**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HƯNG YÊN**

****

**ĐỒ ÁN 3**

**KIỂM THỬ TỰ ĐỘNG VÀ XÂY DỰNG FRAMEWORK DỰA TRÊN SELENIUM CHO WEBSITE SÁCH TAO ĐÀN**

NGÀNH: KỸ THUẬT PHẦN MỀM

CHUYÊN NGÀNH: KIỂM THỬ VÀ

ĐẢM BẢO CHẤT LƯỢNG PHẦN MỀM

SINH VIÊN: **HOÀNG MINH NGUYỆT**

MÃ LỚP: **12522T.1CN**

HƯỚNG DẪN: **TS ĐÀO ANH HIỂN**

**HƯNG YÊN - 2025**

NHẬN XÉT

Nhận xét của giảng viên hướng dẫn:

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

**GIẢNG VIÊN HƯỚNG DẪN**

*(Ký và ghi rõ họ tên)*

**LỜI CAM ĐOAN**

Em xin cam đoan đồ án “Kiểm thử tự động và xây dựng framework dựa trên Slenium cho website Sách Tao Đàn” là kết quả thực hiện của bản thân em dưới sự hướng dẫn của thầy/cô Đào Anh Hiển.

Những phần sử dụng tài liệu tham khảo trong đồ án đã được nêu rõ trong phần tài liệu tham khảo. Các kết quả trình bày trong đồ án và chương trình xây dựng được hoàn toàn là kết quả do bản thân em thực hiện.

Nếu vi phạm lời cam đoan này, em xin chịu hoàn toàn trách nhiệm trước Khoa và Nhà trường.

|  |  |
| --- | --- |
|  | *Hưng Yên, ngày … tháng … năm…..*  Sinh viên |

**LỜI CẢM ƠN**

Để có thể hoàn thành đồ án này, đầu tiên em xin phép gửi lời cảm ơn tới bộ môn Công nghệ phần mềm, Khoa Công nghệ thông tin - Trường Đại học Sư phạm Kỹ thuật Hưng Yên đã tạo điều kiện thuận lợi cho em thực hiện đồ án môn học này.

Đặc biệt em xin chân thành cảm ơn thầy/cô Đào Anh Hiển đã rất tận tình hướng dẫn, chỉ bảo em trong suốt thời gian thực hiện đồ án vừa qua.

Em cũng xin chân thành cảm ơn tất cả các Thầy, các Cô trong Trường đã tận tình giảng dạy, trang bị cho em những kiến thức cần thiết, quý báu để giúp em thực hiện được đồ án này.

Mặc dù em đã có cố gắng, nhưng với kiến thức còn hạn chế, trong quá trình thực hiện đề tài không tránh khỏi những thiếu sót. Em hi vọng sẽ nhận được những ý kiến nhận xét, góp ý của các Thầy giáo, Cô giáo về những kết quả triển khai trong đồ án.

Em xin trân trọng cảm ơn!**MỤC LỤC**

[NHẬN XÉT 2](#_Toc194355660)

[Nhận xét của giảng viên hướng dẫn: 2](#_Toc194355661)

[CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI 10](#_Toc194355662)

[1.1 Lý do chọn đề tài 10](#_Toc194355663)

[1.2 Mục tiêu của đề tài 10](#_Toc194355664)

[1.2.1 Mục tiêu tổng quát 10](#_Toc194355665)

[1.2.2 Mục tiêu cụ thể 10](#_Toc194355666)

[1.3 Giới hạn và phạm vi của đề tài 10](#_Toc194355667)

[1.3.1 Đối tượng nghiên cứu 10](#_Toc194355668)

[1.3.2 Phạm vi nghiên cứu 11](#_Toc194355669)

[1.4 Nội dung thực hiện 11](#_Toc194355670)

[1.5 Phương pháp tiếp cận 11](#_Toc194355671)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 12](#_Toc194355672)

[2.1. Tổng quan về kiểm thử tự động 12](#_Toc194355673)

[2.2. Giới thiệu các công cụ kiểm thử: 12](#_Toc194355674)

[2.3. Các kiến trúc của các Framework kiểm thử tự động 12](#_Toc194355675)

[2.4. Test Script và phương pháp viết Test Script 12](#_Toc194355676)

[CHƯƠNG 3: PHÂN TÍCH YÊU CẦU VÀ LẬP KẾ HOẠCH KIỂM THỬ 13](#_Toc194355677)

[3.1. Phân tích yêu cầu kiểm thử 13](#_Toc194355678)

[3.1.1 Yêu cầu chức năng 13](#_Toc194355679)

[3.1.2 Yêu cầu phi chức năng 13](#_Toc194355680)

[3.2. Kế hoạch kiểm thử 13](#_Toc194355681)

[3.2.1. Phương pháp kiểm thử 13](#_Toc194355682)

[3.2.2. Môi trường kiểm thử 13](#_Toc194355683)

[3.2.3. Lịch trình và phân công công việc 13](#_Toc194355684)

[CHƯƠNG 4: TRIỂN KHAI KIỂM THỬ ỨNG DỤNG 14](#_Toc194355685)

[4.1. Thiết kế kịch bản kiểm thử 14](#_Toc194355686)

[4.1.1. Thiết kế kịch bản Kiểm thử chức năng 14](#_Toc194355687)

[4.1.2. Thiết kế kịch bản Kiểm thử giao diện 14](#_Toc194355688)

[4.1.3. Thiết kế kịch bản Kiểm thử hiệu xuất 14](#_Toc194355689)

[4.2. Xây dựng và thực thi các trường hợp kiểm thử 14](#_Toc194355690)

[4.2.1. Xây dựng và thực thi các trường hợp Kiểm thử chức năng 14](#_Toc194355691)

[4.2.2. Xây dựng và thực thi các trường hợp Kiểm thử giao diện 14](#_Toc194355692)

[4.2.3. Xây dựng và thực thi các trường hợp Kiểm thử hiệu xuất 14](#_Toc194355693)

[4.2.4. Xây dựng và thực thi các trường hợp Kiểm thử hướng dữ liệu/Từ khóa (Nếu có) 14](#_Toc194355694)

[4.3. Báo cáo và phân tích kết quả kiểm thử 14](#_Toc194355695)

[4.3.1. Báo cáo kiểm thử chức năng 14](#_Toc194355696)

[4.3.2. Báo cáo kiểm thử giao diện 14](#_Toc194355697)

[4.3.3. Báo cáo kiểm thử hiệu suất 14](#_Toc194355698)

[CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 15](#_Toc194355699)

[5.1. Kết quả đạt được của đề tài 15](#_Toc194355700)

[5.2. Hạn chế của đề tài 15](#_Toc194355701)

[5.3. Hướng phát triển của đề tài 15](#_Toc194355702)

[TÀI LIỆU THAM KHẢO 16](#_Toc194355703)

DANH MỤC CÁC THUẬT NGỮ

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Từ viết tắt | Cụm từ tiếng anh | Diễn giải |
| 1 | SRS | Software Requirement Specification | Đặc tả yêu cầu phần mềm |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

DANH MỤC CÁC BẢNG

[Bảng 3‑1: Thiết kế bảng Skins lưu trữ các loại khung giao diện](#_Toc388683208) 15

[Bảng 3‑2: Thiết kế bảng lưu trữ các Module được thiết kế](#_Toc388683209) 15

DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ

[Hình 3‑1: Mô hình kiến trúc tổng thể hệ thống SEVN Framework](#_Toc388683110) 14

[Hình 3‑2: Mô hình kiến trúc chi tiết của hệ thống SEVN Framework](#_Toc388683111) 14

[Hình 3‑3: Mô hình lưu trữ cấu trúc website trên Database](#_Toc388683112) 14

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1.1 Lý do chọn đề tài

Trong thời đại công nghệ thông tin phát triển mạnh mẽ, mua sắm trực tuyến đã trở thành xu hướng phổ biến và quen thuộc đối với người tiêu dùng. Đặc biệt, sách là một trong những mặt hàng được lựa chọn mua online nhiều nhờ sự tiện lợi, nhanh chóng và kho sản phẩm đa dạng. Nắm bắt xu thế đó, hàng loạt nền tảng bán sách trực tuyến đã ra đời, trong đó có website Sách Tao Đàn - một nền tảng phân phối sách trực tuyến với số lượng sản phẩm lớn và nhiều chức năng phục vụ nhu cầu người dùng.

Để đảm bảo mang lại trải nghiệm mua sắm tốt nhất cho khách hàng, website Sách Tao Đàn cần vận hành ổn định, nhanh chóng, chính xác và đảm bảo an toàn bảo mật. Vì vậy, công tác kiểm thử phần mềm đóng vai trò then chốt trong việc phát hiện, khắc phục lỗi và duy trì chất lượng hệ thống trước khi sản phẩm được đưa vào sử dụng thực tế. Tuy nhiên, phương pháp kiểm thử thủ công hiện nay vẫn còn tồn tại nhiều hạn chế như tốn nhiều thời gian, nhân lực, dễ mắc sai sót và khó tái sử dụng các kịch bản kiểm thử khi hệ thống thay đổi.

Để khắc phục những hạn chế đó, kiểm thử tự động được áp dụng nhờ những lợi ích vượt trội so với kiểm thử thủ công. Cụ thể, kiểm thử tự động giúp tiết kiệm thời gian và chi phí, tăng độ chính xác, dễ dàng tái sử dụng và mở rộng, nâng cao hiệu suất kiểm thử**,** hỗ trợ kiểm thử liên tục.

Đặc biệt với website Sách Tao Đàn, người dùng thường xuyên thực hiện các thao tác như đăng nhập, đăng ký, tìm kiếm sách, thêm sách yêu thích, thêm vào giỏ hàng và đặt mua. Nếu thực hiện kiểm thử thủ công cho các thao tác này nhiều lần sẽ rất tốn thời gian và dễ xảy ra sai sót. Do đó, việc xây dựng một framework kiểm thử tự động cho website thương mại điện tử Sách Tao Đàn là thực sự cần thiết nhằm đảm bảo hệ thống hoạt động ổn định, tiết kiệm chi phí, nâng cao chất lượng kiểm thử và tối ưu trải nghiệm cho người dùng.

1.2 Mục tiêu của đề tài

1.2.1 Mục tiêu tổng quát

Kiểm thử tự động website Sách Tao Đàn dựa trên framework Selenium.

1.2.2 Mục tiêu cụ thể

- Tìm hiểu và phân tích quy trình kiểm thử tự động cho website Sách Tao Đàn, lựa chọn phương pháp và công cụ kiểm thử tự động phù hợp, cụ thể là công cụ Selenium.

- Đưa ra các giải pháp nhằm tăng chất lượng kiểm thử tự động, giảm thiểu thời gian kiểm thử, nâng cao độ chính xác và hiệu suất hoạt động của hệ thống.

- Xây dựng và triển khai các kịch bản kiểm thử tự động cho các chức năng trong website Sách Tao Đàn như: chức năng đăng nhập, đăng ký, tìm kiếm sách, thêm vào yêu thích, thêm vào giỏ hàng và thực hiện đặt mua.

- Thiết kế và xây dựng framework kiểm thử tự động trên nền tảng Selenium, đảm bảo tính linh hoạt, dễ dàng mở rộng và tái sử dụng cho các đợt kiểm thử tiếp theo.

- Thực hiện kiểm thử, thu thập kết quả và so sánh hiệu quả giữa kiểm thử thủ công và kiểm thử tự động về các tiêu chí: thời gian thực hiện, độ chính xác và khả năng tái sử dụng.

1.3 Giới hạn và phạm vi của đề tài

1.3.1 Đối tượng nghiên cứu

- Đối tượng nghiên cứu: Kiểm thử tự động website Sách Tao Đàn, tập trung vào các phương pháp, công cụ và kỹ thuật kiểm thử chức năng, kiểm thử hiệu năng.

- Khách thể nghiên cứu: Người dùng website Sach Tao Đàn, chuyên gia phát triển phần mềm, chuyên gia kiểm thử phần mềm.

1.3.2 Phạm vi nghiên cứu

* Phạm vi không gian: Website Scahs Tao Đàn.
* Phạm vi thời gian: Thu thập dữ liệu, triển khia vfa đánh giá trong khoảng thời gian từ 02/2025 - 06/2025.
* Ý nghĩa khoa học và thực tiễn của đề tài

**Về mặt khoa học:**

* Đề tài góp phần hệ thống hóa các kiến thức lý thuyết và thực tiễn liên quan đến kiểm thử phần mềm.
* Cung cấp một giải pháp framework kiểm thử tự động dựa trên Selenium WebDriver, giúp làm rõ quy trình thiết kế, triển khai và vận hành hệ thống kiểm thử tự động cho các website thương mại điện tử.
* Là tài liệu tham khảo hữu ích cho các nghiên cứu và đề tài sau này liên quan đến kiểm thử tự động và kiểm thử website thương mại điện tử.

**Về mặt thực tiễn:**

* Framework kiểm thử tự động do đề tài xây dựng giúp tiết kiệm thời gian, chi phí kiểm thử, đồng thời nâng cao độ chính xác và hiệu quả kiểm thử cho website Sách Tao Đàn.
* Hỗ trợ phát hiện và xử lý lỗi kịp thời, đảm bảo hệ thống vận hành ổn định, an toàn, góp phần nâng cao chất lượng trải nghiệm người dùng.
* Có thể mở rộng và áp dụng cho các dự án website thương mại điện tử khác có quy mô và tính năng tương tự.

1.4 Nội dung thực hiện

- Nghiên cứu tài liệu về kiểm thử phần mềm, kiểm thử tự động, các công cụ kiểm thử phổ biến hiện nay.

- Khảo sát và phân tích website Sách Tao Đàn để xác định các chức năng cần kiểm thử như: đăng ký, đăng nhập, tìm kiếm sách, thêm vào giỏ hàng, đặt mua, yêu thích sách…

- Lựa chọn công cụ kiểm thử phù hợp, cụ thể là Selenium WebDriver, dựa trên yêu cầu hệ thống và khả năng đáp ứng các chức năng kiểm thử tự động.

- Xây dựng bộ kịch bản kiểm thử tự động, bao gồm các trường hợp kiểm thử chức năng và kiểm thử bảo mật cho các chức năng chính của website.

- Thiết kế và triển khai framework kiểm thử tự động cho website Sách Tao Đàn, thực hiện kiểm thử thực tế, thu thập và phân tích kết quả kiểm thử.

- Đánh giá hiệu quả kiểm thử tự động so với kiểm thử thủ công, về các tiêu chí: thời gian thực hiện, độ chính xác, khả năng tái sử dụng và hiệu suất hệ thống.

- Viết báo cáo tổng kết quá trình nghiên cứu, trình bày kết quả đạt được, khó khăn gặp phải và đề xuất hướng phát triển trong tương lai.

* 1. Phương pháp tiếp cận

Để thực hiện đề tài **“**Xây dựng framework kiểm thử tự động cho website thương mại điện tử Sách Tao Đàn bằng Selenium”, nhóm thực hiện lựa chọn phương pháp tiếp cận trực tiếp và từng bước như sau:

* Tiếp cận trực tiếp với người sử dụng hệ thống, thu thập yêu cầu và phản hồi từ người dùng, quản trị viên hệ thống và bộ phận kỹ thuật nhằm nắm rõ các chức năng quan trọng, các lỗi thường gặp cũng như nhu cầu kiểm thử hiện tại của website Sách Tao Đàn.
* Khảo sát hệ thống website hiện tại, phân tích quy trình kiểm thử thủ công đang áp dụng để nhận diện các điểm hạn chế và những khu vực chức năng cần kiểm thử liên tục.
* Nghiên cứu, đề xuất và lựa chọn phương án kiểm thử tự động phù hợp, trong đó tập trung vào công cụ Selenium WebDriver nhờ khả năng hỗ trợ kiểm thử giao diện web, đa trình duyệt và dễ tích hợp vào quy trình kiểm thử liên tục (CI/CD).
* Lên kế hoạch xây dựng framework kiểm thử tự động, thiết kế các kịch bản kiểm thử dựa trên các chức năng trọng yếu của hệ thống, đồng thời đảm bảo tính mở rộng và tái sử dụng cho các đợt kiểm thử về sau.
* Tiến hành thử nghiệm framework kiểm thử tự động trên hệ thống thực tế, ghi nhận kết quả kiểm thử và đánh giá hiệu suất hệ thống.
* Đề xuất các phương án cải tiến, hoàn thiện framework kiểm thử, đồng thời kiến nghị hướng áp dụng lâu dài vào quy trình phát triển và vận hành website.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về kiểm thử tự động

2.1.1. Định ngĩa, vai trò của kiểm thử tự động

Trong quá trình phát triển phần mềm, kiểm thử đóng vai trò vô cùng quan trọng nhằm đảm bảo sản phẩm hoạt động đúng chức năng, ổn định và đáp ứng được yêu cầu của người sử dụng. Trong đó, kiểm thử tự động (Automated Testing) là một phương pháp kiểm thử hiện đại, sử dụng các công cụ và phần mềm để tự động hóa việc thực thi các kịch bản kiểm thử thay cho phương pháp kiểm thử thủ công truyền thống.

Kiểm thử tự động giúp kiểm tra phần mềm một cách nhanh chóng, chính xác và hiệu quả, đặc biệt trong những dự án có quy mô lớn, nhiều chức năng hoặc thường xuyên thay đổi, cập nhật tính năng. Các kịch bản kiểm thử được lập trình sẵn và có thể thực hiện lặp đi lặp lại nhiều lần với cùng một tập dữ liệu, giúp phát hiện lỗi sớm và đảm bảo hệ thống luôn vận hành ổn định.

Mục tiêu chính của kiểm thử tự động là:

* Tăng hiệu quả và tốc độ thực hiện kiểm thử.
* Giảm thiểu sai sót do yếu tố con người trong quá trình kiểm tra.
* Tiết kiệm thời gian và chi phí cho việc kiểm tra các chức năng lặp lại nhiều lần.
* Hỗ trợ kiểm thử liên tục (Continuous Testing) trong các quy trình phát triển phần mềm hiện đại như CI/CD.
* Dễ dàng tái sử dụng, chỉnh sửa và mở rộng các kịch bản kiểm thử khi phần mềm thay đổi.

Những lợi ích nổi bật của kiểm thử tự động gồm:

* Tăng tốc độ kiểm thử: Thời gian thực thi các kịch bản kiểm thử được rút ngắn đáng kể so với kiểm thử thủ công.
* Giảm lỗi và nâng cao độ chính xác: Các bước kiểm thử được thực hiện nhất quán, giảm thiểu các sai sót do con người.
* Tái sử dụng và mở rộng: Các kịch bản kiểm thử tự động có thể sử dụng lại cho nhiều phiên bản phần mềm khác nhau.
* Hỗ trợ kiểm thử liên tục: Giúp kiểm tra phần mềm ngay khi có thay đổi mới, đảm bảo sản phẩm luôn ổn định trước khi đến tay người dùng.
* Tiết kiệm chi phí lâu dài: Mặc dù chi phí ban đầu để thiết lập kiểm thử tự động cao hơn, nhưng về lâu dài sẽ tiết kiệm nhân lực và chi phí vận hành.

2.1.2. Phân loại kiểm thử tự động

Có các loại kiểm thử tự động khác nhau tùy vào mục đích phạm vi kiểm thử.

- **Kiểm thử đơn vị (Unit Testing)**

Là quá trình kiểm tra từng thành phần nhỏ nhất của phần mềm như hàm, phương thức, module hoặc đối tượng. Mục đích nhằm đảm bảo từng đơn vị hoạt động chính xác độc lập trước khi tích hợp vào hệ thống lớn hơn. Unit test giúp phát hiện lỗi sớm, giảm chi phí sửa lỗi và tăng khả năng tái sử dụng mã nguồn.

Các loại unit test phổ biến:

* Black Box Testing: Kiểm tra input và output mà không xét logic bên trong.
* White Box Testing: Kiểm tra logic, cấu trúc nội bộ của mã nguồn.
* Gray Box Testing: Kết hợp hai phương pháp trên*.*

**- Kiểm thử tích hợp (Integration Testing)**

Là kiểm tra quá trình kết hợp các module, thành phần của hệ thống để đảm bảo chúng hoạt động đồng bộ, truyền dữ liệu chính xác và không phát sinh lỗi khi tích hợp. Loại kiểm thử này giúp phát hiện sớm lỗi liên quan đến giao tiếp giữa các module.

Kỹ thuật kiểm thử tích hợp:

* Big Bang: Tích hợp tất cả module rồi kiểm thử một lần.
* Top-Down: Tích hợp dần từ trên xuống dưới.
* Bottom-Up: Tích hợp dần từ dưới lên trên.
* Sandwich (Hybrid): Kết hợp cả Top-Down và Bottom-Up.

**- Kiểm thử chức năng (Functional Testing)**

Kiểm tra xem phần mềm có đáp ứng đúng các yêu cầu chức năng đặt ra hay không. Thường sử dụng phương pháp Black Box Testing, tập trung vào đầu vào, đầu ra mà không cần biết đến cấu trúc bên trong.

Bao gồm:

- Kiểm thử đơn vị, kiểm thử tích hợp, kiểm thử khói (Smoke Testing) và kiểm thử chấp nhận (Acceptance Testing).

**- Kiểm thử hồi quy (Regression Testing)**

Là kiểm tra lại các tính năng của hệ thống sau khi có thay đổi về mã nguồn (sửa lỗi, nâng cấp tính năng) nhằm đảm bảo các chức năng cũ vẫn hoạt động tốt và không phát sinh lỗi mới. Đây là loại kiểm thử quan trọng trong quá trình bảo trì phần mềm.

**- Kiểm thử chấp nhận (Acceptance Testing):** Kiểm thử chấp nhận là quá trình kiểm tra tính đúng đắn và đáp ứng yêu cầu của người dùng. Nó được thực hiện bởi người dùng cuối và đảm bảo các tính năng của phần mềm đã đáp ứng chính xác các tiêu chí chấp thuận được đề ra từ ban đầu, dưới góc nhìn của người dùng.

**- Kiểm thử hoạt động (Operational Testing):** Kiểm thử hoạt động tập trung vào việc đảm bảo rằng hệ thống hoạt động một cách đáng tin cậy trong môi trường sản xuất. Bằng cách này, ta có thể phát hiện và khắc phục các vấn đề liên quan đến hiệu suất, bảo mật và khả năng chịu tải của hệ thống.

**- Kiểm thử phi chức năng (Non-functional Testing):** Kiểm thử phi chức năng là quá trình kiểm tra các yếu tố không liên quan đến tính năng của phần mềm như trải nghiệm người dùng, hiệu suất, bảo mật, khả năng mở rộng và khả năng tương thích. Điều này giúp đảm bảo tính linh hoạt và hiệu quả của hệ thống trong các tình huống thực tế.

2.1.3 Quy trình kiểm thử tự động



**1. Lập kế hoạch và thiết kế kịch bản kiểm thử**

Ở giai đoạn đầu tiên, nhóm kiểm thử sẽ tiến hành lập kế hoạch tổng thể cho quá trình kiểm thử tự động. Công việc chính bao gồm:

Xác định mục tiêu và phạm vi kiểm thử tự động, những chức năng, module hoặc thành phần nào của phần mềm sẽ được tự động hóa.

Phân tích các yêu cầu hệ thống để xác định những chức năng có tính ổn định cao, ít thay đổi và có khả năng tự động hóa tốt.

Thiết kế các kịch bản kiểm thử chi tiết dựa trên các yêu cầu chức năng và phi chức năng đã phân tích. Các kịch bản này cần đầy đủ thông tin về:

* Dữ liệu đầu vào
* Các bước thực hiện
* Điều kiện đầu vào và đầu ra mong đợi

Việc thiết kế kịch bản kỹ lưỡng giúp quá trình tự động hóa sau này hiệu quả và giảm thiểu các lỗi phát sinh.

**2. Lựa chọn và thiết lập công cụ kiểm thử**

Tiếp theo, nhóm kiểm thử sẽ tiến hành:

* Khảo sát và lựa chọn công cụ kiểm thử tự động phù hợp với công nghệ phần mềm, ngân sách và nhu cầu dự án. Một số công cụ phổ biến như: Selenium, Katalon, TestComplete, Appium, JMeter,…
* Cài đặt và thiết lập môi trường kiểm thử tự động bao gồm các framework, thư viện và các công cụ hỗ trợ cần thiết.
* Cấu hình tích hợp công cụ kiểm thử với hệ thống quản lý phiên bản (Git), hệ thống CI/CD (Jenkins, GitLab CI), và các công cụ quản lý kiểm thử khác nếu có.

Việc lựa chọn đúng công cụ sẽ giúp tối ưu hóa chi phí và thời gian cho dự án kiểm thử tự động.

**3. Phát triển và tự động hóa kịch bản kiểm thử**

Sau khi có công cụ và môi trường, nhóm kiểm thử sẽ:

* Tiến hành viết mã kiểm thử tự động hoặc thiết lập các bước kiểm thử bằng giao diện tùy thuộc vào loại công cụ.
* Chuyển các kịch bản kiểm thử từ bản thiết kế sang các đoạn mã hoặc test script.
* Áp dụng các kỹ thuật kiểm thử như:
  + **Data-Driven Testing**: kiểm thử với nhiều bộ dữ liệu khác nhau.
  + **Keyword-Driven Testing**: kiểm thử theo các từ khóa đại diện cho hành động.
  + **Modular Testing**: chia nhỏ kịch bản kiểm thử thành các module độc lập để dễ quản lý và tái sử dụng.

Đồng thời tổ chức các test suite để nhóm các test case cùng loại với nhau, phục vụ cho quá trình chạy kiểm thử tự động.

**4. Thực thi và theo dõi kết quả kiểm thử**

Ở bước này, các test suite và test case sẽ được chạy tự động:

* Thực thi các kịch bản kiểm thử tự động đã phát triển.
* Theo dõi quá trình thực thi và ghi lại log, báo cáo kết quả kiểm thử.
* Phân tích kết quả thực thi để xác định các lỗi phát sinh, điểm chưa đạt hoặc các bất thường.
* Thực hiện báo cáo chi tiết cho nhóm phát triển về các lỗi gặp phải và phối hợp xử lý.

Việc chạy kiểm thử tự động giúp phát hiện lỗi nhanh chóng và liên tục sau mỗi lần cập nhật hệ thống.

5. **Bảo trì và tối ưu hóa kiểm thử tự động**

Cuối cùng, để đảm bảo bộ kiểm thử luôn hoạt động chính xác:

* Cập nhật, chỉnh sửa các test script khi phần mềm có sự thay đổi về chức năng hoặc giao diện.
* Xóa bỏ những kịch bản không còn cần thiết, thêm mới những kịch bản cho chức năng mới.
* Tối ưu hóa test case, test suite để giảm thời gian thực thi và tăng độ ổn định.
* Đánh giá hiệu quả của quá trình kiểm thử tự động để có các điều chỉnh phù hợp.

Bảo trì kiểm thử tự động là một công việc định kỳ và rất quan trọng để đảm bảo tính chính xác và khả năng tái sử dụng của hệ thống kiểm thử.

2.1.4. Ưu, nhược điểm kiểm thử tự động

**+ Ưu điểm:**

* Độ tin cậy cao: Công cụ kiểm thử tự động có sự ổn định cao vì được hoạt động dựa trên một quy trình sẵn có. Đặc biệt, trong trường hợp nhiều test case sẽ tránh được tình trạng lỗi do con người tạo ra khi tiến hành kiểm tra thủ công.
* Khả năng lặp: Bạn có thể dễ dàng test các phần mềm xử lý (tính năng/ hiệu năng) khi gặp tình huống chạy lặp đi lặp lại nhiều lần trên cùng script test. Điều này giúp các tester xử lý được trường hợp lặp lại các thao tác như nhập dữ liệu, check kết quả, click,… Đây còn được gọi là performance hoặc Load testing.
* Khả năng tái sử dụng: Với automation testing bạn có thể tái sử dụng ứng dụng trên nhiều phiên bản ngay cả khi có sự thay đổi giao diện. Có thể test ở nhiều môi trường khác như như: beta, production,..
* Tốc độ cao: Automation test giúp chạy test với tốc độ nhanh hơn ít nhất 10 lần so với chạy thủ công bằng tay. Bạn chỉ cần mất khoảng 30s để thực thi việc này một cách tự động thay vì mất 5 phút cho test tay thủ công.
* Chi phí thấp: Nếu biết áp dụng kiểm thử tự động đúng cách, đúng mục đích thì bạn có thể tiết kiệm được rất nhiều chi phí cả về thời gian và nhân lực. Bởi kiểm thử tự động nhanh hơn nhiều so với thủ công. Hơn nữa, nhân lực cần để thực hiện và bảo trì scripts không nhiều.

**+ Nhược điểm:**

* Khó mở rộng, khó bảo trì: Để mở rộng phạm vi cho kiểm thử tự động khó hơn nhiều so với thủ công trong cùng một dự án. Nguyên nhân là bởi vì cập nhật hay chỉnh sửa yêu cầu nhiều công việc như debug, thay đổi dữ liệu đầu vào và cập nhật code mới.
* Khả năng bao phủ thấp: Do đòi hỏi nhiều kỹ năng lập trình và khó mở rộng, nên độ bao phủ của kiểm thử tự động khi xét trên góc nhìn toàn dự án sẽ thấp hơn
* Đối với vấn đề công cụ và nhân lực: Hiện nay, mặc dù có rất nhiều công cụ kiểm thử tự động nhằm hỗ trợ được đánh giá cao, tuy nhiên vẫn còn nhiều hạn chế. Ngoài ra, nhân lực đạt yêu cầu cũng không nhiều.
* Tốn thời gian: Để có thể áp dụng một cách thành thạo các công cụ kiểm thử đòi hỏi thời gian chuẩn bị dài, thiết kế, cài đặt kỹ càng trước khi chạy dự án.

2.2. Giới thiệu các công cụ kiểm thử:

2.2.1. Selenium

Selenium là một framework kiểm thử tự động mã nguồn mở, được sử dụng phổ biến để kiểm thử các ứng dụng web trên nhiều trình duyệt và nền tảng khác nhau. Không giống như các framework kiểm thử chạy trong trình duyệt như Cypress, Selenium điều khiển trình duyệt thông qua WebDriver, cho phép tương tác với trang web tương tự như người dùng thật, thực hiện các thao tác như click, nhập liệu, kiểm tra dữ liệu hiển thị, và xác thực nội dung trang web.

Selenium đặc biệt thích hợp cho các dự án cần kiểm thử chức năng, kiểm thử giao diện người dùng (UI Testing) và kiểm thử End-to-End (E2E), nhất là với các website thương mại điện tử như Sách Tao Đàn — nơi có nhiều thao tác và chức năng người dùng cần kiểm tra.

Đặc điểm nổi bật:

* Hỗ trợ đa trình duyệt: Selenium hỗ trợ kiểm thử trên nhiều trình duyệt phổ biến như Chrome, Firefox, Edge, Safari…, giúp kiểm tra khả năng tương thích giao diện và tính năng.
* Điều khiển trình duyệt qua WebDriver: Selenium sử dụng WebDriver API để điều khiển trình duyệt từ bên ngoài, mô phỏng hành vi người dùng thật như click, nhập văn bản, kéo thả…
* Hỗ trợ nhiều ngôn ngữ lập trình: Selenium cho phép viết kịch bản kiểm thử bằng các ngôn ngữ phổ biến như Java, Python, C#, JavaScript, Ruby…, thuận tiện cho việc tích hợp vào các dự án khác nhau.
* Tích hợp CI/CD: Selenium dễ dàng tích hợp với các công cụ quản lý kiểm thử và hệ thống CI/CD như Jenkins, GitLab CI, Bamboo…, hỗ trợ kiểm thử liên tục.
* Thư viện hỗ trợ phong phú: Với cộng đồng người dùng đông đảo, Selenium có rất nhiều thư viện và tài liệu hướng dẫn, plugin, giúp mở rộng tính năng kiểm thử và gỡ lỗi.
* Có thể kết hợp với nhiều công cụ khác: Selenium có thể kết hợp với TestNG, JUnit (Java) hoặc PyTest (Python) để tổ chức và quản lý các test case, đồng thời tích hợp với Allure hoặc ExtentReports để tạo báo cáo kết quả kiểm thử chuyên nghiệp.

Ứng dụng trong website Sách Tao Đàn:

* Kiểm tra chức năng đăng nhập/đăng ký người dùng.
* Kiểm tra tìm kiếm sách theo từ khóa, danh mục.
* Kiểm tra thêm sách vào giỏ hàng, thanh toán.
* Kiểm tra xem chi tiết thông tin sách.
* Kiểm tra tính năng thêm sách yêu thích.
* Kiểm tra khả năng tương thích giao diện trên nhiều trình duyệt.

2.2.2. Jmeter

JMeter là một công cụ kiểm thử hiệu năng mã nguồn mở, được phát triển bởi Apache Software Foundation, viết hoàn toàn bằng ngôn ngữ Java. Công cụ này được thiết kế chủ yếu để thực hiện kiểm thử hiệu suất, kiểm thử tải và kiểm thử hành vi của các ứng dụng web, dịch vụ web, cũng như nhiều giao thức mạng khác.

Đặc điểm và tính năng chính:

* Kiểm thử hiệu suất và tải trọng: JMeter được sử dụng để đo lường khả năng chịu tải của ứng dụng hoặc dịch vụ web khi phải xử lý nhiều người dùng đồng thời.
* Hỗ trợ nhiều giao thức: Bao gồm HTTP, HTTPS, FTP, JDBC, SMTP, SOAP, REST, WebSocket…
* Giao diện đồ họa thân thiện: Cung cấp giao diện trực quan giúp người dùng dễ dàng thiết kế, quản lý và theo dõi các kịch bản kiểm thử.
* Kiểm thử phân tán (Distributed Testing): Hỗ trợ thực hiện kiểm thử trên nhiều máy tính để mô phỏng lượng tải lớn, giúp đánh giá hiệu suất hệ thống trong môi trường thực tế.
* Tính năng ghi lại kịch bản (Recording): Cho phép ghi lại các hoạt động người dùng trên trình duyệt và tự động tạo ra kịch bản kiểm thử tương ứng.
* Tạo báo cáo và phân tích kết quả: JMeter cung cấp báo cáo dạng biểu đồ, bảng số liệu và xuất file giúp người dùng dễ dàng phân tích hiệu suất, phát hiện bottleneck và vấn đề tiềm ẩn.
* Hỗ trợ mở rộng bằng plugin: Cho phép tích hợp thêm các plugin để mở rộng tính năng, bổ sung giao thức hoặc định dạng báo cáo mới.

Các loại kiểm thử hiệu suất với JMeter:

* Load Testing (Kiểm tra chịu tải): Đánh giá khả năng của hệ thống khi xử lý số lượng lớn yêu cầu đồng thời.
* Stress Testing (Kiểm tra áp lực): Kiểm tra hệ thống khi vượt quá ngưỡng tải thông thường để đánh giá mức độ ổn định và khả năng phục hồi.
* Concurrency Testing (Kiểm thử đồng thời): Đánh giá khả năng xử lý song song nhiều kết nối và yêu cầu của hệ thống.
* Endurance Testing (Kiểm tra bền vững): Kiểm tra hệ thống hoạt động ổn định trong thời gian dài với tải liên tục.
* Spike Testing (Kiểm tra tăng đột biến): Kiểm tra khả năng hệ thống xử lý khi tải người dùng tăng đột ngột trong thời gian ngắn.

2.3. Các kiến trúc của các Framework kiểm thử tự động

2.3.1. Định nghĩa

Framework là tập hợp các đoạn mã, thư viện, mô hình thiết kế, API và các quy tắc đã được đóng gói sẵn nhằm hỗ trợ lập trình viên hoặc kiểm thử viên trong quá trình phát triển và kiểm thử phần mềm. Framework giúp tiêu chuẩn hóa quy trình, giảm thiểu thời gian và công sức cho các thao tác lặp lại, đồng thời đảm bảo tính nhất quán và dễ bảo trì cho dự án.

Nếu ví von, framework giống như bộ khung nhà đã xây sẵn nền móng, cột trụ — lập trình viên và tester chỉ cần xây dựng tiếp phần nội thất, trang trí và chức năng theo ý muốn mà không phải thiết kế lại từ đầu.

2.3.2 Ưu, nhược điểm

**+ Ưu điểm**

|  |  |
| --- | --- |
| Ưu điểm | Mô tả |
| * Tiết kiệm thời gian và công sức | Framework cung cấp sẵn các thành phần, thư viện và công cụ giúp lập trình viên, kiểm thử viên không phải xây dựng mọi thứ từ đầu.  Hạn chế thao tác lặp lại và chuẩn hóa quy trình |
| Tăng hiệu suất làm việc | Giúp tăng tốc quá trình phát triển và kiểm thử phần mềm nhờ các tính năng hỗ trợ và các công cụ tích hợp sẵn. |
| Tính tái sử dụng cao | Các đoạn mã và thành phần trong framework có thể tái sử dụng nhiều lần cho các dự án khác nhau. |
| Giảm thiểu lỗi phát sinh | Do sử dụng các thành phần chuẩn hóa, đã được kiểm tra kỹ lưỡng nên giảm thiểu rủi ro lỗi và tăng độ ổn định của sản phẩm.  Hỗ trợ bảo trì, mở rộng dễ dàng  Kiến trúc rõ ràng, phân tách các lớp, các module giúp việc bảo trì và mở rộng hệ thống dễ dàng hơn. |
| Cộng đồng hỗ trợ lớn | Hầu hết các framework phổ biến đều có cộng đồng người dùng và tài liệu phong phú, dễ tìm kiếm khi gặp sự cố hoặc cần học hỏi. |

**+ Nhược điểm**

|  |  |
| --- | --- |
| Nhược điểm | Mô tả |
| Hạn chế về tính linh hoạt | Do phải tuân theo kiến trúc và quy tắc của framework nên đôi khi khó tùy biến cho các dự án quá đặc thù. |
| Tốn thời gian học tập ban đầu | Để sử dụng thành thạo framework, lập trình viên và kiểm thử viên phải dành thời gian tìm hiểu cấu trúc, cách vận hành và các tính năng của framework đó. |
| Hiệu năng bị ảnh hưởng nếu framework cồng kềnh | Một số framework có thể làm giảm hiệu suất ứng dụng nếu không tối ưu hóa tốt hoặc quá nặng nề. |
| Phụ thuộc vào framework | Khi đã xây dựng hệ thống dựa trên một framework cụ thể, việc chuyển sang framework khác có thể gây khó khăn, mất thời gian và tốn kém |
| Cập nhật và tương thích | Các framework thường xuyên nâng cấp phiên bản. Nếu không theo kịp, có thể dẫn đến lỗi tương thích với dự án cũ. |

2.3.3. Một số framework

- Linear Scripting Framework

Linear Scripting Framework là một khung tự động hóa kiểm tra mức cơ bản ở dạng 'Ghi và Phát lại' theo kiểu tuyến tính.

Khung này còn được gọi là khung 'Ghi và Phát lại'.

Loại khuôn khổ này được sử dụng để kiểm tra các ứng dụng có kích thước nhỏ.

Trong loại này, việc tạo và thực thi các tập lệnh thử nghiệm được thực hiện riêng lẻ cho từng trường hợp thử nghiệm riêng lẻ.

Người kiểm tra nắm bắt từng bước kiểm tra như duyệt, điều hướng, đầu vào của người dùng, thực thi các điểm kiểm tra. Sau đó, người kiểm tra sẽ phát các tập lệnh để thực hiện các bài kiểm tra.

* Module-Based Testing Framework

Với framework này, thì ta sẽ xây dựng các test script độc lập, tương ứng với từng module, các compoment hoặc các function của phần mềm ứng dụng. Việc tránh sử dụng các script phụ thuộc nhau là một yếu tố quan trọng đối với sự ổn định và khả năng bảo trì của framework này.

Nói một cách dễ hiểu, người kiểm thử chia ứng dụng thành nhiều mô-đun và tạo các tập lệnh kiểm tra riêng lẻ. Các tập lệnh thử nghiệm riêng lẻ này có thể được kết hợp để tạo ra các tập lệnh thử nghiệm lớn hơn bằng cách sử dụng một tập lệnh chính để đạt được các kịch bản yêu cầu. Tập lệnh chính này được sử dụng để gọi các mô-đun riêng lẻ để chạy các kịch bản thử nghiệm từ đầu đến cuối.

Lý do chính để sử dụng khuôn khổ này là xây dựng một lớp trừu tượng để bảo vệ mô-đun chính khỏi bất kỳ thay đổi nào được thực hiện trong các thử nghiệm riêng lẻ.

* Library Architecture TesitngFramework

Khung kiểm tra kiến ​​trúc thư viện hay còn gọi là “Kịch bản có cấu trúc” hoặc “Phân rã chức năng” nó dựa trên khuôn khổ mô-đun (Module Based Framework) với một số lợi thế bổ sung.

Trong khuôn khổ kiểm thử mô-đun, chúng ta chia ứng dụng đang được kiểm tra thành các mô-đun trong khi ở đây chúng ta xác định các nhiệm vụ chung và nhóm chúng thành các chức năng. Khi các chức năng được nhóm lại thì các nhóm này sẽ được lưu giữ trong thư viện. Các kịch bản thử nghiệm sử dụng lại các thư viện này để tạo các trường hợp thử nghiệm mới.

2.4. Test Script và phương pháp viết Test Script

2.4.1. Định nghĩa

**Test Script** là một tập hợp các câu lệnh chi tiết được lập trình nhằm tự động hóa quy trình kiểm thử phần mềm. Mục đích của Test Script là mô phỏng và thực hiện các thao tác của người dùng trên phần mềm để kiểm tra tính đúng đắn, độ ổn định và khả năng phản hồi của hệ thống.

**Đặc điểm của Test Script:**

* Thường được lập trình bằng ngôn ngữ hỗ trợ của công cụ kiểm thử (Java, Python, C#, Groovy…)
* Tự động hóa các hành động như click, nhập liệu, chuyển trang, xác minh kết quả…
* Chuyển các bước Test Case thủ công thành mã hóa tự động

2.4.2. Mục tiêu của việc viết Test Script

* Tự động hóa kiểm thử lặp lại, giảm thiểu công việc thủ công.
* Tăng hiệu quả và tính chính xác trong quá trình kiểm thử.
* Giảm chi phí và nhân sự kiểm thử, đặc biệt với các dự án lớn, nhiều môi trường.
* Hỗ trợ kiểm thử hồi quy (Regression Testing) nhanh chóng khi có phiên bản cập nhật.
* Tích hợp vào quy trình CI/CD (Continuous Integration/Continuous Delivery), giúp kiểm thử liên tục.
* Đảm bảo chất lượng phần mềm ở mức cao nhất nhờ phát hiện lỗi sớm.

2.4.3. Phuong pháp viết Test Script

CHƯƠNG 3: PHÂN TÍCH YÊU CẦU VÀ LẬP KẾ HOẠCH KIỂM THỬ

3.1. Phân tích yêu cầu kiểm thử

3.1.1 Yêu cầu chức năng

3.1.2 Yêu cầu phi chức năng

3.2. Kế hoạch kiểm thử

3.2.1. Phương pháp kiểm thử

3.2.2. Môi trường kiểm thử

3.2.3. **Lịch trình và phân công công việc**

CHƯƠNG 4: TRIỂN KHAI KIỂM THỬ ỨNG DỤNG

4.1. Thiết kế kịch bản kiểm thử

4.1.1. Thiết kế kịch bản Kiểm thử chức năng

4.1.2. Thiết kế kịch bản Kiểm thử giao diện

4.1.3. Thiết kế kịch bản Kiểm thử hiệu xuất

4.2. Xây dựng và thực thi các trường hợp kiểm thử

4.2.1. Xây dựng và thực thi các trường hợp Kiểm thử chức năng

4.2.2. Xây dựng và thực thi các trường hợp Kiểm thử giao diện

4.2.3. Xây dựng và thực thi các trường hợp Kiểm thử hiệu xuất

4.2.4. Xây dựng và thực thi các trường hợp Kiểm thử hướng dữ liệu/Từ khóa (Nếu có)

4.3. Báo cáo và phân tích kết quả kiểm thử

4.3.1. Báo cáo kiểm thử chức năng

4.3.2. Báo cáo kiểm thử giao diện

4.3.3. Báo cáo kiểm thử hiệu suất

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết quả đạt được của đề tài

5.2. Hạn chế của đề tài

5.3. Hướng phát triển của đề tài

TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Tên tác giả (năm XB), Tên sách, NXB, Nơi XB |
| [2] | ……….. |