

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
KHOA CÔNG NGHỆ THÔNG TIN**



BÀI TẬP LỚN

MÔN: KIỂM THỦ PHẦN MỀM

Đề tài: KIỂM THỦ TRANG WEB BÁN GIÀY

Giảng viên hướng dẫn : Ths. Nguyễn Thu Hường

Sinh viên thực hiện:

Họ tên Trần Kim Anh

Mã sinh viên 221230742

Hà Nội, 2025

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
KHOA CÔNG NGHỆ THÔNG TIN



BÀI TẬP LỚN

MÔN: KIỂM THỦ PHẦN MỀM

Đề tài: KIỂM THỦ TRANG WEB BÁN GIÀY

Giảng viên hướng dẫn : Ths. Nguyễn Thu Hường

Sinh viên thực hiện:

Họ tên Trần Kim Anh

Mã sinh viên 221230742

Hà Nội, 2025

MỤC LỤC

| | |
|---|----|
| MỤC LỤC | 3 |
| DANH MỤC HÌNH ẢNH..... | 7 |
| DANH MỤC BẢNG BIỂU..... | 8 |
| LỜI MỞ ĐẦU | 9 |
| CHƯƠNG 1: TỔNG QUAN VỀ KIỂM THỬ..... | 11 |
| 1.1 Kiểm thử phần mềm là gì? | 11 |
| 1.2 Phân biệt kiểm định và thẩm định | 12 |
| 1.3 Quy trình kiểm thử phần mềm | 12 |
| 1.3.1 Test Planning (Lập kế hoạch kiểm thử) | 13 |
| 1.3.2 Test Analysis and Design (Phân tích và thiết kế kiểm thử) | 14 |
| 1.3.3 Test Implementation and Execution (Thực thi kiểm thử) | 15 |
| 1.3.4 Evaluating Exit Criteria and Reporting (Đánh giá tiêu chí thoát và báo cáo) | 17 |
| 1.3.5 Test Closure Activities (Hành động kết thúc kiểm thử)..... | 18 |
| 1.4 Testing Levels (Các Mức kiểm thử)..... | 19 |
| 1.5 Các phương pháp kiểm thử phần mềm..... | 21 |
| 1.5.1 Kiểm thử hộp đen | 21 |
| 1.5.2. Kiểm thử hộp trắng..... | 22 |
| 1.5.3 Kiểm thử hộp xám | 24 |
| CHƯƠNG 2: KIỂM THỬ API VỚI POSTMAN | 25 |
| 2.1 Tổng quan về API..... | 25 |
| 2.1.1 API là gì? | 25 |
| 2.1.2 Mục đích của kiểm thử API | 25 |
| 2.1.3 Mô hình hoạt động của API | 26 |
| 2.1.4 Các loại API phổ biến | 27 |

| | |
|--|----|
| 2.1.5 Bảo mật của API..... | 28 |
| 2.1.6 Endpoint và tầm quan trọng của Endpoint | 29 |
| 2.1.7 Ưu điểm và nhược điểm của API | 29 |
| 2.1.8 Ứng dụng của API | 30 |
| 2.2 RESTful API | 31 |
| 2.2.1 RESTful API là gì?..... | 31 |
| 2.2.2 Cách thức hoạt động của RESTful API..... | 32 |
| 2.2.3 Request và response trong RESTful API..... | 32 |
| 2.2.4 Status code trong RESTful API | 35 |
| 2.3. Tổng quan về kiểm thử API | 40 |
| 2.3.1. Kiểm thử API là gì?..... | 40 |
| 2.3.2. Tại sao cần kiểm thử API | 40 |
| 2.3.3. Phương pháp tiếp cận API testing | 41 |
| 2.3.4. Một số phương pháp kiểm thử API phổ biến | 41 |
| 2.3.5. Sự khác nhau giữa API testing và Unit testing | 42 |
| 2.4. Công cụ kiểm thử API – Postman | 43 |
| 2.4.1. Postman là gì? | 43 |
| 2.4.2. Các thành phần chính trong Postman..... | 44 |
| 2.4.3. Các chức năng chính trong Postman | 44 |
| 2.4.4. Các thành phần khác trong Postman | 45 |
| CHƯƠNG 3: KIỂM THỬ TỰ ĐỘNG VỚI SELENIUM | 47 |
| 3.1. Khái quát về Selenium | 47 |
| 3.1.1. Giới thiệu chung | 47 |
| 3.1.2. Lịch sử hình thành và phát triển | 47 |
| 3.1.3. Đặc điểm của Selenium | 48 |
| 3.1.4. Các thành phần của Selenium | 48 |

| | |
|---|-----------|
| 3.2. Selenium Webdriver | 50 |
| 3.2.1 Đặc điểm của Selenium Webdriver..... | 50 |
| 3.2.2 Các tính năng nổi bật của Selenium WebDriver | 50 |
| 3.2.3 Ưu điểm và hạn chế của Selenium WebDriver | 51 |
| CHƯƠNG 4. GIỚI THIỆU TRANG WEB BÁN GIÀY | 53 |
| CHƯƠNG 5: LẬP KẾ HOẠCH KIỂM THỬ..... | 55 |
| 5.1 Tổng quan | 55 |
| 5.1.1 Mục tiêu kiểm thử | 55 |
| 5.1.2 Phạm vi..... | 55 |
| 5.1.3 Các loại kiểm thử cần thực hiện | 55 |
| 5.1.4 Chiến lược kiểm thử..... | 55 |
| 5.1.5 Tiêu chí chấp nhận và từ chối | 56 |
| 5.1.6 Rủi ro và biện pháp giảm thiểu rủi ro..... | 57 |
| CHƯƠNG 6 PHÂN TÍCH VÀ THIẾT KẾ CA KIỂM THỬ | 59 |
| 6.1 Phân tích và thiết kế ca kiểm thử cho các chức năng màn hình..... | 59 |
| 6.1.1 Chức năng của màn hình SR_001 | 59 |
| 6.1.1.1 Chức năng Thêm sản phẩm vào giỏ hàng | 59 |
| 6.1.2 Chức năng màn hình SR_002..... | 62 |
| 6.1.2.1 Chức năng tăng/giảm số lượng sản phẩm | 62 |
| 6.1.2.2 Chuyển đến màn thanh toán | 64 |
| 6.1.3 Chức năng màn hình SR_003..... | 65 |
| 6.1.3.1 Chức năng xóa địa chỉ và đặt địa chỉ làm địa chỉ mặc định...65 | 65 |
| 6.1.3.2 Chức năng thêm địa chỉ..... | 65 |
| 6.1.4 Chức năng màn hình SR_004..... | 70 |
| 6.1.4.1 Chức năng lọc sản phẩm..... | 70 |
| 6.1.4.2 Chức năng tìm kiếm | 74 |

| | |
|---|-----------|
| 6.1.5 Chức năng màn hình SR_005..... | 75 |
| 6.1.5.1 Chức năng nhập thông tin giao hàng | 75 |
| 6.1.5.2 Chức năng thanh toán..... | 77 |
| CHƯƠNG 7: THỰC THI VÀ BÁO CÁO | 80 |
| 7.1 Thực thi kiểm thử | 80 |
| 7.1.1 Kiểm thử tự động..... | 80 |
| 7.1.1.1 Chức năng tìm kiếm sản phẩm | 80 |
| 7.1.1.2 Chức năng thêm địa chỉ đặt hàng..... | 80 |
| 7.1.2 Kiểm thử thủ công | 81 |
| KẾT LUẬN | 83 |

DANH MỤC HÌNH ẢNH

| | |
|---|----|
| Hình 2.1 Minh họa Hệ thống Rest Api..... | 32 |
| Hình 2.2 HTTP Response với mã trạng thái 200 OK | 33 |
| Hình 2.3 HTTP Response với mã trạng thái 404 Not Found | 34 |
| Hình 2.4 Phản hồi API thành công (Status 200 OK) trên Postman | 34 |
| Hình 2.5 Cấu trúc của Response trả về với Body dạng JSON | 35 |
| Hình 2.6 Cấu trúc của Response trả về với Body dạng HTML/XML | 35 |
| Hình 4.1 Sơ đồ phân dã chức năng hệ thống web bán giày | 54 |
| Hình 6.1 Mô tả Button [Add to cart]..... | 59 |
| Hình 6.2 Các ca kiểm thử cho chức năng thêm sản phẩm vào giỏ hàng..... | 60 |
| Hình 6.3 Testcase cho chức năng thêm sản phẩm vào giỏ hàng..... | 61 |
| Hình 6.4 Testcase các button tăng/ giảm số lượng sản phẩm | 64 |
| Hình 6.5 Testcase cho chức năng điều hướng đến màn hình thanh toán | 64 |
| Hình 6.6 Mô tả chi tiết chức năng xóa địa chỉ | 65 |
| Hình 6.7 Testcase cho chức năng xóa và đặt địa chỉ mặc định..... | 65 |
| Hình 6.8 Kĩ thuật phân vùng tương đương cho Chức năng thêm địa chỉ | 66 |
| Hình 6.9 Testcase cho Chức năng thêm địa chỉ | 70 |
| Hình 6.10. Mô tả chi tiết chức năng lọc sản phẩm theo các biến thể | 71 |
| Hình 6.11 Testcase chức năng lọc sản phẩm theo các biến thể | 73 |
| Hình 6.12 Testcase cho chức năng tìm kiếm..... | 74 |
| Hình 6.13. Mô tả chi tiết chức năng nhập thông tin giao hàng | 75 |
| Hình 6.14 Testcase nhập thông tin giao hàng | 77 |
| Hình 6.15 Bảng phân vùng tương đương cho chức năng thanh toán bằng thẻ visa | 78 |
| Hình 6.16 Testcase cho chức năng chức năng thanh toán bằng thẻ visa | 79 |
| Hình 7.1 Kết quả kiểm thử chức năng tìm kiếm sản phẩm | 80 |
| Hình 7.2 Kết quả kiểm thử chức năng thêm địa chỉ | 81 |

DANH MỤC BẢNG BIỂU

| | |
|--|----|
| Bảng 1.1 So sánh giữa kiểm định và thẩm định..... | 12 |
| Bảng 2.1 Một số phương pháp kiểm thử API phổ biến..... | 41 |
| Bảng 2.2 So sánh API testing và Unit testing | 42 |
| Bảng 5.1 Phân chia công việc | 56 |
| Bảng 5.2 Bảng lịch trình kiểm thử | 56 |
| Bảng 6.1 Bảng phân vùng kiểm thử cho chức năng giảm số lượng..... | 62 |
| Bảng 6.2 Bảng phân vùng kiểm thử cho chức năng tăng số lượng..... | 62 |
| Bảng 6.3 Bảng phân vùng kiểm thử cho chức năng điều hướng đến màn hình thanh toán | 64 |

LỜI MỞ ĐẦU

Trong bối cảnh công nghệ thông tin phát triển mạnh mẽ như hiện nay, thương mại điện tử đã trở thành xu hướng tất yếu của các doanh nghiệp bán lẻ. Các cửa hàng giàn dép truyền thống đang dần chuyển đổi sang mô hình kinh doanh trực tuyến để mở rộng thị trường, tối ưu hóa quy trình quản lý và nâng cao trải nghiệm khách hàng. Trong quá trình chuyển đổi số này, việc xây dựng một hệ thống quản lý bán hàng trực tuyến ổn định, đáng tin cậy là vô cùng quan trọng.

Để đảm bảo chất lượng của hệ thống, kiểm thử phần mềm đóng vai trò then chốt trong việc phát hiện và khắc phục các lỗi tiềm ẩn trước khi đưa sản phẩm vào vận hành thực tế. Một hệ thống quản lý shop giàn hoạt động hiệu quả không chỉ yêu cầu giao diện thân thiện, dễ sử dụng mà còn phải đảm bảo tính chính xác trong xử lý nghiệp vụ, khả năng phản hồi nhanh và bảo mật thông tin khách hàng.

Xuất phát từ những thực tiễn đó, em đã lựa chọn đề tài “**Kiểm thử trang web bán giàn**” với mong muốn có thể áp dụng những kiến thức về kiểm thử của bản thân để có thể đánh giá chất lượng của một hệ thống quản lý cửa hàng điển hình. Việc kiểm thử hệ thống ngoài kiểm thử tại các màn hình giao diện mà còn kiểm thử API xử lý phía backend nhằm có thể đảm bảo được hệ thống hoạt động ổn định cả ở phía giao diện và backend trong nhiều tình huống khác nhau.

Trong phạm vi dự án này, các nội dung chính của đề tài này bao gồm:

Tìm hiểu và phân tích hệ thống cần kiểm thử.

Thiết kế các ca kiểm thử cho các màn hình chính

Thực hiện kiểm thử tự động và thủ công, ghi nhận và đánh giá kết quả kiểm thử.

Các phương pháp được áp dụng để thực hiện dự án bao gồm:

Nghiên cứu hệ thống đưa ra các tài liệu đặc tả hệ thống.

Áp dụng các kỹ thuật kiểm thử hộp đen, đoán lỗi và đặc tả UI để đưa ra các ca kiểm thử.

Thiết kế các ca kiểm thử dựa trên từng màn hình với các thông tin về di chuyển màn hình, xác thực dữ liệu...trong tài liệu đặc tả và dựa trên các tài liệu API.

Sử dụng Selenium để thực hiện các ca kiểm thử tự động.

Thực hiện kiểm thử thủ công trên các nền tảng trình duyệt.

Ghi nhận và đánh giá kết quả kiểm thử

Dự án được thực hiện với cấu trúc gồm các chương sau:

CHƯƠNG 1. TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM

CHƯƠNG 2. KIỂM THỬ API VỚI POSTMAN

CHƯƠNG 3. KIỂM THỬ TỰ ĐỘNG VỚI SELENIUM

CHƯƠNG 4. GIỚI THIỆU VỀ TRANG WEB BÁN GIÀY

CHƯƠNG 5. LẬP KẾ HOẠCH KIỂM THỬ

CHƯƠNG 6. PHÂN TÍCH VÀ THIẾT KẾ CA KIỂM THỬ

CHƯƠNG 7. THỰC THI VÀ BÁO CÁO

KẾT LUẬN VÀ KIẾN NGHỊ

CHƯƠNG 1: TỔNG QUAN VỀ KIỂM THỬ

1.1 Kiểm thử phần mềm là gì?

Kiểm thử phần mềm (Software testing) là một quá trình thực hiện trong một chương trình nhằm mục đích tìm kiếm các lỗi có thể xảy ra của chương trình.

Mục đích của kiểm thử là tìm ra lỗi, không phải tìm ra nguyên nhân gây ra lỗi.

Tại sao phải kiểm thử phần mềm?

Phần mềm do con người tạo ra và con người luôn có khả năng mắc lỗi . Lỗi cần được phát hiện sớm nhất có thể, ngay tại bước đầu tiên của quy trình phát triển phần mềm

Đảm bảo đưa sản phẩm chất lượng đến khách hàng nhằm có được độ tin cậy và sự hài lòng của khách hàng

Giảm chi phí cho nhà phát triển trong việc bảo trì, bảo hành sản phẩm

Chi phí khắc phục lỗi thường nhiều hơn so với chi phí xây dựng phần mềm

Đặc điểm của kiểm thử phần mềm:

Process: Kiểm thử phần mềm là một tiến trình không phải là một hành động đơn lẻ

All life cycle activities: Kiểm thử phần mềm được thực hiện trong mọi hoạt động của quy trình phát triển phần mềm cũng như trong mỗi một bước nhỏ của từng hoạt động đó.

As a program: Kiểm thử phần mềm cũng như phát triển một phần mềm, nó cần đầy đủ các bước bao gồm lên kế hoạch, phân tích, thiết kế, cài đặt, chuẩn bị môi trường cũng như thực thi để đưa ra kết quả cuối cùng.

Static test vs dynamic test:

Static test: thực hiện mà không cần phải thực thi chương trình: reviewing, inspection, ...

Dynamic test: thực hiện cần đến việc thực thi chương trình hoặc một phần của chương trình: unit test, integration test, system test ...

1.2 Phân biệt kiểm định và thẩm định

Kiểm định là quá trình đánh giá sản phẩm phần mềm trong các giai đoạn phát triển để đảm bảo phần mềm được xây dựng đúng cách theo đúng các chuẩn, quy trình và tài liệu thiết kế

Thẩm định là quá trình đánh giá sản phẩm cuối cùng của phần mềm để đảm bảo phần mềm thỏa mãn, đáp ứng yêu cầu của người dùng

| | Kiểm định | Thẩm định |
|------------|--|--|
| Giống nhau | <p>Giúp đảm bảo chất lượng phần mềm</p> <p>Giúp phát hiện và loại bỏ các lỗi và sai sót trong phần mềm.</p> <p>Đều là các hoạt động quan trọng trong vòng đời phát triển phần mềm</p> <p>Yêu cầu tài liệu đầy đủ và chính xác</p> | |
| Khác nhau | <p>Đảm bảo đáp ứng các yêu cầu cụ thể của từng giai đoạn không.</p> <p>Kiểm tra xem sản phẩm có được xây dựng đúng theo yêu cầu và đặc điểm kỹ thuật thiết kế không.</p> <p>Kiểm tra xem "Chúng tôi xây dựng sản phẩm đúng không?"</p> <p>Được thực hiện mà không cần chạy phần mềm.</p> <p>Bao gồm tất cả các kỹ thuật test tĩnh.</p> | <p>Đảm bảo đáp ứng được yêu cầu nghiệp vụ không.</p> <p>Xác định xem phần mềm có phù hợp với nhu cầu sử dụng và đáp ứng yêu cầu nghiệp vụ không.</p> <p>Kiểm tra "Chúng tôi xây dựng đúng sản phẩm"?</p> <p>Được thực hiện cùng với việc chạy phần mềm.</p> <p>Bao gồm tất cả các kỹ thuật test động</p> |

Bảng 1.1 So sánh giữa kiểm định và thẩm định.

1.3 Quy trình kiểm thử phần mềm

Quy trình kiểm thử phần mềm (Software Testing Process) là tập hợp các bước có hệ thống, nhằm mục đích đảm bảo chất lượng và phát hiện lỗi trong phần mềm trước khi

đưa vào sử dụng chính thức. Kiểm thử là một phần quan trọng trong vòng đời phát triển phần mềm (SDLC - Software Development Life Cycle).

Đối với một dự án kiểm thử phần mềm trong thực tiễn, quy trình này thường trải qua các bước chính sau:

Lập kế hoạch (Test Planning)

Phân tích và thiết kế Test case (Test Analysis and Design)

Thực thi kiểm thử (Test Implementation and Execution)

Đánh giá tiêu chí thoát và báo cáo (Evaluating Exit Criteria and Reporting)

Hành động kết thúc kiểm thử (Test Closure Activities)

1.3.1 Test Planning (Lập kế hoạch kiểm thử)

Đây là giai đoạn xác định các chiến lược được dùng để kiểm thử nhằm đảm bảo rằng sản phẩm thỏa mãn đặc tả thiết kế phần mềm và các yêu cầu khác. Các hoạt động chính trong giai đoạn này bao gồm:

Xác định các chiến lược: Định nghĩa mục tiêu và phạm vi của việc kiểm thử.

Phân tích sản phẩm: Xác định đối tượng sử dụng, mục đích sử dụng, cách thức hoạt động, và phần cứng/phần mềm sản phẩm sử dụng.

Xây dựng chiến lược kiểm thử:

Định nghĩa phạm vi kiểm thử (những thành phần nào cần và không cần kiểm thử).

Xác định loại kiểm thử.

Tạo và lưu trữ tài liệu về Rủi ro & Vấn đề (Risk & Issues).

Xác định người kiểm thử và thời điểm sẵn sàng kiểm thử.

Xác định mục tiêu kiểm thử: Liệt kê tất cả các tính năng cần kiểm thử (functionality, performance, GUI...).

Xác định tiêu chí kiểm thử: Bao gồm tiêu chí thực hiện, định chỉ, và kết thúc kiểm thử.

Hoạch định nguồn lực: Con người và hệ thống.

Kế hoạch môi trường kiểm thử.

Xây dựng lịch trình.

Xác định deliverable (danh sách các tài liệu, tool và các thành phần khác hỗ trợ nỗ lực kiểm thử).

1.3.2 Test Analysis and Design (Phân tích và thiết kế kiểm thử)

Giai đoạn Phân tích và Thiết kế kiểm thử nhằm xác định những gì cần được kiểm thử và cách thức thực hiện kiểm thử để đảm bảo rằng tất cả yêu cầu phần mềm đều được bao phủ. Đây là bước chuyển tiếp từ kế hoạch kiểm thử sang việc xây dựng các trường hợp và kịch bản kiểm thử cụ thể.

Mục tiêu

Xác định các điều kiện kiểm thử (Test Conditions) dựa trên tài liệu yêu cầu và thiết kế.

Xây dựng các Test Case bao phủ đầy đủ các luồng xử lý và tình huống ngoại lệ của hệ thống.

Chuẩn bị dữ liệu kiểm thử (Test Data) phù hợp cho từng Test Case.

Thiết lập môi trường, hạ tầng và công cụ kiểm thử đảm bảo đúng yêu cầu kỹ thuật.

Nội dung thực hiện:

Phân tích yêu cầu kiểm thử:

Dựa vào tài liệu yêu cầu phần mềm (ADMS) và tài liệu thiết kế, kiểm thử phân tích để xác định các điều kiện cần kiểm tra cho từng chức năng hoặc module của hệ thống.

Thiết kế Test Case và kịch bản kiểm thử:

Xây dựng danh sách Test Case cho từng điều kiện kiểm thử.

Đảm bảo Test Case bao phủ cả luồng chính, luồng phụ và các trường hợp ngoại lệ.

Nếu cần kiểm thử tự động, Test Designer sẽ thiết kế các kịch bản tự động (Test Script) dựa trên các Test Case đã xây dựng.

Xây dựng dữ liệu kiểm thử:

Chuẩn bị dữ liệu đầu vào cho từng Test Case.

Gồm cả dữ liệu hợp lệ (valid data) và dữ liệu không hợp lệ (invalid data).

Đảm bảo dữ liệu phản ánh đúng tình huống thực tế và có khả năng phát hiện lỗi tiềm ẩn.

Chuẩn bị môi trường kiểm thử:

Thiết lập môi trường phần mềm, phần cứng, cơ sở dữ liệu và công cụ hỗ trợ kiểm thử.

Đảm bảo môi trường mô phỏng gần nhất với môi trường sản xuất thực tế để đảm bảo độ chính xác của kết quả kiểm thử.

Đầu vào:

Tài liệu Kế hoạch kiểm thử (Test Planning)

Tài liệu Đặc tả yêu cầu phần mềm (ADMS)

Tài liệu Thiết kế hệ thống (Design Specification)

Đầu ra:

Tài liệu Test Case / Test Scenario

Dữ liệu kiểm thử (Test Data)

Tài liệu cấu hình môi trường kiểm thử (Test Environment Setup Document)

1.3.3 Test Implementation and Execution (Thực thi kiểm thử)

Giai đoạn Thực thi kiểm thử (Test Implementation and Execution) là quá trình thực hiện các Test Case đã được thiết kế nhằm xác định xem phần mềm có hoạt động đúng theo yêu cầu hay không. Ở giai đoạn này, kiểm thử tiến hành chạy các ca kiểm thử, ghi nhận kết quả và báo cáo các lỗi phát hiện được.

Mục tiêu

Thực hiện các Test Case theo đúng kế hoạch đã đề ra.

Ghi nhận, so sánh kết quả thực tế với kết quả mong đợi.

Phát hiện, ghi nhận và theo dõi các lỗi (defect) trong quá trình kiểm thử.

Kiểm thử lại (re-test) và kiểm thử hồi quy (regression test) sau khi lỗi được sửa.

Nội dung thực hiện

Thực hiện kiểm thử:

Chuẩn bị môi trường kiểm thử và dữ liệu đầu vào.

Chạy các Test Case theo trình tự và điều kiện được mô tả trong tài liệu kiểm thử.

Đảm bảo việc thực thi tuân theo kế hoạch kiểm thử và quy trình kiểm thử đã phê duyệt.

Ghi nhận kết quả kiểm thử:

So sánh kết quả thực tế với kết quả mong đợi được nêu trong Test Case.

Ghi lại trạng thái của từng Test Case (Pass/Fail/Blocked/Not Run).

Lưu lại bằng chứng kiểm thử (ảnh chụp màn hình, log, dữ liệu đầu ra...).

Ghi lại lỗi (Defect Logging):

Khi kết quả kiểm thử khác với kết quả mong đợi, kiểm thử tiến hành tạo báo cáo lỗi (Defect Report).

Ghi rõ mô tả lỗi, bước tái hiện, mức độ ảnh hưởng (severity), và độ ưu tiên (priority).

Gửi lỗi đến phát triển để xử lý và theo dõi trạng thái lỗi đến khi được khắc phục.

Kiểm thử lại (Re-test) và kiểm thử hồi quy (Regression Test):

Sau khi lập trình viên sửa lỗi, kiểm thử viên thực hiện re-test để xác nhận lỗi đã được khắc phục.

Tiến hành kiểm thử hồi quy nhằm đảm bảo việc sửa lỗi không ảnh hưởng đến các chức năng khác của hệ thống.

Đầu vào

Tài liệu Test Case và Test Data

Môi trường kiểm thử (Test Environment)

Phiên bản phần mềm cần kiểm thử

Đầu ra

Kết quả thực thi kiểm thử (Test Execution Report)

Báo cáo lỗi (Defect Report)

Báo cáo kiểm thử lại (Re-test Report)

1.3.4 Evaluating Exit Criteria and Reporting (Đánh giá tiêu chí thoát và báo cáo)

Giai đoạn Đánh giá tiêu chí thoát và báo cáo nhằm xác định xem quá trình kiểm thử có thể kết thúc hay cần tiếp tục thực hiện thêm các hoạt động kiểm thử. Việc đánh giá dựa trên các tiêu chí thoát (exit criteria) đã được xác định trong kế hoạch kiểm thử, giúp đảm bảo rằng phần mềm đạt mức chất lượng mong đợi trước khi bàn giao.

Mục tiêu:

Đánh giá kết quả kiểm thử so với các mục tiêu đã đặt ra trong kế hoạch kiểm thử.

Xác định xem có cần thực hiện thêm kiểm thử hay không.

Tổng hợp và báo cáo toàn bộ kết quả kiểm thử cho các bên liên quan.

Nội dung thực hiện:

Xác minh có cần kiểm thử thêm hay không, so sánh kết quả thực tế với các tiêu chí thoát như:

Tỷ lệ Test Case đã được thực thi.

Tỷ lệ lỗi đã được phát hiện và khắc phục.

Mức độ bao phủ yêu cầu (requirement coverage).

Mức độ rủi ro còn tồn tại.

Nếu các tiêu chí chưa được đáp ứng, kiểm thử sẽ đề xuất thực hiện kiểm thử bổ sung hoặc mở rộng phạm vi kiểm thử.

Báo cáo tổng kết kiểm thử (Test Summary Report):

Tổng hợp kết quả thực hiện kiểm thử, bao gồm:

Số lượng Test Case đã chạy, pass, fail, chưa chạy.

Thống kê lỗi: số lỗi phát hiện, đã sửa, còn tồn tại.

Đánh giá chất lượng phần mềm và mức độ sẵn sàng để triển khai.

Gửi báo cáo tổng kết kiểm thử đến các bên liên quan (quản lý dự án, đội phát triển, khách hàng, v.v.).

Đầu vào:

Kế hoạch kiểm thử (Test Plan)

Kết quả thực thi kiểm thử (Test Execution Results)

Báo cáo lỗi (Defect Report)

Đầu ra:

Báo cáo tổng kết kiểm thử (Test Summary Report)

Đề xuất về việc dừng hoặc tiếp tục kiểm thử

1.3.5 Test Closure Activities (Hành động kết thúc kiểm thử)

Giai đoạn Hành động kết thúc kiểm thử được thực hiện sau khi quá trình kiểm thử hoàn tất và các tiêu chí thoát đã được đáp ứng. Mục tiêu của giai đoạn này là tổng kết, đánh giá toàn bộ hoạt động kiểm thử, lưu trữ tài liệu, và rút ra bài học kinh nghiệm cho các dự án sau.

Mục tiêu:

Đảm bảo tất cả hoạt động kiểm thử đã được hoàn thành theo kế hoạch.

Tổng hợp, lưu trữ và bàn giao các tài liệu, kết quả kiểm thử.

Đánh giá hiệu quả của quy trình kiểm thử và đề xuất cải tiến cho lần sau.

Nội dung thực hiện:

Lưu trữ và đóng gói kết quả kiểm thử:

Thu thập, lưu trữ toàn bộ tài liệu kiểm thử bao gồm: kế hoạch kiểm thử, Test Case, Test Data, báo cáo lỗi, báo cáo kết quả kiểm thử và báo cáo tổng kết.

Đảm bảo tất cả tài liệu được lưu trữ có hệ thống, dễ tra cứu và sử dụng lại cho các dự án tương tự trong tương lai.

Bàn giao tài liệu kiểm thử cho các bên liên quan theo quy định của dự án.

Dánh giá và rút kinh nghiệm:

Tổ chức cuộc họp tổng kết để đánh giá quá trình kiểm thử, những điểm đạt được và hạn chế.

Ghi nhận các bài học kinh nghiệm, đề xuất cải tiến quy trình kiểm thử và phương pháp làm việc cho các dự án kế tiếp.

Đầu vào:

Báo cáo tổng kết kiểm thử (*Test Summary Report*)

Tài liệu kiểm thử (*Test Artifacts*)

Đầu ra:

Bộ tài liệu kiểm thử đã lưu trữ hoàn chỉnh (*Test Repository*)

Báo cáo tổng kết kết thúc kiểm thử (*Test Closure Report*)

Bài học kinh nghiệm và đề xuất cải tiến (*Lessons Learned*)

1.4 Testing Levels (Các Mức kiểm thử)

Có 4 cấp độ kiểm thử phần mềm là:

Kiểm thử đơn vị (Unit Testing)

Kiểm thử tích hợp (Integration Testing)

Kiểm thử hệ thống (System Testing)

Kiểm thử chấp nhận (Acceptance Testing)

1. Unit Testing (Kiểm thử đơn vị):

Kiểm thử đơn vị là cấp độ kiểm thử cơ bản, thực hiện test từng module nhỏ trong hệ thống và có thể được thực hiện tách biệt với phần còn lại của hệ thống tùy thuộc vào mô hình vòng đời phát triển được chọn.

Mục đích: Xác nhận mỗi thành phần của phần mềm thực hiện đúng với thiết kế ban đầu.

Người thực hiện: Thường do lập trình viên thực hiện.

Phương pháp: Thường sử dụng cả phương pháp kiểm thử hộp đen (Black-box testing) và kiểm thử hộp trắng (White-box testing).

2. Integration Testing (Kiểm thử tích hợp):

Kiểm thử tích hợp (hay kiểm thử kết hợp) là việc kiểm thử cho nhiều chức năng (hay module) cùng một lúc. Nó sẽ kiểm tra việc truyền dữ liệu giữa các chức năng, các màn hình với nhau để xem chúng có liên lạc với nhau đúng như yêu cầu hay không.

Mục đích:

Đảm bảo rằng các chức năng (module) khi tích hợp lại với nhau thì chạy đúng theo yêu cầu ban đầu.

Là bước chuẩn bị quan trọng để tiến hành kiểm thử ở mức hệ thống (System Testing).

Thời điểm: Được thực hiện sau khi kiểm thử đơn vị và trước khi kiểm thử hệ thống.

Người thực hiện: Được thực hiện bởi một tester hoặc một các tester. Lập trình viên không tham gia vào giai đoạn này.

Phương pháp: Kiểm thử hộp đen (Black-box testing).

3. System Testing (Kiểm thử hệ thống):

Kiểm thử hệ thống là thực hiện kiểm thử một hệ thống đã được tích hợp hoàn chỉnh để xác minh rằng nó đúng yêu cầu của phần mềm. Đây thường là giai đoạn thử nghiệm cuối cùng để xác minh hệ thống đáp ứng đầy đủ các đặc điểm kỹ thuật và yêu cầu của người dùng trước khi giao cho khách hàng.

Phương pháp: Kiểm thử hộp đen (Black-box testing), không yêu cầu kiến thức về thiết kế bên trong của mã nguồn.

Điểm khác biệt với kiểm thử tích hợp:

Kiểm thử tích hợp thường chỉ thực hiện cho các yêu cầu chức năng.

Kiểm thử hệ thống thực hiện cho cả yêu cầu chức năng và phi chức năng.

Người thực hiện: Thường được thực hiện bởi tester. Đôi khi hệ thống quá phức tạp sẽ cần làm chung với đội dự án của khách hàng.

4. Acceptance Testing (Kiểm thử chấp nhận):

Sau giai đoạn kiểm thử hệ thống, phần mềm sẽ được gửi đến người dùng hoặc khách hàng để thực hiện kiểm thử chấp nhận (hay kiểm thử nghiệm thu), còn được gọi là UAT (User Acceptance Testing).

Mục đích: Đảm bảo phần mềm đáp ứng đúng yêu cầu của khách hàng và nhận được sự chấp nhận từ khách hàng hoặc người dùng cuối (end-user).

Điểm khác biệt với kiểm thử hệ thống: Kiểm thử chấp nhận về cơ bản khá giống kiểm thử hệ thống nhưng được thực hiện bởi khách hàng và trên môi trường thật. Môi trường kiểm thử hệ thống thường chỉ là môi trường mô phỏng (giống thật).

Các mức kiểm thử chấp nhận:

Kiểm thử alpha: Được thực hiện tại địa điểm của khách hàng (khi khách hàng chưa phải là người dùng cuối cùng). Ví dụ: nhân viên ngân hàng kiểm thử ứng dụng internet banking do một công ty phần mềm phát triển.

Kiểm thử beta: Được thực hiện bởi khách hàng/người dùng cuối tại địa điểm của chính họ. Ví dụ: Khách hàng của ngân hàng (người dùng thật sự) sử dụng và kiểm thử ứng dụng internet banking.

1.5 Các phương pháp kiểm thử phần mềm

1.5.1 Kiểm thử hộp đen

Kiểm thử hộp đen – black-box testing, hay còn gọi là kiểm thử dựa vào thông tin về các đặc tả của yêu cầu mà không cần quan tâm đến cấu trúc dữ liệu thiết kế bên trong, thuật toán cài đặt, quá trình xử lý dữ liệu. Kiểm thử hộp đen không yêu cầu người kiểm thử phải có kiến thức về lập trình

Mục tiêu chính của kiểm thử hộp đen là xác định xem phần mềm có hoạt động đúng theo các yêu cầu chức năng đã được đề ra hay không. Phương pháp kiểm thử này bao gồm cả kiểm thử chức năng (functional testing) và kiểm thử phi chức năng (non-functional testing), như kiểm thử hiệu năng, khả năng mở rộng, khả năng tương thích hoặc tính bảo mật của hệ thống.

Kiểm thử hộp đen có thể được áp dụng ở nhiều cấp độ kiểm thử khác nhau, đặc biệt là kiểm thử tích hợp (Integration Test), kiểm thử hệ thống (System Test) và kiểm thử chấp

nhận (Acceptance Test) - nơi mục tiêu là đảm bảo phần mềm đáp ứng đúng yêu cầu và mong đợi của người dùng cuối.

Quy trình kiểm thử hộp đen thường bao gồm bốn bước cơ bản sau:

Bước 1: Phân tích các đặc tả yêu cầu của phần mềm để xác định các chức năng hoặc thành phần cần được kiểm thử.

Bước 2: Áp dụng các kỹ thuật thiết kế ca kiểm thử (test case) nhằm xây dựng các tình huống kiểm thử cụ thể. Mỗi ca kiểm thử thường bao gồm:

Giá trị đầu vào (input data), tiền điều kiện, mô tả các bước thực thi,...

Trạng thái hoặc phản hồi của phần mềm (UI, thông báo, phản hồi hệ thống, v.v.)

Giá trị đầu ra mong đợi (expected output)

Bước 3: Thực hiện kiểm thử theo các ca kiểm thử đã thiết kế và ghi nhận kết quả thực tế.

Bước 4: So sánh kết quả thực tế với kết quả mong đợi để đánh giá mức độ đáp ứng của phần mềm đối với yêu cầu đề ra.

1.5.2. Kiểm thử hộp trắng

Kiểm thử hộp trắng (White-box Testing), hay còn gọi là kiểm thử dựa vào cấu trúc, là phương pháp kiểm thử dựa trên việc phân tích cấu trúc dữ liệu và các thuật toán được cài đặt bên trong của chương trình. Người kiểm thử trong trường hợp này cần có kiến thức về ngôn ngữ lập trình, cấu trúc mã nguồn và luồng xử lý của hệ thống.

Mục tiêu của kiểm thử hộp trắng là xác định hoạt động của phần mềm thông qua việc kiểm tra chi tiết mã lệnh, cấu trúc điều khiển và logic bên trong, nhằm đảm bảo rằng mọi nhánh, vòng lặp, và điều kiện đều được thực thi và kiểm tra đúng như mong đợi.

Kiểm thử hộp trắng thường được áp dụng chủ yếu ở các mức kiểm thử đơn vị (Unit Test) và kiểm thử hồi quy (Regression Test), vì đây là hai cấp độ đòi hỏi kiểm tra sâu bên trong mã nguồn. Tuy nhiên, phương pháp này cũng có thể được áp dụng cho kiểm thử tích hợp (Integration Test) và kiểm thử hệ thống (System Test) trong một số trường hợp cần xác minh luồng xử lý giữa các module có hoạt động chính xác hay không.

Ưu điểm và nhược điểm của kiểm thử hộp trắng

Ưu điểm:

Có thể bắt đầu từ giai đoạn sớm trong quá trình phát triển phần mềm, ngay cả khi giao diện người dùng chưa được xây dựng.

Cho phép kiểm thử kỹ lưỡng hơn, có thể bao phủ hầu hết các đường dẫn và cấu trúc điều khiển trong mã lệnh.

Giúp phát hiện sớm lỗi logic, lỗi cú pháp và các vấn đề trong mã nguồn, đặc biệt hiệu quả trong việc tìm lỗi ẩn.

Lập trình viên có thể tự thực hiện kiểm thử, hỗ trợ quá trình phát triển và tối ưu hóa mã.

Do hiểu rõ cấu trúc bên trong phần mềm, người kiểm thử có thể kiểm soát và loại bỏ lỗi tối đa, góp phần nâng cao chất lượng mã nguồn.

Nhược điểm:

Yêu cầu người kiểm thử có kiến thức chuyên sâu về lập trình.

Tốn nhiều thời gian khi kiểm tra các hệ thống lớn, phức tạp.

Khó áp dụng khi sản phẩm chưa hoàn thiện hoặc thiếu công cụ hỗ trợ.

Quá trình kiểm thử phức tạp và chi phí thực hiện cao.

Nếu kiểm thử không đầy đủ, dễ dẫn đến lỗi trong giai đoạn vận hành.

Có hai cách tiếp cận phổ biến thường được dùng trong kiểm thử hộp trắng là:

Kiểm thử dựa trên luồng điều khiển (Control Flow Testing): Kiểm tra các đường đi (path) của chương trình nhằm đảm bảo mọi câu lệnh, nhánh rẽ và vòng lặp đều được thực thi ít nhất một lần.

Kiểm thử dựa trên luồng dữ liệu (Data Flow Testing): Tập trung vào việc theo dõi cách dữ liệu được khai báo, sử dụng và truyền giữa các phần của chương trình để phát hiện lỗi xử lý hoặc sử dụng biến sai.

Kỹ thuật kiểm thử hộp trắng: thường được thực hiện theo hai kỹ thuật chính:

Kiểm thử tĩnh (Static Testing): Xem xét và phân tích mã nguồn mà không cần chạy chương trình, nhằm phát hiện lỗi cú pháp, lỗi logic hoặc sai phạm tiêu chuẩn lập trình.

Kiểm thử động (Dynamic Testing): Thực thi chương trình để kiểm tra luồng xử lý, độ bao phủ mã và hành vi của phần mềm. Các kỹ thuật phổ biến gồm kiểm thử đơn vị (Unit Test), kiểm thử bao phủ mã (Code Coverage) và kiểm thử luồng dữ liệu (Data Flow).

1.5.3 Kiểm thử hộp xám

Kiểm thử hộp xám là phương pháp kiểm thử kết hợp giữa kiểm thử hộp đen và kiểm thử hộp trắng. Phương pháp này vừa xem xét dữ liệu đầu vào và đầu ra của hệ thống như kiểm thử hộp đen, vừa khai thác hiểu biết nhất định về cấu trúc bên trong và thuật toán xử lý của phần mềm như kiểm thử hộp trắng.

Người kiểm thử hộp xám thường có quyền truy cập một phần vào mã nguồn, tài liệu thiết kế hoặc sơ đồ kiến trúc để xây dựng các ca kiểm thử hiệu quả hơn

Ưu điểm:

Kết hợp được các lợi ích của cả kiểm thử hộp đen và kiểm thử hộp trắng.

Có khả năng xác định lỗi tốt hơn nhờ hiểu biết về cấu trúc bên trong hệ thống.

Cho phép kiểm thử dựa trên yêu cầu chức năng, mô tả thiết kế và sơ đồ kiến trúc ngay từ giai đoạn đầu.

Nhược điểm:

Việc kiểm tra từng đường dẫn có thể tốn nhiều thời gian và công sức.

Đối với các hệ thống phức tạp hoặc phân tán, việc xác định và liên kết nguyên nhân lỗi gặp nhiều khó khăn.

Không phù hợp với tất cả các loại chức năng, đặc biệt là các chức năng đòi hỏi kiểm thử sâu về cấu trúc hoặc giao diện người dùng.

CHƯƠNG 2: KIỂM THỬ API VỚI POSTMAN

2.1 Tổng quan về API

2.1.1 API là gì?

API là viết tắt của Application Programming Interface (Giao diện lập trình ứng dụng), nó không phải là một ngôn ngữ lập trình, về cơ bản nó cho phép một chức năng của phần mềm này có thể giao tiếp với một hoặc nhiều phần mềm khác. Hay nói cách khác API là giao thức kết nối giữa ứng dụng và thư viện lưu trữ, tạo liên kết giữa hai ứng dụng thông qua các lệnh đơn giản để trao đổi dữ liệu.

API có tính ứng dụng thực tế cao, nó tạo ra những sự tương tác dễ dàng giữa các hệ thống với nhau. Hiện nay có rất nhiều loại API khác nhau: Windows API, Google API, Youtube API,...

2.1.2 Mục đích của kiểm thử API

Kiểm thử API (API Testing) là một quy trình quan trọng trong hoạt động đảm bảo Chất lượng Phần mềm. Quy trình này tập trung vào việc xác minh API ở tầng logic nghiệp vụ, độc lập với giao diện người dùng.

Mục tiêu chính của kiểm thử API bao gồm:

Xác minh tính đúng đắn của logic nghiệp vụ: Đảm bảo API trả về đúng dữ liệu, đúng định dạng (ví dụ: JSON, XML) và thực hiện chính xác các thao tác nghiệp vụ (tạo, đọc, cập nhật, xóa dữ liệu) tương ứng với các yêu cầu hợp lệ.

Đánh giá khả năng xử lý lỗi: Kiểm tra cách API phản hồi với các yêu cầu không hợp lệ, thiếu tham số, hoặc dữ liệu đầu vào sai định dạng. Hệ thống phải trả về các mã lỗi (Error Codes) và thông điệp lỗi chính xác thay vì gây ra các lỗi không mong muốn hoặc làm sập hệ thống.

Đánh giá hiệu năng và khả năng chịu tải: Xác định thời gian phản hồi (response time) của API dưới điều kiện tải bình thường và tải cao. Mục tiêu là đảm bảo API duy trì hiệu suất ổn định và không suy giảm dịch vụ khi có nhiều yêu cầu đồng thời.

Kiểm soát và xác minh an ninh, bảo mật: Đảm bảo API được bảo vệ khỏi các lỗ hổng bảo mật. Việc này bao gồm kiểm tra cơ chế xác thực, phân quyền và xác thực dữ liệu đầu vào để ngăn chặn truy cập trái phép hoặc các cuộc tấn công (ví dụ: SQL Injection, Broken Access Control).

Đảm bảo độ tin cậy và tính nhất quán: Đảm bảo API hoạt động ổn định và trả về kết quả nhất quán sau nhiều lần gọi.

Việc kiểm thử API cho phép phát hiện các khiếm khuyết ở tầng logic sớm hơn trong chu kỳ phát triển, giúp giảm thiểu chi phí và thời gian khắc phục sự cố.

2.1.3 Mô hình hoạt động của API

Quy trình hoạt động của API, đặc biệt là Web API, tuân theo mô hình Yêu cầu - Phản hồi (Request - Response) giữa Client (bên gọi) và Server (bên cung cấp). Luồng hoạt động tiêu chuẩn diễn ra như sau:

Client khởi tạo Yêu cầu: Ứng dụng Client (ví dụ: ứng dụng di động, trình duyệt web) tạo một yêu cầu (request) để truy cập một tài nguyên hoặc thực thi một chức năng. Yêu cầu này được gửi đến một địa chỉ cụ thể do Server cung cấp, gọi là *Endpoint*.

Cấu thành của Yêu cầu: Một yêu cầu API (thường là HTTP request) bao gồm các thành phần chính:

Phương thức HTTP (HTTP Method): Xác định hành động mong muốn:

GET: Truy xuất tài nguyên.

POST: Gửi dữ liệu để tạo tài nguyên mới.

PUT/PATCH: Cập nhật tài nguyên hiện có.

DELETE: Xóa tài nguyên.

Tiêu đề (Headers): Chứa các thông tin meta về yêu cầu, như định dạng dữ liệu và quan trọng nhất là thông tin xác thực.

Nội dung (Body): Chứa dữ liệu mà Client muốn gửi lên Server, thường ở định dạng JSON.

Server xử lý Yêu cầu: Server nhận yêu cầu, trước tiên sẽ kiểm tra tính hợp lệ của nó. Nếu hợp lệ, Server sẽ xử lý logic nghiệp vụ (ví dụ: truy vấn cơ sở dữ liệu, tính toán dữ liệu).

Server gửi Phản hồi: Sau khi xử lý, Server tạo một phản hồi (response) và gửi lại cho Client. Phản hồi bao gồm:

Mã trạng thái: Một mã số HTTP chỉ báo kết quả của yêu cầu.

Nội dung: Dữ liệu mà Client yêu cầu (thường là JSON) hoặc thông tin chi tiết về lỗi.

Client xử lý Phản hồi: Client nhận phản hồi, phân tích cú pháp dữ liệu và sử dụng thông tin đó để hiển thị trên giao diện người dùng hoặc thực hiện các tác vụ tiếp theo.

2.1.4 Các loại API phổ biến

Dựa vào phạm vi sử dụng, kiến trúc xây dựng và phương thức kết nối mà API được chia thành nhiều loại khác nhau.

2.1.4.1. Phân loại dựa vào phạm vi sử dụng

Dựa vào phạm vi hoạt động chúng ta phân ra làm 4 loại API:

API mở/ API công cộng: được hiểu là API dành cho mọi đối tượng, có sẵn và có thể được sử dụng bởi bất kỳ nhà phát triển nào. Đối với loại API này, thông thường sẽ yêu cầu các biện pháp xác thực hoặc ủy quyền thấp và bị hạn chế khi chia sẻ công khai. Một số API mở sẽ được chia sẻ miễn phí, còn một số khác sẽ yêu cầu tính phí khi sử dụng và những chi phí này thường sẽ được tính dựa trên các lệnh gọi đến API được sử dụng.

API đối tác: đóng vai trò như một bên thứ ba, chỉ phục vụ cho các công ty bên ngoài đã nhận được yêu cầu để thực hiện công việc, cũng như trách nhiệm hợp tác giữa doanh nghiệp này với doanh nghiệp khác. Các loại API này thường sẽ cần có quyền hoặc giấy phép mới có thể truy cập được.

API nội bộ: thường sẽ được sử dụng trong nội bộ công ty, không dành cho các bên thứ ba sử dụng. Công dụng của API này là dùng để kết nối các hệ thống và liên kết mọi dữ liệu thuộc quyền sở hữu và sử dụng trong nội bộ của doanh nghiệp đó.

API tổng hợp: kết hợp hai hoặc nhiều API khác nhau để giải quyết các yêu cầu phức tạp của hệ thống. Nếu cần sử dụng dữ liệu từ nhiều nguồn dữ liệu hoặc các ứng dụng khác nhau hay hệ thống chứa nhiều hành vi phức tạp và có phạm vi lớn thì cần sử dụng API tổng hợp để tăng cường hiệu suất và hiệu quả trong việc xử lý.

2.1.4.2. Phân loại dựa và kiến trúc

Dựa vào kiến trúc chúng ta phân ra làm 4 loại API:

API SOAP: dùng giao thức truy cập tương đối đơn giản, cho phép toàn bộ thông tin sẽ được trao đổi dưới dạng XML, trong đó các phần tử và thuộc tính được định nghĩa rõ ràng, đảm bảo tính nhất quán và dễ sử dụng. Tuy nhiên, đây là loại API kém linh hoạt nhất nhưng lại được sử dụng rộng rãi ở những năm về trước.

API RPC: còn được gọi là “Lệnh gọi thủ tục từ xa”. Trong đó, khi máy khách đã thực hiện xong một thủ tục hoặc một hàm trên hệ thống máy chủ, lúc này máy chủ sẽ trả kết quả về máy khách.

API WebSocket: là một phiên bản API web hiện đại chuyển dữ liệu nhờ vào việc sử dụng các đối tượng của JSON. Máy khách và máy chủ có thể trao đổi thông tin 2 chiều nhờ vào API này. Máy chủ còn có thể truyền tải lệnh gọi lại cho máy khách được phép kết nối, nhờ đó nâng cao hiệu quả của ứng dụng API Websocket.

API REST: được coi là API linh hoạt nhất và được sử dụng rộng rãi nhất hiện nay. Với API này, máy khách sẽ gửi đến một yêu cầu dưới dạng dữ liệu, dạng dữ liệu này là JSON. Sau đó máy chủ sẽ dùng các dữ liệu này để thực hiện các hàm nội bộ và cho ra các dữ liệu đầu ra rồi gửi lại cho máy khách.

2.1.5 Bảo mật của API

Sử dụng Token xác thực: Người dùng thực hiện lệnh để nhận dạng, xác thực thông tin chính xác nhằm bảo mật quyền truy cập như khi đăng nhập email vào máy chủ, máy khách sẽ được dùng để nhận token xác thực.

Khóa API: Để nhận dạng ứng dụng và bảo mật quyền truy cập của khóa để thực hiện lệnh cụ thể. Thay vì bảo mật như token, nó sẽ giám sát hoạt động của nội bộ để thu thập dữ liệu sử dụng.

2.1.6 Endpoint và tầm quan trọng của Endpoint

Điểm cuối API là điểm tiếp xúc cuối cùng trong hệ thống giao tiếp của API, hay còn được gọi là EndPoint. Những điểm cuối này bao gồm URL máy chủ, phương thức và những thông số kỹ thuật cụ thể khác, từ đây thông tin được gửi đi và tiếp nhận giữa các hệ thống.

Điểm cuối API rất quan trọng đối với doanh nghiệp vì 2 lý do chính:

Bảo mật: Điểm cuối API khiến hệ thống dễ bị tấn công. Việc giám sát API để ngăn tình trạng lạm dụng là rất quan trọng. Vì nó có thể được public ra ngoài nhiều người biết đến và truy cập với đầy đủ thông số yêu cầu.

Hiệu năng : Điểm cuối API, nhất là những điểm cuối có lưu lượng truy cập cao, ví dụ login hệ thống chặng hạn, thì có thể gây ra tình trạng nghẽn mạng và ảnh hưởng đến hiệu năng hệ thống.

2.1.7 Ưu điểm và nhược điểm của API

Những ưu điểm đáng chú ý của API

API được sử dụng trên hầu hết những ứng dụng của desktop, ứng dụng mobile và các ứng dụng website.

Nó linh hoạt với các định dạng dữ liệu khác nhau khi trả về client.

Với API, mọi người có thể nhanh chóng xây dựng HTTP service khiến công việc lập trình trở nên đơn giản hơn.

Nó sử dụng mã nguồn mở, có chức năng RESTful đầy đủ. Nhờ vậy, có thể sử dụng bởi bất kỳ client nào hỗ trợ Json, XML quen thuộc như trước.

Có thể giao tiếp 2 chiều, được xác nhận trong các giao dịch khác nhau. Từ đó đảm bảo có được độ tin cậy cao.

API có khả năng hỗ trợ đầy đủ các thành phần MVC như Unit Test, Model Binder, Controller, Action,...

Nhược điểm của API

Dù có nhiều ưu điểm, API vẫn còn khá mới. Những người dùng chưa đánh giá được nhiều về nhược điểm của nó. Tuy nhiên, bạn có thể nhìn thấy những nhược điểm dưới đây:

Website API chưa hoàn toàn là RESTful service như thông thường. Nó mới chỉ hỗ trợ mặc định GET, POST mà thôi.

Để dùng được hiệu quả, mọi người cần có kiến thức chuyên sâu, có kinh nghiệm backend tốt nếu không sẽ khó lòng tận dụng triệt để những lợi thế, tính năng mà API có.

Phát triển, nâng cấp hay vận hành API là một quá trình lâu dài và khó khăn. Thậm chí, tiêu tốn khá nhiều chi phí của người vận hành.

Nếu hệ thống bị tấn công trong khi chủ sở hữu chưa giới hạn điều kiện kỹ, việc bảo mật sẽ rất khó khăn.

2.1.8 Úng dụng của API

Web API: Cho phép bạn cập nhật cơ sở dữ liệu, lấy dữ liệu và kết nối hầu hết các ứng dụng đến website. **Ví dụ:** Bạn xây dựng chức năng đăng nhập thông qua Google, Twitter, Facebook,... Hoặc các ứng dụng di động lấy dữ liệu thông qua API.

API trên hệ điều hành: Linux hay Windows có rất nhiều API, các developer tạo ra các phần mềm ứng dụng có thể tương tác trực tiếp với hệ điều hành thông qua các tài liệu nhận được từ API đó là phương thức kết nối hoặc đặc tả các hàm.

Framework API của thư viện phần mềm: Triển khai một API có nhiều cách triển khai khác nhau nó làm cho một chương trình viết bằng ngôn ngữ này nhưng lại có thể sử dụng thư viện được viết bằng ngôn ngữ khác. Nó quy định và mô tả các hành động mong muốn mà các thư viện cung cấp. **Ví dụ** bạn có thể dùng **Java** để yêu cầu một thư viện tạo tài khoản người dùng được viết bằng PHP

2.2 RESTful API

2.2.1 RESTful API là gì?

RESTful API là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các resource. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động...), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP.

Các thành phần chính của RESTful API:

Resource: Đây là đối tượng mà chúng ta muốn truy xuất từ server. Resource được định danh bởi một URI (Uniform Resource Identifier) và có thể được truy xuất bằng HTTP methods như GET, POST, PUT, DELETE.

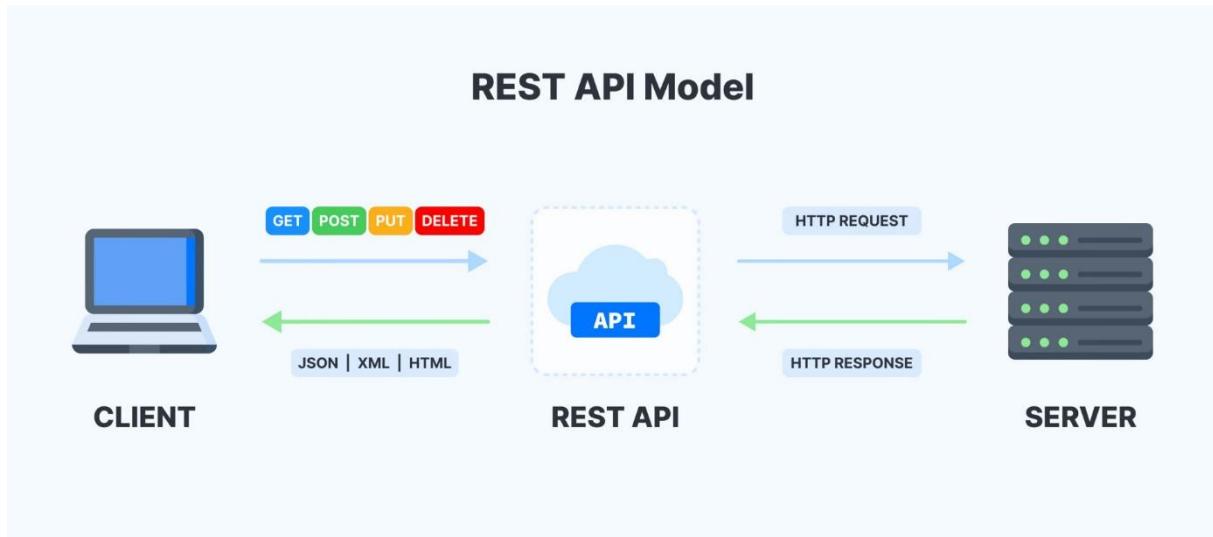
HTTP Method: Đây là cách client gửi yêu cầu đến server. RESTful API hỗ trợ 4 phương thức chính: GET, POST, PUT, DELETE. Những phương thức này phù hợp với các tác vụ cụ thể như lấy thông tin từ server (GET), tạo mới resource (POST), cập nhật resource (PUT) hay xóa resource (DELETE).

Representation: Mỗi resource có thể có nhiều dạng representation khác nhau như HTML, XML, JSON,... Chúng ta có thể yêu cầu server trả về representation của resource bằng cách sử dụng HTTP header Accept.

Status code: Là mã trạng thái mà server trả về sau khi xử lý yêu cầu từ client. RESTful API sử dụng mã trạng thái HTTP để chỉ ra kết quả của yêu cầu, ví dụ như 200 OK cho thành công, 404 Not Found cho không tìm thấy resource,...

Hypermedia: Là các liên kết giữa các resource. Nó giúp cho việc điều hướng giữa các resource dễ dàng hơn, cũng như giúp cho client có thể tự động tìm kiếm và truy xuất các resource liên quan. Hypermedia có thể được định dạng bằng các định dạng như HTML, Atom hay JSON.

2.2.2 Cách thức hoạt động của RESTful API



Hình 2.1 Minh họa Hệ thống Rest Api

Chức năng cơ bản của API RESTful cũng giống như việc duyệt Internet. Client liên hệ với máy chủ bằng cách sử dụng API khi yêu cầu tài nguyên.

Đây là các bước chung cho bất kỳ lệnh gọi REST API:

Client gửi một yêu cầu đến máy chủ. Client làm theo tài liệu API để định dạng yêu cầu theo cách mà máy chủ hiểu được.

Máy chủ xác thực và nhận máy khách có quyền đưa ra yêu cầu đó.

Máy chủ nhận yêu cầu và xử lý trong nội bộ.

Máy chủ trả về một phản hồi đến client. Phản hồi chứa thông tin cho client biết liệu yêu cầu có thành công hay không. Phản hồi cũng bao gồm bất kỳ thông tin nào mà client yêu cầu.

Chi tiết về phản hồi và yêu cầu API REST sẽ khác nhau đôi chút tùy thuộc vào cách các nhà phát triển API thiết kế API.

2.2.3 Request và response trong RESTful API

Trước tiên chúng ta sẽ tìm hiểu một chút về HTTP Response. Khi nhận và phiên dịch một HTTP Request, Server sẽ gửi tín hiệu phản hồi là một HTTP Response bao gồm các thành phần sau:

Một dòng trạng thái (Status-Line)

Không hoặc nhiều hơn các trường Header (General|Response|Entity) được theo sau CRLF

Một phần thân thông báo tùy ý

Dưới đây là một số ví dụ về một HTTP Response:

| X | Headers | Preview | Response | Initiator | Timing | Cookies |
|------------------------|---------|---------|-------------------------------------|-----------|--------|---------|
| ▼ General | | | | | | |
| Request URL: | | | https://anhtester.com/blogs | | | |
| Request Method: | | | GET | | | |
| Status Code: | | | ● 200 OK | | | |
| Remote Address: | | | 14.225.255.250:443 | | | |
| Referrer Policy: | | | strict-origin-when-cross-origin | | | |
| ▼ Response Headers | | | | | | |
| Cache-Control: | | | no-store, no-cache, must-revalidate | | | |
| Content-Encoding: | | | gzip | | | |
| Content-Type: | | | text/html; charset=UTF-8 | | | |
| Date: | | | Wed, 01 Nov 2023 05:46:15 GMT | | | |
| Expires: | | | Thu, 19 Nov 1981 08:52:00 GMT | | | |
| Pragma: | | | no-cache | | | |
| Server: | | | LiteSpeed | | | |
| Vary: | | | Accept-Encoding | | | |
| X-Powered-By: | | | PHP/7.4.33 | | | |
| ► Request Headers (20) | | | | | | |

Hình 2.2 HTTP Response với mã trạng thái 200 OK

X Headers Preview Response Initiator Timing Cookies

▼ General

Request URL: https://anhtester.com/abc

Request Method: GET

Status Code: ● 404 Not Found

Remote Address: 14.225.255.250:443

Referrer Policy: strict-origin-when-cross-origin

▼ Response Headers

Content-Encoding: gzip

Content-Type: text/html; charset=UTF-8

Date: Wed, 01 Nov 2023 05:48:35 GMT

Server: LiteSpeed

Vary: Accept-Encoding

X-Powered-By: PHP/7.4.33

► Request Headers (19)

Hình 2.3 HTTP Response với mã trạng thái 404 Not Found

Body Cookies Headers (12) Test Results

Status: 200 OK Time: 68 ms Size: 492 B

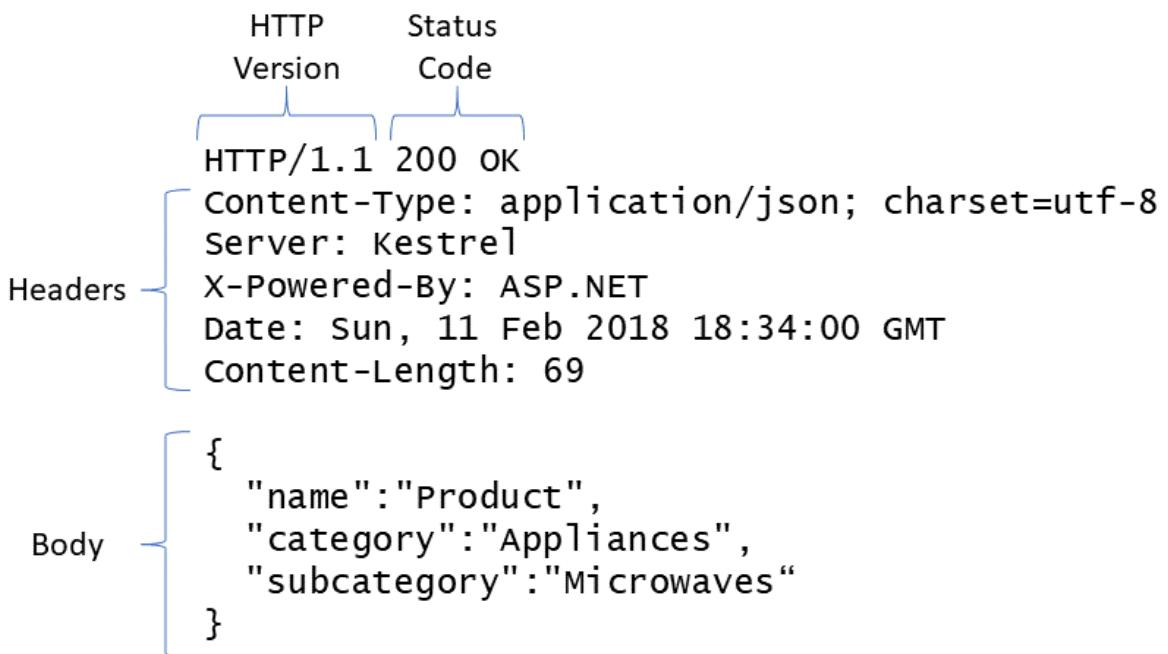
Pretty Raw Preview Visualize JSON 

```

1  {
2      "message": "Success",
3      "response": {
4          "id": 5,
5          "username": "anhtester",
6          "firstName": "Anh",
7          "lastName": "Tester",
8          "email": "thaian.it15@gmail.com",
9          "phone": "0939206009",
10         "userStatus": 1
11     }
12 }
```

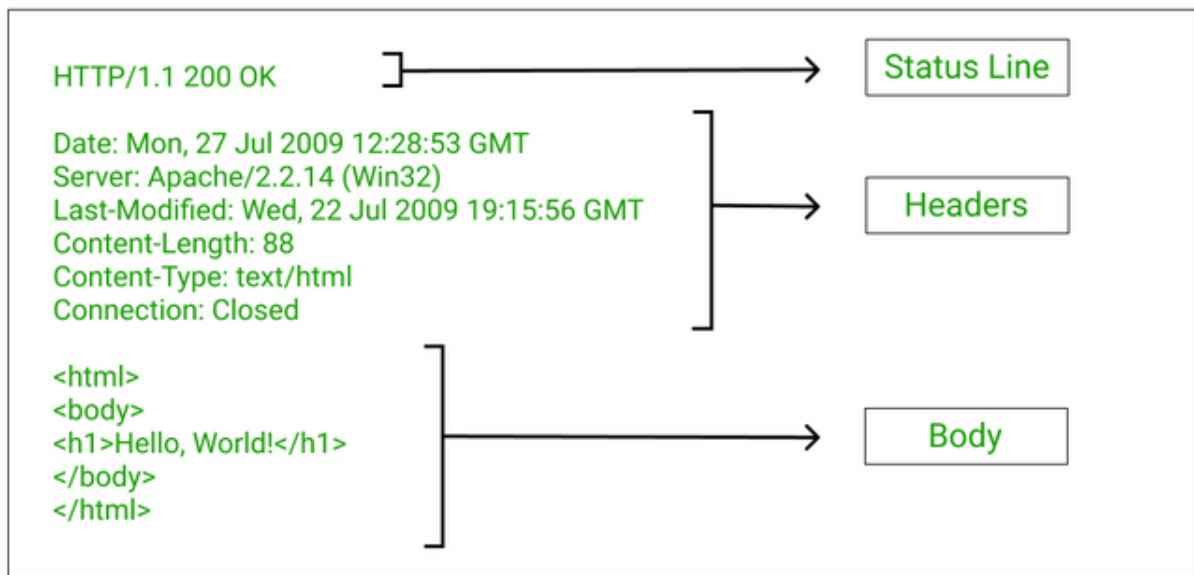
Hình 2.4 Phản hồi API thành công (Status 200 OK) trên Postman

Cấu trúc của Response trả về với Body dạng JSON như sau:



Hình 2.5 Cấu trúc của Response trả về với Body dạng JSON

Cấu trúc của Response trả về với Body dạng HTML/XML như sau:



Hình 2.6 Cấu trúc của Response trả về với Body dạng HTML/XML

2.2.4 Status code trong RESTful API

1xx Information (Thông tin)

100 Continue: Chỉ một phần của Request được nhận bởi Server (có thể là header và Client cần gửi tiếp body), nhưng miễn là nó không bị loại bỏ, Client nên tiếp tục với Request.

101 Switching Protocols: Requester đã hỏi Server về việc thanh đổi Protocol và Server đã chấp nhận điều đó

2xx Success (Thành công)

200 OK: Request đã được tiếp nhận và xử lý thành công. Các Response thực tế trả về sẽ phụ thuộc vào phương thức HTTP của Request. Trong một GET Request, Response sẽ chứa một thực thể tương ứng với các tài nguyên yêu cầu, trong một POST Request, Response sẽ chứa một thực thể mô tả hoặc chứa các kết quả của các action.

201 Created: Request đã được xử lý, kết quả của việc xử lý tạo ra một resource mới.

202 Accepted: Request được chấp nhận cho xử lý, nhưng việc xử lý chưa hoàn thành.

203 Non-authoritative Information (Xuất hiện từ HTTP/1.1): Server là nơi chuyển đổi proxy (ví dụ một Web accelerator) đã nhận được 200 OK nhưng nó trả về một phiên bản thay đổi (có thể là header) của Response nguyên gốc.

204 No Content: Server đã xử lý thành công request nhưng không trả về bất cứ content nào.

205 Reset Content: Server đã xử lý thành công request nhưng không trả về bất cứ content nào. Không giống với 204 No Content Response này yêu cầu phía Client phải thiết lập lại document view.

206 Partial Content: Server chỉ trả về một phần của resouce(dạng byte) do một range header được gửi bởi phía Client. Các Range Header được sử dụng bởi Client để cho phép nối lại các phần của file download bị dán đoạn hoặc chia thành nhiều luồng download.

3xx Redirection (Sự chuyển hướng lại)

300 Multiple Choices: Một danh sách các link. Người sử dụng có thể chọn một link và tới vị trí đó. Tối đa 5 địa chỉ. Ví dụ: List các file video với format khác nhau

301 Moved Permanently: Request hiện tại và các request sau được yêu cầu di chuyển tới một URI mới.

302 Found: Đây là một ví dụ cho thấy sự mâu thuẫn giữa thực tiễn và quy chuẩn. Ở phiên bản HTTP/1.0 nó có nghĩa là yêu cầu Client chuyển hướng đến một URL tạm thời (tương tự như là 301 Moved Permanently) nhưng phần lớn các browser lại thực hiện nó với ý nghĩa của 303 See Other(sẽ nói sau đây). Do đó từ phiên bản HTTP/1.1 có thêm hai mã 303 và 307 để phân biệt rõ hành vi, nhưng một số ứng dụng web và framework vẫn sử dụng 302 như thể là 303.

303 See Other (Xuất hiện từ HTTP/1.1): Response trả về của Request có thể tìm thấy ở một URL khác bằng cách sử dụng phương thức GET.

304 Not Modified: Đây là Status-Code tới một If-Modified-Since hoặc If-None-Match header, nơi mà URL không được chỉnh sửa từ ngày cụ thể.

305 Use Proxy: Tài nguyên yêu cầu chỉ có sẵn thông qua một proxy, địa chỉ mà được cung cấp trong các Response. Nhiều HTTP Client (như Mozilla và Internet Explorer) không xử lý một cách chính xác phản ứng với mã trạng thái này, chủ yếu là vì các lý do an ninh.

306 Switch Proxy: Mã này hiện không còn được sử dụng, ý nghĩa ban đầu của nó là "Các Request tiếp theo nên sử dụng các proxy được chỉ định".

307 Temporary Redirect (xuất hiện từ HTTP/1.1): Trong trường hợp này, Request hiện tại cần được lặp lại một URI khác nhưng các Request trong tương lai vẫn sử dụng URI gốc.

4xx: Client Error (Lỗi Client)

400 Bad Request: Server không thể xử lý hoặc sẽ không xử lý các Request lỗi của phía client (ví dụ Request có cú pháp sai hoặc Request lừa đảo định tuyến ...)

401 Unauthorized: Tương tự như 403 Forbidden nhưng được sử dụng khi yêu cầu xác thực là bắt buộc và đã không thành công. Các Response bắt buộc phải có thành phần WWW-Authenticate chứa các thách thức với tài nguyên được yêu cầu. Một số trang web vấn đề HTTP 401 khi một địa chỉ IP bị cấm từ các trang web (thường là các tên miền trang web) và địa chỉ cụ thể là từ chối quyền truy cập một trang web.

402 Payment Required: Hiện tại mã này chưa được sử dụng và nó được dự trù cho tương lai. Mục đích ban đầu là mã này có thể được sử dụng như là một phần của đề án tiền mặt hoặc micropayment kỹ thuật số, nhưng điều đó đã không xảy ra, và mã này thường không được sử dụng. Google API sử dụng Status-Code này nếu một nhà phát triển đặc biệt đã vượt quá giới hạn số lần yêu cầu.

403 Forbidden: Request là hợp lệ nhưng server từ chối đáp ứng nó. Nó có nghĩa là trái phép, người dùng không có quyền cần thiết để tiếp cận với các tài nguyên.

404 Not Found: Các tài nguyên hiện tại không được tìm thấy nhưng có thể có trong tương lai. Các request tiếp theo của Client được chấp nhận.

405 Method Not Allowed: Request method không được hỗ trợ cho các tài nguyên được yêu cầu. Ví dụ Một GET request đến một POST resource, PUT Request gọi đến một tài nguyên chỉ đọc.

406 Not Acceptable: Server chỉ có thể tạo một Response mà không được chấp nhận bởi Client.

407 Proxy Authentication Required: Bạn phải xác nhận với một Server ủy quyền trước khi Request này được phục vụ.

408 Request Timeout: Request tồn thời gian dài hơn thời gian Server được chuẩn bị để đợi.

409 Conflict: Request không thể được hoàn thành bởi vì sự xung đột, ví dụ như là xung đột giữa nhiều chỉnh sửa đồng thời.

410 Gone: Các resource được yêu cầu không còn nữa và sẽ không có sẵn một lần nữa, khi gặp mã lỗi này Client không nên cố gắng tìm kiếm các tài nguyên này ở những lần sau.

411 Length Required: Content-Length không được xác định rõ. Server sẽ không chấp nhận Request nào không có nó.

412 Precondition Failed: Server sẽ không đáp ứng một trong những điều kiện tiên quyết của Client trong Request.

413 Payload Too Large: Server sẽ không chấp nhận yêu cầu, bởi vì đối tượng yêu cầu là quá rộng. Trước đây nó gọi là "Request Entity Too Large".

414 URI Too Long: URI được cung cấp là quá dài để Server xử lý, thường là kết quả của quá nhiều dữ liệu được mã hóa như là một truy vấn chuỗi của một GET Request, trong trường hợp đó nó phải được chuyển đổi sang một POST Request. Trước đây được gọi là "Request-URI Too Long"

415 Unsupported Media Type: Server sẽ không chấp nhận Request, bởi vì kiểu phương tiện không được hỗ trợ. Ví dụ khi Client upload một ảnh có định dạng image/svg+xml, nhưng server yêu cầu một định dạng khác.

416 Range Not Satisfiable: Client yêu cầu một phần của tập tin nhưng server không thể cung cấp nó. Trước đây được gọi là "Requested Range Not Satisfiable"

417 Expectation Failed: Máy chủ không thể đáp ứng các yêu cầu của trường Expect trong header.

5xx: Server Error (Lỗi Server)

500 Internal Server Error: Một thông báo chung chung, được đưa ra khi Server gặp phải một trường hợp bất ngờ, Message cụ thể là không phù hợp.

501 Not Implemented: Server không công nhận các Request method hoặc không có khả năng xử lý nó.

502 Bad Gateway: Server đã hoạt động như một gateway hoặc proxy và nhận được một Response không hợp lệ từ máy chủ nguồn.

503 Service Unavailable: Server hiện tại không có sẵn (Quá tải hoặc được down để bảo trì). Nói chung đây chỉ là trạng thái tạm thời.

504 Gateway Timeout: Server đã hoạt động như một gateway hoặc proxy và không nhận được một Response từ máy chủ nguồn.

505 HTTP Version Not Supported: Server không hỗ trợ phiên bản “giao thức HTTP”.

Một số status phổ biến thường dùng:

200 OK – Trả về thành công cho những phương thức GET, PUT, PATCH hoặc DELETE.

201 Created – Trả về khi một Resouce vừa được tạo thành công.

204 No Content – Trả về khi Resource xoá thành công.

304 Not Modified – Client có thể sử dụng dữ liệu cache, resource server không đổi gì.

400 Bad Request – Request không hợp lệ

401 Unauthorized – Request cần có xác thực.

403 Forbidden – bị từ chối không cho phép.

404 Not Found – Không tìm thấy resource từ URL

405 Method Not Allowed – Phương thức không cho phép với user hiện tại.

410 Gone – Resource không còn tồn tại, Version cũ đã không còn hỗ trợ.

415 Unsupported Media Type – Không hỗ trợ kiểu Resource này.

422 Unprocessable Entity – Dữ liệu không được xác thực

429 Too Many Requests – Request bị từ chối do bị giới hạn

2.3. Tổng quan về kiểm thử API

2.3.1. Kiểm thử API là gì?

Kiểm thử API (API testing) là một loại kiểm thử phần mềm bao gồm kiểm tra trực tiếp các giao diện lập trình ứng dụng (API) và là một phần của kiểm thử tích hợp để xác định xem chúng có đáp ứng mong đợi về chức năng, độ tin cậy, hiệu suất và bảo mật không.

Kiểm thử API sẽ được tiến hành ở tầng nghiệp vụ (business layer), bởi vì các API đều không có GUI. Mọi dữ liệu đều sẽ được trao đổi từ JSON hoặc XML qua những yêu cầu và phản hồi HTTP trong quá trình kiểm thử API. Đây là các hệ thống công nghệ độc lập, làm việc với đa dạng các ngôn ngữ lập trình và công nghệ khác nhau.

2.3.2. Tại sao cần kiểm thử API

Trong quá trình triển khai dự án, phần server và client làm việc độc lập với nhau nên nhiều khi Client chưa làm xong. Ta không thể đợi làm xong phần client mới test được các API phía server. Khi ấy ta cần test API bằng công cụ khác luôn và việc test hoàn toàn không phụ thuộc gì vào client.

Kể cả khi client làm xong rồi, nếu mình test trên client mà thấy lỗi liên quan đến logic và dữ liệu thì cũng cần test thêm cả API để biết chính xác là server sai hay client sai. Đây là việc sẽ giúp tester tìm ra lỗi nhanh hơn.

Khi làm hệ thống web services, dự án của mình chỉ viết API cho bên khác dùng, mình sẽ không có client để test giống như các dự án khác. Vì vậy phải dùng công cụ để test API hoàn toàn.

2.3.3. Phương pháp tiếp cận API testing

API Testing Approach là chiến lược do Tester thiết lập để kiểm thử API sau khi môi trường kiểm thử đã sẵn sàng. Phương pháp này giúp Tester hiểu rõ chức năng, kỹ thuật kiểm thử, tham số đầu vào và cách thực hiện test case. Để thực hiện API Testing hiệu quả, cần:

Xác định rõ chức năng và phạm vi của API.

Áp dụng các kỹ thuật kiểm thử phù hợp (phân vùng tương đương, giá trị biên, đoán lỗi).

Lên kế hoạch và chọn tham số đầu vào thích hợp.

Thực hiện test case và so sánh kết quả.

2.3.4. Một số phương pháp kiểm thử API phổ biến

Bảng 2.1 Một số phương pháp kiểm thử API phổ biến

| Loại kiểm thử | Mục đích |
|--------------------------|--|
| Unit testing | Kiểm tra từng đơn vị code của API, như hàm, phương thức hay lớp |
| Function testing | Kiểm tra chức năng của API từng phần hoặc toàn bộ. |
| Interface testing | Kiểm tra giao diện API (cấu trúc, kiểu dữ liệu, tham số, mã phản hồi) để đảm bảo rằng các yêu cầu và phản hồi được truyền đi và nhận về đúng cách. |

| | |
|-----------------------------|---|
| Security testing | Kiểm tra tính bảo mật của API bằng cách xác định và khắc phục các lỗ hổng bảo mật (xác thực, phân quyền, kiểm tra đầu vào, mã hóa) |
| Non-function testing | Kiểm tra các yếu tố phi chức năng của API như tương tác người dùng, độ tin cậy, khả năng mở rộng, khả năng khôi phục, và độ ổn định. |
| Integration testing | Kiểm tra tích hợp của API với các dịch vụ hoặc hệ thống khác. |
| Load testing | Kiểm tra khả năng xử lý một lượng lớn yêu cầu và đảm bảo rằng API không gặp vấn đề hiệu suất. |
| Regression testing | Kiểm tra lại API sau khi có các thay đổi hoặc bổ sung mới để đảm bảo rằng những thay đổi đó không làm ảnh hưởng đến tính đúng đắn và chức năng của API. |

2.3.5. Sự khác nhau giữa API testing và Unit testing

Bảng 2.2 So sánh API testing và Unit testing

| Unit testing | API testing |
|--|--|
| Lập trình viên thực hiện | Tester thực hiện |
| Từng chức năng được kiểm thử | Các chức năng liên quan đến nhau cần được kiểm thử |
| Lập trình viên có thể truy cập vào source code | Tester không thể truy cập vào source code |

| | |
|--|------------------------------------|
| Phải kiểm tra cả UI | Chỉ kiểm tra các hàm API |
| Chỉ các chức năng đơn giản được kiểm thử | Tất cả các chức năng được kiểm thử |
| Giới hạn phạm vi | Phạm vi rộng hơn |
| Thường được chạy trước khi build | Thường được chạy sau khi build |

2.4. Công cụ kiểm thử API – Postman

2.4.1. Postman là gì?

Postman là một nền tảng hỗ trợ phát triển và kiểm thử API được sử dụng phổ biến trong lĩnh vực kiểm thử phần mềm. Công cụ này giúp người dùng thiết kế, gửi yêu cầu (request), nhận và phân tích phản hồi (response) từ API một cách trực quan và hiệu quả.

Ban đầu, Postman được phát triển như một tiện ích mở rộng (extension) trên trình duyệt Google Chrome, nhưng hiện nay đã trở thành một ứng dụng độc lập đa nền tảng, có thể cài đặt trên Windows, macOS và Linux.

Mục đích sử dụng:

Hỗ trợ kiểm thử chức năng của API (REST, SOAP, GraphQL, v.v.).

Giúp xác minh phản hồi của máy chủ, bao gồm mã trạng thái (status code), tiêu đề (headers) và dữ liệu trả về (body).

Dùng để tự động hóa quy trình kiểm thử thông qua collection và script.

Cho phép chia sẻ, tài liệu hóa và quản lý API trong môi trường làm việc .

Các tính năng nổi bật:

Giao diện trực quan, dễ thao tác, phù hợp cho cả người mới học kiểm thử.

Hỗ trợ nhiều phương thức HTTP như GET, POST, PUT, PATCH, DELETE.

Có khả năng lưu trữ và quản lý tập hợp các yêu cầu (Collections).

Cho phép chèn script kiểm thử tự động bằng JavaScript.

Tích hợp tính năng báo cáo, môi trường (Environments) và làm việc (Workspaces).

2.4.2. Các thành phần chính trong Postman

Postman được cấu thành từ nhiều thành phần (elements) khác nhau giúp người dùng dễ dàng tổ chức, quản lý và thực hiện kiểm thử API hiệu quả. Một số thành phần chính bao gồm:

Setting: Chứa các cài đặt cơ bản liên quan đến tài khoản, giao diện hiển thị và cấu hình của Postman. Ngoài ra, phần này còn quản lý các nguồn dữ liệu đầu vào nhằm đảm bảo hoạt động ổn định của công cụ.

Collection: Giống như một thư viện trong Postman, nơi lưu trữ các request (yêu cầu) mà người dùng tạo ra. Collection giúp các API theo module, chức năng hoặc dự án, từ đó thuận tiện cho việc quản lý và kiểm thử tự động.

API Content: Là phần quan trọng nhất trong Postman, hỗ trợ người dùng trong việc gửi và nhận dữ liệu khi kiểm thử API.

API Content bao gồm ba thành phần nhỏ:

Environment: Lưu trữ thông tin về môi trường (biến, URL, token, v.v.), giúp dễ dàng chuyển đổi giữa các môi trường kiểm thử mà không cần chỉnh sửa thủ công từng request.

Request: Chứa các thông tin chính của một yêu cầu gửi tới API, như phương thức (GET, POST,...), URL, headers và body.

Response: Hiển thị dữ liệu phản hồi mà API trả về sau khi gửi yêu cầu, giúp người kiểm thử đánh giá kết quả.

Ngoài ba thành phần chính trên, Postman còn hỗ trợ thêm các công cụ như Mock Server, Monitor và Flow nhằm mở rộng khả năng kiểm thử, mô phỏng và tự động hóa quy trình.

2.4.3. Các chức năng chính trong Postman

Postman cho phép người dùng gửi các loại HTTP request như GET, POST, PUT, DELETE, cùng dữ liệu ở dạng text, form-data hoặc JSON.

Xem và phân tích phản hồi (Response): Kết quả phản hồi từ API được hiển thị trực quan dưới dạng JSON, XML, HTML, hoặc text, giúp dễ kiểm tra tính chính xác của dữ liệu trả về.

Hỗ trợ xác thực (Authorization): Hỗ trợ nhiều cơ chế xác thực như API Key, Bearer Token, Basic Auth, OAuth 2.0,... để truy cập các API bảo mật.

Tùy chỉnh Header và Parameter: Cho phép thêm, chỉnh sửa hoặc xóa các Request Headers, Params, giúp mô phỏng nhiều tình huống kiểm thử khác nhau.

Kiểm thử tự động bằng Script (Tests): Cung cấp khu vực viết JavaScript script để tự động kiểm tra phản hồi API, xác minh mã trạng thái, dữ liệu trả về,...

Quản lý tập hợp và môi trường (Collections & Environments): Giúp tổ chức các request, thay đổi nhanh các biến môi trường (như URL, Token) khi kiểm thử ở nhiều môi trường khác nhau.

2.4.4. Các thành phần khác trong Postman

Bên cạnh các thành phần và chức năng chính, Postman còn tích hợp nhiều công cụ hỗ trợ khác giúp người dùng thao tác thuận tiện và nâng cao hiệu quả kiểm thử API. Các thành phần này có thể được chia thành hai : công cụ thao tác trong giao diện làm việc và công cụ hỗ trợ quản lý và cộng tác.

Công cụ thao tác trong giao diện làm việc:

New: Cho phép tạo mới các thành phần như Request, Collection, Environment hoặc API.

Import: Dùng để nhập (import) các tập tin hoặc dự án Postman có sẵn từ bên ngoài, chẳng hạn như file JSON hoặc link chia sẻ.

Tab Request: Hiển thị tiêu đề của request đang mở, giúp người dùng có thể thao tác song song trên nhiều tab.

HTTP Request: Cho phép lựa chọn phương thức HTTP (GET, POST, PUT, DELETE, PATCH,...) phù hợp với yêu cầu kiểm thử.

Request URL: Là địa chỉ endpoint nơi API nhận và xử lý yêu cầu từ client.

Params: Dùng để thêm các tham số truyền trên URL (query parameters) dưới dạng cặp key – value.

Authorization: Cấu hình thông tin xác thực để truy cập API, hỗ trợ nhiều hình thức như API Key, Bearer Token hoặc Basic Auth.

Headers: Cho phép thêm hoặc chỉnh sửa các thông tin tiêu đề (header) của request, ví dụ như Content-Type hoặc Accept.

Body: Là nơi khai báo nội dung dữ liệu gửi đi trong request, thường áp dụng với các phương thức POST hoặc PUT.

Pre-request Script: Cho phép viết các đoạn mã JavaScript thực thi trước khi gửi request, thường dùng để sinh token hoặc thiết lập giá trị biến.

Tests: Cung cấp khu vực viết script để kiểm tra tự động phản hồi (response), ví dụ xác minh mã trạng thái, nội dung trả về hoặc thời gian phản hồi.

Save: Dùng để lưu lại các thay đổi trong request hoặc collection hiện tại.

Công cụ hỗ trợ quản lý và cộng tác:

Collections: Cho phép và quản lý các request theo chức năng hoặc dự án, thuận tiện cho việc tái sử dụng và kiểm thử tự động.

Runner: Hỗ trợ chạy hàng loạt các request trong cùng một collection, phục vụ mục đích kiểm thử tự động hoặc kiểm thử hồi quy.

My Workspace: Là không gian làm việc cá nhân hoặc , nơi người dùng tổ chức và lưu trữ các collection, environment hoặc tài nguyên liên quan.

Invite: Cho phép mời người khác tham gia vào workspace để cùng cộng tác, chia sẻ request hoặc kết quả kiểm thử.

History: Lưu lại danh sách các request đã gửi, giúp dễ dàng xem lại hoặc thực hiện lại khi cần thiết.

CHƯƠNG 3: KIỂM THỬ TỰ ĐỘNG VỚI SELENIUM

3.1. Khái quát về Selenium

3.1.1. Giới thiệu chung

Selenium là một công cụ hỗ trợ kiểm thử tự động cho các ứng dụng Web. Selenium hỗ trợ kiểm thử trên hầu hết các trình duyệt phổ biến hiện nay như Firefox, Internet Explorer, Safari, ... cũng như các hệ điều hành chủ yếu như Windows, Linux, Mac,... Selenium cũng hỗ trợ một số lớn các ngôn ngữ lập trình Web phổ biến hiện nay như C#, Java, Perl, PHP, Python, Ruby,... Công cụ này có thể kết hợp thêm với một số công cụ khác như Junit và TestNG nhưng với người dùng thông thường chỉ cần chạy tự động mà không cần cài thêm các công cụ hỗ trợ.

3.1.2. Lịch sử hình thành và phát triển

Selenium bắt đầu vào năm 2004 bởi Jason Huggins khi làm việc tại ThoughtWorks. Ban đầu, nó là một công cụ đơn giản để kiểm thử nội bộ cho các ứng dụng web của công ty, sau đó nhanh chóng phát triển nhờ vào tính linh hoạt và khả năng mở rộng. Một số cột mốc quan trọng trong quá trình phát triển của Selenium gồm:

Selenium RC (Remote Control): Phiên bản đầu tiên của Selenium cho phép tự động hóa trình duyệt từ xa, giúp thực hiện các thao tác từ nhiều ngôn ngữ lập trình khác nhau.

Selenium WebDriver: Ra mắt năm 2009, WebDriver cải tiến mạnh mẽ Selenium RC bằng cách tương tác trực tiếp với trình duyệt, đem lại hiệu suất cao và độ ổn định tốt hơn.

Selenium 2.0: Đây là phiên bản hợp nhất Selenium RC và WebDriver, tạo nền tảng vững chắc cho Selenium như hiện nay.

Selenium 3 và 4: Phiên bản 3 giới thiệu nhiều tính năng nâng cao và chuẩn hóa WebDriver. Phiên bản 4 (ra mắt vào năm 2021) nâng cấp thêm giao diện người dùng (Selenium IDE) và bổ sung hỗ trợ các tính năng mới như xử lý thao tác trên trình duyệt và cài tiến API.

3.1.3. Đặc điểm của Selenium

Mã nguồn mở: Đây là điểm mạnh nhất của Selenium khi so sánh với các test tool khác. Vì là mã nguồn mở nên chúng ta có thể sử dụng mà không phải lo lắng về phí bản quyền hay thời hạn sử dụng.

Cộng đồng hỗ trợ: vì là mã nguồn mở nên Selenium có một cộng đồng hỗ trợ khá mạnh mẽ. Bên cạnh đó, Google là nơi phát triển Selenium nên chúng ta hoàn toàn có thể yên tâm về sự hỗ trợ miễn phí khi có vấn đề về Selenium. Tuy nhiên, đây cũng là một điểm yếu của Selenium. Vì công cụ này hoàn toàn miễn phí, cộng đồng lại đông nên một vấn đề có thể nhiều giải pháp, và có thể một số giải pháp là không hữu ích. Mặc khác, chúng ta không thể hối thúc hay ra deadline cho sự hỗ trợ:

Selenium hỗ trợ nhiều ngôn ngữ lập trình.

Selenium hỗ trợ chạy trên nhiều hệ điều hành khác nhau với mức độ chỉnh sửa script hầu như là không có. Thực sự thì điều này phụ thuộc phần lớn vào khả năng viết script của người dùng.

Chạy test case ở background. Khi chúng ta thực thi một test script, chúng ta hoàn toàn có thể làm việc khác trên cùng một máy tính. Điều này hỗ trợ chúng ta không cần tốn quá nhiều tài nguyên máy móc khi chạy test script.

Không hỗ trợ Win app. Selenium thực sự chỉ hỗ trợ chúng ta tương tác với Browser mà không hỗ trợ chúng ta làm việc với các Win app, kể cả Win dialog như Download/Upload. Vậy nên, để xử lý các trường hợp cần tương tác với hệ thống hay một app thứ ba, chúng ta cần một hay nhiều thư viện khác như AutoIt hay Coded UI.

Là một công cụ hỗ trợ kiểm tra tính năng nên Selenium không có khả năng giả lập nhiều người dùng ảo cùng một lúc. Công việc của nó là chạy kiểm thử tự động dựa trên một kịch bản đã được thiết kế từ trước. Qua đó chúng ta có thể chắc chắn rằng đối tượng kiểm thử có hoạt động đúng như mong đợi hay không.

3.1.4. Các thành phần của Selenium

Selenium là một bộ công cụ hỗ trợ kiểm thử tự động các tính năng của ứng dụng trên nền Web, bao gồm 4 thành phần: Selenium IDE, Selenium Grid, Selenium 1.0 (hay Selenium Remote Control - Selenium RC) và Selenium 2.0 (hay Selenium WebDriver).

Mỗi loại có một vai trò cụ thể trong việc hỗ trợ sự phát triển của tự động hóa kiểm thử ứng dụng web.

Selenium IDE (Integrated Development Environment): được phát triển dưới hình thức add-on của Firefox. Chúng ta chỉ có thể record trên trình duyệt FireFox, nhưng bù lại, chúng ta có thể playback trên các trình duyệt khác như là IE, Chrome....

Selenium có thể sinh code tự động hoặc nạp các đoạn mã viết tay. Công cụ này cung cấp chức năng “thu và chạy lại”. Sau đó chạy lại các câu lệnh này để kiểm thử. Chức năng này rất hữu dụng giúp tiết kiệm thời gian viết kịch bản kiểm thử. Selenium IDE còn cho phép lưu kịch bản đã thu dưới nhiều loại ngôn ngữ lập trình khác nhau như Java, PHP, C#, Ruby....

Selenium Core: Đã được tích hợp trong Selenium IDE. Selenium Core là một công cụ chạy các test script viết bằng Selenese. Thế mạnh của công cụ này là có thể chạy test script trên gần như tất cả các trình duyệt, nhưng lại yêu cầu được cài đặt trên máy chủ của website cần kiểm tra. Điều này là không thể khi Tester không có quyền truy cập đến máy chủ đó.

Selenium RC (Remote Control): là một framework kiểm thử cho phép thực hiện nhiều hơn và tuyển tính các hành động trên trình duyệt. Nó cho phép cho phép các nhà phát triển tự động hóa kiểm thử sử dụng một ngôn ngữ lập trình cho tính linh hoạt tối đa và mở rộng trong việc phát triển logic thử nghiệm.

Công cụ này có thể nhận các test script được thu bởi Selenium IDE, cho phép chỉnh sửa, cải tiến linh động bằng nhiều ngôn ngữ lập trình khác nhau. Sau đó khởi động một trong các trình duyệt Web được chỉ định để thực thi kiểm thử trực tiếp trên trình duyệt đó. Selenium RC còn cung cấp khả năng lưu lại kết quả kiểm thử; cung cấp một API (Application Programming Interface) và thư viện cho mỗi ngôn ngữ được hỗ trợ: HTML, Java, C#, Perl, PHP, Python và Ruby. Khả năng sử dụng Selenium RC với một ngôn ngữ lập trình bậc cao để phát triển các trường hợp kiểm thử cũng cho phép kiểm thử tự động được tích hợp với một dự án xây dựng môi trường tự động.

Selenium WebDriver: là sự kế thừa từ Selenium Remote Control, làm việc trực tiếp với trình duyệt ở mức hệ điều hành, cho phép gửi lệnh trực tiếp đến trình duyệt và xuất ra kết quả

Selenium-Grid: là một hệ thống hỗ trợ người dùng thực thi test script trên nhiều trình duyệt một cách song song mà không cần phải chỉnh sửa test script.

Thực hiện phương pháp kiểm tra phân bổ, phối hợp nhiều Selenium RC để có thể thực thi trên nhiều trình duyệt Web khác nhau trong cùng một lúc nhằm giảm thiểu thời gian thực hiện.

3.2. Selenium Webdriver

3.2.1 Đặc điểm của Selenium Webdriver

Kiến trúc hướng trực tiếp: Thay vì chạy qua máy chủ trung gian như Selenium RC, WebDriver tương tác trực tiếp với trình duyệt bằng cách sử dụng các API nội tại của trình duyệt. Điều này giúp WebDriver có tốc độ nhanh hơn và độ ổn định cao hơn.

Hỗ trợ nhiều trình duyệt: Selenium WebDriver tương thích với nhiều trình duyệt phổ biến như Google Chrome, Mozilla Firefox, Safari, Microsoft Edge, và Opera. Điều này giúp kiểm thử trên các môi trường khác nhau và đảm bảo tính nhất quán cho ứng dụng web.

Đa nền tảng: WebDriver chạy trên nhiều hệ điều hành, bao gồm Windows, macOS, và Linux, giúp các đội phát triển dễ dàng thực hiện kiểm thử trên các môi trường khác nhau.

Đa ngôn ngữ lập trình: Selenium WebDriver hỗ trợ nhiều ngôn ngữ lập trình như Java, Python, C#, Ruby, và JavaScript. Điều này cho phép các lập trình viên có thể viết kịch bản kiểm thử trong ngôn ngữ mà họ thành thạo.

3.2.2 Các tính năng nổi bật của Selenium WebDriver

Hỗ trợ tương tác phong phú với trang web: WebDriver cho phép mô phỏng hầu hết các thao tác của người dùng như click chuột, nhập văn bản, chọn dropdown, di chuyển con trỏ chuột, kéo thả, và cuộn trang.

Quản lý và chờ đợi trạng thái phần tử: WebDriver có cơ chế chờ đợi cả Implicit Waits (chờ ngầm định) và Explicit Waits (chờ tường minh), giúp đảm bảo các phần tử được tải đầy đủ trước khi thực hiện các thao tác. Điều này rất hữu ích khi kiểm thử các ứng dụng web có nội dung động hoặc thời gian tải không đồng nhất.

Hỗ trợ đa tab và đa cửa sổ: WebDriver cho phép điều khiển các tab hoặc cửa sổ mới, giúp tester có thể kiểm thử các chức năng cần mở liên kết trong tab hoặc cửa sổ khác.

Xử lý JavaScript và Pop-up: WebDriver có thể tương tác với các đoạn mã JavaScript và xử lý các cửa sổ pop-up, thông báo, hoặc hộp thoại xác nhận (alert). Điều này giúp mô phỏng hoàn chỉnh hơn các thao tác người dùng thực tế.

Tích hợp với các framework kiểm thử: Selenium WebDriver có thể dễ dàng tích hợp với các framework kiểm thử như TestNG, JUnit (Java), hoặc PyTest (Python) để tổ chức các kịch bản kiểm thử, thực hiện kiểm thử song song, và tạo báo cáo kết quả kiểm thử.

3.2.3 Ưu điểm và hạn chế của Selenium WebDriver

Ưu điểm của Selenium WebDriver:

Hiệu suất cao và đáng tin cậy: Nhờ vào kiến trúc trực tiếp, WebDriver có khả năng thực thi nhanh và ổn định hơn so với Selenium RC.

Mở rộng và linh hoạt: Có thể mở rộng dễ dàng bằng cách kết hợp với các công cụ khác và framework kiểm thử, hoặc tích hợp với CI/CD.

Phù hợp với môi trường Agile và DevOps: Tốc độ và tính linh hoạt của Selenium WebDriver rất phù hợp để tích hợp vào quy trình Agile và DevOps, giúp cải thiện chất lượng phần mềm liên tục.

Hạn chế của Selenium WebDriver:

Chỉ hỗ trợ ứng dụng web: WebDriver không thể kiểm thử ứng dụng desktop hoặc ứng dụng di động mà chỉ dành cho ứng dụng web.

Khó khăn trong việc kiểm thử hình ảnh: WebDriver không hỗ trợ tốt kiểm thử hình ảnh hoặc xác minh giao diện.

Headless Browser Testing: WebDriver hỗ trợ chế độ headless, tức là kiểm thử không cần mở giao diện trình duyệt. Điều này giúp giảm tải hệ thống và tăng tốc độ kiểm thử, đặc biệt hữu ích trong các môi trường CI/CD.

CHƯƠNG 4. GIỚI THIỆU TRANG WEB BÁN GIÀY

Trang web được sử dụng trong dự án này là một ứng dụng web về cửa hàng giày. Đặc biệt trong thời đại mà mua sắm online đang phát triển như hiện nay thì việc các cửa hàng giày cần một trang web để có thể bán hàng tại cửa hàng và việc bán hàng online là một điều khá cần thiết. Đồng thời, khách hàng cũng có thể truy cập website cửa hàng để mua hàng mà không cần đến tận nơi để mua.

Với giao diện người dùng, khách hàng có thể thực hiện đăng nhập, đăng ký tài khoản trước khi đặt hàng, có thể thực hiện tìm kiếm sản phẩm, xem chi tiết sản phẩm, xác nhận đặt hàng, xác nhận thanh toán. Đồng thời, người dùng cũng có thể quy đổi các mã giảm giá hoặc điểm tích lũy để có thể giảm giá cho đơn hàng.

Với giao diện dành cho nhân viên, người dùng có thể thực hiện thống kê và báo cáo doanh thu, quản lý các đơn hàng, khách hàng, khuyến mãi, nhân viên, tạo và thanh toán hóa đơn cho khách hàng mua hàng tại quầy.

Trang web hoạt động theo mô hình client-server, sử dụng API để xử lý giao tiếp giữa người dùng và giao diện. Các API được xây dựng theo chuẩn RESTful, với định dạng response, request là JSON.

Trang web được phát triển và đảm bảo chất lượng với việc sử dụng các công nghệ:

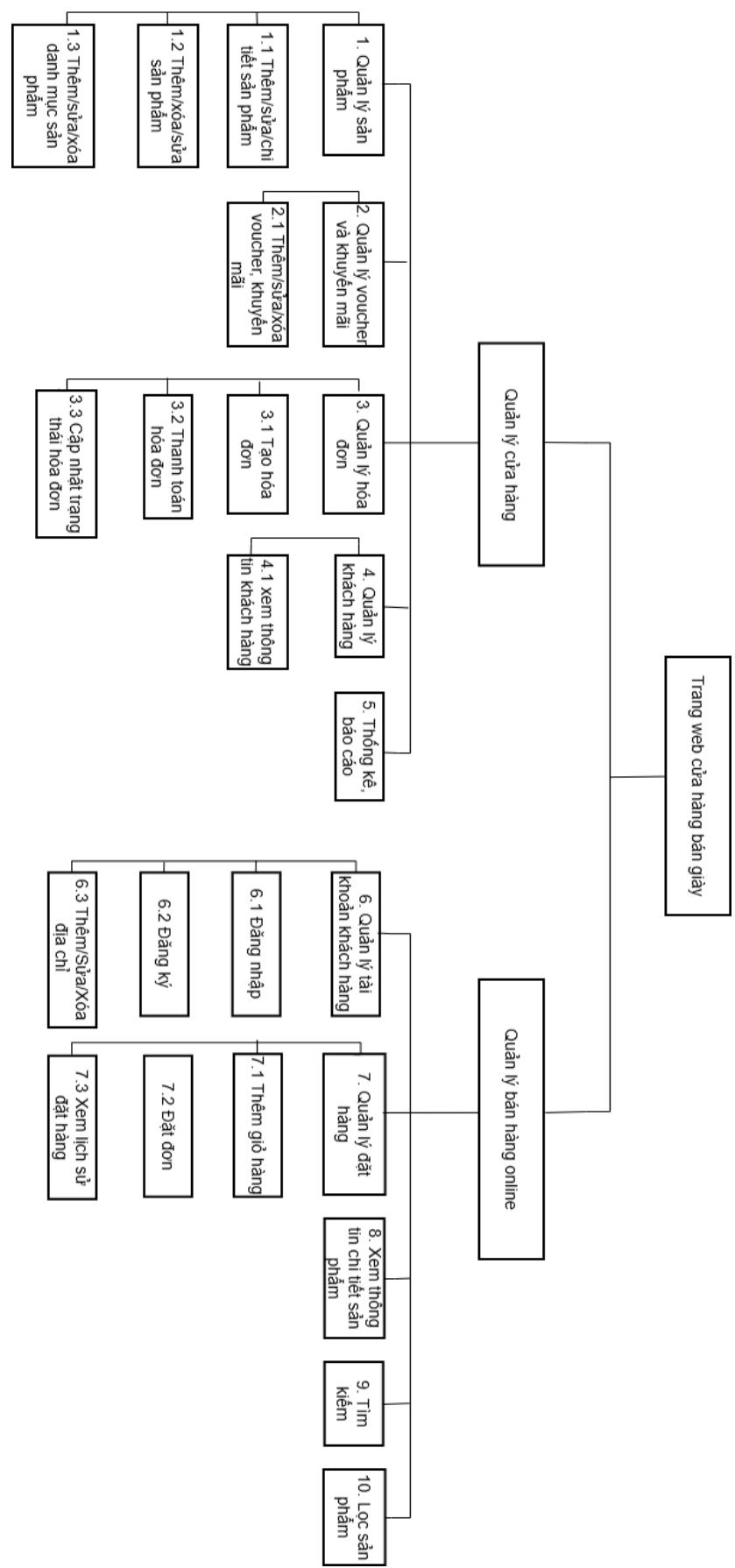
Front-end: HTML/CSS, JavaScripts

Back-end: Typecript

Database: PostgreSQL

Kiểm thử: Kết hợp kiểm thử thủ công và kiểm thử tự động (sử dụng công cụ kiểm thử Selenium Webdriver).

Sơ đồ phân chia chức năng cho hệ thống:



Hình 4.1 Sơ đồ phân cấp chức năng hệ thống web bán giày

CHƯƠNG 5: LẬP KẾ HOẠCH KIỂM THỬ

5.1 Tổng quan

5.1.1 Mục tiêu kiểm thử

Đảm bảo toàn bộ chức năng của trang web bán giày (giành cho User) hoạt động đúng yêu cầu.

Xác minh tính ổn định, khả năng phản hồi và độ bảo mật của hệ thống.

Đảm bảo giao diện người dùng hiển thị chính xác trên các trình duyệt phổ biến (Chrome)

5.1.2 Phạm vi

Phạm vi kiểm thử trang web bao gồm:

Màn hình SR_001 – Chi tiết sản phẩm

Màn hình SR_002 – Giỏ hàng của khách hàng

Màn hình SR_003 – Tài khoản - Thông tin của khách hàng

Màn hình SR_004 – Danh mục - sản phẩm Nữ

Màn hình SR_005 – Thanh toán

5.1.3 Các loại kiểm thử cần thực hiện

Kiểm thử chức năng (Functional Testing): Đảm bảo từng tính năng hoạt động đúng.

Kiểm thử giao diện (UI Testing): Kiểm tra bố cục, hiển thị, tính thân thiện.

Kiểm thử tự động (Automation Testing) áp dụng cho: Chức năng thêm địa chỉ trong màn hình SR_003, Chức năng tìm kiếm

Thực hiện bằng công cụ Selenium WebDriver (Java).

Mục tiêu: giảm thời gian kiểm thử lặp lại và tăng độ chính xác.

5.1.4 Chiến lược kiểm thử

Phương pháp: Kết hợp kiểm thử thủ công (manual) và kiểm thử tự động (automation).

Công cụ:

Selenium WebDriver + Java (Automation)

Google Sheets (Test Case & kết quả)

Jira (Quản lý lỗi)

Dữ liệu kiểm thử: Dữ liệu mẫu được chuẩn bị trước trong cơ sở dữ liệu test.

Môi trường kiểm thử: môi trường staging riêng biệt, có quyền reset dữ liệu.

Phân công công việc:

| Người phụ trách | Nhiệm vụ chính |
|-----------------|---|
| Kim Anh | Kiểm thử manual và automaution trang SR_001, SR_002, SR_003, SR_004, SR_005 |

Bảng 5.1 Phân chia công việc

Lịch trình kiểm thử

| Ngày | Hoạt động | Người phụ trách |
|------|-----------------------------------|-----------------|
| 1-4 | Phân tích yêu cầu, viết test case | Kim Anh |
| 5 | Chuẩn bị dữ liệu và môi trường | Kim Anh |
| 6-7 | Kiểm thử chức năng (User) | Kim Anh |
| 8-9 | Viết và chạy test tự động | Kim Anh |
| 10 | Ghi nhận, xác nhận lỗi fix | Kim Anh |
| 11 | Tổng hợp, lập báo cáo kết quả | Kim Anh |

Bảng 5.2 Bảng lịch trình kiểm thử

5.1.5 Tiêu chí chấp nhận và từ chối

Chấp nhận:

Ít nhất 90% test case đạt.

Không có lỗi nghiêm trọng (Severity 1 hoặc 2).

Các API phản hồi đúng định dạng và trạng thái.

Các test tự động chính đều pass.

Tùy chọn:

Lỗi nghiêm trọng gây gián đoạn chức năng chính (đặt hàng, đăng nhập, thanh toán):

Tỷ lệ pass < 90%.

5.1.6 Rủi ro và biện pháp giảm thiểu rủi ro

Rủi ro 1: Yêu cầu thay đổi trong khi kiểm thử

Ảnh hưởng: Làm trễ tiến độ kiểm thử, test case phải viết lại.

Biện pháp giảm thiểu: Chốt phiên bản yêu cầu trước khi bắt đầu test; mọi thay đổi phải được ghi nhận bằng biên bản và điều chỉnh kế hoạch kịp thời.

Rủi ro 2: Môi trường kiểm thử không ổn định (server hoặc cơ sở dữ liệu bị lỗi)

Ảnh hưởng: Gián đoạn quá trình kiểm thử, mất dữ liệu test.

Biện pháp giảm thiểu: Tạo bản sao (backup) môi trường cục bộ; luôn sao lưu dữ liệu test trước khi chạy kiểm thử.

Rủi ro 3: Thiếu dữ liệu thử nghiệm phù hợp

Ảnh hưởng: Không bao phủ đủ các trường hợp kiểm thử thực tế.

Biện pháp giảm thiểu: Chuẩn bị bộ dữ liệu giả lập đa dạng (nhiều loại sản phẩm, người dùng, đơn hàng) trước khi bắt đầu test.

Rủi ro 4: Lỗi trùng lặp hoặc ghi nhận sai do 2 người kiểm thử song song

Ảnh hưởng: Khó quản lý lỗi, mất thời gian rà soát.

Biện pháp giảm thiểu: Phân chia rõ phạm vi kiểm thử cho từng người. Ghi rõ người tạo lỗi trong file theo dõi (Jira hoặc Google Sheets).

Rủi ro 5: Script kiểm thử tự động bị lỗi hoặc không chạy ổn định

Ảnh hưởng: Kết quả kiểm thử tự động sai lệch, ảnh hưởng độ tin cậy.

Biện pháp giảm thiểu: Kiểm tra lại các script Selenium trước khi chạy; cập nhật định danh (locator) khi giao diện thay đổi; chạy thử nghiệm trên môi trường ổn định.

Rủi ro 6: Thiếu thời gian để bao phủ hết các test case

Ảnh hưởng: Một số chức năng ít quan trọng có thể chưa được kiểm tra đầy đủ.

Biện pháp giảm thiểu: Ưu tiên test các chức năng cốt lõi trước (đăng nhập, đặt hàng, thanh toán); ghi chú các trường hợp chưa test để xử lý sau.

CHƯƠNG 6 PHÂN TÍCH VÀ THIẾT KẾ CA KIỂM THỦ

6.1 Phân tích và thiết kế ca kiểm thử cho các chức năng màn hình

6.1.1 Chức năng của màn hình SR_001

6.1.1.1 Chức năng Thêm sản phẩm vào giỏ hàng

Dựa vào mô tả chi tiết chức năng Thêm sản phẩm vào giỏ hàng trong [đặc tả màn hình SR_001](#) ta có phạm vi kiểm thử

| No. | Item | Type | Required | Default value | Min/Max | Valid format | Descriptions |
|-----|---------------|--------|----------|---------------|---------|--------------|---|
| 19 | [Add to cart] | Button | - | - | Active | - | Click thực hiện action thêm sản phẩm vào giỏ hàng: - Trường hợp thành công: Biến thể màu sắc và biến thể kích thước được chọn, số lượng sản phẩm <= số lượng sản phẩm còn lại cửa hàng - Trường hợp thất bại: + Nếu người dùng chưa chọn biến thể màu sắc hoặc kích thước → Hiển thị alert thông báo lỗi "Please select variant options" + Nếu người dùng nhập số lượng > số lượng sản phẩm còn lại → Hiển thị thông báo lỗi "The quantity entered exceeds the available stock" |

Hình 6.1 Mô tả Button [Add to cart]

Thành phần: Button [Add to cart]

Tính năng: Thêm giỏ hàng

Logic xử lý:

Đăng nhập và click button khi sản phẩm còn hàng

Đăng nhập và click button khi sản phẩm hết hàng

Chưa đăng nhập và click button

Áp dụng kỹ thuật bảng quyết định ta có các ca kiểm thử cho chức năng thêm sản phẩm vào giỏ hàng như dưới đây:

Số lượng chỉ có 2 phân vùng tương đương là <= Số lượng tồn kho và Số lượng > Số lượng tồn kho

| | | TC1 | TC2 | TC3 | TC4 | TC5 | TC6 |
|-----------|---|-----|-----|-----|-----|-----|-----|
| Điều kiện | Màu sắc | Y | - | N | N | Y | Y |
| | Kích thước | Y | - | N | Y | N | Y |
| | Số lượng <= Số lượng hàng tồn | Y | - | - | - | - | N |
| | Đã đăng nhập | Y | N | Y | Y | Y | Y |
| Action | Thêm sản phẩm thành công | X | | | | | |
| | Hiển thị alert thông báo lỗi "Please select variant options" | | | X | X | X | |
| | Hiển thị thông báo lỗi "The quantity entered exceeds the available stock" | | | | | | X |
| | Chuyển đến màn đăng nhập | | X | | | | |

Hình 6.2 Các ca kiểm thử cho chức năng thêm sản phẩm vào giỏ hàng

Dựa vào và phạm vi kiểm thử và kỹ thuật bảng quyết định ta có các ca kiểm thử của button [Add to cart] để kiểm tra UI và Function như dưới đây:

| ID | Test case Description | Test case Procedure | Expected Output | Test Data |
|-------|--|---|---|---------------------------------|
| | GUI | | | |
| PD_16 | [Add to cart] button | | Trạng thái: Cho phép click | |
| PD_17 | Thêm sản phẩm vào giỏ hàng thành công | Precondition: User login vào hệ thống https://demo.evershop.io/ | Mini Cart hiển thị ở góc phải màn hình, chưa có thông tin: • Tiêu đề: "JUST ADDED TO YOUR CART". • Tên sản phẩm chính xác. • Ảnh sản phẩm hiển thị đúng. | |
| | | Trong trang home, kéo xuống mục Featured Products | • Số lượng (QTY) đúng như người dùng chọn. • Hai nút "VIEW CART" và "Continue Shopping" hiển thị. | |
| | | Chọn sản phẩm | | Nike react infinity run flyknit |
| | | Click chọn [Size] button group | "VIEW CART" button: cho phép click | M |
| | | Click chọn [Màu] button group | "Continue Shopping" hyperlink: Cho phép click, đóng mini cart lại, vẫn ở trang hiện tại | Pink |
| | | Nhập [Số lượng] textbox | [X] button: Cho phép click, đóng mini cart lại, vẫn ở trang hiện tại | |
| | | Click [Add to cart] button | Sản phẩm được thêm vào giỏ hàng, Hiển thị mini Cart | |
| PD_18 | Thêm sản phẩm vào giỏ hàng không thành công - Bỏ trống tất cả các trường | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Trong trang home, kéo xuống mục Featured Products | | |
| | | Chọn sản phẩm | Đến trang [Chi tiết sản phẩm] | Nike zoom fly |
| | | Xóa, bỏ trống [Số lượng] | Viền Textbox [Số lượng] chuyển sang màu đỏ và trỏ chuột trả về textbox | |
| | | Click [Add to cart] button | Hiển thị lỗi 'Please select variant options' | |

| ID | Test case Description | Test case Procedure | Expected Output | Test Data |
|-------|--|---|--|---------------|
| FUNC | | | | |
| PD_19 | Thêm sản phẩm vào giỏ hàng không thành công - Chỉ chọn trường [Size] | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Trong trang home, kéo xuống mục Featured Products | | |
| | | Chọn sản phẩm | Đến trang [Chi tiết sản phẩm] | Nike zoom fly |
| | | Click chọn [Size] button group | Nút Size được chọn (hiển thị active). | M |
| | | Click [Add to cart] button | Hiển thị lỗi 'Please select variant options' | |
| PD_20 | Thêm sản phẩm vào giỏ hàng không thành công - Chỉ chọn trường [Màu] | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Trong trang home, kéo xuống mục Featured Products | | |
| | | Chọn sản phẩm | Đến trang [Chi tiết sản phẩm] | Nike zoom fly |
| | | Click chọn [Màu] button group | Nút [Màu] được chọn (hiển thị active). | Blue |
| | | Click [Add to cart] button | Hiển thị lỗi 'Please select variant options' | |
| PD_21 | Thêm sản phẩm không thành công - Đề trống trường [Số lượng] | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Trong trang home, kéo xuống mục Featured Products | | |
| | | Chọn sản phẩm | Đến trang [Chi tiết sản phẩm] | Nike zoom fly |
| | | Click chọn [Size] button group | Nút Size được chọn (hiển thị active). | S |
| | | Click chọn [Màu] button group | Nút [Màu] được chọn (hiển thị active). | Blue |
| | | Xóa, bỏ trống [Số lượng] | Sản phẩm không thêm vào giỏ hàng | |
| | | Click [Add to cart] button | Viền Textbox [Số lượng] chuyển sang màu đỏ và trỏ chuột trở về textbox | |

| ID | Test case Description | Test case Procedure | Expected Output | Test Data |
|-------|---|--|--|---------------------------------|
| FUNC | | | | |
| PD_22 | Thêm sản phẩm không thành công - Trường [Số lượng] không hợp lệ (Dữ liệu chuỗi) | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Trong trang home, kéo xuống mục Featured Products | | |
| | | Chọn sản phẩm | Đến trang [Chi tiết sản phẩm] | Nike zoom fly |
| | | Click chọn [Size] button group | Nút Size được chọn (hiển thị active). | S |
| | | Click chọn [Màu] button group | Nút [Màu] được chọn (hiển thị active). | Blue |
| | | Nhập [Số lượng] | Sản phẩm không thêm vào giỏ hàng | abc, -1, 0 |
| | | Click [Add to cart] button | Hiển thị thông báo lỗi "Qty is invalid" | |
| PD_23 | Thêm sản phẩm không thành công - Trường [Số lượng] lớn hơn số sản phẩm có trong kho | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Trong trang home, kéo xuống mục Featured Products | | |
| | | Chọn sản phẩm | Đến trang [Chi tiết sản phẩm] | Nike react infinity run flyknit |
| | | Click chọn [Size] button group | Nút Size được chọn (hiển thị active). | M |
| | | Click chọn [Màu] button group | Nút [Màu] được chọn (hiển thị active). | Green |
| | | Nhập [Số lượng] | Hiển thị dữ liệu lên textbox | 100 |
| | | Click [Add to cart] button | Sản phẩm không được thêm vào giỏ hàng Hiển thị thông báo "The quantity entered exceeds the available stock" | |
| PD_28 | Chuyển đến trang đăng nhập khi thêm sản phẩm vào giỏ hàng(Trường hợp chưa đăng nhập) | Precondition: User chưa login vào hệ thống https://demo.evershop.io/ | | |
| | | Chọn sản phẩm | Đến trang [Chi tiết sản phẩm] | |
| | | Click chọn [Size] button group | Nút Size được chọn (hiển thị active). | |
| | | Click chọn [Màu] button group | Nút [Màu] được chọn (hiển thị active). | |
| | | Nhập [Số lượng] | Hiển thị số lượng trên textbox | |
| | | Click "Add to cart" | Điều hướng đến trang [Đăng nhập] | |

Hình 6.3 Testcase cho chức năng thêm sản phẩm vào giỏ hàng

Nội dung chi tiết testcase đối với các item khác trên màn hình nhằm kiểm tra về UI, Validate dữ liệu của item, Function khi người dùng thao tác với các item của màn hình SR_001 xem chi tiết tại sheet “SR_001” tại file [EvershopTesting](#)

6.1.2 Chức năng màn hình SR_002

6.1.2.1 Chức năng tăng/giảm số lượng sản phẩm

Áp dụng kỹ thuật phân vùng tương đương có các phạm vi kiểm thử như dưới đây:

| Vùng | Điều kiện đầu vào | Kết quả mong đợi |
|--------------|-------------------|-------------------------------|
| Hợp lệ | Số lượng > 1 | Số lượng giảm 1 |
| Không hợp lệ | Số lượng = 1 | Xóa sản phẩm ra khỏi giỏ hàng |

Bảng 6.1 Bảng phân vùng kiểm thử cho chức năng giảm số lượng

| Vùng | Điều kiện đầu vào | Kết quả mong đợi |
|--------------|--------------------------|------------------|
| Hợp lệ | Số lượng < Số lượng tồn | Số lượng tăng 1 |
| Không hợp lệ | Số lượng >= Số lượng tồn | Hiển thị lỗi |

Bảng 6.2 Bảng phân vùng kiểm thử cho chức năng tăng số lượng

Dựa vào kỹ thuật phân vùng tương đương trên ta có các ca kiểm thử cho các button tăng/giảm số lượng để kiểm tra UI và Function như dưới đây:

| ID | Test case Description | Test case Procedure | Expected Output |
|---------|--|---|---|
| | GUI | | |
| CART_10 | [-] button | | Trạng thái: Cho phép click |
| CART_11 | [+] button | | Trạng thái: Cho phép click |
| FUNC | | | |
| CART_20 | Xóa sản phẩm thành công khi click [Remove] hyperlink(Giỏ hàng có >= 2 sản phẩm) | <p>Precondition: Giỏ hàng có [Product]>=2 - User login vào hệ thống https://demo.evershop.io/</p> <p>Click vào [cart] icon button ở thanh bar, phía bên phải</p> <p>Trong hàng đầu của [Cart] table, Click [Remove] hyperlink trong cột [Product]</p> | Điều hướng đến [Giỏ hàng], có > = 2 sản phẩm Sản phẩm được chọn bị xóa ra khỏi giỏ hàng Sản phẩm được xóa ra khỏi CSDL |
| CART_21 | Xóa sản phẩm thành công khi click [Remove] hyperlink(Giỏ hàng có = 1 sản phẩm) | <p>Precondition: Giỏ hàng có [Product]=1 - User login vào hệ thống https://demo.evershop.io/</p> <p>Click vào [cart] icon button ở thanh bar, phía bên phải</p> <p>Trong hàng đầu của [Cart] table, Click [Remove] hyperlink trong cột [Product]</p> | Điều hướng đến [Giỏ hàng], Giỏ hàng có 1 sản phẩm Giao diện hiển thị "Shopping cart Your cart is empty!" và [Continue Shopping] button |
| CART_22 | Xóa sản phẩm thành công khi click [-] button (Giỏ hàng có = 1 sản phẩm, Quantity input = 1) | <p>Precondition: - User đã đăng nhập vào hệ thống. - Giỏ hàng có đúng 1 sản phẩm (ProductItem = 1). - Ô nhập số lượng (Quantity input) của sản phẩm = 1. - Trang web: https://demo.evershop.io/</p> <p>Click vào [cart] icon button ở thanh bar, phía bên phải</p> <p>Trong hàng đầu của [Cart] table, Click [-] hyperlink trong cột [Quantity]</p> | Giỏ hàng hiển thị với 1 sản phẩm trong bảng [Cart]. "Giao diện hiển thị ""Shopping cart Your cart is empty!"" và [Continue Shopping] button" |

| ID | Test case Description | Test case Procedure | Expected Output |
|---------|--|---|--|
| | FUNC | | |
| CART_23 | Xóa sản phẩm thành công khi click [-] button(Giỏ hàng có >= 2 sản phẩm và Quantity input = 1) | <p>Precondition:- User đã đăng nhập vào hệ thống. - Giỏ hàng có tối thiểu 2 sản phẩm ([ProductItem]≥ 2). - Sản phẩm ở hàng đầu tiên trong bảng [Cart] có Quantity input = 1. - Trang web: https://demo.evershop.io/</p> <p>Click vào [cart] icon button ở thanh bar, phía bên phải</p> <p>Trong hàng sản phẩm đầu tiên của [Cart] table, Click [-] hyperlink trong cột [Quantity]</p> | Điều hướng đến [Giỏ hàng], có > = 2 sản phẩm Sản phẩm được chọn bị xóa ra khỏi giỏ hàng - Số lượng sản phẩm trong giỏ giảm đi 1. - Các sản phẩm còn lại vẫn giữ nguyên thông tin. - Không xuất hiện thông báo lỗi. - Tổng giá trị đơn hàng được cập nhật chính xác. |
| CART_24 | Giảm số lượng sản phẩm thành công khi click [-] button | <p>Precondition: Giỏ hàng có [ProducItemt]>=1, Số lượng sản phẩm >1 - User login vào hệ thống https://demo.evershop.io/</p> <p>Click vào [cart] icon button ở thanh bar, phía bên phải</p> <p>Chọn 1 sản phẩm trong [Cart] table, click [-] button</p> | Điều hướng đến [Giỏ hàng], có > = 1 sản phẩm Số lượng sản phẩm giảm lên mỗi lần click vào [-] button [Total] = [price]*[Quantity] [Sub Total] = [price]*[Quantity] |
| CART_25 | Thêm số lượng sản phẩm thành công khi click [+] button | <p>Precondition: Giỏ hàng có [ProducItemt]>=1 - User login vào hệ thống https://demo.evershop.io/</p> <p>Click vào [cart] icon button ở thanh bar, phía bên phải</p> <p>Chọn 1 sản phẩm trong [Cart] table, click [+] button</p> | Điều hướng đến [Giỏ hàng], có > = 1 sản phẩm Số lượng sản phẩm tăng lên mỗi lần click vào [+] button [Total] = [price]*[Quantity] [Sub Total] = [price]*[Quantity] |
| CART_26 | Thêm số lượng sản phẩm không thành công khi click [+] button (Số lượng sản phẩm > số lượng sản phẩm trong kho) | <p>Precondition: Giỏ hàng có [ProducItemt]>=1 - User login vào hệ thống https://demo.evershop.io/</p> <p>Click vào [cart] icon button ở thanh bar, phía bên phải</p> <p>Chon 1 sản phẩm trong [Cart] table, click [+] button</p> | Điều hướng đến [Giỏ hàng], có > = 1 sản phẩm Hiển thị thông báo "We do not have enough stock" dưới tên sản phẩm Không cho chuyển đến trang thanh toán |

Hình 6.4 Testcase các button tăng/ giảm số lượng sản phẩm

6.1.2.2 Chuyển đến màn thanh toán

Áp dụng kỹ thuật phân vùng tương đương có các bảng phân vùng kiểm thử như dưới đây:

| Vùng | Điều kiện đầu vào | Kết quả mong đợi |
|--------------|----------------------|---|
| Hợp lệ | Giỏ hàng có sản phẩm | Chuyển đến màn hình thanh toán |
| Không hợp lệ | Giỏ hàng trống | Hiển thị giao diện giỏ hàng trống (Như trong đặc tả sheet SR_002) |

Bảng 6.3 Bảng phân vùng kiểm thử cho chức năng điều hướng đến màn hình thanh toán

Từ đó ta thiết kế được các ca kiểm thử:

| ID | Test case Description | Test case Procedure | Expected Output |
|---------|---|---|--|
| GUI | | | |
| CART_13 | Cart table | | Gồm có 4 cột: - Product: Hiển thị tên ảnh, tên sản phẩm, màu sắc, kích cỡ đã được thêm vào giỏ hàng và [Remove] hyperlink - Price: Hiển thị giá tiền của 1 sản phẩm (bao gồm thuế) đã được thêm vào giỏ hàng Quantity: Hiển thị Quantity Selector $Total = [Price]*[Quantity]$ Tên sản phẩm: Có thể click Các sản phẩm trong giỏ hàng được hiển thị theo thứ tự thời gian thêm vào, trong đó sản phẩm được thêm gần nhất sẽ xuất hiện ở đầu danh sách (top of the list). |
| CART_17 | [Continue shopping] button | Precondition: Giỏ hàng trống | Trạng thái: Cho phép click |
| FUNC | | | |
| CART_18 | Điều hướng đến trang [home] thành công khi click [Continue shopping] button | Precondition: Giỏ hàng trống - User login vào hệ thống https://demo.evershop.io/ | |
| | | Click vào [cart] icon button ở thanh bar, phía bên phải | Điều hướng đến [Giỏ hàng] |
| | | Click vào [Continue shopping] button | Điều hướng đến trang [Home] |
| CART_19 | Điều hướng đến trang thanh toán thành công | Precondition: Giỏ hàng có [Product]>=1 - User login vào hệ thống https://demo.evershop.io/ | |
| | | Click vào [cart] icon button ở thanh bar, phía bên phải | Điều hướng đến [Giỏ hàng] |
| | | Click vào [Checkout] button | Điều hướng đến trang thanh toán |

Hình 6.5 Testcase cho chức năng điều hướng đến màn hình thanh toán

Nội dung chi tiết testcase đối với các item khác trên màn hình nhằm kiểm tra về UI, Validate dữ liệu của item, Function khi người dùng thao tác với các item của màn hình SR_002 xem chi tiết tại sheet “SR_002” tại file [EvershopTesting](#)

6.1.3 Chức năng màn hình SR_003

6.1.3.1 Chức năng xóa địa chỉ và đặt địa chỉ làm địa chỉ mặc định

Dựa vào mô tả chi tiết chức năng xóa địa chỉ trong [đặc tả màn hình SR_003](#)

| No. | Item | Type | Required | Default value | Min/Max | Valid format | Descriptions |
|-----|--------------|------|----------|---------------|---------|--------------|--|
| 6 | Address book | Text | - | - | - | - | Hiển thị các địa chỉ |
| 15 | Edit | Link | - | Active | - | - | Hiển thị popup Edit Address, cho phép người dùng sửa địa chỉ |
| 16 | Make default | Link | - | Active | - | - | Đặt địa chỉ làm địa chỉ mặc định |

Hình 6.6 Mô tả chi tiết chức năng xóa địa chỉ

Ta thiết kế được các ca kiểm thử:

| ID | Test case Description | Test case Procedure | Expected Output |
|--------|---|--|---|
| FUNC | | | |
| ACC_41 | Xóa địa chỉ thành công | Precondition: User login vào hệ thống https://demo.evershop.io/ Có ≥ 1 địa chỉ trong Address Book. Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang Tài khoản - Thông tin |
| | | Trong [Address Book] section, Click [delete] hyperlink | Hiển thị popup đến người dùng: "Do you want to delete this address?" |
| | | Click button {Yes} | Button {Yes} {No} cho phép click |
| | | | Xóa Address khỏi [Address Book] section, Hiển thị thông báo xóa |
| ACC_42 | Hủy xóa địa chỉ | Precondition: User login vào hệ thống https://demo.evershop.io/ Có ≥ 1 địa chỉ trong Address Book. Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang Tài khoản - Thông tin |
| | | Trong [Address Book] section, Click [delete] hyperlink | Hiển thị popup đến người dùng: "Do you want to delete this address?" |
| | | Click button {No} | Button {Yes} {No} cho phép click |
| | | | Close popup |
| ACC_43 | Cho địa chỉ làm địa chỉ mặc định thành công | Precondition: User login vào hệ thống https://demo.evershop.io/ Có ≥ 1 địa chỉ trong Address Book. Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang Tài khoản - Thông tin |
| | | Trên một address card chưa mặc định, click [Make default]. | Link [Make default] không còn xuất hiện trên card mặc định. |
| | | (Kiểm chứng) Mở giò hàng → click [Checkout] → kiểm tra các trường địa chỉ trên form checkout. | Các card còn lại vẫn có link [Make default]. Address card chuyển sang viền xanh Hiển thị thông báo: "Address has been set as default" |
| | | | Các trường địa chỉ (Full name, Telephone, Address, City, Country/Province, Postcode) tự động điền bằng address mặc định mới. |

Hình 6.7 Testcase cho chức năng xóa và đặt địa chỉ mặc định

6.1.3.2 Chức năng thêm địa chỉ

Dựa vào kĩ thuật phân vùng tương đương:

| | C1 Full name | C2 Telephone | C3 Address | C4 City | C5 Postcode | Output |
|----|--|-------------------------|--|---|---|---------|
| T | Nguyễn Văn A | 0988956088 | 1600 Pennsylvania Ave NW | Washington | 20500 | Valid |
| C1 | "" "Nguyen Thi Kim Mai Hoang Phuong Thao Tran Minh Chau" | 0988956087 | 1601 Pennsylvania Ave NW | Washington | 43001 | Invalid |
| C2 | Anna | 1234567 012343211234 | 12 Rue Didouche Mourad | Alger | 16000 | Invalid |
| C3 | Ming | 0123211412 | "" "12 Rue Didouche Mourad, Bab El Oued, Wilaya d'Alger, Apartment 101, Building B, Near Central Market, Landmark: Notre-Dame d'Afrique, Additional info: Floor 5, Door 12" | Shanghai | 200000 | Invalid |
| C4 | Ming | 0123211412 | 88 Nanjing Road | "" "A Very Long City Name That Exceeds Fifty Characters For Testing Purposes" | 200000 | Invalid |
| C5 | An | 0345122088 | 29 MG Road | Bengaluru | "" "560001-EXTRA- POSTCODE-TO- EXCEED-LIMIT- 12345" | Invalid |

Hình 6.8 Kỹ thuật phân vùng tương đương cho Chức năng thêm địa chỉ

Từ bảng trên ta xây dựng, thiết kế được các ca kiểm thử:

| ID | Test case Description | Test case Procedure | Expected Output | Test Data |
|--------|---|--|---|--------------------------|
| FUNC | | | | |
| ACC_48 | Thêm địa chỉ thành công | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang [Tài khoản - Thông tin] | |
| | | Trong section [Address book], click hyperlink [Add new address] | Hiển thị popup [Add new address] | |
| | | Nhập [Full name] | | Nguyễn Văn A |
| | | Nhập [Telephone] | | 0988956088 |
| | | Nhập [Address] | | 1600 Pennsylvania Ave NW |
| | | Nhập [City] | | Washington |
| | | Chọn [Country] | Dữ liệu được chọn/nhập thành công | United States |
| | | Chọn[Province] | | Washington |
| | | Nhập [Postcode] | Hiển thị dữ liệu vừa nhập vào textbox | 20500 |
| | | Click [Save] button | Hiển thị thông báo "Address has been saved successfully" | |
| ACC_49 | Thêm địa chỉ không thành công, bỏ trống tất cả các trường | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang [Tài khoản - Thông tin] | |
| | | Trong section [Address book], click hyperlink [Add new address] | Hiển thị popup [Add new address] | |
| | | Click [Save] button | Hệ thống hiển thị thông báo lỗi đó dưới mỗi trường: <ul style="list-style-type: none"> • Full name is required • Telephone is required • Address is required <ul style="list-style-type: none"> • City is required • Country is required • Postcode is required | |
| | | | - Dữ liệu không được lưu, form vẫn giữ nguyên trạng thái | |

| | | | | |
|--------|---|---|---|--------------------------|
| | | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang [Tài khoản - Thông tin] | |
| | | Trong section [Address book], click hyperlink [Add new address] | Hiển thị [Add new address] popup | |
| | | Nhập [Telephone] | | 0988956087 |
| | | Nhập [Address] | Hiển thị dữ liệu vừa nhập vào textbox | 1601 Pennsylvania Ave NW |
| | | Nhập [City] | | Washington |
| | | Chọn [Country] | Dữ liệu được chọn/nhập thành công | United States |
| | | Chọn[Province] | | Ohio |
| | | Nhập [Postcode] | Hiển thị dữ liệu vừa nhập vào textbox | 43001 |
| | | | Không thêm địa chỉ mới, hiển thị thông báo lỗi hợp lệ. | |
| | | Click [Save] button | Hiển thị thông báo lỗi "Full name is required" dưới [Full name] textbox | |
| ACC_50 | Thêm địa chỉ không thành công, không nhập [Full name] | | | |

| ID | Test case Description | Test case Procedure | Expected Output | Test Data |
|--------|--|---|---|---------------------------------|
| FUNC | | | | |
| ACC_51 | Thêm địa chỉ không thành công, nhập [Full name] lớn hơn 50 kí tự | | | |
| | | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang [Tài khoản - Thông tin] | |
| | | Trong section [Address book], click hyperlink [Add new address] | Hiển thị [Add new address] popup | |
| | | Nhập [Full name] | | Nguyen Thi Kim Mai Hoang Phuong |
| | | Nhập [Telephone] | Hiển thị dữ liệu vừa nhập vào textbox | 0345122099 |
| | | Nhập [Address] | | 12 Rue Didouche Mourad |
| | | Nhập [City] | | Algiers |
| | | Chọn [Country] | Dữ liệu được chọn/nhập thành công | Algeria |
| | | Chọn[Province] | | Algeria |
| | | Nhập [Postcode] | Hiển thị dữ liệu vừa nhập vào textbox | 16000 |
| | | | Không thêm địa chỉ mới, hiển thị thông báo lỗi hợp lệ. | |
| | | Click [Save] button | Hiển thị thông báo lỗi "Full name is over 50 chars" dưới [Full name] textbox | |
| ACC_52 | Thêm địa chỉ không thành công, nhập [Telephone] < 8 digits | | | |
| | | Precondition: User login vào hệ thống | | |
| | | Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang [Tài khoản - Thông tin] | |
| | | Trong section [Address book], click hyperlink [Add new address] | Hiển thị [Add new address] popup | |
| | | Nhập [Full name] | | Anna |
| | | Nhập [Telephone] | Hiển thị dữ liệu vừa nhập vào textbox | 1234567 |
| | | Nhập [Address] | | 12 Rue Didouche Mourad |
| | | Nhập [City] | | Alger |
| | | Chọn [Country] | Dữ liệu được chọn/nhập thành công | Algeria |
| | | Chọn[Province] | | Alger |
| | | Nhập [Postcode] | Hiển thị dữ liệu vừa nhập vào textbox | 16000 |
| | | | Không thêm địa chỉ mới, hiển thị thông báo lỗi hợp lệ. | |
| | | Click [Save] button | Hiển thị thông báo lỗi "Telephone is lower" Trường Telephone được đánh dấu lỗi viền đỏ và focus | |
| ACC_53 | Thêm địa chỉ không thành công, nhập [Telephone] > 11 digits | | | |
| | | Precondition: User login vào hệ thống | | |
| | | Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang [Tài khoản - Thông tin] | |
| | | Trong section [Address book], click hyperlink [Add new address] | Hiển thị [Add new address] popup | |
| | | Nhập [Full name] | | Anna Han |
| | | Nhập [Telephone] | Hiển thị dữ liệu vừa nhập vào textbox | 012343211234 |
| | | Nhập [Address] | | 45 Gangnam-daero |
| | | Nhập [City] | | Seoul |
| | | Chọn [Country] | Dữ liệu được chọn/nhập thành công | South Korea |
| | | Chọn[Province] | | Seoul-teukbyeolsi |
| | | Nhập [Postcode] | Hiển thị dữ liệu vừa nhập vào textbox | 6030 |
| | | | Không thêm địa chỉ mới, hiển thị thông báo lỗi hợp lệ. | |
| | | Click [Save] button | Hiển thị thông báo lỗi "Telephone is greater than 11 digits" dưới [Telephone] textbox | |
| | | | Trường Telephone được đánh dấu lỗi viền đỏ và focus | |

| | | | | |
|--------|---|---|---|-----------------|
| | | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| ACC_54 | Thêm địa chỉ không thành công, không nhập [Address] | Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang [Tài khoản - Thông tin] | |
| | | Trong section [Address book], click hyperlink [Add new address] | Hiển thị [Add new address] popup | |
| | | Nhập [Full name] | | Ming |
| | | Nhập [Telephone] | Hiển thị dữ liệu vừa nhập vào textbox | 0123211412 |
| | | Nhập [City] | | Shanghai |
| | | Chọn [Country] | Dữ liệu được chọn/nhập thành công | China |
| | | Chọn[Province] | | Shanghai |
| | | Nhập [Postcode] | Hiển thị dữ liệu vừa nhập vào textbox | 200000 |
| | | | Không thêm địa chỉ mới, hiển thị thông báo lỗi hợp lệ. | |
| | | Click [Save] button | Hiển thị thông báo lỗi "Address is required" dưới [Address] textbox | |
| | | | Trường Address được đánh dấu lỗi viền đỏ và focus | |
| ACC_55 | Thêm địa chỉ không thành công, không nhập [City] | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang [Tài khoản - Thông tin] | |
| | | Trong section [Address book], click hyperlink [Add new address] | Hiển thị [Add new address] popup | |
| | | Nhập [Full name] | | Ming |
| | | Nhập [Telephone] | Hiển thị dữ liệu vừa nhập vào textbox | 0123211412 |
| | | Nhập [Address] | | 88 Nanjing Road |
| | | Chọn [Country] | Dữ liệu được chọn/nhập thành công | China |
| | | Chọn[Province] | | Shanghai |
| | | Nhập [Postcode] | Hiển thị dữ liệu vừa nhập vào textbox | 200000 |
| | | | Không thêm địa chỉ mới, hiển thị thông báo lỗi hợp lệ. | |
| | | Click [Save] button | Hiển thị thông báo lỗi "City is required" dưới [City] textbox | |
| | | | Trường Address được đánh dấu lỗi viền đỏ và focus | |

| | | | | |
|--------|---|--|--|--|
| ACC_56 | Thêm địa chỉ không thành công, không nhập [Postcode] | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang [Tài khoản - Thông tin] | |
| | | Trong section [Address book], click hyperlink [Add new address] | Hiển thị [Add new address] popup | |
| | | Nhập [Full name] | | Ming |
| | | Nhập [Telephone] | | 0123211412 |
| | | Nhập [Address] | | 88 Nanjing Road |
| | | Nhập [City] | | Shanghai |
| | | Chọn [Country] | | China |
| | | Chọn [Province] | | Shanghai |
| | | Click [Save] button | Không thêm địa chỉ mới, hiển thị thông báo lỗi hợp lệ. Hiển thị thông báo lỗi "Postcode is required" dưới [Postcode] textbox Trường Address được đánh dấu lỗi viền đỏ và focus | |
| ACC_57 | Thêm địa chỉ không thành công, nhập [Address] lớn hơn 100 kí tự | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang [Tài khoản - Thông tin] | |
| | | Trong section [Address book], click hyperlink [Add new address] | Hiển thị [Add new address] popup | |
| | | Nhập [Full name] | | An |
| | | Nhập [Telephone] | | 0345122088 |
| | | Nhập [Address] | Hiển thị dữ liệu vừa nhập vào textbox | 12 Rue Didouche Mourad, Bab El Oued, Wilaya d'Alger, Apartment 101, Building B, Near Central Market, Landmark: Notre-Dame d'Afrique, Additional info: Floor 5, Door 12 |
| | | Nhập [City] | | Algiers |
| | | Chọn [Country] | | Algeria |
| | | Chọn [Province] | | Algeria |
| | | Nhập [Postcode] | Hiển thị dữ liệu vừa nhập vào textbox | 16000 |
| ACC_58 | Thêm địa chỉ không thành công, nhập [City] lớn hơn 50 kí tự | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang [Tài khoản - Thông tin] | |
| | | Trong section [Address book], click hyperlink [Add new address] | Hiển thị [Add new address] popup | |
| | | Nhập [Full name] | | An |
| | | Nhập [Telephone] | | 0345122088 |
| | | Nhập [Address] | Hiển thị dữ liệu vừa nhập vào textbox | 24 MG Road |
| | | Nhập [City] | | A Very Long City Name That Exceeds Fifty Characters For Testing Purposes |
| | | Chọn [Country] | | India |
| | | Chọn [Province] | | Karnataka |
| | | Nhập [Postcode] | Hiển thị dữ liệu vừa nhập vào textbox | 560001 |
| ACC_59 | Thêm địa chỉ không thành công, nhập [Postcode] lớn hơn 20 kí tự | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang [Tài khoản - Thông tin] | |
| | | Trong section [Address book], click hyperlink [Add new address] | Hiển thị [Add new address] popup | |
| | | Nhập [Full name] | | An |
| | | Nhập [Telephone] | | 0345122088 |
| | | Nhập [Address] | Hiển thị dữ liệu vừa nhập vào textbox | 29 MG Road |
| | | Nhập [City] | | Bengaluru |
| | | Chọn [Country] | | India |
| | | Chọn [Province] | | Karnataka |
| | | Nhập [Postcode] | Hiển thị dữ liệu vừa nhập vào textbox | 560001-EXTRA-POSTCODE-TO-EXCEED-LIMIT-12345 |
| | | Click [Save] button | Không thêm địa chỉ mới, hiển thị thông báo lỗi hợp lệ. Trường Address được đánh dấu lỗi viền đỏ và focus Hiển thị thông báo lỗi "Postcode is over 20 chars" | |

| ID | Test case Description | Test case Procedure | Expected Output | Test Data |
|--------|---|--|---|--|
| FUNC | | | | |
| ACC_58 | Thêm địa chỉ không thành công, nhập [City] lớn hơn 50 kí tự | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang [Tài khoản - Thông tin] | |
| | | Trong section [Address book], click hyperlink [Add new address] | Hiển thị [Add new address] popup | |
| | | Nhập [Full name] | | An |
| | | Nhập [Telephone] | | 0345122088 |
| | | Nhập [Address] | Hiển thị dữ liệu vừa nhập vào textbox | 24 MG Road |
| | | Nhập [City] | | A Very Long City Name That Exceeds Fifty Characters For Testing Purposes |
| | | Chọn [Country] | | India |
| | | Chọn [Province] | | Karnataka |
| | | Nhập [Postcode] | Hiển thị dữ liệu vừa nhập vào textbox | 560001 |
| ACC_59 | Thêm địa chỉ không thành công, nhập [Postcode] lớn hơn 20 kí tự | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang [Tài khoản - Thông tin] | |
| | | Trong section [Address book], click hyperlink [Add new address] | Hiển thị [Add new address] popup | |
| | | Nhập [Full name] | | An |
| | | Nhập [Telephone] | | 0345122088 |
| | | Nhập [Address] | Hiển thị dữ liệu vừa nhập vào textbox | 29 MG Road |
| | | Nhập [City] | | Bengaluru |
| | | Chọn [Country] | | India |
| | | Chọn [Province] | | Karnataka |
| | | Nhập [Postcode] | Hiển thị dữ liệu vừa nhập vào textbox | 560001-EXTRA-POSTCODE-TO-EXCEED-LIMIT-12345 |
| | | Click [Save] button | Không thêm địa chỉ mới, hiển thị thông báo lỗi hợp lệ. Trường Address được đánh dấu lỗi viền đỏ và focus Hiển thị thông báo lỗi "Postcode is over 20 chars" | |

| ID | Test case Description | Test case Procedure | Expected Output | Test Data |
|--------|--|--|--|------------------------------|
| FUNC | | | | |
| ACC_60 | Thêm địa chỉ không thành công, nhập [Telephone] không hợp lệ | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Click vào [user] icon button ở thanh bar, phia bên phải | Điều hướng đến trang [Tài khoản - Thông tin] | |
| | | Trong section [Address book], click hyperlink [Add new address] | Hiển thị [Add new address] popup | |
| | | Nhập [Full name] | | Kim Hoa |
| | | Nhập [Telephone] | | abc03422113 hoặc 083218337() |
| | | Nhập [Address] | | 12 MG Road |
| | | Nhập [City] | | Bengaluru |
| | | Chọn [Country] | Dữ liệu được chọn/nhập thành công | India |
| | | Chọn[Province] | | Karnataka |
| | | Nhập [Postcode] | Hiển thị dữ liệu vừa nhập vào textbox | 560001 |
| | | | Không thêm địa chỉ mới, hiển thị thông báo lỗi hợp lệ. | |
| | | Click [Save] button | Trường Address được đánh dấu lỗi viền đỏ và focus | |
| | | | Hiển thị thông báo lỗi "Telephone number is invalid" dưới [Postcode] textbox | |
| ACC_61 | Thêm địa chỉ không thành công, nhập dấu cách | Precondition: User login vào hệ thống https://demo.evershop.io/ | | |
| | | Click vào [user] icon button ở thanh bar, phia bên phải | Điều hướng đến trang [Tài khoản - Thông tin] | |
| | | Trong section [Address book], click hyperlink [Add new address] | Hiển thị [Add new address] popup | |
| | | Nhập space vào [Full name] | | |
| | | Nhập space vào [Telephone] | | |
| | | Nhập space vào [Address] | Field hiển thị khoảng trắng. | |
| | | Nhập space vào [City] | | |
| | | Chọn [Country] | Dữ liệu được chọn/nhập thành công | China |
| | | Chọn[Province] | | Hunan |
| | | Nhập space vào [Postcode] | Field hiển thị khoảng trắng. | |
| | | | Form không được lưu. | |
| | | Click [Save] button | - Hiển thị lỗi cho tất cả các field trống như: "Full name is required.", "Telephone is required.", "Address is required.", "City is required.", "Postcode is required." - Popup vẫn mở. - Không thêm địa chỉ mới trong Address Book. | |

Hình 6.9 Testcase cho Chức năng thêm địa chỉ

Nội dung chi tiết testcase đối với các item khác trên màn hình nhằm kiểm tra về UI, Validate dữ liệu của item, Function khi người dùng thao tác với các item của màn hình SR_003 xem chi tiết tại sheet “SR_003” tại file [EvershopTesting](#)

6.1.4 Chức năng màn hình SR_004

6.1.4.1 Chức năng lọc sản phẩm

Dựa vào mô tả chi tiết chức năng lọc sản phẩm theo các biến thể: Giá cả, màu sắc, kích thước, Brand trong [đặc tả màn hình SR_004](#)

| No. | Item | Type | Required | Default value | Min/Max | Valid format | Descriptions |
|-----|--------------------------|--------------|----------|---------------|--------------------------|--------------|---|
| 1 | Breadcrumb: Home / Women | Link | ▼ | - | - | - | - Hiển thị đường dẫn: Home / Women - Click "Home" trong Breadcrumb thực hiện action trả về màn hình [Home] |
| 2 | Tên màn hình | Text | ▼ | - | - | - | Hiển thị title "Women" |
| | Left bar | | ▼ | - | - | - | |
| 3 | Price slider | Range slider | ▼ | - | Min: 133\$ Max: 963\$ | - | Lọc sản phẩm theo khoảng giá |
| 4 | Biến thể [Size] | Check box | ▼ | - | - | - | Cho phép Checked các biến thể kích thước để lọc sản phẩm |
| 5 | Biến thể [Color] | Check box | ▼ | - | - | - | Cho phép Checked các biến thể màu sắc để lọc sản phẩm |
| 6 | Biến thể [Brand] | Check box | ▼ | - | | | Cho phép Checked các biến thể Brand để lọc sản phẩm |
| 7 | Sort by | Select box | ▼ | - | Default | - | Chọn lọc theo giá/ theo tên |
| | Default | Select box | ▼ | - | - | - | Lọc theo thứ tự mặc định của hệ thống |
| | Price | Select box | ▼ | - | - | - | Chọn lọc theo giá |
| | Name | Select box | ▼ | - | - | - | Chọn lọc theo tên |
| 8 | ↓/↑ | Button | ▼ | - | - | - | Sắp xếp ↑ tăng dần, sắp xếp ↓ giảm dần |
| 9 | Hình ảnh sản phẩm | Image | ▼ | - | | - | -Hiển thị hình ảnh sản phẩm - Click vào dẫn đến trang SR_001 |
| 10 | Tên sản phẩm | Link | ▼ | - | - | - | -Hiển thị tên sản phẩm - Click vào dẫn đến trang SR_001 |
| 11 | Giá tiền của sản phẩm | Text | ▼ | - | - | - | Hiển thị giá tiền của sản phẩm |

Hình 6.10. Mô tả chi tiết chức năng lọc sản phẩm theo các biến thể

Từ bảng đặc tả trên, ta thiết kế các ca kiểm thử:

| ID | Test case Description | Test case Procedure | Expected Output |
|--------|--|--|--|
| CAT_20 | Điều hướng đến trang Chi tiết sản phẩm khi click vào Thumbnail | Trong trang [home] hover vào [Shop] dropdown, click vào Women | Hiển thị trang Danh mục sản phẩm nữ |
| | | Trong Product Grid, chọn 1 sản phẩm và click vào Thumbnail của sản phẩm đó | Điều hướng đến trang Chi tiết sản phẩm |
| CAT_21 | Điều hướng đến trang Chi tiết sản phẩm khi click vào Product Name | Trong trang [home] hover vào [Shop] dropdown, click vào Women | Hiển thị trang Danh mục sản phẩm nữ |
| | | Trong Product Grid, chọn 1 sản phẩm và click vào tên của sản phẩm đó | Điều hướng đến trang Chi tiết sản phẩm |
| CAT_22 | Sắp xếp sản phẩm giảm dần theo giá tiền | Tại trang Home, di chuột (hover) vào menu Shop và chọn mục Women. | |
| | | Trong phần Sort By, click vào dropdown và chọn Price. | icon chuyển thành ↓ icon |
| | | Click vào mũi tên hướng lên (↑) bên phải ô Sort. | Sản phẩm được sắp xếp theo chiều giảm dần của giá tiền |
| CAT_23 | Sắp xếp sản phẩm tăng dần theo giá tiền | Tại trang Home, di chuột (hover) vào menu Shop và chọn mục Women. | |
| | | Trong phần Sort By, click vào dropdown và chọn Price. | icon chuyển thành ↑ |
| | | Click vào mũi tên hướng xuống (↓) bên phải ô Sort. | Sản phẩm được sắp xếp theo chiều tăng dần của giá tiền |
| CAT_24 | Sắp xếp sản phẩm tăng dần theo tên (A-Z) | Tại trang Home, di chuột (hover) vào menu Shop và chọn mục Women. | |
| | | Trong phần Sort By, click vào dropdown và chọn Name. | icon chuyển thành ↑ |
| | | Click vào mũi tên hướng xuống (↓) bên phải ô Sort. | Sản phẩm được sắp xếp theo chiều tăng dần theo tên (A-Z) |
| CAT_25 | Sắp xếp sản phẩm giảm dần theo tên (A-Z) | Tại trang Home, di chuột (hover) vào menu Shop và chọn mục Women. | |
| | | Trong phần Sort By, click vào dropdown và chọn Name. | |
| | | Click vào mũi tên hướng xuống (↑) bên phải ô Sort. | icon chuyển thành ↓ Sản phẩm được sắp xếp theo chiều giảm dần theo tên (Z-A) |
| CAT_26 | Hiển thị sản phẩm có giá trị min khi kéo thanh trượt bên phải về mức thấp nhất | Tại trang Home, di chuột (hover) vào menu Shop và chọn mục Women. | Hiển thị trang [danh mục sản phẩm] |
| | | Cuộn đến phần bộ lọc “PRICE” | |
| | | Kéo thanh trượt bên phải (max handle) hết về bên trái | Giá trị min hiển thị đúng giá nhỏ nhất |
| | | Quan sát giá trị hiển thị bên dưới thanh trượt | Danh sách sản phẩm hiển thị tương ứng với khoảng giá mới |
| CAT_27 | Hiển thị sản phẩm có giá trị max khi kéo thanh trượt bên trái về mức cao nhất | Tại trang Home, di chuột (hover) vào menu Shop và chọn mục Women. | Hiển thị trang [danh mục sản phẩm] |
| | | Kéo thanh trượt bên trái (min handle) hết về bên phải | Giá trị max hiển thị đúng giá lớn nhất |
| | | Quan sát giá trị hiển thị bên dưới thanh trượt | Danh sách sản phẩm hiển thị tương ứng với khoảng giá mới |
| CAT_28 | Hiển thị sản phẩm trong khoảng giá chọn | Tại trang Home, di chuột (hover) vào menu Shop và chọn mục Women. | |
| | | Kéo thanh trượt trái đến \$200.00 | |
| | | Kéo thanh trượt phải đến \$800.00 | |
| | | Quan sát danh sách sản phẩm | - Chỉ hiển thị sản phẩm có giá trong khoảng \$200-\$800 - Không có sản phẩm nằm ngoài khoảng - Giá trị hiển thị dưới thanh trượt cập nhật đúng |

| | | | |
|--------|--|--|---|
| CAT_29 | Kiểm tra hiển thị mặc định khi không chọn bất kỳ bộ lọc nào | Tại trang Home, di chuột (hover) vào menu Shop và chọn mục Women. | |
| | | Không tích chọn bất kỳ mục nào trong các bộ lọc (PRICE, SIZE, COLOR, BRAND). | <ul style="list-style-type: none"> - Hiển thị tất cả sản phẩm trong hệ thống. - Thanh trượt giá hiển thị min–max mặc định (\$133.00–\$963.00). |
| CAT_30 | Kiểm tra khi người dùng chọn 1 mục duy nhất trong mỗi bộ lọc | Tại trang Home, di chuột (hover) vào menu Shop và chọn mục Women. | |
| | | Tích chọn: Size = L | |
| | | Tích chọn: Color = Black | |
| | | Tích chọn: Brand = Nike | |
| | | Quan sát danh sách sản phẩm | <ul style="list-style-type: none"> Chi hiển thị các sản phẩm thỏa cả 3 điều kiện: Size L, Color Black, Brand Nike Không hiển thị sản phẩm không thỏa mãn. |
| CAT_31 | Lọc sản phẩm khi chọn tất cả tùy chọn trong một nhóm | Tại trang Home, di chuột (hover) vào menu Shop và chọn mục Women. | |
| | | Tích tất cả các mục trong phần Size (L, X, XL, S, M) | |
| | | Quan sát danh sách sản phẩm. | Hệ thống hiển thị tất cả sản phẩm thỏa mãn tất cả điều kiện trên |
| CAT_32 | Không có sản phẩm phù hợp khi lọc | Tại trang Home, di chuột (hover) vào menu Shop và chọn mục Women. | |
| | | Chọn khoảng giá \$600–\$900. | |
| | | Tích Size = L | |
| | | Tích Color = White | |
| | | Tích Brand = Converse | Hiển thị thông báo "There is no product to display" |
| CAT_33 | Lọc sản phẩm khi tích mỗi bộ lọc size | Tại trang Home, di chuột (hover) vào menu Shop và chọn mục Women. | |
| | | Tích Size = S | Hiển thị các sản phẩm có size S |
| CAT_34 | Lọc sản phẩm khi tích mỗi bộ lọc Color | Tại trang Home, di chuột (hover) vào menu Shop và chọn mục Women. | |
| | | Tích Color = White | <ul style="list-style-type: none"> Hiển thị các sản phẩm có màu trắng Thumbnail hiển thị ảnh sản phẩm màu trắng |
| CAT_35 | Lọc sản phẩm khi tích mỗi bộ lọc Brand | Tại trang Home, di chuột (hover) vào menu Shop và chọn mục Women. | |
| | | Tích Brand = Nike | Hiển thị các sản phẩm có thương hiệu Nike |

Hình 6.11 Testcase chức năng lọc sản phẩm theo các biến thể

Nội dung chi tiết testcase đối với các item khác trên màn hình nhằm kiểm tra về UI, Validate dữ liệu của item, Function khi người dùng thao tác với các item của màn hình SR_004 xem chi tiết tại sheet “SR_004” tại file [EvershopTesting](#)

6.1.4.2 Chức năng tìm kiếm

Thiết kế được một số testcase như sau:

| | Chức năng tìm kiếm | | |
|-----------|---|--|--|
| Search_01 | Tìm kiếm sản phẩm thành công, khi điền chính xác tên của sản phẩm | Precondition: User vào hệ thống https://demo.evershop.io/ | |
| | | Click vào Search icon ở góc phải màn hình | Hiển thị Search textbox |
| | | Nhập tên sản phẩm vào Search textbox | Hiển thị nội dung vừa nhập vào search textbox |
| | | | Con trỏ chuột tự động vào ô nhập tìm kiếm |
| | | | Giao diện hiển thị "Search results for" + Tên sản phẩm |
| | | Nhấn enter | Hiển thị đúng số lượng sản phẩm |
| | | | Hiển thị các sản phẩm có từ khóa vừa nhập |
| | | | Không phân biệt hoa/thường |
| Search_02 | Tìm kiếm sản phẩm thành công, khi điền từ khóa của sản phẩm | Precondition: User vào hệ thống https://demo.evershop.io/ | |
| | | Click vào Search icon ở góc phải màn hình | Hiển thị Search textbox |
| | | Nhập tên sản phẩm vào Search textbox | Con trỏ chuột tự động vào ô nhập tìm kiếm |
| | | | Hiển thị nội dung vừa nhập vào search textbox |
| | | | Giao diện hiển thị "Search results for" + keywords |
| | | Nhấn enter | Hiển thị đúng số lượng sản phẩm |
| | | | Hiển thị các sản phẩm có từ khóa vừa nhập |
| | | | Không phân biệt hoa/thường |
| Search_03 | Tìm kiếm không thành công, nhập sản phẩm không tồn tại | Precondition: User vào hệ thống https://demo.evershop.io/ | |
| | | Click vào Search icon ở góc phải màn hình | Hiển thị Search textbox |
| | | Nhập tên sản phẩm vào Search textbox | Hiển thị nội dung vừa nhập vào search textbox |
| | | | Con trỏ chuột tự động vào ô nhập tìm kiếm |
| | | | Dòng chữ: There is no product to display |
| | | Nhấn enter | Số sản phẩm hiển thị: 0 products |
| | | | Không bị lỗi, không crash, không hiển thị sản phẩm sai |
| Search_04 | Tìm kiếm với khoảng trắng thừa đầu/cuối | Precondition: User vào hệ thống https://demo.evershop.io/ | |
| | | Click vào Search icon ở góc phải màn hình | Hiển thị Search textbox |
| | | Nhập tên sản phẩm vào Search textbox | Hiển thị nội dung vừa nhập vào search textbox |
| | | | Con trỏ chuột tự động vào ô nhập tìm kiếm |
| | | | Giao diện hiển thị "Search results for" + trim(keywords) |
| | | Nhấn enter | " zoom " |
| | | | Hiển thị đúng số lượng sản phẩm |
| Search_05 | Tìm kiếm với từ khóa nhiều từ | Precondition: User vào hệ thống https://demo.evershop.io/ | |
| | | Click vào Search icon ở góc phải màn hình | Hiển thị Search textbox |
| | | Nhập tên sản phẩm vào Search textbox | Con trỏ chuột tự động vào ô nhập tìm kiếm |
| | | | Hiển thị nội dung vừa nhập vào search textbox |
| | | | Giao diện hiển thị "Search results for" + keywords |
| | | Nhấn enter | Hiển thị các sản phẩm chứa đủ cả hai từ |
| | | | Hiển thị đúng số lượng sản phẩm |
| | | | Không phân biệt hoa/thường |
| Search_06 | Tìm kiếm trống (empty) | Precondition: User vào hệ thống https://demo.evershop.io/ | |
| | | Click vào Search icon ở góc phải màn hình | Hiển thị Search textbox |
| | | Nhấn enter | Quay lại giao diện trang chủ |
| Search_07 | Tìm kiếm 1 ký tự | Precondition: User vào hệ thống https://demo.evershop.io/ | |
| | | Click vào Search icon ở góc phải màn hình | Hiển thị Search textbox |
| | | Nhập tên sản phẩm vào Search textbox | Con trỏ chuột tự động vào ô nhập tìm kiếm |
| | | | Hiển thị nội dung vừa nhập vào search textbox |
| | | | Hiển thị các sản phẩm chứa kí tự đó |
| | | Nhấn enter | Hiển thị đúng số lượng sản phẩm |
| | | | Không phân biệt hoa/thường |

Hình 6.12 Testcase cho chức năng tìm kiếm

6.1.5 Chức năng màn hình SR_005

6.1.5.1 Chức năng nhập thông tin giao hàng

Dựa vào mô tả chi tiết chức năng Nhập thông tin giao hàng trong đặc tả màn hình SR_005 ta có phạm vi kiểm thử

| No. | Item | Type | Required | Default value | Min/Max | Valid format | Descriptions |
|-----|-----------------------|------------|----------------------------------|-------------------------------------|---------------------|--------------|--|
| 1 | Breadcrumb: Shipment | Link | - | - | - | - | Hiển thị đường dẫn: Shipment |
| 2 | Contact (email) | Text | - | - | - | - | Hiển thị email liên hệ người mua (email đăng nhập). |
| 3 | Shipping Address | Text | - | - | - | - | Hiển thị title "Shipping Address" |
| 4 | [Full name] textbox | Text box | <input checked="" type="radio"/> | - Blank - Placeholder: Full name | Max length = 50 | - | Nhập Full name hiển thị lên textbox |
| 5 | [Telephone] textbox | Text box | <input checked="" type="radio"/> | - Blank - Placeholder: Telephone | length: 8-11 digits | - | Nhập số điện thoại hiển thị lên textbox |
| 6 | [Address] textbox | Text box | <input checked="" type="radio"/> | - Blank - Placeholder: Address | Max length = 100 | - | Nhập địa chỉ hiển thị lên textbox |
| 7 | [City] textbox | Text box | <input checked="" type="radio"/> | - Blank - Placeholder: City | Max length = 50 | - | Nhập tên thành phố hiển thị lên textbox |
| 8 | [Country] select box | Select box | <input checked="" type="radio"/> | | - | - | Hiển thị các select option tương ứng với [Country] và các select option được sắp xếp tăng dần A-Z |
| 9 | [Province] select box | Select box | <input checked="" type="radio"/> | | - | - | Hiển thị các select option tương ứng với [Province] và các select option được sắp xếp tăng dần A-Z |
| 10 | [Postcode] textbox | Text box | <input checked="" type="radio"/> | - Blank - Placeholder: Postcode | Max length = 20 | - | Nhập postcode và hiển thị lên textbox |

Hình 6.13. Mô tả chi tiết chức năng nhập thông tin giao hàng

Từ bảng đặc tả trên, ta thiết kế các ca kiểm thử:

| FUN | | | | |
|----------|---|--|--|-----------------|
| PAY - 34 | Thêm địa chỉ không thành công, không chọn [Country] | Precondition: User đang ở trang shipment | | |
| | | Nhập [Full name] | Hiển thị dữ liệu vừa nhập vào textbox | Trang |
| | | Nhập [Telephone] | | 033 233 32 48 |
| | | Nhập [Address] | | 90 Nanjing Road |
| | | Nhập [City] | | Shanghai |
| | | Chọn[Province] | Dữ liệu được chọn/nhập thành công | Shanghai |
| | | Nhập [Postcode] | Hiển thị dữ liệu vừa nhập vào textbox | 22123 |
| | | | Không chuyển trang, hiển thị thông báo lỗi hợp lệ. | |
| | | Click [Continue to payment] button | Hiển thị thông báo lỗi dưới [Country] textbox | |
| | | | Trường Country được đánh dấu lỗi viền đỏ và focus | |

| FUN | | | | | |
|----------|--|--|--|-----------------|--|
| PAY - 35 | Thêm địa chỉ không thành công, không chọn [Province] | Precondition: User đang ở trang shipment | | | |
| | | Nhập [Full name] | Hiển thị dữ liệu vừa nhập vào textbox | Trang | - Address được thêm thành công - Chuyển trang Payment |
| | | Nhập [Telephone] | | 033 233 32 48 | |
| | | Nhập [Address] | | 90 Nanjing Road | |
| | | Nhập [City] | | Shanghai | |
| | | Chọn [Country] | Dữ liệu được chọn/nhập thành công | China | |
| | | Nhập [Postcode] | Hiển thị dữ liệu vừa nhập vào textbox | 22123 | |
| | | | Không chuyển trang, hiển thị thông báo lỗi hợp lệ. | | |
| | | Click [Continue to payment] button | Hiển thị thông báo lỗi dưới [Province] textbox | | |
| | | | Trường Province được đánh dấu lỗi viền đỏ và focus | | |

| | | | | | |
|----------|---|---|---------------------------------------|--|--|
| PAY - 36 | Thêm địa chỉ không thành công, nhập [Address] lớn hơn 100 kí tự | Precondition: User đang ở trang shipment | | | |
| | | Nhập [Full name] | Hiển thị dữ liệu vừa nhập vào textbox | Trang | - Address được thêm thành công - Chuyển trang Payment |
| | | Nhập [Telephone] | | 033 233 32 48 | |
| | | Nhập [Address] | | 12 Rue Didouche Mourad, Bab El Oued, Wilaya d'Alger, Apartment 101, Building B, Near Central Market, Landmark: Notre-Dame d'Afrique, Additional info: Floor 5, Door 12 | |
| | | Nhập [City] | | Shanghai | |
| | | Chọn [Country] | | China | |
| | | Chọn[Province] | | Shanghai | |
| | | Nhập [Postcode] | | 22123 | |
| | | Click [Continue to payment] button | | Không chuyển trang, hiển thị thông báo lỗi hợp lệ. | |
| | | | | Hiển thị thông báo lỗi dưới [Address] textbox | |

| | | | | | |
|----------|---|---|---------------------------------------|---|--|
| PAY - 37 | Thêm địa chỉ không thành công, nhập [City] lớn hơn 50 kí tự | Precondition: User đang ở trang shipment | | | |
| | | Nhập [Full name] | Hiển thị dữ liệu vừa nhập vào textbox | Trang | - Address được thêm thành công - Chuyển trang Payment |
| | | Nhập [Telephone] | | 033 233 32 48 | |
| | | Nhập [Address] | | 90 Nanjing Road | |
| | | Nhập [City] | | A Very Long City Name That Exceeds Fifty Characters For | |
| | | Chọn [Country] | | China | |
| | | Chọn[Province] | | Shanghai | |
| | | Nhập [Postcode] | | 22123 | |
| | | Click [Continue to payment] button | | Không chuyển trang, hiển thị thông báo lỗi hợp lệ. | |
| | | | | Hiển thị thông báo lỗi | |

| | | | | | |
|--------|---|---|---|---|--|
| PAY_38 | Thêm địa chỉ không thành công, nhập [Postcode] lớn hơn 20 kí tự | Precondition: User đang ở trang shipment | | | <ul style="list-style-type: none"> - Address được thêm thành công - Chuyển trang Payment |
| | | Nhập [Full name] | Hiển thị dữ liệu vừa nhập vào textbox | Trang | |
| | | Nhập [Telephone] | | 033 233 32 48 | |
| | | Nhập [Address] | | 90 Nanjing Road | |
| | | Nhập [City] | | Shanghai | |
| | | Chọn [Country] | Dữ liệu được chọn/nhập thành công | China | |
| | | Chọn[Province] | | Shanghai | |
| | | Nhập [Postcode] | Hiển thị dữ liệu vừa nhập vào textbox | 560001-EXTRA-POSTCODE -TO-EXCEED-LIMIT-12345 | |
| | | Click [Continue to payment] button | Không chuyển trang, hiển thị thông báo lỗi hợp lệ. | | |
| | | | Hiển thị thông báo lỗi | | |
| | | | Trường được đánh dấu lỗi viền đỏ và focus | | |
| PAY_39 | Thêm địa chỉ không thành công, nhập [Telephone] không hợp lệ | Precondition: User đang ở trang shipment | | | <ul style="list-style-type: none"> - Address được thêm không thành công - Hiển thị thông báo lỗi - không chuyển trang Payment |
| | | Nhập [Full name] | Hiển thị dữ liệu vừa nhập vào textbox | Trang | |
| | | Nhập [Telephone] | | 9933 233 32 48 | |
| | | Nhập [Address] | | 90 Nanjing Road | |
| | | Nhập [City] | | Shanghai | |
| | | Chọn [Country] | Dữ liệu được chọn/nhập thành công | China | |
| | | Chọn[Province] | | Shanghai | |
| | | Nhập [Postcode] | Hiển thị dữ liệu vừa nhập vào textbox | 22123 | |
| | | Click [Continue to payment] button | Không chuyển trang, hiển thị thông báo lỗi hợp lệ. | | |
| | | | Hiển thị thông báo lỗi | | |
| | | | Trường được đánh dấu lỗi viền đỏ và focus | | |
| PAY_40 | Thêm địa chỉ không thành công, nhập dấu cách | Precondition: User đang ở trang shipment | | | <ul style="list-style-type: none"> - Address được thêm không thành công - Hiển thị thông báo lỗi - không chuyển trang Payment |
| | | Click vào [user] icon button ở thanh bar, phía bên phải | Điều hướng đến trang [Tài khoản - Thông tin] | | |
| | | Trong section [Address book], click hyperlink [Add new address] | Hiển thị [Add new address] popup | | |
| | | Nhập space vào [Full name] | Field hiển thị khoảng trắng. | | |
| | | Nhập space vào [Telephone] | | | |
| | | Nhập space vào [Address] | | | |
| | | Nhập space vào [City] | | | |
| | | Chọn [Country] | Dữ liệu được chọn/nhập thành công | China | |
| | | Chọn[Province] | | Shanghai | |
| | | Nhập space vào [Postcode] | Field hiển thị khoảng trắng. | | |
| | | Click [Continue to payment] button | Form không được lưu. | | |
| | | | - Hiển thị lỗi cho tất cả các field trống như: | | |
| | | | <ul style="list-style-type: none"> • “Full name is required.” • “Telephone is required.” • “Address is required.” • “City is required.” • “Postcode is required.” - Popup vẫn mở. | | |
| | | | - Không chuyển trang payment | | |

Hình 6.14 Testcase nhập thông tin giao hàng

6.1.5.2 Chức năng thanh toán

Thanh toán bằng thẻ visa:

Áp dụng kĩ thuật phân vùng tương đương:

Thanh toán bằng phương thức thẻ visa

| | C1 [Số thẻ] | C2 [Ngày hết hạn] định dạng MM/YY | C3 [Mã bảo mật] | Output |
|----|--|---|--------------------|---------|
| T | 4242 4242 4242 4242 | 04/26 | 242 | Valid |
| C1 | 4242 4242 4242 4245 (Số thẻ sai) 4000 0000 0000 9995 (Thẻ có số dư không đủ) | 04/26 | 242 | Invalid |
| C2 | 4242 4242 4242 4242 | 03/26 (Thời gian sai) 03/25 (Thời gian trong quá khứ) | 242 | Invalid |
| C3 | 4242 4242 4242 4242 | 04/26 | 223 | Invalid |

Hình 6.15 Bảng phân vùng tương đương cho chức năng thanh toán bằng thẻ visa

Ta thiết kế các ca kiểm thử:

| | | | | | |
|-------------|---|--|--|--|--------------------------|
| PAY - 50 | Đặt hàng thành công bằng phương thức dùng thẻ visa | Precondition: User đăng nhập vào hệ thống | | Card địa chỉ: Full name: Nguyen Van A Address: 1600 Pennsylvania Ave NW Postcode: 20500 City: Washington Province: Washington Country: United States Telephone: 0988956088 | |
| | | Trong giỏ hàng, nhấn [Checkout] button | Điều hướng đến trang Chọn địa chỉ - Thanh toán | | |
| | | Trong [Shipping Address] section, chọn 1 địa chỉ, click [Ship here] link | Các trường textbox hiển thị tương ứng | | |
| | | Trong [Shipping Method] chọn [Standard Delivery - \$5.00] | Trong Order Summary hiển thị hàng Shipping: Standard Delivery: \$5.00 Hiển thị đúng tổng tiền sản phẩm: Giá sản phẩm + Ship | | |
| | | Click [Continue to payment] button | Điều hướng đến trang Chọn phương thức thanh toán - Thanh toán | | |
| | | Trong [Payment Method] Click [Visa] radio button | Hiển thị Form nhập thông tin thẻ hiển thị gồm: Số thẻ, Ngày hết hạn, Mã bảo mật, Quốc gia | | |
| | | Nhập thông tin thẻ hợp lệ: • Card Number: 4242 4242 4242 4242 • Expiry: 04/26 • CVC: 242 • Country: Việt Nam | Hiển thị dữ liệu lên textbox | | |
| | | Click [Place order] button | Hiển thị trang Thanh toán thành công Nút Continue Shopping: Có thể click | | |
| PAY - 52 | Đặt hàng không thành công, nhập sai mã bảo mật CVC của phương thức visa | Precondition: User đăng nhập vào hệ thống | | Card địa chỉ: Full name: Nguyen Van A Address: 1600 Pennsylvania Ave NW Postcode: 20500 City: Washington Province: Washington Country: United States Telephone: 0988956088 | |
| | | Trong giỏ hàng, nhấn [Checkout] button | Điều hướng đến trang Chọn địa chỉ - Thanh toán | | |
| | | Trong [Shipping Address] section, chọn 1 địa chỉ, click [Ship here] link | Các trường textbox hiển thị tương ứng | | |
| | | Trong [Shipping Method] chọn [Standard Delivery - \$5.00] | Trong Order Summary hiển thị hàng Shipping: Standard Delivery: \$5.00 Hiển thị đúng tổng tiền sản phẩm: Giá sản phẩm + Ship | | |
| | | Click [Continue to payment] button | Điều hướng đến trang Chọn phương thức thanh toán - Thanh toán | | |
| | | Trong [Payment Method] Click [Visa] radio button | Hiển thị Form nhập thông tin thẻ hiển thị gồm: Số thẻ, Ngày hết hạn, Mã bảo mật, Quốc gia | | |
| | | Nhập thông tin thẻ hợp lệ: • Card Number: 4242 4242 4242 4242 • Expiry: 04/26 • CVC: 223 • Country: Việt Nam | Hiển thị thông báo lỗi dưới mã bảo mật textbox: "Sai mã bảo mật vui lòng nhập lại" | | |
| | | | Con trỏ chuột tự động focus vào textbox [Mã bảo mật], viền textbox chuyển đỏ | | Thanh toán thành công |

| | | | | | |
|----------|---|--|---|---|---|
| PAY - 53 | Đặt hàng không thành công, nhập ngày hết hạn < ngày hiện tại trong phương thức visa | Precondition: User đăng nhập vào hệ thống | | Card địa chỉ: Full name: Nguyen Van A Address: 1600 Pennsylvania Ave NW Postcode: 20500 City: Washington Province: Washington Country: United States Telephone: 0988956088 | Không update giá ship lên Order |
| | | Trong giờ hàng, nhấn [Checkout] button | Điều hướng đến trang Chọn địa chỉ - Thanh toán | | |
| | | Trong [Shipping Address] section, chọn 1 địa chỉ, click [Ship here] link | Các trường textbox hiển thị tương ứng | | |
| | | Trong [Shipping Method] chọn [Standard Delivery - \$5.00] | Trong Order Summary hiển thị hàng Shipping: Standard Delivery: \$5.00 | | |
| | | Click [Continue to payment] button | Điều hướng đến trang Chọn phương thức thanh toán - Thanh toán | | |
| | | Trong [Payment Method] Click [Visa] radio button | Hiển thị Form nhập thông tin thẻ hiển thị gồm: Số thẻ, Ngày hết hạn, Mã bảo mật, Quốc gia | | |
| | | Nhập thông tin thẻ hợp lệ: • Card Number: 4242 4242 4242 4242 • Expiry: 03/25 • CVC: 242 • Country: Việt Nam | Hiển thị thông báo lỗi dưới mã bảo mật textbox: "Ngày hết hạn không hợp lệ vui lòng nhập lại" Con trỏ chuột tự động focus vào textbox [Ngày hết hạn], viền textbox chuyển đỏ | | |
| PAY - 54 | Đặt hàng không thành công, nhập sai mã ngày hết hạn của phương thức visa | Precondition: User đăng nhập vào hệ thống | | Card địa chỉ: Full name: Nguyen Van A Address: 1600 Pennsylvania Ave NW Postcode: 20500 City: Washington Province: Washington Country: United States Telephone: 0988956088 | Không update giá ship lên Order summary |
| | | Trong giờ hàng, nhấn [Checkout] button | Điều hướng đến trang Chọn địa chỉ - Thanh toán | | |
| | | Trong [Shipping Address] section, chọn 1 địa chỉ, click [Ship here] link | Các trường textbox hiển thị tương ứng | | |
| | | Trong [Shipping Method] chọn [Standard Delivery - \$5.00] | Trong Order Summary hiển thị hàng Shipping: Standard Delivery: \$5.00 Hiển thị đúng tổng tiền sản phẩm: Giá sản phẩm + Ship | | |
| | | Click [Continue to payment] button | Điều hướng đến trang Chọn phương thức thanh toán - Thanh toán | | |
| | | Trong [Payment Method] Click [Visa] radio button | Hiển thị Form nhập thông tin thẻ hiển thị gồm: Số thẻ, Ngày hết hạn, Mã bảo mật, Quốc gia | | |
| | | Nhập thông tin thẻ hợp lệ: • Card Number: 4242 4242 4242 4242 • Expiry: 03/26 • CVC: 242 • Country: Việt Nam | Hiển thị thông báo lỗi dưới mã bảo mật textbox: "Sai ngày hết hạn vui lòng nhập lại" Con trỏ chuột tự động focus vào textbox [Ngày hết hạn], viền textbox chuyển đỏ | | |
| PAY - 55 | Đặt hàng không thành công, nhập số thẻ visa không hợp lệ | Precondition: User đăng nhập vào hệ thống | | Card địa chỉ: Full name: Nguyen Van A Address: 1600 Pennsylvania Ave NW Postcode: 20500 City: Washington Province: Washington Country: United States Telephone: 0988956088 | Thanh toán thành công |
| | | Trong giờ hàng, nhấn [Checkout] button | Điều hướng đến trang Chọn địa chỉ - Thanh toán | | |
| | | Trong [Shipping Address] section, chọn 1 địa chỉ, click [Ship here] link | Các trường textbox hiển thị tương ứng | | |
| | | Trong [Shipping Method] chọn [Standard Delivery - \$5.00] | Trong Order Summary hiển thị hàng Shipping: Standard Delivery: \$5.00 Hiển thị đúng tổng tiền sản phẩm: Giá sản phẩm + Ship | | |
| | | Click [Continue to payment] button | Điều hướng đến trang Chọn phương thức thanh toán - Thanh toán | | |
| | | Trong [Payment Method] Click [Visa] radio button | Hiển thị Form nhập thông tin thẻ hiển thị gồm: Số thẻ, Ngày hết hạn, Mã bảo mật, Quốc gia | | |
| | | Nhập thông tin thẻ hợp lệ: • Card Number: 4242 4242 4242 4245 • Expiry: 04/26 • CVC: 242 • Country: Việt Nam | Hiển thị thông báo lỗi dưới mã bảo mật textbox: "Số thẻ của quý vị không đầy đủ." Con trỏ chuột tự động focus vào textbox [Số thẻ], viền textbox chuyển đỏ | | |

Hình 6.16 Testcase cho chức năng chúc năng thanh toán bằng thẻ visa

CHƯƠNG 7: THỰC THI VÀ BÁO CÁO

7.1 Thực thi kiểm thử

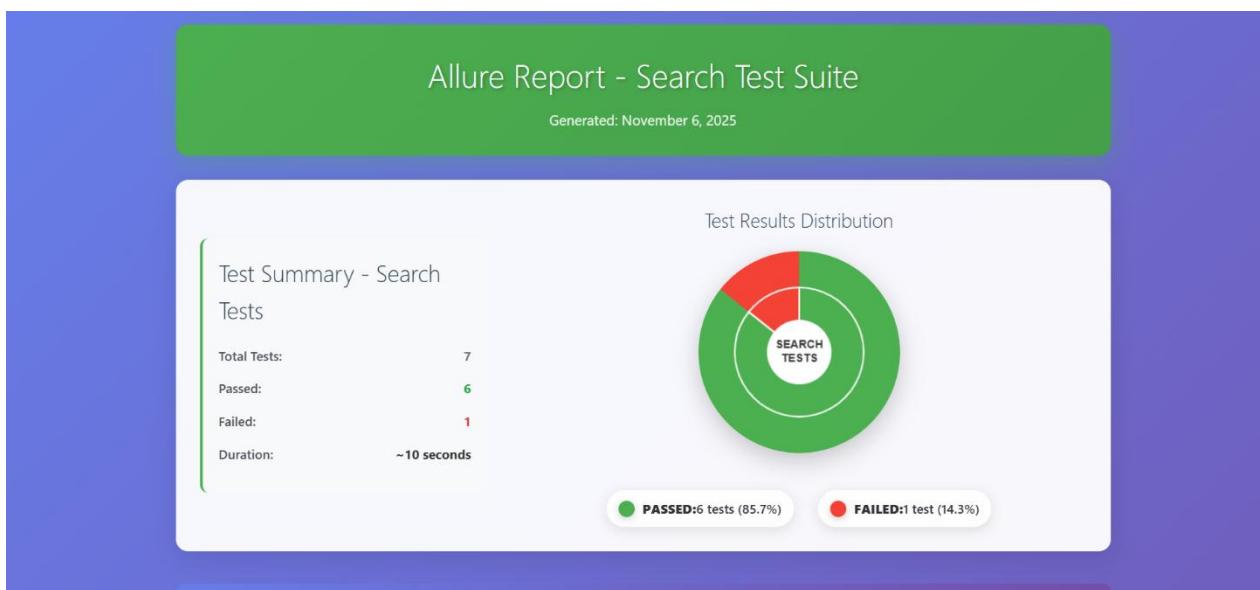
7.1.1 Kiểm thử tự động

7.1.1.1 Chức năng tìm kiếm sản phẩm

Theo phần phân tích và thiết kế testcase các chức năng tìm kiếm sản phẩm tại **Chương 6. Phân tích và thiết kế ca kiểm thử mục 6.1.5**, ta thực hiện thiết kế được:

Với chức năng Tìm kiếm: thiết kế được tổng 7 ca kiểm thử cho textbox [Tìm kiếm] về Function.

Sử dụng Selenium Webdriver và Intelij viết các test scripts để thực hiện các ca kiểm thử cho Tìm kiếm ta có kết quả: pass 6 testcase, fail 1 testcase, không có testcase skip.



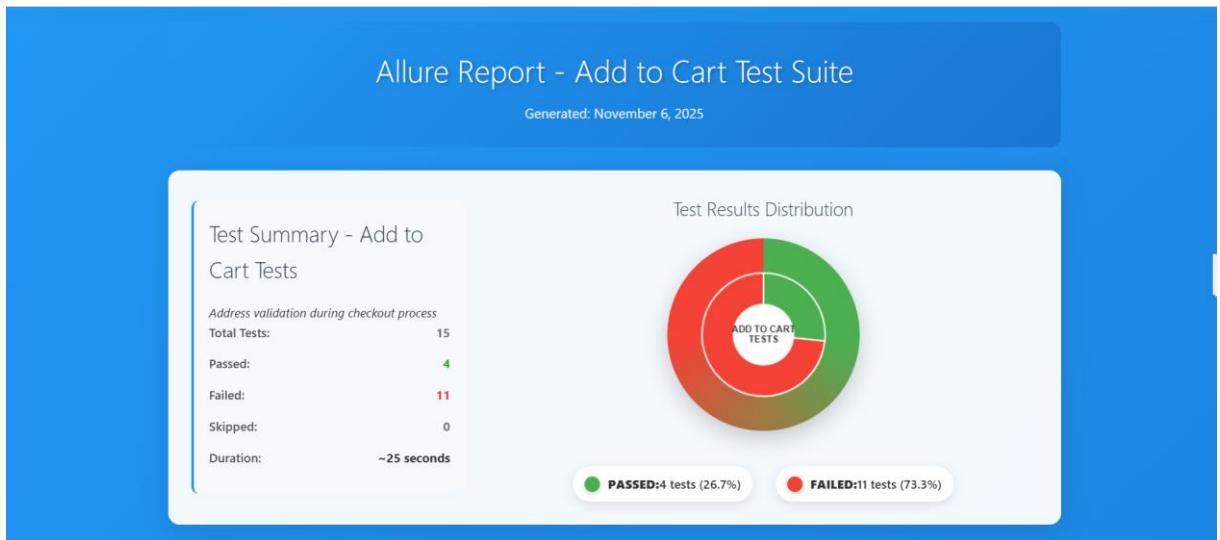
Hình 7.1 Kết quả kiểm thử chức năng tìm kiếm sản phẩm

7.1.1.2 Chức năng thêm địa chỉ đặt hàng

Theo phần phân tích và thiết kế testcase các chức năng thêm địa chỉ tại **Chương 6. Phân tích và thiết kế ca kiểm thử mục 6.1.3.2**, ta thực hiện thiết kế được:

Với chức năng thêm địa chỉ: thiết kế được tổng 15 ca kiểm thử về Function

Sử dụng Selenium Webdriver và Intelij viết các test scripts để thực hiện các ca kiểm thử cho thêm địa chỉ ta có kết quả: pass 4 testcase, fail 11 testcase, không có testcase skip.



Hình 7.2 Kết quả kiểm thử chức năng thêm địa chỉ

7.1.2 Kiểm thử thủ công

Bên cạnh các chức năng kiểm thử tự động thì cũng có các chức năng, UI kiểm thử thủ công. Các testcase thực hiện thủ công được kiểm thử trên trình duyệt Chrome theo các testcase trong file testcase. Dưới đây là tổng số testcase thực hiện thủ công trong từng màn hình

Màn hình SR_001 – Màn hình chi tiết sản phẩm

| | | | |
|------------------|---------------------------|----|--------------|
| RESULT | PASSED | 22 | |
| | FAILED | 7 | |
| | BLOCKED | 0 | |
| | NOT RUN | 0 | |
| | AUTOMATION | 0 | |
| | MANUAL | 29 | |
| EXECUTION | N/A | 0 | UI |
| | Number of Testcase | 29 | FUNCTION |
| | | | VALIDATE |
| | | | TOTAL |
| | | | 16 |
| | | | 13 |
| | | | 0 |
| | | | 29 |

Màn hình SR_002 - Giỏ hàng

| | | | |
|------------------|---------------------------|----|--------------|
| RESULT | PASSED | 24 | |
| | FAILED | 5 | |
| | BLOCKED | 1 | |
| | NOT RUN | 0 | |
| | AUTOMATION | 0 | |
| | MANUAL | 30 | |
| EXECUTION | N/A | 0 | UI |
| | Number of Testcase | 30 | FUNCTION |
| | | | VALIDATE |
| | | | TOTAL |
| | | | 17 |
| | | | 13 |
| | | | 0 |
| | | | 30 |

Màn hình SR_003 - Thông tin tài khoản

| | | | | |
|--------------------|------------|----|----------|----|
| RESULT | PASSED | 37 | | |
| | FAILED | 25 | | |
| | BLOCKED | 0 | | |
| | NOT RUN | 0 | | |
| EXECUTION | AUTOMATION | 15 | | |
| | MANUAL | 47 | | |
| | N/A | 0 | | |
| Number of Testcase | | 62 | | |
| | | | UI | 38 |
| | | | FUNCTION | 24 |
| | | | VALIDATE | 0 |
| | | | TOTAL | 62 |

Màn hình SR_004 - Doanh mục sản phẩm nữ

| | | | | |
|--------------------|------------|----|----------|----|
| RESULT | PASSED | 36 | | |
| | FAILED | 6 | | |
| | BLOCKED | 0 | | |
| | NOT RUN | 0 | | |
| EXECUTION | AUTOMATION | 7 | | |
| | MANUAL | 35 | | |
| | N/A | 0 | | |
| Number of Testcase | | 42 | UI | 19 |
| | | | FUNCTION | 23 |
| | | | VALIDATE | 0 |
| | | | TOTAL | 42 |

Màn hình SR_005 – Thanh toán

| | | | | |
|--------------------|------------|----|----------|----|
| RESULT | PASSED | 59 | | |
| | FAILED | 17 | | |
| | BLOCKED | 3 | | |
| | NOT RUN | 0 | | |
| EXECUTION | AUTOMATION | 0 | | |
| | MANUAL | 77 | | |
| | N/A | 0 | | |
| Number of Testcase | | 79 | UI | 33 |
| | | | FUNCTION | 33 |
| | | | TOTAL | 66 |

Tổng số testcase trong [tất cả các màn hình](#) là 242 testcase trong đó :

+ Pass : 178

+ Fail : 60

+ Block : 4

+ Not run : 0

KẾT LUẬN

Sau quá trình thực hiện đề tài “**Kiểm thử trang web bán giày**”, em đã áp dụng các kiến thức đã học về **kiểm thử phần mềm** để đánh giá chất lượng và độ ổn định của một hệ thống thương mại điện tử thực tế.

đã tiến hành **kiểm thử thủ công trên trình duyệt Chrome** và **kiểm thử tự động bằng Selenium WebDriver** đối với các chức năng chính như: xem chi tiết sản phẩm, thêm sản phẩm vào giỏ hàng, cập nhật số lượng, thanh toán và quản lý tài khoản. Tổng cộng **242 testcase** được thực hiện, trong đó có **178 Pass ($\approx 73.55\%$)**, **60 Fail** và **4 Block**. So với tiêu chí trong kế hoạch kiểm thử (**yêu cầu $\geq 90\%$ Pass**), kết quả này **chưa đạt yêu cầu đề ra**.

Các lỗi phát hiện không chỉ nằm ở giao diện mà còn **ảnh hưởng trực tiếp đến một số chức năng chính** như thêm sản phẩm, xử lý thanh toán và tài khoản người dùng. Điều này cho thấy hệ thống vẫn cần được **kiểm tra và hiệu chỉnh thêm** để đảm bảo tính ổn định và trải nghiệm người dùng tốt hơn.

Dự án giúp hiểu rõ hơn quy trình kiểm thử phần mềm, rèn luyện kỹ năng viết và thực thi **testcase**, đồng thời thành thạo việc sử dụng công cụ **Selenium** để tự động hóa quy trình kiểm thử. Trong thời gian tới, em sẽ tiếp tục **phân tích nguyên nhân lỗi, tái kiểm thử và tối ưu quy trình tự động hóa** để hoàn thiện hệ thống và nâng cao chất lượng kiểm thử.