# MỤC LỤC

[MỤC LỤC i](#_Toc142829712)

[DANH MỤC HÌNH ẢNH iii](#_Toc142829713)

[DANH MỤC BẢNG BIỂU iv](#_Toc142829714)

[LỜI CẢM ƠN 1](#_Toc142829715)

[LỜI CAM ĐOAN 2](#_Toc142829716)

[CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI VÀ KHÁI NIỆM CƠ BẢN 3](#_Toc142829717)

[1.1. LÝ DO CHỌN ĐỀ TÀI 3](#_Toc142829718)

[1.2. MỤC TIÊU VÀ PHẠM VI ĐỀ TÀI 3](#_Toc142829719)

[1.3. Ý NGHĨA ĐỀ TÀI 4](#_Toc142829720)

[1.4. KIỂM THỬ PHẦN MỀM – PHÂN LOẠI KIỂM THỬ 5](#_Toc142829721)

[1.4.1. Kiểm thử phầm mềm 5](#_Toc142829722)

[1.4.2. Kiểm thử thủ công(Manual test) 13](#_Toc142829723)

[1.4.3. Kiểm thử tự động (Automation test) 15](#_Toc142829724)

[1.5. KẾT LUẬN CHƯƠNG 18](#_Toc142829725)

[CHƯƠNG 2. ROBOT FRAMEWORK TRONG KIỂM THỬ TỰ ĐỘNG 19](#_Toc142829726)

[2.1. GIỚI THIỆU VỀ ROBOT FRAMEWORK 19](#_Toc142829727)

[2.2. CÁC THÀNH PHẦN CHÍNH CỦA ROBOT FRAMEWORK 19](#_Toc142829728)

[2.3. THƯ VIỆN DÙNG TRONG ROBOTFRAMEWORK 20](#_Toc142829729)

[2.3.1. Thư viện “SeleniumLibrary” 20](#_Toc142829730)

[2.3.2. Thư viện “Browser” 21](#_Toc142829731)

[2.3.3. Một số keywords được dùng phổ biến 21](#_Toc142829732)

[2.4. GIỚI THIỆU PHẦN MỀM VISUAL STUDIO CODE 22](#_Toc142829733)

[2.5. KẾT LUẬN CHƯƠNG 24](#_Toc142829734)

[CHƯƠNG 3. XÂY DỰNG MÔI TRƯỜNG KIỂM THỬ VÀ CÚ PHÁP VIẾT TEST CASES 25](#_Toc142829735)

[3.1. CÀI ĐẶT PYTHON VÀ PIP 25](#_Toc142829736)

[3.2. CÀI ĐẶT ROBOT FRAMEWORK VÀ CÁC THƯ VIỆN LIÊN QUAN 33](#_Toc142829737)

[3.3. CÚ PHÁP CƠ BẢN CỦA TEST CASE 35](#_Toc142829738)

[3.4. TẠO TEST SUITE VÀ TEST CASE HIERACHY 37](#_Toc142829739)

[3.5. KẾT LUẬN CHƯƠNG 38](#_Toc142829740)

[CHƯƠNG 4. XÂY DỰNG VÀ THỰC HIỆN KIỂM THỬ TỰ ĐỘNG WEBSITE 39](#_Toc142829741)

[4.1. GIỚI THIỆU VỀ WEBSITE MẪU 39](#_Toc142829742)

[4.2. XÂY DỰNG KỊCH BẢN KIỂM THỬ 41](#_Toc142829743)

[4.2.1. Kịch bản kiểm thử cho chức năng đăng nhập 41](#_Toc142829744)

[4.2.2. Kịch bản kiểm thử cho chức năng tạo mới Quản trị viên 43](#_Toc142829745)

[4.2.3. Kịch bản kiểm thử cho chức năng chỉnh sửa người dùng 46](#_Toc142829746)

# DANH MỤC HÌNH ẢNH

# DANH MỤC BẢNG BIỂU

# LỜI CẢM ƠN

Trong hành trình thực hiện đề tài tốt nghiệp **"Tìm hiểu Robot Framework trong kiểm thử tự động Website"**, tôi xin gửi lời cảm ơn sâu sắc và chân thành đến những người đã đồng hành và hỗ trợ em suốt quãng thời gian nghiên cứu và thực hiện đề tài này.

Đầu tiên và trước hết, tôi xin gửi lời cảm ơn chân thành đến thầy hướng dẫn của em, thầy **Vương Quang Phước** đã dành thời gian, kiến thức và sự hướng dẫn tận tâm để tôi có thể hoàn thành đề tài này. Sự hỗ trợ và định hướng từ thầy không chỉ giúp tôi nắm bắt được kiến thức mới mẻ mà còn giúp tôi phát triển kỹ năng nghiên cứu và thực hiện dự án.

Tôi cũng muốn gửi lời cảm ơn đến bạn bè, gia đình và những người thân yêu đã luôn ở bên cạnh tôi, động viên và cổ vũ trong những khoảnh khắc khó khăn và thách thức. Sự ủng hộ của các bạn đã là động lực quý báu giúp tôi vượt qua mọi khó khăn.

Không thể không đề cập đến các giảng viên, bạn bè và đồng nghiệp đã đóng góp ý kiến, gợi ý và hỗ trợ tôi trong quá trình nghiên cứu và thực hiện dự án. Những chia sẻ và kinh nghiệm từ mọi người đã góp phần làm cho đề tài của tôi trở nên phong phú và đa dạng hơn.

Cuối cùng, tôi xin bày tỏ lòng biết ơn chân thành đến Trường Đại học Khoa học – Đại học Huế đã cung cấp môi trường học tập và nghiên cứu tốt nhất, giúp tôi phát triển bản thân và hoàn thành đề tài tốt nghiệp này.

Xin chân thành cảm ơn!

# LỜI CAM ĐOAN

Tôi – tác giả của đề tài tốt nghiệp mang tựa đề "Tìm hiểu Robot Framework trong kiểm thử tự động Website", cam kết rằng tất cả các thông tin và nội dung được trình bày trong báo cáo này là hoàn toàn chân thực và đã được nghiên cứu, thu thập và trình bày một cách trung thực.

Tôi xác nhận rằng mọi nguồn tài liệu, số liệu, thông tin, và kết quả nghiên cứu từ các nguồn khác nhau đã được chúng tôi trích dẫn và ghi rõ trong phần tài liệu tham khảo. Chúng tôi cam kết không sao chép hay sử dụng bất kỳ nội dung nào mà không được trích dẫn một cách đúng đắn.

Tôi hiểu rằng việc vi phạm quy tắc về sự trích dẫn và sử dụng tài liệu của người khác có thể dẫn đến việc mất đạo đức học thuật và bị xem là hành vi gian lận. Tôi cam kết tuân thủ các quy tắc này một cách nghiêm ngặt và tôn trọng quyền sở hữu trí tuệ của người khác.

Bản báo cáo này được viết dưới sự chỉ dẫn và giám sát của thầy **Vương Quang Phước**, người đã hướng dẫn và cung cấp hướng dẫn chi tiết trong quá trình nghiên cứu và viết báo cáo. Tôi xác nhận rằng toàn bộ nội dung được trình bày trong báo cáo này phản ánh kiến thức, quan điểm, và công trình nghiên cứu của chúng tôi.

Xin cam đoan tính chân thực và trung thực của bản báo cáo này.

|  |  |
| --- | --- |
| Xác nhận của người hướng dẫn | Thừa Thiên Huế, ngày…tháng…năm 20…  Tác giả  Hồ Văn Nhật |

# CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI VÀ KHÁI NIỆM CƠ BẢN

## LÝ DO CHỌN ĐỀ TÀI

Trong môi trường phát triển phần mềm ngày càng phức tạp, việc đảm bảo chất lượng sản phẩm là một thách thức quan trọng. Kiểm thử phần mềm đóng một vai trò quan trọng trong việc đảm bảo rằng sản phẩm đáp ứng được các yêu cầu và chức năng một cách đáng tin cậy. Trong tình hình này, kiểm thử tự động đã trở thành một xu hướng thiết thực, giúp tăng hiệu suất và đảm bảo tính nhất quán trong quy trình kiểm thử.

Kiểm thử phầm mềm là một quá trình liên tục, xuyên suốt mọi giai đoạn phát triển phần mềm để đảm bảo rằng phần mềm thỏa mã các yêu cầu thiết kế và các yêu cầu đó đáp ứng các nhu cầu của người dung. Các kỹ thuật kiểm thử phần mềm đã, đang được nghiên cứu, và việc kiểm thử phần mềm đã trở thành quy trình bắt buộc trong các dự án phát triển phần mềm trên thế giới. Kiểm thử phần mềm là một hoạt động rất tốt kém, mất thời gian và khó phát hiện được hết lỗi.

Lý do tôi lựa chọn đề tài "Tìm hiểu Robot Framework trong kiểm thử tự động Website" là để nghiên cứu sâu hơn về một trong những công cụ kiểm thử tự động hàng đầu - Robot Framework. Sự linh hoạt và khả năng tương tác với các thư viện hỗ trợ khác nhau của Robot Framework đã thu hút sự quan tâm và ứng dụng rộng rãi trong cộng đồng phát triển phần mềm. Tôi muốn tìm hiểu cách sử dụng Robot Framework để kiểm thử các chức năng của các trang web, đặc biệt là trong bối cảnh tăng cường xu hướng phát triển ứng dụng web.

Đồ án này không chỉ giúp tôi có cái nhìn tổng quan về Robot Framework, mà còn giúp chúng tôi áp dụng kiến thức thực tế vào việc xây dựng các test cases kiểm thử tự động cho các trang web thực tế. Tôi hi vọng rằng thông qua đồ án này, tôi có thể chia sẻ những kiến thức cơ bản về Robot Framework cùng như trải nghiệm thực tế trong việc sử dụng công cụ này để kiểm thử tự động các ứng dụng web.

## MỤC TIÊU VÀ PHẠM VI ĐỀ TÀI

* **Mục tiêu của đề tài:**
* Nắm vững kiến thức về cách sử dụng Robot Framework trong kiểm thử tự động các trang web.
* Hiểu rõ cách xây dựng các test cases và sử dụng các keywords để kiểm thử tự động chức năng của website.
* Áp dụng Robot Framework để thực hiện các kịch bản kiểm thử trên các trang web mẫu.
* Hiểu được cách tự động hóa quy trình kiểm thử và tạo báo cáo kết quả kiểm thử tự động.
* **Phạm vi của đề tài:**
* Tìm hiểu tổng quan về Robot Framework, bao gồm cách cài đặt, cấu hình môi trường và cú pháp cơ bản.
* Xây dựng các test cases kiểm thử tự động cho một trang web mẫu.
* Sử dụng các keywords và thư viện mở rộng để thực hiện các kịch bản kiểm thử cụ thể.
* Tự động hóa quy trình kiểm thử bằng cách tạo và thực thi test suite.
* So sánh Robot Framework với một số công cụ kiểm thử tự động khác để nhận biết ưu điểm và hạn chế của nó.
* Phạm vi của đề tài sẽ tập trung vào việc áp dụng Robot Framework trong kiểm thử tự động các trang web. Đồ án sẽ không đi sâu vào các khía cạnh phức tạp khác như kiểm thử API hay ứng dụng di động, mà tập trung vào việc hiểu và thực hành việc sử dụng Robot Framework trong kiểm thử web.

## Ý NGHĨA ĐỀ TÀI

* Việc nghiên cứu và thực hiện đề tài "Tìm hiểu Robot Framework trong kiểm thử tự động Website" mang lại nhiều ý nghĩa quan trọng trong lĩnh vực phát triển phần mềm và kiểm thử:
* **Nâng cao hiệu quả kiểm thử tự động:** Sự tăng cường sự nhất quán và khả năng tái sử dụng của Robot Framework giúp cải thiện hiệu quả kiểm thử tự động. Điều này giúp tiết kiệm thời gian và tài nguyên trong việc kiểm thử các ứng dụng web.
* **Tích hợp kiểm thử sớm trong quy trình phát triển:** Hiểu rõ cách sử dụng Robot Framework giúp đưa quy trình kiểm thử vào giai đoạn phát triển sớm hơn, từ đó giúp phát hiện và sửa lỗi một cách nhanh chóng và tiết kiệm chi phí.
* **Phát triển kỹ năng và kiến thức:** Việc nắm vững cách sử dụng một công cụ kiểm thử phổ biến như Robot Framework không chỉ giúp nâng cao kỹ năng thực hiện kiểm thử tự động mà còn giúp mở rộng hiểu biết về quy trình phát triển phần mềm.
* **Chia sẻ kiến thức và kinh nghiệm:** Kết quả nghiên cứu và thực hiện trong đề tài có thể được chia sẻ và đóng góp vào cộng đồng phát triển phần mềm. Bài viết, tài liệu và trải nghiệm cá nhân có thể hỗ trợ người khác hiểu rõ hơn về Robot Framework và kiểm thử tự động.
* **Đóng góp vào sự phát triển của ngành công nghiệp phần mềm:** Nắm vững kiến thức về Robot Framework và kiểm thử tự động giúp bạn đóng góp vào việc cải thiện chất lượng phần mềm và nâng cao hiệu suất của các dự án phát triển.
* Tóm lại, việc thực hiện đề tài này không chỉ mang lại lợi ích cho người thực hiện mà còn góp phần cải thiện quy trình phát triển phần mềm và kiểm thử tự động trên toàn ngành công nghiệp.

## KIỂM THỬ PHẦN MỀM – PHÂN LOẠI KIỂM THỬ

### Kiểm thử phầm mềm

Một chương trình mới được tạo ra thường chứa vài lỗi trong 100 dòng bao gồm lỗi từ quá trình lập trình và lỗi từ quá trình thiết kế. Nếu một chương trình chứa lỗi được dung để vận hành một hệ thống trực tuyến thì những hư hỏng nghiêm trọng phát sinh ra không chỉ ảnh hưởng tới công ty vận hành hệ thống đó mà còn ảnh hưởng cả tới công chúng lớn bên ngoài.

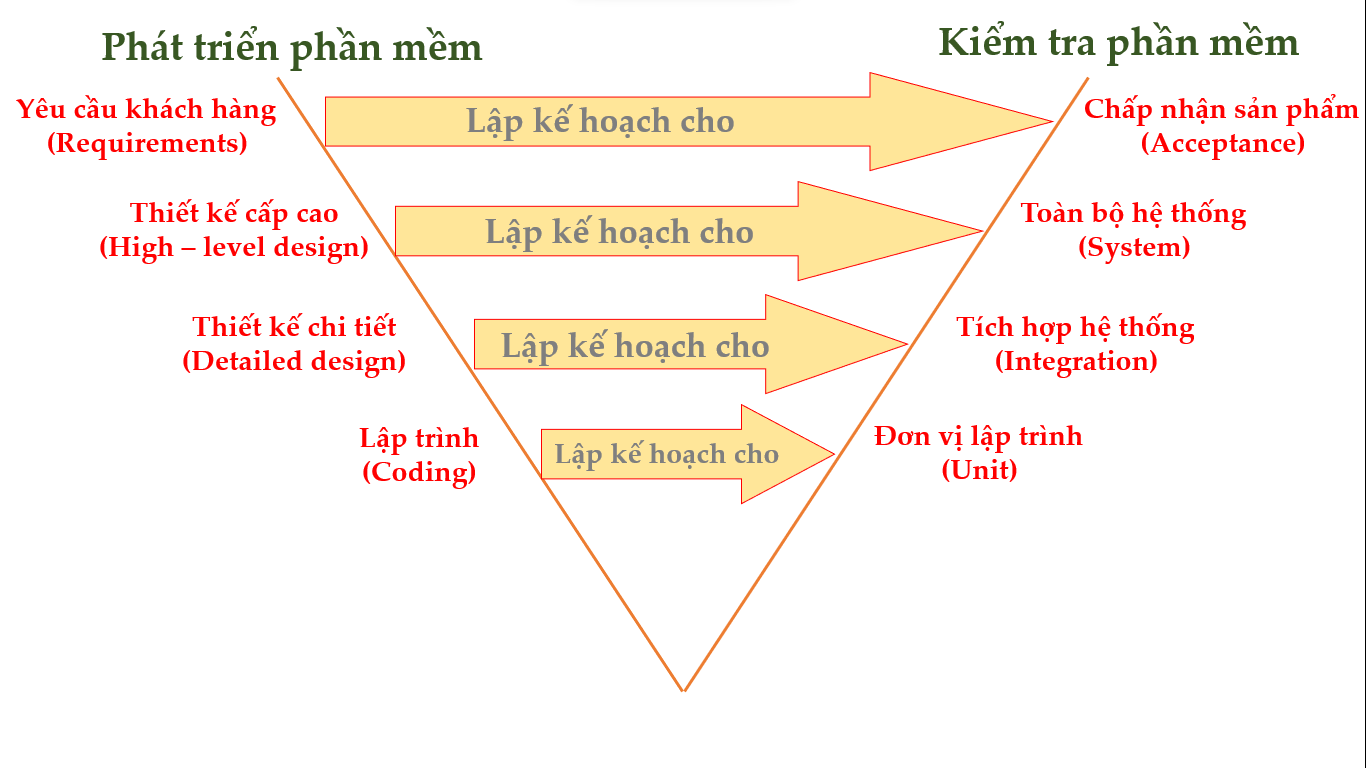
Do đó việc kiểm thử phần mềm phải được tiến hành trước khi chuyển giao sản phẩm công nghiệp. Việc kiểm thử phần mềm cũng phải được tiến hành theo một trình tự kiểm thử đặc biệt để kiểm chứng rằng chương trình và hệ thống mà nó điều khiển có thể vận hành tương ứng với các đặc tả. Mặc dù chúng ta không thể đảm bảo loại bỏ hoàn toàn hết lỗi trong chương trình nhưng chúng ta vẫn có thể giảm số lỗi đó tới mức tối thiểu nhất nếu chúng ta kiểm thử chương trình theo cách chính xác, hiệu quả.

Một hoạt động mang tính sống còn trong các dự án sản xuất hoặc gia công phần mềm, đó là kiểm tra (Testing). Người làm phần mềm chắc hẳn không ai nghi ngờ vai trò quan trọng của nó, tuy nhiên không phải ai cũng hiểu rõ hoạt động này. Bản than công việc kiểm thử phần mềm cũng là một lĩnh vực hoạt động độc lập và khá “hấp dẫn”. Cùng với các dự án gia công sản xuất phần mềm, hiện cũng có khá nhiều dự án mà nội dung công việc chỉ là kiểm tra những phần đã được khách hang phát triển sẵn.

Thực tế cho thấy kiểm thử phần mềm là công việc mà bất cứ người nào từng tham gia phát triển phần mềm đều biết và từng làm. Kiểm thử phần mềm bao gồm việc “chạy thử” phần mềm hay một chức năng của phần mềm, xem nó “chạy” đúng như mong muốn hay không. Việc kiểm tra này có thể thực hiện từng chặng, sau mỗi chức năng hoặc module được phát triển, hoặc thực hiện sau cùng, khi phần mềm đã được phát triển hoàn tất.

Do đó kiểm thử một sản phẩm phần mềm là xây dựng một cách có chủ đích những tập dữ liệu và dãy thao tác nhằm đánh giá một số hoặc toàn bộ các tiêu chuẩn của sản phẩm phần mềm đó.

Kiểm tra phần mềm có nhiều mức độ khác nhau và có mối tương quan với các chặng phát triển trong dự án phát triển phần mềm.



Hình 1.1. Tương quan giữa các chặng trong PTPM và KTPM

#### Vai trò của kiểm thử phần mềm

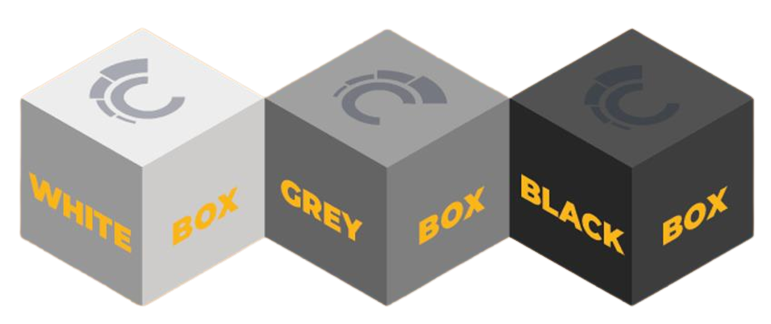
* Việc tạo ra một sản phẩm phần mềm phải trải qua nhiều giai đoạn, người ta gọi là quy trình phát triển phần mềm, bắt đầu từ khi bắt đầu có ý tưởng cho đến khi đưa ra sản phẩm phần mềm thực thi. Khối lượng công việc trong từng giai đoạn của quá trình sản xuất phần mềm cũng thay đổi theo thời gian. Bảng 1.1 minh hoa cụ thể hơn về điều này.

Bảng 1.1. Tỉ lệ công việc của các giai đoạn phát triển phần mềm

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Giai đoạn | Phân tích yêu cầu | Thiết kế sơ bộ | Thiết kế chi tiết | Lập trình và kiểm thử đơn vị | Tích hợp và kiểm thử tích hợp | Kiểm thử hệ thống |
| Hai thập kỉ 1960 - 1970 |  | | |  | 10% | |
| Thập kỉ 1980 |  | |  | |  | |
| Thập kỉ 1990 |  |  | |  | | |
| Hiện nay |  | | |  |  | |

* Như vậy, một sản phẩm phần mềm không chỉ đơn giản là các đoạn mã chương trình mà còn rất nhiều phần ẩn đằng sau nó. Vì vậy, việc mắc lỗi không chỉ xảy ra trong khi lập trình mà còn xảy ra cao hơn trong các công đoạn khác của quy trình phát triển một sản phẩm phần mềm. Việc kiểm thử cũng vì thế phải được tiến hành trong tất cả các phần tạo nên một sản phẩm phần mềm.

#### Các kỹ thuật kiểm thử phần mềm



Hình 1.2. Các loại kỹ thuật kiểm thử

* **Kiểm thử hộp đen (Black box testing):** dung để kiểm tra chức năng mà không xem xét mã nguồn cũng như cấu trúc chương trình bên trong. Thường kiểm thử hộp đen quan tâm nhiều đến các bộ dữ liệu kiểm thử đầu vào

**Ưu điểm:**

* + Các tester khi dùng phương pháp này sẽ không cần liên quan đến code.
  + Có thể tìm được nhiều bug hơn.
  + Việc kiểm thử được thực hiện bởi một cách độc lập với các Dev, cho phép quan điểm khách quan và tránh sự thiên vị.

**Nhược điểm:**

* Chỉ có một số lượng nhỏ các đầu vào có thể được kiểm thử và nhiều đường dẫn chương trình hoặc 1 vài phần cuối sẽ không được kiểm thử.
* Các kiểm thử có thể thừa nếu nhà thiết kế/ nhà phát triển phần mềm đã chạy kiểm thử.

Vì vậy, black box testing có ưu điểm là sản phẩm phần mềm được Kiểm thử theo một quan điểm độc lập tuy vậy vẫn còn khá nhiều nhược điểm đáng lưu ý.

* **Kiểm thử hộp trắng (White box testing):** khác với kiểm thử hộp đen, kiểm thử hộp trắng xem xét mọi module trong chương trình, các luồng thực hiện công việc để từ đó đưa ra các chiến lược kế hoạch cụ thể cho việc kiểm thử.

**Ưu điểm:**

* + Dễ dàng tự động.
  + Cung cấp các quy tắc dựa trên kỹ thuật rõ ràng cho thời điểm ngừng kiểm thử.
  + Buộc các chuyên gia kiểm thử phải suy luận cẩn thận về việc test lỗi vì vậy lỗi sẽ được triệt để.

**Nhược điểm:**

* Khá tốn thời gian và công sức.
* Vẫn sẽ tồn tại lỗi.
* Để kiểm thử được bằng phương pháp này cần có kinh nghiệm và trình độ chuyên sâu về kiểm thử.
* **Kiểm thử hộp xám (Grey box testing):** là một trong các phương pháp kiểm thử phần mềm phổ biến nhất hiện nay, đây là một kĩ thuật kiểm thử mới dựa trên những đặc tính của cả kiểm thử hộp đen và hộp trắng. Mục tiêu chính của kiểm thử hộp xám là kiểm thử các ứng dụng trên nền web (web based).

**Ưu điểm:**

* + Là sự kết hợp của kiểm thử hộp đen và hộp trắng nên sẽ tối ưu hơn.
  + Kiểm thử bằng phương pháp hộp màu xám có thể thiết kế kịch bản kiểm thử phức tạp một cách thông minh hơn.

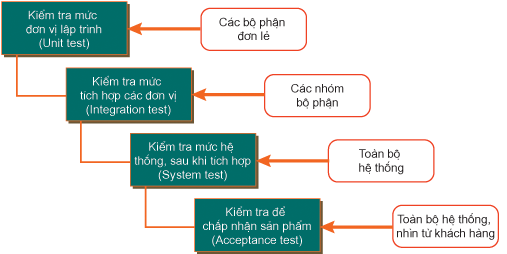
**Nhược điểm:**

* Rất khó để liên kết lỗi khi thực hiện kiểm thử hộp xám cho một ứng dụng có hệ thống phân tán.

Trên đây là 3 phương pháp kiểm thử phần mềm cơ bản nhất mà bất cứ một lập trình viên nào cũng cần nắm. Việc lựa chọn phương pháp nào phụ thuộc vào khả năng cũng như dự án mà bạn đang thực hiện.

#### Các giai đoạn hay cấp độ kiểm thử phần mềm

Kiểm thử phần mềm nói chung có 4 mức độ sau đây:



Hình 1.3. Các mức độ cơ bản của kiểm thử phần mềm

* Kiểm thử mức đơn vị – Unit Test:
* Một đơn vị là một thành phần phần mềm nhỏ nhất mà ta có thể kiểm thử được. Ví dụ, các hàm (Function), thủ tục (Procedure), lớp (Class) hay phương thức (Method) đều có thể được xem là Unit.
* Unit Test thường do lập trình viên thực hiện. Công đoạn này cần được thực hiện càng sớm càng tốt trong giai đoạn viết code và xuyên suốt chu kỳ phát triển phần mềm. Unit Test đòi hỏi kiểm thử viên có kiến thức về thiết kế và code của chương trình. Mục đích của Unit Test là bảo đảm thông tin được xử lý và xuất chính xác, trong mối tương quan với dữ liệu nhập và chức năng của Unit. Tất cả các nhánh bên trong Unit đều phải được kiểm tra để phát hiện nhánh phát sinh lỗi. Một nhánh thường là một chuỗi các lệnh được thực thi trong một Unit. Ví dụ: chuỗi các lệnh điều kiện If và nằm giữa then …else là một nhánh. Việc chọn lựa các nhánh để đơn giản hóa việc kiểm thử và quét hết Unit đòi hỏi phải có kỹ thuật, đôi khi phải dung thuật toán để chọn lựa.
* Cùng với các mục kiểm thử khác, Unit Test cũng đòi hỏi phải chuẩn bị trước các ca kiểm thử (Test case) hoặc kịch bản kiểm thử (Test script), trong đó chỉ định rõ dữ liệu đầu vào, các bước thực hiện và dữ liệu đầu ra mong muốn. Các Test case và Test script này nên được giữ lại để tái sử dụng.
* Kiểm thử tích hợp – Integration Test:
* Integration Test kết hợp các thành phần của một ứng dụng và kiểm thử như một ứng dụng đã hoàn thành. Trong khi Unit Test kiểm tra các thành phần và Unit riêng lẻ thì Integration Test kết hợp chúng lại với nhau và kiểm tra sự giao tiếp giữa chúng.
* Hai mục tiêu chính của Integration Test:
  + Phát hiện lỗi giao tiếp xảy ra giữa các Unit.
  + Tích hợp các Unit đơn lẻ thành các hệ thống nhỏ (Subsystem) và cuối cùng là nguyên hệ thống hoàn chỉnh (System) chuẩn bị cho kiểm thử ở mức hệ thống (System Test).
* Integration Test chỉ nên thực hiện trên những Unit đã được kiểm tra cẩn thận trước đó bằng Unit Test, và tất cả các lỗi ở mức Unit đã được sửa chữa. Một số người hiểu sai rằng Unit một khi đã qua giai đoạn Unit Test với các giao tiếp giả lập thì không cần phải thực hiện Integration Test nữa.
* Một chiến lược cần quan tâm trong Integration Test là nên tích hợp dần từng Unit. Một Unit tại một thời điểm được tích hợp vào một nhóm các Unit khác đã tích hợp trước đó và đã hoàn tất các đợt Integration Test trước đó. Lúc này, ta chỉ cần kiểm thử giao tiếp của Unit mới them vào hệ thống các Unit đã tích hợp trước đó, điều này sẽ làm cho số lượng cần kiểm thử giảm đi rất nhiều và sai sót sẽ giảm đáng kể.
* Có 4 loại kiểm thử trong Integration Test:
  + Kiểm thử cấu trúc (Structure Test): Kiểm thử cấu trúc nhằm bảo đảm các thành phần bên trong của một chương trình chạy đúng và chú trọng đến hoạt động của các thành phần cấu trúc nội tại của chương trình.
  + Kiểm thử chức năng (Function Test): Kiểm thử chức năng chỉ chú trọng đến chức năng của chương trình, không quan tâm đến cấu trúc bên trong, chỉ khảo sát chức năng của chương trình theo yêu cầu kỹ thuật.
  + Kiểm thử hiệu năng (Performance Test): Kiểm thử việc vận hành của hệ thống.
  + Kiểm thử khả năng chịu tải (Stress Test): Kiểm thử các giới hạn của hệ thống.
* Kiểm thử hệ thống – System Test:
  + Mục đích System Test là kiểm thử thiết kế và toàn bộ hệ thống (sau khi tích hợp) có thỏa mãn yêu cầu đặt ra hay không.
  + System Test bắt đầu khi tất cả các bộ phận của phần mềm đã được tích hợp thành công. Loại kiểm thử này rất tốn công sức và thời gian. Trong nhiều trường hợp, việc kiểm thử đòi hỏi một số thiết bị phụ trợ, phần mềm hoặc phần cứng đặc thù, đặc biệt là các ứng dụng thời gian thực, hệ thống phân bố, hoặc hệ thống nhúng. Ở mức độ hệ thống, người kiểm thử cũng tìm kiếm các lỗi, nhung 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.
  + Điểm khác nhau then chốt giữa Integration Test và System Test là System Test chú trọng các hành vi và lỗi trên toàn hệ thống, còn Integration Test chú trọng sự giao tiếp giữa các đơn thể hoặc đối tượng khi chúng làm việc cùng nhau. Ta phải thực hiện Unit Test và Integration Test để bảo đảm mọi Unit và sự tương tác giữa chúng hoạt động chính xác khi thực hiện System Test.
  + Sau khi hoàn thành Integration Test, một hệ thống phần mềm đã được hình thành cùng với các thành phần đã được kiểm tra đầy đủ. Tại thời điểm này, lập trình viên hoặc kiểm thử viên bắt đầu kiểm thử phần mềm như một hệ thống hoàn chỉnh. Việc lập kế hoạch cho System Test nên bắt đầu từ giai đoạn hình thành và phân tích các yêu cầu.
  + System Test kiểm thử tất cả các hành vi, chức năng của phần mềm và các yêu cầu về chất lượng như độ tin cậy, tính tiện lợi khi sử dụng, hiệu năng và bảo mật. Mức kiểm thử này đặc biệt thích hợp cho việc phát hiện lỗi giao tiếp với phần mềm hoặc phần cứng bên ngoài, chẳng hạn các lỗi “tắc nghẽn” (deadlock) hoặc chiếm dụng bộ nhớ. Sau giai đoạn System Test, phần mềm thường đã sẵn sàng cho khách hàng hoặc người dùng, cuối cùng kiểm thử chấp nhận sản phẩm (Acceptance Test) hoặc dùng thử (Alpha/Beta Test). System Test thường được thực hiện bởi một nhóm kiểm thử viên hoàn toàn độc lập với nhóm phát triển dự án. System Test gồm nhiều loại kiểm thử khác nhau, phổ biến nhất gồm:
    - Kiểm thử chức năng (Function Test): Bảo đảm các hành vi cảu hệ thống thỏa màn đúng yêu cầu thiết kế.
    - Kiểm thử hiệu năng (Performance Test): Bảo đảm tối ưu việc phân bố tài nguyên hệ thống (ví dụ bộ nhớ) nhằm đạt các chỉ tiêu như thời gian xử lý hay đáp ứng yêu cầu truy vấn…
    - Kiểm thử khả năng chịu tải (Stress Test hay Load Test): Bảo đảm hệ thống vận hành đúng dưới áp lực cao (ví dụ nhiều người truy xuất cùng lúc). Stress Test tập trung vào các trạng thái tới hạn, các “điểm chết”, các tình huống bất thường như đang giao dịch thì ngắt kết nối (xuất hiện nhiều trong kiểm tra thiết bị như POS, ATM…)…
    - Kiểm thử cấu hình (Configuration Test)
    - Kiểm thử bảo mật (Security Test): Bảo đảm tính toàn vẹn, bảo mật của dữ liệu và của hệ thống.
    - Kiểm thử khả năng phục hồi (Recovery Test): Bảo đảm hệ thống có khả năng khôi phục trạng thái ổn định trước đó trong tình huống mất tài nguyên hoặc dữ liệu; đặc biệt quan trọng đối với các hệ thống giao dịch như ngân hàng trực tuyến…

Nhìn từ quan điểm người dung, các cấp độ kiểm thử trên rất quan trọng: Chúng bảo đảm hệ thống đủ khả năng làm việc trong môi trường thực. Tùy yêu cầu và đặc trưng của từng hệ thống, tùy khả năng và thời gian cho phép của dự án, khi lập kế hoạch, người quản lý dự án sẽ quyết định áp dụng những loại kiểm thử nào.

* Kiểm thử chấp nhận sản phẩm – Acceptance Test:
  + Sau giai đoạn System Test là Acceptance Test, được khách hàng hoặc ủy quyền cho một nhóm thứ ba thực hiện. Mục đích của Acceptance Test là để chứng minh phần mềm thỏa mãn tất cả yêu cầu của khách hàng và khách hàng chấp nhận sản phẩm.
  + Acceptance Test có ý nghĩa hết sức quan trọng. Mặc dù trong hầu hết mọi trường hợp, các phép kiểm thử của System Test và Acceptance Test gần như tương tự, nhưng bản chất và cách thức thực hiện lại rất khác biệt.
  + Đối với những sản phẩm dành bán rộng rãi trên thị trường, thông thường sẽ thông qua hai loại kiểm thử gọi là kiểm thử Alpha – Alpha Test và kiểm thử Beta – Beta Test. Alpha Test, người dung kiểm thử phần mềm ngay tại nơi phát triển phần mềm, lập trình viên ghi nhận các lỗi hoặc phản hồi và lên kế hoạch sửa chữa. Beta Test, phần mềm được gửi tới cho người dùng để kiểm thử ngay trong môi trường thực, lỗi hoặc phản hồi sẽ gửi ngược lại cho lập trình viên để sửa chữa.
  + Nếu khách hàng không quan tâm và không tham gia vào quá trình phát triển phần mềm thì kết quả Acceptance Test sẽ sai lệch rất lớn, mặc dù phần mềm đã trải qua tất cả các kiểm thử trước đó. Sự sai lệch này liên quan đến việc hiểu sai yêu cầu cũng như sự mong chờ của khách hàng.
  + Gắn liền với giai đoạn Acceptance Test thường là một nhóm những dịch vụ và tài liệu đi kèm, phổ biến như hướng dẫn cài đặt, sử dụng v.v… Tất cả tài liệu đi kèm phải được cập nhật và kiểm thử chặt chẽ.

#### Một số loại hình kiểm thử phổ biến

Hiên nay, do sự phát triển mạnh mẽ của công nghệ phần mềm nên có một số loại hình kiểm thử tiêu biểu như:

* Kiểm thử các phần mềm trên Desktop: đây là các ứng dụng được cài đặt trực tiếp trên máy tính cá nhân nhằm thực hiện các chức năng theo yêu cầu của người dung. Đây vẫn đang là những ứng dụng phổ biến nhất.
* Kiểm thử Web hay kiểm thử trên đám mây: với sự lớn mạnh của Internet thì các ứng dụng web cũng ngày càng phát triển và đang dần thay thế các ứng dụng trên Desktop truyền thống như Google Document, Microsoft web, app,…
* Kiểm thử trên Mobile: ngày nay xã hội với sự phát triển nhanh chóng, các thiết bị di động (điện thoại thông minh, máy tính bảng,…) có số lượng người dung cũng đang tăng lên chóng mặt, cùng với đó là số lượng phần mềm phục vụ cho nhu cầu cũng tăng cao, vì vậy đây là một lĩnh vực đầy tiềm năng và thách thức trong công nghệ phần mềm.

### Kiểm thử thủ công(Manual test)

* ***Kiểm thử thủ công*** là quá trình kiểm tra và đánh giá các tính năng, chức năng và giao diện của phần mềm bằng cách thao tác và kiểm tra bằng tay. Dù đã có sự phát triển của kiểm thử tự động, kiểm thử thủ công vẫn đóng một vai trò quan trọng trong việc đảm bảo tính ổn định và chất lượng của sản phẩm phần mềm.
* ***Phương pháp kiểm thử thủ công và quy trình***: Trong quá trình kiểm thử thủ công, chúng tôi đã thực hiện các bước sau:
  + ***Xác định phạm vi kiểm thử***: Đầu tiên, chúng tôi xác định các tính năng và chức năng cần kiểm thử thủ công, dựa trên tài liệu yêu cầu và thiết kế.
  + ***Lập kế hoạch kiểm thử:*** Chúng tôi thiết lập kế hoạch kiểm thử thủ công, xác định các trường hợp kiểm thử, tạo bảng kiểm thử, và định thời gian kiểm thử.
  + ***Thực hiện kiểm thử:*** Chúng tôi tiến hành thực hiện các bước kiểm thử thủ công theo kế hoạch. Điều này bao gồm việc thao tác trên giao diện của ứng dụng và kiểm tra tính năng, chức năng, hiển thị, và dữ liệu.
  + ***Ghi nhận kết quả:*** Chúng tôi ghi nhận kết quả từng bước kiểm thử, bao gồm cả kết quả dương và tiêu cực, và ghi chép các lỗi và vấn đề phát hiện.
  + ***Xác minh và báo cáo:*** Chúng tôi xác minh và kiểm tra lại các lỗi đã tìm thấy, sau đó tạo báo cáo kiểm thử thủ công. Báo cáo này bao gồm mô tả kết quả kiểm thử, danh sách các lỗi, và các yêu cầu cần phải được xem xét.
* ***Lợi ích của kiểm thử thủ công:***
  + ***Phát hiện lỗi động và sâu hơn:*** Kiểm thử thủ công cho phép chúng tôi phát hiện các lỗi mà kiểm thử tự động có thể bỏ sót, bao gồm các tình huống động và phức tạp hơn.
  + ***Đảm bảo trải nghiệm người dùng:*** Kiểm thử thủ công giúp chúng tôi đảm bảo rằng giao diện và trải nghiệm người dùng đáp ứng đúng yêu cầu và mong đợi.
  + ***Kiểm tra tích hợp:*** Kiểm thử thủ công giúp chúng tôi kiểm tra tích hợp giữa các thành phần và tính tương thích.
* **Các loại kiểm thử thủ công:**
  + ***Kiểm thử thăm dò:*** sử dụng khi cần dung kiến thức, kinh nghiệm, phân tích/logic, kỹ năng, sáng tạo và trực giác. (Explode testing). Kiểm thử này được dùng khi có ít tài liệu đặc tả, hoặc một thời gian ngắn để thực hiện.
  + ***Usability Testing:*** sử dụng khi cần để đo độ thân thiện, hiệu quả, thuận tiện của phần mềm hoặc sản phẩm cho người dùng cuối. Khi cần sự quan sát của con người.
  + ***Kiểm thử Ad –* *hoc*:** dùng khi trong kịch bản đặc tả không có phương pháp cụ thể. Nó là phương pháp hoàn toàn không có kế hoạch kiểm thử. Cần có sự hiểu biết và cái nhìn sâu sắc của con người là yếu tố quan trọng duy nhất.

### Kiểm thử tự động (Automation test)

#### Tổng quan về kiểm thử tự động

Ngày nay, tự động hóa được ứng dụng trong rất nhiều lĩnh vực với mục đích là rất đa dạng tùy vào nhu cầu của mỗi lĩnh vực và điểm chung nhất là giảm chi phí, nhân lực, thời gian cũng như sai sót. Tự động hóa trong kiểm thử phần mềm cũng không nằm ngoài mục đích đó. Thực tế đã cho thấy, áp dụng tự động hóa kiểm thử mang lại những hiệu quả, thành công nhất định cho khâu kiểm thử thông qua các công cụ phần mềm.

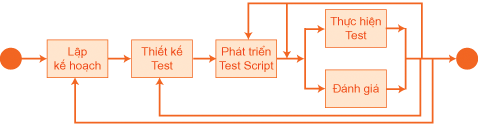
#### Khái niệm

Kiểm thử tự động là phương pháp sử dụng phần mềm hay các công cụ để xử lý tự động các bước thực hiện test case mà không cần sự can thiệp của con người. Trong kiểm thử tự động, phần mềm phải được biên dịch và chạy thực sự.

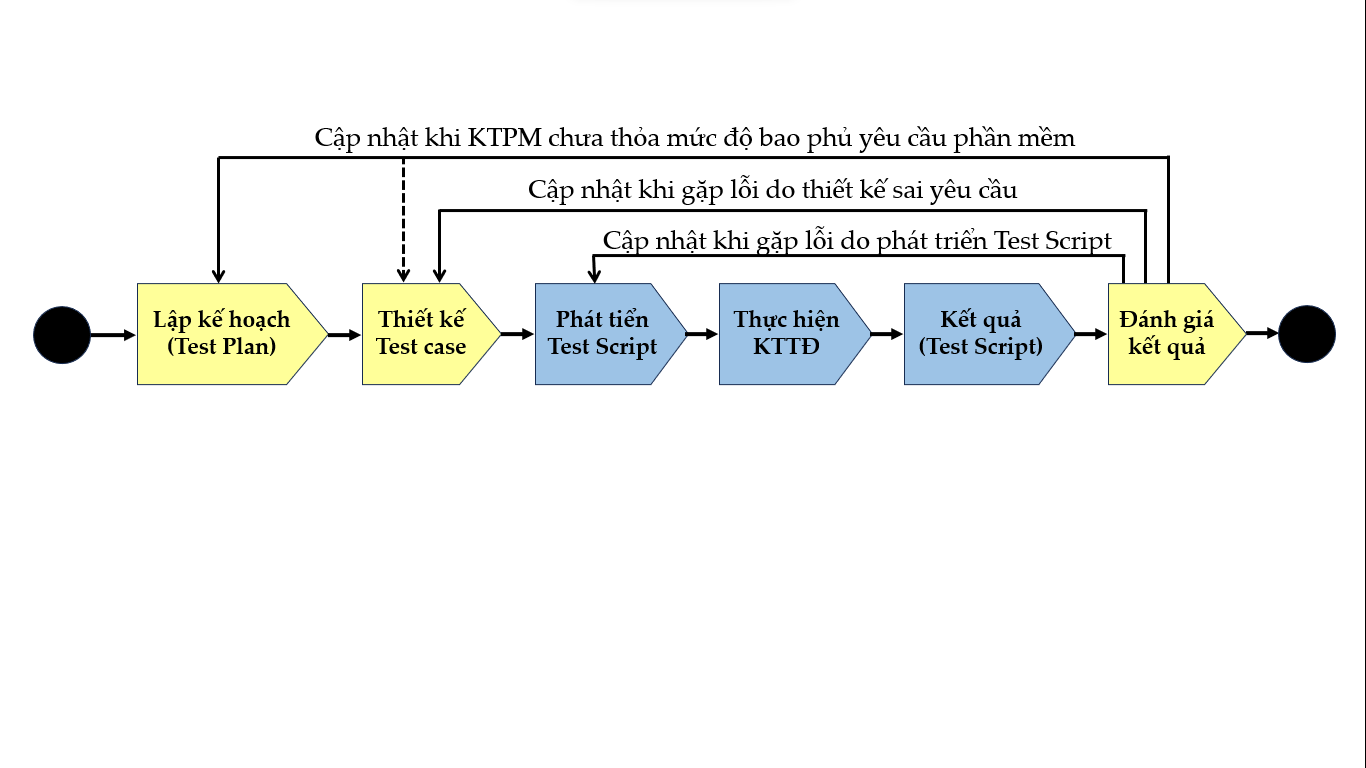
Mục tiêu:

* Giảm bớt công sức và thời gian thực hiện quá trình kiểm thử cho cả một kế hoạch kiểm thử.
* Tăng độ tin cậy.
* Rèn luyện kỹ năng lập trình cho tester.
* Giảm chi phí cho tổng quá trình kiểm thử.

#### Quy trình kiểm thử tự động



Hình 1.4. Quy trình kiểm thử tự động



Hình 1.5. Tương quan giữa kiểm thử tự động và toàn bộ chu trình kiểm tra phần mềm

Việc phát triển kiểm thử tự động cũng tuân theo các bước phát triển phần mềm, ta phải xem việc phát triển phần mềm giống như phát triển một dự án. Giống như phát triển phần mềm, chung ta thực hiện các bước cơ bản sau:

* ***Xây dựng yêu cầu*:** thu thập các đặc tả yêu cầu, lựa chọn những phần cần thực hiện kiểm thử tự động, lập kế hoạch kiểm thử.
* ***Phân tích thiết kế mô hình kiểm thử tự động:*** xây dựng mô hình phát triển kiểm thử tự động, thiết kế và xây dựng các test case để thực thi.
* ***Phát tiển Test Script:*** 
  + Tạo Test Script: giai đoạn này chúng ta sẽ sử dụng test tool để ghi lại các thao tác lên phần mềm cần kiểm tra và tự động sinh ra Test Script.
  + Chỉnh sửa Test Script: chỉnh sửa để Test Script thực hiện kiểm tra theo đúng yêu cầu đặt ra, cụ thể là làm theo test case cần thực hiện.
* ***Chạy* *Test Script:***giám sát các hoạt động kiểm thử phần mềm của Test Script.
* ***Kiểm tra kết quả:*** kiểm tra kết quả thông báo ngay sau khi thực hiện kiểm thử tự động.
* ***Đánh giá kết quả kiểm thử:*** thông qua báo cáo kết quả kiểm thử, bổ sung, chỉnh sửa những sai sót.

#### Các loại kiểm thử

* ***Kiểm thử hồi quy:*** kiểm thử tự động rất hiệu quả khi áp dụng cho giai đoạn re – test và test hồi quy. Hoặc cần chạy một lượng testcases trong một thời gian ngắn.
* ***Thực hiện lặp đi lặp lại:*** khi chúng ta chạy test mỗi ngày cần phải add Data hoặc chạy sẵn các bước nền để test các khâu quan trọng. Khi chạy full flow mà chỉ test một phần nhỏ trong đó thì các bước đã chạy qua rồi sẽ lặp đi lặp lại thì khi đó Automation sẽ thấy tác dụng rõ nhất.
* ***Thử nghiệm tính năng:*** kiểm tra sự kết hợp giữa nhiều giá trị đầu vào ở một bước nào đó. Kiểm tra nhiều màn hình của dữ liệu đầu vào. Khi muốn thực hiện kiểm thử hiệu năng hoặc load test thì kiểm thử tự động gần như là biện pháp duy nhất.

#### Ưu và nhược điểm của kiểm thử tự động

Bảng 1.2. Ưu và nhược điểm của kiểm thử tự động

|  |  |
| --- | --- |
| ƯU ĐIỂM | NHƯỢC ĐIỂM |
| * Không cần đến sự can thiệp của kiểm thử viên. * Giảm chi phí khi thực hiện kiểm tra số lượng lớn test case hoặc test case lặp lại nhiều lần. * Giả lập tình huống khó có thể thực hiện bằng tay. | * Mất chi phí tạo các script để thực hiện kiểm thử tự động. * Tốn chi phí cho bảo trì các script. * Đòi hỏi kiểm thử viên phải có kỹ năng tạo các script kiểm thử tự động. * Không áp dụng được trong việc tìm lỗi mới của phần mềm. |

#### So sánh kiểm thử thủ công (Manual test) và kiểm thử tự động (Automation test)

Bảng 1.3. So sánh kiểm thử thủ công và kiểm thử tự động

|  |  |
| --- | --- |
| MANUAL TEST | AUTOMATION TEST |
| Được thực hiện bởi con người, không chính xác tại mọi thời điểm do đó nó ít tin cậy. | Được thực hiện bởi các công cụ hoặc các kịch bản. Đáng tin cậy hơn. |
| Tốn nhiều thời gian khi Regression Test. | Được thực hiện bởi các công cụ phần mềm, vì vậy nó nhanh hơn. |
| Cần đầu tư nguồn nhân lực nhiều. | Đầu tư cho công cụ kiểm thử. |
| Chỉ phù hợp cho các trường hợp thử nghiệm ít lặp đi lặp lại trong thời gian dài. | Thích hợp cho các trường hợp thử nghiệm được chạy liên tục trong một thời gian dài. |
| Phù hợp khi cần có sự quan sát của con người hoặc trải nghiệm của khách hàng. | Không đòi hỏi sự quan sát của con người, nhưng không thể đảm bảo thân thiện với người dùng trải nghiệm thực tế. |

## KẾT LUẬN CHƯƠNG

Chương 1 đã giới thiệu về đề tài "Tìm hiểu Robot Framework trong kiểm thử tự động Website". Trong chương này, tôi đã trình bày tổng quan về mục tiêu, ý nghĩa, và lý do chọn đề tài, cùng với cấu trúc và nội dung dự kiến của đề tài tốt nghiệp.

Tôi đã bắt đầu bằng việc giới thiệu về vấn đề kiểm thử phần mềm và tầm quan trọng của việc thực hiện kiểm thử tự động, đặc biệt trong việc kiểm thử các ứng dụng web.

Chương 1 cũng đã giới thiệu về cấu trúc tổng quan của đề tài, bao gồm các phần chính như Mục tiêu đề tài, Phạm vi nghiên cứu, Phương pháp nghiên cứu, Cấu trúc luận văn.

Trong những chương tiếp theo, chúng tôi sẽ đi sâu vào việc nghiên cứu và trình bày về Robot Framework, thư viện SeleniumLibrary và thư viện Browser, cách tạo môi trường kiểm thử tự động, cú pháp cơ bản của test case, cũng như cách sử dụng các keywords có sẵn. Ngoài ra, chúng tôi cũng sẽ trình bày về quá trình kiểm thử thủ công và cách tổ chức kiểm thử bằng cách tạo test suite và test case hierarchy.

Với những kiến thức và thông tin được trình bày trong chương 1, tôi hy vọng rằng bạn đọc đã có cái nhìn tổng quan về đề tài và sẵn sàng tiếp tục theo dõi các chương tiếp theo để hiểu rõ hơn về việc thực hiện kiểm thử tự động với Robot Framework trong môi trường kiểm thử phần mềm.

Chương 2 sẽ trình bày chi tiết về Robot Framework và thư viện FakerLibrary để bạn đọc có cái nhìn sâu hơn về công cụ và thư viện được sử dụng trong quá trình kiểm thử tự động.

# CHƯƠNG 2. ROBOT FRAMEWORK TRONG KIỂM THỬ TỰ ĐỘNG

## GIỚI THIỆU VỀ ROBOT FRAMEWORK

* Robot Framework là một công cụ kiểm thử tự động mã nguồn mở và mạnh mẽ, được phát triển nhằm giúp tự động hóa quy trình kiểm thử và kiểm tra chất lượng phần mềm. Đặc biệt, Robot Framework là một công cụ linh hoạt, hỗ trợ nhiều loại kiểm thử, bao gồm kiểm thử giao diện người dùng (UI), kiểm thử API và nhiều loại kiểm thử khác.

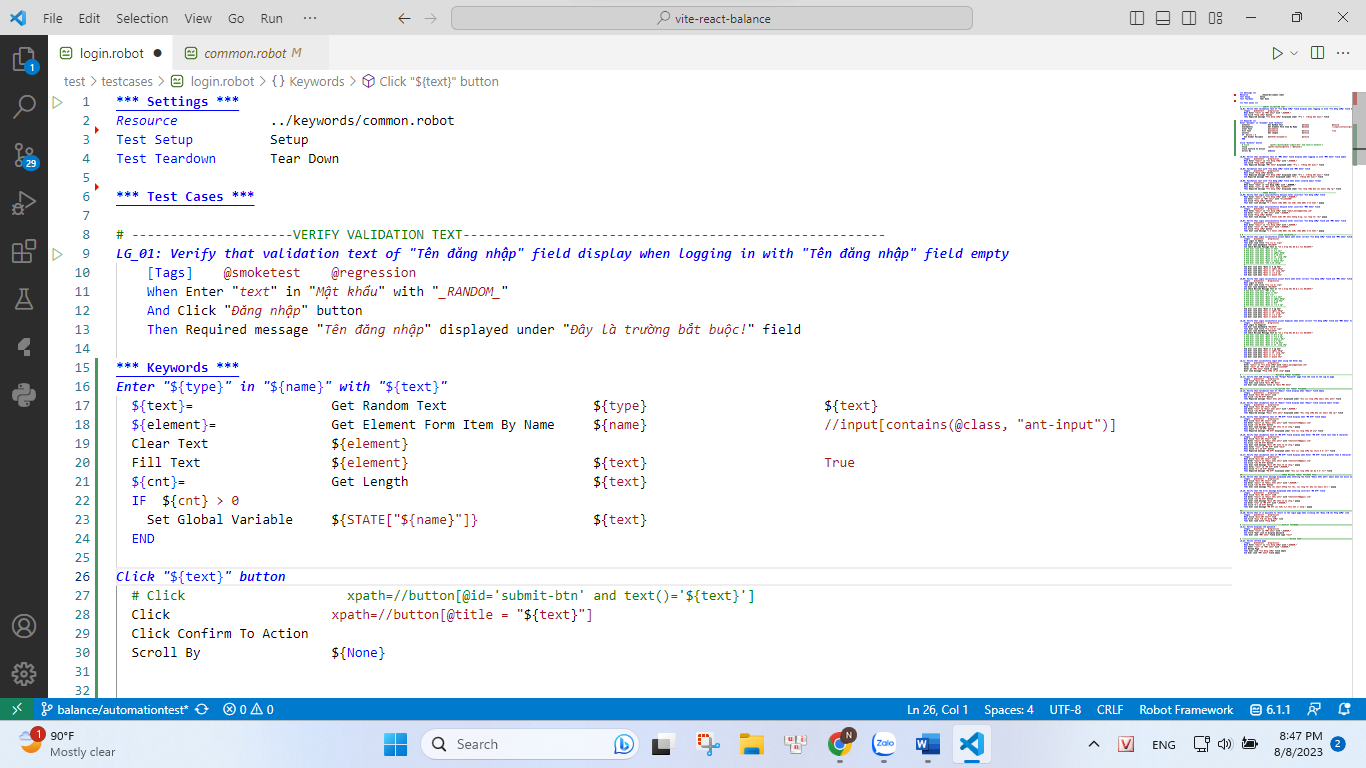


Hình 2.1. Logo Robotframework

* **Tính năng chính của Robot Framework:**
  + **Ngôn ngữ đơn giản và dễ đọc**: Robot Framework sử dụng cú pháp gần với ngôn ngữ tự nhiên, giúp người sử dụng dễ dàng viết và đọc mã kiểm thử.
  + **Tích hợp dễ dàng**: Robot Framework cho phép tích hợp với nhiều thư viện hỗ trợ kiểm thử khác nhau như Selenium, Appium, API testing libraries, và nhiều thư viện tự tạo khác.
  + **Cấu trúc tổ chức linh hoạt**: Test cases và test suites trong Robot Framework có thể được tổ chức thành cấu trúc phân cấp, giúp dễ dàng quản lý và chạy các kiểm thử khác nhau.
  + **Tùy chỉnh keywords**: Bạn có thể tạo ra các keywords tùy chỉnh để mô tả các hành động cụ thể trong quy trình kiểm thử và tái sử dụng chúng trong nhiều test cases khác nhau.

## CÁC THÀNH PHẦN CHÍNH CỦA ROBOT FRAMEWORK

* **Test Case:** Một test case trong Robot Framework là một chuỗi các bước kiểm thử, bao gồm cả việc chuẩn bị dữ liệu và kiểm tra kết quả. Test cases được viết bằng ngôn ngữ có cấu trúc rõ ràng, giúp người viết dễ dàng biểu diễn các bước kiểm thử một cách chi tiết.
* **Keyword:** Keywords là các hành động hoặc chức năng có sẵn hoặc tạo ra bởi người sử dụng để thực hiện các bước kiểm thử cụ thể. Các keywords có thể là các keywords có sẵn trong Robot Framework hoặc là các keywords tùy chỉnh được tạo ra bởi người dùng.

****

Hình 2.2. Ví dụ về 1 test case bằng robotframework

* **Test Suite:** Test suite là tập hợp các test cases có thể được tổ chức thành các nhóm logic. Test suites giúp bạn tổ chức và quản lý các test cases dễ dàng hơn, đồng thời cũng cho phép bạn thực hiện kiểm thử theo nhóm.
* **Thư viện (Library)**: Thư viện là tập hợp các keywords đã được định nghĩa sẵn để hỗ trợ kiểm thử trong các lĩnh vực cụ thể. Robot Framework cung cấp một số thư viện sẵn có như SeleniumLibrary (cho kiểm thử UI), RequestsLibrary (cho kiểm thử API), và nhiều thư viện khác.

Thông qua những thành phần chính này, Robot Framework mang đến một cách tiếp cận linh hoạt và mạnh mẽ cho việc thực hiện kiểm thử tự động và đóng góp vào quá trình tối ưu hóa chất lượng phần mềm.

## THƯ VIỆN DÙNG TRONG ROBOTFRAMEWORK

### Thư viện “SeleniumLibrary”

* Thư viện "SeleniumLibrary" là một trong những thư viện quan trọng trong Robot Framework, được sử dụng để kiểm thử giao diện người dùng (UI) của các ứng dụng web. Thư viện này kết nối trực tiếp với Selenium WebDriver, cho phép kiểm thử viên tương tác với trình duyệt web như Chrome, Firefox, Edge, và Safari.
* **Tính năng và chức năng chính của thư viện "SeleniumLibrary":**
  + **Tương tác với trình duyệt**: SeleniumLibrary cho phép bạn mở trình duyệt web, nhập dữ liệu, nhấn nút, và thực hiện nhiều hành động khác trực tiếp trên trình duyệt.
  + **Xác nhận và kiểm tra**: Bạn có thể kiểm tra các yếu tố trên trang web như văn bản, giá trị của các trường, sự hiện diện của yếu tố, và nhiều khía cạnh khác.
  + **Xử lý cửa sổ và tab**: Thư viện cho phép bạn mở và đóng cửa sổ hoặc tab trình duyệt, chuyển đổi giữa chúng và làm việc với nhiều cửa sổ/tab cùng lúc.

### Thư viện “Browser”

* Thư viện "Browser" là một thư viện mở rộng dựa trên "SeleniumLibrary", giúp tối ưu hóa việc kiểm thử các ứng dụng web bằng cách cung cấp các keywords linh hoạt hơn cho việc tương tác với trình duyệt và các yếu tố trong trang web.
* **Tính năng và chức năng chính của thư viện "Browser":**
  + **Quản lý trình duyệt**: Thư viện "Browser" cung cấp các keywords để mở, đóng và xác định trình duyệt mặc định hoặc nhiều trình duyệt cùng lúc.
  + **Xác định yếu tố**: Bạn có thể tìm kiếm và tương tác với các yếu tố trong trang web bằng cách sử dụng các keywords dễ đọc và gần gũi với ngôn ngữ tự nhiên.
  + **Xử lý popup và iframe**: Thư viện cho phép bạn xử lý các cửa sổ popup và iframe một cách dễ dàng, giúp kiểm thử giao diện người dùng một cách toàn diện hơn.

Việc sử dụng "SeleniumLibrary" và "Browser" không chỉ giúp tự động hóa kiểm thử UI một cách hiệu quả, mà còn tạo điều kiện thuận lợi cho việc quản lý và tương tác với các yếu tố trong trang web.

### Một số keywords được dùng phổ biến

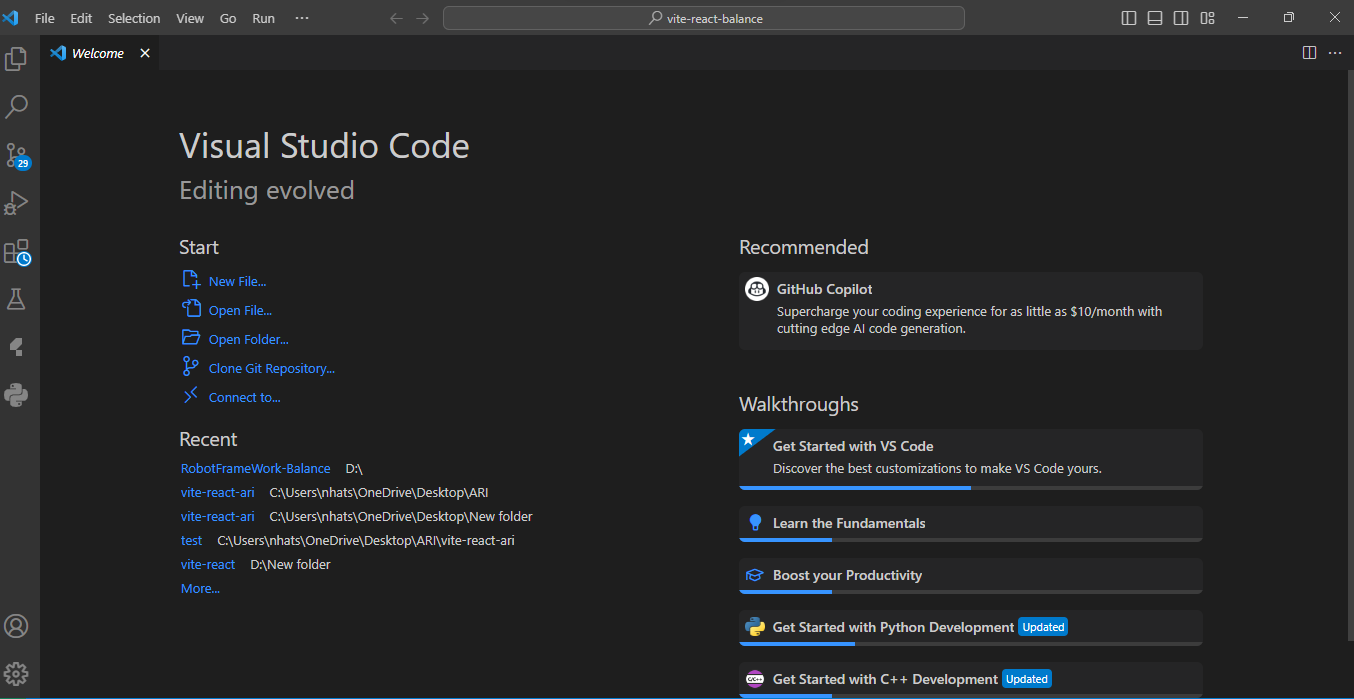
Bảng 2.1. Một số keywords được sử dụng phổ biến trong robotframework

|  |  |
| --- | --- |
| **Keyword** | **Mô tả** |
| **Keywords liên quan đến giao diện người dùng (UI)** | |
| Click Element | Nhấp vào một phần tử trên giao diện. |
| Input Text | Nhập dữ liệu vào trường văn bản. |
| Verify Text | Xác nhận văn bản trên giao diện. |
| **Keywords liên quan đến kiểm tra** | |
| Should Be Equal | So sánh hai giá trị để đảm bảo chúng bằng nhau. |
| Should Be True | Kiểm tra xem một biểu thức logic có đúng không. |
| Should Be False | Kiểm tra xem một biểu thức logic có sai không. |
| **Keywords liên quan đến dữ liệu** | |
| Set Variable | Thiết lập giá trị cho biến. |
| Append To List | Thêm giá trị vào danh sách. |
| **Keywords liên quan đến tệp tin và thư mục** | |
| File Should Exist | Kiểm tra xem một tệp tin có tồn tại không. |
| Create Directory | Tạo thư mục mới. |
| **Keywords liên quan đến kiểm tra ngoại lệ** | |
| Should Raise Error | Kiểm tra xem một hành động có ném ra ngoại lệ không. |
| **Keywords liên quan đến chuỗi văn bản** | |
| Should Contain | Kiểm tra xem một chuỗi có chứa một phần tử con không. |
| Should Not Contain | Kiểm tra xem một chuỗi có không chứa một phần tử con. |
| **Keywords liên quan đến thời gian và ngày tháng** | |
| Sleep | Tạm dừng thực thi trong một khoảng thời gian. |
| Get Current Date | Lấy ngày hiện tại. |
| **Keywords liên quan đến HTTP requests** | |
| GET Request | Gửi yêu cầu HTTP GET. |
| POST Request | Gửi yêu cầu HTTP POST. |
| **Keywords liên quan đến SQL database** | |
| Connect to Database | Kết nối đến cơ sở dữ liệu. |
| Query | Thực thi truy vấn SQL. |
| **Keywords liên quan đến vòng lặp và điều kiện** | |
| For Loop | Vòng lặp for. |
| Run Keyword If | Thực hiện keyword dựa trên điều kiện. |

Đây chỉ là một số keywords phổ biến trong Robot Framework. Mỗi thư viện hay framework cụ thể sẽ có những keywords riêng phù hợp với tác vụ cụ thể mà bạn đang thực hiện.

## GIỚI THIỆU PHẦN MỀM VISUAL STUDIO CODE

* **Visual Studio Code và vai trò của nó trong việc viết kiểm thử tự động:**
  + Visual Studio Code (VS Code) là một mã nguồn mở và môi trường phát triển tích hợp (IDE) nhẹ nhàng, mạnh mẽ được phát triển bởi Microsoft.
  + VS Code không chỉ hỗ trợ việc lập trình thông thường mà còn rất thích hợp cho việc viết kiểm thử tự động với Robot Framework.
* **Lợi ích của việc sử dụng VS Code trong việc viết kiểm thử tự động:**
  + **Phát triển hiệu quả:** VS Code cung cấp môi trường làm việc trực quan, hỗ trợ sáng tạo và tập trung vào quy trình phát triển và kiểm thử.
  + **Hỗ trợ đa ngôn ngữ:** VS Code hỗ trợ nhiều ngôn ngữ lập trình, bao gồm cả Python và Robot Framework, giúp bạn viết mã dễ dàng hơn và tối ưu hóa quy trình viết kiểm thử tự động.
  + **Mở rộng bằng các tiện ích:** Bạn có thể mở rộng khả năng của VS Code bằng cách cài đặt các tiện ích (extensions) liên quan đến Robot Framework, chẳng hạn như hỗ trợ cú pháp, kiểm tra lỗi chính tả, hoặc tích hợp với hệ thống kiểm thử liên tục (CI/CD).



Hình 2.3. Giao diện Visual Studio Code

## KẾT LUẬN CHƯƠNG

Chương 2 đã tập trung vào việc tìm hiểu về Robot Framework và thư viện hỗ trợ cho Robot Framework, đây là hai thành phần chính trong quá trình kiểm thử tự động ứng dụng web. Tôi đã đi sâu vào các khía cạnh cơ bản và quan trọng của hai thành phần này, giúp bạn đọc có cái nhìn rõ ràng hơn về cách sử dụng chúng để thực hiện kiểm thử hiệu quả.

Tôi đã giới thiệu về Robot Framework, một công cụ kiểm thử tự động mã nguồn mở, linh hoạt, và dễ sử dụng. Tôi đã đề cập đến cách cài đặt và cấu hình Robot Framework, cùng với cách cơ bản để tạo và chạy các test case. Chúng tôi cũng đã trình bày về cấu trúc của một test case, bao gồm khai báo biến, sử dụng keywords, và kiểm tra kết quả. Với cấu trúc đơn giản và dễ đọc, Robot Framework cho phép các nhà phát triển và kiểm thử viên dễ dàng triển khai kiểm thử tự động mà không cần có kiến thức sâu về lập trình.

Thư viện Browser đã được giới thiệu như một công cụ hỗ trợ mạnh mẽ để thực hiện kiểm thử tự động các ứng dụng web. Tôi đã trình bày về cách sử dụng Browser để thao tác với trình duyệt, tương tác với các yếu tố trên giao diện, và kiểm tra các điều kiện. Chúng tôi đã trình bày ví dụ về việc sử dụng các keywords của thư viện, giúp bạn đọc thấy rõ sự tiện lợi và hiệu quả của Browser trong việc kiểm thử tự động.

Chương 2 đã cung cấp một cái nhìn tổng quan về khả năng và tiềm năng của Robot Framework và Browser trong việc thực hiện kiểm thử tự động. Những kiến thức được trình bày trong chương này sẽ là nền tảng cho việc thực hiện các phần kiểm thử cụ thể trong các chương tiếp theo. Tôi hy vọng rằng bạn đọc đã có cái nhìn rõ ràng và sẵn sàng tiếp tục học hỏi về việc áp dụng Robot Framework và Browser trong thực tế kiểm thử phần mềm.

Chương 3 sẽ tiếp tục trình bày về cấu hình môi trường kiểm thử tự động và cách viết các test case cơ bản bằng Robot Framework.

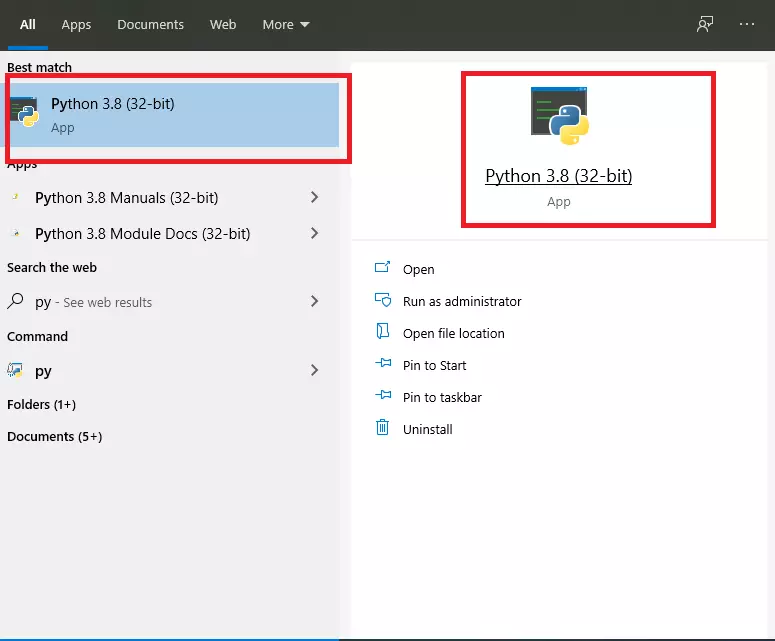
# CHƯƠNG 3. XÂY DỰNG MÔI TRƯỜNG KIỂM THỬ VÀ CÚ PHÁP VIẾT TEST CASES

## CÀI ĐẶT PYTHON VÀ PIP

* **Giới thiệu về Python và vai trò của nó trong phát triển và kiểm thử:**
  + Python là một ngôn ngữ lập trình đa năng, mạnh mẽ và dễ học, được sử dụng rộng rãi trong lĩnh vực phát triển phần mềm và kiểm thử.
  + Python cung cấp cú pháp đơn giản, hỗ trợ nhiều thư viện mạnh mẽ và có khả năng tích hợp dễ dàng với các công cụ khác.
  + Với khả năng này, Python đã trở thành một trong những ngôn ngữ phổ biến trong việc thực hiện kiểm thử tự động.
* **Cài đặt Python:** Trước hết, vì Robot Framework là một nền tảng kiểm thử dựa trên nền tảng Python, nên trước tiên cần cài đặt Python (nên cài Python 2.5 hoặc mới hơn – khuyến cáo cài đặt Python 2.7).

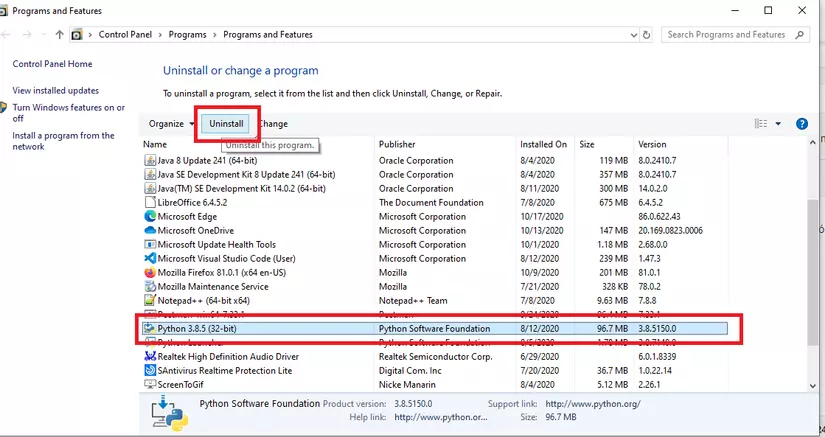
**Bước 1.** Kiểm tra xem trong máy tính của bạn đã cài đặt Python hay chưa?

Bạn hãy nhấp vào biểu tượng kính lúp/vòng tròn cạnh biểu tượng Windown, hoặc nhấn ⊞ Win+S. Nếu tìm thấy kết quả Python thì chứng tỏ là máy của bạn đã sẵn sàng cài đặt Python rồi đó.

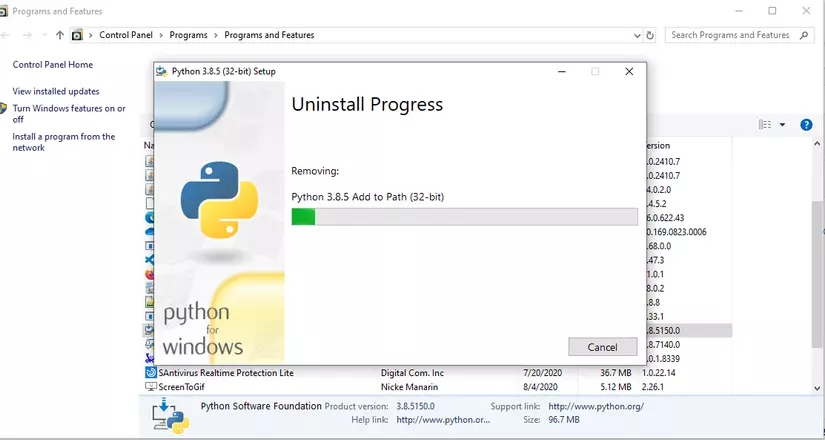


Hình 3.1. Kiểm tra xem Python đã cài hay chưa

**Bước 2.** Gỡ cài đặt và cài lại



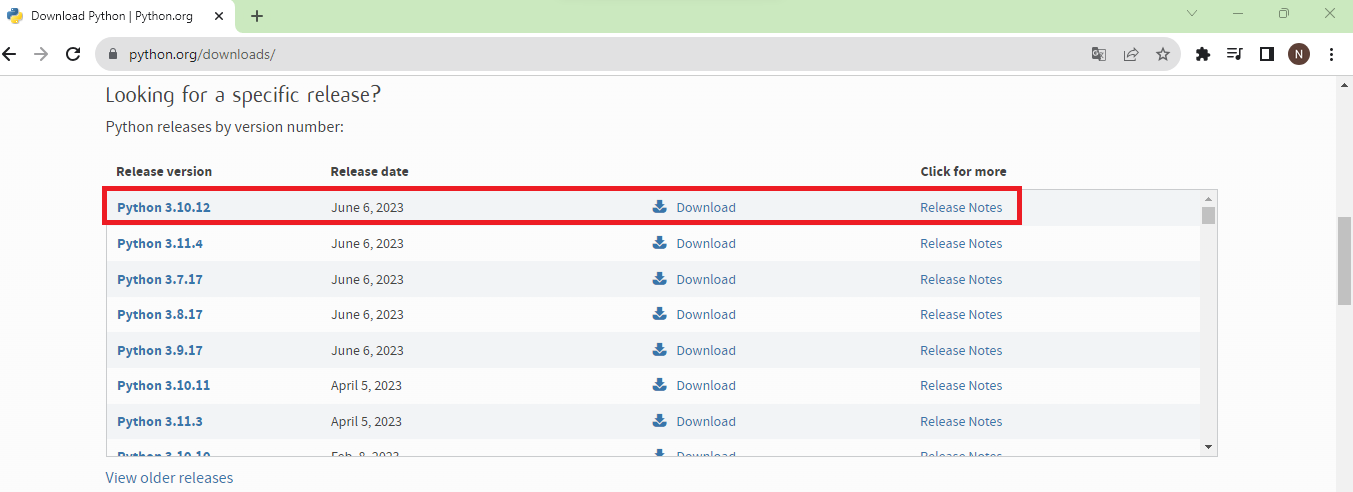
Hình 3.2. Chọn Python 3.8.5 để gỡ cài đặt



Hình 3.3. Quá trình gỡ cài đặt Python bắt đầu

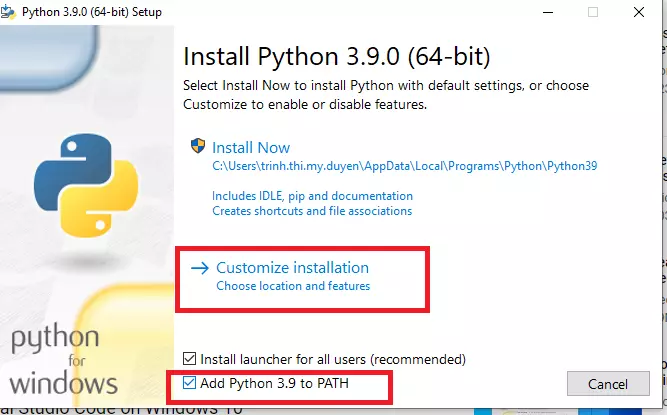
**Bước 3.** Cài đặt lại

Mở trình duyệt, truy cập vào website có đường dẫn sau để tải phần mềm python: <https://www.python.org/>

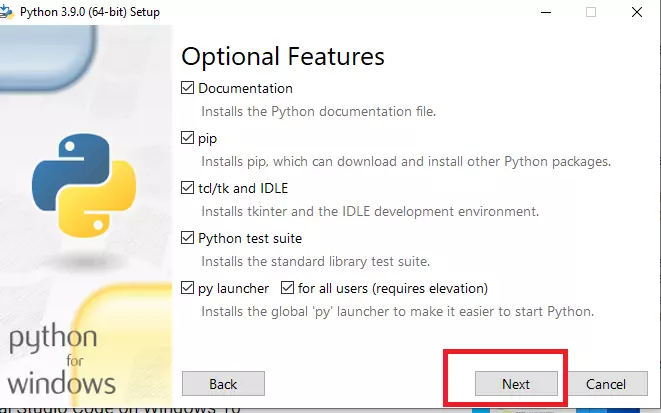


Hình 3.4. Download phiên bản Python cần cài đặt

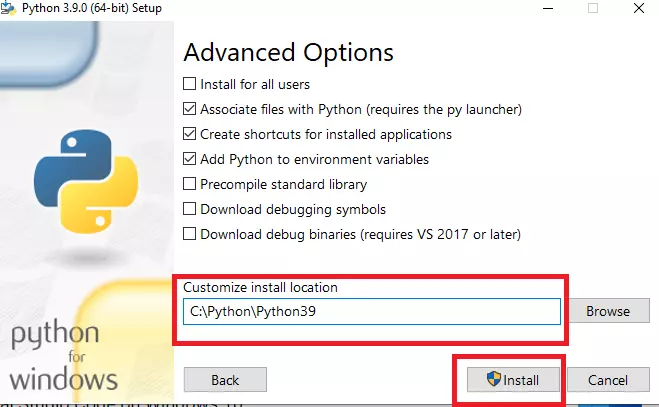
Bắt đầu cài đặt



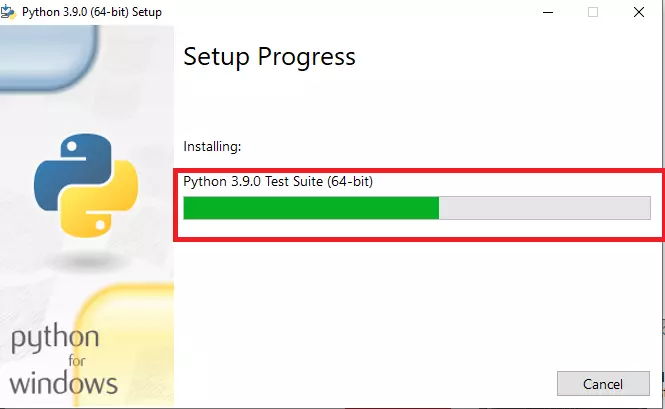
Hình 3.5. Chạy phần mềm python vừa tải về



Hình 3.6. Chọn “Next” để tiếp tục cài đặt

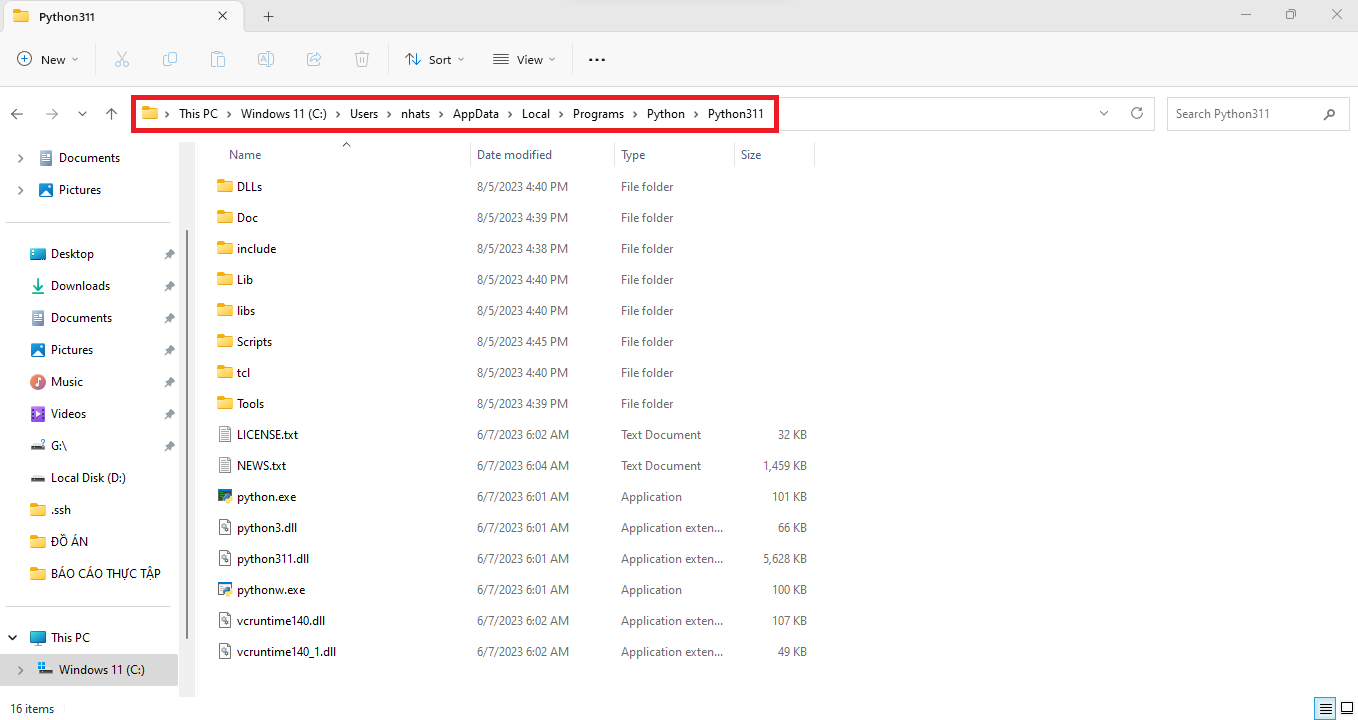


Hình 3.7. Chọn đường dẫn lưu file cài đặt Python và chọn “Install”

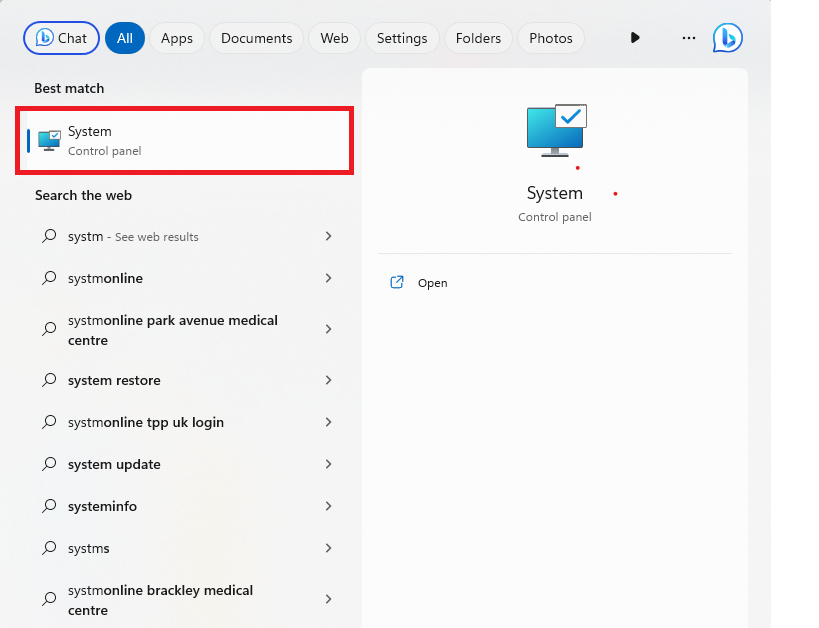


Hình 3.8. Bắt đầu cài đặt python

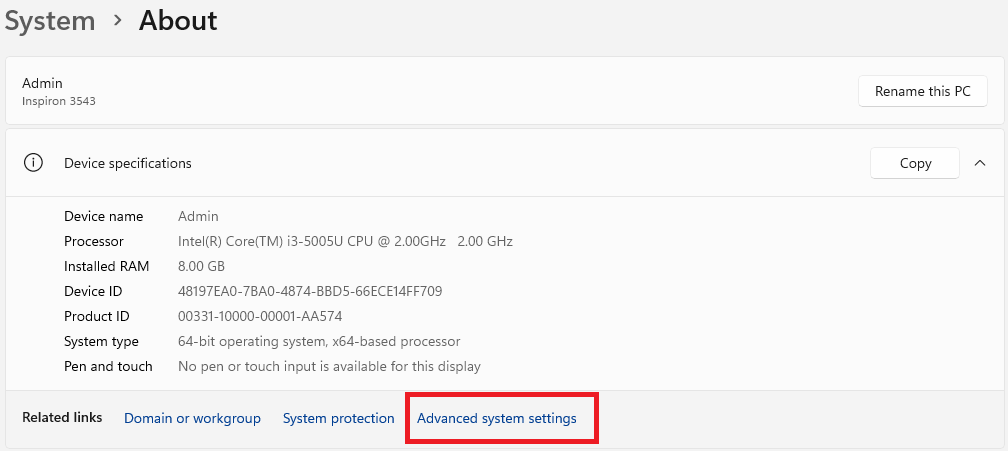
**Bước 4.** Sau khi Python được cài đặt, tiến hành Setup đường dẫn



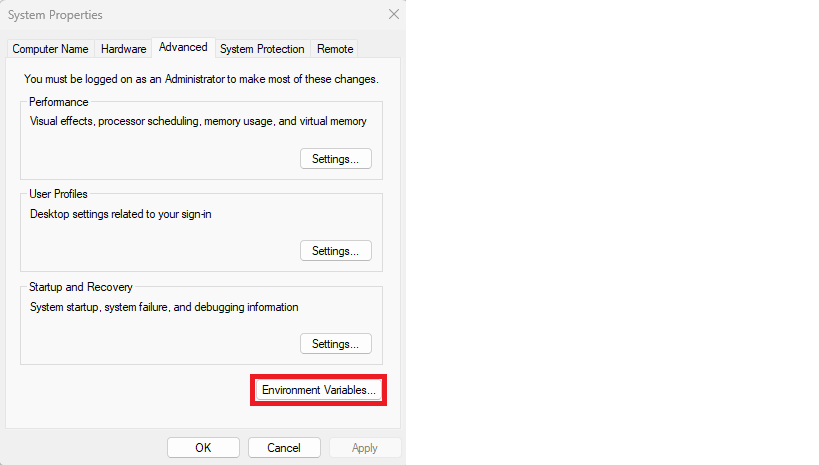
Hình 3.9. Copy đường dẫn thư mục Python311



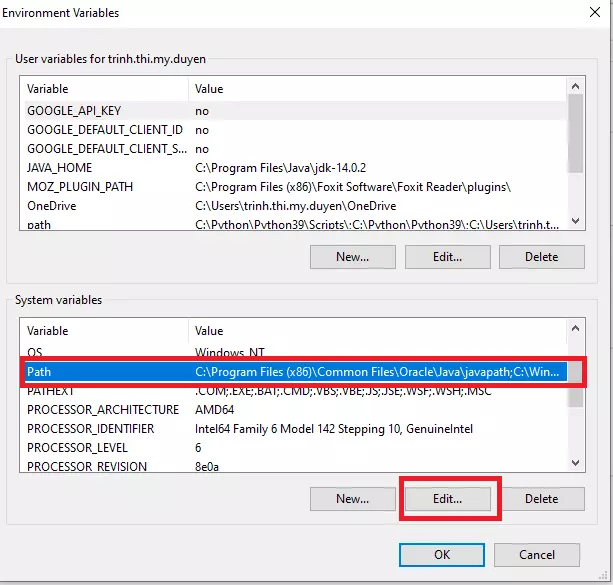
Hình 3.10. Mở “System”



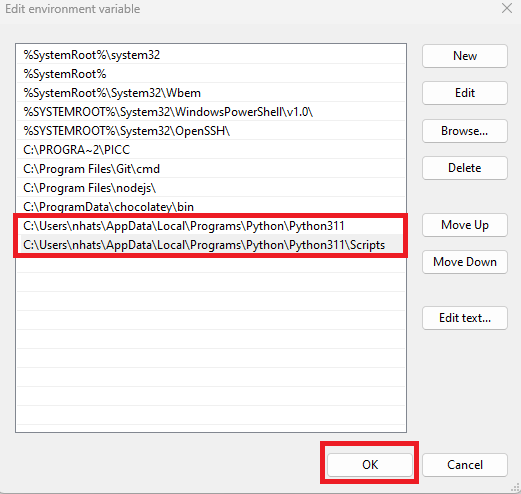
Hình 3.11. Chọn “Advanced system settings”



Hình 3.12. Chọn “Environment Variables…”

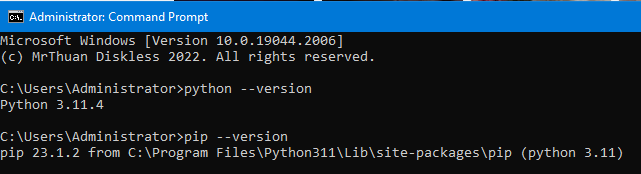


Hình 3.13. Chỉnh sửa biến môi trường path

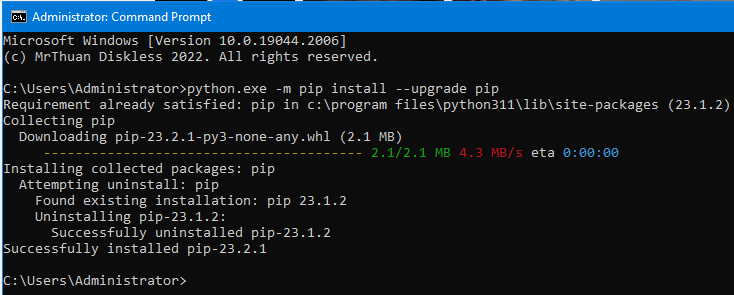


Hình 3.14. Thêm đường dẫn chưa file cài đặt Python vào biến môi trường

**Bước 4.** Kiểm tra xem Python và Pip cài đặt xong chưa



Hình 3.15. Kiểm tra phiên bản Python và pip đã cài đặt

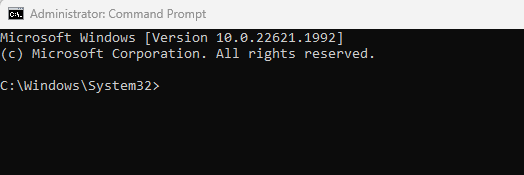


Hình 3.16. Update phiên bản pip

## CÀI ĐẶT ROBOT FRAMEWORK VÀ CÁC THƯ VIỆN LIÊN QUAN

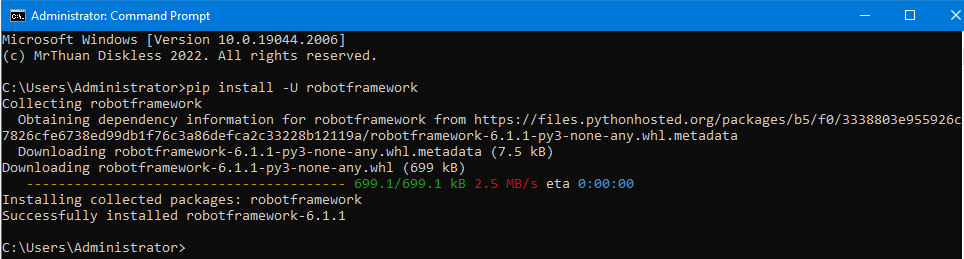
* **Giới thiệu về cách cài đặt Robot Framework và các bước cần thiết:**
  + Robot Framework là một công cụ kiểm thử tự động mạnh mẽ và linh hoạt.
  + Để sử dụng Robot Framework, bạn cần cài đặt nó cùng với các thư viện hỗ trợ.
  + Trong phần này, chúng ta sẽ cài đặt Robot Framework cùng với hai thư viện quan trọng là "FakerLibrary" và "Browser".
* **Cài đặt Robot Framework và thư viện "FakerLibrary":**

**Bước 1.** Mở Command Prompt hoặc Terminal.



Hình 3.17. Giao diện Command Prompt

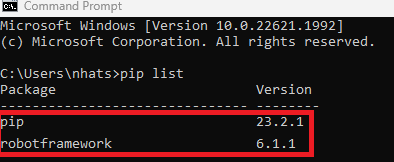
**Bước 2.** Sử dụng pip để cài đặt Robot Framework bằng cách chạy lệnh: “pip install -U robotframework”.



Hình 3.18. Cài đặt thành công Robotframework

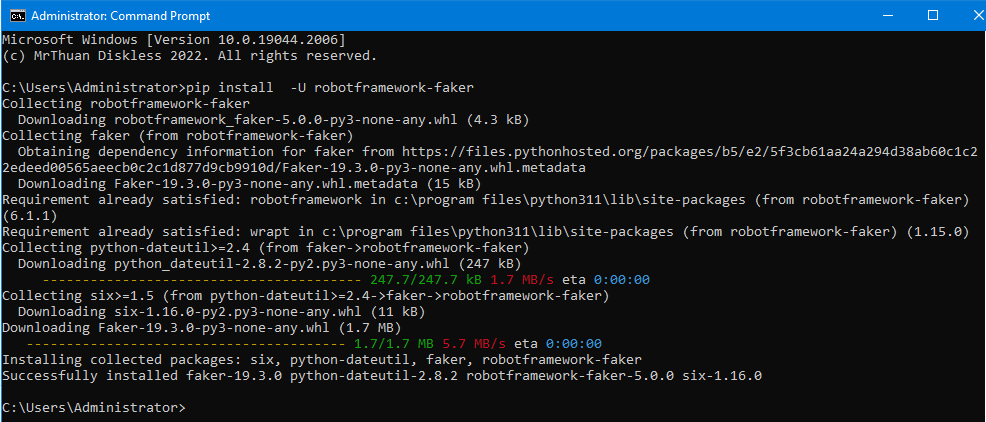
Bước 3. Kiểm tra xem robotframework đã được cài đặt đúng chưa bằng câu lệnh:

“pip list”

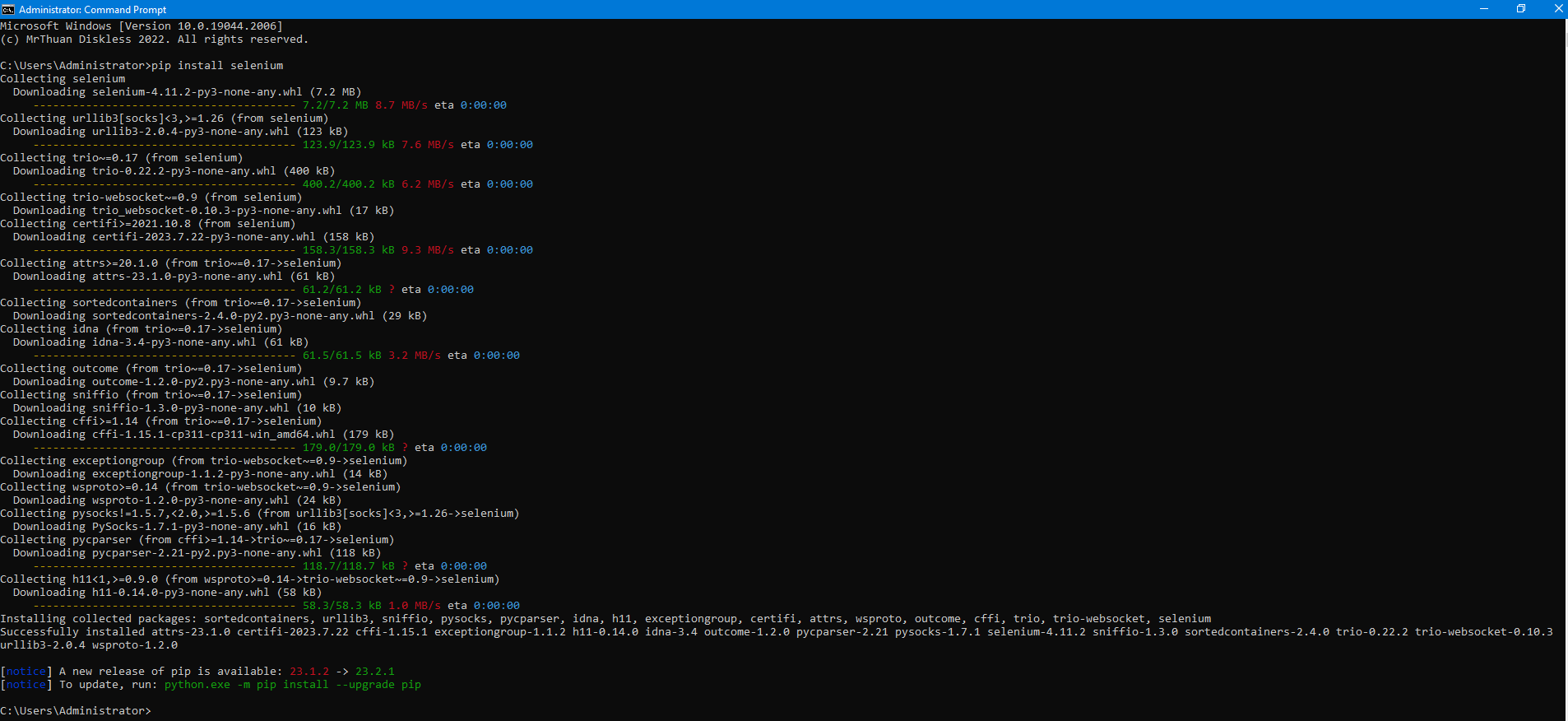


Hình 3.19. Kiểm tra xem phiên bản robotframework đã cài đặt

Bước 4. Sau khi cài đặt xong Robot Framework, bạn cần cài đặt thư viện "FakerLibrary" để làm việc với kiểm thử giao diện người dùng (UI) của các trang web. Sử dụng pip để cài đặt thư viện này bằng lệnh: “pip install -U robotframework-faker”.

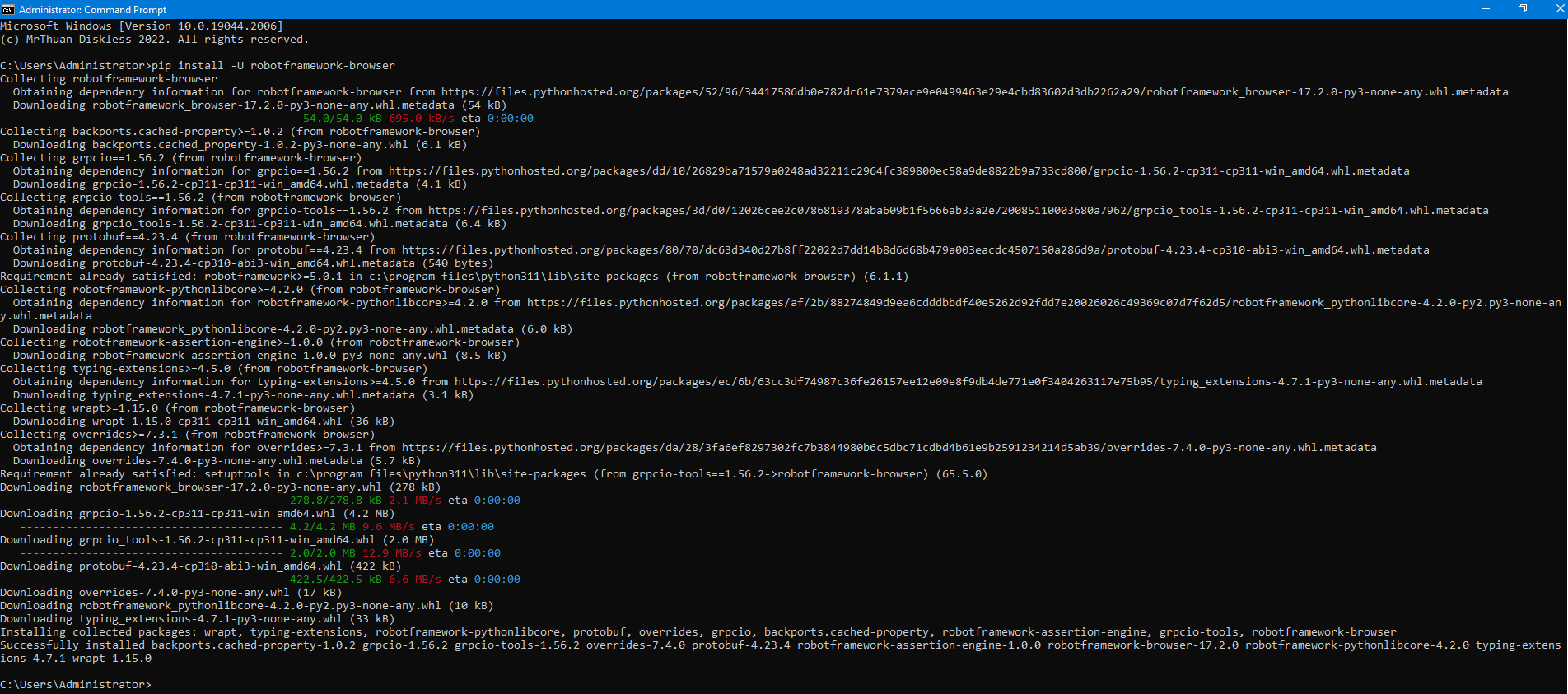


Hình 3.20. Cài đặt thành công thư viện FakerLibrary



Hình 3.21. Cài đặt thành công thư viện Selenium

* **Cài đặt thư viện "Browser" cho việc kiểm thử trực tiếp trình duyệt:** Thư viện "Browser" là một thư viện mở rộng dựa trên "SeleniumLibrary", cung cấp các keywords linh hoạt hơn để tương tác với trình duyệt và các yếu tố trong trang web. Để cài đặt thư viện "Browser", sử dụng pip bằng lệnh: “pip install -U robotframework-browser”.



Hình 3.22. Cài đặt thành công thư viện Browser

## CÚ PHÁP CƠ BẢN CỦA TEST CASE

* **Định nghĩa cấu trúc cơ bản của một test case trong Robot Framework:**
  + Trong Robot Framework, mỗi test case được viết dưới dạng một tệp .robot riêng biệt. Một test case chứa các thông tin cơ bản như tên, mô tả, thẻ, và các bước kiểm thử cụ thể. Dưới đây là cấu trúc cơ bản của một test case:

|  |
| --- |
| **\*\*\* Test Cases \*\*\***  **Tên Test Case**  ***[Documentation]* Mô tả về mục tiêu của test case**  **[Tags] Thẻ liên quan đến test case (tùy chọn)**  **Bước 1**  **Bước 2**  **...** |

* + **Cú pháp chi tiết của một bước kiểm thử:** Mỗi bước kiểm thử trong test case sẽ được biểu diễn bởi một keyword theo sau bởi các tham số. Cấu trúc chi tiết của một bước kiểm thử như sau:

Tên Keyword Tham số 1 Tham số 2 ...

* + **Các keywords có sẵn trong Robot Framework:** Các keywords có sẵn trong Robot Framework là những khối xây dựng sẵn cho các thao tác kiểm thử. Các keywords này được cung cấp bởi thư viện "SeleniumLibrary" và thư viện "Browser" để giúp bạn tương tác với trình duyệt và các yếu tố trong trang web. Có hàng trăm keywords có sẵn cho các thao tác khác nhau như tìm kiếm, nhập liệu, nhấn nút, và kiểm tra kết quả.
  + **Sử dụng các keywords liên quan đến trình duyệt:** Các keywords có sẵn liên quan đến trình duyệt cho phép bạn thực hiện các thao tác trên trình duyệt như mở trang web, nhập liệu, click, và kiểm tra kết quả. Phần này đã được giới thiệu ở mục 2.3.3 của chương 2.
  + **Ví dụ thực tế về việc sử dụng cú pháp trong test case:**



* + **Mô tả các phần cú pháp:**

**Tên Test Case:** Tên của test case, thể hiện mục đích và chức năng cần kiểm thử.

**[Documentation]**: Mô tả chi tiết về mục đích của test case, giúp người đọc hiểu rõ hơn về mục tiêu kiểm thử.

**[Tags]**: Các thẻ liên quan đến test case, giúp phân loại và nhận biết test case thuộc nhóm nào (ví dụ: UI, API).

**Tên Keyword:** Tên của keyword được sử dụng để thực hiện bước kiểm thử cụ thể.

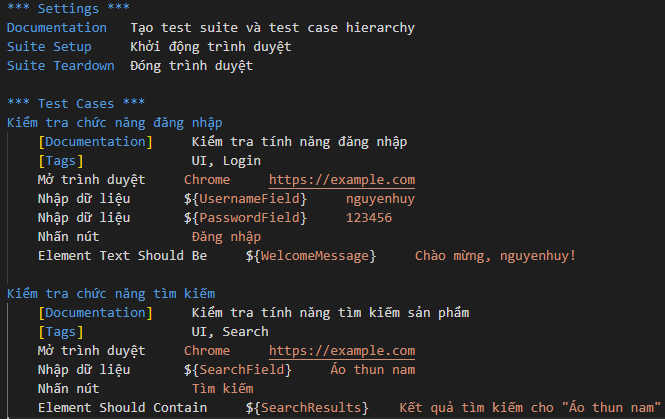
**Tham số:** Các giá trị hoặc dữ liệu cần thiết để thực hiện bước kiểm thử.

**[Tham số tùy chọn]:** Các tham số có thể đi kèm với keyword để tùy chỉnh hành động.

* + **Lợi ích của cú pháp cơ bản trong test case:**
    - ***Rõ ràng và dễ đọc:*** Cú pháp cơ bản giúp mã kiểm thử trở nên dễ đọc, dễ hiểu và dễ theo dõi.
    - ***Tổ chức logic:*** Với cú pháp cơ bản, bạn có thể tổ chức logic kiểm thử thành các bước rõ ràng và có cấu trúc.
    - ***Dễ bảo trì:*** Cấu trúc chuẩn giúp bạn dễ dàng bổ sung, sửa đổi và duyệt test case trong tương lai.

## TẠO TEST SUITE VÀ TEST CASE HIERACHY

* **Tổ chức kiểm thử với test suite và test case hierarchy:** 
  + Trong quá trình thực hiện kiểm thử tự động bằng Robot Framework, việc tổ chức test suite và test case hierarchy là một phần quan trọng giúp bạn tổ chức, quản lý và thực hiện kiểm thử một cách hiệu quả. Test suite là một tập hợp các test case có liên quan đến nhau, còn test case hierarchy là cách bạn tổ chức các test suite vào các mức khác nhau, tạo nên một cấu trúc kiểm thử có hệ thống.
* **Tạo test suite và test case hierarchy trong Robot Framework:**
  + ***Test Suite:*** Một test suite thường tương ứng với một chức năng hoặc một phần trong ứng dụng. Nó có thể chứa một hoặc nhiều test case liên quan đến chức năng đó. Test suite giúp bạn tổ chức các test case có cùng mục tiêu kiểm thử vào cùng một nhóm.
  + ***Test Case Hierarchy:*** Robot Framework cho phép bạn xây dựng test case hierarchy bằng cách tổ chức test suite vào các mức khác nhau. Điều này giúp bạn kiểm thử từng mức độ khác nhau của ứng dụng, từ kiểm thử chức năng cơ bản đến kiểm thử hệ thống tổng thể. Test case hierarchy giúp bạn phân chia và tập trung vào từng khía cạnh cụ thể của ứng dụng.
  + **Ví dụ về việc sử dụng test suite và test case hierarchy:**



Hình 3.23. Ví dụ về việc sử dụng test suite và test case hierarchy

* Lợi ích của test suite và test case hierarchy:
  + Tổ chức rõ ràng: Việc sử dụng test suite và test case hierarchy giúp bạn tổ chức kiểm thử một cách có cấu trúc, dễ quản lý và tìm kiếm khi cần.
  + Tái sử dụng dễ dàng: Test suite và test case hierarchy giúp bạn dễ dàng tái sử dụng các test case trong các dự án khác nhau, giảm thiểu công việc viết lại mã kiểm thử.
  + Kiểm thử từng mức độ: Với test case hierarchy, bạn có thể kiểm thử từng mức độ khác nhau của ứng dụng, từ kiểm thử chức năng cơ bản đến kiểm thử hệ thống tổng thể, giúp đảm bảo tính ổn định và chất lượng của ứng dụng.

## KẾT LUẬN CHƯƠNG

Chương 3 đã tập trung vào việc xây dựng môi trường kiểm thử tự động và viết các test case cơ bản bằng Robot Framework. Chương này đã giới thiệu về quá trình cài đặt và chuẩn bị môi trường để thực hiện kiểm thử tự động, bao gồm việc cài đặt Python, Robot Framework, thư viện Browser và cấu hình môi trường để thao tác với trình duyệt.

Chương này đã hướng dẫn cách cài đặt các thành phần cần thiết để bắt đầu thực hiện kiểm thử tự động bằng Robot Framework. Bằng cách cài đặt Robot Framework thông qua pip, bạn có thể dễ dàng tiến hành các bước chuẩn bị môi trường.

Việc viết test case cơ bản bằng Robot Framework đã được trình bày chi tiết. Chương này đã hướng dẫn cách viết các bước kiểm thử, bao gồm việc mở trình duyệt, nhập liệu, thao tác trên giao diện, và kiểm tra kết quả. Với cú pháp dễ đọc và gọn gàng của Robot Framework, việc viết test case trở nên đơn giản và hiệu quả.

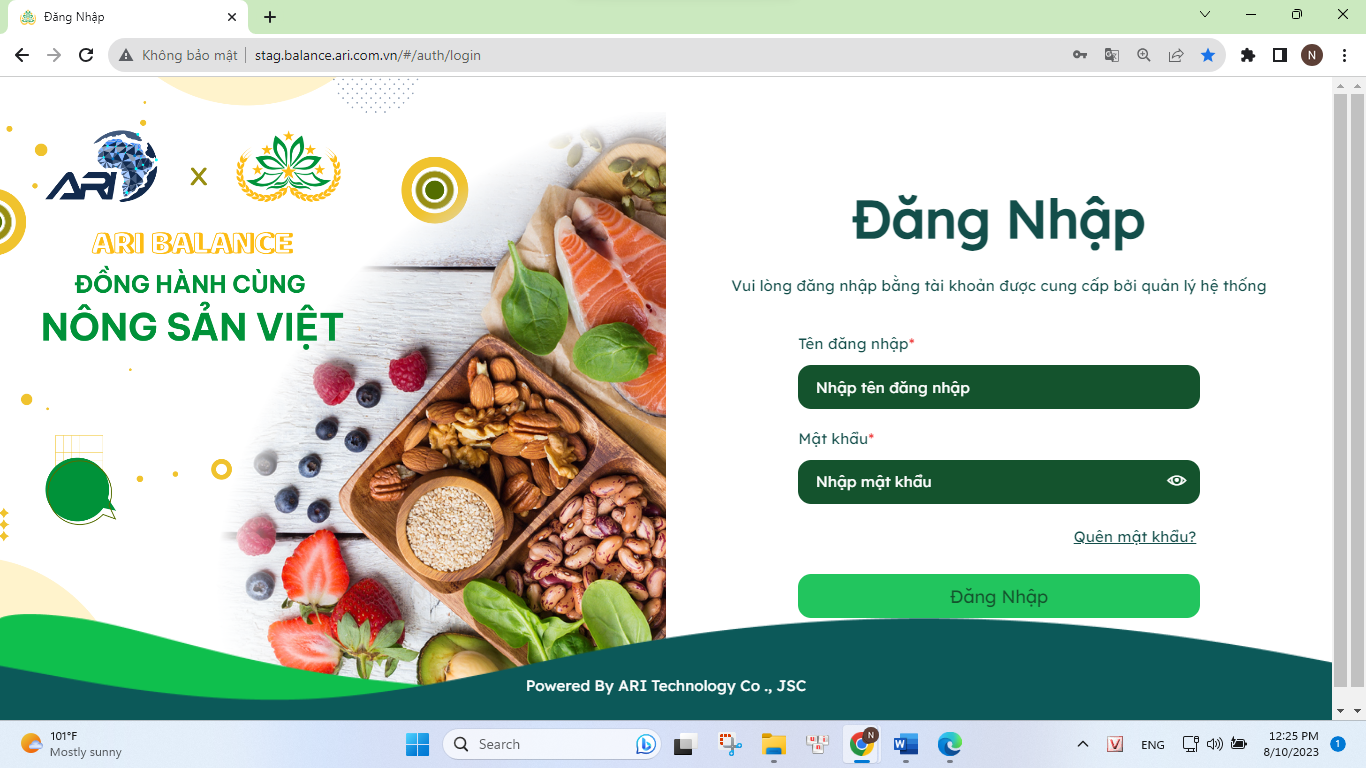
Chương 3 đã giúp bạn đọc nắm vững cách cấu hình môi trường kiểm thử tự động và viết các test case cơ bản. Những kiến thức được trình bày trong chương này là nền tảng để bạn có thể thực hiện kiểm thử tự động một cách hiệu quả và chất lượng. Qua chương này, tôi hy vọng rằng bạn đọc đã có cái nhìn sâu hơn về quá trình chuẩn bị và viết test case, và sẵn sàng áp dụng chúng trong việc kiểm thử thực tế.

Chương 4 sẽ tiếp tục trình bày về việc xây dựng và thực hiện kiểm thử tự động website.

# CHƯƠNG 4. XÂY DỰNG VÀ THỰC HIỆN KIỂM THỬ TỰ ĐỘNG WEBSITE

## GIỚI THIỆU VỀ WEBSITE MẪU

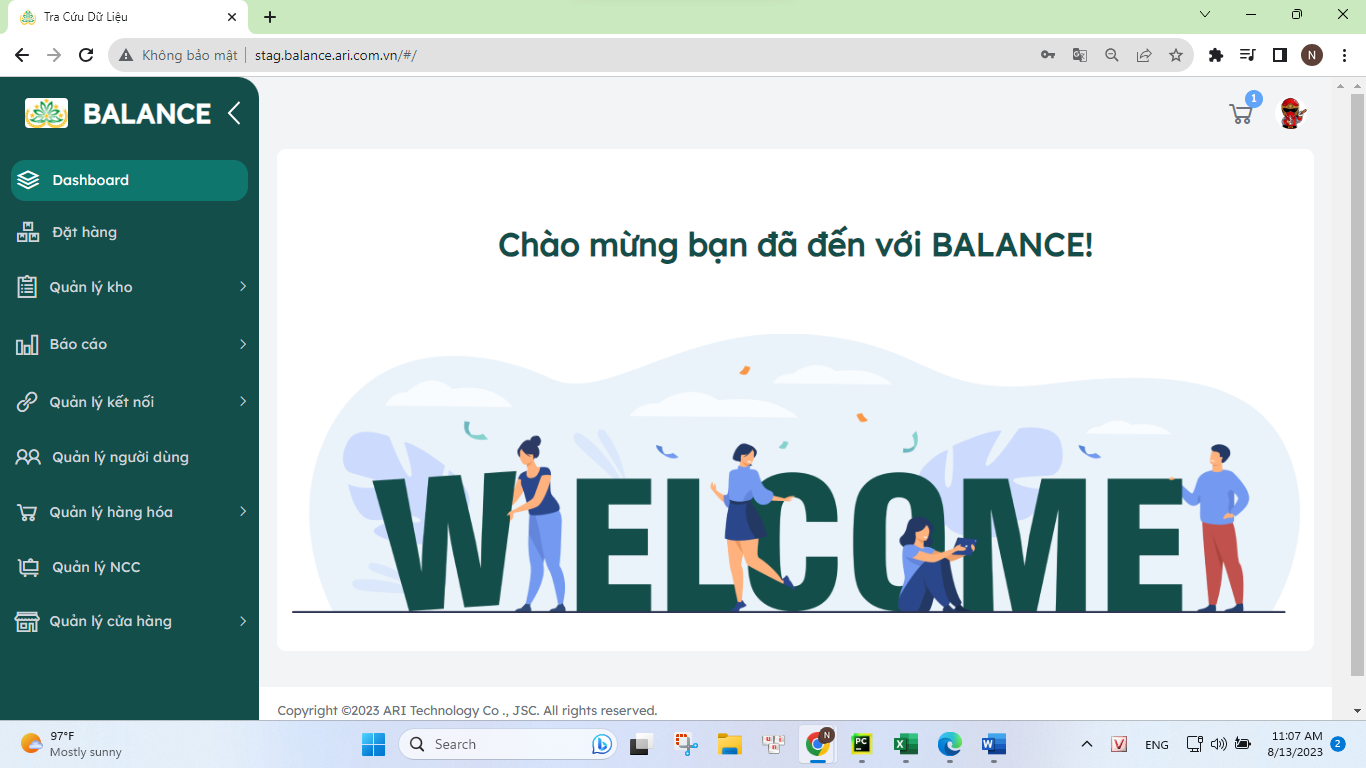
Trong phần này, tôi sẽ giới thiệu về website mẫu mà tôi sẽ sử dụng để thực hiện kiểm thử tự động bằng Robot Framework. Website mẫu này sẽ đóng vai trò là một ứng dụng web cơ bản để chúng tôi thực hiện các bài kiểm thử và trình bày cách sử dụng Robot Framework để kiểm thử tự động.

****

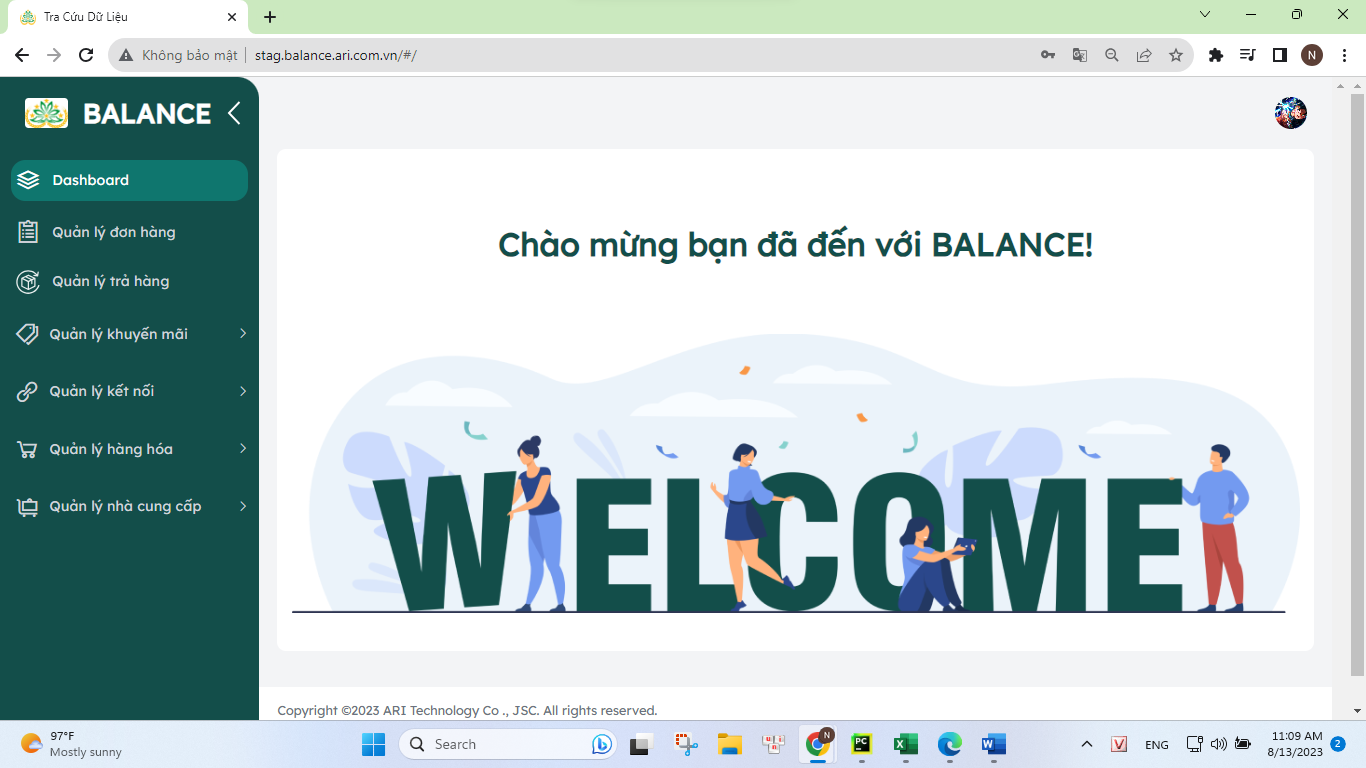
Hình 4.1. Giao diện Website mẫu trang đăng nhập



Hình 4.2. Giao diện và menu trang chủ tài khoản Admin



Hình 4.3. Giao diện và menu trang chủ tài khoản Store



Hình 4.4. Giao diện và menu trang chủ tài khoản Supplier

* **Mục tiêu của website mẫu:**
  + ***Mô phỏng ứng dụng web cơ bản***: Website mẫu được thiết kế để mô phỏng một ứng dụng web cơ bản với các tính năng cơ bản như đăng nhập, tìm kiếm sản phẩm, và hiển thị thông tin.
  + ***Dễ dàng sử dụng***: Giao diện của website mẫu được thiết kế đơn giản và dễ sử dụng, giúp người dùng nhanh chóng làm quen và tương tác với các tính năng.
  + ***Đa dạng chức năng***: Website mẫu cung cấp các chức năng chính như đăng nhập, tìm kiếm sản phẩm, và hiển thị thông tin sản phẩm, cho phép chúng tôi thực hiện các kiểm thử cơ bản và nâng cao.
* **Cấu trúc và tính năng của website mẫu:**
  + ***Trang chủ***: Trang chủ hiển thị thông tin tổng quan về ứng dụng và các tính năng chính khi đăng nhập thành công. Ứng với mỗi tài khoản khác nhau, sẽ có giao diện trang chủ với các mục menu khác nhau.
  + ***Đăng nhập***: Trang đăng nhập cho phép người dùng nhập tên người dùng và mật khẩu để đăng nhập vào hệ thống.
  + ***Quên mật khẩu***: Ngoài ra, website mẫu cũng có chức năng cho phép người dùng lấy lại mật khẩu.
  + ***Các tính năng của website mẫu***:
    - Đăng nhập (Sign In).
    - Admin tạo mới người dùng (Admin – Create User).
    - Admin tạo mới của hàng chính (Admin – Create Main Store).
    - Admin tạo mới của hàng chi nhánh (Admin – Create Branch Store).
    - Admin tạo mới nhà cung cấp (Admin – Create Supplier).
    - Cập nhật thông tin cá nhân (Profile).
    - Supplier tạo mới sản phẩm (Supplier – Create Product).
    - Kết nối cửa hàng với nhà cung cấp (Store & Supplier connect).
    - Store – Đặt hàng (Store – Order flow).
    - Quản lý doanh thu (Revenue).
    - Tạo mới danh mục (Admin – Create Category).
    - Quản lý đơn hàng (Order management).
    - Quản lý hàng khuyến mãi (Promtional goods).
    - Tạo mới thuế (Create Tax).
    - Hủy đơn hàng (Create Product Disposal).
    - Lợi nhuận (Good return).
    - Tạo mới nhà cung cấp Non-BAL (Create Non-BAL Supplier).
    - Tạo mới sản phẩm Non-BAL (Create Non-BAL Product).
    - Nhà cung cấp tạo mới sản phẩm (Create Product – Supplier).
* **Mục tiêu kiểm thử cho website mẫu:** kiểm thử toàn bộ chức năng đã liệt kê ở trên.

## XÂY DỰNG KỊCH BẢN KIỂM THỬ

### Kịch bản kiểm thử cho chức năng đăng nhập

Bảng 4.1. Kịch bản kiểm thử cho chức năng đăng nhập

|  |  |  |
| --- | --- | --- |
| Testcase ID | Yêu cầu test | Yêu cầu kết quả |
| Validation Text | | |
| LG – 01 | Kiểm tra Validation text khi bỏ trống “Tên đăng nhập”. | Hiển thị Validation Text trường “Tên đăng nhập”: “***This is a required field!***”. |
| LG – 02 | Kiểm tra Validation text khi bỏ trống “Mật khẩu”. | Hiển thị Validation Text trường “Mật khẩu”: “***This is a required field!***”. |
| LG – 03 | Kiểm tra Validation text khi bỏ trống cả “Tên đăng nhập” và “Mật khẩu”. | Hiển thị Validation Text trường “Tên đăng nhập” và trường “Mật khẩu”: “***This is a required field!***”. |
| LG – 04 | Kiểm tra Validation text khi nhập “Tên đăng nhập” sai định dạng email. | Hiển thị Validation Text trường “Tên đăng nhập”: “***Please enter a valid email address!***”. |
| Error Message | | |
| LG – 05 | Kiểm tra Error Message khi nhập sai “Tên đăng nhập”. | Hiển thị Error Message: “***Tài khoản chưa được tạo hoặc chưa được kích hoạt.***”. |
| LG – 06 | Kiểm tra Error Message khi nhập sai “Mật khẩu”. | Hiển thị Error Message: “***Tài khoản hoặc mật khẩu không đúng, vui lòng thử lại***”. |
| LG – 07 | Kiểm tra Error Message khi nhập sai cả “Tên đăng nhập” và “Mật khẩu”. | Hiển thị Error Message: “***Tài khoản chưa được tạo hoặc chưa được kích hoạt.***”. |
| Sign in Successfully | | |
| LG – 08 | Kiểm tra đăng nhập thành công tài khoản Admin. | Hiển thị title: “Tra Cứu Dữ Liệu”.  Hiển thị đúng menu của tài khoản Admin. |
| LG – 09 | Kiểm tra đăng nhập thành công tài khoản Store. | Hiển thị title: “Tra Cứu Dữ Liệu”.  Hiển thị đúng menu của tài khoản Store. |
| LG – 10 | Kiểm tra đăng nhập thành công tài khoản Supplier. | Hiển thị title: “Tra Cứu Dữ Liệu”  Hiển thị đúng menu của tài khoản Supplier. |
| LG – 11 | Kiểm tra đăng nhập thành công bằng phím Enter | Hiển thị title: “Tra Cứu Dữ Liệu”  Hiển thị đúng menu của tài khoản đăng nhập. |
| Navigate to Forgot Password Page | | |
| LG – 12 | Kiểm tra có thể điều hướng đến trang “Quên mật khẩu” từ trang “Đăng nhập”. | Điều hướng đến trang “Quên mật khẩu” thành công. |
| Validation Text Forgot Password Page | | |
| LG – 13 | Kiểm tra Validation Text khi bỏ trống Email. | Hiển thị Validation Text tại trường “Email”: “***This is a required field!***” |
| LG – 14 | Kiểm tra Validation Text khi nhập sai định dạng Email. | Hiển thị Validation Text trường “Email”: “***Please enter a valid email address!***”. |
| LG – 15 | Kiểm tra Validation Text khi không nhập Mã OTP | Hiển thị Validation Text tại trường “Mã OTP”: “***This is a required field!***” |
| Error Message Forgot Password Page | | |
| LG – 16 | Kiểm tra Error Message khi nhập email không tồn tại. | Hiển thị Error Message: “***Địa chỉ email không tồn tại, vui lòng thử địa chỉ email khác.***” |
| LG – 17 | Kiểm tra Error Message khi nhập sai Mã OTP. | Hiển thị Error Message: “***Mã OTP sai hoặc hết thời hạn sử dụng.***” |
| Back to login page from forgot password page | | |
| LG – 18 | Kiểm tra có thể quay trở lại trang đăng nhập. | Điều hướng đến trang đăng nhập thành công. |
| Display Password | | |
| LG – 19 | Kiểm tra có thể hiển thị mật khẩu dưới dạng text. | Mật khẩu được hiển thị dưới dạng text. |
| Reload page | | |
| LG – 20 | Kiểm tra thông tin đã nhập sẽ bị xóa khi reload page. | Thông tin đăng nhập tại các trường trống. |

### Kịch bản kiểm thử cho chức năng tạo mới Quản trị viên

Bảng 4.2. Kịch bản kiểm thử cho chức năng tạo mới Quản trị viên

|  |  |  |
| --- | --- | --- |
| Testcase ID | Yêu cầu test | Yêu cầu kết quả |
| Navigate to Create User Page | | |
| CRU – 01 | Kiểm tra có thể điều hướng đến trang tạo mới Quản trị viên | Điều hướng đến trang tạo mới Quản trị viên thành công. |
| Validation Text | | |
| CRU – 02 | Kiểm tra Validation text khi không nhập thông tin nào cả. | * + - * Hiển thị Validation Text trường “Họ và tên: “***Xin vui lòng nhập tên người dùng***”.       * Hiển thị Validation Text trường “Email”: “***Xin vui lòng nhập email người dùng***”.       * Hiển thị Validation Text trường “Số điện thoại”: “***Xin vui lòng nhập số điện thoại người dùng***”. |
| CRU – 03 | Kiểm tra Validation text khi bỏ trống “Họ và tên”. | Hiển thị Validation Text trường “Họ và tên”: “***Xin vui lòng nhập tên người dùng***”. |
| CRU – 04 | Kiểm tra Validation text khi nhập “Họ và tên” không hợp lệ. | Hiển thị Validation Text trường “Họ và tên”: “***Xin vui lòng chỉ nhập chữ***”. |
| CRU – 05 | Kiểm tra Validation text khi không nhập “Email” | Hiển thị Validation Text trường “Email”: “***Xin vui lòng nhập email người dùng***”. |
| CRU – 06 | Kiểm tra Validation text khi nhập sai định dạng “Email”. | Hiển thị Validation Text trường “Email”: “***Vui lòng nhập địa chỉ email hợp lệ!***”. |
| CRU – 07 | Kiểm tra Validation text khi không nhập “Số điện thoại”. | Hiển thị Validation Text trường “Số điện thoại”: “***Xin vui lòng nhập số điện thoại người dùng***”. |
| CRU – 08 | Kiểm tra Validation text khi nhập sai định dạng “Số điện thoại” và nhỏ hơn 8 ký tự. | Hiển thị Validation Text trường “Số điện thoại”: “***Xin vui lòng chỉ nhập số***” và “***Xin vui lòng nhập tối thiểu 8 ký tự số***”. |
| CRU – 09 | Kiểm tra Validation text khi nhập sai định dạng “Số điện thoại” và lớn hơn 12 ký tự. | Hiển thị Validation Text trường “Số điện thoại”: “***Xin vui lòng chỉ nhập số***” và “***Xin vui lòng nhập tối đa 12 ký tự số***”. |
| CRU – 10 | Kiểm tra Validation text khi nhập “Số điện thoại” nhỏ hơn 8 ký tự. | Hiển thị Validation Text trường “Số điện thoại”: “***Xin vui lòng nhập tối thiểu 8 ký tự số***”. |
| CRU – 11 | Kiểm tra Validation text khi nhập “Số điện thoại” lớn hơn 12 ký tự. | Hiển thị Validation Text trường “Số điện thoại”: “***Xin vui lòng nhập tối đa 12 ký tự số***”. |
| CRU – 12 | Kiểm tra Validation text khi nhập sai định dạng “Số điện thoại”. | Hiển thị Validation Text trường “Số điện thoại”: “***Xin vui lòng chỉ nhập số***”. |
| CRU – 13 | Kiểm tra Validation text khi nhập “Ghi chú” quá 500 ký tự. | Hiển thị Validation Text trường “Ghi chú”: “***Chỉ được nhập tối đa 500 kí tự***”. |
| Error Message | | |
| CRU – 14 | Kiểm tra Error Message khi nhập “Email” đã tồn tại. | Hiển thị Error Message: “***Email đã được đăng ký trước đó.***”. |
| CRU – 15 | Kiểm tra Error Message khi nhập “Số điện thoại” đã tồn tại. | Hiển thị Error Message: “***Số điện thoại đã được đăng ký trước đó.***”. |
| Create User Successfully | | |
| CRU – 16 | Kiểm tra tạo mới người dùng thành công. | Tạo mới người dùng thành công. |
| CRU – 17 | Kiểm tra điều hướng đến trang “Quản lý người dùng” từ trang “Tạo mới người dùng” | Điều hướng đến trang “Quản lý người dùng” thành công. |
| Reload Page | | |
| CRU – 18 | Kiểm tra thông tin đã nhập sẽ bị xóa khi reload page. | Thông tin đăng nhập tại các trường trống. |
| View list of Users | | |
| CRU – 19 | Kiểm tra hiển thị thành công danh sách người dùng đã tạo. | Hiển thị thành công danh sách người dùng đã tạo. |
| CRU – 20 | Kiểm tra tìm kiếm thành công với “Mã người dùng” hợp lệ. | Hiển thị người dùng ứng với “Mã người dùng” hợp lệ. |
| CRU – 21 | Kiểm tra tìm kiếm thành công với “Họ và tên” hợp lệ. | Hiển thị người dùng ứng với “Họ và tên” hợp lệ. |
| CRU – 22 | Kiểm tra tìm kiếm thành công với “Email” hợp lệ. | Hiển thị người dùng ứng với “Email” hợp lệ. |
| CRU – 23 | Kiểm tra tìm kiếm thành công với “Số điện thoại” hợp lệ. | Hiển thị người dùng ứng với “Số điện thoại” hợp lệ. |
| CRU – 24 | Kiểm tra tìm kiếm thành công với “Vai trò” hợp lệ. | Hiển thị người dùng ứng với “Vai trò” hợp lệ. |
| CRU – 25 | Kiểm tra tìm kiếm không thành công với “Mã người dùng” không hợp lệ. | Không có user nào được hiển thị |
| CRU – 26 | Kiểm tra tìm kiếm không thành công với “Họ và tên” không hợp lệ. | Không có user nào được hiển thị |
| CRU – 27 | Kiểm tra tìm kiếm không thành công với “Email” không hợp lệ. | Không có user nào được hiển thị |
| CRU – 28 | Kiểm tra tìm kiếm không thành công với “Số điện thoại” không hợp lệ. | Không có user nào được hiển thị |
| CRU – 29 | Kiểm tra có thể xem danh sách trang kế tiếp, trước trước, 10 trang kế tiếp, 10 trang trước | Hiển thị danh sách người dùng ứng với trang đó. |

### Kịch bản kiểm thử cho chức năng chỉnh sửa người dùng

Bảng 4.3. Kịch bản kiểm thử cho chức năng chỉnh sửa người dùng

|  |  |  |
| --- | --- | --- |
| Testcase ID | Yêu cầu test | Yêu cầu kết quả |
| Navigate to Edit User Page | | |
| EDU – 01 | Kiểm tra có thể điều hướng đến trang chỉnh sửa người dùng | Điều hướng đến trang Chỉnh sửa người dùng thành công. |
| Validation Text | | |
| EDU – 02 | Kiểm tra Validation text khi bỏ trống “Họ và tên”. | * + - * Hiển thị Validation Text trường “Họ và tên”: “***Xin vui lòng nhập tên người dùng***”. |
| EDU – 03 | Kiểm tra Validation text khi nhập “Họ và tên” không hợp lệ. | Hiển thị Validation Text trường “Họ và tên”: “***Xin vui lòng chỉ nhập chữ***”. |
| EDU – 04 | Kiểm tra Validation text khi không nhập “Số điện thoại”. | Hiển thị Validation Text trường “Số điện thoại”: “***Xin vui lòng nhập số điện thoại người dùng***”. |
| EDU – 05 | Kiểm tra Validation text khi nhập sai định dạng “Số điện thoại” và nhỏ hơn 8 ký tự. | Hiển thị Validation Text trường “Số điện thoại”: “***Xin vui lòng chỉ nhập số***” và “***Xin vui lòng nhập tối thiểu 8 ký tự số***”. |
| EDU – 06 | Kiểm tra Validation text khi nhập sai định dạng “Số điện thoại” và lớn hơn 12 ký tự. | Hiển thị Validation Text trường “Số điện thoại”: “***Xin vui lòng chỉ nhập số***” và “***Xin vui lòng nhập tối đa 12 ký tự số***”. |
| EDU – 07 | Kiểm tra Validation text khi nhập “Số điện thoại” nhỏ hơn 8 ký tự. | Hiển thị Validation Text trường “Số điện thoại”: “***Xin vui lòng nhập tối thiểu 8 ký tự số***”. |
| EDU – 08 | Kiểm tra Validation text khi nhập “Số điện thoại” lớn hơn 12 ký tự. | Hiển thị Validation Text trường “Số điện thoại”: “***Xin vui lòng nhập tối đa 12 ký tự số***”. |
| EDU – 09 | Kiểm tra Validation text khi nhập sai định dạng “Số điện thoại”. | Hiển thị Validation Text trường “Số điện thoại”: “***Xin vui lòng chỉ nhập số***”. |
| EDU – 10 | Kiểm tra Validation text khi nhập “Ghi chú” quá 500 ký tự. | Hiển thị Validation Text trường “Ghi chú”: “***Chỉ được nhập tối đa 500 kí tự***”. |
| Error Message | | |
| EDU – 11 | Kiểm tra Error Message khi nhập “Số điện thoại” đã tồn tại. | Hiển thị Error Message: “***Số điện thoại hoặc email đã tồn tại trong hệ thống.***”. |
| Edit User Successfully | | |
| EDU – 12 | Kiểm tra Chỉnh sửa người dùng thành công khi thay đổi “Họ và tên” | Chỉnh sửa người dùng thành công. |
| EDU – 13 | Kiểm tra Chỉnh sửa người dùng thành công khi thay đổi “Số điện thoại” | Chỉnh sửa người dùng thành công. |
| EDU – 14 | Kiểm tra Chỉnh sửa người dùng thành công khi thay đổi “Ghi chú” | Chỉnh sửa người dùng thành công. |
| EDU – 15 | Kiểm tra điều hướng đến trang “Quản lý người dùng” từ trang “Chỉnh sửa người dùng” | Điều hướng đến trang “Quản lý người dùng” thành công. |

3.3. Xây dựng các test cases cho từng kịch bản

3.4. Tạo và sử dụng custom keywords

3.5. Sử dụng thư viện hỗ trợ kiểm thử web

**CHƯƠNG 4: TỔNG KẾT VÀ ĐÁNH GIÁ**

4.1. Xử lý kết quả kiểm thử

4.2. Tự động hóa việc thực hiện test suite

4.3. So sánh Robot Framework với các công cụ kiểm thử khác

4.4. Tổng kết nội dung đồ án

4.5. Đánh giá đạt được mục tiêu đề ra ban đầu

4.6. Hướng phát triển và nâng cao trong tương lai