Nhóm-4 Kiểm-thử-phần-mềm-an-toàn

Nhập môn công nghệ phần mềm (Trường Đại học Bách khoa Hà Nội)

BAN CƠ YẾU CHÍNH PHỦ

HỌC VIỆN KỸ THUẬT MẬT MÃ

KHOA AN TOÀN THÔNG TIN

BÁO CÁO

Môn: Chuyên Đề Cơ Sở

Đề tài: KIỂM THỬ PHẦN MỀM AN TOÀN

Giảng viên hướng dẫn: **Th. Thái Thị Thanh Vân**

Sinh viên thực hiện:

1. Nguyễn Cao Phi	AT170136
2. Nguyễn Ngọc Anh	AT170103
3. Nguyễn Đình Hoàng Anh	AT170303

Khóa: AT17

Hà Nội, 2023

MỤC LỤC

DANH MỤC HÌNH

LỜI NÓI ĐẦU

Hiện nay, trong xu hướng phát triển kinh tế công nghiệp hóa, hiện đại hóa, trong thời đại mà công nghệ thông tin đã có những bước tiến vượt bậc và hỗ trợ phần lớn các hoạt động của con người như nghiên cứu khoa học, thương mại, tìm kiếm thông tin,.... thì lĩnh vực công nghệ phần mềm đang ngày một chiếm vị trí quan trọng. Cùng với sự phát triển của công nghệ phần mềm, không thể tránh khỏi lỗi phần mềm và chất lượng phần mềm. Thực tế chứng minh, kiểm thử phần mềm là giai đoạn chiếm đến hơn 30%-70% thời gian, kinh phí và nguồn nhân lực phát triển dự án phần mềm (tùy theo loại phần mềm và lĩnh vực). Tuy nhiên ở Việt Nam hiện nay, việc kiểm thử phần mềm vẫn chưa thực sự được nhìn nhận đúng với tầm quan trọng của nó. Điều này thể hiện ở tỷ lệ kỹ sư kiểm thử phần mềm ở Việt Nam còn khá thấp, cứ 5 lập trình viên thì mới có 1 kỹ sư kiểm thử , trong khi tỷ lệ này theo chuẩn quốc tế là 3:1. Thêm vào đó, mức độ đáp ứng của kỹ sư kiểm thử phần mềm ở Việt Nam chưa cao. Nguyên nhân của việc này đến từ sự thiếu hụt các đơn vị đào tạo chuyên sâu về kiểm thử và nguyên nhân sâu xa vẫn là vấn đề kiểm thử phần mềm ở Việt Nam vẫn chưa được chuyên nghiệp hóa và đầu tư đúng mức. Ngày nay, tự động hóa đang được nghiên cứu và ứng dụng trong nhiều lĩnh vực trong đó công nghệ phần mềm nói chung và kiểm thử phần mềm nói riêng cũng không ngoại lệ. Khi mà kiểm thử phần mềm vẫn tiêu tốn một lượng lớn thời gian, kinh phí và nhân lực trong một dự án phần mềm thì song song với kiểm thử truyền thống thủ công, sự ra đời của các công cụ hỗ trợ kiểm thử tự động như Quick Test Professional, Nunit, Junit, Load Runner ... là tất yếu. Selenium là một công cụ kiểm thử các ứng dụng web có khá nhiều ưu điểm như có thể kiểm thử trên nhiều trình duyệt, hỗ trợ nhiều ngôn ngữ lập trình, giao tiếp được với các công cụ kiểm thử khác như Junit, TestNG (với Java) hay Nunit (với C#), và ưu điểm đặc biệt của công cụ này là nó là một bộ mã nguồn mở, do đó các tổ chức sẽ không tốn kinh phí mua bản quyền. Tuy chưa được ứng dụng nhiều trong các tổ chức ở Việt Nam, song với Tìm hiểu về kiểm thử tự động và ứng dụng kiểm thử website sử dụng công cụ kiểm thử tự đông Selenium. Những ưu điểm trên, Selenium hứa hẹn sẽ

ngày càng phát triển và trở lên thông dụng hơn trong các tổ chức phát triển phần mềm ở nước ta.

Quá trình kiểm thử phần mềm an toàn có ý nghĩa rất quan trọng trong việc đảm bảo chất lượng và độ tin cậy của phần mềm. Mục đích chính của kiểm thử phần mềm an toàn là tìm ra các lỗ hổng bảo mật của phần mềm và đảm bảo rằng phần mềm hoạt động đúng như mong đợi trong mọi trường hợp, đảm bảo tính năng và hiệu suất của phần mềm, từ đó đảm bảo độ tin cậy của phần mềm và đáp ứng được yêu cầu của người dùng.

Xuất phát từ khía cạnh này và với mong muốn có cái nhìn xác thực, rõ ràng hơn về kiểm thử phần mềm, chúng em quyết định đề tài bài tập lớn cho môn Kiểm thử phần mềm là "Kiểm thử website bằng công cụ Selenium". Trong khuôn khổ bài tập lớn, do thời gian và kinh nghiệm thực tế còn hạn chế nên có những phần thực hiện chưa được tốt, chúng em rất mong nhận được sự góp ý của thầy và các bạn.

LỜI CẢM ƠN

Báo cáo chuyên đề cơ sở chuyên ngành là kết quả tìm hiểu và nghiên cứu của nhóm 86 chúng em. Để có thể thực hiện và hoàn thành báo cáo này, em đã nhận được sự hướng dẫn và giúp đỡ nhiệt tình của các thầy cô và bạn bè trong Học viện Kỹ thuật Mật Mã. Nhóm em xin chân thành cảm ơn đến các thầy cô đã tận tình dạy bảo, truyền đạt kiến thức cho chúng em trong suốt quá trình học tập.

Đặc biệt, chúng em xin gửi lời cảm ơn sâu sắc đến cô Thái Thị Thanh Vân– người đã tận tình cung cấp tài liệu, kiến thức và chỉ bảo chúng em trong suốt quá trình thực hiện đề tài này.

Trong quá trình làm đề tài, do thời gian có hạn và trình độ kiến thức, kinh nghiệm còn hạn chế nên chắc chắn không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự góp ý từ thầy, cô để chúng em có thể học hỏi thêm nhiều điều, hoàn thiện hơn bài báo cáo của mình cũng như hoàn thành tốt hơn các bài báo cáo sắp tới.

Chúng em xin chân thành cảm ơn!

Nhóm sinh viên thực hiện
Nguyễn Cao Phi
Nguyễn Ngọc Anh
Nguyễn Đình Hoàng Anh

PHẦN I. TỔNG QUAN VỀ ĐỀ TÀI

1.1. Lý do chọn đề tài

"Lỗi phần mềm là chuyện hiển nhiên trong cuộc sống. Chúng ta có cố gắng đến mức nào thì thực tế là ngay cả những lập trình viên xuất sắc nhất cũng không có thể lúc nào cũng viết được những đoạn mã không có lỗi. Tính trung bình, ngay cả một lập trình viên loại tốt thì cũng có từ 1 đến 3 lỗi trên 100 dòng lệnh. Người ta ước lượng rằng việc kiểm tra để tìm ra các lỗi này chiếm phân nửa khối lượng công việc phải làm để có được một phần mềm hoạt động được".

(Software Testing Techniques, Second Edition, by Boris Beizer, Van Nostrand Reinhola, 1990, ISBN 1850328803).

Thật vậy, ngày nay càng ngày các chương trình (các phần mềm) càng trở lên phức tạp và đồ sộ. Việc tạo ra một sản phẩm có thể bán được trên thị trường đòi hỏi sự nỗ lực của hàng chục, hàng trăm thậm chí hàng ngàn nhân viên. Số lượng dòng mã lên đến hàng triệu. Và để tạo ra một sản phẩm thì không phải chỉ do một tổ chức đứng ra làm từ đầu đến cuối, mà đòi hỏi sự liên kết, tích hợp của rất nhiều sản phẩm, thư viện lập trình, của nhiều tổ chức khác nhau Từ đó đòi hỏi việc kiểm nghiệm phần mềm càng ngày càng trở nên rất quan trọng và rất phức tạp.

Song song với kiểm thử truyền thống thủ công, sự ra đời của các công cụ hỗ trợ kiểm thử tự động như Quick Test Professional, Nunit, Junit, . . . là tất yếu.

Selenium là bộ kiểm thử tự động miễn phí(mã nguồn mở) dành cho các ứng dụng web trên các trình duyệt và nền tảng khác nhau. Với Selenium cùng một số công cụ hỗ trợ khác, kiểm thử viên có thể phát triển thành các framework hỗ trợ cho viết các kịch bản kiểm thử và chạy các kịch bản này một cách tự động, giảm nguồn lực, tăng độ tin cậy và nhàm chán của công việc kiểm thử.

1.2. Mục tiêu

- Nắm được lý thuyết kiểm thử phần mềm, kiểm thử tự động phần mềm.
- Nắm được lý thuyết về công cụ kiểm thử tự động Selenium.
- Ứng dụng được công cụ Selenium Webdriver vào kiểm thử website

1.3. Giới hạn và phạm vi của đề tài

- Tập trung vào lý thuyết kiểm thử và công cụ Selenium.
- Ứng dụng được công cụ selenium Webdriver vào kiểm thử website.

1.4. Nội dung thực hiện

- Trình bày được lý thuyết kiểm thử phần mềm, kiểm thử website.
- Trình bày được lý thuyết về công cụ kiểm thử tự động Selenium.
- Ứng dụng được bô công cụ kiểm thử tự động Selenium vào kiểm thử website.
- Tìm hiểu về kiểm thử tự động và ứng dụng kiểm thử tự động website sử dụng công cụ kiểm thử Selenium.

1.5. Phương pháp tiếp cận

Sử dụng các phương pháp nghiên cứu:

- Phương pháp đọc tài liệu.
- Phương pháp phân tích mẫu.
- Phương pháp thực nghiệm (làm giảm bớt thời gian kiểm thử sau này.)

PHẦN II. TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM

2.1. Tổng quan về phần mềm

2.1.1. Khái niệm

Phần mềm là một tập hợp các chương trình, ứng dụng và tài liệu được thiết kế để thực hiện một hoặc nhiều chức năng hoặc công việc trên một máy tính hoặc các thiết bị điện tử khác. Phần mềm có thể được cài đặt và chạy trên các hệ điều hành khác nhau như Windows, macOS, Linux, iOS, Android và nhiều hệ điều hành khác.

Phần mềm là một chương trình được thiết kế để thực hiện các chức năng nhất định trên một máy tính hoặc thiết bị điện tử khác. Khi phần mềm được chạy, nó được tải lên bộ nhớ của máy tính và thực hiện các chức năng được thiết kế. Phần mềm có thể tương tác với các phần khác của hệ thống, hiển thị thông tin trên màn hình và tương tác với người dùng thông qua giao diện người dùng

Phần mềm được phát triển để đáp ứng các nhu cầu của người dùng, từ việc quản lý dữ liệu, tạo tài liệu, đến giải trí và giáo dục. Phần mềm có thể được phát triển bởi các nhà sản xuất phần mềm hoặc các nhà phát triển độc lập, và sau đó được phân phối và sử dụng bởi người dùng cuối.

2.1.2. Đặc trưng

Một số đặc trưng của phần mềm bao gồm:

- *Tính tương thích*: phần mềm cần có khả năng hoạt động trên các nền tảng và thiết bị khác nhau mà không gây ra sự cố.
- *Tính ổn định*: phần mềm cần hoạt động một cách ổn định và không gây ra các lỗi, sự cố hoặc đóng ứng dụng đột ngột.
- *Tính bảo mật*: phần mềm cần được thiết kế để bảo vệ dữ liệu và thông tin người dùng khỏi các cuộc tấn công và đánh cắp.
- *Tính linh hoạt*: phần mềm cần có khả năng thích nghi với các tình huống khác nhau và có thể được cấu hình để đáp ứng nhu cầu của người dùng.

- *Tính dễ sử dụng*: phần mềm cần có giao diện người dùng thân thiện và dễ sử dụng, giúp người dùng tương tác với phần mềm một cách dễ dàng và hiệu quả.
- *Tính mở rộng*: phần mềm cần có khả năng mở rộng để đáp ứng nhu cầu của người dùng và có thể được cải tiến hoặc phát triển thêm vào sau này.
- *Tính hiệu suất*: phần mềm cần có khả năng hoạt động một cách nhanh chóng và hiệu quả để đáp ứng nhu cầu của người dùng

2.1.3. Lỗi phần mềm

2.1.3.1. Thế nào là lỗi phần mềm

Có rất nhiều định nghĩa khác nhau về lỗi phần mềm, nhưng tựu chung, có thể phát biểu một cách tổng quát: "Lỗi phần mềm là sự không khớp giữa chương trình và đặc tả của nó."

Một hình thức khác nữa cũng được xem là lỗi, đó là phần mềm khó hiểu, khó sử dụng, chậm hoặc dễ gây cảm nhận rằng phần mềm hoạt động không đúng.

2.1.3.2. Các nguyên nhân gây ra lỗi phần mềm

Khác với sự cảm nhận thông thường, lỗi xuất hiện nhiều nhất không phải do lập trình. Nhiều nghiên cứu đã được thực hiện trong các dự án từ rất nhỏ đến các dự án rất lớn và kết quả luôn giống nhau. Số lỗi do đặc tả gây ra là nhiều nhất, chiếm khoảng 80%. Có một số nguyên nhân làm cho đặc tả tạo ra nhiều lỗi nhất. Trong nhiều trường hợp, đặc tả không được viết ra. Các nguyên nhân khác có thể do đặc tả không đủ cẩn thận, nó hay thay đổi, hoặc do chưa phối hợp tốt trong toàn nhóm phát triển. Sự thay đổi yêu cầu của khách hàng cũng là nguyên nhân dễ gây ra lỗi phần mềm. Khách hàng thay đổi yêu cầu không cần quan tâm đến những tác động sau khi thay đổi yêu cầu như phải thiết kế lại, lập lại kế hoạch, làm lại những việc đã hoàn thành. Nếu có nhiều sự thay đổi, rất khó nhận biết hết được phần nào của dự án phụ thuộc và phần nào không phụ thuộc vào sự thay đổi. Nếu không giữ được vết thay đổi rất dễ phát sinh ra lỗi.

Nguồn gây ra lỗi lớn thứ hai là thiết kế. Đó là nền tảng mà lập trình viên dựa vào để nỗ lực thực hiện kế hoạch cho phần mềm. Lỗi do lập trình gây ra cũng khá dễ hiểu. Thời kì đầu, phát triển phần mềm có nghĩa là lập trình, công việc lập trình thì nặng nhọc, do đó lỗi do lập trình gây ra là chủ yếu. Ngày nay, công việc lập trình chỉ là một phần việc của quá trình phát triển phần mềm, cộng với sự hỗ trợ của nhiều công cụ lập trình cao cấp, việc lập trình trở nên nhẹ nhàng hơn, mặc dù độ phức tạp phần mềm lớn hơn rất nhiều. Do đó, lỗi do lập trình gây ra cũng ít hơn. Tuy nhiên, nguyên nhân để lập trình tạo ra lỗi lại nhiều hơn. Đó là do độ phức tạp của phần mềm, do tài liệu nghèo nàn, do sức ép thời gian hoặc chỉ đơn giản là những lỗi "không nói lên được". Một điều cũng hiển nhiên là nhiều lỗi xuất hiện trên bề mặt lập trình nhưng thực ra lại do lỗi của đặc tả hoặc thiết kế.

Ngoài ra còn có các nguyên nhân như: Hiểu sai yêu cầu, thất bại trong giao tiếp giữa người phát triển và khách hàng, lỗi logic trong thiết kế, lỗi mã hóa, không tuân theo tài liệu và cấu trúc code, lỗi công cụ phát triển phần mềm…

2.1.3.3. Các yếu tố ảnh hưởng tới chất lượng phần mềm

- Quy trình phát triển: quy trình phát triển phần mềm sẽ ảnh hưởng đến chất lượng của phần mềm. Một quy trình phát triển phần mềm tốt, được thực hiện đúng cách, có thể giúp đảm bảo chất lượng sản phẩm.
- Thiết kế: Thiết kế phần mềm phải được thực hiện một cách chặt chẽ và rõ ràng để đảm bảo tính tương thích, tính bảo mật, tính ổn định, tính linh hoạt và tính mở rộng của phần mềm.
- Kiểm thử: Việc kiểm thử phần mềm được thực hiện đúng cách và đầy đủ cũng là yếu tố quan trọng để đảm bảo chất lượng phần mềm. Quá trình kiểm thử giúp phát hiện và sửa lỗi trước khi phần mềm được triển khai.
- Kinh nghiệm phát triển: Kinh nghiệm của các nhà phát triển phần mềm cũng ảnh hưởng đến chất lượng phần mềm. Các nhà phát triển có kinh nghiệm sẽ có khả năng giải quyết các vấn đề phức tạp hơn và tạo ra phần mềm chất lượng cao hơn.

- Sử dụng công nghệ phù hợp: Sử dụng công nghệ phù hợp cũng là yếu tố quan trọng để đảm bảo chất lượng phần mềm. Sử dụng công nghệ mới nhất và phù hợp sẽ giúp tăng tính linh hoạt, hiệu suất và tính mở rộng của phần mềm.
- Đội ngũ phát triển: Đội ngũ phát triển phần mềm cũng ảnh hưởng đến chất lượng phần mềm. Một đội ngũ phát triển tốt, có kiến thức chuyên môn và kinh nghiệm sẽ giúp tạo ra phần mềm chất lượng cao.

2.2. Tổng quan kiểm thử phần mềm

2.2.1. Khái niệm

Kiểm thử là gì? Kiểm thử (testing) là quá trình đánh giá một phần mềm hoặc hệ thống để đảm bảo tính đúng đắn, đầy đủ chức năng và đáp ứng được các yêu cầu của người dùng. Nó bao gồm việc thiết kế và thực thi các ca kiểm thử, đánh giá kết quả kiểm thử và xác định các lỗi hoặc vấn đề cần sửa chữa. Kiểm thử được coi là một phần quan trọng trong quy trình phát triển phần mềm để đảm bảo chất lượng và độ tin cậy của phần mềm trước khi nó được phát hành cho khách hàng sử dụng.

Kiểm thử phần mềm là gì?

Kiểm thử phần mềm là quá trình khảo sát một hệ thống hay thành phần dưới những điều kiện xác định, quan sát và ghi lại các kết quả, và đánh giá một khía cạnh nào đó của hệ thống hay thành phần đó. (Theo *Bảng chú giải thuật ngữ chuẩn IEEE của Thuật ngữ kỹ nghệ phần mềm- IEEE Standard Glossary of Software Engineering Terminology*).

Kiểm thử phần mềm là quá trình thực thi một chương trình với mục đích tìm lỗi. (Theo *"The Art of Software Testing" – Nghệ thuật kiểm thử phần mềm*).

Kiểm thử phần mềm là hoạt động khảo sát thực tiễn sản phẩm hay dịch vụ phần mềm trong đúng môi trường chúng dự định sẽ được triển khai nhằm cung cấp cho người có lợi ích liên quan những thông tin về chất lượng của sản phẩm hay dịch vụ phần mềm ấy. Mục đích của kiểm thử phần mềm là tìm ra các lỗi hay khiếm khuyết phần mềm nhằm đảm bảo hiệu quả hoạt động tối ưu của phần mềm trong nhiều ngành khác nhau (theo *Bách khoa toàn thư mở Wikipedia*).

Kiểm thử phần mềm an toàn là gì? Kiểm thử phần mềm an toàn (Safety Testing) là quá trình kiểm tra và đánh giá tính an toàn của phần mềm để đảm bảo rằng phần mềm hoạt động đúng đắn và không gây ra nguy hiểm cho con người, môi trường hoặc tài sản. Mục tiêu chính của kiểm thử phần mềm an toàn là đảm bảo rằng các rủi ro liên quan đến sự cố, thương tích hoặc tổn thất có thể xảy ra khi sử dụng phần mềm đã được đánh giá và giảm thiểu tối đa.

Kiểm thử phần mềm an toàn là rất quan trọng trong các ứng dụng đòi hỏi tính an toàn cao như trong các lĩnh vực y tế, hàng không, ô tô, thiết bị y tế, thiết bị điện tử và các ứng dụng liên quan đến an toàn và bảo mật. Việc thực hiện kiểm thử phần mềm an toàn sẽ giúp đảm bảo rằng sản phẩm phần mềm được phát triển đáp ứng đầy đủ các yêu cầu an toàn và tăng cường sự tin tưởng của người dùng và các tổ chức liên quan.

2.2.2. Mục tiêu

Mục tiêu của kiểm thử là đảm bảo tính đúng đắn, chất lượng và độ tin cậy của phần mềm, cũng như đảm bảo rằng phần mềm đáp ứng các yêu cầu của khách hàng, yêu cầu của sản phẩm đã đặt ra. Kiểm thử có thể cung cấp cho doanh nghiệp một quan điểm, một cách nhìn độc lập về phần mềm để từ đó cho phép đánh giá và thấu hiểu được những rủi ro trong quá trình triển khai phần mềm .

Các mục tiêu chính của kiểm thử phần mềm bao gồm:

- Trong thời gian xác định trước, kiểm thử viên tìm được càng nhiều lỗi càng tốt.
- Tìm ra lỗi chưa được phát hiện, tìm một cách sớm nhất và đảm bảo rằng lỗi đã được sửa, mà kiểm thử phần mềm không làm công việc chẩn đoán nguyên nhân gây ra lỗi đã được phát hiện và sửa lỗi trước khi nó được phát hành cho khách hàng sử dụng.
- Thiết kế tài liệu kiểm thử một cách có hệ thống và thực hiện nó sao cho có hiệu quả, nhưng tiết kiệm được thời gian, công sức và chi phí.
- Đảm bảo rằng phần mềm cuối cùng phù hợp với các yêu cầu đặc tả của nó.

- Đảm bảo rằng phần mềm đáp ứng được các yêu cầu chức năng và phi chức năng của khách hàng.
- Đảm bảo rằng phần mềm đáp ứng được các tiêu chuẩn và quy định liên quan đến an ninh thông tin và bảo mật dữ liệu.
- Đo lường chất lượng của sản phẩm và dự án.
- Viết kịch bản kiểm thử (test case) chất lượng cao, thực hiện kiểm thử hiệu quả và đưa ra các báo cáo chính xác.

Tóm lại, mục tiêu của kiểm thử phần mềm là đảm bảo chất lượng, tính đúng đắn và độ tin cậy của phần mềm, giúp đảm bảo rằng khách hàng sử dụng phần mềm có được trải nghiệm tốt nhất và giảm thiểu rủi ro cho doanh nghiệp

2.2.3. Tầm quan trọng

Ý nghĩa của kiểm thử phần mềm là đảm bảo rằng phần mềm đáp ứng được các yêu cầu chức năng và phi chức năng của khách hàng, và đảm bảo tính đúng đắn, chất lượng và độ tin cậy của phần mềm. Kiểm thử phần mềm giúp phát hiện các lỗi và vấn đề trong phần mềm trước khi nó được phát hành cho khách hàng sử dụng, giảm thiểu rủi ro và chi phí trong việc sửa chữa lỗi sau khi phần mềm đã phát hành.

Kiểm thử phần mềm giúp nhanh chóng phát hiện các lỗi của phần mềm, giúp giảm chi phí sửa chữa. Sản phẩm được phát hiện và sửa lỗi giúp loại bỏ các rủi ro và các vấn đề sớm, làm tăng độ tin cậy cho sản phẩm. Đối với ngành công nghệ phần mềm, vấn đề bảo mật là yếu tố cực kỳ nhạy cảm, nó liên quan trực tiếp đến việc sở hữu, sử dụng của người dùng. Vì vậy, việc kiểm thử phần mềm giúp hoàn thiện nhất sản phẩm phần mềm, tránh những lỗ hổng bảo mật đáng tiếc, tăng độ tin tưởng cho người sử dụng.

Ngoài ra, kiểm thử phần mềm còn giúp đảm bảo tính đáp ứng của phần mềm trong mọi trường hợp sử dụng, giúp cải thiện chất lượng sản phẩm và tăng độ tin cậy của phần mềm. Điều này giúp doanh nghiệp xây dựng một danh tiếng tốt và tăng cường niềm tin của khách hàng. Một sản phẩm càng chỉn chu, càng hoàn

thiện, chất lượng càng cao sẽ tạo ra những trải nghiệm người dùng tốt nhất, từ đó càng tạo được niềm tin và uy tín với khách hàng và đối tác.

Cuối cùng, kiểm thử phần mềm giúp đảm bảo tuân thủ các tiêu chuẩn và quy định liên quan đến an ninh thông tin và bảo mật dữ liệu, giảm thiểu nguy cơ lỗ hổng bảo mật và bảo vệ thông tin của khách hàng và doanh nghiệp.

2.2.4. Các nguyên tắc của kiểm thử

Trong kiểm thử phần mềm, có 7 nguyên tắc kiểm thử. Tìm hiểu chúng là 1 điều rất quan trọng bởi vì nó giúp tiết kiệm thời gian cũng như công sức truy lùng lỗi ẩn trong các ứng dụng:

1. Kiểm thử phần mềm chứng minh sự hiện diện của lỗi

Bằng việc kiểm thử, chúng ta có thể làm giảm lượng bugs khi áp dụng nhiều phương pháp kiểm thử lên phần mềm. Tuy nhiên khi được đưa lên môi trường thật, người dùng cuối hoàn toàn có thể thấy nhiều lỗi khác không tìm thấy trong quá trình kiểm thử. Kiểm thử chứng minh được sản phẩm có lỗi nhưng không thể chứng minh rằng sản phẩm không còn lỗi. Điều này có nghĩa là, sẽ luôn có lỗi không được phát hiện trong phần mềm, ngay cả khi không tìm thấy lỗi, cũng không đồng nghĩa rằng phần mềm đúng hoàn toàn.

2. Kiểm thử toàn bộ là không khả thi

Đúng vậy, rất khó để kiểm tra toàn bộ các module cũng như các tính năng, kết hợp với đầu vào và đầu ra trong suốt quá trình kiểm tra. Các sản phẩm phần mềm hiện nay cực kỳ đa dạng và phức tạp, được phát triển trên nhiều nền tảng, thêm vào đó, ngày càng có nhiều công nghệ mới, khả năng kết nối dữ liệu lớn… khiến việc kiểm thử toàn bộ gần như là không thể. Thay vì cố gắng kiểm thử toàn bộ, hãy xác định mức độ quan trọng và độ ưu tiên của các module để kiểm thử những phần cần thiết hoặc gặp nhiều nguy cơ hơn.

3. Kiểm thử càng sớm càng tốt

Nguyên tắc kiểm thử sớm có nghĩa là việc kiểm thử cần được thực hiện càng sớm càng tốt trong vòng đời phát triển phần mềm. Vậy ở bước nào thì được coi là

sớm? Nguyên tắc này cho thấy ta cần phát hiện bug ngay từ các bước đầu tiên như nghiên cứu yêu cầu (requirement) hay design. Phát hiện lỗi càng muộn, chi phí bỏ ra để xử lý càng cao. Vì vậy, việcthay đổi yêu cầu không đúng từ sớm sẽ khiến tốn ít chi phí để thay đổi tính năng hơn.

4. Lỗi thường được phân bố tập trung

Nguyên lý về phân bố lỗi chỉ ra rằng, chỉ một số ít module chứa phần lớn số lỗi phát hiện được. Những module này thường là những thành phần, chức năng chính của hệ thống. Điều này cũng đúng theo nguyên lý Pareto: 80 – 20: 80% số lỗi tìm thấy ở chỉ 20% module. Bằng kinh nghiệm, các QA/ Tester có thể xác định được những module có tính rủi ro và nhiều lỗi như vậy, giúp tập trung tìm kiếm lỗi nhanh và hiệu quả hơn. Tuy nhiên, cách tiếp cận này cũng ẩn chứa vấn đề: nếu thực hiện kiểm thử tương tự lặp đi lặp lại, dễ thấy rằng những test case cũ sẽ khó tìm thêm được bug mới.

5. Nghịch lý thuốc trừ sâu

Trong trồng trọt, nếu người nông dân sử dụng lặp lại một liều trừ sâu, các loại sâu bệnh sẽ dần dần thích nghi và trở nên "nhờn" với loại thuốc trừ sâu đó. Tương tự với kiểm thử phần mềm, khi lặp đi lặp lại một test case, thì xác suất tìm được lỗi là rất thấp. Nguyên nhân là hệ thống hoàn thiện hơn, lỗi tìm thấy đã được sửa theo test case cũ. Để xử lý hiệu ứng "thuốc trừ sâu" này, test case cần được thường xuyên xem lại và chỉnh sửa, thêm nhiều test case mới để tìm lỗi mới (regression test).

Thêm vào đó, QA/ Tester cũng không nên phụ thuộc quá nhiều vào các kỹ thuật test sẵn có. Bạn cần liên tục cải tiến các phương pháp có sẵn để làm việc kiểm thử hiệu quả hơn.

6. Kiểm thử phụ thuộc vào ngữ cảnh

Kiểm thử phụ thuộc vào ngữ cảnh, nói một cách đơn giản, là việc kiểm thử một trang thương mại điện tử sẽ phải khác cách test một ứng dụng đọc tin tức. Tất cả

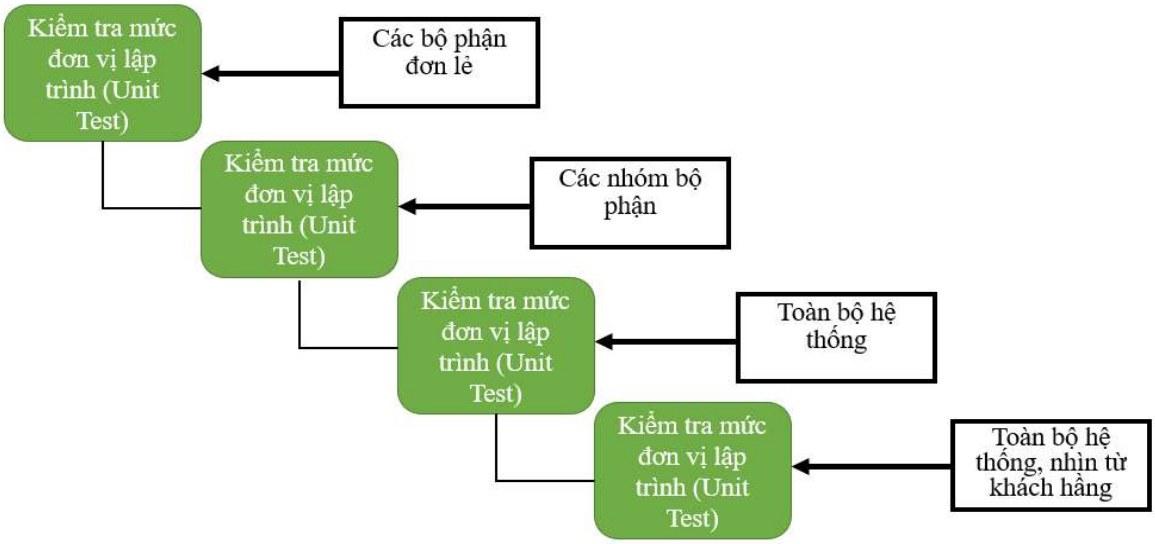
các phần mềm đều được phát triển theo cách khác nhau. Và việc áp dụng chung một "cách giải" là sai lầm. Bạn cần sử dụng cách tiếp cận khác nhau, phương thức, kỹ thuật test khác nhau, loại test phụ thuộc vào loại phần mềm/ ứng dụng/ website.

7. Quan niệm sai lầm về việc "hết lỗi"

Một phần mềm sạch bug 99% vẫn có thể không sử dụng được. Đây là trường hợp phần mềm được kiểm thử bằng một requirement sai. Kiểm thử không chỉ để tìm ra lỗi, mà còn để kiểm tra xem phần mềm có đáp ứng được đúng nhu cầu hay không. Chính vì vậy, việc Không còn lỗi hay Hết lỗi là quan niệm sai lầm.

2.2.5. Phân loại kiểm thử

2.2.5.1. Phân loại dựa vào mục đích, phạm vi kiểm thử



Hình 2.1. 4 cấp độ kiểm thử phần mềm

2.2.5.1.1. Unit Test - Kiểm thử mức đơn vị

Unit Test là một loại kiểm thử phần mềm trong đó các đơn vị hay thành phần riêng lẻ của phần mềm được kiểm thử. Kiểm thử đơn vị được thực hiện trong quá trình phát triển ứng dụng. Nó tập trung vào việc kiểm tra các đơn vị phần mềm nhỏ nhất, gọi là "đơn vị" (unit), để đảm bảo rằng các đơn vị đó hoạt động đúng và phù hợp với thiết kế. Mục tiêu của unit test là đảm bảo rằng từng đơn vị của mã nguồn là đúng và có thể hoạt động độc lập với các đơn vị khác.

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 PTPM. Thông thường, Unit Test đòi hỏi kiểm tra 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 (khỏi Unit) là chính xác, trong mối tương quan với dữ liệu nhập và chức năng của Unit. Điều này thường đòi hỏi 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. Thực tế việc chọn lựa các nhánh để đơn giản hóa việc kiểm tra và quét hết Unit đòi hỏi phải có kỹ thuật, đôi khi phải dùng thuật toán để chọn lựa.

Mỗi Unit Test sẽ gửi đi một thông điệp và kiểm tra câu trả lời nhận được đúng hay không, bao gồm:

- Các kết quả trả về mong muốn
- Các lỗi ngoại lệ mong muốn

Các đoạn mã Unit Test hoạt động liên tục hoặc định kỳ để thăm dò và phát hiện các lỗi kỹ thuật trong suốt quá trình phát triển, do đó Unit Test còn được gọi là kỹ thuật kiểm nghiệm tự động. Unit Test có các đặc điểm sau:

- Đóng vai trò như những người sử dụng đầu tiên của hệ thống.
- Chỉ có giá trị khi chúng có thể phát hiện các vấn đề tiềm ẩn hoặc lỗi kỹ thuật.

Cũng như các mức kiểm tra khác, Unit Test cũng đòi hỏi phải chuẩn bị trước các tình huống (test case) hoặc kịch bản (script), trong đó chỉ định rõ dữ liệu vào, các bước thực hiện và dữ liệu mong chờ sẽ xuất ra. Các test case và script này nên được giữ lại để tái sử dụng.

2.2.5.1.2. Integration Test - Kiểm thử tích hợp

Integration test là một loại kiểm thử phần mềm nhằm kiểm tra tính tương tác giữa các đơn vị của phần mềm, gọi là "đơn vị tích hợp" (integration units). Mục tiêu của integration test là:

- Đảm bảo rằng các đơn vị tích hợp hoạt động đúng và phù hợp với thiết kế, cũng như đảm bảo tính toàn vẹn và độ tin cậy của hệ thống khi được tích hợp với nhau.
- 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).

Trong Unit Test, lập trình viên cố gắng phát hiện lỗi liên quan đến chức năng và cấu trúc nội tại của Unit. Có một số phép kiểm tra đơn giản trên giao tiếp giữa Unit với các thành phần liên quan khác, tuy nhiên mọi giao tiếp liên quan đến Unit thật sự được kiểm tra đầy đủ khi các Unit tích hợp với nhau trong khi thực hiện Integration Test.

Integration test nên được thực hiện sau khi đã hoàn thành unit test và các thành phần của phần mềm đã được đóng gói thành các module hay thành phần con để có thể tích hợp và kiểm thử tích hợp giữa các thành phần đó. Việc thực hiện integration test sớm trong quá trình phát triển phần mềm giúp phát hiện sớm các lỗi tích hợp giữa các thành phần và giảm thiểu số lượng lỗi phát hiện sau khi phần mềm được tích hợp hoàn chỉnh.

Khi thực hiện integration test, có một số điều cần lưu ý để đảm bảo tính hiệu quả và độ chính xác của quá trình kiểm thử như:

- Xác định đầy đủ các thành phần của phần mềm cần tích hợp và kiểm thử tích hợp.
- Thiết kế các ca kiểm thử tích hợp một cách tổng quát, chứ không chỉ dành riêng cho một thành phần cụ thể.
- Đảm bảo rằng dữ liệu đầu vào và kết quả đầu ra sau khi tích hợp là chính xác và đúng với các yêu cầu của phần mềm.
- Kiểm tra tính tương thích của các giao diện giữa các thành phần.
- Xác định các yếu tố độc lập và phụ thuộc để tăng tính hiệu quả của quá trình kiểm thử.

- Lưu trữ kết quả kiểm thử và theo dõi quá trình sửa chữa các lỗi được phát hiện trong quá trình tích hợp.

Trong Integration Test có 4 loại kiểm thử:

- Kiểm thử cấu trúc (structure): chú trọng đến hoạt động của các thành phần cấu trúc nội tại của chương trình.
- Kiểm thử chức năng (functional): 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): kiểm tra việc vận hành của hệ thống.
- Kiểm thử khả năng chịu tải (stress): kiểm tra giới hạn của hệ thống.

2.2.5.1.3. System Test - Kiểm thử hệ thống

System Test là một loại kiểm thử phần mềm, được thực hiện nhằm đánh giá toàn bộ hệ thống phần mềm đã được phát triển. Nó bao gồm việc kiểm tra chức năng, hiệu suất, bảo mật, độ tin cậy và các yêu cầu chất lượng khác của hệ thống.

Mục tiêu của System Test là đảm bảo rằng hệ thống đã đáp ứng được tất cả các yêu cầu chức năng và phi chức năng, và có thể hoạt động một cách ổn định trong môi trường sản xuất thực tế. Kết quả của System Test sẽ được đánh giá để quyết định liệu hệ thống đã sẵn sàng để triển khai hoặc không.

Khi thực hiện system test, có một số lưu ý cần được xem xét để đảm bảo độ chính xác và hiệu quả của quá trình kiểm thử:

- Xác định mục tiêu kiểm thử rõ ràng và đầy đủ để đảm bảo đủ kiểm thử tất cả các chức năng và tính năng của hệ thống.
- Sử dụng các phương pháp kiểm thử chính xác và phù hợp như kiểm thử hộp đen, kiểm thử hộp trắng hoặc kiểm thử hộp xám.
- Đảm bảo rằng tất cả các yêu cầu kỹ thuật đã được thực hiện đúng cách.
- Thiết lập một môi trường kiểm thử giống với môi trường thực tế để đảm bảo độ chính xác và khả năng tái sử dụng của các bài kiểm thử.
- Sử dụng một kế hoạch kiểm thử chi tiết, bao gồm các bước kiểm thử cụ thể để kiểm tra các chức năng và tính năng của hệ thống.

2.2.5.1.4. Acceptance Test - Kiểm thử chấp nhận sản phẩm

Acceptance Test là loại kiểm thử phần mềm nhằm đảm bảo rằng hệ thống phần mềm đã đáp ứng yêu cầu và tiêu chuẩn của khách hàng. Nó được thực hiện bởi người dùng cuối hoặc đại diện của khách hàng để đảm bảo rằng hệ thống phần mềm đã hoàn thành và hoạt động đúng như yêu cầu, cung cấp sự tin cậy.

Acceptance Test góp phần hỗ trợ cho việc phát triển phần mềm:

- Nắm bắt trực tiếp yêu cầu khách hàng và cân nhắc xem liệu yêu cầu đó có phù hợp với cơ sở hệ thống hay không.
- Cho thấy được những phần thiếu sót, lỗi của unit test.
- Cung cấp một thước đo cho biết tình trạng của hệ thống, đảm bảo rằng sản phẩm phần mềm đã được kiểm tra kỹ lưỡng và đáp ứng các tiêu chuẩn chất lượng được đặt ra trước khi được chuyển giao cho khách hàng.

Đối với những sản phẩm dành rộng rãi trên thị trường cho nhiều người sử dụng, thông thường sẽ thông qua hai loại kiểm tra gọi là Alpha Test và Beta Test. Với Alpha Test, người sử dụng (tiềm năng) kiểm tra phần mềm ngay tại nơi phát triển, lập trình viên sẽ ghi nhận các lỗi hoặc phản hồi, và lên kế hoạch sửa. Với Beta Test, phần mềm sẽ được gửi tới cho người sử dụng (tiềm năng) để kiểm tra ngay trong môi trường thực, lỗi hoặc phản hồi cũng sẽ gửi ngược lại cho lập trình viên để sửa chữa. Thực tế cho thấy, nếu khách hàng không tham gia vào quá trình phát triển phần mình thì kết quả Acceptance Test sẽ có sai lệch (có thể rất lớn), mặc dù đã trải qua tất cả các kiểm tra trước đó. Sự sai lệch này liên quan đến việc hiểu sai yêu cầu cũng như sự mong chờ của khách hàng.

Acceptance test có vai trò xác nhận rằng phần mềm đã đáp ứng đầy đủ các yêu cầu và mong đợi của người sử dụng cuối. Nó là một bước quan trọng trong quá trình phát triển phần mềm và giúp đảm bảo rằng phần mềm sẽ hoạt động đúng và đáp ứng được nhu cầu của người dùng, tăng niềm tin của khách hàng vào hệ thống xây dựng.

2.2.5.1.5. Regression Test - Kiểm thử hồi quy

Regression Test thực chất không phải là một mức kiểm thử như các mức đã nói ở trên. Nó đơn thuần kiểm tra lại phần mềm sau khi có thay đổi, để bảo đảm phiên bản phần mềm mới sẽ thực hiện tốt các chức năng như phiên bản cũ và sự thay đổi không gây ra lỗi mới trên những chức năng vốn đã làm việc tốt.

Regression test có thể thực hiện tại mọi mức kiểm thử. Mặc dù không là một mức kiểm tra, thực tế lại cho thấy Regression Test là một trong những loại kiểm tra tốn nhiều thời gian và công sức nhất. Tuy thế, việc bỏ qua Regression Test là "không được phép" vì có thể dẫn đến tình trạng phát sinh hoặc tái xuất hiện những lỗi nghiêm trọng, mặc dù ta "tưởng rằng" những lỗi đó hoặc không có hoặc đã được kiểm tra và sửa chữa rồi.

2.2.5.2. Phân loại dựa vào mức độ tự động

2.2.5.2.1. Manual Test - Kiểm thử thủ công

Manual Testing là 1 trong những công việc theo dạng kiểm thử phần mềm, hoặc là một chương trình được thực hiện bằng tay bởi các tester mà không thông qua bất kỳ công cụ hỗ trợ nào.

Nó hoạt động dựa vào mục đích phát hiện các lỗi bug từ nhỏ cho đến lớn trong phần mềm. Từ đó, đưa ra các định hướng giải quyết để có thể đảm bảo cho phần mềm hoạt động ổn định nhất lúc giao cho khách hàng.

2.2.5.2.2. Automation Test - Kiểm thử tự động

Automation testing (Kiểm thử tự động) là quá trình sử dụng các công cụ, script và phần mềm để thực hiện những trường hợp kiểm thử, bằng cách lặp lại những hành động được xác định trước. Automation testing tập trung vào việc thay thế hoạt động thủ công của con người bằng các hệ thống hoặc thiết bị.

Bởi vì Automation testing được thực hiện thông qua một công cụ tự động hóa, nên nó tiêu tốn ít thời gian hơn trong các thử nghiệm khám phá và hiệu quả hơn trong việc duy trì các script kiểm tra, đồng thời nâng cao phạm vi kiểm tra tổng thể.

Automation testing thích hợp nhất cho các dự án lớn yêu cầu kiểm tra lặp lại các khu vực giống nhau và những dự án đã trải qua quá trình thử nghiệm thủ công ban đầu.

2.2.5.3. Phân loại dựa vào phương pháp kiểm thử

2.2.5.3.1. Static Testing - Kiểm thử tĩnh

Là phương pháp kiểm thử phần mềm đòi hỏi phải duyệt lại các yêu cầu và các đặc tả bằng tay, thông qua việc sử dụng giấy, bút để kiểm tra logic, lần từng chi tiết mà không cần chạy chương trình. Kiểu kiểm thử này thường được sử dụng bởi chuyên viên thiết kế người mà viết mã lệnh một mình.

Kiểm thử tĩnh cũng có thể được tự động hóa. Nó sẽ thực hiện kiểm tra toàn bộ bao gồm các chương trình được phân tích bởi một trình thông dịch hoặc biên dịch mà xác nhận tính hợp lệ về cú pháp của chương trình.

2.2.5.3.2. Dynamic Testing - Kiểm thử động

Là phương pháp kiểm thử phần mềm thông qua việc dùng máy chạy chương trình để điều tra trạng thái tác động của chương trình. Đó là kiểm thử dựa trên các ca kiểm thử xác định bằng sự thực hiện của đối tượng kiểm thử hay chạy các chương trình. Kiểm thử động kiểm tra cách thức hoạt động của mã lệnh, tức là kiểm tra sự phản ứng vật lý từ hệ thống tới các biến luôn thay đổi theo thời gian. Trong kiểm thử động, phần mềm phải thực sự được biên dịch và chạy. Kiểm thử động thực sự bao gồm làm việc với phần mềm, nhập các giá trị đầu vào và kiểm tra xem liệu đầu ra có như mong muốn hay không. Các phương pháp kiểm thử động gồm có kiểm thử Unit Tests, Integration Tests, System Tests, và Acceptance Tests.

2.2.5.4. Phân loại dựa vào kỹ thuật kiểm thử

2.2.5.4.1. Kiểm thử hộp trắng - dựa vào cấu trúc:

Kiểm thử hộp trắng hay còn gọi là kiểm thử hướng logic, cho phép kiểm tra cấu trúc bên trong của phần mềm với mục đích đảm bảo rằng tất cả các câu lệnh và điều kiện sẽ được thực hiện ít nhất một lần. Hộp trắng đúng nghĩa phải gọi là hộp

trong suốt . Chính vì vậy, kỹ thuật này còn có một số tên khác là kiểm thử hộp thuỷ tinh (Glass-Box Testing), hay kiểm thử hộp trong suốt (Clear-Box Testing). Người kiểm thử truy nhập vào mã nguồn chương trình, chọn các giá trị đầu vào và có thể kiểm tra nó, lấy đó làm cơ sở để hỗ trợ việc kiểm thử và so sánh kết quả đầu ra với kết quả dự kiến.

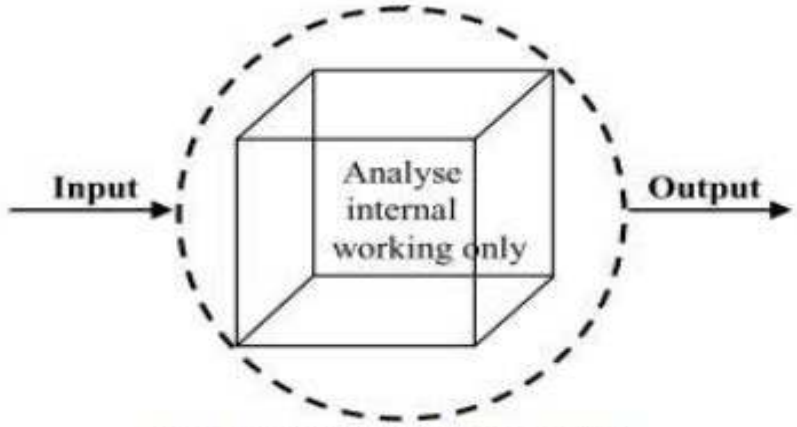


Fig. 4: White-box Testing [22]

Hình 2.2. Kiểm thử hộp trắng

Các phương pháp kiểm thử hộp trắng

- Kiểm thử đường cơ bản – Đồ thị dòng: là phương pháp kiểm tra được phát minh đầu tiên bởi Tom McCabe. Với phương thức kiểm tra này khá giống với đồ thị trong luồng điều khiển của mỗi chương trình.

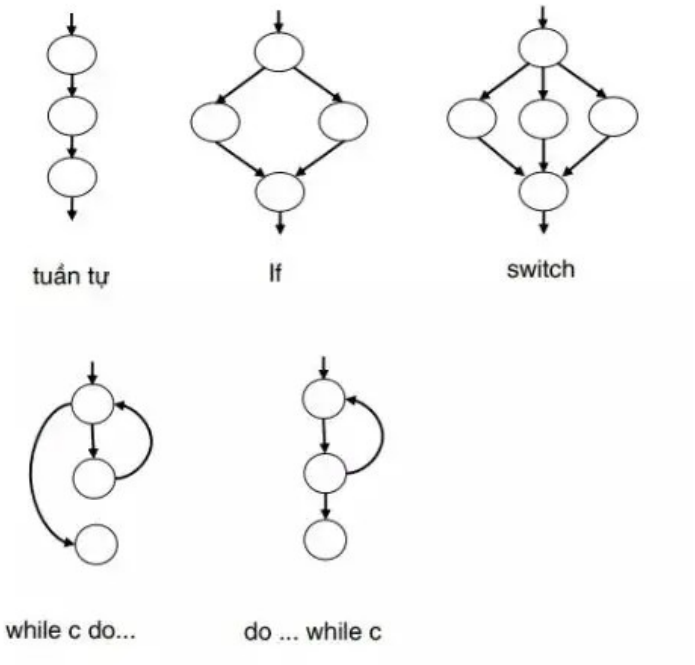
Bên cạnh đó, đây cũng là phương pháp kiểm tra thuật giải được sử dụng phổ biến nhất hiện nay. Với chế độ hiển thị trực quan từ đó các thuật giả và thành phần của mối quan có thể dễ dàng nhìn thấy được.

Thông thường với phương pháp kiểm thử đường cơ bản – đồ thị dòng thì người ta thường chia thành 2 thành phần chính là các cung và các nút kết nối.

- Các thành phần có trong nút điều khiển:

Hình 2.3. Các thành phần có trong nút điều khiển

- Các cấu trúc có trong đồ thị dòng:



Hình 2.4. Các cấu trúc có trong đồ thị dòng

● Kiểm thử dựa trên luồng dữ liệu (Data – flow Testing): là

Kiểm thử dòng dữ liệu (data flow testing) là một kỹ thuật kiểm thử phần mềm tập trung vào việc phân tích các dòng dữ liệu trong chương trình để phát hiện các lỗi tiềm ẩn và các bất thường liên quan đến dữ liệu.

Với quá trình kiểm thử này thông thường sẽ bao gồm quá trình xây dựng phần mềm từ đó kiểm tra các vấn đề xảy ra có liên quan. Sẽ có 2 cách kiểm tra tích hợp trên hệ thống phần mềm đó là:

– Kiểm tra tích hợp theo thứ tự từ trên xuống dưới: Quá trình này bao gồm việc xây dựng hệ thống khung bên trong và quá trình đưa toàn bộ thành phần vào bên trong đó.

– Kiểm tra tích hợp theo thứ tự từ dưới lên trên: Là quá trình kiểm tra các thành phần trong cơ sở và từ đó có thể bổ sung thêm các thành phần khác có liên quan đến chức năng.

● Kiểm thử đột biến (Mutation Testing)

Đây là 1 trong các loại kiểm thử phần mềm khá phổ biến hiện nay với chức năng chính là để kiểm tra các câu lệnh. Đặc biệt là kiểm tra test case và kiểm tra câu lệnh xem có xảy ra lỗi gì hay không.

Hơn thế nữa, với phương pháp kiểm thử này hệ thống còn đưa ra cho người dùng những thiếu sót đang xảy ra và từ đó chương trình phần mềm sẽ hoàn chính nhất.

Quy trình kiểm thử hộp trắng

Tom McCabe đề nghị qui trình kiểm thử TPPM gồm các bước công việc sau:

1. Từ TPPM cần kiểm thử, xây dựng đồ thị dòng điều khiển tương ứng, rồi chuyển thành đồ thị dòng điều khiển nhị phân, rồi chuyển thành đồ thị dòng điều khiển cơ bản.
2. Tính độ phức tạp Cyclomatic của đồ thị (C = P +1).
3. Xác định C đường thi hành tuyến tính độc lập cơ bản cần kiểm thử.
4. Tạo từng test case cho từng đường thi hành tuyến tính độc lập cơ bản.
5. Thực hiện kiểm thử trên từng test case.
6. So sánh kết quả có được với kết quả được kỳ vọng.
7. Lập báo cáo kết quả để phản hồi cho những người có liên quan.

Ưu - nhược điểm: Kiểm thử hộp trắng có thể tối ưu mã hóa nguồn, phát hiện những lỗi ẩn trong các mã nguồn, đảm bảo rằng các thành phần phần mềm hoạt động chính xác theo thiết kế và cũng giúp tăng độ bảo mật, hiệu suất của phần mềm. Tuy nhiên, phương pháp này cũng có hạn chế, bao gồm khó khăn trong việc tìm ra tất cả các lỗi ẩn và giải quyết chúng với thời hạn cho phép; việc bảo trì tốn kém và khó khăn…

2.2.5.4.2. Kiểm thử hộp đen - dựa vào đặc tả

Kiểm thử hộp đen là 1 phương pháp kiểm thử phần mềm được thực hiện mà không biết được cấu tạo bên trong của phần mềm, là cách mà các tester kiểm tra xem hệ thống như một chiếc hộp đen, không có cách nào nhìn thấy bên trong của cái hộp.

- Nó còn được gọi là kiểm thử hướng dữ liệu hay là kiểm thử hướng in/out.
- Nó có thể là kiểm thử chức năng (như kiểm thử tích hợp) hoặc kiểm thử phi chức năng (như kiểm thử hiệu năng), tuy nhiên thường thì nó là kiểm thử chức năng.
- Người kiểm thử nên xây dựng những nhóm giá trị đầu vào mà sẽ thực thi hầu hết đa số các yêu cầu chức năng của chương trình.

Khi thực hiện kỹ thuật kiểm thử này, người kiểm thử không cần quan tâm bên trong hệ thống hoạt động ra sao, không cần hiểu source code thế nào, chỉ cần quan tâm đến việc tìm các hiện tượng mà phần mềm không hành xử theo đúng đặc tả của nó. Thông thường, trong lúc thực hiện kiểm thử hộp đen, tester sẽ tương tác với giao diện người dùng của hệ thống bằng cách cung cấp đầu vào và kiểm tra kết quả đầu ra mà không cần biết cách thức làm việc bên trong của hệ thống.

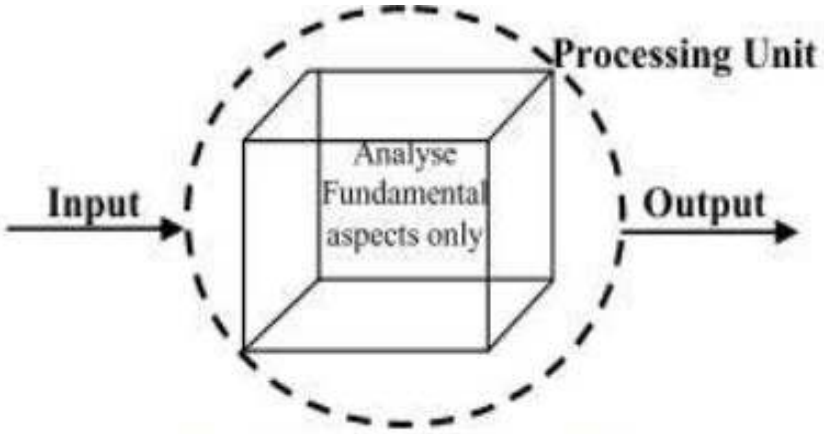


Fig. 5: Black-box Testing [22]

Hình 2.5. Kiểm thử hộp đen

Kiểm thử hộp đen không thay thế kỹ thuật hộp trắng, nhưng nó bổ sung khả năng phát hiện các lớp lỗi khác với các phương pháp hộp trắng như:

- Các chức năng thiếu hoặc không đúng.
- Các lỗi giao diện.
- Các lỗi cấu trúc dữ liệu trong truy cập cơ sở dữ liệu bên ngoài.
- Các lỗi thi hành.
- Các lỗi khởi tạo hoặc kết thúc.

Các phương pháp kiểm thử hộp đen

- Phân lớp tương đương – Equivalence Partitioning.
- Phân tích giá trị biên – Boundary value analysis.
- Kiểm thử mọi cặp – All-pairs testing.
- Kiểm thử fuzz – Fuzz Testing.
- Kiểm thử dựa trên mô hình – Model-based testing.
- Ma trận dấu vết – Traceability Matrix.
- Kiểm thử thăm dò – Exploratory Testing.
- Kiểm thử dựa trên đặc tả – Specification-based testing.

Ưu - nhược điểm: Trong kiểm thử hộp đen, các kiểm thử được thiết kế dựa trên các yêu cầu của khách hàng hoặc các yêu cầu chức năng của sản phẩm. Kiểm thử hộp đen giúp đảm bảo rằng chương trình đáp ứng được các yêu cầu chức năng của khách hàng và cũng giúp xác định các lỗi có thể xảy ra trong các tình huống khác nhau. Phương pháp này phù hợp và hiệu quả khi số lượng các dòng lệnh của hệ thống là lớn, Không cần truy cập mã nguồn, phân biệt rõ ràng quan điểm của người dùng với quan điểm của nhà phát triển thông qua các vai trò được xác định rõ ràng Ngoài ra, một số lượng lớn tester có kỹ năng vừa phải có thể kiểm tra ứng dụng mà không cần có nhiều kiến thức, ngôn ngữ lập trình hoặc hệ điều hành. Mặt khác, nó lại bị giới hạn bởi độ bao phủ của các trường hợp kiểm thử, không cung cấp cho người kiểm thử sự chi tiết về cách thức hoạt động của chương trình và còn rất khó để thiết kế được hầu hết các trường hợp kiểm thử cho hệ thống.

2.2.5.4.3. Kiểm thử hộp xám

Gray Box Testing là một phương pháp kiểm thử phần mềm được kết hợp giữa Phương pháp Kiểm thử Black Box (hộp đen) và White Box (hộp trắng). Trong Kiểm thử hộp đen, Tester kiểm thử các hạng mục mà không cần biết cấu trúc bên trong của nó, còn trong Kiểm thử Hộp trắng thì Tester biết được cấu trúc bên trong của chương trình. Trong Kiểm thử Hộp xám, cấu trúc bên trong sản phẩm chỉ được

biết một phần, Tester có thể truy cập vào cấu trúc dữ liệu bên trong và thuật toán của chương trình với mục đích là để thiết kế test case, nhưng khi test thì test như là người dùng cuối hoặc là ở mức hộp đen.

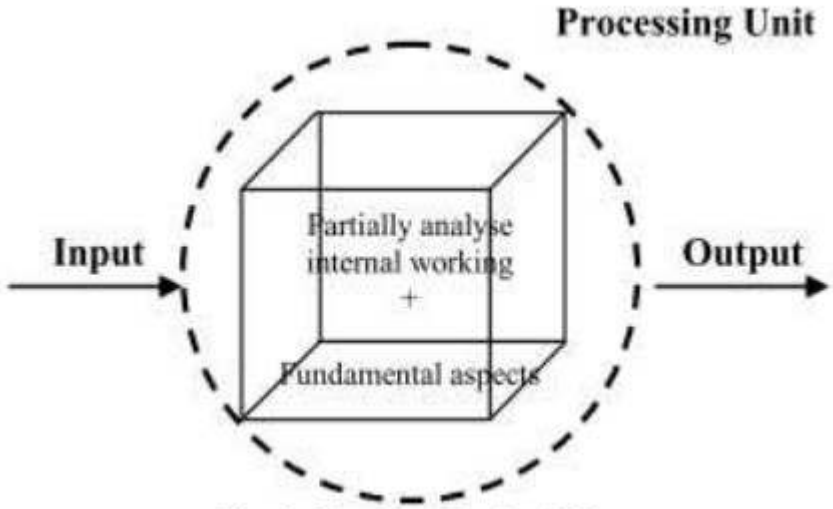
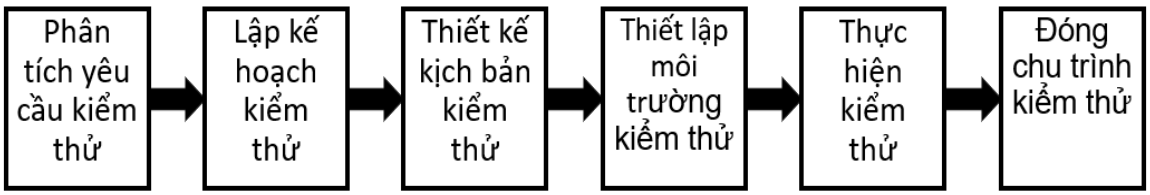


Fig. 6: Grey-box Testing [22]

Hình 2.6. Kiểm thử hộp xám

Ưu - nhược điểm: Kiểm thử hộp trắng sẽ dựa trên các đặc tả chức năng, mô tả của người dùng và sơ đồ kiến trúc hệ thống, từ đó xác nhận các yêu cầu ngay từ ban đầu, Việc kiểm tra sẽ tường minh vì sẽ có nhiều sự quan tâm giữa người kiểm thử phần mềm và người thiết kế hoặc kỹ sư. Bên cạnh đó, phương pháp này cũng cung cấp cả chức năng của kiểm thử hợp trắng và hộp đen. Thế nhưng kiểm thử hộp xám lại có thể mất nhiều thời gian để kiểm tra từng đường dẫn và đôi khi điều này là không thực tế. no còn khó để liên kết lỗi khi thực hiện kiểm tra hộp xám cho một ứng dụng có hệ thống phân tán và có thể không phù hợp để thử nghiệm một số loại chức năng.

2.2.6. Quy trình kiểm thử:



Hình 2.7. Quy trình kiểm thử

1. Phân tích yêu cầu kiểm thử: phân tích, xác định yêu cầu khách hàng (trong đó có cả yêu cầu chức năng và phi chức năng) và phạm vi test.

2. Lập kế hoạch kiểm thử: Dựa vào tài liệu từ bước 1 để lên kế hoạch kiểm thử phần mềm và để xác định lại 1 vài vấn đề (phạm vi dự án, phương pháp tiếp cận)

3. Thiết kế kích bản kiểm thử:Trong giai đoạn này, các Tester sẽ đọc hiểu tất cả các tài liệu, từ đó xác định những việc cần làm, chức năng nào cần test hoặc không. Sau đó, dựa vào kế hoạch và kỹ thuật thiết kế kịch bản kiểm thử, Tester sẽ bắt đầu viết test case. Ngoài test case, Tester cũng cần chuẩn bị các dữ liệu cần thiết khác như test data, test script, test design, test automation script.

4. Thực hiện kiểm thử: Theo test case đã thiết kế và môi trường kiểm thử đã hoàn tất cài đặt và báo cáo bug lên tool quản lý lỗi và theo dõi đến khi fix bug thành công. Tiếp đó, Tester thực hiện retest để verify các fix bug và regression test trong trường hợp có sự thay đổi. Sau khi hoàn tất giai đoạn này, các chuyên viên kiểm thử cần có được test results (kết quả kiểm thử) và defect reports (danh sách các lỗi tìm được).

5. Đóng chu trình kiểm thử: đóng chu trình khi đã có được những tài liệu được tổng hợp từ những giai đoạn trước. Sau đó là tổng kết và báo cáo về quá trình và kết quả đạt được.

PHẦN III. KIỂM THỬ WEBSITE BẰNG CÔNG CỤ KIỂM THỬ TỰ ĐỘNG

3.1. Kiểm thử website

3.1.1. Khái quát

Trong thời đại công nghệ 4.0 hiện nay, internet đã trở thành một phần tất yếu trong cuộc sống của chúng ta.

Hầu hết chúng ta đưa ra quyết định của mình bằng cách tìm kiếm thông tin trên website. Do đó, việc sở hữu một trang web không còn là tùy chọn mà đã là bắt buộc đối với hầu hết mọi loại hình doanh nghiệp. Đây là bước đầu để đưa sản phẩm của mình tiếp cận tới thị trường. Một trang web đơn thuần là chưa đủ, một trang web có nhiều thông tin, dễ truy cập và thân thiện với người dùng là tất cả những gì một doanh nghiệp cần. Để duy trì những điều đó, trang web cần được kiểm tra một cách tốt nhất. Hôm nay chúng ta cùng tìm hiểu về nó.

Kiểm thử tự động là thực hiện kiểm thử bằng một chương trình đặc biệt với rất ít hoặc không có sự tương tác của con người, giúp cho người thực hiện việc kiểm thử phần mềm (tester) không phải lặp đi lặp lại các bước nhàm chán. Công cụ kiểm thử tự động có thể lấy dữ liệu từ file bên ngoài (excel, csv…) nhập vào ứng dụng, so sánh kết quả mong đợi (từ file excel, csv…) với kết quả thực tế và xuất ra báo cáo kết quả kiểm thử

Kiểm thử website nhằm kiểm tra và phát hiện ra những lỗi tiềm ẩn từ trang web hoặc ứng dụng website. Một hệ thống dựa trên website cần phải được kiểm tra hoàn chỉnh từ đầu đến cuối trước khi hệ thống này tới tay người dùng cuối (End-user).

Bằng cách kiểm thử website, một tổ chức / doanh nghiệp có thể đảm bảo rằng hệ thống website của mình đang hoạt động đúng và có thể được chấp nhận bởi người dùng thực của website.

Các phương pháp kiểm thử thông thường là các kỹ thuật tập trung đánh giá các chức năng yêu cầu của ứng dụng. Tuy nhiên, không thể tập trung vào hết các chức năng yêu cầu bởi có rất nhiều chức năng quan trọng cho người sử dụng như: tính

hiệu năng, tính dễ sử dụng, độ tin cậy và tính bảo mật cần được xem xét. Những yêu cầu và mong đợi của người sử dụng, những vấn đề về nền tảng và cấu hình, mô hình nghiệp vụ, sự phát triển và chi phí cho việc kiểm thử là vấn đề hay gặp phải và thay đổi liên tục suốt chu trình của 1 ứng dụng Web. Vì thế cần phát triển 1 chiến lược hiệu quả cho việc kiểm thử mà có thể bao quát được yêu cầu, chức năng cho ứng dụng Web, qua đó giúp cho việc cài đặt, hoàn thành ứng dụng và tránh rủi ro có thể gặp phải.

3.1.2. Đặc điểm về chất lượng của ứng dụng Website

Ứng dụng Web là một loại chương trình máy tính thường chạy với sự hỗ trợ của trình duyệt web và công nghệ web để thực hiện các tác vụ khác nhau trên internet. Web Application thường được lưu trữ trên một máy chủ từ xa và người dùng có thể truy cập nó thông qua việc sử dụng Phần mềm được gọi là trình duyệt web.

Một ứng dụng web có thể được coi là một hệ thống phân tán, với máy chủ của khách hàng hoặc kiến trúc đa tầng, bao gồm các đặc điểm sau:

- Nó có thể được truy cập đồng thời bởi một số lượng lớn người dùng được phân phối trên toàn thế giới
- Nó chạy trên các môi trường thực thi phức tạp, không đồng nhất, bao gồm các phần cứng khác nhau, các kết nối mạng, hệ điều hành, máy chủ Web và trình duyệt Web khác nhau
- Nó có một bản chất cực kỳ không đồng nhất phụ thuộc vào sự đa dạng lớn của các thành phần, phần mềm. Các thành phần này có thể được xây dựng bởi các công nghệ khác nhau (nghĩa là các ngôn ngữ và mô hình lập trình khác nhau)và có thể có tính chất khác nhau (tức là các thành phần mới được tạo ra từ đầu,các thành phần kế thừa, các thành phần hypermedia, COTS, v.v.)
- Nó có thể tạo ra các thành phần, phần mềm trong thời gian chạy theo đầu vào của người dùng và trạng thái máy chủ.

Bên cạnh đó, người sử dụng không chỉ muốn chương trình của họ vận hành một cách ổn định, chính xác mà còn mong đợi những điểm sau: tính dễ sử dụng, độ tin

cậy cao, tốc độ, tương thích với các hệ thống khác nhau và các phiên bản trong tương lai.

- Chức năng: đáp ứng được các yêu cầu xác định, có tính phù hợp, chính xác, khả năng tương tác, tuân thủ và bảo mật.
- Độ tin cậy: duy trì sự hiệu quả của nó trong một điều kiện cụ thể và trong một khoảng thời gian xác định.
- Khả năng sử dụng: tính dễ sử dụng và hiệu quả của một ứng dụng.
- Hiệu quả: tỷ lệ giữa mức độ hiệu quả của một ứng dụng và các tài nguyên mà nó sử dụng trong các điều kiện cụ thể.

3.2. Công việc khi kiểm thử một ứng dụng Web

Kiểm thử là một hoạt động để đánh giá chất lượng của một sản phẩm và quan trọng là cải thiện nó bằng cách tìm ra những thiếu sót, khiếm khuyết. Một vấn đề thường xảy ra trong quá trình phát triển Website đó là các yêu cầu thường không đầy đủ, tường minh và có thể thay đổi bất cứ lúc nào.

Một website thường sẽ có nhiều người dùng với nhiều nền tảng khác nhau nên rất khó đoán được số lượng người dùng một ứng dụng web là bao nhiêu, thời gian hồi đáp của người dùng,…. Điều này dẫn đến việc phải kiểm thử website trước khi đảm bảo website có thể hoạt động bình thường.

3.2.1. Kiểm thử chức năng:

Kiểm thử chức năng đòi hỏi tester phải thực hiết test tất cả chức năng chính của các link trong trang web, định dạng được dùng trong các trang web để gửi và nhận thông tin cần thiết từ người dùng.

- Kiểm tra các liên kết

Liên kết bên trong một cấu trúc siêu văn bản mà điểm đến không tồn tại một nút (các trang web, hình ảnh, ...) gọi là liên kết hỏng thường xuyên xảy ra sai sót trong các ứng dụng web. Link được kiểm thử gồm:

- Liên kết ngoài trang web

- Liên kết nội bộ
- Liên kết tới các vị trí trong cùng trang
- Liên kết sử dụng để gửi email tới admin hoặc người dùng khác trong trang…

- Kiểm tra các hình thức Web trên trang
  - Kiểm tra các lĩnh vực logic xác nhận cho từng lĩnh vực.
  - Kiểm tra các giá trị mặc định cho từng lĩnh vực.
  - Kiểm tra xem các lĩnh vực mật khẩu không hiển thị nội dung mật khẩu.
  - Kiểm tra giá trị đầu vào không hợp lệ cho từng lĩnh vực.
  - Xác nhận đáp ứng với một hình thức gửi.

- **Kiểm tra cookies:** Cookies là các tệp được tạo bởi trang web đã truy cập để lưu trữ thông tin duyệt web như các tùy chọn trang web hoặc thông tin đăng nhập của người dùng. Người dùng có thể tùy chỉnh trình duyệt để quản lý cookies, thực hiện thao tác xóa, chặn, lưu,… để kiểm thử các tính năng lưu hoặc không lưu trạng thái đăng nhập, tính năng bảo mật của ứng dụng web. Test cookies gồm:
  - Kiểm tra cookie (sessions) sẽ bị xóa khi xóa bộ nhớ cache hoặc khi chúng hết hạn.
  - Xóa cookie (sessions) và kiểm tra thông tin đăng nhập có được yêu cầu khi bạn truy cập trang web lần sau.
- Kiểm tra nội dung động (kiểm tra cơ sở dữ liệu):
  - Kiểm tra dữ liệu thống nhất trong các hình thức web cơ sở dữ liệu theo định hướng.

- Kiểm tra chức năng tạo, chỉnh sửa, xóa, cập nhật công việc.
- Kiểm tra dữ liệu cung cấp dữ liệu chính xác.
- Xác định kết nối cơ sở dữ liệu và các lỗi truy vấn.

3.2.2. Kiểm thử khả năng tương thích

Sự khác biệt trong các trình duyệt Web, môi trường hoạt động, và các thiết bị phần cứng ảnh hưởng đến các hoạt động chính xác của ứng dụng Web của bạn. Test khả năng tương thích sẽ đảm bảo ứng dụng website của bạn hiển thị chính xác trên các thiết bị khác nhau, lưu ý rằng:

- Tương thích với trình duyệt (trên máy tính và trên điện thoại di động):
  - Cần kiểm thử ứng dụng web có hiển thị chính xác trên các trình duyệt không.
  - Kiểm tra chức năng ứng dụng với một loạt các cài đặt cấu hình bảo mật trình duyệt.
  - Cần phải kiểm thử ứng dụng web trên càng nhiều trình duyệt càng tốt (IE, Firefox, Chrome, Safari, Opera…) để kiểm thử tương thích.
  - Kiểm tra trên cả các phiên bản khác nhau của trình duyệt.
  - Kiểm thử trên cả trình duyệt của thiết bị điện thoại thông minh.
  - Nếu ứng dụng chạy tốt hơn, hoặc có ưu tiên tương thích hơn với trình duyệt nào đó thì cần có thông báo tới người dùng.
- Tương thích với hệ điều hành: một số chức năng của ứng dụng có thể không tương thích với một số hệ điều hành, hoặc có những lưu ý khác khi sử dụng, điều này cần phải được kiểm thử kỹ và thông báo cho người dùng được biết.

- Tương thích với các thiết bị ngoại vi (máy in…): khi người dùng có lệnh in trang thì phải đảm bảo tính chính xác của fonts, cỡ chữ, cỡ giấy…mà người dùng đã chọn.
- Thiết bị di động tương thích:
    - Kiểm tra khả năng tương thích ứng dụng với các dịch vụ thiết bị, bao gồm cả vị trí và dịch vụ quay số.
    - Kiểm tra giao diện người dùng vẽ trên điện thoại di động kích thước màn hình thiết bị, bao gồm cả màn hình xoay.
    - Kiểm tra hoạt động ứng dụng chính xác khi điện thoại đang trong và ra khỏi phạm vi dịch vụ mạng.

3.2.3. Kiểm thử tính khả dụng

Đây là phần rất quan trọng của bất kỳ dự án nào. Nó có thể được thực hiện bởi tester dev hoặc bất cứ người nào trong dự án. Tính khả dụng của website là trang web dễ sử dụng, có hướng dẫn sử dụng rõ ràng, mỗi trang đều có menu chính và phải nhất quán. Tester cần chú ý:

- Kiểm tra Danh mục chính:
    - Kiểm tra người sử dụng có kiểm soát rõ ràng và dễ dàng di chuyển từ trang này sang trang khác.
    - Kiểm tra dòng chảy của một ứng dụng web bằng cách quan sát cách người sử dụng hoàn thành mục tiêu của họ.
    - Kiểm tra xem người dùng có thể tìm thấy hướng dẫn nếu họ không biết làm gì với một chức năng.
    - Kiểm tra các đối tượng chuyển hướng chung xuất hiện trên tất cả các trang.
    - Chức năng tìm kiếm thử nghiệm cho các chức năng ứng dụng thích hợp.

- Kiểm tra nội dung:
    - Nội dung phải dễ đọc không có lỗi chính tả hoặc ngữ pháp, thân thiện với người dùng.
    - Hình ảnh được sắp xếp gọn gàng, hợp lý và có kích thước phù hợp.
    - Kiểm tra xem màu sắc và hoa văn hướng dẫn phong cách, bao gồm phông chữ, khung hình, và viền.

3.2.4. Kiểm thử bảo mật

Kiểm thử bảo mật rất quan trọng với những trang web thương mại điện tử, lưu trữ thông tin khách hàng hoặc thông tin nhạy cảm như thẻ tín dụng. Gồm:

- Gõ trực tiếp URL vào thanh địa chỉ của trình duyệt mà không qua đăng nhập. Các trang nội bộ phải được bảo mật.
- Kiểm tra xác thực cơ bản sử dụng tên giả và các thông tin mật khẩu.
- Thử nghiệm cho các chức năng ứng dụng chính xác dựa trên các giá trị thuộc tính không hợp lệ URL.
- Kiểm tra các chức năng ứng dụng với các lĩnh vực đầu vào không hợp lệ, bao gồm các lĩnh vực văn bản.
- Kiểm tra, bảo vệ các thư mục Web và các tập tin không thể truy cập của máy chủ Web.
    - Kiểm tra để xác định ứng dụng Web vi phạm an ninh, bao gồm cả thông báo lỗi và vi phạm an ninh nỗ lực đang được đăng nhập.
    - Không thể tải xuống các tệp bị hạn chế nếu không có quyền truy cập phù hợp
    - Kiểm tra các lĩnh vực CAPTCHA cho các hình thức web và đăng nhập.
    - Tất cả các phiên giao dịch, các thông báo lỗi, các hành vi cố gắng xâm phạm an ninh phải ghi trong log và lưu tại web server.

3.2.5. Kiểm thử hiệu năng (Performance)

Kiểm tra hiệu năng sẽ đảm bảo website hoạt động dưới tất cả các tải, gồm các yêu cầu sau:

- Thời gian phản hồi của ứng dụng trang web ở các tốc độ kết nối khác nhau
- Xác định các vấn đề về độ trễ mạng về chức năng ứng dụng Web.
- Stress test ứng dụng web để xác định hành vi của nó vẫn hoạt động bình thường vào tầm cao điểm.
- Stress test trang web để xác định điểm dừng của nó khi được đẩy vượt quá tải bình thường vào tầm cao điểm sẽ ra sao.
- Kiểm tra xem nếu có sự cố xảy ra do tải cao điểm, làm thế nào để trang web phục hồi sau sự cố đó.
- Đảm bảo các kỹ thuật tối ưu hóa như nén zip, bộ đệm phía trình duyệt và máy chủ được bật để giảm thời gian tải.

PHẦN IV. THỰC NGHIỆM

4.1. Các công cụ kiểm thử tự động

4.1.1. Công cụ kiểm thử hiệu năng:

Dưới đây là danh sách một số công cụ kiểm thử hiệu năng được sử dụng rộng rãi nhất để đo hiệu suất ứng dụng Web và khả năng chịu tải của chúng. Các công cụ kiểm tra tải này sẽ đưa ra đánh giá về hiệu suất của ứng dụng trong thời gian có lưu lượng truy cập cao điểm.

4.1.1.1. WebLoad:

Cho phép thực hiện kiểm thử khả năng chịu tải và độ chịu lỗi của ứng dụng Web bằng cách sử dụng Ajax, Adobe Flex, .NET, Oracle. Forms, HTML5 và nhiều công nghệ khác. Điểm mạnh của WebLoad là là dễ sử dụng với các tính năng như

cho phép ghi/phát lại dựa trên DOM, tương quan tự động và ngôn ngữ kịch bản Javascript. Công cụ này hỗ trợ thử nghiệm hiệu suất quy mô lớn với các kịch bản phức tạp và đưa ra những phân tích rõ ràng.

4.1.1.2. NeoLoad:

Công cụ sử dụng để đo và phân tích hiệu suất của ứng dụng Web. NeoLoad phân tích hiệu suất của ứng dụng Web bằng cách tăng lưu lượng truy cập vào ứng dụng. Nhờ đó, kiểm thử viên có thể biết được năng lực chịu tải của ứng dụng. Công cụ này được viết trên nền Java, tương thích với nhiều hệ điều hành khác nhau và hỗ trợ 2 ngôn ngữ tiếng anh và tiếng pháp.

4.1.1.3. Apache JMeter:

Đây là một công cụ phát triển trên mã nguồn mở. Apache Jmeter được coi như một công cụ kiểm thử hiệu năng, có khả năng tích hợp với kế hoạch kiểm thử. Ngoài việc kiểm thử hiệu năng, Apache JMeter còn có thể sử dụng để kiểm tra các chức năng của ứng dụng Web.

4.1.2. Công cụ kiểm thử bảo mật:

4.1.2.1. OWASP Zed Attack Proxy:

Tương tự như Burp Suite, OWASP Zed Attack Proxy là công cụ để thâm nhập, đánh giá an ninh mạng, bảo mật của các ứng dụng Web.

4.1.2.2. Nikto:

Công cụ đánh giá hệ thống Nikto là một máy quét lỗ hổng máy chủ Web mã nguồn mở. Nó phát hiện việc cài đặt phần mềm và cấu hình đã lỗi thời, các tệp tin có khả năng nguy hiểm, v.v.

4.1.2.3. Exploit-Me:

Là một công cụ kiểm tra bảo mật ứng dụng Web có thể tích hợp trên trình duyệt Firefox được thiết kế nhỏ gọn, dễ sử dụng. Exploit- Me bao gồm các gói: XSS-Me

và SQL Inject-Me. Cross-Site Scripting (XSS) là một lỗ hổng được tìm thấy trong nhiều ứng dụng Web hiện nay. . Lỗ hổng XSS có thể gây ra thiệt hại nghiêm trọng cho một ứng dụng Web. XSS-Me là công cụ giúp phát hiện ra các lỗ hổng XSS này. Trong khi đó, SQL Inject- Me được sử dụng để kiểm tra các lỗ hổng SQL Injection trong ứng dụng Web.

4.1.3. Công cụ kiểm thử chức năng

4.1.3.1. BrowserStack:

Đây là một công cụ giúp kiểm thử hoạt động của các chức năng trên ứng dụng Web trên nhiều trình duyệt khác nhau. Ứng dụng Web có thể được kiểm tra bằng thao tác của người dùng hoặc tự động thông qua Selenium. Ngoài ra, BrowserStack còn cung cấp tính năng chụp ảnh màn hình ứng dụng Web trên 650 trình duyệt khác nhau và kiểm tra khả năng hiển thị responsive trên các loại màn hình.

4.1.3.2. Ranorex:

Công cụ kiểm thử tự động cho các ứng dụng Web, desktop và di động. Chỉ với một tài khoản, người dùng có thể sử dụng Ranorex để kiểm thử cho 3 loại ứng dụng kể trên. Việc tích hợp này sẽ giúp rút ngắn thời gian khi kiểm thử ứng dụng được thiết kế chạy trên nhiều nền tảng khác nhau. Tuy nhiên, bản trả phí của Ranorex khá đắt, lên tới 3500$/năm.

4.1.3.3. Công cụ Selenium:

Selenium là bộ kiểm thử tự động miễn phí (mã nguồn mở) dành cho các ứng dụng web trên các trình duyệt và nền tảng khác nhau. Nó khá là giống với HP Quick Test Pro (QTP bây giờ là UFT) chỉ khác là Selenium thì tập trung vào việc tự động hoá các ứng dụng dựa trên nền tảng web. Kiểm thử được thực hiện bằng cách sử dụng công cụ Selenium thường được gọi là Kiểm thử Selenium. Selenium không chỉ là 1 công cụ độc lập mà là 1 bộ công cụ của phần mềm, mỗi bộ đều đáp ứng được nhu cầu kiểm thử khác nhau của 1 tổ chức.

Nó có 4 thành phần.

- Selenium Integrated Development Environment (IDE) là một công cụ cho phép chúng ta Record/Playback một test script. Đây là một add-on hỗ trợ cho FireFox. Chúng ta chỉ có thể Record trên trình duyệt FireFox, nhưng bù lại, chúng ta có thể Playback trên các trình duyện khác như là IE, Chrome….
- Selenium Remote Control (RC) là một framework kiểm thử cho phép thực hiện nhiều hơn và tuyến tính các hành động trên trình duyệt. Nó cho phép cho phép các nhà phát triển tự động hóa kiểm thử sử dụng một ngôn ngữ lập trình cho tính linh hoạt tối đa và mở rộng trong việc phát triển logic thử nghiệm
- Selenium WebDriver là sự kế thừa từ Selenium Remote Control, làm việc trực tiếp với trình duyệt ở mức hệ điều hành, cho phép gửi lệnh trực tiếp đến trình duyệt và cho ra kết quả
- Selenium Grid là một hệ thống hỗ trợ người dùng thực thi test script trên nhiều trình duyệt một cách song song mà không cần phải chỉnh sửa test script.

Các đặc điểm của Selenium

1. Mã nguồn mở. Phải nói điểm này là điểm mạnh nhất của Selenium khi so sánh với các test tool khác. Vì là mã nguồn mở nên chúng ta có thể sử dụng mà không phải lo lắng về phí bản quyền hay thời hạn sử dụng.

2. Cộng đồng hỗ trợ. Vì là mã nguồn mở nên Selenium có một cộng đồng hỗ trợ khá mạnh mẽ. Bên cạnh đó, Google là nơi phát triển Selenium nên chúng ta hoàn toàn có thể yên tâm về sự hổ trợ miễn phí khi có vấn đề về Selenium. Tuy nhiên, đây cũng là một điểm yếu của Selenium. Cơ bản vì là hàng miễn phí, cộng đồng lại đông nên một vấn đề có thể nhiều giải pháp, và có thể một số giải pháp là không hữu ích. Mặc khác, chúng ta không thể hối thúc hay ra deadline cho sự hỗ trợ.

3. Selenium hỗ trợ nhiều trình duyệt khác nhau (Google Chrome, Mozilla Firefox, MS Edge, Opera và Safari) để thực thi các test case.

4. Selenium hỗ trợ chạy trên nhiều OS khác nhau với mức độ chỉnh sửa script hầu như là không có. Thực sự thì điều này phụ thuộc phần lớn vào khả năng viết script của chúng ta.

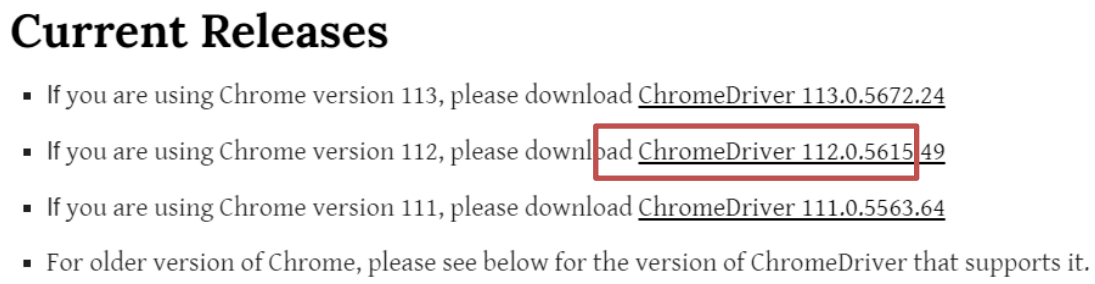
5. Chạy test case ở background. Khi chúng ta thực thi một test scrpit, chúng ta hoàn toàn có thể làm việc khác trên cùng một PC. Điều này hỗ trợ chúng ta không cần tốn quá nhiều tài nguyên máy móc khi chạy test script.

6. Không hỗ trợ Win app. Selenium thực sự chỉ hỗ trợ chúng ta tương tác với Browser mà không hỗ trợ chúng ta làm việc với các Win app, kể cả Win dialog như Download/Upload – ngoại trừ Browser Alarm. Vậy nên, để xử lý các trường hợp cần tương tác với hệ thống hay một app thứ ba, chúng ta cần một hay nhiều thư viện khác như AutoIt hay Coded UI.

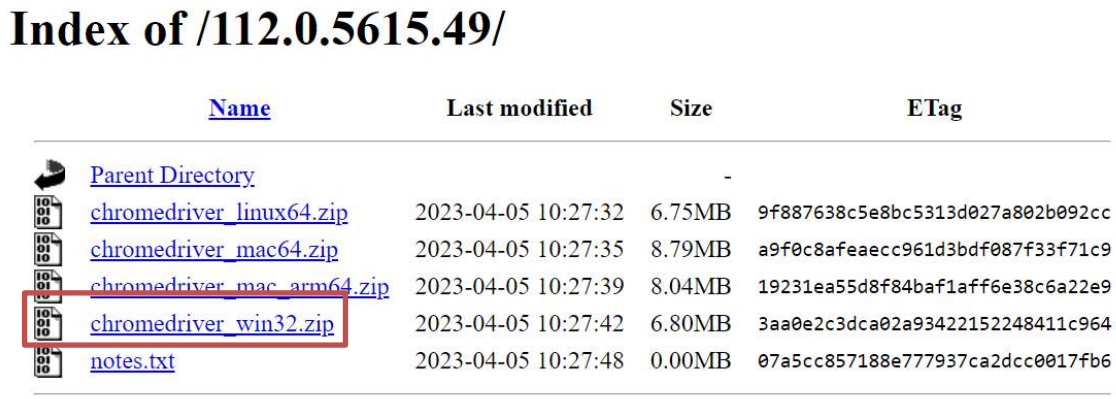
7. Selenium hỗ trợ thực hiện kiểm thử song song giúp giảm thời gian và tăng hiệu quả của các kiểm thử.

Cài đặt và chạy Selenium

- Tải chrome driver và lựa chọn phiên bản phù hợp



Hình 3.1. Lựa chọn phiên bản trình duyệt muốn cài



Hình 3.2. Chọn driver thuộc hệ điều hành mà mình sử dụng

4.2. Xây dựng testcase

Test case về giao diện và chức năng "Đăng nhập"

Mã trường hợp kiểm thử	Mục đích kiểm thử	Tiền điều kiện	Các bước thực hiện	Kết quả mong muốn	Kết quả hiện tại	Mã lỗi	Ghi chú
1. Chức năng Đăng nhập							
ĐN_1	Kiểm tra giao diện màn hình đăng nhập	1. Thiết bị có thể truy cập Internet	1. Di chuyển đến màn hình "Login" theo đường link sau : https://www.facebook.com/ 2.Check giao diện mặc định của màn hình "Đăng nhập"	Bao gồm các trường: Email và số điện thoại, Mật khẩu, button [Đăng nhập], button [Đăng ký], link "Quên mật khẩu?", link "Đăng nhập bằng facebook" 			
ĐN_2	Kiểm tra bố cục, căn chỉnh các field, color, font chữ (icon, logo, place holder)	1. Thiết bị có thể truy cập Internet	1.Check bố cục, căn chỉnh các field, font, color 2. Check label, textbox	1. Căn chỉnh các field hợp lý, thẳng hàng Bố cục giống như hình, cùng một font, size, màu chữ 2. Các label, textbox có độ dài, khoảng cách bằng nhau, không xô lệch Không có lỗi chính tả, ngữ pháp			

Hình 3.3. Test case về giao diện chức năng "Đăng Nhập"

Test case về giao diện và chức năng “Tìm kiếm”

2. Chức năng Tìm kiếm							
TK_1	Kiểm tra giao diện màn hình Tìm Kiếm	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống	1. Di chuyển đến thanh tìm kiếm trên màn hình chính 2. Check giao diện, bố cục, màu sắc, kích thước, ...	2. Hiển thị lịch sử tìm kiếm trước đó "Tìm Kiếm gần đây"			
TK_2	Kiểm tra bố cục, căn chỉnh các field, clolor, font chữ (icon, logo, place holder)	1. Thiết bị có thể truy cập Internet	1.Check bố cục, căn chỉnh các field, font, color 2. Check label, textbox	1. Căn chỉnh các field hợp lý, thẳng hàng Bố cục giống như hình, cùng một font, size, màu chữ 2. Các label, textbox có độ dài, khoảng cách bằng nhau, không xô lệch Không có lỗi chính tả, ngữ pháp			
TK_3	Kiểm tra Placeholder thanh tìm kiếm	1. Thiết bị có thể truy cập Internet	1. Check Placeholder Tìm Kiếm bị xóa khi click vào textbox 2.Check placeholder Tìm Kiếm có bị xoá khi nhập value vào textbox	1. Placeholder sẽ bị mờ hoặc xóa khi click vào textbox 2.Placeholder có bị xóa khi nhập value vào textbox			
TK_4	Kiểm tra chức năng tìm kiếm khi không nhập kí tự vào textbox	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống	1. Di chuyển đến thanh tìm kiếm trên màn hình chính 2. Không nhập kí tự nào vào thanh tìm kiếm 3. Nhấn Enter	3. Hệ thống tự thoát ra khỏi thanh tìm kiếm			
TK_5	Kiểm tra chức năng tìm kiếm khi nhập 1 kí tự vào textbox	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống	1. Di chuyển đến thanh tìm kiếm trên màn hình chính 2. Nhập 1 ký tự bất kì vào thanh tìm kiếm 3. Nhấn Enter	3. Màn hình sẽ hiển thị kết quả tìm kiếm những người có tên tài khoản có chứa kí tự đã nhập. Kết quả được sắp xếp từ trên xuống theo độ tương tác			
TK_6	Kiểm tra Copy keyboard, right click hoạt động với thanh Tìm Kiếm	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống	1. Di chuyển đến thanh tìm kiếm trên màn hình chính 2. Sử dụng chức năng copy đối với thanh Tìm Kiếm 3. Nhấn Enter	4. Copy keyboard, right click có hoạt động với thanh Tìm Kiếm			
TK_7	Kiểm tra chức năng trimspace với thanh Tìm Kiếm	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống	1. Nhập dữ liệu và thêm space vào đầu và cuối(hoặc thực hiện copy paste một số có chứa space ở đầu, cuối) 2. Nhấn Enter	2. Tìm kiếm thành công			
TK_6	Kiểm tra chức năng tìm kiếm khi nhập đúng tên tài khoản muốn tìm vào textbox	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống	1. Di chuyển đến thanh tìm kiếm trên màn hình chính 2. Nhập tên tài khoản mà người dùng muốn tìm kiếm vào thanh tìm kiếm 3. Nhấn Enter	3. Màn hình tìm kiếm sẽ hiển thị kết quả tìm kiếm những người có tên tài khoản hợp lệ nhất. Kết quả được sắp xếp từ trên xuống theo độ chính xác và độ tương tác			
TK_7	Kiểm tra chức năng tìm kiếm khi nhập tên tài khoản không tồn tại vào textbox	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống	1. Di chuyển đến thanh tìm kiếm trên màn hình chính 2. Nhập tên tài khoản mà người dùng muốn tìm kiếm vào thanh tìm kiếm 3. Nhấn Enter	3. Màn hình tìm kiếm sẽ hiển thị thông báo "Không tìm thấy kết quả"			
TK_8	Kiểm tra chức năng tìm kiếm khi nhập quá max length vào textbox	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống	1. Di chuyển đến thanh tìm kiếm trên màn hình chính 2. Nhập vào thanh tìm kiếm với độ dài quá độ dài max	2. Hệ thống tự động chặn không cho phép nhập quá maxlenght			
TK_9	Kiểm tra chức năng tìm kiếm khi ấn (Chỉnh sửa) trong popup	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống	1. Di chuyển đến thanh tìm kiếm trên màn hình chính 2. Nhấn vào thanh tìm kiếm sẽ thấy hiện ra các tìm kiếm trước đó 3. Nhấn vào (Chỉnh sửa)	3. Hệ thống sẽ chuyển qua màn hình Nhật ký hoạt động với Lịch sử tìm kiếm trước đó			
TK_10	Kiểm tra chức năng Tìm Kiếm khi bị mất kết nối với Server	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống 3. Hệ thống mất kết nối với Server	1. Di chuyển đến thanh tìm kiếm trên màn hình chính 2. Nhập tên tài khoản mà người dùng muốn tìm kiếm vào thanh tìm kiếm 3. Nhấn Enter	3. Tìm Kiếm không thành công			
TK_11	Kiểm tra chức năng tìm kiếm khi ấn (x) trong popup	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống	1. Di chuyển đến thanh tìm kiếm trên màn hình chính 2. Nhấn vào thanh tìm kiếm sẽ thấy hiện ra các tìm kiếm trước đó 3. Nhấn vào (x)	3. Khi nhấn dấu (x) bên cạnh tìm kiếm nào trước đó thì sẽ bị xóa đi			

Hình 3.4. Test case về giao diện và chức năng "Tìm kiếm"

Test case về giao diện và chức năng “Theo dõi”

3. Chức năng	Theo dõi						
KB_1	Kiểm tra chức năng Theo dõi thành công	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống 3. Người dùng tìm kiếm được người muốn kết bạn	1. Người dùng truy cập vào trang cá nhân của người muốn theo dõi 2. Người dùng nhấn button [Theo dõi]	2. Gửi lời mời kết bạn thành công, button [Theo dõi] chuyển thành button [Đang theo dõi]			
KB_2	Kiểm tra chức năng Theo dõi thành công	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống	1. Người dùng thấy người muốn kết bạn ở mục Gợi ý cho bạn" 2. Người dùng nhấn button [Theo dõi] ngay dưới avatar tài khoản được gợi ý	2. Gửi lời mời kết bạn thành công, button [Theo dõi] chuyển thành button [Đang theo dõi]			
KB_3	Kiểm tra chức năng Theo dõi không thành công khi người dùng [Hủy theo dõi]	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống	1. Người dùng truy cập vào trang cá nhân của người muốn kết bạn 2. Người dùng nhấn button [Theo dõi] 3. Người dùng nhấn button [Đang theo dõi] sau đó chọn Bỏ theo dõi	3. Người dùng bỏ theo dõi			
KB_4	Kiểm tra chức năng Theo dõi không thành công khi người dùng nhấn button [Hủy theo dõi]	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống	1. Người dùng thấy người muốn kết bạn ở mục "Gợi ý cho bạn" 2. Người dùng nhấn button [Theo dõi] ngay dưới avatar tài khoản được gợi ý 3. Người dùng nhấn button [Hủy theo dõi]	3. Người dùng bỏ theo dõi			
KB_5	Kiểm tra chức năng Theo dõi khi bị mất kết nối với Server	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống 3. Hệ thống mất kết nối với Server	1. Người dùng thấy người muốn kết bạn ở mục "Gợi ý cho bạn" 2. Người dùng nhấn button [Theo dõi] ngay dưới avatar tài khoản được gợi ý	3. Gửi lời mời kết bạn không thành công			

Hình 3.5. Test case về giao diện và chức năng "Theo dõi"

Test case về giao diện và chức năng “Hủy theo dõi”

HKB_1	Kiểm tra chức năng Hủy theo dõi thành công	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống 3. Người dùng tìm kiếm được người muốn hủy theo dõi	1. Người dùng truy cập vào trang cá nhân của người muốn hủy theo dõi (hoặc tìm kiếm trong danh sách theo dõi của mình) 2. Người dùng nhấn button [Người theo dõi] 3. Người dùng tiếp tục chọn chức năng [Hủy theo dõi]	2. Hiển thị popup "Bỏ theo dõi với xxx" và 2 button [Bỏ theo dõi] và [Hủy]			
HKB_2	Kiểm tra chức năng Hủy theo dõi thành công khi chọn button [Xác nhận]	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống 3. Đã thực hiện các bước như ở HKB_1	1. Người dùng nhấn button [Bỏ theo dõi] ở popup	2. Hủy theo dõi thành công, button [Đang theo dõi] chuyển thành button [Theo dõi]			
HKB_3	Kiểm tra chức năng Hủy theo dõi không thành công khi người dùng nhấn button [Hủy]	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống 3. Đã thực hiện các bước như ở HKB_1	1. Người dùng nhấn button [Hủy] ở popup	3. Hủy theo dõi không thành công, popup biến mất			
HKB_4	Kiểm tra chức năng Hủy theo dõi không thành công khi người dùng nhấn button [x] trên popup	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống 3. Đã thực hiện các bước như ở HKB_1	1. Người dùng nhấn button [x] ở popup	3. Hủy theo dõi không thành công, popup biến mất			
HKB_5	Kiểm tra chức năng Hủy theo dõi khi bị mất kết nối với Server	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống 3. Hệ thống mất kết nối với Server	1. Người dùng truy cập vào trang cá nhân của người muốn hủy theo dõi (hoặc tìm kiếm trong danh sách theo dõi của mình) 2. Người dùng nhấn button [Người theo dõi] 3. Người dùng tiếp tục chọn chức năng [Hủy theo dõi]	3. Hủy theo dõi không thành công			
HKB_6	Kiểm tra chức năng Hủy theo dõi khi nhấn button [Nhắn tin]	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống 3. Người dùng tìm kiếm được người muốn hủy kết bạn	1. Người dùng truy cập vào trang cá nhân của người muốn hủy theo dõi (hoặc tìm kiếm trong danh sách theo dõi của mình) 2. Người dùng nhấn button [Nhắn tin]	2. Chuyển qua màn hình nhắn tin hoặc hiển thị thanh chat phía dưới bên phải màn hình			

Hình 3.6. Test case về giao diện và chức năng “Hủy theo dõi”

Test case về giao diện và chức năng “Đăng xuất”

5. Chức năng Đăng xuất							
ĐX_1	Kiểm tra giao diện màn hình đăng xuất	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống	1. Người dùng ấn vào biểu tượng tài khoản bên góc phải màn hình	1. Hiển thị popup như hình: 			
ĐX_2	Check bố cục, căn chỉnh các field, clolor, font chữ (icon, logo, place holder)	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống	1.Check bố cục, căn chỉnh các field, font, color 2. Check label, textbox	1. Căn chỉnh các field hợp lý, thẳng hàng Bố cục giống như hình, cùng một font, size, màu chữ 2. Các label, textbox có độ dài, khoảng cách bằng nhau, không xô lệch Không có lỗi chính tả, ngữ pháp			
ĐX_3	Kiểm tra chức năng đăng xuất khi người dùng bấm chọn "Đăng xuất"	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống	1. Người dùng ấn vào biểu tượng tài khoản bên góc phải màn hình 2. Nhấn chọn "Đăng xuất"	1. Hệ thống hiển thị popup "Đang đăng xuất. Bạn cần đăng nhập lại" và sau đó thoát ra khỏi tài khoản			
ĐX_6	Kiểm tra chức năng đăng xuất khi người dùng bấm chọn [x]	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống	1. Người dùng ấn vào biểu tượng tài khoản bên góc phải màn hình 2. Nhấn chọn "Đăng xuất" 3. Chọn [x]	3. Hệ thống thoát popup và trở lại màn hình chính, không đăng xuất khỏi thiết bị hiện tại			
ĐX_7	Kiểm tra chức năng Đăng xuất khi bị mất kết nối với Server	1. Thiết bị có thể truy cập Internet 2. Người dùng đã đăng nhập thành công vào hệ thống 3. Hệ thống mất kết nối với Server	1. Người dùng ấn vào biểu tượng tài khoản bên góc phải màn hình 2. Nhấn chọn "Đăng xuất" 3. Chọn button "Có"	3. Đăng xuất không thành công			

Hình 3.7. Test case về giao diện và chức năng " Đăng xuất"

4.3. Thực hiện kiểm thử tự động

4.3.1. Kiểm thử chức năng – sử dụng framework seleniumbase với ngôn ngữ python

- Cài đặt framework seleniumbase và webdriver trong terminal.

```
(venv) PS D:\ppp\pythonproject> pip install selenium
Requirement already satisfied: selenium in d:\ppp\pythonproject\venv\lib\site-packages (4.8.3)
Requirement already satisfied: trio-websocket~=0.9 in d:\ppp\pythonproject\venv\lib\site-packages (from selenium) (0.10.2)
Requirement already satisfied: trio~=0.17 in d:\ppp\pythonproject\venv\lib\site-packages (from selenium) (0.22.0)
Requirement already satisfied: urllib3[socks]~=1.26 in d:\ppp\pythonproject\venv\lib\site-packages (from selenium) (1.26.15)
Requirement already satisfied: certifi>=2021.10.8 in d:\ppp\pythonproject\venv\lib\site-packages (from selenium) (2022.12.7)
```

Hình 4.1 Cài đặt webdriver trong cmd

- Thực hiện kiểm thử

a) Kiểm thử chức năng “Đăng nhập”

Hình 4.2. Scrips test chức năng "Đăng nhập"

b) Kiểm thử chức năng “Tìm kiếm”

```
from loginTest import loginSuccess
from seleniumbase import BaseCase
BaseCase.main(__name__, __file__)

5 usages
def search(self, key):
    self.assert_element("(//div/a[@role='link']/div)[3]")
    self.click_xpath("(//div/a[@role='link']/div)[3]")
    self.assert_element("input[aria-label='Search input']")
    self.type("input[aria-label='Search input']", key)
    self.assert_element("(//div[@role='none']/a)[1]")
    self.highlight("(//div[@role='none']/a)[1]/div/div")
    self.save_screenshot_to_logs("Search")

class TestInstaSeach(BaseCase):
    def test_insta_seach(self):
        loginSuccess(self)
        search(self, "qyn.trg")
```

Hình 4.3. Script test chức năng "Tìm kiếm"

c) Kiểm thử chức năng “Theo dõi” và “Hủy theo dõi”

- Theo dõi

```
from search import search
from loginTest import loginSuccess
from seleniumbase import BaseCase
BaseCase.main(__name__, __file__)

1 usage
def follow(self):
    try:
        self.assert_element("(//div[@role='none']/a)[1]")
        self.click_xpath("(//div[@role='none']/a)[1]")
        self.assert_element("div:contains('Follow')")
        self.click("button:contains('Follow')")
        self.assert_element("img[alt='Profile photo']")
        self.assert_element("button:contains('Requested')")
        self.save_screenshot_to_logs("Follow")
    except Exception as e:
        print("An exception occurred", e)


class TestInstaFollow(BaseCase):
    def test_insta_follow(self):
        loginSuccess(self)
        search(self, "qyn.trg")
        follow(self)
```

Hình 4.4. Scrips test chức năng “Theo dõi”

- Hủy theo dõi

```
from search import search
from loginTest import loginSuccess
from seleniumbase import BaseCase
BaseCase.main(__name__, __file__)

1 usage
def follow(self):
    try:
        self.assert_element("(//div[@role='none']/a)[1]")
        self.click_xpath("(//div[@role='none']/a)[1]")
        self.assert_element("button:contains('Requested')")
        self.click("button:contains('Requested')")
        self.assert_element("div:contains('Unfollow')")
        self.click("button:contains('Unfollow')")
        self.assert_element("header button:contains('Follow')")
        self.save_screenshot_to_logs("unFollow")
    except Exception as e:
        print("An exception occurred", e)

class TestInstaUnFollow(BaseCase):
    def test_insta_unfollow(self):
        loginSuccess(self)
        search(self, "qyn.trg")
        follow(self)
```

Hình 4.5. Scripts test chức năng "Hủy theo dõi"

d) Kiểm thử chức năng thích bài viết

```
from loginTest import loginSuccess
from seleniumbase import BaseCase
BaseCase.main(__name__, __file__)

1 usage
def like(self):
    try:
        self.click("a:contains('Home')")
        selectorList = ["(//span[contains(@class, '_aamw')])[1]", "(//span[contains(@class, '_aamw')])[2]",
                        "(//span[contains(@class, '_aamw')])[3]"]
        for index, selector in enumerate(selectorList):
            self.assert_element(selector)
            self.click_xpath(selector)
            screeen_index = "Like" + str(index)
            self.save_screenshot_to_logs(screeen_index)
            self.wait(1)
    except Exception as e:
        print("An exception occurred", e)


class TestInstaUnFollow(BaseCase):
    def test_insta_unfollow(self):
        loginSuccess(self)
        like(self)
```

Hình 4.6. Scripts test chức năng "Thích bài viết'

e) Kiểm thử chức năng "Đăng xuất"

```
from loginTest import loginSuccess
from seleniumbase import BaseCase
BaseCase.main(__name__, __file__)

1 usage
def logout(self):
    # Click more
    self.assert_element("(//div/a[@role='link']/div)[10]")
    self.click_xpath("(//div/a[@role='link']/div)[10]")
    # click logout
    self.assert_element("._aa61")
    self.click("._aa61 > a:last-child")
    self.save_screenshot_to_logs("Logout")

class TestInstaLogout(BaseCase):
    def test_insta_logout(self):
        loginSuccess(self)
        logout(self)
```

Hình 4.7. Scripts test chức năng "Đăng xuất"

KẾT LUẬN

Kiểm thử phần mềm là hoạt động quan trọng nhằm đảm báo chất lượng phân mềm. Việc nghiên cứu lựa chọn các kỹ thuật và chiến lược kiểm thử phần mềm phù hợp giúp cho việc kiểm thử có hiệu quả, giảm chi phí và thời gian. Việc xây dựng tài liệu kiểm thử phần mềm hợp lý sẽ giúp cho việc tổ chức, quản lý và thực hiện kiểm thử có hiệu quả.

Kết quả đạt được:

Trong thời gian làm bài với sự định hướng và sự giúp đỡ tận tình của giáo viên hướng dẫn, chúng em đã đạt được kết quả sau:

- Nắm được tổng quan về kiểm thử phần mềm: các phương pháp, kỹ thuật kiểm thử phần mềm, các vấn đề liên quan, ...

- Tìm hiểu và nắm được phương pháp thiết kế testcase trong kiểm thử phần mềm.

- Tìm hiểu chi tiết cách cài đặt và sử dụng tiện ích Selenium kết hợp Codeception trên trình duyệt Chrome.

- Tìm hiểu cách cài đặt và sử dụng công cụ Apache Jmater để kiểm thử hiệu năng.

Hướng phát triển

- Khi nghiên cứu về kiểm thử phần mềm nói chung và công cụ Selenium nói riêng, chúng em đã hiểu được kiểm thử là rất quan trọng trong quy trình sản xuất phần mềm, đảm bảo chất lượng phần mềm. Sự áp dụng với kiến thức tìm hiểu được mới chỉ dừng lại ở một bài toán nhỏ. Hướng phát triển của chúng em:

+ Tìm hiểu và nghiên cứu các kỹ thuật nâng cao về công cụ kiểm thử tự động Selenium kết hợp Codeception trên trình duyệt Chrome, và một số công cụ kiểm thử tự động khác.

+ Thực hiện kiểm thử trên mô hình bài toán phần mềm rộng hơn, phức tạp hơn.

Tài liệu tham khảo

[1] Tài liệu "Các loại kiểm thử phần mềm" https://www.academia.edu/29409876/Cac_loai_kiem_thu_phan_mem

[2] Tài liệu hướng dẫn sử dụng công cụ Apache Jmeter bằng Tiếng Việt - https://jmetervietnam.wordpress.com/.

[3] Giáo trình "Kiểm thử phần mềm" của Th. Phạm Ngọc Hùng, TS. Trương Anh Hoàng, TS. Đặng Văn Hưng.

[4]https://bizflycloud.vn/tin-tuc/selenium-la-gi-20220328105303215.htm#:~:text=Selenium%20l%C3%A0%20c%C3%B4ng%20c%E1%BB%A5%20th%E1%BB%AD,test%20tr%C3%AAn%20nhi%E1%BB%81u%20tr%C3%ACnh%20duy%E1%BB%87t.

[5]https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing

Bảng phân chia công việc

Stt	Họ và tên	Phân chia công việc	Đánh giá
1	Trần Thị Hồng	- Tìm hiểu các công việc khi kiểm thử một ứng dụng Website. - Tìm hiểu và thực hiện quy trình kiểm thử hiệu năng.	Đã hoàn thành
2	Bùi Thị Quỳnh Trang	- Tìm hiểu kiểm thử phần mềm. - Tìm hiểu và thực hiện kiểm thử chức năng "Đăng nhập", "Đăng xuất". - Thiết kế Slide.	Đã hoàn thành
3	Lê Ngọc Quỳnh	- Tìm hiểu các công cụ kiểm thử tự động. - Tìm hiểu và thực hiện kiểm thử chức năng "Theo dõi", "Hủy theo dõi".	Đã hoàn thành
4	Hoàng Hải Yến	- Tìm hiểu và Thiết kế Testcase. - Tìm hiểu và thực hiện kiểm thử chức năng "Tìm kiếm", "Thích bài viết".	Đã hoàn thành