** BỘ** **CÔNG** **THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**





**NGUYỄN THỊ HẢI YẾN**

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

NGÀNH HỆ THỐNG THÔNG TIN

|  |
| --- |
| **TÌM HIỂU VÀ ỨNG DỤNG SELENIUM WEBDRIVER TRONG KIỂM THỬ WEBSITE BÁN BÁNH TRUNG THU** |

|  |  |
| --- | --- |
| **CBHD** | **: ThS.Trần Việt Hà** |
| **Sinh viên** | **: Nguyễn Thị Hải Yến** |
| **Mã sinh viên** | **: 2021607634** |

**HỆ THỐNG THÔNG TIN**

**Hà Nội – Năm 2025**

****

|  |
| --- |
| **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **---------------------------------------** |
| ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC  NGÀNH HỆ THỐNG THÔNG TIN |
|  |

|  |
| --- |
| **TÌM HIỂU VÀ ỨNG DỤNG SELENIUM WEBDRIVER TRONG KIỂM THỬ WEBSITE BÁN BÁNH TRUNG THU** |

|  |  |
| --- | --- |
| **CBHD** | **: ThS.Trần Việt Hà** |
| **Sinh viên** | **: Nguyễn Thị Hải Yến** |
| **Mã sinh viên** | **: 2021607634** |
| **Hà Nội – Năm 2025** |

LỜI CẢM ƠN

Trong quá trình thực hiện đồ án tốt nghiệp ngành Hệ thống thông tin, em luôn nhận được sự hỗ trợ, giúp đỡ và khuyến khích từ mọi người. Đó là động lực to lớn để giúp em có thể thực hiện thành công đồ án tốt nghiệp của mình. Lời đầu tiên, em xin được gửi đến các thầy cô trường Công nghệ thông tin và Truyền thông thuộc trường Đại học Công nghiệp Hà Nội – những người đã truyền dạy rất nhiều kiến thức và kỹ năng cần thiết cho em trong suốt thời gian tham gia học tập và rèn luyện tại trường để em có thể thực hiện tốt đồ án tốt nghiệp cũng như trong nghề nghiệp tương lai. Đặc biệt, em xin được gửi lời cảm ơn đến thầy ThS.Trần Việt Hà là người hướng dẫn, đồng hành cùng em trong suốt quá trình thực hiện đồ án tốt nghiệp. Nhờ có sự chỉ bảo tận tình và giúp đỡ của thầy, em đã có thể xây dựng và hoàn thành đồ án tốt nghiệp của mình một cách tốt nhất.

Em cũng muốn gửi lời cảm ơn tới Công ty Cổ phần VTI đã tạo cơ hội để em được học tập và thực hành trong lĩnh vực kiểm thử phần mềm, giúp em có nhiều thêm kiến thức, kỹ năng và định hướng phát triển cho nghề nghiệp sau này.

Trong quá trình thực hiện đồ án tốt nghiệp, em đã cố gắng hết mình để cố gắng đạt được kết quả tốt nhất, nhưng em vẫn rất mong nhận được thêm sự góp ý từ thầy cô và các bạn để tiếp tục hoàn thiện hơn trong tương lai.

Em xin chân thành cảm ơn!

MỤC LỤC

# DANH MỤC CÁC THUẬT NGỮ, KÝ HIỆU VÀ CÁC TỪ VIẾT TẮT

# 

# DANH MỤC HÌNH VẼ

# DANH MỤC BẢNG BIỂU

1. MỞ ĐẦU
2. **Tên đề tài:**

Tìm hiểu và ứng dụng Selenium Webdriver trong kiểm thử website bán bánh Trung Thu

1. **Lý do chọn đề tài**

Trong xu thế hội nhập hiện nay, việc nâng cao chất lượng sản phẩm để tạo niềm tin với khách hàng trở nên quan trọng và cấp thiết. Sản phẩm phần mềm phải được kiểm tra, kiểm thử kỹ lưỡng để đảm bảo không có lỗi và tuân thủ đúng yêu cầu của khách hàng. Kiểm thử thủ công giúp đánh giá tính trực quan, trải nghiệm người dùng tuy nhiên tiêu tốn nhiều thời gian và nguồn lực, dễ bỏ sót lỗi khi hệ thống ngày càng phức tạp. Vì thế cần kết hợp với kiểm thử tự động giúp tăng tốc độ kiểm thử, giảm thiểu sai sót khi thực hiện kiểm thử hồi quy và đảm bảo tính ổn định của hệ thống trên nhiều môi trường khác nhau. Kiểm thử tự động đang dần trở thành trợ thủ đắc lực cho đội ngũ phát triển phần mềm.

Trong đó, Selenium WebDriver là một công cụ kiểm thử tự động mã nguồn mở được sử dụng rộng rãi trên toàn cầu. Theo thống kê của Stack Overflow Developer Survey 2023, Selenium nằm trong top 5 công cụ kiểm thử tự động phổ biến nhất, được sử dụng bởi hơn 35% các kỹ sư kiểm thử và phát triển phần mềm.[1]

Với mục tiêu có thể áp dụng kiểm thử tự động trong thực tế, em đã quyết định thực hiện đề tài “Tìm hiểu và ứng dụng Selenium Webdriver trong kiểm thử website bán bánh Trung Thu”. Đề tài này giúp em nắm vững các kỹ thuật kiểm thử, cách thiết kế ca kiểm thử và kết hợp kiểm thử tự động với kiểm thử thủ công để tối ưu chất lượng phần mềm.

1. **Mục tiêu của đề tài**

* Hiểu được quy trình kiểm thử trong thực tế.
* Tìm hiểu và thực hành thiết kế ca kiểm thử và báo cáo kết quả nhằm đảm bảo chất lượng phần mềm.
* Tìm hiểu tính năng, cách cài đặt và sử dụng Selenium WebDriver kết hợp các công cụ và thư viện hỗ trợ khác trong kiểm thử tự động.
* Áp dụng các kỹ thuật kiểm thử và thực thi các kịch bản kiểm thử bằng Selenium WebDriver để kiểm thử chức năng cho trang web bán bánh Trung Thu Nguyệt Việt.

1. **Nội dung nghiên cứu**

Đề tài tập trung vào nghiên cứu và áp dụng các phương pháp kiểm thử phần mềm trong việc kiểm thử website bán bánh Trung Thu Nguyệt Việt với công cụ Selenium Webdriver. Các nội dung chính bao gồm:

* Lý thuyết cơ bản của kiểm thử phần mềm
* Phương pháp kiểm thử phần mềm tự động
* Kiểm thử tự động các chức năng với công cụ Selenium Webdriver
* Thư viện hỗ trợ TestNG
* Ứng dụng: Vận dụng các kiến thức kiểm thử vào việc xây dựng kịch bản kiểm thử và thực hiện kiểm thử cho website bán bánh Trung Thu

1. **Giới hạn và phạm vi của đề tài**

Đề tài được thực hiện trong khoảng thời gian 2 tháng, tính từ ngày 14/04/2025 đến 14/06/2025.

Phạm vi nghiên cứu đề tài bao gồm việc tìm hiểu lý thuyết cơ bản về kiểm thử, phương pháp kiểm thử tự động sử dụng công cụ Selenium Webdriver và ứng dụng trong kiểm thử website bán bánh Trung Thu với các chức năng cơ bản.

1. **Phương pháp thực hiện**

* Phương pháp phân tích tài liệu: Tìm hiểu các kiến thức tổng quan về kiểm thử phần mềm, kiểm thử ứng dụng web, kiểm thử tự động. Tìm hiểu cụ thể về công cụ Selenium WebDriver cũng như việc sử dụng công cụ này trong kiểm thử phần mềm
* Phương pháp nghiên cứu định tính: Tìm hiểu các dự án thực tế đã sử dụng Selenium WebDriver để kiểm thử ứng dụng web, phân tích cách thức áp dụng và hiệu quả của phương pháp này trong kiểm thử tự động.
* Phương pháp nghiên cứu định lượng: Thực hiện kiểm thử tự động trên hệ thống bán bánh Trung Thu với Selenium WebDriver.
* Phương pháp phân tích số liệu: Sử dụng các phương pháp thống kê để xử lý dữ liệu kiểm thử, tổng hợp kết quả một cách trực quan và đánh giá tính hiệu quả từ đó đưa ra đề xuất và hướng phát triển sau này.

1. **Bố cục đề tài**

**Chương 1:** **Tổng quan về kiểm thử phần mềm**

Chương này trình bày về các khái niệm, mục tiêu, quy trình của kiểm thử phần mềm. Bên cạnh đó còn đưa ra các mức độ kiểm thử và kỹ thuật thiết kế kịch bản kiểm thử.

**Chương 2:** **Công cụ kiểm thử Selenium WebDriver**

Trong chương này sẽ giới thiệu về công cụ Selenium WebDriver: đặc điểm, thành phần, hướng dẫn cài đặt và sử dụng, kết hợp với thư viện hỗ trợ TestNG.

**Chương 3:** **Ứng dụng Selenium WebDriver kiểm thử ứng dụng web**

Tại chương 3, em giới thiệu chức năng của website, biểu diễn kịch bản kiểm thử, thực thi kiểm thử và báo cáo kết quả kiểm thử tự động các chức năng chính của website.

# TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM

## Khái niệm kiểm thử phần mềm

Kiểm thử phần mềm được định nghĩa theo nhiều tiêu chuẩn khác nhau. Theo IEEE Standard 610.12-1990, kiểm thử phần mềm là “quá trình phân tích một sản phẩm phần mềm để phát hiện ra sự khác biệt giữa điều kiện hiện tại và điều kiện được yêu cầu (tức là lỗi) và đánh giá các tính năng của sản phẩm phần mềm đó. [2]

Trong khi đó, theo ISTQB Glossary, kiểm thử phần mềm lại được định nghĩa như sau: kiểm thử phần mềm là một quá trình bao gồm tất cả các hoạt động trong vòng đời, bao gồm cả tĩnh và động, liên quan đến việc lập kế hoạch, chuẩn bị và đánh giá sản phẩm phần mềm cũng như các sản phẩm liên quan để xác định rằng chúng đáp ứng yêu cầu đã xác định, chứng minh rằng chúng phù hợp với mục đích sử dụng và phát hiện lỗi. [3]

Từ đó có thể thấy, cả hai định nghĩa đều coi kiểm thử phần mềm là một quá trình quan trọng nhằm xác định lỗi và đảm bảo sản phẩm phần mềm đáp ứng chính xác, đầy đủ và đúng theo yêu cầu của khách hàng, yêu cầu sản phẩm đã đặt ra. Người thực hiện kiểm thử được gọi là kiểm thử viên (Tester).

Trong ngành công nghiệp phần mềm hiện đại, kiểm thử không chỉ là một bước riêng biệt sau khi lập trình xong mà được tích hợp trong toàn bộ chu trình phát triển (Software Development Life Cycle - SDLC). Việc kiểm thử phần mềm hiệu quả có thể giúp giảm chi phí, tăng độ tin cậy và cải thiện sự hài lòng của khách hàng.

Dựa vào chiến lược kiểm thử thì kiểm thử phần mềm bao gồm:

* Kiểm thử thủ công (Manual Testing): Là quá trình kiểm thử phần mềm mà tester kiểm thử bằng tay bao gồm viết kịch bản kiểm thử, nhập dữ liệu và thực hiện kiểm thử. Hiện nay, phần lớn các tổ chức và các công ty phần mềm đều kiểm thử thủ công là chủ yếu
* Kiểm thử tự động (Automation Testing): Là quá trình sử dụng các công cụ và phần mềm để thực hiện các thử nghiệm kiểm tra một cách tự động, thay vì phải thực hiện thủ công. Các kịch bản kiểm thử được tự động hóa và thực thi bởi các công cụ phần mềm, giúp cho việc kiểm tra tính đúng đắn của ứng dụng phần mềm trở nên hiệu quả hơn và tiết kiệm thời gian.

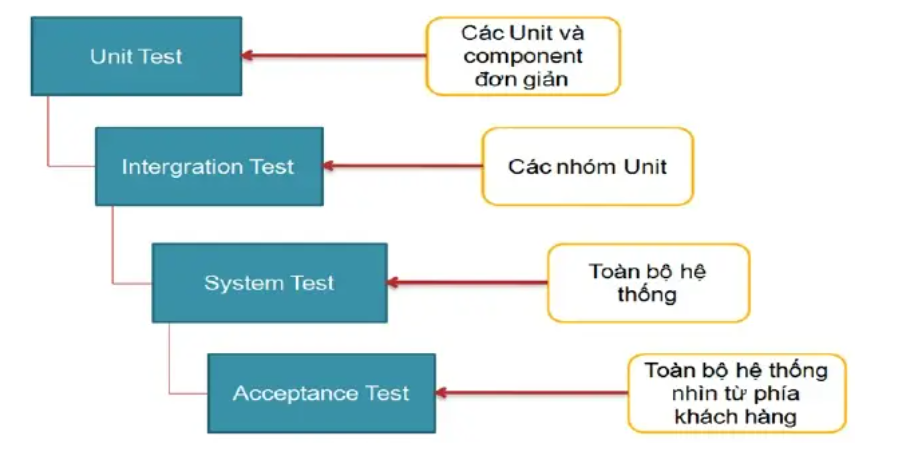
## Mục đích của kiểm thử phần mềm

Mục đích chính của kiểm thử phần mềm:

* Ngăn ngừa lỗi bằng cách đánh giá các chức năng theo yêu cầu và thiết kế.
* Xác minh xem tất cả các yêu cầu đã đáp ứng hay chưa.
* Kiểm tra xem đối tượng thử nghiệm đã hoàn thành chưa và xác nhận.
* Xây dựng niềm tin về mức độ chất lượng.
* Tìm ra lỗi, khiếm khuyết để giảm rủi ro của chất lượng phần mềm không đầy đủ.
* Cung cấp đủ thông tin cho các bên liên quan để họ có thể đưa ra quyết định sáng suốt, đặc biệt về chất lượng của đối tượng thử nghiệm.
* Tuân thủ các yêu cầu hoặc tiêu chuẩn về hợp đồng, pháp lý.
* Tạo các test case chất lượng cao, thực hiện kiểm thử hiệu quả và tạo ra các báo cáo vấn đề đúng và hữu dụng.

Kiểm thử phần mềm giữ vai trò then chốt trong việc đảm bảo chất lượng phần mềm. Một hệ thống không được kiểm thử kỹ lưỡng có thể gặp nhiều rủi ro trong quá trình vận hành thực tế, gây mất uy tín cho doanh nghiệp và ảnh hưởng đến người dùng. Đặc biệt trong các hệ thống quan trọng như ngân hàng, y tế, thương mại điện tử,... kiểm thử phần mềm được xem như một mắt xích sống còn trong chuỗi phát triển phần mềm.

## Các mức độ kiểm thử phần mềm

Các mức độ kiểm thử:

*Hình 1.2: Các mức độ kiểm thử*

*Nguồn: PhotoLina, Issuu (2024)[4]*

* Kiểm thử đơn vị (Unit testing): Kiểm thử đơn vị là một quy trình kiểm thử phần mềm cơ bản để kiểm thử các thành phần hệ thống phần mềm riêng lẻ. Cấp độ này kiểm tra kiến ​​trúc bên trong của từng đơn vị và đảm bảo chúng hoạt động như mong đợi. Người tiến hành kiểm thử đơn vị thường là lập trình viên.[5]
* Kiểm thử tích hợp (Integration testing): Sau khi kiểm thử đơn vị, kiểm thử tích hợp nhằm xác minh rằng các module hoặc thành phần riêng lẻ của hệ thống có thể hoạt động cùng nhau một cách chính xác sau khi được tích hợp. Người thực hiện thường là Tester.[5]
* Kiểm thử hệ thống (System testing): Kiểm thử hệ thống là đánh giá toàn diện về chức năng và hiệu suất chung của hệ thống phần mềm. Nó đảm bảo rằng toàn bộ hệ thống đáp ứng các tiêu chí phát triển. Mức độ kiểm thử này rất quan trọng để phát hiện các vấn đề trên toàn hệ thống như sự cố tương thích, tắc nghẽn hiệu suất và điểm yếu bảo mật. Người thực hiện thường là Tester.[5]
* Kiểm thử chấp nhận (Acceptance testing): Kiểm thử chấp nhận là giai đoạn cuối cùng của quá trình kiểm thử phần mềm, đảm bảo phần mềm hoạt động bình thường trong môi trường làm việc của người dùng. Nó xác định xem phần mềm có phù hợp với các yêu cầu kinh doanh và tiêu chí chấp nhận của người dùng như mong đợi hay không. Không giống như các cấp độ kiểm thử trước đó, kiểm thử chấp nhận đánh giá hệ thống theo quan điểm của người dùng chứ không phải chức năng kỹ thuật.[5]

## Phương pháp kiểm thử phần mềm

* + 1. **Phương pháp kiểm thử hộp đen**

Kiểm thử hộp đen là một phương pháp kiểm thử phần mềm mà ở đó việc thực hiện kiểm tra các chức năng của một phần mềm mà không quan tâm cấu trúc bên trong, thiết kế hoặc mã hóa của nó, tập trung vào kiểm tra hệ thống làm gì chứ không kiểm tra hệ thống làm như thế nào. Nó còn được gọi là kiểm thử hướng dữ liệu hay là kiểm thử hướng vào/ra. Thường được sử dụng trong cấp độ kiểm thử hệ thống, kiểm thử chấp nhận người dùng và loại hình kiểm thử chức năng.

Ưu điểm: Kiểm thử hộp đen dễ tiếp cận, áp dụng được cho nhiều mức độ kiểm thử và có thể thực hiện mà không cần biết mã nguồn hoặc cấu trúc nội bộ của phần mềm.

Nhược điểm: Khó bao quát hết tất cả các trường hợp kiểm thử và không tối ưu cho kiểm thử hiệu năng hoặc bảo mật.

Các kỹ thuật kiểm thử hộp đen được sử dụng bao gồm:

* ***Kỹ thuật phân vùng tương đương****:* là một kỹ thuật kiểm thử phần mềm có liên quan đến phân chia các giá trị đầu vào thành các phân vùng hợp lệ và không hợp lệ, sau đó chúng ta sẽ viết ra các kịch bản kiểm thử cho từng phần, chọn giá trị đại diện từ mỗi phân vùng làm dữ liệu thử nghiệm.

Phân tích cách làm gồm 2 bước:

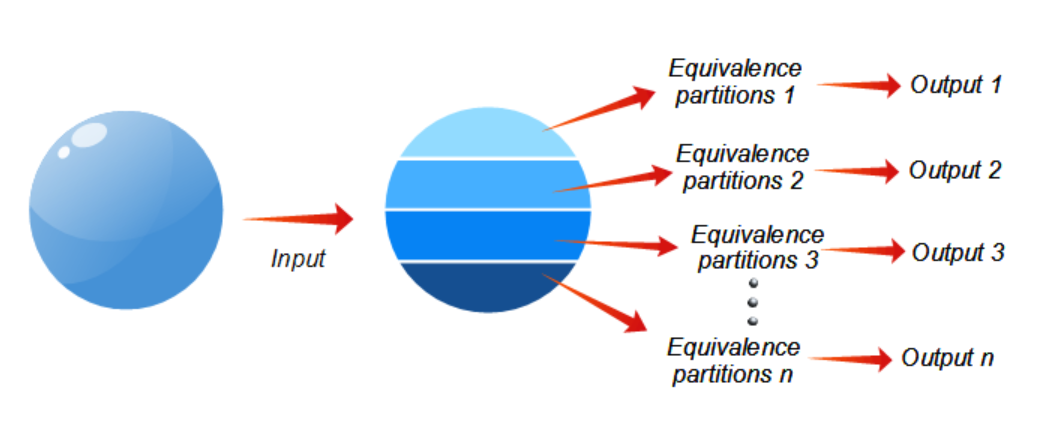
Bước 1: Xác định các lớp tương đương theo đầu vào hoặc đầu ra. Với mỗi điều kiện đầu vào hoặc đầu ra được mô tả trong đặc tả yêu cầu thì lấy ra: lớp thỏa mãn điều kiện và lớp không thỏa mãn điều kiện.

Bước 2: Lựa chọn giá trị đại diện

Ví dụ 1: Viết test case cho form Update giỏ hàng

Điều kiện ràng buộc: Trường Qty (Quantity) chỉ cho phép nhập số nguyên từ 1 đến 99. Trong đó từ 1-10 dành cho khách hàng mua lẻ (không được chiết khấu), từ 11-99 dành cho khách hàng mua sỉ (có chiết khấu).

* Ta có thể chia thành 5 phân vùng như sau khi kiểm thử: < 1; [1,10]; [11,99]; > 99 và nhập vào không phải số nguyên.



*Hình 1.3: Phân vùng tương đương*

*Nguồn: Trần Thị Hương Trang, viblo.asia (2017)[6]*

* ***Kỹ thuật phân tích giá trị biên***: là một kỹ thuật kiểm thử phần mềm có liên quan đến việc xác định biên (ranh giới) của điều kiện mô tả cho các giá trị đầu vào và chọn giá trị ở biên và bên cạnh giá trị biên làm dữ liệu kiểm thử.[6]

Phân tích cách làm gồm 2 bước:

Bước 1: Xác định các điểm biên [a, b]

Bước 2: Thiết kế test case dựa vào các điểm biên

Test Case 1: giá trị a – 1 => kết quả mong đợi: Không hợp lệ

Test Case 2: giá trị a => Kết quả mong đợi: Hợp lệ

Test Case 3: giá trị b => Kết quả mong đợi: Hợp lệ

Test Case 4: giá trị b + 1 => Kết quả mong đợi: Không hợp lệ

Ví dụ 2: Phân tích giá trị biên cho điểm nằm trong khoảng [0,100]:

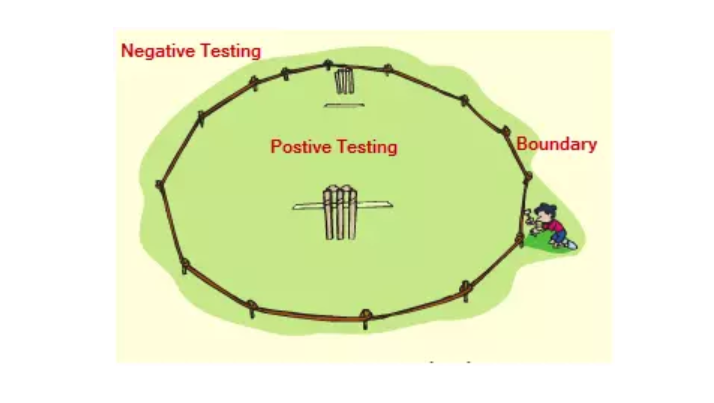
+ Giá trị nhỏ nhất: 0

+ Giá trị lớn nhất: 100

+ Giá trị nhỏ hơn giá trị nhỏ nhất: -1

+ Giá trị lớn hơn giá trị lớn nhất: 101

+ Giá trị trung bình: 50



*Hình 1.4: Phân tích giá trị biên*

*Nguồn: Trần Thị Hương Trang, viblo.asia (2017)[6]*

* ***Kỹ thuật bảng quyết định****:* Là một kỹ thuật hộp đen được sử dụng để kiểm tra các hành vi hệ thống với các cách kết hợp input đầu vào khác nhau. Đây là một cách tiếp cận có hệ thống, kết quả của các kết hợp và hành vi hệ thống tương ứng của chúng (output) sẽ được ghi lại dưới dạng bảng (Cause-Effect table). Đặc điểm của bảng là giá trị của các conditions chỉ có 2 giá trị True hoặc False và các cột (Rules) tương ứng với số lượng test case. Phương pháp này thường được áp dụng với các chức năng cần test có logic phức tạp và sự kết hợp nhiều điều kiện. [7]

Phân tích cách làm gồm 4 bước:

Bước 1: Xác định các conditions (điều kiện đầu vào/input)

Bước 2: Tính toán số rules = 2^condition

Bước 3: Đặt toàn bộ các kết hợp vào bảng

Bước 4: Giảm số lượng các kết hợp và quyết định test case

Các nguyên tắc xác định condition:

Nguyên tắc 1: Cần xác định mục đích cuối cùng mà bài toán hướng tới là gì? Những điều kiện ảnh hưởng đến mục đích cuối cùng của bài toán thì chính là conditions.

Nguyên tắc 2: Không lấy chiều ngược lại của 1 condition đã chọn.

Nguyên tắc 3: Chỉ lấy các condition đơn lẻ, không được gộp các condition với nhau.

Ví dụ 3: Xác định test case cho bài toán mua vé tàu có được giảm giá hay không với các điều kiện sau: Nếu bạn có thẻ đường sắt "over 60s" thì được giảm giá 34% trên tất cả các vé bạn mua. Nếu bạn có thẻ "family rail card”, nếu bạn đi cùng với trẻ em (dưới 16 tuổi), thì bạn sẽ được giảm 50%, ngược lại bạn sẽ được giảm 10%. Bạn chỉ được sử dụng 1 loại thẻ đường sắt

Bảng quyết định của bài toán trên được thể hiện như sau:

*Bảng 1.1: Ví dụ về Bảng quyết định*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Điều kiện | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 |
| Thẻ “over 60s” | T | T | T | T | F | F | F | F |
| Thẻ “family rail” | T | T | F | F | F | F | T | T |
| Đi cùng trẻ em | T | F | T | F | T | F | T | F |
| **Giảm giá?** | **50%** | **34%** | **34%** | **34%** | **0** | **0** | **50%** | **10%** |

* ***Kỹ thuật Pairwise testing:*** Là một kỹ thuật kiểm thử hộp đen được sử dụng để kiểm tra các sự kết hợp của các cặp điều kiện có thể từ các tham số liên quan. Thường được áp dụng cho các chức năng cần test có logic phức tạp có sự kết hợp của quá nhiều các tổ hợp điều kiện.

Phân tích cách làm gồm 5 bước:

Bước 1: Liệt kê toàn bộ điều kiện đầu vào/Input

Bước 2: Xác định số lượng test case thủ công = số lượng vùng giá trị lớn nhất của các biến \* số lượng vùng giá trị lớn thứ 2 trong số các biến

Bước 3: Thực hiện kết hợp 2 giá trị đầu vào lớn nhất bảng

Bước 4: Thực hiện mapping các giá trị cho đến khi bảng được điền hết

Bước 5: Tối ưu hóa các cặp điều kiện

*Chú ý:* Thứ tự các điều kiện sắp xếp giảm dần theo số lượng vùng giá trị. Số lần lặp lại các giá trị của conditions lớn nhất sẽ phụ thuộc vào số lượng vùng giá trị của điều kiện lớn thứ hai.

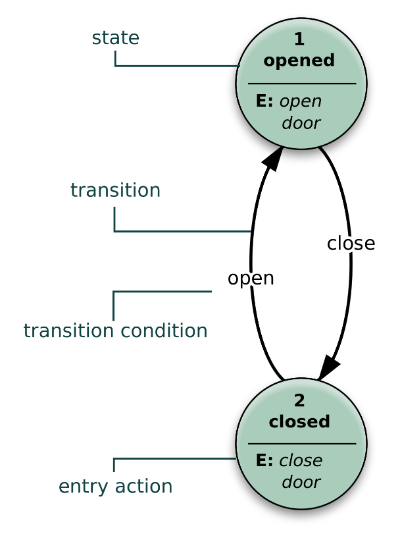
Ví dụ 4: Một ứng dụng gồm có: Listbox đơn giản với 10 phần tử (giả sử 0,1,2,3,4,5,6,7,8,9); Checkbox; Radio Nút; Textbox có thể nhận những giá trị nguyên dương từ 1 đến 100 và OK nút

Áp dụng pairwise testing và phân vùng tương đương ta có:

*Bảng 1.2: Ví dụ về Pairwise testing*

|  |  |  |  |
| --- | --- | --- | --- |
| **Texbox** | **List box** | **Checkbox** | **Radio Nút** |
| Int Valid | 0 | Checked | On |
| Int Valid | Other | Unchecked | Off |
| Int Invalid | 0 | Unchecked | On |
| Int Invalid | Other | Checked | Off |
| AlphaSpecialCharacter | 0 | Checked | Off |
| AlphaSpecialCharacter | Other | Unchecked | On |

* ***Sơ đồ chuyển đổi trạng thái:*** Là một kỹ thuật giúp thiết kế test case dựa vào mô hình (model) hoặc lược đồ (diagram) mô tả các trạng thái (state) và chuyển đổi (transition) của chúng. Thường áp dụng đối với các trường hợp có sự chuyển đổi trạng thái của các flow (thường sử dụng đối với việc kiểm thử End-To-End). Mô hình một sơ đồ chuyển trạng thái gồm 4 phần cơ bản:
* Các trạng thái hữu hạn
* Các sự dịch chuyển trạng thái
* Các sự kiện kích hoạt sự dịch chuyển trạng thái
* Các hành động là kết quả của sự dịch chuyển



*Hình 1.5: Mô hình sơ đồ chuyển trạng thái*

*Nguồn: Testing vn (2010)[7]*

* + 1. **Phương pháp kiểm thử hộp trắng**

Kiểm thử hộp trắng là một phương pháp kiểm thử phần mềm dựa trên việc phân tích cấu trúc bên trong, thiết kế và mã nguồn để làm rõ đầu vào, đầu ra của phần mềm. Chính vì thế nên kỹ thuật kiểm thử hộp trắng thường được thực hiện bởi các lập trình viên và được áp dụng chủ yếu cho mức độ kiểm thử đơn vị và kiểm thử tích hợp. Kỹ thuật kiểm thử hộp trắng thuộc loại hình kiểm thử cấu trúc.

Ưu điểm: Có thể cô lập được lỗi dẫn đến việc sửa lỗi dễ dàng hơn và có thể thực hiện kiểm thử sớm hơn.

Nhược điểm: Không phù hợp với tất cả mọi người, không phải kiểm thử viên nào cũng có thể thực hiện được.

## Quy trình kiểm thử phần mềm

Tùy vào từng tổ chức, hệ thống, ngữ cảnh, mức độ rủi ro của phần mềm mà quy trình kiểm thử phần mềm có thể gồm nhiều bước khác nhau. Nhưng nhìn chung mọi quy trình kiểm thử đều có những bước cơ bản sau:

Bước 1: Phân tích yêu cầu (Requirement Analysis).

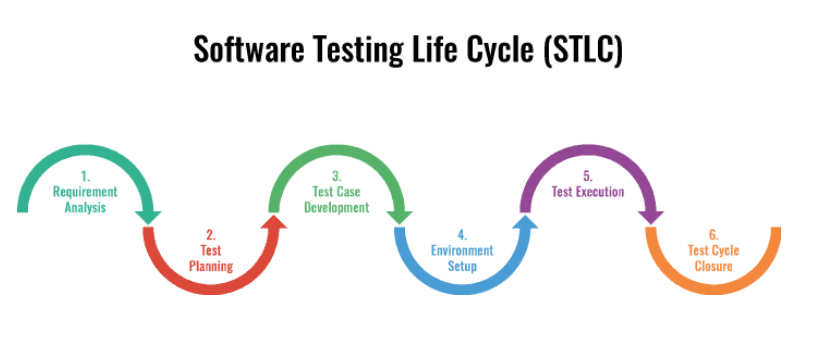
Bước 2: Lập kế hoạch kiểm thử (Test Planning)

Bước 3: Thiết kế kịch bản kiểm thử (Test Case Development)

Bước 4: Thiết lập môi trường kiểm thử (Environment Setup)

Bước 5: Thực hiện kiểm thử (Test Execution)

Bước 6: Kết thúc chu kỳ kiểm thử **(**Test Cycle Closure)[8]



*Hình 1.1: Quy trình kiểm thử phần mềm*

*Nguồn: HR1Tech (2021)[9]*

### Phân tích yêu cầu

Phân tích yêu cầu là giai đoạn đầu tiên trong quy trình kiểm thử phần mềm, gồm các hoạt động sau:

* Tìm hiểu, nghiên cứu và phân tích cụ thể các yêu cầu trong tài liệu đặc tả của dự án hoặc tài liệu khách hàng.
* Đưa ra câu hỏi với các bên liên quan như chuyên viên phân tích nghiệp vụ, người quản lý dự án, trưởng nhóm và khách hàng để hiểu chính xác hơn về yêu cầu của sản phẩm.

### Lập kế hoạch kiểm thử

Lập kế hoạch kiểm thử là quá trình tạo ra bản kế hoạch kiểm thử. Bản kế hoạch kiểm thử là tài liệu thường mô tả các nội dung cơ bản sau:

* Xác định phạm vi, rủi ro và mục đích của hoạt động kiểm thử.
* Xác định cách tiếp cận kiểm thử.
* Xác định quy định kiểm thử hoặc chiến lược kiểm thử.
* Xác định yêu cầu về nguồn nhân lực như con người, môi trường kiểm thử, thiết bị,...
* Lên lịch trình cho việc phân tích kiểm thử và thiết kế các trường hợp. kiểm thử, thực thi kiểm thử và đánh giá kết quả kiểm thử.
* Xác định các tiêu chí kết thúc việc kiểm thử.

### Thiết kế kịch bản kiểm thử

Thiết kế kịch bản kiểm thử là một bước hết sức quan trọng giúp đảm bảo rằng hệ thống được kiểm tra một cách có hệ thống và đầy đủ. Một kịch bản kiểm thử được xây dựng tốt sẽ giúp xác định các tình huống cụ thể, bao gồm đầu vào, hành động, kết quả mong đợi và điều kiện tiên quyết. Thiết kế kịch bản kiểm thử gồm các hoạt động sau:

* Kiểm tra lại tất cả các tài liệu để xác định công việc cần làm, các công việc có khác gì so với dự án trước khách hàng đưa cho.
* Viết kịch bản kiểm thử (test case hoặc checklist).
* Chuẩn bị dữ liệu kiểm thử.
* Thành viên trong đội kiểm thử đánh giá lại test case/checklist.

### Thiết lập môi trường kiểm thử

Thiết lập môi trường kiểm thử nhằm đảm bảo rằng quá trình kiểm thử được thực hiện trong điều kiện tương đồng với môi trường thực tế, giúp phát hiện và khắc phục lỗi một cách chính xác. Việc cài đặt môi trường kiểm thử gồm các hoạt động: Quyết định môi trường kiểm thử, chuẩn bị một số ca kiểm thử để kiểm tra xem môi trường cài đặt đã sẵn sàng cho việc kiểm thử hay chưa.

### Thực hiện kiểm thử

Thực hiện kiểm thử là quá trình phát triển, chạy thử một phần chức năng hay cả một hệ thống dựa trên các sản phẩm, tài liệu đã chuẩn bị ở các bước trên nhằm đưa ra kết quả thực tế. Các hoạt động chính trong giai đoạn thực hiện kiểm thử này bao gồm:

* Thực hiện các ca kiểm thử như thiết kế và mức độ ưu tiên đã đưa ra trên môi trường đã được cài đặt.
* So sánh với kết quả mong đợi sau báo cáo các lỗi xảy ra lên công cụ quản lý lỗi.
* Kiểm thử lại và kiểm thử hồi quy để xác minh các lỗi đã được sửa.
* Đo và phân tích tiến độ.
* Báo cáo thường xuyên cho người quản lý dự án và khách hàng về tình hình thực hiện dự án.

### Đóng chu trình kiểm thử

Đóng chu trình kiểm thử là giai đoạn cuối cùng trong quy trình kiểm thử phần mềm với mục tiêu chính là thu thập dữ liệu nhằm cung cấp cho việc bàn giao sản phẩm phần mềm, lưu trữ dữ liệu và phân tích cho các hoạt động cải tiến sau này. Hoạt động kết thúc kiểm thử thường thực hiện các công việc:

* Tổng kết, báo cáo kết quả về việc thực thi test case.
* Đánh giá các tiêu chí hoàn thành như phạm vi kiểm tra, chất lượng, chi phí, thời gian, mục tiêu kinh doanh quan trọng.
* Thảo luận những điểm tốt, điểm chưa tốt và rút ra bài học kinh nghiệm.

## Kiểm thử tự động

Kiểm thử tự động (Automation Testing) là quá trình sử dụng các công cụ phần mềm để thực thi các kịch bản kiểm thử đã được định sẵn nhằm xác minh chất lượng và độ ổn định của hệ thống phần mềm. Không giống như kiểm thử thủ công, nơi các thao tác được thực hiện bởi con người, kiểm thử tự động sử dụng mã lệnh để mô phỏng thao tác của người dùng trên giao diện ứng dụng hoặc để kiểm tra logic bên trong hệ thống. [10]

Kiểm thử tự động được triển khai thông qua các công cụ hỗ trợ viết test script, thực thi test case và so sánh kết quả thực tế với kết quả mong đợi. Các công cụ kiểm thử tự động có thể hoạt động trên nhiều nền tảng như giao diện người dùng (UI), lập trình giao tiếp ứng dụng (API), hoặc kiểm thử đơn vị (Unit testing). Kiểm thử tự động đóng vai trò quan trọng trong quy trình phát triển phần mềm hiện đại, đặc biệt trong các mô hình phát triển như Agile hay DevOps, nơi mà việc kiểm thử thường xuyên và liên tục là cần thiết.

* Ưu điểm của kiểm thử tự động:
* Tiết kiệm thời gian: Sau khi thiết lập, kiểm thử tự động có thể lặp đi lặp lại nhiều lần mà không cần can thiệp thủ công.
* Tăng độ chính xác: Giảm thiểu lỗi do con người gây ra trong quá trình thực hiện kiểm thử.
* Hiệu quả với các test lặp đi lặp lại: Tự động hóa phù hợp với các trường hợp cần chạy kiểm thử hồi quy thường xuyên hoặc test trên nhiều môi trường.
* Khả năng mở rộng cao: Dễ dàng thực thi nhiều kịch bản kiểm thử song song trên nhiều nền tảng.
* Hỗ trợ CI/CD: Tích hợp dễ dàng với các công cụ tích hợp liên tục giúp kiểm tra tự động sau mỗi lần build.
* Nhược điểm của kiểm thử tự động:
* Chi phí ban đầu cao: Việc xây dựng bộ test script và môi trường kiểm thử tự động đòi hỏi thời gian và kỹ năng lập trình.
* Không linh hoạt bằng kiểm thử thủ công: Một số kịch bản đòi hỏi nhận thức và đánh giá của con người (như kiểm thử giao diện đồ họa) vẫn cần kiểm thử thủ công.
* Bảo trì phức tạp: Khi phần mềm thay đổi nhiều, các test script cũng phải cập nhật tương ứng, có thể gây tốn kém thời gian và tốn tài nguyên.

Vì vậy, kiểm thử tự động chỉ phù hợp trong các trường hợp sau:

* Khi cần thực hiện các kiểm thử hồi quy thường xuyên sau mỗi lần cập nhật phần mềm.
* Khi có khối lượng test lớn và lặp lại nhiều lần, ví dụ kiểm thử tính năng giỏ hàng trên nhiều loại trình duyệt.
* Khi phần mềm sử dụng CI/CD và cần đảm bảo không có lỗi sau mỗi lần build.
* Khi cần kiểm thử hiệu năng, độ chịu tải hoặc chạy test xuyên suốt 24/7 mà không thể dùng sức người

## Kiểm thử trên nền tảng web

### Khái quát về kiểm thử trên nền tảng web

Trong bối cảnh kinh doanh hiện đại, việc triển khai các ứng dụng web là yếu tố quan trọng giúp doanh nghiệp mở rộng phạm vi tiếp cận khách hàng và tối ưu hóa cơ hội phát triển. Các ứng dụng web đóng vai trò then chốt trong việc kết nối, trao đổi thông tin và hỗ trợ hoạt động của nhiều tổ chức.

Để đạt được hiệu quả cao, ứng dụng web cần đảm bảo chất lượng, hiệu suất tối ưu cũng như mang lại trải nghiệm người dùng tốt. Không giống như các ứng dụng di động hay desktop, ứng dụng web phải hoạt động trên nhiều nền tảng khác nhau, đòi hỏi sự linh hoạt trong cấu hình và thao tác sử dụng. Điều này đặt ra nhiều thách thức đối với các nhà phát triển trong việc đảm bảo tính ổn định và tương thích của sản phẩm.

Do đó, việc xây dựng một chiến lược kiểm thử toàn diện là điều cần thiết để giảm thiểu rủi ro và nâng cao chất lượng ứng dụng web, giúp sản phẩm hoạt động ổn định trên nhiều môi trường khác nhau.

### Các hạng mục kiểm thử Website

Kiểm thử chức năng: Kiểm tra tính đúng đắn và hiệu quả của các chức năng trong ứng dụng web, bao gồm các liên kết, form nhập liệu, tìm kiếm và các tính năng khác. Mục tiêu là phát hiện và báo cáo về bất kỳ lỗi nào xuất hiện khi sử dụng các chức năng này.

Kiểm thử giao diện: So sánh thiết kế UI với thực tế trên trình duyệt, kiểm tra mỗi thành phần trên trang web để đảm bảo đúng với bản thiết kế. Kiểm tra các liên kết và menu, đảm bảo chúng hoạt động đúng và không trỏ về chính nó.

Kiểm thử cookie và session: Kiểm tra tính an toàn của việc đăng nhập bằng cách thử các tập tin cookie khác nhau và kiểm tra khả năng bảo mật của ứng dụng khi xóa các tập tin cookie.

Kiểm thử nội dung đa ngôn ngữ: Đảm bảo dữ liệu được dịch sang các ngôn ngữ khác nhau mà không mất đi ý nghĩa hoặc gây ra lỗi.

Kiểm thử cơ sở dữ liệu: Kiểm tra kết nối và truy vấn cơ sở dữ liệu để đảm bảo dữ liệu chính xác và các chức năng hoạt động đúng.

### Một số công cụ hỗ trợ kiểm thử website

Một số công cụ kiểm thử chức năng:

* BrowserStack: Kiểm thử chức năng của ứng dụng web trên nhiều trình duyệt khác nhau và kiểm tra khả năng hiển thị responsive.
* Ranorex: Công cụ tự động kiểm thử cho ứng dụng web, desktop và di động.
* Selenium: Công cụ mạnh mẽ cho kiểm thử tự động ứng dụng web, được phát triển và hỗ trợ bởi Selenium team từ Google.

Một số công cụ kiểm thử hiệu năng:

* WebLoad: Cho phép kiểm tra khả năng chịu tải và độ chịu lỗi của ứng dụng web bằng nhiều công nghệ khác nhau.
* Apache JMeter: Công cụ mã nguồn mở để kiểm tra hiệu năng và chức năng của ứng dụng web.
* NeoLoad: Đo và phân tích hiệu suất của ứng dụng web bằng cách tăng lưu lượng truy cập.

Một số công cụ kiểm thử bảo mật:

* Burp Suite: Công cụ kiểm tra lỗ hổng bảo mật cho ứng dụng web với nhiều tính năng tích hợp như Spider và Intruder.
* OWASP Zed Attack Proxy: Đánh giá an ninh mạng và bảo mật của các ứng dụng web.
* Nikto: Máy quét lỗ hổng máy chủ web mã nguồn mở để phát hiện cài đặt và cấu hình lỗi.

**Kết luận chương 1:**

Chương 1 đã trình bày tổng quan về kiểm thử phần mềm, bao gồm khái niệm, mục đích, các mức độ và phương pháp, kỹ thuật kiểm thử phổ biến. Nội dung chương giúp làm rõ vai trò của kiểm thử phần mềm cũng như các phương pháp áp dụng trong thực tế. Việc giới thiệu về kiểm thử tự động và kiểm thử ứng dụng web là bước đệm quan trọng cho các chương tiếp theo.

# CÔNG CỤ KIỂM THỬ SELENIUM WEBDRIVER

## Giới thiệu về công cụ Selenium

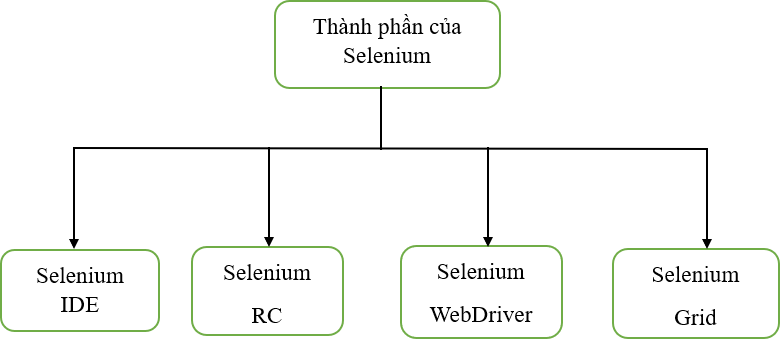
### Giới thiệu chung

Selenium (thường được viết tắt là SE) là một bộ kiểm thử tự động mã nguồn mở được phát triển bởi Jason Huggins, sau đó tiếp tục phát triển bởi nhóm ThoughtWorks vào năm 2004. Selenium là công cụ miễn phí cho các ứng dụng web trên các trình duyệt và nền tảng khác nhau.

Selenium hỗ trợ kiểm tra hầu hết trên các trình duyệt phổ biến hiện nay như Chrome, Internet Explorer, Firefox, Safari, v.v cũng như các hệ điều hành phổ biến như Windows, MacOS, Linux. Selenium hỗ trợ một số lớn các ngôn ngữ lập trình như Java, C#, Perl, PHP, Python, Ruby, v.v giúp tăng linh hoạt trong quá trình kiểm thử. Không những vậy, Selenium có thể kết hợp thêm với một số công cụ khác để nâng cao hiệu quả kiểm thử tự động như:

* TestNG, Junit: Hỗ trợ quản lý và thực thi các kịch bản kiểm thử.
* Maven, Gradle: Quản lý thư viện và tự động hóa quy trình build dự án.
* Allure Report, Extent Reports, Cucumber Report: Tạo báo cáo kiểm thử chi tiết và trực quan.
* Jenkins, GitHub Actions: Tích hợp CI/CD để tự động chạy kiểm thử khi có thay đổi mã nguồn.
* Appium: Hỗ trợ kiểm thử tự động ứng dụng di động sử dụng cùng cú pháp với Selenium.
* Docker: Chạy kiểm thử song song trên nhiều trình duyệt và môi trường khác nhau.
* BrowserStack, Sauce Labs: Kiểm thử trên đám mây, hỗ trợ nhiều trình duyệt và thiết bị.

### Thành phần của Selenium

Selenium không chỉ là một công cụ duy nhất mà nó là một bộ phần mềm, mỗi bộ nó cung cấp các nhu cầu thử nghiệm khác nhau của một tổ chức. Selenium gồm có 4 thành phần sau: 

*Hình STYLEREF 1 \s 2. SEQ Hình \\* ARABIC \s 1 1: Thành phần của Selenium*

*Hình 2.1: Thành phần của Selenium*

**Selenium Integrated Development Environment (Selenium IDE)**: là framework đơn giản nhất trong bộ Selenium. Nó là một browser plugin có thể cài đặt dễ dàng như các plugin khác. [11] Đây là một công cụ cho phép ghi (record) và phát lại (playback) một test script. Selenium IDE cho phép xuất ra kịch bản đã thu dưới nhiều loại ngôn ngữ lập trình khác nhau như Java, PHP, C#, Ruby, Perl hay Python.

**Selenium Remote Control (Selenium RC)**: là 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. [11]

**Selenium WebDriver**: là sự kế thừa từ Selenium RC, nó thực hiện một cách tiếp cận hiện đại và ổn định hơn trong việc tự động hóa các hành động của trình duyệt. WebDriver không giống như Selenium RC, không dựa vào JavaScript cho tự động hóa. Nó kiểm soát trình duyệt bằng cách giao tiếp trực tiếp với nó. [11]

**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. [11]

## Công cụ kiểm thử tự động Selenium WebDriver

### Tổng quan về Selenium WebDriver

Selenium WebDriver là một trong những thành phần quan trọng và được sử dụng rộng rãi nhất trong bộ công cụ Selenium. Đây là một công cụ kiểm thử tự động mạnh mẽ, cho phép tương tác trực tiếp với trình duyệt web mà không cần thông qua bất kỳ thành phần trung gian nào. WebDriver được phát triển nhằm thay thế Selenium RC, khắc phục những hạn chế của phiên bản trước và mang lại hiệu suất kiểm thử cao hơn.

Không giống như Selenium IDE, vốn chỉ hoạt động trên một số trình duyệt nhất định, Selenium WebDriver hỗ trợ kiểm thử trên nhiều trình duyệt và nền tảng hiện đại như Chrome, Firefox, Edge, Safari và Opera. Đồng thời, WebDriver không yêu cầu phải khởi động máy chủ Selenium trước khi thực thi các tập lệnh kiểm thử như Selenium RC, giúp đơn giản hóa quá trình kiểm thử và nâng cao tốc độ thực thi.

Với khả năng mô phỏng hành vi người dùng một cách chân thực, Selenium WebDriver cho phép tự động hóa các thao tác như nhập liệu, nhấp chuột, cuộn trang, xác minh nội dung trang web và thực hiện các kịch bản kiểm thử phức tạp. Nhờ đó, WebDriver trở thành công cụ lý tưởng cho việc kiểm thử chức năng, kiểm thử giao diện và kiểm thử hồi quy trên các ứng dụng web.

### Kiến trúc của Selenium WebDriver

Selenium WebDriver là công cụ kiểm thử tự động mạnh mẽ cho các ứng dụng web, hoạt động theo mô hình client – server. Kiến trúc của nó được thiết kế để cung cấp khả năng kiểm soát trực tiếp và hiệu quả đối với trình duyệt web.

Selenium WebDriver hoạt động bằng cách giao tiếp trực tiếp với trình duyệt thông qua trình điều khiển trình duyệt (browser driver). Trình điều khiển trình duyệt là một chương trình riêng biệt hoạt động như một cầu nối giữa Selenium WebDriver và trình duyệt web. Mỗi trình duyệt (Chrome, Firefox, Safari, v.v.) đều có trình điều khiển trình duyệt riêng. Các thành chính trong kiến trúc Selenium WebDriver bao gồm:

* Thư viện client của Selenium (Selenium Client Libraries): Đây là các thư viện ngôn ngữ lập trình (Java, C#, Python, v.v.) cung cấp các API để viết các kịch bản kiểm thử. Các thư viện này cho phép người dùng tương tác với trình duyệt, thực hiện các hành động như nhấp chuột, nhập liệu, v.v.
* WebDriver Language Bindings: Là các thư viện hỗ trợ đa ngôn ngữ lập trình cho phép người dùng có thể viết kịch bản kiểm thử bằng nhiều ngôn ngữ khác nhau.
* Trình điều khiển trình duyệt (Browser Driver): Đây là các trình điều khiển cụ thể cho từng trình duyệt, giúp Selenium WebDriver giao tiếp với trình duyệt. Ví dụ ChromeDriver cho Chrome, GeckoDriver cho Firefox, v.v.
* Trình duyệt (Browser): Selenium WebDriver hỗ trợ nhiều trình duyệt web phổ biến, cho phép kiểm thử trên nhiều nền tảng khác nhau.

Cách thức hoạt động của Selenium WebDriver như sau: Khi một kịch bản kiểm thử được thực thi, các lệnh Selenium được gửi đến trình điều khiển trình duyệt. Trình điều khiển trình duyệt diễn giải các lệnh này và giao tiếp với trình duyệt tương ứng. Trình duyệt thực hiện các hành động được chỉ định và trả kết quả về cho trình điều khiển trình duyệt. Trình điều khiển trình duyệt sau đó chuyển kết quả trở lại kịch bản kiểm thử.

### Các tính năng của Selenium WebDriver

*Hỗ trợ đa trình duyệt*:Selenium WebDriver được thiết kế để hỗ trợ kiểm thử trên nhiều trình duyệt web phổ biến như Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge và cả trình duyệt Internet Explorer (dù Selenium đã chuyển sang sử dụng Edge Chromium từ phiên bản 3.141.59). [12] Điều này đảm bảo rằng cách kịch bản kiểm thử có thể chạy trên cách trình duyệt khác nhau một cách đồng nhất và đáng tin cậy, giúp đảm bảo tính tương thích của ứng dụng web trên các nền tảng khác nhau.

*Hỗ trợ đa ngôn ngữ lập trình*:Selenium WebDriver cung cấp khả năng viết kịch bản kiểm thử bằng một loạt các ngôn ngữ lập trình phổ biến. Các ngôn ngữ lập trình được Selenium WebDriver hỗ trợ: Java, C#, Python, PHP, Pearl, Ruby, v.v. Điều này tạo điều kiện linh hoạt cho các nhà phát triển và kiểm thử viên lựa chọn ngôn ngữ phù hợp với nhu cầu cụ thể của dự án hoặc tổ chức.

*Tương tác trực tiếp với trình duyệt*:WebDriver giao tiếp trực tiếp với trình duyệt web, cho phép người dùng mô phỏng các hành động như nhấp chuột, nhập liệu, cuộn trang và gửi biểu mẫu.

*Khả năng định vị phần tử linh hoạt*:Selenium WebDriver cung cấp nhiều phương pháp để định vị các phần tử web, bao gồm ID, tên, lớp, thẻ, CSS selector và Xpath. Điều này giúp ta dễ dàng tìm và tương tác với các phần tử trên trang web.

*Kiểm tra tính tương tác của người dùng*:Công cụ cho phép ta kiểm tra tính tương tác của người dùng với các phần tử trên trang web, chẳng hạn như kiểm tra xem một nút có nhấp vào hay không hoặc một trường văn bản có được nhập liệu chính xác hay không.

*Chụp ảnh màn hình*:Selenium WebDriver có thể chụp ảnh màn hình của trang web, giúp người kiểm thử ghi lại trạng thái của ứng dụng web trong quá trình kiểm tra.

*Tích hợp với Selenium Grid*:Selenium WebDriver có thể được tích hợp với Selenium Grid, cho phép chạy các kiểm tra song song trên nhiều máy và trình duyệt khác nhau. Như thế sẽ giúp tăng tốc độ kiểm thử và giảm thời gian thực hiện.

*Hỗ trợ kiểm thử trên thiết bị di động*:Công cụ có thể dùng để kiểm thử trên các trình duyệt di động bằng cách sử dụng Appium, một thư viện mã nguồn mở cho phép Selenium tương tác với các ứng dụng di động.

Tóm lại, Selenium WebDriver là một công cụ kiểm thử tự động mạnh mẽ và linh hoạt, cung cấp nhiều tính năng để kiểm tra các ứng dụng web trên nhiều trình duyệt và nền tảng khác nhau.

### Ưu điểm và nhược điểm của Selenium WebDriver

***Ưu điểm*:**

* Mã nguồn mở: Selenium WebDriver là một phần mềm miễn phí, dễ sử dụng và mang lại lợi ích trong việc kiểm thử phần mềm tự động.
* Tương thích nhiều trình duyệt: Selenium WebDriver tương thích với gần như tất cả các trình duyệt web phổ biến hiện nay, đảm bảo khả năng kiểm thử đa nền tảng.
* Hỗ trợ đa ngôn ngữ lập trình: Selenium WebDriver hỗ trợ gần hết các ngôn ngữ lập trình phổ biến hiện nay (C#, Java, Python, Ruby, Perl, v.v).
* Thân thiện, dễ sử dụng: Xét về độ thân thiện với người dùng, Selenium WebDriver vượt trội so với các thành phần khác của Selenium. Những câu lệnh của nó khá dễ nhớ và tiện dụng. Selenium cũng cung cấp cho người dùng nhiều tài liệu hướng dẫn cần thiết.
* Tốc độ thực thi nhanh: Selenium WebDriver tận dụng khá tốt khả năng hỗ trợ tự động hóa của các trình duyệt web. Tốc độ thực thi của Selenium WebDriver sẽ nhanh hơn nhiều nếu so với các công cụ khác trong bộ Selenium. [12]

***Nhược điểm*:**

* Chỉ hỗ trợ ứng dụng web: Selenium WebDriver chỉ hỗ trợ kiểm thử trên các ứng dụng web, hạn chế sự đa dạng trong kiểm thử ứng dụng desktop hoặc di động.
* Đòi hỏi kinh nghiệm lập trình: Sử dụng Selenium WebDriver đòi hỏi tester phải có kiến thức vững về một ngôn ngữ lập trình cụ thể và kinh nghiệm sử dụng các công cụ kiểm thử tự động.
* Thiếu tính năng báo cáo tự động: Là cung cụ kiểm thử nhưng Selenium Webdriver chưa có sẵn chức năng báo cáo tự động nên việc khắc phục sự cố và sửa lỗi khó khăn hơn. Điều này đồng nghĩa với việc kiểm thử viên sẽ mất nhiều thời gian hơn để viết báo cáo kiểm thử. [12]

## Các công cụ và thư viện hỗ trợ Selenium WebDriver

Selenium WebDriver là một công cụ mạnh mẽ để tự động hóa kiểm thử ứng dụng web. Tuy nhiên, để tối ưu hóa quá trình kiểm thử và nâng cao hiệu quả, có thể kết hợp Selenium WebDriver với các công cụ và thư viện sau:

### Công cụ Eclipse

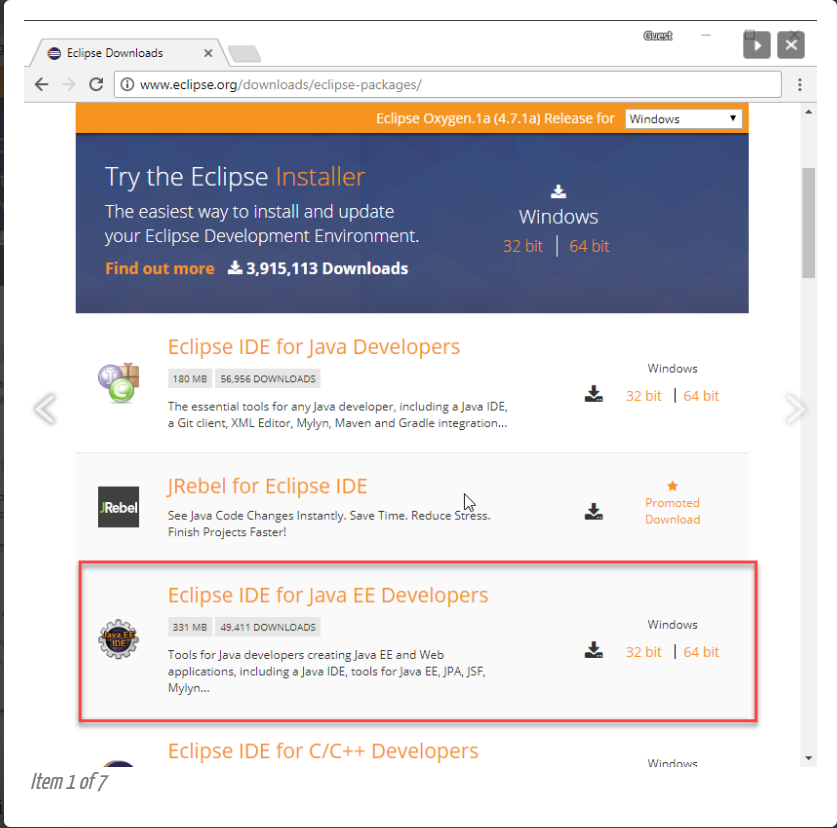
Eclipse là một môi trường phát triển tích hợp (IDE – Integrated Development Environment) mã nguồn mở, được thiết kế chủ yếu để lập trình ngôn ngữ Java. Eclipse được phát triển bởi tổ chức Eclipse Foundation và hiện là một trong những IDE phổ biến nhất trong cộng đồng lập trình viên Java trên toàn thế giới.

Một số tính năng nổi bật của Eclipse bao gồm:

* Tự động hoàn thành mã nguồn (code completion).
* Tích hợp trình gỡ lỗi (debugger) mạnh mẽ.
* Quản lý dự án theo cấu trúc thư mục rõ ràng.
* Hỗ trợ tích hợp với các công cụ kiểm thử như JUnit và Selenium.
* Tích hợp hệ thống quản lý phiên bản như Git.

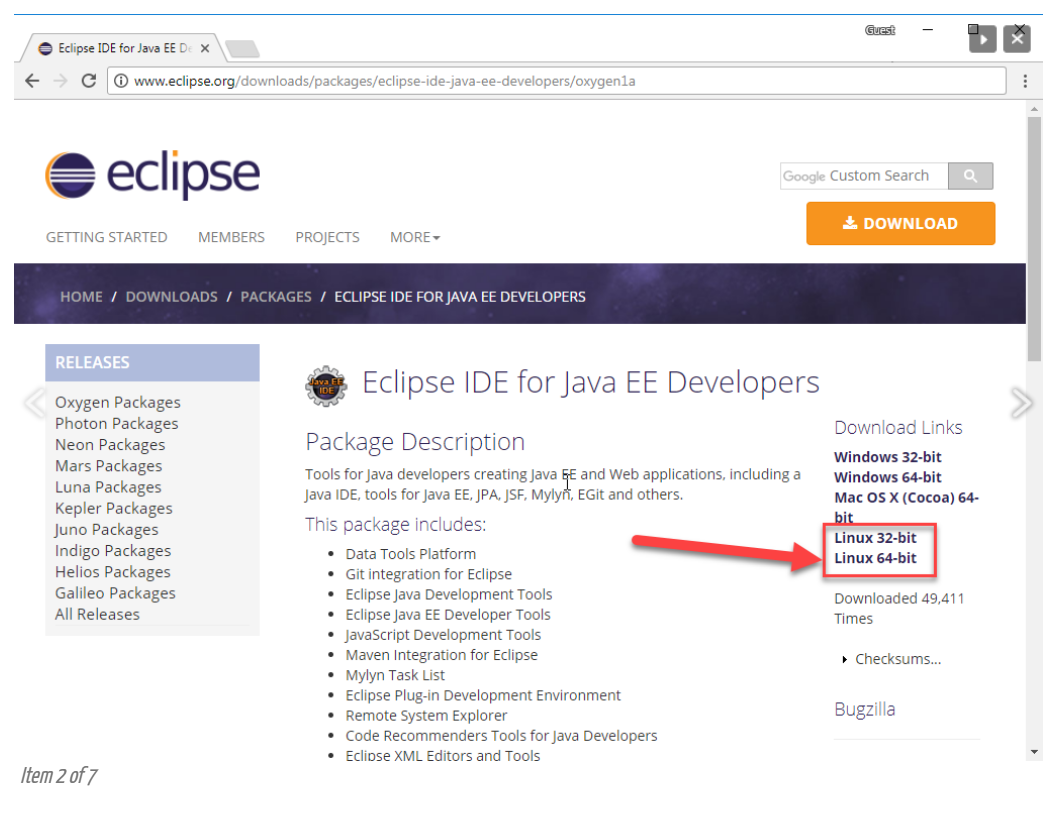
Cách cài đặt Eclipse:

* Đầu tiên download Eclipse tại: <http://www.eclipse.org/downloads/eclipse-packages/>
* Có rất nhiều các gói để download, chúng chỉ khác nhau về các plugin có sẵn trong đó, nên download gói Eclipse IDE for Java EE Developers.



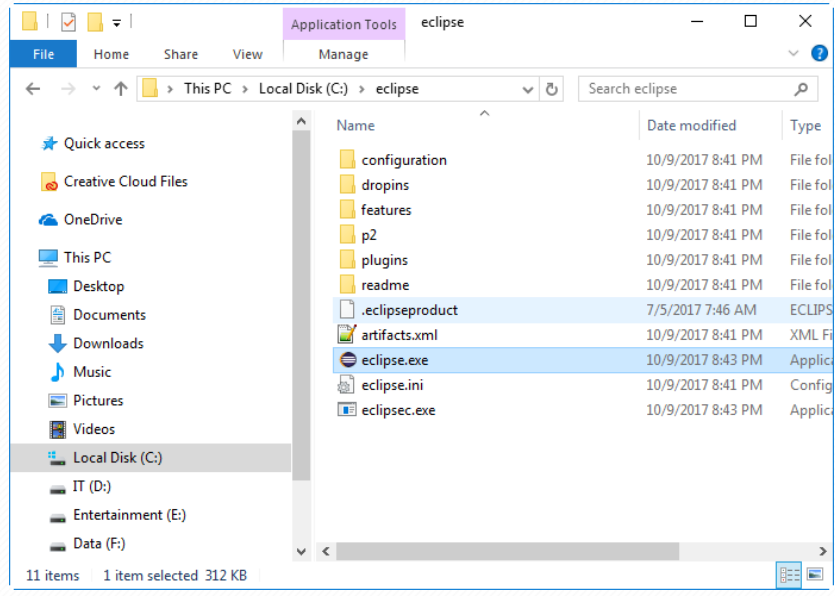
*Hình 2.2. Download Eclipse*

* Chọn phiên bản ứng với hệ điều hành: 32 bit hoặc 64 bit rồi download.



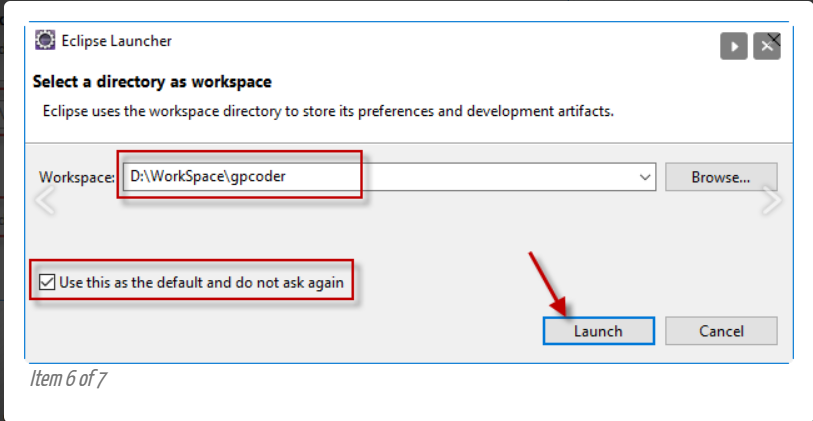
*Hình 2.3. Chọn phiên bản Eclipse tương ứng với hệ điều hành*

* Sau khi download xong, giải nén tập thư mục vừa download rồi run file eclipse.exe



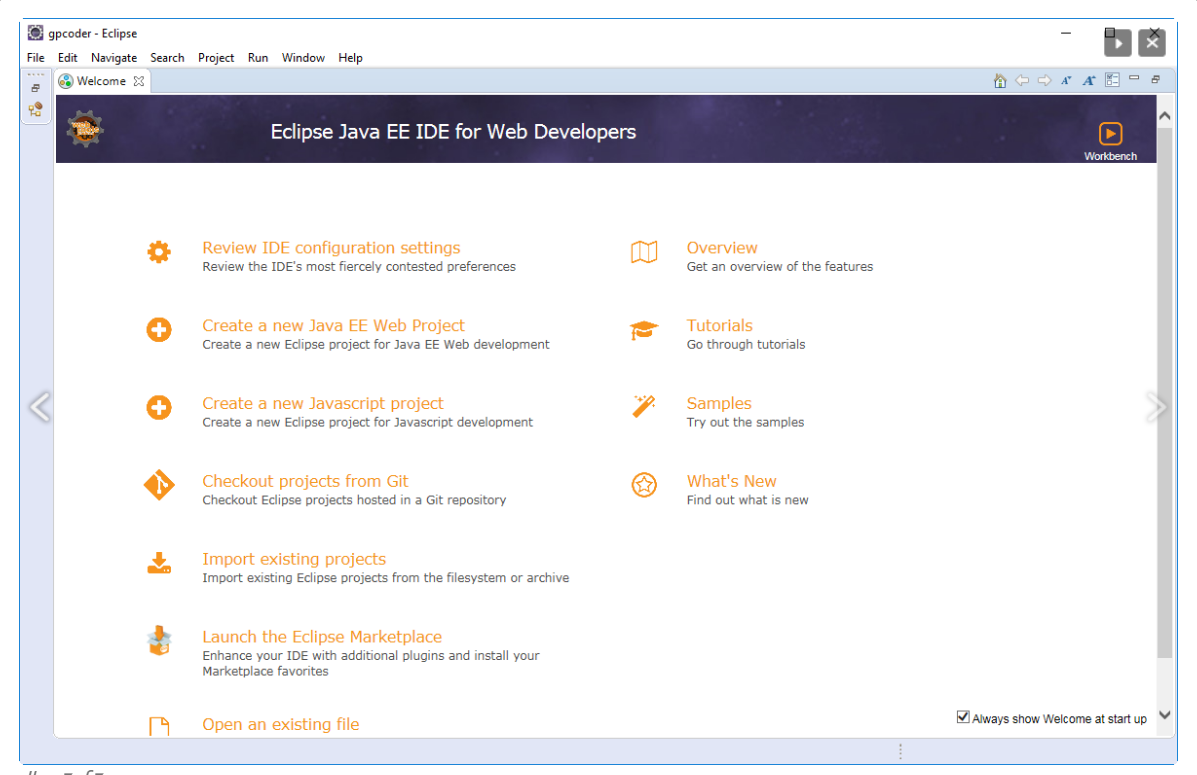
*Hình 2.4. Run file eclipse.exe vừa giải nén*

* Chọn workplace:



*Hình 2.5. Chọn workplace*

* Cài đặt thành công:



*Hình 2.6. Cài đặt Eclipse thành công*

### Thư viện TestNG

* TestNG (Testing Next Generation) là một framework kiểm thử Java phổ biến, cung cấp nhiều tính năng hữu ích cho việc quản lý và thực thi các trường hợp kiểm thử.
* Cung cấp các annotation như @Test, @BeforeMethod, @AfterMethod để đánh dấu các phương thức là các bước kiểm thử, trước hoặc sau toàn bộ bộ kiểm thử hoặc nhóm kiểm thử. [13]
* Quản lý thời gian: Có thể cấu hình TestNG để thực hiện các bước kiểm thử theo thứ tự nhất định hoặc song song, cũng như đặt thời gian timeout cho mỗi kiểm thử.
* Báo cáo và ghi nhật ký: TestNG tạo ra báo cáo chi tiết về kết quả kiểm thử, bao gồm thông tin về số lượng kiểm thử thành công và thất bại, thời gian chạy, và các chi tiết về lỗi nếu có.

### Trình điều khiển trình duyệt ChromeDriver

* ChromeDriver là một trình điều khiển web cho phép Selenium WebDriver điều khiển trình duyệt Google Chrome. Nó hoạt động như một cầu nối giữa Selenium WebDriver và Chrome, cho phép tự động hóa

# TÀI LIỆU THAM KHẢO

1. Stack Overflow, “Developer Survey Results 2023,” [Trực tuyến]. Có tại: <https://survey.stackoverflow.co/2023/>. [Truy cập: 8-6-2025].
2. IEEE, *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Std 610.12-1990, IEEE Standards Board, 1990.
3. ISTQB, *Giáo trình Kiểm thử phần mềm cơ bản,* Brussels: ISTQB, 2018.
4. PhotoLina, "*A Guide to Software Testing,*" *Issuu, 2024.*
5. Quoc Bao, “*What are 4 Levels Of Testing In Software Testing? With Examples,*” [Trực tuyến]. Có tại: <https://reliasoftware.com/blog/levels-of-testing-in-software-testing>. [Truy cập: 8-6-2025].
6. Trần Thị Hương Trang, *“Tìm hiểu về kỹ thuật phân tích giá trị biên và phân vùng tương đương trong kiểm thử hộp đen,” Viblo, 2017.*  [Trực tuyến]. Có tại:[*https://viblo.asia/p/tim-hieu-ve-ky-thuat-phan-tich-gia-tri-bien-va-phan-vung-tuong-duong-trong-kiem-thu-hop-den-bWrZnBJrZxw*](https://viblo.asia/p/tim-hieu-ve-ky-thuat-phan-tich-gia-tri-bien-va-phan-vung-tuong-duong-trong-kiem-thu-hop-den-bWrZnBJrZxw). [Truy cập: 8-6-2025].
7. Testing VN, *“State Transition Testing”, testingvn, 2010.*  [Trực tuyến]. Có tại:[*https://www.testingvn.com/viewtopic.php?t=119*](https://www.testingvn.com/viewtopic.php?t=119). [Truy cập: 8-6-2025].
8. STC Admin, *“Software Testing Life Cycle STLC,” Software Testing Class*, [Trực tuyến]. Có tại: <https://www.softwaretestingclass.com/software-testing-life-cycle-stlc/>. [Truy cập: 8-6-2025].
9. HR1 Tech, “*Quy trình kiểm thử phần mềm – Software Testing Life Cycle (STLC),” hr1tech.com, 2021.* [Trực tuyến]. Có tại: [*https://hr1tech.com/vi/news/software-testing-life-cycle-stlc.html*](https://hr1tech.com/vi/news/quy-trinh-kiem-thu-phan-mem-software-testing-life-cycle-stlc-121.html). [Truy cập: 8-6-2025].
10. Thạc Bình Cường, *Kiểm thử và đảm bảo chất lượng phần mềm*, Hà Nội: Đại học Bách khoa Hà Nội, 2008.
11. Tenten, *“Selenium là gì? Toàn tập về 4 công cụ Selenium Automation Testing bạn cần biết,” tenten.vn, 2023.* [Trực tuyến]. Có tại:[*https://tenten.vn/tin-tuc/selenium-la-gi/*](https://tenten.vn/tin-tuc/selenium-la-gi/). [Truy cập: 8-6-2025].
12. Got It Vietnam (n.d.), *“Selenium WebDriver là gì? Top những ưu nhược điểm mà bạn cần biết,” Got It AI,* [Trực tuyến]. Có tại:<https://vn.got-it.ai/blog/selenium-webdriver-la-gi-top-nhung-uu-nhuoc-diem-ma-ban-can-biet>. [Truy cập: 8-6-2025].
13. Nguyễn Thị Hoa, *“Sử dụng TestNG Framework trong tạo kịch bản Selenium,” Viblo, 2020.* [Trực tuyến]. Có tại:<https://viblo.asia/p/su-dung-testng-framework-trong-tao-kich-ban-selenium-V3m5W03EKO7>. [Truy cập: 8-6-2025].