**BỘ** **CÔNG** **THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**



**NGUYỄN TRANG NHUNG**

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

NGÀNH HỆ THỐNG THÔNG TIN

|  |
| --- |
| **NGHIÊN CỨU VÀ ỨNG DỤNG SELENIUM WEBDRIVER TRONG KIỂM THỬ WEBSITE ĐẶT ĐỒ ĂN TRỰC TUYẾN** |

**HỆ THỐNG THÔNG TIN**

|  |  |
| --- | --- |
| **CBHD** | **: ThS.Trần Việt Hà** |
| **Sinh viên** | **: Nguyễn Trang Nhung** |
| **Mã sinh viên** | **: 2021606577** |

**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 |
|  |

|  |
| --- |
| **NGHIÊN CỨU VÀ ỨNG DỤNG SELENIUM WEBDRIVER TRONG KIỂM THỬ WEBSITE ĐẶT ĐỒ ĂN TRỰC TUYẾN** |

|  |  |
| --- | --- |
| **CBHD** | **: ThS.Trần Việt Hà** |
| **Sinh viên** | **: Nguyễn Trang Nhung** |
| **Mã sinh viên** | **: 2021606577** |

|  |
| --- |
| **Hà Nội – Năm 2025** |

LỜI CẢM ƠN

Trong thời gian thực hiện đồ án tốt nghiệp ngành hệ thống thông tin, em đã nhận được nhiều sự hỗ trợ, giúp đỡ và khuyến khích từ rất nhiều người. Đó là một phần quan trọng 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 DevPro Việt Nam 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 HÌNH VẼ

# DANH MỤC BẢNG BIỂU

1. MỞ ĐẦU
2. **Tên đề tài:**

Nghiên cứu và sử dụng Selenium Webdriver trong kiểm thử website đặt đồ ăn trực tuyến

1. **Lý do chọn đề tài**

Trong bối cảnh công nghệ phát triển mạnh mẽ, các website hay ứng dụng ngày càng trở nên phổ biến, đặc biệt là lĩnh vực thương mại điện tử. Sự phát triển nhanh chóng đó đặt ra yêu cầu cao về chất lượng phần mềm, đòi hỏi các hệ thống phải vận hành ổn định, chính xác và đáp ứng tốt nhu cầu người dùng.

Trong lĩnh vực phát triển phần mềm, kiểm thử đóng vai trò quan trọng không thể thiết trong việc đảm bảo chất lượng. Kiểm thử thủ công giúp đánh giá tính trực quan, trải nghiệm người dùng và xử lý những tình huống khóa tự động hóa nhưng sẽ 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 các bài kiểm thử lặp đi lặp lại và đảm bảo tính ổn định của hệ thống trên nhiều môi trường khác nhau.

Hiện nay, có rất nhiều công cụ phổ biến để kiểm thử tự động, tùy thuộc vào loại ứng dụng và mục đích kiểm thử. Trong đó, theo Stack Overflow Developer Survey 2023, Selenium là một trong những công cụ kiểm thử hàng đầu với hơn 50% lập trình viên, kiểm thử viên sử dụng trong lĩnh vực kiểm thử tự động.

Với mục tiêu tìm hiểu cách áp dụng kiểm thử tự động trong thực tế, em đã quyết định thực hiện đề tài “Nghiên cứu và sử dụng Selenium Webdriver trong kiểm thử website đặt đồ ăn trực tuyến”. 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 đặt đồ ăn trực tuyến Haui Food.

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 wbsite Haui Food với công cụ Selenium Webdriver. Các nội dung chính bao gồm:

* Lý thuyết kiểm thử phần mềm: Tìm hiểu quy trình kiểm thử phần mềm, các loại kiểm thử như kiểm thử chức năng, kiểm thử giao diện người dùng cũng như các kỹ thuật kiểm thử như kiểm thử hộp đen, kiểm thử hộp trắng.
* Phương pháp kiểm thử phần mềm tự động
* Công cụ hỗ trợ kiểm thử: Selenium Webdriver kết hợp với thư viện TestNG, Allure Report, Maven
* Ứng dụng: Vận dụng các kiến thức kiểm thử vào việc xây dựng và thực hiện kiểm thử cho website đặt đồ ăn trực tuyến Haui Food

1. **Giới hạn và phạm vi của đề tài**

Đề tài được thực hiện trong thời gian từ 17/03/2025 đến 17/05/2025.

Phạm vi tập trung vào kiểm thử website đặt đồ ăn trực tuyến do nhóm sinh viên thực hiện với các chức năng cơ bản với công cụ Selenium WebDriver trên nền tảng web.

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 đặt đồ ăn trực tuyến với Selenium WebDriver.
* Phương pháp phân tích số liệu: Sử dụng các phương pháp thống kể 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:**

**Chương 2:**

**Chương 3:**

# TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM

## Khái niệm và mục tiêu của 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.

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 mình rằng chúng phù hợp với mục đích sử dụng và phát hiện lỗi.

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

* Kiểm thử thủ công: Là quá trình kiểm thử phần mềm mà kiểm thử viên thực hiện kiểm thử mọi thứ bằng tay, từ viết ca kiểm thử đến thực hiện test
* Kiểm thử tự động: Thực hiện một cách tự động các bước trong kịch bản kiểm thử bằng cách dùng một công cụ trợ giúp. Kiểm thử tự động nhằm tiết kiệm thời gian kiểm thử.

Mục đích chính của kiểm thử phần mềm bao gồm:

* Tìm ra được càng nhiều lỗi càng tốt trong điều kiện về thời gian đã định và nguồn lực sẵn có
* Chứng minh rằng sản phẩm phần mềm phù hợp với các đặc tả của nó
* Xác thực chất lượng kiểm thử phần mềm đã dùng chi phí và nỗ lực tối thiểu
* Thiết kế tài liệu kiểm thử một cách có hệ thống và thực hiện nó sao cho có hiệu quả, tiết kiệm được thời gian công sức

## Các nguyên lý cơ bản của kiểm thử

Trong quá trình phát triển phần mềm, kiểm thử đóng vai trò quan trọng nhằm đảm bảo chất lượng sản phẩm trước khi đưa vào sử dụng. Tuy nhiên, kiểm thử không thể phát hiện mọi lỗi và cũng không thể thực hiện một cách ngẫu nhiên mà cần có phương pháp tiếp cận khoa học. Để tối ưu hóa quá trình kiểm thử, ngành kiểm thử phần mềm đã xây dựng và tuân theo một số nguyên lý cơ bản giúp định hướng chiến lược kiểm thử, nâng cao hiệu quả và tối ưu nguồn lực. Dưới đây là 7 nguyên lý cơ bản của kiểm thử:

Thứ nhất, kiểm thử toàn bộ là không thể. Thay vì kiểm thử toàn bộ, chúng ta có thể dựa vào đánh giá rủi ro, độ ưu tiên và sử dụng kỹ thuật thiết kế kịch bản kiểm thử phù hợp để kiểm thử.

Thứ hai, sự phân cụm của lỗi (defect clustering). Nguyên lý này cho rằng một số ít các mô-đun chứa hầu hết các lỗi được phát hiện. Đây là việc áp dụng nguyên lý Pareto (80-20 Rule) để kiểm thử phần mềm: khoảng 80% các vẫn đề được tìm thấy trong 20% các mô-đun.

Thứ ba là nguyên lý thuốc trừ sâu. Theo nguyên lý này, một bộ dữ liệu kiểm thử được sử dụng lặp đi lặp lại sẽ không tìm được ra các lỗi mới. Để khắc phục điều này, các trường hợp kiểm thử cần phải được xem xét và sửa đổi thường xuyên, bổ sung thêm các trường hợp kiểm thử mới và khác để giúp tìm ra nhiều lỗi hơn. Người kiểm thử không chỉ đơn giản là phụ thuộc vào các kỹ thuật kiểm thử hiện có mà cần phải nhìn ra liên tục để cải thiện các phương pháp hiện có để làm cho kiểm thử hiệu quả hơn. Khi một thay đổi được thực hiện, việc kiểm thử hồi quy là rất cần thiết.

Thứ tư, kiểm thử chỉ ra sự hiện diện của các lỗi. Kiểm thử chỉ ra sự tồn tại của các lỗi nhưng không thể chứng minh rằng phần mềm không có lỗi. Kiểm thử làm giảm xác suất của các lỗi chưa được phát hiện trong phần mềm.

Thứ năm, sự sai lầm về việc phần mềm không có lỗi. Có thể phần mềm 99% không có lỗi vẫn không sử dụng được nếu hệ thống được xây dựng trên yêu cầu sai. Kiểm thử không chỉ là tìm ra các lỗi, mà còn để kiểm tra xem phần mềm có đáp ứng yêu cầu nghiệp vụ không.

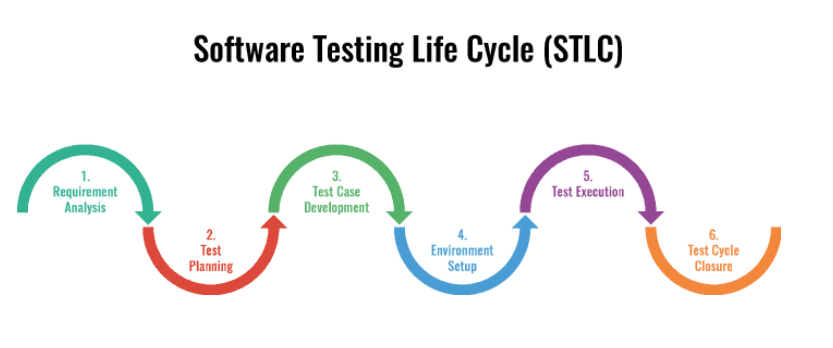
Thứ sáu, nên thực hiện kiểm thử càng sớm càng tốt. Kiểm thử nên bắt đầu càng sớm càng tốt trong chu kỳ vòng đời phát triển phần mềm. Lỗi được phát hiện sớm sẽ có chi phí sửa lỗi thấp hơn. Nhưng nên kiểm thử sớm như thế nào còn phụ thuộc vào việc sản phẩm phần mềm đó đang được phát triển theo quy trình như thế nào.

Thứ bảy, kiểm thử phụ thuộc vào ngữ cảnh. Cách tiếp cận, phương pháp, kỹ thuật và các loại kiểm thử tùy thuộc vào ứng dụng bơi tất cả các phần mềm đã phát triển đều không giống nhau. Ví dụ: Kiểm thử một trang web thương mại điện tử sẽ khác so với kiểm thử một trang web giới thiệu về một công ty.

## 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:

* Phân tích yêu cầu (Requirement Analysis).
* Lập kế hoạch kiểm thử (Test Planning)
* Thiết kế kịch bản kiểm thử (Test Case Development)
* Thiết lập môi trường kiểm thử (Environment Setup)
* Thực hiện kiểm thử (Test Execution)
* Kết thúc chu kỳ kiểm thử **(**Test Cycle Closure)



Hình 1.1: Quy trình kiểm thử phần mềm

### 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 theo các bước quan trọng 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 thiêt 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 sau:

* + - * 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 gồm các hoạt động sau:

* + - * 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.
      * Thực hiện 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.

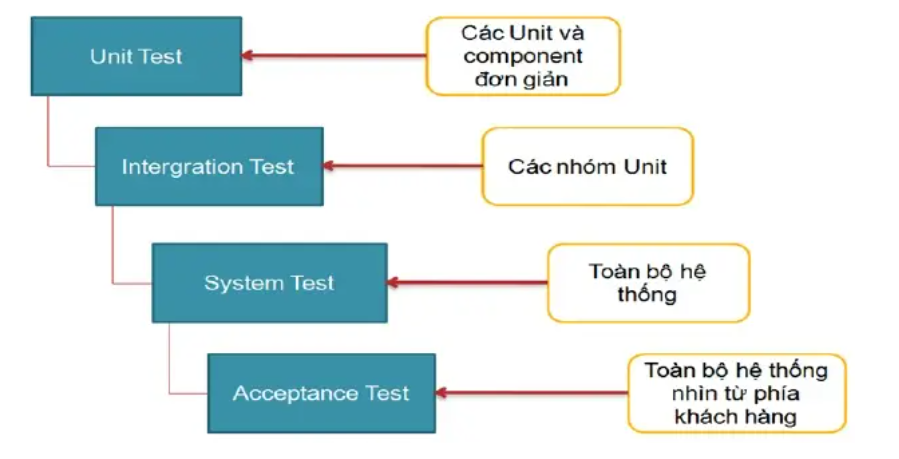
### Đóng chu trình kiểm thử

Đây là giai đoạn cuối cùng trong quy trình kiểm thử phần mềm, gồm các hoạt động sau:

* + - * 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 tất cả những điểm tốt, điểm chưa tốt và rút ra bài học kinh nghiệm.

## Các mức độ và loại hình kiểm thử

* Các mức độ kiểm thử:



Hình 1.2: Các mức độ kiểm thử

* Kiểm thử đơn vị (Unit testing): Nhằm kiểm tra đơn vị thiết kế nhỏ nhất – một module phần mềm. Người tiến hành kiểm thử đơn vị thường là lập trình viên. Kỹ thuật kiểm thử đơn vị chủ yếu là kiểm thử hộp trắng, trong các trường hợp cần thiết có thể sử dụng thêm kỹ thuật kiểm thử hộp đen.
* Kiểm thử tích hợp (Intergation testing): 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 nhóm phát triển phần mềm.
* Kiểm thử hệ thống (System test): Nhằm tìm kiếm các lỗi, nhưng trọng tâm là đánh giá về hoạt động, thao tác, sự tin cậy và các yêu cầu khác liên quan đến chất lượng của toàn hệ thống. Người thực hiện thường là kiểm thử viên. Kỹ thuật kiểm thử hệ thống chủ yếu là kiểm thử hộp đen.
* Kiểm thử chấp nhận (Acceptance test): Là cấp độ kiểm thử cuối cùng trong các cấp độ kiểm thử, được thực hiện để xác nhận rằng hệ thống đáp ứng đầy đủ các yêu cầu của khách hàng hoặc người dùng cuối trước khi đưa vào sử dụng chính thức.
* Các loại hình kiểm thử:
* Kiểm thử chức năng (Functional test): Quy trình cố gắng tìm ra các khác biệt giữa đặc tả bên ngoài của phần mềm và thực tế mà phần mềm cung cấp.
* Kiểm thử phi chức năng (Non-funtional test): Tập trung vào kiểm thử phần mề, hệ thống phần mềm có những đặc tính tốt như thế nào, tức là kiểm thử tập trung vào mặt chất lượng. Kiểm thử phi chức năng có thể được sử dụng ở mọi cấp độ kiểm thử nhưng thường được sử dụng hiệu quả nhất trong cấp độ kiểm thử hệ thống và kiểm thử chấp nhận.
* Kiểm thử cấu trúc (Structural test): Nhằm đánh giá sự bao phủ của code, tìm ra các lỗi tiềm ẩn trong cấu trúc logic của chương trình, xác minh xem mã nguồn có được viết đúng theo yêu cầu hay không. Kỹ thuật được sử dụng thường là kiểm thử hộp trắng và kiểm thử hộp xám.
* Kiểm thử liên quan đến sự thay đổi (Change related test): Kiểm thử xem sản phẩm có vận hành trơn tru qua các lần thay đổi hay không. Hai kỹ thuật sử dụng chủ yếu là kiểm thử lại và kiểm thử hồi quy.

## Các kỹ thuật kiểm thử

Trong kiểm thử phần mềm có 3 kỹ thuật chính bao gồm: Kiểm thử hộp đen, kiểm thử hộp trắng và kiểm thử hộp xám. Trong đề tài này, kỹ thuật được sử dụng là kiểm thử hộp đen.

### Kỹ thuật 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 phương pháp kiểm thử hộp đen được sử dụng bao gồm:

* ***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:

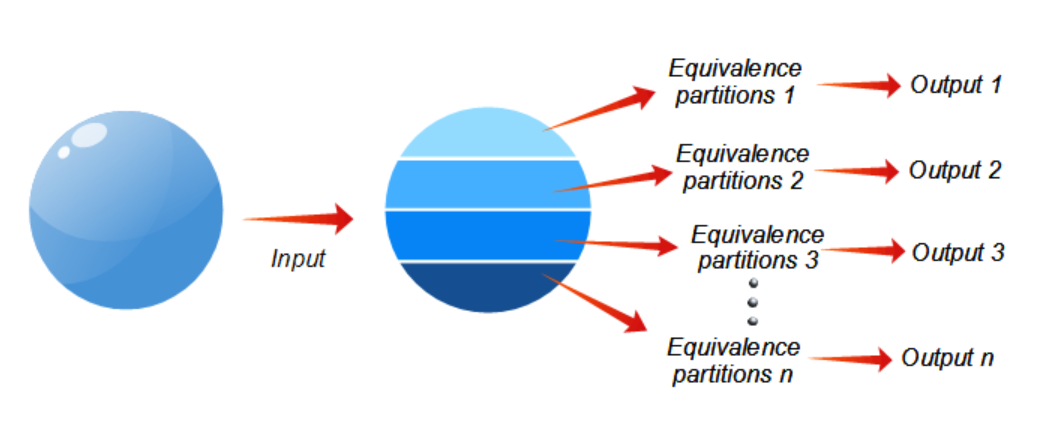
Bước 1: Xác định các lớp tương đương theo đầu vào hoặc đầu ra. Với mỗi điều kiện đầu vào hoặc đầu ra được mô tả trong đặc tả yêu cầu thì lấy ra: lớp thỏa mãn điều kiện và lớp không thỏa mãn điều kiện.

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
* Nhập vào không phải số nguyên



Hình 1.3: Phân vùng tương đương

* ***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.

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]:

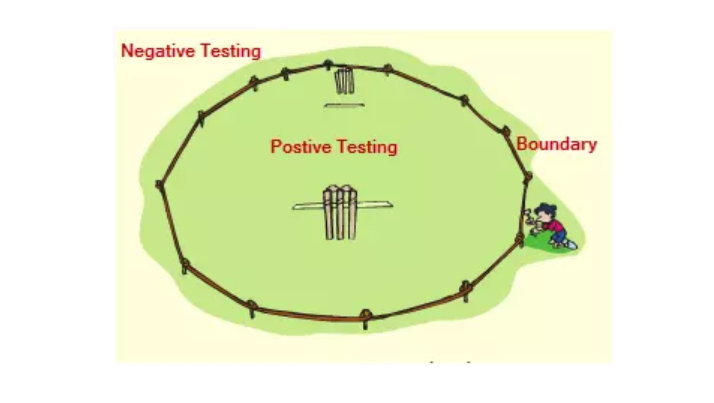
+ Giá trị nhỏ nhất: 0

+ 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

* ***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.

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 dịnh

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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%** |

* ***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 Button

Textbox có thể nhận những giá trị nguyên dương từ 1 đến 100

Và OK button

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

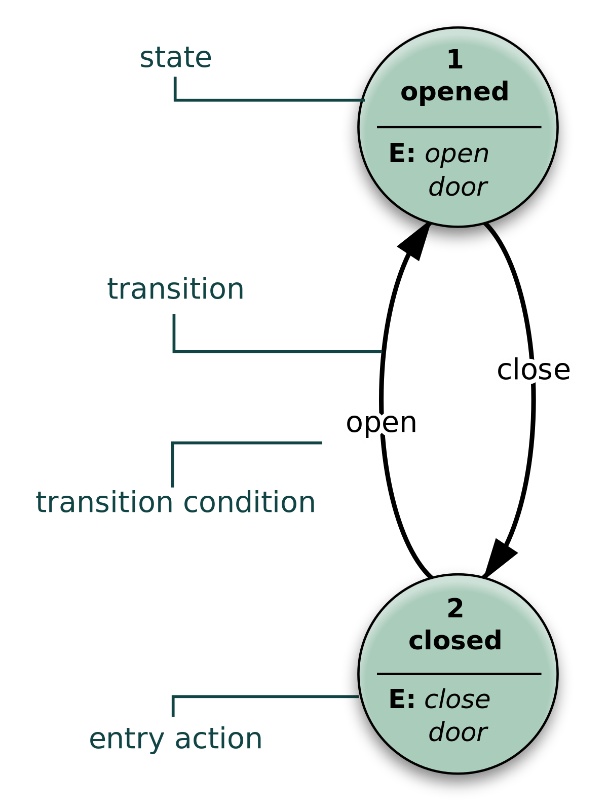
Bảng 1.2: Ví dụ về Pairwise testing

|  |  |  |  |
| --- | --- | --- | --- |
| **Texbox** | **List box** | **Checkbox** | **Radio Button** |
| 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

### Kỹ thuật 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.

### Kỹ thuật kiểm thử hộp xám

Kiểm thử hộp xám là phương pháp kiểm thử khá mới mẻ, mới hình thành và đòi hỏi trình độ cao. Kiểm thử hộp xám là sự kết hợp của 2 loại kiểm thử hộp đen và hộp trắng. Với kỹ thuật này, người kiểm thử cần phải có một phần sự hiểu biết về cấu trúc của hệ thống cũng như là quyền truy cập vào cơ sở dữ liệu. Kiểm thử hộp xám thường được sử dụng cho mức độ kiểm thử tích hợp. Trong quá trình tiến hành kiểm thử hộp xám, người kiểm thử sẽ tạo đầu vào từ phía front-end, sau đó xác minh dữ liệu từ phía back-end.

Ưu điểm: Tiếp cận từ góc độ người dùng, giúp đảm bảo tính toàn diện của kiểm thử. Dựa trên các đặc tả chức năng, mô tả người dùng và sơ đồ kiến trúc, giúp xác nhận yêu cầu sớm. Kết hợp lợi ích của cả kiểm thử hộp đen và hộp trắng.

Nhược điểm: Có thể tốn nhiều thời gian để kiểm tra từng luồng, đôi khi không thực tế. Phạm vi kiểm tra thường thấp hơn so với kiểm thử hộp đen và hộp trắng riêng biệt.

## 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.

# CÔNG CỤ KIỂM THỬ SELENIUM WEBDRIVER

## Giới thiệu về công cụ Selenium

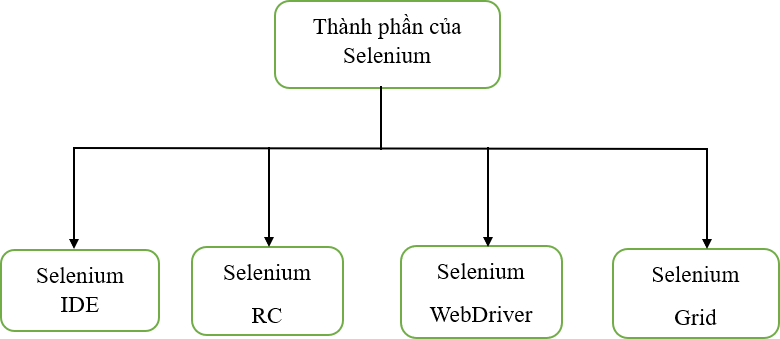
### Giới thiệu chung về Selenium

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 cấ các nhu cầu thử nghiệm khác nhau của một tổ chức. Selenium gồm có 4 thành phần sau:

Hình 2.1: Thành phần của Selenium

* Selenium Intergrated 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. 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 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.
* 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.
* 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.

## 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 một 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). 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 WebDrver 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 miền 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.

**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.

## 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 qua, chúng ta có thể kết hợp Selenium WebDriver với các công cụ và thư viện sau:

* **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, @BeforeSuite, @AfterSuite để đánh dấu các phương thức là các bước kiểm thử, trước hoặc sau toàn bộ bộ kiểm thử hoặc nhóm kiểm thử.
* Quản lý thời gian: Bạn 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.
* **IntelliJ IDEA:**
* IntelliJ IDEA là một môi trường phát triển tích hợp mạnh mẽ cho Java.
* Cung cấp các công cụ hỗ trợ viết mã, kiểm tra lỗi cú pháp và tích hợp dễ dàng với TestNG, Maven.
* Hỗ trợ chạy kiểm thử trực tiếp trong IDE mà không cần thiết lập môi trường phức tạp.
* Tích hợp tốt với Git, Github, giúp quản lý mã nguồn hiệu quả
* Plugin: Hỗ trợ nhiều plugin hữu ích cho kiểm thử tự động hóa, ví dụ như plugin TestNG, Maven và Selenium.
* **Maven (Công cụ quản lý dự án):**
* Maven là một công cụ quản lý dự án và xây dựng phần mềm, giúp tự động hóa quá trình xây dựng, kiểm thử và triển khai dự án. Nó sử dụng một file xml (pom.xml) để định nghĩa cấu hình dự án, bao gồm các phụ thuộc, plugin và cấu hình xây dựng.
* Maven hỗ trợ tự động hóa quá trình biên dịch, đóng gói và triển khải dự án.
* Hỗ trợ chạy các trường hợp kiểm thử tự động hóa và quản lý vòng đời dự án: Định nghĩa các giai đoạn vòng đời dự án, ví dụ như biên dịch, kiểm thử và triển khai.
* Ứng dụng thực tế: Quản lý các thư viện Selenium WebDriver và TestNG, tự động hóa quá trình xây dựng và chạy các trường hợp kiểm thử Selenium, tích hợp với các công cụ CI/CD để tự động hóa quá trình kiểm thử.
* **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 các thao tác trên trình duyệt Chrome, ví dụ như mở trang web, nhập dữ liệu, nhấp chuột và kiểm tra các phần tử trên trang.
* ChromeDriver tương thích tốt với Selenium WebDriver, cho phép viết các trường hợp kiểm thử tự động hóa dễ dàng.
* Được cập nhật thường xuyên để hỗ trợ các phiên bản mới nhất của Chrome.
* **Allure Report:**
* Allure Report là một framework tạo báo cáo kiểm thử linh hoạt và dễ hiểu, giúp trực quan hóa kết quả kiểm thử. Nó cung cấp các báo cáo chi tiết về kết quả kiểm thử bao gồm các bước kiểm thử, ảnh chụp màn hình và video.
* Cung cấp giao diện đồ họa trực quan và dễ hiểu, giúp theo dõi tiến độ kiểm thử và xác định các vấn đề một cách nhanh chóng.
* Cho phép đính kèm ảnh chụp màn hình và video vào báo cáo kiểm thử, giúp ghi lại các bước kiểm thử và lỗi.
* Allure Report hỗ trợ nhiều ngôn ngữ lập trình và framework kiểm thử, bao gồm Java, Python, JavaScript và thư viện TestNG.
* Tích hợp tốt với các công cụ CI/CD, cho phép tự động hóa quá trình tạo báo cáo kiểm thử.

Ngoài các công cụ và thư viện đã đề cập, Selenium WebDriver có thể kết hợp với nhiều công cụ và thư viện khác để tạo hiệu quả cao trong quá trình tự động hóa kiểm thử phần mềm như Selenium Grid, Cucumber, framework Junit, Docker, v.v.

## Các bước cài đặt Selenium WebDriver sử dụng ngôn ngữ Java

**Bước 1:** Cài đặt Chrome Brower

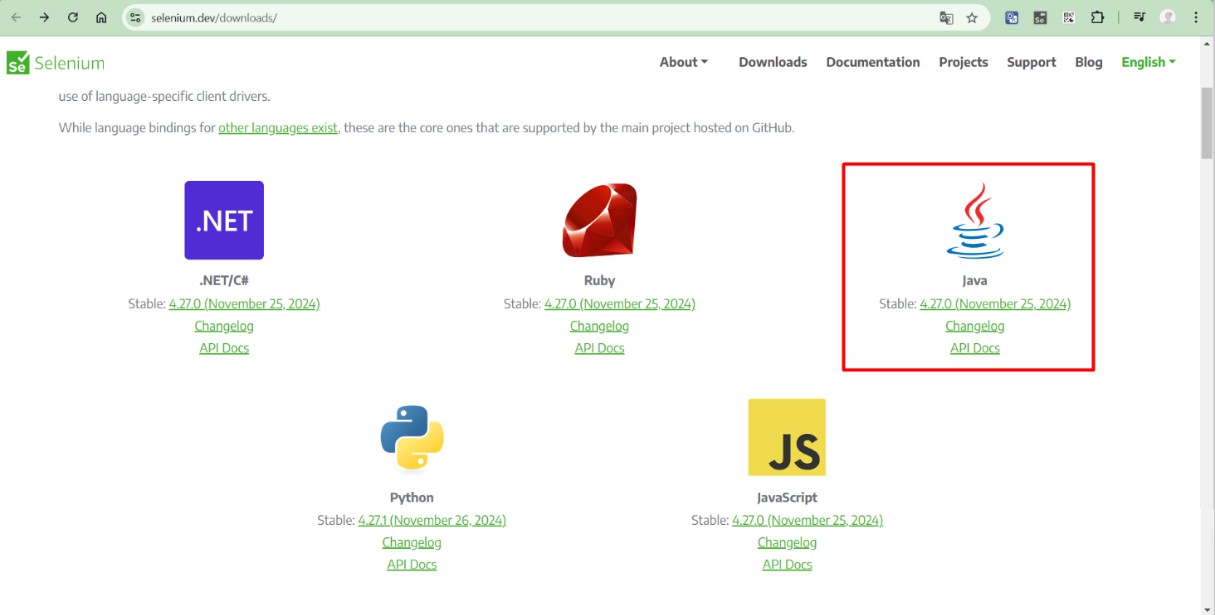
* + Vào trang web <https://www.google.com/intl/vi_vn/chrome/> và nhấn nút “Tải xuống tại đây”.
  + Thực hiện cài đặt Chrome.

**Bước 2:** Cài đặt Chrome Driver

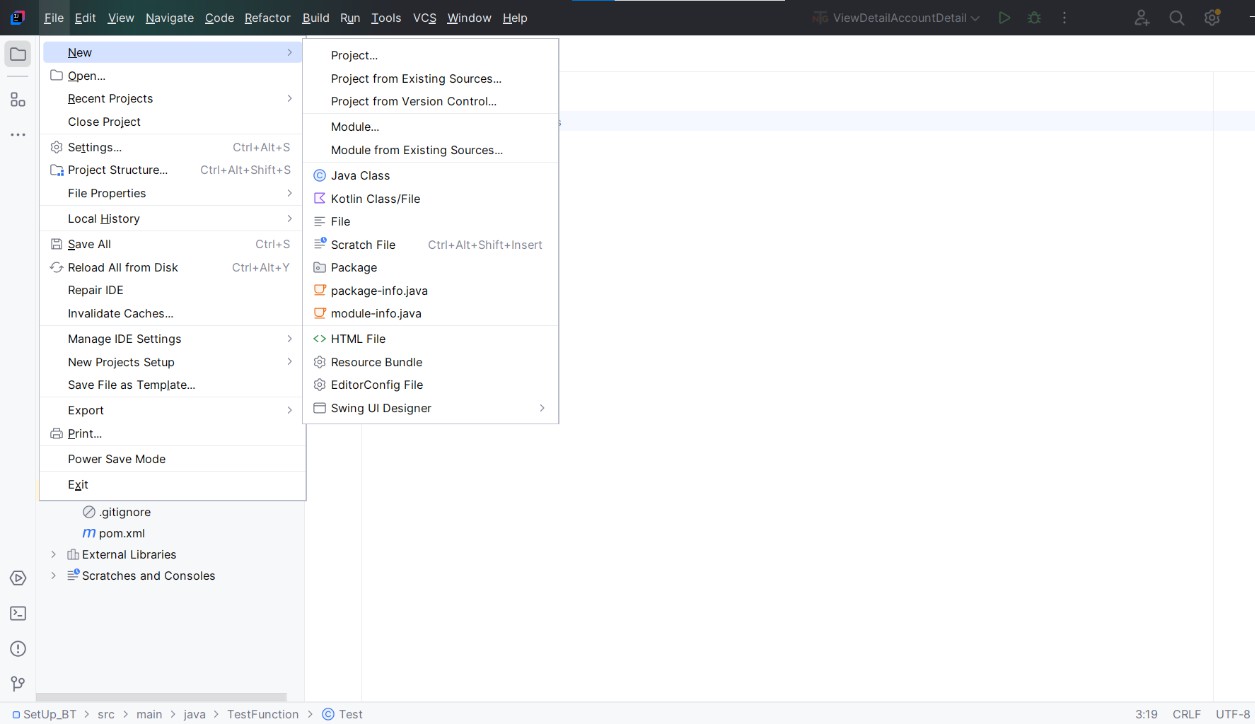
 Vào trang web <https://chromedriver.chromium.org/downloads> và tải phiên bản phù hợp với Chrome Browser đã tải ở bước 1.

**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.

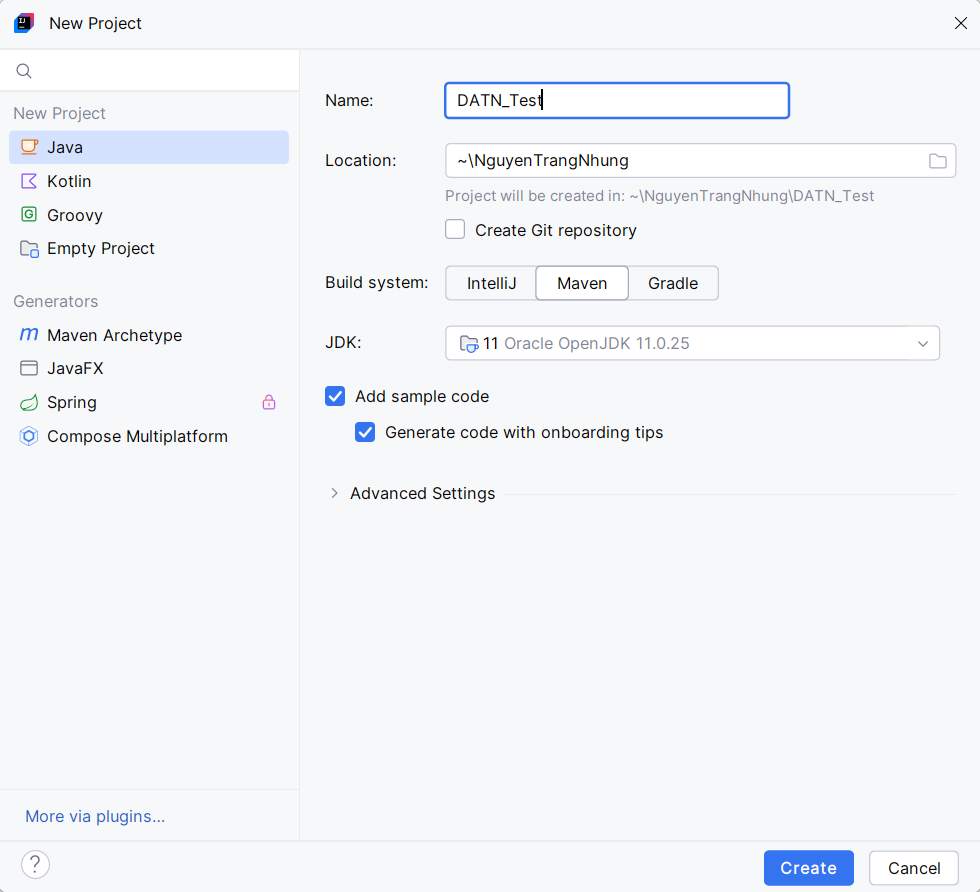


Hình 2.2: Màn hình trang tải về Selenium WebDriver

Sau đó, mở công cụ IntelliJ IDEA lên và tạo một project mới.

Hình 2.3: Thao tác tạo một project mới trong IntelliJ

Chọn cấu hình JDK đúng với phiên bản Java



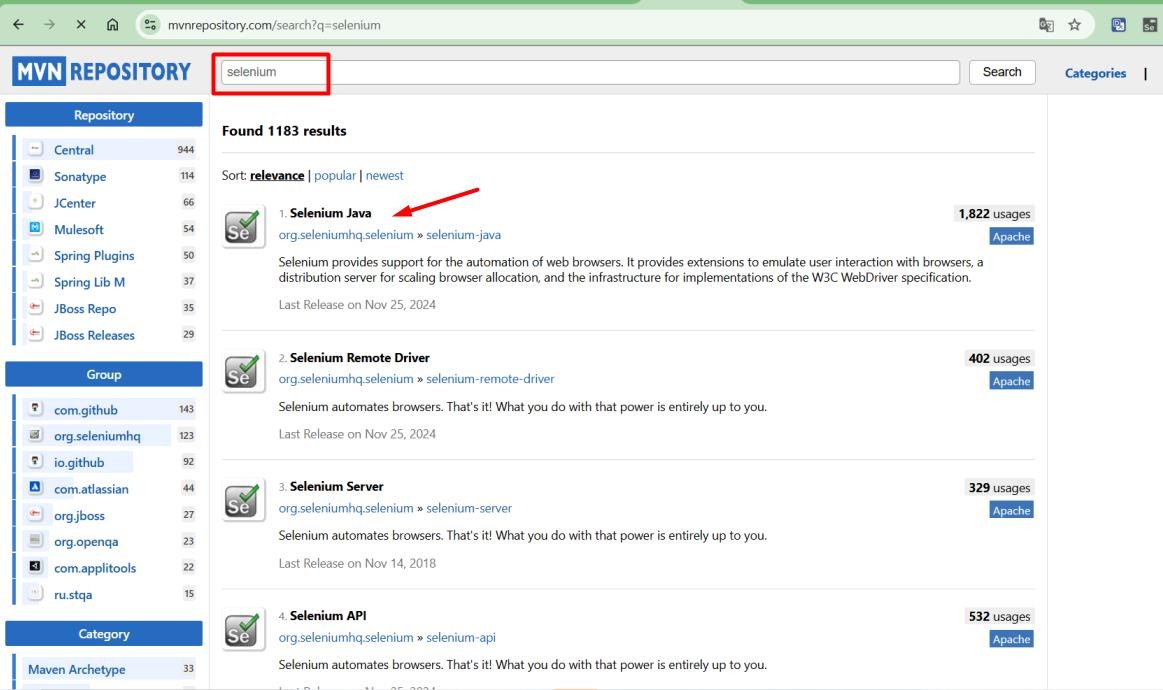
Hình 2.4: Chọn cấu hình JDK đúng với phiên bản Java ở IntelliJ

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.

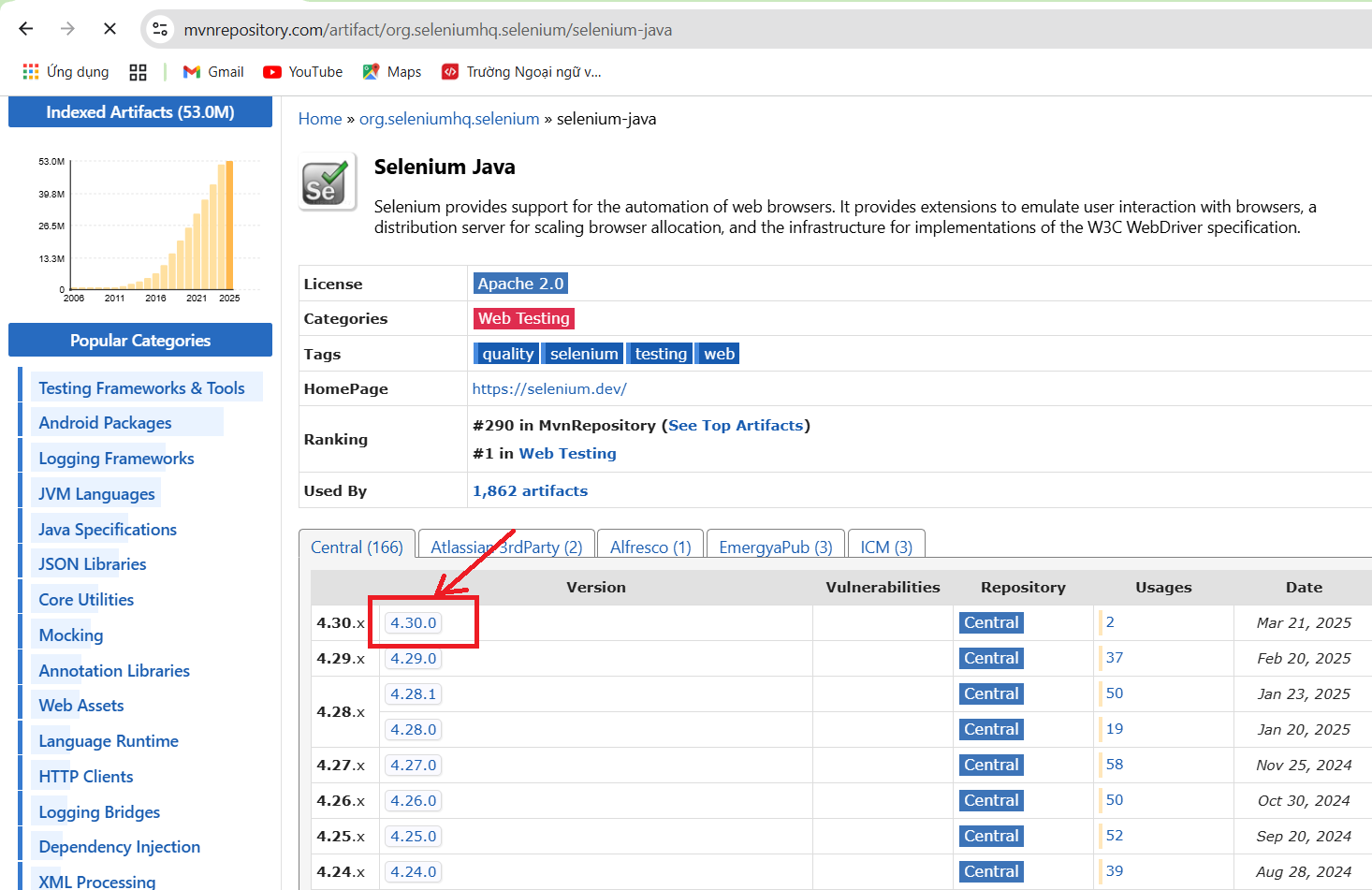
**Bước 4:** Cài đặt thư viện Selenium vào Maven project trên IntelliJ

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

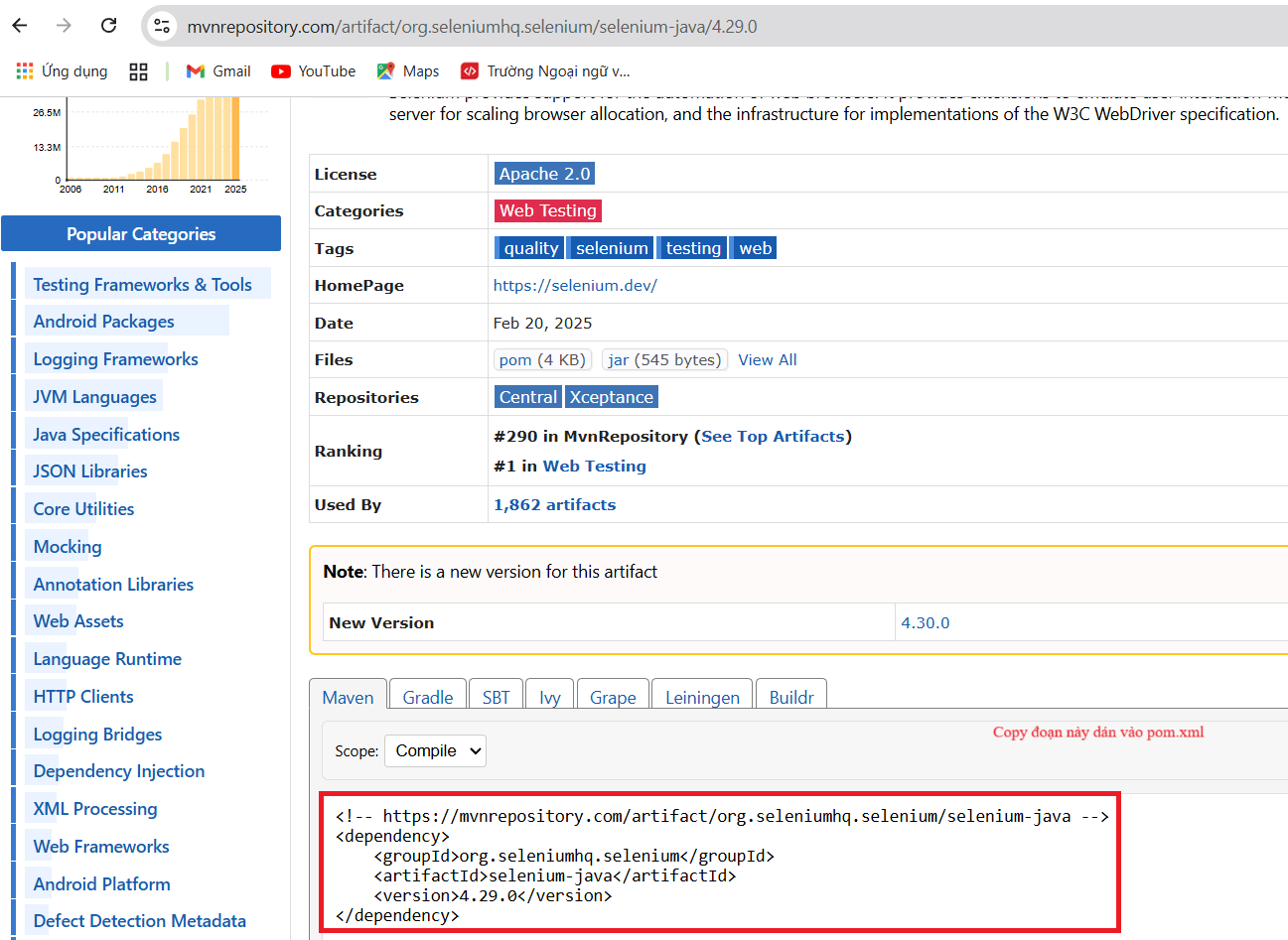
Thư viện được lấy tại trang web: <https://mvnrepository.com/>. Ta cần tìm Selenium Java, chọn phiên bản phù hợp. Sau đó, cần lấy ra dependency (phụ thuộc) dành cho Maven, được sử dụng để khai báo thư viện Selenium Java trong dự án Java sử dụng Maven.



Hình 2.5: Thao tác tải Selenium Java



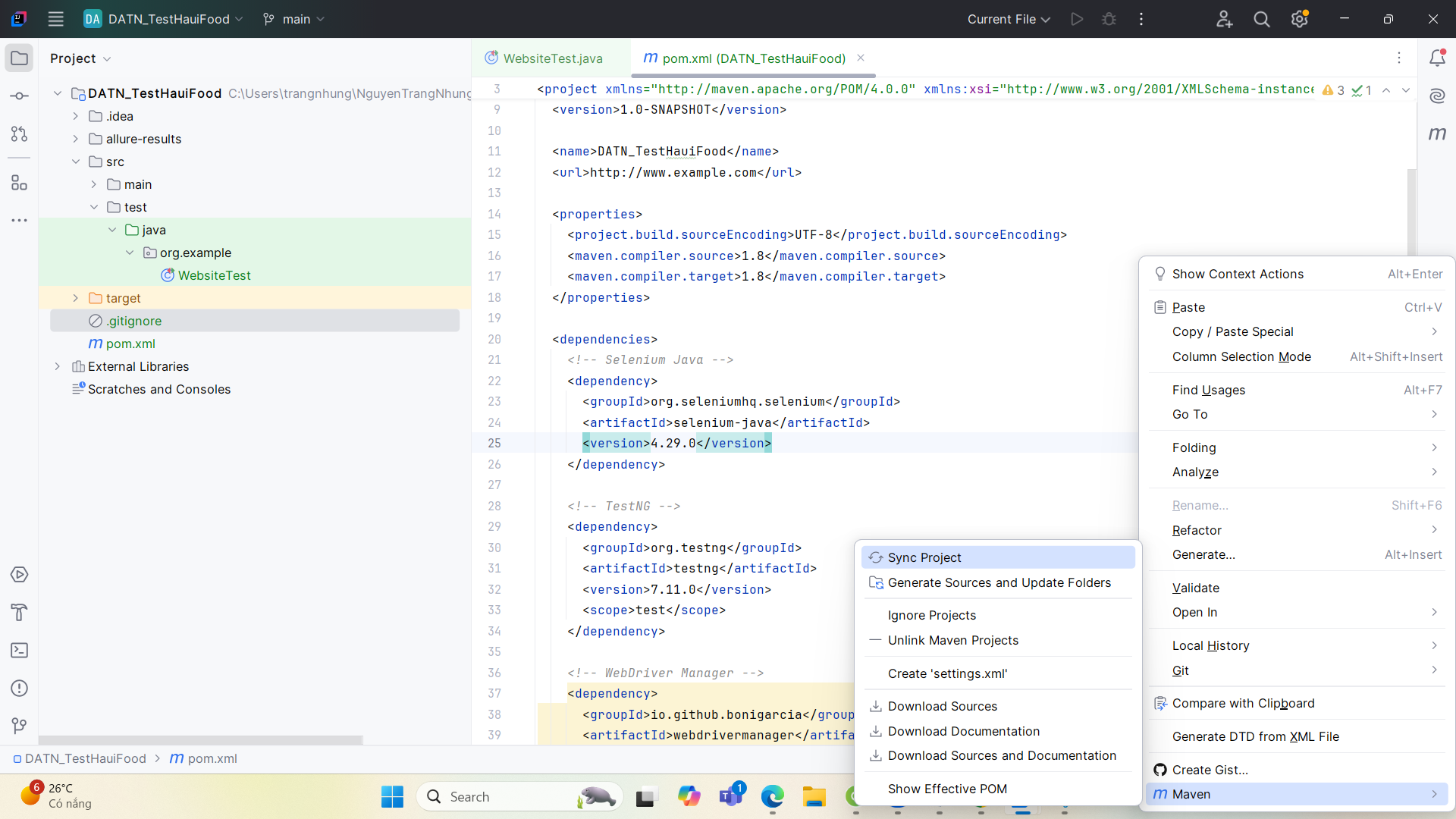
Hình 2.6: Chọn phiên bản cho Selenium Java



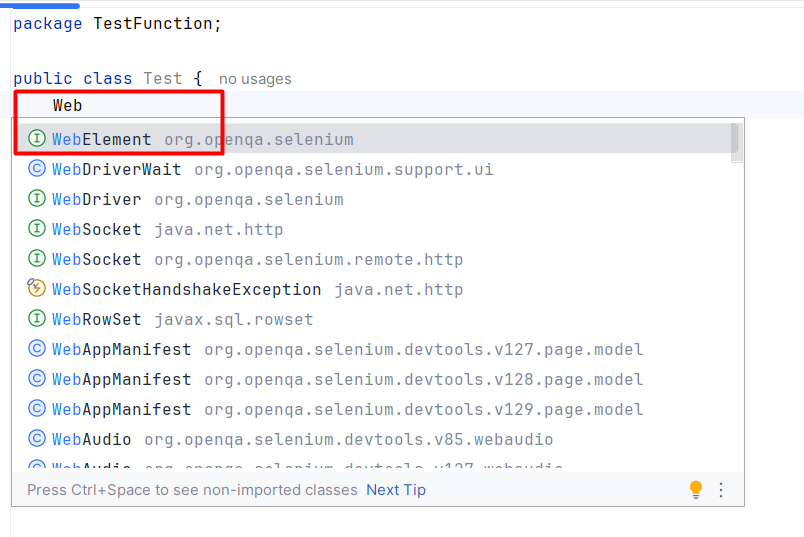
Hình 2.7: Dependency để khai khai báo thư viện Selenium Java

Sau khi lấy được dependency của phiên bản phù hợp như trên hình 2.7, ta cần thực hiện dán vào file pom.xml trên IntelliJ và phải dán đoạn mã trong cặp thẻ <dependencies> </dependencies>.

Khi đã thêm dependency vào pom.xml, cần khởi động lại Maven bằng cách nhấn vào biểu tượng Reload hoặc nhấp chuột phải > Maven > Reload project.



Hình 2.8: Thực hiện khởi động lại Maven sau khi thêm dependency

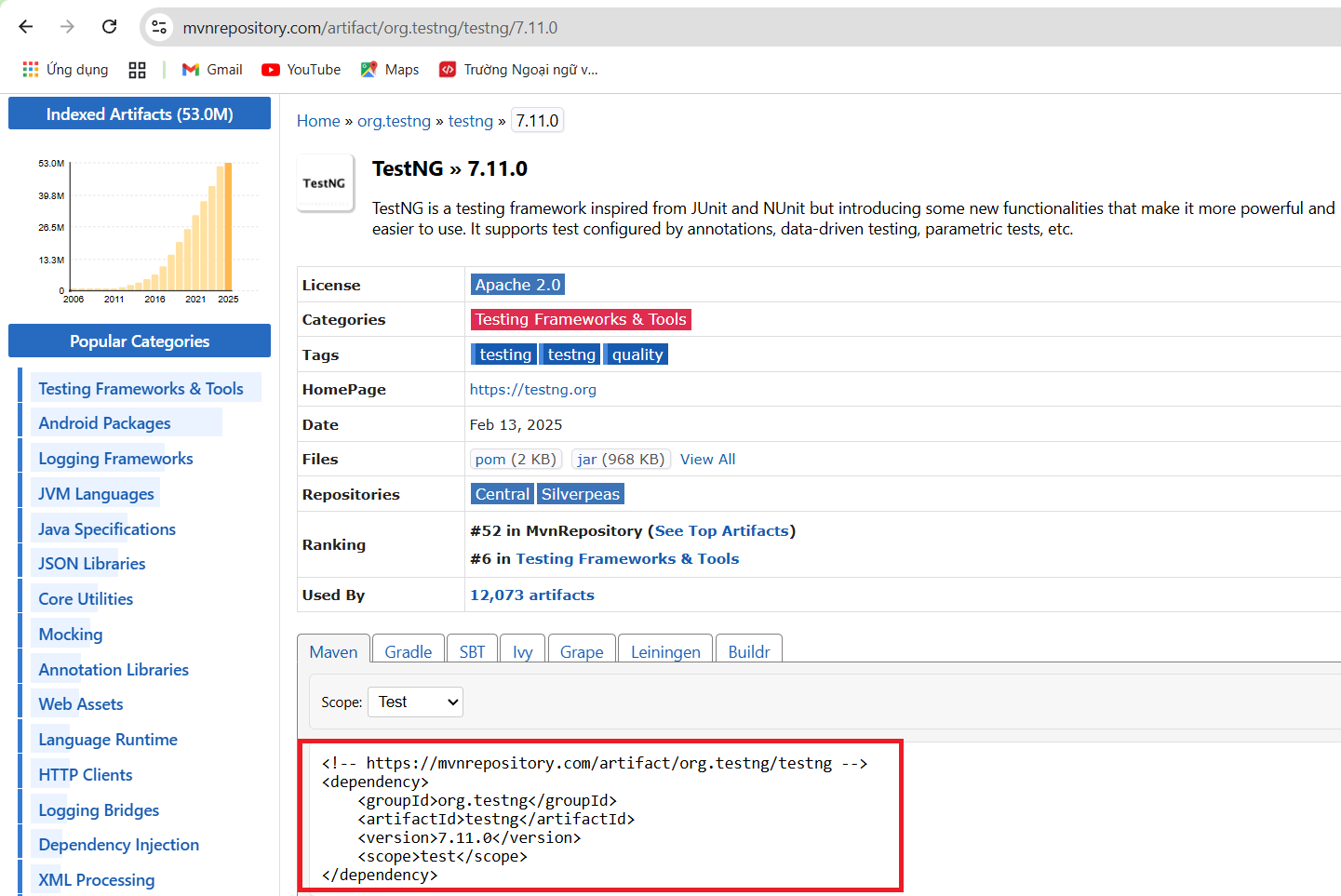
Kiểm tra Selenium đã cài đặt vào maven project: Nếu gợi ý hiển thị lên như hình 2.9 dưới đây là đã cài đặt được Selenium thành công.

Hình 2.9: Hoàn thiện cài đặt thư viện Selenium

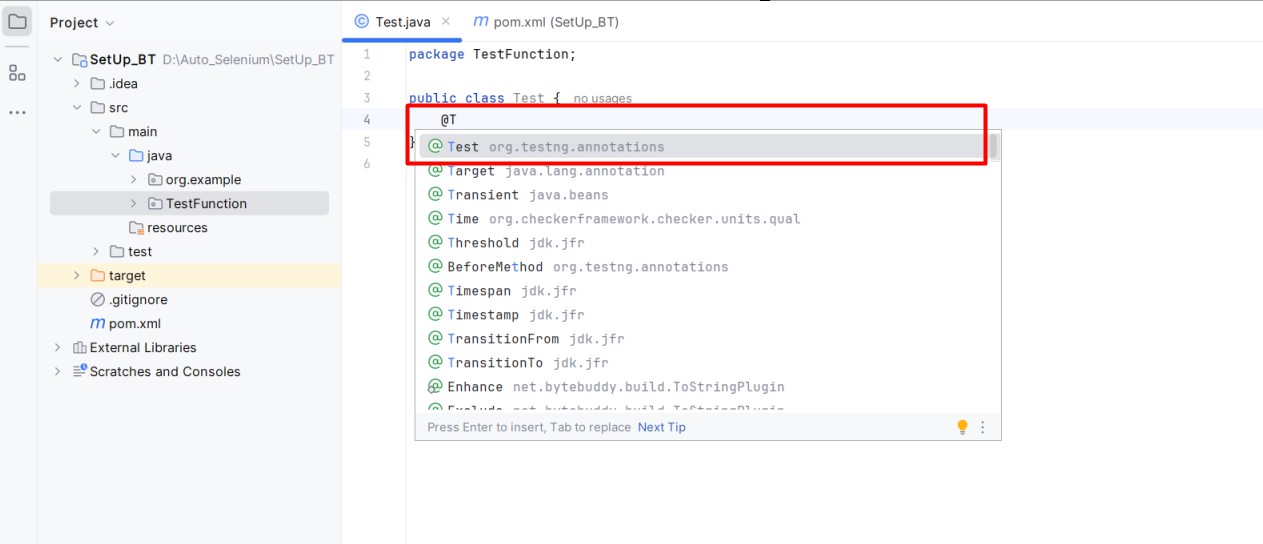
**Bước 4:** Cài đặt thư viện TestNG vào IntelliJ

Để thực hiện thêm thư viện TestNG vào file pom.xml cần tiến hành tương tự như Selnium Java:

Trên trang Maven Repository thực hiện tìm kiếm TestNG > Chọn phiên bản phù hợp > Lấy thông tin dependency (phụ thuộc) để thêm vào pom.xml của maven project đã tạo.



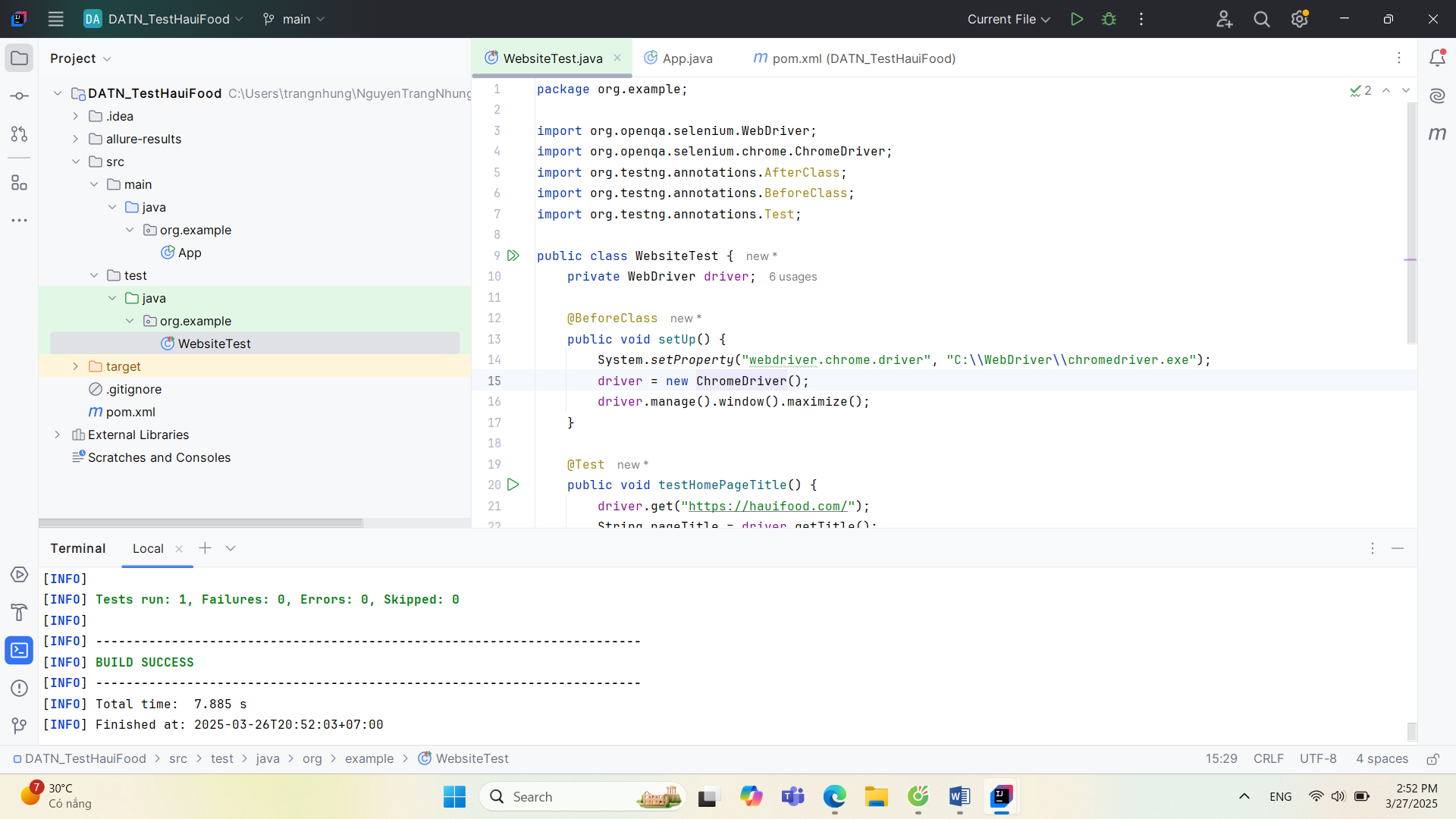
Hình 2.10: Dependency để khai báo thư viện TestNG

Sau đó, thêm dependency để khai báo thư viện TestNG vào file pom.xml và thực hiện reload Maven để hoàn tất cài đặt. Để kiểm tra xem thư viện TestNG đã được cài đặt thành công hay chưa, ta cần kiểm tra gợi ý hiển thị lên như hình dưới đây.

Hình 2.11: Hoàn thiện cài đặt TestNG

**Bước 5:** Viết test script đầu tiên

* Tạo class kiểm thử: Để viết một test script ta cần tạo class kiểm thử. Trước tiên, trong IntelliJ, tạo một package mới như sau: src/test/java/org/example. Sau đó tạo một file Java mới, ví dụ: WebsiteTest.java
* Viết script kiểm thử theo kịch bản kiểm thử
* Chạy test script: Có 2 cách để thực hiện:
* Nhấn chuột phải vào file cần chạy test script và chọn Run
* Chạy bằng terminal với lệnh sau: mvn test hoặc mvn clean test.

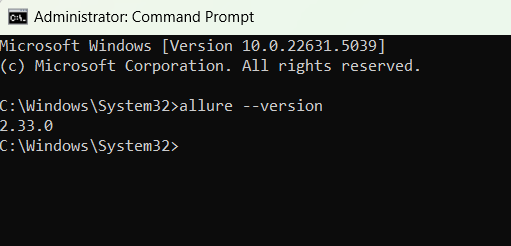


Hình 2.12: Kết quả của test script khi chạy bằng terminal

**Bước 6 (Nếu cần):** Tạo báo cáo Allure Report

1. Cài đặt Allure:

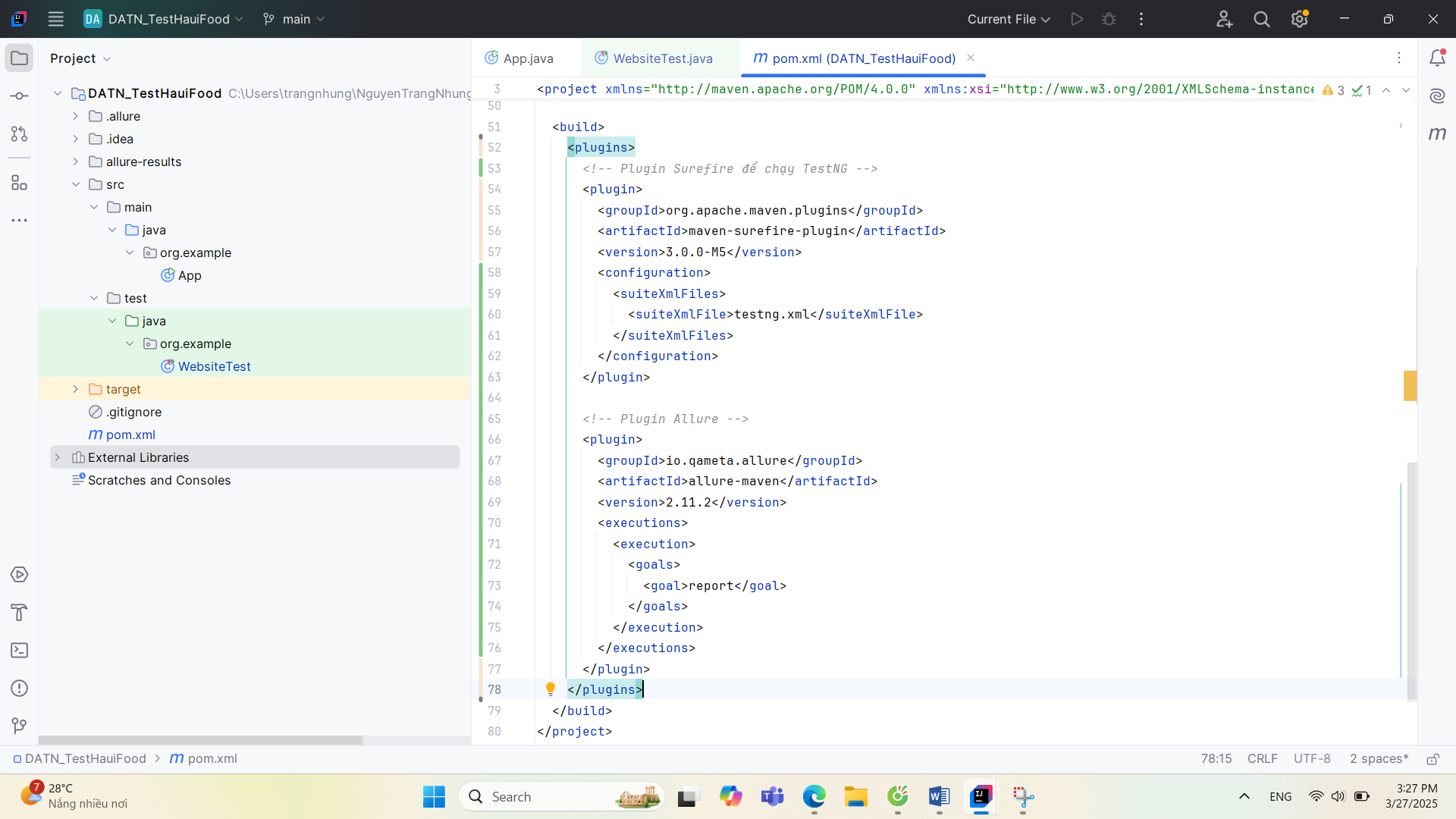
* Tải và cài đặt Allure từ <https://github.com/allure-framework/allure2>
* Kiểm tra Allure đã cài đặt thành công bằng lệnh: allure - -version



Hình 2.13: Kiểm tra phiên bản Allure

1. Cấu hình plugin allure trong Maven:

* Thêm vào file pom.xml phần plugin (lưu ý: cần thêm trong cặp thẻ <plugins></plugins>)

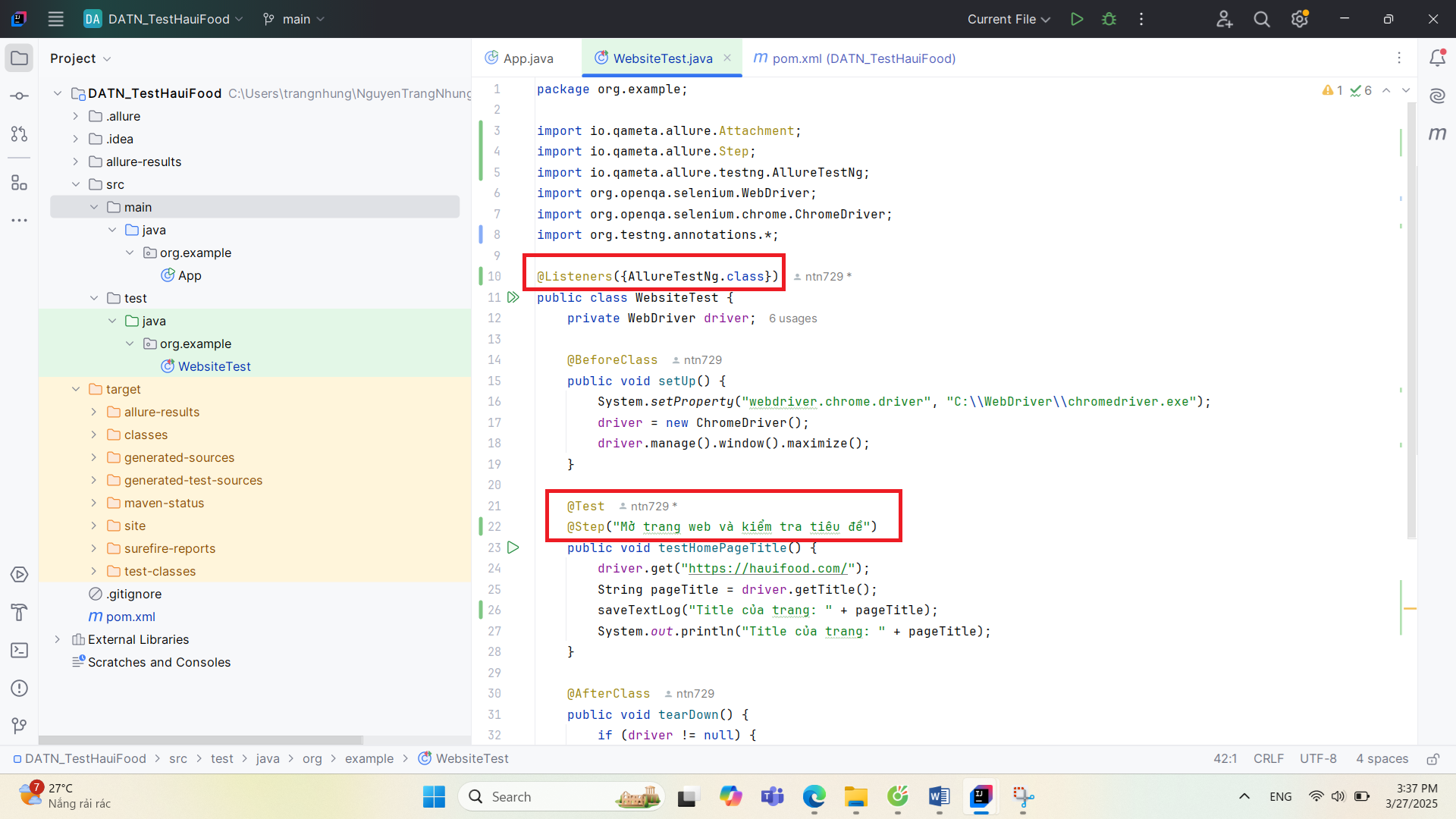


Hình 2.14: Cấu hình plugin Allure trong Maven

* Thực hiện reload lại maven

1. Cấu hình Listener của Allure:

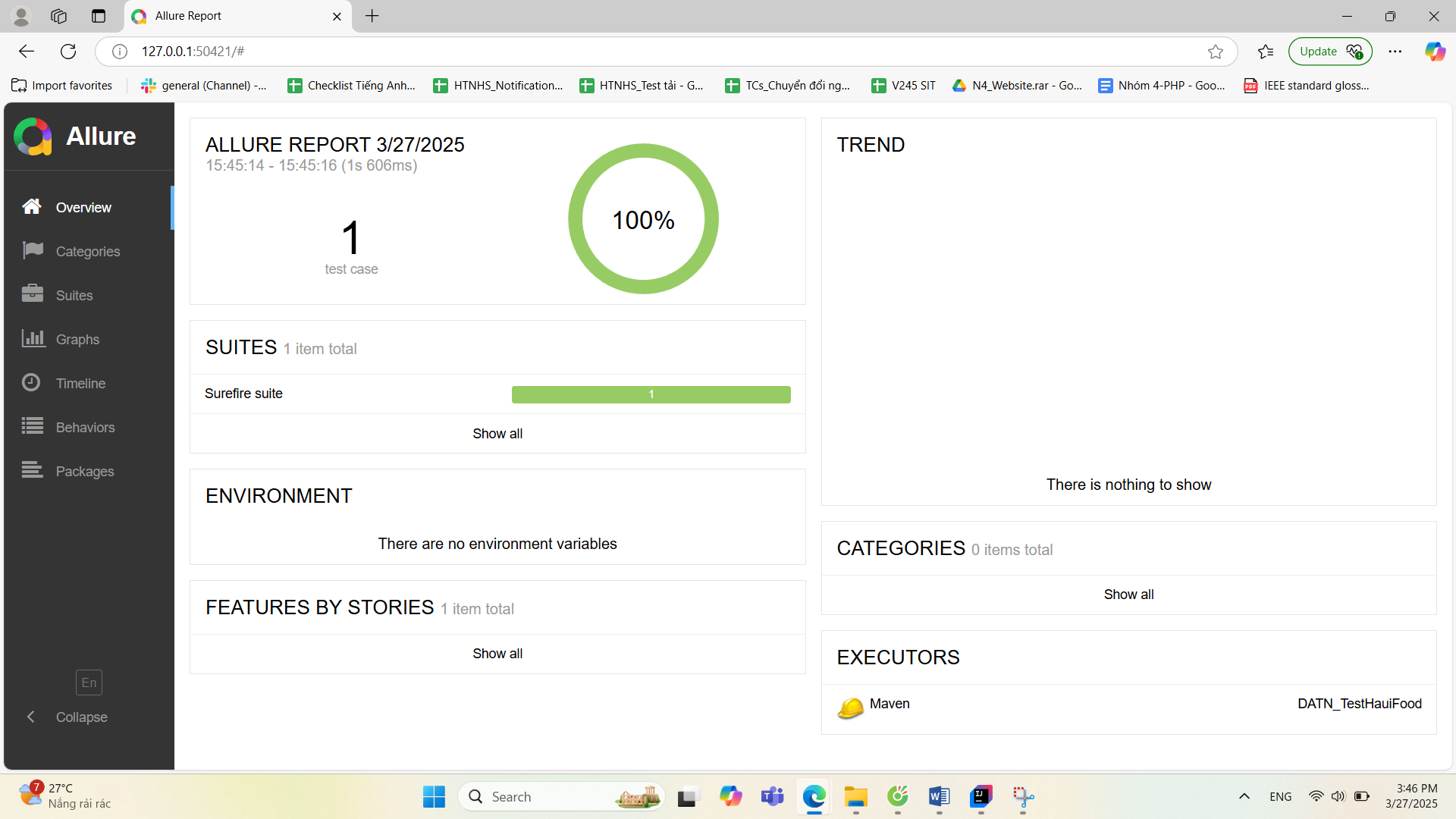
* Cần thêm @Listeners vào class test để Allure có thể ghi nhận kết quả.
* Có thể thêm các annotation thuộc thư viện Allure để mô tả, phân loại và đánh giá mức độ quan trọng của các test case trong báo cáo Allure Report. Các annotation thường dùng bao gồm:
* @Description: Mô tả test case
* @Epic: Xác định test case liên quan đến một tính năng lớn trong hệ thống
* @Feature: Xác định tính năng nhỏ trong hệ thống
* @Story: Xác định user story (tình hướng cụ thể)
* @Severity: Xác định mức độ quan trọng của test case
* @Step: Các bước thực hiện



Hình 2.15: Cấu hình Listener và các annotation của Allure

1. Chạy kiểm thử và sinh báo cáo

* Chạy test script trên terminal bằng lệnh: mvn test hoặc mvn clean test.
* Sinh báo cáo Allure bằng lệnh: mvn allure:report
* Xem báo cáo Allure bằng: allure serve target/allure-results
* Kết quả mong đợi: Allure sẽ tự động tạo một báo cáo HTML từ dữ liệu từ trong thư mục target/allure-results. Một trình duyệt web sẽ mở ra, hiển thị giao diện báo cáo của Allure.



Hình 2.16: Giao diện báo cáo Allure