ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

A blue button with white text

AI-generated content may be incorrect.

**BÁO CÁO ĐỒ ÁN KIỂM THỬ PHẦN MỀM**

**KIỂM THỬ WEBSITE THƯƠNG MẠI ĐIỆN TỬ SHOESHOP**

Thành viên:

Đỗ Phú Thành - 3122411189

Huỳnh Duy Khang – 3122411088

Võ Thành Danh – 3122411024

Huỳnh Minh Quân – 3122411167

Giảng viên hướng dẫn: TS. Đỗ Như Tài

**TP. HỒ CHÍ MINH, 2025 - 2026**

# **MỤC LỤC**

[MỤC LỤC 1](#_heading=h.ou4zr2sipdjj)

[CHƯƠNG 1 – INTRODUCTION 4](#_heading=h.tjvcelqmdsax)

[Tổng quan chương 4](#_heading=h.xytuy81yno8o)

[1.1. Tổng quan đề tài 4](#_heading=h.xiyss4502tz)

[1.1.1. Bối cảnh và lý do chọn đề tài 4](#_heading=h.4z86vh9x4shv)

[1.1.2. Mục tiêu của hệ thống Shoeshop 5](#_heading=h.4vqe97ycuui)

[1.1.3. Phạm vi dự án và phạm vi kiểm thử 5](#_heading=h.1umulbglyc5l)

[1.1.4. Quy trình phát triển phần mềm (Agile Scrum) 6](#_heading=h.rdglvxayn4c4)

[1.1.5. Tổng quan CI/CD và vai trò trong kiểm thử 6](#_heading=h.dgiu94en8g5w)

[1.1.6. Tổng quan kiểm thử phần mềm 6](#_heading=h.r06uhjjhfls)

[1.2. Phân tích yêu cầu hệ thống 6](#_heading=h.glpsei95yszz)

[1.2.1. Khảo sát hiện trạng và pain point nghiệp vụ 6](#_heading=h.by0edwbup9wu)

[1.2.2. Phạm vi chức năng hệ thống (Module khách hàng – Module admin) 7](#_heading=h.393gy79tkvuy)

[1.2.3. Công nghệ sử dụng (ReactJS – Spring Boot – MySQL – Vercel/Railway) 7](#_heading=h.wnqasgdpd9fs)

[1.3 Quy trình nghiệp vụ 8](#_heading=h.dzghktc5pas4)

[1.4. Yêu cầu nghiệp vụ 10](#_heading=h.fcaeyyjpk198)

[1.4.2. Điều kiện chấp nhận (UAC Test) 11](#_heading=h.lef86vqedel5)

[1.4.3. Yêu cầu chức năng (Functional Requirements) 11](#_heading=h.yfgioic82dhd)

[1.4.4. Yêu cầu phi chức năng (NFR: hiệu năng, bảo mật, khả dụng) 12](#_heading=h.oxjr05nbezyc)

[1.5. Kế hoạch triển khai 14](#_heading=h.vxstyliq16va)

[1.5.1. Kế hoạch phát triển (Development Plan) 14](#_heading=h.y4to6xcuunfr)

[1.5.2. Kế hoạch kiểm thử tổng quan (Overview Test Plan) 14](#_heading=h.bl2b0pqhbmjt)

[CHƯƠNG 2 – PHÂN TÍCH & THIẾT KẾ HỆ THỐNG 17](#_heading=h.48a99pl0h5g0)

[TỔNG QUAN CHƯƠNG 2 17](#_heading=h.sbtqoa5hdhfo)

[2.1. Thiết kế hệ thống (System Design – Business View) 17](#_heading=h.2bzt8ky3lnao)

[2.1.1. Thiết kế chức năng 17](#_heading=h.coh8369fkr5m)

[2.1.2. Thiết kế dữ liệu 24](#_heading=h.1zpdi1ojldtn)

[2.1.3. Thiết kế giao diện 26](#_heading=h.7a0t587ht8ad)

[2.2. Thiết kế kiến trúc (Architecture Design – Technical View) 31](#_heading=h.uft7fulgsewj)

[2.2.1. Tổng quan kiến trúc 31](#_heading=h.1dz58i7lrs5d)

[2.2.2. Phân tích thành phần 32](#_heading=h.q11uepx83xyb)

[2.2.3. Phân tích giao tiếp 43](#_heading=h.ln2qn72e2cqp)

[CHƯƠNG 3 – TEST PLAN 44](#_heading=h.u54oycvyzai3)

[3.1. Tổng quan kế hoạch kiểm thử 44](#_heading=h.56q9mxn8f4rf)

[3.1.1. Test Scope (In-scope/Out-of-scope) 44](#_heading=h.hke6v3ffmfoe)

[3.2. Phương pháp kiểm thử 45](#_heading=h.3hvo8bmcbahu)

[3.2.1 Kiểm thử tĩnh (Static Testing) 45](#_heading=h.ia2ldq14wk1m)

[3.2.2 Kiểm thử động (Dynamic Testing) 45](#_heading=h.1xj980hejbei)

[3.3. Hạng mục kiểm thử 46](#_heading=h.ggbdad2lpmzz)

[3.4 Hạng mục không được kiểm thử 56](#_heading=h.sj18sbw6fsrk)

[3.3.1. Các chức năng thuộc hệ thống bên thứ ba 56](#_heading=h.aszktqg7rhu3)

[3.3.2. Kiểm thử bảo mật nâng cao 56](#_heading=h.8xtlnmh40fyr)

[3.5. Chiến lược kiểm thử (Test Strategy) 57](#_heading=h.woasaq98g2p8)

[3.5.1. Các loại kiểm thử (Test Types) 57](#_heading=h.asir6336x2fm)

[3.5.2. Ma trận phân bổ kiểm thử (Test Stage Matrix) 60](#_heading=h.aqvk66dktage)

[3.6. Test Environment 61](#_heading=h.v7wnr0e5n7ba)

[3.6.1 Hardware 61](#_heading=h.27gg296730du)

[3.6.2 Software 61](#_heading=h.xvimepupg4rl)

[3.6.3. Nhân lực (Human Resources) 62](#_heading=h.soqjoms74qdi)

[CHƯƠNG 4 – THIẾT KẾ KIỂM THỬ (TEST DESIGN) 63](#_heading=h.vkv8n1jrlgzl)

[4.1. Giới thiệu 63](#_heading=h.11iuw5cb5v4h)

[4.2. Quy trình thiết kế kiểm thử theo V-Model 64](#_heading=h.bqhbv43rjd6r)

[4.2.1. Phân tích yêu cầu - 1a 64](#_heading=h.c5wcqizg35if)

[4.2.2. Thiết kế hệ thống - 2a 64](#_heading=h.n6yjzeh38qe)

[4.2.3. Thiết kế kiến trúc - 3a 65](#_heading=h.c7sn1i1hskci)

[4.2.4. Thiết kế module - 4a 65](#_heading=h.9if5637toguy)

[4.2.5. Kiểm thử đơn vị - 1b 65](#_heading=h.uy7exu5e7p9i)

[4.2.6. Kiểm thử tích hợp - 2b 66](#_heading=h.3f01gccwe7co)

[4.2.7. Kiểm thử hệ thống - 3b 67](#_heading=h.5o3vwe4g1uwq)

[4.2.8. Kiểm thử chấp nhận - 4b 67](#_heading=h.6duw2s6nk3pg)

[4.3. Kỹ thuật thiết kế kiểm thử 68](#_heading=h.8lx58cd5815q)

[4.3.1. Kĩ thuật test: Hộp trắng (White Box) 68](#_heading=h.orp0fahhq583)

[4.3.2. Kĩ thuật test: Hộp đen (Black Box) 74](#_heading=h.y6xk2im6l3at)

[4.4. Phương pháp thiết kế kiểm thử 77](#_heading=h.md0qr4hf6yvm)

[4.4.1. Kiểm thử thủ công 77](#_heading=h.4bil8okcc0r8)

[4.4.2. Kiểm thử tự động 78](#_heading=h.x2xm85cmgwtq)

[4.5. Áp dụng GenAI trong việc sinh Test Case 79](#_heading=h.m2q4o935l5vu)

[4.5.1. Đầu vào GenAi 79](#_heading=h.ckhrrjubnrcj)

[4.5.2. Giai đoạn xử lý của GenAI 81](#_heading=h.xhhll4r5gifz)

[4.5.3. Hoàn thiện test case 82](#_heading=h.z388f6impxvd)

[4.5.4. Phương pháp Prompt Engineering áp dụng trong sinh Test Case 82](#_heading=h.wxklcpiturv0)

[4.5.5. Prompt mẫu sinh Test Case với GenAI 83](#_heading=h.6zww93m21wuz)

[CHƯƠNG 5. BÁO CÁO KIỂM THỬ 85](#_heading=h.moyxjmmahvpu)

[5.1. Tổng quan về quá trình kiểm thử 85](#_heading=h.u58mj3spc1a8)

[5.2. Báo cáo trường hợp kiểm thử 85](#_heading=h.rehki5qth2m)

[5.2.1. Giới thiệu 85](#_heading=h.nkw2xr3f7fgs)

[5.2.2. Phạm vi bao phủ 86](#_heading=h.69sdvb26dzus)

[5.2.3. Kết quả thực thi 86](#_heading=h.1ydtbdx2vwnn)

[5.3. Báo cáo lỗi 86](#_heading=h.clkbvymvbty6)

[5.3.1. Thống kê lỗi 87](#_heading=h.1p47dzte5674)

# 

# 

# CHƯƠNG 1 – GIỚI THIỆU

## Tổng quan chương

Chương 1 cung cấp bức tranh toàn cảnh về hệ thống Shoeshop dưới góc nhìn KIỂM THỬ, nhằm tạo nền tảng để xây dựng kế hoạch kiểm thử (Test Plan) và phương pháp luận kiểm thử (Test Methodology) ở các chương tiếp theo.

Mục tiêu của chương này không phải mô tả chi tiết hệ thống, mà là xác định đúng và đủ các thông tin cần thiết phục vụ cho hoạt động kiểm thử, bao gồm:

* Mục tiêu và phạm vi dự án → để xác định *phạm vi kiểm thử (test scope)*.
* Quy trình nghiệp vụ → để hiểu *luồng xử lý* và xác định *điểm kiểm thử quan trọng*.
* Yêu cầu nghiệp vụ (User Story, Acceptance Criteria) → để làm *cơ sở xây dựng Test Scenario và Test Case*.
* Yêu cầu chức năng và phi chức năng → để xác định *loại kiểm thử* cần thực hiện (functional, UI, API, performance, security…).
* Quy trình phát triển (Agile, CI/CD) → để xác định *cách kiểm thử được lồng ghép theo sprint* và *trong pipeline CI/CD*.

## 1.1. Tổng quan đề tài

### 1.1.1. Bối cảnh và lý do chọn đề tài

Trong bối cảnh chuyển đổi số đang diễn ra mạnh mẽ, thương mại điện tử đã trở thành một phần không thể thiếu trong hoạt động kinh doanh hiện đại, đặc biệt là ở lĩnh vực thời trang và giày dép. Người tiêu dùng ngày nay có xu hướng mua sắm trực tuyến nhiều hơn, nhờ sự tiện lợi trong việc tìm kiếm, so sánh giá, đặt hàng và thanh toán ngay tại nhà. Tuy nhiên, khảo sát thực tế cho thấy nhiều hệ thống bán hàng trực tuyến hiện nay vẫn tồn tại các hạn chế về chất lượng phần mềm, dẫn đến trải nghiệm người dùng chưa tối ưu và rủi ro trong vận hành. Vậy nên để đảm bảo chất lượng phần mềm trước khi được đưa vào vận hành và tạo sự yên tâm cho doanh nghiệp trước khi sử dụng, nhóm đề tài lựa chọn dự án **kiểm thử hệ thống Website bán giày trực tuyến – ShoeShop Store** nhằm đánh giá, kiểm tra và đảm bảo chất lượng phần mềm của một nền tảng thương mại điện tử mô phỏng thực tế. Đề tài tập trung vào việc xây dựng kế hoạch kiểm thử, thiết kế ca kiểm thử, thực thi và phân tích lỗi, nhằm xác định mức độ ổn định, tính chính xác và khả năng vận hành của hệ thống trước khi đưa vào sử dụng.

### 1.1.2. Mục tiêu của hệ thống Shoeshop

Hệ thống ShoeShop Store được xây dựng với mục tiêu trở thành một nền tảng thương mại điện tử chuyên biệt cho ngành hàng giày dép, giải quyết các vấn đề như:

* **Đối với khách hàng**: Cung cấp trải nghiệm mua sắm trực tuyến mượt mà, tiện lợi và an toàn, từ khâu tìm kiếm, lựa chọn sản phẩm, quản lý giỏ hàng đến thanh toán (COD, VNPay) và theo dõi đơn hàng
* **Đối với Vận hành (Staff, Manager):** Tối ưu hóa quy trình nghiệp vụ nội bộ, bao gồm quản lý đơn hàng theo thời gian thực, kiểm soát tồn kho chính xác, và quản lý linh hoạt các chương trình khuyến mãi.
* **Đối với Quản trị (Admin):** Đảm bảo an ninh, an toàn dữ liệu thông qua cơ chế phân quyền chi tiết cho bốn vai trò (Customer, Staff, Manager, Admin), và cung cấp công cụ giám sát hoạt động tổng thể.

Mục tiêu của dự án kiểm thử là xác minh rằng hệ thống Shoeshop đã đạt được các mục tiêu trên.

### 1.1.3. Phạm vi dự án và phạm vi kiểm thử

**Phạm vi dự án (Project Scope):** Phạm vi dự án tập trung vào hệ thống website bán giày ShoeShop Store, bao gồm các chức năng được phát triển cho bốn vai trò: Customer, Staff, Manager và Admin mà tập trung chủ yếu vào 4 chức năng cốt lõi: tìm kiếm sản phẩm, giỏ hàng và thanh toán, đơn hàng, bảo mật và phân quyền . Dự án không bao gồm các hệ thống của bên thứ ba, các hoạt động marketing, hay quy trình logistics vật lý bên ngoài.

**Phạm vi kiểm thử (Testing Scope):** Phạm vi kiểm thử được xác định bám sát phạm vi dự án, tập trung vào việc xác minh và đánh giá chất lượng của các chức năng cốt lõi.

* **Đối với Customer:** Kiểm thử quy trình người dùng cuối (End-to-End) bao gồm: đăng ký, đăng nhập, tìm kiếm sản phẩm, quản lý giỏ hàng, áp dụng mã giảm giá, thanh toán (COD & VNPay), theo dõi và hủy đơn hàng.
* **Đối với Staff:** Kiểm thử quy trình nghiệp vụ xử lý đơn hàng: xác nhận đơn, cập nhật trạng thái (từ CREATED đến DELIVERED), và kiểm tra tính chính xác của tồn kho khi xử lý
* **Đối với Admin:** Kiểm thử các chức năng quản trị hệ thống: phân quyền, quản lý tài khoản người dùng, và bảo mật truy cập.
* **Đối với Manager:** Kiểm thử các chức năng quản lý nghiệp vụ: tạo và quản lý mã giảm giá, giám sát và can thiệp đơn hàng, xem báo cáo doanh thu

Hoạt động kiểm thử bao gồm cả kiểm thử chức năng (Functional Testing) và kiểm thử phi chức năng (Non-functional Testing)

### 1.1.4. Quy trình phát triển phần mềm (Agile Scrum)

Dự án được phát triển theo mô hình Agile, cụ thể là phương pháp luận Scrum. Dự án được chia thành các chu kỳ phát triển ngắn (Sprint) với mục tiêu cung cấp các phần chức năng hoàn chỉnh sau mỗi Sprint. Kiểm thử được thực hiện liên tục trong suốt Sprint (Testing in-sprint) để đảm bảo chất lượng, phát hiện lỗi sớm và cung cấp phản hồi nhanh cho đội ngũ phát triển (Dev) trước khi Sprint kết thúc. Cách tiếp cận này giúp giảm thiểu rủi ro và đảm bảo sản phẩm bám sát yêu cầu người dùng.

### 1.1.5. Tổng quan CI/CD và vai trò trong kiểm thử

Dự án áp dụng phương pháp tự động hóa CI/CD các bước trong quy trình phát triển phần mềm, từ lúc mã nguồn được thay đổi cho đến khi được triển khai. **Continuous Integration (CI)** tự động hợp nhất mã nguồn từ nhiều lập trình viên vào một kho lưu trữ chung. Quan trọng hơn, mỗi khi hợp nhất, hệ thống sẽ tự động chạy các bài kiểm thử cơ bản (như Unit Test, Integration Test) để phát hiện xung đột hoặc lỗi sớm. **Continuous Delivery (CD):** Tự động triển khai các thay đổi đã vượt qua CI lên môi trường kiểm thử (Testing/Staging).

CI/CD được sử dụng trong dự án để đảm bảo các thay đổi mới không phá vỡ chức năng cũ (thông qua kiểm thử hồi quy tự động), cung cấp một môi trường Staging ổn định và cập nhật liên tục cho QA thực hiện kiểm thử thủ công (manual testing) các tính năng mới, và rút ngắn đáng kể chu kỳ phản hồi lỗi

### 1.1.6. Tổng quan kiểm thử phần mềm

Dự án đánh giá hệ thống phần mềm nhằm phát hiện sự khác biệt giữa yêu cầu đã định nghĩa và kết quả thực tế. Mục tiêu không chỉ là tìm lỗi (bug) mà còn là đảm bảo phần mềm đáp ứng các yêu cầu về chức năng, hiệu năng, bảo mật và trải nghiệm người dùng. Các quy trình phân tích yêu cầu, lập kế hoạch kiểm thử (Test Planning), thiết kế kiểm thử (Test Design), thực thi kiểm thử (Test Execution), quản lý lỗi (Bug Management), báo cáo (Reporting) được thực hiện trong suốt vòng đời phát triển sẽ được trình bày chi tiết và cụ thể trong bài báo cáo

## 1.2. Phân tích yêu cầu hệ thống

### 1.2.1. Khảo sát hiện trạng và pain point nghiệp vụ

Bối cảnh chung: Trong xu hướng chuyển đổi số mạnh mẽ, thương mại điện tử đã trở thành một phần không thể thiếu trong hoạt động kinh doanh, đặc biệt là trong ngành thời trang và giày dép.

Nhu cầu người dùng: Người tiêu dùng ngày càng có xu hướng mua sắm trực tuyến nhiều hơn nhờ sự tiện lợi trong việc tìm kiếm, so sánh giá, đặt hàng và thanh toán ngay tại nhà.

Mục tiêu dự án: Dự án này được lựa chọn nhằm kiểm thử hệ thống Website bán giày trực tuyến – ShoeShop Store để đánh giá, kiểm tra và đảm bảo chất lượng phần mềm của một nền tảng thương mại điện tử mô phỏng thực tế.

Phương pháp khảo sát: Quá trình khảo sát được thực hiện thông qua việc quan sát quy trình vận hành tại các cửa hàng truyền thống và tham khảo các website thương mại điện tử phổ biến để xác định luồng hoạt động thực tế.

### 1.2.2. Phạm vi chức năng hệ thống (Module khách hàng – Module admin)

Danh mục sản phẩm (Product Catalog Service): Xem danh sách sản phẩm, lọc theo tiêu chí (thương hiệu, kích cỡ, màu sắc, tìm kiếm sản phẩm theo từ khóa. Xem chi tiết sản phẩm, bao gồm mô tả, hình ảnh và tồn kho. Thêm mới, chỉnh sửa, xóa sản phẩm và thông tin liên quan. Quản lý các danh mục, thương hiệu, kích cỡ, màu sắc.

Giỏ hàng (Shopping Cart Service): thêm/sửa/xóa số lượng sản phẩm trong giỏ hàng. Áp dụng mã giảm giá. Tính toán lại tổng tiền giỏ hàng sau khi áp dụng khuyến mãi. Điền thông tin nhận hàng, chọn phương thức vận chuyển. Tích hợp các cổng thanh toán trực tuyến hoặc chọn Thanh toán khi nhận hàng (COD).

Quản lý đơn hàng (Order Service): Xem danh sách các đơn hàng đã đặt. Theo dõi trạng thái xử lý chi tiết của từng đơn hàng. Tiếp nhận, xác nhận, và thay đổi trạng thái đơn hàng từ Khách hàng. Theo dõi, lọc, và tìm kiếm đơn hàng theo nhiều tiêu chí.

Quản lý người dùng (User Management Service): Đăng ký/Đăng nhập/Đăng xuất cho khách hàng và nhân viên. Quản lý/cập nhật hồ sơ cá nhân và thay đổi mật khẩu. Quản trị viên có thể xem và quản lý danh sách người dùng.

Kiểm soát truy cập (Access Control Service): Thiết lập và quản lý các nhóm như Admin, Manager, Staff, Customer. Đảm bảo mỗi vai trò chỉ truy cập được các chức năng theo đúng quyền hạn.

### 1.2.3. Công nghệ sử dụng (ReactJS – Spring Boot – MySQL – Vercel/Railway)

Hệ thống ShoeShop E-commerce được xây dựng theo kiến trúc Full-stack hiện đại, phân tách rõ ràng giữa Frontend và Backend để đảm bảo tính linh hoạt và khả năng mở rộng.

1. Frontend (Giao diện người dùng):

Thành phần giao diện người dùng được phát triển bằng ReactJS. Đây là một thư viện JavaScript mạnh mẽ, cho phép xây dựng giao diện tương tác cao dưới dạng ứng dụng trang đơn (Single Page Application - SPA). Việc sử dụng ReactJS giúp tối ưu hóa tốc độ tải trang và mang lại trải nghiệm mua sắm mượt mà, thân thiện với người dùng cuối.

2. Backend (Xử lý nghiệp vụ):

Phần xử lý nghiệp vụ cốt lõi được xây dựng trên framework Spring Boot sử dụng ngôn ngữ lập trình Java. Spring Boot chịu trách nhiệm cung cấp các RESTful APIs để quản lý toàn bộ logic hệ thống, bao gồm quản lý sản phẩm, giỏ hàng, tiến trình thanh toán, và các chức năng dành cho quản trị viên. Việc lựa chọn Spring Boot đảm bảo hệ thống có hiệu suất cao, tính bảo mật tốt và dễ dàng mở rộng khi quy mô kinh doanh tăng lên.

3. Cơ sở dữ liệu (Database):

Hệ quản trị cơ sở dữ liệu quan hệ MySQL được sử dụng để lưu trữ dữ liệu bền vững của hệ thống. MySQL được chọn vì tính ổn định, độ tin cậy cao và khả năng xử lý tốt các giao dịch phức tạp liên quan đến đơn hàng, tồn kho và thông tin khách hàng.

4. Triển khai (Deployment):

Ứng dụng được triển khai trên nền tảng đám mây để đảm bảo tính sẵn sàng và dễ dàng quản lý. Cụ thể, giao diện Frontend (ReactJS) được hosting trên Vercel nhờ khả năng tích hợp CI/CD (Continuous Integration/Continuous Deployment) và hiệu suất phân phối nội dung cao. Phần Backend (Spring Boot) và MySQL được triển khai trên Railway, một nền tảng hosting linh hoạt, hỗ trợ tốt cho kiến trúc microservices/API. Ngoài ra, Docker Compose được sử dụng trong môi trường phát triển để đóng gói và quản lý các thành phần Backend và Database, đảm bảo tính đồng nhất giữa các môi trường làm việc.

## 1.3 Quy trình nghiệp vụ

**Danh mục sản phẩm**

Khách hàng truy cập vào hệ thống có thể duyệt qua danh sách các sản phẩm giày đang được bày bán. Hệ thống hỗ trợ các công cụ tìm kiếm và bộ lọc nâng cao, cho phép người dùng lọc sản phẩm theo thương hiệu, kích cỡ, màu sắc hoặc khoảng giá mong muốn. Khi chọn một sản phẩm cụ thể, hệ thống hiển thị trang chi tiết bao gồm hình ảnh, mô tả, thông số kỹ thuật và tình trạng tồn kho thực tế, giúp khách hàng có đầy đủ thông tin trước khi quyết định mua hàng .

**Quy trình kiểm soát truy cập**

Để đảm bảo an toàn và phân quyền chính xác, hệ thống yêu cầu người dùng (bao gồm Khách hàng, Nhân viên, Quản lý và Quản trị viên) phải đăng nhập để truy cập các chức năng tương ứng. Khách hàng mới có thể thực hiện đăng ký tài khoản bằng cách cung cấp thông tin cá nhân cơ bản. Hệ thống xác thực thông tin đăng nhập và cấp quyền truy cập dựa trên vai trò (Role): Khách hàng được truy cập các tính năng mua sắm; Nhân viên,Quản lý truy cập quản lý đơn hàng;Admin truy cập các chức năng quản trị cấp cao. Khi kết thúc phiên làm việc, người dùng thực hiện đăng xuất để bảo mật tài khoản .

**Quy trình xử lý giỏ hàng**

Sau khi chọn được sản phẩm ưng ý, khách hàng thêm sản phẩm vào giỏ hàng. Tại đây, khách hàng có thể xem lại danh sách sản phẩm, điều chỉnh số lượng mua hoặc xóa bớt các mặt hàng không cần thiết. Hệ thống sẽ tự động tính toán lại tổng tiền hàng (tạm tính) mỗi khi có sự thay đổi về số lượng. Nếu khách hàng sở hữu mã giảm giá, họ có thể áp dụng ngay tại bước này để xem giá trị được chiết khấu trước khi tiến hành thanh toán .

**Quy trình quản lý đơn hàng**

Quá trình đặt hàng bắt đầu khi khách hàng xác nhận giỏ hàng và cung cấp thông tin giao nhận. Hệ thống hỗ trợ hai phương thức thanh toán: thanh toán khi nhận hàng (COD) hoặc thanh toán trực tuyến qua VNPay.

* Sau khi đặt hàng thành công, đơn hàng được khởi tạo với trạng thái CREATED.
* Nhân viên hoặc Quản lý sẽ kiểm tra tồn kho và xác nhận đơn hàng (CONFIRMED).
* Quy trình xử lý tiếp tục qua các bước chuẩn bị hàng (PREPARING), sẵn sàng giao (READY\_FOR\_DELIVERY) và giao cho đơn vị vận chuyển (OUT\_FOR\_DELIVERY).
* Khi khách hàng nhận được hàng, trạng thái đơn hàng chuyển sang DELIVERED.
* Trong trường hợp hủy đơn hoặc trả hàng, hệ thống sẽ thực hiện quy trình cập nhật trạng thái và hoàn tiền (đối với đơn VNPay) theo quy định .

**Quy trình quản lý người dùng**

Module này cho phép người dùng (Khách hàng) tự quản lý và cập nhật thông tin cá nhân như họ tên, địa chỉ giao hàng và mật khẩu để thuận tiện cho các lần mua sắm sau. Đối với Quản trị viên (Admin), hệ thống cung cấp quyền hạn cao nhất để quản lý toàn bộ danh sách người dùng trong hệ thống. Admin có thể xem chi tiết, tạo mới, cập nhật vai trò hoặc vô hiệu hóa (khóa) các tài khoản vi phạm chính sách hoặc các tài khoản nhân viên đã nghỉ việc .

**Quy trình quản lý mã giảm giá**

Trang quản lý khuyến mãi dành riêng cho Quản lý (Manager) để thiết lập các chiến dịch kinh doanh. Người quản lý có thể tạo mới các mã giảm giá (Coupon) với các thông số chi tiết như: mã code, mức giảm (theo phần trăm hoặc số tiền cố định), ngày bắt đầu/kết thúc, số lượng giới hạn và giá trị đơn hàng tối thiểu. Khách hàng khi thanh toán sẽ nhập mã này, hệ thống tự động kiểm tra tính hợp lệ (thời hạn, điều kiện) và trừ trực tiếp vào tổng tiền thanh toán nếu mã hợp lệ.

## 1.4. Yêu cầu nghiệp vụ

**1.4.1. User Story (As a… I want… so that…)**

| **Vai trò (As a)** | **Mong muốn (I want)** | **Mục đích (So that)** |
| --- | --- | --- |
| **Customer** | Xem danh sách sản phẩm và lọc theo thương hiệu, kích cỡ, khoảng giá | Để nhanh chóng tìm được mẫu giày phù hợp với nhu cầu và túi tiền |
| Xem chi tiết sản phẩm (mô tả, hình ảnh, tồn kho) | Để có đủ thông tin ra quyết định mua hàng |
| Thêm sản phẩm vào giỏ hàng, cập nhật số lượng hoặc xóa sản phẩm | Để quản lý những món đồ muốn mua trước khi thanh toán |
| Nhập mã giảm giá tại giỏ hàng hoặc bước thanh toán | Để được hưởng ưu đãi và giảm tổng chi phí đơn hàng |
| Thanh toán đơn hàng qua COD hoặc VNPay | Để linh hoạt lựa chọn phương thức thanh toán thuận tiện nhất |
| Theo dõi trạng thái đơn hàng (Created Delivered) và xem lịch sử đơn | Để biết tiến độ giao hàng và quản lý chi tiêu |
| **Staff** | Xem danh sách đơn hàng và xác nhận đơn dựa trên tồn kho | Để đảm bảo có hàng để giao và chuyển sang quy trình đóng gói |
| Cập nhật trạng thái đơn hàng theo tiến trình giao vận | Để khách hàng và quản lý theo dõi được hành trình đơn hàng |
| **Manager** | Tạo, chỉnh sửa và vô hiệu hóa các mã giảm giá (Discount) | Để chạy các chiến dịch kích cầu, khuyến mãi theo mùa vụ |
| **Admin** | Quản lý người dùng, phân quyền và khóa/mở khóa tài khoản | Để kiểm soát quyền truy cập và bảo mật hệ thống |

### **1.4.2. Điều kiện chấp nhận (UAC Test)**

1. **UAC 1 - Logic Đơn hàng (Order Logic):**

* Trạng thái đơn hàng phải tuân thủ trình tự: CREATED CONFIRMED PREPARING READY\_FOR\_DELIVERY DELIVERED.
* Khách hàng chỉ được phép Hủy đơn (Cancel) khi trạng thái là CREATED hoặc CONFIRMED.
* Không được phép chỉnh sửa đơn hàng sau khi trạng thái đã chuyển sang READY\_FOR\_DELIVERY.

1. **UAC 2 - Logic Tồn kho (Inventory Logic):**

* Khi đơn hàng được tạo thành công, số lượng tồn kho phải tự động trừ đi tương ứng với số lượng trong đơn hàng.
* Nhân viên không thể xác nhận đơn hàng nếu tồn kho không đủ.

1. **UAC 3 - Logic Giảm giá (Discount Logic):**

* Mã giảm giá chỉ áp dụng được khi thỏa mãn đủ 3 điều kiện:
  + Còn trong thời gian hiệu lực
  + Đơn hàng đạt giá trị tối thiểu
  + Số lượt sử dụng chưa đạt giới hạn.
* Hệ thống phải hiển thị thông báo lỗi cụ thể nếu mã không hợp lệ hoặc hết hạn.

1. **UAC 4 - Thanh toán (Payment Logic):**

* Nếu chọn VNPay, hệ thống phải chuyển hướng sang cổng thanh toán Sandbox và cập nhật trạng thái đơn hàng sau khi nhận phản hồi giao dịch thành công.

### **1.4.3. Yêu cầu chức năng (Functional Requirements)**

1. **Quản lý Danh mục (Product Catalog):**
   * Hệ thống hiển thị danh sách sản phẩm với bộ lọc và sắp xếp.
   * Hệ thống hiển thị chi tiết sản phẩm gồm: Tên, Giá, Mô tả, Size, Brand, Hình ảnh, Tồn kho.
   * Admin có quyền CRUD.
2. **Giỏ hàng (Shopping Cart):**
   * Hệ thống tính toán tổng tiền, thuế, phí ship và giảm giá theo thời gian thực khi người dùng thay đổi số lượng .
   * Hệ thống lưu trạng thái giỏ hàng của người dùng đã đăng nhập vào Database.

**3. Đặt hàng & Thanh toán (Checkout & Payment):**

* + Hệ thống cho phép nhập địa chỉ giao hàng và chọn phương thức thanh toán (COD/VNPay).
  + Hệ thống tạo bản ghi đơn hàng (CustomerOrder) và chi tiết đơn hàng (OrderDetail) vào Database khi checkout thành công.

**4. Quản lý Đơn hàng (Order Management):**

* + Staff/Admin có thể thay đổi trạng thái đơn hàng.
  + Admin/Manager có thể xem báo cáo doanh thu và tình trạng đơn hàng

**5. Quản lý Người dùng & Bảo mật (User & Access Control):**

* + Hệ thống hỗ trợ Đăng ký, Đăng nhập, Đăng xuất.
  + Admin quản lý danh sách người dùng, kích hoạt hoặc vô hiệu hóa tài khoản.
  + Người dùng có thể cập nhật thông tin cá nhân.

### **1.4.4. Yêu cầu phi chức năng (NFR: hiệu năng, bảo mật, khả dụng)**

* **Hiệu năng (Performance):**
  + Thời gian tải trang (Load page) < 2 giây.
  + Thời gian xử lý Checkout < 3 giây.
  + Hệ thống đảm bảo hoạt động ổn định với 1.000 - 2.000 người dùng đồng thời (Concurrent users).

| **Giai đoạn** | **Hoạt động chính** | **Kết quả đạt được** |
| --- | --- | --- |
| **1. Phân tích yêu cầu** | Thu thập và phân tích yêu cầu nghiệp vụ; xác định User Story, Use Case, FR/NFR; xác định 4 role (Customer, Staff, Manager, Admin). | Hoàn thành tài liệu SRS và User Story; danh sách Use Case; xác định phạm vi kiểm thử. |
| **2. Thiết kế hệ thống** | Thiết kế kiến trúc (C4), thiết kế DB (ERD), thiết kế API, thiết kế UI Flow và Wireframe; mô tả workflow xử lý cho Cart – Checkout – Order., Thiết kế kiến trúc API và Sequence Diagram cho các luồng chính | Mô hình kiến trúc hoàn chỉnh; sơ đồ dữ liệu; UI Flow; danh sách API. |
| **3. Phát triển** | Xây dựng Backend (Spring Boot), xây dựng Frontend (ReactJS); kết nối MySQL; triển khai các API chính: Authentication, Product, Cart, Order, Discount; tích hợp thanh toán; đóng gói bằng Docker Compose. | Hệ thống hoạt động đầy đủ module: Login, Catalog, Cart, Checkout, Order, RBAC; API và giao diện vận hành ổn định. |
| **4. Kiểm thử** | Thiết kế Test Case và Test Data; thực hiện kiểm thử theo mô hình chữ V: Unit Test, Integration Test, System Test và UAT .; chạy Automation bằng Selenium; chạy Regression và UAT. | Bộ Test Case hoàn chỉnh; Test plan,Test Report; danh sách Bug và mức độ; xác nhận chất lượng hệ thống. |
| **5. Triển khai & Đánh giá** | Triển khai hệ thống lên Vercel (Frontend) và Railway (Backend + MySQL); tích hợp CI/CD GitHub Actions: build → test → deploy; đánh giá kết quả chạy thử. | Phiên bản triển khai ổn định; hệ thống chạy trên môi trường Production/Staging; báo cáo đánh giá cuối sprint. |

* **Bảo mật (Security):**
  + Mật khẩu người dùng phải được mã hóa.
  + Thanh toán trực tuyến thực hiện qua giao thức HTTPS và cổng bảo mật chuẩn SSL/TLS.
  + Kiểm soát truy cập dựa trên vai trò (Role-based: Admin, Manager, Staff, Customer).
* **Tính khả dụng (Usability):**
  + Giao diện hỗ trợ Responsive Design, hiển thị tốt trên Desktop và Mobile.
* **Độ tin cậy (Reliability):**
  + Thời gian hoạt động (Uptime) đạt 99%.
  + Thời gian phục hồi sau sự cố (Recoverability) < 5 phút.

## 1.5. Kế hoạch triển khai

### 1.5.1. Kế hoạch phát triển (Development Plan)

Hệ thống ShoeShop Store được phát triển theo mô hình Agile Scrum, với các chu kỳ Sprint. Mỗi Sprint cung cấp một phần chức năng hoàn chỉnh để kiểm thử liên tục. Áp dụng theo mô hình chữ V phân chia việc phát triển phần mềm thành thiết kế, triển khai, tích hợp và kiểm tra chất lượng. Chiến lược có hệ thống này đảm bảo mỗi giai đoạn phát triển đều có một bài kiểm thử rõ ràng, từ đó giúp tạo ra phần mềm chất lượng cao. Việc kiểm thử trong từng giai đoạn giúp phát hiện sớm các lỗi, tăng tốc độ hoàn thành dự án, giảm chi phí và cải thiện chất lượng.

### 1.5.2. Kế hoạch kiểm thử tổng quan (Overview Test Plan)

#### 1.5.2.a. Mục tiêu kiểm thử

Mục đích chính của hoạt động kiểm thử đối với hệ thống ShoeShop Store là đảm bảo chất lượng phần mềm toàn diện trước khi đưa vào vận hành, cụ thể bao gồm:

* Xác minh chức năng nghiệp vụ **:** Đảm bảo toàn bộ nghiệp vụ thương mại điện tử hoạt động đúng theo yêu cầu đã định nghĩa. Các module trọng tâm bao gồm: Danh mục Sản phẩm (Product Catalog), Giỏ hàng và Thanh toán (Shopping Cart và Checkout), Quản lý Đơn hàng (Order Management), Quản lý Người dùng và Phân quyền (User Management và Access Control), Quản lý Giảm giá (Discount Management).
* Đánh giá phi chức năng**:** Kiểm tra các yếu tố về hiệu năng, bảo mật, khả năng sử dụng, độ tin cậy và khả năng tương thích của hệ thống
* Phát hiện và giảm thiểu rủi ro**:** Tìm kiếm lỗi (bug) sớm nhất có thể thông qua quy trình kiểm thử liên tục trong từng Sprint (Testing in-sprint) và mô hình chữ V, giúp giảm chi phí sửa lỗi và rủi ro khi vận hành
* Đảm bảo tính ổn định**:** Xác nhận hệ thống đạt đủ tiêu chuẩn để người dùng cuối chấp nhận (UAT) thông qua các luồng kiểm thử End-to-End

1.5.2.b. Phương pháp kiểm thử

Kế hoạch kiểm thử áp dụng kết hợp nhiều phương pháp để đảm bảo chất lượng toàn diện, đặc biệt kết hợp với quy trình Agile Scrum và mô hình chữ V để đảm bảo kiểm thử được thực hiện xuyên suốt từ giai đoạn phân tích yêu cầu cho đến phát triển, tích hợp và nghiệm thu sản phẩm

* Các cấp độ kiểm thử

Quy trình kiểm thử được triển khai tuần tự qua bốn cấp độ chính, tương ứng với giai đoạn Validation trong mô hình chữ V, nhằm đảm bảo chất lượng từ chi tiết nhỏ nhất đến tổng thể hệ thống: Khởi đầu là Unit Test (Kiểm thử đơn vị) do đội ngũ phát triển thực hiện. Sử dụng các công cụ như JUnit và Mockito, các lập trình viên sẽ kiểm tra tính đúng đắn của từng hàm và service riêng lẻ, tập trung vào các logic nghiệp vụ cốt lõi như tính toán tổng tiền, kiểm tra tồn kho hay xác thực tài khoản. Sau khi các đơn vị code đã ổn định, quy trình chuyển sang Integration Test (Kiểm thử tích hợp) nhằm xác minh sự tương tác liền mạch giữa các module quan trọng (ví dụ: luồng dữ liệu từ Giỏ hàng sang Thanh toán) cũng như tính toàn vẹn dữ liệu giữa Backend và Database.  
Khi các thành phần đã được tích hợp hoàn chỉnh, đội ngũ QA sẽ tiến hành System Test (Kiểm thử hệ thống). Giai đoạn này thực hiện kiểm thử End-to-End dựa trên các quy trình nghiệp vụ thực tế của người dùng, từ lúc đăng nhập, tìm kiếm sản phẩm cho đến khi hoàn tất thanh toán và theo dõi đơn hàng, nhằm đánh giá toàn diện về tính ổn định, hiệu năng và trải nghiệm UI/UX. Cuối cùng, trước khi bàn giao sản phẩm, Acceptance Test (UAT) sẽ được thực hiện trên môi trường Staging. Đây là bước kiểm tra dưới góc nhìn của người dùng cuối để xác nhận hệ thống đã đáp ứng đầy đủ các yêu cầu nghiệp vụ đã đề ra.

* Kỹ thuật và Loại hình kiểm thử

Để tối ưu hóa hiệu quả, dự án áp dụng linh hoạt giữa kiểm thử thủ công và tự động hóa:  
Đối với các luồng nghiệp vụ phức tạp, đòi hỏi tư duy logic của con người hoặc các yếu tố liên quan đến giao diện và cảm nhận người dùng (UI/UX), nhóm sẽ áp dụng Kiểm thử thủ công (Manual Testing). Phương pháp này giúp phát hiện các vấn đề về tính dễ dùng, khả năng điều hướng và tính chính xác của các thông báo lỗi mà máy móc khó nhận biết.  
Ngược lại, Kiểm thử tự động (Automation Testing) được ưu tiên cho các tác vụ lặp lại thường xuyên nhằm tiết kiệm nguồn lực và đảm bảo tính ổn định. Cụ thể, *Selenium WebDriver* được sử dụng để tự động hóa các kịch bản giao diện như đăng nhập hay checkout, trong khi *Postman/Newman* chịu trách nhiệm gửi request tự động để xác minh cấu trúc JSON và mã trạng thái HTTP của các API Backend. Bên cạnh đó, Kiểm thử hiệu năng (Performance Testing) cũng được chú trọng bằng việc sử dụng *Apache JMeter* để đo lường khả năng chịu tải (load/stress test) của các API có tần suất truy cập cao như danh sách sản phẩm hay đặt hàng.

* Tích hợp kiểm thử vào quy trình CI/CD

Nhằm đảm bảo hệ thống luôn ở trạng thái sẵn sàng triển khai, hoạt động kiểm thử được tích hợp sâu vào pipeline GitHub Actions. Hệ thống được cấu hình để tự động kích hoạt Unit Test và Integration Test mỗi khi có Pull Request hoặc thao tác hợp nhất mã nguồn (Merge). Đồng thời, các bộ test API cũng được chạy tự động để kiểm tra tính nhất quán của dữ liệu. Quy trình này giúp phát hiện lỗi sớm, giảm thiểu rủi ro hồi quy (Regression) và đảm bảo rằng mọi thay đổi mới đều không phá vỡ các tính năng đang hoạt động ổn định trước khi deploy sang môi trường Staging.

| Nhóm nội dung | Tóm tắt phương pháp |
| --- | --- |
| Các cấp độ kiểm thử (Test Levels) | * Unit Test: Dev kiểm thử từng hàm bằng JUnit/Mockito. * Integration Test: Kiểm tra tương tác giữa module và giữa Backend – Database. * System Test: QA thực hiện kiểm thử End-to-End toàn bộ quy trình nghiệp vụ. * Acceptance Test (UAT): Xác nhận yêu cầu nghiệp vụ trên môi trường Staging trước khi bàn giao. |
| Kỹ thuật & loại hình kiểm thử | * Manual Testing: Áp dụng cho UI/UX, luồng nghiệp vụ phức tạp, kiểm thử trải nghiệm người dùng. * Automation Testing: Selenium cho UI; Postman/Newman cho API; phù hợp với test lặp lại. * Performance Testing: JMeter kiểm tra tải, stress, thời gian phản hồi API quan trọng. |
| Tích hợp trong CI/CD | * Kiểm thử được tích hợp vào GitHub Actions. * Tự động chạy Unit Test + Integration Test khi có Pull Request hoặc Merge. * Tự động chạy API Test để kiểm tra tính nhất quán dữ liệu. * Đảm bảo không phát sinh lỗi hồi quy trước khi deploy sang Staging. |

# CHƯƠNG 2 – PHÂN TÍCH & THIẾT KẾ HỆ THỐNG

## TỔNG QUAN CHƯƠNG 2

Chương 2 trình bày thiết kế hệ thống và kiến trúc kỹ thuật của Shoeshop với mục tiêu phục vụ trực tiếp cho hoạt động kiểm thử phần mềm. Nội dung tập trung mô tả cách hệ thống vận hành ở hai góc nhìn: nghiệp vụ (System Design) và kỹ thuật (Architecture Design), qua đó giúp xác định đối tượng kiểm thử, luồng dữ liệu quan trọng và các điểm rủi ro cần được ưu tiên khi thiết kế kịch bản kiểm thử.

Ở mức nghiệp vụ, chương *mô tả phân rã chức năng của hệ thống, luồng xử lý chính của khách hàng và quản trị viên, cùng mô hình dữ liệu khái niệm*. => giúp kiểm thử viên hiểu rõ hệ thống “làm gì” và “xử lý theo trình tự nào”, làm cơ sở xây dựng Test Scenario và xác định vùng kiểm thử quan trọng.

Ở mức kỹ thuật, chương mô tả kiến trúc hệ thống gồm các lớp Frontend – Backend – Database, cơ chế giao tiếp thông qua API và luồng dữ liệu nội bộ. Thông tin này hỗ trợ việc lựa chọn loại kiểm thử phù hợp như kiểm thử API, kiểm thử tích hợp, kiểm thử cơ sở dữ liệu hay kiểm thử hiệu năng.

Chương 2 đóng vai trò nền tảng cho Chương 3 và Chương 4: các thiết kế, luồng xử lý và kiến trúc trình bày tại đây được sử dụng để xây dựng kế hoạch kiểm thử, chiến lược kiểm thử và phương pháp luận kiểm thử của toàn đề tài. Các mô hình chi tiết như ERD đầy đủ, sequence diagram, API list và mockup giao diện sẽ được trình bày tại Phụ lục.

## 2.1. Thiết kế hệ thống (System Design – Business View)

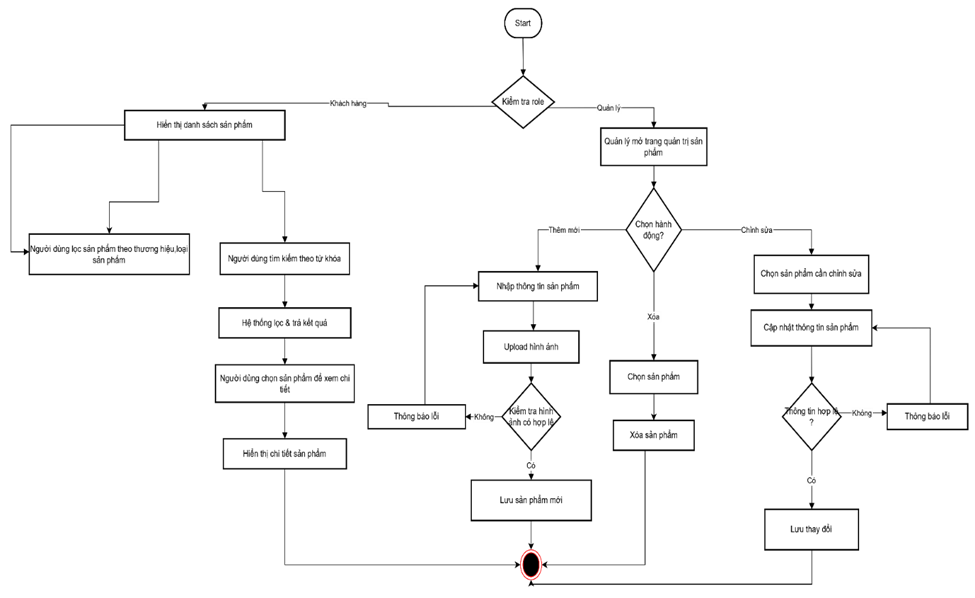
### 2.1.1. Thiết kế chức năng

Sơ đồ bối cảnh nghiệp vụ (Business Context)

Mục này trình bày chi tiết về các chức năng nghiệp vụ cốt lõi của hệ thống **ShoeShop E-commerce**.

**1. Danh mục sản phẩm (Product Catalog Service)**

**Mô tả:** Module này đóng vai trò quản lý trung tâm và hiển thị toàn bộ dữ liệu liên quan đến sản phẩm của cửa hàng. Chức năng này cho phép khách hàng dễ dàng tiếp cận sản phẩm thông qua các công cụ tìm kiếm và lọc, đồng thời cung cấp cho quản lý khả năng kiểm soát và cập nhật thông tin hàng hóa một cách chính xác.



*Hình 1.Activity Product*

**Người dùng (Khách hàng) có thể:**

* + **Lọc sản phẩm:** Khách hàng có thể sử dụng các bộ lọc có sẵn như thương hiệu (brand), loại sản phẩm (category), Hệ thống sẽ áp dụng các tiêu chí này và chỉ hiển thị danh sách những sản phẩm thỏa mãn điều kiện lọc mà người dùng đã chọn.
  + **Tìm kiếm sản phẩm:** Khách hàng nhập tên hoặc từ khóa liên quan đến sản phẩm cần tìm vào ô tìm kiếm. Hệ thống thực hiện truy vấn trong cơ sở dữ liệu và trả về danh sách các sản phẩm có tên hoặc đặc điểm khớp với từ khóa
  + **Xem chi tiết sản phẩm**: bao gồm mô tả, hình ảnh và thông tin về **số lượng tồn kho** (Availability).

**Quản lý có thể:**

* + Thực hiện các thao tác **Thêm mới, Chỉnh sửa, Xóa** sản phẩm và thông tin liên quan (giá, mô tả, hình ảnh).

**2. Giỏ hàng & Thanh toán (Shopping Cart & Checkout Service)**

**Mô tả:** Module này quản lý toàn bộ quy trình từ lúc khách hàng chọn sản phẩm, áp dụng các ưu đãi cho đến khi hoàn tất thanh toán và khởi tạo đơn hàng.



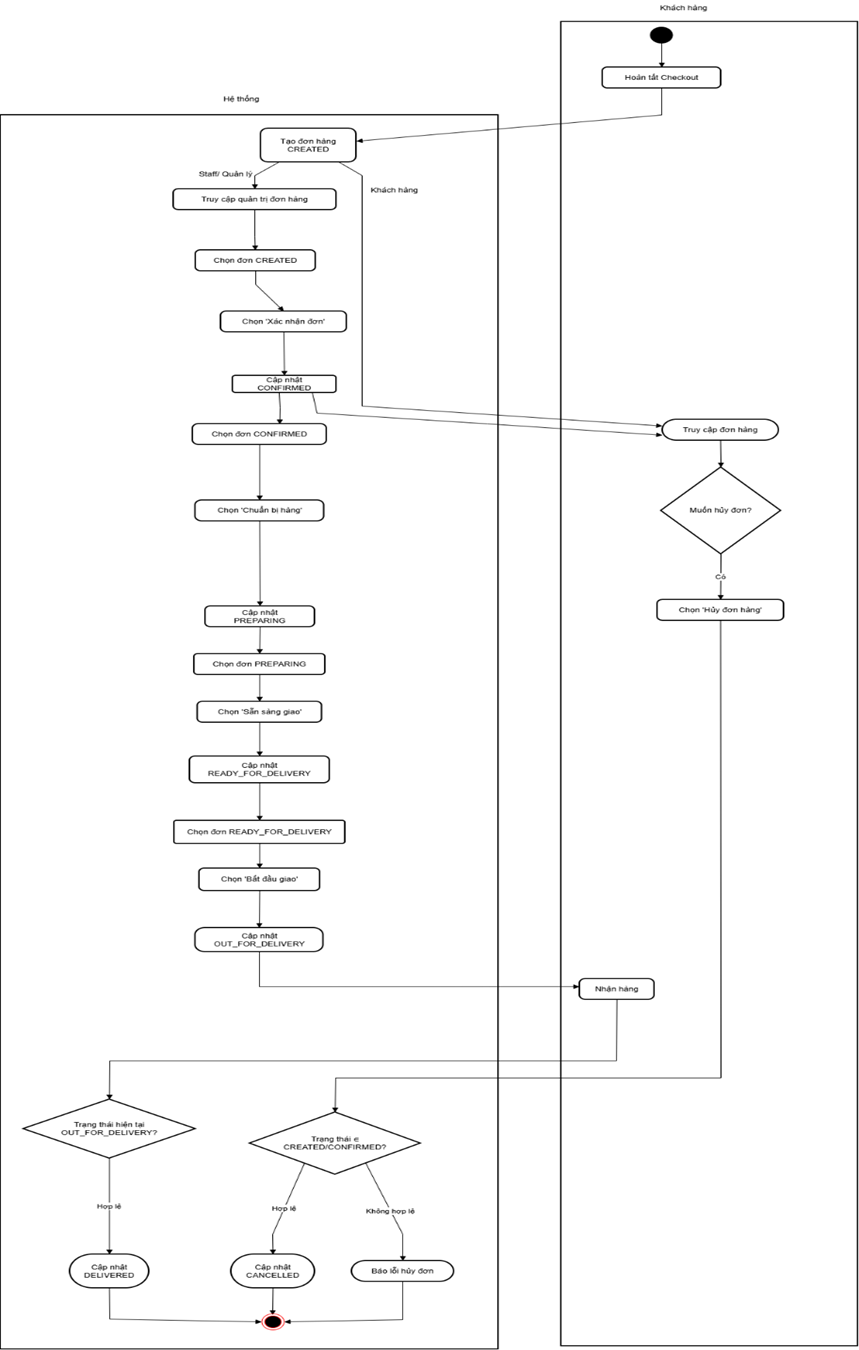
*Hình 2.Activity Cart*

**Người dùng (Khách hàng) có thể:**

* + **Quản lý giỏ hàng:** Khách hàng có thể thực hiện các thao tác **Thêm/Sửa/Xóa** sản phẩm trong giỏ hàng một cách linh hoạt. Khi người dùng thay đổi số lượng hoặc loại bỏ sản phẩm, hệ thống sẽ tự động cập nhật lại danh sách và tổng giá trị tạm tính của giỏ hàng.
  + **Áp dụng mã giảm giá:** Khách hàng có thể nhập mã giảm giá (Coupon Code) nếu có. Hệ thống sẽ kiểm tra tính hợp lệ của mã (thời hạn, điều kiện áp dụng) và tự động tính toán lại tổng tiền phải thanh toán sau khi đã trừ đi giá trị khuyến mãi.
  + **Điền thông tin nhận hàng**: Tại bước thanh toán, khách hàng cần điền thông tin nhận hàng đầy đủ (tên, số điện thoại, địa chỉ cụ thể) và chọn phương thức vận chuyển phù hợp với nhu cầu và chi phí.
  + Chọn phương thức thanh toán: **Tích hợp các cổng thanh toán trực tuyến** hoặc chọn **Thanh toán khi nhận hàng (COD)**.

**3. Quản lý đơn hàng (Order Service)**

**Mô tả:** Module này đóng vai trò trung tâm trong việc đảm bảo quá trình xử lý đơn hàng diễn ra suôn sẻ và minh bạch. Chức năng này cho phép cả khách hàng và nhân viên theo dõi sát sao tiến độ đơn hàng từ lúc khởi tạo đến khi giao thành công.



*Hình 3. Activity Order*

**Người dùng (Khách hàng) có thể:**

* + Xem danh sách các đơn hàng đã đặt trong lịch sử.
  + **Theo dõi trạng thái:** Khách hàng nắm bắt được trạng thái xử lý chi tiết của từng đơn hàng theo thời gian thực (ví dụ: Chờ xác nhận, Đang chuẩn bị hàng, Đang giao, Đã giao thành công).
  + **Hủy đơn hàng:** Trong trường hợp thay đổi ý định, khách hàng có thể chủ động hủy đơn hàng nếu đơn hàng đó vẫn đang ở trạng thái **"Chờ xác nhận"** (chưa được nhân viên xử lý).
* **Quản lý/Nhân viên có thể:**
  + **Cập nhật trạng thái:** Nhân viên thực hiện quy trình nghiệp vụ như Tiếp nhận -> Xác nhận (kiểm tra kho) -> Đóng gói -> Giao hàng. Mỗi bước đều được cập nhật trạng thái lên hệ thống để khách hàng theo dõi.
  + **Quản lý danh sách:** Quản lý có thể sử dụng các công cụ **Lọc và Tìm kiếm** nâng cao để tra cứu đơn hàng theo nhiều tiêu chí khác nhau (theo ngày đặt, trạng thái đơn, tên khách hàng), giúp việc quản lý số lượng lớn đơn hàng trở nên hiệu quả.

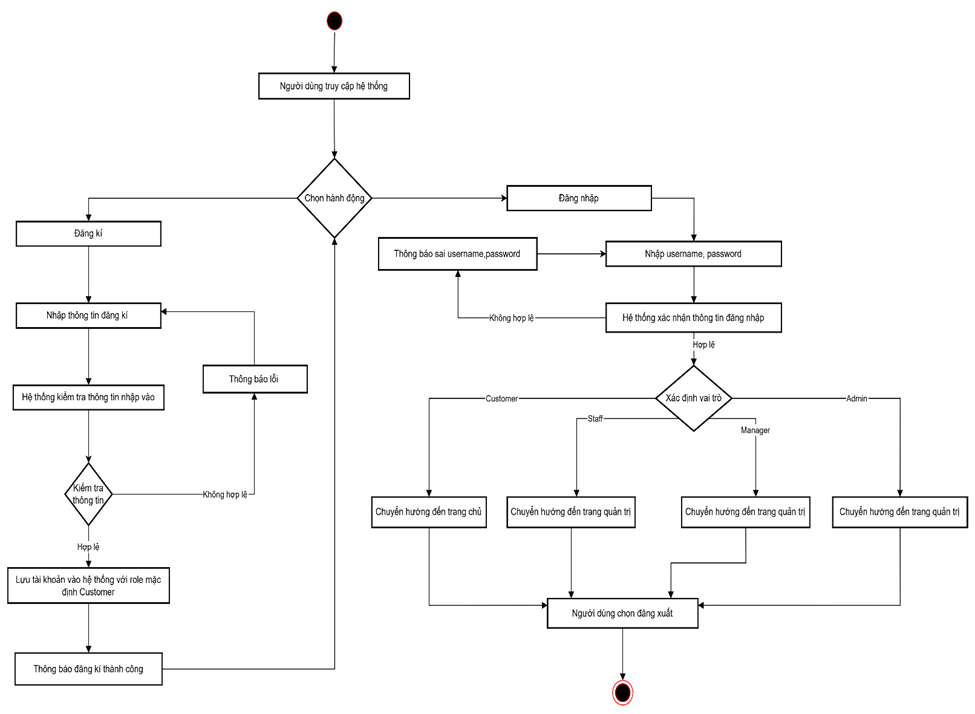
**4. Quản lý người dùng (User Management Service)**

Module này quản lý thông tin tài khoản và hồ sơ người dùng trong hệ thống.

* **Chức năng chung:**
  + Hỗ trợ quy trình **Đăng ký/Đăng nhập/Đăng xuất** cho cả khách hàng và nhân viên.
  + Cho phép người dùng **Quản lý/cập nhật hồ sơ cá nhân** và **thay đổi mật khẩu**.
* **Admin có thể:**
  + **Xem và quản lý** danh sách tài khoản người dùng, bao gồm cả khách hàng và nhân viên.

**5. Kiểm soát truy cập (Access Control Service)**

**Mô tả:** Module này đóng vai trò quan trọng trong việc bảo vệ hệ thống và đảm bảo tính toàn vẹn dữ liệu. Chức năng chính là xác thực danh tính người dùng và cấp quyền truy cập vào các tài nguyên phù hợp dựa trên vai trò của họ.



*Hình 4. Activity Access Control*

**Chức năng chính:**

* + **Đăng ký:** Người dùng mới có thể tạo tài khoản bằng cách cung cấp thông tin cá nhân (email, mật khẩu). Hệ thống sẽ kiểm tra tính hợp lệ và duy nhất của thông tin trước khi tạo tài khoản với vai trò mặc định là Khách hàng (Customer).
  + **Đăng nhập:** Người dùng truy cập hệ thống bằng tên đăng nhập và mật khẩu. Hệ thống xác thực thông tin; nếu đúng, người dùng được cấp quyền truy cập (thường thông qua token hoặc session).

**Phân quyền (Authorization):** Sau khi đăng nhập thành công, hệ thống sẽ tự động điều hướng người dùng đến giao diện làm việc phù hợp với vai trò của họ:

o **Customer:** Truy cập trang chủ, xem sản phẩm, giỏ hàng, lịch sử đơn hàng.

o **Staff (Nhân viên):** Truy cập trang quản lý đơn hàng để xử lý.

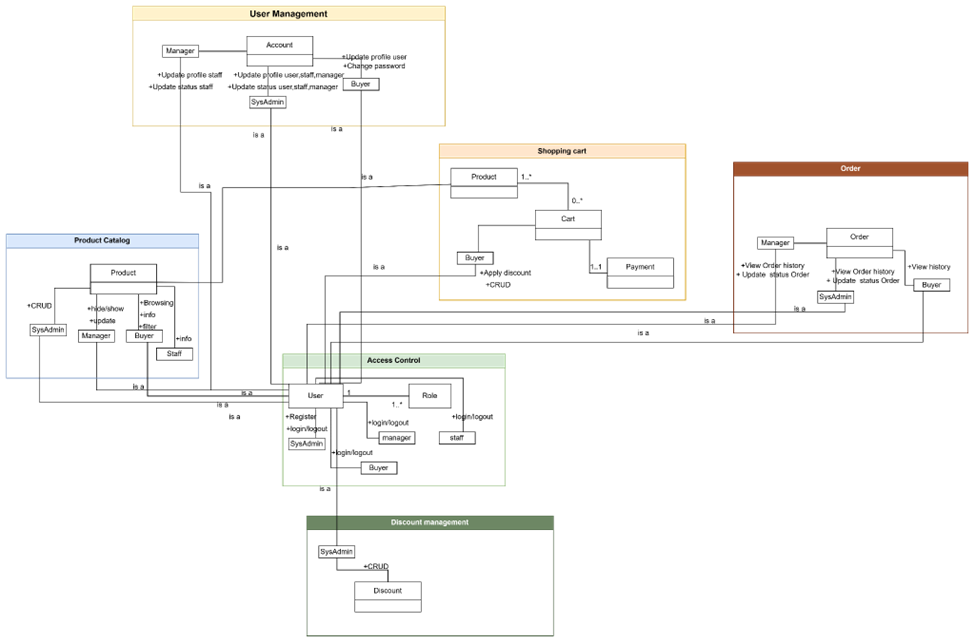
o **Manager (Quản lý):** Truy cập các chức năng quản lý sản phẩm, khuyến mãi, báo cáo.

o **Admin (Quản trị viên):** Truy cập toàn quyền hệ thống, bao gồm quản lý người dùng và cấu hình hệ thống.

* + **Đăng xuất:** Người dùng có thể kết thúc phiên làm việc an toàn. Hệ thống sẽ hủy token/session hiện tại và đưa người dùng về trang đăng nhập hoặc trang chủ công khai.

### 2.1.2. Thiết kế dữ liệu

Conceptual Data Model (ERD mức khái niệm)



*Hình 5. Conceptual Data Model*

Sơ đồ ERD mô tả cấu trúc dữ liệu của hệ thống được tổ chức chặt chẽ theo 5 nhóm chức năng chính:

- **User Management & Access Control:** Thực thể trung tâm User kết hợp với Role để thực hiện phân quyền. Các vai trò cụ thể như Buyer (Khách hàng), Manager, SysAdmin đều có mối quan hệ kế thừa "is a" với thực thể Account để quản lý thông tin đăng nhập tập trung.

- **Product Catalog:** Tập trung vào thực thể Product. Dữ liệu này liên kết trực tiếp với các hành động của người dùng: Buyer thực hiện xem/lọc (**Browsing/Filter**) và Manager thực hiện cập nhật/quản lý (**Update/CRUD**).

- **Shopping Cart:** Cart đóng vai trò lưu trữ tạm thời danh sách các Product mà Buyer chọn (quan hệ 1..\*). Thực thể này liên kết 1-1 với Payment để ghi nhận thông tin thanh toán trước khi chuyển đổi thành đơn hàng.

- **Order Service:** Order là thực thể dữ liệu chính được tạo ra sau quy trình mua hàng. Dữ liệu này được Buyer truy xuất để xem lịch sử (**View history**) và Manager tác động để cập nhật trạng thái xử lý (**Update status**).

- **Discount Management:** Thực thể Discount chứa thông tin mã giảm giá do Manager khởi tạo (**CRUD**), và được liên kết vào quy trình mua hàng khi Buyer thực hiện áp dụng mã (**Apply discount**).

ERD mức logic



*Hình 6. ERD Logic*

Ghi chú: Data dictionary: file sheet data dictionary

### 2.1.3. Thiết kế giao diện

**UI Flow (Customer → Cart → Payment → Orders)**

* + Trang chủ/Danh sách (Product List): Người dùng xem và tìm kiếm sản phẩm (Chọn sản phẩm)
  + Chi tiết sản phẩm (Product Detail): Xem thông tin, chọn size, số lượng Nhấn "Thêm vào giỏ".
  + Giỏ hàng (Cart Page): Xem lại danh sách, sửa số lượng, nhập mã giảm giá Nhấn "Thanh toán”.
  + Thanh toán (Checkout Page): Nhập địa chỉ, Chọn phương thức thanh toán (COD/VNPay) Nhấn "Đặt hàng".
    - *(Nếu chọn VNPay):* Chuyển hướng sang trang thanh toán VNPay Quay lại.
    - Thành công (Order Success): Thông báo đặt hàng thành công.
  + Lịch sử đơn hàng (Order History): Người dùng xem trạng thái xử lý đơn hàng.

**Wireframe các trang chính**

Màn hình Danh sách sản phẩm (Product List UI)

* Header: Logo, Icon Giỏ hàng (có badge số lượng), Avatar User (Đăng nhập/Đăng ký).

A screenshot of a website

AI-generated content may be incorrect.

*Hình 7. UI Trang Chủ*

* Sidebar (Bộ lọc):
  + Filter theo hang giày (Nike, Adidas...)
  + Filter theo loại giày (Running, Casual...)
  + Filter theo giới tính (Nam, Nữ, Unisex)
* Main Content (Lưới sản phẩm): Hiển thị dạng Grid. Mỗi thẻ sản phẩm gồm:
  + Hình ảnh.
  + Tên sản phẩm.
  + Giá bán và Giá gốc gạch ngang.
  + Nút nhanh: "Xem chi tiết".

A screenshot of a website

AI-generated content may be incorrect.

*Hình 8. UI Sản Phẩm*

Màn hình Giỏ hàng (Cart UI)

* Bảng danh sách:
  + Cột Sản phẩm: Hình ảnh thumbnail + Tên + Size đã chọn.
  + Cột Đơn giá.
  + Cột Số lượng: Input số hoặc nút +/- (Cập nhật giá Real-time).
  + Cột Thành tiền (Đơn giá \* Số lượng).
  + Nút Xóa.
* Bảng tổng kết (Order Summary):
  + Tạm tính (Subtotal).
  + Mã giảm giá: Ô input code + Nút "Áp dụng". Hiển thị số tiền được giảm.
  + Tổng cộng (Total): Số tiền cuối cùng phải thanh toán.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 9. UI Giỏ Hàng*

Màn hình Thanh toán (Checkout UI)

* Thông tin vận chuyển: Form nhập (Họ tên, SĐT, Địa chỉ chi tiết). *Yêu cầu validate không được để trống.*
* Review đơn hàng: Tóm tắt lại danh sách món hàng và tổng tiền lần cuối trước khi chốt .
* Phương thức thanh toán:
  + Thanh toán khi nhận hàng (COD).
  + Thanh toán VNPay (Redirect sang cổng thanh toán).

A screenshot of a computer

AI-generated content may be incorrect.

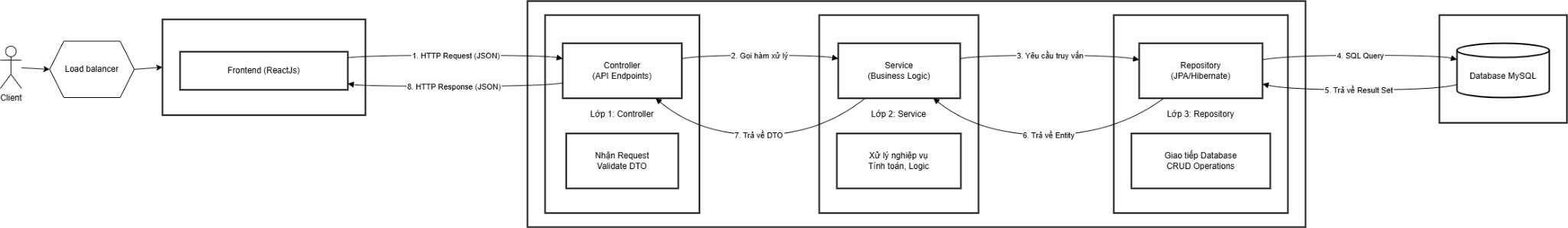
*Hình 10. UI Thanh Toán*

**Mapping UI → Test nhóm giao diện**

| Màn hình (UI) | Thành phần (Element) | Kịch bản kiểm thử (Test Scenario) | Hình ảnh |
| --- | --- | --- | --- |
| Product List | Bộ lọc (Filter) | Chọn Brand "Adidas" Chỉ hiển thị giày Adidas |  |
| Product Detail | Nút "Thêm vào giỏ" | Nhấn thêm Icon giỏ hàng trên Header phải nhảy số (+1). |  |
| Cart | Mã giảm giá | Nhập mã hết hạn/sai Thông báo lỗi đỏ "Mã không hợp lệ". |  |
| Order History | Trạng thái đơn | Đơn hàng vừa đặt phải có trạng thái CREATED (hoặc PENDING nếu chưa thanh toán). |  |

## 2.2. Thiết kế kiến trúc (Architecture Design – Technical View)

### 2.2.1. Tổng quan kiến trúc

****

*Hình 11. Architecture Design*

* Luồng đi của dữ liệu sẽ là: Client (Frontend) -> Controller -> Service -> Repository -> Database
* Khối Client (Phía người dùng): Đại diện là trình duyệt web (Browser). Chức năng là gửi yêu cầu HTTP (GET, POST, PUT, DELETE) đến Server, gửi dữ liệu dưới dạng JSON (ví dụ: thông tin đặt hàng, thông tin đăng nhập), nhận phản hồi (Response) từ Server để hiển thị cho người dùng.
* Khối Controller Layer (Tầng Giao diện/Điều hướng): Đại diện trong code: Các class ShoeController, CartController, AuthController. Chức năng là hứng các request từ Client gửi lên, validate dữ liệu đầu vào, không xử lý logic phức tạp, nó chỉ điều phối việc "ai sẽ làm gì". Gọi xuống Service Layer để yêu cầu xử lý nghiệp vụ, trả về kết quả (APIResponse, HTTP Status 200, 400, 500) cho Client.

#### Khối Service Layer (Tầng Nghiệp vụ): Đại diện trong code là các class ShoeService, CartService, AuthService, chức năng: Là "Bộ não" của hệ thống. Chứa toàn bộ logic kinh doanh (Business Logic), Ví dụ: Kiểm tra tồn kho, tính tổng tiền giỏ hàng, áp mã giảm giá, mã hóa mật khẩu, chuyển đổi dữ liệu giữa DTO (Data Transfer Object) và Entity, gọi xuống Repository Layer để lấy hoặc lưu dữ liệu

* Khối Repository Layer (Tầng Truy xuất dữ liệu): Đại diện trong code là các interface ShoeRepository, CartRepository, UserRepository. Chức năng: Trực tiếp làm việc với Cơ sở dữ liệu, thực hiện các câu lệnh truy vấn (Query) như: findById, save, delete, findByStatusTrue…, map dữ liệu từ bảng trong Database thành các Java Object (Entity) thông qua Hibernate/JPA.

#### Khối Database (Cơ sở dữ liệu): Đại diện: MySQL. Chức năng là lưu trữ bền vững toàn bộ dữ liệu (Bảng User, Shoe, Order, Cart...).

* Luồng dữ liệu (Data Flow) ví dụ cho chức năng "Tạo sản phẩm":

1. Client: Gửi POST /shoes kèm JSON thông tin giày.
2. Controller (ShoeController):

* Nhận request.
* Kiểm tra @Valid (tên không rỗng, giá > 0).
* Gọi shoeService.createShoe(request).

1. Service (ShoeService):

* Nhận ShoeCreateRequest.
* Kiểm tra Brand có tồn tại không?
* Chuyển ShoeCreateRequest -> Shoe Entity.
* Gọi shoeRepository.save(shoe).

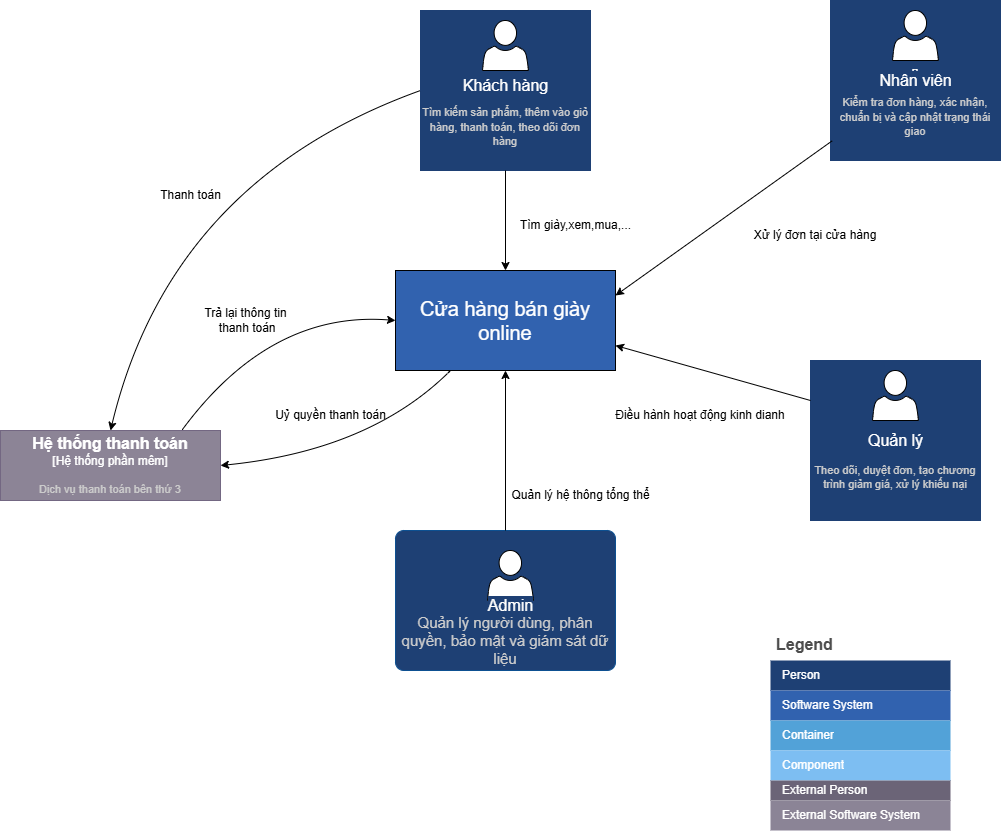
1. Repository (ShoeRepository):

* Sinh câu lệnh INSERT INTO shoe ....
* Gửi xuống Database.

1. Database: Lưu dữ liệu và trả về ID mới tạo.
2. Service: Nhận lại Entity đã lưu, chuyển thành ShoeResponse.
3. Controller: Đóng gói ShoeResponse vào APIResponse và trả về 200

### 2.2.2. Phân tích thành phần

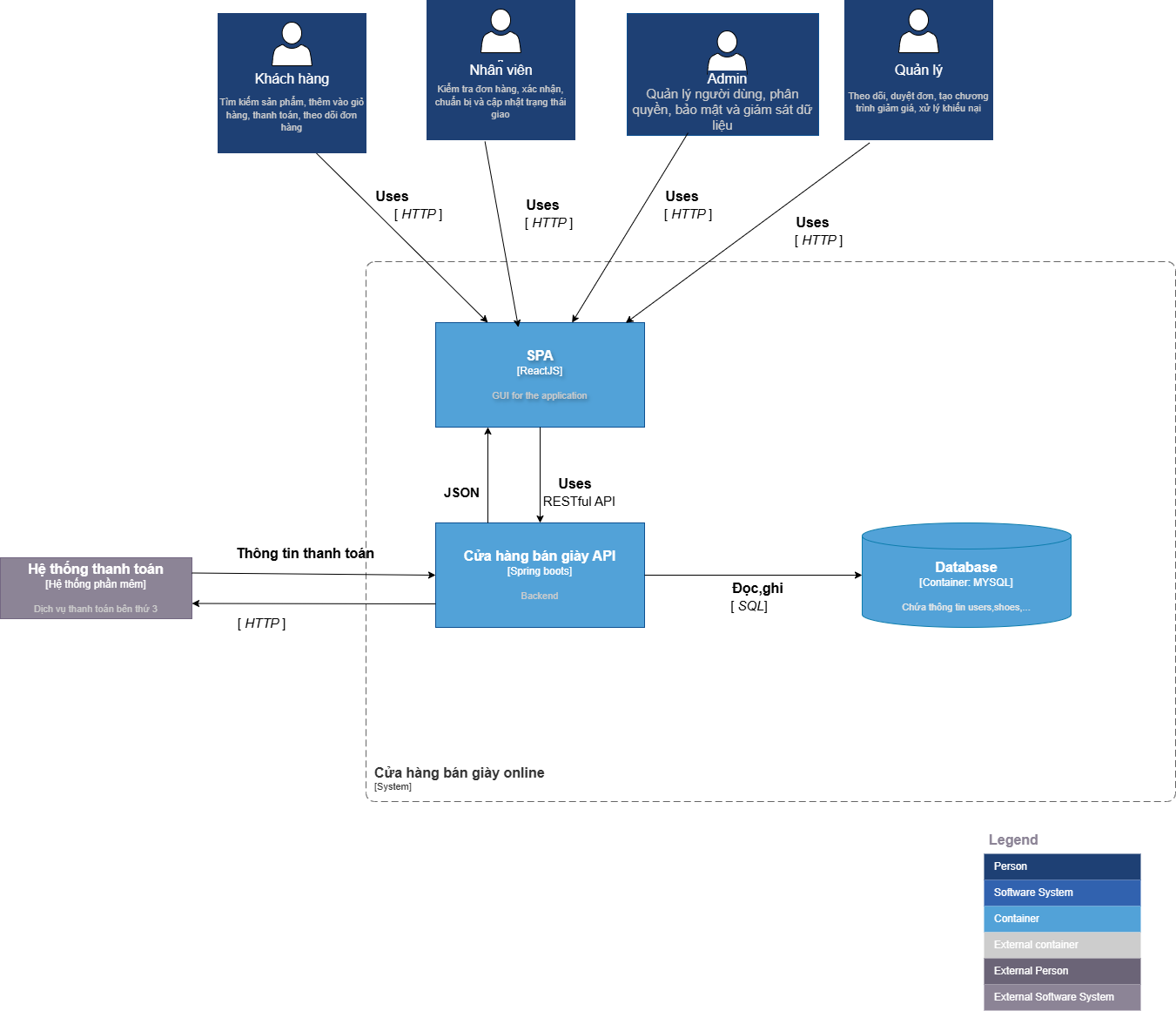
1. Lược đồ C1



*Hình 12. C1 Model*

* Hệ thống gồm 4 người dùng là admin, khách hàng,quản lý và nhân viên.
* Customer (Khách hàng) là Người dùng cuối, mua sắm sản phẩm,có thể tìm kiếm sản phẩm, thêm vào giỏ hàng, thanh toán, theo dõi đơn hàng
* Nhân viên cửa hàng là Người xử lý đơn hàng tại cửa hàng, có thể Kiểm tra đơn hàng, xác nhận, chuẩn bị và cập nhật trạng thái giao
* Quản lý cửa hàng là người chịu trách nhiệm điều hành hoạt động kinh doanh có thể theo dõi, duyệt đơn, tạo chương trình giảm giá, xử lý khiếu nại
* Admin là người quản lý hệ thống tổng thể, có thể quản lý người dùng, phân quyền, bảo mật và giám sát dữ liệu

1. Lược đồ C2



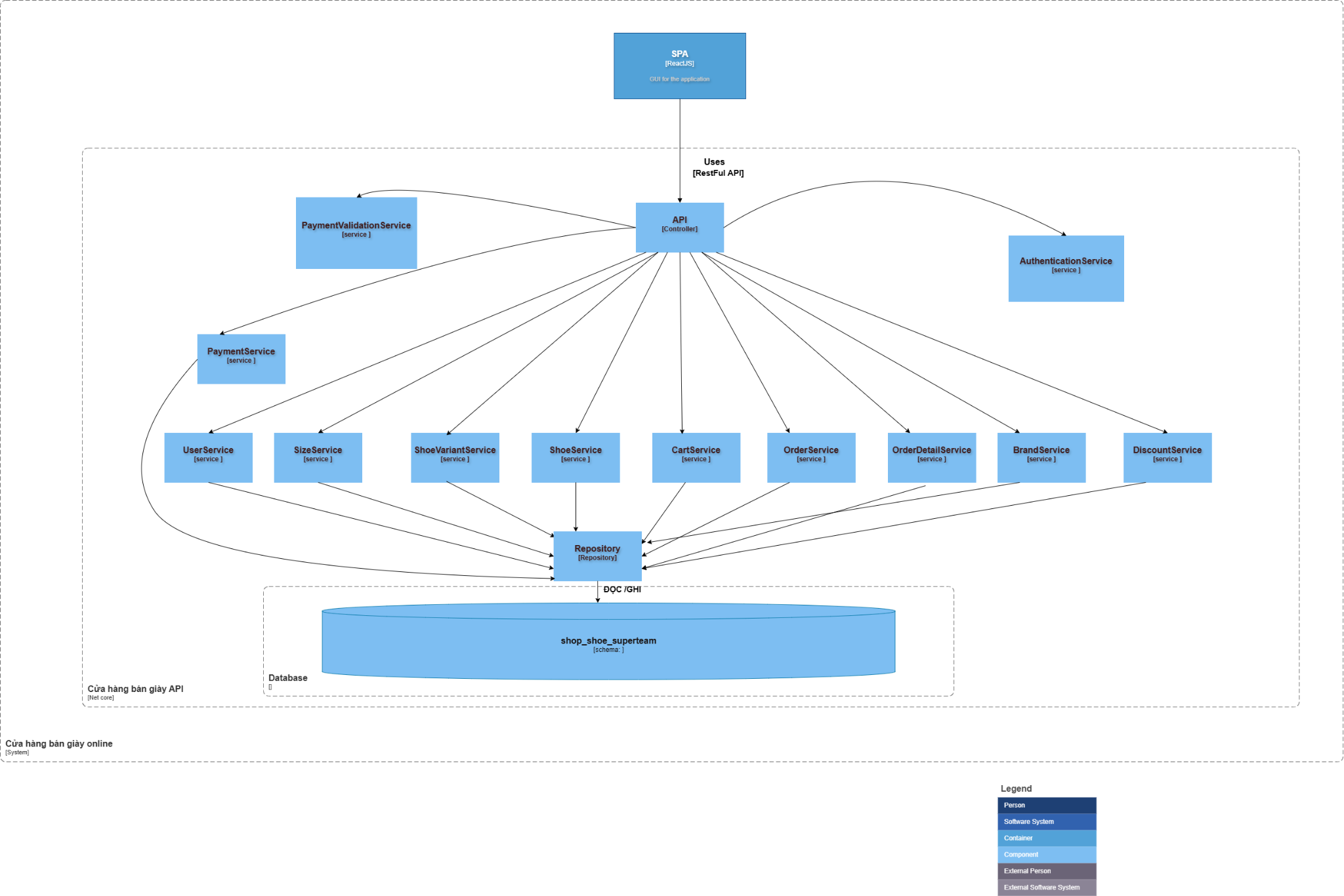
*Hình 12. C2 Model*

* SPA (Single Page Application - Ứng dụng một trang):
* Loại: Container (biểu diễn bằng hình chữ nhật màu xanh lam nhạt).
* Công nghệ: ReactJS.
* Vai trò: Cung cấp Giao diện người dùng đồ họa (GUI) cho toàn bộ ứng dụng. Đây là nơi người dùng (Khách hàng, Nhân viên, Quản lý, Admin) tương tác trực tiếp.
* Tương tác: Nhận yêu cầu qua HTTP từ tất cả 4 loại người dùng. Giao tiếp với "Cửa hàng bán giày API" thông qua JSON để gửi/nhận dữ liệu.
* Cửa hàng bán giày API (Backend)  
   + Loại: Container (biểu diễn bằng hình chữ nhật màu xanh lam nhạt).
* Công nghệ: Spring Boot (ngôn ngữ Java).
* Vai trò: Đây là trái tim của hệ thống, xử lý toàn bộ logic nghiệp vụ (business logic) của ứng dụng. Nó cung cấp các dịch vụ RESTful API để SPA có thể gọi và tương tác
* Tương tác: Nhận yêu cầu (Uses) RESTful API từ SPA. Đọc/Ghi (Đọc/ghi) dữ liệu vào "Database". Tương tác (Thông tin thanh toán) với "Hệ thống thanh toán" bên ngoài
* Các Hệ thống Bên ngoài (External Software System): Hệ thống tương tác với một hệ thống phần mềm bên ngoài là hệ thống thanh toán (Hệ thống ngân hàng):
* Loại: External Software System (biểu diễn bằng hình chữ nhật màu xám).
* Công nghệ: Dịch vụ thanh toán bên thứ 3.
* Vai trò: Xử lý các giao dịch thanh toán.
* Tương tác: "Cửa hàng bán giày API" gửi yêu cầu thanh toán (HTTP) tới hệ thống này và nhận lại kết quả.

#### Cơ sở dữ liệu (Database)

* Loại: Container (biểu diễn bằng hình trụ màu xanh lam đậm).
* Công nghệ: MySQL.
* Vai trò: Lưu trữ tất cả thông tin về giày, người dùng, đơn hàng, v.v. (Chứa thông tin users, shoes...).
* Tương tác: "Cửa hàng bán giày API" là thành phần duy nhất đọc và ghi trực tiếp vào cơ sở dữ liệu.
* Mối quan hệ và Giao thức
* Mũi tên: Biểu thị luồng thông tin hoặc sự phụ thuộc.
* Nhãn trên mũi tên: Mô tả loại thông tin được truyền tải hoặc giao thức sử dụng (HTTP, JSON, RESTful API, Đọc/ghi, Thông tin thanh toán).

1. Lược đồ C3

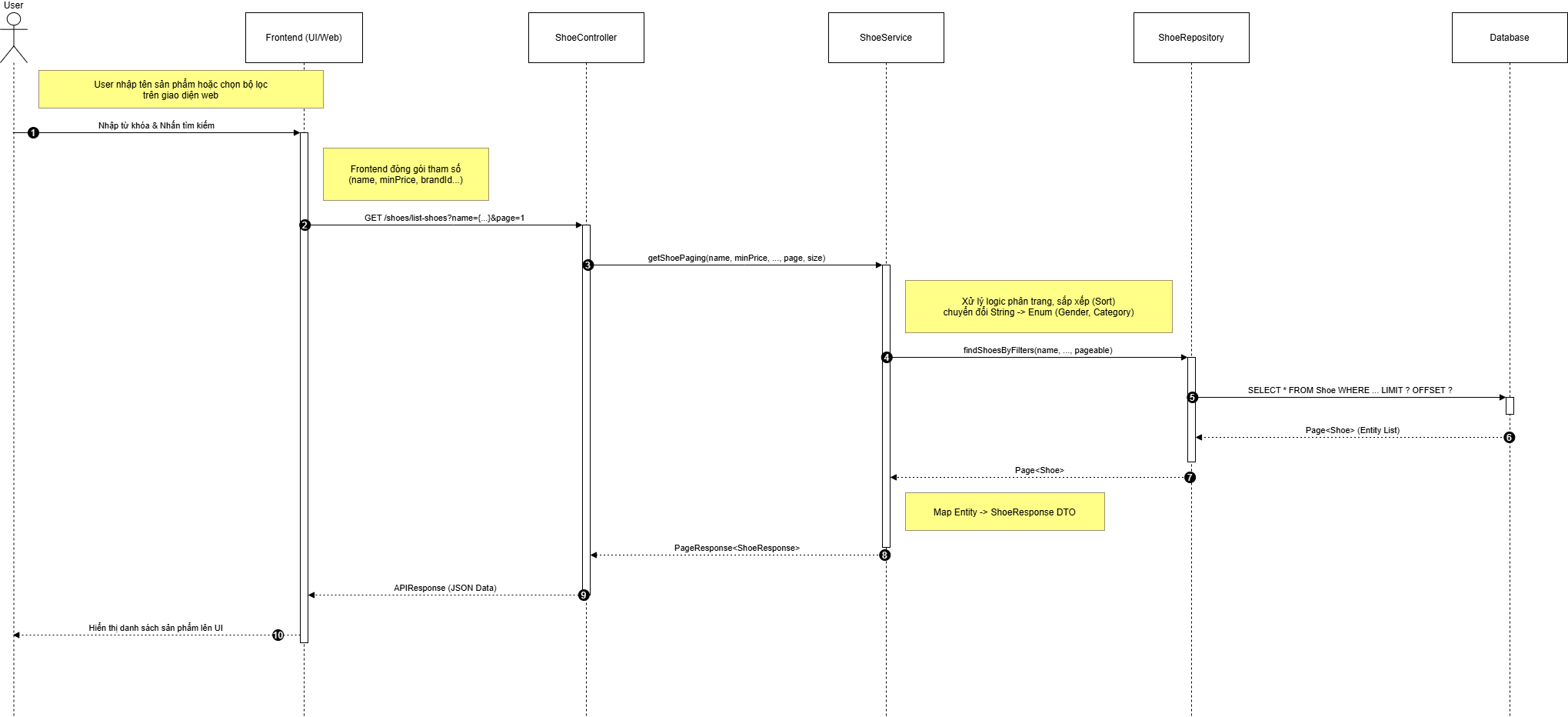


*Hình 3. C3 Model*

* Luồng dữ liệu từ Trình duyệt Web đi vào Ứng dụng API thông qua giao thức HTTP với định dạng dữ liệu JSON
* Bên trong phần API được chia thành các thành phần Controller riêng biệt xử lý từng nghiệp vụ cụ thể, ví dụ như AuthenticationController chịu trách nhiệm về xác thực, phân quyền, đăng nhập và đăng ký tài khoản. ShoeController quản lý các chức năng liên quan đến sản phẩm giày như xem danh sách, chi tiết và tìm kiếm. CartController xử lý các thao tác với giỏ hàng như thêm sản phẩm, cập nhật số lượng và xóa. OrderController quản lý vòng đời của đơn hàng từ lúc khởi tạo, xác nhận đến khi giao hàng. PaymentController xử lý các giao dịch thanh toán, DiscountController quản lý việc tạo và áp dụng các mã giảm giá cho đơn hàng. UserController cung cấp các chức năng quản lý thông tin cá nhân và tài khoản người dùng. BrandController quản lý thông tin về thương hiệu và phân loại sản phẩm
* Các controller này để thực hiện chức năng của mình sẽ gọi với các service tương ứng, các service muốn thao tác với cơ sở dữ liệu sẽ gọi tới các repository tương ứng.

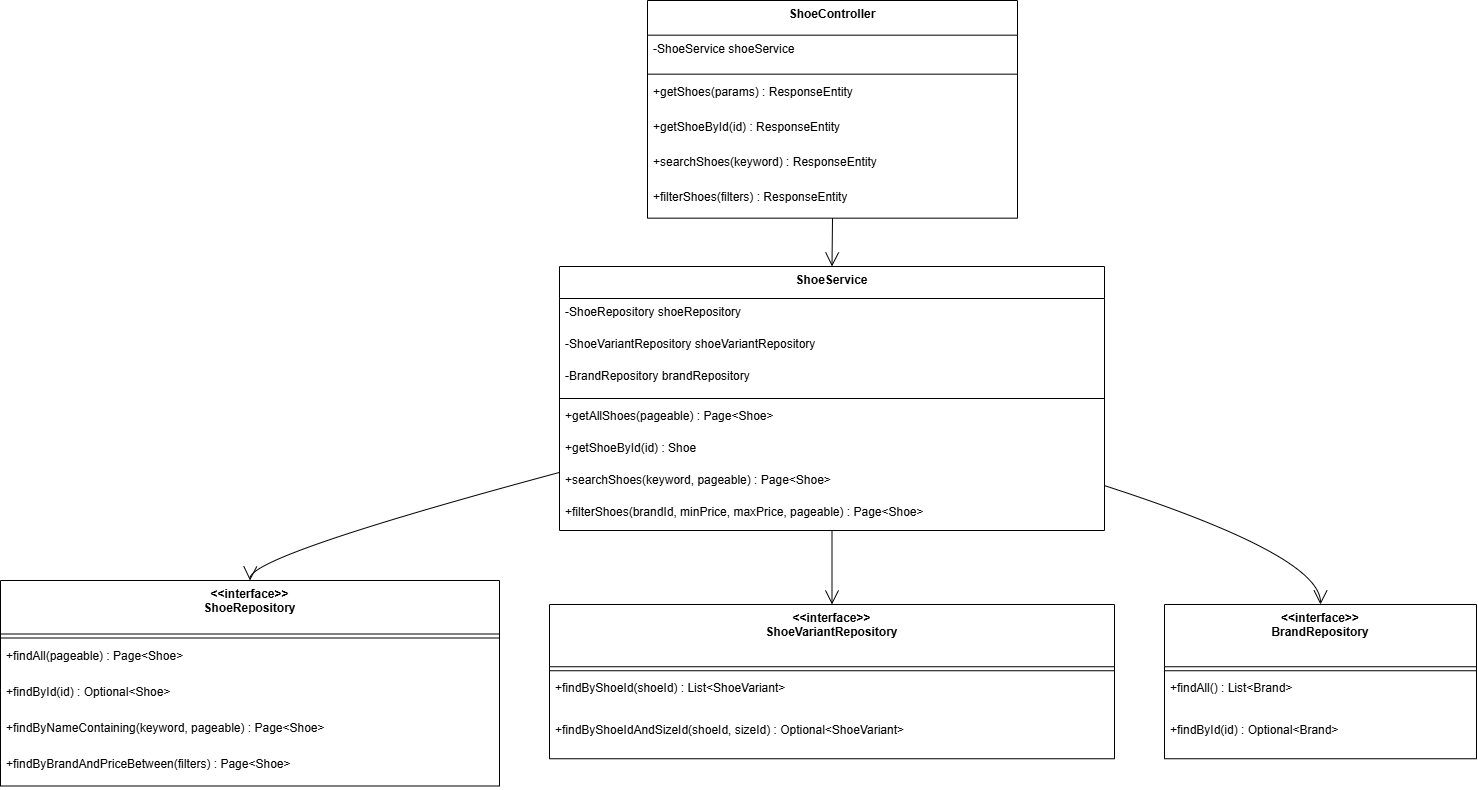
b) Phân tích luồng dữ liệu

Lược đồ Sequence của tìm kiếm sản phẩm



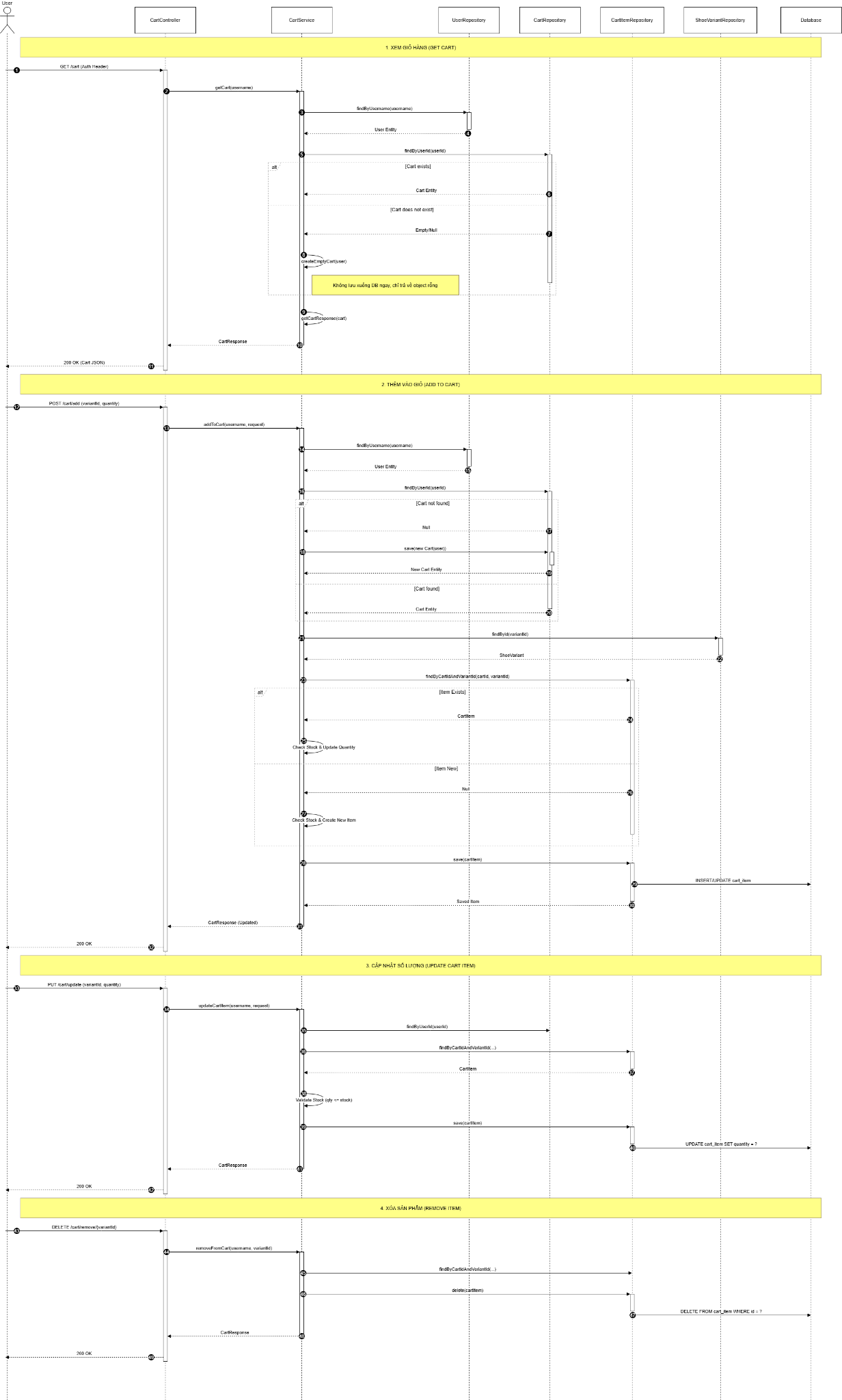
*Hình 14. Sequence Tìm Kiếm Sản Phẩm*

Class diagram của tìm kiếm sản phẩm



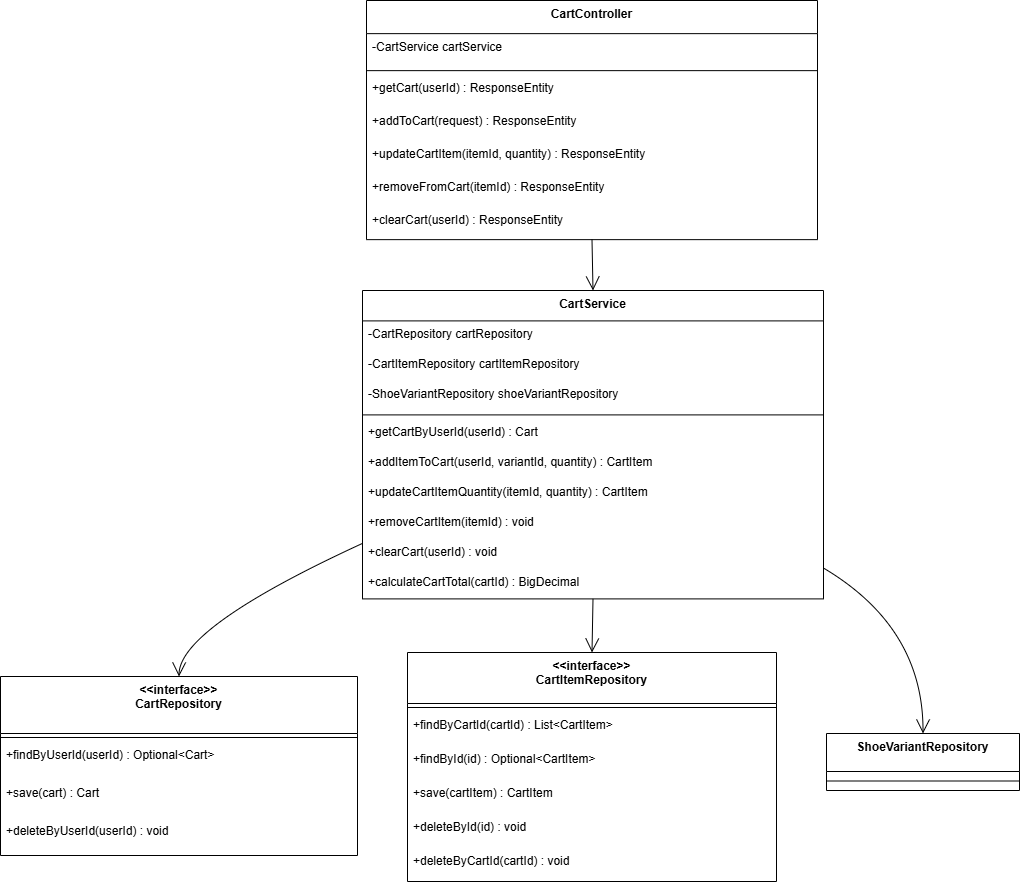
*Hình 15. Class Diagram Tìm Kiếm Sản Phẩm*

Lược đồ mô tả quy trình Khách hàng tìm kiếm sản phẩm bằng từ khóa. Khách hàng nhập từ khóa và nhấn nút tìm kiếm, Giao diện kiểm tra tính hợp lệ của từ khóa. Nếu không hợp lệ, lỗi được hiển thị ngay. Nếu hợp lệ, Giao diện gửi yêu cầu đến ProductController, controller chuyển tiếp cho ProductService xử lý, và service truy vấn ProductRepository. Repository thực hiện câu lệnh SQL xuống Database để lấy danh sách sản phẩm chứa từ khóa. Sau khi nhận kết quả, nếu danh sách rỗng, hệ thống thông báo không tìm thấy sản phẩm; nếu có dữ liệu, ProductController trả danh sách về Giao diện để hiển thị cho Khách hàng.  
  
Lược đồ sequence của chức năng giỏ hàng



*Hình 16. Sequence Giỏ Hàng*

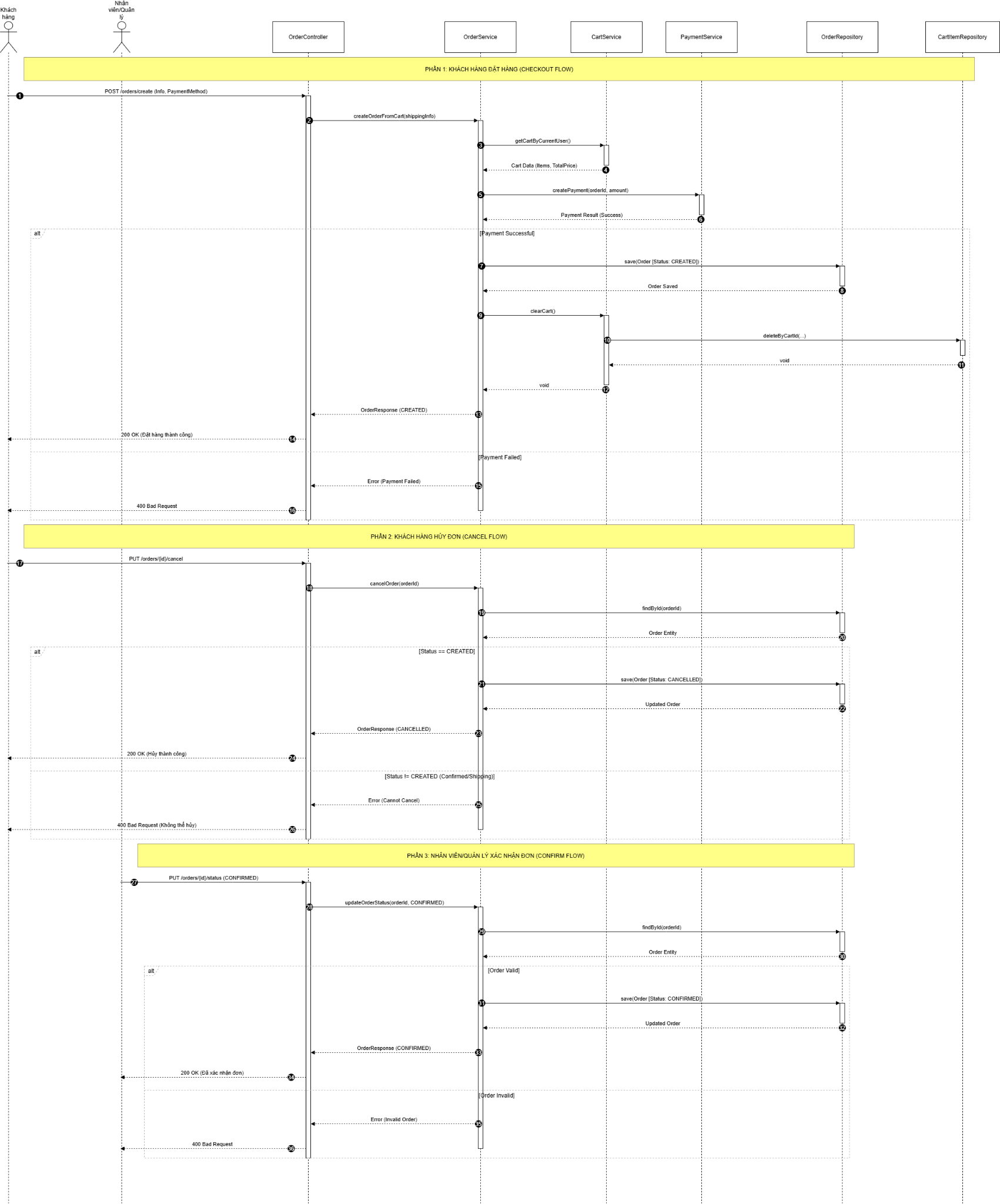
Class diagram của chức năng giỏ hàng



*Hình 17. Class Diagram Giỏ Hàng*

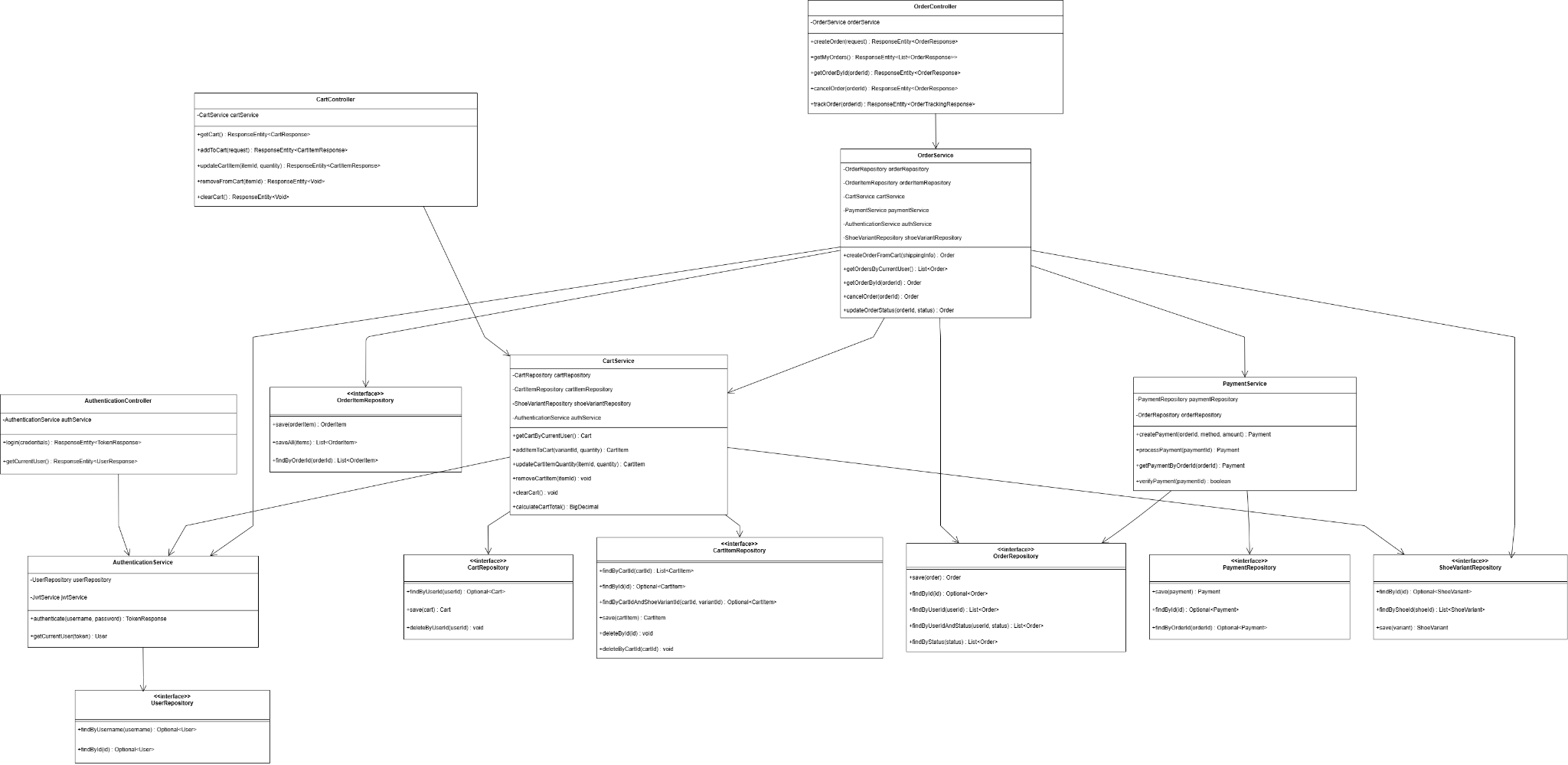
* Lược đồ mô tả quy trình Khách hàng thêm sản phẩm vào giỏ hàng. Khi nhấn “Thêm vào giỏ”, Giao diện gửi yêu cầu addToCart (gồm mã sản phẩm và số lượng) đến CartController, sau đó controller chuyển tiếp cho CartService. Service kiểm tra sản phẩm qua ProductRepository; nếu Database không trả về sản phẩm, hệ thống báo lỗi “Sản phẩm không tồn tại”. Nếu có, CartService lấy giỏ hàng hiện tại của người dùng từ CartRepository; nếu chưa có giỏ, hệ thống tạo giỏ mới. Tiếp theo, service duyệt danh sách sản phẩm trong giỏ: nếu sản phẩm đã tồn tại thì tăng số lượng, nếu chưa thì thêm mới. Sau khi cập nhật, CartRepository lưu dữ liệu xuống Database. Khi lưu thành công, CartController trả phản hồi để Giao diện hiển thị thông báo “Thêm thành công” cho Khách hàng.

Lược đồ sequence cho chức năng đặt hàng



*Hình 18. Sequence Đặt Hàng*

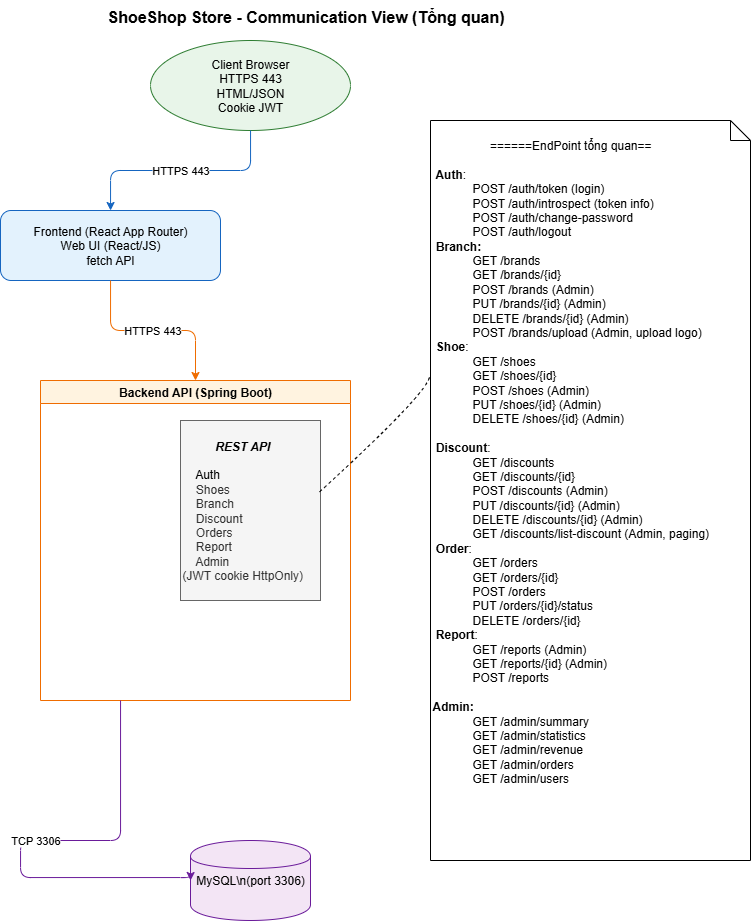
Class diagram cho chức năng đặt hàng



*Hình 19. Class Diagram Đặt Hàng*

* Lược đồ mô tả ba quy trình chính: Khách hàng đặt hàng, Khách hàng hủy đơn và Nhân viên xác nhận đơn. Khi Khách hàng tạo đơn (POST), OrderController gọi OrderService, service lấy dữ liệu giỏ hàng từ CartService và gửi thanh toán qua PaymentService. Nếu thanh toán thành công, OrderService lưu đơn trạng thái CREATED, xóa giỏ hàng và trả về 200 OK; nếu thất bại, trả về 400 Bad Request. Ở quy trình hủy đơn, Khách hàng gửi yêu cầu hủy (PUT cancel). OrderService tìm đơn trong OrderRepository và kiểm tra trạng thái. Nếu đơn đang ở trạng thái CREATED, hệ thống cập nhật sang CANCELLED và trả thành công; nếu trạng thái khác, báo lỗi không thể hủy. Trong quy trình xác nhận đơn, Nhân viên gửi yêu cầu cập nhật trạng thái sang CONFIRMED. OrderService truy vấn đơn hàng và kiểm tra điều kiện hợp lệ (như tồn kho, thanh toán). Nếu hợp lệ, đơn được cập nhật sang CONFIRMED và lưu xuống database; nếu không hợp lệ, hệ thống trả về 400 Bad Request.

### 2.2.3. Phân tích giao tiếp

****

*Hình 10. Communication View*

# **CHƯƠNG 3 – TEST PLAN**

TỔNG QUAN CHƯƠNG 3 – KẾ HOẠCH KIỂM THỬ (TEST PLAN)

Chương 3 trình bày kế hoạch kiểm thử tổng thể cho hệ thống Shoeshop, nhằm xác định rõ phạm vi kiểm thử, đối tượng kiểm thử, cách tiếp cận và chiến lược kiểm thử sẽ áp dụng trong dự án.

Nội dung chương tập trung mô tả phương pháp luận ở mức tổng quan, bao gồm: phạm vi kiểm thử (in-scope/out-of-scope), phương pháp kiểm thử (tĩnh – động, hộp trắng – hộp đen – hộp xám), các mức kiểm thử (Unit Test, Integration Test, System Test, Acceptance Test), tiêu chí chấp nhận, môi trường kiểm thử, nguồn lực và lộ trình triển khai theo Agile.

## **3.1. Tổng quan kế hoạch kiểm thử**

### 3.1.1. Test Scope (In-scope/Out-of-scope)

- Mục tiêu kiểm thử: Xác định phạm vi, chiến lược, môi trường, nguồn lực và deliverables của hoạt động kiểm thử cho hệ thống SuperShoe.

- Mục đích tổng thể: Đảm bảo toàn bộ các nghiệp vụ thương mại điện tử (Product Catalog, Shopping Cart, Order , Discount, User Management, Access Control) hoạt động đúng yêu cầu chức năng và phi chức năng; giảm thiểu rủi ro trước khi release chính thức.

Các chuẩn & công cụ áp dụng:

| Chuẩn/Công cụ | Mục đích |
| --- | --- |
| Selenium | Kiểm thử giao diện tự động |
| JMeter | Kiểm thử tải và hiệu năng |
| Postman | Kiểm thử API và xác minh dữ liệu backend |

− **Functional scope**: Product Catalog, Shopping Cart, Order, Discount, User Management, Access Control.

− **Non-functional scope**: Performance, Security, Usability, Maintainability, Reliability, Compatibility.

## **3.2. Phương pháp kiểm thử**

### **3.2.1 Kiểm thử tĩnh (Static Testing)**

Kỹ thuật kiểm thử được thực hiện mà không cần thực thi mã nguồn chương trình. Quá trình này tập trung vào việc rà soát tài liệu và phân tích cấu trúc mã nguồn.

**Mục tiêu:**

- Phát hiện sớm các lỗi về logic, thiết kế và thiếu sót trong yêu cầu.

- Đảm bảo mã nguồn tuân thủ các quy chuẩn lập trình (coding conventions).

- Giảm thiểu chi phí sửa lỗi trước khi chuyển sang giai đoạn kiểm thử động.

### **3.2.2 Kiểm thử động (Dynamic Testing)**

Quá trình thực thi mã nguồn hệ thống ShoeShop trên môi trường thực tế để tìm kiếm lỗi và xác minh chức năng.

**Phương pháp tiếp cận:** Kiểm thử động được triển khai thông qua sự phối hợp của 3 chiến lược:

1. **Kiểm thử Hộp đen (Black-box Testing):**

- Kiểm tra chức năng dựa trên **User Story** và giao diện người dùng (UI) mà không quan tâm cấu trúc code.

- **Áp dụng cho ShoeShop:**

* + - Thao tác trên Frontend ReactJS: Thêm giỏ hàng, bộ lọc sản phẩm, hiển thị responsive.
    - Luồng nghiệp vụ (End-to-End): Quy trình Checkout, áp dụng mã giảm giá, phân quyền (Customer/Staff/Admin/Manager).

1. **Kiểm thử Hộp trắng (White-box Testing):**

- Kiểm tra cấu trúc nội tại, luồng logic và các nhánh điều kiện trong mã nguồn.

- **Áp dụng cho ShoeShop:**

* + - **Unit Test:** Sử dụng **JUnit 5** để kiểm tra các hàm xử lý logic tại Backend Spring Boot.
    - **Logic nghiệp vụ:** Kiểm tra thuật toán tính tổng tiền, logic giảm giá và bảo mật token JWT.

1. **Kiểm thử Hộp xám (Grey-box Testing):**

- Kết hợp việc kiểm tra chức năng bên ngoài với sự hiểu biết về cấu trúc dữ liệu bên trong.

- **Áp dụng cho ShoeShop:**

* + - **API Testing:** Sử dụng **Postman** để kiểm tra các phương thức HTTP (GET/POST/PUT).
    - **Database:** Xác minh tính toàn vẹn dữ liệu trong MySQL (Ràng buộc khóa ngoại, trừ tồn kho, trạng thái đơn hàng).

**Hình thức thực hiện:**

* **Manual Testing:** Kiểm thử thủ công cho UI và các luồng nghiệp vụ phức tạp.
* **Automation Testing:** Sử dụng Selenium (UI) và JUnit (Backend) để chạy hồi quy.

## **3.3. Hạng mục kiểm thử**

**Yêu cầu chức năng**

**1. Danh mục sản phẩm (Product Catalog)**

– Xác minh rằng hệ thống hiển thị danh sách sản phẩm với đầy đủ thông tin chi tiết: tên, mô tả, giá bán, thương hiệu, kích cỡ và trạng thái tồn kho.

– Xác minh rằng hệ thống cho phép khách hàng thực hiện tìm kiếm sản phẩm theo từ khóa, lọc theo danh mục/thương hiệu và sắp xếp theo giá/tên.

– Xác minh rằng hệ thống hiển thị chính xác hình ảnh và thông tin chi tiết khi người dùng chọn xem một sản phẩm cụ thể.

– Xác minh rằng hệ thống cho phép Quản lý thực hiện các thao tác thêm mới, chỉnh sửa, xóa (hoặc ẩn) sản phẩm và cập nhật tồn kho.

– Xác minh rằng hệ thống hiển thị thông báo lỗi phù hợp khi nhập dữ liệu sản phẩm không hợp lệ (ví dụ: giá âm, thiếu tên).

**2. Giỏ hàng & Thanh toán (Shopping Cart)**

– Xác minh rằng hệ thống cho phép khách hàng thêm sản phẩm vào giỏ hàng từ trang chi tiết sản phẩm.

– Xác minh rằng hệ thống hiển thị đúng danh sách sản phẩm trong giỏ, bao gồm đơn giá, số lượng và tổng tiền tạm tính.

– Xác minh rằng hệ thống tự động cập nhật tổng tiền thanh toán ngay lập tức khi người dùng thay đổi số lượng hoặc xóa sản phẩm khỏi giỏ.

– Xác minh rằng hệ thống cho phép người dùng áp dụng mã giảm giá hợp lệ và từ chối các mã không hợp lệ hoặc hết hạn.

– Xác minh rằng hệ thống lưu trữ trạng thái giỏ hàng chính xác (đồng bộ) khi người dùng đăng nhập lại hoặc tải lại trang.

**3. Quản lý đơn hàng (Order)**

– Xác minh rằng hệ thống cho phép khách hàng hoàn tất đặt hàng (checkout) với đầy đủ thông tin giao nhận và phương thức thanh toán (COD/VNPay).

– Xác minh rằng hệ thống xử lý giao dịch thanh toán trực tuyến chính xác và cập nhật trạng thái đơn hàng tương ứng.

– Xác minh rằng hệ thống cho phép khách hàng xem lại lịch sử đơn hàng và chi tiết trạng thái xử lý của từng đơn.

– Xác minh rằng hệ thống chỉ cho phép khách hàng hủy đơn hàng khi đơn đang ở trạng thái hợp lệ (Created/Confirmed).

– Xác minh rằng hệ thống cho phép Nhân viên/Quản lý cập nhật trạng thái đơn hàng theo đúng quy trình nghiệp vụ (Confirmed → Preparing → Delivery → Delivered).

– Xác minh rằng hệ thống tự động trừ số lượng tồn kho khi đơn hàng được xác nhận và hoàn lại tồn kho nếu đơn hàng bị hủy.

**4. Quản lý mã giảm giá (Discount)**

– Xác minh rằng hệ thống cho phép Quản lý tạo mới mã giảm giá với các tham số: mức giảm (tiền/phần trăm), ngày hiệu lực, giới hạn số lượng.

– Xác minh rằng hệ thống cho phép chỉnh sửa hoặc vô hiệu hóa mã giảm giá đang hoạt động.

– Xác minh rằng hệ thống áp dụng đúng mức giảm giá cho đơn hàng đủ điều kiện và ngăn chặn việc áp dụng cho đơn hàng không đủ điều kiện.

**5. Quản lý người dùng (User Management)**

– Xác minh rằng hệ thống cho phép khách hàng tự cập nhật thông tin hồ sơ cá nhân (Họ tên, địa chỉ, số điện thoại) và thay đổi mật khẩu.

– Xác minh rằng hệ thống cho phép Admin xem danh sách toàn bộ người dùng và tìm kiếm tài khoản theo tiêu chí.

– Xác minh rằng hệ thống cho phép Admin thực hiện khóa (disable) hoặc mở khóa (enable) tài khoản người dùng và ngăn chặn tài khoản bị khóa đăng nhập.

**6. Kiểm soát truy cập (Access Control)**

– Xác minh rằng hệ thống cho phép đăng nhập đối với các tài khoản hợp lệ (Khách hàng, Nhân viên, Quản lý, Admin).

– Xác minh rằng hệ thống cho phép khách hàng đăng ký tài khoản mới và hiển thị thông báo lỗi nếu thông tin không hợp lệ (trùng email, sai định dạng).

– Xác minh rằng hệ thống cho phép đăng xuất an toàn và hủy phiên làm việc hiện tại đối với tất cả người dùng.

– Xác minh rằng hệ thống thực thi phân quyền truy cập đúng, đảm bảo người dùng chỉ truy cập được các trang và API phù hợp với vai trò (Role) của mình.

– Xác minh rằng hệ thống ngăn chặn các truy cập trái phép vào trang quản trị từ tài khoản khách hàng hoặc người dùng chưa đăng nhập.

### **7. Khả năng sử dụng**

* Xác minh rằng hệ thống cung cấp thanh điều hướng rõ ràng, cho phép người dùng truy cập các trang chính như trang chủ, danh sách sản phẩm, giỏ hàng và đơn hàng.
* Xác minh rằng người dùng có thể dễ dàng quay lại trang trước hoặc trang chủ trong quá trình sử dụng.
* Xác minh rằng các nút chức năng chính được hiển thị rõ ràng và dễ thao tác

### **8. Tương thích**

* Xác minh rằng giao diện website hiển thị đúng và đồng nhất trên các trình duyệt phổ biến như Chrome, Firefox, Safari.
* Xác minh rằng hệ thống tự động điều chỉnh giao diện để phù hợp với các kích thước màn hình khác nhau (Desktop, Tablet, Mobile).
* Xác minh rằng các thành phần giao diện không bị vỡ hoặc lệch khi thay đổi kích thước màn hình.
* Xác minh rằng người dùng có thể thao tác đầy đủ các chức năng trên các thiết bị khác nhau.

**9. Giao diện**

* Xác minh rằng các màu sắc, font chữ và kiểu hiển thị đúng theo thiết kế.
* Xác minh rằng logo, hình ảnh, icon được hiển thị đúng vị trí và không bị biến dạng.
* Xác minh rằng các thông báo lỗi, cảnh báo, hoặc thông tin trạng thái hiển thị đúng theo ngữ cảnh nghiệp vụ.

### **10. Hiệu năng**

* Xác minh rằng hệ thống đáp ứng nhanh khi tải bình thường.
* Xác minh rằng hệ thống vẫn hoạt động ổn định khi có nhiều người dùng truy cập đồng thời (load testing).
* Xác minh rằng hệ thống có thể chịu được tải cực đại mà không bị crash (stress testing).

**11.API**

* Xác minh rằng các endpoint API hoạt động đúng chức năng theo thiết kế.
* Xác minh rằng dữ liệu trả về từ API chính xác, đầy đủ và tuân theo định dạng yêu cầu.
* Xác minh rằng quyền truy cập và bảo mật của các API được đảm bảo, người dùng không được phép truy cập trái phép.
* Xác minh rằng các API tương tác đúng với cơ sở dữ liệu, giao diện UI và các endpoint liên quan khác.
* Xác minh rằng các luồng nghiệp vụ chính thông qua API vẫn chạy ổn định sau khi có thay đổi hệ thống.

## 3.4 Hạng mục không được kiểm thử

### 3.4.1. Các chức năng thuộc hệ thống bên thứ ba

Các chức năng xử lý nội bộ của cổng thanh toán VNPay không thuộc phạm vi kiểm thử của đề tài.  
Nhóm không thực hiện kiểm thử các nghiệp vụ, thuật toán xử lý thanh toán, hay cơ chế bảo mật nội bộ do VNPay quản lý.

Phạm vi kiểm thử của nhóm chỉ tập trung vào các luồng tích hợp giữa hệ thống ShoeShop và cổng thanh toán VNPay (sandbox/demo) thông qua API, bao gồm:

* Gửi yêu cầu tạo giao dịch thanh toán từ hệ thống ShoeShop đến VNPay.
* Nhận và xử lý kết quả phản hồi từ VNPay (thành công, thất bại, hủy giao dịch).
* Kiểm tra tính đúng đắn của dữ liệu phản hồi và việc cập nhật trạng thái đơn hàng trong hệ thống ShoeShop.

### 3.4.2. Kiểm thử bảo mật nâng cao

Các hoạt động kiểm thử bảo mật nâng cao, bao gồm kiểm thử xâm nhập (Penetration Testing) và các bài kiểm thử bảo mật chuyên sâu, không nằm trong phạm vi thực hiện của đề tài.

Nhóm không thực hiện đánh giá các lỗ hổng bảo mật ở mức:

* Hệ điều hành và hạ tầng máy chủ.
* Mạng, firewall, và các cấu hình bảo mật hệ thống.
* Các cơ chế bảo mật nội bộ của dịch vụ bên thứ ba.

Nhóm chỉ thực hiện kiểm thử bảo mật ở mức cơ bản, tập trung vào kiểm soát truy cập và phân quyền người dùng.

## 3.5. Chiến lược kiểm thử (Test Strategy)

Dự án được triển khai theo mô hình V-Model trong kiểm thử phần mềm, trong đó các hoạt động kiểm thử được tiến hành song song với từng giai đoạn phát triển, giúp kiểm soát chất lượng xuyên suốt vòng đời phần mềm.  
 Mỗi giai đoạn phát triển được liên kết với một cấp độ kiểm thử tương ứng, bao gồm kiểm thử đơn vị, kiểm thử tích hợp, kiểm thử hệ thống và kiểm thử chấp nhận, nhằm đảm bảo sản phẩm đáp ứng đầy đủ yêu cầu đề ra.

Các Test Case được xây dựng dựa trên yêu cầu chức năng và các Use Case của hệ thống. Đồng thời, dự án áp dụng kiểm thử tự động trong quy trình CI/CD bằng GitHub Actions, giúp tự động hóa các bước Build, Test và Deploy, nâng cao tính ổn định và hiệu quả của quá trình phát triển.

### 3.5.1. Các loại kiểm thử (Test Types)

Bảng dưới đây tóm tắt các loại kiểm thử sẽ được áp dụng trong dự án, mục tiêu và tiêu chí chấp nhận tương ứng.

| **Loại kiểm thử** | **Mục tiêu** | **Nội dung kiểm thử** | **Tiêu chí chấp nhận** |
| --- | --- | --- | --- |
| 1. Kiểm thử chức năng (Functional Testing) | Đảm bảo hệ thống vận hành đúng theo yêu cầu nghiệp vụ và các Use Case (BR1 – BR5) | - Tự động hóa kiểm thử chức năng**- Unit Test** bằng JUnit, Mockito- **Integration** **Test** bằng Spring Boot Test, MockMvc, H2- System Test bằng Selenium | - Tất cả Test Case chức năng Pass- Không tồn tại lỗi nghiêm trọng hoặc lỗi chặn (Blocker/Critical) |
| 2. Kiểm thử giao diện & khả năng sử dụng (UI/UX Testing) | Đánh giá trải nghiệm người dùng, khả năng điều hướng và tính nhất quán của giao diện ReactJS | - Kiểm tra giao diện, bố cục, màu sắc- Kiểm tra điều hướng giữa các màn hình- Kiểm thử cơ bản bằng Selenium trên một số trang- Kiểm thử thủ công trên nhiều trình duyệt và thiết bị | - Giao diện hiển thị đúng bố cục- Hoạt động ổn định trên các trình duyệt và thiết bị phổ biến |
| 3. Kiểm thử API (API Testing) | Đảm bảo giao tiếp giữa Client và Server chính xác, bảo mật và tuân thủ chuẩn RESTful | - Kiểm tra các phương thức HTTP (GET, POST, PUT, DELETE)- Kiểm tra mã trạng thái (200, 400, 401, 403, 500)- Kiểm tra cấu trúc JSON trả về- Kiểm tra xác thực Token/JWT | - API hoạt động đúng logic nghiệp vụ- Trả về mã lỗi phù hợp khi input sai- Token không hợp lệ bị chặn (401/403)- Thời gian phản hồi < 1 giây (mạng nội bộ) |
| 4. Kiểm thử hiệu năng (Performance Testing) | Đảm bảo tốc độ phản hồi, độ trễ và mức sử dụng tài nguyên ổn định Đánh giá khả năng chịu tải khi số lượng người dùng | - Load test: hệ thống hoạt động dưới tải dự kiến- Stress test: kiểm tra tại tải vượt mức để xác định giới hạn  - Load testing và stress testing bằng K6 | - Hệ thống xử lý tốt tải dự kiến (ví dụ: 100 users)- Không bị crash khi quá tải- Khi vượt tải, hệ thống degrade gracefully |
| 5. Kiểm thử đám đông (Crowd Testing) | Phát hiện lỗi, vấn đề tương thích và đánh giá trải nghiệm người dùng thực tế | - Thử nghiệm thủ công trên nhiều thiết bị và trình duyệt- Thu thập phản hồi từ nhiều tester | - Người dùng thực tế thao tác đầy đủ chức năng- Không có lỗi nghiêm trọng- Hệ thống hoạt động ổn định, nhất quán |
| 6. Kiểm thử hồi quy (Regression Testing) | Đảm bảo các chức năng đã ổn định không bị ảnh hưởng sau khi cập nhật hoặc sửa lỗi | - Tự động hóa kiểm thử kết hợp CI/CD và Selenium | - Tỷ lệ Pass Test Case hồi quy ≥ 98%- Pipeline GitHub Actions chạy thành công |

| **Cấp độ** | **Mục đích** | **Kỹ thuật** | **Phương pháp** |
| --- | --- | --- | --- |
| Kiểm thử đơn vị | Kiểm tra tính đúng đắn của các hàm/phương thức | Hộp trắng | Kiểm thử tự động bằng JUnit và Mockito ở môi trường Development |
| Kiểm thử tích hợp | Kiểm tra sự tương tác giữa các module chức năng | Hộp trắng | Kiểm thử tự động bằng Spring Boot Test, MockMvc và H2 ở môi trường Development |
| Kiểm thử hệ thống | Kiểm tra các luồng nghiệp vụ và giao diện | Hộp đen | Kiểm thử tự động bằng Selenium (UI, E2E) và K6 (hiệu năng) trong môi trường Development |
| Kiểm thử chấp nhận | Kiểm tra và nghiệm thu sản phẩm | Hộp đen | Kiểm thử thủ công ở môi trường Production |

### 3.5.2. Ma trận phân bổ kiểm thử (Test Stage Matrix)

Bảng dưới đây xác định giai đoạn kiểm thử mà mỗi loại kiểm thử sẽ được áp dụng.

| **Loại kiểm thử** | **Static** | **Unit** | **Integration** | **System** | **Acceptance** |
| --- | --- | --- | --- | --- | --- |
| **Functional Testing** |  | X | X | X | X |
| **UI/UX & Usability Testing** |  |  |  | X | X |
| **API Testing** |  |  | **X** | **X** |  |
| **Performance Testing** |  |  |  | X | X |
| **Load & Stress Testing** |  |  |  | X |  |

| **Loại** | **Cấp độ kiểm thử** | | | |
| --- | --- | --- | --- | --- |
| **Kiểm thử đơn vị** | **Kiểm thử tích hợp** | **Kiểm thử hệ thống** | **Kiểm thử chấp nhận** |
| Kiểm thử chức năng | x | x | x | x |
| Kiểm giao diện |  |  | x | x |
| Kiểm thử hiệu năng |  |  | x |  |
| Kiểm thử đám đông |  |  | x | x |
| Kiểm thử hồi quy | x | x | x |  |
| Kiểm thử API | x | x |  |  |

## **3.6.** Test Environment

### **3.6.1 Hardware**

| **Hardware** | **Quantity** | **Usage** |
| --- | --- | --- |
| Application Server | 2 | Deploy backend (Spring Boot) |
| Database Server | 1 | MySQL test DB |
| Client Test Devices | 5 | PC cho Tester, Chrome/Edge/ test |

### **3.6.2 Software**

| **Software** | **Version** | **Purpose** |
| --- | --- | --- |
| **Hệ điều hành** | Windows 10,11 (client) | Window cho server & client |
| **Backend** | Spring Boot 3.x | API backend |
| **Frontend** | React + Vite | UI web app |
| **Database** | MySQL 8.0 | Lưu trữ dữ liệu test |
| **CI/CD** | Docker | Tự động build/test/deploy |
| **Automation Test** | Selenium 4.x | UI testing |
| **Performance Test** | K6 | Load/Stress test |
| **Unit Test** | JUnit 5.x | Unit testing cho backend |

### **3.6.3. Nhân lực (Human Resources)**

Phân công trách nhiệm cụ thể cho các thành viên trong đội kiểm thử (QA Team):

| **Thành viên** | **Vai trò & Trách nhiệm chính** |
| --- | --- |
| **Võ Thành Danh** (Test Lead) | - Xây dựng và duyệt Test Plan.  - Phân công task cho các tester.  - Thiết kế & thực thi Test Case cho module **Access Control**. |
| **Đỗ Phú Thành** (Tester) | - Thiết kế & thực thi Test Case cho module **Order Management**.  - Viết báo cáo tổng hợp (Test Summary Report). |
| **Huỳnh Minh Quân** (Tester) | - Chịu trách nhiệm module **Discount Management**.  - Kiểm tra tính logic của mã giảm giá. |
| **Huỳnh Duy Khang** (Tester) | - Chịu trách nhiệm module **Product Catalog** và **Shopping Cart**.    - Kiểm tra hiển thị sản phẩm và tính toán giỏ hàng. |

### 

# CHƯƠNG 4 – THIẾT KẾ KIỂM THỬ (TEST DESIGN)

## 4.1. Giới thiệu

Chương này trình bày quy trình thiết kế và kết quả xây dựng các trường hợp kiểm thử (Test Case) cho hệ thống ShoeShop dựa trên mô hình kiểm thử V-Model.  
Việc thiết kế Test Case được thực hiện dựa trên các tài liệu đã phân tích ở Chương 1 (Tổng quan hệ thống), Chương 2 (Phân tích và thiết kế hệ thống) và Chương 3 (Chiến lược và phạm vi kiểm thử).

Mục tiêu của chương là làm rõ mối liên hệ giữa yêu cầu – thiết kế – kiểm thử, đồng thời chứng minh rằng mọi chức năng của hệ thống ShoeShop đều được xác minh thông qua các mức kiểm thử phù hợp theo V-Model.

## 4.2. Quy trình thiết kế kiểm thử theo V-Model

Quy trình thiết kế kiểm thử của nhóm được xây dựng hoàn toàn dựa trên mô hình V-Model, một mô hình kiểm thử nhấn mạnh sự song hành và đối xứng giữa các giai đoạn phát triển phần mềm và các mức kiểm thử tương ứng.

Trong mô hình này, các hoạt động phân tích và thiết kế ở nhánh trái của chữ V sẽ lần lượt được xác minh bởi các hoạt động kiểm thử ở nhánh phải. Cách tiếp cận này giúp đảm bảo rằng mọi yêu cầu và quyết định thiết kế đều có thể được kiểm chứng thông qua các Test Case cụ thể, từ đó nâng cao chất lượng và độ tin cậy của hệ thống.

### 4.2.1. Phân tích yêu cầu - 1a

Ở giai đoạn đầu tiên, nhóm tiến hành thu thập, phân tích và làm rõ các yêu cầu nghiệp vụ của hệ thống bán giày trực tuyến ShoeShop.  
Mục tiêu của giai đoạn này là đảm bảo các yêu cầu được mô tả đầy đủ, đúng phạm vi, không mâu thuẫn và có thể kiểm thử được.

Dựa trên quá trình phân tích, nhóm xây dựng tài liệu Đặc tả yêu cầu phần mềm (SRS) làm cơ sở cho toàn bộ hoạt động thiết kế và kiểm thử. Hệ thống được xác định gồm 4 nhóm người dùng chính là *Customer (khách hàng), Staff nhân viên, Manager quản lý, Admin quản trị viên*, với các module chức năng tiêu biểu như: Quản lý sản phẩm, Giỏ hàng, Đơn hàng, Thanh toán.

Kết quả của giai đoạn phân tích yêu cầu là tiền đề để xây dựng các trường hợp kiểm thử chấp nhận, trong đó mỗi yêu cầu nghiệp vụ được chuyển hóa thành tiêu chí nghiệm thu nhằm xác minh rằng hệ thống đáp ứng đúng mong đợi của người dùng.

### 4.2.2. Thiết kế hệ thống - 2a

Sau khi các yêu cầu được xác định rõ ràng, nhóm tiến hành thiết kế hệ thống ở mức tổng thể. Giai đoạn này tập trung mô tả các chức năng chính, các luồng nghiệp vụ, sơ đồ use case và mối quan hệ tương tác giữa người dùng với hệ thống ShoeShop.

Thông tin thu được từ bước thiết kế hệ thống được sử dụng để xây dựng các kịch bản kiểm thử hệ thống. Mỗi use case đại diện cho một hoặc nhiều luồng nghiệp vụ sẽ trở thành đối tượng kiểm thử, nhằm đảm bảo hệ thống vận hành đúng theo các kịch bản nghiệp vụ đã được thiết kế.

### 4.2.3. Thiết kế kiến trúc - 3a

Trong giai đoạn thiết kế kiến trúc, nhóm áp dụng C4 Model (Context – Container – Component – Code) để mô tả và phân rã hệ thống ShoeShop từ mức độ tổng quan đến chi tiết.

Việc thiết kế kiến trúc giúp xác định rõ:

* Sự phân tách trách nhiệm giữa frontend, backend và database.
* Các điểm giao tiếp giữa các module thông qua API.
* Luồng dữ liệu và cách thức trao đổi thông tin giữa các thành phần.

Kết quả của giai đoạn này là cơ sở quan trọng cho việc xây dựng kiểm thử tích hợp, trong đó các điểm giao tiếp giữa các thành phần được phân tích và chuyển hóa thành các trường hợp kiểm thử nhằm đánh giá tính chính xác và nhất quán của dữ liệu trao đổi.

### 4.2.4. Thiết kế module - 4a

Ở bước thiết kế module, nhóm tiến hành phân tích chi tiết từng module chức năng của hệ thống. Các sơ đồ lớp (Class Diagram) được sử dụng để mô tả cấu trúc dữ liệu, thuộc tính, phương thức và mối quan hệ giữa các lớp trong từng module.

Những tài liệu này là cơ sở để xác định các thành phần trọng yếu cần được kiểm thử ở mức đơn vị. Việc thiết kế rõ ràng module giúp nhóm xây dựng các trường hợp kiểm thử đơn vị nhằm phát hiện lỗi sớm ngay từ cấp độ thấp nhất trước khi các module được tích hợp với nhau.

### 4.2.5. Kiểm thử đơn vị - 1b

Dựa trên thiết kế module, nhóm tiến hành xây dựng và thực thi kiểm thử đơn vị nhằm đánh giá tính đúng đắn của từng phương thức, lớp hoặc hàm độc lập.

Mục tiêu của kiểm thử đơn vị là đảm bảo các thành phần nhỏ nhất của hệ thống hoạt động chính xác theo thiết kế, từ đó hạn chế tối đa lỗi phát sinh ở các giai đoạn tích hợp và kiểm thử mức cao hơn.

Các ca kiểm thử đơn vị được triển khai tự động bằng JUnit kết hợp với Mockito để mô phỏng các đối tượng phụ thuộc. Việc sử dụng mock giúp cô lập logic kiểm thử khỏi các thành phần bên ngoài như database hoặc service khác, đảm bảo tính chính xác và khả năng lặp lại của kiểm thử.

Toàn bộ kiểm thử đơn vị được tích hợp vào quy trình CI/CD thông qua GitHub Actions, cho phép các bài test được thực thi tự động mỗi khi có thay đổi mã nguồn, qua đó hỗ trợ phát hiện lỗi sớm và duy trì chất lượng phần mềm liên tục.

| **Tên hàm test** | **Mô tả** | **Dữ liệu nhập** | **Kết quả mong đợi** | **Kết quả chạy thực tế** | **Trạng thái** |
| --- | --- | --- | --- | --- | --- |
| addToCart\_existingItemExceedStock\_shouldThrowOutOfStock | Thêm item đã tồn tại vượt stock | username='unit\_test\_user', variantId='variant-001', oldQty=2, addQty=9, stock=10 | Ném AppException với ErrorCode.OUT\_OF\_STOCK | Exception: Product is out of stock (OUT\_OF\_STOCK) | PASS |
| addToCart\_exceedStock\_shouldThrowAppException | Thêm vào giỏ vượt quá stock | username='unit\_test\_user', variantId='variant-001', quantity=15, stock=10 | Ném AppException với ErrorCode.OUT\_OF\_STOCK | Exception: Product is out of stock (OUT\_OF\_STOCK) | PASS |
| addToCart\_userNotFound\_shouldThrowUserNotExisted | Thêm vào giỏ khi user không tồn tại | username='unknown\_user' | Ném AppException với ErrorCode.USER\_NOT\_EXISTED | Exception: User does not exist (USER\_NOT\_EXISTED) | PASS |
| addToCart\_variantNotFound\_shouldThrowVariantNotFound | Thêm vào giỏ khi variant không tồn tại | username='unit\_test\_user', variantId='non\_existent\_variant', quantity=2 | Ném AppException với ErrorCode.VARIANT\_NOT\_FOUND | Exception: Variant Not Found (VARIANT\_NOT\_FOUND) | PASS |
| addToCart\_existingItem\_shouldUpdateQuantity | Thêm item đã tồn tại trong giỏ | username='unit\_test\_user', variantId='variant-001', oldQty=2, addQty=3, stock=10 | Cập nhật quantity từ 2 lên 5 | Updated existing item. New Quantity: 5 | PASS |
| addToCart\_newItem\_shouldSaveCartItem | Thêm item mới vào giỏ | username='unit\_test\_user', variantId='variant-001', quantity=2, stock=10 | Lưu CartItem mới với quantity=2 | Saved new CartItem. Quantity: 2, Total Price: 400.0 | PASS |

### 4.2.6. Kiểm thử tích hợp - 2b

Sau khi các module đã được xác nhận ổn định thông qua kiểm thử đơn vị, nhóm tiến hành kiểm thử tích hợp để đánh giá sự tương tác giữa các thành phần trong hệ thống ShoeShop.

Kiểm thử tích hợp tập trung xác minh:

* Luồng xử lý giữa Controller – Service – Repository.
* Việc truyền nhận dữ liệu qua các API.
* Sự tương thích giữa backend và cơ sở dữ liệu.

Nhóm sử dụng Spring Boot Test kết hợp với MockMvc để kiểm thử các API trong môi trường mô phỏng gần với thực tế. Cơ sở dữ liệu được thay thế bằng H2 Database (in-memory) nhằm đảm bảo kiểm thử có tính độc lập, dễ tái lập và không ảnh hưởng đến dữ liệu thật.

Các Test Case tích hợp được thiết kế dựa trên các điểm giao tiếp đã xác định ở giai đoạn thiết kế kiến trúc, đảm bảo bao phủ đầy đủ các luồng tương tác quan trọng của hệ thống.

### 4.2.7. Kiểm thử hệ thống - 3b

Ở mức cao hơn, nhóm thực hiện kiểm thử hệ thống nhằm đánh giá toàn bộ hệ thống ShoeShop như một thể thống nhất. Kiểm thử hệ thống được thực hiện chủ yếu theo phương pháp thủ công, dựa trên bộ Test Case đã được xây dựng từ các use case và luồng nghiệp vụ chính.

Các hoạt động kiểm thử bao gồm:

* Kiểm thử chức năng: kiểm tra các luồng nghiệp vụ từ đầu đến cuối (end-to-end) như đăng nhập, mua hàng, thanh toán…
* Kiểm thử giao diện: đảm bảo giao diện trực quan, nhất quán và đáp ứng tiêu chuẩn hiển thị.
* Kiểm thử bảo mật cơ bản: xác minh luồng đăng nhập, phân quyền và kiểm soát truy cập.

Kết quả của giai đoạn này phản ánh chất lượng tổng thể của hệ thống trước khi chuyển sang kiểm thử chấp nhận.

### 4.2.8. Kiểm thử chấp nhận - 4b

Kiểm thử chấp nhận là giai đoạn cuối cùng trong quy trình kiểm thử, nhằm đánh giá hệ thống dưới góc nhìn người dùng cuối. Ở giai đoạn này, các tiêu chí nghiệm thu được đối chiếu trực tiếp với tài liệu SRS để xác nhận rằng hệ thống ShoeShop đáp ứng đầy đủ các yêu cầu đã đề ra.

Sau khi hoàn tất kiểm thử nội bộ, hệ thống được bàn giao cho Product Owner thực hiện kiểm thử chấp nhận. Product Owner đóng vai trò đại diện người dùng, mô phỏng các tình huống sử dụng thực tế và đưa ra đánh giá về mức độ đáp ứng nghiệp vụ, tính tiện dụng và độ ổn định của hệ thống.

Kết quả kiểm thử chấp nhận là cơ sở quan trọng để đưa ra quyết định cuối cùng về việc triển khai hệ thống ShoeShop vào môi trường vận hành.

.

## 4.3. Kỹ thuật thiết kế kiểm thử

### 4.3.1. Kĩ thuật test: Hộp trắng (White Box)

#### Kiểm thử thêm vào giỏ hàng

public CartResponse addToCart(String username, AddToCartRequest request) {

log.info("Adding item to cart for user: {}", username);

// 1. Lấy user

User user = userRepository.findByUsername(username)

.orElseThrow(() -> new AppException(ErrorCode.USER\_NOT\_EXISTED));

// 2. Lấy hoặc tạo cart cho user

Cart cart = cartRepository.findByUserId(user.getId())

.orElseGet(() -> {

Cart newCart = Cart.builder()

.user(user)

.build();

return cartRepository.save(newCart);

});

// 3. Lấy shoe variant

ShoeVariant variant = shoeVariantRepository.findById(request.getVariantId())

.orElseThrow(() -> new AppException(ErrorCode.VARIANT\_NOT\_FOUND));

// 4. Kiểm tra item đã tồn tại trong cart chưa

CartItem cartItem = cartItemRepository

.findByCartIdAndVariantId(cart.getId(), variant.getId())

.orElse(null);

if (cartItem != null) {

// 5a. Cập nhật số lượng nếu item đã tồn tại

int newQuantity = cartItem.getQuantity() + request.getQuantity();

if (newQuantity > variant.getStockQuantity()) {

throw new AppException(ErrorCode.OUT\_OF\_STOCK);

}

cartItem.setQuantity(newQuantity);

} else {

// 5b. Tạo item mới nếu chưa tồn tại

if (request.getQuantity() > variant.getStockQuantity()) {

throw new AppException(ErrorCode.OUT\_OF\_STOCK);

}

cartItem = CartItem.builder()

.cart(cart)

.variant(variant)

.quantity(request.getQuantity())

.build();

cart.addItem(cartItem);

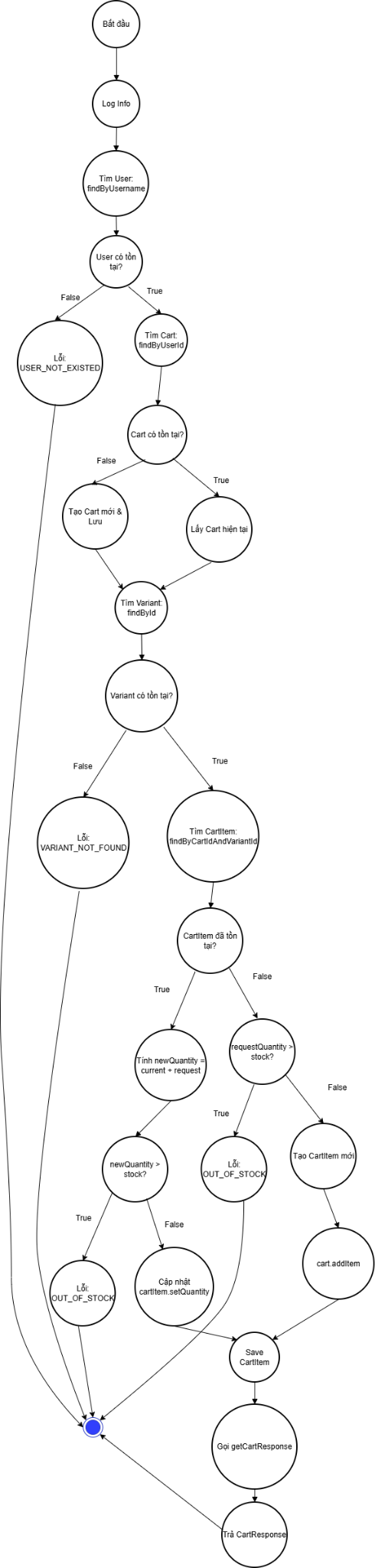
}

// 6. Lưu cart item

cartItemRepository.save(cartItem);

// 7. Trả về response

return getCartResponse(cart);

}

*Hình 21. WorkFlow thêm hàng vào giỏ hàng*

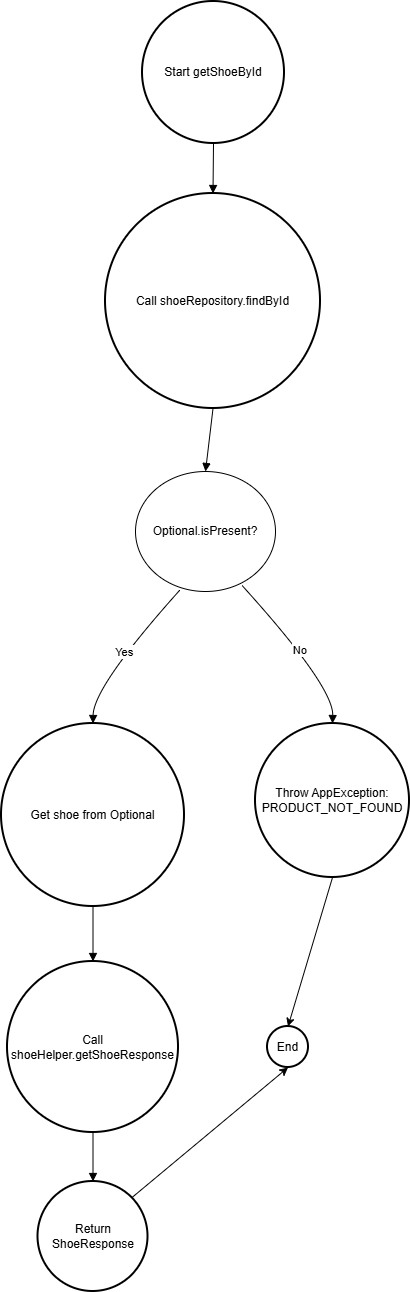
Từ đồ thị trên, có 6 đường độc lập chính:

* User not exists
* Variant not exists
* New item, quantity ≤ stock
* New item, quantity > stock
* Existing item, newQuantity ≤ stock
* Existing item, newQuantity > stock

| **Tên hàm test** | **Mô tả** | **Dữ liệu nhập** | **Kết quả mong đợi** | **Kết quả chạy thực tế** | **Trạng thái** |
| --- | --- | --- | --- | --- | --- |
| addToCart\_existingItemExceedStock\_shouldThrowOutOfStock | Thêm item đã tồn tại vượt stock | username='unit\_test\_user', variantId='variant-001', oldQty=2, addQty=9, stock=10 | Ném AppException với ErrorCode.OUT\_OF\_STOCK | Exception: Product is out of stock (OUT\_OF\_STOCK) | PASS |
| addToCart\_exceedStock\_shouldThrowAppException | Thêm vào giỏ vượt quá stock | username='unit\_test\_user', variantId='variant-001', quantity=15, stock=10 | Ném AppException với ErrorCode.OUT\_OF\_STOCK | Exception: Product is out of stock (OUT\_OF\_STOCK) | PASS |
| addToCart\_userNotFound\_shouldThrowUserNotExisted | Thêm vào giỏ khi user không tồn tại | username='unknown\_user' | Ném AppException với ErrorCode.USER\_NOT\_EXISTED | Exception: User does not exist (USER\_NOT\_EXISTED) | PASS |
| addToCart\_variantNotFound\_shouldThrowVariantNotFound | Thêm vào giỏ khi variant không tồn tại | username='unit\_test\_user', variantId='non\_existent\_variant', quantity=2 | Ném AppException với ErrorCode.VARIANT\_NOT\_FOUND | Exception: Variant Not Found (VARIANT\_NOT\_FOUND) | PASS |
| addToCart\_existingItem\_shouldUpdateQuantity | Thêm item đã tồn tại trong giỏ | username='unit\_test\_user', variantId='variant-001', oldQty=2, addQty=3, stock=10 | Cập nhật quantity từ 2 lên 5 | Updated existing item. New Quantity: 5 | PASS |
| addToCart\_newItem\_shouldSaveCartItem | Thêm item mới vào giỏ | username='unit\_test\_user', variantId='variant-001', quantity=2, stock=10 | Lưu CartItem mới với quantity=2 | Saved new CartItem. Quantity: 2, Total Price: 400.0 | PASS |

1. Xem chi tiết sản phẩm

public ShoeResponse getShoeById(int id) { Shoe shoe = shoeRepository.findById(id) .orElseThrow(() -> new AppException(ErrorCode.PRODUCT\_NOT\_FOUND)); return shoeHelper.getShoeResponse(shoe); }



*Hình 22. WorkFlow xem chi tiết sản phẩm*

Đồ thị có 2 đường độc lập

1. Tìm ra sản phẩm và lấy thông tin chi tiết
2. Không tìm ra sản phẩm

| **Tên hàm test** | **Mô tả** | **Dữ liệu nhập** | **Kết quả mong đợi** | **Kết quả chạy thực tế** | **Trạng thái** |
| --- | --- | --- | --- | --- | --- |
| getShoeById\_WhenNotExists\_ShouldThrowException | Lấy giày theo ID không tồn tại | shoeId=999 | Ném AppException với ErrorCode.PRODUCT\_NOT\_FOUND | Exception: Product Not Found (PRODUCT\_NOT\_FOUND) | PASS |
| getShoeById\_WhenExists\_ShouldReturnShoe | Lấy giày theo ID tồn tại | shoeId=1, name='Air Jordan' | Trả về ShoeResponse với tên 'Air Jordan' | Retrieved Shoe Name: Air Jordan | PASS |

### 4.3.2. Kĩ thuật test: Hộp đen (Black Box)

Kiểm thử hộp đen được sử dụng để kiểm tra chức năng của hệ thống dựa trên yêu cầu nghiệp vụ, mà không cần quan tâm đến cấu trúc bên trong của phần mềm. Trong dự án ShoeShop, phương pháp này được áp dụng cho kiểm thử hệ thống và kiểm thử chấp nhận, vì mục tiêu là xác minh phần mềm hoạt động đúng như kỳ vọng của người dùng cuối và đáp ứng các tiêu chí nghiệm thu.

Việc sử dụng hộp đen giúp nhóm mô phỏng các tình huống thực tế và kiểm tra toàn bộ luồng nghiệp vụ từ đầu đến cuối, phù hợp với thực hiện thủ công hoặc tự động hóa ở mức giao diện.

Áp dụng vào Test Cases Cart Management

Nhóm đã áp dụng 2 kỹ thuật hộp đen chính cho module Cart Management:

*1. KỸ THUẬT BẢNG QUYẾT ĐỊNH (Decision Table)*

Quy tắc nghiệp vụ được xác định:

1. User phải đăng nhập để thêm sản phẩm
2. Sản phẩm phải còn tồn kho
3. Số lượng phải hợp lệ (không vượt quá tồn kho)
4. Mã giảm giá phải hợp lệ và đủ điều kiện

| **Quy tắc** | **R1(Thêm hợp lệ)** | **R2(Hết hàng)** | **R3(Số lượng không hợp lệ)** | **R4(Giảm giá hợp lệ)** |
| --- | --- | --- | --- | --- |
| **Điều kiện** |  |  |  |  |
| Đã đăng nhập | ✓ | ✓ | ✓ | ✓ |
| Còn tồn kho | ✓ | ✗ | ✓ | ✓ |
| Số lượng hợp lệ | ✓ | – | ✗ | ✓ |
| Mã giảm giá hợp lệ | – | – | – | ✓ |
| Đủ điều kiện giảm giá | – | – | – | ✓ |
| **Kết quả** |  |  |  |  |
| Hệ thống xử lý | Thêm thành công | Báo lỗi hết hàng | Báo lỗi số lượng | Áp dụng giảm giá |

* TC\_DT\_001: Thêm sản phẩm hợp lệ → Thành công
* TC\_DT\_002: Thêm sản phẩm hết hàng → Lỗi
* TC\_DT\_003: Số lượng vượt tồn kho → Lỗi
* TC\_DT\_004: Áp dụng giảm giá hợp lệ → Thành công

*2. KỸ THUẬT SƠ ĐỒ CHUYỂN TRẠNG THÁI (State Transition)*

* Các trạng thái của giỏ hàng:
* Empty Cart (S1): Giỏ trống
* Has Items (S2): Có sản phẩm
* Ready for Checkout (S3): Sẵn sàng thanh toán
* Error State (S4): Trạng thái lỗi
* 

*Hình 23. State Transition*

Bảng chuyển trạng thái (State Transition Table)

| **Trạng thái hiện tại** | **Sự kiện** | **Hành động** | **Trạng thái kế tiếp** |
| --- | --- | --- | --- |
| Empty Cart | addItem (hợp lệ) | Thêm sản phẩm, tính lại tổng tiền | Has Items |
| Empty Cart | addItem (không hợp lệ) | Hiển thị thông báo lỗi | Error State |
| Has Items | updateQty | Cập nhật số lượng, tính lại tổng tiền | Has Items |
| Has Items | removeItem | Xóa sản phẩm, tính lại tổng tiền | Empty Cart |
| Has Items | checkout | Chuẩn bị dữ liệu thanh toán | Ready for Checkout |
| Error State | retry | Reset trạng thái giỏ hàng | Empty Cart |

Danh sách Test Case từ bảng chuyển trạng thái

| **Test Case ID** | **Luồng kiểm thử** | **Mô tả** |
| --- | --- | --- |
| TC\_ST\_001 | Empty Cart → addItem → Has Items | Thêm sản phẩm hợp lệ vào giỏ hàng trống |
| TC\_ST\_002 | Has Items → updateQty → Has Items | Cập nhật số lượng khi giỏ hàng đã có sản phẩm |
| TC\_ST\_003 | Has Items → removeItem → Empty Cart | Xóa sản phẩm khỏi giỏ hàng |
| TC\_ST\_004 | Has Items → checkout → Ready for Checkout | Thực hiện chuyển sang bước thanh toán |

## **4.4. Phương pháp thiết kế kiểm thử**

### **4.4.1. Kiểm thử thủ công**

Trong dự án ShoeShop, kiểm thử thủ công được áp dụng chủ yếu ở giai đoạn kiểm thử chấp nhận, nhằm đánh giá hệ thống dưới góc nhìn của người dùng cuối. Phương pháp này cho phép kiểm tra trực tiếp các luồng nghiệp vụ chính và xác nhận mức độ đáp ứng của hệ thống so với các yêu cầu đã được đặc tả trong tài liệu SRS.

Các ca kiểm thử thủ công được thực hiện dựa trên Test Case và Review Checklist đã được xây dựng trước đó. Trong quá trình kiểm thử, kết quả thực tế được ghi nhận, so sánh và đối chiếu với kết quả mong đợi để làm cơ sở đánh giá mức độ hoàn thành của từng yêu cầu, trước khi tiến hành nghiệm thu hệ thống.

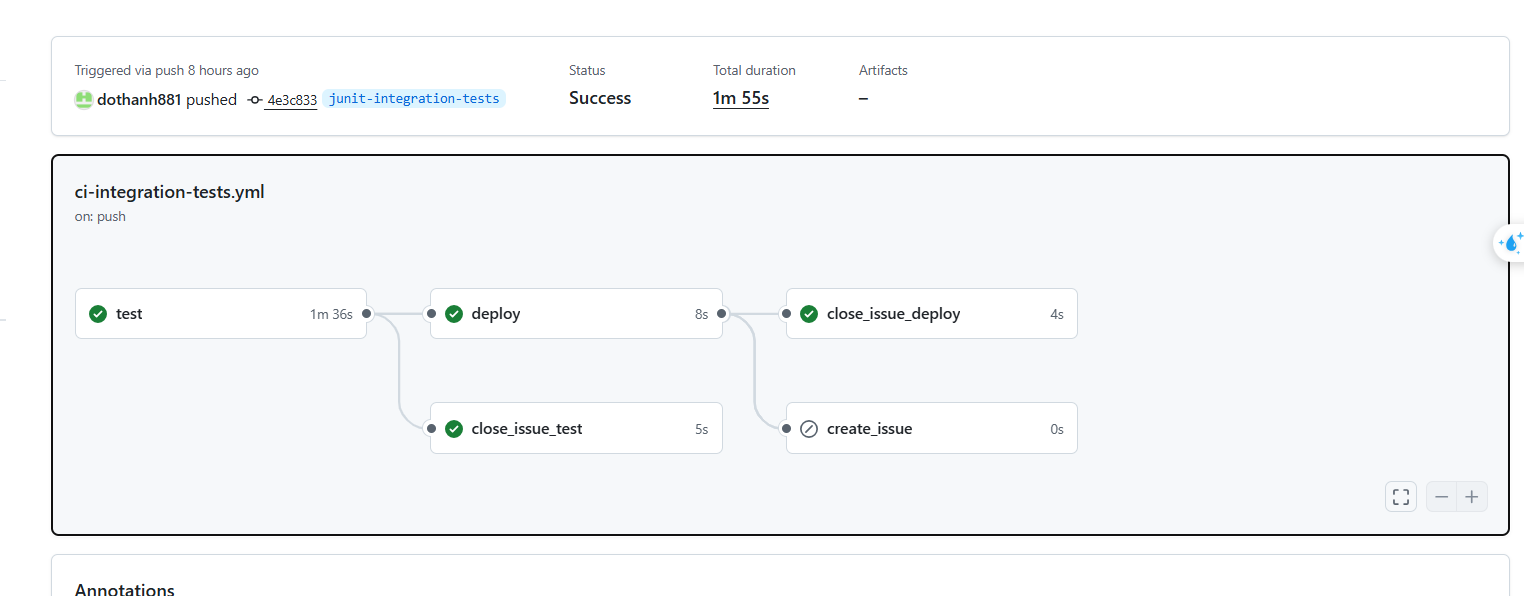
### **4.4.2. Kiểm thử tự động**

Bên cạnh kiểm thử thủ công, dự án ShoeShop áp dụng kiểm thử tự động thông qua việc tích hợp quy trình CI/CD sử dụng GitHub Actions, nhằm tự động hóa các bước build, kiểm thử và triển khai hệ thống.

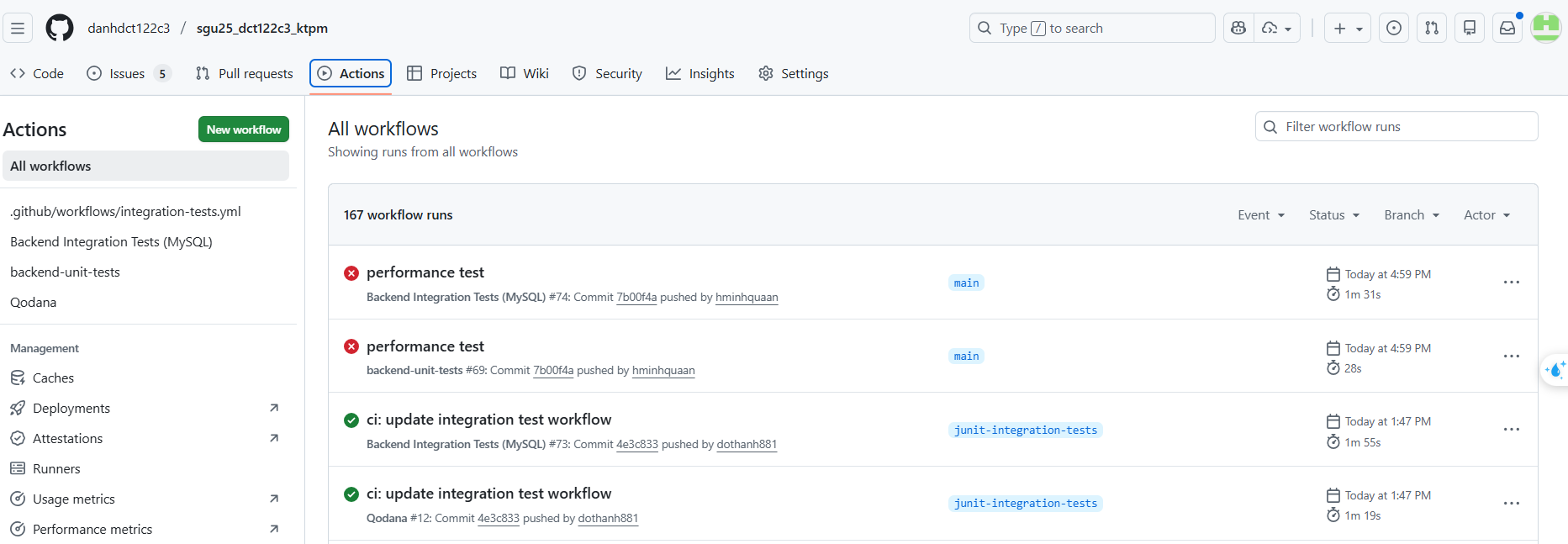
Kiểm thử tự động được sử dụng cho các hình thức kiểm thử hộp trắng, bao gồm kiểm thử đơn vị, kiểm thử tích hợp và một phần kiểm thử hệ thống. Mỗi khi có thay đổi mã nguồn được đẩy lên kho lưu trữ GitHub, pipeline CI/CD sẽ tự động được kích hoạt và thực hiện job kiểm thử. Job này tiến hành build dự án và chạy toàn bộ các bộ kiểm thử tự động đã được thiết kế.

* Trong trường hợp job kiểm thử thất bại, pipeline sẽ tự động tạo Issue trên GitHub để ghi nhận lỗi phát sinh, hỗ trợ nhóm phát triển theo dõi và xử lý kịp thời.
* Trong trường hợp job kiểm thử thành công, pipeline sẽ tự động cập nhật trạng thái và đóng các Issue liên quan đến lỗi kiểm thử trước đó, đảm bảo thông tin về chất lượng hệ thống luôn được phản ánh chính xác.

Khi mã nguồn được hợp nhất vào nhánh chính (Main), pipeline CI/CD sẽ tiếp tục kích hoạt job triển khai, thực hiện kết nối đến nền tảng Render để tiến hành triển khai lại hệ thống ShoeShop trên môi trường vận hành.



*Hình 24. CI/CD Pipeline Integration Test*



*Hình 135. Danh sách các lần chạy Git Actions*

## **4.5. Áp dụng GenAI trong việc sinh Test Case**

Trong đồ án này, Generative AI (GenAI) được áp dụng như một công cụ hỗ trợ trong giai đoạn thiết kế test case. GenAI không thay thế vai trò của kiểm thử viên (tester) mà đóng vai trò hỗ trợ phân tích yêu cầu, sinh các test scenario và test case ban đầu. Các kết quả sinh ra từ GenAI đều được tester rà soát, chỉnh sửa và phê duyệt trước khi sử dụng chính thức trong quá trình kiểm thử.

Việc áp dụng GenAI giúp:

* Giảm thời gian thiết kế test case thủ công
* Mở rộng phạm vi bao phủ kiểm thử
* Hạn chế bỏ sót các trường hợp biên và trường hợp lỗi

### 4.5.1. Đầu vào GenAi

Đầu vào của GenAI bao gồm các tài liệu nghiệp vụ và kỹ thuật của hệ thống, cụ thể:

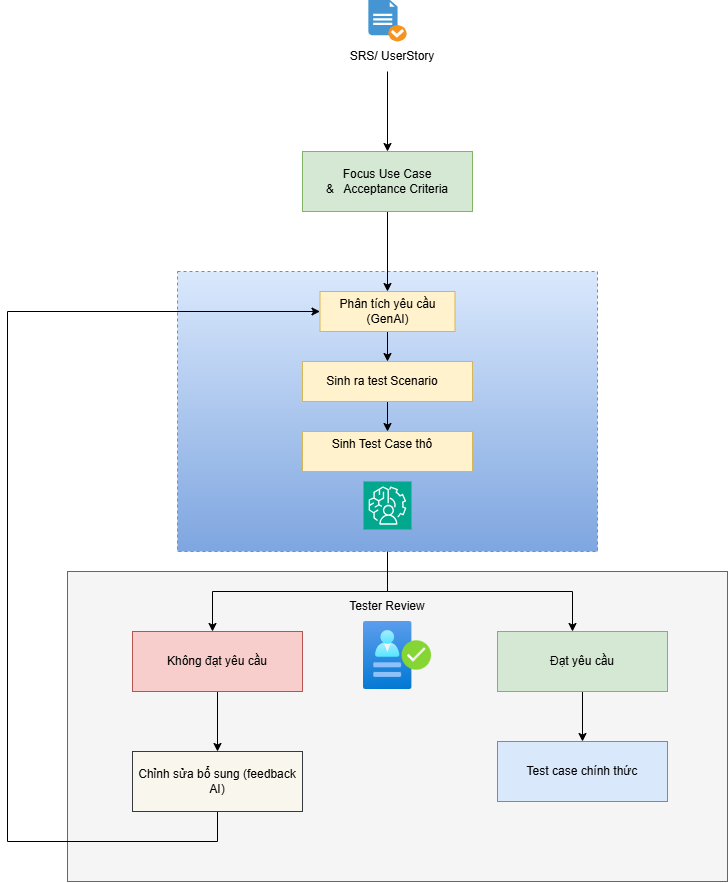
* Tài liệu yêu cầu hệ thống (Business Requirement)
* Tài liệu đặc tả yêu cầu phần mềm (SRS)
* User Story và Use Case

Các tài liệu này cung cấp thông tin về:

* Chức năng hệ thống
* Luồng nghiệp vụ chính và luồng phụ
* Các ràng buộc và điều kiện nghiệp vụ

Những tài liệu này được cung cấp cho GenAI thông qua prompt, đóng vai trò là ngữ cảnh (context) để GenAI hiểu đúng phạm vi và mục tiêu kiểm thử.

### 4.5.2. Giai đoạn xử lý của GenAI



*Hình 26. Quy trình sinh Test Case Gen AI*

Trong giai đoạn xử lý, Generative AI (GenAI) thực hiện một chuỗi các bước nhằm hỗ trợ kiểm thử viên trong việc thiết kế kiểm thử. Trước hết, GenAI tiến hành phân tích ngữ nghĩa yêu cầu, trong đó nội dung của các tài liệu như SRS, user story và use case được phân tích để xác định rõ mục tiêu chức năng của hệ thống, các nghiệp vụ cần đáp ứng cũng như những điều kiện ràng buộc liên quan. Quá trình phân tích này giúp GenAI hiểu được bối cảnh nghiệp vụ, phạm vi chức năng và các yếu tố ảnh hưởng trực tiếp đến hoạt động của hệ thống.

Tiếp theo, dựa trên kết quả phân tích yêu cầu và các use case đã xác định, GenAI thực hiện sinh các test scenario. Các test scenario được tạo ra nhằm bao phủ đầy đủ các luồng nghiệp vụ quan trọng của hệ thống, bao gồm luồng chính (main flow), các luồng thay thế (alternate flow) và các luồng lỗi hoặc trường hợp ngoại lệ (exception flow). Việc sinh test scenario theo nhiều luồng khác nhau giúp mở rộng phạm vi kiểm thử, hạn chế nguy cơ bỏ sót các tình huống có thể xảy ra trong quá trình vận hành thực tế của hệ thống.

Từ các test scenario đã được sinh ra, GenAI tiếp tục sinh các test case thô (draft test case). Các test case này được tạo ở mức đề xuất ban đầu và bao gồm nhiều loại trường hợp kiểm thử khác nhau, chẳng hạn như các trường hợp kiểm thử cơ bản, các trường hợp biên và các trường hợp lỗi. Tuy nhiên, các test case thô này chưa được sử dụng trực tiếp trong quá trình thực thi kiểm thử. Thay vào đó, chúng đóng vai trò là đầu vào tham khảo để kiểm thử viên rà soát, chỉnh sửa và hoàn thiện trước khi trở thành các test case chính thức của hệ thống.

### 4.5.3. Hoàn thiện test case

Sau khi GenAI sinh test case thô, kiểm thử viên thực hiện các công việc sau:

* Rà soát nội dung test case
* Chỉnh sửa lại các bước kiểm thử cho phù hợp với hệ thống thực tế
* Bổ sung dữ liệu test và điều kiện tiên quyết
* Loại bỏ các test case trùng lặp hoặc không phù hợp

Các test case sau khi được hoàn thiện sẽ trở thành test case chính thức, được sử dụng trong quá trình kiểm thử thủ công hoặc kiểm thử tự động.

### 4.5.4. Phương pháp Prompt Engineering áp dụng trong sinh Test Case

Trong đồ án này, nhóm áp dụng một số phương pháp Prompt Engineering nhằm nâng cao chất lượng kết quả sinh test case từ Generative AI (GenAI). Trước hết, phương pháp Role-based Prompting được sử dụng để xác định rõ vai trò của GenAI trong từng bước xử lý, chẳng hạn như đóng vai trò là kiểm thử viên (Tester) hoặc kỹ sư đảm bảo chất lượng (QA Engineer). Việc xác định vai trò cụ thể giúp định hướng cách GenAI phân tích yêu cầu và sinh nội dung phù hợp với mục tiêu kiểm thử.

Bên cạnh đó, nhóm áp dụng phương pháp Context-based Prompting bằng cách cung cấp đầy đủ ngữ cảnh cho GenAI thông qua các tài liệu như SRS, user story và Business Requirement. Việc cung cấp ngữ cảnh chi tiết giúp GenAI hiểu đúng phạm vi nghiệp vụ và chức năng của hệ thống, từ đó hạn chế các kết quả sinh ra mang tính chung chung hoặc không phù hợp với yêu cầu thực tế.

Ngoài ra, phương pháp Step-by-step Prompting được sử dụng để yêu cầu GenAI sinh nội dung theo từng bước rõ ràng. Cụ thể, GenAI được hướng dẫn sinh test scenario dựa trên use case, sau đó tiếp tục sinh test case chi tiết từ các test scenario đã được tạo. Cách tiếp cận theo từng bước này giúp kiểm soát tốt hơn quá trình sinh test, đồng thời đảm bảo tính nhất quán giữa các artefact kiểm thử.

Cuối cùng, nhóm áp dụng phương pháp Constraint-based Prompting bằng cách đặt ra các ràng buộc cụ thể trong prompt, bao gồm yêu cầu về cấu trúc test case, phạm vi sinh test và việc không giả định các hành vi của hệ thống ngoài những nội dung đã được mô tả trong yêu cầu. Các ràng buộc này giúp đảm bảo các test case sinh ra có cấu trúc rõ ràng, bám sát yêu cầu nghiệp vụ và phù hợp để kiểm thử viên tiếp tục rà soát, chỉnh sửa và hoàn thiện.

### 4.5.5. Prompt mẫu sinh Test Case với GenAI

**Prompt 1 – Sinh Test Scenario từ Use Case (kèm theo tài liệu doc)**

Bạn hãy đóng vai trò là một kiểm thử viên phần mềm.

Dựa trên use case sau, hãy sinh các test scenario.

Yêu cầu:

- Bao phủ luồng chính, luồng phụ và luồng lỗi

- Không sinh test case chi tiết

- Phù hợp với hệ thống ShoeShop

Use Case:

Khách hàng đặt hàng sản phẩm giày trực tuyến

**Prompt 2 – Sinh Test Case từ Test Scenario**

Bạn hãy đóng vai trò là QA Engineer.

Chuyển test scenario dưới đây thành các test case chi tiết.

Yêu cầu:

- Sử dụng cấu trúc: Điều kiện tiên quyết – Các bước thực hiện – Kết quả mong đợi

- Không giả định các hành vi chưa được mô tả

- Mỗi test case độc lập

Test Scenario:

Khách hàng đặt hàng với sản phẩm đã hết hàng

# **CHƯƠNG 5. BÁO CÁO KIỂM THỬ**

## **5.1. Tổng quan về quá trình kiểm thử**

Sau khi hoàn tất giai đoạn thiết kế kiểm thử ở Chương 4, nhóm tiến hành thực hiện kiểm thử cho hệ thống ShoeShop theo đúng Test Plan đã xây dựng.Bao gồm đầy đủ các mức kiểm thử: kiểm thử đơn vị, kiểm thử tích hợp, kiểm thử hệ thống .

Hoạt động kiểm thử tập trung vào toàn bộ các chức năng nghiệp vụ chính của hệ thống ShoeShop. Các Test Case được thực hiện một cách có hệ thống, chi tiết và kết quả được ghi nhận đầy đủ trong các tài liệu kiểm thử nhằm phục vụ cho việc đánh giá chất lượng và bàn giao hệ thống.

## **5.2. Báo cáo trường hợp kiểm thử**

### **5.2.1. Giới thiệu**

Các Test Case trong dự án ShoeShop được chia thành hai nhóm:

•Positive Test: kiểm tra các hành vi hợp lệ và luồng nghiệp vụ đúng của hệ thống

•Negative Test: kiểm tra khả năng xử lý lỗi, ngoại lệ và các điều kiện không hợp lệ

| **Chức năng** | **Số TC** | **Kết quả** |
| --- | --- | --- |
| Kiểm soát truy cập | 16 | 16 Pass |
| Sản phẩm | 13 | 12 Pass - 1 fail |
| Giỏ hàng | 23 | 21 Pass - 2 Fail |
| Đơn hàng | 21 | 21 Pass |

### **5.2.2. Phạm vi bao phủ**

Nhóm đã xây dựng **73** Test Case bao phủ hầu hết các chức năng nghiệp vụ chính của hệ thống ShoeShop. Do phạm vi đồ án tập trung vào kiểm thử chức năng, các yêu cầu phi chức năng chưa được đưa vào phạm vi kiểm thử

### **5.2.3. Kết quả thực thi**

Tổng cộng  **73 test case** đã được thực thi với kết quả:

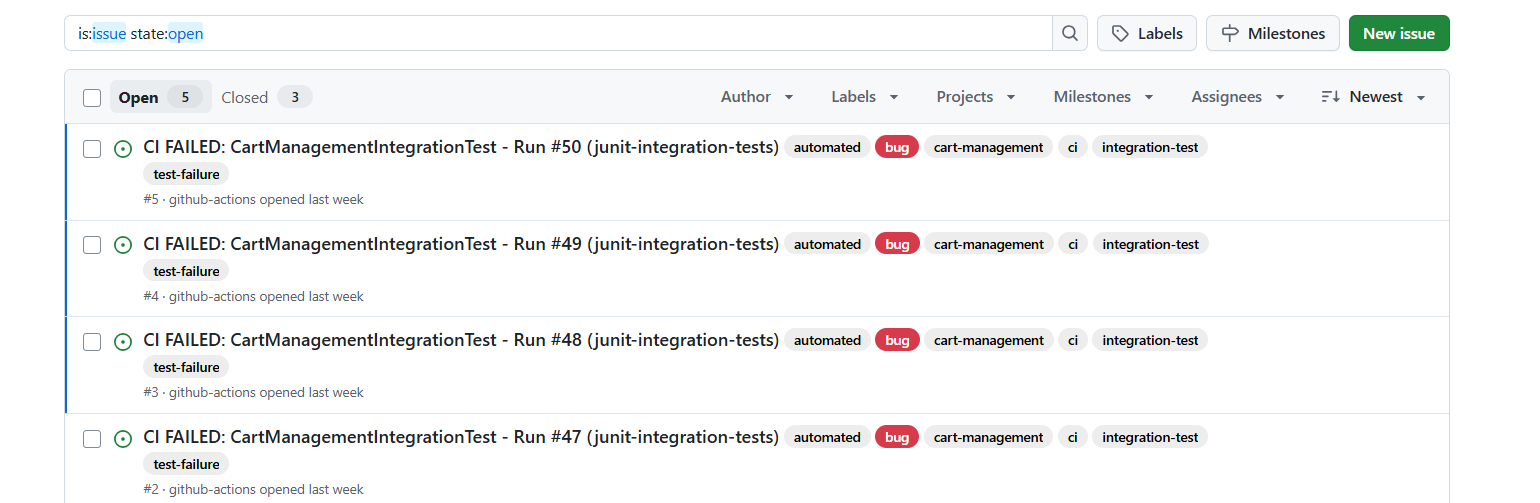
* **70 Test Case Passed (95.9%)**
* **3 Test Case Failed (4.1%)**
* **0 Test Case Not Run**

Kết quả cho thấy hệ thống ShoeShop đạt mức độ ổn định cao trước khi chuyển sang giai đoạn kiểm thử chấp nhận.

## **5.3. Báo cáo lỗi**

Trong quá trình kiểm thử thủ công, nhóm phát hiện **3 lỗi**, được phân loại theo mức độ nghiêm trọng như sau:

* Lỗi số lượng bị gấp đôi sau khi bấm dấu cộng trong trang giỏ hàng (TC\_CART\_007) (medium)
* Lỗi không xét vượt tồn kho trong trang giỏ hàng (TC\_CART\_008) (medium)
* Lỗi không hiển thị số lượng tổng sản phẩm mà chỉ hiển thị theo size trong trang quản lý giày (TC\_PRODUCT\_009) (low)



*Hình 27. Issues Git Actions*