**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN**

**ĐỀ TÀI : Kiểm thử chất lượng hệ thống**

**thương mại điện tử cho nền tảng bán lẻ**

**giày trực tuyến EverShop.**

Giảng viên hướng dẫn: TS. Đỗ Như Tài

Học phần: Kiểm thử phần mềm

Sinh viên thực hiện:

Châu Gia Anh  3122411002

Dương Lê Khánh 3122411093

Đào Thị Thanh Tâm 3122411182

**TP Hồ Chí Minh, ngày 13 tháng 10 năm 2025**

**MỤC LỤC**

[CHƯƠNG 1: GIỚI THIỆU 1](#_Toc2670)

[1. Tổng quan về đề tài 1](#_Toc9233)

[1.1. Bối cảnh 1](#_Toc7233)

[1.2. Phạm vi đề tài 1](#_Toc149)

[1.3. Quy trình Agile và CI/CD 2](#_Toc22623)

[1.3.1. Agile 2](#_Toc16485)

[1.3.2. CI/CD 2](#_Toc275)

[2. Phân tích yêu cầu 3](#_Toc16373)

[2.1. Khảo sát hiện trạng 3](#_Toc32504)

[2.2. Bối cảnh kinh doanh 3](#_Toc5027)

[2.2.1. Sơ đồ các tác nhân tham gia hệ thống 5](#_Toc21112)

[2.2.2. Sơ đồ ngữ cảnh hệ thống 5](#_Toc2475)

[2.4. Công nghệ sử dụng 6](#_Toc2668)

[2.5. Yêu cầu nghiệp vụ 6](#_Toc20681)

[2.5.1. Yêu cầu chức năng 6](#_Toc12485)

[2.5.2. Yêu cầu phi chức năng 7](#_Toc4293)

[CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG VÀ KIẾN TRÚC HỆ THỐNG 9](#_Toc25948)

[1. Phân tích thiết kế hệ thống 9](#_Toc27772)

[1.1. Sơ đồ quy trình nghiệp vụ 9](#_Toc24669)

[1.1.1. Sơ đồ quy trình nghiệp vụ quản lý sản phẩm 9](#_Toc24874)

[1.1.2. Sơ đồ quy trình nghiệp vụ mua hàng 10](#_Toc8422)

[1.1.3. Sơ đồ quy trình nghiệp vụ Quản lý đơn hàng 12](#_Toc21754)

[1.1.4. Sơ đồ quy trình nghiệp vụ Quản lý phân quyền truy cập 13](#_Toc29141)

[1.2. Use Case 14](#_Toc30509)

[1.2.1. Quản lý sản phẩm 14](#_Toc32222)

[1.2.2. Quản lý mua hàng 15](#_Toc26192)

[1.2.3. Quản lý đơn hàng 16](#_Toc13654)

[1.2.4. Quản lý phân quyền truy cập 17](#_Toc30049)

[1.3. User story 17](#_Toc7003)

[1.4. Dữ liệu 19](#_Toc28365)

[1.4.1. Conceptual ERD 19](#_Toc32067)

[1.4.2. Logical ERD 20](#_Toc27822)

[1.4.3. Physical ERD 21](#_Toc28278)

[1.5. Từ điển dữ liệu 25](#_Toc22454)

[2. Kiến trúc hệ thống 26](#_Toc29724)

[2.1. Sơ đồ tổng quan hệ thống 26](#_Toc27300)

[2.2. Kiến trúc C4 27](#_Toc26446)

[2.2.1. Level 1: Context Diagram 27](#_Toc17979)

[2.2.2. Level 2: Container Diagram 28](#_Toc2467)

[2.2.3. Level 3: Component Diagram 29](#_Toc16912)

[2.2.4. Level 4: Code-level Diagram 30](#_Toc26448)

[2.3. Communication View 31](#_Toc18602)

[2.4. Deployment View 32](#_Toc15165)

[CHƯƠNG 3: KẾ HOẠCH KIỂM THỬ 34](#_Toc16845)

[1. Giới thiệu 34](#_Toc27829)

[1.1. Mục tiêu 34](#_Toc25571)

[1.2. Phạm vi áp dụng 34](#_Toc15258)

[2. Các mục kiểm thử 34](#_Toc19933)

[2.1. Kiểm thử chức năng 34](#_Toc5963)

[2.2. Kiểm thử phi chức năng 36](#_Toc5388)

[2.3. Không trong phạm vi kiểm thử 38](#_Toc15768)

[3. Tiêu chí kiểm thử chấp nhận 38](#_Toc3768)

[3.1. Độ bao phủ kiểm thử 38](#_Toc27257)

[3.2. Số lượng trường hợp kiểm thử 38](#_Toc7046)

[3.3. Số lượng lỗi 38](#_Toc26213)

[3.4. Độ bao phủ mã nguồn 38](#_Toc10980)

[4. Chiến lược kiểm thử 39](#_Toc28797)

[4.1. Phương pháp kiểm thử 39](#_Toc7967)

[4.2. Các giai đoạn kiểm thử 39](#_Toc16070)

[4.3. Các loại kiểm thử 40](#_Toc8953)

[5. Môi trường và nguồn lực 41](#_Toc1673)

[5.1. Môi trường kiểm thử 41](#_Toc7251)

[5.2. Nguồn nhân lực 42](#_Toc1720)

[5.3. Milestones 43](#_Toc15534)

[6. Tài liệu bàn giao 44](#_Toc4619)

[CHƯƠNG 4: THIẾT KẾ KIỂM THỬ 45](#_Toc26666)

[1. Tổng quan 45](#_Toc28508)

[1.1. Mục tiêu thiết kế kiểm thử 45](#_Toc22663)

[1.2. Kiểm thử đơn vị 47](#_Toc3702)

[1.2.1. Định nghĩa và Đặc điểm 47](#_Toc3910)

[1.2.2. Cấu trúc phân tầng trong Unit Test của EverShop 47](#_Toc47)

[1.2.3. Chiến lược giả lập (Mocking Strategy) 48](#_Toc473)

[1.2.4. Lợi ích của việc áp dụng Unit Test trong dự án 48](#_Toc13403)

[1.2.5. Áp dụng unit test trong evershop 49](#_Toc6214)

[1.3. Kiểm thử tích hợp 51](#_Toc21601)

[1.3.1. Cấu trúc các thành phần chính 52](#_Toc6633)

[1.3.2. Quy trình vận hành và Luồng dữ liệu (Sequence Flow) 53](#_Toc1915)

[2. Phân tích khung nhìn V-Model 54](#_Toc16694)

[3. Phân tích khung nhìn Agile CI/CD 55](#_Toc5604)

[3.1. Quy trình vận hành CI/CD chi tiết 55](#_Toc13999)

[3.1.1. Giai đoạn Tích hợp liên tục (Continuous Integration - CI) 55](#_Toc19071)

[3.1.2. Giai đoạn Triển khai liên tục (Continuous Deployment - CD) 56](#_Toc29803)

[3.2. Lợi ích trong thiết kế và thực thi kiểm thử 57](#_Toc11971)

[4. Phương pháp kiểm thử 57](#_Toc24281)

[4.1. Kiểm thử Tĩnh 57](#_Toc23273)

[4.2. Kiểm thử Động 58](#_Toc1197)

[5. Phân tích các kỹ thuật nâng cao 59](#_Toc9397)

[5.1 Manual test / Auto test 59](#_Toc18195)

[5.2 Gen AI trong kiểm thử 60](#_Toc2379)

[5.2.1 Các hướng ứng dụng trọng tâm 61](#_Toc29852)

[5.2.2 Lợi ích kỳ vọng 61](#_Toc21234)

[5.2.3 Lộ trình triển khai chi tiết (Roadmap) 62](#_Toc29830)

[5.3. Tự động hóa quy trình kiểm thử 62](#_Toc16822)

[5.3.1 Cấu trúc hệ thống tự động hóa (Automation Framework) 62](#_Toc19788)

[5.3.2 Quy trình vận hành tự động trong CI/CD 63](#_Toc30277)

[5.3.3 Lợi ích vượt trội của tự động hóa 63](#_Toc26782)

[5.3.4 Thách thức và Giải pháp 63](#_Toc6262)

[5.3.5. Áp dụng tự động hóa cho đồ án 64](#_Toc18483)

[CHƯƠNG 5: BÁO CÁO KIỂM THỬ 67](#_Toc16226)

[1. Tổng quan quá trình kiểm thử 67](#_Toc14865)

[2. Báo cáo kết quả kiểm thử 67](#_Toc3836)

[2.1. Tổng Quan Quá Trình Kiểm Thử 67](#_Toc20549)

[2.2. Báo Cáo Chi Tiết Kiểm Thử 69](#_Toc24443)

[2.2.1 Cypress API Testing 69](#_Toc16157)

[2.2.2 Jest Unit & Integration Testing 70](#_Toc9500)

[2.3 Báo Cáo Độ Phủ Code (Code Coverage) 71](#_Toc7773)

**DANH MỤC BẢNG**

[Bảng 1.2.5. 1. Bảng yêu cầu chức năng 14](#_Toc217179869)

[Bảng 1.2.5. 2. Bảng yêu cầu phi chức năng 15](#_Toc217179870)

[Bảng 3.2. 1. Bảng giai đoạn kiểm thử 42](#_Toc217181073)

[Bảng 3.2. 2. Bảng các loại kiểm thử được áp dụng 42](#_Toc217181074)

[Bảng 4.1. 1. Bảng thiết kế kiểm thử theo module 53](#_Toc217243513)

[Bảng 4.5.1. 1. Bảng phân bổ tỷ lệ kiểm thử 66](#_Toc217189810)

[Bảng 5.2.1. 1. Bảng tóm tắt kết quả kiểm thử 73](#_Toc217190188)

[Bảng 5.2.1. 2. Bảng phân bố kết quả test case theo Module 74](#_Toc217190189)

[Bảng 5.2.1. 3. Bảng đối chiếu kết quả kiểm thử giữa các công cụ 75](#_Toc217190190)

[Bảng 5.2.3. 1. Bảng độ phủ code 77](#_Toc217190616)

**DANH MỤC HÌNH ẢNH**

[Hình 2.1.1. 1. Sơ dồ quy trình nghiệp vụ quản lý sản phẩm 15](#_Toc217180142)

[Hình 2.1.1. 2. Sơ đồ quy trình nghiệp vụ mua hàng 17](#_Toc217180143)

[Hình 2.1.1. 3. Sơ đồ quy trình nghiệp vụ quản lý đơn hàng 19](#_Toc217180144)

[Hình 2.1.1. 4. Sơ đồ quy trình nghiệp vụ quản lý phân quyền truy cập 20](#_Toc217180145)

[Hình 2.1.2. 1. Use case quản lý sản phẩm 21](#_Toc217180336)

[Hình 2.1.2. 2. Use case quản lý mua hàng 22](#_Toc217180337)

[Hình 2.1.2. 3. Use case quản lý đơn hàng 24](#_Toc217180338)

[Hình 2.1.2. 4. Use case quản lý phân quyền truy cập 25](#_Toc217180339)

[Hình 2.1.3. 1. Conceptual ERD 26](#_Toc217180523)

[Hình 2.1.3. 2. Logical ERD 27](#_Toc217180524)

[Hình 2.1.3. 3. Physical Product-Catalog 28](#_Toc217180525)

[Hình 2.1.3. 4. Physical Cart - Order 30](#_Toc217180526)

[Hình 2.1.3. 5. Physical Customer 31](#_Toc217180527)

[Hình 2.2.1. 1. Sơ đồ tổng quan hệ thống 33](#_Toc217180845)

[Hình 2.2.2. 1. Level 1: Context Diagram 34](#_Toc217180850)

[Hình 2.2.2. 2. Level 2: Container Diagram 35](#_Toc217180851)

[Hình 2.2.2. 3. Level 3: Component Diagram 36](#_Toc217180852)

[Hình 2.2.2. 4. Code-level Diagram 37](#_Toc217180853)

[Hình 2.2.3. 1. Communication View 39](#_Toc217180906)

[Hình 2.2.4. 1. Deployment View 40](#_Toc217180950)

[Hình 4.1.2. 1. Hình mô phỏng cách hoạt động của Unit Test trong evershop 55](#_Toc217189484)

[Hình 4.1.3. 1. Hình mô phỏng cách hoạt động của Integration Test trong evershop 58](#_Toc217189608)

[Hình 4.1.3. 2. Hình ảnh quy trình vận hành 59](#_Toc217189609)

[Hình 4.2. 1. Hình khung nhìn V-Model 60](#_Toc217189619)

[Hình 4.3.1. 1. Hình giai đoạn tích hợp liên tục (CI) 62](#_Toc217189659)

[Hình 4.3.1. 2. Hình giai đoạn triển khia liên tục (CD) 62](#_Toc217189660)

[Bảng 5.2.2. 1. Bảng kết quả dùng Cypress test API 76](#_Toc217190440)

[Bảng 5.2.2. 2. Bảng kết quả dùng Jest cho Unit test và Integration test 77](#_Toc217190441)

# 

# CHƯƠNG 1: GIỚI THIỆU

## **1. Tổng quan về đề tài**

### **1.1. Bối cảnh**

Trong những năm gần đây, sự phát triển mạnh mẽ của thương mại điện tử đã thúc đẩy việc xây dựng và vận hành các website bán hàng trực tuyến nhằm đáp ứng nhu cầu mua sắm ngày càng cao của người tiêu dùng. Trong đó, các website bán giày là một mô hình điển hình, với nhiều chức năng nghiệp vụ quan trọng như tìm kiếm sản phẩm, xem thông tin chi tiết, quản lý giỏ hàng, đặt hàng và thanh toán trực tuyến. Các chức năng này đòi hỏi hệ thống phải hoạt động chính xác, ổn định và đảm bảo an toàn thông tin người dùng.

Tuy nhiên, cùng với sự gia tăng về số lượng chức năng và mức độ phức tạp của hệ thống, nguy cơ phát sinh lỗi trong quá trình phát triển và vận hành phần mềm cũng ngày càng cao. Các lỗi liên quan đến chức năng, giao diện, dữ liệu hoặc hiệu năng có thể ảnh hưởng trực tiếp đến trải nghiệm người dùng, làm giảm uy tín của doanh nghiệp và gây thiệt hại về kinh tế. Do đó, kiểm thử phần mềm trở thành một hoạt động không thể thiếu nhằm đảm bảo chất lượng của hệ thống trước và trong quá trình đưa vào sử dụng.

Trong bối cảnh các mô hình phát triển phần mềm hiện đại như **Agile** và **CI/CD** được áp dụng rộng rãi, kiểm thử phần mềm không chỉ được thực hiện ở giai đoạn cuối mà cần được tích hợp xuyên suốt vòng đời phát triển phần mềm. Xuất phát từ yêu cầu thực tiễn đó, đồ án này tập trung nghiên cứu và triển khai kiểm thử phần mềm cho một website bán giày, với trọng tâm là xây dựng kế hoạch kiểm thử, áp dụng các phương pháp và kỹ thuật kiểm thử phù hợp, cũng như tổng hợp và đánh giá kết quả kiểm thử thông qua các báo cáo kiểm thử.

### **1.2. Phạm vi đề tài**

Đề tài tập trung vào việc kiểm thử phần mềm cho website bán giày, trong đó phạm vi nghiên cứu và thực hiện bao gồm: Phân tích yêu cầu nghiệp vụ, phân tích thiết kế và hệ thống và kiến trúc hệ thống của website bán giày làm cơ sở cho hoạt động kiểm thử; Xây dựng kế hoạch kiểm thử nhằm xác định mục tiêu, phạm vi, chiến lược và nguồn lực kiểm thử; Thiết kế và thực hiện các ca kiểm thử cho các chức năng nghiệp vụ chính của hệ thống, bao gồm kiểm thử chức năng, kiểm thử giao diện và kiểm thử dữ liệu; Áp dụng các phương pháp và kỹ thuật kiểm thử trong môi trường phát triển theo mô hình **Agile** và tích hợp kiểm thử trong quy trình **CI/CD** và cuối cùng tổng hợp kết quả kiểm thử thông qua các tài liệu như test report, bug report và test summary.

### **1.3. Quy trình Agile và CI/CD**

#### 1.3.1. Agile

Trong quá trình phát triển và kiểm thử website bán giày, nhóm áp dụng phương pháp Agile theo mô hình Scrum nhằm tổ chức và quản lý công việc một cách linh hoạt, phù hợp với đặc thù thay đổi thường xuyên của yêu cầu nghiệp vụ. Các yêu cầu của hệ thống được phân tách thành các User Story và đưa vào Product Backlog, làm cơ sở cho việc lập kế hoạch và triển khai các vòng lặp phát triển.

Dựa trên mức độ ưu tiên, các User Story được lựa chọn và thực hiện trong từng Sprint, với thời gian mỗi Sprint kéo dài từ một đến hai tuần. Mỗi Sprint hướng tới việc tạo ra một phiên bản phần mềm có thể vận hành và đủ điều kiện để kiểm thử, qua đó cho phép đánh giá chất lượng hệ thống một cách liên tục. Trong phạm vi mỗi Sprint, các hoạt động được thực hiện theo trình tự:

* Phân tích và làm rõ yêu cầu nghiệp vụ của các User Story.
* Thiết kế và phát triển các chức năng tương ứng ở các tầng giao diện, xử lý nghiệp vụ và dữ liệu.
* Thiết kế và thực hiện các kịch bản kiểm thử ở nhiều mức độ, bao gồm Unit Test, Integration Test, System Test và UAT.
* Triển khai phiên bản mới lên môi trường chạy thử nhằm phục vụ việc kiểm thử và đánh giá thực tế.

Kết thúc mỗi Sprint, nhóm tiến hành Sprint Review để đánh giá mức độ hoàn thành các mục tiêu đề ra. Việc áp dụng Agile/Scrum đảm bảo bám sát mục tiêu kiểm thử ban đầu, vừa linh hoạt điều chỉnh kế hoạch khi có yêu cầu bổ sung hoặc chính sửa.

#### 1.3.2. CI/CD

Triển khai quy trình CI/CD (Continuous Integration/Continuous Deployment) nhằm tự động hóa quá trình tích hợp, kiểm thử và triển khai hệ thống. Mã nguồn của dự án được quản lý tập trung trên nền tảng GitHub, cho phép kiểm soát phiên bản và phối hợp làm việc hiệu quả. Mỗi khi có thay đổi mã nguồn thông qua các thao tác commit hoặc pull request trên các nhánh chính, hệ thống GitHub Actions sẽ tự động kích hoạt các workflow đã được cấu hình sẵn. Quy trình CI/CD được chia thành hai giai đoạn chính:

Continuous Integration (CI):

* Tự động thiết lập môi trường và cài đặt các thư viện phụ thuộc cho cả phía máy chủ và giao diện người dùng.
* Tự động thực thi các bài kiểm thử như Unit Test và Integration Test nhằm phát hiện sớm các lỗi ở mức logic nghiệp vụ và API.
* Kiểm tra quá trình build của Frontend và Backend để đảm bảo mã nguồn ở trạng thái sẵn sàng triển khai.

Continuous Deployment (CD):

* Sau khi các bước tích hợp và kiểm thử hoàn tất thành công, pipeline tự động triển khai phiên bản mới của hệ thống lên môi trường chạy thử.
* Sử dụng các biến môi trường để cấu hình các thông số cần thiết như kết nối cơ sở dữ liệu, khóa bảo mật và các dịch vụ tích hợp bên ngoài.
* Mỗi lần cập nhật mã nguồn đều tạo ra một phiên bản triển khai ổn định, cho phép nhóm thực hiện kiểm thử liên tục trên môi trường gần với thực tế.

Việc áp dụng CI/CD giúp rút ngắn chu kỳ phản hồi, nâng cao hiệu quả kiểm thử và đảm bảo rằng các thay đổi của hệ thống đều được kiểm tra và đánh giá kịp thời trước khi đưa vào sử dụng.

## **2. Phân tích yêu cầu**

### **2.1. Khảo sát hiện trạng**

Website bán giày là một hệ thống thương mại điện tử hoạt động trên nền tảng web, cung cấp các chức năng cơ bản như xem và tìm kiếm sản phẩm, quản lý giỏ hàng, đặt hàng trực tuyến và xem lại những đơn hàng đã được đặt. Hệ thống được xây dựng theo mô hình client–server, trong đó phía máy chủ xử lý nghiệp vụ và dữ liệu, phía máy khách đảm nhiệm giao diện và tương tác với người dùng.

Qua khảo sát, hệ thống đã đáp ứng được các chức năng nghiệp vụ chính, tuy nhiên vẫn tồn tại các rủi ro tiềm ẩn liên quan đến lỗi chức năng, tính nhất quán dữ liệu và độ ổn định khi hệ thống được mở rộng. Vì vậy, việc thực hiện kiểm thử là cần thiết nhằm đánh giá chất lượng hệ thống, phát hiện sớm lỗi và làm cơ sở cho việc xây dựng kế hoạch kiểm thử trong các giai đoạn tiếp theo.

### **2.2. Bối cảnh kinh doanh**

Hệ thống EShoe có các nghiệp vụ cơ bản của một hệ thống thương mại điện tử bao gồm các chức năng: tìm kiếm sản phẩm và quản lý giỏ hàng, quy trình đặt hàng và thanh toán, phân quyền truy cập người dùng

**Quy trình quản lý sản phẩm:**

Khách hàng vào trang web có thể xem được các sản phẩm đang được bày bán trên web, với các chức năng như tìm kiếm sản phẩm theo tên nằm trên khung tìm kiếm của trang chủ hoặc chọn lọc theo giới tính ở đầu trang chủ, chọn size giày, màu sắc, giá tiền. Khi nhấn vào một sản phẩm trên trang chủ hay sản phẩm sau khi được lọc hoặc tìm kiếm, khách hàng sẽ được chuyển sang trang chi tiết sản phẩm. Trang chi tiết sản phẩm chứa các thông tin của sản phẩm: Tên sản phẩm, mô tả, màu sắc, size, giá bán. Quản trị viên có thể quản lý dữ liệu sản phẩm như: thêm sản phẩm mới khi cửa hàng nhập sản phẩm mới về bán, chỉnh sửa thông tin sản phẩm khi muốn sửa đổi thông tin sản phẩm, ẩn sản phẩm khi cửa hàng không còn bán sản phẩm này nữa.

**Quy trình mua hàng:**

Sau khi khách hàng lựa chọn được sản phẩm muốn mua, khách hàng có thể nhấn vào nút thêm vào giỏ hàng ở trang chủ hoặc trang chi tiết sản phẩm, các thông tin của sản phẩm sẽ hiển thị trong giỏ hàng. Khách hàng có thể xem lại giỏ hàng bằng cách nhấn vào giỏ hàng, ở đó khách hàng có thể tăng số lượng sản phẩm muốn mua hoặc xóa sản phẩm khỏi giỏ hàng khi không muốn mua nữa, khi thêm số lượng hoặc xóa sản phẩm khỏi giỏ hàng thì giỏ hàng sẽ cập nhật lại giá tiền, số lượng sản phẩm,.. . Khách hàng tiến hành đặt hàng bằng cách nhấn vào nút “đặt hàng” trong giỏ hàng, người dùng được chuyển qua trang đặt hàng, ở đó người dùng có thể chọn địa chỉ giao hàng, xem lại các sản phẩm trong đơn hàng, lựa chọn hình thức thanh toán bằng tiền mặt. Nếu khách hàng không sửa đổi lựa chọn thì nhấn nút “xác nhận” để tiến hành đặt đơn. Nếu khách hàng sửa đổi lựa chọn thì nhấn quay về giỏ hàng, hủy quá trình đặt hàng, các sản phẩm vẫn còn trong giỏ hàng.

**Quy trình quản lý đơn hàng:**

Khi khách hàng xác nhận đặt hàng thành công, khách hàng có thể xem lại đơn hàng mình đã đặt ở trang lịch sử đơn hàng. Các đơn hàng có các trạng thái: chờ xác nhận, đã xác nhận, đang giao, đã giao, đã hủy. Khách hàng có thể hủy đơn hàng khi quản trị viên chưa nhấn xác nhận. Đối với quản trị viên, đơn hàng mới sẽ hiển thị bên trang đơn hàng của quản trị viên, quản trị viên có thể tiến hành xử lý đơn hàng: xác nhận đơn hàng, đã chuyển qua đơn vị vận chuyển, xác nhận đã giao, nếu hủy đơn sẽ thông báo đến khách hàng. Trạng thái đơn sẽ được cập nhật bên phía khách hàng và quản trị viên

**Quy trình phân quyền truy cập:**

Hệ thống hỗ trợ phân quyền theo hai vai trò: khách hàng và quản trị viên. Với vai trò khách hàng đăng ký, đăng nhập thành công sẽ chuyển khách hàng đến trang chủ phía khách hàng, thấy được các sản phẩm đang được bày bán, có thể tiến hành đặt hàng. Đăng nhập với vai trò quản trị viên sẽ được chuyển đến trang quản lý sản phẩm và đơn hàng. Khách hàng và quản trị viên có thể đăng xuất khi không sử dụng trang web nữa.

#### 2.2.1. Sơ đồ các tác nhân tham gia hệ thống

Khách hàng là người sử dụng hệ thống để mua hàng.

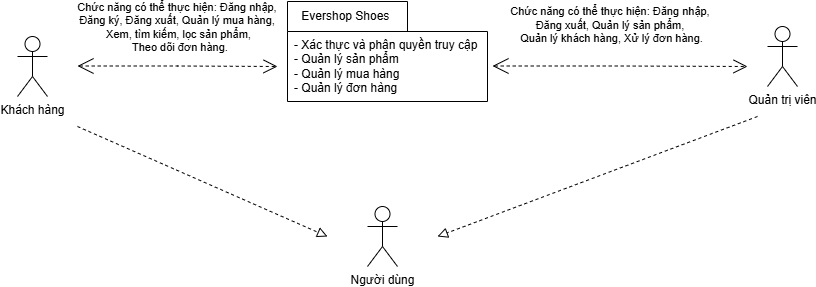
Quản trị viên hệ thống là quản trị viên có chức vụ cao nhất của hệ thống, có quyền thực hiện tất cả các chức năng quản trị.

A white dots on a black background

Description automatically generated

Hình 1.2.2 1. . Sơ đồ các tác nhân tham gia hệ thống

#### 2.2.2. Sơ đồ ngữ cảnh hệ thống



Hình 1.2.2 2. Sơ đồ ngữ cảnh hệ thống

Hình ảnh minh họa luồng tương tác giữa các tác nhân (Actors) và hệ thống Evershop Shoes.

Các thành phần chính:

* Khách hàng: Thực hiện các chức năng như Đăng nhập, Đăng ký, Quản lý mua hàng, Tìm kiếm/Lọc sản phẩm và Theo dõi đơn hàng.
* Quản trị viên: Thực hiện Đăng nhập, Đăng xuất, Quản lý sản phẩm, Quản lý khách hàng và Xử lý đơn hàng.
* Hệ thống Evershop Shoes: Đóng vai trò trung tâm xử lý xác thực, phân quyền và quản lý các nghiệp vụ lõi.

### **2.4. Công nghệ sử dụng**

* **Node.js**: Môi trường thực thi phía máy chủ mã nguồn mở được xây dựng trên công cụ JavaScript V8 của Chrome. Nó cung cấp một môi trường thực thi hướng sự kiện, không chặn (bất đồng bộ) và đa nền tảng để xây dựng các ứng dụng phía máy chủ có khả năng mở rộng cao bằng JavaScript.
* **PostgreSQL 13**: Hệ quản trị cơ sở dữ liệu quan hệ, dùng cho lưu trữ và query.
* **TypeScript**: Ngôn ngữ mở rộng JavaScript với type safety, dùng cho code backend.
* **React:** Thư viện JavaScript mã nguồn mở và miễn phí dành cho giao diện người dùng dựa trên các thành phần UI. EverShop triển khai việc render các thành phần React phía máy chủ với cơ chế hydration để cung cấp trải nghiệm nhanh, hiệu quả và tối ưu hóa SEO.
* **GraphQL:**  là ngôn ngữ truy vấn dành cho API và là môi trường thực thi để đáp ứng các truy vấn đó với dữ liệu hiện có của bạn. EverShop sử dụng GraphQL và React để xây dựng giao diện người dùng linh hoạt và có khả năng mở rộng.

### **2.5. Yêu cầu nghiệp vụ**

#### 2.5.1. Yêu cầu chức năng

|  |  |  |  |
| --- | --- | --- | --- |
| **Mã yêu cầu** | **Người yêu cầu** | **Nội dung yêu cầu** | **Mức thiết yếu** |
| **BR1** | **Quản lý sản phẩm** | | |
| BR1.1 | Khách hàng | Xem được tất cả các sản phẩm hiện đang bán trên website. | High |
| BR1.2 | Khách hàng | Tìm kiếm theo tên và lọc theo giá, size, màu sắc có sắp xếp theo thứ tự tăng dần hoặc bảng chữ cái, có phân trang. | High |
| BR1.3 | Khách hàng | Xem được chi tiết sản phẩm khi nhấn vào sản phẩm đó. | High |
| BR1.4 | Quản trị viên | Thêm sản phẩm mới | High |
| BR1.5 | Quản trị viên | Chỉnh sửa thông tin sản phẩm | Medium |
| BR1.6 | Quản trị viên | Ẩn sản phẩm khi không bán nữa | Medium |
| **BR2** | **Quy trình mua hàng** | | |
| BR2.1 | Khách hàng | Thêm sản phẩm vào giỏ hàng | High |
| BR2.2 | Khách hàng | Xóa sản phẩm khỏi giỏ hàng | High |
| BR2.3 | Khách hàng | Xem danh sách các sản phẩm có trong giỏ hàng | High |
| BR2.4 | Khách hàng | Chỉnh sửa số lượng sản phẩm trong giỏ hàng | High |
| BR2.5 | Khách hàng | Chọn địa chỉ giao hàng | High |
| BR2.6 | Khách hàng | Chọn hình thức thanh toán (thanh toán tiền mặt) | High |
| BR2.7 | Khách hàng | Quay về giỏ hàng nếu không muốn mua nữa | Medium |
| **BR3** | **Quản lý đơn hàng** | | |
| BR3.1 | Khách hàng | Xem lại lịch sử đơn hàng | High |
| BR3.2 | Khách hàng | Xem chi tiết đơn hàng | High |
| BR3.3 | Khách hàng | Hủy đơn hàng | High |
| BR3.4 | Quản trị viên | Xem danh sách đơn hàng | Medium |
| BR3.5 | Quản trị viên | Cập nhật trạng thái đơn: đã xác nhận -> đang vận chuyển -> đã giao. | High |
| BR3.6 | Quản trị viên | Nhập lý do hủy đơn hàng và thông báo cho khách | Medium |
| **BR4** | **Phân quyền truy cập** | | |
| BR4.1 | Khách hàng | Đăng ký tài khoản bằng tên đăng nhập và mật khẩu | High |
| BR4.2 | Khách hàng | Đăng nhập bằng tên và mật khẩu đã đăng ký | High |
| BR4.3 | Quản trị viên | Đăng nhập bằng tên và mật khẩu riêng được cấp | Medium |
| BR4.4 | Khách hàng Quản trị viên | Đăng xuất khỏi hệ thống | Medium |

Bảng 1.2.5. 1. Bảng yêu cầu chức năng

#### 2.5.2. Yêu cầu phi chức năng

|  |  |  |  |
| --- | --- | --- | --- |
| **Mã yêu cầu** | **Người yêu cầu** | **Nội dung yêu cầu** | **Mức thiết yếu** |
| **BR1** | **Hiệu năng** | | |
| BR1.1 | Khách hàng, quản trị viên | Tốc độ tải trang phải nhanh, không giật lag. | High |
| BR1.2 | Khách hàng | Thời gian phản hồi của các thao tác (thêm vào giỏ, thanh toán) phải nhanh chóng < 3 giây, người dùng biết được sản phẩm đã được thêm, không gây khó chịu cho người dùng. | High |
| **BR2** | **Bảo mật** | | |
| BR2.1 | Khách hàng, Quản trị viên | Quá trình đăng nhập/đăng ký phải an toàn (mã hóa mật khẩu). | High |
| BR2.2 | Khách hàng, Quản trị viên | Phân quyền rõ ràng giữa vai trò Quản trị viên và Người mua. | High |
| **BR3** | **Tính khả dụng** | | |
| BR3.1 | Khách hàng | Giao diện website phải thân thiện, dễ hiểu, và dễ sử dụng. | High |
| BR3.2 | Quản trị viên | Trang quản trị phải được tổ chức logic, giúp quản trị viên dễ dàng quản lý. | High |

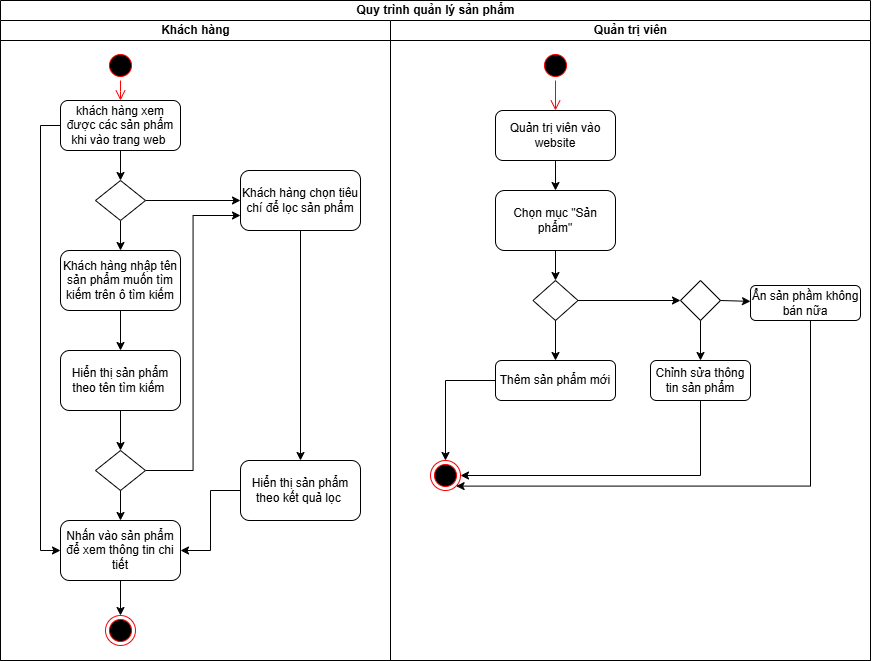
Bảng 1.2.5. 2. Bảng yêu cầu phi chức năng

# CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG VÀ KIẾN TRÚC HỆ THỐNG

## **1. Phân tích thiết kế hệ thống**

### **1.1. Sơ đồ quy trình nghiệp vụ**

#### 1.1.1. Sơ đồ quy trình nghiệp vụ quản lý sản phẩm



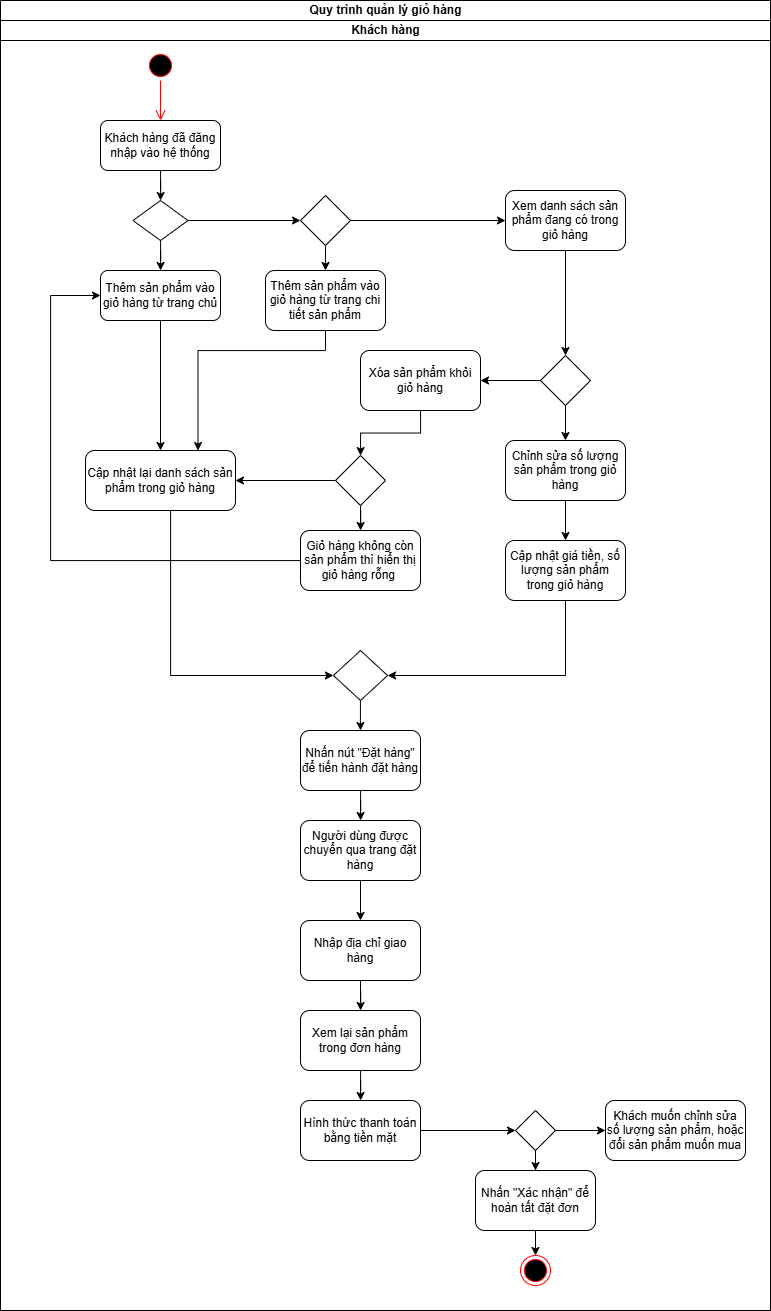
Hình 2.1.1. 1. Sơ dồ quy trình nghiệp vụ quản lý sản phẩm

**Mô tả:**

Khách hàng sau khi đăng nhập vào trang web có thể xem được các sản phẩm đang được bày bán và chi tiết sản phẩm, tìm kiếm sản phẩm thông qua khung tìm kiếm và lọc theo tiêu chí tăng trải nghiệm người dùng.

Quản trị viên sau khi đăng nhập có thể tiến hành thêm sản phẩm mới khi đến lúc nhập hàng, chỉnh sửa lại thông tin của sản phẩm và ẩn sản phẩm đi khi không bán nữa ở mục “Sản phẩm”

#### 1.1.2. Sơ đồ quy trình nghiệp vụ mua hàng

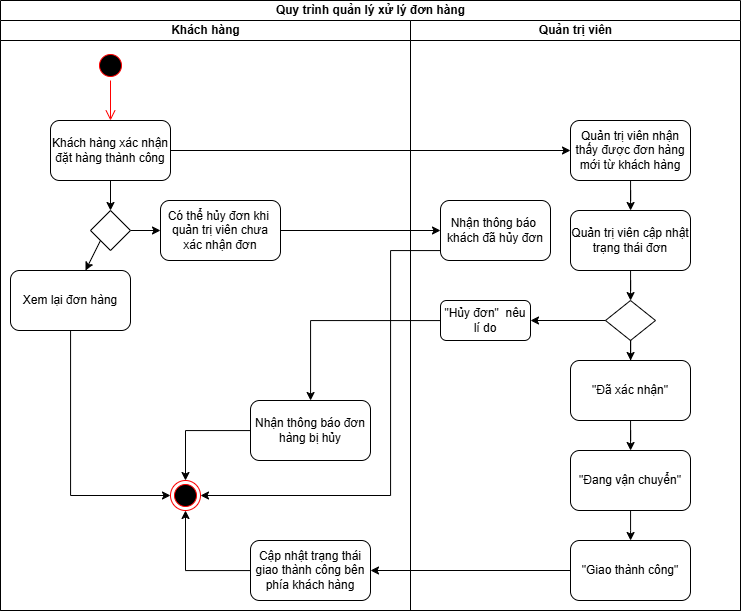


Hình 2.1.1. 2. Sơ đồ quy trình nghiệp vụ mua hàng

**Mô tả:**

Khách hàng đã đăng nhập vào hệ thống có thể thêm sản phẩm vào giỏ hàng ở trang chủ hoặc trang chi tiết sản phẩm, sau khi thêm sản phẩm cập nhật lại danh sách sản phẩm trong giỏ hàng. Khách hàng có thể xem lại giỏ hàng, khách có thể tiến hành chỉnh sửa số lượng sản phẩm trong giỏ hàng, khi chỉnh sửa số lượng giá tiền và số lượng sản phẩm sẽ được cập nhật lại. Khách có thể xóa sản phẩm khỏi giỏ hàng khi đó danh sách sản phẩm trong giỏ hàng sẽ được cập nhật lại, nếu đó là sản phẩm duy nhất trong giỏ hàng thì khi xóa giỏ hàng sẽ rỗng. Sau đó nhấn nút “Đặt hàng” khách được chuyển qua trang đặt hàng. Ở đó khách hàng nhập địa chỉ giao hàng, xem xét lại các sản phẩm đang có trong đơn hàng với hình thức thanh toán bằng tiền mặt trước khi nhấn xác nhận. Khi nhấn xác nhận khách hàng sẽ hoàn tất quá trình đặt đơn. Khi khách hàng muốn chỉnh sửa số lượng sản phẩm hoặc đổi sản phẩm, không muốn mua sản phẩm này nữa khách có thể trở về trang giỏ hàng.

#### 1.1.3. Sơ đồ quy trình nghiệp vụ Quản lý đơn hàng

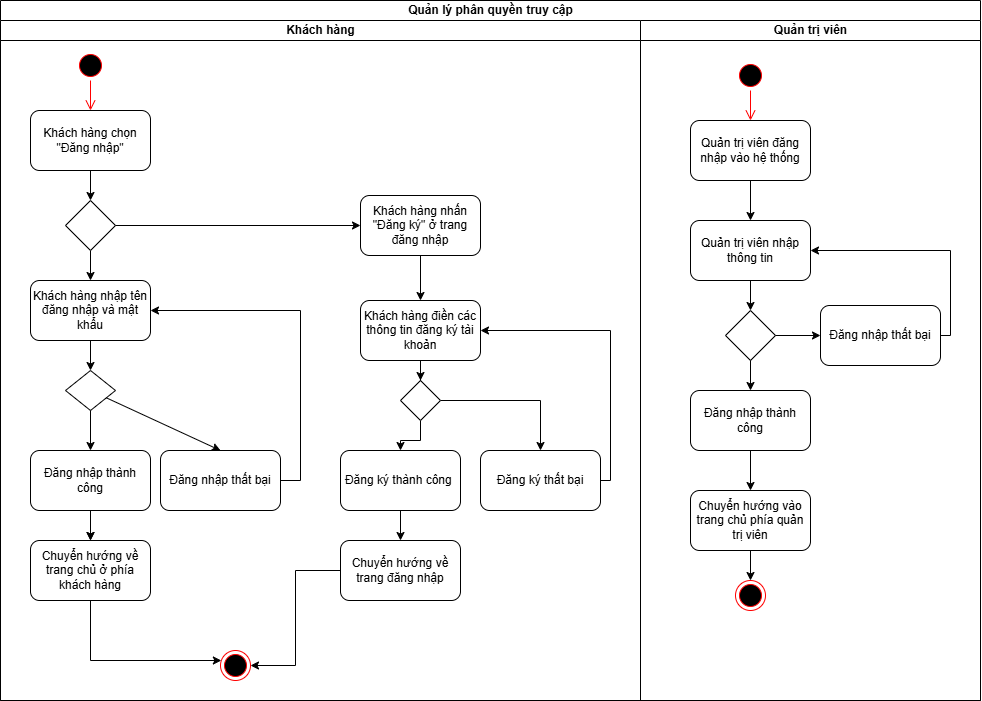


Hình 2.1.1. 3. Sơ đồ quy trình nghiệp vụ quản lý đơn hàng

**Mô tả:**

Sau khi khách hàng xác nhận đặt hàng thành công, khách có thể xem lại đơn hàng hoặc đổi ý không mua nữa có thể hủy đơn nhưng chỉ được hủy trước khi quản trị viên xác nhận đơn hàng, khi hủy đơn thành công thông báo cho quản trị viên khách đã hủy đơn. Quản trị viên thấy được đơn hàng mới, tiến hành cập nhật trạng thái đơn nếu khách hàng chưa hủy, xác nhận đơn, đang vận chuyển và cuối cùng là giao thành công, giao thành công cập nhật trạng thái bên phía khách hàng. Quản trị viên cũng có thể hủy đơn nhưng phải đưa ra lí do thông báo cho khách hàng.

#### 1.1.4. Sơ đồ quy trình nghiệp vụ Quản lý phân quyền truy cập



Hình 2.1.1. 4. Sơ đồ quy trình nghiệp vụ quản lý phân quyền truy cập

**Mô tả:**

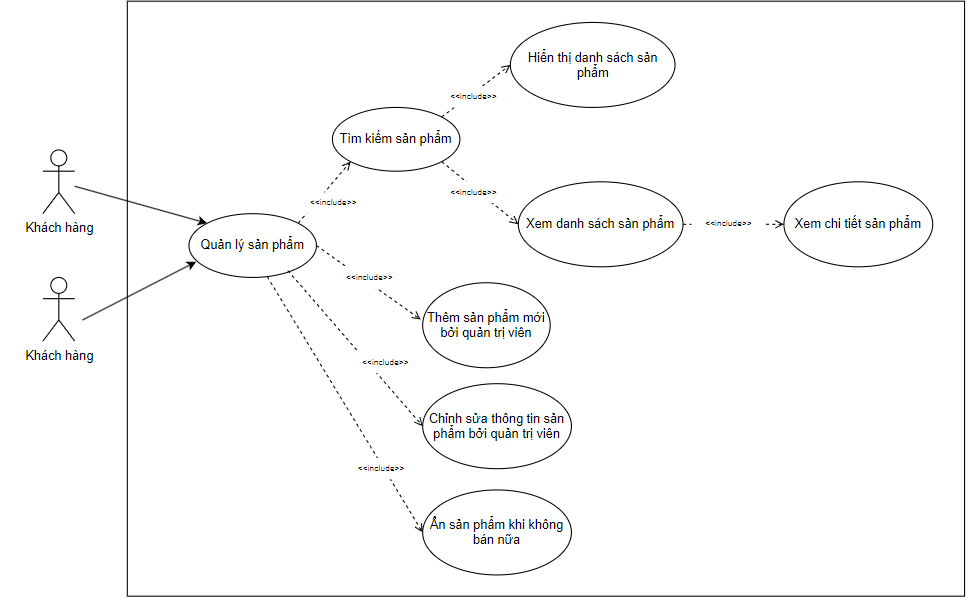
Đối với khách hàng đã có tài khoản có thể đăng nhập vào website bằng email và mật khẩu, đăng nhập thành công sẽ chuyển hướng khách hàng đến trang chủ phía khách hàng để khách hàng có thể mua sắm. Nếu đăng nhập thất bại cho khách hàng nhập lại.

Đối với khách chưa có tài khoản có thể tạo tài khoản bằng cách nhấn vào “Đăng ký” ở trang đăng nhập và điền các thông tin để đăng ký tài khoản. Đăng ký thành công chuyển hướng về trang đăng nhập để khách hàng đăng nhập vào website. Đăng ký không thành công cho khách hàng điền lại thông tin.

Quản trị viên đăng nhập vào website bằng email và mật khẩu được cấp riêng, đăng nhập thành công chuyển hướng đến trang chủ phía quản trị viên để quản trị viên có thể thao tác. Khi đăng nhập thất bại cho nhập lại.

### **1.2. Use Case**

#### 1.2.1. Quản lý sản phẩm



Hình 2.1.2. 1. Use case quản lý sản phẩm

**Mô tả:**

Khách hàng có thể xem danh sách sản phẩm, tìm kiếm hoặc lọc sản phẩm theo giá tiền, size, màu sắccó thể sắp xếp theo bảng chữ cái và xem chi tiết sản phẩm. Quản trị viên có thể quản lý sản phẩm bằng cách thêm mới sản phẩm khi hết hàng, chỉnh sửa thông tin sản phẩm khi cần thiết và ẩn sản phẩm khi không bán mặt hàng đó nữa

#### 1.2.2. Quản lý mua hàng

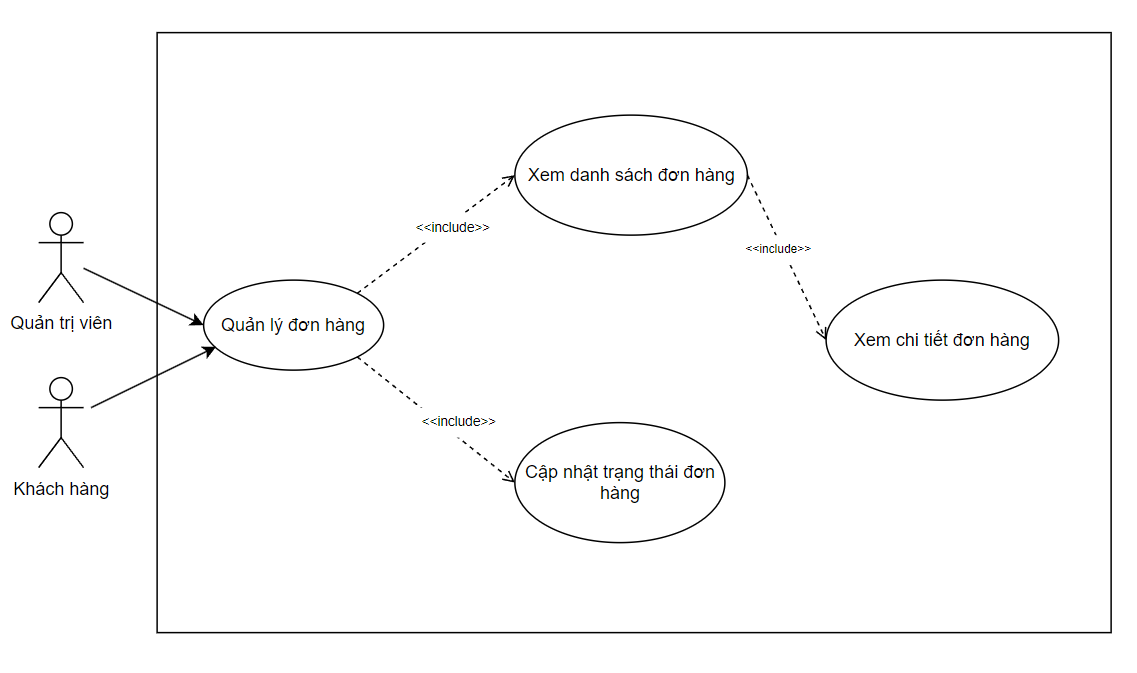


Hình 2.1.2. 2. Use case quản lý mua hàng

**Mô tả:**

Khách hàng có thể xem các sản phẩm có trong giỏ hàng, khi mua sản phẩm mới nhấn “thêm sản phẩm” ở trang chủ và trang chi tiết sản phẩm. Khách hàng có thể xóa sản phẩm khỏi giỏ hàng khi không muốn mua nữa hoặc chỉnh sửa số lượng sản phẩm đang có trong giỏ hàng. Khách hàng nhấn đặt hàng, hiển thị thông tin của đơn hàng, khách xác nhận địa chỉ, chọn phương thức thanh toán rồi nhất “xác nhận” để hoàn tất đặt hàng.

#### 1.2.3. Quản lý đơn hàng

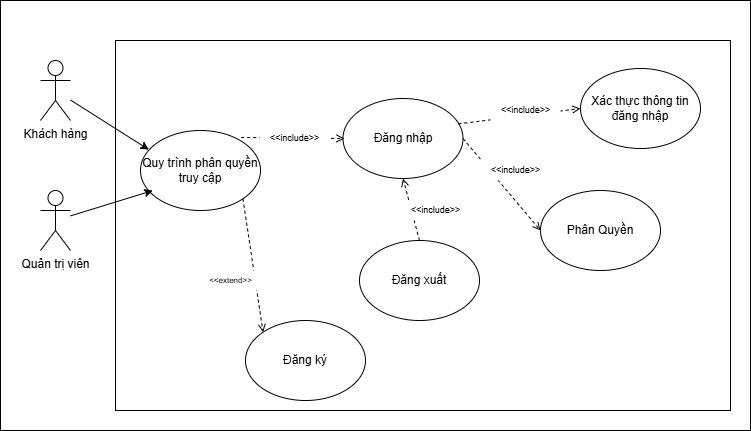


Hình 2.1.2. 3. Use case quản lý đơn hàng

**Mô tả:**

Khách hàng có thể xem lại danh sách đơn hàng và chi tiết đơn hàng sau khi đặt mua. Quản trị viên có thể cập nhật trạng thái đơn hàng để giao hàng cho khách hoặc hủy đơn hàng.

#### 1.2.4. Quản lý phân quyền truy cập



Hình 2.1.2. 4. Use case quản lý phân quyền truy cập

**Mô tả:**

Khách hàng có thể đăng ký, đăng nhập vào hệ thống để thực hiện mua hàng, sau khi sử dụng hệ thống xong khách hàng có thể đăng xuất. Quản trị viên có thể đăng nhập vào hệ thống, đăng xuất khỏi hệ thống. Khi đăng nhập tiến hành xác thực thông tin và phân quyền theo vai trò người dùng.

### **1.3. User story**

**User story quản lý sản phẩm**

Là một khách hàng, tôi muốn xem tất cả sản phẩm có trên website để lựa chọn sản phẩm phù hợp với bản thân.

Là một khách hàng, tôi muốn xem được chi tiết sản phẩm khi ấn vào sản phẩm trên trang chủ để biết thêm về thông tin sản phẩm

Là một khách hàng, tôi muốn tìm kiếm sản phẩm theo tên để có thể tìm ra được sản phẩm mà tôi muốn.

Là một khách hàng, tôi muốn lọc sản phẩm theo giá tiền, size, màu sắc của giày để tìm ra đôi giày ưng ý.

Là một khách hàng, tôi muốn có thể kết hợp vừa tìm theo tên vừa lọc được.

Là một quản trị viên, tôi muốn thêm sản phẩm mới vào cửa hàng để có thêm nhiều loại giày mới cập nhật xu hướng.

Là một quản trị viên, tôi muốn có thể chỉnh sửa thông tin sản phẩm khi cần thiết để đảm bảo thông tin sản phẩm chính xác.

Là một quản trị viên, tôi muốn ẩn sản phẩm khi không còn bán nữa.

**User story quản lý mua hàng**

Là một khách hàng, tôi muốn thêm sản phẩm muốn mua từ trang chủ hoặc trang chi tiết sản phẩm vào giỏ hàng .

Là một khách hàng, tôi muốn xem lại danh sách các sản phẩm trong giỏ hàng trước khi thanh toán.

Là một khách hàng, tôi muốn thay đổi số lượng sản phẩm trong giỏ hàng để đáp ứng nhu cầu của tôi.

Là một khách hàng, tôi muốn có thể xóa sản phẩm trong giỏ hàng khi không muốn mua nữa.

Là một khách hàng, tôi muốn xem thông tin đơn hàng để chắc chắn trước khi xác nhận mua.

Là một khách hàng, tôi muốn có thể đổi địa chỉ giao hàng để giao đến nơi mà tôi muốn.

Là một khách hàng, tôi muốn có thể chọn phương thức thanh toán.

**User story quản lý đơn hàng**

Là một khách hàng, tôi muốn xem danh sách đơn hàng mà tôi đã đặt.

Là một khách hàng, tôi muốn xem chi tiết đơn hàng đã đặt để coi lại thông tin của đơn hàng khi tôi muốn.

Là một khách hàng, tôi muốn có thể hủy đơn hàng để chỉnh sửa lại số lượng hoặc không mua sản phẩm đó nữa.

Là một khách hàng, tôi muốn xem được trạng thái đơn hàng để biết khi nào đơn hàng đến.

Là một quản trị viên, tôi muốn xem danh sách đơn hàng để có thể tiến hành cập nhật đơn.

Là một quản trị viên, tôi muốn cập nhật đơn hàng thành xác nhận đơn để bán sản phẩm.

Là một quản trị viên, tôi muốn hủy đơn hàng khi mặt hàng đó bị lỗi.

**User story quản lý phân quyền truy cập**

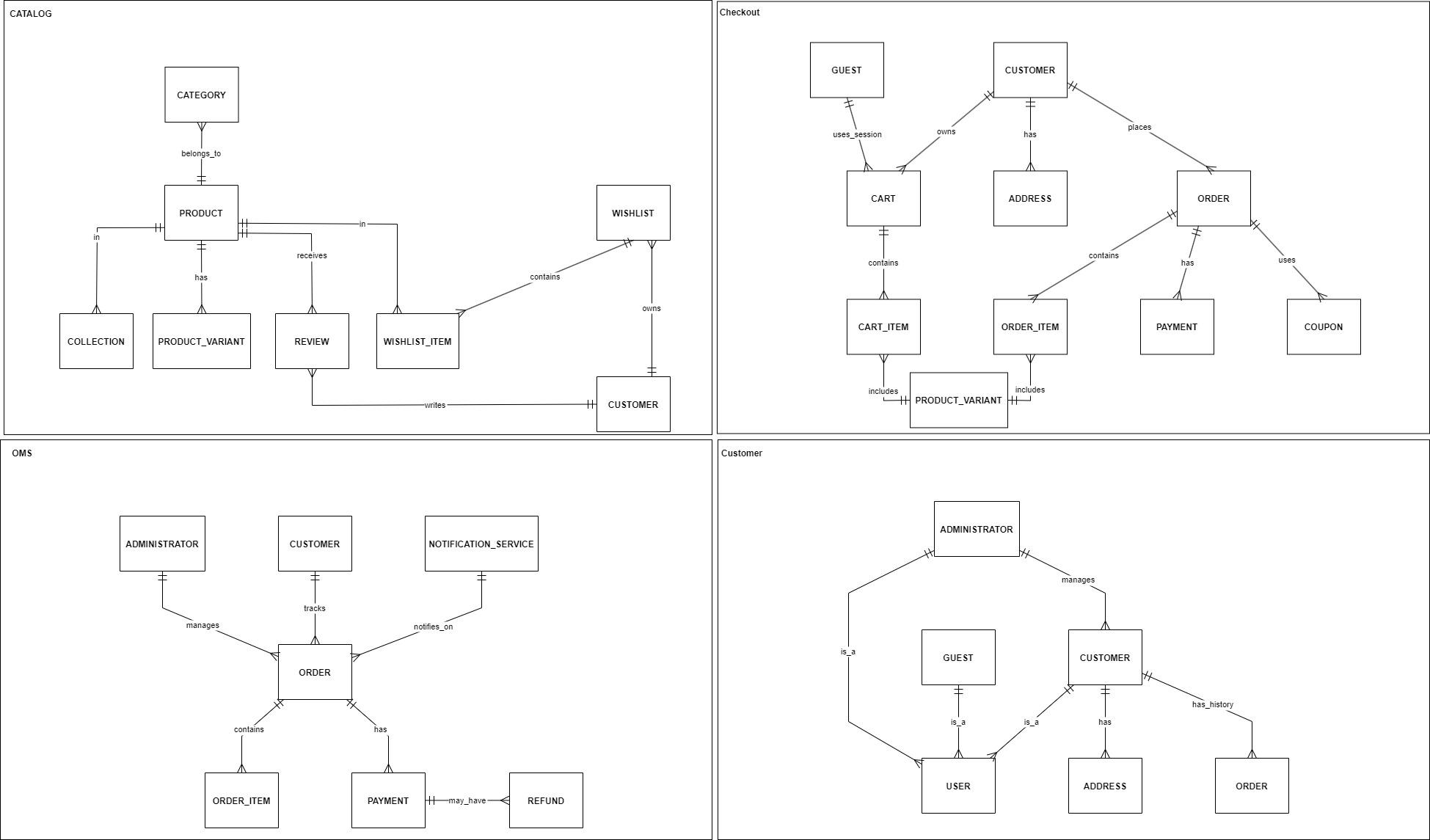
Là một khách hàng truy cập vào trang web, tôi muốn đăng ký tài khoản mới để có thể đăng nhập và sử dụng các chức năng của hệ thống.

Là một quản trị viên, tôi muốn đăng nhập vào tài khoản để quản lý đơn hàng và sản phẩm và đăng xuất.

Là một quản trị viên hệ thống, tôi muốn khóa hoặc mở khóa tài khoản người dùng để kiểm soát quyền truy cập của các tài khoản vi phạm hoặc không còn hoạt động.

### **1.4. Dữ liệu**

#### 1.4.1. Conceptual ERD



Hình 2.1.3. 1. Conceptual ERD

**Mô tả:**

Biểu đồ Conceptual ERD mô tả các thực thể chính trong hệ thống "Online Shoe Store", bao gồm Catalog (Category belongs\_to Product, Product has Collection/Variant/Review), Wishlist (Wishlist contains Wishlist\_Item, owns by Customer).

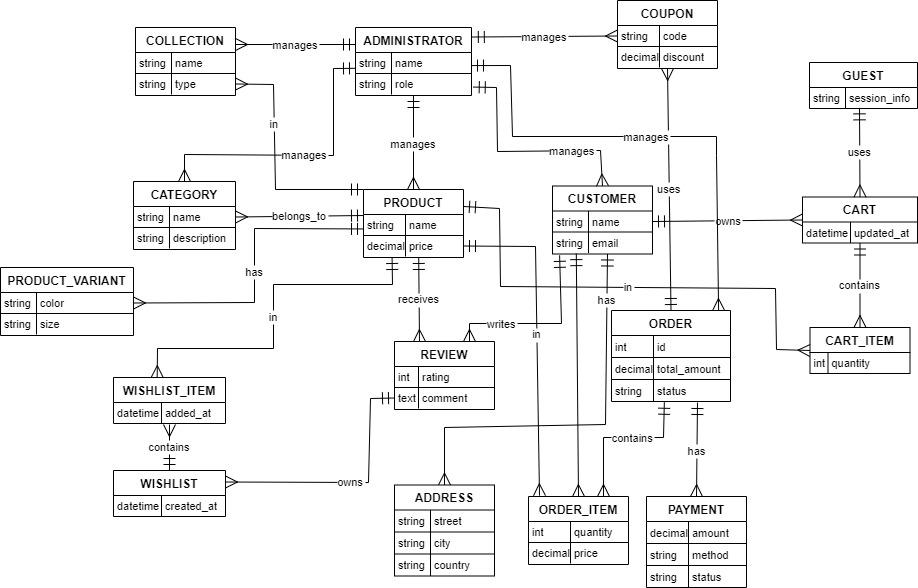
Phần Checkout: Guest/Customer uses/owns Cart contains Cart\_Item (includes Product Variant), places Order contains Order\_Item has Payment uses Coupon.

OMS (Order Management System): Administrator manages Order, Customer tracks Order notifies Notification Service, Order contains Order\_Item has Payment may\_have Refund.

Phần Customer: Administrator manages, Guest/User is\_a Customer has Address has\_history Order.

Tổng thể, ERD nhấn mạnh mối quan hệ giữa sản phẩm, khách hàng, đơn hàng và quản lý, tập trung vào khái niệm trừu tượng mà không chỉ định thuộc tính chi tiết.

#### 1.4.2. Logical ERD



Hình 2.1.3. 2. Logical ERD

**Mô tả:**

Biểu đồ Logical ERD mô tả schema cơ sở dữ liệu cho hệ thống "Online Shoe Store" với các entity chi tiết thuộc tính: Administrator (string name, role) manages Collection (string name, type), Category (string name, description), Product (string name, decimal price, has Product Variant: string color/size), Customer (string name, email), Coupon (string code, decimal discount), Order (int id, decimal total\_amount, string status).

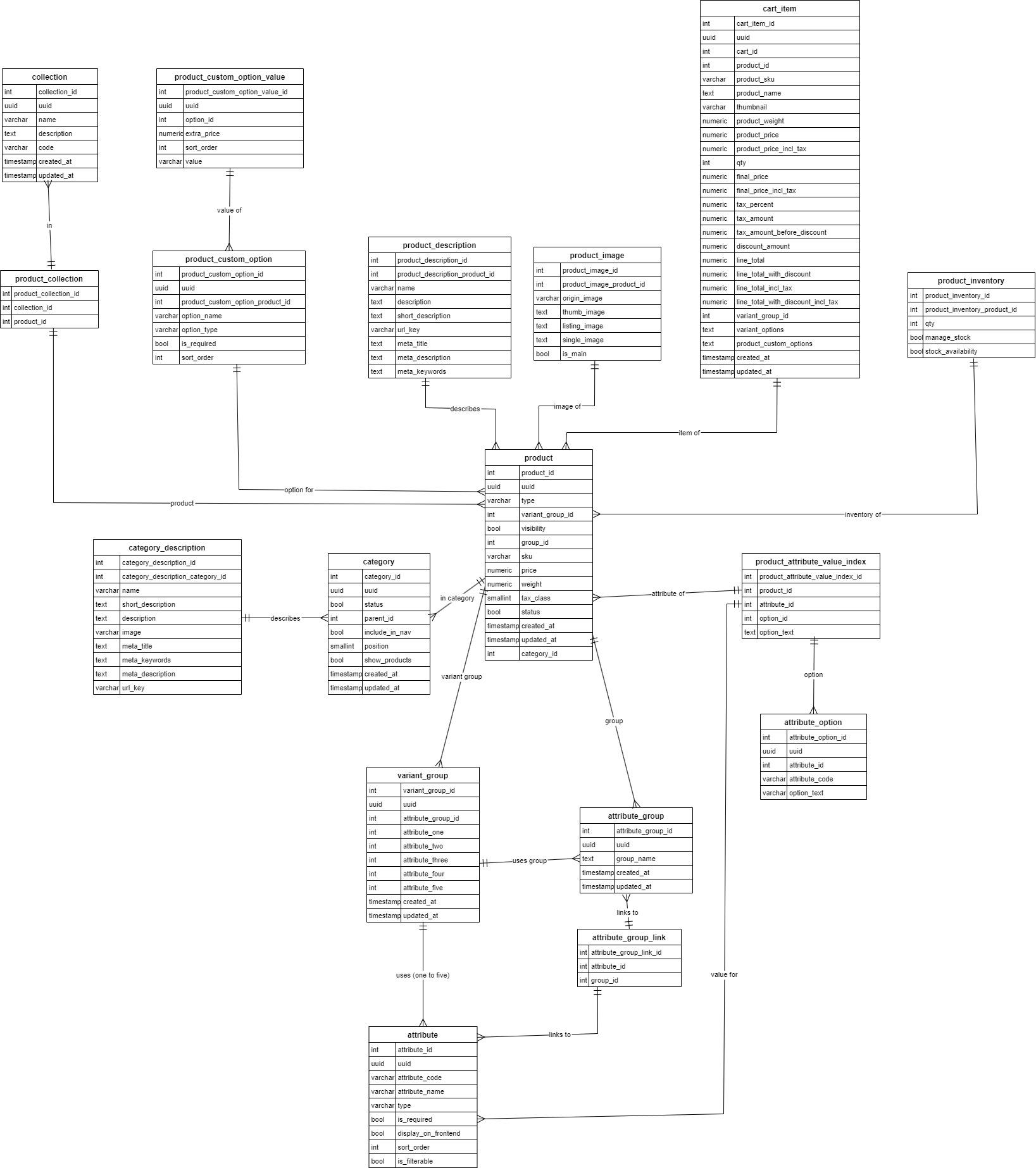
Customer owns Wishlist (datetime created\_at, contains Wishlist\_Item: datetime added\_at), owns Address (string street, city, country), receives Review (int rating, text comment, writes by Customer), uses Cart (datetime updated\_at, contains Cart\_Item: int quantity).

Order has Order\_Item (int quantity, decimal price), has Payment (decimal amount, string method, status), uses Coupon.

Guest (string session\_info) uses Cart tương tự Customer.

Tổng thể, ERD nhấn mạnh mối quan hệ (belongs\_to, has, owns, manages, contains) với kiểu dữ liệu cụ thể để hỗ trợ triển khai database.

#### 1.4.3. Physical ERD



Hình 2.1.3. 3. Physical Product-Catalog

**Mô tả:**

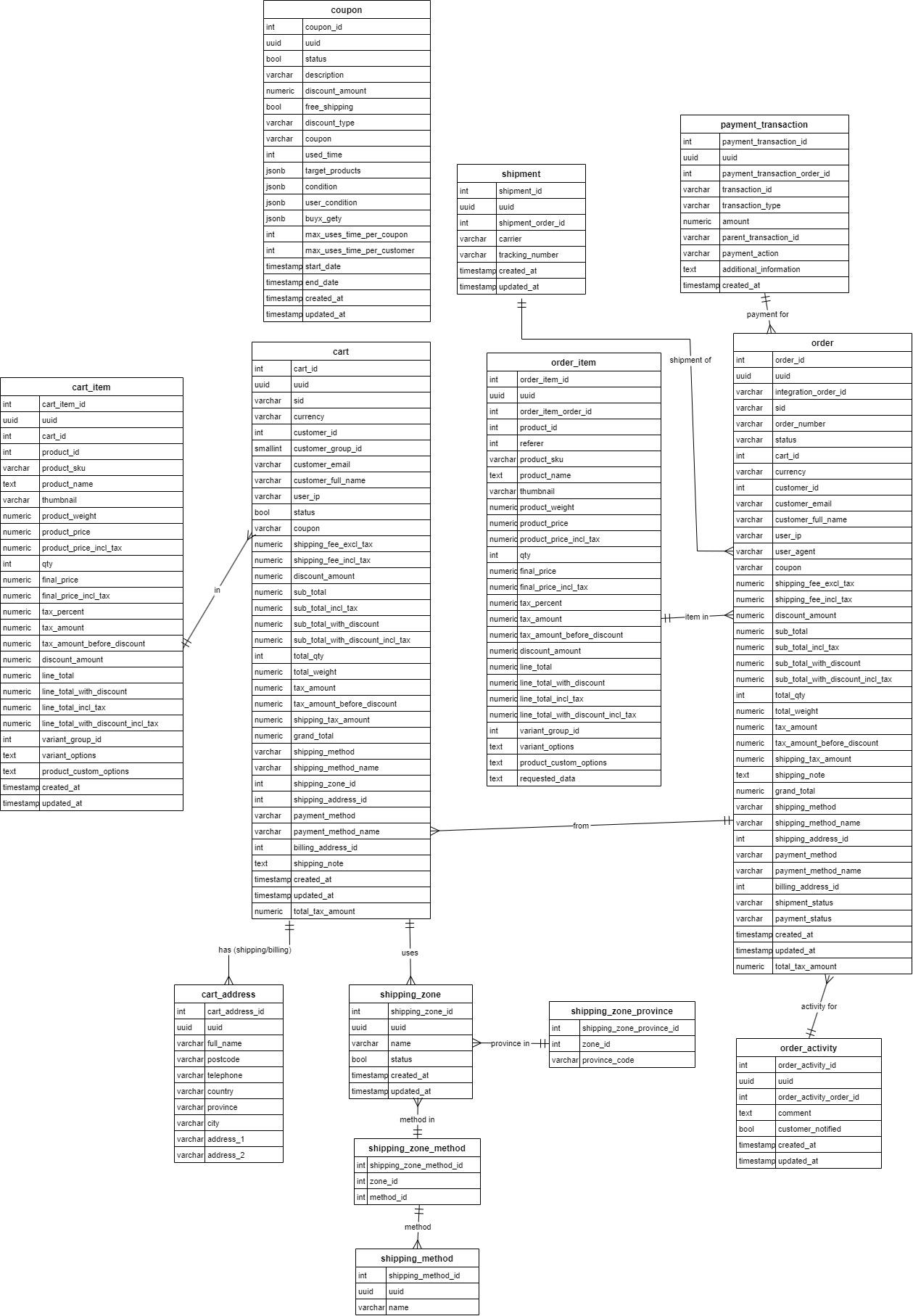
Biểu đồ Physical ERD của module Catalog mô tả schema cơ sở dữ liệu với các bảng chính: Collection (int id, varchar name, varchar type, timestamp created\_at/updated\_at), liên kết với Product\_Collection (int product\_id/collection\_id) để quản lý sản phẩm thuộc bộ sưu tập.

Product (int id, varchar name, int description\_id, numeric price, int stock, int category\_id, timestamp created\_at/updated\_at) kết nối với Product\_Description (int id, varchar short\_description, text meta\_description, varchar meta\_keywords, etc.), Product\_Image (int id, varchar image, bool main\_image), và nhận Review (int id, int rating, text comment).

Category (int id, varchar name, int description\_id, bool active, timestamp created\_at/updated\_at) belongs\_to Product và có Category\_Description tương tự.

Variant\_Group (int id, varchar name, int attribute\_group\_id), Variant (int id, int variant\_group\_id, int attribute\_id, varchar value, timestamp created\_at/updated\_at), Option (int id, int attribute\_id, varchar value/index), và Attribute (int id, varchar name, varchar type, bool required, timestamp created\_at/updated\_at) hỗ trợ biến thể sản phẩm với mối quan hệ 1:N và N:N.

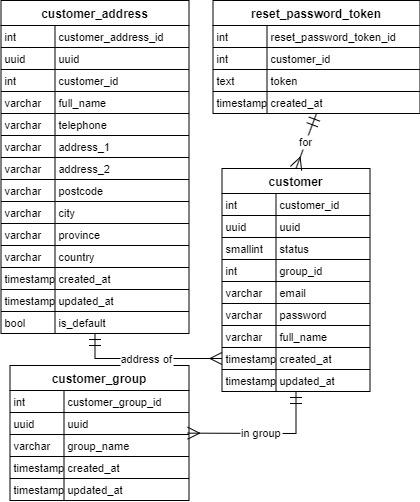
Tổng thể, ERD nhấn mạnh khóa chính (int id), khóa ngoại, kiểu dữ liệu cụ thể (varchar, numeric, timestamp), và chỉ mục để triển khai database PostgreSQL cho module quản lý danh mục sản phẩm.



Hình 2.1.3. 4. Physical Cart - Order

**Mô tả:**

Biểu đồ ERD logic cho module checkout-OMS mô tả các thực thể cốt lõi trong hệ thống xử lý đơn hàng thương mại điện tử. Ở trung tâm là thực thể Order, liên kết với Order\_Item để chi tiết hóa sản phẩm, số lượng và giá cả, đồng thời kết nối với Shipment để quản lý chi tiết giao hàng như phương thức và khu vực. Payment\_Transaction liên kết với Order và Payment để ghi nhận giao dịch, bao gồm loại, số tiền và trạng thái. Thực thể Cart dẫn vào Order, kết nối với Cart\_Item cho các mặt hàng trước thanh toán và Coupon cho các mã giảm giá áp dụng. Các thực thể hỗ trợ bao gồm Shipping\_Zone, Shipping\_Method và Activity để ghi nhật ký sự kiện, với các mối quan hệ được thực thi qua khóa ngoại như order\_id và shipment\_id để duy trì tính nhất quán dữ liệu.



Hình 2.1.3. 5. Physical Customer

**Mô tả:**

Biểu đồ mô tả sơ đồ cơ sở dữ liệu vật lý với bốn bảng chính: Customer\_address, Reset\_password\_token, Customer và Customer\_group.

Bảng Customer\_address chứa các trường như int customer\_address\_id (khóa chính), int customer\_id (khóa ngoại), varchar full\_name, telephone, address\_1, address\_2, postcode, city, province, country, timestamp created\_at, updated\_at, và bool is\_default. Nó liên kết với bảng Customer qua mũi tên "address of Customer".

Bảng Reset\_password\_token bao gồm int reset\_password\_token\_id (khóa chính), int customer\_id (khóa ngoại), text token, và timestamp created\_at. Nó liên kết trực tiếp với bảng Customer.

Bảng Customer là trung tâm, với các trường int customer\_id (khóa chính), uuid uuid, smallint status, int group\_id (khóa ngoại), varchar email, password, full\_name, timestamp created\_at và updated\_at.

Bảng Customer\_group có int customer\_group\_id (khóa chính), varchar group\_name, timestamp created\_at và updated\_at.

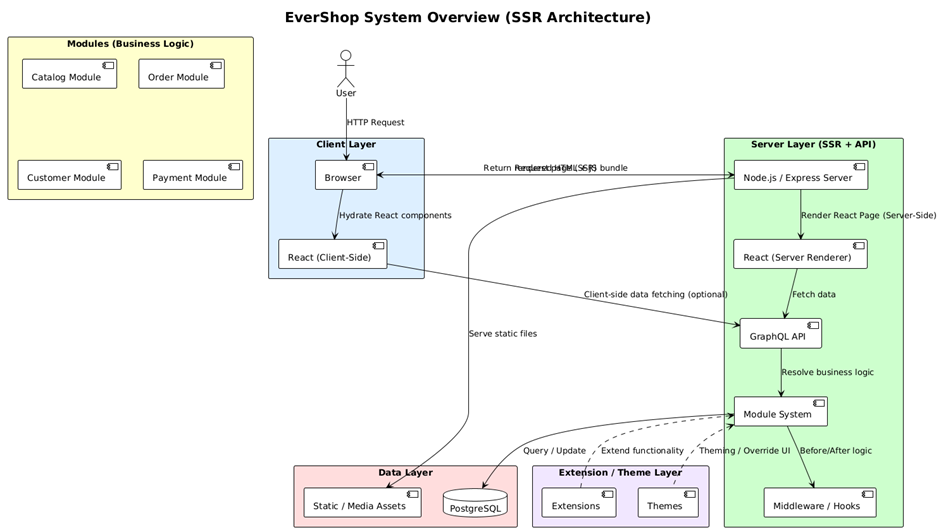
Mũi tên "in group" nối từ Customer đến Customer\_group, cho thấy mối quan hệ nhóm khách hàng.

### **1.5. Từ điển dữ liệu**

Vì danh sách bảng của dữ liệu quá nhiều nhóm đã làm một file .docx riêng đề trình bày về từ điển của dữ liệu.

## **2. Kiến trúc hệ thống**

### **2.1. Sơ đồ tổng quan hệ thống**



Hình 2.2.1. 1. Sơ đồ tổng quan hệ thống

**Mô tả:**

Biểu đồ tổng quan hệ thống EverShop (kiến trúc SSR) bao gồm các lớp chính: Modules (Business Logic) với Catalog, Order, Customer, Payment; Client Layer với User qua Browser gửi HTTP Request đến React (Client-Side) để hydrate components.

Server Layer (SSR + API) sử dụng Node.js/Express Server để render React Page phía server, fetch data qua GraphQL API, resolve business logic qua Module System, và trả về bundle HTML/JS/CSS.

Data Layer bao gồm Static/Media Assets và PostgreSQL để lưu trữ dữ liệu.

Extension/Theme Layer hỗ trợ mở rộng với Extensions, Themes, Middleware/Hooks, cho phép tùy chỉnh chức năng.

Quá trình chảy dữ liệu: Từ client-side fetching (tùy chọn) đến tương tác với GraphQL, đảm bảo rendering server-side và client-side đồng bộ.

### **2.2. Kiến trúc C4**

#### 2.2.1. Level 1: Context Diagram



Hình 2.2.2. 1. Level 1: Context Diagram

**Mô tả:**

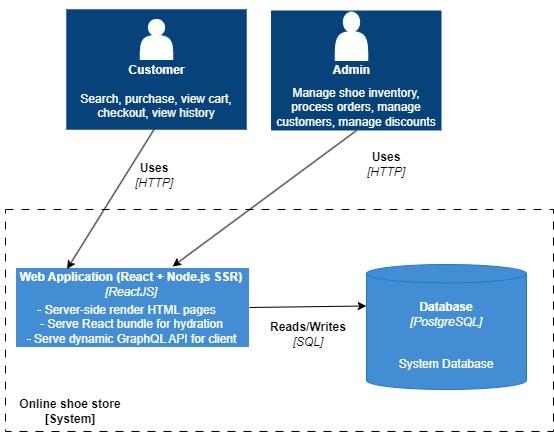
Biểu đồ Context Diagram minh họa hệ thống "Online Shoe Store" làm trung tâm, tương tác với hai thực thể bên ngoài: Khách hàng (Customer) và Quản trị viên (Admin).

Khách hàng thực hiện tìm kiếm sản phẩm, mua hàng, xem giỏ hàng, thanh toán và kiểm tra lịch sử đơn hàng, với hệ thống hỗ trợ quy trình mua sắm trực tuyến.

Quản trị viên quản lý kho hàng, xử lý đơn hàng, tài khoản khách hàng và áp dụng giảm giá để duy trì hoạt động nội bộ.

Dòng chảy dữ liệu hai chiều giúp xác định ranh giới hệ thống và các tương tác chính.

#### 2.2.2. Level 2: Container Diagram



Hình 2.2.2. 2. Level 2: Container Diagram

**Mô tả**:

Biểu đồ minh họa kiến trúc hệ thống "Online Shoe Store" với Web Application làm trung tâm, sử dụng React + Node.js SSR (Server-Side Rendering) thay vì SPA (Single Page Application), giúp render HTML phía server để tối ưu SEO và tốc độ tải ban đầu.

Khách hàng (Customer) và Quản trị viên (Admin) tương tác qua HTTP, với Customer thực hiện tìm kiếm, mua hàng, xem giỏ, thanh toán và lịch sử; Admin quản lý kho hàng, đơn hàng, khách hàng và giảm giá.

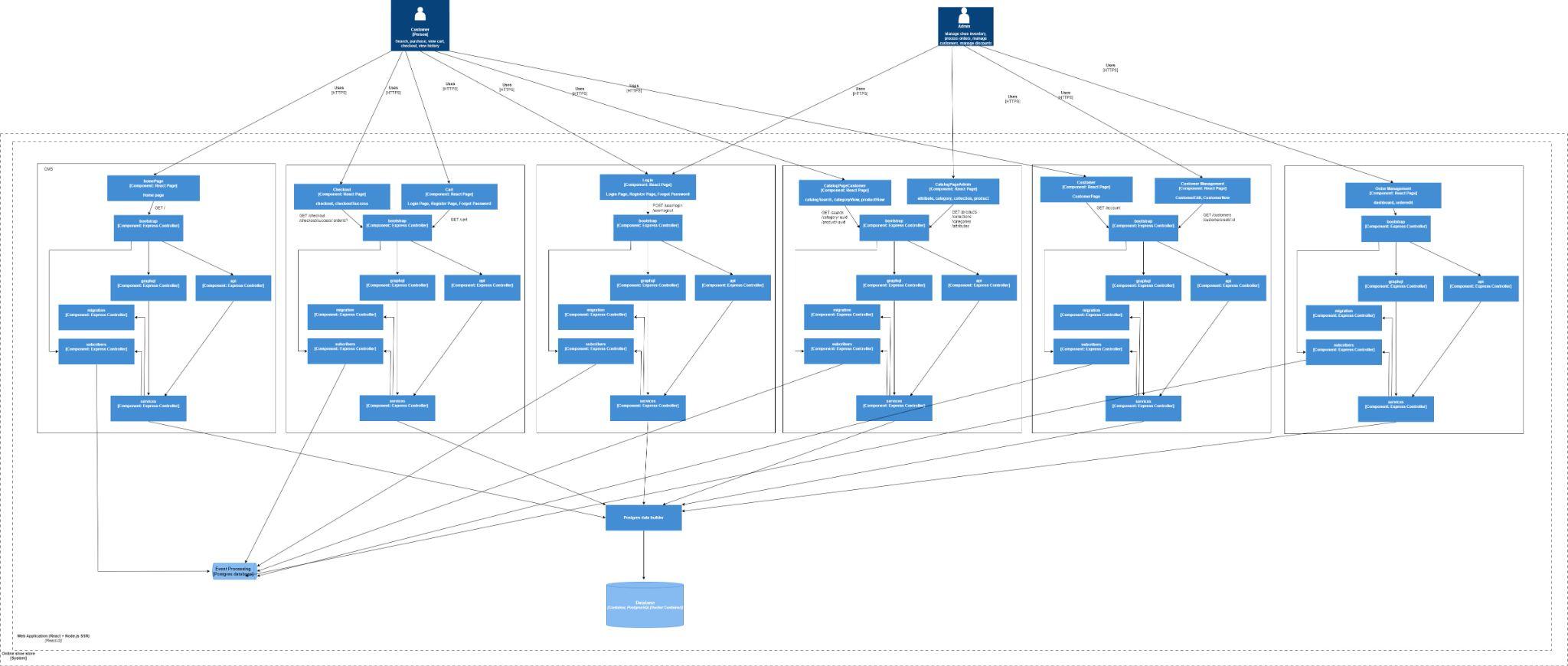
Web Application server-side render HTML pages, serve React bundle cho hydration phía client, và cung cấp dynamic GraphQL API để xử lý dữ liệu động.

Hệ thống kết nối với Database (PostgreSQL) qua reads/writes SQL để lưu trữ và truy xuất dữ liệu.

Sử dụng SSR thay vì SPA giúp giảm tải client-side, cải thiện trải nghiệm người dùng với nội dung sẵn sàng ngay lập tức, đặc biệt phù hợp cho ứng dụng e-commerce.

Tổng thể, biểu đồ nhấn mạnh dòng chảy dữ liệu hai chiều giữa người dùng, ứng dụng web và cơ sở dữ liệu, xác định ranh giới hệ thống.

#### 2.2.3. Level 3: Component Diagram



Hình 2.2.2. 3. Level 3: Component Diagram

**Mô tả:**

Biểu đồ Component Diagram minh họa cấu trúc hệ thống "Online Shoe Store" với hai người dùng chính: Customer (tìm kiếm, mua hàng, xem giỏ, thanh toán, lịch sử) và Admin (quản lý kho, đơn hàng, khách hàng, giảm giá).

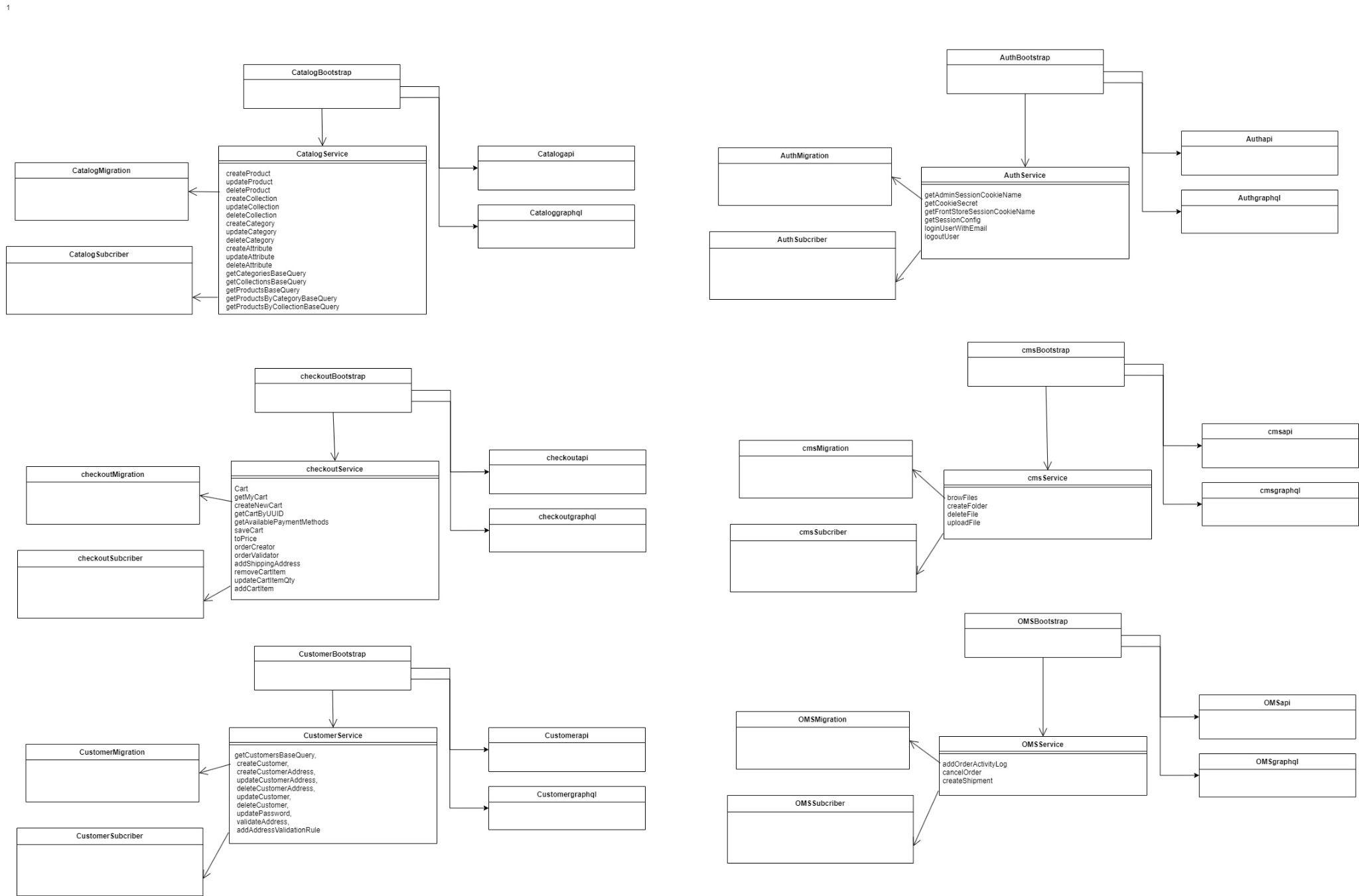
Lớp giao diện bao gồm các component như Home Page, Cart Component, Login/Register Page, Checkout Page, Category/Collection Page, Customer Management Page, Order Management Page, v.v., sử dụng GET/POST để tương tác.

Các service như Continuation Service, Checkout Service, Auth Service, Category Service, Customer Service, Order Service hỗ trợ xử lý logic kinh doanh.

Hệ thống kết nối với Model Layer qua API, sử dụng Express với PostgreSQL Database để lưu trữ dữ liệu.

Tổng thể, biểu đồ nhấn mạnh sự phân tầng giữa frontend (React components), backend services (Node.js/Express), và cơ sở dữ liệu, với dòng chảy dữ liệu hai chiều.

#### 2.2.4. Level 4: Code-level Diagram



Hình 2.2.2. 4. Code-level Diagram

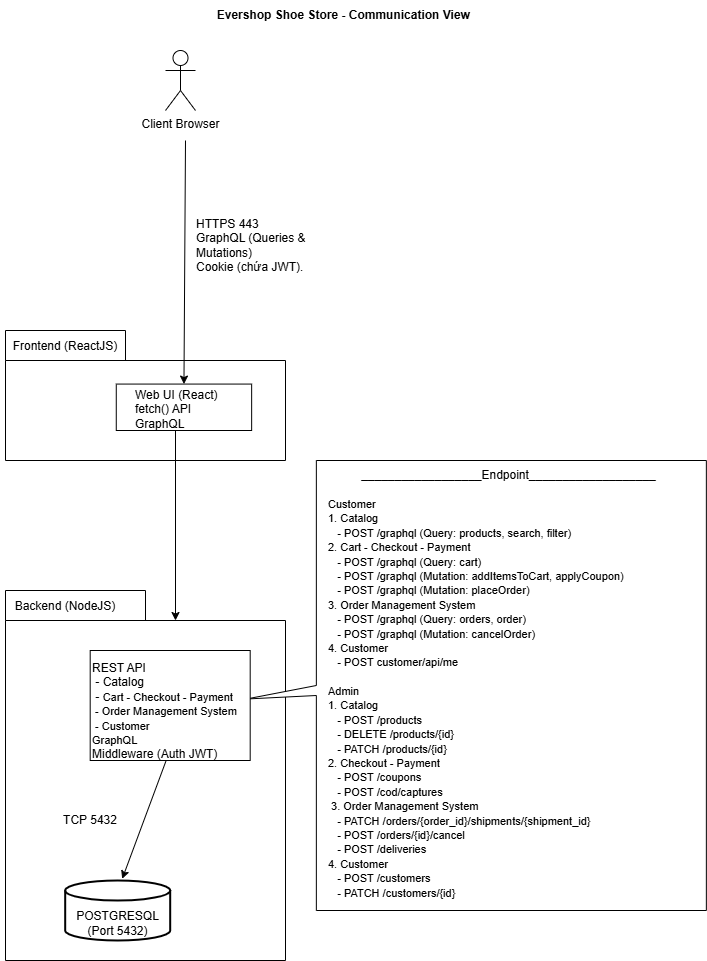
**Mô tả:**

Biểu đồ mô tả cấu trúc mã nguồn với sáu bootstrap chính: CatalogBootstrap, AuthBootstrap, CheckoutBootstrap, CmsBootstrap, CustomerBootstrap và OMSBootstrap.

Mỗi bootstrap kết nối với Migration (di cư dữ liệu), Subscriber (đăng ký sự kiện), Service (dịch vụ với phương thức cụ thể như createProduct, loginUserByEmail), rồi phân nhánh ra api và graphql.

Các mũi tên thể hiện mối quan hệ phân cấp, giúp tổ chức mã cho các module như danh mục sản phẩm, xác thực, thanh toán, quản lý nội dung, khách hàng và quản lý đơn hàng.

### **2.3. Communication View**

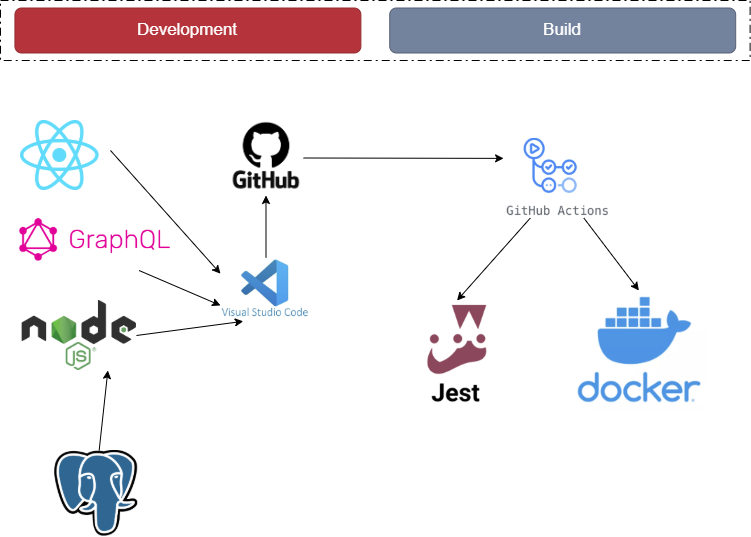


Hình 2.2.3. 1. Communication View

**Mô tả:**

Sơ đồ "Communication View" của hệ thống Evershop Shoe Store mô tả luồng dữ liệu khép kín bắt đầu từ phía người dùng (Client Browser), nơi các yêu cầu được gửi đi qua giao thức bảo mật HTTPS (cổng 443) kèm theo cookie chứa JWT để xác thực. Tầng Frontend được xây dựng bằng ReactJS, chịu trách nhiệm hiển thị giao diện và sử dụng fetch() API và GraphQL để tương tác với Backend. Tại tầng Backend phát triển trên nền tảng NodeJS, hệ thống xử lý các logic nghiệp vụ quan trọng như quản lý danh mục, giỏ hàng, thanh toán và đơn hàng thông qua các Middleware xác thực. Điểm nổi bật của kiến trúc này là việc áp dụng song song hai chuẩn giao tiếp: trong khi người dùng phổ thông (Customer) chủ yếu tương tác qua GraphQL để tối ưu hóa truy vấn dữ liệu, thì quản trị viên (Admin) lại sử dụng hệ thống RESTful API tiêu chuẩn để quản lý tài nguyên. Cuối cùng, mọi dữ liệu được lưu trữ bền vững tại cơ sở dữ liệu PostgreSQL, được Backend kết nối trực tiếp qua giao thức TCP tại cổng 5432.

### **2.4. Deployment View**



Hình 2.2.4. 1. Deployment View

**Mô tả:**

Hình trên mô tả quy trình triển khai và xây dựng hệ thống theo mô hình CI/CD, bao gồm hai giai đoạn chính là Development và Build.

Trong giai đoạn Development, hệ thống được phát triển trên môi trường máy tính cá nhân bằng công cụ Visual Studio Code. Sử dụng React cho phía giao diện người dùng, Node.js cho phía máy chủ và GraphQL để xây dựng các API phục vụ việc trao đổi dữ liệu. Dữ liệu của hệ thống được lưu trữ và quản lý trong cơ sở dữ liệu PostgreSQL. Sau khi hoàn thành việc phát triển và chỉnh sửa mã nguồn, lập trình viên thực hiện đẩy (push) mã nguồn lên kho lưu trữ GitHub.

Trong giai đoạn Build, khi có sự thay đổi mã nguồn trên GitHub, GitHub Actions sẽ tự động được kích hoạt để thực hiện quy trình tích hợp liên tục. Tại đây, hệ thống tiến hành chạy các kịch bản kiểm thử tự động bằng công cụ Jest nhằm đảm bảo tính đúng đắn và ổn định của mã nguồn. Sau khi kiểm thử thành công, ứng dụng được đóng gói dưới dạng container bằng Docker, giúp hệ thống dễ dàng triển khai, mở rộng và đảm bảo tính nhất quán giữa các môi trường.

# CHƯƠNG 3: KẾ HOẠCH KIỂM THỬ

## **1. Giới thiệu**

### **1.1. Mục tiêu**

Mục tiêu của tài liệu là đảm bảo hệ thống Evershop Shoes Store đáp ứng đầy đủ các yêu cầu nghiệp vụ và kỹ thuật đã được đặc tả, phát hiện sớm các lỗi trong quá trình phát triển, từ đó nâng cao chất lượng và độ ổn định của phần mềm trước khi đưa vào sử dụng. Kế hoạch kiểm thử đóng vai trò làm cơ sở thống nhất cho các hoạt động kiểm thử, hỗ trợ kiểm soát tiến độ, đánh giá kết quả kiểm thử và phục vụ cho quá trình nghiệm thu hệ thống.

### **1.2. Phạm vi áp dụng**

Kế hoạch kiểm thử này áp dụng cho toàn bộ quá trình kiểm thử website Evershop Shoes Store trong suốt vòng đời phát triển phần mềm, bao gồm các giai đoạn từ phát triển, tích hợp, kiểm thử hệ thống đến kiểm thử chấp nhận người dùng.

## **2. Các mục kiểm thử**

### **2.1. Kiểm thử chức năng**

|  |  |
| --- | --- |
| **Chức năng cần kiểm thử** | **Điều kiện chấp nhận** |
| Phân quyền truy cập | Xác minh rằng hệ thống cho phép đăng nhập cho chức vụ là khách hàng, quản trị viên.  Xác minh rằng hệ thống cho phép đăng ký tài khoản cho khách hàng.  Xác minh rằng hệ thống cho phép đăng xuất đối với tất cả người dùng đã đăng nhập.  Xác minh rằng hệ thống thực thi phân quyền truy cập đúng, chỉ cho phép người dùng truy cập các chức năng phù hợp với vai trò của mình.  Xác minh rằng hệ thống cho phép quản trị viên thực hiện quản lý người dùng. |
| Quản lý sản phẩm | Xác minh rằng hệ thống cung cấp các thao tác CRUD chức năng quản lý sản phẩm cho quản trị viên.  Xác minh rằng hệ thống cho phép khách hàng truy cập và xem danh sách sản phẩm.  Xác minh rằng hệ thống hỗ trợ các chức năng tìm kiếm, lọc và sắp xếp sản phẩm nhằm giúp khách hàng dễ dàng tìm kiếm sản phẩm mong muốn.  Xác minh rằng hệ thống đảm bảo tính nhất quán và chính xác của dữ liệu sản phẩm giữa giao diện người dùng và hệ thống backend. |
| Quản lý mua hàng | Xác minh rằng hệ thống cung cấp các thao tác CRUD cho giỏ hàng của khách hàng.  Xác minh rằng hệ thống tự động cập nhật và hiển thị chính xác thông tin giỏ hàng, bao gồm danh sách sản phẩm, số lượng và tổng tiền.  Xác minh rằng hệ thống đảm bảo tính nhất quán dữ liệu giỏ hàng trong suốt quá trình người dùng thao tác và chuyển đổi giữa các trang.  Xác minh rằng hệ thống có các ràng buộc nghiệp vụ đối với giỏ hàng giới hạn số lượng mua một sản phẩm theo số lượng hiện có.  Xác minh rằng hệ thống cho phép khách hàng tiếp tục quy trình mua hàng từ giỏ hàng để chuyển sang bước đặt hàng.  Xác minh rằng hệ thống cho phép người chỉnh sửa địa chỉ giao hàng |
| Quản lý nội dung | Xác minh rằng hệ thống cho phép quản trị viên thực hiện các thao tác CRUD (Thêm, sửa, xóa, xem) đối với các trang nội dung tĩnh (như Trang chủ, Về chúng tôi, Chính sách bảo mật).  Xác minh rằng hệ thống cho phép quản trị viên quản lý các thành phần giao diện như Slider, Banner quảng cáo và các khối nội dung trên trang web.  Xác minh rằng các thay đổi về nội dung từ trang quản trị (Admin Dashboard) được cập nhật và hiển thị chính xác trên giao diện người dùng (Storefront) ngay lập tức.  Xác minh rằng hệ thống hỗ trợ trình soạn thảo văn bản (WYSIWYG) để quản trị viên có thể định dạng văn bản, chèn hình ảnh/video vào trang nội dung một cách dễ dàng. |
| Quản lý đơn hàng | Xác minh rằng hệ thống tự động tạo đơn hàng với mã định danh duy nhất ngay sau khi khách hàng hoàn tất quy trình thanh toán.  Xác minh rằng hệ thống cho phép quản trị viên xem danh sách chi tiết tất cả các đơn hàng, bao gồm thông tin khách hàng, trạng thái thanh toán và trạng thái vận chuyển.  Xác minh rằng hệ thống cho phép quản trị viên cập nhật trạng thái đơn hàng (ví dụ: Đang xử lý, Đang giao hàng, Đã hoàn thành, Đã hủy) và thông báo thay đổi này đến khách hàng qua email.  Xác minh rằng hệ thống tự động trừ số lượng tồn kho của sản phẩm tương ứng sau khi đơn hàng được đặt thành công.  Xác minh rằng hệ thống hỗ trợ quản trị viên tạo hóa đơn (Invoice) và phiếu đóng gói (Shipment) cho từng đơn hàng cụ thể.  Xác minh rằng khách hàng có thể theo dõi trạng thái đơn hàng và xem lịch sử đơn hàng của mình trong trang quản lý tài khoản cá nhân. |

Bảng 3.3. 1. Bảng kiểm thử chức năng

Tất cả các tính năng cần kiểm thử sẽ được trình bày chi tiết trong các tài liệu Kịch bản Kiểm thử và tài liệu Trường hợp Kiểm thử tương ứng dựa trên các loại kiểm thử được đề cập trong phần Phạm vi.

Tất cả các tài liệu Kịch bản Kiểm thử sẽ được chuyển giao để xem xét trong giai đoạn Tiền xây dựng và các tài liệu Trường hợp Kiểm thử sẽ được chuyển giao vào giữa giai đoạn Xây dựng, ngay trước khi bắt đầu kiểm thử tích hợp.

### **2.2. Kiểm thử phi chức năng**

|  |  |
| --- | --- |
| **Loại** | **Điều kiện chấp nhận** |
| Kiểm thử 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, không đi vào trang chết.  Xác minh rằng các nút chức năng chính được hiển thị rõ ràng và dễ thao tác. |
| Kiểm thử tương thích | Xác minh rằng giao diện website hiển thị đúng và đồng nhất trên các trình duyệt phổ biến như Chrome, Firefox.  Xác minh rằng hệ thống tự động điều chỉnh giao diện để phù hợp với các kích thước màn hình khác nhau.  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. |
| Kiểm thử 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 lệch, méo.  Xác minh rằng các thông báo lỗi, cảnh báo, hoặc thông tin trạng thái hiển thị đúng theo ngữ cảnh nghiệp vụ. |
| Kiểm thử bảo mật | Xác minh rằng hệ thống mã hóa mật khẩu người dùng.  Xác minh rằng hệ thống kiểm soát truy cập đúng vai trò và không cho phép truy cập trái phép.  Xác minh rằng các yêu cầu HTTP/HTTPS được bảo mật, và dữ liệu không bị lộ qua URL.  Xác minh rằng các phiên đăng nhập tự động hết hạn sau thời gian quy định hoặc khi người dùng đăng xuất. |
| Kiểm thử 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. |

Bảng 3.3. 2. Bảng kiểm thử phi chức năng

### **2.3. Không trong phạm vi kiểm thử**

* Kiểm thử các bảo mật chuyên sâu như: kiểm thử xâm nhập, kiểm thử lỗ hổng hệ thống, hạ tầng mạng. Nhóm chỉ thực hiện kiểm thử bảo mật ở mức cơ bản, tập trung vào xác thực, phân quyền và kiểm tra dữ liệu đầu vào.
* Tích hợp cổng thanh toán bên thứ ba.
* Kiểm thử khả năng phục hồi sau sự cố.

## **3. Tiêu chí kiểm thử chấp nhận**

### **3.1. Độ bao phủ kiểm thử**

Ít nhất 60% yêu cầu chức năng của hệ thống phải được thiết kế và thực thi test case.

### **3.2. Số lượng trường hợp kiểm thử**

Toàn bộ test case ở các cấp độ sau phải được thực thi:

* Kiểm thử đơn vị (Unit Test).
* Kiểm thử tích hợp (Integration Test).

### **3.3. Số lượng lỗi**

Không được tồn tại lỗi mức Critical tại thời điểm chấp nhận.

Số lỗi mức Major không vượt quá 2 lỗi và phải có giải pháp khắc phục được thống nhất.

Các lỗi mức Minor không ảnh hưởng đến chức năng chính của hệ thống.

### **3.4. Độ bao phủ mã nguồn**

**Các chỉ số bao phủ mã nguồn phải đạt:**

* Statement Coverage ≥ 90%.
* Branch Coverage = 100%.
* Path Coverage = 100%

## **4. Chiến lược kiểm thử**

### **4.1. Phương pháp kiểm thử**

Dự án áp dụng mô hình V-Model trong kiểm thử phần mềm, trong đó các hoạt động kiểm thử được thực hiện song song với các giai đoạn phát triển. Mỗi giai đoạn phát triển tương ứng với một cấp độ kiểm thử nhằm đảm bảo chất lượng phần mềm xuyên suốt vòng đời phát triển.

Cụ thể, dự án thực hiện các cấp độ kiểm thử bao gồm: kiểm thử đơn vị, kiểm thử tích hợp, kiểm thử api. Các Test Case được thiết kế dựa trên yêu cầu chức năng và các use case của hệ thống.

Bên cạnh đó, dự án áp dụng kiểm thử tự động thông qua quy trình CI/CD sử dụng GitHub Actions, nhằm tự động hóa các bước Build, Test và Deploy.

### **4.2. Các giai đoạn kiểm thử**

|  |  |  |  |
| --- | --- | --- | --- |
| **Giai đoạn** | **Mục tiêu Ánh xạ từ Sprint Backlog** | **Nhóm Chịu trách nhiệm** | **Tần suất** |
| Unit Test (UT) | Xác minh logic nghiệp vụ cấp độ hàm/phương thức (Services, Controllers, Components). Đảm bảo các Tasks trong Sprint Backlog được mã hóa đúng logic. | Development Team | Chạy tự động khi developer commit code và khi merge vào nhánh chính. |
| Integration Test (IT) | Đảm bảo sự tương tác giữa các module (React ↔ API Spring Boot) và với Cơ sở Dữ liệu (Spring Boot ↔ MySQL) là chính xác. | Development/Testing Team | Chạy tự động sau Unit Test và trước khi Build. |
| System Test (ST) | Kiểm thử toàn bộ hệ thống (end-to-end) cho các User Story được chọn trong Sprint. Đảm bảo toàn bộ luồng nghiệp vụ của Sprint đó hoạt động. | Testing Team | Chạy tự động sau khi bản Build mới được triển khai lên môi trường Staging. Mô phỏng luồng người dùng cuối |
| Regression Test (RT) | Tái kiểm tra toàn bộ các chức năng đã hoàn thành ở các Sprint trước đó (ví dụ: Auth, Catalog) sau khi mã Sprint mới được hợp nhất. | Testing Team | Toàn bộ bộ Test Tự động (Unit, IT, E2E) từ các Sprint trước sẽ được chạy lại để bảo vệ các tính năng đã hoàn thành. |
| Acceptance Test (UAT) | Xác nhận rằng Product Increment (tổng sản phẩm sau Sprint) đáp ứng Definition of Done (DoD) và nhu cầu của đại diện SysAdmin/Buyer. | Đại diện Người dùng | Cuối mỗi Sprint, trên môi trường Staging. |

### **4.3. Các loại kiểm thử**

|  |  |  |
| --- | --- | --- |
| **Loại kiểm thử** | **Mục đích chính theo Product Backlog** | **Kỹ thuật áp dụng** |
| Kiểm thử chức năng | Xác minh từng chức năng theo các module cốt lõi: AUTH (Xác thực), CATALOG (Sản phẩm), CUSTOMER (Khách hàng), CHECKOUT (Thanh toán), OMS (Đơn hàng), và CMS (Nội dung). | Áp dụng Phân vùng tương đương và Phân tích giá trị biên cho các trường nhập liệu như số lượng giày, đơn giá, thông tin thanh toán và mã giảm giá |
| Kiểm thử chu kỳ nghiệp vụ | Đảm bảo chu kỳ nghiệp vụ thương mại điện tử hoàn chỉnh: Tìm kiếm giày -> Thêm vào giỏ -> Đặt hàng -> Xử lý đơn hàng -> Giao hàng thành công. | Sử dụng các luồng nghiệp vụ cốt lõi (User Stories độ ưu tiên cao) làm kịch bản kiểm thử toàn trình (End-to-End) bằng công cụ Cypress. |
| Kiểm thử tính toàn vẹn dữ liệu | Đảm bảo dữ liệu tồn kho, giá sản phẩm và trạng thái đơn hàng được cập nhật chính xác đồng bộ giữa các bảng PostgreSQL | Kiểm tra các ràng buộc dữ liệu và tính toàn vẹn sau các giao dịch (ví dụ: tồn kho giảm đúng số lượng tương ứng ngay sau khi đặt hàng thành công). |
| Kiểm thử hiệu năng | Đo lường hiệu năng của các GraphQL API quan trọng và khả năng xử lý của trang web sử dụng kiến trúc SSR (Server-Side Rendering) | Sử dụng công cụ tạo tải (JMeter/k6) để đo Thời gian Phản hồi (Response Time) và Tỷ lệ Lỗi dưới áp lực truy cập đồng thời. |
| Kiểm thử bảo mật/phân quyền | Đảm bảo hệ thống thực thi phân quyền nghiêm ngặt giữa các vai trò Khách hàng và Quản trị viên. | Thử truy cập các tài nguyên Admin (quản lý kho, đơn hàng) bằng tài khoản khách hàng; kiểm tra việc xác thực và giải mã mã thông báo JWT |

## **5. Môi trường và nguồn lực**

### **5.1. Môi trường kiểm thử**

Môi trường này được sử dụng bởi nhóm để phát triển tính năng và chạy Unit Test/Integration Test.

* Phần cứng

|  |  |  |
| --- | --- | --- |
| **Mục** | **Thành phần** | **Mục đích** |
| Development | Máy tính cá nhân | Lập trình, chạy kiểm thử đơn vị và kiểm thử tích hợp |
| Thiết bị | Desktop, Android, iOS | Kiểm thử giao diện và kiểm thử tương thích |

* Phần mềm

|  |  |  |  |
| --- | --- | --- | --- |
| **Loại** | **Tên** | **Phiên bản** | **Ghi chú** |
| **OS** | Windows 11 | 64-bit | Môi trường Host |
| **Runtime / App Server** | Node.js | 16.x hoặc 18.x (LTS) | Evershop yêu cầu Node.js để vận hành |
| **Package Manager** | npm | 8.x trở lên | Quản lý thư viện |
| **Frontend Framework** | ReactJS | 17.x / 18.x | Framework hiển thị chính |
| **Styling** | Tailwind CSS | Latest | Framework CSS được Evershop sử dụng |
| **Scripting Language** | TypeScript | 5.x | Ngôn ngữ lập trình chính |
| **API Standard** | GraphQL | Latest | Giao thức giao tiếp Client-Server |
| **Database** | PostgreSQL | 13.x hoặc 14.19 | Cơ sở dữ liệu chính |
| **Containerization** | Docker Desktop | 4.x (Engine 24+) | Để chạy môi trường cô lập |
| **IDE** | VS Code | Latest | Code Editor khuyến nghị |

* Hạ tầng

|  |  |  |
| --- | --- | --- |
| **Mục** | **Thành phần** | **Mục đích** |
| Production | Frontend deploy trên Vercel | Triển khai phần mềm thực tế, thực hiện kiểm thử hệ thống và chấp nhận |
| CI/CD | GitHub Actions | Tự động hóa quy trình Build, Test và Deploy |

### **5.2. Nguồn nhân lực**

Bảng này hiển thị số lượng nhân viên cần thiết cho dự án:

|  |  |  |
| --- | --- | --- |
| **Người thực hiện** | **Vai trò** | **Công việc cụ thể** |
| Châu Gia Anh | Dev/Tester | - Tham gia lập kế hoạch kiểm thử - Thực hiện kiểm thử hệ thống (System Test) - Viết tài liệu kiểm thử (Test Report, Test Summary) - Tổng hợp kết quả kiểm thử và đề xuất cải tiến |
| Dương Lê Khánh | Dev/Tester | - Thiết kế Test Case cho các chức năng nâng cao (giỏ hàng, thanh toán, quản lý đơn hàng) - Lập review checklist cho tài liệu và Test Case - Thực hiện kiểm thử tích hợp (Integration Test) - Ghi nhận, theo dõi và xác nhận lỗi |
| Đào Thị Thanh Tâm | Dev/Tester | - Tham gia lập kế hoạch kiểm thử tổng thể - Thiết kế Test Case cho các chức năng chính (đăng nhập, tìm kiếm, đặt hàng) - Thực hiện kiểm thử đơn vị (Unit Test) - Hỗ trợ viết pipeline CI/CD cho kiểm thử tự động |

### **5.3. Milestones**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nhiệm vụ** | **Người phụ trách** | **Thời gian bắt đầu** | **Thời gian kết thúc** |
| Lập kế hoạch kiểm thử | Châu Gia Anh, Đào Thị Thanh Tâm | 01/10/2025 | 12/10/2025 |
| Thiết kế Test Case | Châu Gia Anh, Dương Lê Khánh,  Đào Thị Thanh Tâm | 13/10/2025 | 22/10/2025 |
| Lập review checklist | Dương Lê Khánh | 23/10/2025 | 31/10/2025 |
| Thực hiện kiểm thử đơn vị và tích hợp | Châu Gia Anh, Dương Lê Khánh | 01/11/2025 | 18/11/2025 |
| Viết mã CI/CD trong GitHub Actions | Châu Gia Anh | 19/11/2025 | 25/11/2025 |
| Ghi chép tài liệu kiểm thử | Đào Thị Thanh Tâm | 26/11/2025 | 16/12/2025 |

## **6. Tài liệu bàn giao**

Lập tài liệu word bao gồm tài liệu Database Design, Test Plan và báo cáo tổng thể về phần mềm và kiểm thử.

Lập tài liệu excel bao gồm tài liệu tất cả giao diện, đặc tả các Use Case, Review Checklist, Test Report, Test Case, Test Design.

Mã nguồn của dự án được lưu trên GitHub bao gồm các file kiểm thử đơn vị, kiểm thử tích hợp và kiểm thử hệ thống và pipeline CI/CD trên GitHub Actions

# CHƯƠNG 4: THIẾT KẾ KIỂM THỬ

## **1. Tổng quan**

Thiết kế kiểm thử là giai đoạn quan trọng trong quy trình kiểm thử phần mềm, tập trung vào việc xây dựng các chiến lược, kỹ thuật và tài liệu cụ thể để đảm bảo bao phủ toàn diện các yêu cầu chức năng cũng như phi chức năng. Dựa trên kế hoạch kiểm thử đã lập ở Chương 3, chương này mô tả chi tiết cách thiết kế kiểm thử cho hệ thống Shoes Store EverShop.

### **1.1. Mục tiêu thiết kế kiểm thử**

Việc thiết kế các kịch bản kiểm thử cho hệ thống EverShop Shoes Store không chỉ đơn thuần là tìm lỗi, mà là một quá trình hoạch định chiến lược nhằm đảm bảo tính toàn vẹn của sản phẩm trước khi đến tay người dùng. Mục tiêu cốt lõi bao gồm:

Đảm bảo bao phủ tối đa các yêu cầu (Requirement Coverage)

Mục tiêu hàng đầu của quy trình là xây dựng một "ma trận bao phủ" (Traceability Matrix) chặt chẽ, nơi mọi tính năng được mô tả trong tài liệu nghiệp vụ đều phải có ít nhất một kịch bản kiểm thử tương ứng.

* Bao phủ luồng nghiệp vụ chính: Đảm bảo tất cả các tính năng quan trọng nhất của một sàn thương mại điện tử như: Quản lý danh mục sản phẩm (Catalog), quy trình thanh toán (Checkout), xác thực người dùng (Auth) và quản lý kho hàng được vận hành trơn tru theo đúng thiết kế ban đầu.
* Kiểm soát các kịch bản ngoại lệ (Edge Cases): Không chỉ dừng lại ở các luồng đi đúng (Happy Path), mục tiêu thiết kế còn phải bao quát được các tình huống bất thường như: Nhập sai định dạng dữ liệu, giỏ hàng trống khi thanh toán, hoặc mất kết nối cơ sở dữ liệu đột ngột.
* Đảm bảo tính nhất quán trên các môi trường: Kịch bản kiểm thử được thiết kế để xác minh rằng ứng dụng hoạt động ổn định và hiển thị đồng nhất trên nhiều trình duyệt và kích thước màn hình khác nhau (Responsive Testing).

Phát hiện lỗi sớm thông qua các kỹ thuật kiểm thử hiệu quả

Dự án áp dụng chiến lược "Shift-left testing" – đưa việc kiểm thử vào giai đoạn sớm nhất có thể trong chu kỳ phát triển phần mềm (SDLC) để tối ưu hóa chi phí và thời gian.

* Giảm thiểu chi phí sửa lỗi: Theo nguyên lý kiểm thử, lỗi được phát hiện ở giai đoạn thiết kế hoặc lập trình có chi phí sửa chữa thấp hơn hàng chục lần so với khi đã triển khai (Production). Việc thiết kế các bài Unit Test (với Jest) và kiểm tra tĩnh (với Lint) giúp loại bỏ các lỗi cú pháp và logic cơ bản ngay lập tức.
* Tăng cường độ tin cậy của mã nguồn: Thông qua việc thiết kế các bộ kiểm thử tự động, mỗi khi có một tính năng mới được thêm vào, hệ thống sẽ tự động chạy lại các bài kiểm thử cũ (Regression Testing). Mục tiêu là đảm bảo mã nguồn mới không làm hỏng (break) các tính năng hiện có, từ đó duy trì sự ổn định bền vững cho hệ thống.
* Tối ưu hóa quy trình phản hồi (Feedback Loop): Thiết kế kiểm thử tự động giúp rút ngắn thời gian phản hồi cho đội ngũ phát triển. Thay vì đợi tester thủ công, các kỹ thuật kiểm thử tự động trong CI/CD cung cấp kết quả chất lượng mã nguồn chỉ sau vài phút, giúp lập trình viên điều chỉnh kịp thời.

Đảm bảo trải nghiệm người dùng và hiệu năng

Bên cạnh chức năng, mục tiêu thiết kế còn hướng tới chất lượng sử dụng thực tế:

* Xác minh tính khả dụng (Usability): Đảm bảo giao diện trực quan, dễ thao tác và các thông báo lỗi được hiển thị rõ ràng cho người dùng.
* Đảm bảo an toàn dữ liệu: Đặc biệt quan trọng đối với các module thanh toán (COD) và thông tin cá nhân, mục tiêu thiết kế phải xác minh rằng không có lỗ hổng rò rỉ thông tin trong quá trình truyền tải dữ liệu giữa Client và Server.

Phạm vi và các module chính

Thiết kế kiểm thử tập trung vào các module cốt lõi:

|  |  |  |
| --- | --- | --- |
| **Module** | **Mô tả chính** | **Cấp độ kiểm thử chính** |
| Auth | Phân quyền bảo mật hệ thống | Unit, Intergration, API |
| Catalog | Quản lý sản phẩm, danh mục | Unit, Integration, API, E2E |
| Customer | Đăng ký, đăng nhập, hồ sơ khách hàng | Unit, Integration, API, E2E |
| Checkout | Giỏ hàng, thanh toán, giao hàng | Unit, Integration, API, E2E |
| OMS | Quản lý đơn hàng, vận chuyển | Unit, Integration, API, E2E |
| CMS | Quản lý nội dung, banner, widget | Unit, Integration, API, E2E |

Bảng 4.1. 1. Bảng thiết kế kiểm thử theo module

### **1.2. Kiểm thử đơn vị**

#### 1.2.1. Định nghĩa và Đặc điểm

Kiểm thử đơn vị là mức độ kiểm thử nhỏ nhất trong quy trình kiểm thử phần mềm, tập trung vào việc xác minh tính đúng đắn của các đơn vị mã nguồn độc lập như hàm (function), phương thức (method) hoặc lớp (class).

Trong hệ thống EverShop, Kiểm thử đơn vị (Unit Testing) đóng vai trò là nền móng của kim tự tháp kiểm thử. Sơ đồ kiến trúc thể hiện cách tổ chức các bài test theo cấu trúc phân tầng (Layered Architecture), giúp cô lập logic và đảm bảo tính chính xác của từng thành phần nhỏ nhất trong hệ thống.

**Cơ sở lý luận và Nguyên tắc thiết kế**

Kiểm thử đơn vị là mức độ kiểm thử nhỏ nhất trong quy trình kiểm thử phần mềm, tập trung vào việc xác minh tính đúng đắn của các đơn vị mã nguồn độc lập như hàm (function), phương thức (method) hoặc lớp (class).

**Các nguyên tắc cốt lõi được áp dụng:**

* **Chiến lược "Shift-Left":** Đưa kiểm thử vào giai đoạn sớm nhất của chu kỳ phát triển (SDLC). Việc phát hiện lỗi ngay tại dòng code vừa viết giúp giảm thiểu chi phí sửa lỗi xuống mức thấp nhất.
* **Tính độc lập (Isolation):** Mỗi đơn vị mã nguồn được kiểm thử trong một môi trường cô lập hoàn toàn. Các phụ thuộc bên ngoài (như Cơ sở dữ liệu, API bên thứ ba, hay Hệ thống tệp) không tham gia vào quá trình này nhằm đảm bảo kết quả kiểm thử chỉ phản ánh duy nhất logic của đơn vị đang xét.
* **Nguyên tắc FIRST:** Đảm bảo các bài kiểm thử đơn vị phải chạy Nhanh (Fast), Độc lập (Independent), Có thể lặp lại (Repeatable), Tự xác thực kết quả (Self-validating) và Kịp thời (Timely)..

#### 1.2.2. Cấu trúc phân tầng trong Unit Test của EverShop

Hệ thống tổ chức các bài kiểm thử theo các tầng xử lý riêng biệt để đảm bảo tính dễ bảo trì:

* Tầng Controllers: Kiểm tra việc tiếp nhận và xử lý tham số của các HTTP Request (như authController, userController). Mục tiêu là đảm bảo Controller gọi đúng Service tương ứng và trả về mã trạng thái đúng.
* Tầng Middleware: Xác minh các logic kiểm tra quyền truy cập, định dạng dữ liệu đầu vào (như auth.test.ts).
* Tầng Services: Tập trung vào các quy tắc nghiệp vụ (Business Logic). Đây là tầng quan trọng nhất, nơi xử lý các phép tính giá sản phẩm, logic đơn hàng hoặc báo cáo.
* Tầng Utils: Kiểm tra các hàm tiện ích dùng chung như trình ghi nhật ký (logger), bộ tạo PDF hoặc dịch vụ gửi Email.

#### 1.2.3. Chiến lược giả lập (Mocking Strategy)

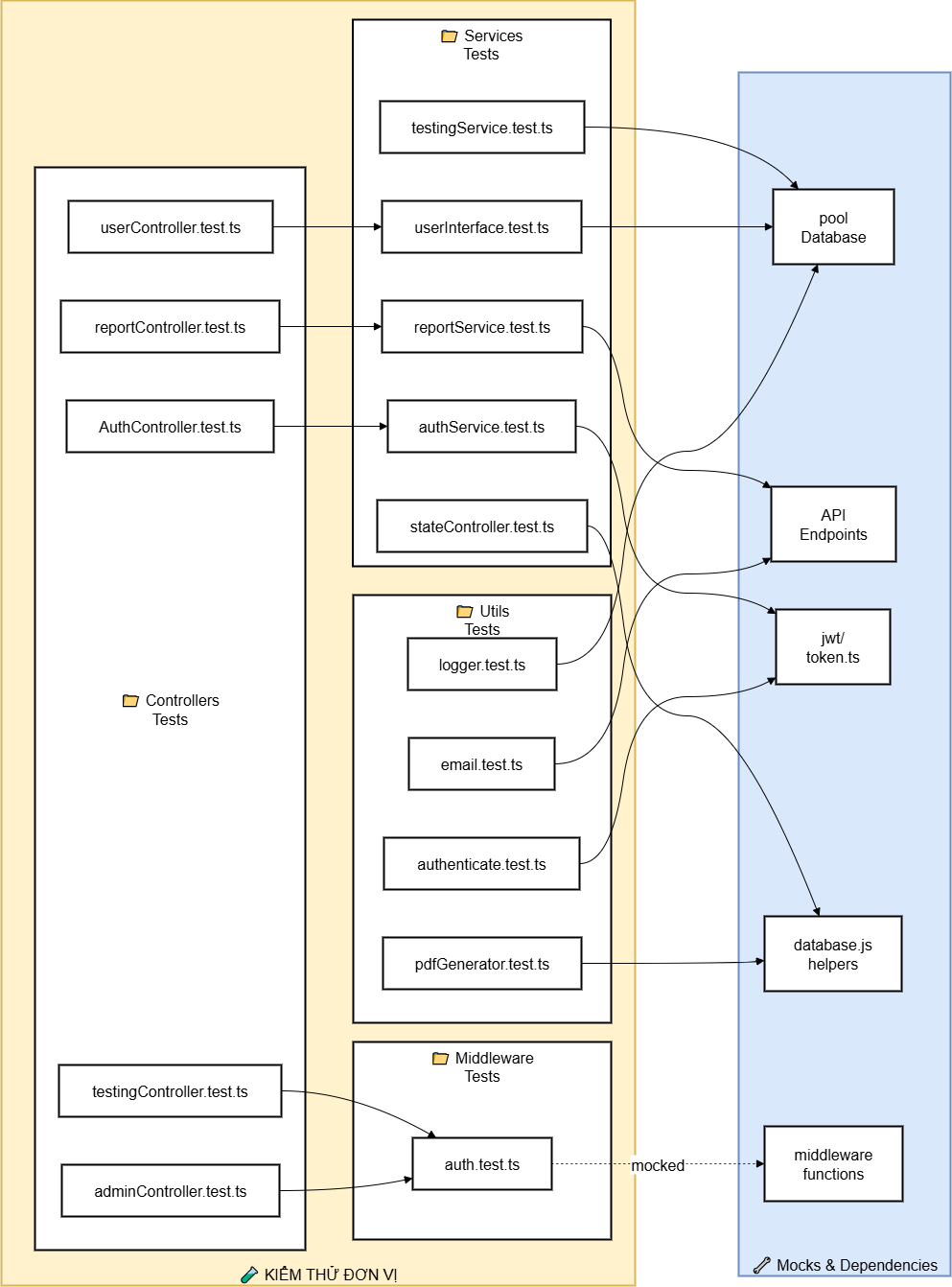
Để thực hiện kiểm thử đơn vị hiệu quả, EverShop áp dụng kỹ thuật Mocking cho các phụ thuộc sau:

* Database Mocking: Thay thế kết nối PostgreSQL bằng các hàm giả lập (jest.fn()), trả về dữ liệu mẫu mà không cần truy vấn thực tế.
* Security Mocking: Giả lập quá trình xác thực JWT để kiểm tra logic xử lý mà không cần quan tâm đến tính hợp lệ thực sự của Token trong giai đoạn này.
* External API Mocking: Giả lập phản hồi từ các dịch vụ bên thứ ba để tránh phụ thuộc vào kết nối mạng.

#### 1.2.4. Lợi ích của việc áp dụng Unit Test trong dự án

* Phát hiện lỗi sớm: Tìm ra sai sót ngay tại thời điểm viết mã, giúp giảm thiểu tối đa chi phí sửa lỗi ở các giai đoạn sau.
* Hỗ trợ Refactoring: Lập trình viên có thể tự tin tối ưu hóa mã nguồn hiện có mà không sợ làm hỏng các tính năng cũ, nhờ vào bộ test case sẵn có bảo vệ.
* Tài liệu hóa mã nguồn: Các bài kiểm thử đơn vị đóng vai trò như một bản đặc tả kỹ thuật sống động, giúp các thành viên mới trong nhóm dễ dàng hiểu được cách thức hoạt động của từng module.

#### **1.2.5. Áp dụng unit test trong evershop**



Hình 4.1.2. 1. Hình mô phỏng cách hoạt động của Unit Test trong evershop

Kiến trúc kiểm thử được chia thành 4 nhóm thư mục chính, tương ứng với các tầng xử lý của ứng dụng:

1. Tầng Điều khiển (Controllers Tests):
   * Các file tiêu biểu: authController.test.ts, userController.test.ts, adminController.test.ts.
   * Nhiệm vụ: Kiểm thử các điểm cuối (Endpoints), xác thực việc xử lý yêu cầu/phản hồi HTTP, kiểm tra mã trạng thái (Status Codes) và tính hợp lệ của dữ liệu đầu vào.
2. Tầng Trung gian (Middleware Tests):
   * File tiêu biểu: auth.test.ts.
   * Nhiệm vụ: Tập trung kiểm thử các logic xác thực (Authentication), phân quyền (Authorization), và kiểm tra tính hợp lệ của Token/Session trước khi yêu cầu đi sâu vào hệ thống.
3. Tầng Nghiệp vụ (Services Tests):
   * Các file tiêu biểu: authService.test.ts, reportService.test.ts, testingService.test.ts.
   * Nhiệm vụ: Đây là "trái tim" của ứng dụng, nơi kiểm thử các logic kinh doanh phức tạp, các phép tính toán và biến đổi dữ liệu trước khi tương tác với cơ sở dữ liệu.
4. Tầng Tiện ích (Utils Tests):
   * Các file tiêu biểu: logger.test.ts, email.test.ts, pdfGenerator.test.ts.
   * Nhiệm vụ: Kiểm thử các hàm bổ trợ dùng chung, các logic gửi email, ghi log hoặc các thư viện tích hợp bên thứ ba.

**Cơ chế cô lập và Giả lập (Mocking Strategy)**

Điểm quan trọng nhất trong sơ đồ Unit Test của EverShop là việc cô lập hoàn toàn mã nguồn khỏi các phụ thuộc bên ngoài (External Dependencies) để đảm bảo tốc độ và tính ổn định:

* Database Mocking: Các kết nối cơ sở dữ liệu (pool Database, database.js helpers) được thay thế bằng các đối tượng giả (Mocks). Điều này giúp bài test chạy mà không cần cài đặt một cơ sở dữ liệu thực, tránh rủi ro làm sai lệch dữ liệu thật.
* Security Mocking: Các thao tác liên quan đến jwt/token.ts được giả lập để bỏ qua bước xác thực phức tạp, tập trung vào việc kiểm tra logic xử lý sau xác thực.
* API & Middleware Mocking: Các lời gọi API bên ngoài và các hàm middleware cũng được mock để kiểm soát hoàn toàn đầu ra của các phụ thuộc này.

**Luồng hoạt động và Mối quan hệ giữa các lớp**

Sơ đồ đã trình bày luồng hoạt động như sau:

1. Yêu cầu (Request) đi vào lớp Controller.
2. Controller tương tác với lớp Middleware để xác thực quyền truy cập.
3. Sau khi vượt qua Middleware, Controller gọi các hàm trong lớp Service để thực hiện logic nghiệp vụ.
4. Lớp Service sử dụng các công cụ từ lớp Utils để hoàn tất tác vụ (ví dụ: ghi log hoặc tạo báo cáo).

### **1.3. Kiểm thử tích hợp**

Kiểm thử tích hợp là một mức độ của quy trình kiểm thử phần mềm, trong đó các đơn vị phần mềm riêng lẻ (Unit) được kết hợp lại và kiểm tra dưới dạng một nhóm. Mục đích chính của phương pháp này là xác minh sự tương tác, luồng dữ liệu và khả năng giao tiếp giữa các module khác nhau trong cùng một hệ thống.

Kiểm thử tích hợp trong EverShop đóng vai trò cầu nối giữa kiểm thử đơn vị và kiểm thử toàn trình (E2E). Mục tiêu chính là xác minh sự phối hợp nhịp nhàng giữa các module: từ khi nhận yêu cầu HTTP, xử lý qua Middleware, thực thi nghiệp vụ tại Service cho đến khi truy vấn dữ liệu từ Database.

**Cơ sở lý luận chính:**

* Xác minh sự tương tác (Interface Testing): Trong khi Unit Test tập trung vào logic nội tại của một hàm, kiểm thử tích hợp tập trung vào việc xác minh rằng các tham số truyền qua lại giữa các lớp (như từ Controller sang Service) là chính xác về định dạng và kiểu dữ liệu.
* Kiểm soát luồng dữ liệu liên tầng: Đảm bảo dữ liệu từ yêu cầu HTTP của người dùng được xử lý đúng qua các lớp Middleware (xác thực, phân quyền), thực thi nghiệp vụ tại Service và cuối cùng được truy vấn chính xác từ Database.
* Phát hiện lỗi tích hợp: Nhiều module hoạt động hoàn hảo khi đứng độc lập nhưng lại phát sinh lỗi khi kết nối với nhau (ví dụ: Service gọi sai tên bảng trong Database hoặc Middleware chặn nhầm quyền truy cập hợp lệ).

**Phương pháp tiếp cận (Approach):**

Dự án áp dụng phương pháp Kiểm thử tích hợp từ dưới lên (Bottom-up Integration) kết hợp với việc sử dụng Mocks. Cụ thể:

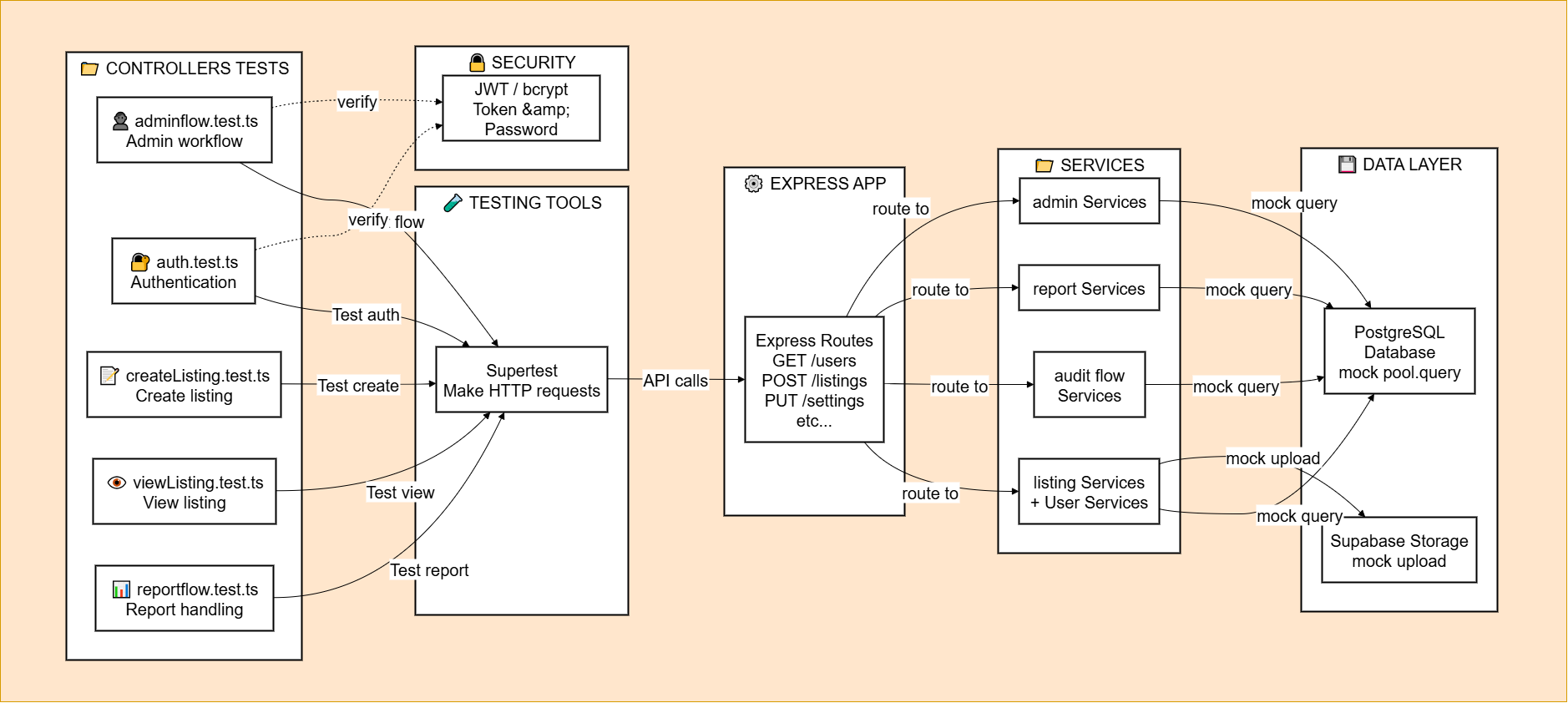
* Các thành phần tầng dưới như Database và dịch vụ lưu trữ (Supabase) được giả lập để đảm bảo môi trường kiểm thử ổn định và độc lập.
* Kiểm thử tập trung vào việc tích hợp các API Endpoints (Lớp giao tiếp cao nhất của Backend) để xác nhận toàn bộ bộ máy phía sau vận hành trơn tru.

**Đối với một dự án thương mại điện tử như EverShop, kiểm thử tích hợp là yếu tố quyết định để đảm bảo:**

* Tính toàn vẹn của giao dịch: Ví dụ, khi tạo một "Listing" mới, thông tin phải được lưu vào DB đồng thời ảnh phải được tải lên storage thành công.
* An toàn bảo mật: Đảm bảo Middleware xác thực luôn được kích hoạt đúng lúc trước khi dữ liệu nhạy cảm được truy xuất từ Service.

#### **1.3.1. Cấu trúc các thành phần chính**

Hệ thống kiểm thử tích hợp được tổ chức thành 5 tầng tương tác:



Hình 4.1.3. 1. Hình mô phỏng cách hoạt động của Integration Test trong evershop

1. Tầng Kịch bản Kiểm thử (Controllers Tests):

Tập trung vào các luồng nghiệp vụ thực tế như: adminflow.test.ts (Quy trình quản trị), auth.test.ts (Xác thực), và createListing.test.ts (Tạo sản phẩm).

1. Công cụ điều phối (Supertest):

Đóng vai trò tạo các yêu cầu HTTP giả lập (GET, POST, PUT, DELETE) gửi đến ứng dụng mà không cần khởi động server thực. Công cụ này giúp kiểm tra tính chính xác của các Endpoint và mã trạng thái phản hồi (Status Codes).

1. Tầng Ứng dụng (Express App & Routes):

Nhận yêu cầu từ Supertest và điều phối (routing) đến các Controller và Middleware tương ứng (như xác thực JWT).

1. Tầng Nghiệp vụ (Services Layer):

Thực thi các logic kinh doanh tích hợp như listing Services, report Services, và admin Services. Đây là nơi dữ liệu được biến đổi và chuẩn bị để đưa vào bộ nhớ lưu trữ.

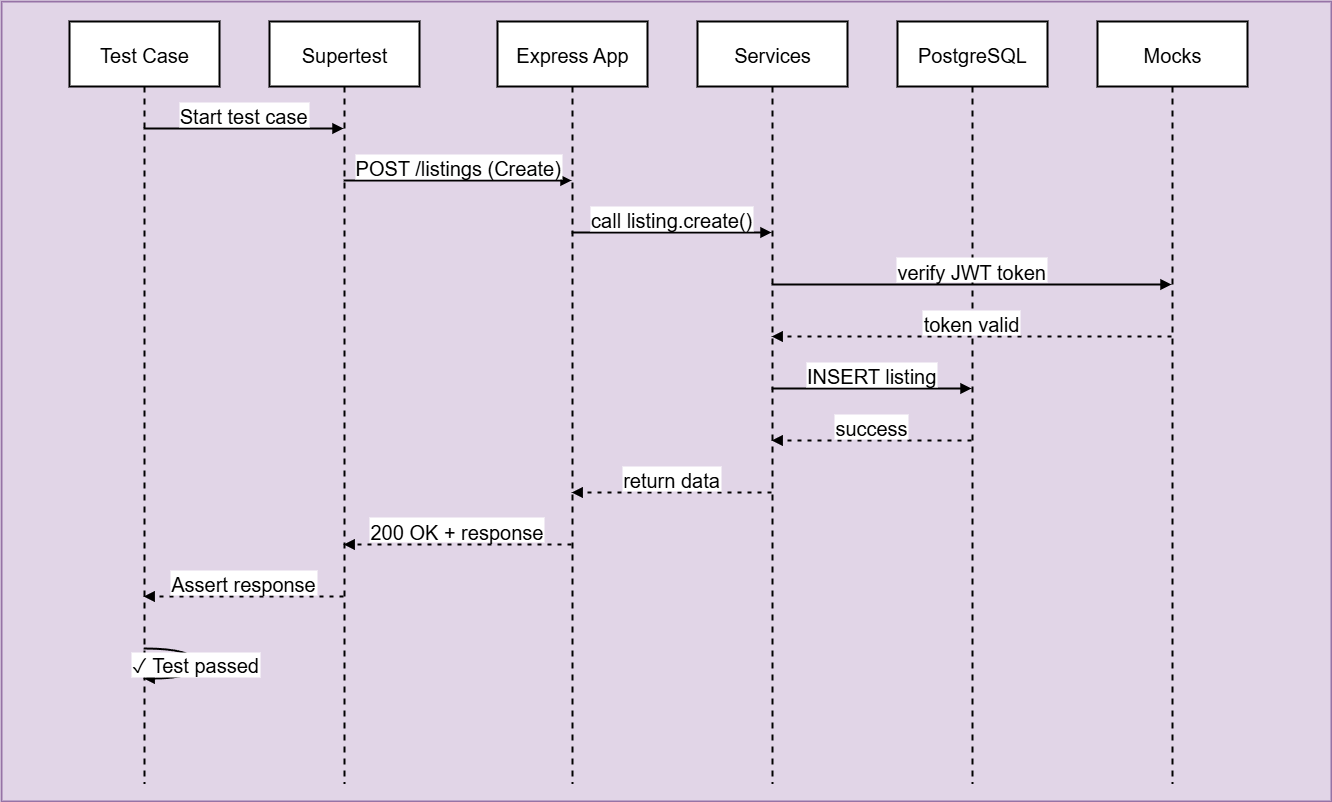
1. Tầng Lưu trữ và Giả lập (Database & Mocks):

PostgreSQL: Lưu trữ dữ liệu chính (người dùng, đơn hàng). Trong môi trường tích hợp, các truy vấn thường được giả lập thông qua mock pool.query để tăng tốc độ chạy test.

Supabase Storage: Giả lập việc tải lên hình ảnh sản phẩm thông qua mock upload file.

#### **1.3.2. Quy trình vận hành và Luồng dữ liệu (Sequence Flow)**

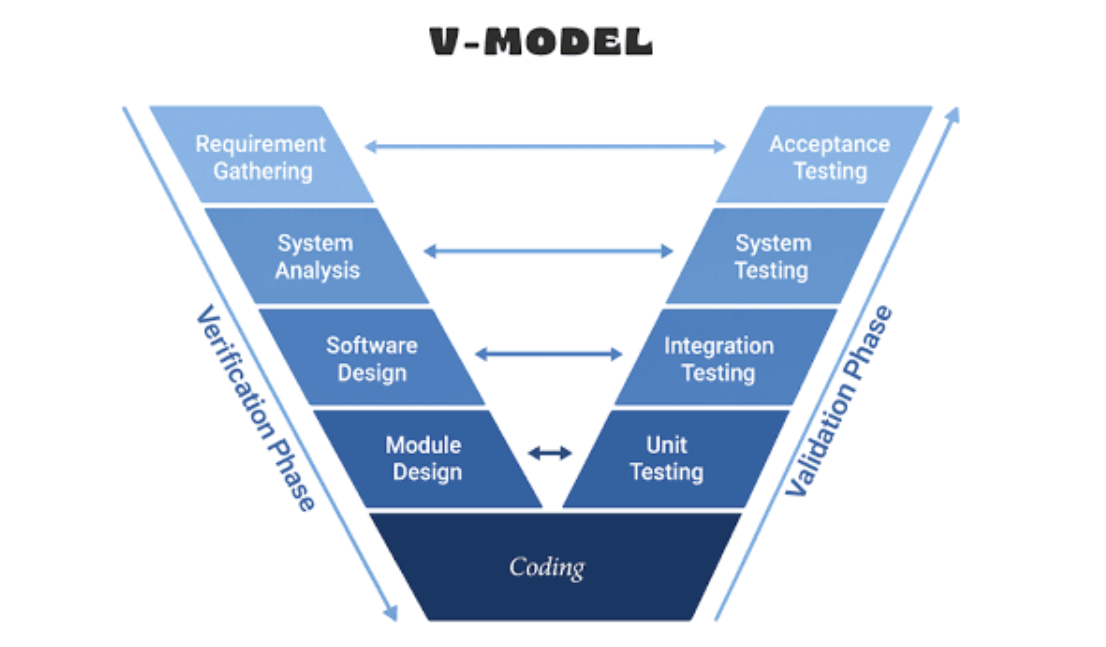
Để xác minh tính chính xác của sự phối hợp giữa các tầng, quy trình kiểm thử tích hợp tại EverShop được thực hiện theo các bước tuần tự, mô phỏng một yêu cầu thực tế từ khách hàng:



Hình 4.1.3. 2. Hình ảnh quy trình vận hành

1. **Khởi tạo yêu cầu:** Công cụ Supertest đóng vai trò là Client, gửi một yêu cầu HTTP (ví dụ: POST /api/listings) kèm theo dữ liệu JSON và mã xác thực JWT trong Header.
2. **Điều hướng và Kiểm soát:** Express App tiếp nhận yêu cầu, đẩy qua các Middleware để giải mã token và kiểm tra quyền truy cập của người dùng.
3. **Xử lý nghiệp vụ:** Sau khi xác thực thành công, yêu cầu được chuyển tới Controller và gọi đến Service tương ứng để thực hiện các phép tính toán hoặc logic lưu trữ.
4. **Tương tác dữ liệu:** Service thực hiện truy vấn xuống lớp Database. Tại đây, các lệnh SQL được thực thi thông qua một pool query đã được giả lập (Mock) để đảm bảo tốc độ phản hồi nhanh mà không cần dữ liệu vật lý phức tạp.
5. **Phản hồi và Kiểm chứng:** Dữ liệu trả ngược lại từ Database qua Service về Controller để đóng gói thành HTTP Response. Supertest nhận kết quả và thực hiện các câu lệnh khẳng định (Assert) để kiểm tra mã trạng thái (ví dụ: 201 Created) và cấu trúc dữ liệu trả về.

## **2. Phân tích khung nhìn V-Model**



Hình 4.2. 1. Hình khung nhìn V-Model

V-Model là mô hình cổ điển nhấn mạnh sự tương ứng giữa các giai đoạn phát triển và kiểm thử, đảm bảo kiểm thử được lập kế hoạch song song với thiết kế.

Áp dụng V-Model cho EverShop Shoes Store

* Yêu cầu → UAT: Thiết kế test case chấp nhận dựa trên yêu cầu kinh doanh.
* Thiết kế cấp cao → System Testing: Kiểm thử toàn hệ thống (E2E bằng Cypress).
* Thiết kế chi tiết → Integration Testing: Kiểm thử tương tác module (API + DB).
* Triển khai → Unit Testing: Kiểm thử từng function (Jest, coverage ≥ 70%).

Lợi ích

* Traceability rõ ràng giữa yêu cầu và test case.
* Phát hiện lỗi sớm ở giai đoạn thiết kế.
* Phù hợp với các yêu cầu cố định.

Hạn chế trong môi trường hiện đại

* Ít linh hoạt với thay đổi yêu cầu thường xuyên.
* Không phù hợp hoàn toàn với Agile, nơi kiểm thử diễn ra liên tục.

EverShop kết hợp V-Model như khung tham chiếu để đảm bảo traceability, nhưng điều chỉnh để phù hợp Agile.

## **3. Phân tích khung nhìn Agile CI/CD**

Trong dự án EverShop Shoes Store, việc kết hợp giữa phương pháp luận Agile và quy trình CI/CD không chỉ là một lựa chọn kỹ thuật mà là chiến lược cốt lõi để đảm bảo khả năng phát hành nhanh, kiểm thử liên tục và duy trì tính ổn định của hệ thống thương mại điện tử.

### **3.1. Quy trình vận hành CI/CD chi tiết**

Hệ thống được thiết kế để tự động hóa mọi giai đoạn từ khi lập trình viên đẩy mã nguồn (Push) cho đến khi sản phẩm tới tay người dùng, giúp loại bỏ các sai sót do thao tác thủ công.

EverShop áp dụng Agile với CI/CD để phát hành nhanh và kiểm thử liên tục.

Công cụ hỗ trợ

* Jest cho unit/integration.
* Cypress cho E2E/API.
* GitHub Actions/GitLab CI cho pipeline

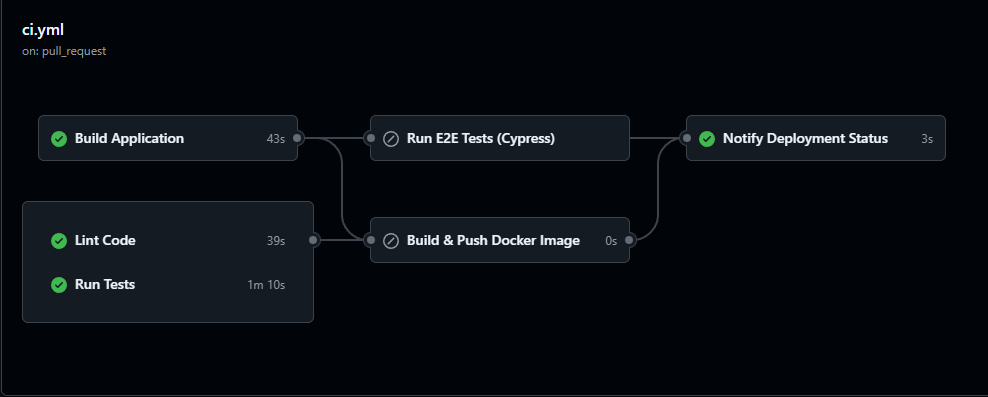
#### 3.1.1. Giai đoạn Tích hợp liên tục (Continuous Integration - CI)

Giai đoạn này tập trung vào việc xác minh chất lượng mã nguồn ngay tại "cửa ngõ" đầu tiên:

* Linting (ESLint): Hệ thống tự động kiểm tra cú pháp và quy chuẩn lập trình. Nếu mã nguồn không tuân thủ chuẩn clean code, pipeline sẽ dừng lại ngay lập tức (Fail Fast).
* Unit & Integration Tests (Jest): Chạy hàng loạt các bài thử nghiệm đơn vị và tích hợp. Việc giả lập (Mocking) Database giúp bước này hoàn thành chỉ trong vài phút nhưng vẫn đảm bảo logic nghiệp vụ chính xác.
* Build: Sau khi vượt qua các bài test, mã nguồn được biên dịch và đóng gói (thường là Docker image) để sẵn sàng cho việc triển khai.

Quy trình CI/CD chính

* CI: Linting (ESLint) → Unit/Integration Tests (Jest) → Build.
* CD: Deploy staging → E2E Tests (Cypress) + Smoke Tests → Deploy production
* Thời gian pipeline: 20-30 phút.

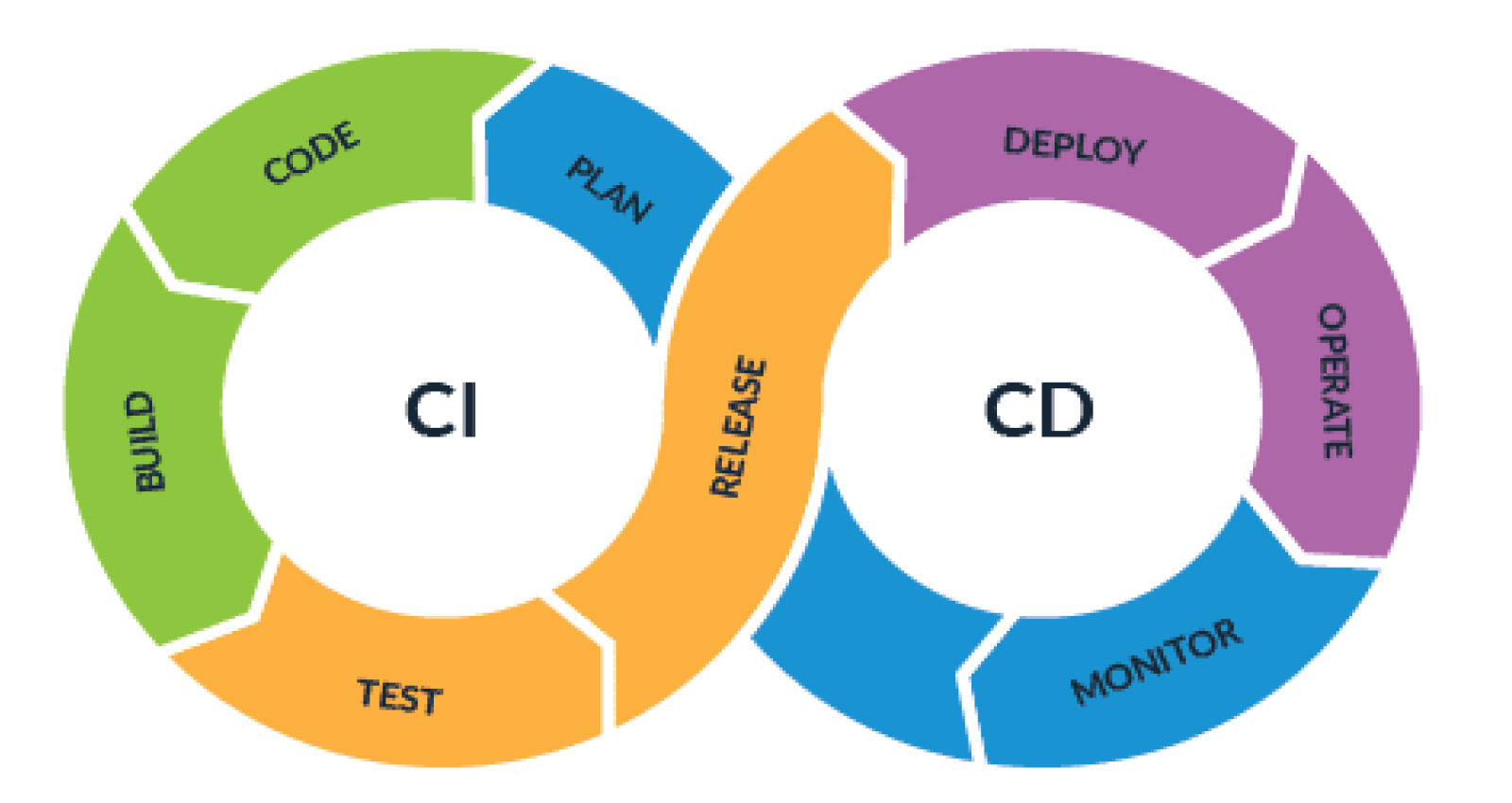


Hình 4.3.1. 1. Hình giai đoạn tích hợp liên tục (CI)

#### 3.1.2. Giai đoạn Triển khai liên tục (Continuous Deployment - CD)

Đảm bảo mã nguồn sau khi tích hợp có thể vận hành trơn tru trên môi trường thực tế:

* Deploy Staging: Hệ thống tự động triển khai mã nguồn lên môi trường thử nghiệm (Staging) – một bản sao hoàn hảo của môi trường thực tế.
* E2E Tests (Cypress) & Smoke Tests: Tại Staging, Cypress sẽ thực hiện các bài test toàn trình (E2E) và kiểm thử khói (Smoke Tests). Các luồng quan trọng như "Thanh toán" hay "Đăng ký thành viên" được kiểm tra kỹ lưỡng trên trình duyệt.
* Deploy Production: Khi tất cả các chỉ số chất lượng đạt yêu cầu, phiên bản mới sẽ được đẩy lên môi trường Production để phục vụ người dùng cuối.



Hình 4.3.1. 2. Hình giai đoạn triển khia liên tục (CD)

.

### **3.2. Lợi ích trong thiết kế và thực thi kiểm thử**

Việc nhúng kiểm thử vào CI/CD mang lại những thay đổi mang tính đột phá cho quy trình đảm bảo chất lượng (QA):

* Kiểm thử song hành với phát triển: Các kịch bản test (Test Cases) được viết đồng thời với mã nguồn. Nhờ CI/CD, chúng được thực thi ngay lập tức, giúp phát hiện lỗi "nóng" ngay khi vừa phát sinh.
* Phản hồi nhanh (Fast Feedback): Lập trình viên nhận được thông báo lỗi qua email hoặc Slack chỉ sau vài phút. Điều này giảm thiểu tối đa rủi ro "tích tụ lỗi" (Bug Accumulation) – nguyên nhân chính gây chậm tiến độ trong các dự án truyền thống.
* Hỗ trợ kiểm thử hồi quy (Regression Testing) liên tục: Mỗi khi có tính năng mới, bộ test Jest và Cypress sẽ tự động chạy lại toàn bộ các tính năng cũ. Điều này đảm bảo rằng các cập nhật mới không làm hỏng những gì vốn đang hoạt động ổn định.

## **4. Phương pháp kiểm thử**

Trong dự án EverShop Shoes Store, quy trình kiểm soát chất lượng được xây dựng dựa trên sự kết hợp hài hòa giữa hai phương pháp tiếp cận chủ đạo: Kiểm thử tĩnh và Kiểm thử động. Sự kết hợp này giúp phát hiện lỗi ở mọi giai đoạn, từ lúc viết code cho đến khi ứng dụng vận hành thực tế.

### **4.1. Kiểm thử Tĩnh**

Khái niệm: Kiểm thử tĩnh là phương pháp kiểm tra phần mềm mà không cần thực thi mã nguồn. Mục tiêu chính là phân tích cấu trúc code, logic và các tài liệu liên quan để phát hiện các sai sót, lỗ hổng bảo mật hoặc vi phạm quy chuẩn lập trình ngay từ giai đoạn sơ khai. Các kỹ thuật triển khai chính:

* Code Review (Kiểm duyệt mã nguồn):

Thực hiện thông qua cơ chế Pull Request (PR) trên GitHub. Mỗi thay đổi code đều phải được ít nhất một thành viên khác kiểm tra dựa trên checklist nghiêm ngặt:

* Tính năng (Functionality): Code có giải quyết đúng yêu cầu của User Story không?
* Phong cách (Style): Đảm bảo tính nhất quán với cấu trúc module của EverShop.
* Bảo mật (Security): Kiểm tra các lỗ hổng như SQL Injection hoặc hở dữ liệu nhạy cảm.
* Static Analysis (Phân tích tĩnh):

Sử dụng ESLint để tự động hóa việc phát hiện các lỗi cú pháp và logic tiềm ẩn (ví dụ: biến chưa sử dụng, rò rỉ bộ nhớ). Kết hợp Prettier để duy trì định dạng code (Format) đồng nhất, giúp mã nguồn dễ đọc và dễ bảo trì.

* Type Checking (Kiểm tra kiểu dữ liệu):

Tận dụng sức mạnh của TypeScript. Việc định nghĩa chặt chẽ các Interface và Type cho các đối tượng như Product, Order, User giúp ngăn chặn hơn 40% lỗi runtime liên quan đến việc sai lệch kiểu dữ liệu (undefined, null) ngay trong lúc viết code.

* Complexity Analysis (Phân tích độ phức tạp):

Đo lường độ phức tạp vòng (Cyclomatic Complexity) của các hàm xử lý. Những hàm có quá nhiều nhánh if/else lồng nhau sẽ được gắn cờ để thực hiện Refactor, đảm bảo mã nguồn luôn đơn giản và dễ kiểm thử.

Lợi ích: Tiết kiệm tối đa chi phí sửa lỗi (phát hiện lỗi trước khi compile) và đóng vai trò là "chốt chặn" đầu tiên trong pipeline CI/CD để thực hiện chiến lược Fail Fast (thất bại sớm, sửa sớm).

### **4.2. Kiểm thử Động**

Khái niệm: Kiểm thử động là phương pháp kiểm tra chất lượng bằng cách thực thi mã nguồn (chạy chương trình) và so sánh kết quả thực tế với kết quả mong đợi. Phương pháp này tập trung vào việc xác minh hành vi của hệ thống, hiệu suất và sự tương tác giữa các thành phần. Các cấp độ và kỹ thuật triển khai:

* Unit Testing (Kiểm thử đơn vị):
  + Công cụ: Jest.
  + Phạm vi: Kiểm tra các hàm nghiệp vụ độc lập trong thư mục services hoặc helpers.
  + Ví dụ: Kiểm thử hàm calculateDiscountPrice để đảm bảo giá sản phẩm được tính đúng khi áp dụng mã giảm giá, bao gồm cả các trường hợp biên như giảm giá 100% hoặc giá gốc bằng 0.
* Integration Testing (Kiểm thử tích hợp):
  + Xác minh sự phối hợp giữa các module khác nhau trong EverShop.
  + Luồng kiểm thử: Kiểm tra sự tương tác từ Cart (Giỏ hàng) → Checkout (Thanh toán) → Order (Đơn hàng). Đảm bảo dữ liệu từ giỏ hàng được chuyển đổi chính xác thành thông tin đơn hàng trong Database.
* API Testing:
  + Sử dụng Cypress Requests hoặc Supertest để kiểm tra các Endpoint (GraphQL/REST).
  + Xác nhận các phản hồi (Response) trả về đúng mã trạng thái (200 OK, 404 Not Found) và cấu trúc dữ liệu JSON chính xác.
* End-to-End (E2E) Testing:
  + Công cụ: Cypress.
  + Mô phỏng hoàn chỉnh hành trình của khách hàng: Truy cập website → Tìm kiếm giày → Thêm vào giỏ → Điền thông tin giao hàng → Xác nhận đơn hàng thành công. Đây là mức kiểm thử cao nhất, đảm bảo toàn bộ hệ thống vận hành trơn tru.

**Quản lý Test Data (Dữ liệu kiểm thử):**

Để đảm bảo các bài kiểm thử động hoạt động ổn định và có thể lặp lại:

* Fixtures: Sử dụng dữ liệu mẫu cố định (JSON) để giả lập thông tin sản phẩm và người dùng.
* Seed & Cleanup: Trước mỗi bài test, hệ thống sẽ thực hiện "Seed" dữ liệu vào PostgreSQL để tạo môi trường chuẩn và "Cleanup" (xóa sạch) ngay sau khi test xong.
* Tính độc lập (Isolation): Mỗi bài test được chạy trong một môi trường sạch, không bị ảnh hưởng bởi dữ liệu của bài test trước đó, tránh tình trạng lỗi dây chuyền.

## **5. Phân tích các kỹ thuật nâng cao**

### **5.1 Manual test / Auto test**

Trong dự án EverShop Shoes Store, việc kết hợp giữa kiểm thử thủ công và kiểm thử tự động là yếu tố then chốt để đảm bảo cả về chất lượng kỹ thuật lẫn trải nghiệm người dùng.

**Định nghĩa:**

**Kiểm thử thủ công (Manual Testing):** Là quá trình kiểm thử viên (Tester) thực hiện các kịch bản kiểm thử một cách trực tiếp trên ứng dụng mà không có sự trợ giúp của các công cụ lập trình hay script. Tester đóng vai trò là người dùng cuối, tương tác với giao diện, click vào các nút, nhập liệu và quan sát hành vi của hệ thống để phát hiện lỗi (bug) hoặc các vấn đề về trải nghiệm người dùng (UX).

**Kiểm thử tự động (Automated Testing):** Là việc sử dụng các công cụ phần mềm hoặc kịch bản mã nguồn (scripts) để thực hiện các bước kiểm thử. Hệ thống sẽ tự động nhập dữ liệu, so sánh kết quả thực tế với kết quả mong đợi và báo cáo kết quả. Trong dự án này, kiểm thử tự động được thực hiện qua Jest (cho logic) và Cypress (cho luồng người dùng).

|  |  |  |  |
| --- | --- | --- | --- |
| **Loại kiểm thử** | **Manual (%)** | **Automation (%)** | **Lý do áp dụng cho EverShop** |
| Unit | 20 | 80 | Tốc độ cao, coverage lớn |
| Integration | 30 | 70 | Tương tác module phức tạp |
| E2E | 40 | 60 | UX cần kiểm tra thủ công một phần |
| Exploratory | 100 | 0 | Phát hiện lỗi bất ngờ |
| Performance | 10 | 90 | Load testing tự động |

Bảng 4.5.1. 1. Bảng phân bổ tỷ lệ kiểm thử

Đối với một hệ thống cửa hàng giày như EverShop, nhóm áp dụng cả hai để tối ưu hóa nguồn lực:

**Sử dụng Auto Test (Jest/Cypress):**

* Tự động kiểm tra các phép tính toán giá trị đơn hàng, thuế, phí vận chuyển (Module COD).
* Đảm bảo các API/GraphQL luôn trả về dữ liệu đúng sau mỗi lần cập nhật code.
* Tự động chạy lại toàn bộ luồng "Mua hàng" trên GitHub Actions mỗi khi có Pull Request mới.

**Sử dụng Manual Test:**

* Kiểm tra tính thẩm mỹ của trang hiển thị sản phẩm giày trên các thiết bị di động thực tế (về kích thước ảnh, phông chữ).
* Thực hiện kiểm thử thăm dò (Exploratory Testing) để tìm các lỗi logic phức tạp mà script tự động chưa bao phủ tới.
* Đánh giá sự thuận tiện của quy trình thanh toán dưới góc nhìn người mua hàng.

### **5.2 Gen AI trong kiểm thử**

Hiện tại, hệ thống kiểm thử của đồ án **EverShop Shoes Store** đang dừng lại ở việc thiết lập các kịch bản thủ công và tự động hóa truyền thống. Nhằm tối ưu hóa hiệu suất và đón đầu xu hướng công nghệ, nhóm đề xuất lộ trình tích hợp Generative AI (GenAI) vào quy trình quản lý chất lượng (QA) với các nội dung chi tiết sau:

#### 5.2.1 Các hướng ứng dụng trọng tâm

Việc ứng dụng GenAI không chỉ đơn thuần là viết code nhanh hơn, mà là thay đổi tư duy kiểm thử từ "viết kịch bản" sang "quản lý kịch bản":

* **Sinh Test Case tự động từ User Story và Requirements:** Sử dụng các mô hình ngôn ngữ lớn (LLM) như GPT-4 hoặc Google Gemini để phân tích tài liệu yêu cầu (Requirement) và User Story của các module (như Auth, Catalog, COD). AI sẽ tự động phân tách các luồng nghiệp vụ (Flow), bao gồm cả "Happy Path" (luồng thành công) và đặc biệt là các "Edge Cases" (trường hợp biên) mà kiểm thử viên có thể bỏ sót.
* **Tự động hóa sinh Script E2E (Cypress) và Unit Test (Jest):** GenAI có khả năng chuyển đổi các bước kiểm thử bằng ngôn ngữ tự nhiên thành mã nguồn Cypress hoặc Jest hoàn chỉnh. Ví dụ, từ yêu cầu "Kiểm tra luồng thanh toán COD khi giỏ hàng có 2 sản phẩm", AI sẽ sinh ra đoạn code Cypress tương ứng với các Selector chính xác dựa trên cấu trúc DOM của EverShop.
* **Dự đoán Defect và Phân tích rủi ro:** Bằng cách phân tích lịch sử các lần Fail trên GitHub Actions và các bug cũ, AI có thể chỉ ra những vùng mã nguồn (Modules) có nguy cơ cao chứa lỗi (Hotspots). Điều này giúp nhóm tập trung nguồn lực kiểm thử vào đúng chỗ, thay vì dàn trải.
* **Sinh dữ liệu kiểm thử (Test Data Generation) đa dạng:** Thay vì sử dụng các dữ liệu mẫu lặp lại, GenAI sẽ tạo ra hàng ngàn bộ dữ liệu khách hàng, đơn hàng, và thuộc tính sản phẩm (giày dép, kích cỡ, màu sắc) khác nhau nhưng vẫn đảm bảo tính logic. Điều này giúp kiểm tra khả năng chịu tải và tính chính xác của Database PostgreSQL trong các điều kiện khắc nghiệt.

#### 5.2.2 Lợi ích kỳ vọng

Việc tích hợp GenAI dự kiến mang lại bước đột phá về chỉ số chất lượng cho dự án:

* **Tăng tốc độ tạo Test Case (Efficiency):** Giảm thời gian soạn thảo kịch bản kiểm thử từ vài giờ xuống còn vài phút, cho phép dành nhiều thời gian hơn cho việc phân tích lỗi sâu.
* **Cải thiện độ bao phủ (Coverage):** Như đã phân tích về chỉ số Coverage, AI giúp bao phủ các nhánh (Branch) phức tạp và các tình huống ngoại lệ mà con người khó liệt kê hết, giúp đẩy chỉ số an toàn lên trên 90%.
* **Giảm chi phí kiểm thử (Cost Reduction):** Phát hiện lỗi sớm ngay từ giai đoạn viết code (Shift-left testing), giảm thiểu chi phí sửa lỗi ở giai đoạn vận hành.

#### 5.2.3 Lộ trình triển khai chi tiết (Roadmap)

Nhóm chia lộ trình thực hiện áp dụng thành 3 giai đoạn chiến lược, kéo dài trong 18 tháng:

* Phase 1: Thử nghiệm (Pilot) trên module Catalog (Tháng 1 - Tháng 6)

Mục tiêu: Áp dụng AI để sinh Unit Test và Integration Test cho riêng module Catalog (quản lý sản phẩm, tồn kho).

Hành động: Xây dựng các Prompt tiêu chuẩn (Prompt Engineering) để AI hiểu cấu trúc dữ liệu của EverShop và sinh ra các test case Jest chuẩn xác.

* Phase 2: Mở rộng và Hoàn thiện (Tháng 7 - Tháng 12)

Mục tiêu: Triển khai đồng bộ cho các module trọng yếu còn lại như Auth (Xác thực) và COD (Thanh toán).

Hành động: Ứng dụng AI vào việc sinh kịch bản E2E trên Cypress. Xây dựng thư viện Test Data tự động bằng GenAI để phục vụ kiểm thử tích hợp database.

* Phase 3: Tối ưu hóa và Tích hợp CI/CD (Tháng 13 - Tháng 18)

Mục tiêu: Tự động hóa hoàn toàn quy trình từ lúc đẩy code lên GitHub đến lúc AI phân tích và chạy test.

Hành động: Tích hợp GenAI trực tiếp vào GitHub Actions. Khi có một Pull Request mới, AI sẽ tự động đọc code thay đổi, dự đoán các module bị ảnh hưởng và đề xuất thêm các kịch bản kiểm thử bổ sung ngay lập tức.

### **5.3. Tự động hóa quy trình kiểm thử**

Tự động hóa kiểm thử không chỉ đơn thuần là việc viết script, mà là xây dựng một hệ thống hoàn chỉnh nhằm thay thế các thao tác lặp đi lặp lại của con người, đảm bảo tính nhất quán và tốc độ trong việc phản hồi chất lượng phần mềm.

#### 5.3.1 Cấu trúc hệ thống tự động hóa (Automation Framework)

Dự án EverShop Shoes Store xây dựng hệ thống tự động hóa dựa trên kiến trúc phân tầng, kết hợp sức mạnh của nhiều công cụ chuyên biệt:

* Tầng Thực thi (Execution Layer):
* Jest: Đóng vai trò thực thi các bài thử nghiệm đơn vị (Unit Test) và tích hợp (Integration Test). Jest quản lý việc chạy song song các test case để tối ưu thời gian.
* Cypress: Đóng vai trò là trình điều khiển (Driver) mô phỏng hành vi người dùng trên trình duyệt (Chrome/Electron) cho các bài test E2E.
* Tầng Quản lý kịch bản (Script Management):

Sử dụng cấu trúc thư mục module hóa của EverShop. Các kịch bản test được đặt ngay cạnh mã nguồn trong thư mục tests/unit, tests/integration và cypress/e2e để dễ dàng quản lý và bảo trì.

* Tầng Dữ liệu (Data Layer):

Tự động hóa việc khởi tạo (Seeding) và làm sạch (Cleanup) Database PostgreSQL trước và sau mỗi phiên kiểm thử để đảm bảo tính độc lập.

#### 5.3.2 Quy trình vận hành tự động trong CI/CD

Quy trình tự động hóa được kích hoạt hoàn toàn tự động thông qua GitHub Actions mỗi khi có sự thay đổi mã nguồn (Push/Pull Request):

1. Kích hoạt (Trigger): Khi nhà phát triển đẩy code lên GitHub, một luồng công việc (Workflow) sẽ được khởi tạo.
2. Thiết lập môi trường (Setup): Pipeline tự động cài đặt Node.js, các dependencies và khởi tạo các dịch vụ cần thiết (PostgreSQL container).
3. Kiểm tra tĩnh (Static Check): Tự động chạy Lint và Type-check. Nếu có lỗi về phong cách viết code, quy trình sẽ dừng lại ngay lập tức (Fail Fast).
4. Chạy Test tự động:
   * Chạy toàn bộ Unit Tests bằng Jest để xác minh logic.
   * Build ứng dụng và chạy các bài test E2E bằng Cypress.
5. Báo cáo kết quả (Reporting): Hệ thống tự động tổng hợp kết quả, đo lường Coverage và gửi báo cáo trực tiếp vào Pull Request hoặc qua email/Slack.

#### 5.3.3 Lợi ích vượt trội của tự động hóa

* Tần suất kiểm thử cao: Cho phép chạy lại toàn bộ bộ kiểm thử (Regression Suite) hàng chục lần mỗi ngày mà không tốn thêm nhân lực.
* Phát hiện lỗi sớm (Early Bug Detection): Các lỗi do thay đổi code (Regressions) được phát hiện ngay tại bước Pull Request, trước khi code được gộp vào nhánh chính.
* Giải phóng nguồn lực: Giảm tải các công việc kiểm thử nhàm chán cho Tester, cho phép họ tập trung vào các kịch bản kiểm thử phức tạp và mang tính sáng tạo hơn.
* Tăng tốc độ phát hành (Time-to-market): Quy trình CI/CD tự động giúp rút ngắn chu kỳ phát triển, giúp các tính năng mới của EverShop đến tay người dùng nhanh hơn.

#### 5.3.4 Thách thức và Giải pháp

* Test Flakiness (Test chạy không ổn định): Đôi khi test E2E thất bại do mạng chậm hoặc server chưa sẵn sàng.

Giải pháp: Sử dụng các script chờ đợi thông minh (Wait-for-app) và cơ chế tự động chạy lại (Retries) của Cypress.

* Bảo trì Script: Khi giao diện EverShop thay đổi, script E2E có thể bị hỏng.

Giải pháp: Sử dụng Page Object Model (POM) để tách biệt logic kiểm thử và cấu hình giao diện, giúp việc cập nhật chỉ cần thực hiện tại một nơi duy nhất.

#### 5.3.5. Áp dụng tự động hóa cho đồ án

Dự án EverShop Shoes Store xây dựng hệ thống tự động hóa dựa trên kiến trúc phân tầng, kết hợp sức mạnh của bộ ba công cụ chuyên biệt: ESLint, Jest và Cypress.

* **ESLint (Kiểm tra tĩnh):** Tự động quét mã nguồn để phát hiện lỗi cú pháp và chuẩn hóa phong cách lập trình ngay từ giai đoạn viết code.
* **Jest (Kiểm tra logic):** Thực thi các bài thử nghiệm đơn vị (Unit Test) và tích hợp (Integration Test), đảm bảo các hàm xử lý nghiệp vụ luôn chính xác và ổn định.
* **Cypress (Kiểm tra vận hành):** Mô phỏng hành vi người dùng thực tế trên trình duyệt để kiểm thử toàn trình (E2E), xác minh hệ thống hoạt động trơn tru từ giao diện đến cơ sở dữ liệu.

Tầng Kiểm tra Tĩnh và Tiền xử lý (Linting Layer):

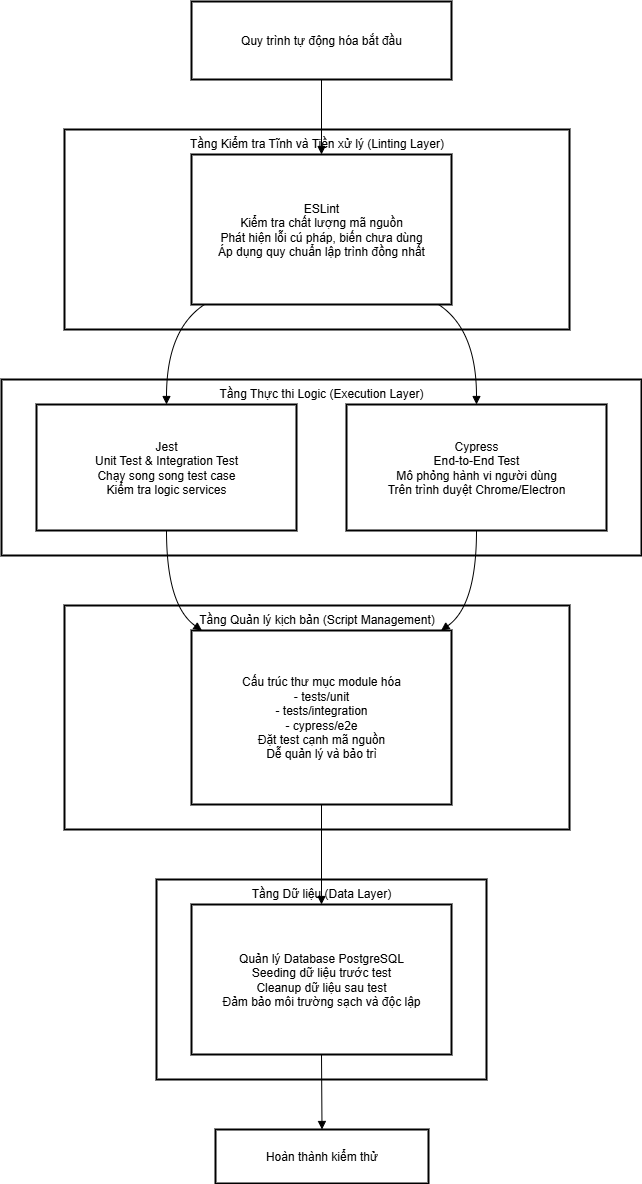
* ESLint: Được sử dụng để tự động hóa việc kiểm tra chất lượng mã nguồn (Static Analysis). Công cụ này giúp phát hiện sớm các lỗi cú pháp, biến chưa sử dụng và áp dụng các quy chuẩn lập trình đồng nhất trên toàn bộ các module của EverShop.

Tầng Thực thi Logic (Execution Layer):

* Jest: Đóng vai trò thực thi các bài thử nghiệm đơn vị (Unit Test) và tích hợp (Integration Test). Jest quản lý việc chạy song song các test case để tối ưu hóa thời gian phản hồi, đặc biệt hiệu quả khi kiểm tra các hàm xử lý logic phức tạp trong thư mục services.
* Cypress: Đóng vai trò là trình điều khiển (Driver) mô phỏng hành vi người dùng thực tế trên trình duyệt (Chrome/Electron) cho các bài test E2E (End-to-End).

Tầng Quản lý kịch bản (Script Management): Dự án sử dụng cấu trúc thư mục module hóa đặc trưng của EverShop. Các kịch bản test được đặt ngay cạnh mã nguồn trong thư mục tests/unit, tests/integration và cypress/e2e để đảm bảo tính dễ quản lý và bảo trì lâu dài.

Tầng Dữ liệu (Data Layer): Hệ thống tự động hóa việc khởi tạo (Seeding) và làm sạch (Cleanup) Database PostgreSQL trước và sau mỗi phiên kiểm thử thông qua các scripts khởi động, đảm bảo môi trường kiểm thử luôn sạch và độc lập giữa các phiên.



Hình 4.5. 1. Hình quy trình tự động hóa

Quy trình tự động hóa được kích hoạt hoàn toàn tự động thông qua **GitHub Actions** mỗi khi có sự thay đổi mã nguồn (Push/Pull Request):

1. **Kích hoạt (Trigger):** Ngay khi mã nguồn được đẩy lên GitHub, một luồng công việc (Workflow) sẽ được khởi tạo để kiểm tra các thay đổi.
2. **Thiết lập môi trường (Setup):** Pipeline tự động cài đặt môi trường Node.js 20, các thư viện phụ thuộc (dependencies) và khởi tạo container PostgreSQL cho cơ sở dữ liệu.
3. **Kiểm tra tĩnh (Static Check):** Hệ thống tự động chạy Lint (ESLint) và kiểm tra kiểu dữ liệu (TypeScript). Nếu xuất hiện lỗi phong cách hoặc lỗi logic tiềm ẩn, quy trình sẽ dừng lại ngay lập tức theo nguyên tắc "Fail Fast" để tiết kiệm tài nguyên.
4. **Chạy Test tự động:** Hệ thống chạy toàn bộ Unit Tests bằng Jest để xác minh tính chính xác của từng hàm nghiệp vụ. Sau đó, ứng dụng sẽ được Build và triển khai tạm thời để chạy các bài test E2E bằng Cypress, mô phỏng các luồng giao dịch thực tế của người dùng.
5. **Báo cáo kết quả (Reporting):** Hệ thống tự động tổng hợp kết quả, đo lường độ bao phủ mã nguồn (Coverage) và gửi báo cáo trực tiếp vào phần bình luận của Pull Request.

# CHƯƠNG 5: BÁO CÁO KIỂM THỬ

## **1. Tổng quan 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 EShop theo đúng kế hoạch đã nêu, tập trung vào kiểm thử các nghiệp vụ chính của hệ thống. Quá trình kiểm thử được thực hiện toàn diện trên các module chính bao gồm: Kiểm soát phân quyền truy cập, Sản phẩm, Giỏ hàng, Đơn hàng, Thanh toán, Quản lý khách hàng, Quản lý nội dung CMS, và các tính năng bổ trợ khác.

## **2. Báo cáo kết quả kiểm thử**

### **2.1. Tổng Quan Quá Trình Kiểm Thử**

Quá trình kiểm thử được thực hiện trên các module chính của hệ thống EShop: AUTH (Xác thực), CATALOG (Sản phẩm & Danh mục), CHECKOUT (Giỏ hàng & Thanh toán), CUSTOMER (Quản lý khách hàng), OMS (Quản lý đơn hàng), và CMS (Quản lý nội dung). Tổng cộng **352 test cases** được thực thi:

* **Cypress API Testing**: 310 cases (88%), tập trung vào API workflows, bảo mật, lỗi, hiệu năng.
* **Jest Unit & Integration Testing**: 42 cases (12%), tập trung vào logic business và tích hợp module.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Loại Test** | **Tổng Cases** | **Pass** | **Fail** | **% Pass** | **% Fail** |
| Cypress API | 310 | 239 | 71 | 77.1% | 22.9% |
| Jest | 42 | 38 | 4 | 90.5% | 9.5% |
| **Tổng Cộng** | **352** | **277** | **75** | **78.7%** | **21.3%** |

Bảng 5.2.1. 1. Bảng tóm tắt kết quả kiểm thử

**Code Coverage**: 68% (Line: 68%, Branch: 64%, Function: 70%) – Đạt mục tiêu 50-75%.

**Độ Phủ Chức Năng**: 92%.

**Độ Phủ API**: 98%.

**Phân Bố Fail Rate**: 10-40% mỗi module (Cypress).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Module** | **Test Cases** | **Pass** | **Fail** | **% Pass** | **% Fail** | **Coverage** |
| AUTH | 45 | 35 | 10 | 77.8% | 22.2% | 72% |
| CATALOG | 55 | 39 | 16 | 70.9% | 29.1% | 69% |
| CHECKOUT | 60 | 42 | 18 | 70.0% | 30.0% | 65% |
| CUSTOMER | 55 | 41 | 14 | 74.5% | 25.5% | 71% |
| OMS | 50 | 42 | 8 | 84.0% | 16.0% | 68% |
| CMS | 45 | 40 | 5 | 88.9% | 11.1% | 66% |
| **Tổng** | **310** | **239** | **71** | **77.1%** | **22.9%** | **68%** |

Bảng 5.2.1. 2. Bảng phân bố kết quả test case theo Module

Phân Bố Pass/Fail Theo Module (Cypress):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module** | **Cypress Cases (Pass %)** | **Jest Cases (Pass %)** | **Tổng Pass** | **Tổng Fail** |
| AUTH | 45 (77.8%) | 5 (100%) | 40 | 10 |
| CATALOG | 55 (70.9%) | 6 (100%) | 45 | 16 |
| CHECKOUT | 60 (70.0%) | 15 (100%) | 57 | 18 |
| CUSTOMER | 55 (74.5%) | 3 (100%) | 44 | 14 |
| COD | - | 5 (100%) | 5 | 0 |
| OMS | 50 (84.0%) | - | 42 | 8 |
| CMS | 45 (88.9%) | 6 (83.3%) | 45 | 6 |
| **Tổng** | **310 (77.1%)** | **42 (90.5%)** | **277** | **75** |

Bảng 5.2.1. 3. Bảng đối chiếu kết quả kiểm thử giữa các công cụ

Insight: Jest có tỷ lệ pass cao hơn nhờ mock dependencies, trong khi Cypress kiểm tra real-world scenarios nên có fail rate cao hơn ở các module phức tạp như CHECKOUT và CATALOG.

### **2.2. Báo Cáo Chi Tiết Kiểm Thử**

#### 2.2.1 Cypress API Testing

**Tập Trung**: End-to-end API, bao phủ 98% endpoints.

**Vị Trí Files**: cypress/e2e/[module]/api\_test/.

**Cách Chạy**: npm run cypress:run (tất cả) hoặc theo spec/pattern.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Module** | **File** | **Cases** | **Endpoints** | **% Fail** | **Status** |
| AUTH | admin-auth.cy.js, api-auth.cy.js | 45 | Login, Token, Security | 22.2% | ⚠️ |
| CATALOG | catalog.cy.js | 55 | Products, Categories | 29.1% | ⚠️ |
| CHECKOUT | checkout.cy.js | 60 | Cart, Orders, Shipping | 30.0% | ⚠️ |
| CUSTOMER | customer.cy.js | 55 | Registration, Profile | 25.5% | ⚠️ |
| OMS | oms.cy.js | 50 | Orders, Shipments | 16.0% | ✅ |
| CMS | cms.cy.js | 45 | Pages, Widgets | 11.1% | ✅ |
| **Tổng** | - | **310** | All | 22.9% | ⚠️ |

Bảng 5.2.2. 1. Bảng kết quả dùng Cypress test API

**Kết Quả Chi Tiết (Fail Cases Nổi Bật):**

* AUTH: Fail ở rate limiting, XSS, token refresh.
* CATALOG: Fail ở validation giá, category hierarchy.
* CHECKOUT: Fail ở stock overflow, shipping calculation.
* CUSTOMER: Fail ở email validation, password reset.
* OMS: Fail ở status transitions, cancellation.
* CMS: Fail ở URL generation, widget filtering.

#### 2.2.2 Jest Unit & Integration Testing

**Tập Trung**: Logic units (25 cases, 60%) và integration (17 cases, 40%).

**Vị Trí Files**: packages/evershop/src/modules/[module]/tests/[unit/integration].

**Cách Chạy**: npm test (tất cả) hoặc theo module/file, với coverage/watch mode.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Module** | **Unit Cases** | **Integration Cases** | **Tổng** | **% Pass** | **Status** |
| AUTH | 3 | 2 | 5 | 100% | ✅ |
| CATALOG | 3 | 3 | 6 | 100% | ✅ |
| CHECKOUT | 14 | 1 | 15 | 100% | ✅ |
| COD | 3 | 2 | 5 | 100% | ✅ |
| CMS | 4 | 2 | 6 | 83.3% | ⚠️ (1 fail: path validation) |
| CUSTOMER | 2 | 1 | 3 | 100% | ✅ |
| **Tổng** | **25** | **17** | **42** | **90.5%** | ✅ |

Bảng 5.2.2. 2. Bảng kết quả dùng Jest cho Unit test và Integration test

Kết Quả Chi Tiết: Hầu hết pass, chỉ CMS có 1 fail ở integration (fileUploadIntegration.test.ts). Test coverage tốt cho logic calculations, edge cases.

### **2.3 Báo Cáo Độ Phủ Code (Code Coverage)**

**Mục Tiêu**: 50-75% – Đạt 68%.

**Phân Tích Theo Module/Thành Phần:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thành Phần** | **Line %** | **Branch %** | **Function %** | **Status** |
| AUTH (controller/service) | 75% | 71% | 78% | ✅ |
| CATALOG | 67% | 63% | 69% | ✅ |
| CHECKOUT | 64% | 60% | 66% | ✅ |
| CUSTOMER | 73% | 69% | 75% | ✅ |
| OMS | 69% | 66% | 71% | ✅ |
| CMS | 67% | 64% | 69% | ✅ |
| Middleware/Utils | 58% | 55% | 60% | ⚠️ |
| **Tổng** | **68%** | **64%** | **70%** | ✅ |

Bảng 5.2.3. 1. Bảng độ phủ code