp vụ của toàn bộ hệ thống.

– **Trách nhiệm:**

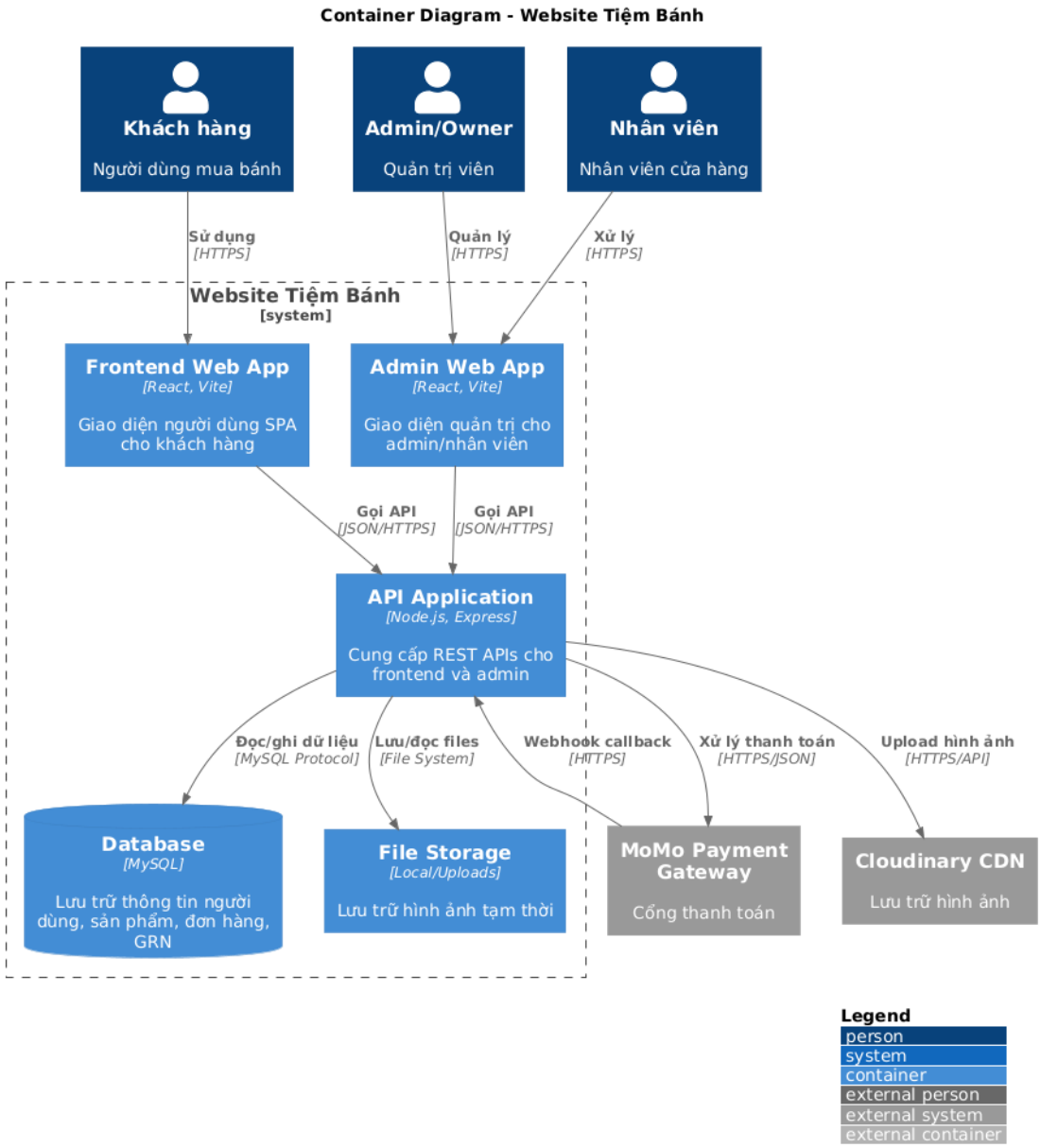
- **Xử lý Logic:** Tính toán tổng tiền, kiểm tra tồn kho xử lý logic nhập kho (GRN) và quản lý hạn sử dụng (Expiry Date).
- **Bảo mật:** Xác thực người dùng và phân quyền dựa trên bảng Admin_users và Users.
- **Kết nối Cổng thanh toán:** Tích hợp API với **MoMo** để xử lý link thanh toán và nhận Callback.
- **Quản lý Media:** Gọi API sang **Dịch vụ lưu trữ hình ảnh** (ví dụ: Cloudinary) để upload ảnh bánh.
- **Thông báo:** Kết nối **SMTP Server** để gửi email xác nhận đơn hàng hoặc reset mật khẩu.

– **Giao tiếp:** Nhận request từ Web App (JSON/REST API) và truy vấn dữ liệu từ Database.

Database

– **Chức năng:** Lưu trữ toàn bộ dữ liệu bền vững của hệ thống.
– **Cấu trúc:** Dựa trên thiết kế ERD đã xây dựng, bao gồm các bảng chính:
  - *Nhóm Bán hàng:* Orders, Order_Items, Reviews.
  - *Nhóm Kho:* Items, Suppliers, GRN_headers, GRN_details.
  - *Nhóm Người dùng:* Users, Admin_users.
– **Tương tác:** Chỉ cho phép API Application truy cập (Đọc/Ghi), không mở kết nối trực tiếp ra Internet.

Sơ đồ C2 – Container



Hình 32. Sơ đồ C2 – Container

Hệ thống được thiết kế theo mô hình **Client-Server** hiện đại, tách biệt hoàn toàn giữa Frontend và Backend. Kiến trúc này sử dụng **Single Page Application (SPA)** cho phía người dùng để tối ưu trải nghiệm và tập trung xử lý logic nghiệp vụ tại API trung tâm.

Hệ thống bao gồm 5 container chính nằm trong phạm vi của "Website Tiệm Bánh":

Frontend Web App

- **Công nghệ:** React, Vite.
- **Chức năng:** Là giao diện SPA dành riêng cho **Khách hàng**.
- **Trách nhiệm:** Hiển thị danh sách bánh, giỏ hàng và thực hiện quy trình đặt hàng. Ứng dụng này giao tiếp với Backend thông qua giao thức **JSON/HTTPS**.

Admin Web App

- **Công nghệ:** React, Vite.
- **Chức năng:** Là giao diện quản trị dành cho **Admin/Owner** và **Nhân viên cửa hàng**.
- **Trách nhiệm:** Cung cấp các công cụ quản lý sản phẩm, xử lý đơn hàng, nhập kho và xem báo cáo. Tương tự như ứng dụng khách hàng, nó kết nối với API qua **JSON/HTTPS**.

API Application

- **Công nghệ:** Node.js, Express.
- **Chức năng:** Là trung tâm xử lý logic của toàn bộ hệ thống, cung cấp **REST APIs** cho cả hai ứng dụng Frontend và Admin.
- **Trách nhiệm:**
    - Xử lý nghiệp vụ (Business Logic).
    - Điều phối luồng dữ liệu giữa Database, File Storage và các dịch vụ bên ngoài.
    - Xử lý thanh toán và webhook từ MoMo.
    - Quản lý upload hình ảnh lên Cloudinary.

Database

- **Công nghệ:** MySQL.
- **Chức năng:** Lưu trữ dữ liệu có cấu trúc của hệ thống.
- **Dữ liệu lưu trữ:** Thông tin người dùng, sản phẩm, đơn hàng, và phiếu nhập kho (GRN).

– **Giao tiếp:** API Application đọc/ghi dữ liệu thông qua **MySQL Protocol**.

File Storage

– **Công nghệ:** Local/Uploads (File System).

– **Chức năng:** Lưu trữ hình ảnh tạm thời trước khi xử lý hoặc upload lên cloud. API Application tương tác trực tiếp qua hệ thống tệp (Lưu/đọc files).

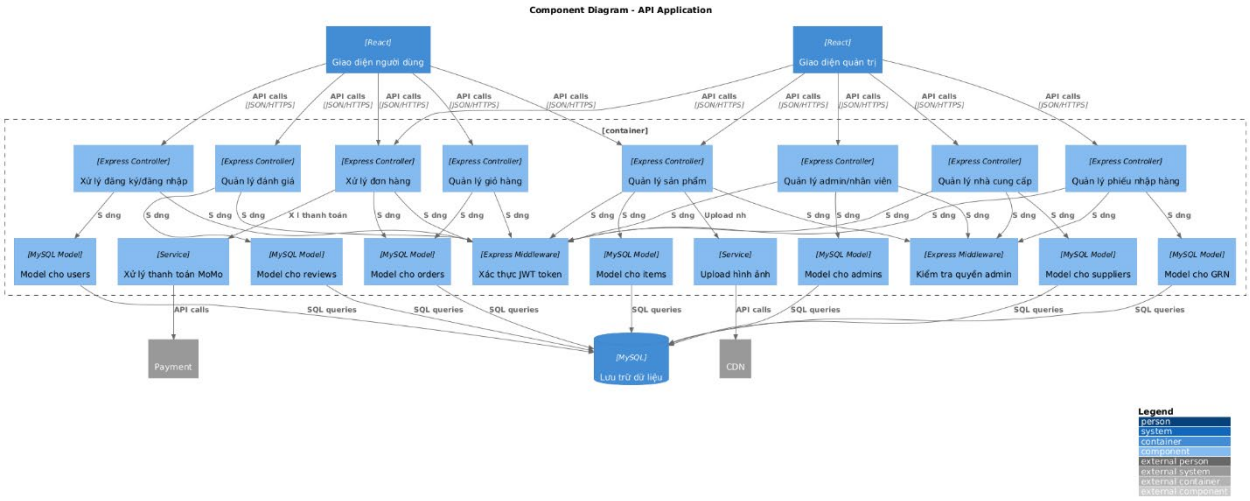
Tích hợp hệ thống bên ngoài

Hệ thống kết nối với các dịch vụ thứ ba để mở rộng tính năng:

– **MoMo Payment Gateway:** Xử lý giao dịch thanh toán trực tuyến. API gọi sang MoMo để tạo giao dịch và nhận **Webhook callback** qua HTTPS để cập nhật trạng thái đơn hàng.

– **Cloudinary CDN:** Dịch vụ lưu trữ và phân phối hình ảnh. API thực hiện **Upload hình ảnh** qua HTTPS/API để giảm tải cho server chính.

Sơ đồ C3 – Component



Hình 33. Sơ đồ C3 – Component

Hệ thống được chia thành 3 tầng xử lý chính:

Tầng Controller

Đây là nơi đầu tiên tiếp nhận các yêu cầu API từ Frontend (React) thông qua giao thức JSON/HTTPS. Các controller được chia theo chức năng:

– Nhóm phục vụ Người dùng:

- Xử lý đăng ký/đăng nhập: Quản lý xác thực người dùng ban đầu.
- Quản lý giỏ hàng & Đơn hàng: Tiếp nhận yêu cầu đặt hàng, tính toán giỏ hàng từ phía người mua.
- Quản lý đánh giá: Cho phép người dùng gửi nhận xét về sản phẩm.

– **Nhóm phục vụ Quản trị:**

- Quản lý sản phẩm: Thêm/sửa/xóa thông tin bánh.
- Quản lý nhà cung cấp & Phiếu nhập: Xử lý nghiệp vụ nhập kho từ Supplier.
- Quản lý admin/nhân viên: Quản lý tài khoản nội bộ.

Tầng Middleware & Security

Các thành phần này đứng giữa Request và Controller để đảm bảo an toàn hệ thống:

– Xác thực JWT token: Kiểm tra tính hợp lệ của token đăng nhập trước khi cho phép truy cập tài nguyên.

– Kiểm tra quyền admin: Phân quyền, đảm bảo chỉ nhân viên có thẩm quyền mới được truy cập các API quản lý.

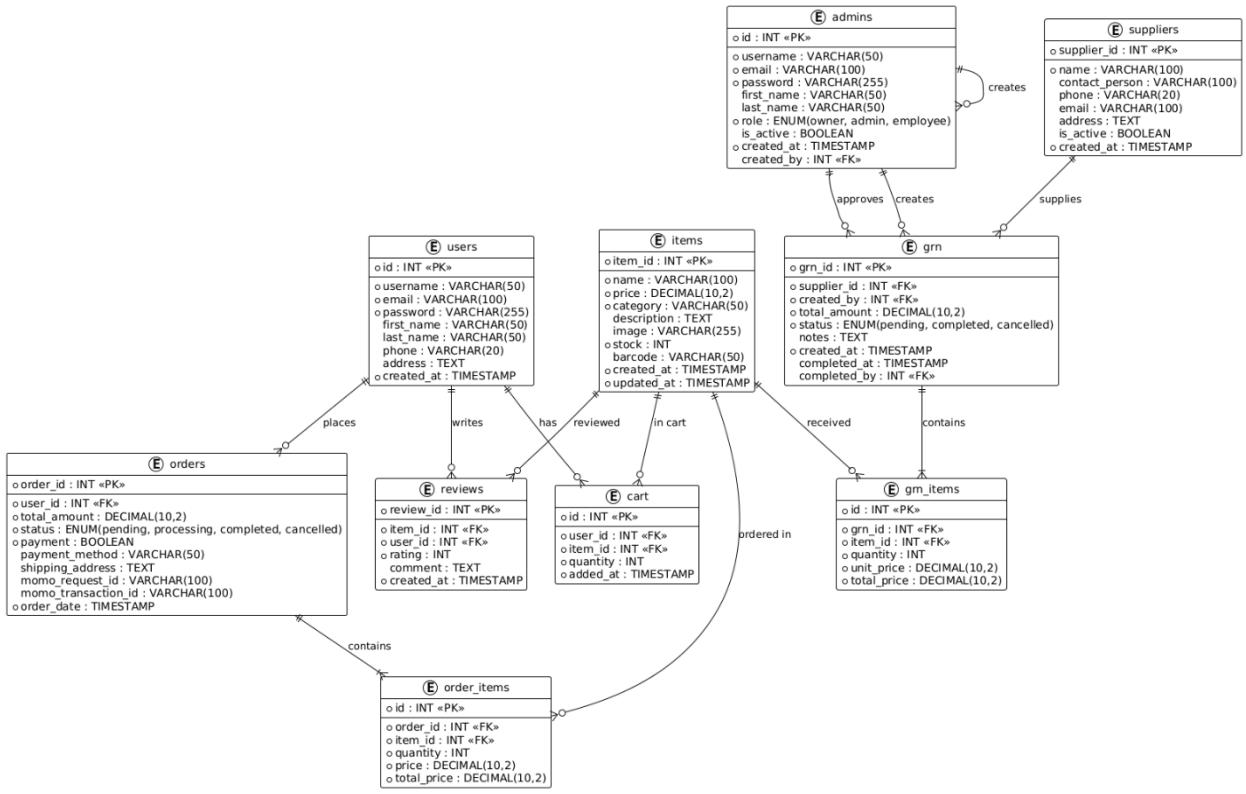
Tầng Service & Model

– MySQL Models: Các thành phần này (Model cho users, items, orders, GRN, suppliers...) chịu trách nhiệm giao tiếp trực tiếp với Cơ sở dữ liệu MySQL thông qua các câu truy vấn SQL (SQL queries).

– Integration Services:

- Xử lý thanh toán MoMo: Component chuyên biệt để gọi API sang cổng thanh toán MoMo.
- Upload hình ảnh: Component xử lý việc upload file sang hệ thống CDN (Cloudinary).

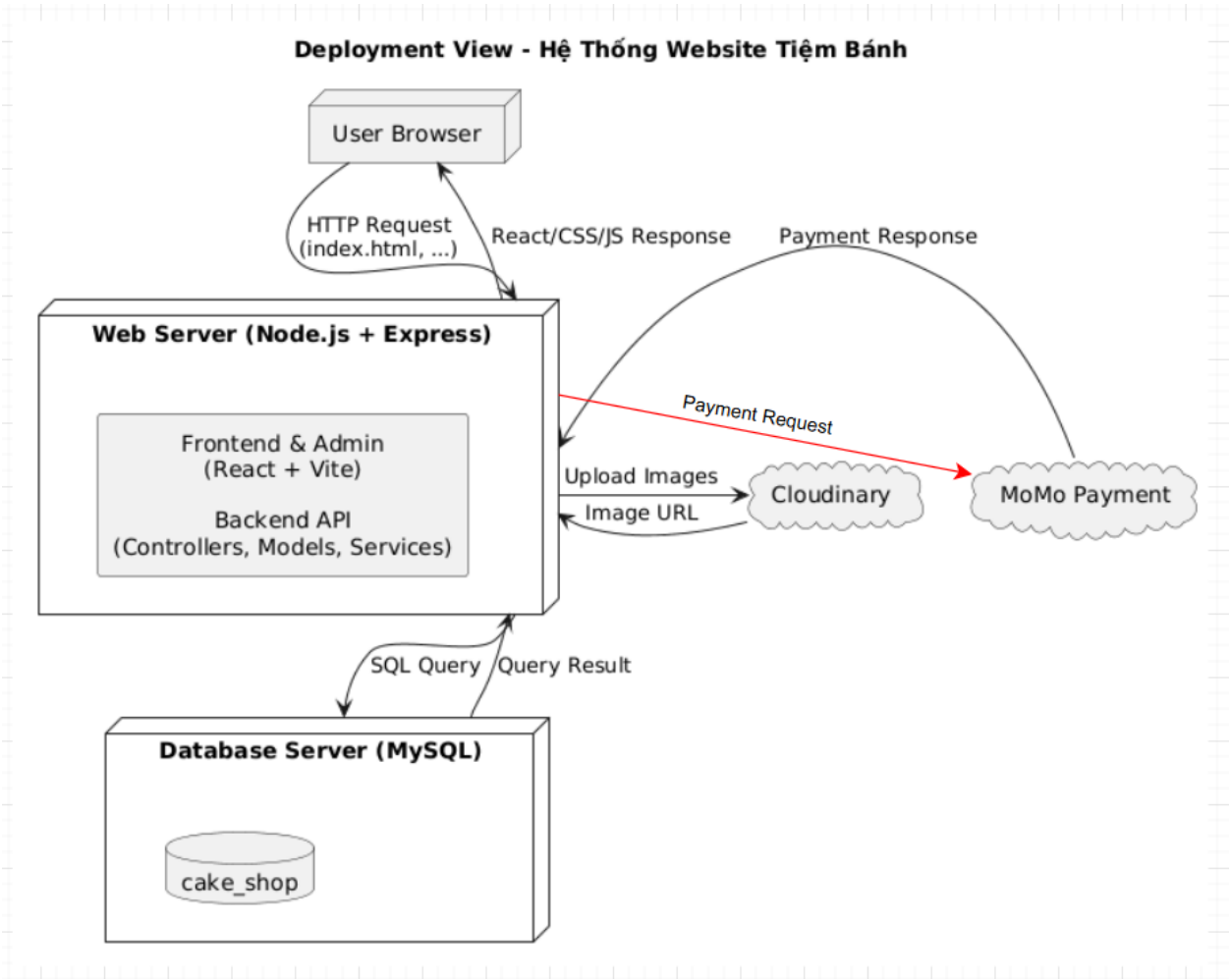
Sơ đồ C4 – Code



Hình 34. Sơ đồ C4 – Code

Phần này mô tả chi tiết thiết kế hướng đối tượng cho **Module Đơn hàng**. Đây là thành phần chịu trách nhiệm tiếp nhận yêu cầu đặt hàng, tính toán giá trị, tương tác với cổng thanh toán và lưu trữ dữ liệu bền vững.

2.3.5. Deployment View



Hình 35. Deployment View của hệ thống

Sơ đồ triển khai mô tả cấu hình phần cứng và môi trường thực thi của hệ thống Website Tiệm Bánh. Hệ thống được triển khai theo mô hình Client-Server 3 tầng điển hình, bao gồm: Tầng Client, Tầng Ứng dụng và Tầng Dữ liệu.

CHƯƠNG 3: THIẾT KẾ KIỂM THỬ

3.1. Tổng quản

3.1.1. Mục tiêu

Tài liệu Kế hoạch Kiểm thử cho dự án Cake Fantasy Hệ thống quản lý tiệm bánh được xây dựng nhằm mục đích sau:

- Xác định phạm vi kiểm thử và các thành phần phần mềm của hệ thống Cake Fantasy cần được kiểm thử, bao gồm Frontend dành cho khách hàng (React + Vite), Admin Panel (React + Vite), Backend API (Node.js + Express) và cơ sở dữ liệu MySQL, cùng các tích hợp bên ngoài (Cloudinary cho lưu trữ ảnh, MoMo cho thanh toán, và dịch vụ lưu trữ uploads cục bộ).
- Liệt kê và làm rõ các yêu cầu chức năng cấp cao cần được kiểm thử để đảm bảo hệ thống vận hành đúng nghiệp vụ trên tất cả các phân hệ cốt lõi. Cụ thể các phân hệ và chức năng chính bao gồm:
    - Quản lý danh mục sản phẩm và thuộc tính (danh mục, thương hiệu, màu sắc, kích cỡ).
    - Quản lý sản phẩm và biến thể (CRUD sản phẩm, upload ảnh, quản lý *cloudinary_id*).
    - Giỏ hàng (thêm/sửa/xóa sản phẩm, đồng bộ trạng thái giữa client và server).
    - Quản lý đơn hàng (tạo đơn, xem lịch sử, chi tiết đơn, cập nhật trạng thái, hủy đơn).
    - Thanh toán (hỗ trợ MoMo integration và mock payment service, xử lý callback/IPN, timeout/auto-cancel).
    - Quản lý tồn kho & biến thể (trừ tồn khi đơn được xác nhận, hoàn trả khi hủy đơn, cập nhật từ GRN).
    - Quản lý nhà cung cấp và phiếu nhập hàng.
    - Đánh giá & bình luận sản phẩm (reviews, rating).
    - Quản lý người dùng & phân quyền truy cập (User & Admin authentication, JWT).
- Đề xuất và mô tả các chiến lược kiểm thử sẽ được áp dụng trong dự án, gồm:
    - Kiểm thử chức năng để xác minh từng yêu cầu nghiệp vụ và hành vi hệ thống.

- Kiểm thử giao diện người dùng (UI Testing) cho các luồng khách hàng và admin, bao gồm xác thực hiển thị, validation và trải nghiệm responsive.
- Kiểm thử tích hợp (Integration Testing) tập trung vào backend API, tương tác với MySQL và dịch vụ bên ngoài (Cloudinary, MoMo — sử dụng mock khi cần).
- Kiểm thử đầu-cuối (End-to-End Testing) với Cypress để đảm bảo các luồng người dùng chính (tìm kiếm sản phẩm, đặt hàng, thanh toán) hoạt động xuyên suốt.
- Kiểm thử khả năng tương thích trên các trình duyệt chính (Chrome, Firefox, Edge) và trên các kích thước màn hình khác nhau (desktop, tablet, mobile).

– Xác định các nguồn lực cần thiết cho hoạt động kiểm thử:
  - Nhân sự: QA (1–2), lập trình viên backend/frontend hỗ trợ, người quản lý dự án/PO để giải đáp yêu cầu nghiệp vụ.
  - Công cụ và môi trường: Node.js/npm, Docker/Docker Compose (chạy toàn bộ stack), Jest & supertest (backend unit/integration), Vitest (frontend), Cypress (E2E), Postman (API manual), DBeaver/MySQL Workbench (DB inspection), GitHub Actions (CI), mock tools (nock hoặc local mock servers).
  - Tài nguyên hạ tầng: máy local hoặc runner CI có khả năng chạy Docker; biến môi trường và secrets (không commit vào repo).

– Đưa ra ước tính khối lượng công việc kiểm thử (dự kiến sơ bộ):
  - Soạn test case cho các chức năng critical (order, payment, inventory): 2–3 ngày.
  - Viết unit/integration tests (backend): 2–4 ngày tùy module và coverage mục tiêu.
  - Viết E2E tests (Cypress) cho các luồng chính: 3–5 ngày.
  - Thực thi test và xử lý defect (một vòng): 2–3 ngày.
  - Lưu ý: ước tính phụ thuộc vào nguồn lực thực tế và mức độ chi tiết test yêu cầu.

– Liệt kê các sản phẩm bàn giao sẽ được tạo ra sau quá trình kiểm thử:
  - Bộ test case chi tiết (Excel/Google Sheets hoặc file quản lý test), bao gồm traceability tới yêu cầu.

- Báo cáo thực thi kiểm thử sau mỗi đợt test, chứa số lượng TC pass/fail, danh sách defect và logs.
- Nhật ký lỗi (Defect Log) quản lý bằng GitHub Issues hoặc công cụ quản lý bug, có phân loại severity và trạng thái fix.
- Ma trận bao phủ yêu cầu liên kết các yêu cầu chức năng với test case tương ứng.
- Báo cáo tổng kết kiểm thử cuối cùng (Test Summary Report) bao gồm kết quả tổng hợp, các lỗi còn tồn, rủi ro và khuyến nghị trước khi release.

3.1.2. Định nghĩa, Thuật ngữ và Ký hiệu

Thuật ngữ	Ý nghĩa / Định nghĩa
Cake Fantasy	Tên dự án — Hệ thống quản lý tiệm bánh (Customer site + Admin + Backend)
SUT	System Under Test — Hệ thống đang được kiểm thử (Cake Fantasy)
Frontend	Phần giao diện người dùng (Customer site) — React + Vite
Admin Panel	Giao diện quản trị nội bộ — React + Vite
Backend	Phần máy chủ (API) — Node.js + Express
DB / MySQL	Database Management System — MySQL 8.0 dùng để lưu trữ dữ liệu ứng dụng
API	Application Programming Interface — các endpoint REST do Backend cung cấp
JWT	JSON Web Token — phương thức xác thực token-based
MoMo	Cổng thanh toán MoMo (tích hợp/ mock trong dự án)
Cloudinary	Dịch vụ lưu trữ và quản lý ảnh (external service)
GRN	Goods Receipt Note — Phiếu nhập hàng (cập nhật tồn kho)

Thuật ngữ	Ý nghĩa / Định nghĩa
CRUD	Create, Read, Update, Delete — Các thao tác cơ bản trên dữ liệu
E2E	End-to-End Testing — Kiểm thử đầu-cuối (Cypress)
CI/CD	Continuous Integration / Continuous Delivery — GitHub Actions trong dự án
TC	Test Case — Trường hợp kiểm thử
UAT	User Acceptance Testing — Kiểm thử chấp nhận người dùng
DBMS	Database Management System — Hệ quản trị cơ sở dữ liệu
REST	Representational State Transfer — Kiến trúc API được sử dụng
Seed	Script seed dữ liệu (seed.js) dùng để khởi tạo dữ liệu test mẫu
Docker Compose	Công cụ để khởi dựng toàn bộ stack (Backend + Frontend + MySQL) cho local
Nodemon	Công cụ hỗ trợ tự động restart backend trong quá trình phát triển
Supertest	Thư viện dùng để viết integration test cho HTTP API (backend)
Jest	Framework kiểm thử cho backend (unit/integration)
Vitest	Framework kiểm thử cho frontend (unit)
Cypress	Công cụ E2E test cho các luồng UI

Thuật ngữ	Ý nghĩa / Định nghĩa
Defect	Lỗi phần mềm được log trong defect log / GitHub Issues
Traceability Matrix	Ma trận liên kết giữa yêu cầu và test case (REQ ↔ TC)
Seed data	Dữ liệu mẫu được dùng cho test (import từ sql hoặc seed.js)

3.1.3. Tài liệu tham khảo

STT	Tên tài liệu / Tài nguyên	Mô tả	Đường dẫn (trong repo)
1	Repository README	Tổng quan dự án, hướng dẫn cài đặt, ports, CI/CD, testing overview	README.md
2	Test Design	Tài liệu thiết kế kiểm thử, chiến lược và ma trận	docs/Test Design.xlsx
3	Test Case (Backend & Integration)	Bộ test case chi tiết cho backend (unit/integration)	Test_Case.xlsx
4	Test Case E2E	Bộ test case cho E2E (Cypress) / kịch bản luồng người dùng	Test_Case_E2E.xlsx
5	Backend tests	Thư mục chứa unit & integration tests cho backend	tests (files: auth.test.js, cart.test.js, order.test.js, search.test.js, setup.js, testUtils.js)

STT	Tên tài liệu / Tài nguyên	Mô tả	Đường dẫn (trong repo)
6	Frontend unit tests	Test components frontend (Vitest)	__tests__
7	Admin unit tests	Test components admin panel (Vitest)	__tests__
8	Frontend package.json	Scripts & dependencies cho Cypress / vitest	package.json
9	Backend package.json & jest config	Scripts & cấu hình Jest cho chạy test, coverage	package.json, jest.config.cjs
10	Seed script	Script seed dữ liệu mẫu để phục vụ test/integration	seed.js
11	SQL schema	Dùng để reset/import database cho test	cake_fantasy_db.sql
12	Docker Compose	File để khởi dựng toàn bộ stack phục vụ integration/E2E testing	docker-compose.yml
13	Mock Payment Service	Dùng để mô phỏng responses thanh toán trong test flows	mockPaymentService.js
14	MoMo Payment Service	Implementation/adapter cho MoMo (dùng trong test/integration)	momoPaymentService.js
15	Cloudinary config	Cấu hình kết nối Cloudinary (liên quan upload image tests)	cloudinary.js

STT	Tên tài liệu / Tài nguyên	Mô tả	Đường dẫn (trong repo)
16	API controllers / routes	Tham chiếu các endpoint để viết API tests	controllers, routes
17	CI Workflows	GitHub Actions workflows có job liên quan test/build	workflows
18	Backend .github README	Hướng dẫn config CI và secrets cho test trên CI	README.md
19	Docs folder	Chứa các tài liệu test case, test design và các báo cáo	docs
20	Dockerfiles	Docker configuration cho từng service (test in container)	Dockerfile

3.1.4. Bối cảnh

Cake Fantasy là một hệ thống quản lý tiệm bánh toàn diện bao gồm website bán hàng cho khách, giao diện quản trị nội bộ (Admin Panel) và Backend API phục vụ lưu trữ, xử lý nghiệp vụ và kết nối cơ sở dữ liệu. Hệ thống được phát triển dưới dạng monorepo với ba ứng dụng độc lập nhưng tích hợp chặt chẽ:

– Công nghệ chính:

- Frontend: React + Vite (port local: 5173).
- Admin Panel: React + Vite (port local: 5174).
- Backend: Node.js + Express, triển khai dạng ESM (main: *server.js*) (port local: 4000).
- Database: MySQL 8.0 (local container thường dùng port 3306).

– Dịch vụ bên thứ ba và hỗ trợ:

- Cloudinary: lưu trữ và quản lý ảnh sản phẩm (sử dụng public_id/secure_url).
- MoMo: cổng thanh toán (dự án có *momoPaymentService.js* và *mockPaymentService.js* để test).
- Local uploads: thư mục *uploads*/ chứa ảnh mẫu.

– Công cụ & test infra đã có trong repo: Jest + supertest (backend unit/integration), Vitest (frontend/admin), Cypress (E2E) — cùng các file cấu hình và test scripts trong *tests*, *__tests__*.

– Khả năng chạy toàn bộ stack: Docker Compose (*docker-compose.yml*) hỗ trợ khởi tạo Backend + Frontend + Admin + MySQL cho local integration tests.

– Các tính năng nghiệp vụ nổi bật: quản lý sản phẩm/biến thể, giỏ hàng, tạo & quản lý đơn hàng, tích hợp thanh toán (MoMo hoặc mock), quản lý tồn kho tự động (trừ khi order CONFIRMED; hoàn trả khi CANCELLED), quản lý Supplier và GRN (phiếu nhập hàng), review/rating sản phẩm, phân quyền admin/user.

– Hành vi tự động quan trọng: auto-cancel đơn MoMo sau 15 phút nếu chưa thanh toán; scheduler kiểm tra trạng thái đơn định kỳ.

3.1.5. Phạm vi kiểm thử

3.1.5.1. Phạm vi

Kế hoạch kiểm thử này bao phủ toàn bộ các hoạt động kiểm thử nhằm đảm bảo các luồng nghiệp vụ chính của Cake Fantasy hoạt động đúng như yêu cầu. Cụ thể:

– Authentication & Authorization

- Đăng ký, đăng nhập, đăng xuất (user & admin)
- Phân quyền route/role (admin vs user)
- Quản lý JWT, token expiry

– Product Management

- CRUD sản phẩm
- Upload / cập nhật ảnh (Cloudinary integration, lưu *cloudinary_id*)
- Tìm kiếm, lọc, hiển thị chi tiết sản phẩm

- Shopping Cart
    - Thêm/xóa/cập nhật số lượng
    - Đồng bộ cart giữa client và server
    - Validation tồn kho khi thêm vào giỏ
- Order Management
    - Tạo đơn (COD & MoMo), xem chi tiết đơn
    - Cập nhật trạng thái đơn (Pending, Confirmed, Cancelled, Paid)
    - Hủy đơn thủ công và tự động (auto-cancel thanh toán MoMo)
    - Retry payment flow cho đơn pending
- Payment Integration
    - Tạo payment request (MoMo) và xử lý callback/IPN (mock khi cần)
    - Xử lý timeout (auto-cancel sau 15 phút) và trạng thái thanh toán
- Inventory & GRN
    - Trừ tồn khi đơn được CONFIRMED
    - Hoàn trả tồn khi đơn CANCELLED
    - Tạo GRN và cập nhật tồn kho tương ứng
- Supplier Management
    - CRUD nhà cung cấp (Supplier)
- Reviews & Ratings
    - Thêm / sửa / xóa đánh giá; hiển thị rating trung bình
- API & Integration Testing
    - Kiểm tra tất cả các REST endpoints chính (/api/user, /api/item, /api/cart, /api/order, /api/review, v.v.) bằng integration tests (supertest/Postman).
    - Kiểm tra luồng end-to-end (Cypress) cho các kịch bản chính: tìm kiếm → thêm giỏ → đặt hàng → thanh toán.

- Database Testing
    - Kiểm tra consistency sau các thao tác (orders, order_items, inventory, grn) bằng truy vấn DB trực tiếp (DBeaver/MySQL Workbench).
- Other test types included
    - Functional Testing, UI Testing (component và tương tác), Integration Testing, API Testing, E2E Testing, Compatibility Testing (Chrome/Firefox/Edge, responsive view), Regression Testing.

3.1.5.2. Out of Scope (Ngoài phạm vi)

Phần kiểm thử hiện tại không bao gồm các hoạt động sau (sẽ thực hiện ở các giai đoạn sau hoặc không nằm trong phạm vi đồ án):

- Performance / Load / Stress Testing quy mô lớn (JMeter, kịch bản concurrent users).
- Penetration testing chuyên sâu và vulnerability scanning (security audit).
- Kiểm thử native mobile apps (chỉ test responsive web).
- Kiểm thử nội bộ của third-party services (Cloudinary, MoMo, SendGrid).
- Disaster recovery drills và backup/restore production-scale.
- Kiểm thử quốc tế hóa (i18n) và accessibility (WCAG) toàn diện.

3.1.6. Các ràng buộc

1. Ràng buộc về nguồn lực

- Nhân lực: đội QA hạn chế (1–2 tester) — developer hỗ trợ khi cần.
- Thiết bị: kiểm thử chủ yếu trên máy cá nhân/lab; không có lab thiết bị di động chuyên dụng.
- Thời gian: tiến độ đồ án giới hạn, ưu tiên test các luồng nghiệp vụ quan trọng.

2. Ràng buộc môi trường

- Môi trường chính để test là local (Docker Compose) và các preview/staging deploy nếu có.

- Một số dịch vụ (MoMo sandbox, Cloudinary) có thể không ổn định hoặc bị giới hạn; khi đó sử dụng mock services (*mockPaymentService.js*) để kiểm thử.
- Chưa có staging environment hoàn chỉnh cho mọi team; nhiều test integration sẽ chạy trên môi trường local.

3. Ràng buộc kỹ thuật

- Một số biến môi trường/secret (JWT_SECRET, Cloudinary keys, MOMO keys) phải được cấu hình thủ công trong *.env* local - không được commit vào repo.
- DB reset và seeding cần thực hiện thủ công hoặc qua seed.js trước mỗi test cycle để đảm bảo tính tái lập.
- Scheduler (ví dụ auto-cancel payment) chạy theo lịch; để test behavior time-dependent có thể cần thay đổi thời gian hoặc mock scheduler.
- Lazy loading / JPA tương đương - trong dự án Node, cần chú ý các truy vấn liên quan quan hệ khi test integration để tránh lỗi thiếu dữ liệu do lazy-loading-like behavior hoặc transaction scope.

4. Ràng buộc dữ liệu

- Dữ liệu test phải được tách biệt khỏi dữ liệu production; sử dụng seed scripts hoặc SQL dump (*cake_fantasy_db.sql*).
- Không sử dụng dữ liệu sản xuất trực tiếp cho test (vì lý do bảo mật).
- Không có generator dữ liệu tự động phức tạp; cần chuẩn bị dataset thủ công cho các kịch bản đặc thù.

5. Ràng buộc phạm vi và lịch

- Do giới hạn nguồn lực và thời gian cho đồ án, một số edge-case hoặc permutations hiếm gặp sẽ không được kiểm thử chi tiết.
- Compatibility testing chỉ thực hiện trên các trình duyệt chính và phiên bản gần đây; không đảm bảo tương thích với trình duyệt cũ.

3.1.7. Rủi ro

ID	Rủi ro	Mức độ	Tác động	Biện pháp giảm thiểu
R01	MoMo Sandbox không ổn định	High	Không thể test luồng thanh toán MoMo, làm chậm tiến độ test payment	- Chuẩn bị test data cho cả COD và MoMo - Kiểm tra trạng thái MoMo Sandbox trước khi test -Có plan B: sử dụng *mockPaymentService.js* để mock MoMo responses
R02	Secrets/keys lộ trong repo	High	Có thể dẫn tới rò rỉ dữ liệu, truy cập trái phép	- Remove keys khỏi repo và rotate/revoke keys đã lộ - Sử dụng env vars & GitHub Secrets cho CI/CD - Thêm secret-scanning và .gitignore kiểm soát
R03	Inventory không được restore khi order bị hủy	High	Mất dữ liệu tồn kho, ảnh hưởng nghiệp vụ và trải nghiệm khách	- Viết test case kiểm tra inventory restore (manual cancel & auto-cancel) - Kiểm tra log của scheduler và transaction handling - Đảm bảo transactional integrity (rollback) trong code
R04	Thay đổi requirements giữa chừng	High	Test case phải viết lại, mất thời gian, gây trễ tiến độ	- Freeze requirements trước khi bắt đầu test cycle - Mọi change request phải qua PM/PO phê duyệt - Thực hiện regression testing sau mỗi thay đổi

ID	Rủi ro	Mức độ	Tác động	Biện pháp giảm thiểu
R05	Test data không nhất quán	Medium	Kết quả test không tin cậy, false positives/negatives	- Sử dụng SQL script / *seed.js* để reset DB trước mỗi test cycle - Document rõ test data requirements và chuẩn schema - Lưu backup DB state và tái tạo nhanh
R06	Thiếu nhân sự kiểm thử	Medium	Không đủ thời gian test toàn bộ use case, giảm coverage	- Ưu tiên test các critical flows (checkout, payment, inventory) - Dev hỗ trợ test API & viết unit tests - Lên kế hoạch automation cho sau này
R07	Browser compatibility issues	Medium	UI hiển thị sai/không tương thích trên một số trình duyệt	- Test trên tối thiểu Chrome, Firefox, Edge - Sử dụng responsive testing và CSS prefixes/polyfills khi cần - Kiểm thử trên các viewport chính (desktop/tablet/mobile)
R08	Third-party service downtime (Cloudinary, SendGrid)	Low	Một số tính năng (upload ảnh, email) không test được tạm thời	- Mock third-party responses khi cần (nock hoặc mock server) - Test offline scenarios và ghi chú dependency trong test plan

ID	Rủi ro	Mức độ	Tác động	Biện pháp giảm thiểu
R09	Port conflict / environment unstable	Low	Không thể khởi động service, làm gián đoạn test	- Document các port được sử dụng (Backend 4000, Frontend 5173, Admin 5174, MySQL 3306) - Kiểm tra port availability trước khi start - Dùng Docker Compose để isolate services
R10	Database connection timeout / DB instability	Low	Test bị lỗi do mất kết nối, ảnh hưởng kết quả	- Configure connection pool & retry logic - Restart Docker container khi cần - Monitor MySQL logs và tăng timeouts trong config test
R11	Session timeout khi test	Low	Phải login lại nhiều lần, gây gián đoạn E2E test	- Tăng session TTL trong môi trường test hoặc dùng test accounts - Sử dụng token refresh script trong test runner
R12	Deployment / CI issues	Medium	Không thể deploy preview/staging, CI fail, chậm release	- Kiểm thử deployment trên local trước khi push - Document deployment steps và CI variables - Sử dụng Docker để đảm bảo consistency trong CI

Mức độ rủi ro:

- **High**: cần xử lý ngay
- **Medium**: theo dõi và có kế hoạch dự phòng
- **Low**: chấp nhận hoặc xử lý khi cần.

3.2. Chi tiết kế hoạch kiểm thử

3.2.1. Kiểm thử hộp trắng

Mục tiêu: Kiểm tra cấu trúc mã nguồn và luồng xử lý nội bộ của Backend (Node.js + Express). Đảm bảo các logic nghiệp vụ, các nhánh điều kiện (if/else), vòng lặp và cơ chế xử lý lỗi (try/catch) được thực thi chính xác.

Phạm vi áp dụng:

- *Unit Tests:* Kiểm tra các hàm tiện ích (utils), validators, và logic nghiệp vụ độc lập (ví dụ: tính tổng tiền đơn hàng, kiểm tra mã giảm giá).
- *Integration Tests:* Kiểm tra sự tương tác giữa các lớp Controller → Service → Model và Cơ sở dữ liệu (MySQL). Bao gồm kiểm tra các tác động (side-effects) như tạo đơn hàng, trừ tồn kho, và giao tiếp với dịch vụ bên ngoài (Cloudinary, MoMo) thông qua Mocking.

Chiến lược thiết kế:

- Tập trung bao phủ các trường hợp biên và luồng lỗi.
- Sử dụng chỉ số Code Coverage để đánh giá, ưu tiên đạt độ bao phủ cao cho các module quan trọng như Order và Payment.

3.2.2. Kiểm thử hộp đen

Mục tiêu: Kiểm tra chức năng và hành vi của hệ thống từ góc độ người dùng cuối mà không quan tâm đến cấu trúc mã bên trong.

Phạm vi áp dụng:

- *System & Acceptance Tests (E2E):* Kiểm thử các luồng nghiệp vụ hoàn chỉnh: Tìm kiếm → Thêm giỏ hàng → Đặt hàng → Thanh toán → Lịch sử đơn hàng.
- *UI Tests:* Kiểm tra tính hợp lệ của biểu mẫu (form validation), hiển thị thông báo lỗi và độ phản hồi của giao diện.

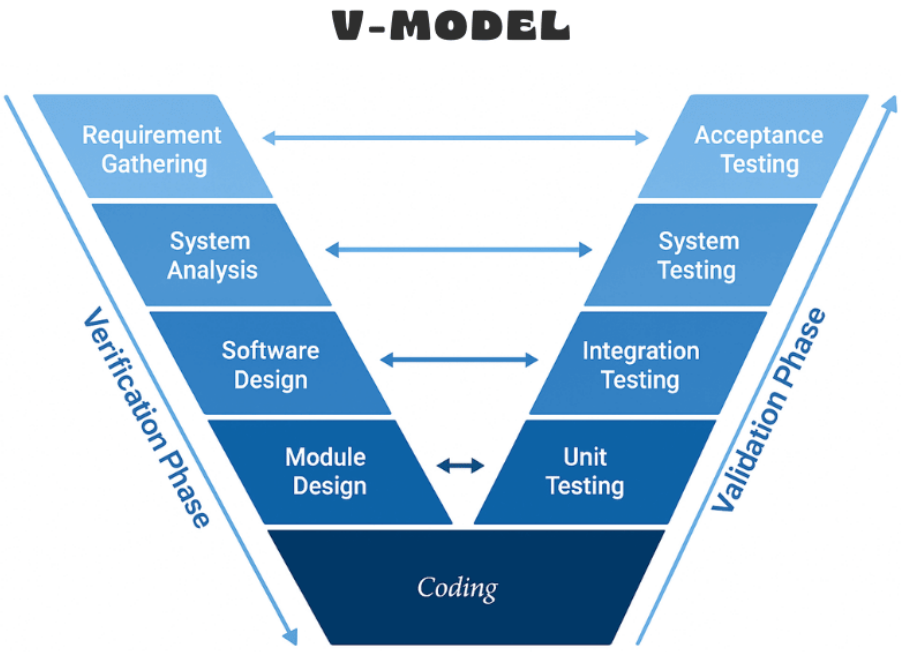
Chiến lược thiết kế:

- Xây dựng Test Case dựa trên User Stories và yêu cầu chức năng.
- Sử dụng kỹ thuật phân hoạch tương đương và phân tích giá trị biên.
- Sử dụng Cypress để thực hiện E2E testing gọi API thực tế.

3.2.3. Chiến lược kiểm thử theo mô hình V-Model

Dự án áp dụng mô hình **V-Model** để đảm bảo quy trình kiểm thử diễn ra song song và tương ứng chặt chẽ với quy trình phát triển. Điều này giúp phát hiện lỗi sớm và đảm bảo tính truy vết cao giữa yêu cầu và sản phẩm thực tế:

- **Phân tích yêu cầu ↔ Kiểm thử chấp nhận:** Các kịch bản kiểm thử luồng nghiệp vụ (End-to-End) được xây dựng dựa trên tài liệu yêu cầu để đảm bảo hệ thống đáp ứng đúng nhu cầu sử dụng thực tế của khách hàng.
- **Thiết kế hệ thống ↔ Kiểm thử tích hợp:** Kiểm tra sự giao tiếp giữa các thành phần kiến trúc (Frontend - Backend - Database) và các dịch vụ bên ngoài (Payment, Cloudinary) để đảm bảo hệ thống vận hành đồng bộ.
- **Thiết kế chi tiết ↔ Kiểm thử đơn vị:** Tương ứng với việc thiết kế từng module/hàm, các bài test đơn vị (Unit Tests) được viết ngay trong quá trình code để kiểm tra logic nội tại và xử lý lỗi ở mức độ nhỏ nhất.



Hình ?:

3.3. Môi trường kiểm thử

3.3.1. Công cụ sử dụng

Phân loại	Công cụ / Kỹ thuật	Ghi chú

Backend Testing	Jest, Supertest	Unit & Integration test cho Node.js API.
Frontend Testing	Vitest, Cypress	Component test & End-to-End testing.
Database Tool	MySQL Workbench / DBeaver	Kiểm tra dữ liệu trực quan.
Environment	Docker, Docker Compose	Đóng gói môi trường kiểm thử đồng nhất.
CI/CD	GitHub Actions	Tự động chạy test khi có commit mới.
Mocking	Jest Mocks, Mock Service	Giả lập MoMo/Cloudinary (*mockPaymentService*.js).

3.3.2. Dữ liệu kiểm thử

Dữ liệu kiểm thử (Test Data) Dữ liệu kiểm thử được chuẩn bị sẵn (Seeding) để đảm bảo tính nhất quán trước mỗi lần chạy test:

- Users: 10 tài khoản mẫu (bao gồm 1 Admin, 9 Customers).
- Items: 20 sản phẩm bánh với đa dạng thuộc tính (size, category).
- Khác: Danh mục (Categories), Nhà cung cấp (Suppliers), Phiếu nhập mẫu (GRN).
- *Cơ chế:* Sử dụng script seed.js để reset và nạp lại dữ liệu sạch vào Database test trước các chu kỳ Integration/E2E Test.

3.4. Thiết kế Test Case chi tiết

Đội dự án tập trung nguồn lực để kiểm thử sâu các chức năng có mức độ rủi ro cao và ảnh hưởng trực tiếp đến nghiệp vụ bán hàng như: *Xác thực người dùng, Giỏ hàng, Đặt hàng & Thanh toán, và Quản lý tồn kho*. Mục tiêu là tối đa hóa khả năng phát hiện lỗi nghiêm trọng trong thời gian phát triển giới hạn.

3.4.1. Cấu trúc Test Case chuẩn

Mỗi Test Case trong dự án được chuẩn hóa theo bảng dữ liệu gồm 8 trường thông tin sau:

Tên cột	Mô tả

ID	Mã định danh duy nhất (Ví dụ: TC_WB_01, TC_E2E_01).
Test Case Description	Mô tả ngắn gọn mục đích kiểm thử.
Test Case Procedure	Các bước thực hiện (Steps) để tái hiện kịch bản.
Expected Output	Kết quả mong đợi hệ thống trả về (UI, API status, Database).
Inter-test case Dependence	Sự phụ thuộc vào Test Case khác hoặc điều kiện tiên quyết (Pre-condition).
Result	Kết quả thực tế (*Pass / Fail / Pending*).
Test date	Ngày thực hiện kiểm thử.
Note	Ghi chú bổ sung hoặc link tới lỗi (Defect ID) nếu có.

3.4.2. Thiết kế Test Case cho Hộp trắng

Hệ thống Backend được phân chia thành 4 Module chính để thực hiện kiểm thử Đơn vị (Unit Test) và Tích hợp (Integration Test). Tổng số lượng Test Case được thiết kế là **64 test cases**, đảm bảo bao phủ các logic nghiệp vụ quan trọng và tính toàn vẹn của dữ liệu.

Module 1: Authentication & User Management

Mục tiêu: Kiểm tra tính bảo mật, quy trình xác thực và phân quyền người dùng.

Đăng ký:

- Kiểm tra đăng ký thành công với dữ liệu hợp lệ (Lưu password dạng hash).
- Kiểm tra lỗi khi **Email đã tồn tại** (Duplicate entry).
- Validate định dạng Email không hợp lệ (Missing '@', domain).
- Validate độ mạnh mật khẩu (Độ dài tối thiểu).
- Kiểm tra ràng buộc các trường bắt buộc (Missing required fields).

Đăng nhập:

- Đăng nhập thành công trả về **JWT Token** hợp lệ.
- Xử lý lỗi khi sai Email hoặc sai Mật khẩu.
- Kiểm tra cơ chế khóa tài khoản (nếu có logic thử sai nhiều lần).

Authorization & Token:

- Kiểm tra truy cập API bảo mật với Token hợp lệ.

- Xử lý lỗi khi Token hết hạn hoặc không đúng định dạng
- **Phân quyền:** Đảm bảo User thường không thể gọi API của Admin (Ví dụ: API xóa sản phẩm).

Module 2: Product Catalog & Search Engine

Mục tiêu: Kiểm tra các thao tác CRUD sản phẩm và độ chính xác của thuật toán tìm kiếm.

Quản lý sản phẩm:

- **Thêm mới:** Kiểm tra thêm sản phẩm thành công vào DB, xử lý upload ảnh lên Cloudinary.
- **Cập nhật:** Kiểm tra cập nhật thông tin (giá, tên) và thay đổi ảnh.
- **Xóa:** Kiểm tra xóa sản phẩm (Xóa mềm) và xóa ảnh tương ứng trên Cloudinary.

Tìm kiếm & Lọc:

- Tìm kiếm theo từ khóa chính xác và gần đúng.
- Tìm kiếm với từ khóa không tồn tại.
- Tìm kiếm với ký tự đặc biệt.
- Lọc sản phẩm theo Danh mục và khoảng giá.

Module 3: Shopping Cart & Order Processing

Mục tiêu: Đây là module quan trọng nhất, kiểm tra logic tính toán tiền và quy trình xử lý đơn hàng.

Giỏ hàng:

- Thêm sản phẩm mới vào giỏ.
- Thêm sản phẩm đã có trong giỏ (Cộng dồn số lượng).
- Validate số lượng tồn kho khi thêm (Không cho phép thêm quá số lượng tồn).
- Cập nhật số lượng: Test với số lượng âm, bằng 0 (Xóa khỏi giỏ), và số lượng cực lớn.

Đặt hàng:

- **Tính toán:** Kiểm tra hàm calculateTotal tính đúng tổng tiền = (Giá * Số lượng) + Phí Ship.
- **Tạo đơn:** Kiểm tra transaction tạo record trong bảng orders và order_items đồng thời.
- **Thanh toán:**
    - Tạo Signature thanh toán MoMo chính xác.
    - Xử lý Callback từ MoMo: Cập nhật trạng thái đơn hàng khi thanh toán thành công/thất bại.

Module 4: Inventory & Administration

Mục tiêu: Kiểm tra quy trình quản lý kho và các báo cáo thống kê dành cho Admin.

Quản lý Nhập kho:

- Tạo phiếu nhập kho mới: Kiểm tra ghi nhận nhà cung cấp và danh sách nguyên liệu.
- **Cập nhật tồn kho:** Đảm bảo khi phiếu nhập được duyệt, số lượng tồn kho của sản phẩm tăng lên tương ứng.

Quản lý Nhà cung cấp:

- Thêm/Sửa/Xóa thông tin nhà cung cấp.
- Ràng buộc toàn vẹn: Không cho phép xóa nhà cung cấp đã có lịch sử giao dịch nhập hàng.

Thống kê:

- Kiểm tra API thống kê doanh thu theo ngày/tháng (Aggregation queries).
- Kiểm tra đếm tổng số đơn hàng, khách hàng mới.

3.4.2. Thiết kế Test Case cho Backend

Hệ thống Backend được phân chia thành 4 Module chính để thực hiện kiểm thử Đơn vị (Unit Test) và Tích hợp (Integration Test). Tổng số lượng Test Case được

thiết kế là **64 test cases**, sử dụng **Jest Framework** để đảm bảo bao phủ các logic nghiệp vụ quan trọng và tính toàn vẹn của dữ liệu.

Module 1: User Management

Mục tiêu: Kiểm tra tính bảo mật, quy trình xác thực (Authentication) và phân quyền (Authorization) người dùng.

Chi tiết kịch bản kiểm thử:

- **Đăng ký:**
    - *Happy Path:* Đăng ký thành công với thông tin hợp lệ, mật khẩu được băm (hash) bằng bcrypt trước khi lưu vào DB.
    - *Validation:* Kiểm tra định dạng Email (thiếu @, domain sai), độ mạnh mật khẩu (độ dài tối thiểu), và các trường bắt buộc.
    - *Negative Case:* Xử lý lỗi khi Email đã tồn tại (Duplicate entry) - API trả về mã 400/409.
- **Đăng nhập:**
    - *Happy Path:* Đăng nhập đúng Email/Pass trả về **JWT Token** chứa thông tin userId và role.
    - *Negative Case:* Sai mật khẩu hoặc Email không tồn tại -> Trả về lỗi "Invalid credentials".
    - *Security:* Kiểm tra cơ chế bảo vệ chống Brute-force (nếu có logic khóa sau nhiều lần thử sai).
- **Authorization & Token:**
    - *Token Validation:* Gửi request kèm Token hợp lệ -> API xử lý thành công.
    - *Token Error:* Gửi request với Token hết hạn (Expired) hoặc chuỗi Token giả mạo (Malformed) -> API trả về 401/403.
    - *Role Based Access Control (RBAC):* Đảm bảo User thường không thể gọi API dành riêng cho Admin (ví dụ: DELETE /api/item/:id).

Module 2: Product Catalog & Search Engine

Mục tiêu: Kiểm tra các thao tác CRUD sản phẩm và độ chính xác của thuật toán tìm kiếm/lọc.

Chi tiết kịch bản kiểm thử:

- **Quản lý sản phẩm:**
    - *Create:* Thêm sản phẩm mới thành công, kiểm tra tích hợp **Mock Cloudinary** để giả lập upload ảnh (tránh gọi API thật khi test).
    - *Update:* Cập nhật thông tin (giá, tên, mô tả) và thay đổi ảnh đại diện.
    - *Delete:* Kiểm tra xóa sản phẩm (Soft delete hoặc Hard delete) và đồng bộ xóa ảnh trên Cloudinary.
- **Tìm kiếm & Lọc:**
    - *Keyword Search:* Tìm kiếm chính xác (Exact match) và tìm kiếm gần đúng (Partial match/Like query).
    - *Empty Result:* Tìm kiếm với từ khóa không tồn tại -> Trả về danh sách rỗng (Empty Array), không gây lỗi server.
    - *Advanced Filter:* Lọc sản phẩm theo nhiều tiêu chí đồng thời: Danh mục (category_id) và Khoảng giá (min_price, max_price).
    - *Security:* Kiểm tra đầu vào chứa ký tự đặc biệt để ngăn chặn SQL Injection cơ bản.

Module 3: Shopping Cart & Order Processing

Mục tiêu: Đây là module quan trọng nhất, kiểm tra logic tính toán tiền tệ chính xác và quy trình xử lý giao dịch đơn hàng.

Chi tiết kịch bản kiểm thử:

- **Giỏ hàng:**
    - *Add New:* Thêm sản phẩm chưa có vào giỏ -> Tạo dòng mới trong bảng cart.
    - *Update Quantity:* Thêm sản phẩm đã có -> Cộng dồn số lượng.
    - *Stock Validation:* Không cho phép thêm số lượng lớn hơn tồn kho hiện tại (Inventory Check).
    - *Edge Cases:* Cập nhật số lượng về 0 (Xóa khỏi giỏ), thử nhập số lượng âm (Validation Error).
- **Đặt hàng:**

- *Calculation:* Kiểm tra hàm calculateTotal tính đúng công thức: Tổng tiền = (Giá * Số lượng) + Phí Ship.
- *Transaction:* Kiểm tra tính nguyên vẹn (Atomicity) - đảm bảo record trong bảng orders và order_items được tạo đồng thời. Nếu một trong hai lỗi, toàn bộ transaction phải rollback.

– **Thanh toán:**

- *MoMo Signature:* Tạo chữ ký (Signature) chính xác theo thuật toán HMAC_SHA256 để gửi sang cổng thanh toán.
- *Callback Handling:* Xử lý IPN/Callback từ MoMo giả lập. Cập nhật trạng thái đơn hàng thành Paid (nếu thành công) hoặc Cancelled (nếu thất bại/timeout).

Module 4: Inventory & Administration

Mục tiêu: Kiểm tra quy trình quản lý nguồn lực (kho, nhà cung cấp) và các báo cáo thống kê doanh thu.

Chi tiết kịch bản kiểm thử:

– **Quản lý Nhập kho:**

- *Create GRN:* Tạo phiếu nhập kho ghi nhận Nhà cung cấp và danh sách nguyên liệu/sản phẩm nhập.
- *Stock Update:* Quan trọng - Đảm bảo khi phiếu nhập được duyệt, số lượng tồn kho (quantity trong bảng items) phải tăng lên tương ứng.

– **Quản lý Nhà cung cấp:**

- *CRUD:* Thêm/Sửa/Xóa thông tin nhà cung cấp.
- *Constraint Check:* Không cho phép xóa Nhà cung cấp đã có lịch sử giao dịch (Ràng buộc khóa ngoại).

– **Thống kê:**

- *Revenue Stats:* Kiểm tra API thống kê doanh thu theo ngày/tháng (sử dụng SQL Aggregation Queries SUM, COUNT).
- *Counters:* Kiểm tra số liệu đếm tổng quan: Tổng đơn hàng, Tổng khách hàng mới, Tổng sản phẩm bán chạy.

CHƯƠNG 4: QUY TRÌNH KIỂM THỬ

4.1. Tổng quan

Mục tiêu: Cốt lõi của kế hoạch kiểm thử là đảm bảo chất lượng toàn diện cho hệ thống Cake Fantasy trước khi bàn giao cho người dùng cuối. Quá trình kiểm thử tập trung vào việc xác minh sự tuân thủ của phần mềm đối với các yêu cầu nghiệp vụ đã đề ra, đồng thời đánh giá trải nghiệm người dùng nhằm giảm thiểu rủi ro vận hành và nâng cao độ tin cậy của hệ thống.

Phạm vi kiểm thử bao gồm các phân hệ quan trọng: Người dùng (Buyer), Quản trị viên (Admin), và các quy trình nghiệp vụ cốt lõi như: Giỏ hàng, Đặt hàng và Thanh toán.

4.2. Thiết kế theo khung nhìn V-Model

Hệ thống áp dụng tư duy của mô hình V-Model để ánh xạ trực tiếp giữa yêu cầu và kiểm thử, đảm bảo không có chức năng nào được phát triển mà thiếu sự kiểm chứng:

- **Kiểm thử đơn vị:** Tương ứng với giai đoạn Thiết kế chi tiết (Component Design). Do lập trình viên thực hiện để kiểm tra các hàm xử lý logic nhỏ nhất, đảm bảo tính đúng đắn ở mức code. (Ví dụ: hàm calculateTotal tính tổng tiền giỏ hàng trong Backend).
- **Kiểm thử tích hợp:** Tương ứng với giai đoạn Thiết kế kiến trúc (Architecture Design). Kiểm tra sự giao tiếp giữa Frontend (ReactJS) và Backend (Node.js/Express), hoặc giữa Backend và Database (MySQL).
- **Kiểm thử hệ thống:** Tương ứng với giai đoạn Yêu cầu hệ thống (System Design). Đây là trọng tâm của đồ án, kiểm tra toàn bộ chức năng của website trên giao diện hoàn chỉnh (End-to-End) để đảm bảo hệ thống vận hành đúng thiết kế.
- **Kiểm thử chấp nhận:** Tương ứng với giai đoạn Phân tích yêu cầu. Đảm bảo hệ thống đáp ứng đúng nhu cầu thực tế của người dùng cuối (User Needs) như đặt hàng thành công, quản lý kho chính xác.

4.3. Thiết kế theo Agile CI/CD

Hệ thống Cake Fantasy áp dụng quy trình CI/CD (Continuous Integration / Continuous Deployment) tự động hóa nhằm đảm bảo chất lượng phần mềm được

duy trì liên tục qua mỗi lần thay đổi mã nguồn, hỗ trợ đội ngũ phát triển nhận phản hồi nhanh chóng theo đúng tinh thần Agile.

4.3.1. Mục tiêu

- Tự động hóa: Giảm thiểu thao tác thủ công trong các bước Build, Test và Deploy, giúp tiết kiệm thời gian và giảm sai sót của con người.
- Phát hiện lỗi sớm (Shift-Left Testing): Các lỗi logic hoặc lỗi tích hợp được phát hiện ngay khi Developer đẩy code (Push) hoặc tạo Pull Request, thay vì đợi đến cuối chu kỳ phát triển.
- Quản lý rủi ro: Đảm bảo mã nguồn trên nhánh chính (main) luôn ở trạng thái ổn định (Deployable) nhờ các cổng kiểm soát chất lượng (Quality Gates).

4.3.2. Kiến trúc Pipeline

Quy trình được thực hiện thông qua GitHub Actions với các luồng công việc (Workflows) được định nghĩa trong thư mục .github/workflows/.

- Trigger:
  - *Push to Feature Branch:* Kích hoạt Pipeline cơ bản (Linting, Unit Tests) để kiểm tra nhanh.
  - *Open Pull Request :* Kích hoạt Full Pipeline (Lint, Unit, Integration, E2E Smoke Test) để đảm bảo tính toàn vẹn trước khi merge.
  - *Push to Main/Release:* Kích hoạt Release Pipeline (Build Docker Image, Deploy Staging/Production).

4.3.3. Các giai đoạn chi tiết

1. Install & Setup:

- Thiết lập môi trường Node.js (v20.x).
- Khôi phục Cache (node_modules) để tăng tốc độ build.
- Cài đặt dependencies cho cả Backend, Frontend và Admin (npm install).

2. Lint & Static Analysis:

- Sử dụng ESLint để kiểm tra cú pháp và quy chuẩn code cho toàn bộ Monorepo.
- *Chính sách:* Pipeline sẽ thất bại (Fail) ngay lập tức nếu phát hiện lỗi cú pháp nghiêm trọng.

3. Automated Testing:

- **Unit Tests**: Chạy Jest cho Backend để kiểm tra logic nghiệp vụ. Yêu cầu độ bao phủ mã (Coverage) cho các module quan trọng phải đạt ngưỡng quy định (ví dụ: >80%).
- **Integration Tests:**
  - Sử dụng Service Containers trong GitHub Actions để khởi tạo một MySQL Database tạm thời.
  - Chạy script seed.js để nạp dữ liệu mẫu sạch.
  - Thực thi test các luồng API tương tác với Database.
- **E2E Smoke Tests:** Chạy bộ test nhẹ bằng Cypress (Headless mode) để kiểm tra các luồng chính yếu (Login -> Xem sản phẩm -> Checkout) đảm bảo hệ thống không bị "chết" (Smoke test).

4. **Build & Artifacts:**

- Frontend/Admin: Chạy vite build để tạo các file tĩnh (HTML/CSS/JS) trong thư mục dist.
- Backend: Đóng gói ứng dụng thành Docker Image và đẩy lên Docker Hub hoặc Registry của Railway.

5. **Deployment:**

- Backend & Database: Tự động deploy lên Railway thông qua Webhook hoặc CLI khi Docker Image mới được publish.
- Frontend & Admin: Tự động deploy lên Vercel khi có commit mới vào nhánh main.

4.3.4. Chiến lược quản lý cấu hình và bảo mật

- Secrets Management: Tuyệt đối không lưu trữ thông tin nhạy cảm (API Keys, DB Passwords) trong mã nguồn (Repository). Thay vào đó, sử dụng GitHub Actions Secrets để inject các biến môi trường vào quá trình chạy CI/CD.
- Mocking Services: Để đảm bảo kết quả kiểm thử nhất quán (Deterministic) và không phụ thuộc vào dịch vụ bên thứ 3, hệ thống sử dụng kỹ thuật Mocking:
  - Sử dụng mockPaymentService.js thay thế cho cổng thanh toán MoMo thực trong môi trường Test/CI.
  - Mock Cloudinary cho chức năng upload ảnh.
- Database Isolation: Sử dụng container database riêng biệt cho mỗi lần chạy CI, đảm bảo dữ liệu test của lần chạy trước không ảnh hưởng đến lần chạy sau.

4.3.5. Kết luận

Thiết kế quy trình CI/CD này giúp nhóm phát triển Cake Fantasy duy trì nhịp độ phát hành nhanh (Agile) mà vẫn kiểm soát chặt chẽ chất lượng sản phẩm, giảm thiểu rủi ro khi triển khai lên môi trường Production.

4.4. Các phương pháp kiểm thử

Để đảm bảo chất lượng toàn diện cho hệ thống Cake Fantasy, dự án áp dụng chiến lược kiểm thử đa lớp, kết hợp ba phương pháp chính: Kiểm thử hộp trắng, Hộp xám và Hộp đen. Mỗi phương pháp tập trung vào một khía cạnh khác nhau của hệ thống nhằm tối đa hóa khả năng phát hiện lỗi.

4.4.1. Kiểm thử Hộp trắng

Khái niệm: Kiểm thử dựa trên cấu trúc mã nguồn bên trong.

Đối tượng thực hiện: Lập trình viên.

Áp dụng:

- Viết Unit Test cho các Controller và Service trong Node.js/Express (Backend).
- Kiểm tra các nhánh logic (If/Else) trong code xử lý tồn kho, tính toán giá và xác thực.

1. Chiến lược kiểm thử

Mục tiêu:

- Đảm bảo logic xử lý nghiệp vụ bên trong các function/module hoạt động chính xác tuyệt đối trước khi tích hợp.
- Đạt độ bao phủ mã nguồn tối đa cho các nhánh điều kiện và dòng lệnh.

Phương pháp kiểm thử:

- Unit Testing (Kiểm thử đơn vị): Sử dụng Jest để chạy test và mock các phụ thuộc như Database query, External Service.
- Mocking: Giả lập hành vi của MySQL, không kết nối DB thật trong unit tests.

Phạm vi kiểm thử:

- Các module Controller: *cartController.js, orderController.js, userController.js, itemController.js.*

- Các module Service: *momoPaymentService.js, mockPaymentService.js.*
- Các hàm Utility/Helper trong utils/.
- Không test: Các cấu hình framework mặc định, middleware đơn giản.

Kỹ thuật sinh Test-case:

- Basis Path Testing: Đảm bảo mọi đường dẫn logic trong hàm đều được đi qua ít nhất một lần.
- Control Structure Testing: Kiểm tra kỹ các vòng lặp, điều kiện rẽ nhánh.

2. Phương pháp thiết kế Test Case

- Đầu vào: Các object JavaScript (request body, params) được khởi tạo trong code test.
- Môi trường: Cô lập hoàn toàn. Các Dependency (như db.query, User.findByEmail) được mock.
- Kết quả mong đợi (Assertion):
  - Giá trị trả về của hàm đúng với tính toán (sử dụng *expect().toBe()).*
  - Các hàm phụ thuộc được gọi đúng số lần (sử dụng *jest.fn()* và *toHaveBeenCalledTimes()).*
  - Ngoại lệ (Exception) được ném ra đúng loại khi gặp lỗi (sử dụng *expect().toThrow()).*

3. Kiểm thử chức năng: Giỏ hàng

3.1. Mô tả chức năng và nghiệp vụ

File cần test: *cartController.js*

Function cần test: *addToCart(req, res)*

Logic cần bao phủ:

1. Validate input: kiểm tra *item_id* và *quantity* có tồn tại không.
2. Tìm Item theo *item_id*. Nếu không thấy → trả về 404 "Item not found".
3. Validate quantity là số dương.
4. Check tồn kho: if (quantity > stock_quantity) → trả về lỗi "Vượt quá số lượng có sẵn".
5. Check item trong giỏ:
   - Nhánh True (Đã có): existingItem.quantity += quantity.
   - Nhánh False (Chưa có): Tạo mới cart item.

6. Gọi *db.query()* để lưu vào database.

3.2. Áp dụng phương pháp sinh Test Case

ID	Test Case	Setup	Hành động	Kết quả mong đợi
UT-C-01	Thêm sản phẩm mới thành công	- db.query trả về Item (stock=10) - Cart item chưa tồn tại	Gọi *addToCart({item_id:1, quantity:2})*	- db.query INSERT được gọi 1 lần - Response status 200 - Không có Exception
UT-C-02	Cộng dồn số lượng khi sản phẩm đã có	- db.query trả về Item (stock=10) - Cart item đã tồn tại (qty=3)	Gọi *addToCart({item_id:1, quantity:2})*	- Item cũ có quantity mới = 5 - db.query UPDATE được gọi 1 lần
UT-C-03	Lỗi vượt quá tồn kho (Exception Path)	- db.query trả về Item (stock=5)	Gọi *addToCart({item_id:1, quantity:6})*	- Response status 400 - Message "Vượt quá số lượng" - db.query INSERT KHÔNG được gọi
UT-C-04	Lỗi sản phẩm không tồn tại	- db.query trả về mảng rỗng []	Gọi *addToCart({item_id:999, quantity:1})*	- Response status 404 - Message "Item not found"

4. Kiểm thử chức năng: Đơn hàng và thanh toán

4.1. Mô tả chức năng và nghiệp vụ

File cần test: orderController.js

Function cần test: *placeOrder(req, res)*

Logic cần bao phủ:

Validate các trường bắt buộc: **firstName**, **lastName**, **contactNumber1**, **address**. Nếu thiếu → trả về 400.

Validate items array: nếu rỗng → trả về 400 "Đơn hàng không có sản phẩm nào".

Tính tổng tiền: Loop qua items, sum += price * quantity.

Gọi **Order.create()** để lưu đơn hàng vào DB.

Gọi **Order.clearCart()** để xóa giỏ hàng.

Xử lý thanh toán:

Nhánh COD: Trả về success với orderId.

Nhánh MoMo: Gọi **momoPaymentService.createPaymentUrl()** → trả về URL thanh toán.

4.2. Áp dụng phương pháp sinh Test Case

ID	Tên Test Case	Điều kiện đầu vào	Luồng xử lý	Kết quả mong đợi
WB-O-01	Tạo đơn hàng thành công (Happy Path)	Giỏ hàng hợp lệ, Tồn kho đủ, paymentMethod='COD'	Thực thi hết các bước 1→6 không có lỗi	- Order.create() được gọi - Order.clearCart() được gọi - Response status 200 với orderId
WB-O-02	Kiểm thử lỗi khi thiếu thông tin (Fail Path)	Thiếu trường **contactNumber1**	Bước 1 (Validate) gặp lỗi	- Response status 400 - Message "Thiếu thông tin giao hàng bắt buộc" - Order.create() KHÔNG được gọi
WB-O-03	Kiểm thử giỏ hàng rỗng	items = []	Bước 2 (Validate items) gặp lỗi	- Response status 400 - Message "Đơn hàng không có sản phẩm nào"
WB-O-04	Xử lý thanh toán MoMo (Integration)	paymentMethod='momo'	Service gọi API tạo URL thanh toán → Trả về URL	- momoPaymentService.createPaymentUrl() được gọi - Response

ID	Tên Test Case	Điều kiện đầu vào	Luồng xử lý	Kết quả mong đợi
				chứa **url** thanh toán hợp lệ

4.3. Kiểm thử quy trình

Kiểm tra luồng dữ liệu đi qua các tầng kiến trúc:

Controller Layer:

- Gửi Request JSON thiếu trường bắt buộc (ví dụ: thiếu **address**).
- Mong đợi: Validation chặn lại, trả về HTTP 400 + Message lỗi chi tiết.

Service Layer:

- Mock Database để giả lập trường hợp DB bị timeout.
- Mong đợi: Service bắt lỗi và trả về message "Lỗi hệ thống, vui lòng thử lại sau".

Database Layer:

- Thử insert một đơn hàng với user_id không tồn tại.
- Mong đợi: MySQL báo lỗi Foreign Key Constraint Violation.

4.4.2. Kiểm thử Hộp đen

1. Chiến lược kiểm thử

Mục tiêu: Để đảm bảo chất lượng phần mềm cho dự án Website Tiệm Bánh Cake Fantasy, nhóm thực hiện chiến lược kiểm thử toàn diện, tập trung vào việc xác minh các yêu cầu nghiệp vụ và trải nghiệm người dùng.

Phương pháp kiểm thử: Sử dụng Kiểm thử Hộp đen. Nhóm kiểm thử tập trung vào đầu vào và đầu ra của các chức năng dựa trên đặc tả yêu cầu mà không can thiệp vào mã nguồn nội bộ.

Phạm vi kiểm thử:

- Kiểm thử chức năng: Đảm bảo các tính năng hoạt động đúng nghiệp vụ (Giỏ hàng, Đặt hàng, Thanh toán).
- Kiểm thử giao diện (UI/UX): Đảm bảo tính thẩm mỹ và dễ sử dụng.

Kỹ thuật sinh Test-case:

- Phân vùng tương đương
- Phân tích giá trị biên
- Bảng quyết định
- Kiểm thử dựa trên quy trình

2. Phương pháp thiết kế Test Case

Để sinh ra các trường hợp kiểm thử hiệu quả, nhóm áp dụng các kỹ thuật thiết kế sau:

- Phân vùng tương đương Chia dữ liệu đầu vào thành các lớp tương đương (hợp lệ và không hợp lệ) để giảm số lượng test case cần thiết.
- Phân tích giá trị biên: Tập trung kiểm thử tại các biên của vùng dữ liệu (giá trị nhỏ nhất, lớn nhất) vì đây là nơi dễ phát sinh lỗi.
- Bảng quyết định Sử dụng cho các chức năng có logic phức tạp với nhiều điều kiện kết hợp (ví dụ: tính phí ship).
- Kiểm thử dựa trên quy trình: Thiết kế kịch bản dựa trên luồng nghiệp vụ thực tế của người dùng.

3. Kiểm thử chức năng: Đăng nhập

3.1. Mô tả chức năng và nghiệp vụ

Mô tả giao diện:

- Người dùng nhập Email và Mật khẩu.
- Hệ thống kiểm tra tính hợp lệ của dữ liệu.
- Hệ thống xác thực thông tin với Cơ sở dữ liệu.
- Nếu đúng: Cấp Token (JWT), lưu session và chuyển hướng về Trang chủ.
- Nếu sai: Hiển thị thông báo lỗi cụ thể.

Luồng hoạt động:

- Người dùng nhập Email và Mật khẩu.
- Hệ thống kiểm tra định dạng dữ liệu (Validation).
- Hệ thống đối chiếu thông tin với Cơ sở dữ liệu (MySQL).
- Nếu hợp lệ: Hệ thống cấp Token JWT, chuyển hướng về Trang chủ và hiển thị tên người dùng.
- Nếu không hợp lệ: Hệ thống hiển thị thông báo lỗi tương ứng ("Tài khoản không tồn tại", "Tài khoản hoặc mật khẩu không đúng").

3.2. Áp dụng phương pháp sinh Test Case

Nhóm sử dụng kỹ thuật Phân vùng tương đương (Equivalence Partitioning) để chia dữ liệu đầu vào thành các lớp hợp lệ và không hợp lệ:

Trường dữ liệu	Lớp hợp lệ	Lớp không hợp lệ
Email	Email đúng định dạng và đã được đăng ký (VD: user@cakefantasy.com)	- Email sai định dạng (thiếu @, thiếu domain) - Email chưa được đăng ký - Để trống
Mật khẩu	Mật khẩu chính xác khớp với email (tối thiểu 8 ký tự)	- Mật khẩu sai - Để trống

Bảng Test Case: Chức năng đăng nhập

ID	Tên kịch bản	Các bước thực hiện	Dữ liệu kiểm thử	Kết quả mong đợi
TC-LOGIN-01	Đăng nhập thành công	1. Truy cập trang Đăng nhập 2. Nhập Email và Mật khẩu đúng 3. Nhấn nút "Đăng nhập"	Email: **user@cakefantasy.com**, pass: 12345678	1. Chuyển hướng về Trang chủ 2. Header hiển thị tên người dùng
TC-LOGIN-02	Đăng nhập sai mật khẩu	1. Nhập Email đã đăng ký 2. Nhập Mật khẩu sai 3. Nhấn nút "Đăng nhập"	Email: **user@cakefantasy.com**, pass: wrongpass	1. Hiển thị lỗi "Tài khoản hoặc mật khẩu không đúng" 2. Người dùng vẫn ở trang đăng nhập

ID	Tên kịch bản	Các bước thực hiện	Dữ liệu kiểm thử	Kết quả mong đợi
TC-LOGIN-03	Đăng nhập với Email chưa tồn tại	1. Nhập Email chưa đăng ký 2. Nhập mật khẩu bất kỳ 3. Nhấn "Đăng nhập"	Email: **notexist@test.com**, pass: 123123	Hiển thị lỗi "Tài khoản không tồn tại"
TC-LOGIN-04	Kiểm tra Validation (Bỏ trống)	1. Để trống ô Email hoặc Mật khẩu 2. Nhấn "Đăng nhập"	Email: [Rỗng], pass: [Rỗng]	Hiển thị thông báo validation dưới ô input

4. Kiểm thử chức năng: Giỏ hàng

4.1. Mô tả chức năng và nghiệp vụ

Mô tả giao diện:

- Giao diện trang giỏ hàng hiển thị danh sách các sản phẩm đã thêm.
- Mỗi sản phẩm hiển thị hình ảnh, tên, đơn giá, số lượng và thành tiền.
- Người dùng có thể tăng/giảm số lượng (có giới hạn stock_quantity) hoặc xóa sản phẩm.
- Khu vực tóm tắt hiển thị Tạm tính, Phí vận chuyển và Tổng cộng.
- Nút "Thanh toán" điều hướng đến trang Checkout.

Sơ đồ hoạt động (Nghiệp vụ):

1. Người dùng xem sản phẩm (trang Product Detail).
2. Người dùng chọn số lượng và nhấn "Thêm vào giỏ".
3. Hệ thống gọi API (/api/cart/add) để thêm sản phẩm. Nếu đã tồn tại, tăng số lượng.
4. Người dùng truy cập trang Giỏ hàng.
5. Người dùng thay đổi số lượng hoặc xóa sản phẩm.
6. Hệ thống tự động cập nhật lại Tạm tính và Tổng cộng.
7. Không cho phép nhập số lượng vượt quá tồn kho.
8. Người dùng nhấn "Thanh toán".

Ràng buộc:

- Số lượng (quantity) phải > 0.
- Số lượng không được vượt quá tồn kho (stock_quantity).
- Phí vận chuyển là 15.000đ.

4.2. Áp dụng phương pháp sinh Test Case

Áp dụng phân vùng tương đương và giá trị biên:

Kỹ Thuật	Điều kiện kiểm thử	Giá trị mẫu	Kết quả mong đợi
Hợp lệ	1 <= Số lượng <= Tồn kho	1	Cập nhật thành tiền, tổng cộng chính xác
Biên	Giá trị nhỏ nhất	1	Hiển thị bình thường. Nút giảm (-) bị vô hiệu hóa
Biên	Giá trị biên lớn nhất (Giả sử tồn = 10)	10	Không cho tăng thêm
Không hợp lệ	Số lượng > Tồn kho (Giả sử tồn = 10)	11	Báo lỗi "Vượt quá số lượng có sẵn". Tự động về mức max
Không hợp lệ	Số lượng <= 0	0, -1	Không cho phép nhập
Không hợp lệ	Ký tự không phải số	"abc", "#$%"	Không cho nhập

Bảng Test Case: Chức năng giỏ hàng

ID	Tên Test Case	Các bước thực hiện	Dữ liệu mẫu	Kết quả mong đợi
TC-CART-01	Thêm sản phẩm vào giỏ	1. Tại trang chi tiết, nhấn	Bánh Tiramisu (Giá: 150k, SL: 2)	Sản phẩm xuất hiện trong giỏ. Thành tiền = 300k

ID	Tên Test Case	Các bước thực hiện	Dữ liệu mẫu	Kết quả mong đợi
		"Thêm vào giỏ"		
TC-CART-02	Cập nhật tăng số lượng sản phẩm	1. Tại giỏ hàng, nhấn nút (+) của sản phẩm	Sản phẩm A (Giá: 100k, SL hiện tại: 1)	Số lượng tăng lên 2. Thành tiền = 200k. Tổng cập nhật đúng
TC-CART-03	Xóa sản phẩm khỏi giỏ	1. Nhấn icon thùng rác tại sản phẩm	Sản phẩm A đang có trong giỏ	Sản phẩm biến mất khỏi danh sách. Tổng tiền cập nhật lại
TC-CART-04	Kiểm tra phí vận chuyển	1. Thêm sản phẩm giá 300k 2. Thêm tiếp sản phẩm để tổng >= 500k	Sản phẩm A (300k), Sản phẩm B (200k)	Lúc đầu: Phí ship 30k. Sau khi thêm: Phí ship miễn phí

5. Kiểm thử chức năng: Đơn hàng và thanh toán

5.1. Mô tả chức năng và nghiệp vụ

Mô tả giao diện:

- Quy trình bắt đầu từ trang Giỏ hàng, chuyển sang trang Thanh toán (Checkout) và kết thúc tại trang Thông báo.
- Trang Thanh toán: Bao gồm form nhập thông tin giao hàng (Họ tên, SĐT, Địa chỉ...), khu vực chọn phương thức thanh toán (COD hoặc MoMo), và bảng tóm tắt đơn hàng.

Sơ đồ hoạt động:

1. Người mua nhấn nút "Thanh toán" từ giao diện Giỏ hàng.
2. Hệ thống kiểm tra giỏ hàng có sản phẩm không.

3. Khởi tạo đơn hàng.
4. Tính toán chi phí: Hệ thống tính tổng chi phí đơn hàng và phí vận chuyển.
5. Xử lý thanh toán: Hệ thống gửi yêu cầu đến cổng thanh toán (đối với MoMo) hoặc ghi nhận phương thức COD.
6. Theo dõi đơn hàng.

5.2. Áp dụng phương pháp sinh Test Case

Áp dụng Kiểm thử dựa trên trường hợp sử dụng (Use Case Testing):

Loại luồng	Tên kịch bản	Mô tả luồng đi	Kết quả mong đợi
Luồng chính	Đặt hàng thành công (COD)	Người dùng nhập đủ thông tin → Chọn COD → Nhấn "Đặt hàng"	1. Tạo đơn hàng mới (status='pending') 2. Chuyển cart_items thành order items 3. Xóa sạch giỏ hàng và trừ tồn kho
Luồng thay thế	Đặt hàng thành công (MoMo)	Người dùng nhập đủ thông tin → Chọn MoMo → Nhấn "Đặt hàng"	1. Tạo đơn hàng trong DB 2. Chuyển hướng sang cổng thanh toán MoMo
Luồng ngoại lệ	Xử lý khi thiếu thông tin	Người dùng bỏ trống các trường bắt buộc và nhấn "Đặt hàng"	Hiển thị lỗi validation, không tạo đơn hàng

Bảng Test Case: Chức năng đơn hàng

ID	Tên Test Case	Các bước thực hiện	Kết quả mong đợi
TC-ORDER-01	Tạo đơn hàng thành công (COD)	1. Checkout 2. Điền đầy đủ thông tin 3. Chọn COD 4. Nhấn "Đặt hàng"	1. Tạo order mới (status 'pending') 2. Chuyển cart items thành order items 3. Giỏ hàng được xóa sạch 4. Tồn kho bị trừ đi

ID	Tên Test Case	Các bước thực hiện	Kết quả mong đợi
TC-ORDER-02	Tạo đơn hàng thành công (MoMo)	1. Checkout 2. Điền đầy đủ thông tin 3. Chọn MoMo 4. Nhấn "Đặt hàng"	1. Tạo order trong database 2. Chuyển hướng sang cổng thanh toán MoMo
TC-ORDER-03	Tạo đơn hàng thiếu thông tin	1. Tới trang checkout 2. Bỏ trống Số điện thoại 3. Nhấn "Đặt hàng"	1. Hiển thị lỗi "Thiếu thông tin giao hàng bắt buộc" 2. Không tạo order mới
TC-ORDER-04	Tạo đơn hàng khi sản phẩm hết hàng (Race Condition)	1. User A có sản phẩm X (tồn kho = 1) trong giỏ 2. User B mua hết sản phẩm X 3. User A nhấn "Đặt hàng"	1. Hiển thị thông báo lỗi "Sản phẩm đã hết hàng" 2. Yêu cầu cập nhật lại giỏ hàng 3. Không tạo đơn hàng

Mô tả quy trình: Thanh toán đơn hàng

Quy trình này bắt đầu từ trang Giỏ hàng, đi qua trang Thanh toán và kết thúc ở trang Thông báo kết quả.

Bước 1: Giỏ hàng (CartPage) - Người dùng kiểm tra sản phẩm, số lượng và nhấn "Thanh toán".

Bước 2: Thanh toán (CheckoutPage) - Người dùng điền thông tin giao hàng (Tên, SĐT, Địa chỉ). Chọn phương thức thanh toán (COD hoặc MoMo).

Bước 3: Xử lý (orderController.placeOrder) - Nhấn "Đặt hàng". Hệ thống tạo đơn hàng (Order.create), giảm tồn kho và xóa giỏ hàng (Order.clearCart).

Bước 4A (COD): Chuyển hướng đến trang "Đơn hàng của tôi".

Bước 4B (MoMo): Chuyển hướng đến cổng thanh toán MoMo. Sau khi thanh toán, callback về hệ thống để xác thực.

CHƯƠNG 5: KẾT QUẢ KIỂM THỬ

5.1. Môi trường kiểm thử

- Repository: WebsiteTiemBanh
- Thư mục thực thi: Backend/
- Lệnh chạy test: npm test (Sử dụng Jest, file cấu hình: jest.config.cjs)
- Phiên bản Node/npm: Theo cấu hình trong Backend/package.json (sử dụng jest v30).
- Cơ sở dữ liệu (Database):
  - Test chạy chủ yếu bằng Mocks và thiết lập local (tests/setup.js).
  - Schema tham khảo: sql/cake_fantasy_db.sql (dùng khi cần seed data).
- Biến môi trường (ENV): Các biến quan trọng như DB connection, CLOUDINARY_URL, MOMO credentials được lưu ngoài file (tham khảo env.example).

5.2. Thống kê số lượng Test Case

Dựa trên kịch bản kiểm thử đã thiết kế, tổng số lượng Test Case được thực thi cho toàn bộ dự án là **64 Test Case** employee E2E (Cypress). Số liệu thống kê chi tiết như sau:

5.3. Báo cáo kết quả kiểm thử

Bảng kết quả **unit test:**

STT	Module	Chức năng	Pass	Fail	Skip	Total
1	SEARCH	Tìm kiếm & filter sản phẩm	10	1	0	11
2	CART	Quản lý giỏ hàng & tính tổng	13	1	0	14
3	ORDER	Tạo, cập nhật, hủy & lấy đơn hàng	17	1	0	18
4	AUTH	Phân quyền user/admin/employee & auth middleware	19	2	0	21

Bảng kết quả **E2E Cypress**:

STT	Module	Chức năng	Pass	Fail	Skip	Total

1	User Management	Đăng ký, Đăng nhập, Validate form, Logout, Token storage.	10	0	0	10
2	Product Browsing	Xem list, Filter category, Search keyword, Sort giá, Pagination.	11	0	0	11
3	Cart & Checkout	Thêm/Xóa giỏ hàng, Tăng giảm SL, Checkout flow, Thanh toán COD.	6	0	0	6

Tài liệu tham khảo

[1]. Viblo, “Kiểm thử phần mềm là gì? Quy trình kiểm thử phần mềm,” truy cập: 15/12/2025. [Online]. Available: https://viblo.asia/p/kiem-thu-phan-mem-la-gi-quy-trinh-kiem-thu-phan-mem-1VgZvayYKAw

[2]. GeeksforGeeks, “Software Engineering | SDLC V-Model,” truy cập: 15/12/2025. [Online]. Available: https://www.geeksforgeeks.org/software-engineering/software-engineering-sdlc-v-model/

[3]. GeeksforGeeks, “Introduction to Cypress Testing Framework,” truy cập: 15/12/2025. [Online]. Available: https://www.geeksforgeeks.org/software-testing/introduction-to-cypress-testing-framework/

[4]. Ragab, “Unit Tests vs Integration vs End-to-End,” LinkedIn Pulse, truy cập: 15/12/2025. [Online]. Available: https://www.linkedin.com/pulse/unit-tests-vs-integration-end-to-end-ashraf-ragab

[5]. IBM, “What is End-to-End Testing?,” truy cập: 15/12/2025. [Online]. Available: https://www.ibm.com/think/topics/end-to-end-testing

[6]. IBM, “What is System Testing?,” truy cập: 15/12/2025. [Online]. Available: https://www.ibm.com/think/topics/system-testing