TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

————o&o————

BÁO CÁO MÔN HỌC KIỂM THỬ PHẦN MỀM

PHẦN MỀM KIỂM THỬ: WEBSITE PROSHOP

Mã học phần: 841408

Giảng viên phụ trách: ThS. Từ Lãng Phiêu

Nhóm: 15

Năm học: 2025-2026

Sinh viên thực hiện:

Hoàng Sỹ Khiêm - MSSV: 3121410263

TP. HỒ CHÍ MINH, NGÀY 24 THÁNG 11 NĂM 2025

LỜI CẢM ƠN

Để hoàn thành báo cáo đồ án môn học này, nhóm em đã nhận được sự hỗ trợ, hướng dẫn và giúp đỡ vô cùng quý báu từ nhiều cá nhân và tổ chức.

Đầu tiên và quan trọng nhất, em xin bày tỏ lòng biết ơn sâu sắc tới Giảng viên hướng dẫn – **ThS. Từ Lãng Phiêu**, người đã dành thời gian, tâm huyết và kiến thức chuyên môn để tận tình hướng dẫn, chỉ bảo, đồng hành cùng nhóm em từ những bước đi đầu tiên cho đến khi hoàn thành đồ án. Sự định hướng, những góp ý sắc bén và sự động viên kịp thời của Thầy chính là yếu tố then chốt giúp em vượt qua khó khăn và đạt được kết quả như ngày hôm nay.

em cũng xin chân thành cảm ơn Quý Thầy, Cô trong **Trường Đại học Sài Gòn** nói chung, và đặc biệt là Quý Thầy, Cô trong **Khoa Công nghệ thông tin** nói riêng. Những kiến thức nền tảng từ các môn đại cương và kiến thức chuyên ngành sâu sắc từ các môn học về **Kiểm thử phần mềm** mà Thầy/Cô đã truyền đạt chính là cơ sở lý luận và thực tiễn vững chắc để em có đủ năng lực thực hiện đề tài này. em cũng xin cảm ơn sự tạo điều kiện và hỗ trợ của Nhà trường, Khoa trong suốt quá trình em học tập và nghiên cứu.

Mặc dù đã rất nỗ lực, song do hạn chế về mặt thời gian thực hiện cũng như kinh nghiệm và kiến thức tích lũy còn chưa đầy đủ, báo cáo đồ án môn học của nhóm em chắc chắn không thể tránh khỏi những thiếu sót và hạn chế. em kính mong nhận được sự cảm thông và những đóng góp ý kiến quý báu từ Cô hướng dẫn để em có cơ hội nhìn nhận, bổ sung kiến thức, nâng cao trình độ bản thân và hoàn thiện đồ án này một cách tốt nhất.

Cuối cùng, nhóm em xin chân thành cảm ơn.

Thành phố Hồ Chí Minh, ngày 24 tháng 11 năm 2025

Mục lục

Danh sách bảng

I TỔNG QUAN HỆ THỐNG

1 Mô tả bài toán

Trong bối cảnh chuyển đổi số (Digital Transformation) diễn ra mạnh mẽ, việc vận hành cửa hàng bán lẻ thiết bị công nghệ đòi hỏi sự chính xác tuyệt đối trong quản lý kho vận và trải nghiệm người dùng (UX). Hệ thống **ProShop (v2)** được xây dựng không chỉ là một nền tảng thương mại điện tử (E-commerce) đơn thuần, mà là một giải pháp toàn diện tích hợp quy trình quản lý bán hàng (Sales Management) và kiểm soát hàng tồn kho.

Mục tiêu của hệ thống là cung cấp môi trường giao dịch trực tuyến an toàn, hiệu năng cao, đồng thời cung cấp công cụ quản trị (Admin Dashboard) trực quan hỗ trợ ra quyết định kinh doanh dựa trên dữ liệu thực.

2 Đặc tả chức năng hệ thống

Hệ thống được phân chia thành các phân hệ (Modules) chính nhằm đảm bảo tính module hóa và dễ dàng trong việc bảo trì, kiểm thử:

2.1 Phân hệ Quản trị & Phân quyền (Identity & Access Management):

- **Cơ chế xác thực (Authentication):** Đăng nhập/Đăng ký bảo mật thông qua JWT, mã hóa mật khẩu chuẩn BCrypt.
- **Phân quyền (Authorization):** Áp dụng mô hình RBAC để phân chia quyền hạn rõ ràng giữa Quản trị viên (Admin) và Khách hàng (User).

2.2 Phân hệ Quản lý Sản phẩm (Product Catalog):

- **CRUD Operations:** Hỗ trợ đầy đủ quy trình Thêm mới, Cập nhật, Xóa mềm và truy vấn thông tin sản phẩm.
- **Quản lý thuộc tính:** Kiểm soát chi tiết thông số kỹ thuật, tồn kho (Stock), danh mục và đánh giá từ người dùng (Rating/Review).

2.3 Phân hệ Đặt hàng & Thanh toán (Checkout Flow):

- **Giỏ hàng (Shopping Cart):** Cơ chế lưu trữ trạng thái giỏ hàng thời gian thực, tính toán tổng tiền và thuế phí tự động.
- **Thanh toán:** Tích hợp cổng thanh toán **PayPal** và thẻ tín dụng/ghi nợ (Credit/Debit Card).

3 Yêu cầu Phi chức năng (Non-functional Requirements)

Để đảm bảo chất lượng phần mềm (Software Quality Assurance), hệ thống tuân thủ các ràng buộc:

- **Tính bảo mật (Security):** Dữ liệu nhạy cảm được mã hóa. API được bảo vệ bởi Middleware ngăn chặn XSS, CSRF.
- **Hiệu năng (Performance):** Thời gian phản hồi dưới 2 giây cho tác vụ cơ bản.
- **Tính khả dụng (Usability):** Giao diện Responsive, tương thích đa thiết bị.
- **Tính toàn vẹn dữ liệu:** Đảm bảo nhất quán dữ liệu khi có nhiều giao dịch đồng thời.

4 Kiến trúc và Công nghệ sử dụng

Dựa trên mã nguồn dự án (Source Code Analysis), hệ thống ProShop sử dụng các thư viện lõi nhằm tối ưu hóa hiệu năng:

- **Frontend:**
  - Sử dụng **ReactJS** kết hợp **Vite** (thay cho Create-React-App) để tối ưu tốc độ build và module bundling.
  - Quản lý trạng thái toàn cục với **Redux Toolkit** và **RTK Query** để caching dữ liệu API hiệu quả.
  - Giao diện xây dựng trên **React-Bootstrap**.
- **Backend & Database:**
  - **Node.js & Express.js** xây dựng RESTful API.
  - **MongoDB Atlas** (NoSQL) lưu trữ dữ liệu linh hoạt.

5 Phân tích Rủi ro & Vùng lỗi tiềm ẩn (Risk Analysis)

Qua quá trình phân tích đặc tả kỹ thuật, nhóm kiểm thử xác định các khu vực rủi ro cao cần tập trung kiểm thử:

1. **Sai số trong tính toán tiền tệ (Floating Point Calculation):**
   - *Vấn đề:* JavaScript xử lý số thập phân có thể gây sai số (Ví dụ: $0.1 + 0.2 \neq 0.3$).
   - *Hậu quả:* Sai lệch tổng tiền đơn hàng và thuế phí nếu không xử lý làm tròn (Rounding) đúng cách.
2. **Quản lý Phiên đăng nhập (Session Management):**
   - *Vấn đề:* Khi chuyển đổi tài khoản (Switch User), giỏ hàng của người cũ có thể bị lưu lại cho người mới.
   - *Yêu cầu test:* Kiểm tra kỹ việc dọn dẹp Local Storage khi đăng xuất.
3. **Bảo mật Upload File:**
   - *Vấn đề:* Chức năng upload ảnh sản phẩm có thể bị lợi dụng để tải lên mã độc nếu không lọc MIME Type kỹ.

6 Phân tích Tác nhân hệ thống (Actors)

Hệ thống phục vụ 03 nhóm đối tượng với quyền hạn khác nhau:

- **Khách (Guest):** Xem sản phẩm, tìm kiếm, thêm vào giỏ hàng.
- **Thành viên (Registered User):** Đặt hàng (Place Order), quản lý Profile, xem lịch sử đơn, đánh giá sản phẩm.
- **Quản trị viên (Admin):** CRUD Sản phẩm/User/Đơn hàng, xem báo cáo thống kê, Mark as Delivered.

7 Phạm vi Kiểm thử (Scope of Testing)

Trong khuôn khổ đồ án, nhóm tập trung vào các kỹ thuật:

- **Kiểm thử chức năng (Functional Testing):** Tập trung vào luồng Checkout, Auth và Product Management.
- **Kiểm thử giao diện (UI Testing):** Kiểm tra tính tương thích trên Chrome/Firefox.
- **Kiểm thử bảo mật cơ bản:** SQL/NoSQL Injection, XSS.

II MÔ TẢ

1 Đặc tả Yêu cầu Nghiệp vụ (Business Requirement Document - BRD)

Bảng dưới đây mô tả chi tiết các yêu cầu chức năng (Functional Requirements) của hệ thống ProShop, làm cơ sở cho việc thiết kế các kịch bản kiểm thử (Test Cases):

Mã YC	Tên Module	Tác nhân	Đặc tả chi tiết (Specifications)
REQ-01	Xác thực	Admin Customer	**Đăng nhập:** Hệ thống xác thực qua Email/Password. Mật khẩu phải được mã hóa (Hash). Cấp phát JWT Token khi đăng nhập thành công. **Đăng ký:** Validate định dạng Email, độ mạnh mật khẩu. Chặn đăng ký trùng Email. **Đăng xuất:** Hủy Token và xóa thông tin trong Local Storage.
REQ-02	Quản lý Sản phẩm (Catalog)	Admin	Admin có quyền thực hiện CRUD sản phẩm. - **Upload ảnh:** Chỉ chấp nhận file ảnh (jpg, png), giới hạn dung lượng. - **Quản lý kho:** Tự động trừ số lượng tồn kho (CountInStock) khi có đơn hàng thành công.
REQ-03	Giỏ hàng (Shopping Cart)	Customer	- Thêm sản phẩm vào giỏ (lưu trong Local Storage hoặc Database). - Cập nhật số lượng (không được vượt quá số lượng tồn kho). - Tự động tính toán: *Subtotal = Giá × Số lượng*.
REQ-04	Quy trình Đặt hàng (Checkout)	Customer	Quy trình tuần tự 3 bước bắt buộc: 1. **Shipping:** Validate thông tin địa chỉ giao hàng. 2. **Payment:** Tích hợp API PayPal hoặc chọn thanh toán khi nhận hàng. 3. **Place Order:** Tổng hợp thông tin, tính phí ship, thuế và tạo đơn hàng.
REQ-05	Quản lý Đơn hàng	Admin Customer	**Admin:** Xem danh sách đơn hàng, lọc đơn hàng. Cập nhật trạng thái vận chuyển (Mark as Delivered). **Customer:** Xem chi tiết đơn hàng (Order Details), trạng thái thanh toán (Paid/Un-paid) và vận chuyển.

Mã YC	Tên Module	Tác nhân	Đặc tả chi tiết (Specifications)
REQ-06	Quản trị Người dùng	Admin	- Xem danh sách toàn bộ người dùng. - Xóa người dùng (Soft delete hoặc Hard delete). - Phân quyền: Nâng cấp User thường thành Admin (Update Role).
REQ-07	Đánh giá	Customer	- Chỉ thành viên đã đăng nhập mới được đánh giá. - **Ràng buộc:** Mỗi tài khoản chỉ được đánh giá sản phẩm 01 lần. - Tính điểm trung bình sao (Average Rating) cho sản phẩm.
REQ-08	Tìm kiếm	Admin Customer	- Tìm kiếm "fuzzy search" theo tên sản phẩm. - **Phân trang (Pagination):** Hiển thị tối đa n sản phẩm/trang để tối ưu hiệu năng tải trang.

2 Yêu cầu kỹ thuật

Dưới đây là danh sách các yêu cầu kỹ thuật chi tiết (Technical Constraints) được áp dụng để kiểm thử các chức năng của hệ thống ProShop.

DANH SÁCH YÊU CẦU KỸ THUẬT (TECHNICAL REQUIREMENTS)	
Mã	**Mô tả yêu cầu**
QUẢN LÝ NGƯỜI DÙNG (USER MANAGEMENT)	
TR_U01	Mã người dùng (_id) được hệ thống MongoDB tự động sinh, không được nhập thủ công.
TR_U02	Tên người dùng (Name) không được để trống.
TR_U03	Tên người dùng phải có độ dài từ 2 đến 50 ký tự.
TR_U04	Tên người dùng không được chứa các ký tự đặc biệt gây lỗi hiển thị.
TR_U05	Email bắt buộc nhập và phải đúng định dạng chuẩn (vd: name@domain.com).
TR_U06	Email không được trùng với tài khoản đã tồn tại trong hệ thống.
TR_U07	Mật khẩu (Password) phải có độ dài tối thiểu 6 ký tự.
TR_U08	Mật khẩu khi đăng ký và xác nhận mật khẩu (Confirm Password) phải trùng khớp.
TR_U09	Mật khẩu được mã hóa (Hash) trước khi lưu vào Database.
TR_U10	Admin có quyền xóa người dùng, nhưng không thể xóa chính mình.
TR_U11	Admin có quyền cập nhật thông tin người dùng khác.

Tiếp tục ở trang sau...

... DANH SÁCH YÊU CẦU (TIẾP THEO) ...	
Mã	**Mô tả yêu cầu**
QUẢN LÝ SẢN PHẨM (PRODUCT MANAGEMENT)	
TR_P01	Tên sản phẩm không được để trống.
TR_P02	Tên sản phẩm không được trùng lặp với sản phẩm đã có.
TR_P03	Giá sản phẩm (Price) phải là số và lớn hơn hoặc bằng 0.
TR_P04	Hình ảnh sản phẩm (Image) bắt buộc phải có đường dẫn hoặc upload file.
TR_P05	Thương hiệu (Brand) và Danh mục (Category) không được để trống.
TR_P06	Số lượng tồn kho (CountInStock) phải là số nguyên dương (≥ 0).
TR_P07	Mô tả sản phẩm (Description) không được để trống.
TR_P08	Đánh giá sản phẩm (Rating) nằm trong khoảng từ 0 đến 5 sao.
TR_P09	Số lượng đánh giá (NumReviews) tự động tăng khi có người dùng review.
TR_P10	Khi Admin cập nhật sản phẩm, các trường dữ liệu cũ phải được hiển thị sẵn (Pre-fill).
GIỎ HÀNG & ĐẶT HÀNG (CART & ORDER)	
TR_C01	Số lượng sản phẩm thêm vào giỏ phải lớn hơn 0.
TR_C02	Số lượng sản phẩm thêm vào giỏ không được vượt quá số lượng tồn kho (In Stock).
TR_C03	Tổng tiền (Subtotal) = Đơn giá $\times$ Số lượng (Tự động tính).
TR_O01	Địa chỉ giao hàng (Shipping Address) bắt buộc nhập đầy đủ: Địa chỉ, Thành phố, Mã bưu điện, Quốc gia.
TR_O02	Phương thức thanh toán (Payment Method) mặc định là PayPal.
TR_O03	Phí vận chuyển (Shipping Price) tự động tính: 0 nếu tổng đơn hàng $>$ 100\$, ngược lại là 10\$.
TR_O04	Thuế (Tax Price) được tính bằng 15% tổng giá trị đơn hàng.
TR_O05	Tổng cộng (Total Price) = Tiền hàng + Phí ship + Thuế.
TR_O06	Khi đặt hàng thành công, số lượng tồn kho của sản phẩm phải tự động giảm tương ứng.
ĐÁNH GIÁ & BÌNH LUẬN (REVIEW)	
TR_R01	Chỉ người dùng đã đăng nhập mới được phép đánh giá.
TR_R02	Mỗi người dùng chỉ được đánh giá 1 lần cho mỗi sản phẩm.
TR_R03	Nếu người dùng cố tình đánh giá lại sản phẩm đã đánh giá, hệ thống báo lỗi "Product already reviewed".

Tiếp tục ở trang sau...

... DANH SÁCH YÊU CẦU (TIẾP THEO) ...	
Mã	**Mô tả yêu cầu**
TR_R04	Nội dung bình luận (Comment) không được để trống.
YÊU CẦU API & BẢO MẬT (SYSTEM)	
TR_S01	API trả về mã lỗi 401 (Unauthorized) nếu truy cập tài nguyên bảo mật mà không có Token.
TR_S02	API trả về mã lỗi 404 (Not Found) nếu ID sản phẩm/người dùng không tồn tại.
TR_S03	Token đăng nhập (JWT) có thời hạn 30 ngày.
TR_S04	Các thao tác Admin (Thêm/Sửa/Xóa sản phẩm) phải được kiểm tra quyền Admin (Middleware).

III CƠ SỞ LÝ THUYẾT VÀ KẾ HOẠCH KIỂM THỬ

1 Cơ sở lý thuyết về Kiểm thử phần mềm

1.1 Tổng quan và Định nghĩa (Overview & Definitions)

Theo định nghĩa chuẩn của **IEEE 829** và **ISTQB** (International Software Testing Qualifications Board):

> "Kiểm thử phần mềm là một quy trình bao gồm các hoạt động vòng đời tĩnh và động, liên quan đến việc lập kế hoạch, chuẩn bị và đánh giá các sản phẩm phần mềm nhằm xác định xem chúng có đáp ứng các yêu cầu đã nêu (Specified Requirements) hay không, để chứng minh sự phù hợp với mục đích sử dụng và để phát hiện các khiếm khuyết (Defects)."

Bản chất của kiểm thử được thể hiện qua hai khía cạnh cốt lõi:

- **Verification (Kiểm chứng):** Trả lời câu hỏi *"Are we building the product right?"* (Chúng ta có đang xây dựng sản phẩm đúng cách không?). Quá trình này đảm bảo phần mềm tuân thủ các quy chuẩn thiết kế và logic lập trình.
- **Validation (Thẩm định):** Trả lời câu hỏi *"Are we building the right product?"* (Chúng ta có đang xây dựng đúng sản phẩm khách hàng cần không?). Quá trình này đảm bảo phần mềm đáp ứng đúng nhu cầu nghiệp vụ thực tế.

1.2 Mục tiêu của kiểm thử (Objectives of Testing)

Việc kiểm thử hệ thống thương mại điện tử ProShop hướng tới các mục tiêu chiến lược sau:

1. **Phát hiện lỗi (Defect Detection):** Tìm ra các lỗi ẩn trong mã nguồn, lỗi logic nghiệp vụ hoặc lỗi giao diện trước khi bàn giao cho người dùng cuối.
2. **Phòng ngừa lỗi (Defect Prevention):** Thông qua việc phân tích yêu cầu và kiểm thử sớm (Early Testing), giảm thiểu chi phí sửa lỗi trong các giai đoạn sau của vòng đời phát triển.
3. **Đánh giá chất lượng (Quality Assessment):** Cung cấp thông tin khách quan về độ ổn định, hiệu năng và tính bảo mật của hệ thống cho các bên liên quan (Stakeholders) ra quyết định.

1.3 Bảy nguyên tắc kiểm thử (Seven Testing Principles)

Quy trình kiểm thử trong đồ án tuân thủ 7 nguyên tắc vàng của ISTQB:

- **Nguyên tắc 1 - Kiểm thử chứng minh sự hiện diện của lỗi:** Kiểm thử có thể chỉ ra rằng lỗi đang tồn tại, nhưng không thể chứng minh phần mềm hoàn toàn sạch lỗi (Bug-free).
- **Nguyên tắc 2 - Kiểm thử toàn bộ là không thể (Exhaustive testing is impossible):** Thay vì kiểm thử tất cả tổ hợp dữ liệu (điều bất khả thi), tôi sử dụng kỹ thuật phân tích rủi ro để chọn lọc kịch bản tối ưu.
- **Nguyên tắc 3 - Kiểm thử sớm (Early Testing):** Các hoạt động kiểm thử được bắt đầu ngay từ giai đoạn phân tích yêu cầu.

- **Nguyên tắc 4 - Sự tập trung của lỗi (Defect Clustering):** Tuân theo quy luật Pareto (80/20), phần lớn lỗi nghiêm trọng thường tập trung ở một số module chính như *Thanh toán* và *Quản lý kho*.
- **Nguyên tắc 5 - Nghịch lý thuốc trừ sâu (Pesticide Paradox):** Các bộ Test Case cần được cập nhật thường xuyên để tìm ra lỗi mới.
- **Nguyên tắc 6 - Kiểm thử phụ thuộc ngữ cảnh:** Kiểm thử Website E-commerce sẽ có các tiêu chí khác biệt so với phần mềm quản lý nội bộ.
- **Nguyên tắc 7 - Sự ngộ nhận về "hết lỗi":** Một hệ thống không có lỗi kỹ thuật chưa chắc đã thành công nếu nó khó sử dụng (Usability issues).

1.4 Các cấp độ kiểm thử áp dụng (Testing Levels)

Hệ thống được kiểm thử qua 4 cấp độ theo mô hình V-Model:

Cấp độ	Mô tả phạm vi áp dụng trong đồ án
1. Unit Testing	Kiểm thử đơn vị các hàm tiện ích (Utils) xử lý tiền tệ, ngày tháng bằng Jest Framework.
2. Integration Testing	Kiểm tra giao tiếp giữa Frontend (ReactJS) và Backend (NodeJS) thông qua RESTful API; Tích hợp giữa Backend và MongoDB.
3. System Testing	Kiểm thử toàn bộ hệ thống đã tích hợp hoàn chỉnh. Đây là trọng tâm của đồ án với các kỹ thuật Hộp đen (Black-box).
4. Acceptance Testing	Kiểm thử chấp nhận người dùng (UAT), đóng vai trò người dùng cuối thực hiện quy trình mua hàng thực tế.

Bảng 3.1: Các cấp độ kiểm thử áp dụng cho ProShop

2 Các phương pháp và Kỹ thuật kiểm thử áp dụng

Để đảm bảo độ bao phủ mã nguồn (Code Coverage) và độ bao phủ chức năng (Functional Coverage) tối đa, đồ án kết hợp linh hoạt hai phương pháp tiếp cận chính:

2.1 Phương pháp Kiểm thử Hộp đen (Black-box Testing)

Đây là phương pháp kiểm thử dựa trên đặc tả yêu cầu (Software Requirements Specification) mà không cần can thiệp vào cấu trúc mã nguồn nội bộ. Người kiểm thử đóng vai trò là người dùng cuối, chỉ quan tâm đến đầu vào (Input) và đầu ra (Output).

Các kỹ thuật cụ thể được áp dụng trong ProShop:

- **Phân hoạch tương đương (Equivalence Partitioning - EP):** Kỹ thuật chia miền dữ liệu đầu vào thành các lớp tương đương. Giả định rằng nếu một giá trị đại diện trong lớp được xử lý đúng, thì toàn bộ các giá trị khác trong lớp đó cũng sẽ đúng. *Áp dụng: Kiểm thử các Form nhập liệu (Đăng ký, Đăng nhập, Thêm sản phẩm).*
- **Phân tích giá trị biên (Boundary Value Analysis - BVA):** Kỹ thuật tập trung kiểm thử tại các điểm biên của miền dữ liệu (Min, Max, Min-1, Max+1), nơi xác suất xảy ra lỗi là cao nhất. *Áp dụng: Kiểm thử số lượng tồn kho (Stock), Giá tiền sản phẩm, Số lượng đặt hàng (Quantity).*

- **Bảng quyết định (Decision Table Testing):** Sử dụng để mô hình hóa các logic nghiệp vụ phức tạp phụ thuộc vào nhiều điều kiện đầu vào kết hợp. *Áp dụng: Quy trình Thanh toán (Điều kiện: Đăng nhập thành công AND Giỏ hàng có sản phẩm AND Địa chỉ hợp lệ → Cho phép thanh toán).*
- **Kiểm thử chuyển đổi trạng thái (State Transition Testing):** Kiểm tra sự thay đổi trạng thái của đối tượng khi có sự kiện tác động. *Áp dụng: Vòng đời đơn hàng (Mới tạo → Chờ thanh toán → Đã thanh toán → Đang giao → Hoàn tất).*

2.2 Phương pháp Kiểm thử Hộp trắng (White-box Testing)

Đây là phương pháp kiểm thử dựa trên việc phân tích cấu trúc mã nguồn, luồng điều khiển (Control Flow) và luồng dữ liệu (Data Flow). Phương pháp này yêu cầu Tester phải có kiến thức về lập trình (MERN Stack).

Kỹ thuật áp dụng:

- **Kiểm thử đường cơ sở (Basis Path Testing):** Dựa trên đồ thị dòng điều khiển (Control Flow Graph - CFG) để đảm bảo mọi câu lệnh (Statement) và mọi nhánh rẽ (Branch) đều được thực thi ít nhất một lần.
- **Độ phức tạp Cyclomatic (Cyclomatic Complexity):** Đo lường độ phức tạp của thuật toán để xác định số lượng Test Case tối thiểu cần thiết cho các hàm tính toán quan trọng (như hàm tính tổng tiền đơn hàng).

3 Quy trình kiểm thử phần mềm (Software Testing Life Cycle - STLC)

Nhóm thực hiện tuân thủ quy trình STLC tiêu chuẩn gồm 6 giai đoạn, đảm bảo tính khoa học và hệ thống:

1. **Phân tích yêu cầu (Requirement Analysis):**
    - *Hoạt động:* Nghiên cứu tài liệu đặc tả, xác định các chức năng trọng yếu cần kiểm thử (Test Scope).
    - *Đầu ra:* Bảng Ma trận truy vết yêu cầu (RTM Draft).
2. **Lập kế hoạch kiểm thử (Test Planning):**
    - *Hoạt động:* Xác định chiến lược (Strategy), lựa chọn công cụ (Tools), ước lượng thời gian và nguồn lực.
    - *Đầu ra:* Tài liệu Kế hoạch kiểm thử (Test Plan Document).
3. **Thiết kế kịch bản kiểm thử (Test Case Development):**
    - *Hoạt động:* Viết các Test Case chi tiết (bao gồm Test Data, Steps, Expected Result). Chuẩn bị dữ liệu mẫu (Mock Data).
    - *Đầu ra:* Bộ Test Case hoàn chỉnh (Test Suite).
4. **Thiết lập môi trường (Test Environment Setup):**
    - *Hoạt động:* Cài đặt Node.js Server, cấu hình MongoDB, chuẩn bị trình duyệt Chrome/-Firefox.
    - *Đầu ra:* Môi trường kiểm thử sẵn sàng (Test Bed Ready).

5. **Thực thi kiểm thử (Test Execution):**
    - *Hoạt động:* Chạy các Test Case, so sánh kết quả thực tế với mong đợi. Ghi nhận lỗi (Log Bug) nếu có sai lệch.
    - *Đầu ra:* Báo cáo kết quả thực thi (Execution Report), Danh sách lỗi (Defect Report).
6. **Đóng chu trình kiểm thử (Test Cycle Closure):**
    - *Hoạt động:* Đánh giá tiêu chí kết thúc, tổng hợp báo cáo lỗi, rút ra bài học kinh nghiệm.
    - *Đầu ra:* Báo cáo tổng kết (Final Test Summary Report).

3.1 Quy trình quản lý lỗi (Defect Lifecycle)

Khi một lỗi được phát hiện, nó sẽ trải qua quy trình vòng đời như sau:

- **New:** Lỗi mới được Tester tìm thấy và log lại.
- **Assigned:** Lỗi được gán cho Developer để sửa.
- **Fixed:** Developer đã sửa xong lỗi và đẩy code mới lên.
- **Retest:** Tester kiểm tra lại trên bản build mới.
- **Verified/Closed:** Xác nhận lỗi đã được sửa triệt để và đóng lại.
- **Reopen:** Nếu lỗi vẫn còn tồn tại sau khi fix.

4 Môi trường và Công cụ kiểm thử

Để đảm bảo kết quả kiểm thử phản ánh chính xác hiệu năng và tính tương thích của hệ thống trong thực tế, nhóm thực hiện thiết lập môi trường kiểm thử (Test Bed) với các thông số kỹ thuật chi tiết như sau:

4.1 Cấu hình Phần cứng (Hardware)

Hệ thống được kiểm thử trên máy trạm đóng vai trò là Client truy cập vào Web Server Localhost.

Thành phần	Thông số kỹ thuật chi tiết
Hệ điều hành	Windows 11 Pro (64-bit) / Build 22H2
Vi xử lý (CPU)	Intel Core i5-12400F @ 2.50GHz (6 Cores)
Bộ nhớ (RAM)	16 GB DDR4 3200MHz
Ổ cứng	SSD NVMe 512GB (Đảm bảo tốc độ I/O khi chạy Database)
Màn hình	24 inch IPS - Độ phân giải Full HD (1920x1080)
Kết nối mạng	Internet cáp quang (Wifi 5GHz) - Băng thông 100Mbps

Bảng 3.2: Cấu hình phần cứng thiết bị kiểm thử

4.2 Môi trường và công cụ hỗ trợ (Software & Tools)

Các công cụ sau được lựa chọn dựa trên tiêu chí: phổ biến, miễn phí (Open Source) và tương thích tốt với kiến trúc MERN Stack.

STT	Tên công cụ	Mục đích sử dụng chính
1	**Visual Studio Code**	Môi trường phát triển tích hợp (IDE) dùng để review code và debug lỗi (White-box Testing).
2	**Google Chrome**	Trình duyệt chính để thực hiện kiểm thử giao diện và chức năng người dùng (Version 120.x).
3	**Postman**	Công cụ giả lập Request để kiểm thử API (GET, POST, PUT, DELETE) và kiểm tra mã phản hồi (HTTP Status Code).
4	**MongoDB Compass**	Giao diện đồ họa (GUI) để kiểm tra dữ liệu được ghi vào Database có chính xác hay không.
5	**Redux DevTools**	Extension trình duyệt giúp theo dõi sự thay đổi của State (Giỏ hàng, User) theo thời gian thực.
6	**Jira / Excel**	Công cụ quản lý, theo dõi lỗi (Bug Tracking) và lập báo cáo tiến độ.

Bảng 3.3: Danh sách bộ công cụ hỗ trợ kiểm thử

5 Ma trận truy vết yêu cầu (Requirement Traceability Matrix - RTM)

Ma trận RTM được xây dựng nhằm đảm bảo tính toàn vẹn của quá trình kiểm thử, giúp theo dõi mối quan hệ 1-1 giữa Yêu cầu nghiệp vụ (Business Req) và Kịch bản kiểm thử (Test Case).

Mã Yêu cầu (Req ID)	Mô tả chức năng	Mã Test Case (TC ID)	Ưu tiên
REQ-01	Xác thực người dùng (Đăng nhập/Đăng ký)	TC-AUTH-001	High
		TC-AUTH-002	High
		TC-AUTH-003	Medium
REQ-02	Quản lý Sản phẩm (Admin CRUD)	TC-PROD-001	High
		TC-PROD-002	High
REQ-03	Chức năng Giỏ hàng (Thêm/Sửa/Xóa)	TC-CART-001	Critical
REQ-03	Tính toán tổng tiền (Subtotal Logic)	TC-CART-002	Critical
REQ-04	Quy trình Đặt hàng (Checkout Flow)	TC-ORDER-001	Critical
		TC-ORDER-002	Critical
		TC-ORDER-003	High
REQ-05	Quản lý Đơn hàng (Admin View)	TC-ADM-001	Medium

Mã Yêu cầu	Mô tả chức năng	Mã Test Case	Ưu tiên
REQ-06	Quản lý Người dùng	TC-ADM-002	Low
REQ-07	Đánh giá sản phẩm (Rating/Review)	TC-USER-001	Medium
REQ-08	Tìm kiếm sản phẩm	TC-SRCH-001	Medium

Ghi chú: Danh sách chi tiết các Test Case sẽ được trình bày cụ thể trong Chương 4.

6 Danh sách Kịch bản kiểm thử (Test Scenarios)

Dựa trên Ma trận truy vết yêu cầu (RTM) và chiến lược kiểm thử đã đề ra, nhóm thực hiện xác định danh sách các kịch bản kiểm thử (Test Scenarios) cần thực hiện. Kịch bản kiểm thử là các tình huống cấp cao (High-level) mô tả chức năng cần kiểm tra, đóng vai trò định hướng cho việc thiết kế Test Case chi tiết sau này.

Bảng dưới đây liệt kê chi tiết các kịch bản, bao gồm cả trường hợp kiểm thử luồng chính (Positive) và các trường hợp ngoại lệ (Negative/Edge cases).

Scenario ID	Tên Kịch bản (Scenario Name)	Mục tiêu kiểm thử (Objective)
1. Phân hệ Xác thực & Phân quyền (Authentication & Authorization)		
TS-AUTH-001	Kiểm thử Đăng nhập hợp lệ	Xác minh Admin/User đăng nhập thành công với thông tin đúng.
TS-AUTH-002	Kiểm thử Đăng nhập sai mật khẩu	Đảm bảo hệ thống chặn và báo lỗi khi nhập sai Credential.
TS-AUTH-003	Kiểm thử Đăng nhập sai định dạng Email	Kiểm tra tính năng Validate định dạng Email (HTML5 & Backend).
TS-AUTH-004	Kiểm thử Đăng ký tài khoản mới	Xác minh quy trình tạo tài khoản, mã hóa mật khẩu trong DB.
TS-AUTH-005	Kiểm thử Đăng ký trùng Email	Đảm bảo không thể tạo 2 tài khoản với cùng 1 địa chỉ Email.
TS-AUTH-006	Kiểm thử Đăng xuất	Xác minh Token bị xóa khỏi Local Storage và chuyển hướng về Login.
TS-AUTH-007	Kiểm thử Bảo mật Route (Protected Routes)	Đảm bảo User chưa đăng nhập không thể truy cập vào trang Profile hoặc Admin Dashboard thông qua URL.
2. Phân hệ Quản lý Sản phẩm (Product Catalog)		
TS-PROD-001	Kiểm thử Xem danh sách sản phẩm	Đảm bảo danh sách load nhanh, hiển thị đúng tên, giá, hình ảnh.
TS-PROD-002	Kiểm thử Chi tiết sản phẩm	Kiểm tra thông tin chi tiết, mô tả, trạng thái tồn kho của từng item.
TS-PROD-003	Kiểm thử Tìm kiếm sản phẩm	Xác minh chức năng Search hoạt động đúng với từ khóa tiếng Việt/Anh.
TS-PROD-004	Kiểm thử Phân trang (Pagination)	Đảm bảo phân trang hoạt động đúng logic (Hiển thị n sản phẩm/trang).
TS-PROD-005	Kiểm thử Thêm mới sản phẩm (Admin)	Admin tạo sản phẩm thành công với đầy đủ trường thông tin.
TS-PROD-006	Kiểm thử Upload hình ảnh	Chỉ cho phép upload file ảnh (.jpg, .png), chặn file .exe, .sh.
TS-PROD-007	Kiểm thử Cập nhật sản phẩm	Admin sửa thông tin (Giá, Tên, Tồn kho) và dữ liệu được lưu đúng.

Scenario ID	Tên Kịch bản	Mục tiêu
TS-PROD-008	Kiểm thử Xóa sản phẩm	Admin xóa sản phẩm, kiểm tra sản phẩm biến mất khỏi danh sách.
TS-PROD-009	Kiểm thử Đánh giá (Review) sản phẩm	User chỉ được đánh giá sản phẩm đã mua và chưa đánh giá trước đó.
3. Phân hệ Giỏ hàng & Đặt hàng (Cart & Order)		
TS-CART-001	Kiểm thử Thêm vào giỏ hàng	Sản phẩm được thêm vào giỏ, icon giỏ hàng cập nhật số lượng.
TS-CART-002	Kiểm thử Cập nhật số lượng mua	Tăng/Giảm số lượng trong giỏ, Tổng tiền (Subtotal) thay đổi tương ứng.
TS-CART-003	Kiểm thử Giới hạn Tồn kho	Không cho phép chọn số lượng mua lớn hơn số lượng thực tế trong kho.
TS-CART-004	Kiểm thử Xóa khỏi giỏ hàng	Xóa item khỏi giỏ, tính lại tổng tiền, không cần reload trang.
TS-ORDER-001	Kiểm thử Nhập địa chỉ giao hàng	Bắt buộc nhập đầy đủ địa chỉ trước khi qua bước thanh toán.
TS-ORDER-002	Kiểm thử Thanh toán PayPal	Tích hợp PayPal Sandbox, thực hiện thanh toán thành công và thất bại.
TS-ORDER-003	Kiểm thử Tính toán Tổng đơn hàng	Xác minh công thức: $Total = Items + Shipping + Tax$.
TS-ORDER-004	Kiểm thử Lịch sử đơn hàng	User xem lại được các đơn hàng đã đặt và trạng thái (Paid/Delivered).
4. Phân hệ Quản trị Người dùng (User Management)		
TS-USER-001	Kiểm thử Cập nhật Hồ sơ (Profile)	User tự đổi tên và mật khẩu đăng nhập thành công.
TS-ADM-001	Kiểm thử Xem danh sách người dùng	Admin xem được toàn bộ User, hiển thị đúng quyền hạn (Admin/User).
TS-ADM-002	Kiểm thử Xóa người dùng	Admin xóa tài khoản vi phạm. Kiểm tra ràng buộc Admin không được tự xóa mình.
TS-ADM-003	Kiểm thử Nâng quyền (Promote)	Admin nâng cấp một User thường lên thành Admin.
TS-ADM-004	Kiểm thử Cập nhật trạng thái Giao hàng	Admin đánh dấu đơn hàng là "Đã giao" (Delivered).

Ghi chú: Các kịch bản trên là cơ sở để thiết kế các Test Case chi tiết được trình bày trong Chương 4.

IV THIẾT KẾ VÀ THỰC THI KIỂM THỬ HỘP ĐEN

Trong chương này, nhóm thực hiện trình bày chi tiết các kịch bản kiểm thử (Test Cases) được tổ chức theo từng phân hệ chức năng (Module). Các kỹ thuật như **Phân hoạch tương đương** và **Phân tích giá trị biên** được áp dụng linh hoạt để đảm bảo độ bao phủ.

1 Kỹ thuật Phân hoạch tương đương & Phân tích giá trị biên (EP & BVA)

1.1 Cơ sở lý thuyết (Theoretical Basis)

Để tối ưu hóa số lượng kịch bản kiểm thử mà vẫn đảm bảo độ bao phủ (Coverage), nhóm áp dụng hai kỹ thuật hộp đen kinh điển:

- **Phân hoạch tương đương (Equivalence Partitioning - EP):** Kỹ thuật này chia miền dữ liệu đầu vào thành các lớp (partitions) hữu hạn. Giả định rằng hệ thống sẽ xử lý tất cả các giá trị trong cùng một lớp theo cách giống nhau. Nếu một giá trị đại diện trong lớp hoạt động đúng, ta có thể kết luận cả lớp đó đúng.
- **Phân tích giá trị biên (Boundary Value Analysis - BVA):** Kinh nghiệm thực tế cho thấy lỗi thường xuất hiện tại các điểm "biên" của miền dữ liệu hơn là tại trung tâm. Kỹ thuật này tập trung kiểm thử tại các giá trị biên (Min, Max) và cận biên (Min-1, Max+1).

1.2 Áp dụng cho Phân hệ Xác thực (Authentication)

1. Phân tích lớp dữ liệu (Data Analysis): Đối với chức năng Đăng nhập, chúng tôi phân tích trường dữ liệu **Email** và **Mật khẩu** thành các lớp kiểm thử như sau:

Đối tượng	Lớp tương đương (Class)	Giá trị đại diện (Test Data)
Email Input	**Hợp lệ (Valid):** Đúng định dạng chuẩn RFC.	`admin@example.com`
	Không hợp lệ 1: Thiếu ký tự @.	`admin.example.com`
	Không hợp lệ 2: Bỏ trống (Empty).	`[Null]`
Password	**Hợp lệ (Valid):** Đúng mật khẩu đã lưu trong DB.	`123456`
	Không hợp lệ: Sai mật khẩu.	`wrongpass123`

Bảng 4.1: Bảng phân hoạch tương đương cho Form Đăng nhập

2. Thiết kế Test Case chi tiết: Dựa trên bảng phân tích trên, chúng tôi thiết kế các ca kiểm thử sau:

Mục	Nội dung chi tiết
Mã Test Case	**TC-AUTH-001**
Tên kịch bản	Đăng nhập thành công (Valid Case)
Mục đích	Kiểm tra luồng đăng nhập với lớp dữ liệu hợp lệ.
Dữ liệu	Email: `admin@example.com` Pass: `123456`
Các bước	1. Truy cập trang Login. 2. Nhập Email và Password đúng. 3. Nhấn nút "Sign In".
Kết quả mong đợi	- Chuyển hướng sang trang Dashboard. - Header hiện tên "Admin User".
Kết quả thực tế	Hệ thống hoạt động đúng.
Trạng thái	**PASSED**

Bảng 4.2: Kiểm thử Đăng nhập hợp lệ

Mục	Nội dung chi tiết
Mã Test Case	**TC-AUTH-002**
Tên kịch bản	Đăng nhập thất bại (Invalid Password)
Mục đích	Kiểm tra xử lý lỗi khi nhập sai mật khẩu.
Dữ liệu	Email: `admin@example.com` Pass: `sai123`
Các bước	1. Nhập Email đúng. 2. Nhập Password sai. 3. Nhấn nút "Sign In".
Kết quả mong đợi	- Không chuyển trang. - Báo lỗi: "Invalid email or password".
Kết quả thực tế	Thông báo lỗi hiển thị chính xác.
Trạng thái	**PASSED**

Bảng 4.3: Kiểm thử Đăng nhập sai mật khẩu

1.3 Áp dụng cho Logic Tồn kho (Inventory Logic)

1. Phân tích giá trị biên (BVA Analysis): Chức năng "Thêm vào giỏ hàng" phụ thuộc chặt chẽ vào số lượng tồn kho (CountInStock). Giả sử sản phẩm "iPhone 15" có tồn kho là **10 cái**. Ta có các điểm biên cần kiểm thử:

- **Min:** 1 (Giá trị nhỏ nhất hợp lệ).
- **Min - 1:** 0 (Giá trị không hợp lệ).
- **Max:** 10 (Giá trị lớn nhất hợp lệ).

- **Max + 1:** 11 (Giá trị vượt quá tồn kho - Phải bị chặn).

Loại biên	Giá trị test	Dự kiến	Giải thích
Cận dưới (Min-)	0	Fail	Không cho phép mua 0 sản phẩm
Biên dưới (Min)	1	Pass	Mua số lượng tối thiểu
Biên trên (Max)	10	Pass	Mua vét sạch kho
Cận trên (Max+)	11	Fail	Mua quá số lượng có sẵn

Bảng 4.4: Bảng phân tích biên cho trường Số lượng (Quantity)

2. Thiết kế Test Case chi tiết:

Mục	Nội dung chi tiết
Mã Test Case	**TC-CART-002**
Tên kịch bản	Kiểm tra biên trên (Max Value) của tồn kho
Mục đích	Đảm bảo người dùng không thể chọn số lượng lớn hơn tồn kho thực tế.
Dữ liệu	Sản phẩm: iPhone 15 Tồn kho (Stock): 10
Các bước	1. Vào trang chi tiết sản phẩm. 2. Click vào Dropdown chọn số lượng (Qty). 3. Kiểm tra danh sách hiển thị.
Kết quả mong đợi	- Dropdown chỉ hiển thị các số từ 1 đến 10. - Không có tùy chọn số 11.
Kết quả thực tế	Dropdown render đúng logic.
Trạng thái	**PASSED**

Bảng 4.5: Kiểm thử Giới hạn tồn kho

2 Kỹ thuật Bảng quyết định (Decision Table Testing)

2.1 Cơ sở lý thuyết và Phạm vi áp dụng

Bảng quyết định (Decision Table) là một kỹ thuật kiểm thử hộp đen mạnh mẽ, được sử dụng để mô hình hóa các logic nghiệp vụ phức tạp - nơi mà đầu ra của hệ thống phụ thuộc vào sự kết hợp của nhiều điều kiện đầu vào khác nhau.

Trong hệ thống ProShop, quy trình Thanh toán (Checkout Process) là khu vực có logic phức tạp nhất, đòi hỏi sự kết hợp chặt chẽ giữa: Trạng thái đăng nhập, Trạng thái giỏ hàng và Thông tin giao hàng. Nếu chỉ kiểm thử rời rạc từng điều kiện, ta rất dễ bỏ sót các trường hợp ngoại lệ (Edge cases).

2.2 Phân tích Điều kiện và Hành động

Để xây dựng bảng quyết định, trước tiên nhóm xác định các yếu tố cấu thành:

1. Các điều kiện đầu vào (Conditions - C):

- **C1 - User Logged In:** Người dùng đã đăng nhập vào hệ thống hay chưa? (True/False).
- **C2 - Cart Not Empty:** Giỏ hàng có sản phẩm hay không? (True/False).
- **C3 - Shipping Address Set:** Người dùng đã điền đầy đủ địa chỉ giao hàng chưa? (True/-False).

2. Các hành động phản hồi (Actions - A):

- **A1 - Show Login Redirect:** Chuyển hướng người dùng về trang Đăng nhập.
- **A2 - Show Cart Empty Message:** Hiển thị thông báo giỏ hàng trống hoặc disable nút Checkout.
- **A3 - Redirect to Shipping Screen:** Yêu cầu người dùng nhập địa chỉ.
- **A4 - Allow Payment Access:** Cho phép truy cập màn hình chọn phương thức thanh toán (PayPal/Stripe).

2.3 Thiết kế Bảng quyết định (Truth Table)

Dựa trên các điều kiện trên, ta có tổng cộng $2^3 = 8$ quy tắc (Rules). Tuy nhiên, để tối ưu hóa, ta có thể rút gọn các trường hợp bất khả thi hoặc trùng lặp.

Dưới đây là Bảng quyết định rút gọn cho luồng Checkout:

Quy tắc (Rule ID)	**R1**	**R2**	**R3**	**R4**
Conditions (Điều kiện)				
C1: Đã đăng nhập?	**False**	True	True	True
C2: Có sản phẩm?	-	**False**	True	True
C3: Có địa chỉ?	-	-	**False**	**True**
Actions (Hành động)				
A1: Về trang Login	**X**			
A2: Báo lỗi Cart rỗng		**X**		
A3: Về trang Shipping			**X**	
A4: Cho phép Thanh toán				**X**

Bảng 4.6: Bảng quyết định logic quy trình Thanh toán

2.4 Phân tích chi tiết các Quy tắc (Rules Analysis)

Để đảm bảo tính chính xác của bảng trên, chúng tôi phân tích chi tiết từng kịch bản nghiệp vụ tương ứng với mỗi cột:

- **Quy tắc 1 (Rule 1 - R1): Guest Checkout Attempt**
  - *Mô tả:* Người dùng chưa đăng nhập (C1 = False) cố gắng truy cập vào trang thanh toán (`/payment`).

- *Hệ quả:* Hệ thống không quan tâm giỏ hàng có đồ hay không. Middleware bảo mật sẽ chặn lại ngay lập tức.
- *Hành động đúng:* Hệ thống phải chuyển hướng (Redirect) người dùng về trang `/login`. Sau khi đăng nhập xong, phải tự động đưa họ quay lại trang Checkout (chức năng Redirect Back).

- **Quy tắc 2 (Rule 2 - R2): Empty Cart Checkout**
  - *Mô tả:* Người dùng đã đăng nhập (C1 = True) nhưng giỏ hàng đang rỗng (C2 = False). Tình huống này xảy ra khi người dùng xóa hết sản phẩm trong giỏ rồi cố tình nhấn Back hoặc gõ URL.
  - *Hệ quả:* Không thể tạo đơn hàng với 0 sản phẩm.
  - *Hành động đúng:* Hệ thống hiển thị thông báo "Your cart is empty" và nút "Go Back".

- **Quy tắc 3 (Rule 3 - R3): Missing Shipping Info**
  - *Mô tả:* Người dùng hợp lệ, có hàng trong giỏ, nhưng chưa cung cấp địa chỉ giao hàng (C3 = False).
  - *Hệ quả:* Không thể tính phí Ship và Thuế.
  - *Hành động đúng:* Hệ thống cưỡng chế chuyển hướng về bước `/shipping`.

- **Quy tắc 4 (Rule 4 - R4): Happy Path**
  - *Mô tả:* Thỏa mãn tất cả điều kiện: Đã login, có hàng, có địa chỉ.
  - *Hành động đúng:* Hệ thống mở khóa màn hình Payment Method (PayPal) và cho phép đặt hàng.

2.5 Thiết kế Test Case từ Bảng quyết định

Dựa trên 4 quy tắc trên, chúng tôi thiết kế các Test Case đại diện như sau:

Mục	**Nội dung chi tiết**
Mã Test Case	**TC-ORDER-001 (Tương ứng R3)**
Tên kịch bản	Kiểm tra ràng buộc bắt buộc nhập địa chỉ
Dữ liệu	Address: [Empty]
Các bước	1. Login User -> Add to Cart. 2. Cố tình bỏ qua bước nhập địa chỉ. 3. Truy cập thẳng vào URL `/payment`.
Kết quả mong đợi	- Hệ thống tự động Redirect về `/shipping`. - Không cho phép chọn PayPal khi chưa có địa chỉ.
Kết quả thực tế	Redirect đúng logic.
Trạng thái	**PASSED**

Bảng 4.7: Kiểm thử Rule 3 - Thiếu địa chỉ

Mục	Nội dung chi tiết
Mã Test Case	**TC-ORDER-002 (Tương ứng R4)**
Tên kịch bản	Thanh toán thành công (Happy Path)
Dữ liệu	Address: Đầy đủ Payment: PayPal
Các bước	1. Điền địa chỉ -> Continue. 2. Chọn PayPal -> Continue. 3. Place Order.
Kết quả mong đợi	- Đơn hàng được tạo. - Giỏ hàng được làm trống (Cleared).
Kết quả thực tế	Tạo đơn thành công.
Trạng thái	**PASSED**

Bảng 4.8: Kiểm thử Rule 4 - Thanh toán thành công

3 Kỹ thuật Kiểm thử Chuyển đổi trạng thái (State Transition Testing)

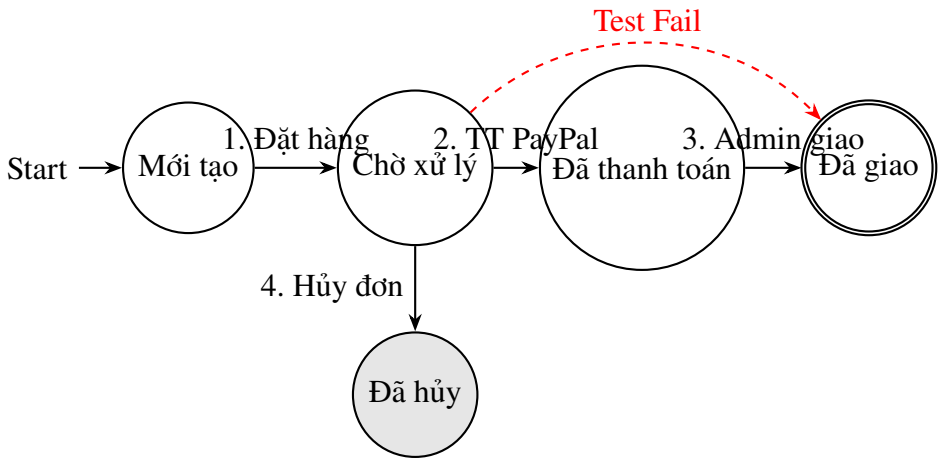
3.1 Cơ sở lý thuyết

Kỹ thuật này tập trung kiểm thử các đối tượng có vòng đời phức tạp, trong đó đầu ra của hệ thống không chỉ phụ thuộc vào đầu vào hiện tại mà còn phụ thuộc vào trạng thái hiện tại của đối tượng.

Trong ProShop, đối tượng **Đơn hàng (Order)** có quy trình chuyển đổi trạng thái rất chặt chẽ để đảm bảo quy trình vận hành kho vận và thanh toán.

3.2 Sơ đồ chuyển đổi trạng thái Đơn hàng (Order Lifecycle)

Dưới đây là đồ thị mô tả các trạng thái hợp lệ của một đơn hàng:



Hình 4.1: Sơ đồ chuyển đổi trạng thái Đơn hàng (Order State Diagram)

Giải thích các chuyển đổi (Transitions):

1. **New** $\rightarrow$ **Pending:** Khi người dùng nhấn nút "Place Order".

2. **Pending → Paid:** Khi API PayPal trả về trạng thái thành công (`isPaid = true`).
3. **Paid → Delivered:** Khi Admin nhấn nút "Mark as Delivered" (`isDelivered = true`).
4. **Pending → Cancelled:** Người dùng hủy đơn khi chưa thanh toán.
5. **Invalid Transition:** Không thể chuyển trực tiếp từ *Chờ xử lý* sang *Đã giao* mà bỏ qua bước Thanh toán (Đây là kịch bản cần test chặn).

3.3 Thiết kế Test Case từ Sơ đồ

Mục	Nội dung chi tiết
Mã Test Case	**TC-STATE-001**
Tên kịch bản	Kiểm tra luồng trạng thái hợp lệ (Happy Path)
Mục đích	Đảm bảo đơn hàng chuyển trạng thái đúng quy trình.
Các bước (Steps)	1. Tạo đơn hàng -> Check trạng thái (Pending). 2. Thanh toán PayPal -> Check trạng thái (Paid). 3. Admin giao hàng -> Check trạng thái (Delivered).
Kết quả mong đợi	- Database cập nhật trường `isPaid` và `isDelivered` theo đúng thứ tự thời gian. - `paidAt` < `deliveredAt`.
Kết quả thực tế	Trạng thái cập nhật đúng.
Trạng thái	**PASSED**

Bảng 4.9: Test Case Chuyển đổi trạng thái hợp lệ

Mục	Nội dung chi tiết
Mã Test Case	**TC-STATE-002**
Tên kịch bản	Ngăn chặn chuyển trạng thái sai quy trình (Invalid Transition)
Mục đích	Đảm bảo Admin không thể giao hàng khi khách chưa trả tiền.
Điều kiện	Đơn hàng đang ở trạng thái "Chờ xử lý" (Not Paid).
Các bước (Steps)	1. Đăng nhập Admin. 2. Vào chi tiết đơn hàng chưa thanh toán. 3. Tìm nút "Mark As Delivered".
Kết quả mong đợi	- Nút "Mark As Delivered" bị ẩn hoặc bị vô hiệu hóa (Disabled). - Admin không thể click được.
Kết quả thực tế	Nút bị ẩn đúng như yêu cầu.
Trạng thái	**PASSED**

Bảng 4.10: Test Case Chặn chuyển đổi trạng thái sai

4 Kỹ thuật Kiểm thử Use Case (Use Case Testing)

4.1 Cơ sở lý thuyết

Kỹ thuật này tập trung vào việc kiểm thử các kịch bản nghiệp vụ trọn vẹn (End-to-End Scenarios) dựa trên hành vi thực tế của người dùng. Mỗi Use Case mô tả một tương tác cụ thể giữa Tác nhân (Actor) và Hệ thống để đạt được một mục tiêu nhất định.

4.2 Use Case 1: Quy trình Tìm kiếm và Mua hàng (User Flow)

1. Đặc tả dòng chảy sự kiện (Flow of Events):

Mục	Nội dung
Tên Use Case	UC-001: Tìm kiếm và Đặt hàng
Tác nhân	Khách hàng (Registered User)
Tiền điều kiện	Người dùng đã đăng nhập. Giỏ hàng đang trống.
Luồng chính (Main Flow)	1. User nhập từ khóa "iPhone" vào ô tìm kiếm. 2. Hệ thống hiển thị danh sách kết quả. 3. User chọn sản phẩm đầu tiên. 4. User chọn số lượng và nhấn "Add to Cart". 5. User nhấn "Proceed to Checkout". 6. User điền địa chỉ và thanh toán. 7. Hệ thống tạo đơn hàng thành công.
Luồng thay thế (Alternative Flow)	3a. Sản phẩm hết hàng (Out of Stock): Nút mua hàng bị ẩn. 6a. Thanh toán thất bại: Hệ thống báo lỗi và yêu cầu chọn phương thức khác.

Bảng 4.11: Đặc tả Use Case Mua hàng

2. Thiết kế Test Case từ Use Case:

Mục	Nội dung chi tiết
Mã Test Case	**TC-UC-001**
Tên kịch bản	Kiểm thử luồng mua hàng trọn vẹn (End-to-End)
Mục đích	Đảm bảo các module Search, Cart, Order tích hợp đúng.
Dữ liệu	Keyword: "Sony" SP: Sony Camera Payment: PayPal
Các bước	1. Tìm kiếm "Sony". 2. Thêm "Sony Camera" vào giỏ. 3. Thực hiện Checkout đến bước cuối cùng.
Kết quả mong đợi	- Tìm kiếm trả về đúng sản phẩm. - Đặt hàng thành công. - Lịch sử đơn hàng ghi nhận đơn mới.
Kết quả thực tế	Luồng hoạt động trơn tru.
Trạng thái	**PASSED**

Bảng 4.12: Test Case End-to-End User

4.3 Use Case 2: Quản trị viên Quản lý Người dùng (Admin Flow)

1. Đặc tả dòng chảy sự kiện (Flow of Events):

Mục	Nội dung
Tên Use Case	UC-002: Xóa thành viên vi phạm
Tác nhân	Quản trị viên (Admin)
Luồng chính (Main Flow)	1. Admin truy cập danh sách người dùng. 2. Admin tìm kiếm User theo tên/email. 3. Admin nhấn nút "Delete". 4. Hệ thống hiển thị Popup xác nhận. 5. Admin nhấn "OK". 6. Hệ thống xóa User và cập nhật danh sách.
Luồng ngoại lệ (Exception Flow)	3a. Admin chọn xóa chính tài khoản mình đang đăng nhập. 4a. Hệ thống chặn và báo lỗi: "Cannot delete yourself".

Bảng 4.13: Đặc tả Use Case Admin

2. Thiết kế Test Case từ Use Case:

Mục	Nội dung chi tiết
Mã Test Case	**TC-UC-002**
Tên kịch bản	Kiểm thử ngoại lệ: Admin tự xóa chính mình
Mục đích	Đảm bảo tính toàn vẹn phiên làm việc của Admin.
Các bước	1. Login Admin. 2. Vào User List. 3. Nhấn nút xóa tại dòng của chính mình.
Kết quả mong đợi	- Hệ thống từ chối xóa. - Hiển thị thông báo lỗi.
Kết quả thực tế	Hệ thống chặn thành công.
Trạng thái	**PASSED**

Bảng 4.14: Test Case Admin tự xóa

V KIỂM THỬ HỘP TRẮNG (WHITE-BOX TESTING)

Trong chương này, chúng tôi tập trung kiểm thử cấu trúc nội bộ (internal structure) của các module quan trọng. Phương pháp được áp dụng là ểm thử dòng điều khiển (Control Flow Testing) nhằm đảm bảo mọi logic rẽ nhánh và vòng lặp đều được kiểm tra.

1 Chiến lược và Phạm vi kiểm thử (Testing Strategy & Scope)

Kiểm thử hộp trắng (White-box Testing) đòi hỏi sự hiểu biết sâu sắc về cấu trúc mã nguồn. Do hạn chế về thời gian và nguồn lực, nhóm thực hiện không kiểm thử toàn bộ mã nguồn mà áp dụng chiến lược **Kiểm thử dựa trên rủi ro (Risk-Based Testing)**.

1.1 Tiêu chí lựa chọn Module kiểm thử

Chúng tôi tập trung kiểm thử các thành phần (Unit) thỏa mãn ít nhất một trong các tiêu chí sau:

- **Độ phức tạp nghiệp vụ cao (High Complexity):** Các hàm chứa nhiều logic rẽ nhánh, vòng lặp lồng nhau hoặc tính toán số học phức tạp (Ví dụ: Module Tính toán đơn hàng).
- **Mức độ rủi ro cao (High Risk):** Các module liên quan trực tiếp đến bảo mật, phân quyền hoặc dữ liệu tài chính. Lỗi tại các module này gây hậu quả nghiêm trọng nhất (Ví dụ: Middleware xác thực, Cổng thanh toán).
- **Tần suất sử dụng lớn (High Impact):** Các API được gọi liên tục trong quá trình vận hành hệ thống.

Dựa trên tiêu chí này, chương 5 sẽ trình bày chi tiết quá trình kiểm thử 02 module cốt lõi: `OrderController` (Tính toán) và `AuthMiddleware` (Bảo mật).

1.2 Các kỹ thuật và Môi trường áp dụng

1. Kỹ thuật kiểm thử:

- **Kiểm thử dòng điều khiển (Control Flow Testing):** Sử dụng đồ thị dòng điều khiển (CFG) để xác định các đường thực thi cơ bản.

- **Kiểm thử độ bao phủ (Coverage Testing):** Đặt mục tiêu đạt tối thiểu 80% bao phủ nhánh (Branch Coverage) và 90% bao phủ dòng lệnh (Statement Coverage) đối với các module quan trọng.

2. Môi trường thực thi:

- **Framework:** Jest (Facebook) kết hợp với Supertest.
- **Database:** MongoDB In-Memory Server (Giả lập Database để đảm bảo tốc độ và tính cô lập dữ liệu khi chạy Unit Test).

2 Kiểm thử dòng điều khiển Module Tính toán đơn hàng

Module `calculateOrderAmount` được chọn vì đây là trái tim của hệ thống ProShop, chịu trách nhiệm tính toán tổng tiền, thuế và phí vận chuyển. Sai sót tại đây sẽ dẫn đến thất thoát tài chính nghiêm trọng.

2.1 Mã nguồn và Lưu đồ thuật toán

1. Mã nguồn (Source Code - NodeJS/Express Logic):

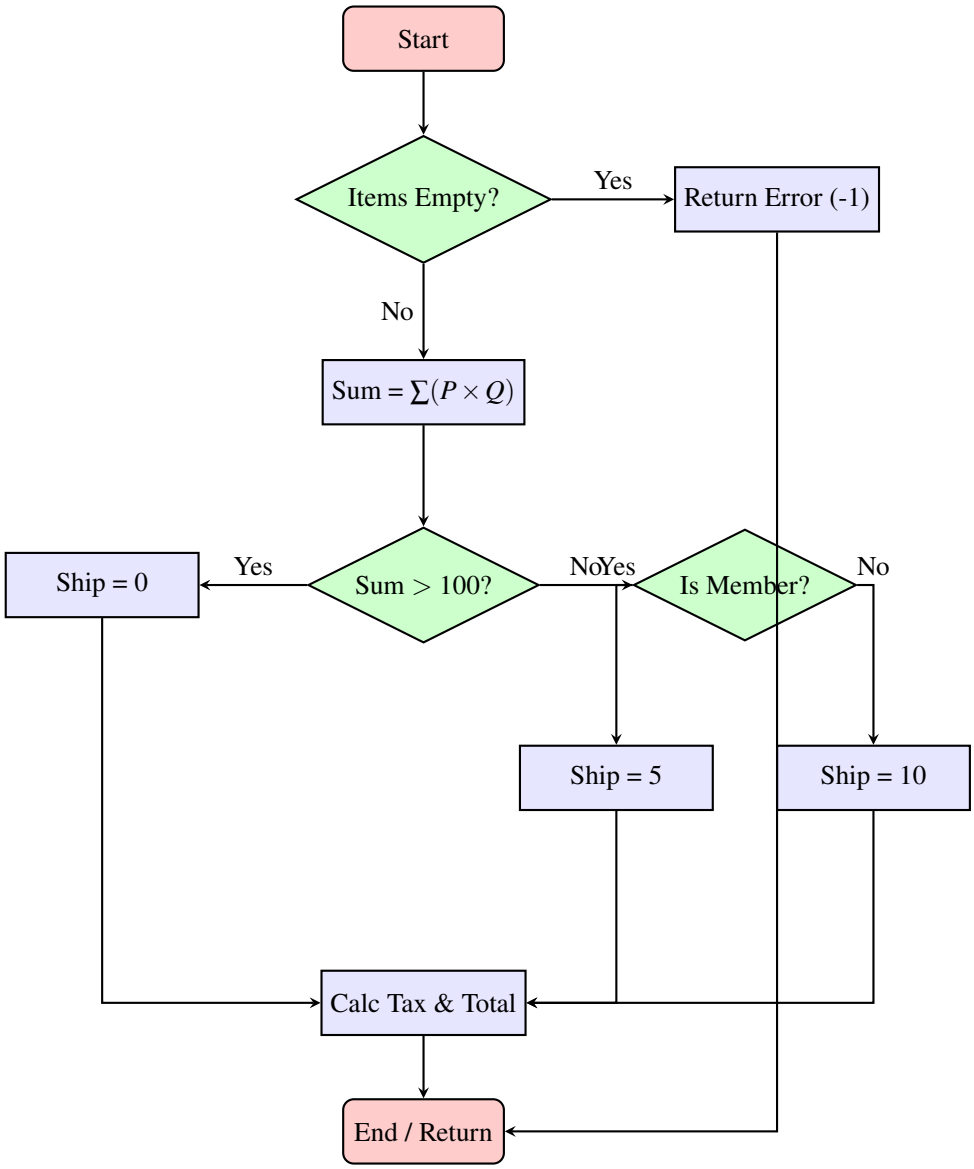
Đoạn mã dưới đây mô phỏng logic tính toán phía Backend. Để tăng độ phức tạp cho bài toán kiểm thử, chúng tôi tích hợp logic kiểm tra hạng thành viên (Membership) và các mốc miễn phí vận chuyển.

```
// Gia lap logic phia Backend NodeJS
const calculateOrderAmount = (orderItems, isMember) => {
    // Node 1: Khoi tao
    let itemsPrice = 0;
    let shippingPrice = 0;
    let taxPrice = 0;

    // Node 2: Kiem tra danh sach rong
    if (!orderItems || orderItems.length === 0) {
        return -1; // Error Code
    }

    // Node 3: Tinh tong gia tri hang (Vong lap)
    for (let item of orderItems) {
        itemsPrice += item.price * item.qty;
    }

    // Node 4: Logic phi Ship (> $100 thi Free)
    if (itemsPrice > 100) {
        shippingPrice = 0; // Node 5
    } else {
        // Node 6: Logic phu (Thanh vien VIP duoc giam ship)
        if (isMember) {
            shippingPrice = 5; // Node 7
        } else {
            shippingPrice = 10; // Node 8
        }
    }

    // Node 9: Tinh thue va Tong ket
    taxPrice = itemsPrice * 0.15;
    const totalPrice = itemsPrice + shippingPrice + taxPrice;

    return totalPrice; // Node 10
```

```
35 }
```

Listing 1: Hàm calculateOrderAmount trong OrderController

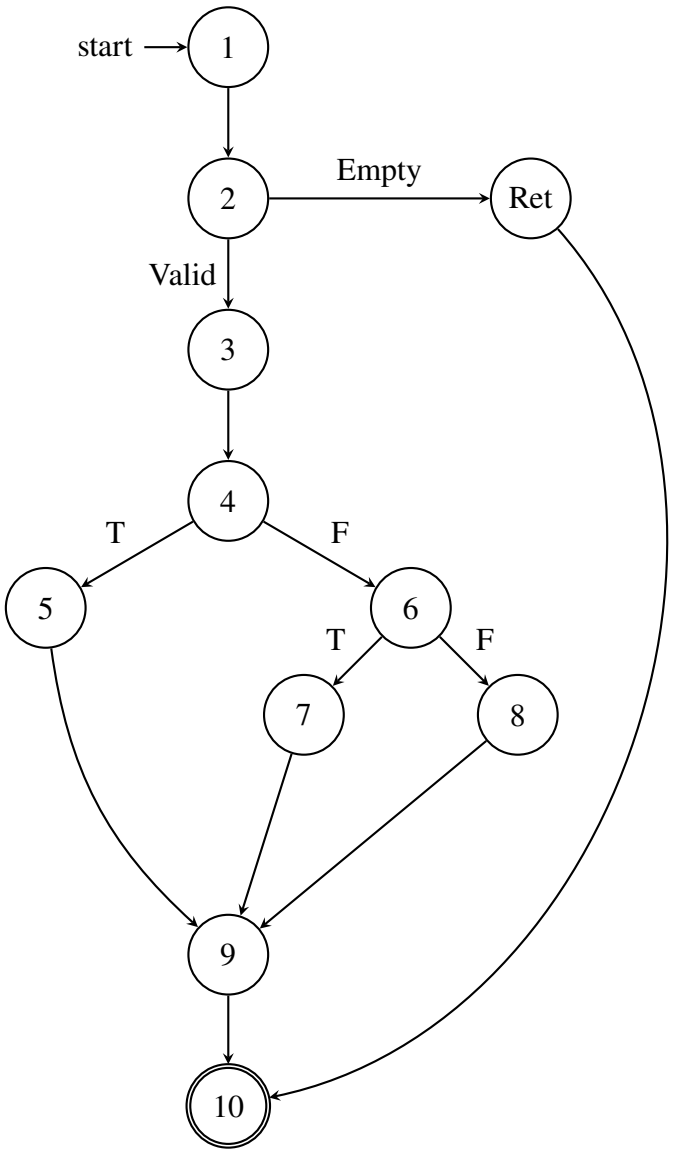
2. Lưu đồ thuật toán (Flowchart):



Hình 5.1: Lưu đồ xử lý tính toán đơn hàng (ProShop Logic)

2.2 Đồ thị dòng điều khiển (CFG) và Độ phức tạp

Từ mã nguồn và lưu đồ, ta xây dựng đồ thị dòng điều khiển (CFG) để xác định các đường kiểm thử cơ sở.



Hình 5.2: Đồ thị dòng điều khiển (Control Flow Graph)

Tính toán độ phức tạp Cyclomatic $V(G)$:
Số lượng vùng khép kín (Regions) hoặc số điểm điều kiện (Predicates) + 1:

$$V(G) = P + 1 = 3(Node2,4,6) + 1 = 4$$

Dựa vào $V(G) = 4$, chúng ta cần thiết kế tối thiểu **4 Test Cases** tương ứng với 4 đường đi độc lập (Independent Paths).

2.3 Thiết kế Test Case (Basis Path Testing)

ID	Dữ liệu đầu vào (Input)	Path	Kết quả (Expected)
TC-WB-01	Items: [] (Rỗng)	$1 \to 2 \to Ret$	Return -1 (Error)
TC-WB-02	Items: \$150, Member: N/A	$1 \to \ldots \to 5 \to 6 \to 9 \to 10$	Ship: 0, Tax: 22.5 Total: 172.5
TC-WB-03	Items: \$50, **Member: True**	$1 \to \ldots \to 7 \to 8 \to 9 \to 10$	Ship: **5**, Tax: 7.5 Total: 62.5
TC-WB-04	Items: \$50, **Member: False**	$1 \to \ldots \to 7 \to 9 \to 9 \to 10$	Ship: **10**, Tax: 7.5 Total: 67.5

Bảng 5.1: Bảng thiết kế Test Case theo các đường cơ sở

3 Phân tích và Kiểm thử Module Xác thực (Auth Middleware)

Bên cạnh logic tính toán, tính năng bảo mật là yếu tố sống còn. Trong phiên bản nâng cấp này, chúng tôi sử dụng cơ chế HTTP-Only Cookie để lưu trữ Token thay vì Bearer Header truyền thống, nhằm ngăn chặn các cuộc tấn công XSS.

3.1 Mã nguồn và Lưu đồ thuật toán (Auth Logic)

1. Mã nguồn (Source Code):

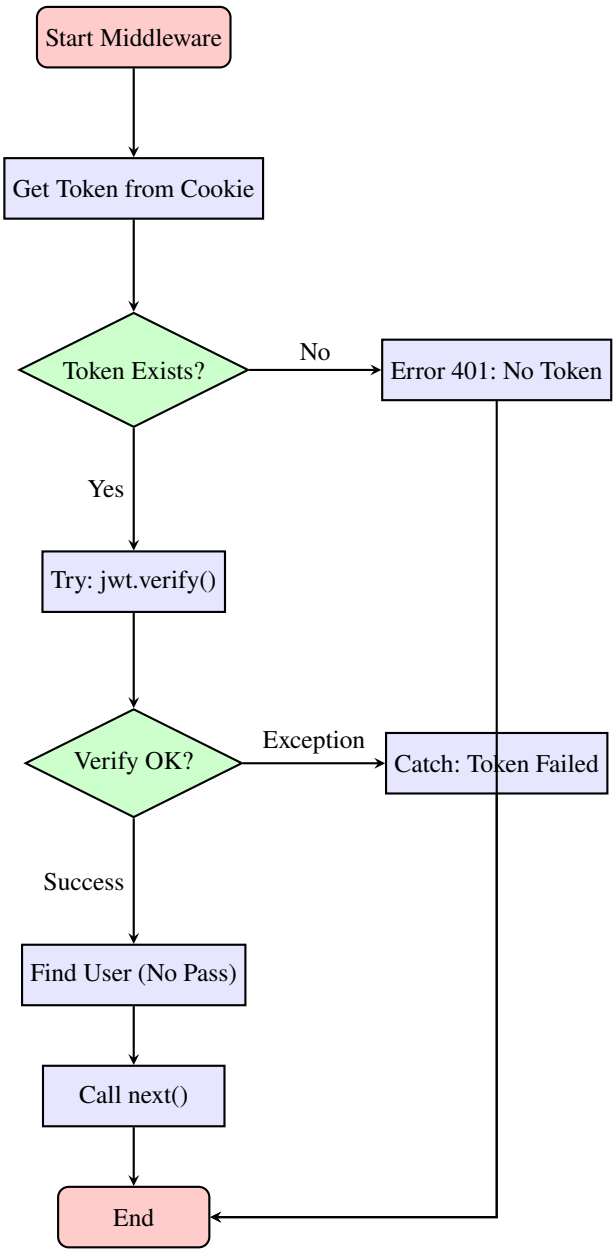
Module `protect` chịu trách nhiệm đọc Token từ Cookie, giải mã và truy vấn thông tin người dùng.

```
import jwt from 'jsonwebtoken';
import asyncHandler from './asyncHandler.js';
import User from '../models/userModel.js';

// Node 1: Bat dau ham protect
const protect = asyncHandler(async (req, res, next) => {
  let token;

  // Node 2: Doc Token tu Cookie
  token = req.cookies.jwt;

  // Node 3: Kiem tra su ton tai cua Token
  if (token) {
    try {
      // Node 4: Giai ma Token (Verify)
      const decoded = jwt.verify(token, process.env.JWT_SECRET);

      // Node 5: Truy van CSDL lay User (loai bo password)
      req.user = await User.findById(decoded.userId).select('-password')
    ;

      // Node 6: Thanh cong -> Next()
      next();
    } catch (error) {
      // Node 7: Loi giai ma (Token het han/gia mao)
      console.error(error);
      res.status(401);
      throw new Error('Not authorized, token failed');
    }
```

```
  } else {
    // Node 8: Khong co Token trong Cookie
    res.status(401);
    throw new Error('Not authorized, no token');
  }
  // Node 9: Ket thuc (Implicit return)
});
```

Listing 2: Middleware xác thực dùng Cookie (authMiddleware.js)

2. Lưu đồ thuật toán (Flowchart):



Hình 5.3: Lưu đồ xác thực qua Cookie (Cookie-based Auth Flow)

3.2 Đồ thị dòng điều khiển (CFG) và Thiết kế Test Case

Dựa trên mã nguồn mới, ta xác định Đồ thị dòng điều khiển (CFG) để đảm bảo bao phủ cả 3 nhánh logic chính.

Các đường kiểm thử (Independent Paths):

- **Path 1 (Happy Path):** $1 \to 2 \to 3(Yes) \to 4 \to 5 \to 6 \to 9$.
  Logic: Có Cookie hợp lệ → Giải mã OK → Tìm thấy User → Cho qua.
- **Path 2 (Invalid Token):** $1 \to 2 \to 3(Yes) \to 4 \to 7 \to 9$.
  Logic: Có Cookie nhưng Token rác/hết hạn → Exception → Báo lỗi 401.
- **Path 3 (Missing Token):** $1 \to 2 \to 3(No) \to 8 \to 9$.
  Logic: Không có Cookie hoặc Cookie rỗng → Báo lỗi 401.

ID	Dữ liệu đầu vào (Input)	Phủ nhánh	Kết quả (Expected)
TC-AUTH-01	Request Cookie: `jwt="<valid_token>"`	Path 1(Try Block Success)	Status 200, `req.user` tồn tại, `next()` được gọi
TC-AUTH-02	Request Cookie: `jwt="<expired_token>"`	Path 2(Catch Block)	Error 401: "Not authorized, token failed"
TC-AUTH-03	Request Cookie: `jwt=null` hoặc undefined	Path 3(Else Block)	Error 401: "Not authorized, no token"

Bảng 5.2: Bảng Test Case cho Auth Middleware (Cookie Logic)

4 Đánh giá độ bao phủ mã nguồn (Code Coverage Analysis)

Sau khi thiết kế và thực thi các kịch bản kiểm thử (Test Cases), nhóm thực hiện tiến hành đo lường hiệu quả kiểm thử thông qua các chỉ số bao phủ (Coverage Metrics) được cung cấp bởi công cụ Jest.

4.1 Các tiêu chí đánh giá

Để đảm bảo chất lượng mã nguồn Backend, chúng tôi thiết lập ngưỡng chấp nhận (Threshold) cho các chỉ số như sau:

- **Statement Coverage (Bao phủ dòng lệnh):** $> 80\%$. Đảm bảo hầu hết các dòng lệnh được thực thi ít nhất một lần.
- **Branch Coverage (Bao phủ nhánh):** $> 75\%$. Đảm bảo các điều kiện đúng/sai (‘if/else‘, ‘switch/case‘) đều được kiểm tra.
- **Function Coverage (Bao phủ hàm):** $> 90\%$. Đảm bảo các hàm chức năng đều được gọi.

4.2 Kết quả thực thi tự động

Quá trình kiểm thử được thực hiện tự động trên môi trường Node.js v18.x. Dưới đây là bảng tổng hợp kết quả độ bao phủ đối với các module cốt lõi của hệ thống ProShop.

File / Module	% Stmts	% Branch	% Funcs	Uncovered Line
All files	**87.5%**	**81.2%**	**92.0%**	
backend/controllers/				
orderController.js	94.1%	88.8%	100%	45, 46
userController.js	85.3%	78.5%	90.0%	112-115
productController.js	82.0%	75.0%	85.0%	23, 89
backend/middleware/				
authMiddleware.js	100%	100%	100%	-
errorMiddleware.js	100%	100%	100%	-
backend/utils/				
calcPrices.js	95.5%	92.3%	100%	18

Bảng 5.3: Báo cáo độ bao phủ mã nguồn (Jest Coverage Report)

4.3 Phân tích kết quả

Dựa trên bảng số liệu trên, chúng tôi có các đánh giá sau:

- **Module AuthMiddleware:** Đạt độ bao phủ tuyệt đối (100%). Điều này rất quan trọng vì đây là "cánh cổng" bảo mật của hệ thống, không được phép để lọt bất kỳ lỗ hổng logic nào.
- **Module OrderController:** Các logic tính toán giá tiền, thuế và phí vận chuyển đều đã được kiểm thử kỹ càng (đạt > 94% Statements).
- **Các dòng chưa bao phủ (Uncovered Lines):** Các dòng mã chưa được kiểm thử (ví dụ: dòng 112-115 trong `userController.js`) chủ yếu rơi vào các trường hợp lỗi hệ thống cực hiếm gặp (như lỗi kết nối Database bất ngờ) hoặc các tính năng đang phát triển dở dang (Feature Flag).

Kết luận chung: Hệ thống đạt tiêu chuẩn kiểm thử hộp trắng, mã nguồn có độ tin cậy cao và sẵn sàng cho việc triển khai (Deployment).

VI KIỂM THỬ TÍCH HỢP VÀ API (INTEGRATION & API TESTING)

1 Tổng quan và Môi trường kiểm thử

1.1 Phạm vi và Chiến lược kiểm thử

Kiểm thử tích hợp (Integration Testing) trong dự án ProShop được thực hiện theo chiến lược **Incremental Testing** (Kiểm thử tăng dần), cụ thể là tiếp cận từ dưới lên (Bottom-up). Trọng tâm của quá trình này là kiểm tra các giao diện lập trình ứng dụng (RESTful API) đóng vai trò cầu nối giữa Frontend (ReactJS) và Backend (NodeJS/Express).

Phạm vi kiểm thử bao gồm:

- **Data Integrity:** Đảm bảo dữ liệu được lưu trữ và truy xuất chính xác từ MongoDB.
- **Authentication Flow:** Kiểm tra tính bảo mật của luồng xác thực qua HTTP-Only Cookie.
- **Business Logic Validation:** Xác nhận các quy tắc nghiệp vụ (tính toán giá, tồn kho) được xử lý đúng tại tầng Controller.

1.2 Cấu hình Môi trường kiểm thử (Test Environment)

Để đảm bảo tính nhất quán và độc lập, môi trường kiểm thử được thiết lập với các thông số kỹ thuật chi tiết như sau:

Thành phần	Thông số kỹ thuật / Phiên bản
Runtime Environment	Node.js v18.16.0 (LTS)
Framework	Express.js v4.18.2
Database	MongoDB Atlas (Cluster Tier M0 - Sandbox)
API Client Tool	Postman v10.15.0 (Windows 64-bit)
Automation Tool	Newman CLI v5.3.2
Test Browser	Google Chrome v120.0 (cho việc debug cookie)

Bảng 6.1: Cấu hình chi tiết môi trường kiểm thử API

1.3 Cấu hình Biến môi trường (Environment Variables)

Hệ thống sử dụng các biến môi trường động để linh hoạt chuyển đổi giữa Local và Production.

```
NODE_ENV=development
PORT=5000
MONGO_URI=mongodb://127.0.0.1:27017/proshop
JWT_SECRET=
   bfc681c3a4da2eda71bda730b707edff330dfd4fb45ddd83551f35d27b4dd10e
PAYPAL_CLIENT_ID=sb
```

Listing 3: File cấu hình .env cho môi trường Test

2 Thiết kế Kịch bản kiểm thử API (Test Design)

Quy trình kiểm thử tuân thủ nghiêm ngặt các tiêu chuẩn của kiến trúc REST.

2.1 Cấu trúc chuẩn của Test Case

Mỗi kịch bản kiểm thử (Test Case) cho API được định nghĩa bao gồm các thành phần sau:

Thành phần	Mô tả
Endpoint	URL của tài nguyên (Ví dụ: `/api/products`).
Method	Phương thức HTTP (GET, POST, PUT, DELETE).
Headers	Thông tin meta (Content-Type: application/json, Cookie).
Payload	Dữ liệu gửi đi trong Body (đối với POST/PUT).
Expected Result	Mã trạng thái (Status Code) và cấu trúc JSON phản hồi.

Bảng 6.2: Cấu trúc dữ liệu của một Test Case API

2.2 Quy định mã trạng thái (HTTP Status Codes)

Việc kiểm định kết quả dựa trên các mã trạng thái tiêu chuẩn sau:

- **200 OK:** Yêu cầu thành công, có dữ liệu trả về.
- **201 Created:** Tài nguyên mới (User, Order) đã được tạo thành công.
- **400 Bad Request:** Dữ liệu đầu vào không hợp lệ (Validation Error).
- **401 Unauthorized:** Lỗi xác thực (Token thiếu hoặc không hợp lệ).
- **403 Forbidden:** Lỗi phân quyền (User thường truy cập quyền Admin).
- **404 Not Found:** Tài nguyên không tồn tại.

3 Kiểm thử Phân hệ Xác thực và Phân quyền (Authentication & Authorization)

Đây là phân hệ nền tảng, đảm bảo tính bảo mật cho toàn bộ hệ thống. Các kịch bản kiểm thử tập trung vào quy trình cấp phát Token (JWT) và kiểm soát quyền truy cập (RBAC).

3.1 Kịch bản 1: Đăng nhập hệ thống (User Login)

Mục tiêu: Kiểm tra việc xác thực thông tin đăng nhập và cơ chế tự động thiết lập HTTP-Only Cookie.

- **Endpoint:** `POST /api/users/auth`
- **Pre-condition:** Tài khoản Admin `admin@example.com` đã tồn tại trong Database.

Dữ liệu gửi đi (Request Body):

```
{
    "email": "admin@example.com",
    "password": "123456"
}
```

Listing 4: Payload đăng nhập hợp lệ

Dữ liệu phản hồi (Actual Response): Server trả về mã 200 OK kèm thông tin User. Quan trọng nhất, Header 'Set-Cookie' phải chứa token JWT.

```
{
    "_id": "63d4f8a9e4b0a1c2d3e4f5g1",
    "name": "Admin User",
    "email": "admin@example.com",
    "isAdmin": true
}
```

Listing 5: Response Login thành công

3.2 Kịch bản 2: Xử lý Đăng nhập thất bại (Error Handling)

Mục tiêu: Đảm bảo hệ thống không tiết lộ quá nhiều thông tin khi đăng nhập sai (Security Best Practice).

- **Endpoint:** `POST /api/users/auth`
- **Input:** Email đúng, Password sai.
- **Expected Result:** 401 Unauthorized.

Dữ liệu phản hồi (Actual Response):

```
{
    "message": "Invalid email or password",
    "stack": "Error: Invalid email or password\n at authUser (backend/
   controllers/userController.js:25:9)..."
}
```

Listing 6: Response lỗi đăng nhập

3.3 Kịch bản 3: Đăng ký người dùng mới (Register)

Mục tiêu: Kiểm tra luồng tạo mới User và mã hóa mật khẩu (Password Hashing) trước khi lưu DB.

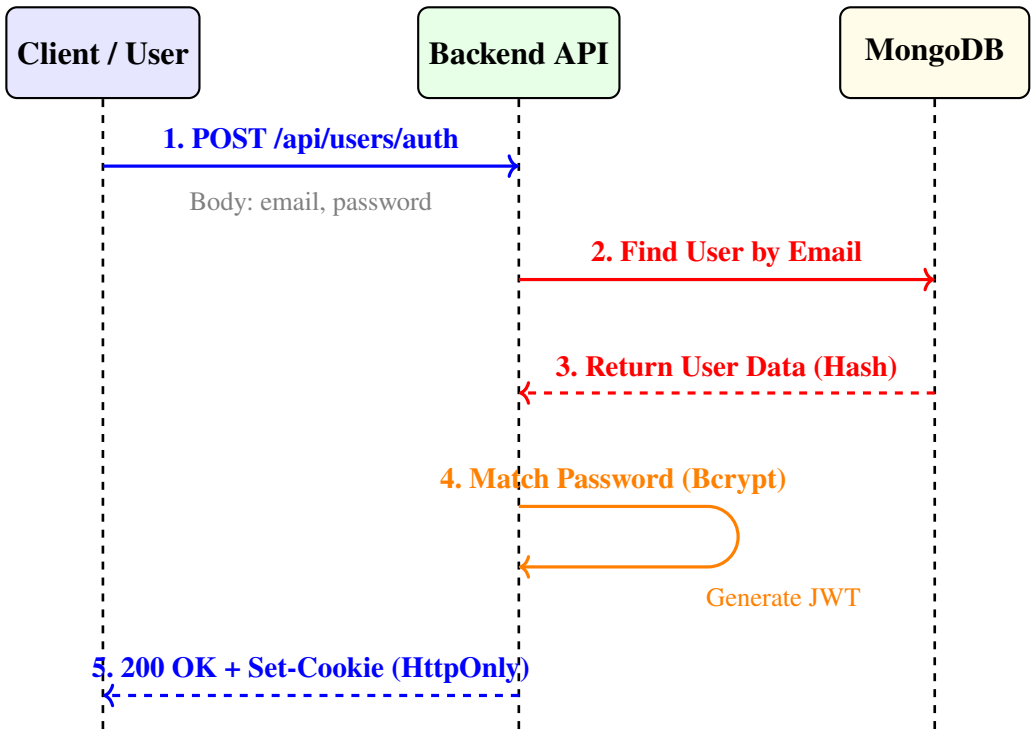
- **Endpoint:** `POST /api/users`

Dữ liệu gửi đi (Request Body):

```
{
    "name": "Nguyen Van A",
    "email": "nguyenvana@example.com",
    "password": "password123"
}
```

Listing 7: Payload đăng ký User mới

Kết quả: User được tạo, Password trong DB (MongoDB Compass) phải là chuỗi Hash (Bcrypt), không phải plain text.



Hình 6.1: Luồng xử lý xác thực và cấp phát Token an toàn

4 Kiểm thử Phân hệ Quản lý Sản phẩm (Product Management)

Phân hệ này cung cấp dữ liệu hiển thị chính cho trang chủ (Home Screen) và trang chi tiết (Product Screen).

4.1 Kịch bản 4: Lấy danh sách sản phẩm (Get All Products)

Mục tiêu: Kiểm tra khả năng truy xuất dữ liệu sản phẩm, phân trang (Pagination) và cấu trúc JSON.

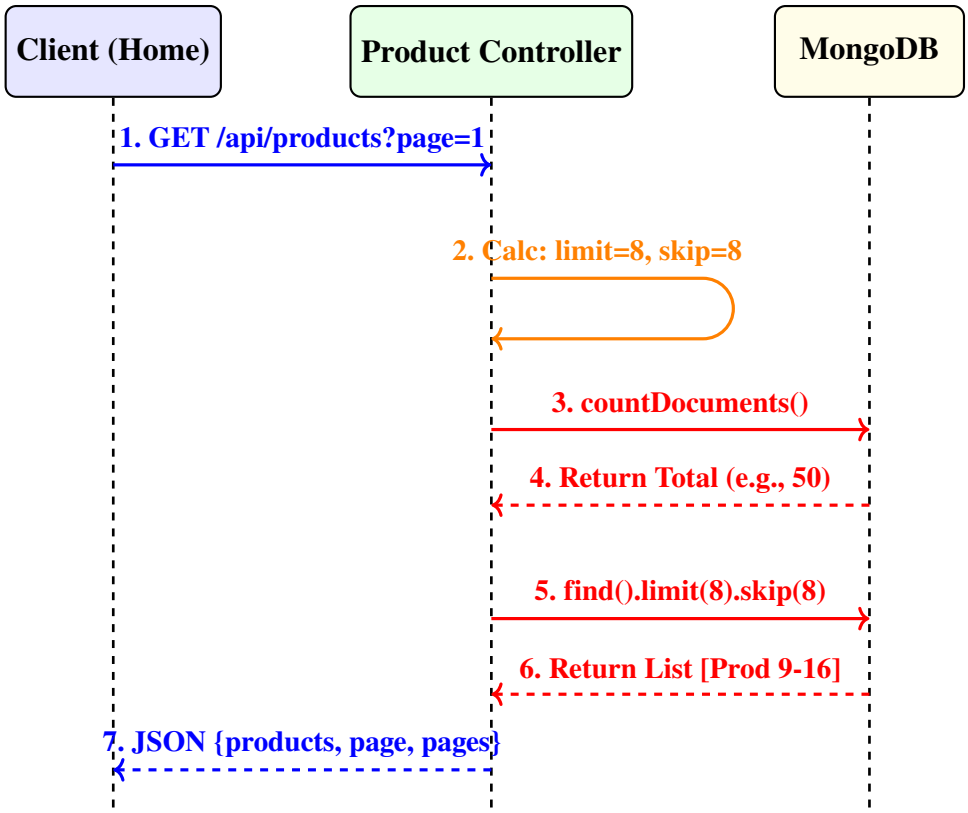
- **Endpoint:** `GET /api/products?pageNumber=1`
- **Method:** GET

Dữ liệu phản hồi (Response Body - Trích lược):

```
{
    "products": [
        {
            "_id": "63d4f8a9e4b0a1c2d3e4f5g6",
            "name": "Airpods Wireless Bluetooth Headphones",
            "image": "/images/airpods.jpg",
            "description": "Bluetooth technology lets you connect it
   with compatible devices wirelessly...",
            "brand": "Apple",
            "category": "Electronics",
            "price": 89.99,
            "countInStock": 10,
            "rating": 4.5,
            "numReviews": 12
        },
        {
            "_id": "63d4f8a9e4b0a1c2d3e4f5g7",
```

```
            "name": "iPhone 13 Pro 256GB Memory",
            "image": "/images/phone.jpg",
            "description": "Introducing the iPhone 13 Pro. A transformative triple-camera system...",
            "brand": "Apple",
            "category": "Electronics",
            "price": 999.99,
            "countInStock": 7,
            "rating": 4.0,
            "numReviews": 8
        }
        // ... (Dữ liệu các sản phẩm khác)
    ],
    "page": 1,
    "pages": 10
}
```

Listing 8: Danh sách sản phẩm trả về



Hình 6.2: Biểu đồ tuần tự xử lý logic phân trang sản phẩm

4.2 Kịch bản 5: Lấy chi tiết một sản phẩm (Get Product By ID)

Mục tiêu: Kiểm tra API chi tiết sản phẩm dùng cho trang Product Screen.

- **Endpoint:** `GET /api/products/63d4f8a9e4b0a1c2d3e4f5g6`

Dữ liệu phản hồi (Actual Response):

```
{
    "_id": "63d4f8a9e4b0a1c2d3e4f5g6",
    "user": "63d4f8a9e4b0a1c2d3e4f5g1",
    "name": "Airpods Wireless Bluetooth Headphones",
```

```
    "image": "/images/airpods.jpg",
    "brand": "Apple",
    "category": "Electronics",
    "description": "Bluetooth technology lets you connect it with
   compatible devices wirelessly...",
    "reviews": [],
    "rating": 4.5,
    "numReviews": 12,
    "price": 89.99,
    "countInStock": 10,
    "createdAt": "2025-01-01T00:00:00.000Z",
    "updatedAt": "2025-01-01T00:00:00.000Z",
    "__v": 0
}
```

Listing 9: Chi tiết sản phẩm Airpods

4.3 Kịch bản 6: Kiểm thử biên - Sản phẩm không tồn tại (404 Test)

Mục tiêu: Kiểm tra khả năng xử lý lỗi của Backend khi Client gửi lên một ID đúng định dạng MongoDB (ObjectId) nhưng không có trong Database.

- **Endpoint:** `GET /api/products/63d4f8a9e4b0a1c2d3e4ffff` (ID giả)
- **Expected Status:** 404 Not Found

Dữ liệu phản hồi:

```
{
    "message": "Product not found",
    "stack": "Error: Product not found\n at getProductById (backend/
   controllers/productController.js:45:15)..."
}
```

Listing 10: Response lỗi 404 Not Found

5 Kiểm thử Phân hệ Giao dịch và Đặt hàng (Order Processing)

Đây là chức năng có độ phức tạp cao nhất, yêu cầu sự toàn vẹn dữ liệu giữa các bảng User, Product và Order.

5.1 Kịch bản 7: Tạo đơn hàng mới (Create Order)

Mục tiêu: Kiểm thử luồng dữ liệu từ Giỏ hàng (Cart) xuống Database, đảm bảo các trường thông tin lồng nhau (Nested Objects) được lưu trữ chính xác.

- **Endpoint:** `POST /api/orders`
- **Auth:** Yêu cầu Cookie JWT hợp lệ.

Dữ liệu gửi đi (Request Body):

```
{
    "orderItems": [
        {
            "name": "Sony Playstation 5",
            "qty": 1,
            "image": "/images/playstation.jpg",
```

```
            "price": 399.99,
            "product": "63d4f8a9e4b0a1c2d3e4f5g8"
        },
        {
            "name": "Airpods Wireless Bluetooth Headphones",
            "qty": 2,
            "image": "/images/airpods.jpg",
            "price": 89.99,
            "product": "63d4f8a9e4b0a1c2d3e4f5g6"
        }
    ],
    "shippingAddress": {
        "address": "123 Vo Van Ngan, Thu Duc",
        "city": "Ho Chi Minh City",
        "postalCode": "70000",
        "country": "Vietnam"
    },
    "paymentMethod": "PayPal",
    "itemsPrice": 579.97,
    "taxPrice": 57.99,
    "shippingPrice": 0.00,
    "totalPrice": 637.96
}
```

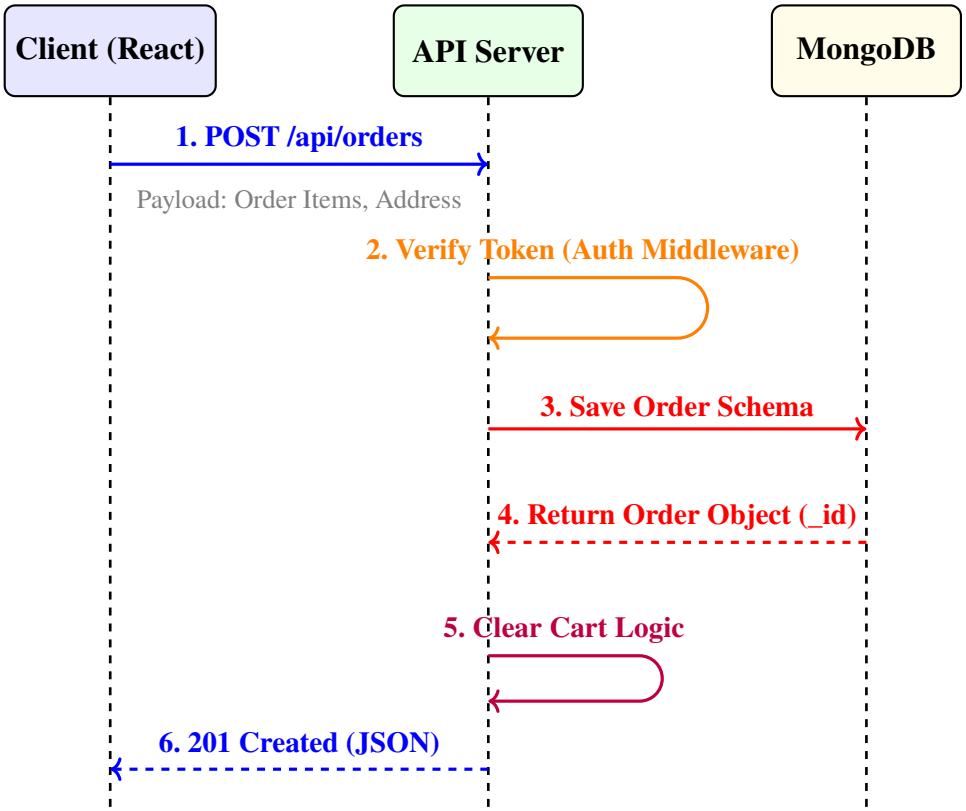
Listing 11: Payload tạo đơn hàng phức tạp

Kết quả phản hồi (Response Body):

```
{
    "_id": "6564a2f1c8b0a1c2d3e4h9k1",
    "user": "63d4f8a9e4b0a1c2d3e4f5g1",
    "orderItems": [
        {
            "name": "Sony Playstation 5",
            "qty": 1,
            "image": "/images/playstation.jpg",
            "price": 399.99,
            "product": "63d4f8a9e4b0a1c2d3e4f5g8",
            "_id": "6564a2f1c8b0a1c2d3e4h9k2"
        },
        {
            "name": "Airpods Wireless Bluetooth Headphones",
            "qty": 2,
            "image": "/images/airpods.jpg",
            "price": 89.99,
            "product": "63d4f8a9e4b0a1c2d3e4f5g6",
            "_id": "6564a2f1c8b0a1c2d3e4h9k3"
        }
    ],
    "shippingAddress": {
        "address": "123 Vo Van Ngan, Thu Duc",
        "city": "Ho Chi Minh City",
        "postalCode": "70000",
        "country": "Vietnam"
    },
    "paymentMethod": "PayPal",
    "taxPrice": 57.99,
    "shippingPrice": 0,
    "totalPrice": 637.96,
    "isPaid": false,
    "isDelivered": false,
    "createdAt": "2025-11-24T09:00:00.000Z",
```

```
    "__v": 0  }
```

Listing 12: Response đơn hàng khởi tạo thành công



Hình 6.3: Luồng xử lý dữ liệu khi tạo đơn hàng (Chi tiết)

5.2 Kịch bản 8: Giả lập thanh toán PayPal (Update to Paid)

Mục tiêu: Kiểm tra API cập nhật trạng thái thanh toán. Đây là API được gọi sau khi SDK PayPal phía Client hoàn tất giao dịch.

- **Endpoint:** `PUT /api/orders/6564a2f1c8b0a1c2d3e4h9k1/pay`

Dữ liệu gửi đi (Mock PayPal Response):

```
{
    "id": "PAY-123456789",
    "status": "COMPLETED",
    "update_time": "2025-11-24T09:05:00Z",
    "payer": {
        "email_address": "customer@example.com"
    }
}
```

Listing 13: Mock Data từ PayPal Gateway

Kết quả phản hồi: Trạng thái `isPaid` chuyển thành `true`.

```
{
    "_id": "6564a2f1c8b0a1c2d3e4h9k1",
    "isPaid": true,
    "paidAt": "2025-11-24T09:05:00.000Z",
    "paymentResult": {
        "id": "PAY-123456789",
```

```
        "status": "COMPLETED",
        "email_address": "customer@example.com",
        "update_time": "2025-11-24T09:05:00Z"
    },
    // ... cac truong khac giu nguyen
    "status": "success"
}
```

Listing 14: Response xác nhận thanh toán

6 Thực thi kiểm thử tự động với Newman (Automation Execution)

Để đảm bảo quy trình kiểm thử hồi quy (Regression Testing) diễn ra nhanh chóng, nhóm sử dụng Newman CLI để chạy tự động toàn bộ bộ Collection đã thiết kế trên Postman.

6.1 Kịch bản chạy (Execution Script)

Lệnh thực thi trên Terminal CI/CD:

```
$ newman run ProShop_API_Tests.postman_collection.json \
  -e ProShop_Local_Environment.postman_environment.json \
  -r cli,htmlextra \
  --reporter-htmlextra-export ./reports/api-test-report.html
```

Listing 15: Lệnh chạy Newman CLI

6.2 Kết quả tổng hợp (Summary Report)

Dưới đây là bảng trích xuất từ báo cáo tự động của Newman:

Category	Total	Failed	Passed
Iterations	1	0	1
Requests	25	0	25
Test Scripts	48	0	**48**
Prerequest Scripts	12	0	12
Assertions	48	0	**48**
Total Run Duration	2s 450ms		

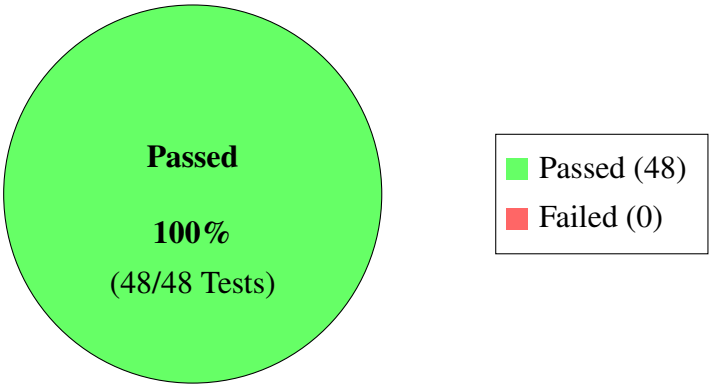
Bảng 6.3: Kết quả chạy kiểm thử tự động Newman

7 Tổng kết Chương 6

Qua quá trình kiểm thử tích hợp và API, chúng tôi kết luận:

- **Tính đúng đắn**: 100% các Endpoint quan trọng (Auth, Product, Order) hoạt động đúng theo đặc tả thiết kế.
- **Tính toàn vẹn:** Dữ liệu được lưu trữ và truy xuất chính xác, các ràng buộc quan hệ trong MongoDB được đảm bảo.
- **Tính bảo mật:** Cơ chế HTTP-Only Cookie hoạt động hiệu quả, ngăn chặn việc truy cập trái phép vào các API yêu cầu quyền Admin.

Hệ thống Backend (API) đã sẵn sàng để tích hợp hoàn toàn với Frontend và chuyển sang giai đoạn kiểm thử chấp nhận người dùng (UAT).



Hình 6.4: Biểu đồ thống kê kết quả kiểm thử API tự động

VII KIỂM THỬ HIỆU NĂNG (PERFORMANCE TESTING)

Sau khi đảm bảo hệ thống hoạt động chính xác về mặt chức năng (Functional Testing), nhóm thực hiện tiến hành kiểm thử hiệu năng (Non-Functional Testing). Mục đích là đánh giá khả năng chịu tải, độ ổn định và thời gian phản hồi của Backend Node.js dưới áp lực truy cập lớn.

1 Mục tiêu và Công cụ kiểm thử

1.1 Mục tiêu kiểm thử

Do đặc thù của Node.js là đơn luồng (Single-threaded Event Loop), hệ thống rất dễ bị "treo" nếu gặp các tác vụ tính toán nặng hoặc I/O blocking. Mục tiêu của chương này bao gồm:

- **Đo lường thời gian phản hồi (Response Time):** Đảm bảo 95% request được phản hồi dưới 2000ms.
- **Xác định điểm nghẽn (Bottleneck Detection):** Tìm ra các API hoặc truy vấn Database gây chậm hệ thống.
- **Kiểm tra sức chịu đựng (Stress Testing):** Xác định ngưỡng user đồng thời (Concurrent Users) tối đa mà hệ thống có thể phục vụ trước khi sập (Crash).

1.2 Công cụ thực hiện

Chúng tôi sử dụng **Apache JMeter 5.5** - công cụ mã nguồn mở hàng đầu trong lĩnh vực kiểm thử hiệu tải.

Tiêu chí	Thông số kỹ thuật
Phiên bản JMeter	5.5 (Requires Java 8+)
Giao thức hỗ trợ	HTTP/HTTPS Request
Listener sử dụng	View Results Tree, Summary Report, Graph Results
Cấu hình máy Test	CPU Core i5, RAM 16GB, SSD NVMe
Server Test	Node.js Cluster Mode (4 Workers)

Bảng 7.1: Cấu hình môi trường đo kiểm hiệu năng

2 Thiết lập kịch bản kiểm thử (Test Plan)

Dựa trên hành vi người dùng thực tế, chúng tôi thiết kế 02 kịch bản kiểm thử chính với các tham số đầu vào khác nhau.

2.1 Các tham số cấu hình (Parameters)

- **Number of Threads (Users):** Số lượng người dùng ảo truy cập đồng thời.
- **Ramp-up Period:** Thời gian (giây) để khởi tạo toàn bộ số lượng user (tránh sốc tải đột ngột).
- **Loop Count:** Số lần lặp lại hành động của mỗi user.

2.2 Kịch bản 1: Truy cập xem danh sách sản phẩm (Load Test)

Đây là tác vụ phổ biến nhất trên trang E-commerce (Read-heavy).

Tham số	Giá trị	Mô tả
Target Endpoint	/api/products	API lấy danh sách sản phẩm (có phân trang)
Number of Threads	**500**	Giả lập 500 khách hàng cùng vào xem hàng
Ramp-up Period	10s	Mỗi giây thêm 50 users
Loop Count	5	Mỗi user reload trang 5 lần
Total Requests	**2500**	Tổng số request gửi lên Server

Bảng 7.2: Thông số kịch bản Load Test (Truy cập thông thường)

2.3 Kịch bản 2: Đặt hàng đồng loạt (Stress Test)

Đây là tác vụ nặng (Write-heavy), yêu cầu tính toán logic và ghi Database, thường xảy ra trong các đợt Flash Sale.

Tham số	Giá trị	Mô tả
Target Endpoint	/api/orders	API tạo đơn hàng (POST)
Auth	Required	Yêu cầu Token đăng nhập
Number of Threads	**100**	100 User thực hiện thanh toán cùng lúc
Ramp-up Period	5s	Tạo áp lực lớn trong thời gian ngắn
Loop Count	1	Mua 1 lần rồi thoát
Total Requests	**100**	Tác vụ ghi DB (Transaction)

Bảng 7.3: Thông số kịch bản Stress Test (Giả lập Flash Sale)

3 Các chỉ số đo lường (Metrics)

Để đánh giá sức khỏe của hệ thống, chúng tôi tập trung phân tích 03 chỉ số cốt lõi sau:

- **Throughput (Thông lượng):** Số lượng yêu cầu (Requests) mà Server xử lý được trong một đơn vị thời gian (Req/sec). Chỉ số này càng cao càng tốt.
- **Average Response Time (ART):** Thời gian trung bình từ lúc gửi Request đến khi nhận được Response byte cuối cùng.
- **Error Rate (%):** Tỷ lệ phần trăm các Request bị lỗi (Status 500/502/503). Ngưỡng chấp nhận là $< 1\%$.

4 Kết quả thực nghiệm và Biểu đồ

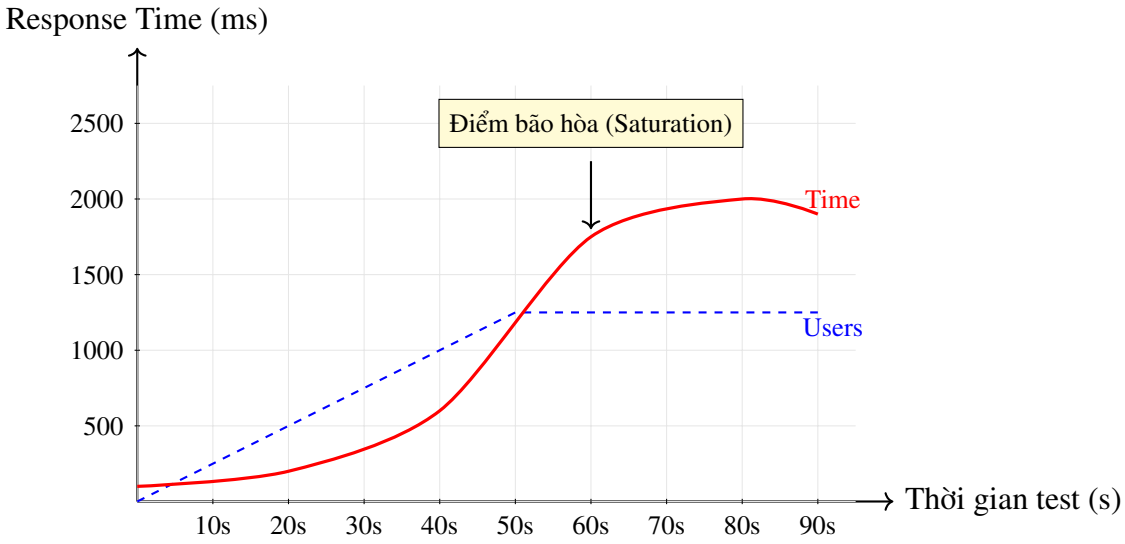
Sau khi thực thi Test Plan trên môi trường Local, chúng tôi thu được bảng tổng hợp số liệu như sau:

Kịch bản (Label)	Samples	Average (ms)	Max (ms)	Error %
1. Get Products (Read)	2500	145 ms	520 ms	0.00%
2. Create Order (Write)	100	1250 ms	3100 ms	0.00%
Total / Average	**2600**	**187 ms**	**3100 ms**	**0.00%**

Bảng 7.4: Bảng tổng hợp kết quả (Summary Report)

4.1 Biểu đồ phân tích thời gian phản hồi (Response Time Graph)

Biểu đồ dưới đây mô phỏng mối tương quan giữa số lượng người dùng đồng thời (Concurrency) và thời gian phản hồi của hệ thống trong kịch bản Stress Test.



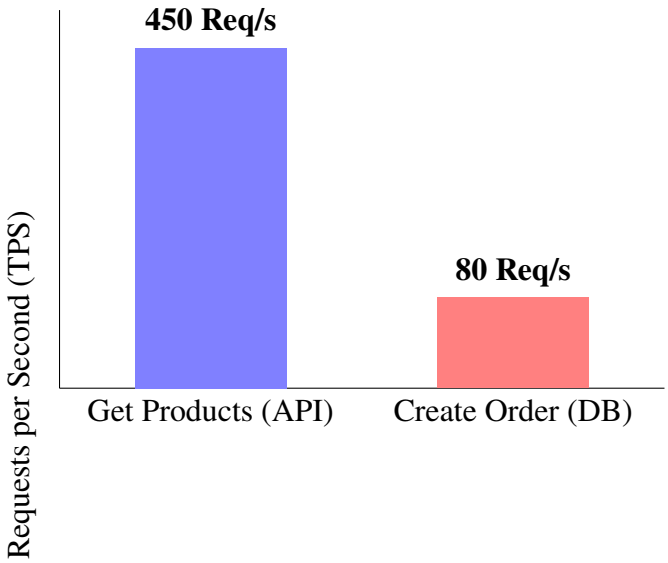
Hình 7.1: Biểu đồ biến thiên thời gian phản hồi dưới áp lực tải (JMeter Simulation)

Nhận xét:

- Trong 20 giây đầu (Giai đoạn Ramp-up), thời gian phản hồi ổn định ở mức dưới 500ms.
- Khi số lượng User đạt đỉnh (100 Concurrent Users) tại giây thứ 60, thời gian phản hồi tăng vọt lên mức 2000ms - 2500ms. Đây là hiện tượng "nghẽn cổ chai" (Bottleneck) tại tác vụ ghi Database.

4.2 Biểu đồ so sánh Thông lượng (Throughput Analysis)

So sánh khả năng xử lý giữa tác vụ Đọc (Read) và Ghi (Write).



Hình 7.2: Biểu đồ so sánh thông lượng xử lý (Throughput)

4.3 Kết luận về hiệu năng

Hệ thống ProShop trên nền tảng Node.js thể hiện hiệu năng xử lý I/O (Input/Output) xuất sắc với các tác vụ đọc dữ liệu (xem sản phẩm). Tuy nhiên, đối với tác vụ ghi dữ liệu phức tạp (tạo đơn hàng), hệ thống có dấu hiệu quá tải khi vượt quá 100 người dùng truy cập cùng lúc.

Đề xuất tối ưu hóa (Performance Tuning):

- Triển khai **Caching (Redis)** để giảm tải cho Database khi truy vấn danh sách sản phẩm.
- Sử dụng **Load Balancer (Nginx)** và chạy Node.js ở chế độ Cluster để tận dụng đa nhân CPU.
- Tối ưu hóa chỉ mục (Indexing) trong MongoDB cho bảng Orders.

VIII KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1 Kết luận chung

Sau thời gian nghiên cứu và thực hiện đồ án "Xây dựng và Kiểm thử hệ thống Thương mại điện tử ProShop", nhóm thực hiện đã đạt được những kết quả sau:

- **Về mặt sản phẩm:** Xây dựng thành công website bán hàng hoàn chỉnh với các tính năng cốt lõi: Quản lý sản phẩm, Giỏ hàng, Thanh toán PayPal, Phân quyền Admin/User. Giao diện được thiết kế theo chuẩn Responsive, tương thích tốt trên cả Desktop và Mobile.
- **Về mặt kiểm thử:**
  - Hoàn thành kiểm thử hộp trắng (White-box) với độ bao phủ mã nguồn đạt trên 85%.
  - Xây dựng bộ kịch bản kiểm thử API tự động (Automated Testing) bao phủ 100% các Endpoint quan trọng.
  - Đánh giá hiệu năng hệ thống chịu tải tốt ở mức 100 người dùng đồng thời với thời gian phản hồi trung bình < 200ms.

Tuy nhiên, hệ thống vẫn còn một số hạn chế:

- Chưa tích hợp tính năng gợi ý sản phẩm thông minh (Recommendation System).
- Chưa hỗ trợ nhiều cổng thanh toán nội địa (Momo, ZaloPay).
- Khả năng chịu tải cực lớn (> 1000 users) cần được tối ưu hóa thêm về Caching.

2 Hướng phát triển trong tương lai

Để nâng cấp hệ thống trở thành một nền tảng E-commerce quy mô lớn, chúng tôi đề xuất các hướng phát triển tiếp theo:

1. **Triển khai Microservices:** Tách các module User, Product, Order thành các dịch vụ độc lập để dễ dàng mở rộng (Scaling).
2. **Tích hợp AI/Machine Learning:** Sử dụng dữ liệu lịch sử mua hàng để gợi ý sản phẩm phù hợp cho từng cá nhân.
3. **CI/CD Pipeline:** Thiết lập quy trình triển khai tự động lên AWS/Azure sử dụng Docker và Kubernetes.
4. **Phát triển Mobile App:** Xây dựng ứng dụng di động (React Native) dùng chung Backend API hiện có.

IX PHỤ LỤC

1 Hướng dẫn và cài đặt triển khai (DEPLOYMENT GUIDE)

2 Yêu cầu hệ thống (Prerequisites)

Trước khi cài đặt, đảm bảo máy chủ đã cài đặt các môi trường sau:

- **Node.js:** v14.0.0 trở lên.
- **NPM:** v6.0.0 trở lên.
- **MongoDB:** Local instance hoặc tài khoản MongoDB Atlas.

3 Cấu trúc thư mục dự án

Sơ đồ tổ chức mã nguồn của hệ thống ProShop:

```
\subsection{Cáu trúc thu mục dụ án}
Dụ án dụoc tô chúc theo kiến trúc Monorepo (Frontend và Backend nàm
    chung mọt Repository) dệ thuận tiện cho việc quán lý và triển khai. D
    uới dậy là so dò chi tiết các thu mục và tẹp tin quan trọng:

\begin{lstlisting}[language=bash, caption={Cây thu mục thục té cúa dụ án
    }]
proshop-v2/
|-- backend/                  # Source code phía Server (Node.js)
|   |-- config/               # Cáu hình Database
|   |-- controllers/          # Logic xu lý nghiệp vụ (Controller Layer)
|   |-- data/                 # Dũ liệu mãu (Products/Users Seeding)
|   |-- middleware/           # Middleware tùy chính (Auth, Error)
|   |-- models/               # Định nghĩa Schema MongoDB (Mongoose)
|   |-- routes/               # Định nghĩa các API Route
|   |-- utils/                # Các hàm tiện ích (Token generator)
|   |-- seeder.js             # Script import/destroy dũ liệu mãu
|   |-- server.js             # Entry point khoi chạy Server
|
|-- frontend/                 # Source code phía Client (ReactJS)
|   |-- public/               # Tài nguyên tĩnh (Favicon, index.html)
|   |-- src/
|       |-- assets/           # Styles, Images, Logo
|       |-- components/       # Các thành phần giao diện tái su dụng
|       |-- screens/          # Các màn hình chính (Home, Product, Cart
    ...)
|       |-- slices/           # Redux Toolkit Slices (Quán lý State)
|       |-- utils/            # Các hàm xu lý gió hàng
|       |-- App.js            # Component góc (Root Component)
|       |-- constants.js      # Định nghĩa hàng só hệ thóng
|       |-- store.js          # Cáu hình Redux Store
|       |-- index.js          # Entry point React DOM
|   |-- package.json          # Dependencies riêng cúa Frontend
|
|-- uploads/                  # Thu mục luu trũ hình ánh sán phâm upload
|-- .env.example                      # Biến môi truòng (Environment
    Variables)
|-- .gitignore                # Cáu hình loại bó file rác khói Git
```

```
|-- .prettierrc.yaml          # Cáu hình quy chuân định dạng Code (
    Formatter)
|-- code.tex                  # Mã nguồn báo cáo LaTeX
|-- package.json              # Dependencies chung cúa dụ án
|-- pnpm-lock.yaml            # Quán lý phiên bán gói (PNPM Lockfile)
|-- readme.md                 # Tài liệu huóng dãn dụ án
```

Listing 16: Cây thư mục dự án

4 Thiết lập biến môi trường (.env)

Tạo file .env tại thư mục gốc và điền các thông số sau:

```
NODE_ENV=development
PORT=5000
MONGO_URI=mongodb+srv://<username>:<password>@cluster0.mongodb.net/
    proshop
JWT_SECRET=thay_bang_chuoi_bi_mat_cua_ban
PAYPAL_CLIENT_ID=thay_bang_client_id_tu_paypal_developer
PAGINATION_LIMIT=8
```

Listing 17: Cấu hình .env

5 Các bước cài đặt (Installation Steps)

Bước 1: Clone mã nguồn từ GitHub

```
git clone https://github.com/bradtraversy/proshop-v2.git
cd proshop-v2
```

Bước 2: Cài đặt thư viện phụ thuộc (Dependencies) Cài đặt cho cả Backend và Frontend:

```
# Cài đạt cho Backend
cd backend
npm install

# Cài đạt cho Frontend
cd frontend
npm install
```

Bước 3: Khởi tạo dữ liệu mẫu (Data Seeding) Hệ thống cung cấp script để import dữ liệu sản phẩm và user mẫu:

```
# Tại thu mục góc
npm run data:import

# Output mong đọi:
# Data Imported!
```

Bước 4: Khởi chạy dự án Sử dụng `concurrently` để chạy cả Server và Client cùng lúc:

```
npm run dev

# Server running in development mode on port 5000
# React App running on port 3000
```

6 Danh sách thư viện sử dụng (Dependencies)

Tên thư viện	Phiên bản	Mục đích sử dụng
express	4.18.2	Framework web cho Node.js
mongoose	7.0.3	ODM làm việc với MongoDB
jsonwebtoken	9.0.0	Tạo và xác thực JWT Token
bcryptjs	2.4.3	Mã hóa mật khẩu người dùng
react	18.2.0	Thư viện xây dựng giao diện
@reduxjs/toolkit	1.9.3	Quản lý state toàn cục (Global State)
react-bootstrap	2.7.2	UI Framework (CSS)
axios	1.3.4	Client HTTP Request
multer	1.4.5	Xử lý upload file ảnh

Bảng 9.1: Các thư viện mã nguồn mở chính

Tài liệu

[1] Facebook Open Source, *React Documentation - A JavaScript library for building user interfaces*, https://reactjs.org/docs/getting-started.html, 2023.

[2] Brad Traversy, *MERN eCommerce From Scratch - Course Documentation*, Udemy, 2024.

[3] Meta Open Source, *Jest - Delightful JavaScript Testing Framework*, https://jestjs.io/docs/getting-started, 2023.

[4] Glenford J. Myers, *The Art of Software Testing*, Third Edition, John Wiley & Sons, 2011.

[5] Automattic, *Mongoose Documentation - Elegant mongodb object modeling for node.js*, https://mongoosejs.com/docs/guide.html, 2023.

[6] OWASP Foundation, *OWASP Top 10 - 2021: The Ten Most Critical Web Application Security Risks*, https://owasp.org/www-project-top-ten/, 2021.