|  |  |
| --- | --- |
|  |  |
| **ĐÀO THỊ THUỲ LINH** | **BỘ CÔNG THƯƠNG**  **TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**  **---------------------------------------** |
|  |
| **ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC** |
| **NGÀNH HỆ THỐNG THÔNG TIN** |
| **TÌM HIỂU CÔNG CỤ SELENIUM WEBDRIVER**  **VÀ VẬN DỤNG VÀO KIỂM THỬ ỨNG DỤNG WEB** |
|  |
|  |
| **CBHD :TS. Nguyễn Bá Nghiễn** |
| **HỆ THỐNG THÔNG TIN** | **Sinh viên : Đào Thị Thuỳ Linh** |
| **Mã số sinh viên: 2021600594** |
|  |
|  |
|  |
| **Hà Nội – Năm 2025** |

# **LỜI CẢM ƠN**

Để có được kết quả như ngày hôm nay, trước tiên em xin gửi lời cảm ơn sâu sắc nhất tới thầy Nguyễn Bá Nghiễn, giảng viên khoa Công nghệ thông tin, Đại học Công nghiệp Hà Nội đã tận tình hướng dẫn, giúp đỡ em trong suốt quá trình nghiên cứu và thực đồ án tốt nghiệp.

Em cũng xin bày tỏ lòng biết ơn chân thành tới các thầy cô trong Đại học Công nghiệp Hà Nội đặc biệt là các thầy cô trong khoa Công nghệ thông tin đã tận tụy dạy dỗ em trong quá trình học tập, rèn luyện suốt thời gian qua, giúp em có được những kiến thức không chỉ để hoàn thành đồ án mà còn học tập và làm việc sau này.

Và cuối cùng tôi xin gửi lời cảm ơn tới những người bạn luôn ở bên, chia sẻ cùng tôi những lúc khó khăn, đóng góp ý kiến và giúp đỡ trong quá trình học tập, nghiên cứu và hoàn thành đồ án tốt nghiệp.

Trong quá trình thực hiện đồ án, mặc dù đã cố gắng để hoàn thiện đề tài tốt nhất có thể, nhưng chắc chắn không thể tránh khỏi những thiếu sót. Vì vậy, em rất mong nhận được sự góp ý của thầy cô và các bạn để đồ án của em được hoàn thiện hơn.

Em xin chân thành cảm ơn!

Hà Nội, ngày tháng năm 2025

Sinh viên thực hiện

**MỤC LỤC**

[LỜI CẢM ƠN i](#_Toc198905150)

[DANH SÁCH TỪ VIẾT TẮT, THUẬT NGỮ iv](#_Toc198905151)

[DANH MỤC HÌNH VẼ v](#_Toc198905152)

[DANH MỤC BẢNG BIỂU vi](#_Toc198905153)

[LỜI MỞ ĐẦU 1](#_Toc198905154)

[CHƯƠNG 1. TỔNG QUAN VỀ KIỂM THỬ 4](#_Toc198905155)

[1.1. Kiểm thử phần mềm 4](#_Toc198905156)

[1.1.1. Khái niệm kiểm thử phần mềm 4](#_Toc198905157)

[1.1.2. Quy trình kiểm thử phần mềm 4](#_Toc198905158)

[1.1.3. Các loại kiểm thử 5](#_Toc198905159)

[1.1.4. Các phương pháp kiểm thử 8](#_Toc198905160)

[1.1.5. Thiết kế test case 10](#_Toc198905161)

[1.2. Khái quát về kiểm thử tự động 14](#_Toc198905162)

[1.2.1. Khái niệm kiểm thử tự động 15](#_Toc198905163)

[1.2.2. Quy trình kiểm thử tự động 15](#_Toc198905164)

[1.2.3. So sánh kiểm thử thủ công và kiểm thử tự động 16](#_Toc198905165)

[1.2.4. Một số công cụ kiểm thử tự động phổ biến 18](#_Toc198905166)

[CHƯƠNG 2. NGHIÊN CỨU CÔNG CỤ KIỂM THỬ TỰ ĐỘNG SELENIUM WEBDRIVER 19](#_Toc198905167)

[2.1. Công cụ kiểm thử tự động Selenium 19](#_Toc198905168)

[2.1.1. Khái quát về Selenium 19](#_Toc198905169)

[2.1.2. Đặc điểm của Selenium 19](#_Toc198905170)

[2.1.3. Các thành phần của Selenium 20](#_Toc198905171)

[2.2. Những vấn đề mới của Selenium 23](#_Toc198905172)

[2.3. Selenium Webdriver 24](#_Toc198905173)

[2.3.1. Tiền thân của Selenium WebDriver 24](#_Toc198905174)

[2.3.2. Đặc trưng của Selenium WebDriver 25](#_Toc198905175)

[2.3.3.Cách lấy xpath của element trên website 25](#_Toc198905176)

[2.3.4. Các câu lệnh sử dụng trong Selenium WebDriver 27](#_Toc198905177)

[2.2.5. Ngôn ngữ lập trình Java 32](#_Toc198905178)

[2.3. Cài đặt Selenium Webdriver 34](#_Toc198905179)

[CHƯƠNG 3. ỨNG DỤNG CÔNG CỤ KIỂM THỬ CHO HỆ THỐNG WEBSITE BÁN THỰC PHẨM CHỨC NĂNG 43](#_Toc198905180)

[3.1. Mô tả về website bán thực phẩm chức năng 43](#_Toc198905181)

[3.1.1. Giới thiệu 43](#_Toc198905182)

[3.2. Kế hoạch kiểm thử 43](#_Toc198905183)

[3.2.1. Phạm vi kiểm thử 43](#_Toc198905184)

[3.2.2. Mục tiêu kiểm thử 44](#_Toc198905185)

[3.2.3. Chiến lược kiểm thử 44](#_Toc198905186)

[3.2.4 Tiêu chí kiểm thử 45](#_Toc198905187)

[3.2.5 Nguồn nhân lực 45](#_Toc198905188)

[3.2.6 Môi trường kiểm thử 45](#_Toc198905189)

[3.2.7 Lịch trình và dự toán 46](#_Toc198905190)

[3.2.8 Xác định phân phối kiểm thử 46](#_Toc198905191)

[3.3. Thiết kế ca kiểm thử 46](#_Toc198905192)

[3.3.1. Đăng ký 46](#_Toc198905193)

[3.3.2. Đăng nhập 50](#_Toc198905194)

[3.3.3. Tìm kiếm sản phẩm 52](#_Toc198905195)

[3.3.4. Cập nhật thông tin cá nhân 52](#_Toc198905196)

[3.3.5. Quên mật khẩu 54](#_Toc198905197)

[3.3.6. Quản lý giỏ hàng 55](#_Toc198905198)

[3.3.7. Mua hàng 57](#_Toc198905199)

[3.3.8. Đăng xuất 58](#_Toc198905200)

[3.4. Xây dựng test script 59](#_Toc198905201)

[3.5. Kết quả kiểm thử 64](#_Toc198905202)

[KẾT LUẬN 66](#_Toc198905203)

[TÀI LIỆU THAM KHẢO 68](#_Toc198905204)

# **DANH SÁCH TỪ VIẾT TẮT, THUẬT NGỮ**

|  |  |
| --- | --- |
| Tester | Kiểm thử viên |
| Testcase | Kịch bản kiểm thử |
| Software Testing | Kiểm thử phần mềm |
| Manual Testing | Kiểm thử phần mềm thủ công |
| Automation Testing | Kiểm thử phần mềm tự động |
| STLC  (Software Testing Life Cycle) | Quy trình kiểm thử phần mềm |

# **DANH MỤC HÌNH VẼ**

[Hình 1.1: Quy trình kiểm thử phần mềm 4](#_Toc199083214)

[Hình 2.1: Màn hình trang tải về Selenium Webdriver 35](#_Toc199083215)

[Hình 2.2: Thao tác tạo một project mới trong IntelliJ 36](#_Toc199083216)

[Hình 2.3: Chọn cấu hình JDK đúng với version Java ở IntelliJ 37](#_Toc199083217)

[Hình 2.4: Thao tác tải Selenium Java 38](#_Toc199083218)

[Hình 2.5: Chọn phiên bản phù hợp 39](#_Toc199083219)

[Hình 2.6: Thao tác cài đặt Selenium Java về project 39](#_Toc199083220)

[Hình 2.7: Thực hiện khởi động lại Maven 40](#_Toc199083221)

[Hình 2.8: Hoàn thiện cài đặt thư viện Selenium 41](#_Toc199083222)

[Hình 2.9: Cài đặt thư viện TestNG 41](#_Toc199083223)

[Hình 2.10: Hoàn thiện cài đặt TestNG 42](#_Toc199083224)

[Hình 3.1: Tạo project 60](#_Toc199083225)

[Hình 3.2: Tạo file chức các biến và phương thức chung 61](#_Toc199083226)

[Hình 3.3: Tạo file chứa các đường dẫn xpath và các hàm kiểm tra website 62](#_Toc199083227)

[Hình 3.4:Tạo file chứa các ca kiểm thử của từng chức năng 62](#_Toc199083228)

[Hình 3.5: Chạy các ca kiểm thử 63](#_Toc199083229)

[Hình 3.6: Hình ảnh minh họa kết quả kiểm thử 63](#_Toc199083230)

[Hình 3.7: Kết quả tổng quan thực hiện các test case 64](#_Toc199083231)

[Hình 3.8: Hình ảnh kết quả chi tiết của từng chức năng 65](#_Toc199083232)

[Hình 3.9: Hình ảnh thanh thời gian thực hiện test case 65](#_Toc199083233)

# **DANH MỤC BẢNG BIỂU**

[Bảng 1.1: Bảng quyết định 21](#_Toc198903236)

# **LỜI MỞ ĐẦU**

1. **Lý do chọn đề tài**

Trong những năm gần đây, công nghệ thông tin đã phát triển nhanh chóng, mạnh mẽ và được ứng dụng ở hầu hết các mặt đời sống, kinh tế, văn hóa, xã hội. Công nghệ thông tin phát triển cũng đồng nghĩa với nó là ngành công nghệ phần mềm cũng phát triển. Việc xây dựng các ứng dụng web ngày càng phổ biến và nó ngày càng trở nên thân thiết với con người qua các hoạt động giới thiệu, quảng cáo, mua bán, quản lí, ... trên mạng. Cuộc sống càng lên cao nhu cầu về chất lượng của một sản phẩm cũng tăng lên. Một sản phẩm không chỉ dừng lại ở việc thân thiện dễ sử dụng mà nó còn phải luôn chính xác, kịp thời và đảm bảo tính bảo mật tốt. Lỗi của một phần mềm nói chung hay của một ứng dụng web nói riêng có thể gây ra những hậu quả nghiêm trọng. Do đó, hoạt động đảm bảo chất lượng phần mềm là rất quan trọng. Trong khi đó hoạt động này lại tiêu tốn và chiếm tỷ trọng khá lớn công sức và thời gian của trong một dự án. Chính vì lý do đó, việc đưa vào quy trình kiểm thử là vô cùng cần thiết.

Bên cạnh đó, xu hướng áp dụng tự động hoá đang được triển khai rộng rãi ở nhiều lĩnh vực, trong đó có kiểm thử phần mềm. Đặc biệt, khi kiểm thử phần mềm là công đoạn chiếm phần lớn thời gian trong quá trình phát triển dự án phần mềm thì sự ra đời của các công cụ kiểm thử tự động càng có ý nghĩa hơn bao giờ hết, giúp tiết kiệm thời gian, công sức và tiền bạc. Selenium là một công cụ hỗ trợ kiểm thử tự động dành cho các ứng dụng Web, hoạt động trên hầu hết các trình duyệt phổ biến hiện nay như Firefox, Chrome, Internet Explorer, Safari, … cũng như hỗ trợ số lượng lớn các ngôn ngữ lập trình Web phổ biến. Công cụ Selenium hiện được đánh giá là một trong những công cụ tốt nhất cho kiểm thử tự động các ứng dụng Web kiểm thử là vô cùng cần thiết.

Vì những vấn đề cấp thiết đó cùng với mong muốn được tìm hiểu và tiếp cận gần hơn đến công cụ kiểm thử tự động Selenium góp phần tích lũy kiến thức, kinh nghiệm, nâng cao kỹ năng kiểm thử phần mềm để hướng đến những sản phẩm tốt nhất cho người sử dụng. Đồ án tốt nghiệp của mình, em xin lựa chọn đề tài “Tìm hiểu công cụ selenium webdriver và ứng dụng vào kiểm thử ứng dụng web”.

1. **Mục tiêu**

Vận dụng kiến thức về kiểm thử để thực hiện đánh giá, kiểm thử phần mềm và các tài liệu phần mềm.

Viết tài liệu hướng dẫn sử dụng, xây dựng kế hoạch kiểm thử, thiết kế test case, test report.

Biết cách cài đặt và sử dụng công cụ Selenium Webdriver với ngôn ngữ lập trình Java trong quá trình kiểm thử tự động.

Áp dụng tiến hành kiểm thử tự động chức năng trên website thương mại điện tử

1. **Đối tượng và phạm vi nghiên cứu**

Quyển đồ án tốt nghiệp trình bày kiến thức về kiểm thử phần mềm, các sản phẩm của kiểm thử chức năng cho hệ thống web thương mại điện tử cho các module chức năng như sau:

* Đăng ký
* Đăng nhập
* Đăng xuất
* Tìm kiếm sản phẩm
* Cập nhật thông tin cá nhân
* Quên mật khẩu
* Quản lý giỏ hàng: thêm, cập nhật, xóa sản phẩm trong giỏ hàng

1. **Ý nghĩa khoa học và thực tiễn của đề tài**

Từ những phân tích trên, ý tưởng cho một ứng dụng website bán hàng hình thành nhằm mục đích đáp ứng người tiêu dùng, chỉ với những thao tác đơn giản trên máy tính hoặc điện thoại có kết nối internet, người mua sắm sẽ có tận tay những sản phẩm (thông qua hình thức giao hàng tại nhà) mà mình ưng ý mà không cần phải ra trực tiếp cửa hàng. Điều này vẫn có thể giữ vững được sự cung cầu giữa người mua sắm và người bán, góp phần giữ vững nền kinh tế nước nhà.

Trong báo cáo đồ án này, em sẽ tập trung trình bày những nghiên cứu kiến thức cơ bản về quy trình kiểm thử phần mềm. Vai trò, mục tiêu và các kỹ thuật kiểm thử phần mềm, đi sâu hơn vào kiểm thử chức năng, các cách tiếp cận, quy trình và phương pháp kiểm thử chức năng. Phân tích và kiểm thử chức năng website bằng cách triển khai các bước: lập kế hoạch test, thiết kế test, tạo các testcase, thực hiện test và báo cáo kết quả test.

Trong quá trình tìm hiểu lý thuyết và áp dụng những kiến thức, kinh nghiệm của em có được vào hệ thống cũng chưa nhiều và còn nhiều hạn chế cũng như sai sót mong thầy cô và các bạn đóng góp ý kiến để đề tài được hoàn thiện hơn.

# 

# **TỔNG QUAN VỀ KIỂM THỬ**

## **1.1. Kiểm thử phần mềm**

### **1.1.1. Khái niệm kiểm thử phần mềm**

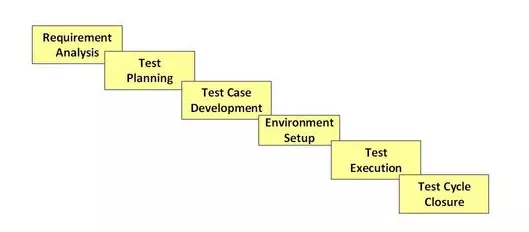
Kiểm thử phần mềm (software testing) là hoạt động nhằm tìm kiếm và phát hiện ra các lỗi của phần mềm, đảm bảo phần mềm chính xác, đúng và đầy đủ theo yêu cầu của khách hàng, yêu cầu của sản phẩm đã đặt ra.

Kiểm thử phần mềm đóng vai trò rất quan trọng:

* Kiểm thử phần mềm là hoạt động đảm bảo chất lượng phần mềm và mang tính sống còn trong các dự án sản xuất phần mềm. Vì vậy nó đã trở thành quy trình bắt buộc trong các dự án phần mềm hiện nay.
* Kiểm thử phần mềm để tránh những rủi ro, lỗi phát sinh trong suốt quá trình tạo ra sản phẩm.
* Lỗi càng phát hiện ra sớm càng giúp tránh được rủi ro và chi phí.

### **1.1.2. Quy trình kiểm thử phần mềm**

Software Testing Life Cycle đề cập đến một quy trình Test (Testing process) trong đó các bước cụ thể được thực hiện theo một trình tự nhất định để đảm bảo mục tiêu chất lượng được đáp ứng. Trong quy trình kiểm thử phần mềm mỗi hoạt động được thực hiện một cách có kế hoạch và hệ thống. Mỗi một giai đoạn có các mục tiêu khác nhau.



Hình 1.1: Quy trình kiểm thử phần mềm

1. Phân tích yêu cầu: thu thập, phân tích yêu cầu từ khách hàng
2. Lập kế hoạch kiểm thử: nhằm xác định các mục tiêu, phạm vi và chiến lược kiểm thử.
3. Phát triển kịch bản kiểm thử
4. Thiết lập môi trường thử nghiệm: bắt đầu cùng với giai đoạn phát triển kịch bản kiểm thử.
5. Thực hiện kiểm thử: thực hiện các ca kiểm thử đã thiết kế và ghi lại kết quả của mỗi ca kiểm thử.
6. Đóng chu trình kiểm thử: báo cáo kết thúc kiểm thử

### **1.1.3. Các loại kiểm thử**

#### **1.1.3.1. Kiểm thử chức năng**

Kiểm thử chức năng là xác minh hệ thống hoạt động theo đúng theo các yêu cầu nghiệp vụ. Hình thức kiểm thử này có thể được thực hiện từ hai khía cạnh: dựa trên yêu cầu (requirements-based) và dựa trên quy trình nghiệp vụ (business – process – based).

Kiểm thử chức năng bao gồm 5 bước:

* Xác định các chức năng mà phần mềm sẽ thực hiện.
* Tạo các dữ liệu đầu vào dựa trên các tài liệu đặc tả kỹ thuật của các chức năng.
* Xác định các kết quả đầu ra dựa trên các tài liệu đặc tả kỹ thuật của các chức năng.
* Thực hiện các trường hợp kiểm thử.
* So sánh kết quả thực tế và kết quả mong muốn.

Ưu điểm của kiểm thử chức năng:

* Hình thức kiểm thử này mô phỏng việc sử dụng hệ thống thực tế
* Được thực hiện trong các điều kiện gần với điều kiện của khách hàng
* Không có giả định nào về cấu trúc hệ thống được đưa ra trong khi kiểm thử chức năng
* Rất dễ dàng để thực hiện test thủ công

Ngược lại, kiểm thử chức năng có những giới hạn sau:

* Khả năng cao xảy ra tình trạng test dư thừa
* Các lỗi logic trong phần mềm có thể bị bỏ sót trong khi kiểm thử chức năng

Kiểm thử phần mềm có nhiều hình thức khác nhau và được phân loại theo một số tiêu chí.

#### **1.1.3.2 Kiểm thử phi chức năng**

Kiểm thử phi chức năng là kiểm tra các đặc tính chất lượng của hệ thống. Ví dụ, kiểm tra xem bao nhiêu người có thể đăng nhập đồng thời vào một phần mềm. Kiểm tra phi chức năng cũng quan trọng không kém như kiểm tra chức năng và ảnh hưởng đến sự hài lòng của khách hàng.

Tương tự, kiểm thử phi chức năng cũng được chia thành các loại:

* Kiểm thử độ ổn định (Stability testing): đánh giá phần mềm có thể liên tục hoạt động tốt trong hoặc ngay trên khoảng thời gian có thể chấp nhận hay không
* Kiểm thử khả năng chịu tải (Load testing): đánh giá hoạt động của hệ thống khi khối lượng công việc ngày càng tăng
* Kiểm thử áp lực (Stress testing): ước tính hoạt động của hệ thống ở trong hoặc vượt quá giới hạn khối lượng công việc dự kiến
* Kiểm thử tính khả dụng (Usability testing): sản phẩm được test về tính thân thiện với người dùng
* Kiểm thử bảo trì (Maintainability testing): kiểm tra mức độ đánh giá, thay đổi và test sản phẩm
* Kiểm thử độ tin cậy (Reliability testing): sử dụng công cụ để tìm, ngăn chặn và loại bỏ lỗi trước khi hệ thống được triển khai
* Kiểm thử tính tương thích (Portability testing): xác định mức độ dễ dàng hoặc khó khăn mà phần mềm có thể di chuyển từ môi trường này sang môi trường khác

#### **1.1.3.3. Kiểm thử cấu trúc**

Kiểm thử cấu trúc thường được gọi là “hộp trắng” hoặc “hộp thủy tinh” bởi vì phương pháp này quan tâm đến việc tìm kiếm những gì đang xảy ra bên trong, kiểm tra dựa trên phân tích cấu trúc bên trong của thành phần hoặc hệ thống. Nó thường được sử dụng như một cách đo lường của kiểm thử, thông qua độ bao phủ của một tập hợp các yếu tố cấu trúc. Kiểm thử cấu trúc chủ yếu được áp dụng ở kiểm thử thành phần, kiểm thử tích hợp.

Các mục tiêu chính của kiểm thử cấu trúc bao gồm:

* Nhận ra những điểm bất cập
* Test chức năng bổ sung
* Xác định những phần bị thiếu trong bộ kiểm thử

Ưu điểm của kiểm thử cấu trúc:

* Loại bỏ code chết
* Có khả năng tìm ra lỗi ở giai đoạn đầu
* Đảm bảo kiểm tra phần mềm kỹ lưỡng hơn
* Tiết kiệm thời gian
* Bên cạnh đó, nhược điểm của kiểm thử cấu trúc:
* Kiểm tra kết cấu khá tốn kém
* Yêu cầu kiến thức về code
* Đòi hỏi kiến thức vững chắc về công cụ được sử dụng để test

Kiểm thử cấu trúc đảm bảo kiểm tra kỹ lưỡng hơn và tiết kiệm thời gian.

#### **1.1.3.4 Kiểm thử liên quan đến các thay đổi**

* Kiểm thử xác nhận (Confirmation testing)

Khi kiểm thử gặp lỗi, Tester phải xác định nguyên nhân lỗi là do lỗi phần mềm. Sau khi Tester phát hiện lỗi và báo cho Developer để sửa thì phần mềm

sau đó sẽ cập nhật phiên bản vá lỗi. Cuối cùng, Tester cần thực hiện kiểm tra thêm một lần nữa để xác định rằng lỗi thực sự đã được giải quyết.

Khi thực hiện kiểm tra xác nhận, điều quan trọng nhất là phải đảm bảo rằng các trường hợp kiểm thử phải được thực hiện chính xác giống như lần đầu tiên, sử dụng cùng một đầu vào, dữ liệu và môi trường kiểm thử để đảm bảo rằng các lỗi đã được sửa. Tester cần phải biết rằng trong lần kiểm thử sau khi vá lỗi khả năng sinh ra lỗi khác trong phần mềm là điều hoàn toàn có thể xảy ra. Vì vậy kiểm thử chính xác ở phiên bản hiện tại của phần mềm là chưa đủ. Cách phát hiện các điểm ngoài ý muốn của việc kiểm lỗi là thực hiện kiểm thử hồi quy.

* Kiểm thử hồi quy (Regression testing)

Tương tự như kiểm thử xác nhận thì kiểm thử hồi quy liên quan đến việc lặp lại các trường hợp kiểm thử đã được thực hiện trước đó. Kiểm thử hồi quy được thực hiện khi phần mềm thay đổi do sửa lỗi, chức năng mới.

Mục đích của kiểm thử hồi quy để xác minh rằng các sửa đổi trong phần mềm hoặc môi trường không gây ra bất lợi ngoài ý muốn, ảnh hưởng hoặc làm hư các chức năng và hệ thống vẫn đáp ứng các yêu cầu của phần mềm. Tất cả các trường hợp trong quá trình kiểm thử hồi quy sẽ được thực hiện mỗi khi một phiên bản vá lỗi của phần mềm được release, và điều này khiến chúng trở nên lý tưởng cho tự động hóa.

### **1.1.4. Các phương pháp kiểm thử**

#### **1.1.4.1. Kiểm thử hộp đen**

Kiểm thử hộp đen hay còn gọi là kiểm thử hướng dữ liệu. Là phương pháp kiểm thử không quan tâm đến cấu trúc bên trong của phần mềm (cấu trúc dữ liệu, thuật toán, xử lý…) mà chỉ kiểm tra dữ liệu đầu vào, dữ liệu đầu ra ứng với các trường hợp trong đặc tả (Testcase).

Kiểm thử hộp đen không có “mối ràng buộc” nào với code và nhận thức của một tester rất đơn giản: một source code có rất nhiều lỗi => Các tester sẽ tìm thấy được nhiều lỗi nơi mà Developer không tìm thấy.

#### **1.1.4.2. Kiểm thử hộp trắng**

Kỹ thuật kiểm thử hộp trắng hay còn gọi là “kiểm thử cấu trúc” là kỹ thuật kiểm thử dựa vào giải thuật cụ thể, cấu trúc dữ liệu bên trong của đơn vị phần mềm cần kiểm thử để xác định đơn vị phần mềm đó có thực hiện đúng hay không?

Tiêu chuẩn của kiểm thử hộp trắng phải đáp ứng các yêu cầu như sau:

* Bao phủ dòng lệnh: mỗi dòng lệnh ít nhất phải được thực thi 1 lần.
* Bao phủ nhánh: mỗi nhánh trong sơ đồ điều khiển phải được đi qua một lần.
* Bao phủ đường: tất cả các đường từ điểm khởi tạo đến điểm cuối cùng trong sơ đồ dòng điều khiển phải được đi qua.

Ưu/ Nhược điểm:

* Ưu điểm: Buộc các chuyên gia kiểm thử phải suy luận cẩn thận về việc kiểm thử lỗi vì vậy lỗi sẽ được triệt để, cho phép tìm kiếm các lỗi ẩn bên trong. Do yêu cầu kiến thức cấu trúc bên trong của phần mềm, nên việc kiểm soát lỗi tối đa nhất.
* Nhược điểm: Khá mất thời gian và công sức nhưng vẫn sẽ tồn tại lỗi. Đòi hỏi người kiểm thử có kinh nghiệm và am hiểu về kiểm thử cũng như về cấu trúc bên trong của phần mềm được thử nghiệm.

#### **1.1.4.3. Kiểm thử hộp xám**

Kiểm thử hộp xám là một phương pháp kiểm thử phần mềm được kết hợp giữa phương pháp kiểm thử hộp đen và kiểm thử hộp trắng. Trong kiểm thử hộp đen tester kiểm thử các hạng mục mà không cần biết cấu trúc bên trong của chương trình, còn kiểm thử hộp trắng thì tester biết được cấu trúc bên trong của chương trình. Trong kiểm thử hộp xám, cấu trúc bên trong sản phẩm chỉ được biết một phần. Tester có thể truy cập vào cấu trúc dữ liệu bên trong và thuật toán của chương trình với mục đích là để thiết kế các trường hợp kiểm thử, nhưng khi thực hiện kiểm thử thì test như người dùng cuối hoặc là ở mức hộp đen.

Kiểm thử hộp xám nhằm tìm ra tối đa số lỗi về cấu trúc dữ liệu của hộp trắng và lỗi chức năng của hộp đen. Trong kiểm thử hộp xám viết các trường hợp kiểm thử dựa vào yêu cầu và nội dung Source Code (can thiệp vào bên trong Code của chương trình). Thực hiện kiểm thử trên giao diện của chương trình (yêu cầu chương trình phải chạy được mới kiểm thử được, không can thiệp vào code).

Ưu/ Nhược điểm:

* Ưu điểm: Hoạt động tốt cho các đoạn mã lớn, các vai trò được xác định rõ ràng cho người dùng và nhà phát triển trong quá trình thử nghiệm. Thử nghiệm dựa trên yêu cầu của người dùng chứ không phải người thiết kế.
* Nhược điểm: Hầu hết các trường hợp kiểm thử đều khó thiết kế. Kiểm thử hộp xám không được coi là một phương pháp hiệu quả vì không có nhiều kịch bản để kiểm thử

### **1.1.5. Thiết kế test case**

Test case mô tả một dữ liệu đầu vào (input), hành động (action) hoặc sự kiện (event) và một kết quả mong đợi (expected response), để xác định một chức năng của ứng dụng phần mềm hoạt động đúng hay không.

Mục đích sử dụng:

* Tạo ra các ca kiểm thử tốt nhất có khả năng phát hiện ra lỗi, sai sót của phần mềm một cách nhiều nhất.
* Tạo ra các ca kiểm thử có chi phí rẻ nhất, đồng thời tốn ít thời gian và công sức nhất.
* Giúp người tester ít kinh nghiệm có thể biết những việc cần làm để đảm bảo chất lượng của phần mềm.
* Giúp nhà quản lý dự án biết được chức năng nào có lỗi, chức năng nào lỗi nhiều, lỗi ít.

Cấu trúc một test case thường bao gồm các thông tin:

* Test Case ID (Mã và tên của test case): Giá trị cần để xác định số lượng trường hợp cần để kiểm thử.
* Test Items (Mục đích kiểm thử): Mô tả mục đích sử dụng của test case. Giúp Tester hiểu và thực hiện đúng khi kiểm thử phần mềm theo test case mô tả.
* Pre-condition (Điều kiện tiên quyết): Mô tả điều kiện cần có để có thể thực hiện test case này.
* Test Steps (Mô tả các bước): Mô tả cụ thể các bước cần thực hiện để tái hiện nội dung test case khi Tester thực hiện kiểm thử phần mềm.
* Test Data (Dữ liệu đầu vào): Là dữ liệu nhập vào các trường của phần mềm để thực hiện kiểm thử.
* Expected results (Kết quả mong đợi): Một test case được viết tốt cần phải đề cập một cách rõ ràng kết quả mong đợi của ứng dụng hoặc hệ thống. Chỉ ra những gì mong đợi như là đầu ra của bước kiểm tra đó.
* Actual results (Kết quả thực tế): là kết quả thực kế của một test case.
* Trạng thái: Là trạng thái để đánh giá một test case có đạt hay không. Nó bao gồm các trạng thái: pass/ fail/ pending.

#### **1.1.5.1. Các bước viết một test case**

Bước 1: Xác định mục đích test, cần hiểu rõ đặc tả yêu cầu của khách hàng.

Bước 2: Xác định chức năng testing, cần phải biết làm thế nào phần mềm được sử dụng bao gồm các hoạt động, tổ chức chức năng khác nhau. Các bước thực hiện chỉ mô tả các bước thực hiện đứng từ phía người dùng cuối bao gồm nhập dữ liệu, nhấn button.

Bước 3: Xác định các yêu cầu phi chức năng, yêu cầu phần cứng, hệ điều hành, các khía cạnh an ninh.

Bước 4: Xác định biểu mẫu cho test case, bao gồm giao diện UI, chức năng, khả năng tương thích và hiệu suất…

Bước 5: Xác định tính ảnh hưởng giữa các nguyên tắc module test case nên được thiết kế để có thể bao phủ được sự ảnh hưởng của các module với nhau ở mức độ cao nhất.

#### **1.1.5.2. Các kỹ thuật viết test case**

**a) Phân vùng tương đương**

Ý tưởng: Phân vùng tương đương là phương pháp chia các điều kiện đầu vào thành những vùng tương đương nhau. Tất cả các giá trị trong một vùng tương đương sẽ cho một kết quả đầu ra giống nhau. Vì vậy chúng ta có thể kiểm tra một giá trị đại diện trong vùng tương đương.

Thiết kế test case bằng kỹ thuật phân vùng tương đương tiến hành theo 2 bước:

* Bước 1: Xác định các lớp tương đương.
* Bước 2: Xác định các ca kiểm thử.

Ưu/ Nhược điểm của kỹ thuật phân vùng tương đương:

* Ưu điểm: Vì mỗi vùng tương đương ta chỉ cần test trên các phần tử đại diện nên số lượng test case được giảm đi khá nhiều nhờ đó mà thời gian thực hiện test cũng giảm đáng kể.
* Nhược điểm: Không phải với bất kỳ bài toán nào đều có thể áp dụng kỹ thuật này. Có thể bị sót lỗi ở biên nếu chỉ chọn giá trị ở khoảng giữa của miền tương đương. Vì vậy việc kết hợp linh hoạt giữa kỹ thuật phân vùng tương đương và phân tích giá trị biên dưới đây sẽ mang lại hiệu quả cao hơn để vừa tối ưu số lượng test case và vẫn đảm bảo đươc chất lượng phần mềm.

**b) Phân tích giá trị biên**

Ý tưởng: Phân tích giá trị biên là trường hợp đặc biệt của phân vùng tương đương, dựa trên những phân vùng tương đương tester sẽ xác định giá trị biên giữa những phân vùng này và lựa chọn test case phù hợp.

Các case chuẩn được lựa chọn dựa vào quy tắc sau:

* Giá trị biên nhỏ nhất ± 1
* Giá trị biên nhỏ nhất
* Giá trị biên lớn nhất
* Giá trị biên lớn nhất ± 1

Ưu/ Nhược điểm của kỹ thuật phân tích giá trị biên:

* Ưu điểm: Thay vì phải kiểm tra hết toàn bộ các giá trị trong từng vùng tương đương, kỹ thuật phân tích giá trị biên tập trung vào việc kiểm thử các giá trị biên của miền giá trị inputs để thiết kế test case do “lỗi thường tiềm ẩn tại các ngõ ngách và tập hợp tại biên”. Do đó, tiết kiệm thời gian thiết kế test case và thực hiện test.
* Nhược điểm: Phương pháp này chỉ hiệu quả trong trường hợp các đối số đầu vào độc lập với nhau và mỗi đối số đều có một miền giá trị hữu hạn.

**c) Bảng quyết định**

Bảng quyết định còn được gọi là bảng Nguyên nhân – Kết quả (Cause-Effect). Kỹ thuật kiểm thử phần mềm này được sử dụng cho các chức năng cần sự kết hợp của các yếu tố đầu vào các biến.

Đối với mọi chức năng, cần tạo một bảng và liệt kê tất cả các loại kết hợp đầu vào và đầu ra tương ứng. Điều này giúp xác định các điều kiện bị tester bỏ qua.

Các bước để tạo bảng quyết định:

* Nhập đầu vào theo hàng.
* Nhập tất cả các quy tắc trong cột.
* Điền vào bảng với sự kết hợp của đầu vào.
* Trong hàng cuối cùng, ghi chú đầu ra so với kết hợp đầu vào.

Ví dụ: Bảng quyết định các hành vi hệ thống cho form đăng nhập như sau:

Bảng 1.1: Bảng quyết định

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Role 1 | Role 2 | Role 3 | Role 4 |
| Email | T | T | F | F |
| Password | T | F | T | F |
| Submit | H | Error | Error | Error |

Gồm 2 trường đầu vào: Email và Password

Chú thích:

* T: Email hoặc Password đúng
* F: Email hoặc Password sai
* H: Đăng nhập thành công, chuyển đến màn hình trang chủ
* Error: Hiển thị thông báo lỗi, đăng nhập không thành công

**d) Đoán lỗi**

Ý tưởng: Phương pháp này không có quy trình cụ thể vì có tính trực giác cao và không thể dự đoán trước. Phương pháp chỉ phù hợp với những Tester có kinh nghiệm. Họ phỏng lỗi phần mềm dựa vào trực giác, dựa vào kinh nghiệm, dữ liệu lịch sử về các lỗi đã từng xảy ra với chương trình trước đó… và sau đó viết các ca kiểm thử để đưa ra các lỗi đó.

Ưu/ Nhược điểm:

* Ưu điểm: Sử dụng phương pháp này có thể giúp tester tìm ra những lỗi điển hình thường xảy ra trong phần mềm hoặc những lỗi không thể tìm thấy khi thiết kế test case theo hình thức formal.
* Nhược điểm: Kỹ thuật này thường được thực hiện bởi các Tester có kinh nghiệm và không theo một quy tắc nhất định, thiết kế test case dựa nhiều vào cảm tính.

Để giảm thiểu số case đến mức tối ưu mà vẫn đảm bảo chất lượng phần mềm, mỗi tester cần linh hoạt trong việc lựa chọn các kỹ thuật thiết kế các trường hợp kiểm thử.

## **1.2. Khái quát về kiểm thử tự động**

Kiểm thử phần mềm tốn nhiều chi phí, nhân lực và thời gian. Trong một số dự án, chi phí kiểm thử phần mềm chiếm 40% tổng giá trị của dự án. Do đó một trong các mục tiêu của kiểm thử là tự động hóa nhiều, nhờ đó mà giảm thiểu chi phí, giảm lỗi, đặc biệt giúp việc kiểm thử dễ dàng và nhanh chóng hơn.

### **1.2.1. Khái niệm kiểm thử tự động**

Kiểm thử tự động là quá trình xử lý một cách tự động các bước thực hiện các test case. Kiểm thử tự động bằng một công cụ nhằm rút ngắn thời gian kiểm thử.

Lợi ích của kiểm thử tự động:

* Tiết kiệm thời gian và tiền bạc
* Chính xác hơn
* Độ bao phủ cao

### **1.2.2. Quy trình kiểm thử tự động**

Quy trình kiểm thử tự động bao gồm các bước sau:

* Lập kế hoạch kiểm thử
* Thiết kế test case
* Phát triển test script
* Thực hiện kiểm thử
* Đánh giá kết quả kiểm thử

#### **1.2.2.1. Lập kế hoạch kiểm thử**

Mục đích: Nhằm chỉ định và mô tả các loại kiểm thử sẽ được triển khai và thực hiện. Kết quả của bước lập kế hoạch là bản tài liệu kế hoạch kiểm thử phần mềm, bao gồm:

* Các giai đoạn kiểm thử áp dụng cho dự án phần mềm
* Chiến lược kiểm thử
* Các công cụ kiểm thử
* Nguồn lực kiểm thử
* Môi trường kiểm thử, bao gồm tài nguyên phần cứng và phần mềm
* Mốc bàn giao các tài liệu kiểm thử

#### **1.2.2.2. Thiết kế test case**

Mục đích: Nhằm xác định các test case và các bước kiểm tra chi tiết cho mỗi phiên bản phần mềm. Giai đoạn thiết kế test case là hết sức quan trọng, nó đảm bảo tất cả các tình huống kiểm thử “quét” hết tất cả các yêu cầu cần kiểm tra.

#### **1.2.2.3. Phát triển test script**

Mục đích: Bước này thường không bắt buộc trong các loại và mức kiểm thử, chỉ yêu cầu trong những trường hợp đặc thù cần thiết, tạo ra các test script có khả năng chạy trên máy tính giúp tự động hóa việc thực thi các bước kiểm tra đã định nghĩa ở bước thiết kế test case.

Trong đó một test script được hiểu là một nhóm mã lệnh dạng đặc tả kịch bản dùng để tự động hóa một trình tự kiểm thử, giúp cho việc kiểm thử nhanh hơn, hoặc cho những trường hợp mà kiểm thử bằng tay sẽ rất khó khăn hoặc không khả thi. Các test script có thể tạo thủ công hoặc tạo tự động dùng công cụ kiểm thử tự động.

#### **1.2.2.4. Thực hiện kiểm thử**

Mục đích: Thực hiện các bước kiểm tra đã thiết kế hoặc thực thi các test script nếu tiến hành kiểm tra tự động và ghi nhận kết quả.

#### **1.2.2.5. Đánh giá kết quả kiểm thử**

Mục đích: Đánh giá toàn bộ quá trình kiểm thử, bao gồm xem xét và đánh giá kết quả kiểm thử, liệt kê lỗi, chỉ định các yêu cầu thay đổi, và tính toán các số liệu liên quan đến quá trình kiểm thử như số giờ, thời gian kiểm thử, số lượng lỗi, phân loại lỗi…

### **1.2.3. So sánh kiểm thử thủ công và kiểm thử tự động**

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Kiểm thử thủ công** | **Kiểm thử tự động** |
| Định nghĩa | Testcase được thực hiện thủ công bởi tester | Tester phải viết test script và lựa chọn công cụ để tự động hóa việc test |
| Thời gian xử lý | Cần nhiều thời gian và nhân lực | Thời gian kiểm thử nhanh hơn so với manual testing |
| Kiểm thử khám phá | Kiểm thử khám phá được thực hiện | Không cho phép kiểm thử khám phá |
| Thay đổi UI | Sự thay đổi nhỏ như ID, Class hoặc 1 button nhưng không ảnh hưởng đến thực thi test | Chỉ 1 vài thay đổi nhỏ trong UI, người dùng phải update script để đảm bảo có kết quả như mong đợi |
| Độ tin cậy | Kết quả kiểm thử không đáng tin cậy vì có khả năng xảy ra lỗi do con người | Do được thực thi bằng tool và scripts nên kết quả đáng tin cậy hơn |
| Đầu tư | Cần nhiều nguồn nhân lực | Bắt buộc phải đầu tư tool để test và những kỹ sư auto |
| Báo cáo | Manual test thường lưu lại kết quả ở Excel, Word, … | Tất cả stakeholders có thể đăng nhập vào hệ thống auto và kiểm tra lại kết quả test |
| Sự quan sát của con người | Cần có sự quan sát của con người để giúp cho hệ thống thân thiện với người dùng | Không có sự quan sát của con người |
| Kiểm thử hiệu năng | Không thực hiện được Kiểm thử hiệu năng | Kiểm thử hiệu năng phải được thực hiện bởi 1 tool phù hợp |
| Kiến thức lập trình | Không cần có khả năng code | Phải có kiến thức về lập trình để tạo ra các test script |
| Cách tiếp cận tốt | Manual testing hữu ích khi chúng ta chạy lại bộ testcase 1 hoặc 2 lần | Auto test rất hữu ích khi ta chạy lại bộ script nhiều lần |
| Sử dụng khi nào? | Kiểm thử thủ công phù hợp cho test khám phá, khả năng sử dụng và kiểm thử dựa vào thực tế | Test auto thích hợp cho kiểm thử hồi quy, hiệu năng hoặc các trường hợp có khả năng lặp lại nhiều lần |

### **1.2.4. Một số công cụ kiểm thử tự động phổ biến**

Selenium

QuickTest Professional (QTP)

Test Complete

LoadTest

Jmeter

Visual Studio

…

# **NGHIÊN CỨU CÔNG CỤ KIỂM THỬ TỰ ĐỘNG SELENIUM WEBDRIVER**

## **2.1. Công cụ kiểm thử tự động Selenium**

### **2.1.1. Khái quát về Selenium**

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, 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.

### **2.1.2. 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. Mặt 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 không có. Thực sự 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 nhau 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.

### **2.1.3. Các thành phần của Selenium**

Selenium là một bộ công cụ tự động hóa kiểm thử phần mềm trên web, được phát triển bởi Jason Huggins vào năm 2004 khi ông làm việc tại ThoughtWorks. Ban đầu, Selenium được tạo ra để tự động kiểm thử ứng dụng web của mình. Từ đó, Selenium đã trải qua nhiều giai đoạn phát triển và cải tiến, trở thành công cụ kiểm thử tự động phổ biến nhất cho các ứng dụng web.

Selenium bao gồm 4 thành phần: Selenium IDE, Selenium RC, Selenium Grid và 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):**

Ra mắt năm 2006

Cho phép viết kiểm thử bằng nhiều ngôn ngữ lập trình khác nhau như Java, C#, Perl, Python và Ruby

Là môi trường phát triển tích hợp cho việc xây dựng trường hợp kiểm thử Selenium. Nó hoạt động như một add-on của Firefox và cung cấp một giao diện dễ sử dụng để phát triển và chạy trường hợp kiểm thử. Selenium IDE có tính năng thu lại kịch bản kiểm thử để tái sử dụng. Selenium IDE cũng cung cấp các chức năng chỉnh sửa các trường hợp kiểm thử chính xác và dễ kiểm soát hơn. Mặc dù Selenium IDE chỉ là một Firefox add-on, nhưng các test case tạo ra bằng Selenium IDE vẫn có thể chạy trên các trình duyệt khác bằng cách sử dụng Selenium RC.

IDE đã ngừng phát triển vào tháng 8 năm 2017 khi Firefox nâng cấp lên phiên bản Firefox 55 mới, không còn hỗ trợ Selenium IDE. Applitools đã viết lại Selenium IDE cũ và phát hành phiên bản mới vào năm 2019. Phiên bản mới nhất có một số cải tiến.

Selenium IDE vẫn còn gặp phải một số thiếu sót sau:

* Không hỗ trợ kiểm thử theo hướng dữ liệu
* Không thực hiện được kiểm tra cơ sở dữ liệu
* Không cung cấp được báo cáo kiểm thử chi tiết
* Không thể xuất sang tập lệnh Webdriver
* **Selenium RC (Remote Control):**

Phát hành vào năm 2004

Là một framework kiểm thử cho phép thực hiện nhiều hơn và tuyến tính các hành động trên trình duyệt. Nó 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 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:**

Được giới thiệu vào năm 2006 và tích hợp chính thức vào năm 2011

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.

WebDriver không yêu cầu một công cụ cốt lõi như RC và tương tác tự nhiên với các ứng dụng trình duyệt. WebDriver cũng hỗ trợ nhiều ngôn ngữ lập trình khác nhau như Python, Ruby, PHP và Perl, trong số những ngôn ngữ khác và có thể được tích hợp với các framwork như TestNG và JUnit để quản lý kiểm thử.

Kiến trúc của Selenium WebDriver rất đơn giản và dễ hiểu:

* Selenium test script: Tập lệnh kiểm thử Selenium được viết bằng bất kỳ ngôn ngữ lập trình nào cí thể là Java, Perl, PHP hoặc Python mà trình điều khiển có thể hiểu được.
* JSON Wire Protocol: Cung cấp cơ chế vận chuyển để truyền dữ liệu giữa máy chủ và máy khách. Giao thức này phục vụ như một tiêu chuẩn công nghiệp cho các dịch vụ Web khác nhau
* Browser Driver: Selenium sử dụng driver dành riêng cho từng trình duyệt để thiết lập kết nối an toàn.
* Browsers: Selenium WebDriver hỗ trợ các trình duyệt web khác nhau để kiểm thử và chạy các ứng dụng trên đó.
* **Selenium-Grid:**

Phát hành lần đầu vào năm 2008

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.

## **2.2. Những vấn đề mới của Selenium**

* **Khả năng tương thích với trình duyệt:**

Các Bản Cập Nhật Thường Xuyên Của Trình Duyệt: Các trình duyệt web như Chrome, Firefox, và Edge được cập nhật thường xuyên, dẫn đến việc Selenium phải liên tục cập nhật để duy trì sự tương thích. Đôi khi, các cập nhật này có thể phá vỡ các bài kiểm thử hiện có.

Trình Duyệt Mới: Sự xuất hiện của các trình duyệt mới hoặc các phiên bản trình duyệt trên thiết bị di động yêu cầu Selenium phải cập nhật và mở rộng khả năng hỗ trợ.

* **Hiệu suất và độ ổn định:**

Hiệu Suất Kiểm Thử: Khi kiểm thử các ứng dụng web phức tạp hoặc có giao diện người dùng động, hiệu suất của Selenium có thể bị ảnh hưởng, dẫn đến thời gian kiểm thử kéo dài.

Độ Ổn Định: Các bài kiểm thử không ổn định có thể xảy ra do vấn đề đồng bộ hóa giữa các hành động kiểm thử và trạng thái của ứng dụng web. Điều này đặc biệt phổ biến với các ứng dụng có nhiều tương tác AJAX hoặc các thành phần động.

* **Hỗ trợ Mobile:**

Thiết Lập và Bảo Trì Phức Tạp: Dù Selenium có thể tích hợp với Appium để kiểm thử ứng dụng di động, việc thiết lập và bảo trì môi trường kiểm thử di động vẫn phức tạp và đòi hỏi nhiều công sức.

Hạn Chế Trong Kiểm Thử Di Động: Selenium chủ yếu được thiết kế cho kiểm thử trình duyệt web trên máy tính, do đó, việc kiểm thử trên các thiết bị di động có thể gặp một số hạn chế và thách thức.

* **Đồng bộ hóa:**

Vấn Đề Đồng Bộ: Selenium đôi khi gặp khó khăn trong việc đồng bộ hóa các lệnh kiểm thử với trạng thái của ứng dụng web, dẫn đến các lỗi không ổn định và khó tái hiện. Việc xử lý các yếu tố không đồng bộ như AJAX, thời gian tải trang không đồng đều, hoặc các thành phần động là một thách thức liên tục.

* **Cấu hình và bải trì môi trường kiểm thử:**

Cấu Hình Phức Tạp: Thiết lập và cấu hình môi trường kiểm thử Selenium Grid để kiểm thử phân tán và song song có thể phức tạp và yêu cầu kiến thức kỹ thuật sâu.

Bảo Trì Môi Trường: Việc duy trì và cập nhật các node và hub trong Selenium Grid để đảm bảo chúng hoạt động hiệu quả và tương thích với các bản cập nhật mới của Selenium và trình duyệt là một thách thức liên tục.

* **Bảo mật:**

Vấn Đề Bảo Mật: Khi thực hiện kiểm thử trên môi trường phân tán, đặc biệt là trên các cloud service hoặc các môi trường không kiểm soát, bảo mật dữ liệu và bảo mật thông tin là một mối quan tâm quan trọng.

## **2.3. Selenium Webdriver**

### **2.3.1. Tiền thân của Selenium WebDriver**

Trước khi Selenium WebDriver ra đời và phát triển thì Selenium RC là công cụ chính trong suốt một thời gian dài. Hiện nay, Selenium RC không được sử dụng nhiều như Selenium WebDriver nữa, tuy nhiên người dùng vẫn có thể tiếp tục phát triển các kịch bản kiểm thử với Selenium RC.

Selenium RC là công cụ phục vụ cho các công việc kiểm thử đòi hỏi nhiều hơn việc thao tác với các website trên giao diện. Nó cho phép viết các kịch bản kiểm thử tự động ứng dụng Web với sự hỗ trợ của các ngôn ngữ lập trình như Java, C#, Python, Perl, PHP để tạo ra các trường hợp kiểm thử phức tạp hơn như đọc và viết các tệp tin, truy vấn cơ sở dữ liệu, gửi mail kết quả kiểm thử.

Các thành phần của Selenium RC gồm:

* Máy chủ Selenium: Thực hiện phân tích và chạy các lệnh được gửi đến từ ứng dụng cần kiểm thử và các thao tác như HTTP proxy, phân tích và xác minh các thông điệp HTTP, giữa trình duyệt và ứng dụng cần kiểm tra.
* Các thư viện máy khách: Cung cấp sự hỗ trợ lập trình cho phép chạy lệnh Selenium từ chương trình. Các thư viện máy khách hỗ trợ cho các ngôn ngữ lập trình khác nhau thì khác nhau. Giao diện lập trình là tập các chức năng chạy lệnh Selenium, trong mỗi giao diện có một chức năng lập trình hỗ trợ Selenium.

### **2.3.2. Đặc trưng của Selenium WebDriver**

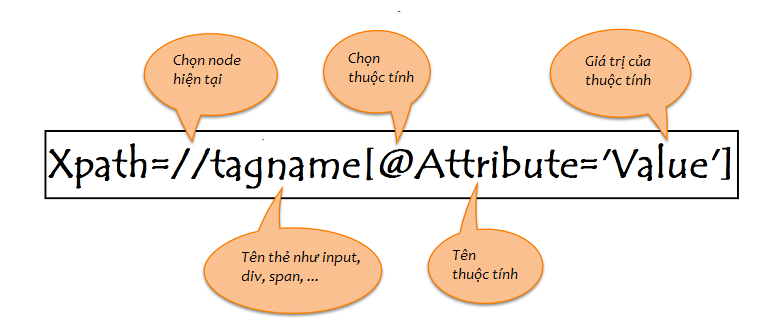
Selenium WebDriver (hay còn gọi là Selenium 2.0) kế thừa và phát triển từ Selenium IDE, Selenium RC, Selenium Grid. Selenium WebDriver tương tác trực tiếp với trình duyệt mà không cần thông qua bất kỳ trung gian, không giống như Selenium RC phụ thuộc vào một máy chủ.

Tính năng chính trong Selenium 2.0 là việc tích hợp WebDriver API. Ngoài việc giải quyết một số hạn chế trong Selenium RC API, Selenium WebDriver còn được thừa kế để mang đến một giao diện lập trình đơn giản hơn. Nó cho phép sử dụng một trong số các ngôn ngữ lập trình như HTML, Java, .Net, Perl, Ruby, … để tạo kịch bản kiểm thử kết hợp với sử dụng các điều kiện, vòng lặp, … khiến cho test script trở nên chính xác hơn.

Selenium WebDriver còn được phát triển tốt hơn để hỗ trợ cho các trang web động do các phần tử trong một trang web động có thể thay đổi bất cứ lúc nào, ngay cả khi trang đó không được tải lại. Vì vậy, Selenium WebDriver được phát triển để hỗ trợ quá trình kiểm thử mà không cần phải thực hiện lại khi có thay đổi xảy ra.

### **2.3.3.Cách lấy xpath của element trên website**

Xpath được sử dụng để xác định các phần tử web dựa trên path XML của nó. Đây được xem là kỹ thuật hiệu quả nhất, có thể xác định tất cả các phần tử trên trang web thường là các element không định vị được bằng id, name, class. Tuy nhiên khi có sự thay đổi trong cấu trúc trang thì bạn sẽ cần phải update lại thông tin các element liên quan.



Có 2 loại Xpath: Xpath tuyệt đối và Xpath tương đối.

* Xpath tuyệt đối:
  + Xpath tuyệt đối bắt đầu bằng dấu gạch chéo đơn "/", cho phép xác định một đường dẫn tuyệt đối đến đối tượng UI.
  + Xpath tuyệt đối được xem là là cách tìm kiếm phần tử dễ dàng nhất, tuy nhiên nhược điểm của nó là nếu có bất kỳ thay đổi nào trên đường dẫn của element thì Xpath lấy sẽ sai.
* Xpath tương đối: Xpath tương đối bắt đầu bằng 2 dấu gạch chéo "//", cho phép xác định một đối tượng UI ở bất kỳ đâu trên trang web, không cần bắt đầu bởi thẻ html trong đường dẫn.

Cách lấy Xpath từ Inspect của Chrome:

* Bước 1: Chạy và khởi động trình duyệt.
* Bước 2: Trỏ chuột vào vị trí muốn lấy. Sau đó, bấm chuột phải rồi chọn Inspect.
* Bước 3: Kiểm tra lại xem vị trí đã được xác định đúng chưa. Sau đó, bấm chuột phải rồi chọn Copy và chọn Copy Xpath.

### **2.3.4. Các câu lệnh sử dụng trong Selenium WebDriver**

#### **2.3.4.1 Các câu lệnh trình duyệt**

Selenium WebDriver có một số các câu lệnh thao tác với trình duyệt như mở, đóng, lấy tiêu đề của trang Web như dưới đây:

**a) Câu lệnh get**

- Mục đích: Câu lệnh này sử dụng để mở một trang Web mới trong trình duyệt hiện tại.

- Cú pháp: drive.get(URL);

- Trong đó: URL là url để tải trang, nên sử dụng một url đầy đủ.

- Ví dụ: driver.get(“https://xedaptotnhat.vn”);

**b) Câu lệnh lấy tiêu đề getTitle**

- Mục đích: Câu lệnh này sử dụng để lấy tiêu đề của trang Web hiện tại.

- Cú pháp: drive.getTitle();

**c) Câu lệnh lấy URL hiện tại getCurrentUrl**

- Mục đích: Câu lệnh này sử dụng để lấy URL của trang hiện tại đã được tải trên trình duyệt.

- Cú pháp: drive.getCurrentUrl();

**d) Câu lệnh lấy source của trang Web getPageSource**

- Mục đích: Câu lệnh này sử dụng để lấy source của trang được tải cuối cùng.

- Cú pháp: drive.getPageSource();

**e) Câu lệnh đóng trang web close**

- Mục đích: Câu lệnh này dùng để đóng cửa sổ hiện tại của trình duyệt.

- Cú pháp: driver.close();

**f) Câu lệnh thoát trình duyệt quit**

- Mục đích: Câu lệnh này dùng để thoát khỏi trình duyệt và tất cả các cửa sổ đã mở.

- Cú pháp: driver.quit();

**g) Câu lệnh làm mới refresh**

- Mục đích: Câu lệnh này dùng để làm mới trình duyệt hiện tại.

- Cú pháp: driver.navigate().refresh();

#### **2.3.4.2 Các câu lệnh WebElement**

Để tương tác với một trang web, cần xác định vị trí của phần tử trên trang web, WebDriver cung cấp 2 phương thức “FindElement” và “FindElements” để xác định vị trí của phần tử trên trang web.

Phương thức “FindElement” và “FindElements”

Sự khác nhau giữa phương thức “FindElement” và “FindElements” là trả về đối tượng WebElement, nếu không ném một ngoại lệ và trả về một danh sách WebElement, có thể sẽ trả về danh sách rỗng nếu không cố định phần tử DOM phù hợp với truy vấn. Phương thức “Find” lấy vị trí hoặc đối tượng truy vấn gọi bằng phương thức “By”.

**a) Tìm phần tử bằng ID: By ID**

- Mục đích: Tìm vị trí của phần tử bằng ID, nếu tìm được id phù hợp sẽ trả về vị trí của phần tử, nếu không có phần tử phù hợp với id sẽ xuất hiện NoSuchElementException.

- Cú pháp: drive.findElement(By.id(“”));

- Ví dụ: <input id=”Email”></input>

WebElement element=driver.findElement(By.id(“Email”));

**b) Tìm phần tử bằng Name: By Name**

- Mục đích: Tìm vị trí của phần tử bằng name, nếu tìm được giá trị thuộc tính name phù hợp sẽ trả về vị trí của phần tử, nếu không có phần tử phù hợp với thuộc tính name sẽ xuất hiện NoSuchElementException.

- Cú pháp: drive.findElement(By.name(“”));

- Ví dụ: <input id=”UserID”name=”huong”></input>

WebElement element=driver.findElement(By.name(“huong”));

**c) Tìm phần tử bằng Class Name: By className**

- Mục đích: Tìm phần tử dựa trên giá trị của thuộc tính “class”

- Cú pháp: drive.findElement(By.className(“”));

- Ví dụ: <input class = “test”></input>

WebElement element=driver.findElement(By.className(“test”));

**d) Tìm phần tử bằng Tag Name: By tagName**

- Mục đích: Tìm phần tử dựa vào tag names.

- Cú pháp: drive.findElement(By.tagName(“”));

- Ví dụ: <dt><a href="select\_tag.html">TagSelected</a></dt>

WebElement element=driver.findElement(By.tagName("dt"));

**e) Tìm phần tử bằng Link Text**

- Mục đích: Tìm phần tử của thẻ a bằng tên của link.

- Cú pháp: driver.findElement(By.tagName(""));

- Ví dụ: <a href="link.html">NameoftheLink</a>

WebElement element=driver.findElement(By.linkText("Name of the Link"));

#### **2.3.4.3 Các câu lệnh điều hướng trình duyệt**

**a) Câu lệnh To**

- Mục đích: Lệnh này dùng để chuyển hướng đến trang web hoặc URL yêu cầu.

- Cú pháp: driver.navigate().to(URL)

- Trong đó: URL: là url để tải trang, nên sử dụng url đầy đủ

- Ví dụ: driver.navigate().to("https://xedaptotnhat.vn");

**b) Câu lệnh Forward**

- Mục đích: Câu lệnh này dùng để đi đến trang tiếp theo, giống với nút forward trên trình duyệt.

- Cú pháp: drive.navigate().forward();

**c) Câu lệnh Back**

- Mục đích: Câu lệnh này dùng để quay về trang trước, giống với nút back trên trình duyệt.

- Cú pháp: drive.navigate().back();

**d) Câu lệnh Refresh**

- Mục đích: Câu lệnh này dùng để làm mới trang hiện tại.

- Cú pháp: drive.navigate().refresh();

#### **2.3.4.4 Các câu lệnh Switch**

Một số trang web có nhiều frames hoặc nhiều cửa sổ. Selenium WebDriver gán id cho mỗi cửa sổ ngay khi đối tượng WebDriver được khởi tạo. ID này được gọi là cửa sổ xử lý.

Selenium sử dụng id duy nhất này để điều khiển nhiều cửa sổ. Trong đó, mỗi cửa sổ có một id duy nhất, do đó Selenium có thể phân biệt được khi nó được chuyển sang điều khiển một cửa sổ khác.

**a) Câu lệnh getWindowHandle**

- Mục đích: Câu lệnh này dùng để lấy cửa sổ xử lý của cửa sổ hiện tại.

- Cú pháp: drive.getWindowHandle();

**b) Câu lệnh getWindowHanles**

- Mục đích: Câu lệnh này dùng để lấy cửa sổ xử lý của tất cả các cửa sổ hiện tại.

- Cú pháp: drive.getWindowHandles();

**c) Câu lệnh switch to window**

- Mục đích: Câu lệnh này dùng để hỗ trợ di chuyển giữa các cửa sổ khác thông qua tên của chúng bằng cách sử dụng phương thức “switchTo”.

- Cú pháp: drive.switchTo().window(“windowName”);

#### **2.3.4.5 Các câu lệnh wait**

**a) Câu lệnh implicitlyWait**

- Mục đích: Đợi một thời gian nhất định trước khi ném một ngoại lệ khi không thể tìm thấy các phần tử trên trang web. Lưu ý rằng chờ đời ngầm luôn thực hiện trong suốt thời gian trình duyệt mở. Điều này có nghĩa tìm kiếm bất kỳ các phần tử nào trên trang có thể mất thời gian chờ đợi ngầm cho việc thiết lập câu lệnh này.

- Cú pháp:

drive.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

**b) Câu lệnh FluentWait**

- Mục đích: Xác định số lượng tối đa thời gian để chờ đợi một điều kiện. giống như tần suất để kiểm tra điều kiện xuất hiện. Hơn nữa, người dùng có thể cấu hình để loại bỏ việc chờ đợi bằng cách ném một ngoại lệ trong khi chờ đợi như là NoSuchElementExceptions.

- Cú pháp:

Wait wait=newFluentWait(driver).withTimeout(30, SECONDS)

.pollingEvery(5, SECONDS)

.ignoring(NoSuchElementException.class);

**c) Câu lệnh ExpectedConditions**

- Mục đích: Một điều kiện hợp lý được mong đợi để giá trị cuối cùng không phải là null hoặc sai.

- Cú pháp:

WebDriverWait wait = new WebDriverWait(driver, 10);  
 WebElement element

=wait.until(ExpectedConditions.elementToBeClickable(By.id(>someid>)));

**d) Câu lệnh pageLoadTimeout**

- Mục đích: Thiết lập giá trị thời gian để chờ đợi cho trang web hoàn thành tải trước khi ném một lỗi.

- Cú pháp:

drive.manage().timeouts().pageLoadTimeout(100, TimeUnit.SECONDS);

**e) Câu lệnh setScriptTimeout**

- Mục đích: Thiết lập giá trị thời gian chờ đợi một kịch bản không đồng bộ để kết thúc việc thực hiện trước khi ném một lỗi. Nếu thời gian chờ là tiêu cực, sau đó kịch bản sẽ chạy vô hạn.

- Cú pháp:

drive.manage().timeouts().setScriptTimeout(100TimeUnit.SECONDS);

**f) Câu lệnh sleep**

- Mục đích: Câu lệnh này hiếm khi sử dụng vì nó luôn luôn buộc các trình duyệt chờ đợi một thời gian cụ thể.

- Cú pháp: thread.sleep(1000);

### **2.2.5. Ngôn ngữ lập trình Java**

Java là một ngôn ngữ lập trình hướng đối tượng, mạnh mẽ và phổ biến, được biên dịch thành mã máy trước khi thực thi, có khả năng đáp ứng nhiều nhu cầu khác nhau trong quá trình phát triển phần mềm.

Các lý do chọn Java là ngôn ngữ để thực hiện kiểm thử tự động:

* + Đa năng và phổ biến
  + Dễ học và sử dụng
  + Thư viện và framework đa dạng
  + Hỗ trợ đa nền tảng
  + Cộng đồng và tài liệu đa dạng
  + Khả năng tích hợp và mở rộng
  + Tính ổn định và độ tin cậy

Với Java unit test (còn được gọi là JUnit) là một nền tảng kiểm thử (testing framework) được phát triển dành cho Java có một số annotaion và assert statement trong unit test:

* + @Test: Đánh dấu phương thức kiểm thử.
  + @Before: Được thực thi trước khi thực thi mỗi function test, nó được sử dụng để chuẩn bị môi trường cho việc test (đọc, input data, khởi tạo class, ...).
  + @After: Được thực thi sau mỗi function test, nó được sử dụng để làm sạch môi trường test.
  + @BeforeClass: Được thực thi một lần trước khi bắt đầu thực thi tất cả các function test. Các phương thức được đánh dấu bằng annotaion này thì cần được xác định là static để làm việc với JUnit.
  + @AfterClass: Được thực thi một lần, sau khi tất cả các function test kết thúc, nó được sử dụng để dọn dẹp một số thứ đã chuẩn bị trước đó cho việc test (connect database, ...). Các phương thức được đánh dấu bằng annotaion này thì cần được xác định là static để làm việc với JUnit.
  + @Ignore or @Ignore("Why disabled"): Đánh dấu test đó là disabled. Nó hữu ích trong trường hợp code thay đổi mà function test vẫn chưa được điều chỉnh.
  + @Test (expected = Exception.class): Failed nếu method không ném ra được tên một exception.
  + @Test(timeout=100): Failed nếu phương thức kiểm thử đó chạy quá 100 ms.
  + assertTrue(message, boolean condition): Đánh giá biểu thức luận lý. Test sẽ được chấp nhận nếu biểu thức đó đúng.
  + assertFalse(message, boolean condition): Đánh giá biểu thức luận lý. Test sẽ được chấp nhận nếu biểu thức đó sai.
  + assertNull(message, object): So sánh tham chiếu của một đối tượng với giá trị null. Test sẽ được chấp nhận nếu tham chiếu đối tượng đó là null.
  + assertNotNull(message, object): So sánh tham chiếu của một đối tượng với null. Test sẽ được chấp nhận nếu tham chiếu đối tượng đó khác null.
  + assertSame(message,expected, actual): So sánh địa chỉ vùng nhớ của 2 tham chiếu đối tượng bằng cách sử dụng toán tử ==. Test sẽ được chấp nhận nếu cả 2 đều tham chiếu đến cùng một đối tượng.
  + assertNotSame(message,expected, actual): So sánh địa chỉ vùng nhớ của 2 tham chiếu đối tượng bằng cách sử dụng toán tử ==. Test sẽ được chấp nhận nếu cả 2 đều tham chiếu đến các đối tượng khác nhau.
  + assertThat(): So sánh một giá trị thực tế có thỏa mãn với 1 Matcher được xác định hay không. Với matchers có thể kiểm tra kết quả của một string, number, collections …
  + fail(): Phương thức này làm cho test hiện hành thất bại, phương thức này thường được sử dụng khi xử lý các ngoại lệ. Mặc dù chúng ta có thể chỉ cần sử dụng phương thức assertTrue() cho gần như hầu hết các test case, tuy nhiên thì việc sử dụng một trong các phương thức assertXXX() cụ thể sẽ làm cho các test dễ hiểu hơn và cung cấp các thông điệp khi một test bị fail rõ ràng hơn.

## **2.3. Cài đặt Selenium Webdriver**

- Các bước cài đặt Selenium WebDriver:

Bước 1: Cài đặt Chrome Browser

- Vào trang web <https://www.google.com/intl/vi_vn/chrome/> và nhấn nút “Tải xuống tại đây”.

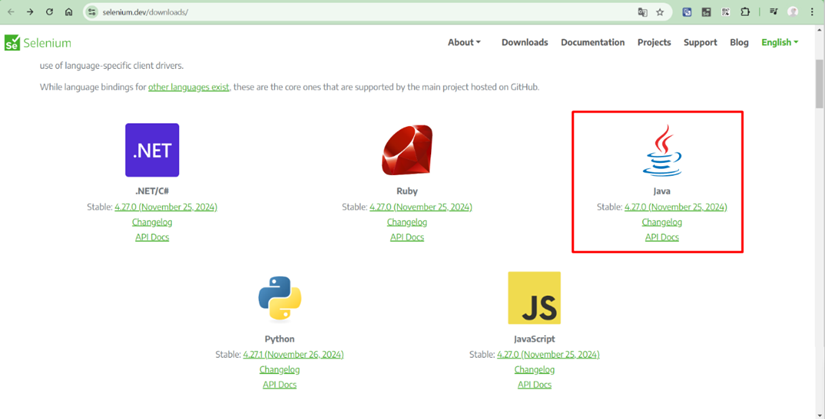
- Thực hiện cài đặt Chrome.

Bước 2: Cài đặt Chrome Driver

- Vào trang web<https://chromedriver.chromium.org/downloads> và tải phiên bản phù hợp với Chrome Browser đã tải ở bước 1.

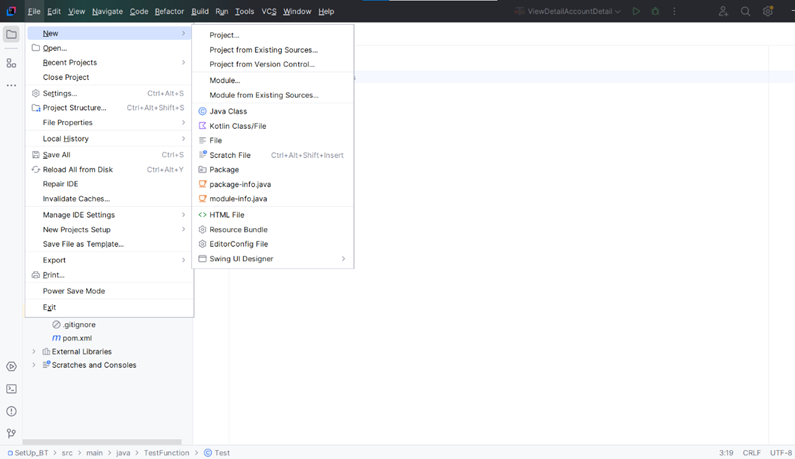
Bước 3: Cài đặt thư viện Selenium Webdriver sử dụng ngôn ngữ Java vào IntelliJ IDEA.

Trước khi cài đặt ta phải có điều kiện sử dụng Java 8 trở lên và có công cụ hỗ trợ code (Intellij IDEA, Eclipse,..). Ở đây ta thực hiện cài đặt Selenium WebDriver với công cụ IntelliJ IDEA.



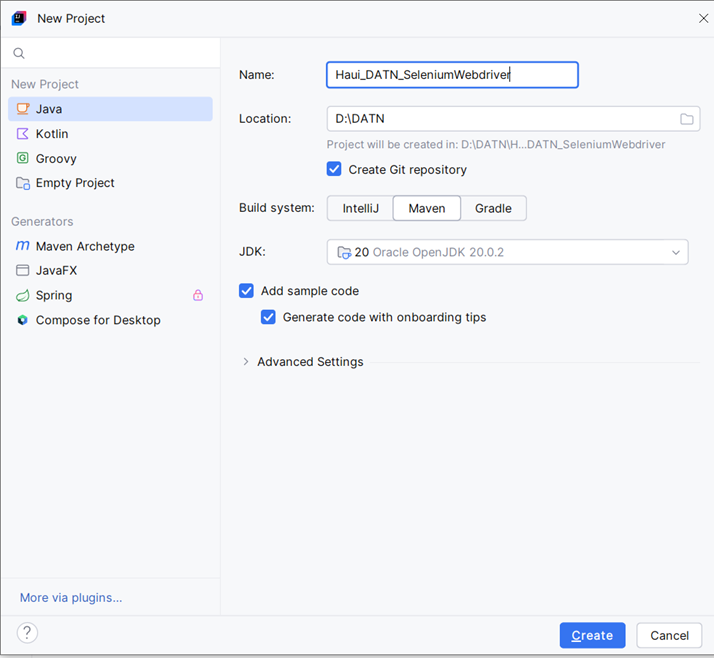
Hình 2.1: Màn hình trang tải về Selenium Webdriver

Mở IntelliJ IDEA lên, tạo một project mới.



Hình 2.2: Thao tác tạo một project mới trong IntelliJ

Chọn cấu hình JDK đúng với version Java.



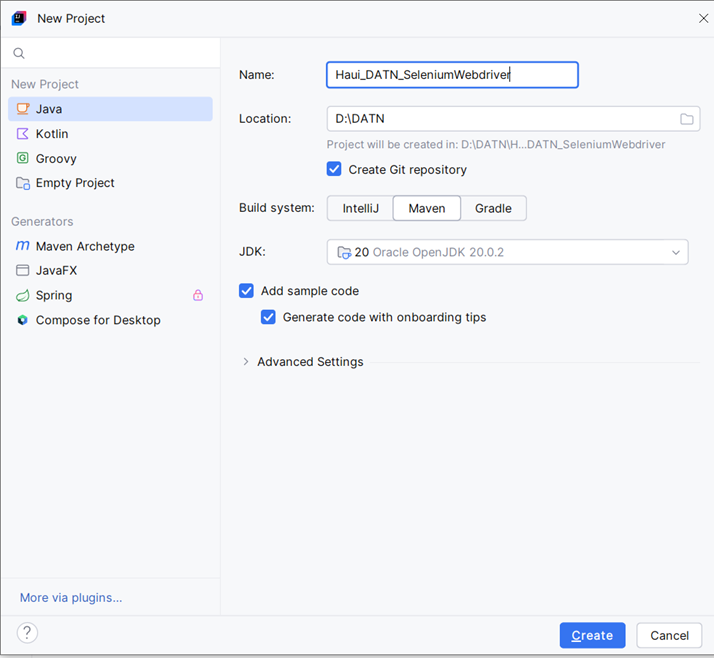
Hình 2.3: Chọn cấu hình JDK đúng với version Java ở IntelliJ

Sau khi tạo Project Java xong, tới một bước quan trọng là cài đặt thư viện của Selenium WebDriver vào project để nó có thể chạy được các yêu cầu của WebDriver.

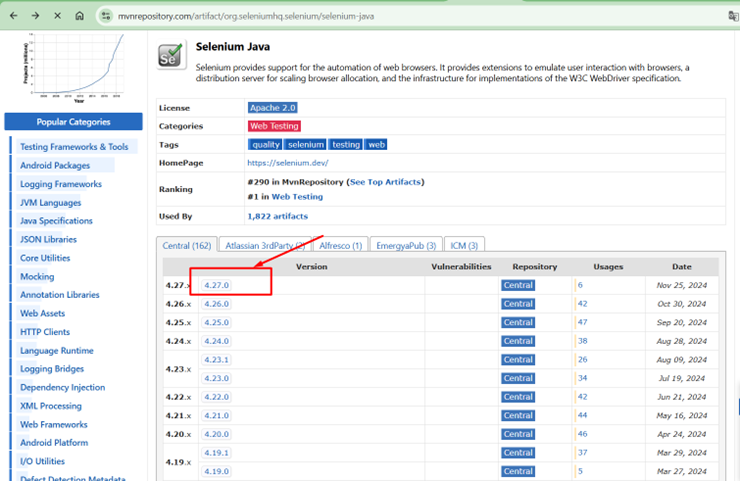
- Cài đặt thư viện Selenium vào Maven project trên IntelliJ

Để cài đặt thư viện Selenium vào Maven project thì sẽ cài nó vào file pom.xml

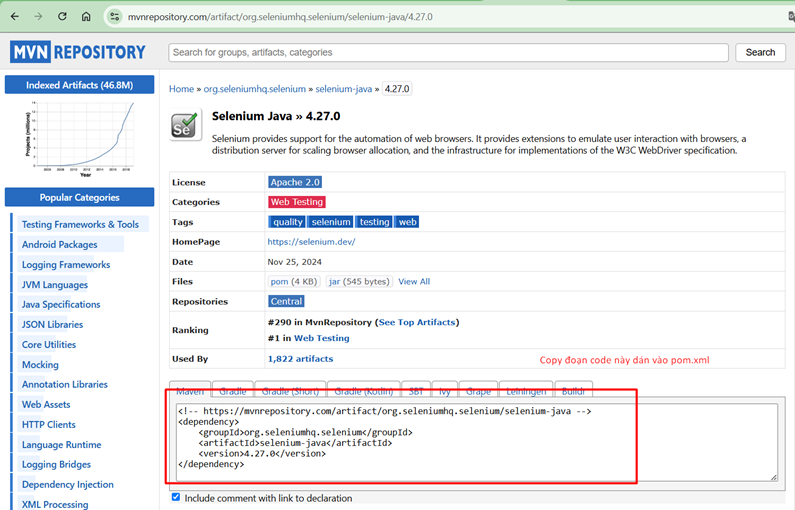
Thư viện được lấy tại trang web:<https://mvnrepository.com/>



Hình 2.4: Thao tác tải Selenium Java

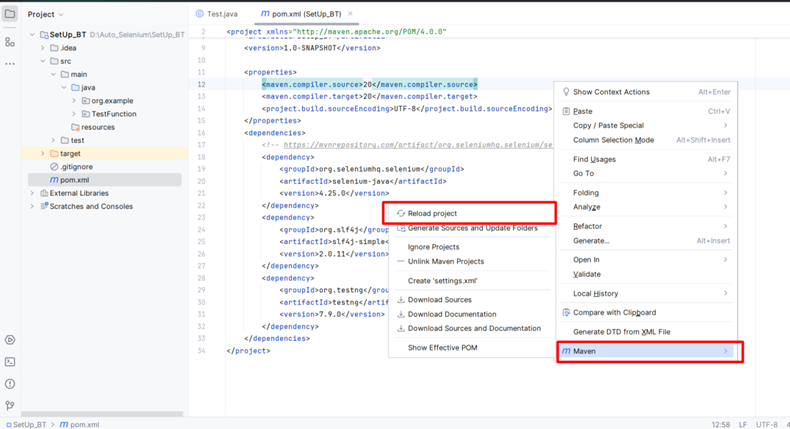


Hình 2.5: Chọn phiên bản phù hợp



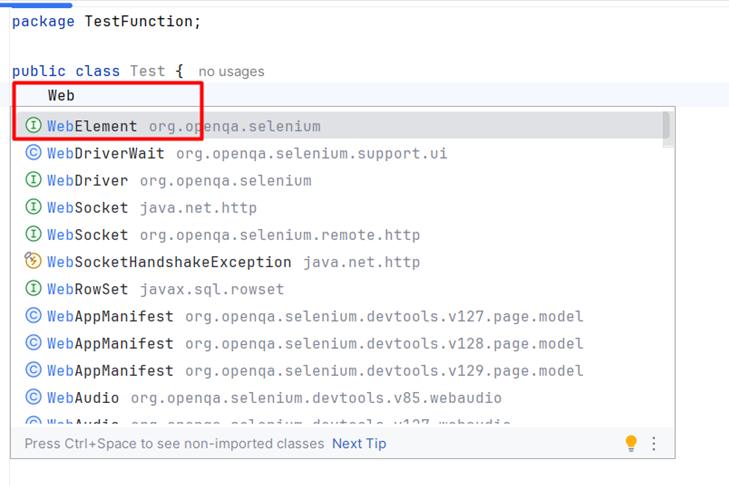
Hình 2.6: Thao tác cài đặt Selenium Java về project

Khi dán vào file pom.xml nhớ để trong cặp thẻ <dependencies> </dependencies>. Sau khi dán xong nhớ reload project lại bằng cách nhấn vào biểu tượng Reload hoặc chuột phải > Maven > reload project.



Hình 2.7: Thực hiện khởi động lại Maven

Kiểm tra Selenium đã cài đặt vào maven project

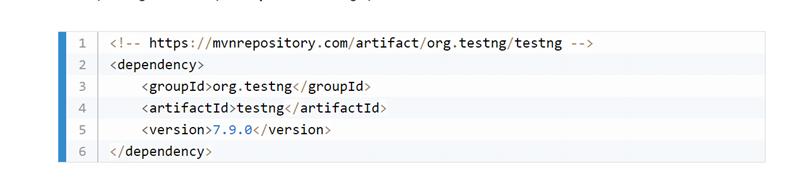


Hình 2.8: Hoàn thiện cài đặt thư viện Selenium

Nếu gợi ý hiển thị lên như hình là đã cài đặt được Selenium thành công

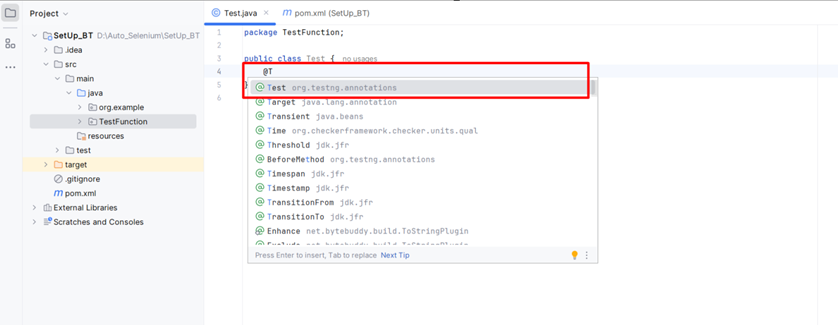
Bước 4: Cài đặt thư viện TestNG vào IntelliJ

Thêm thư viện vào file pom.xml tương tự Selenium



Hình 2.9: Cài đặt thư viện TestNG

Thực hiện Reload Maven để hoàn tất cài đặt



Hình 2.10: Hoàn thiện cài đặt TestNG

# **ỨNG DỤNG CÔNG CỤ KIỂM THỬ CHO HỆ THỐNG WEBSITE BÁN THỰC PHẨM CHỨC NĂNG**

## **3.1. Mô tả về website bán thực phẩm chức năng**

### **3.1.1. Giới thiệu**

Hệ thống em thực hiện kiểm thử là một website thương mại điện tử được xây dựng nhằm cung cấp một hệ thống bán hàng trực tuyến, giúp cho khách hàng dù ở xa vẫn có thể mua được sản phẩm mà không cần trực tiếp tới cửa hàng.

Hoạt động bán các sản phẩm trên nhiều lĩnh vực thời trang, đồ công nghệ.... Website hỗ trợ người dùng mua và tìm hiểu thông tin của sản phẩm, đặt hàng, theo dõi đơn hàng hơn nữa nó còn hỗ trợ việc quản lý sản phẩm, đơn hàng, …

Bên cạnh những chức năng nghiệp vụ, hệ thống phải đảm bảo những đặc tính sau:

* Tính tích hợp cao: Mọi hoạt động từ nhập thông tin về sản phẩm, số lượng, tạo đơn hàng, tạo các báo cáo được thực hiện trên cùng một dữ liệu đầu vào.
* Hệ thống phải đảm bảo tính bảo mật cao.
* Thời gian đáp ứng của hệ thống phải đủ nhanh, có thể đáp ứng được số lượng giao dịch lớn trên một đơn vị thời gian.
* Hệ thống có giao diện thân thiện với người sử dụng.
* Hệ thống có khả năng mở rộng, dễ dàng tích hợp với các hệ thống khác.

## **3.2. Kế hoạch kiểm thử**

### **3.2.1. Phạm vi kiểm thử**

Thực hiện tạo ca kiểm thử, viết test script và thực thi với các chức năng của website bán thực phẩm chức năng.

Phía khách hàng:

* Đăng ký
* Đăng nhập
* Đăng xuất
* Quên mật khẩu
* Tìm kiếm sản phẩm
* Quản lý giỏ hàng
* Đặt hàng
* Cập nhật thông tin cá nhân

### **3.2.2. Mục tiêu kiểm thử**

Tập trung vào kiểm tra chức năng cơ bản với các hạng mục kiểm tra sau:

* Kiểm tra chức năng
* Kiểm tra giá trị biên
* Kiểm tra xử lý lỗi

### **3.2.3. Chiến lược kiểm thử**

#### **3.2.3.1. Đối với tạo các ca kiểm thử**

Tạo các ca kiểm thử dựa vào các kịch bản.

Các kỹ thuật được áp dụng để tạo các ca kiểm thử là phân vùng tương đương, phân tích giá trị biên, bảng quyết định và đoán lỗi.

#### **3.2.3.2. Đối với viết test script**

Viết test script cho các ca kiểm thử nhiều nhất có thể.

Tên ca kiểm thử trong test script sẽ được đặt giống với ID của ca kiểm thử.

#### **3.2.3.3. Đối với thực thi kiểm thử**

Người thực hiện kiểm thử ở 2 vòng được đổi chỗ cho nhau.

Thứ tự thực hiện kiểm thử của các ca kiểm thử ở 2 vòng cũng được xáo trộn.

### **3.2.4 Tiêu chí kiểm thử**

#### **3.2.4.1. Tiêu chí đình chỉ kiểm thử**

#### **3.2.4.2. Tiêu chí kết thúc kiểm thử**

Các hoạt động kiểm thử sẽ được hoàn thành khi:

Mật độ kiểm thử phải bằng hoặc lớn hơn kế hoạch.

Tất cả các trường hợp thử nghiệm được thực thi. Tất cả các lỗi được tìm thấy đều được đóng lại.

Tất cả các tính năng liệt kê từ phần 3.2.1 đều được tiến hành thử nghiệm.

### **3.2.5 Nguồn nhân lực**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Phân quyền** | **Trách nhiệm** | **Người thực hiện** |
| 1 | Tester | - Tạo các ca kiểm thử  - Viết test script  - Thực hiện kiểm thử  - Báo cáo bug | Đào Thị Thuỳ Linh |

### **3.2.6 Môi trường kiểm thử**

Link ứng dụng: https://demowebshop.tricentis.com

Trình duyệt kiểm thử: Chrome

Hệ điều hành: Windows 10

### **3.2.7 Lịch trình và dự toán**

|  |  |  |  |
| --- | --- | --- | --- |
| **Công việc** | **Ngày bắt đầu** | **Ngày kết thúc** | **Ngày hoàn thành thực tế** |
| Lập kế hoạch kiểm thử | 18/3/2025 | 25/3/2025 | 25/3/2025 |
| Thiết kế các ca kiểm thử | 25/3/2025 | 5/4/2025 | 5/4/2025 |
| Viết test script | 5/4/2025 | 30/4/2024 | 30/4/2024 |
| Thực thi kiểm thử | 30/4/2025 | 14/5/2025 | 14/5/2025 |

### **3.2.8 Xác định phân phối kiểm thử**

Nhóm phát triển phần mềm phải đảm bảo phần mềm đã sẵn sàng trước mỗi vòng kiểm thử.

Vòng kiểm thử cuối cùng phải vượt qua tất cả các ca kiểm thử mà không có bất kỳ lỗi nào.

Nhóm kiểm thử sẽ thực hiện kiểm tra hồi quy và thông qua allure report để thực hiện báo cáo lỗi.

## **3.3. Thiết kế ca kiểm thử**

### **3.3.1. Đăng ký**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Mục tiêu kiểm thử** | **Các bước thực hiện** | **Kết quả mong muốn** |
| TEST\_01 | Kiểm tra đăng ký thành công | 1. Chọn thông tin giới tính  2. Nhập thông tin trường first name  3. Nhập thông tin trường last name  4. Nhập mật khẩu đúng  5. Nhập lại mật khẩu  6. Nhấn "Register" | 6. Đăng ký thành công tài khoản |
| TEST\_02 | Kiểm tra để trống trường first name | 1. Chọn các thông tin khác chính xác  2. Để trống trường first name  3. Nhấn "Register" | 3. Hiển thị thông báo lỗi chưa nhập trường first name |
| TEST\_03 | Kiểm tra trường first name chứa kí tự đặc biệt- kí tự số | 1. Chọn các thông tin khác chính xác  2. Trường first name chứa kí tự đặc biệt- chữ số  3. Nhấn "Register" | 3. Hiển thị thông báo lỗi first name chứa kí tự đặc biệt |
| TEST\_04 | Kiểm tra trường first name chứa hơn 25 ký tự | 1. Chọn các thông tin khác chính xác  2. Trường first name nhập dài hơn 25 ký tự  3. Nhấn "Register" | 3. Hiển thị thông báo lỗi first name quá dài |
| TEST\_05 | Kiểm tra để trống trường last name | 1. Chọn các thông tin khác chính xác  2. Để trống trường last name  3. Nhấn "Register" | 3. Hiển thị thông báo lỗi chưa nhập trường last name |
| TEST\_06 | Kiểm tra trường last name chứa các kí tự đặc biệt- kí tự số | 1. Chọn các thông tin khác chính xác  2. Trường last name chứa kí tự đặc biệt- chữ số  3. Nhấn "Register" | 3. Hiển thị thông báo lỗi last name chứa kí tự đặc biệt |
| TEST\_07 | Kiểm tra trường last name chứa hơn 25 ký tự | 1. Chọn các thông tin khác chính xác  2. Trường last name nhập dài hơn 25 ký tự  3. Nhấn "Register" | 3. Hiển thị thông báo lỗi last name quá dài |
| TEST\_08 | Kiểm tra không chọn giới tính | 1. Chọn các thông tin khác chính xác  2. Không chọn thông tin giới tính  3. Nhấn "Register" | 3. Hiển thị thông báo lỗi chưa chọn trường giới tính |
| TEST\_9 | Kiểm tra trường email để trống | 1. Chọn các thông tin khác chính xác  2. Để trống trường email  3. Nhấn "Register" | 3. Hiển thị thông báo lỗi chưa nhập trường email |
| TEST\_10 | Kiểm tra email không đúng định dạng | 1. Chọn các thông tin khác chính xác  2. Nhập trường email không đúng định dạng không chứa các ký tự "@" và "."  3. Nhấn "Register" | 3. Hiển thị thông báo lỗi trường email chưa đúng định dạng |
| TEST\_11 | Kiểm tra email đã đăng ký | 1. Chọn các thông tin khác chính xác  2. Nhập một email đã được đăng kí trước đó  3. Nhấn "Register" | 3. Hiển thị thông báo lỗi trường email này đã tồn tại |
| TEST\_12 | Kiểm tra password để trống | 1. Chọn các thông tin khác chính xác  2. Để trống trường password  3. Nhấn "Register" | 3. Hiển thị thông báo lỗi chưa nhập trường password |
| TEST\_13 | Kiểm tra password chứa ít hơn 6 kí tự | 1. Chọn các thông tin khác chính xác  2. Trường password chứa ít hơn 6 kí tự  3. Nhấn "Register" | 3. Hiển thị thông báo lỗi trường password chưa đủ số lượng kí tự |
| TEST\_14 | Kiểm tra trường Retype password để trống | 1. Chọn các thông tin khác chính xác  2. Để trống trường Retype password  3. Nhấn "Register" | 3. Hiển thị thông báo lỗi chưa nhập trường Retype password |
| TEST\_15 | kiểm tra retype password không trùng với password | 1. Chọn các thông tin khác chính xác  2. Nhập trường re-password không giống trường password  3. Nhấn "Register" | 3. Hiển thị thông báo lỗi trường re\_password chưa khớp với trường password |

### 

### **3.3.2. Đăng nhập**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Mục tiêu kiểm thử** | **Các bước thực hiện** | **Kết quả mong muốn** |
| TEST\_01 | Kiểm thử nhập thông tin các trường hợp lệ | 1. Nhập email chính xác  2. Nhập mật khẩu chính xác  3. Nhấn nút "Log in" | 3. Đăng nhập thành công |
| TEST\_02 | Kiểm tra không nhập email | 1. Trường email để rỗng  2. Nhập mật khẩu chính xác  3. Nhấn nút "Log in" | 3. Hiển thị thông báo lỗi chưa nhập email |
| TEST\_03 | Kiểm tra email chưa được đăng ký | 1. Nhập email chưa đã đăng kí vào hệ thống  2. Nhập mật khẩu chính xác  3. Nhấn nút "Log in" | 3. Hiển thị thông báo lỗi email chưa được đăng ký |
| TEST\_04 | Kiểm tra email không đúng định dạng | 1. Nhập email không đúng định dạng  2. Nhập mật khẩu chính xác  3. Nhấn nút "Log in" | 3. Hiển thị thông báo lỗi email chưa đúng định dạng |
| TEST\_05 | Kiểm tra email chứa kí tự đặc biệt | 1. Nhập email chứa các kí tự đặc biệt  2. Nhập mật khẩu chính xác  3. Nhấn nút "Log in" | 3. Hiển thị thông báo lỗi email chưa đúng định dạng |
| TEST\_06 | Kiểm tra email chứa kí tự khoảng trắng | 1. Nhập email chưa ký tự khoảng trắng  2. Nhập mật khẩu chính xác  3. Nhấn nút "Log in" | 3. Đăng nhập thành công |
| TEST\_07 | Kiểm tra không nhập mật khẩu | 1.Nhập email chính xác  2. Trường mật khẩu để trống  3. Nhấn nút "Log in" | 3. Hiển thị thông báo lỗi chưa nhập mật khẩu |
| TEST\_08 | Kiểm tra mật khẩu nhỏ hơn 6 ký tự | 1.Nhập email chính xác  2. Trường mật khẩu không đủ 6 ký tự  3. Nhấn nút "Log in" | 3. Hiển thị thông báo chưa đủ số lượng ký tự |
| TEST\_09 | Kiểm tra mật khẩu không chính xác | 1.Nhập email chính xác  2. Trường mật khẩu đủ 6 ký tự không chính xác  3. Nhấn nút "Log in" | 3. Hiển thị thông báo đăng nhập không thành công |

### 

### **3.3.3. Tìm kiếm sản phẩm**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Mục tiêu kiểm thử** | **Các bước thực hiện** | **Kết quả mong muốn** |
| TEST\_01 | Kiểm thử nhập thông tin các trường hợp lệ | 1. Nhập email chính xác  2. Nhập mật khẩu chính xác  3. Nhấn nút "Log in" | 3. Đăng nhập thành công |

### **3.3.4. Cập nhật thông tin cá nhân**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Mục tiêu kiểm thử** | **Các bước thực hiện** | **Kết quả mong muốn** |
| TEST\_01 | Kiểm tra thay đổi lại thông tin giới tính | 1. Chọn lại ô giới tính  2. Không thay đổi các giá trị khác  3. Nhấn "Save" | 3. Thông tin giới tính mới trùng khớp với giới tính đã thay đổi |
| TEST\_02 | Kiểm tra trường first name để trống | 1. Trường first name để trống  2. Nhập các thông tin khác hợp lệ  3. Nhấn "Save" | 3. Hiển thị thông báo lỗi first name không được để trống |
| TEST\_03 | Kiểm tra trường fist name chứa kí tự đặc biệt- kí tự số | 1. Trường first name chứa các kí tự đặc biệt và kí tự số  2. Nhập các thông tin khác hợp lệ  3. Nhấn "Save" | 3. Hiển thị thông báo lỗi first name không được chứ kí tự đặc biệt và số |
| TEST\_04 | Kiểm tra trường firstname chứa khoảng rỗng ở trước và sau | 1. Trường first name chứa các kí tự khoảng trống ở trước và sau  2. Nhập các thông tin khác hợp lệ  3. Nhấn "Save" | 3. Lưu thông tin vừa thay đổi thông tin mới không chứa kí tự khoảng trắng ở đầu và cuối |
| TEST\_05 | Kiếm tra trường first name chứa hơn 25 kí tự | 1. Trường first name >25 kí tự  2. Nhập các thông tin khác hợp lệ  3. Nhấn "Save" | 3. Hiển thị thông báo lỗi first name không được vượt quá 25 kí tự |
| TEST\_06 | Kiểm tra để trống trường last name | 1. Trường last name để trống  2. Nhập các thông tin khác hợp lệ  3. Nhấn "Save" | 3. Hiển thị thông báo lỗi last name không được để trống |
| TEST\_07 | Kiểm tra trường last name chứa các kí tự đặc biệt- kí tự số | 1. Trường last name chứa các kí tự đặc biệt và kí tự số  2. Nhập các thông tin khác hợp lệ  3. Nhấn "Save" | 3. Hiển thị thông báo lỗi last name không được chứ kí tự đặc biệt và số |
| TEST\_08 | Kiểm tra trường last name chưa khoảng trắng ở trước và sau | 1. Trường last name chứa các kí tự khoảng trống ở trước và sau  2. Nhập các thông tin khác hợp lệ  3. Nhấn "Save" | 3. Lưu thông tin vừa thay đổi thông tin mới không chứa kí tự khoảng trắng ở đầu và cuối |
| TEST\_09 | Kiểm tra trường last name chứa hơn 25 kí tự | 1. Trường last name >25 kí tự  2. Nhập các thông tin khác hợp lệ  3. Nhấn "Save" | 3. Hiển thị thông báo lỗi last name không được vượt quá 25 kí tự |
| TEST\_10 | Kiểm tra trường email để trống | 1. Trường email để trống  2. Nhập các thông tin khác hợp lệ  3. Nhấn "Save" | 3. Hiển thị thông báo lỗi email không được để trống |
| TEST\_11 | Kiểm tra email không đúng định dạng | 1. Trường email không đúng định dạng  2. Nhập các thông tin khác hợp lệ  3. Nhấn "Save" | 3. Hiển thị thông báo lỗi email không đúng định dạng |
| TEST\_12 | Kiểm tra nhập đúng các thông tin nhấn save | 1. Nhập đúng các thông tin  2. Nhấn Save | 2. Hiển thị đúng như các thông tin đã tạo |

### 

### **3.3.5. Quên mật khẩu**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Mục tiêu kiểm thử** | **Các bước thực hiện** | **Kết quả mong muốn** |
| TEST\_01 | Kiểm tra nhập đúng email đã được đăng ký | 1. Nhập đúng email đã được đăng ký  2. Nhấn "Recover" | 2. Hiển thị thông báo thành công và hướng dẫn vào mail lấy mã |
| TEST\_02 | Kiểm tra email chưa được đăng kí | 1. Nhập email chưa được đăng ký  2. Nhấn "Recover" | 2. Hiển thị lỗi email chưa được đăng kí |
| TEST\_03 | Kiểm tra trường email rỗng | 1. Để trống trường email  2. Nhấn "Recover" | 2. Hiển thị thông báo lỗi email để trống |
| TEST\_04 | Kiểm tra trường email không đúng định dạng | 1. Nhập email không đúng định dạng thiếu các kí tự "@", "." chứa các kí tự đặc biệt  2. Nhấn "Recover" | 2. Hiển thị email không đúng định dạng |
| TEST\_05 | Email chứa khoảng trắng ở trước và sau | 1. Nhập email chứa khoảng trắng ở trước và sau 1 email chính xác  2. Nhấn "Recover" | 2. Hiển thị thông báo thành công và hướng dẫn vào mail lấy mã |

### 

### **3.3.6. Quản lý giỏ hàng**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Mục tiêu kiểm thử** | **Các bước thực hiện** | **Kết quả mong muốn** |
| Add to cart | | | |
| TEST\_01 | Kiểm tra với số lượng mặc định | 1. Nhấn "Add to cart" | 1. Hiển thị thông báo thêm vào giỏ hàng thành công |
| TEST\_02 | Kiểm tra nhập số lượng hợp lệ | 1. Nhập số lượng hợp lệ  2. Nhấn "Add to cart" | 2. Hiển thị thông báo thêm vào giỏ hàng thành công |
| TEST\_03 | Kiểm tra nhập số lượng chứa kí tự chữ | 1. Nhập số lượng chứa kí tự chữ cái  2. Nhấn "Add to cart" | 2. Hiển thị thông báo nhập số lượng lỗi |
| TEST\_04 | Kiểm tra nhập số lượng chứa kí tự đặc biệt | 1. Nhập số lượng chứa kí tự đặc biệt  2. Nhấn "Add to cart" | 2. Hiển thị thông báo nhập số lượng lỗi |
| TEST\_05 | Kiểm tra nhập số lượng chữ kí tự khoảng trắng | 1. Nhập số lượng chứa kí tự khoảng trắng  2. Nhấn "Add to cart" | 2. Hiển thị thông báo nhập số lượng lỗi |
| TEST\_06 | Kiểm tra với số lượng trống | 1. Để trống trường số lượng  2. Nhấn "Add to cart" | 2. Hiển thị thông báo nhập số lượng lỗi |
| TEST\_07 | Kiểm tra với số lượng chứa toàn ký tự khoảng trắng | 1. Trường số lượng toàn ký tự khoảng trắng  2. Nhấn "Add to cart" | 2. Hiển thị thông báo nhập số lượng lỗi |
| TEST\_08 | Kiểm tra nhập số lượng âm | 1. Nhập trường số lượng là một số âm  2. Nhấn "Add to cart" | 2. Hiển thị thông báo nhập số lượng lỗi |
| TEST\_09 | Kiểm tra nhập số lượng bằng 0 | 1. Nhập trường số lượng bằng 0  2. Nhấn "Add to cart" | 2. Hiển thị thông báo nhập số lượng lỗi |
| TEST\_10 | Kiểm tra nhập số lượng là số thập phân | 1. Nhập trường số lượng là số thập phân  2. Nhấn "Add to cart" | 2. Hiển thị thông báo nhập số lượng lỗi |
| Xoá sản phầm | | | |
| TEST\_11 | Xóa 1 sản phẩm trong giỏ | 1. Chọn 1 sản phẩm trong giỏ hàng  2. Nhấn "Update cart" | 2. Xóa sản phẩm được chọn |
| TEST\_12 | Xóa tất cả các sản phẩm trong giỏ | 1. Chọn tất cả sản phẩm trong giỏ hàng  2. Nhấn "Update cart" | 2. Xóa tất cả sản phẩm trong giỏ hàng |

### 

### **3.3.7. Mua hàng**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Mục tiêu kiểm thử** | **Các bước thực hiện** | **Kết quả mong muốn** |
| Chưa chấp nhận điều khoản | | | |
| TEST\_01 | Chưa chấp nhận các điều khoản | 1. Vào giỏ hàng  2. Nhấn mua hàng | 2. Hiển thị dialog lỗi chưa chấp nhận điều khoản |
| Đã chấp nhận các điều khoản | | | |
| TEST\_02 | Chưa log in tài khoản | 1. Vào giỏ hàng  2. Chấp nhận điều khoản  3. Nhấn mua hàng | 3. Chuyển hướng sang màn đăng nhập tài khoản |
| TEST\_03 | Mua hàng thành công | 1. Đăng nhập  2. Vào giỏ hàng  3. Chấp nhận điều khoản  4. Mua hàng | 4. Hiển thị thông báo mua hàng thành công |

### 

### **3.3.8. Đăng xuất**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Mục tiêu kiểm thử** | **Các bước thực hiện** | **Kết quả mong muốn** |
| TEST\_01 | Kiểm tra đăng xuất thành công | 1. Nhấn chọn "Tên đăng nhập"  2. Nhấn chọn "Sign out" | 2. Đăng xuất thành công |

### 

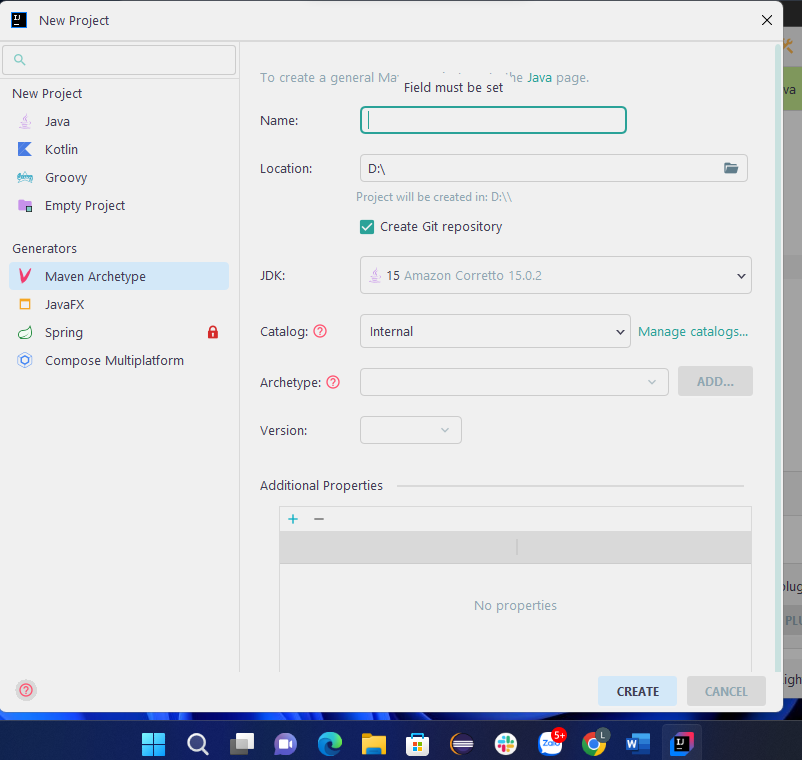
## **3.4. Xây dựng test script**

Để xây dựng test script, ban đầu người dùng cần phải tải xuống bản webdriver tương thích với mỗi browser để có thể chạy được các auto test script. Download công cụ Intellij để thực hiện xây dựng và thực thi câu lệnh (tùy vào nhu cầu mỗi người dùng, có thể viết và thực thi câu lệnh trên các công cụ khác nhau có hỗ trợ ngôn ngữ và thư viện java).

Với mỗi browser sẽ có một bản webdriver riêng như với Chrome - ChromeDriver, với Firefox - GeckoDriver, …

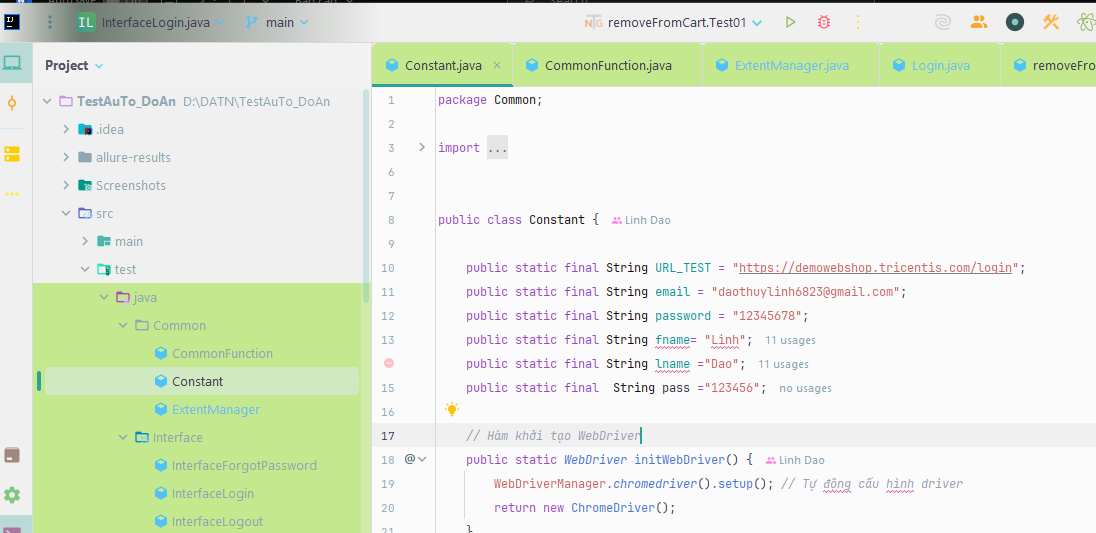
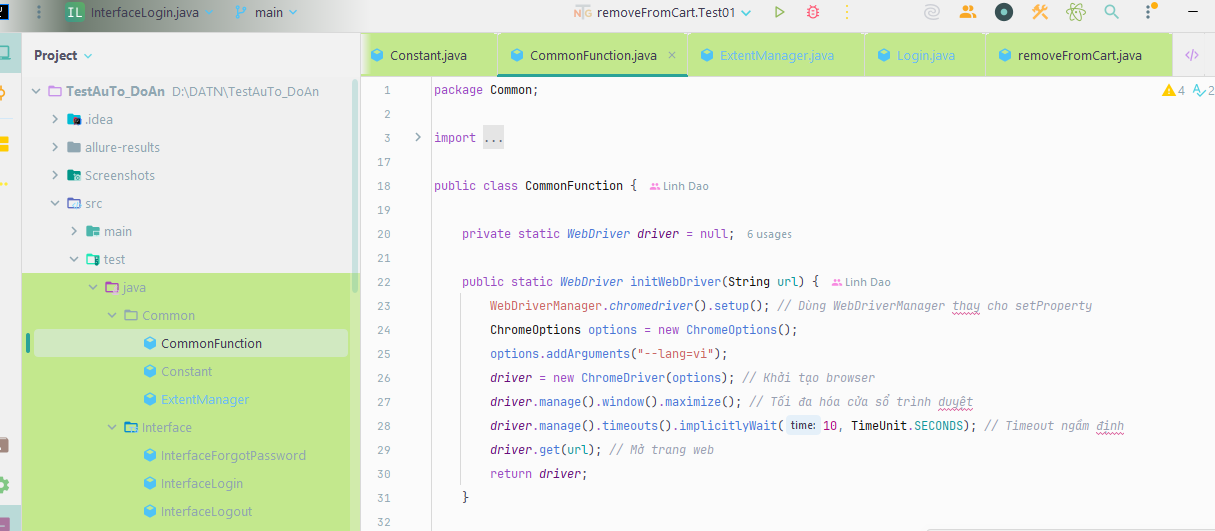
Các bước triển khai:

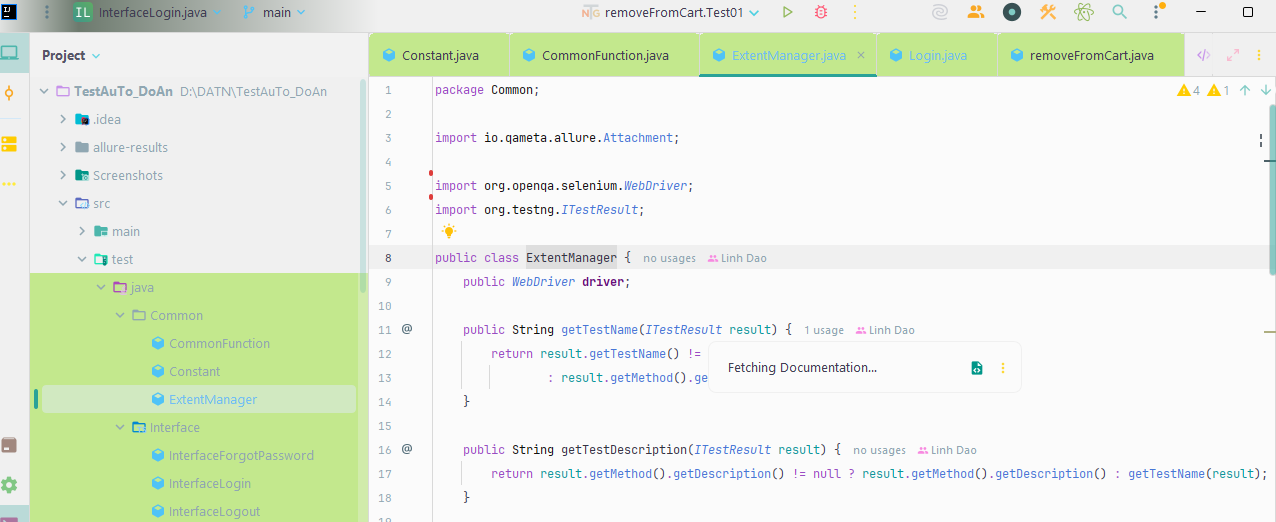
Bước 1: Tạo mới một Maven project: chọn File → New → Maven Project → Nhập thông tin của project → Nhấn Finish



Hình 3.1: Tạo project

Bước 2: Tạo package Commons để lưu trữ các biến và phương thức chung





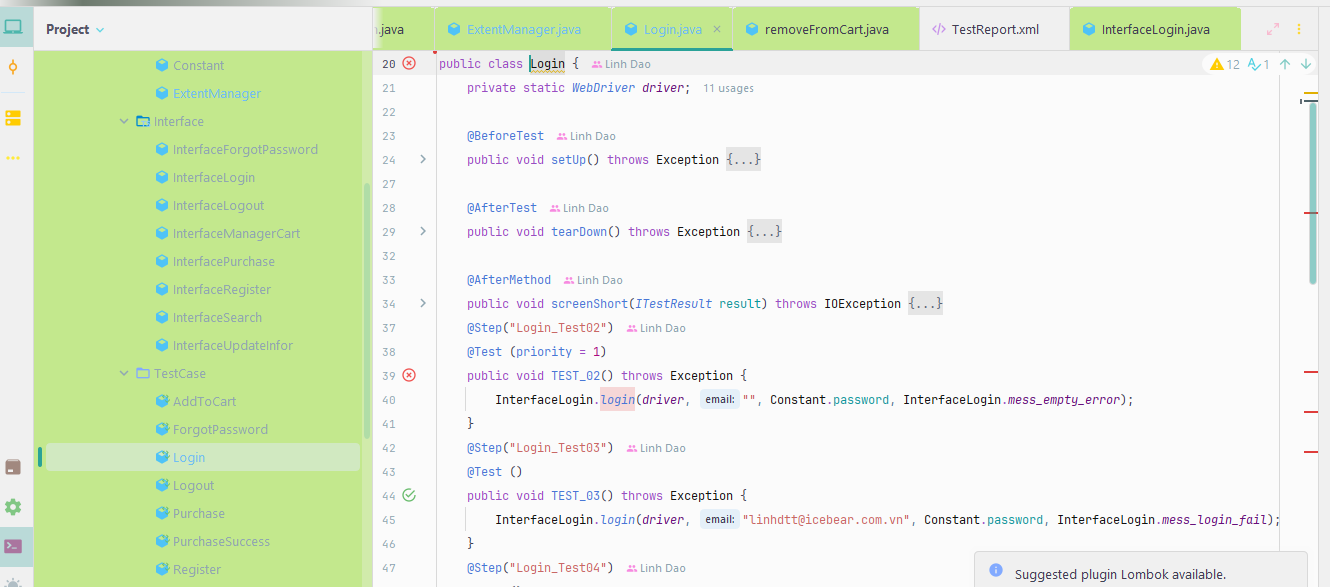
Hình 3.2: Tạo file chức các biến và phương thức chung

Bước 3: Tạo package Interfaces để lưu trữ các đường dẫn xpath và các hàm thao tác kiểm tra website



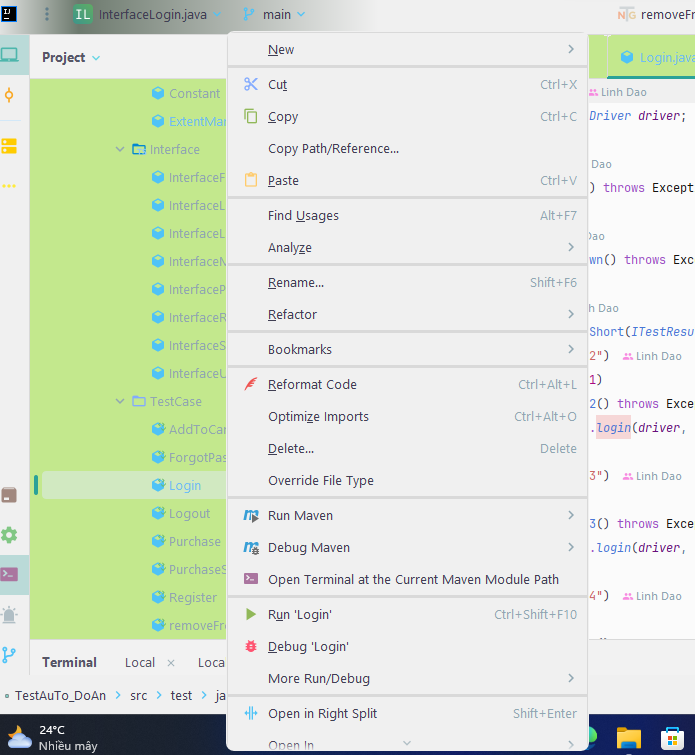
Hình 3.3: Tạo file chứa các đường dẫn xpath và các hàm kiểm tra website

Bước 4: Tạo package Test Cases để lưu trữ các ca kiểm thử



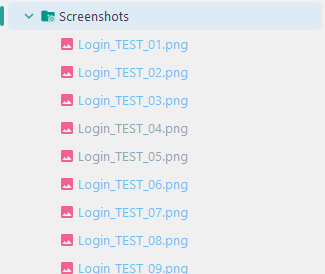
Hình 3.4:Tạo file chứa các ca kiểm thử của từng chức năng

Bước 5: Để chạy các ca kiểm thử của từng chức năng thì mở file testcase rồi nhấn nút  trên thanh menu hoặc chuột phải chọn Run as



Hình 3.5: Chạy các ca kiểm thử

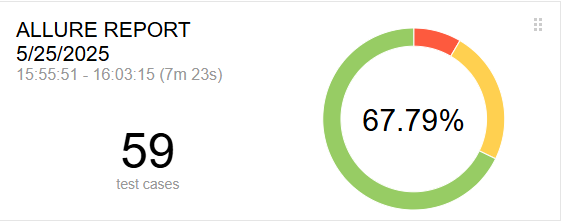
Bước 7: Sau khi chạy kiểm thử tự động thì tự động sinh ra kết quả kiểm thử và lưu lại ảnh thực hiện kiểm thử trong folder “Screenshots”



Hình 3.6: Hình ảnh minh họa kết quả kiểm thử

## **3.5. Kết quả kiểm thử**

Trong chương 3 của đồ án, em đã trình bày các chức năng của trang web thương mại điện tử và chatbot, kịch bản và kết quả kiểm thử tự động và sử dụng công cụ allure report để xuất kết quả tổng quan của kiểm thử:

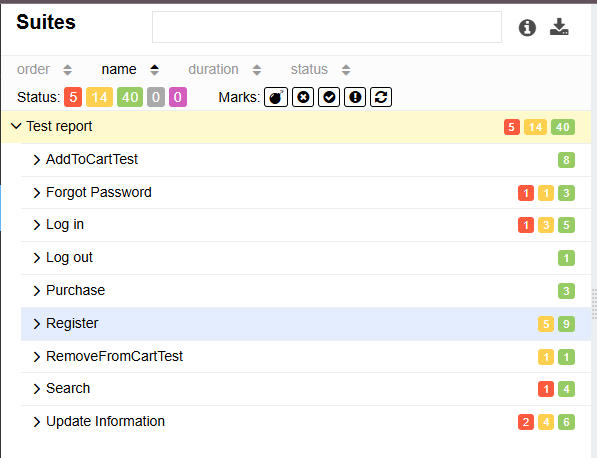
****

Hình 3.7: Kết quả tổng quan thực hiện các test case

Tổng quan về kết quả thực thi test case:

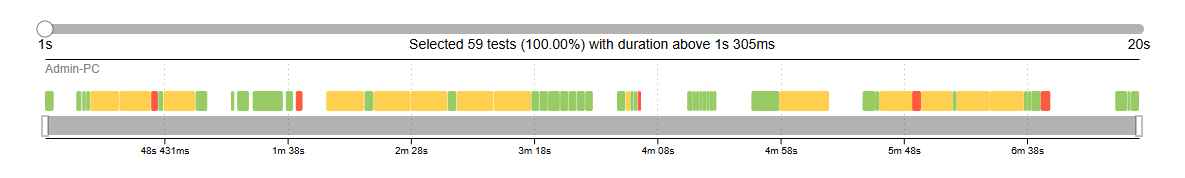
* + Số lượng test case thực hiện: 59 test case
  + Thời gian thực hiện các test case: 7 phút 23 giây
  + Ngày thực hiện: 25/5/2025
  + Số lượng test case thành công chiếm 67,79%

Chi tiết các test case thành công và không thành công trong từng chức năng



Hình 3.8: Hình ảnh kết quả chi tiết của từng chức năng

Chi tiết về thời gian thực hiện test case



Hình 3.9: Hình ảnh thanh thời gian thực hiện test case

# **KẾT LUẬN**

Kiểm thử phần mềm hiện nay giữ vai trò hết sức quan trọng trong việc đảm bảo chất lượng của một sản phẩm phần mềm. Mặc dù việc sử dụng các công cụ kiểm thử tự động đã giúp giảm thời gian, kinh phí, nhân lực trong quá trình phát triển phần mềm. Nhưng cũng có những việc mà công cụ kiểm thử tự động không thể thay thế được kiểm thử viên, hoặc nếu có thì việc cấu hình cho nó khó khăn hơn rất nhiều so với kiểm thử thủ công. Do đó, kiểm thử viên giỏi phải là người nhận biết được khi nào nên sử dụng công cụ kiểm thử tự động và khi nào nên kiểm thử thủ công.

Sau thời gian thực hiện đồ án dưới sự hướng dẫn của TS. Nguyễn Bá Nghiễn kết quả mà em thu được cụ thể như sau:

Trình bày được các kiến thức cơ bản về kiểm thử phần mềm nói chung và kiểm thử phần mềm tự động cho các ứng dụng web nói riêng.

Giới thiệu được các đặc điểm, thành phần của công cụ kiểm thử tự động Selenium. Kết hợp công cụ Selenium WebDriver và framework TestNG trong kiểm thử tự động ứng dụng web.

Áp dụng các kiến thức đã tìm hiểu vào tạo test case và thực hiện kiểm thử tự động các chức năng chính của ứng dụng web bán thực phẩm chức năng trực tuyến.

Tuy nhiên, do thời gian hạn chế em chưa thể giả lập nhiều người dùng ảo chạy đồng thời để kiểm thử khả năng xử lý xung đột khi có nhiều khách hàng cùng đặt hàng vào một thời điểm. Phạm vi mới kiểm thử với ứng dụng nhỏ, chưa kiểm thử với ứng dụng thực tế.

Trong tương lai gần em sẽ phát triển tìm hiểu về các loại test (ví dụ như kiểm thử api, kiểm thử giao diện, kiểm thử hiệu năng, …); áp dụng các kiểu kiểm thử khác nhau vào các sản phẩm thực tiễn, mở rộng thêm lĩnh vực kiểm thử từ đó đúc rút thêm được kinh nghiệm kiểm thử phần mềm cũng như kinh nghiệm làm việc.

Tìm hiểu chuyên sâu hơn và thực hiện kiểm thử tự động để có thể hoàn thành tốt việc kiểm thử của một trang website hoàn chỉnh hơn và đạt được các kết quả tốt hơn. Từ đó, ngày càng cải thiện và trau dồi các kỹ năng viết test script, kỹ thuật về kiểm thử chức năng cũng như trau dồi kinh nghiệm sử dụng các công cụ kiểm thử tự động và tăng thêm kiến thức cho bản thân.

# **TÀI LIỆU THAM KHẢO**

1. Hoàng Quang Huy, Phùng Đức Hòa, Nguyễn Đức Lưu, Trịnh Bá Quý, (2019), “Giáo trình kiểm thử phần mềm”, Nhà xuất bản Thống kê.
2. Kiểm thử phần mềm: <https://viblo.asia/newest>
3. Selenium WebDriver: https://www.guru99.com/selenium-tutorial.html
4. Tài liệu về JUnit: <https://junit.org/junit5/docs/current/user-guide/>
5. Tài liệu về Take Screenshot: <https://www.guru99.com/take-screenshot-selenium-webdriver.html>
6. Create a Selenium Maven project: <https://www.geeksforgeeks.org/how-to-create-a-selenium-maven-project-with-eclipse-to-open-chrome-browser/>