BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC ĐÔNG Á
KHOA CÔNG NGHỆ THÔNG TIN

Báo cáo đồ án
Kiểm Thử Phần Mềm

Sinh viên thực hiện : Nguyễn Thị Thảo Trinh
Nguyễn Thị Kim Huyền
Nguyễn Thị Huyền Trang
Nguyễn Trần Tuyên
Nguyễn Văn Quốc
Trần Công Đô

Giảng viên hướng dẫn : Tạ Quốc Ý

Lớp : IT18A1.11

Đà Nẵng, ngày 14/01/2022

LỜI MỞ ĐẦU

Trong giai đoạn bùng nổ công nghệ thông tin như hiện nay , nhu cầu phát triển phần mềm cả về chất lượng và số lượng đang trở nên bức thiết. Kéo theo đó là nhu cầu kiểm thử phần mềm để đạt được chất lượng tối ưu trước khi đến tay khách hàng. Vì thế mà các công cụ hỗ trợ kiểm tra tự động đang dần trở thành các trợ thủ đắc lực cho đội ngũ phát triển phần mềm. Một công cụ tiêu biểu trong số đó là Selenium. Đây là một công cụ mã nguồn mở, mạnh mẽ, hỗ trợ các ứng dụng trên nền web, nhiều platform và các trình duyệt phổ biến.

Nội dung của báo cáo trình bày một số hiểu biết cơ bản về Software Testing, đồng thời cũng trình bày kết quả của việc ứng dụng Selenium vào kiểm thử một số ứng dụng trên nền web. Do điều kiện hạn chế về thời gian, khả năng lập trình, tài liệu hỗ trợ nên nhóm chúng em chỉ tìm hiểu giới hạn trong Selenium WebDriver.

Đây là công cụ kiểm thử tự động đầu tiên mà các thành viên trong nhóm được tiếp xúc, lại được hoàn thành trong quỹ thời gian hạn hẹp vì thế không tránh khỏi khiếm khuyết, chúng em kính mong nhận được sự cảm thông và chỉ bảo tận tình của thầy.

Nhóm chúng em xin được gửi lời cảm ơn chân thành tới Th.S Tạ Quốc Ý, người trực tiếp hướng dẫn cho chúng em những định hướng và ý kiến quý báu trong quá trình thực hiện.

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM

1.1. Khái niệm về phần mềm

Phần mềm thường được mô tả bởi ba thành phần cấu thành:

Tập các lệnh (chương trình máy tính) trên máy tính khi thực hiện sẽ tạo ra các dịch vụ và đem lại những kết quả mong muốn cho người dùng.

Các cấu trúc dữ liệu (lưu giữ trên các bộ nhớ) làm cho chương trình thao tác hiệu quả với các thông tin thích hợp và nội dung thông tin được số hóa.

Các tài liệu để mô tả thao tác, cách sử dụng và bảo trì phần mềm (hướng dẫn sử dụng, tài liệu kỹ thuật, tài liệu phân tích, thiết kế, kiểm thử, v.v.).

1.2. Kiểm thử phần mềm và một số khái niệm liên quan

1.2.1. Kiểm thử phần mềm

Kiểm thử phần mềm là hoạt động khảo sát thực tiễn trong môi trường dự định sẽ 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 đó.

Mục đích của kiểm thử phần mềm là tìm ra các lỗi hay khiếm khuyết của phần mềm nhằm đảm bảo hiệu quả hoạt động tối ưu của phần mềm.

Tùy thuộc vào từng phương pháp, việc kiểm thử có thể được thực hiện bất cứ lúc nào trong quá trình phát triển phần mềm. Theo truyền thống thì các nỗ lực kiểm thử được tiến hành sau khi các yêu cầu được xác định và việc lập trình được hoàn tất nhưng trong Agile (là một tập hợp các phương pháp phát triển phần mềm linh hoạt dựa trên việc lặp đi lặp lại và gia tăng giá trị) thì việc kiểm thử được tiến hành liên tục trong suốt quá trình xây dựng phần mềm. Như vậy, mỗi một phương pháp kiểm thử bị chi phối theo một quy trình phát triển phần mềm nhất định.

1.2.2. Mục tiêu kiểm thử phần mềm

Trong lúc kiểm thử, công nghệ phần mềm phát sinh một chuỗi các trường hợp kiểm thử được sử dụng để "tách từng phần" phần mềm. Kiểm thử là một bước trong quy trình phần mềm mà có thể được xem xét bởi đội ngũ phát triển bằng cách phá vỡ thay vì xây dựng. Các kỹ sư phần mềm chính là những người xây dựng và kiểm thử yêu cầu họ vượt qua các khái niệm cho trước về độ chính xác và giải quyết mâu thuẫn khi các lỗi được xác định.

Các nguyên tắc được xem như mục tiêu kiểm thử là:

- Kiểm thử là một quá trình thực thi chương trình với mục đích tìm lỗi.
- Một trường hợp kiểm thử tốt là trường hợp kiểm thử mà có khả năng cao việc tìm thấy các lỗi chưa từng được phát hiện.
- Một kiểm thử thành công là kiểm thử mà phát hiện lỗi chưa từng được phát hiện.

1.2.3. Một số khái niệm liên quan

Chất lượng phần mềm (Software quality): là mức độ mà một hệ thống, thành phần hay quy trình đáp ứng các yêu cầu của đặc tả phần mềm, các nhu cầu mong đợi của khách hàng hoặc người sử dụng.

Đảm bảo chất lượng phần mềm (Software quality assurance): là một quy trình có kế hoạch và hệ thống của tất cả các hành động cần thiết để cung cấp các thông tin đầy đủ để đảm bảo các sản phẩm có phù hợp với các yêu cầu về kỹ thuật hay không. Mục đích cuối cùng là để đánh giá quy trình sản xuất sản phẩm phần mềm.

Xác nhận (Validation): là quá trình để đảm bảo rằng sản phẩm phần mềm đáp ứng được yêu cầu của người dùng.

Xác minh, kiểm chứng (Verification): là quá trình để đảm bảo rằng một sản phẩm phần mềm thỏa mãn đặc tả của n

Chúng ta cần thực hiện kiểm chứng trước khi thực hiện việc thẩm định sản phẩm. Thuật ngữ V&V là việc chúng ta cần đảm bảo sản phẩm đúng với đặc tả trước. Nếu thực hiện việc thẩm định trước, một khi phát hiện ra lỗi chúng ta sẽ không thể xác định được lỗi này do đặc tả sai hay hay do lập trình sai so với đặc tả.

Lỗi (Error): Lỗi là những vấn đề mà con người mắc phải trong quá trình phát triển các sản phẩm phần mềm.

Sai (Fault): Sai là kết quả của lỗi, hay nói khác đi, lỗi sẽ dẫn đến sai.

Thất bại (Failure): Thất bại xuất hiện khi một lỗi được thực thi.

Sự cố (Incident): Khi thất bại xuất hiện, nó có thể hiển thị hoặc không, tức là rõ ràng hoặc không rõ ràng đối với người dùng hoặc người kiểm thử. Sự cố là triệu chứng liên kết với một thất bại và thể hiện cho người dùng hoặc người kiểm thử về sự xuất hiện của thất bại này.

1.3. Nguyên tắc kiểm thử phần mềm

Có 7 nguyên tắc trong kiểm thử phần mềm:

Nguyên tắc 1: Kiểm thử đưa ra lỗi

Kiểm thử chỉ có thể chứng minh được rằng sản phẩm có lỗi. Kiểm thử phần mềm không thể chứng minh rằng sản phẩm không còn lỗi. Nghĩa là sản phẩm luôn có lỗi cho dù có kiểm thử nhiều bao nhiêu. Do đó, điều quan trọng là chúng ta phải thiết kế các trường hợp kiểm thử (test case) sao cho có thể tìm được càng nhiều lỗi càng tốt.

Nguyên tắc 2: Kiểm thử mọi thứ là không thể

Trừ khi sản phẩm được kiểm thử quá đơn giản cũng như không có nhiều giá trị đầu vào (chẳng hạn như "Hello World") thì việc chứng minh sản phẩm không còn bug cho dù có kiểm thử nhiều đến đâu là không khả thi. Hầu hết các sản phẩm ngày nay rất đa dạng và phức tạp do được phát triển trên nhiều nền tảng, công nghệ phong phú cũng như khả năng lưu trữ kết nối dữ liệu lớn, khiến việc kiểm thử trở nên khó khăn và việc kiểm thử toàn bộ là gần như không thể. Do đó, việc chúng ta có thể làm là chọn thực thi những loại kiểm thử quan trọng nhất dựa trên phân tích rủi ro cũng như tầm quan trọng và độ ưu tiên của việc kiểm thử. Nghĩa là phải lên kế hoạch kiểm thử, thiết kế trường hợp kiểm thử sao cho có độ bao phủ nhiều nhất và giảm thiểu rủi ro sót lỗi khi đến tay người dùng.

Nguyên tắc 3: Kiểm thử sớm

Nguyên tắc này yêu cầu bắt đầu thử nghiệm phần mềm trong giai đoạn đầu của vòng đời phát triển phần mềm. Các hoạt động kiểm thử phần mềm từ giai đoạn đầu sẽ giúp phát hiện bug sớm hơn. Nó cho phép chuyển giao phần mềm theo yêu cầu đúng thời gian với chất lượng dự kiến.

Nguyên tắc 4: Sự tập trung của lỗi

Trong quá trình kiểm thử, chúng ta sẽ có thể dễ dàng quan sát thấy đa phần những lỗi tìm được thường chỉ liên quan đến một vài tính năng của hệ thống. Điều này cũng thuận theo nguyên lý Pareto: 80% số lượng lỗi được tìm thấy trong 20% tính năng của hệ thống. Điều này cũng có nghĩa là khi tìm lỗi, chúng ta nên tập trung vào những module, thành phần chức năng chính của hệ thống. Một lỗi trên những module chính này có giá trị gấp nhiều lần lỗi tìm được trên những module phụ khác.

Nguyên tắc 5: Nghịch lý thuốc trừ sâu

Nếu bạn sử dụng cùng một tập hợp các trường hợp kiểm thử liên tục, sau đó một thời gian các trường hợp kiểm thử không tìm thấy lỗi nào mới. Hiệu quả của các trường hợp kiểm thử bắt đầu giảm xuống sau một số lần thực hiện, vì vậy luôn luôn chúng ta phải luôn xem xét và sửa đổi các trường hợp kiểm thử trên một khoảng thời gian thường xuyên.

Nguyên tắc 6: Kiểm thử theo các ngữ cảnh độc lập

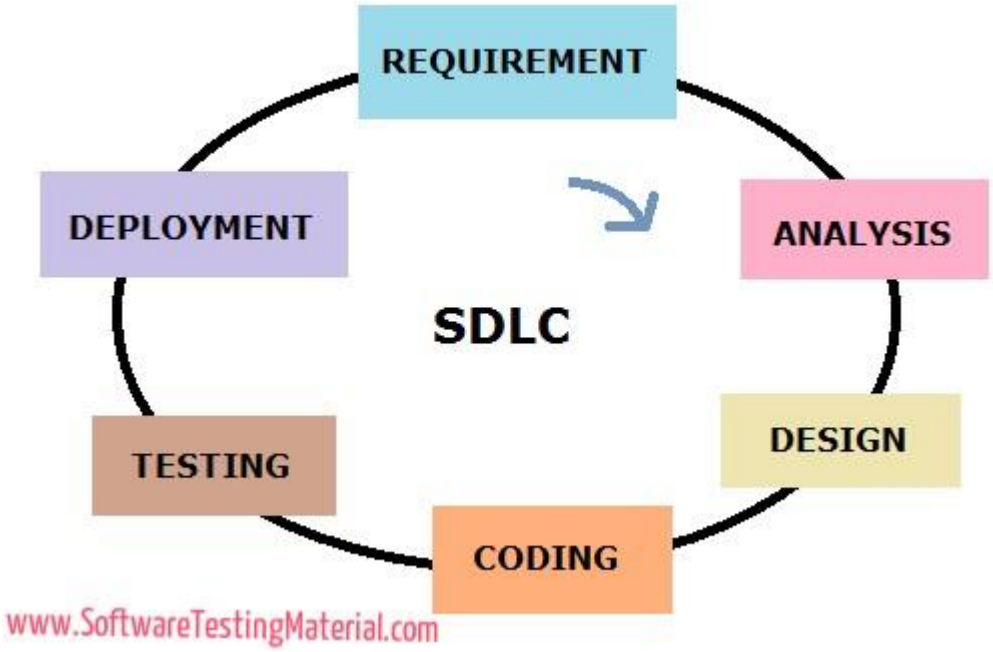
Theo nguyên tắc này thì việc kiểm thử phụ thuộc vào ngữ cảnh và chúng ta phải tiếp cận kiểm thử theo nhiều ngữ cảnh khác nhau Nếu bạn đang kiểm thử ứng dụng web và ứng dụng di động bằng cách sử dụng chiến lược kiểm thử giống nhau, thì đó là sai. Chiến lược để kiểm thử ứng dụng web sẽ khác với kiểm thử ứng dụng cho thiết bị di động của Android.

Nguyên tắc 7: Sự sai lầm về việc không có lỗi

Việc không tìm thấy lỗi trên sản phẩm không đồng nghĩa với việc sản phẩm đã sẵn sàng để tung ra thị trường. Việc không tìm thấy lỗi cũng có thể là do bộ trường hợp kiểm thử được tạo ra chỉ nhằm kiểm tra những tính năng được làm đúng theo yêu cầu thay vì nhằm tìm kiếm lỗi mới.

1.4. Vòng đời phát triển phần mềm(SDLC) và các mô hình phổ biến

1.4.1. Vòng đời phát triển phần mềm(SDLC)



Hình 2.1. Vòng đời phát triển phần mềm

Vòng đời phát triển phần mềm (SDLC – Software Development Life Cycle) là một quá trình theo sau cho một dự án phần mềm, trong một tổ chức phần mềm. Nó bao gồm một kế hoạch chi tiết mô tả làm thế nào để phát triển, duy trì, thay đổi hoặc nâng cấp phần mềm cụ thể.

- *Requirement:* Ở pha này bộ phận phân tích yêu cầu sẽ đi gặp và trao đổi với khách hàng cũng như làm rõ các chức năng, các yêu cầu mà khách hàng mong muốn xây dựng trong phần mềm của mình.

- *Analysis:* Đây là pha sẽ được thực hiện sau khi đã ghi nhận các yêu cầu của khách hàng về bộ phận phân tích sẽ thực hiện làm rõ các yêu cầu và hiện thực hóa bằng một tài liệu **SRS** gọi là "Tài liệu đặc tả". Bao gồm tất cả các yêu cầu sản phẩm được thiết kế và phát triển trong suốt vòng đời dự án. Các bộ phận liên quan như lập trình, kiểm thử viên,… sẽ thực hiện công việc dựa trên mô tả các chức năng chi tiết trong tài liệu và nó sẽ trả lời câu hỏi "Phần mềm sẽ làm gì ?".

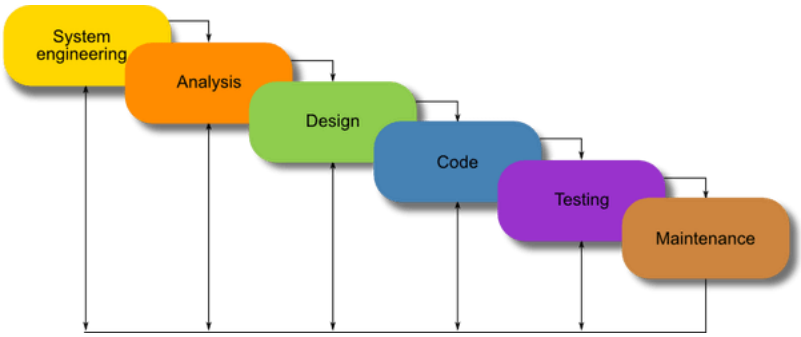
- *Design:* Ở pha này sau khi căn cứ vào tài liệu đặc tả, bộ phận thiết kế sẽ thực hiện thiết kế và tổng hợp tài liệu thiết kế: thiết kế tổng thể, thiết kế CSDL, thiết kế chi tiết.

- *Coding:* Ở pha này các lập trình viên (developer) sẽ lập trình xử lý chức năng, module theo yêu cầu được giao sau đó sẽ chuyển cho kiểm thử viên thực hiện kiểm tra các chức năng theo testcase được xây dựng dựa trên tài liệu đặc tả.

- *Testing:* Ở pha này, các kiểm thử viên sẽ nhận được các bàn giao chức năng thực hiện từ lập trình viên. Sau đó các kiểm thử viên sẽ thực hiện kiểm tra các chức năng này theo các testcase được xây dựng. Trong quá trình kiểm thử theo testcase nếu có phát sinh lỗi, kiểm thử viên sẽ thực hiện báo bug lỗi để lập trình viên xử lý chức năng đó fix lỗi.

- *Deployment:* Một khi sản phẩm sau khi được kiểm thử và sẵn sàng triển khai, sản phẩm được phát hành chính thức trên thị trường thích hợp. Trong quá trình sử dụng phần mềm của khách hàng, công ty phát triển phần mềm sẽ phải hỗ trợ, xử lý lỗi nếu có phát sinh trong quá trình sử dụng.

1.4.2. Mô hình thác nước



Hình 2.2. Mô hình thác nước

Đây được coi như là mô hình phát triển phần mềm đầu tiên dược sử dụng. Mô hình này được áp dụng tuần tự các giai đoạn của phát triển phần mềm. Đầu ra của giai đoạn trước là đầu vào của giai đoạn sau. Giai đoạn sau chỉ được thực hiện khi giai đoạnn trước đã kết thúc. Đặc biệt không đươc quay lại giai đoạn trước để xử lý các yêu cầu khi muốn thay đổi.

Áp dụng Waterfall khi nắm được rõ yêu cầu của dự án một cách tốt nhất, yêu cầu là rõ ràng và có tính ổn định cao. Nắm vững được công nghệ phát triển. Không có những yêu cầu không rõ ràng. Tài nguyên phát triển phong phú và chuyên môn kỹ thuật cao. Thích hợp với những dự án nhỏ và ngắn hạn.

- Ưu điểm

+ Dễ sử dụng, dễ tiếp cận, dễ quản lý.
+ Sản phẩm được phát triển theo các giai đoạn rõ ràng.
+ Xác nhận ở từng giai đoạn, đảm bảo phát hiện sớm các lỗi.

- Nhược điểm:

+ Ít linh hoạt, phạm vi điều chỉnh hạn chế.
+ Rất khó để đo lường sự phát triển trong từng giai đoạn.
+ Mô hình không thich hợp với những dự án dài, đang diễn ra, hay những dự án phức tạp, có nhiều thay đổi về yêu cầu trong vòng đời phát triển.
+ Khó quay lại khi giai đoạn nào đó kết thúc.

1.4.3. Mô hình Agile

Hình 2.3. Mô hình Agile

Dựa trên mô hình iterative and incremental. Các yêu cầu và giải pháp phát triển dựa trên sự kết hợp của các function. Các tác vụ được chia thành các khung thời gian nhỏ để cung cấp các tính năng cụ thể cho bản phát hành cuối.

Có thể được sử dụng với bất kỳ loại hình dự án nào, nhưng cần sự tham gia và tính tương tác của khách hàng. Sử dụng khi khách hàng yêu cầu chức năng sẵn sàng trong khoảng thời gian ngắn.

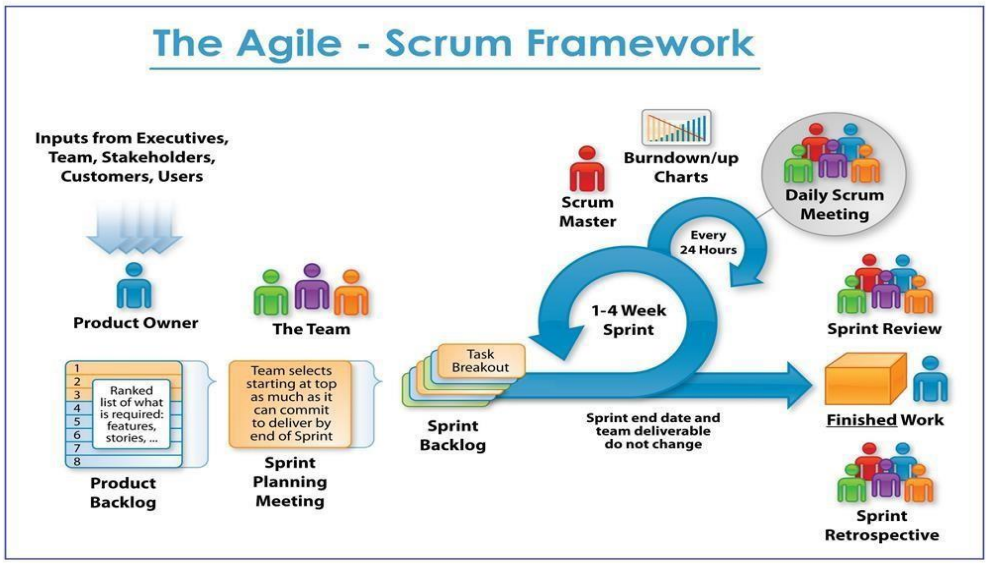
- Ưu điểm

+ Tăng cường tinh thần làm việc nhóm và trao đổi công việc hiệu quả.
+ Các chức năng được xây dựng nhanh chóng và rõ ràng, dễ quản lý.
+ Dễ dàng bổ, sung thay đổi yêu cầu.
+ Quy tắc tối thiểu, tài liệu dễ hiểu, dễ sử dụng.

- Nhược điểm

+ Không thích hợp để xử lý các phụ thuộc phức tạp.
+ Có nhiều rủi ro về tính bền vững khả năng bảo trì và khả năng mở rộng. Cần một nhóm có kinh nghiệm.
+ Phụ thuộc rất nhiều vào sự tương tác rõ ràng của khách hàng.
+ Chuyển giao công nghệ cho các thành viên mới trong nhóm có thể khó khăn do thiếu tài liệu.

1.4.4. Scrum Framework



Hình 2.4. Scrum framework

Scrum là một khung làm việc đã phát triển bền vững các sản phẩm phức tạp. Có thể hiểu đây là khung tổ chức công việc tổng quát hướng đến phát triển các sản phẩm phức tạp chủ yếu là phần mềm.

Scrum có ba vai trò, ba giá trị, bốn cuộc họp các yếu tố này được giải thích như sau:

- **Product Owner:** là người chịu trách nhiệm về sự thành công của dự án người định nghĩa các yêu cầu và đánh giá cuối cùng đầu ra của các nhà phát triển phần mềm.
- **Scrum Master:** là người có hiểu biết sâu sắc về Scrum và đảm bảo nhóm có thể làm việc hiệu quả với Scrum.
- **Development Team:** một nhóm liên chức năng(cross-functional) từ quản lý để tiến hành chuyển đổi các yêu cầu được tổ chức trong Product Backlog thành chức năng của hệ thống.
- **Sprint planning:** Cuộc họp lên kế hoạch của đội dự án, nhằm xác định những gì cần hoàn thành trong Sprint sắp tới.
- **Daily Scum:** Một cuộc họp nhỏ 15 phút mỗi ngày để trao đổi công việc giữa đội phát triển.
- **Sprint demo:** Một cuộc họp chia sẻ, nơi mà các thành viên chỉ ra những gì họ đã làm được trong Sprint đó.

- **Sprint retrospective:** Sự đánh giá, nhìn lại những gì đã làm được và chưa làm được của sprint hiện tại, và đưa ra giải pháp hành động cho Sprint tiếp theo được tốt và hoàn thiện hơn.

- Ưu điểm

+ Một người có thể làm nhiều việc ví dụ như dev có thể test
+ Phát hiện lỗi sớm hơn rất nhiều so với các phương pháp truyền thống
+ Khách hàng nhanh chóng thấy được sản phẩm qua đó đưa ra phản hồi sớm.
+ Có khả năng áp dụng được cho những dự án mà yêu cầu khách hàng không rõ ràng ngay từ đầu.

- Nhược điểm

+ Trình độ của nhóm là có một kỹ năng nhất định +
Phải có sự hiểu biết về mô hình agile.
+ Khó khăn trong việc xác định ngân sách và thời gian.
+ Luôn nghe ý kiến phản hồi từ khách hàng và thay đổi theo nên thời gian sẽ kéo dài khi có quá nhiều yêu cầu thay đổi từ khách hàng.
+ Vai trò của PO rất quan trọng, PO là người định hướng sản phẩm.Nếu PO làm không tốt sẽ ảnh hưởng đến kết quả chung.

1.5. Các cấp độ kiểm thử

1.5.1. Kiểm thử đơn vị

Một đơn vị kiểm thử là một thành phần phần mềm nhỏ nhất mà ta có thể kiểm thử được. Theo định nghĩa này, các hàm (Function), thủ tục (Procedure), lớp (Class), hoặc các phương thức (Method) đều có thể được xem là đơn vị kiểm thử.

1.5.2. Kiểm thử tích hợp

Kiểm thử tích hợp kết hợp các thành phần của một ứng dụng và kiểm thử như một ứng dụng đã hoàn thành. Trong khi kiểm thử đơn vị kiểm thử các thành phần và đơn vị phần mềm riêng lẻ thì kiểm thử tích hợp kết hợp chúng lại với nhau và kiểm thử sự giao tiếp giữa chúng.

Kiểm thử tích hợp có 2 mục tiêu chính:

- Phát hiện lỗi giao tiếp xảy ra giữa các đơn vị kiểm thử.
- Tích hợp các đơn vị kiểm thử 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.

1.5.3. Kiểm thử hệ thống

Mục đích kiểm thử hệ thống là kiểm tra thiết kế và toàn bộ hệ thống (sau khi tích hợp) có thỏa mãn yêu cầu đặt ra hay không. Điểm khác nhau then chốt giữa kiểm thử tích hợp và kiểm thử hệ thống là kiểm thử hệ thống chú trọng các hành vi và lỗi trên toàn hệ thống, còn kiểm thử tích hợp chú trọng sự giao tiếp giữa các đơn vị hoặc đối tượng khi chúng làm việc cùng nhau.

1.5.4. Kiểm thử chấp nhận

Thông thường, sau giai đoạn kiểm thử hệ thống là kiểm thử chấp nhận, được khách hàng thực hiện (hoặc ủy quyền cho một nhóm thứ ba thực hiện). Mục đích của kiểm thử chấp nhận là để chứng minh phần mềm thỏa mãn tất cả yêu cầu của khách hàng và khách hàng chấp nhận sản phẩm.

1.6. Các kỹ thuật kiểm thử phần mềm

1.6.1. Kỹ thuật kiểm thử hộp trắng (White-Box Testing)

Kiểm thử Hộp trắng là một phương pháp kiểm thử phần mềm trong đó tester biết về cấu trúc nội bộ / thiết kế. Người kiểm tra chọn đầu vào để thực hiện các đường dẫn thông qua mã và xác định đầu ra thích hợp. Kiến thức lập trình và kiến thức thực hiện là rất cần thiết trong kiểm thử hộp trắng. Phương pháp kiểm tra hộp trắng chủ yếu áp dụng cho kiểm thử đơn vị.

- Ưu điểm

+ Test có thể bắt đầu ở giai đoạn sớm hơn, không cần phải chờ đợi cho GUI để có thể test
+ Test kỹ càng hơn, có thể bao phủ hầu hết các đường dẫn
+ Thích hợp trong việc tìm kiếm lỗi và các vấn đề trong mã lệnh
+ Cho phép tìm kiếm các lỗi ẩn bên trong
+ Các lập trình viên có thể tự kiểm tra
+ Giúp tối ưu việc mã hoá
+ Do yêu cầu kiến thức cấu trúc bên trong của phần mềm, nên việc kiểm soát lỗi tối đa nhất.

- Nhược điểm

+ Vì các bài kiểm tra rất phức tạp, đòi hỏi phải có các nguồn lực có tay nghề cao, với kiến thức sâu rộng về lập trình và thực hiện.
+ Maintenance test script có thể là một gánh nặng nếu thể hiện thay đổi quá thường xuyên.
+ Vì phương pháp thử nghiệm này liên quan chặt chẽ với ứng dụng đang được test, nên các công cụ để phục vụ cho mọi loại triển khai / nền tảng có thể không sẵn có.

1.6.2. Kỹ thuật kiểm thử hộp đen (Black-Box Testing)

Kiểm thử hộp đen: là một phương pháp kiểm thử phần mềm được thực hiện mà không cần biết cấu trúc 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.
Phương pháp này cố gắng tìm ra các lỗi trong các loại sau:

- Chức năng không chính xác hoặc thiếu.
- Lỗi giao diện.
- Lỗi trong cấu trúc dữ liệu hoặc truy cập cơ sở dữ liệu bên ngoài.
- Hành vi hoặc hiệu suất lỗi.
- Khởi tạo và chấm dứt các lỗi.

- Ưu điểm

+ Các tester được thực hiện từ quan điểm của người dùng và sẽ giúp đỡ trong việc sáng tỏ sự chênh lệch về thông số kỹ thuật.
+ Các tester theo phương pháp black box không có "mối ràng buộc" nào với code, và nhận thức của một tester rất đơn giản: một source code có nhiều lỗi. Sử dụng nguyên tắc, "Hỏi và bạn sẽ nhận" các tester black box tìm được nhiều bug ở nơi mà các DEV không tìm thấy.
+ Tester có thể không phải IT chuyên nghiệp, không cần phải biết ngôn ngữ lập trình hoặc làm thế nào các phần mềm đã được thực hiện.
+ Các tester có thể được thực hiện bởi một cơ quan độc lập từ các developer, cho phép một cái nhìn khách quan và tránh sự phát triển thiên vị.
+ Hệ thống thật sự với toàn bộ yêu cầu của nó được kiểm thử chính xác.
+ Thiết kế kịch bản kiểm thử khá nhanh, ngay khi mà các yêu cầu chức năng được xác định.

- Nhược điểm

+ Dữ liệu đầu vào yêu cầu một khối lượng mẫu (sample) khá lớn
+ Nhiều dự án không có thông số rõ ràng thì việc thiết kế test case rất khó và do đó khó viết kịch bản kiểm thử do cần xác định tất cả các yếu tố đầu vào, và thiếu cả thời gian cho việc tập hợp này.
+ Khả năng để bản thân kỹ sư lạc lối trong khi kiểm thử là khá cao.
+ Chỉ có một số nhỏ các đầu vào có thể được kiểm tra và nhiều đường dẫn chương trình sẽ được để lại chưa được kiểm tra.
+ Kiểm thử black box được xem như "là bước đi trong mê cung tối đen mà không mang đèn pin" bởi vì tester không biết phần mềm đang test đã được xây dựng như thế nào. Có nhiều trường hợp khi một tester viết rất nhiều trường hợp test để kiểm tra một số thứ có thể chỉ được test bằng một trường hợp test và/hoặc một vài phần cuối cùng không được test hết.

1.6.3. Kỹ thuật kiểm thử hộp xám (Gray-Box Testing)

Kiểm thử hộp xám là sự kết hợp của cả 2 loại kiểm thử hộp đen và kiểm thử hộp trắng. Trong kiểm thử hộp trắng sự hiểu biết về cấu trúc bên trong hệ thống là thiết yếu và phải rõ ràng còn trong kiểm thử hộp đen sự hiểu biết này lại không quá cần thiết, nhưng với kiểm thử hộp xám người kiểm thử cần phải có một phần sự hiểu biết về cấu trúc của hệ thống cũng như là quyền truy cập vào cơ sở dữ liệu.

- Ưu điểm

+ Quan điểm kiểm thử của kiểm thử hộp xám là từ quan điểm của người dùng.
+ Cung cấp các lợi ích của cả thử nghiệm hộp đen và hộp trắng cùng nhau.
+ 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.

- Nhược điểm

+ Kiểm tra hộp xám cũng có thể mất nhiều thời gian để kiểm tra từng đường dẫn và đôi khi điều này là không thực tế.
+ Rất khó để liên kết lỗi khi thực hiện kiểm tra hộp xám cho một ứng dụng có hệ thống phân tán.
+ Thông thường sẽ dẫn đến phạm vi kiểm tra thấp hơn so với thực hiện kiểm tra hộp trắng và đen riêng biệt.
+ Có thể không phù hợp để thử nghiệm một số loại chức năng.

CHƯƠNG 2: KIỂM THỬ THỦ CÔNG

2.1. Khái niệm kiểm thử thủ công

Manual testing là việc thử nghiệm một phần mềm hoàn toàn được làm bằng tay bởi người tester. Nó được thực hiện nhằm phát hiện lỗi trong phần mềm đang được phát triển. Trong manual testing, tester sẽ thực hiện các trường hợp kiểm thử và tạo báo cáo kiểm thử hoàn toàn thủ công mà không có bất kỳ sự trợ giúp của công cụ tự động nào.

2.2. Mục tiêu của kiểm thử thủ công

Đảm bảo rằng ứng dụng hoạt động phù hợp với các yêu cầu chức năng được chỉ định. Test Suites hoặc cases, được thiết kế trong giai đoạn kiểm thử, có phạm vi kiểm thử 100%. Đảm bảo rằng các lỗi đã tìm thấy được sửa chữa bởi các developer và được kiểm thử lại bởi những tester sau khi các lỗi được khắc phục. Kiểm tra chất lượng của hệ thống và cung cấp sản phẩm không có lỗi cho khách hàng.

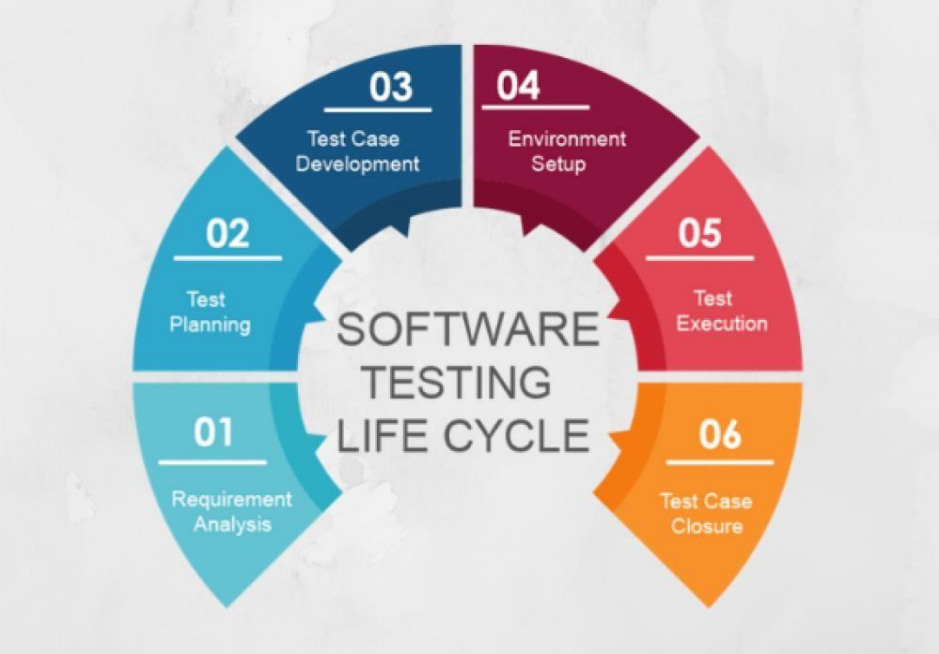
- Ưu điểm

+ Nhận phản hồi trực quan nhanh và chính xác
+ Ít tốn kém hơn vì không cần phải chi ngân sách cho các công cụ và quy trình tự động hóa
+ Sự phán đoán và trực giác của con người luôn có lợi cho yếu tố thủ công
+ Trong khi kiểm thử một thay đổi nhỏ, một kiểm thử tự động hóa sẽ yêu cầu coding có thể tốn thời gian. Trong khi kiểm thử thủ công lại không tốn thời gian.

- Nhược điểm

+ Phương pháp kiểm thử thủ công ít đáng tin cậy hơn vì được thực thi bởi con người. Do đó, dễ mắc sai lầm & không tìm thấy lỗi.
+ Quá trình kiểm thử thủ công không thể được ghi lại, do đó không thể sử dụng lại.
+ Một số phần nhất định khó thực hiện thủ công, có thể cần thêm thời gian.

2.3. Quy trình kiểm thử thủ công



Hình 2.12. Quy trình kiểm thử thủ công

Giai đoạn 1: Phân tích yêu cầu

Ở giai đoạn này, các tester sẽ nghiên cứu tài liệu được yêu cầu để có cái nhìn tổng quan về phần mềm. Từ đó có thể xác định được những yêu cầu cần kiểm tra. Đội Tester/QA có thể tương tác với phía khách hàng để hiểu được hệ thống thông qua trao đổi trực tiếp hoặc tạo file Q&A. Các yêu cầu ở đây là các yêu cầu về chức năng (xác định phần mềm sẽ phải làm gì) và các yêu cầu phi chức năng (hiệu năng hoạt động, bảo mật của phần mềm,…)

Giai đoạn 2: Lập kế hoạch kiểm thử

Kế hoạch kiểm thử là tài liệu tổng quan về kiểm thử: phạm vi của dự án, hướng tiếp cận, quy trình kiểm thử, tài nguyên và nhân lực, các tính năng được yêu cầu test và những tính năng không cần test, các tools mà quá trình kiểm thử cần nó. Kế hoạch kiểm thử sẽ là cơ sở để test sản phẩm phần mềm của dự án.

Giai đoạn 3: Thiết kế kịch bản kiểm thử

Trong giai đoạn này, tester sẽ thiết kế kịch bản kiểm thử theo các tài liệu đặc tả đã được cập nhật bằng cách sử dụng những kỹ thuật thiết kế test case. Các dữ liệu kiểm thử cũng được chuẩn bị từ giai đoạn này.

Giai đoạn 4: Thiết lập môi trường kiểm thử

Môi trường kiểm thử là 1 thiết lập của phần cứng và phần mềm để đội kiểm thử tiến hành kiểm tra các test case. Môi trường kiểm thử sẽ được quyết định dựa trên những yêu

cầu của khách hàng hoặc đặc thù của sản phẩm (server, client, network,…)

Giai đoạn 5: Thực thi kiểm thử

Trong giai đoạn này, tester sẽ thực hiện kiểm thử dựa trên kịch bản kiểm thử. Trong quá trình thực hiện kiểm thử, tester cũng có thể hỗ trợ, đề xuất cho cả đội dự án để đưa ra được các giải pháp hợp lý và kết hợp để quá trình kiểm thử đạt hiệu quả. Tester cần kiểm soát chặt chẽ tiến độ các công việc của mình bằng cách so sánh tiến độ thực tế với kế hoạch. Nếu chậm cần phải điều chỉnh ngay sao cho phù hợp với tiến độ của dự án. Kiểm thử viên cần thường xuyên báo cho PM và khách hàng về tình hình dự án, cung cấp các thông tin trong quá trình kiểm thử.

Giai đoạn 6: Đóng quy trình kiểm thử

Đây là giai đoạn cuối cùng trong quá trình kiểm thử phần mềm. Ở giai đoạn này, QA của team kiểm thử sẽ tổng kết, báo các kết quả về việc thực thi kiểm thử. Chức năng nào của phần mềm đã hoàn hoàn thành test/ chưa hoàn thành test/ chậm tiến độ. Ngoài ra, giai đoạn này cũng thảo luận những điểm tốt, chưa tốt và rút ra kinh nghiệm cho các dự án sau để cải thiện quy trình kiểm thử.

2.4. Ưu điểm của kiểm thử thủ công

- Nhận phản hồi trực quan nhanh và chính xác
- Ít tốn kém hơn vì không cần phải chi ngân sách cho các công cụ và quy trình tự động hóa
- Sự phán đoán và trực giác của con người luôn có lợi cho yếu tố thủ công
- Trong khi kiểm thử một thay đổi nhỏ, một kiểm thử tự động hóa sẽ yêu cầu coding có thể tốn thời gian. Trong khi kiểm thử thủ công lại không tốn thời gian.

2.5 Kỹ thuật thiết kế testcase

2.5.1. Cấu trúc của một testcase

- **Test Case ID**: Giá trị cần để xác định số lượng trường hợp cần để kiểm thử.
- **Description (Test Scenario)**: Mô tả sơ lược về mục đích của ca kiểm thử đó.
- **Pre-condition**: Điều kiện tiền đề nếu có.
- **Test step**: Các bước thực hiện 1 ca kiểm thử.
- **Test Data**: Những dữ liệu đầu vào cần chuẩn bị để test.
- **Step condition to perform**: Mô tả các bước thực hiện kiểm thử.
- **Expected results**: Kết quả mong đợi từ các bước thực hiện trên.
- **Actual result**: Kết quả thực tế khi chạy chương trình.
- **Result**: Đánh giá về kết quả, thông thường sẽ là pass, fail.
- **Note**: Cột này dùng để ghi chú những thông tin liên quan khi thực hiện ca kiểm thử.

Các bước xác định Ca kiểm thử:

Bước 1: Xác định mục đích kiểm thử: cần hiểu rõ đặc tả yêu cầu của khách hàng.

Bước 2: Xác định chức năng cần kiểm tra: cần phải biết làm thế nào phần mềm được sử dụng bao gồm các hoạt động, tổ chức chức năng khác nhau. Các bước thực hiện chỉ mô tả các bước thực hiện đứng từ phía người dùng cuối bao gồm nhập dữ liệu, nhấn button, v.v.

Bước 3: Xác định các yêu cầu phi chức năng: yêu cầu phần cứng, hệ điều hành, các khía cạnh an ninh.

Bước 4: Xác định biểu mẫu cho Ca kiểm thử: bao gồm giao diện UI, chức năng, khả năng tương thích và hiệu suất.

Bước 5: Xác định tính ảnh hưởng giữa các nguyên tắc mô-đun: mỗi một ca kiểm thử nên được thiết kế để có thể che phủ được sự ảnh hưởng của các mô-đun với nhau ở mức độ cao nhất. Dưới đây là minh họa của một ca kiểm thử

2.5.2. Phân vùng tương đương

Phân vùng tương đương là phương pháp chia các điều kiện đầu vào thành những vùng tương đương nhau. Tất cả các giá trị trong một vùng tương đương sẽ cho một kết quả đầu ra giống nhau. Vì vậy chúng ta có thể test một giá trị đại diện trong vùng tương đương. Mục đích sẽ giảm đáng kể số lượng ca kiểm thử cần phải thiết kế vì với mỗi lớp tương đương ta chỉ cần test trên các phần tử đại diện.

Thiết kế ca kiểm thử bằng kỹ thuật phân vùng tương đương tiến hành theo 2 bước:

- Xác định các lớp tương đương: ta chia miền dữ liệu kiểm thử thành các miền con sao cho dữ liệu trong mỗi miền con có cùng tính chất đối với chương trình. Sau khi chia miền dữ liệu của chương trình thành các miền con tương đương, ta chỉ cần chọn một phần tử đại diện của mỗi miền con này làm bộ dữ liệu kiểm thử. Các miền con này chính là các lớp tương đương.
- Xây dựng các ca kiểm thử tương ứng với mỗi lớp tương đương.

Ví dụ: kiểm thử phân vùng tương đương cho hàm NextDay. NextDay là một hàm gồm ba biến ngày, tháng, năm và chúng có các khoảng xác định như sau:

1 <= day <= 31

1 <= month <= 12

1812<=year<=2012

Phân vùng tương đương các vùng tương đương hợp lệ là:

D 1 = {day | 1 < = day < = 31 }

M 1 = {month | 1 < = month < = 12}

Y 1 = {year | 1812 < = year < = 2012}

Các lớp tương đương không hợp lệ là:

D 2 = {day | day < 1}

D 3 = {day | day > 31}

M 2 = {month | month < 1}

M 3 = {month | month > 12}

Y 2 = {year | year < 1812}

Y 3 = {year | year > 2012}

Dựa trên các lớp tương đương này ta có thể có bộ kiểm thử như Hình 2.6 (các giá trị hợp lệ được lấy giá trị giữa khoảng)

TT	Tháng	Ngày	Năm	Kết quả mong đợi
1	6	15	1912	15/6/1912
2	-1	15	1912	Ngày không hợp lệ
3	13	15	1912	Ngày không hợp lệ
4	6	-1	1912	Ngày không hợp lệ
5	6	32	1912	Ngày không hợp lệ
6	6	15	1811	Ngày không hợp lệ
7	6	15	2013	Ngày không hợp lệ

- Ưu điểm

Vì mỗi vùng tương đương ta chỉ cần kiểm tra trên các phần tử đại diện nên số lượng ca kiểm thử được giảm đi khá nhiều nhờ đó mà thời gian thực hiện kiểm thử cũng giảm đáng kể.

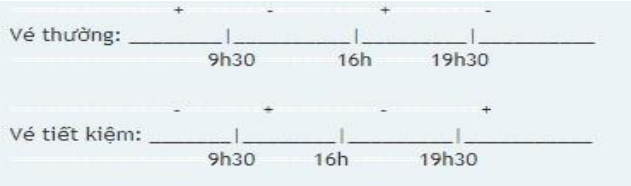
- Nhược điểm

Không phải với bất kỳ bài toán nào đều có thể áp dụng kỹ thuật này. Có thể bị thiếu sót lỗi ở biên nếu chỉ chọn giá trị ở khoảng giữa của miền tương đương.

2.5.3. Phân tích giá trị biên

Đây là phương pháp test mà chúng ta sẽ test tất cả các giá trị ở vùng biên của dữ liệu vào và dữ liệu ra. Chúng ta sẽ tập trung vào các giá trị biên chứ không test toàn bộ dữ liệu. Thay vì chọn nhiều giá trị trong lớp đương tương để làm đại diện, phân tích giá trị biên yêu cầu chọn một hoặc vài giá trị là các cạnh của lớp tương đương để làm điều kiện test.

Ví dụ: Nếu bạn đi xe điện chuyến trước 9:30 sáng hoặc từ sau 4:00 chiều đến 7:30 tối (giờ cao điểm), thì bạn phải mua vé thường. Vé tiết kiệm (giá thấp hơn vé thường) có hiệu lực cho các chuyến xe từ 9:30 sáng đến 4:00 chiều và sau 7:30 tối.



Hình 2.8: Hình liệt kê các vùng tương đương

Các vùng: (0 – 9h30), (9hh31 – 16h), (16h01 – 19h30), (19h31 – 23h59)

Các vùng hợp lệ là vùng thuộc dấu (+) và vùng không hợp là vùng thuộc dấu (-) Các giá trị biên: 9h30, 16h và 19h30.

Ta cần: 4 testcase để kiểm tra các vùng: (0 – 9h30), (9hh31 – 16h), (16h01 – 19h30), (19h31 – 23h59). 6 testcase kiểm tra các giá trị biên: 9h29, 9h31, 5h59, 16h01, 19h29 và 19h31. 1 testcase kiểm tra giá trị < 0 và 1 test case cho giá trị > 23h59

- Ưu điểm

Thay vì phải kiểm tra hết toàn bộ các giá trị trong từng vùng tương đương, kỹ thuật phân tích giá trị biên tập trung vào việc kiểm thử các giá trị biên của miền giá trị đầu vào để thiết kế ca kiểm thử do "lỗi thường tiềm ẩn tại các ngõ ngách và tập hợp tại biên" nên sẽ tiết kiệm thời gian thiết kế ca kiểm thử và thực hiện kiểm thử.

- Nhược điểm

Phương pháp phân tích giá trị biên chỉ hiệu quả trong trường hợp các đối số đầu vào độc lập với nhau và mỗi đối số đều có một miền giá trị hữu hạn.

2.5.4. Bảng quyết định

Bảng quyết định là một kỹ thuật test được sử dụng để kiểm tra các hành vi hệ thống (system behavior) với các cách kết hợp input đầu vào khác nhau. Đây là một cách tiếp cận có hệ thống, kết quả của các kết hợp đó và hành vi hệ thống tương ứng của chúng (output) sẽ được ghi lại dưới dạng bảng.

Ví dụ: Hệ thống máy rút tiền tự động ATM: Để có thể rút được tiền từ máy ATM, người dùng cần một trong hai điều kiện: tiền trong tài khoản vẫn còn và lớn hơn số tiền muốn rút (Đối với thẻ debit) hoặc là, người dùng được cấp cho một khoản tín dụng từ trước (Đối với thẻ Credit).

Điều kiện	**Trường hợp 1**	**Trường hợp 2**	**Trường 3**
Số tiền trong tài khoản lớn hơn số tiền định rút	Đúng (T)	Sai (F)	Sai (F)
Đã được cấp tín dụng	–	Đúng (T)	Sai (F)
Hành động của hệ thống			
Cho rút	Có (T)	Có (T)	Không (F)

Hình 2.9: Bảng quyết định

CHƯƠNG 3. KIỂM THỬ TỰ ĐỘNG

3.1. Khái niệm kiểm thử tự động

Automation Test có thể hiểu rất đơn giản là thay vì test bằng tay, ta để máy thực hiện việc testing mà tester phải làm (khởi động hệ thống, nhập dữ liệu đầu vào, kiểm tra so sánh với dữ liệu đầu ra và ghi kết quả). Automation Testing đóng một vai trò quan trọng góp phần nâng cao năng suất kiểm thử, giảm thiểu lỗi cũng như sự nhàm chán với việc kiểm thử bằng tay trong một thời gian dài hoặc lặp đi lặp lại.

Automation Test là một quá trình xử lý tự động các bước thực hiện một test case và được thực hiện bởi phần mềm là Automation Testing Tool. Mục đích của Tester là tìm bug nhưng mục đích cuối cùng vẫn là hỗ trợ để làm ra sản phẩm tốt nhất.

3.2. Mục tiêu của kiểm thử tự động

- Kiểm thử tự động trong các tình huống sau:

+ Không đủ tài nguyên: Khi số lượng TestCase quá nhiều mà tester không thể hoàn tất trong thời gian cụ thể.

+ Kiểm tra hồi quy: Nâng cấp phần mềm, kiểm tra lại các tính năng đã chạy tốt và những tính năng đã sửa. Tuy nhiên, việc này khó đảm bảo về mặt thời gian.

+Kiểm tra khả năng vận hành phần mềm trong môi trường đặc biệt (Đo tốc độ trung bình xử lý một yêu cầu của Web server, xác định cấu hình máy thấp nhất mà phần mềm vẫn có thể hoạt động tốt).

- Mục tiêu của kiểm thử tự động:
  + Giảm bớt công sức và thời gian thực hiện.
  + Tăng độ tin cậy.
  + Giảm sự nhàm chán.
  + Giảm chi phí tổng cho quá trình kiểm thử.

3.3. Quy trình kiểm thử tự động

Quy trình kiểm thử tự động phần mềm cũng giống như quy trình thực hiện kiểm thử thủ công chỉ khác ở chỗ kiểm thử tự động có hỗ trợ của công cụ ít hoặc nhiều như tạo test script (có thể bằng tay hoặc công cụ), công cụ hỗ trợ về ghi lại kết quả và lưu trữ kết quả trong máy tính. Quy trình này cũng gần tương tự với quy trình phát triển phần mềm, được thực hiện qua nhiều bước, được tiến hành rất sớm trong quy trình phát triển phần mềm và đội kiểm thử tiến hành gần như song song cùng đội phát triển phần mềm.

Hình 2.13. Quy trình kiểm thử

Giai đoạn 1: Lựa chọn công cụ kiểm thử.

Trước khi áp dụng Automation testing, bạn nên xác định mục tiêu. Bây giờ, khi bạn chắc chắn mình đang thực hiện loại kiểm tra nào, bạn cần chọn công cụ kiểm thử phần mềm. Bạn cần cân nhắc những điểm sau khi chọn công cụ: Nó có dễ dàng để phát triển và duy trì các script cho công cụ hay không? Nó có hoạt động trên các nền tảng như web, điện thoại di động, máy tính để bàn, v.v... không? Công cụ có chức năng báo cáo kiểm thử không? Công cụ này có thể hỗ trợ bao nhiêu loại kiểm thử? Công cụ hỗ trợ bao nhiêu ngôn ngữ?

Giai đoạn 2: Xác định phạm vi tự động hóa.

Tiếp theo, bạn cần xác định phạm vi tự động hóa. Vì vậy, bạn cần quyết định trường hợp kiểm thử nào sẽ tự động hóa dựa trên những điều sau: Các tình huống có một lượng lớn dữ liệu, những trường hợp thử nghiệm có chức năng chung trên các ứng dụng, tính khả thi về kỹ thuật, mức độ có thể sử dụng lại các thành phần của doanh nghiệp, sự phức tạp của các trường hợp kiểm thử.

Giai đoạn 3: Lập kế hoạch, thiết kế, phát triển.

Sau khi xác định mục tiêu và loại thử nghiệm nào cần tự động hóa, bạn nên quyết định những hành động mà Automation testing sẽ thực hiện. Việc lập kế hoạch, thiết kế và phát triển bao gồm:

Phát triển các trường hợp kiểm thử: Các bài kiểm tra tự động lớn, phức tạp luôn rất khó chỉnh sửa và gỡ lỗi. Tốt nhất nên chia các bài kiểm tra thành nhiều bài kiểm tra đơn giản, logic và nhỏ hơn.

Phát triển bộ kiểm thử: Bộ thử nghiệm đảm bảo rằng các trường hợp thử nghiệm tự động chạy lần lượt mà không cần bất kỳ sự can thiệp thủ công nào. Bây giờ, điều này có thể dễ dàng được thực hiện bằng cách tạo một bộ kiểm thử có nhiều trường hợp thử nghiệm, một thư viện và công cụ dòng lệnh chạy bộ kiểm thử.

Giai đoạn 4: Thực thi kiểm thử

Các script tự động hóa được thực thi trong giai đoạn này. Ngoài ra, việc thực thi có thể được thực hiện bằng cách sử dụng công cụ tự động hóa trực tiếp hoặc thông qua công cụ quản lý kiểm thử sẽ gọi công cụ tự động hóa.

Giai đoạn 5: Bảo trì

Khi các trường hợp kiểm thử được thực thi, bước tiếp theo là tạo báo cáo để những hành động thực hiện trong quá trình thử nghiệm được ghi lại. Khi các chức năng mới được thêm vào phần mềm mà bạn đang thử nghiệm với những chu kỳ liên tiếp, các script tự động hóa cần được thêm, xem xét và duy trì cho mỗi chu kỳ phát hành. Do đó, việc bảo trì trở nên cần thiết để nâng cao hiệu quả của tự động hóa.

3.4. Ưu điểm của kiểm thử tự động

- Độ tin cậy: Công cụ kiểm thử tự động mang tính ổn định cao hơn so với con người. Đặc biệt, trong những trường hợp nhiều test case nên độ tin cậy cậy của kiểm thử sẽ đạt mức tối ưu hơn so với kiểm thử thủ công.

- Khả năng lặp lại: Với những công cụ kiểm thử tự động sẽ ra đời nhiều các tester không phải trải qua quá trình lặp lại nhiều thao tác giảm được sự nhàm chán cũng như độ tin cậy cao và ổn định.

- Khả năng tái sử dụng: Đây chính là một bộ kiểm thử tự động được nhiều người sử dụng với nhiều những phiên bản khác nhau và được gọi là tái tính sử dụng.

- Tốc độ cao: Với tốc độ kiểm thử nhanh so hơn nhiều với tốc độ của con người. Những thực thi của một test case một cách thủ công có thể hoàn thành hay thực thi trong thời gian ngắn nhất một cách tự động.

3.5. Một số công cụ kiểm thử tự động

PROTRACTOR là một công cụ kiểm tra hướng hành vi tự động hóa và end-toend đóng vai trò quan trọng trong kiểm tra các ứng dụng AngularJS và hoạt động như một bộ tích hợp Giải pháp kết hợp các công nghệ mạnh mẽ như Selenium, Jasmine, Trình điều khiển web, v.v. Mục đích của Kiểm tra thước đo không chỉ để kiểm tra các ứng dụng AngularJS mà còn để viết các bài kiểm tra hồi quy tự động cho các Ứng dụng Web thông thường.

Selenium là một công cụ kiểm tra phần mềm được sử dụng để kiểm tra hồi quy. Đây là một công cụ kiểm tra mã nguồn mở cung cấp chức năng phát lại và thu âm để kiểm tra hồi quy. Các Selenium IDE chỉ hỗ trợ trình duyệt web Mozilla Firefox.

QTP (HP UFT) được sử dụng rộng rãi để kiểm thử chức năng và hồi quy, giải quyết các ứng dụng phần mềm và môi trường. Để đơn giản hóa việc tạo và bảo trì thử nghiệm, nó sử dụng khái niệm kiểm tra từ khóa.

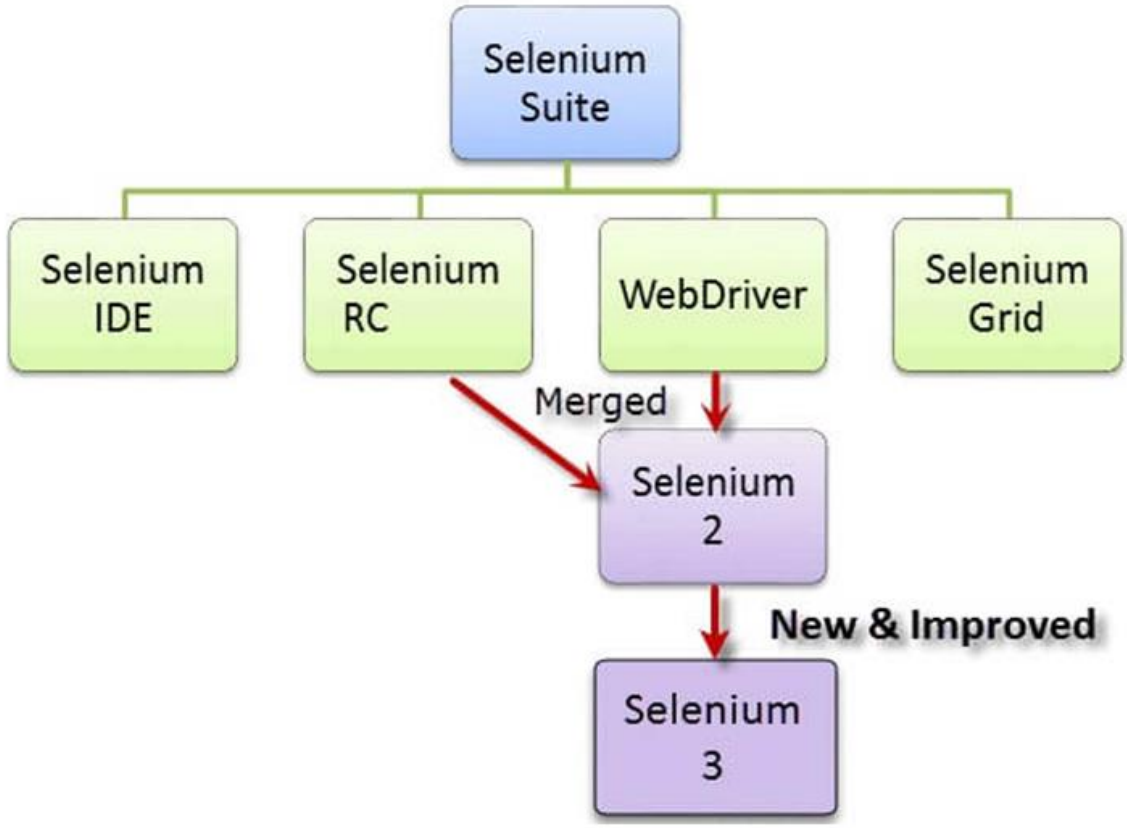
Rational Functional Tester là 1 công cụ kiểm tra tự động hướng đối tượng có khả năng tự động kiểm tra dữ liệu, kiểm tra giao diện, và kiểm thử hồi quy.

WATIR là một phần mềm kiểm tra mã nguồn mở để kiểm thử hồi quy. Watir chỉ hỗ trợ khám phá Internet trên các cửa sổ trong khi Watir webdriver hỗ trợ Chrome, Firefox, IE, Opera, v.v

3.6. Công cụ áp dụng - 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)
  - Selenium Remote Control (RC)
  - WebDriver
  - Selenium Grid



3.6.1. Selenium Integrated Development Environment (IDE)

Selenium IDE là viết tắt của cụm từ Môi trường phát triển tích hợp Selenium, là một công cụ kiểm thử tự động được phát hành dưới dạng trình cắm Firefox (Firefox plug-in). Đây là một trong những công cụ đơn giản và dễ dàng sử dụng nhất để cài đặt, tìm hiểu và tạo các kịch bản test. Công cụ này được đặt trên một bản ghi, có thể phát lại tập lệnh và cũng cho phép chỉnh sửa các tập lệnh được ghi (chỉnh sửa kịch bản kiểm thử).

3.6.2. Selenium Remote Control (RC)

Selenium Remote Control (RC) là dự án Selenium chính duy trì trong một thời gian dài trước khi Selenium WebDriver (Selenium 2.0) ra đời. Hiện nay Selenium RC hầu như không được sử dụng vì WebDriver cung cấp nhiều tính năng mạnh mẽ hơn, tuy nhiên người dùng vẫn có thể tiếp tục phát triển các script bằng RC.

Nó cho phép chúng tôi viết các bài kiểm tra giao diện người dùng ứng dụng web tự động với sự trợ giúp của toàn bộ sức mạnh của các ngôn ngữ lập trình như Java, C #, Perl, P.

Perl, Python và PHP để tạo các bài kiểm tra phức tạp hơn như đọc và ghi tệp, truy vấn cơ sở dữ liệu và gửi kết quả kiểm tra qua email .

3.6.3. Selenium WebDriver

WebDriver là một công cụ để tự động thử nghiệm các ứng dụng web. Nó thường được gọi là Selenium 2.0. WebDriver sử dụng một khung cơ bản khác, trong khi Selenium RC sử dụng JavaScript Selenium-Core được nhúng trong trình duyệt có một số hạn chế. WebDriver tương tác trực tiếp với trình duyệt mà không cần bất kỳ trung

gian nào, không giống như Selenium RC phụ thuộc vào máy chủ. Nó được sử dụng trong ngữ cảnh sau:

Kiểm tra nhiều trình duyệt bao gồm cải thiện chức năng cho các trình duyệt không được hỗ trợ tốt bởi Selenium RC (Selenium 1.0).

- Xử lý nhiều khung, nhiều cửa sổ trình duyệt, cửa sổ bật lên và cảnh báo.
- Điều hướng trang phức tạp.
- Điều hướng người dùng nâng cao chẳng hạn như kéo và thả.
- Các phần tử giao diện người dùng dựa trên AJAX.

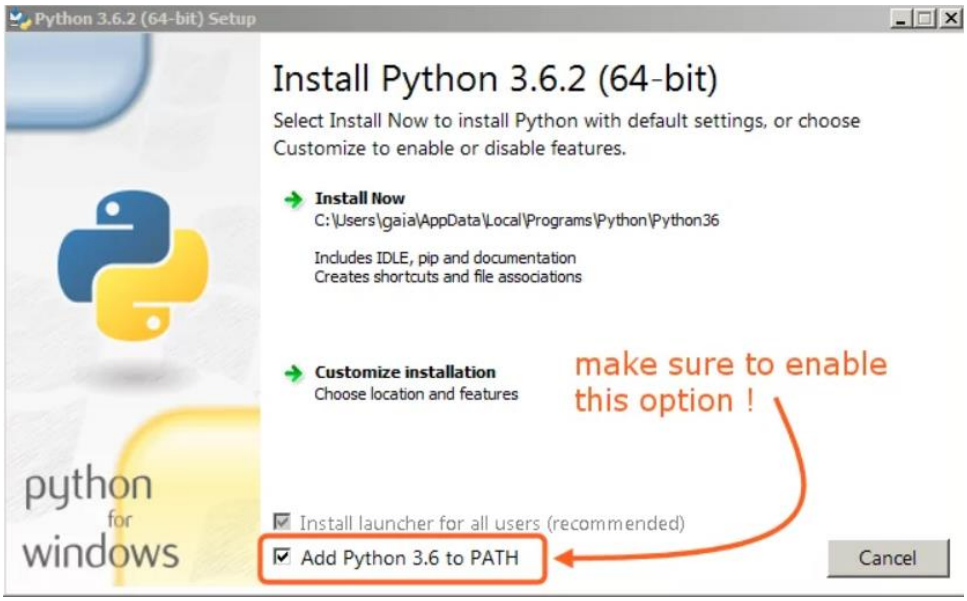
3.6.4. Selenium Grid

Thực hiện phương pháp kiểm tra phân bố , phối hợp nhiều kết quả của Selenium RC để có thể thực thi trên nhiều trình duyệt web khác nhau trong cùng mọt lúc. Cũng cho phép lưu lại kết quả kiểm tra .

3.7. Cài đặt Selenium WebDriver Python.

3.7.1. Cài đặt Python trên Windows

Bạn hãy vào trang https://www.python.org/downloads/ để download phiên bản Python mà bạn muốn cài.



3.7.2. Cài đặt PyCharm IDE hoặc Visual Studio Code.

Download công cụ để dễ dàng thực hiện viết kiểm thử tự động.

3.7.3. Cài đặt môi trường sử dụng Selenium

Cài đặt Selenium với Python.

Việc cài đặt Selenium với ngôn ngữ Python cực kỳ đơn giản, bạn chỉ cần chạy command sau:

```
0
1 pip install selenium
2
```

Tải Selenium WebDriver

Các bạn nên dùng WebDriver FireFox hoặc Chrome. Trong series này mình sẽ dùng Chrome driver(geckodriver) nhé.

Bạn chỉ cần tải : Chrome Driver: http://chromedriver.chromium.org/downloads

Sau khi tải xong, bạn hãy giải nén và để file chromedriver vào cùng folder với file Python của bạn

CHƯƠNG 4: KẾT QUẢ THỰC HIỆN

4.1 Mô tả

Sau thời gian tìm hiểu và nghiên cứu chúng em đã thu thập được các kiến thức để xây dựng một bản demo về Manual testing và Automation testing được thực hiện trên website https://www.expedia.com/.

4.2 Các testcase phải thực hiện

ID	Summary	Precondition	Steps	Expected Result
TC_1	Search for flights from HaNoi to DaNang sorting prices from low to high	Go to website https://www.expedia.com	Select More Travel -> Flights Select "Hanoi" in Leaving from Select "Da Nang" in Going to Click Search Select "Price (Lowest)" in Sort By filter	Go to search Flights successful. Only show the flights from "Hanoi" to "Da Nang" The prices are sorted from Lowest to Highest
TC_2	Search for flights from HaNoi to DaNang sorting prices from high to low	Go to website https://www.expedia.com	Select More Travel -> Flights Select "Hanoi" in Leaving from Select "Da Nang" in Going to Click Search Select "Price (Highest)" in Sort By filter	Go to search Flights successful. Only show the flights from "Hanoi" to "Da Nang" The prices are sorted from Highest to Lowest
TC_3	Search for flights from HaNoi to DaNang Sort time from shortest to longest	Go to website https://www.expedia.com	Select More Travel -> Flights Select "Hanoi" in Leaving from Select "Da Nang" in Going to Click Search Select "Duration (Shortest)" in Sort By filter	Go to search Flights successful. Only show the flights from "Hanoi" to "Da Nang" The Time of Flights are sorted from Shortest to Longest
TC_4	Search for flights from HaNoi to DaNang Sort time from longest to shortest	Go to website https://www.expedia.com	Select More Travel -> Flights Select "Hanoi" in Leaving from Select "Da Nang" in Going to Click Search Select "Duration (Longest)" in Sort By filter	Go to search Flights successful. Only show the flights from "Hanoi" to "Da Nang" The Time of Flights are sorted from Longestto Shortest

4.3 Kiểm thử tự động

Dựa trên các testcase đã xây dựng nhóm đã tiến hành triển khai thực hiện kiểm thử tự động bằng ngôn ngữ Python dựa trên framework Selenium.

Testcase 01

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import unittest
import time
Leaving_from = "Ha Noi"
Going_to = "Da Nang"
Local_HaNoi = f".//ul[@class='no-bullet']//button[contains(string(), 'Noi Bai')]"
verify_mess = "Hanoi (HAN) - Da Nang (DAD)"
class Testcase2(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome()
        time.sleep(3)
        self.base_url = "https://www.expedia.com/"
    def test_case2(self):
        driver = self.driver
        driver.get(self.base_url)
        time.sleep(5)
        driver.find_element_by_xpath('.//button[contains(string(),"More travel")]').click()
        time.sleep(5)
        driver.find_element_by_xpath('.//a[@href="/Flights"]').click()
        time.sleep(45)
        driver.find_element_by_xpath('.//button[@aria-label="Leaving from"]').click()
        driver.find_element_by_xpath('.//input[@placeholder="Where are you leaving from?"]').send_keys(Leaving_from)
        time.sleep(5)
        driver.find_element_by_xpath(Local_HaNoi).click()
        time.sleep(5)
        driver.find_element_by_xpath('.//button[@aria-label="Going to"]').click()
        driver.find_element_by_xpath('.//input[@placeholder="Where are you going to?"]').send_keys(Going_to + Keys.ENTER)
        time.sleep(3)
        driver.find_element_by_xpath('.//button[@data-testid="submit-button"]').click()
        time.sleep(30)
        driver.find_element_by_xpath('.//select[@id="listings-sort"]').click()
        time.sleep(1)
        driver.find_element_by_xpath('.//select[@id="listings-sort"]/option[contains(string(), "Price (Lowest)")]').click()
        pricelist =[]
        for j in range(1,9):
            priceitem = driver.find_element_by_xpath('.//ul/li['+(str(j))+']//span[@class="uitk-lockup-price"]')
            priceitem = priceitem.text
            priceitem = priceitem.split('$')
            priceitem = priceitem[1]
            priceitem = str(priceitem)
            pricelist.append(priceitem)
        verify_local  =  driver.find_elements_by_xpath('.//div[@data-test-id="arrival-departure"]')
        if any(verify_mess in e.text for e in verify_local):
            for i in range(len(pricelist)):
                for j in range(len(pricelist)-1):
                    if(int(pricelist[i]) <= int(pricelist[j])):
                        print("Testcase PASS")
        else:
            print("Testcase FAIL")
```

Kết quả:

```
Testcase PASS
.
----------------------------------------------------------------------
Ran 1 test in 147.829s

OK
```

Testcase 02

```python
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import unittest
import time
Leaving_from = "Ha Noi"
Going_to = "Da Nang"
Local_HaNoi = f".//ul[@class='no-bullet']//button[contains(string(), 'Noi Bai')]"
verify_mess = "Hanoi (HAN) - Da Nang (DAD)"
class Testcase3(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome()
        time.sleep(3)
        self.base_url = "https://www.expedia.com/"
    def test_case(self):
        driver = self.driver
        driver.get(self.base_url)
        time.sleep(5)
        driver.find_element_by_xpath('.//button[contains(string(),"More travel")]').click()
        time.sleep(5)
        driver.find_element_by_xpath('.//a[@href="/Flights"]').click()
        time.sleep(45)
        driver.find_element_by_xpath('.//button[@aria-label="Leaving from"]').click()
        driver.find_element_by_xpath('.//input[@placeholder="Where are you leaving from?"]').send_keys(Leaving_from)
        time.sleep(5)
        driver.find_element_by_xpath(Local_HaNoi).click()
        time.sleep(5)
        driver.find_element_by_xpath('.//button[@aria-label="Going to"]').click()
        driver.find_element_by_xpath('.//input[@placeholder="Where are you going to?"]').send_keys(Going_to + Keys.ENTER)
        time.sleep(3)
        driver.find_element_by_xpath('.//button[@data-testid="submit-button"]').click()
        time.sleep(30)
        driver.find_element_by_xpath('.//select[@id="listings-sort"]').click()
        time.sleep(1)
        driver.find_element_by_xpath('.//select[@id="listings-sort"]/option[contains(string(), "Price (Highest)")]').click()
        time.sleep(5)
        pricelist =[]
        for j in range(1,9):
            priceitem = driver.find_element_by_xpath('.//ul/li['+(str(j))+']//span[@class="uitk-lockup-price"]')
            priceitem = priceitem.text
            priceitem = priceitem.split('$')
            priceitem = priceitem[1]
            priceitem = str(priceitem)
            pricelist.append(priceitem)
        print(pricelist)
        verify_local  =  driver.find_elements_by_xpath('.//div[@data-test-id="arrival-departure"]')
        if any(verify_mess in e.text for e in verify_local):
            for i in range(len(pricelist)):
                for j in range(len(pricelist)-1):
                    if(int(pricelist[i]) >= int(pricelist[j])):
                        print("Testcase PASS")
```

```python
    def tearDown(self):
        self.driver.quit()

if __name__ == "__main__":
    unittest.main()
```

Kết quả:

```
Testcase PASS
.
----------------------------------------------------------------------
Ran 1 test in 147.829s

OK
```

Testcase 03

```python
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import unittest
import time
Leaving_from = "Ha Noi"
Going_to = "Da Nang"
Local_HaNoi = f".//ul[@class='no-bullet']//button[contains(string(), 'Noi Bai')]"
verify_mess = "Hanoi (HAN) - Da Nang (DAD)"
class Testcase4(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome()
        #self.driver = webdriver.Chrome("C:/Users/internntrantuyen/Downloads/chromedriver.exe")
        time.sleep(3)
        self.base_url = "https://www.expedia.com/"

    def test_case4(self):
        driver = self.driver
        driver.get(self.base_url)
        time.sleep(5)
        driver.find_element_by_xpath('.//button[contains(string(),"More travel")]').click()
        time.sleep(5)
        driver.find_element_by_xpath('.//a[@href="/Flights"]').click()
        time.sleep(45)
        driver.find_element_by_xpath('.//button[@aria-label="Leaving from"]').click()
        driver.find_element_by_xpath('.//input[@placeholder="Where are you leaving from?"]').send_keys(Leaving_from)
        time.sleep(5)
        driver.find_element_by_xpath(Local_HaNoi).click()
        time.sleep(5)
        driver.find_element_by_xpath('.//button[@aria-label="Going to"]').click()
        driver.find_element_by_xpath('.//input[@placeholder="Where are you going to?"]').send_keys(Going_to + Keys.ENTER)
        time.sleep(3)
        driver.find_element_by_xpath('.//button[@data-testid="submit-button"]').click()
        time.sleep(30)
        driver.find_element_by_xpath('.//select[@id="listings-sort"]').click()
        time.sleep(1)
        driver.find_element_by_xpath('.//select[@id="listings-sort"]/option[contains(string(), "Duration (Shortest)")]').click()
        time.sleep(5)
        timelist =[]
        for j in range(1,9):
            time_f = driver.find_element_by_xpath('.//ul/li['+(str(j))+']//div[@data-test-id="journey-duration"]')
            time_f = time_f.text
            time_f = time_f.split()
            time_h = time_f[0]
            time_h = time_h.split('h')
            time_h = time_h[0]
            print(time_h)
            time_h = int(time_h)
            time_m = time_f[1]
            time_m = time_m.split('m')
            time_m = time_m[0]
            time_m = time_m[0]
            print(time_m)
            time_m = int(time_m)
            print('type h',type(time_h))
            print('type m' ,type(time_h))
            time_h =  time_h*60
            time_target = time_h + time_m
            print(time_target)
            timelist.append(time_target)
        print(timelist)
        verify_local  =  driver.find_elements_by_xpath('.//div[@data-test-id="arrival-departure"]')
        if any(verify_mess in e.text for e in verify_local):
            for i in range(len(timelist)):
                for j in range(len(timelist)-1):
                    if(int(timelist[i]) <= int(timelist[j])):
                        print("Testcase PASS")
        else:
            print("Testcase FAIL")
    def tearDown(self):
        self.driver.quit()

if __name__ == "__main__":
    unittest.main()
```

Kết quả:

```
Testcase PASS
.
----------------------------------------------------------------------
Ran 1 test in 147.829s

OK
```

Testcase 04

```python
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import unittest
import time
Leaving_from = "Ha Noi"
Going_to = "Da Nang"
Local_HaNoi = f".//ul[@class='no-bullet']//button[contains(string(), 'Noi Bai')]"
verify_mess = "Hanoi (HAN) - Da Nang (DAD)"
class Testcase5(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome()
        #self.driver = webdriver.Chrome("C:/Users/internntrantuyen/Downloads/chromedriver.exe")
        time.sleep(3)
        self.base_url = "https://www.expedia.com/"
    def test_case5(self):
        driver = self.driver
        driver.get(self.base_url)
        time.sleep(5)
        driver.find_element_by_xpath('.//button[contains(string(),"More travel")]').click()
        time.sleep(5)
        driver.find_element_by_xpath('.//a[@href="/Flights"]').click()
        time.sleep(45)
        driver.find_element_by_xpath('.//button[@aria-label="Leaving from"]').click()
        driver.find_element_by_xpath('.//input[@placeholder="Where are you leaving from?"]').send_keys(Leaving_from)
        time.sleep(5)
        driver.find_element_by_xpath(Local_HaNoi).click()
        time.sleep(5)
        driver.find_element_by_xpath('.//button[@aria-label="Going to"]').click()
        driver.find_element_by_xpath('.//input[@placeholder="Where are you going to?"]').send_keys(Going_to + Keys.ENTER)
        time.sleep(3)
        driver.find_element_by_xpath('.//button[@data-testid="submit-button"]').click()
        time.sleep(30)
        driver.find_element_by_xpath('.//select[@id="listings-sort"]').click()
        time.sleep(1)
        driver.find_element_by_xpath('.//select[@id="listings-sort"]/option[contains(string(), "Duration (Longest)")]').click()
        time.sleep(5)
        timelist =[]
        for j in range(1,9):
            time_f = driver.find_element_by_xpath('.//ul/li['+(str(j))+']//div[@data-test-id="journey-duration"]')
            time_f = time_f.text
            time_f = time_f.split()
            time_h = time_f[0]
            time_h = time_h.split('h')
            time_h = time_h[0]
            print(time_h)
            time_h = int(time_h)
            time_m = time_f[1]
            time_m = time_m.split('m')
            time_m = time_m[0]
            print(time_m)
            time_m = int(time_m)
            print('type h',type(time_h))
            print('type m' ,type(time_h))
            time_h =  time_h*60
            time_target = time_h + time_m
            print(time_target)
            timelist.append(time_target)
        print(timelist)
        verify_local  =  driver.find_elements_by_xpath('.//div[@data-test-id="arrival-departure"]')
        if any(verify_mess in e.text for e in verify_local):
            for i in range(len(timelist)):
                for j in range(len(timelist)-1):
                    if(int(timelist[i]) >= int(timelist[j])):
                        print("Testcase PASS")
        else:
            print("Testcase FAIL")
    def tearDown(self):
        self.driver.quit()

if __name__ == "__main__":
    unittest.main()
```

Kết quả:

```
Testcase PASS
.
----------------------------------------------------------------------
Ran 1 test in 147.829s

OK
```

4.4 Kết quả kiểm thử

Sau khi thực hiện chạy các testcase thì sẽ thu được kết quả như sau:

ID	Summary	Precondition	Steps	Expected Result	Actual Result	Result
TC_1	Search for flights from HaNoi to DaNang sorting prices from low to high	Go to website https://www.expedia.com	Select More Travel -> Flights Select "Hanoi" in Leaving from Select "Da Nang" in Going to Click Search Select "Price (Lowest)" in Sort By filter	Go to search Flights successful. Only show the flights from "Hanoi" to "Da Nang" The prices are sorted from Lowest to Highest	Go to search Flights successful. Only show the flights from "Hanoi" to "Da Nang" The prices are sorted from Lowest to Highest	PASS
TC_2	Search for flights from HaNoi to DaNang sorting prices from low to high	Go to website https://www.expedia.com	Select More Travel -> Flights Select "Hanoi" in Leaving from Select "Da Nang" in Going to Click Search Select "Price (Highest)" in Sort By filter	Go to search Flights successful. Only show the flights from "Hanoi" to "Da Nang" The prices are sorted from Highest to Lowest	Go to search Flights successful. Only show the flights from "Hanoi" to "Da Nang" The prices are sorted from Highest to Lowest	PASS
TC_3	Search for flights from HaNoi to DaNang sorting prices from low to high	Go to website https://www.expedia.com	Select More Travel -> Flights Select "Hanoi" in Leaving from Select "Da Nang" in Going to Click Search Select "Duration (Shortest)" in Sort By filter	Go to search Flights successful. Only show the flights from "Hanoi" to "Da Nang" The Time of Flights are sorted from Shortest to Longest	Go to search Flights successful. Only show the flights from "Hanoi" to "Da Nang" The Time of Flights are sorted from Shortest to Longest	PASS
TC_4	Search for flights from HaNoi to DaNang sorting prices from low to high	Go to website https://www.expedia.com	Select More Travel -> Flights Select "Hanoi" in Leaving from Select "Da Nang" in Going to Click Search Select "Duration (Longest)" in Sort By filter	Go to search Flights successful. Only show the flights from "Hanoi" to "Da Nang" The Time of Flights are sorted from Longestto Shortest	Go to search Flights successful. Only show the flights from "Hanoi" to "Da Nang" The Time of Flights are sorted from Longestto Shortest	PASS

Kết Luận:

Trong thời điểm hiện tại, khi mà nhu cầu về phần mềm mới ngày càng nhiều và phức tạp hơn – thì yêu cầu tối ưu kiểm thử trước khi đưa đến tay người sử dụng càng đòi hỏi cấp thiết hơn. Việc tự động hóa một phần quy trình kiểm thử sẽ giúp hỗ trợ các tester nâng cao nâng suất và chất lượng dự án. Tuy nhiên, chi phí cho các phần mềm kiểm thử thương mại thường rất cao, đôi khi làm cho dự án không có khả năng sinh lời. Vì thế chọn lựa các công cụ kiểm tra tự động mã nguồn mở là một lựa chọn rất sáng suốt một trong số các lựa chọn đó.

Báo cáo đã trình bày cơ bản đầy đủ về kiến thức kiểm thử phần mềm bao gồm kiểm thử thủ công và kiểm thử tự động. Sản phẩm thực hiện kiểm thử tự động dựa trên công cụ Selenium WebDriver.