-----🙞🙜🕮🙞🙜-----

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐÔNG Á**



BÁO CÁO MÔN HỌC

**Kiểm Thử Phần Mềm**

**Đề tài:**

**“ Báo cáo về kiểm thử phần mềm và selenium “**

**GVHD: Ths. Tạ Quốc Ý**

**SVTH: Lê Quang Mẫn - 49116**

**Lớp:** **ST19A1B**

**LỜI MỞ ĐẦU**

Do sự phất triển mạnh của công nghệ thông tin những năm gần đây, ngành công nghệ dần chiếm một vị trí không thể thiếu trong đời sống. Điều đó khiến khiến cho các nhà phát triển dần phải nâng cao chất lượng sản phầm của mình cũng nhưu tối ưu hiệu suất sản phẩm . Vì vậy kiểm thử phần mềm đang ngày càng đóng vai trò quan trọng trong ngành công nghiệp phát triển phần mềm không chỉ ở Việt Nam mà trên thế giới. Kiểm thử phần mềm là một khâu rất quan trọng trong quá trình phát triển phần mềm. Kiểm thử phần mềm để kiểm tra phần mềm có đúng với đặc tả và thiết kế hệ thống không, có đáp ứng yêu cầu người dùng không, có lỗi lập trình không, hoạt động có hiệu quả không,…Như vậy, kiểm thử phần mềm là để đáp ứng yêu cầu người dùng, phát triển lỗi để từ đó nâng cao chất lượng phần mềm.

Với sự giúp đỡ của thầy Tạ Quốc Ý hôm nay em xin làm báo cáo về đề tài Kiểm thử phầm mềm và selenium.

1. **Kiểm Thử Phần Mềm ( Software testing** ) :

Kiểm thử phần mềm (Software testing) là hoạt động nhằm tìm kiếm, phát  
hiện các lỗi của phần mềm, đảm bảo sản phẩm phần mềm đáp ứng chính xác, đầy đủ và đúng theo yêu cầu của khách hàng đã đặt ra; cung cấp mục tiêu, cái nhìn độc lập về phần mềm, cho phép việc đánh giá và hiểu rõ hơn các rủi ro khi thực thiphần mềm, tạo điều kiện cho bạn tận dụng tối đa tư duy đánh giá và sáng tạo đểbạn có thể phát hiện ra những điểm mà người khác chưa tìm thấy.

**1.1 Nguyên Nhân Lỗi Phần mềm :**

* Định nghĩa sai yêu cầu của khách hàng: hiểu sai yêu cầu, triển khai thiếu yêu cầu , khách hàng đưa ra quá nhiều yêu cầu
* Thất bại trong việc giao tiếp giữa người phát triển và khách hàng: thiếu sự chú ý tới yêu cầu khách hàng, những thay đổi của khác khôgn được lưu dưới dạng văn bản
* Lỗi logic trong thiết kế phần mềm: thiết kế sai thuật toán, ghi nhận sai tuần tự, ghi nhận sai các điều kiện biên, ghi nhận sai các trạng thái của hệ thống
* Lỗi tài liệu:
* Lỗi mã hóa: lỗi logic, lỗi cú pháp, lỗi thời gian chạy
* Không tuân theo tài liệu và cấu trúc code
* Rút ngắn quá trình kiểm thử

**1.2 Mục tiêu của kiểm thử :**

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

**1.3 Phân loại kiểm thử**

Dựa trên các yếu tố:

* Mục đích kiểm thử
* Chiến lược kiểm thử
* Phương pháp kiểm thử
* Kỹ thuật kiểm thử

**1.4 Mục đích kiểm thử**

Có 2 loại chiến lược kiểm thử đó là Manul Testing và Automation Testing:

* Manual Testing : là việc thử nghiệm một phầm mềm hoàn toàn được làm bằng tay bởi người tester. Nó được thực hiện nhằm phát hiện lỗi trong phầm 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.

\*\*Ưu điểm:

* Dễ dàng cho việc test giao diện, người tester sẽ có phản hồi nhanh và trực quan về giao diện ứng dụng
* Mất ít chi phí cho các tool tự động và quy trình
* Khi có thay đổi nhỏ manual testing manual testing không bị mất nhiều thời gian để thay đổi các trường hợp kiểm thử

\*\*Nhược điểm:

* + - Kết quả kiểm thử ít tin cậy hơn vì có thể sai xót do yếu tố con người
    - Qúa trình thực hiện các ca kiểm thử không được ghi lại, do vậy nó không có tính tái sử dụng
    - Với một số task khó thực hiện thủ công như performance testing và stress testing thì manual testing rất khó để thực hiện
* Automation testing là phương pháp kiểm thử tự động. Người tester sẽ phải viết các kịch bản kiểm thử sau đó sử dụng các tool hỗ trợ để thực hiện kiểm thử, phương pháp này sẽ giúp việc kiểm thử hiệu quả và tốn ít thời gian hơn. Automation testing giúp chạy các kịch bản kiểm thử lặp lại nhiều lần và các task kiểm thử khác khó thực hiện bằng tay như performance testing và stress testing.

\*\* Ưu điểm:

* + - Sử dụng tool tự động giúp tìm kiếm được nhiều lỗi hơn
    - Automation testing nhanh và hiệu quả
    - Qúa trình kiểm thử được ghi lại, điều đó giúp chạy lại kịch bản kiểm thử nhiều lần và thực hiện trên nhiều nền tảng khác nhau
    - Automation testing được thực hiện bằng các công cụ phầm mềm, do đó nó hoạt động không mệt mỏi không giống như người kiểm thử tester
    - Automation testing năng suất và chính xác
    - Phạm vi kiểm thử rộng vì kiểm tra tự động không quên kiểm tra ngay cả đơn vị nhỏ nhất

\*\* Nhược điểm:

* + - Rất khó có cái nhìn đúng và trực quan về giao diện người dùng như màu sắc, font chữ, vị trí, kích thước các button nếu như không có yếu tố con người
    - Chi phí cho các tool kiểm thử có thể tốn kém, có thể làm tăng chi phí trong khâu kiểm thử của dự án
    - Nếu có một thay đổi nhỏ cũng sẽ mất thời gian để update kịch bản kiểm thử

**1.5Các loại kiểm thử phầm mềm:**

* + 1. **Kiểm thử đơn vị (Unit testing):**

Một đơn vị (Unit) là một thành phần phần mềm nhỏ nhất mà ta có thể  
kiểm thử được, ví dụ: các hàm (Function), thủ tục (Procedure), lớp (Class), hoặc các phương thức (Method).

Unit testing đề cập đến các kiểm thử để chứng thực (xác minh - verify)  
chức năng của một phần riêng biệt của code, thường ở mức hàm (function level). Trong một môi trường hướng đối tượng (object-oriented environment), kiểm thử đơn vị thường được sử dụng ở mức lớp (class) và kiểm thử các đơn vị nhỏ nhất bao gồm các hàm constructor và destructor.

Kiểm thử đơn vị thường do lập trình viên thực hiện. Công đoạn này  
cần được thực hiện càng sớm càng tốt trong giai đoạn viết code và xuyên suốt chu kỳ phát triển phần mềm. Một hàm có thể có nhiều kiểm thử, để bắt được các trường hợp hoặc các nhánh trong code.  
Cũng như các mức kiểm thử khác, kiểm thử đơn vị cũng đòi hỏi phải  
chuẩn bị trước các ca kiểm thử (hay trường hợp kiểm thử) (test case) hoặc kịch bản (test script), trong đó chỉ định rõ dữ liệu vào, các bước thực hiện và dữ liệu mong muốn sẽ xuất ra. Các test case và test script được giữ lại để sử dụng sau này.

* + 1. **Kiểm thử tích hợp ( Integration testing ):**

Integration test kết hợp các thành phần của một ứng dụng và kiểm thử như một ứng dụng đã hoàn thành. Trong khi Unit Test kiểm tra các thành phần và Unit riêng lẻ thì Intgration Test kết hợp chúng lại với nhau và kiểm tra sự giao tiếp giữa chúng.  
Integration Test có 2 mục tiêu chính:  
• 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).  
Có 4 loại kiểm thử trong Integration Test:  
• Kiểm thử cấu trúc (Structure Test): Tương tự White Box Test (kiểm thử nhằm bảo đảm các thành phần bên trong của một chương trình chạy đúng),  
chú trọng đến hoạt động của các thành phần cấu trúc nội tại của chương rình chẳng hạn các lệnh và nhánh bên trong.  
• Kiểm thử chức năng (Functional Test): Tương tự Black BoxTest (kiểm thử chỉ chú trọng đến chức năng của chương trình, không quan tâm đến cấu trúc bên trong), chỉ khảo sát chức năng của chương trình theo yêu cầu kỹ thuật.  
• Kiểm thử hiệu năng (Performance Test): kiểm thử việc vận hành của hệ  
thống.  
• Kiểm thử khả năng chịu tải (Stress Test): kiểm thử các giới hạn của hệ thống.

**1.5.3 Kiểm thử Hệ Thống( System Testing) :**

Kiểm thử hệ thống (ST) bao gồm một loạt những kiểm nghiệm nhằm xác minh toàn bộ các thành phần của hệ thống được tích hợp một cách đúng đắn. Mục đích của ST là đảm bảo toàn bộ hệ thống hoạt động như khách hàng mong muốn.  
System Test bắt đầu khi tất cả các bộ phận của PM đã được tích hợp thành công. Thông thường loại kiểm tra này tốn rất nhiều công sức và thời gian. Trong nhiều trường hợp, việc kiểm tra đòi hỏi một số thiết bị phụ trợ, phần mềm hoặc phần cứng đặc thù, đặc biệt là các ứng dụng thời gian thực,  
hệ thống phân bố, hoặc hệ thống nhúng.

Ở mức độ hệ thống, người kiểm tra cũng tìm kiếm các lỗi, nhưng trọng tâm là đánh giá về hoạt động, thao tác, sự tin cậy và các yêu cầu khác liên quan đến chất lượng của toàn hệ thống. Điểm khác nhau then chốt giữa Integration Test và System Test là System Test chú trọng các hành vi và lỗi trên toàn hệ thống, còn Integration Test chú trọng sự giao tiếp giữa các đơn thể hoặc đối tượng khi chúng làm việc cùng nhau.

Thông thường ta phải thực hiện Unit Test và Integration Test để bảo đảm mọi Unit và sự tương tác giữa chúng hoạt động chính xác trước khi thực hiện System Test.  
Kiểm thử hệ thống thường có các loại kiểm thử sau:  
 + Kiểm tra chức năng (Functional Test): bảo đảm các hành vi của hệ thống thỏa mãn đúng yêu cầu thiết kế.

+ Kiểm tra khả năng vận hành (Performance Test): bảo đảm tối ưu việc phân bổ tài nguyên hệ thống (ví dụ bộ nhớ) nhằm đạt các chỉ tiêu như thời gian xử lý hay đáp ứng câu truy vấn…

+ Kiểm tra khả năng chịu tải (Stress Test hay Load Test): bảo đảm hệ thống vận hành đúng dưới áp lực cao (ví dụ nhiều người truy xuất cùng lúc). Stress Test tập trung vào các trạng thái tới hạn, các “điểm chết”, các tình huống bất thường…  
 + Kiểm tra cấu hình (Configuration Test)

+ Kiểm tra khả năng bảo mật (Security Test): bảo đảm tính toàn vẹn, bảo mật của dữ liệu và của hệ thống.  
+ Kiểm tra khả năng phục hồi (Recovery Test): bảo đảm hệ thống có khả năng khôi phục trạng thái ổn định trước đó trong tình huống mất tài nguyên hoặc dữ liệu; đặc biệt quan trọng đối với các hệ thống giao dịch như ngân hàng trực tuyến

\*Và còn rất nhiều loại khác như :

* Kiểm thử cấu hình
* Kiểm thử sơ lược (smoke testing)
* Kiểm thử chức năng
* Kiểm thử hồi quy
* Kiểm thử tải dữ liệu (load testing)
* Kiểm thử tải trọng (stress testing)
* Kiểm thử hiệu suất (performance testing)
* Kiểm thử chấp nhận (UAT)
* Kiểm thử bảo mật (security testing

**2.Các Kỹ Thuật Kiểm Thử:**

Một sản phẩm đều có thể được kiểm thử theo 2 cách:

* Hiểu rõ một chức năng cụ thể của một hàm hay một module. Các trường hợp kiểm thử có thể xây dựng để kiểm thử tất cả các thao tác đó.
* Hiểu rõ cách hoạt động của một hàm/module hay sản phẩm. Các trường hợp kiểm thử có thể được xây dựng để đảm bảo tất cả các thành phần con khớp với nhau. Đó là tất cả các thao tác nội bộ của hàm dựa vào các mô tả và tất cả các thành phần nội bộ đã được kiểm thủ một cách thoả đáng.

Cách tiếp cận đầu tiên được gọi là kiểm thử hộp đen (Black-box testing) và cách tiếp cận thứ hai là gọi là kiểm thử hộp trắng (White-box testing).

**2.1 Kiểm Thử Hộp Đen( Black-box testing):**

Kiểm thử hộp đen hay còn gọi là kiểm tra chức năng thửnghiệm hành vi. Xem chương trình như là một “hộp đen”, hoàn toàn không quan tâm về cách cư xử và cấu trúc bên trong của chương trình. Thay vào đó, Tập trung vào tìm các trường hợp mà chương trình không thực hiện theo các đặc  
tả của nó.   
  
Phương pháp kiểm thử hộp đen bao gồm:

* Phân vùng 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
* Đồ thị nguyên nhân-kết quả - Cause Effect Graphing
* Kiểm thử dựa trên đặc tả – Specification-base testing

**2.1.1 Phân vùng tương đương - Equivalence partitioning** Phân vùng tương đương (Equivalence Class): là một kỹ thuật kiểm thử phần mềm có liên quan đến phân chia các giá trị đầu vào thành các phân vùng hợp lệ và không hợp lệ, sau đó chúng ta sẽ viết ra các kịch bản kiểm thử cho từng phần, chọn giá trị đại diện từ mỗi phân vùng làm dữ liệu thử  
nghiệm

Phân vùng tương đương: là kỹ thuật thực hiện test theo từng class đồng giá trị (tập hợp điều kiện cùng một thao tác).Tập hợp giá trị input có cùng một kết quả xử lý, tập hợp thời gian có cùng một kết quả xử lý, tập hợp kết quả export được xử lý cùng một giá trị nhập.

Mục đích : Giảm đáng kể số lượng test case cần phải thiết kế vì với mỗi lớp tương đương ta chỉ cần test trên các phần tử đại diện. Chọn tối thiểu một giá trị đại diện từ các class đồng giá trị để tiến hành test. Thiết kế ca kiểm thử cho phân vùng tương đương dựa trên sự đánh giá về các vùng tương đương với một điều kiện vào. Vùng tương đương biểu thị một tập cho các trạng thái hợp lệ hay không hợp lệ đối với điều kiện vào.

Thiết kế test case bằng phân vùng tương đương tiến hành theo hai bước:  
 - Xác định các lớp tương đương

-Xác định các ca kiểm thử

=> Nếu có lỗi xảy ra thì các giá trị khác trong class đồng giá trị cũng sẽ có lỗi giống nhau.

**2.1.2 Phân tích giá trị biên – Boundary Value Analysis** Phân tích giá trị biên (Boundary Value Analysis): là một kỹ thuật kiểm thử phần mềm có liên quan đến việc xác định biên (ranh giới) của điều kiện mô tả cho các giá trị đầu vào và chọn giá trị ở biên và bên cạnh giá trị biên làm dữ liệu kiểm thử. Phương pháp phân tích giá trị biên sẽ đưa ra các giá  
trị đặc biệt, bao gồm loại dữ liệu, giá trị lỗi, bên trong, bên ngoài biên giá trị, lớn nhất và nhỏ nhất.  
Test giá trị biên được thực hiện theo trình tự dưới đây:  
Tìm ra đường biên

* Quyết định giá trị biên
* Quyết định giá trị để test
* Giá trị biên.
* Dưới giá trị biên. (Nếu là class đồng giá trị)
* Trên 1 giá trị biên. (Nếu là class đồng giá trị)

**2.1.3 Sử dụng bảng quyết định – Decision Table**

* Làm giảm số lượng tets casse không cần thiết so với 2 kỹ thuật trên vì nó loại trừ các phép kết hợp không cần thiết giữa các giá trị biến đầu vào.
* Liệt kê nguyên nhân (cause) – kết quả (result) trong một ma trận. Mỗi cột ma trận đại diện cho 1 phép kết hợp giữa các cause trong trong việc tạo ra 1 result.
* Các bước để tạo bảng quyết định :
* Liệt kê các nguyên nhân trong bảng quyết định
* Tính tổng số lượng kết hợp giữa các cause
* Điền vào các cột với tất cả các kết hợp có thể có
* Rút bớt số lượng các phép kết hợp dư thừa
* Kiểm tra các phép kết hợp có bao phủ hết mọi trường hợp hay không
* Bổ sung kết quả vào bảng quyết định

**2.1.3 Đồ thị nguyên nhân - kết quả:**  
 Là một kỹ thuật thiết kế kiểm thử phần mềm liên quan đến việc xác định các trường hợp (điều kiện đầu vào) và các hiệu ứng (điều kiện đầu ra). Vì các hệ thống hiện nay đều được phát triển trên nền tảng OOP, do đó, chúng ta có thể có được một đồ thị các đối tượng mà hệ thống định nghĩa và kết  
nối. Từ đồ thị này, chúng ta dễ dàng biết các mối quan hệ của những đối tượng mà hệ thống xử lý, từ đó sẽ cho chúng ta các kịch bản kiểm thử phù hợp.

**2.2. Kiểm thử hộp trắng:**

Kiểm thử hộp trắng (White box testing) là một kỹ thuật xác minh giúp các kỹ sư phần mềm có thể sử dụng để kiểm tra mã code của họ hoạt động như dự kiến.  
Có các loại white box testing đang tồn tại như sau:

* API testing (application programming interface) – Kiểm thử ứng dụng  
  bằng cách sử dụng các hàm API public và private
* Code coverage – Là việc tạo các trường hợp test để thỏa mãn một số điều kiện bao phủ code - code coverage (ví dụ như, người thiết kế test có  
  thể tạo ra các trường hợp test sao cho tất cả các câu lệnh của chương trình đều được thực thi ít nhất 1 lần)
* Fault injection methods – cải tiến bao phủ một trường hợp bằng cách  
  đưa một số lỗi vào để test các đường dẫn code.
* Mutation testing methods
* Static testing - White box testing bao gồm tất cả các phương pháp kiểmthử tĩnh (ví dụ review code).

Kiểm thử độ bao phủ:

Phương pháp kiểm thử white box cũng có thể được sử dụng để ước lượng tính trọn vẹn đầy đủ của các tập hợp kiểm thử (test suit) đã được tạo ra bằng phương pháp kiểm thử black box. Điều này cho phép nhóm sản xuất phần mềm xem xét lại các phần của hệ thống ít được test nhất và để chắc chắn rằng các chức năng quan trọng nhất đã được tập trung test kỹ.  
  
Hai hình thức chung của kiểm thử độ bao phủ code:

* Bao phủ chức năng - Function coverage, dựa trên việc thực thi các chứcnăng.
* Bao phủ câu lệnh - Statement coverage, dựa trên số lượng các dòng lệnh đã được thực thi để hoàn thành kiểm thử

**II. SELENEUM**

**1. Giới thiệu chung về Selenium :**

Trong lĩnh vực kiểm thử tự động hiện có khá nhiều test tool thương mại nổi tiếng, phổ biến như Selenium, QuickTest Professional, WinRunner, Rational Robot, SilkTest, JTest,... Trong số đó, Selenium bao gồm nhiều chức năng điển hình của một công cụ kiểm thử tự động. Selenium (thường được viết tắt là SE) là một trong những công cụ kiểm thử phần mềm tự động mã nguồn mở (open source test automation tool) mạnh mẽ nhất hiện nay cho việc kiểm thử ứng dụng Web.  
Selenium là một công cụ kiểm thử phần mềm tự động, được phát triển bởi ThoughtWorks từ năm 2004 với tên ban đầu là JavaScriptTestRunner. Đến năm 2007, tác giả Jason Huggins rời ThoughtWorks và gia nhập Selenium team, một phần của Google và phát triển thành Selenium như hiện nay.  
Selenium là một tập hợp mạnh mẽ của các công cụ hỗ trợ phát triển  
nhanh chóng của các thử nghiệm tự động hóa cho các ứng dụng dựa trên web.Selenium cung cấp một tập phong phú của các thử nghiệm chức năng đặc biệt hướng đến các nhu cầu của các thử nghiệm của một ứng dụng web.  
Các hoạt động này là rất linh hoạt, cho phép nhiều tùy chọn cho vị trí các thành phần UI và so sánh kết quả thử nghiệm dự kiến sẽ chống lại hành vi ứng dụng thực tế.  
Lợi ích của Selenium :

* Hỗ trợ các trường hợp mà việc execute test lặp đi lặp lại  
  Hỗ trợ các trường hợp execute test một ma trận thử nghiệm lớn
* Có thể thực hiện execute test song song
* Có thể thực hiện execute test mà không cần người giám sát
* Cải thiện độ chính xác, giảm tối đa các lỗi do con người tạo ra
* Tiết kiệm thời gian và tiền bạc

Các đặc điểm của Selenium:  
 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.

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.

Selenium hỗ trợ nhiều ngôn ngữ lập trình C#, Java, Python, PHP vàSelenium còn có thể kết hợp với một số công cụ kiểm thử khác như Junit,Bromien, Nunit..

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.

Chạy test case ở backround. Khi chúng ta thực thi một test scrpit, chúng tahoàn 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.

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.

**2.Các thành phần của Selenium:**

Selenium gồm 4 thành phần chính, mỗi thành phần đều đóng một vai trò cụ thể trong việc hỗ trợ kiểm thử các ứng dụng Web. Các thành phần đó là:

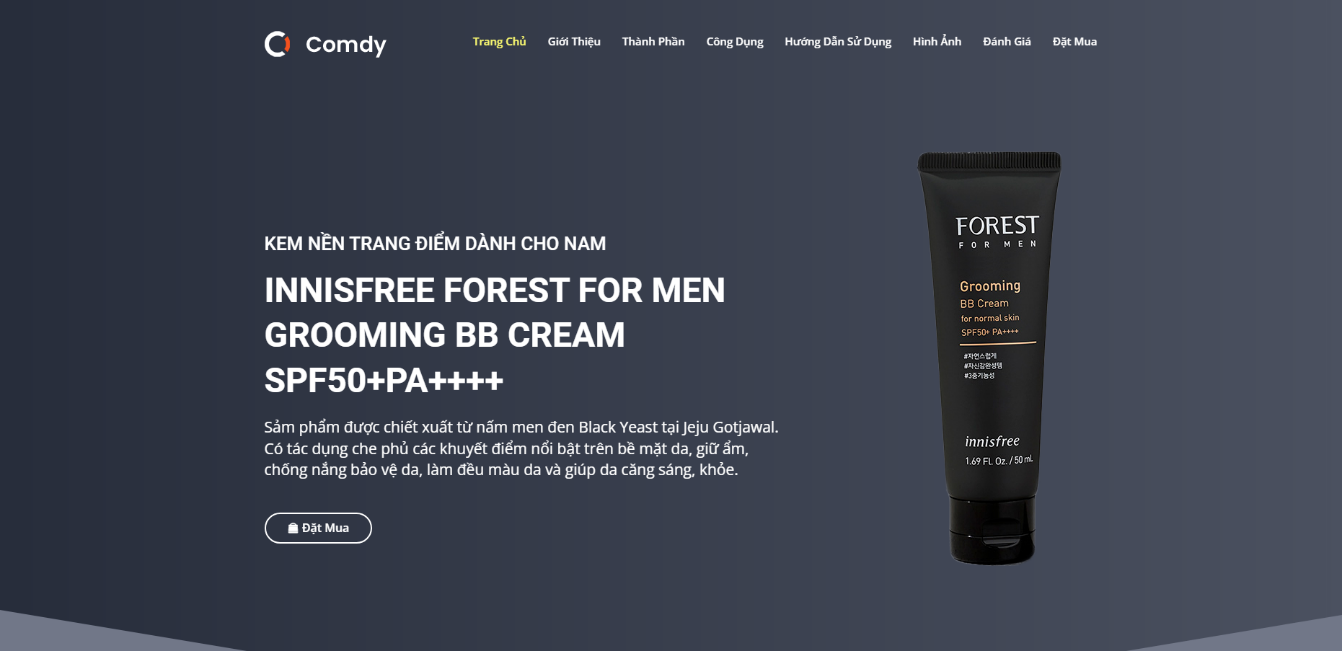
* Selenium IDE: là môi trường phát triển tích hợp cho việc xây dựng trường hợp kiểm thử Selenium. Nó hoạt động như một add-on của Firefox và cung cấp một giao diện dễ sử dụng để phát triển và chạy trường hợp kiểm thử. Selenium- IDE có tính năng thu lại kịch bản kiểm thử để tái sử dụng. Nó cũng có một menu ngữ cảnh tích hợp với trình duyệt Firefox, cho phép người dùng chọn từ một danh sách xác minh (verify) và khẳng định (assert) cho các yếu tố giao diện đã chọn. Selenium- IDE cũng cung cấp các chức năng chỉnh sửa các trường hợp kiểm thử chính xác và dễ kiểm soát hơn. Mặc dù Selenium-IDE chỉ là một Firefox add-on, nhưng các test case tạo ra bằng Selenium-IDE vẫn có thể chạy trên các trình duyệt khác bằng cách sử dụng Selenium- RC.
* Selenium Core: Công cụ này đã được tích hợp trong Selenium IDE. Selenium Core là một công cụ chạy các test script viết bằng Selenese. Thế mạnh của công cụ này là có thể chạy test script trên hần hết các trình duyệt, nhưng lại yêu cầu được cài đặt trên máy chủ của ứng dụng web cần kiểm tra. Điều này là không thể khi nhân viên kiểm thử không có quyền truy cập đến máy chủ
* Selenium RC (Remote Control): Selenium- RC 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. Ví dụ, nếu trình ứng dụng trả về một tập kết quả của việc kiểm thử, và nếu chương trình thử nghiệm tự động cần chạy thử nghiệm trên mỗi phần tử trong tập hợp kết quả, hỗ trợ lặp đi lặp lại các ngôn ngữ lập trình có thể được sử dụng để chuyển đổi thông qua việc tập hợp kết quả, kêu gọi lệnh Selenium chạy thử nghiệm trên mỗi mục. Selenium-RC cung cấp một API (Application Programming Interface) và thư viện cho mỗi ngôn ngữ được hỗ trợ: HTML, Java, C #, Perl, PHP, Python, và Ruby. Khả năng sử dụng Selenium- RC với một ngôn ngữ lập trình bậc cao để phát triển các trường hợp thử nghiệm cũng cho phép thử nghiệm tự động được tích hợp với một dự án xây dựng môi trường tự động
* 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 một lúc. Cũng cho phép lưu lại kết quả kiểm tra.

**III.Tiến hành kiểm thử:**

**1.bài toán kiểm thử:**

**-** Kiểm thử trang website bán mỹ phẩm

- website được kiểm thử trên 3 trình duyệt: chorm, mozillam, firefox

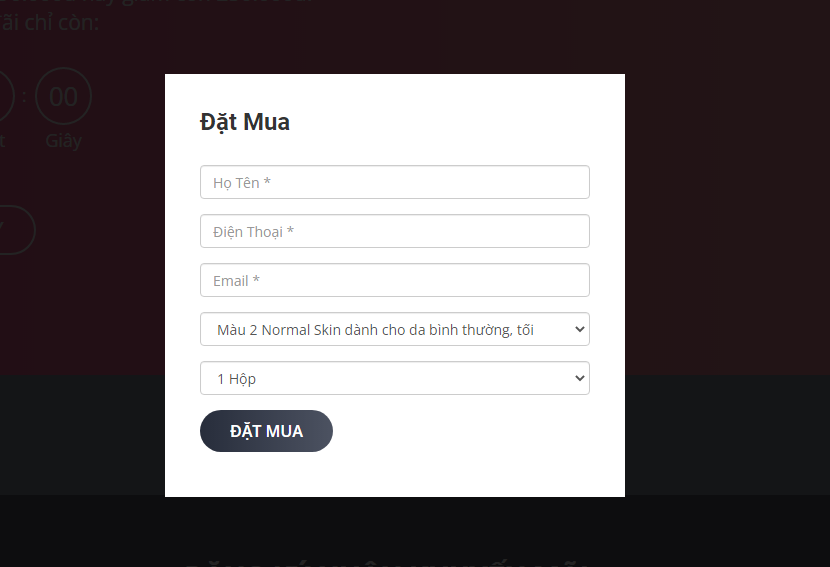
****

**2.Kịch bản kiểm thử (thủ công) :**

**-** Giao diện: Kiểm thử các yếu tố giao diện như: kiểm tra khi lướt chuột, kiểm tra hiệu ứng theo thiết kế , kiểm tra giao diện có bị vỡ khi reponsive,

- Chức năng:

- kiểm tra các nút button, các nút điều hướng trên thanh header

- Kiểm tra nút đặt mua 

- Kiểm tra đặt mua không thành công khi nhập số email không hợp lệ

- Kiểm tra đặt mua không thành công khi nhập thiếu tên/ email/ số điện thoại

- Kiểm tra đặt mua thành công khi nhập tên, email, số điện thoại hợp lệ

**IV. Kết Luận**

Qua thời gian học và làm việ thì em đã hoàn thành báo cáo về kiểm thử phần mềm, tuy nhiên còn nhiều hạn chế và thiếu sót trong kinh nghiệm cũng như kiến thức nên em sẽ tiếp tục nghiên cứu sâu hơn về các vấn đề của kiểm thử phần mềm và đặc biệt là bộ công cụ web selenium trong tương lai. Và em xin cảm ơn thầy đã hỗ trợ và giúp đỡ chúng em trong tương lai