

LỜI CẢM ƠN

Em xin chân thành cảm ơn các thầy giáo, cô giáo Khoa Công nghệ thông tin I của Học viện công nghệ bưu chính viễn thông Hà Nội và các thầy cô của viện CNTT&TT-CDIT đã tạo điều kiện thuận lợi cho em trong quá trình học tập 4 năm qua và trong quá trình thực hiện đồ án tốt nghiệp.

Em xin gửi lời cảm ơn đặc biệt đến thạc sĩ Đỗ Mạnh Hùng - Viện CNTT&TT-CDIT đã nhiệt tình hướng dẫn và chỉ bảo em trong suốt thời gian thực hiện đồ án.

Em xin cũng xin gửi lời cảm ơn chân thành đến gia đình, bạn bè và các anh chị đồng nghiệp tại trung tâm phần mềm viễn thông Viettel đã hết lòng hỗ trợ em trong thời gian thực hiện đồ án.

Hà Nội, ngày.....tháng.....năm 2012

Sinh viên:

Nguyễn Huyền Trang

(Của giảng viên hướng dẫn)

[illegible]

Đồng ý/Không đồng ý cho sinh viên bảo vệ trước hội đồng chấm đồ án tốt nghiệp?

CÁN BỘ - GIẢNG VIÊN HƯỚNG DẪN

(Của giảng viên phản biện)

[illegible]

Đồng ý/Không đồng ý cho sinh viên bảo vệ trước hội đồng chấm đồ án tốt nghiệp?

CÁN BỘ - GIẢNG VIÊN PHẢN BIỆN

MỤC LỤC

MỞ ĐẦU.....	- 1 -
1. Lý do chọn đề tài.....	- 1 -
2. Mục tiêu nghiên cứu	- 1 -
3. Bố cục nội dung của đồ án	- 1 -
CHƯƠNG 1: TỔNG QUAN VỀ PHẦN MỀM VÀ LỖI PHẦN MỀM	- 1 -
1.1. Định nghĩa phần mềm.....	- 1 -
1.2. Định nghĩa công nghệ phần mềm.....	- 1 -
1.3. Vòng đời phần mềm.....	- 1 -
1.4. Định nghĩa chất lượng phần mềm và đảm bảo chất lượng phần mềm.....	- 2 -
1.4.1. Định nghĩa chất lượng phần mềm.....	- 2 -
1.4.2. Định nghĩa đảm bảo chất lượng phần mềm	- 3 -
1.5. Lỗi phần mềm	- 3 -
1.5.1. Định nghĩa lỗi phần mềm và phân loại lỗi phần mềm	- 3 -
1.5.2. Các nguyên nhân gây lỗi phần mềm.....	- 3 -
1.5.3. Chi phí cho việc sửa lỗi phần mềm.....	- 4 -
1.6. Quy trình xử lý lỗi phần mềm	- 5 -
1.6.1. Bước 1: Đưa lỗi lên phần mềm quản lý lỗi.....	- 7 -
1.6.2. Bước 2: Gán lỗi cho nhân viên phát triển	- 7 -
1.6.3. Bước 3: Xử lý lỗi	- 8 -
1.6.4. Bước 4: Kiểm thử lại.....	- 8 -
1.7. Tổng kết chương 1	- 8 -
CHƯƠNG 2: TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM.....	- 9 -
2.1. Định nghĩa kiểm thử phần mềm.....	- 9 -
2.2. Mục tiêu của kiểm thử phần mềm.....	- 9 -
2.2.1. Mục tiêu trực tiếp	- 9 -
2.2.2. Mục tiêu gián tiếp	- 10 -
2.3. Các nguyên tắc cơ bản của kiểm thử phần mềm	- 10 -
2.4. Quy trình kiểm thử phần mềm.....	- 10 -
2.5. Các kỹ thuật kiểm thử phần mềm.....	- 11 -
2.5.1. Kiểm thử hộp đen.....	- 11 -
2.5.2. Kiểm thử hộp trắng	- 11 -
2.5.3. Kiểm thử hộp xám.....	- 12 -
2.6. Các giai đoạn kiểm thử phần mềm	- 12 -
2.6.1. Kiểm thử đơn vị	- 12 -
2.6.2. Kiểm thử tích hợp	- 13 -
2.6.3. Kiểm thử hệ thống.....	- 13 -
2.6.3.1. Kiểm thử chức năng	- 13 -
2.6.3.2. Kiểm thử hiệu năng.....	- 15 -
2.6.3.3. Kiểm thử an toàn thông tin	- 15 -
2.6.4. Kiểm thử chấp nhận	- 23 -
2.6.4.1. Kiểm thử alpha	- 23 -
2.6.4.2. Kiểm thử Beta.....	- 24 -

2.6.5. Kiểm thử hồi qui	- 24 -
2.7. Kiểm thử tự động	- 24 -
2.7.1. Kiểm thử tự động là gì? Qui trình kiểm thử tự động	- 24 -
2.7.2. Ưu điểm và nhược điểm của kiểm thử tự động.....	- 24 -
2.7.3. Các trường hợp nên áp dụng kiểm thử tự động	- 25 -
2.8. Tổng kết chương 2	- 26 -
CHƯƠNG 3: CÔNG CỤ KIỂM THỬ TỰ ĐỘNG SELENIUM.....	- 27 -
3.1. Tổng quan về Selenium	- 27 -
3.1.1. Selenium là gì?.....	- 27 -
3.1.2. Các thành phần của Selenium	- 27 -
3.2. Selenium IDE	- 28 -
3.2.1. Cài đặt Selenium IDE	- 28 -
3.2.2. Các icon của Selenium IDE	- 29 -
3.2.3. Các thao tác thực hiện kiểm thử tự động với Selenium.....	- 31 -
3.2.3.1. Recording_Thực hiện thu một kịch bản với Selenium IDE.....	- 31 -
3.2.3.2. Thêm các lệnh khẳng định và xác nhận với menu ngữ cảnh.....	- 32 -
3.2.3.3. Các thao tác chỉnh sửa.....	- 33 -
3.2.3.4. Mở và lưu lại một test case	- 33 -
3.2.3.5. Chạy các test case	- 33 -
3.2.4. Selenese.....	- 34 -
3.2.4.1. Cú pháp Script.....	- 34 -
3.2.4.2. Một số lệnh thường sử dụng trong Selenium.....	- 35 -
3.3. Selenium Remote Control (Selenium RC)	- 35 -
3.3.1. Các thành phần của Selenium Remote Control	- 36 -
3.3.1.1. Máy chủ Selenium.....	- 36 -
3.3.1.2. Các thư viện máy khách	- 36 -
3.3.2. Cài đặt Selenium Remote Control	- 36 -
3.3.2.1. Cài đặt máy chủ Selenium.....	- 37 -
3.3.2.2. Chạy Selenium Server	- 37 -
3.3.2.3. Sử dụng Java Client Driver.....	- 37 -
3.3.2.4. Sử dụng Python Client Driver.....	- 38 -
3.3.2.5. Sử dụng .NET Client Driver	- 38 -
3.3.2.6. Sử dụng Ruby Client Driver.....	- 38 -
3.3.3. Các thao tác với Selenium RC	- 38 -
3.3.3.1. Chạy các kịch bản kiểm thử Selenium IDE với Selenium Remote Control	- 38 -
3.3.3.2. Tạo một kịch bản kiểm thử mới với ngôn ngữ lập trình Java và Eclipse	- 40 -
3.3.3.3. Dịch một kịch bản kiểm thử Selenium IDE thành kịch bản kiểm thử Selenium RC.....	- 42 -
3.3.3.4. Báo cáo kết quả kiểm thử	- 45 -
3.4. Tổng kết chương 3	- 47 -
CHƯƠNG 4: THỬ NGHIỆM.....	- 49 -
4.1. Bài toán thử nghiệm	- 49 -
4.2. Sự khác nhau giữa kịch bản kiểm thử tự động và kịch bản kiểm thử thủ công ...	- 49 -
4.3. Kịch bản kiểm thử thủ công	- 50 -

4.3.1. Chức năng đăng nhập.....	- 50 -
4.2.2. Chức năng soạn thảo và gửi email	- 50 -
4.4. Kịch bản kiểm thử tự động	- 51 -
4.5. Kết quả thử nghiệm.....	- 57 -
4.5.1. Chức năng đăng nhập.....	- 57 -
4.5.2. Chức năng gửi email	- 58 -
4.6. Tổng kết chương 4	- 58 -
KẾT LUẬN.....	- 59 -
DANH MỤC TÀI LIỆU THAM KHẢO	- 60 -
PHỤ LỤC: KỊCH BẢN KIỂM THỬ THỦ CÔNG CHO ỨNG DỤNG THỬ NGHIỆM....	-61-

DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

Thuật ngữ/Từ viết tắt	Ý nghĩa	Ghi chú
IEEE	Institute of Electrical and Electronic Engineers	
Test case	Trường hợp kiểm thử	
Test suite	Tập hợp các trường hợp kiểm thử trong Selenium (Kịch bản kiểm thử tự động trong Selenium)	
Test script	Tập hợp các trường hợp kiểm thử (Kịch bản kiểm thử)	
Selenium RC	Selenium Remote Control	
Account	Tài khoản đăng nhập vào hệ thống	
User	Tên đăng nhập	
Mô-đun	Là một phần của chương trình, mỗi mô-đun đảm nhiệm một chức năng riêng	
Validate	Một thuật ngữ trong kiểm thử phần mềm dùng để chỉ sự kiểm tra tính hợp lệ của dữ liệu trên một yếu tố của ứng dụng	
Framework	Trong phần mềm, Framework là một tập các thư viện hoặc các lớp có thể sử dụng lại được	

DANH MỤC BẢNG, HÌNH VẼ, SƠ ĐỒ**Danh mục sơ đồ:**

Sơ đồ 1.1 : Chi phí cho việc sửa lỗi phần mềm	- 5 -
Sơ đồ 1.2: Các trạng thái của lỗi	- 5 -
Sơ đồ 1.3: Quy trình xử lý lỗi	- 6 -
Sơ đồ 2.1: Quy trình kiểm thử phần mềm	- 10 -
Sơ đồ 2.2: Các giai đoạn kiểm thử phần mềm	- 12 -

Danh mục hình vẽ:

Hình 2.1: Kiểm tra lỗi SQL Injection	- 17 -
Hình 2.2: Lỗi SQL Injection	- 17 -
Hình 2.3: Lỗi XSS_1	- 18 -
Hình 2.4: Kiểm tra Lỗi XSS_2	- 18 -
Hình 2.5: Lỗi XSS_2	- 18 -
Hình 2.6: Kiểm tra lỗ hổng CFRS_1	- 19 -
Hình 2.7: Kiểm tra lỗ hổng CFRS_2	- 19 -
Hình 2.8: Kiểm tra lỗ hổng CFRS_3	- 19 -
Hình 2.9: Kiểm tra lỗi Path Traversal_1	- 20 -
Hình 2.10: Kiểm tra lỗi Path Traversal_2	- 20 -
Hình 2.11: Kiểm tra lỗi Path Traversal_3	- 21 -
Hình 2.12: Kiểm tra lỗi Path Traversal_4	- 21 -
Hình 2.13: Kiểm tra lỗi xác thực phân quyền	- 21 -
Hình 2.14: Kiểm tra lỗi Session fixation	- 22 -
Hình 2.15: Lỗi HTTP Only Cookie	- 23 -
Hình 3.1: Pop up cài đặt Selenium	- 28 -
Hình 3.2: Kiểm tra cài đặt Selenium thành công	- 29 -
Hình 3.3: Các icon của Selenium IDE	- 29 -
Hình 3.4: Test case Selenium IDE	- 30 -
Hình 3.5: Thực hiện thu các trường hợp kiểm thử_1	- 31 -
Hình 3.6: Thực hiện thu các trường hợp kiểm thử_2	- 31 -
Hình 3.7: Lệnh xác minh (verify) một yếu tố trên trang web	- 32 -
Hình 3.8: Vai trò của Remote Control Server	- 35 -
Hình 3.9: Chạy máy chủ Selenium thành công	- 37 -
Hình 3.10: Câu lệnh chạy testcase Selenium IDE bằng Selenium RC	- 38 -
Hình 3.11: Chạy kịch bản kiểm thử Selenium IDE trên Selenium RC	- 39 -
Hình 3.12: Chạy kịch bản kiểm thử Selenium IDE trên Selenium RC	- 39 -
Hình 3.13: Tạo project Java	- 40 -

Hình 3.14 : Thêm các file jar vào thư viện	- 41 -
Hình 3.15: Mẫu test case Selenium RC	- 42 -
Hình 3.16: Export Test Case Selenium IDE sang Test Case Selenium Remote Control	- 43 -
Hình 3.17: Test Case Selenium Remote Control được export từ test case Selenium IDE	- 44 -
Hình 3.18 : Kết quả chạy test case trên Junit 4	- 45 -
Hình 3.19: Tạo file build.xml	- 45 -
Hình 3.20: Thêm file junit.jar vào Global Entries của Ant	- 46 -
Hình 3.21: Lựa chọn các mục tiêu để thực thi file build.xml	- 46 -
Hình 3.22: Mẫu báo cáo kết quả kiểm thử Selenium dựa trên JUnit	- 47 -
Hình 4.1: Test case đăng nhập bằng Firefox	- 51 -
Hình 4.2: Test case đăng nhập bằng Internet Explore	- 52 -
Hình 4.3: Test case đăng nhập bằng Googlechrome	- 53 -
Hình 4.4: File build.xml	- 56 -
Hình 4.5: Báo cáo kết quả kiểm thử	- 57 -

Danh mục bảng:

Bảng 2.1 : Kiểm thử giao diện người sử dụng	- 14 -
Bảng 2.2 : Kiểm thử luồng nghiệp vụ	- 14 -
Bảng 2.3 : Kiểm thử hiệu năng	- 15 -
Bảng 2.4: Kiểm thử an toàn thông tin	- 16 -
Bảng 2.5: Kiểm thử hồi qui	- 24 -
Bảng 3.1: Cú pháp câu lệnh Selenese	- 34 -

MỞ ĐẦU

1. Lý do chọn đề tài

Trong giai đoạn phát triển của công nghệ thông tin, ngành công nghệ phần mềm đang ngày một chiếm vị trí quan trọng trong xu hướng phát triển kinh tế công nghiệp hóa, hiện đại hóa của đất nước ta. Cùng với sự phát triển của công nghệ phần mềm, lỗi phần mềm và chất lượng phần mềm luôn là thách thức lớn với bản thân ngành phần mềm khi thực tế đã chứng minh, kiểm thử phần mềm là giai đoạn chiếm đến hơn 40% thời gian, kinh phí và nguồn nhân lực phát triển dự án phần mềm. 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 (số liệu thống kê năm 2011 của công ty LogiGear), 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 (thường dùng trong kiểm thử hiệu năng) 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 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.

Với mong muốn có cái nhìn xác thực, rõ ràng hơn về kiểm thử phần mềm và tiếp cận được với công cụ kiểm thử tự động Selenium để làm tiền đề cho định hướng tương lai khi tốt nghiệp đại học sẽ trở thành một kỹ sư kiểm thử phần mềm, cá nhân em lựa chọn đề tài “Nghiên cứu các vấn đề về kiểm thử phần mềm và công cụ kiểm thử tự động Selenium” làm đề tài cho đồ án tốt nghiệp đại học của mình. Trong khuôn khổ đồ án, do thời gian và kinh nghiệm thực tế còn hạn chế nên có những phần thực hiện chưa được tốt, em rất mong nhận được sự góp ý của thầy cô và các bạn.

2. Mục tiêu nghiên cứu

- Có cái nhìn đúng đắn và sâu sắc hơn về các vấn đề cơ bản của công nghệ phần mềm, lỗi phần mềm và kiểm thử phần mềm.
- Hiểu rõ về các thành phần của bộ công cụ Selenium.
- Nắm được cách thức sử dụng của hai công cụ là Selenium IDE và Selenium Remote Control.

- Ứng dụng các kiến thức kiểm thử phần mềm và kiến thức về Selenium để viết kịch bản kiểm thử cho một ứng dụng cụ thể.

3. **Bố cục nội dung của đồ án**

Đồ án được chia thành 6 chương với nội dung như sau:

- **Mở đầu:** Chương này trình bày về lý do chọn đề tài, mục tiêu nghiên cứu đồ án và bố cục nội dung của đồ án.
- **Chương 1: Tổng quan về phần mềm và lỗi phần mềm:** Chương này trình bày về những định nghĩa cơ bản về phần mềm, ngành công nghệ phần mềm, lỗi phần mềm, và qui trình xử lý lỗi phần mềm.
- **Chương 2: Tổng quan về kiểm thử phần mềm:** Chương này trình bày những kiến thức cơ bản về kiểm thử phần mềm như các nguyên tắc kiểm thử, các phương pháp kiểm thử, các giai đoạn kiểm thử phần mềm.
- **Chương 3: Công cụ kiểm thử tự động Selenium:** Chương này trình bày tổng quan về bộ công cụ Selenium, đi sâu vào các thao tác với Selenium IDE và Selenium RC.
- **Chương 4: Thử nghiệm:** Chương này trình bày kịch bản kiểm thử viết cho một số chức năng cơ bản của ứng dụng web <https://mail.viettel.com.vn> và thử nghiệm một số trường hợp kiểm thử tự động viết bằng Selenium IDE và Selenium RC.
- **Kết luận:** Chương này đưa ra những kết quả đồ án đạt được, những thiếu sót chưa thực hiện được và hướng phát triển đề tài trong tương lai.

Hà Nội, tháng 12/2012

Người thực hiện

Sinh viên: Nguyễn Huyền Trang

CHƯƠNG 1: TỔNG QUAN VỀ PHẦN MỀM VÀ LỖI PHẦN MỀM

1.1. Định nghĩa phần mềm

- **Định nghĩa theo IEEE:** Phần mềm là các chương trình máy tính, các thủ tục, các tài liệu có liên quan và các dữ liệu để vận hành của một hệ thống máy tính.
- Theo như định nghĩa của IEEE, phần mềm gồm 4 thành phần:
 - **Chương trình máy tính (mã nguồn):** Thành phần này giúp cho máy tính thực thi các ứng dụng được yêu cầu.
 - **Các thủ tục:** Các thủ tục được yêu cầu, để định nghĩa thứ tự và lịch trình mà chương trình sẽ thực thi, các hàm triển khai, người thực thi các hành động cần thiết cho ứng dụng phần mềm.
 - **Các tài liệu:** Có rất nhiều những tài liệu cần thiết với nhân viên phát triển, người sử dụng và nhân viên bảo trì như: tài liệu thiết kế, tài liệu phân tích, tài liệu hướng dẫn sử dụng, tài liệu hướng dẫn bảo trì.
 - **Dữ liệu cần cho việc vận hành hệ thống:** Dữ liệu bao gồm các biến, mã nguồn, danh sách tên thích ứng phần mềm với yêu cầu xác định của khách hàng để vận hành phần mềm.

1.2. Định nghĩa công nghệ phần mềm

Công nghệ phần mềm là việc áp dụng các công cụ, kỹ thuật một cách hệ thống trong việc phát triển các ứng dụng dựa trên máy tính. Nói cách khác đó là việc áp dụng các quan điểm, các tiến trình có kỷ luật và có bài bản vào để phát triển, vận hành và bảo trì phần mềm.

Về cơ bản, công nghệ phần mềm thực hiện các tác vụ chủ yếu sau: thiết kế phần mềm, xây dựng phần mềm, kiểm thử phần mềm và bảo trì phần mềm.

Mục tiêu của công nghệ phần mềm là tạo ra những phần mềm tốt, giảm đến mức tối thiểu sai sót xảy ra trong quá trình vận hành, cũng như tạo điều kiện thuận lợi trong việc bảo trì và nâng cấp phần mềm.

1.3. Vòng đời phần mềm

Vòng đời phần mềm là một loạt các pha mà phần mềm phải trải qua từ việc khám phá các khái niệm đến khi phần mềm bị loại bỏ hoàn toàn. Mô hình vòng đời phần mềm gồm 7 pha:

- **Pha yêu cầu (requirements phase):** Là pha đầu tiên trong quá trình xây dựng phần mềm, còn gọi là pha tìm hiểu khái niệm (concept exploration). Ở pha này, đại diện nhóm phát triển và khách hàng gặp nhau, khách hàng nêu ra những yêu cầu mà phần mềm phải có, đại diện nhóm phát triển ghi chép lại. Nếu dịch theo tiếng Anh "requirements phase" là pha yêu cầu. Tuy nhiên cách gọi này có thể hơi khó hiểu, do đó người ta thường gọi là pha xác định yêu cầu.
- **Pha đặc tả (specification phase):** Pha này phân tích các yêu cầu của khách hàng. Mô tả các kết quả phân tích dưới dạng "tài liệu đặc tả". Cuối giai đoạn này, kế hoạch quản lý dự án phần mềm được đưa ra, mô tả chi tiết quá trình phát triển phần mềm. Câu hỏi mà pha này cần trả lời là: "Phần mềm sẽ làm gì?".
- **Pha thiết kế (design phase):** Là pha tiếp theo pha đặc tả. Căn cứ vào tài liệu đặc tả, pha này mô tả cách thức mà phần mềm thực hiện các công việc cụ thể. Pha này trả lời câu hỏi "phần mềm sẽ được làm như thế nào?". Pha thiết kế thường có hai phần: thiết kế kiến trúc (architecture

design) và thiết kế chi tiết (detail design). Thiết kế kiến trúc phân chia phần mềm thành các thành phần gọi là mô-đun. Thiết kế chi tiết là thiết kế từng mô-đun một cách chi tiết.

- **Pha cài đặt** (implementation phase): Ở pha này, người ta thực hiện viết chương trình bằng một ngôn ngữ lập trình cụ thể.

- **Pha tích hợp** (integration phase): Kết nối các mô-đun đã viết của chương trình thành một phần mềm thống nhất và chạy thử, chỉnh sửa cho đến khi phần mềm chạy tốt.

- **Pha bảo trì** (maintenance phase): Khi chương trình đã được cài đặt trên máy của khách hàng vẫn có thể có lỗi phát sinh trong phần mềm cần loại trừ và điều chỉnh lại theo yêu cầu khách hàng. Công việc bảo trì được chia thành hai loại: bảo trì sửa lỗi (software repair) và bảo trì cập nhật (software update). Bảo trì sửa lỗi là sửa các lỗi có thể vẫn còn xuất hiện trong khi chạy chương trình. Bảo trì cập nhật lại được chia làm hai loại: bảo trì hoàn thiện (perfective maintenance) và bảo trì thích nghi (adaptive maintenance). Bảo trì hoàn thiện là sửa đổi phần mềm theo ý khách hàng để nâng cao hiệu quả. Bảo trì thích nghi là sửa đổi để phần mềm thích nghi với môi trường mới. Người ta tính toán rằng bảo trì sửa lỗi và thích nghi chiếm thời gian gần bằng nhau và bằng khoảng 20% thời gian bảo trì, còn bảo trì hoàn thiện chiếm thời gian khoảng gấp ba lần mỗi loại bảo trì kia (khoảng 60%).

- **Pha loại bỏ** (retirement phase): Đến một thời điểm nào đó, do sự thay đổi của môi trường làm việc và một số yếu tố khác, chi phí bảo trì phần mềm sẽ lớn hơn rất nhiều so với chi phí phát triển một phần mềm mới. Đó là lúc việc loại bỏ phần mềm được thực hiện.

1.4. Định nghĩa chất lượng phần mềm và đảm bảo chất lượng phần mềm

1.4.1. Định nghĩa chất lượng phần mềm

Có rất nhiều định nghĩa về chất lượng phần mềm được đưa ra bởi các tổ chức, cá nhân khác nhau. Trong phạm vi của đồ án này trình bày một số định nghĩa tiêu biểu.

- **Định nghĩa theo IEEE (1991):**

- **Định nghĩa 1:** Chất lượng phần mềm là một mức độ mà một hệ thống, thành phần hệ thống hay tiến trình đáp ứng được yêu cầu đã được đặc tả.

- **Định nghĩa 2:** Chất lượng phần mềm là mức độ mà một hệ thống, thành phần hệ thống hay tiến trình đáp ứng được yêu cầu và sự mong đợi của khách hàng hay người sử dụng.

- **Phân tích hai quan điểm của IEEE:**

- Theo quan điểm thứ nhất của IEEE: Nếu theo quan điểm này, chúng ta sẽ bị phụ thuộc quá nhiều vào tài liệu đặc tả yêu cầu, dẫn đến nếu việc xác định yêu cầu bị sai, thiếu thì một phần mềm được làm đúng với đặc tả chưa chắc đã là một phần mềm có chất lượng.

- Theo quan điểm thứ hai của IEEE: Khách hàng đôi khi không có nhiều kiến thức về công nghệ, họ có thể đưa ra những mong muốn hết sức vô lý và có thể thay đổi yêu cầu với phần mềm nhiều lần thậm chí thay đổi ngay trong giai đoạn cuối. Điều này gây rất nhiều khó khăn cho việc phát triển phần mềm.

- **Định nghĩa theo Pressman:** Chất lượng phần mềm là sự phù hợp của các yêu cầu cụ thể về hiệu năng và chức năng, các tiêu chuẩn phát triển phần mềm được ghi lại rõ ràng bằng tài liệu với các đặc tính ngầm định của tất cả các phần mềm được phát triển chuyên nghiệp.

- Định nghĩa của Pressman đề xuất ba yêu cầu với chất lượng phần mềm phải được đáp ứng khi phát triển phần mềm:

- Các yêu cầu chức năng rõ ràng là nhân tố chính quyết định chất lượng đầu ra của phần mềm.
- Các tiêu chuẩn chất lượng phần mềm sẽ được nói đến trong hợp đồng.

- Các đặc tính ngầm định cần được đáp ứng trong quá trình phát triển cho dù không được nói đến rõ ràng trong hợp đồng.

1.4.2. Định nghĩa đảm bảo chất lượng phần mềm

- **Định nghĩa theo Daniel Galin:** Đảm bảo chất lượng phần mềm (Software Quality Assure) là một tập hợp các hành động cần thiết được lên kế hoạch một cách hệ thống để cung cấp đầy đủ niềm tin rằng quá trình phát triển phần mềm phù hợp để thành lập các yêu cầu chức năng kỹ thuật cũng như các yêu cầu quản lý theo lịch trình và hoạt động trong giới hạn ngân sách.

1.5. Lỗi phần mềm

1.5.1. Định nghĩa lỗi phần mềm và phân loại lỗi phần mềm

- Định nghĩa lỗi phần mềm: Có rất nhiều định nghĩa về lỗi phần mềm nhưng có thể hiểu và phát biểu một cách đơn giản thì "Lỗi phần mềm là sự không khớp giữa chương trình và đặc tả của nó".

- Dựa vào định nghĩa, ta có thể phân loại lỗi phần mềm thành 3 dạng:
 - Lỗi sai: Sản phẩm phần mềm được xây dựng khác với đặc tả.
 - Lỗi thiếu: Các yêu cầu của sản phẩm phần mềm đã có trong đặc tả nhưng lại không có trong sản phẩm thực tế.
 - Lỗi thừa: Sản phẩm thực tế có những tính năng không có trong tài liệu đặc tả.

1.5.2. Các nguyên nhân gây lỗi phần mềm

Lỗi phần mềm có thể đến từ nhiều nguyên nhân khác nhau, trong đó có cả các nguyên nhân chủ quan và các nguyên nhân khách quan. Dưới đây là chín nguyên nhân chủ yếu gây ra lỗi phần mềm:

- **Định nghĩa các yêu cầu bị lỗi:** Những lỗi trong việc xác định yêu cầu thường nằm ở phía khách hàng. Một số lỗi thường gặp là: định nghĩa sai yêu cầu, lỗi không hoàn chỉnh, thiếu các yêu cầu quan trọng hoặc là quá chú trọng các yêu cầu không thật sự cần thiết.

- **Các lỗi trong giao tiếp giữa khách hàng và nhà phát triển:** Hiểu lầm trong giao tiếp giữa khách hàng và nhà phát triển cũng là nguyên nhân gây lỗi. Những lỗi này thường xuất hiện trong giai đoạn đầu của dự án. Một số lỗi hay gặp phải: hiểu sai chỉ dẫn trong tài liệu yêu cầu, hiểu sai thay đổi khi khách hàng trình bày bằng lời nói và tài liệu, hiểu sai về phản hồi và thiếu quan tâm đến những đề cập của khách hàng.

Giải pháp khắc phục: Cần có ủy ban liên kết giữa khách hàng và nhà cung cấp để tránh lỗi trong giao tiếp. Ủy ban do quản trị dự án đứng đầu và khách hàng phải giới thiệu những người hiểu về mặt nghiệp vụ vào ủy ban đó.

- **Sai lệch có chủ ý với các yêu cầu phần mềm:** Trong một số trường hợp các nhà phát triển cố tình làm sai lệch các yêu cầu trong tài liệu đặc tả. Nguyên nhân của việc này đến từ các áp lực thời gian, ngân sách, hay cố tình sử dụng lại các mô-đun từ các dự án trước mà chưa phân tích đầy đủ những thay đổi để thích nghi với các yêu cầu mới.

Giải pháp khắc phục: Dựa trên những thông kê để quyết định xem giải pháp như thế nào, sắp xếp ưu tiên xem bỏ được yêu cầu nào hay sử dụng lại được mô-đun nào.

- **Các lỗi thiết kế logic:** Lỗi phần mềm xảy ra trong quá trình các chuyên gia thiết kế hệ thống, các kiến trúc sư hệ thống, kỹ sư phần mềm, các nhà phân tích xây dựng phần mềm theo yêu cầu. Các lỗi điển hình bao gồm:

- Định nghĩa các yêu cầu phần mềm bằng các thuật toán sai
- Quy trình định nghĩa có chứa trình tự lỗi
- Sai sót trong các định nghĩa biên như > 3 hay ≥ 3
- Thiếu sót các trạng thái hệ thống phần mềm được yêu cầu

- **Các lỗi lập trình:** Có rất nhiều lý do dẫn đến việc các lập trình viên gây ra các lỗi lập trình. Những lý do này bao gồm: sự hiểu sai các tài liệu thiết kế, ngôn ngữ; sai sót trong ngôn ngữ lập trình; sai sót trong việc áp dụng các công cụ phát triển; sai sót trong lựa chọn dữ liệu...

- **Không tuân thủ theo các tài liệu hướng dẫn và tiêu chuẩn lập trình:** Các lỗi phần mềm có thể đến từ việc không tuân thủ các tài liệu và tiêu chuẩn lập trình của các tổ chức phát triển phần mềm.

- **Thiếu sót trong quá trình kiểm thử:** Lỗi phần mềm có thể đến từ chính quá trình kiểm thử khi mà người kiểm thử để lọt lỗi.

Những lỗi này đến từ các nguyên nhân sau đây:

- Kế hoạch kiểm thử chưa hoàn chỉnh, để sót yêu cầu cần kiểm thử.
- Lỗi trong tài liệu và báo cáo kiểm thử.
- Việc sửa chữa các lỗi được phát hiện không hoàn chỉnh do áp lực thời gian hay do thiếu cẩn thận.

Giải pháp: Lên kế hoạch kiểm thử cụ thể tại giai đoạn đầu của dự án.

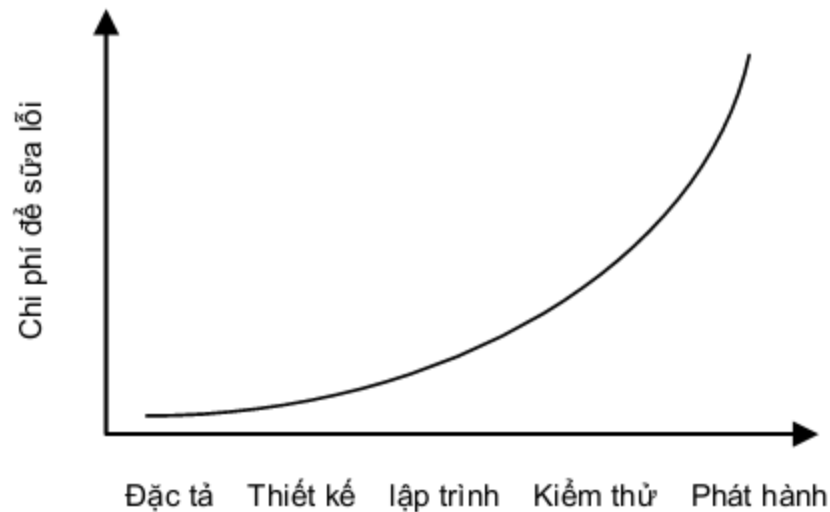
- **Các lỗi thủ tục:** Các thủ tục hướng dẫn cho người sử dụng tại từng bước của tiến trình. Chúng có tầm quan trọng đặc biệt trong các hệ thống phần mềm phức tạp mà các tiến trình được thực bằng nhiều bước, mỗi bước có thể có nhiều kiểu dữ liệu và cho phép kiểm tra các kết quả trung gian. Các lỗi có thể đến từ việc viết các thủ tục.

- **Các lỗi về tài liệu:** Các lỗi về tài liệu là vấn đề của các đội phát triển và bảo trì khi có những sai sót trong các tài liệu liên quan. Những lỗi này có thể là nguyên nhân gây ra lỗi trong giai đoạn phát triển kế tiếp và giai đoạn bảo trì.

1.5.3. Chi phí cho việc sửa lỗi phần mềm

Việc kiểm thử và sửa lỗi phần mềm có thể thực hiện trong bất cứ giai đoạn nào của vòng đời phần mềm. Tuy nhiên công việc này nên được thực hiện càng sớm càng tốt vì càng về giai đoạn sau của vòng đời phần mềm, chi phí cho việc tìm và sửa lỗi càng tăng, đặc biệt là đến giai đoạn đã triển khai phần mềm thì chi phí cho sửa lỗi sẽ trở nên rất lớn và ảnh hưởng trực tiếp đến uy tín của tổ chức phát triển phần mềm.

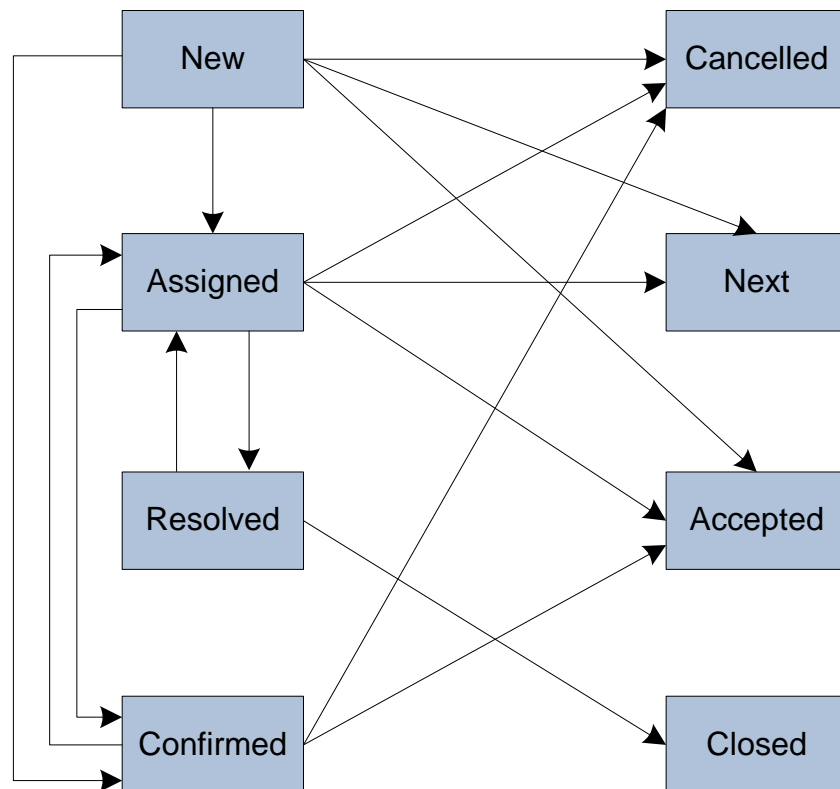
Theo tài liệu của Boehm, chi phí cho việc tìm và sửa lỗi phần mềm sẽ tăng theo hàm mũ trong biểu đồ sau:



Sơ đồ 1.1 : Chi phí cho việc sửa lỗi phần mềm

1.6. Quy trình xử lý lỗi phần mềm

Trước khi giới thiệu về quy trình xử lý lỗi phần mềm, đồ án sẽ trình bày về các trạng thái có thể có của lỗi.



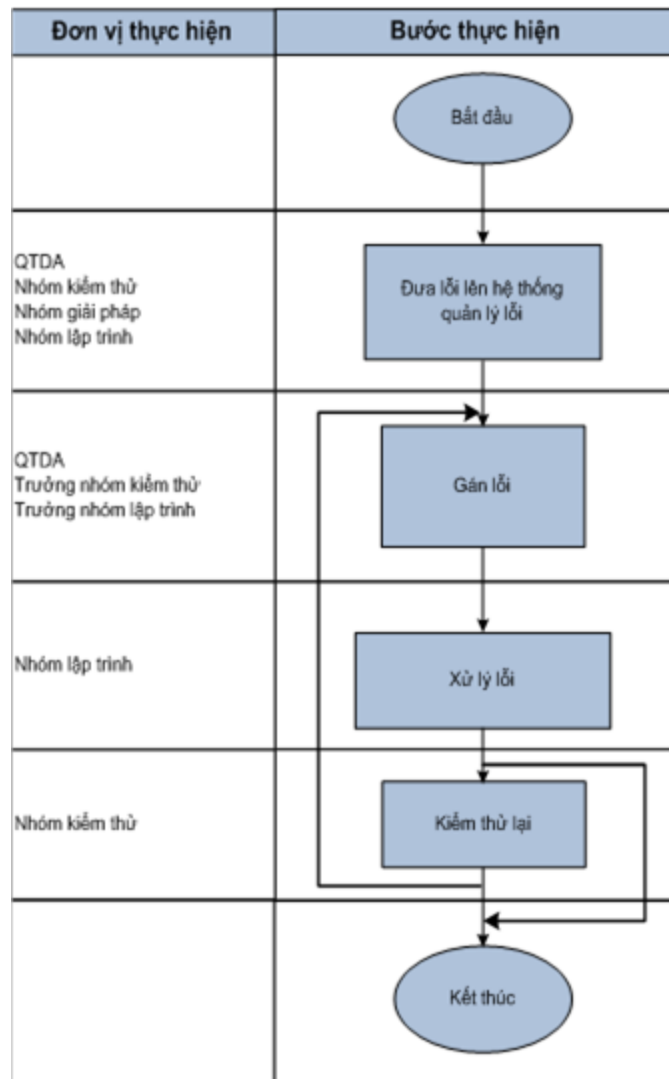
Sơ đồ 1.2: Các trạng thái của lỗi

Trong đó:

- New: Lỗi mới
- Assigned: Lỗi đã được gán cho nhân viên phát triển
- Resolved: Lỗi đã được xử lý
- Confirmed: Lỗi cần được chứng thực, thảo luận lại

- Canceled: Lỗi được xác định là không phải lỗi, lỗi được bỏ qua
- Next: Lỗi không thuộc phạm vi của dự án, hoặc sẽ được xử lý trong một giai đoạn khác của dự án
- Accept: Các lỗi có thể chấp nhận được. Ví dụ: Các lỗi do Framework
- Closed: Trạng thái đóng. Lỗi đã được sửa thành công.

Mỗi tổ chức phát triển phần mềm sẽ sử dụng một công cụ quản lý lỗi riêng, trong đó có thể kể đến Mantis là một công cụ được sử dụng khá phổ biến hiện nay. Phần tiếp theo sẽ trình bày một qui trình xử lý lỗi phần mềm hiện đang được sử dụng trong thực tế ở một số tổ chức phát triển và gia công phần mềm:



Sơ đồ 1.3: Qui trình xử lý lỗi

Theo đó, qui trình xử lý lỗi có thể bao gồm 6 bước chính:

- Bước 0_Bắt đầu: Phát hiện phần mềm có lỗi
- Bước 1_Đưa lỗi lên hệ thống quản lý lỗi
- Bước 2_Gán lỗi cho nhân viên phát triển
- Bước 3_Xử lý lỗi
- Bước 4_Kiểm thử lại
- Bước 5_Đóng lỗi

1.6.1. Bước 1: Đưa lỗi lên phần mềm quản lý lỗi

- Người thực hiện: tất cả các thành viên trong đội dự án như quản trị dự án, nhóm kiểm thử, nhóm giải pháp, nhóm lập trình.
- Trạng thái của lỗi là NEW.
- **Một số thông tin cần có về lỗi:**
 - Category: Thư mục lỗi dùng để phân loại lỗi, lỗi thuộc phần chức năng nào phải chọn đúng phần thư mục lỗi tương ứng để thuận tiện cho việc tra cứu, thống kê lỗi của chức năng.
 - Severity (trọng số của lỗi): Thông số này biểu hiện độ nghiêm trọng của lỗi, thông thường lỗi sẽ thuộc một trong ba trọng số dưới đây:
 - Minor: Các lỗi định dạng (font chữ, cỡ chữ, màu sắc của các đối tượng, chiều dài của các đối tượng), lỗi chính tả, lỗi validate dữ liệu.
 - Major: Các lỗi ràng buộc dữ liệu, lỗi chức năng nghiệp vụ của hệ thống (nhưng chưa gây ra treo hệ thống hay không làm cho hệ thống không xử lý được tiếp).
 - Crash: Các lỗi chức năng nghiệp vụ của hệ thống gây treo hệ thống, không xử lý được tiếp.
 - Reproducibility: Khả năng tái tạo lỗi. Khi phát hiện ra lỗi, nhân viên kiểm thử cần thực hiện lại phần chức năng phát hiện ra lỗi để xét khả năng tái tạo lỗi và lựa chọn đúng khả năng tái tạo lỗi.
 - Priority: Mức độ ưu tiên trong việc sửa lỗi.
 - Summary: Tóm tắt nội dung lỗi, có thể coi là tiêu đề của lỗi.
 - Description: Đây là phần mô tả lỗi, phải mô tả rõ 3 phần nội dung:
 - Các bước thực hiện
 - Kết quả trả về từ hệ thống
 - Kết quả mong muốn
 - Notes: Dùng để đưa các lưu ý, trao đổi về lỗi của các thành viên trong dự án.

1.6.2. Bước 2: Gán lỗi cho nhân viên phát triển

- Nhân viên kiểm thử thực hiện gán lỗi cho nhân viên phát triển, người sẽ chịu trách nhiệm về phần chức năng bị lỗi.
- Trạng thái của lỗi là ASSIGNED.
- Lưu ý:
 - Trưởng nhóm kiểm thử, quản trị dự án có thể xem xét lại các lỗi có trạng thái NEW và ASSIGNED trên hệ thống phần mềm quản lý lỗi:
 - Nếu thấy không phải là lỗi thì đưa lỗi về trạng thái CANCELLED và nêu rõ nguyên nhân đưa lỗi về CANCELLED.
 - Nếu thấy nhân viên kiểm thử mô tả không rõ ràng, thiếu thông tin thì chuyển lỗi sang trạng thái CONFIRMED và yêu cầu nhân viên kiểm thử bổ sung thêm thông tin.

- Nếu thấy lỗi không thuộc phạm vi phát triển của dự án trong giai đoạn hiện tại thì chuyển lỗi về trạng thái NEXT.
- Quản trị dự án xem xét lại các lỗi có trạng thái NEW hoặc ASSIGNED, nếu thấy là lỗi nhưng có thể chấp nhận được thì chuyển lỗi sang trạng thái ACCEPTED và nêu rõ nguyên nhân đưa lỗi về trạng thái ACCEPTED.

1.6.3. Bước 3: Xử lý lỗi

Nhân viên phát triển xem xét các lỗi được gán cho mình:

- Nếu thấy đúng là lỗi và đã mô tả lỗi rõ ràng, đầy đủ thông tin, nhân viên phát triển thực hiện sửa lỗi và chuyển lỗi về trạng thái RESOLVED, đồng thời bắt buộc nêu rõ hướng giải quyết và các chức năng bị ảnh hưởng trong phần NOTES.
- Các trường hợp khác: Nếu thấy không phải là lỗi của hệ thống, nhân viên phát triển sẽ yêu cầu nhân viên kiểm thử bổ sung thêm thông tin, hoặc thấy là lỗi nhưng có thể chấp nhận được thì nhân viên phát triển chuyển lỗi sang trạng thái CONFIRMED và nêu rõ lý do chuyển lỗi sang trạng thái CONFIRMED trong phần NOTES.

1.6.4. Bước 4: Kiểm thử lại

Theo phân công của trưởng nhóm kiểm thử, nhân viên kiểm thử thực hiện kiểm thử lại các chức năng có lỗi đang ở trạng thái RESOLVED:

- Nếu lỗi đã được sửa thì chuyển lỗi về trạng thái CLOSED.
- Nếu lỗi chưa được sửa hoặc mới chỉ sửa một phần thì chuyển lỗi về trạng thái ASSIGNED và nêu rõ những phần chức năng nào chưa chỉnh sửa để nhân viên phát triển tiến hành sửa tiếp.
- Nếu thấy có thể chấp nhận lỗi thì chuyển lỗi về trạng thái ACCEPTED. Đồng thời khi cập nhật kịch bản kiểm thử thì sẽ để kết quả của case đó là fail (vì vẫn là lỗi).
- Lưu ý: Nếu phần chức năng bị ảnh hưởng gây ra lỗi mới thì đưa một lỗi mới lên phần mềm quản lý lỗi.

1.7. Tổng kết chương 1

Chương 1 của đồ án đã trình bày được những định nghĩa và những vấn đề cơ bản xung quanh phần mềm, công nghệ phần mềm, lỗi phần mềm và xử lý lỗi phần mềm. Các vấn đề cụ thể bao gồm:

- Các định nghĩa về phần mềm, công nghệ phần mềm, chất lượng phần mềm, bảo đảm chất lượng phần mềm, lỗi phần mềm.
- Các vấn đề liên quan đến lỗi phần mềm và quy trình xử lý lỗi phần mềm.

CHƯƠNG 2: TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM

2.1. Định nghĩa kiểm thử phần mềm

Kiểm thử phần mềm có nhiều định nghĩa khác nhau đề xuất bởi nhiều tổ chức hay cá nhân khác nhau. Phần này của đồ án sẽ trình bày một số định nghĩa nổi bật:

- **Định nghĩa của Myer (1979):** "Kiểm thử là quá trình thực thi một chương trình với mục đích tìm ra lỗi". Theo như định nghĩa này, quá trình kiểm thử bao gồm tất cả các hoạt động từ kiểm tra mã nguồn được thực hiện bởi trưởng nhóm phát triển, đến việc chạy thử chương trình được tiến hành bởi các đồng nghiệp khác. Tất cả các hoạt động trên đều được coi là các hoạt động kiểm thử.
- **Hai định nghĩa của IEEE (1990):**
 - **Định nghĩa 1:** Kiểm thử phần mềm là quá trình vận hành một hệ thống hoặc một thành phần của hệ thống với các điều kiện xác định, nhận xét và ghi lại các kết quả, tạo ra đánh giá về những khía cạnh của hệ thống hay thành phần hệ thống.
 - **Định nghĩa 2:** Kiểm thử phần mềm là quá trình phân tích các yếu tố phần mềm để phát hiện những khác biệt giữa chương trình với các điều kiện yêu cầu và đánh giá các đặc điểm của các yếu tố phần mềm.

Theo như định nghĩa 2, việc chạy chương trình như một phần của tiến trình kiểm thử phần mềm là không cần thiết.

- **Định nghĩa của Daniel Galin:** "Kiểm thử phần mềm là một quá trình được tiến hành bởi một nhóm chuyên viên kiểm thử, trong đó một đơn vị phần mềm, một nhóm các đơn vị được tích hợp, hoặc cả gói phần mềm được kiểm tra bằng cách chạy các chương trình trên máy tính. Tất cả các bước kiểm tra liên được tiến hành theo các thủ tục kiểm thử và các trường hợp kiểm thử đã được thông qua".

Định nghĩa của Daniel Galin là một định nghĩa khá hoàn thiện về kiểm thử phần mềm. Một số thuật ngữ có trong định nghĩa của Daniel Galin:

- **Nhóm chuyên viên kiểm thử:** Một nhóm độc lập hoặc nhóm tư vấn từ bên ngoài, những người chuyên kiểm thử được chỉ định để thực hiện các nhiệm vụ chủ yếu là để phát hiện và loại bỏ sai lệch và để đảm bảo kiểm thử hiệu quả bởi các chuyên gia kiểm thử được đào tạo.
- **Các thủ tục kiểm thử đã được thông qua:** Quá trình kiểm thử được thực hiện theo kế hoạch kiểm thử và các thủ tục kiểm thử được thông qua phù hợp với các thủ tục đảm bảo chất lượng phần mềm được thông qua bởi tổ chức phát triển phần mềm.
- **Các trường hợp kiểm thử được thông qua:** Các trường hợp kiểm thử được định nghĩa đầy đủ trong kế hoạch kiểm thử. Không có sự thiếu sót hoặc bỏ sung nào được mong đợi xảy ra trong suốt quá trình thực thi kiểm thử.

2.2. Mục tiêu của kiểm thử phần mềm

2.2.1. Mục tiêu trực tiếp

- Phát hiện và xác định càng nhiều lỗi càng tốt ở các phần mềm được kiểm thử.
- Tiến hành sửa lỗi ở các phần mềm được kiểm thử và kiểm thử lại cho đến khi đạt một mức độ chất lượng phần mềm chấp nhận được.
- Thực thi những trường hợp kiểm thử một cách hiệu quả trong một giới hạn ngân sách và lịch trình cho phép.

2.2.2. Mục tiêu gián tiếp

- Để biên dịch một tài liệu về các lỗi phần mềm thường gặp nhằm mục đích ngăn ngừa và sửa chữa lỗi.

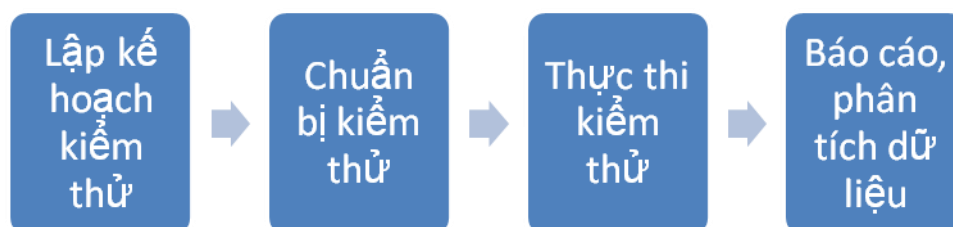
2.3. Các nguyên tắc cơ bản của kiểm thử phần mềm

Có bảy nguyên tắc cơ bản cần chú ý khi kiểm thử phần mềm, các nguyên tắc đó là:

- **Kiểm thử để chứng minh sự có mặt của lỗi và không chứng minh điều ngược lại:** Kiểm thử có thể cho thấy sự có mặt của lỗi nhưng không thể chứng minh điều ngược lại là chương trình không có lỗi. Việc kiểm thử giảm nguy cơ không tìm thấy lỗi trong phần mềm nhưng ngay cả khi không tìm thấy lỗi thì cũng không thể chứng minh được sản phẩm phần mềm được phát triển hoàn toàn chính xác.
- **Không thể kiểm thử vét cạn:** Việc kiểm thử không thể thực hiện được cho tất cả mọi trường hợp kiểm thử. Do vậy thay vì kiểm thử mọi khía cạnh, ta phải tập trung vào kiểm thử những yếu tố quan trọng và nhiều rủi ro.
- **Kiểm thử sớm:** Các hoạt động kiểm thử nên bắt đầu càng sớm càng tốt trong vòng đời phát triển phần mềm, và nên tập trung vào những mục tiêu kiểm thử nhất định.
- **Phân cụm lỗi:** Một số lượng nhỏ các mô-đun phần mềm có thể chứa hầu hết các lỗi được phát hiện ra trong suốt quá trình kiểm thử hoặc tập trung hầu hết các lỗi vận hành.
- **Kiểm thử ngược:** Nếu một phương pháp kiểm thử được lặp đi lặp lại nhiều lần, các trường hợp kiểm thử giống nhau sẽ không phát hiện được triệt để lỗi mới. Để khắc phục điều này ta có thể sử dụng nguyên tắc "kiểm thử ngược", các trường hợp kiểm thử cần phải được xem xét và duyệt lại một cách đều đặn, và việc kiểm thử mới cần phải được viết lại để thực thi những phần khác của phần mềm hay hệ thống để tìm ra những lỗi tiềm ẩn.
- **Kiểm thử phụ thuộc vào ngữ cảnh:** Việc kiểm thử được thực hiện trong những hoàn cảnh khác nhau thì khác nhau.
- **Sai lầm về việc không có lỗi:** Tìm kiếm và sửa lỗi không thể giúp được gì nếu hệ thống không dùng được hoặc không đáp ứng được yêu cầu và sự mong đợi của khách hàng.

2.4. Quy trình kiểm thử phần mềm

Tùy vào từng tổ chức, hệ thống, ngữ cảnh, mức độ rủi ro của phần mềm mà quy trình kiểm thử phần mềm có thể gồm nhiều bước khác nhau. Nhưng nhìn chung mọi quy trình kiểm thử đều có những bước cơ bản như quy trình dưới đây:



Sơ đồ 2.1: Quy trình kiểm thử phần mềm

Theo đó một quy trình kiểm thử phần mềm cơ bản gồm 4 giai đoạn:

- **Lập kế hoạch kiểm thử:** Nhiệm vụ quan trọng trong phần lập kế hoạch kiểm thử là xác định được các yếu tố sau:

- Các giai đoạn kiểm thử áp dụng cho dự án
- Các phương pháp kiểm thử
- Các công cụ kiểm thử
- Nguồn lực kiểm thử
- Tài nguyên môi trường kiểm thử, bao gồm các tài nguyên phần cứng và phần mềm
- Mốc bàn giao các tài liệu kiểm thử
- **Chuẩn bị kiểm thử:** Nhiệm vụ chiến lược của giai đoạn này là:
 - Tìm hiểu nghiệp vụ của hệ thống phải kiểm thử
 - Xây dựng kịch bản kiểm thử, phát triển các thủ tục và các kịch bản kiểm thử tự động (trong trường hợp kiểm thử tự động)
 - Chuẩn bị dữ liệu kiểm thử
 - Xem xét phê duyệt các tài liệu kiểm thử
- **Thực thi kiểm thử:**
 - Thực hiện kiểm thử dựa trên các kịch bản kiểm thử, test script, thủ tục, dữ liệu có sẵn từ bước chuẩn bị kiểm thử
 - Tham gia quá trình quản lý lỗi: báo lỗi, sửa lỗi
- **Báo cáo và phân tích dữ liệu kiểm thử:**
 - Báo cáo kiểm thử
 - Phân tích nguyên nhân và đề xuất các hành động khắc phục

2.5. Các kỹ thuật kiểm thử phần mềm

Có 3 kỹ thuật kiểm thử phần mềm chính là:

- Kiểm thử hộp đen
- Kiểm thử hộp trắng
- Kiểm thử hộp xám

2.5.1. Kiểm thử hộp đen

- Kỹ thuật kiểm thử hộp đen xem chương trình như là một “hộp đen”, trong đó người kiểm thử không quan tâm đến cấu trúc bên trong của chương trình mà chỉ quan tâm tới dữ liệu đầu vào và đầu ra sau khi được xử lý.

- Mục đích của chiến lược này là tìm kiếm các trường hợp mà chương trình không thực hiện theo các đặc tả của nó.

- Ưu, nhược điểm: Kiểm thử hộp đen có ưu điểm là có thể đánh giá phần mềm một cách khách quan, người kiểm thử có thể không hiểu biết về mã lệnh và có thể tìm ra các lỗi mà nhân viên phát triển không tìm ra. Song kiểm thử hộp đen lại có nhược điểm là thăm dò mù, do nhân viên kiểm thử không biết các chương trình thực sự được xây dựng như thế nào, dẫn đến trường hợp nếu kiểm thử hộp đen phải viết rất nhiều trường hợp kiểm thử trong khi chỉ cần viết một ca kiểm thử duy nhất để có thể kiểm tra được.

2.5.2. Kiểm thử hộp trắng

- Kỹ thuật kiểm thử hộp trắng hay còn gọi là “kiểm thử cấu trúc” là kỹ thuật kiểm thử cho phép khảo sát kiến trúc bên trong của chương trình. Kiểm thử hộp trắng là chiến lược được thực hiện trên ba trong sáu loại kiểm thử cơ bản trong các giai đoạn kiểm thử phần mềm là: kiểm thử đơn vị, kiểm thử tích hợp và kiểm thử hồi quy. Mục tiêu của kiểm thử hộp trắng

là kiểm thử bao phủ nhiều nhất các câu lệnh, điểm quyết định và các rẽ nhánh trong mã nguồn nếu có thể.

2.5.3. Kiểm thử hộp xám

Kiểm thử hộp xám là kỹ thuật kiểm thử có sự kết hợp giữa kiểm thử hộp đen và kiểm thử hộp trắng. Trong đó ta cũng quan tâm đến dữ liệu đầu vào và đầu ra giống như trong kiểm thử hộp đen, song lại đòi hỏi có sự truy cập đến cấu trúc dữ liệu và giải thuật để thiết kế các trường hợp kiểm thử.

2.6. Các giai đoạn kiểm thử phần mềm

- Kiểm thử phần mềm gồm 4 giai đoạn chính:

- Kiểm thử đơn vị
- Kiểm thử tích hợp
- Kiểm thử hệ thống
- Kiểm thử nghiệm thu



Sơ đồ 2.2: Các giai đoạn kiểm thử phần mềm

2.6.1. Kiểm thử đơn vị

- Đơn vị: Là thành phần nhỏ nhất của phần mềm có thể kiểm thử được. Ví dụ: Các hàm, lớp, thủ tục, phương thức. Đơn vị thường có kích thước nhỏ, chức năng hoạt động đơn giản, không gây nhiều khó khăn trong việc kiểm thử, ghi nhận và phân tích kết quả do đó nếu phát hiện lỗi việc tìm kiếm nguyên nhân và sửa lỗi cũng đơn giản và tốn ít chi phí hơn. Một nguyên lý đúc kết từ thực tiễn là thời gian dành cho kiểm thử đơn vị sẽ được đền bù bằng việc tiết kiệm được khá nhiều thời gian và chi phí cho việc kiểm thử và sửa lỗi ở các mức độ kiểm thử sau đó.
- Mục đích: Đảm bảo thông tin được xử lý đúng và có đầu ra chính xác trong mối tương quan giữa dữ liệu nhập và chức năng của đơn vị.
- Người thực hiện: Do việc kiểm thử đơn vị đòi hỏi phải kiểm tra từng nhánh lệnh, nên đòi hỏi người kiểm thử có kiến thức về lập trình cũng như về thiết kế của hệ thống nên người thực hiện thường là lập trình viên.

2.6.2. Kiểm thử tích hợp

- Kiểm thử tích hợp là kiểm thử sự kết hợp và giao tiếp giữa các đơn vị của một chương trình và kiểm thử như một chương trình đã hoàn thành.
- Mục đích:
 - Phát hiện lỗi giao tiếp xảy ra giữa các đơn vị cũng như lỗi của bản thân từng đơn vị (nếu có).
 - Tích hợp các đơn vị đơn lẻ thành các hệ thống nhỏ (subsystem) và cuối cùng là tích hợp các hệ thống nhỏ thành một hệ thống hoàn chỉnh (system) để chuẩn bị cho kiểm thử hệ thống.
- Người thực hiện: Thường là lập trình viên.
- Lưu ý:
 - Kiểm thử tích hợp chỉ nên thực hiện trên từng đơn vị đã được kiểm tra cẩn thận trước đó bằng kiểm thử đơn vị, và tất cả các lỗi mức đơn vị đã được sửa chữa.
 - Nên tích hợp dần từng đơn vị: Một đơn vị nên được tích hợp vào một nhóm các đơn vị khác đã được tích hợp và hoàn thành kiểm thử tích hợp trước đó vì khi đó chỉ cần kiểm tra giao tiếp giữa đơn vị mới được thêm vào với nhóm các đơn vị đã được tích hợp trước đó.

2.6.3. Kiểm thử hệ thống

- Kiểm thử hệ thống bắt đầu khi tất cả các đơn vị của hệ thống được tích hợp thành công. Đây là công đoạn kiểm thử tốn nhiều công sức và thời gian hơn cả. Và đặc biệt, công đoạn này thường đòi hỏi được thực hiện bởi một nhóm nhân viên tách biệt với nhóm phát triển, có chuyên môn và kinh nghiệm kiểm thử.
- Kiểm thử hệ thống gồm nhiều loại kiểm thử khác nhau, trong số đó, các mục tiêu kiểm thử quan trọng nhất là:
 - Kiểm thử chức năng
 - Kiểm thử hiệu năng
 - Kiểm thử an toàn thông tin
- Mục đích: kiểm tra xem hệ thống được làm ra có thỏa mãn yêu cầu hay không về nhiều khía cạnh: hoạt động, độ tin cậy, hiệu năng của hệ thống.
- Người thực hiện: Nhóm nhân viên kiểm thử.
- Lưu ý:
 - Việc lập kế hoạch cho kiểm thử hệ thống nên bắt đầu từ giai đoạn bắt đầu dự án.

Phần tiếp theo sẽ đi sâu vào phân tích các bước kiểm thử quan trọng nhất, được coi là không thể bỏ qua khi tiến hành kiểm thử bất cứ hệ thống nào.

2.6.3.1. Kiểm thử chức năng

Việc kiểm thử chức năng chú trọng đến 2 phần chính là kiểm thử giao diện người dùng (User interface) và kiểm thử luồng nghiệp vụ (Business Flow Testing).

a) Kiểm thử giao diện người sử dụng

Kiểm thử giao diện người sử dụng gọi tắt kiểm thử giao diện là việc kiểm tra các tương tác của người dùng với phần mềm. Mục tiêu của kiểm thử giao diện là để đảm bảo rằng giao diện người dùng cung cấp cho người sử dụng cách truy cập và sử dụng các chức năng

của hệ thống một cách thích hợp. Ngoài ra, kiểm thử giao diện còn để đảm bảo rằng các đối tượng trên giao diện giống như thiết kế và phù hợp với tổ chức hoặc chuyên ngành.

Mục đích kiểm thử:	Kiểm tra: <ul style="list-style-type: none"> - Việc sử dụng thông qua mục tiêu kiểm thử phản ánh đúng các chức năng và yêu cầu nghiệp vụ, bao gồm màn hình đến màn hình, trường đến trường và sử dụng các phương pháp truy cập (phím tabs, di chuột, tổ hợp phím) - Các đối tượng và thuộc tính màn hình như menus, size, position, state.
Cách thực hiện:	<ul style="list-style-type: none"> - Tạo ra và chỉnh sửa kịch bản kiểm thử cho mỗi màn hình để kiểm tra việc sử dụng đúng cách và tình trạng các đối tượng cho mỗi màn hình và đối tượng của ứng dụng
Điều kiện hoàn thành:	<ul style="list-style-type: none"> - Mỗi màn hình được kiểm tra thành công đúng với phiên bản kiểm tra hoặc phạm vi chấp nhận được
Các vấn đề đặc biệt:	<ul style="list-style-type: none"> - Không phải toàn bộ các thuộc tính của các đối tượng đều truy cập được

Bảng 2.1 : Kiểm thử giao diện người sử dụng

b) Kiểm thử luồng nghiệp vụ

Mục đích của kiểm thử luồng nghiệp vụ là kiểm tra các yêu cầu chức năng và nghiệp vụ của hệ thống bao gồm các hoạt động để kiểm tra tính đúng đắn của dữ liệu, qui trình, báo cáo và việc thực hiện đúng những qui tắc nghiệp vụ. Kiểu kiểm thử này dựa vào kỹ thuật kiểm thử hộp đen, tức là kiểm tra ứng dụng và các xử lý bên trong ứng dụng bằng cách tương tác với ứng dụng thông qua giao diện người sử dụng và phân tích các kết quả hoặc đầu ra. Bảng sau liệt kê một số gợi ý đối với mỗi ứng dụng:

Mục đích kiểm thử:	Đảm bảo mục tiêu kiểm thử đúng đắn của chức năng, bao gồm dữ liệu đầu vào, xử lý dữ liệu và dữ liệu nhận được. Kiểm thử chức năng đảm bảo các yêu cầu sau: <ul style="list-style-type: none"> - Nhập dữ liệu hợp lệ thì chương trình phải cho nhập - Luồng nghiệp vụ đúng - Quá trình xử lý dữ liệu và kết quả đầu ra phải đúng - Phục hồi được dữ liệu
Cách thực hiện:	Thực hiện các chức năng, sử dụng các dữ liệu hợp lệ và không hợp lệ để kiểm tra. Cụ thể như sau: <ul style="list-style-type: none"> - Kết quả mong đợi với dữ liệu hợp lệ. - Lỗi thích hợp hoặc thông báo hiển thị khi dữ liệu không hợp lệ. - Mỗi qui tắc nghiệp vụ đều được áp dụng đúng
Điều kiện hoàn thành:	<ul style="list-style-type: none"> - Toàn bộ kế hoạch kiểm thử đã được thực hiện. - Toàn bộ các lỗi phát hiện ra đã được ghi nhận.
Các vấn đề đặc biệt:	<ul style="list-style-type: none"> - Xác định hoặc mô tả các vấn đề (nội bộ hoặc bên ngoài) ảnh hưởng đến việc kiểm thử chức năng

Bảng 2.2 : Kiểm thử luồng nghiệp vụ

2.6.3.2. Kiểm thử hiệu năng

Mục đích của kiểm thử hiệu năng là kiểm tra các yêu cầu về hiệu năng có đạt được hay không.

Mục đích kiểm thử:	Kiểm tra các biểu hiện về hiệu năng cho các giao dịch hoặc chức năng nghiệp vụ theo những điều kiện sau: <ul style="list-style-type: none"> - Khối lượng công việc bình thường đã biết trước - Khối lượng công việc xấu đã biết trước
Cách thực hiện:	<ul style="list-style-type: none"> - Sử dụng các thủ tục cho kiểm thử luồng nghiệp vụ - Chỉnh sửa file dữ liệu để tăng số lượng các giao dịch hoặc scripts để tăng số tương tác xảy ra trong mỗi giao dịch - Scripts phải được chạy trên một máy (trường hợp tốt nhất để đánh giá người dùng đơn lẻ, giao dịch đơn lẻ) và phải lặp lại trên nhiều máy trạm.
Điều kiện hoàn thành:	<ul style="list-style-type: none"> - Giao dịch đơn lẻ hoặc người dùng đơn lẻ: Thực hiện thành công test script không có lỗi và trong phạm vi mong đợi hoặc thời gian phản hồi cho mỗi giao dịch - Nhiều giao dịch hoặc nhiều người dùng: Thực hiện thành công test script không có lỗi và trong thời gian chấp nhận được
Các vấn đề đặc biệt:	

Bảng 2.3: Kiểm thử hiệu năng

2.6.3.3. Kiểm thử an toàn thông tin

Kiểm thử an toàn thông tin tập trung vào hai lĩnh vực bảo mật chính:

- Bảo mật ở mức ứng dụng: bao gồm truy cập dữ liệu và các chức năng nghiệp vụ.
- Bảo mật ở mức hệ thống: bao gồm truy cập vào hệ thống hoặc truy cập từ xa.

Bảo mật mức ứng dụng đảm bảo rằng, dựa trên bảo mật đã yêu cầu, người dùng bị hạn chế sử dụng một số chức năng hoặc tình huống sử dụng, hoặc bị hạn chế trong giới hạn dữ liệu phù hợp với họ. Ví dụ, người dùng có thể được phép nhập dữ liệu để tạo account nhưng chỉ có người quản lý có thể xóa chúng. Nếu là bảo mật ở mức dữ liệu, việc kiểm thử đảm bảo rằng “người dùng nhóm 1” có thể nhìn thấy các thông tin khách hàng, bao gồm dữ liệu tài chính, tuy nhiên “người dùng nhóm 2” chỉ nhìn thấy các thông tin chung chung cho cùng một khách hàng.

Bảo mật mức hệ thống đảm bảo rằng chỉ những người dùng được cho quyền truy cập vào hệ thống mới có khả năng truy cập vào ứng dụng và chỉ bằng các cổng thích hợp.

Mục đích kiểm thử:	<ul style="list-style-type: none"> - Bảo mật mức ứng dụng: Đảm bảo rằng một người dùng chỉ có thể truy cập vào những chức năng hoặc dữ liệu mà nhóm người dùng đó được phép. - Bảo mật mức hệ thống: Đảm bảo rằng chỉ những người được phép truy cập hệ thống và ứng dụng được phép truy cập chúng.
Cách thực hiện:	<ul style="list-style-type: none"> - Bảo mật ứng dụng: Xác định và liệt kê từng nhóm người dùng và các chức năng hoặc dữ liệu mà họ được phép truy cập - Tạo kịch bản kiểm thử cho mỗi nhóm người dùng và kiểm tra từng quyền bằng cách tạo các giao dịch xác định cho mỗi nhóm - Sửa lại nhóm người dùng và chạy lại tình huống kiểm thử cho cùng những người dùng. Với mỗi trường hợp, kiểm tra các chức năng thêm vào hoặc dữ liệu có đúng không hay bị từ chối. - Truy cập mức hệ thống: tham khảo các điều kiện đặc biệt dưới đây
Điều kiện hoàn thành:	<ul style="list-style-type: none"> - Với mỗi nhóm người dùng đều có các chức năng hoặc dữ liệu thích hợp, và toàn bộ các chức năng giao dịch đều như dự kiến và chạy trong các kiểm thử chức năng ứng dụng trước đó
Các vấn đề đặc biệt:	<ul style="list-style-type: none"> - Truy cập vào hệ thống phải được xem xét hoặc thảo luận với quản trị hệ thống hoặc quản trị mạng, có thể không cần nếu nó là chức năng của quản trị mạng hoặc quản trị hệ thống

Bảng 2.4: Kiểm thử an toàn thông tin

Đồ án sẽ trình bày một số lỗi an toàn thông tin quan trọng và thường gặp trong quá trình kiểm thử và cách kiểm tra các lỗi an toàn thông tin đó:

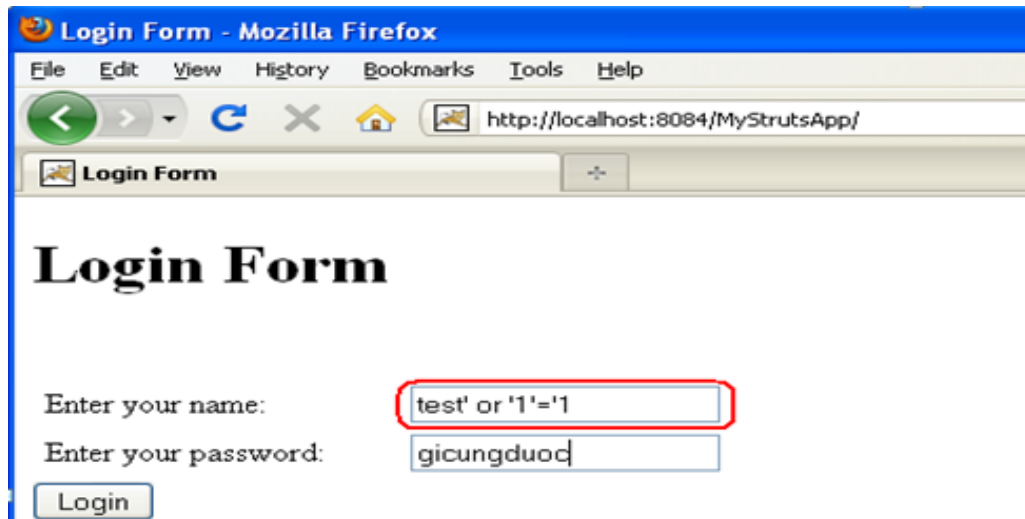
- Kiểm tra lỗi SQL Injection

- Mô tả: Lỗi SQL Injection xảy ra khi các biến do người dùng truyền lên (GET, POST) được đưa thẳng vào các câu truy vấn cơ sở dữ liệu mà không qua xử lý, khi đó kẻ tấn công có thể truyền các kí tự đặc biệt mang ngữ nghĩa SQL truyền vào câu truy vấn để thực hiện các thao tác độc hại như thêm, xóa hay sửa các dữ liệu trên cơ sở dữ liệu và thực hiện các biện pháp tấn công leo thang khác.

- Hướng dẫn kiểm tra lỗi: Các lỗi SQL Injection thường xuất hiện tại các chức năng của ứng dụng có tương tác với cơ sở dữ liệu, trong đó có một số biến được truyền vào ứng dụng từ trình duyệt. Các biến GET thường tồn tại dưới dạng các ký tự mang ngữ nghĩa SQL, để kiểm tra lỗi SQL Injection, ta thử bằng cách:

○ Truyền các ký tự đặc biệt mang ngữ nghĩa SQL như ' vào các biến dạng số trên URL hoặc chuỗi tổ hợp các ký tự đặc biệt vào các form. Nếu ứng dụng xuất lỗi 500, hoặc trên trình duyệt in ra câu truy vấn SQL lỗi hay đăng nhập thành công, khi đó có thể xác định ứng dụng đã bị mắc lỗi SQL Injection.

- Ví dụ: Truyền vào form đăng nhập chuỗi ký tự đặc biệt: test' or '1'='1



Hình 2.1: Kiểm tra lỗi SQL Injection

Nếu hệ thống cho phép đăng nhập thành công thì hệ thống đã bị lỗi SQL Injection:



Hình 2.2: Lỗi SQL Injection

- Kiểm tra lỗi XSS

- Mô tả:

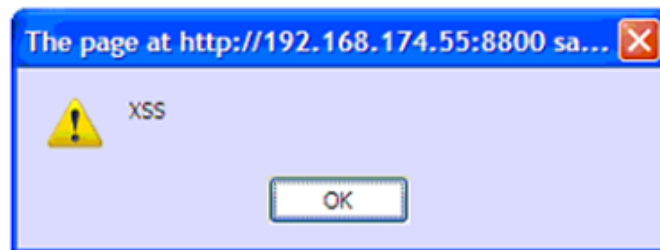
- Lỗi XSS trên ứng dụng là lỗi mà kẻ tấn công có thể truyền các biến độc hại vào ứng dụng để tấn công người dùng.

- Lỗi hỏng XSS thường xuất hiện ở các chức năng cho phép người dùng nhập dữ liệu vào hệ thống, sau đó các dữ liệu này được lưu vào cơ sở dữ liệu hoặc đưa ra hiển thị trực tiếp mà không qua xử lý, khi đó kẻ tấn công có thể truyền vào các kí tự HTML hoặc Javascript. Khi người dùng khác truy cập vào các trang có hiển thị các dữ liệu này (có thể do chủ động hoặc bị dẫn dụ) thì các script sẽ được thực thi, kẻ tấn công có thể chiếm đoạt phiên của người dùng hợp lệ.

- Cách kiểm tra: Nhập vào chuỗi kí tự `<script>alert("XSS")</script>` vào các form hay trên URL, nếu ứng dụng lưu lại và cho phép thực thi script thì trên trình duyệt sẽ xuất hiện cửa sổ có dòng chữ "XSS", khi đó ứng dụng bị mắc lỗi XSS.

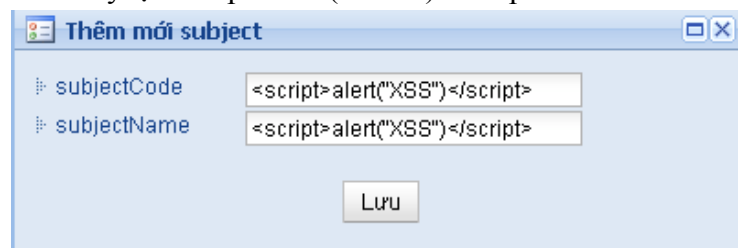
- Ví dụ 1: Truyền ký chuỗi ký tự `<script>alert("XSS")</script>` vào biến user của trang [http://192.168.174.96:9999/index.php?user=<script>alert\("XSS"\)</script>](http://192.168.174.96:9999/index.php?user=<script>alert()

Ứng dụng sẽ trả về lỗi như hình dưới, như vậy ứng dụng đã bị lỗi XSS



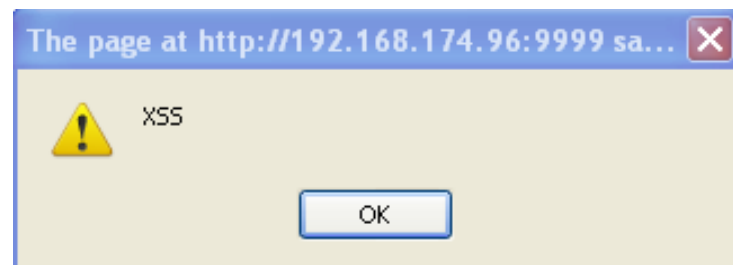
Hình 2.3: Lỗi XSS_1

- Ví dụ 2: Nhập chuỗi ký tự `<script>alert("XSS")</script>` vào form:



Hình 2.4: Kiểm tra Lỗi XSS_2

Ứng dụng trả về như hình:



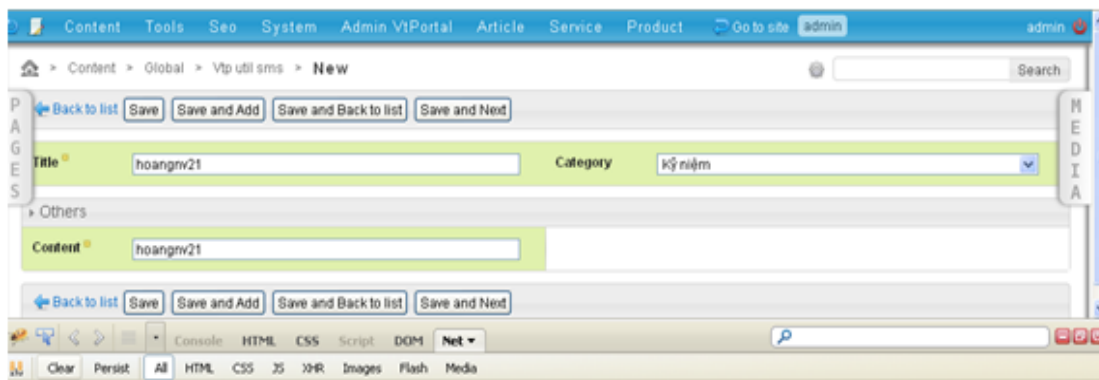
Hình 2.5: Lỗi XSS_2

- Kiểm tra lỗi hỏng CSRF (Cross-site request forgery)

- Mô tả: Khi gặp lỗi này kẻ tấn công có thể thực hiện mượn tay của người có quyền trên hệ thống thực thi tác vụ mà kẻ tấn công mong muốn mà không người thực hiện không hề

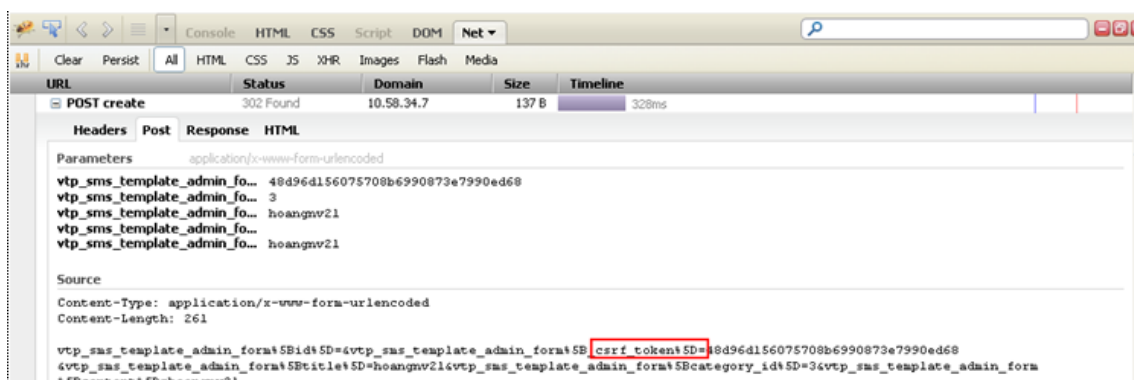
biết.

- Hướng dẫn kiểm tra lỗi: Để kiểm tra lỗi này ta thực hiện như sau:
 - Sử dụng một user
 - Trên trình duyệt, mở hai TAB để đăng nhập vào ứng dụng
 - Đăng nhập theo thứ tự: đăng nhập vào ứng dụng trên TAB1 rồi TAB2
 - Quay về TAB1 thực hiện các tác vụ cần thiết như thêm, xóa, sửa.
 - Sang TAB2 mô phỏng tác vụ giống hệt bên TAB1 nếu vẫn thực hiện được thì ứng dụng bị lỗi CSRF.
 - Ngoài ra, sử dụng thêm FireBug (Add-ons của Firefox) kiểm tra ứng dụng có dùng biến token hay không.

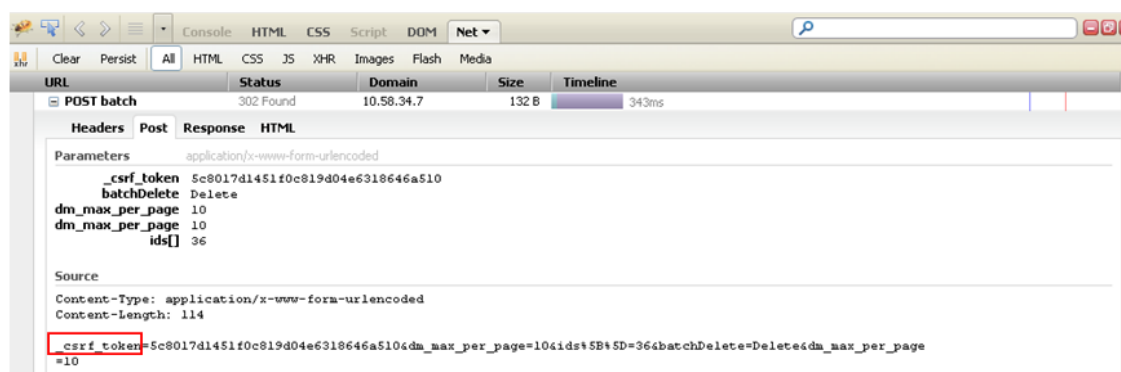


Hình 2.6: Kiểm tra lỗ hổng CFRS_1

Sau khi Save hoặc Delete, kiểm tra ứng dụng có sử dụng biến token hay không?



Hình 2.7: Kiểm tra lỗ hổng CFRS_2



Hình 2.8: Kiểm tra lỗ hổng CFRS_3

- Kiểm tra lỗi Path Traversal

- Mô tả: Lỗi Path Traversal xảy ra khi các biến do người dùng truyền lên (GET, POST) được đưa thẳng vào các hàm mở file, download file mà không qua xử lý. Khi đó, kẻ tấn công có thể truyền vào các kí tự đặc biệt như ../, ..\ để nhảy thư mục, hoặc truyền vào đường dẫn tuyệt đối của 1 file như /etc/passwd, từ đó có thể truy cập các thông tin nhạy cảm của ứng dụng và hệ điều hành và tấn công leo thang để chiếm quyền điều khiển ứng dụng và hệ điều hành.

- Hướng dẫn kiểm tra lỗi:

- Lỗi Path Traversal thường xuất hiện ở các chức năng cho phép xem file trên máy chủ hay chức năng download file, nếu các biến GET, POST có kiểu giá trị như tên file, tên thư mục thì nhiều khả năng chức năng này bị mắc lỗi Path Traversal.

- Có thể kiểm tra bằng cách truyền vào chuỗi kí tự /, \ hoặc đường dẫn tuyệt đối của 1 file thuộc hệ điều hành như /etc/passwd, và theo dõi hồi đáp từ phía server, trong trường hợp tấn công thành công có thể lấy được nội dung file.

- Ví dụ minh họa

`http://some_site.com.br/get-files?file=../../../../some dir/some file`

Hoặc

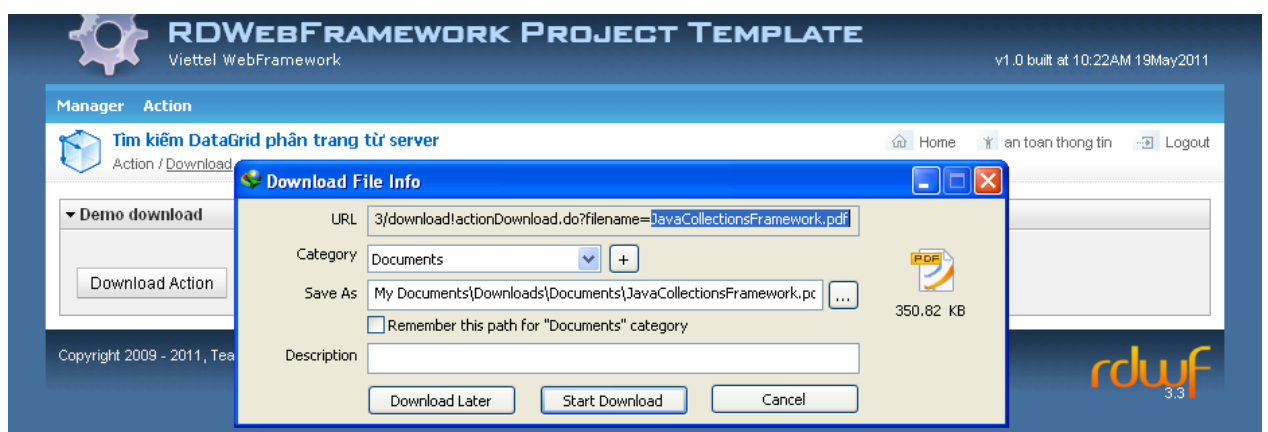
`http://some_site.com.br/../../../../etc/shadow`

`http://some_site.com.br/get-files?file=/etc/passwd`

Hình 2.9: Kiểm tra lỗi Path Traversal_1

Trong ứng dụng trên, chức năng get-files sử dụng 1 biến GET có dạng đường dẫn đến file, ta có thể thử bằng cách truyền vào các giá trị đường dẫn đến các file của hệ điều hành và ứng dụng, nếu có thể get được các file đó thì ứng dụng bị mắc lỗi Path Traversal.

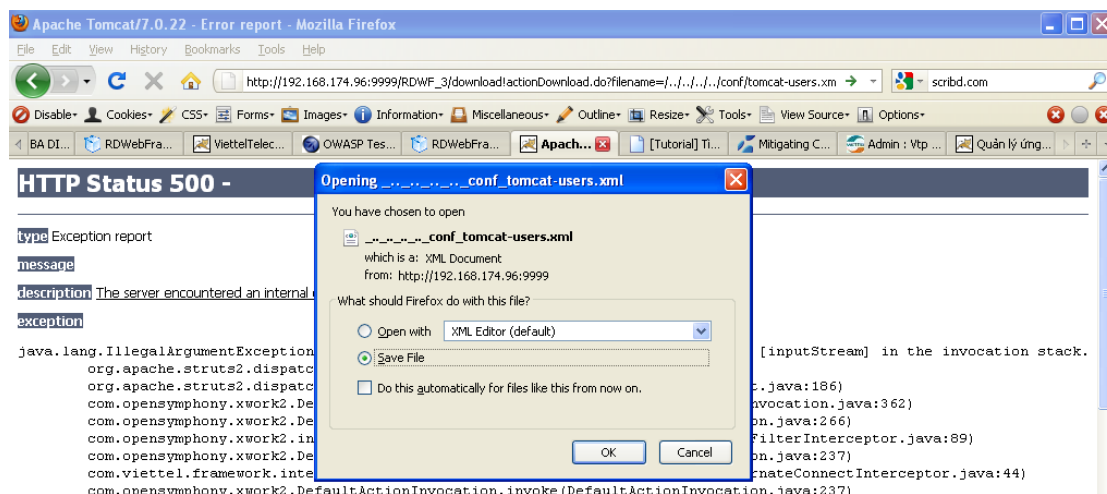
- Ví dụ:



Hình 2.10: Kiểm tra lỗi Path Traversal_2

- Kẻ tấn công lấy file omcat-users.xml để có thông tin về user/pass của tomcat.

`http://192.168.174.96:9999/RDWF_3/download!actionDownload.do?filename=../../../../
../conf/tomcat-users.xml`



Hình 2.11: Kiểm tra lỗi Path Traversal_3

Thông tin user/pass tomcat

```
<?xml version="1.0" encoding="utf-8" ?>
- <tomcat-users>
  <role rolename="manager-gui" />
  <role rolename="manager-script" />
  <role rolename="manager" />
  <role rolename="admin-gui" />
  <role rolename="admin-script" />
  <role rolename="admin" />
  <user username="admin" password="admin" roles="manager-gui,admin-gui,manager,admin,manager-script,admin-script" />
</tomcat-users>
```

Hình 2.12: Kiểm tra lỗi Path Traversal_4

- Kiểm tra lỗi xác thực/phân quyền

• Mô tả: Trong ứng dụng, đôi khi việc thực hiện xác thực và phân quyền không đầy đủ, có khi chỉ dựa trên việc che dấu các link hay không hiển thị các nút chức năng đối với người dùng thông thường. Tuy nhiên, nếu người dùng thông thường biết được các link, nút đó thì có thể tạo các request trực tiếp tới server mà không cần click vào link, nút, để thực hiện thao tác. Khi đó, cơ chế xác thực/phân quyền của ứng dụng hoàn toàn bị phá vỡ.

• Hướng dẫn kiểm tra lỗi: Để kiểm tra lỗi này có thể sử dụng hai account, trong đó có một account có quyền thấp và một account có quyền cao hơn. Khi sử dụng account có quyền cao truy cập vào các chức năng dành riêng ta ghi lại các đường dẫn trên URL, sau đó đăng nhập vào bằng account quyền thấp và thử truy cập vào link đó. Nếu ứng dụng cho phép account truy cập thì cơ chế xác thực phân quyền của ứng dụng không có tác dụng.

• Ví dụ:

Ngày	Đăng ký	Đăng ký SMS	Đăng ký WAP	Gia hạn	Download	Xem online	Gửi tặng	Thành tiền
15-09-2011	406	37	369	0	69	133	0	2,503,000
Tổng	406	37	369	0	69	133	0	2,503,000

Hình 2.13: Kiểm tra lỗi xác thực phân quyền

Trong ví dụ trên, ta dùng account viettel_cp1 để truy cập vào chức năng thống kê doanh thu của dịch vụ. Thông thường, khi truy cập vào account Admin trên bảng điều khiển có 1 nút “Thống kê” để thống kê doanh thu, còn khi truy cập bằng account viettel_cp1 thì không có nút này. Tuy nhiên, nhìn trên thanh URL ta có thể thấy đường dẫn để truy cập vào chức năng này, ta sao chép lại, sau đó đăng nhập bằng account viettel_cp và thử truy cập thì thấy thành công. Từ đó có thể kết luận chức năng xác thực, phân quyền của ứng dụng không có tác dụng.

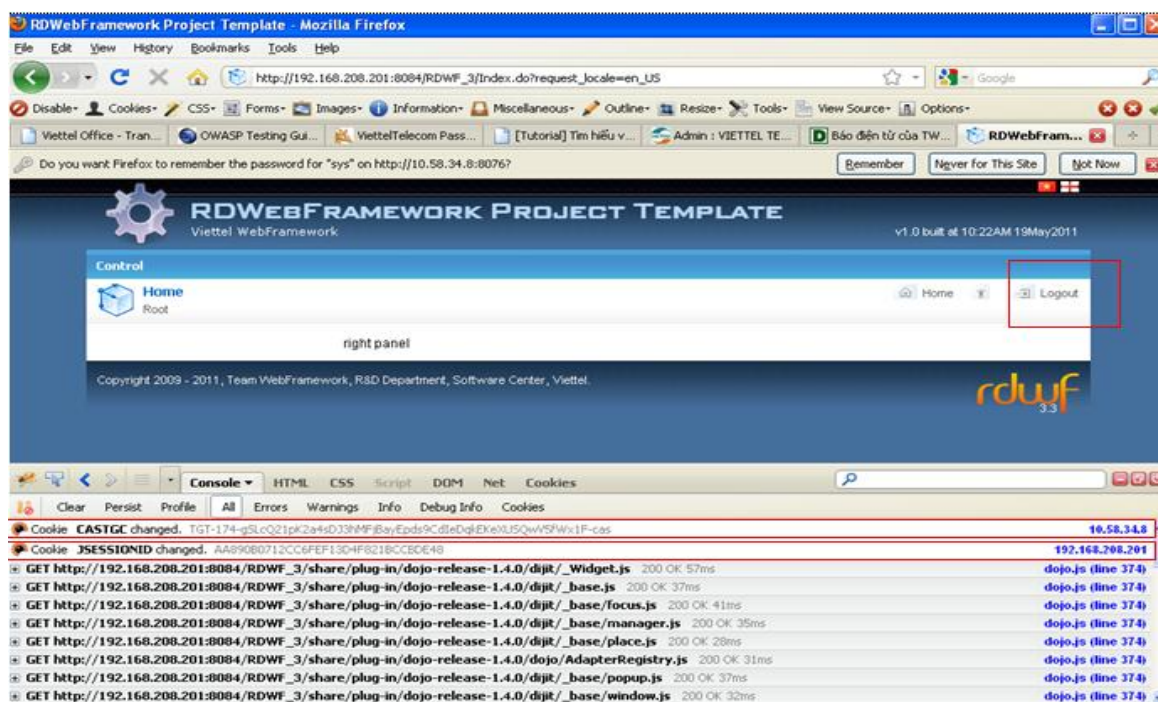
- Kiểm tra lỗi User enumeration

Đăng nhập vào ứng dụng (cố tình nhập tên đăng nhập hoặc mật khẩu sai). Nếu ứng dụng trả về câu thông báo cụ thể là “sai mật khẩu” hoặc “sai tên đăng nhập” thì ứng dụng mắc lỗi. Còn nếu trả về câu thông báo chung ví dụ “Tên đăng nhập hoặc mật khẩu không đúng” thì không bắt lỗi.

- Kiểm tra lỗi Session fixation

• Mô tả: Lỗi này xảy ra khi session-id trước và sau khi đăng nhập vào ứng dụng không thay đổi. Kẻ tấn công có thể lợi dụng lỗ hổng này để đăng nhập mà không cần user/pass.

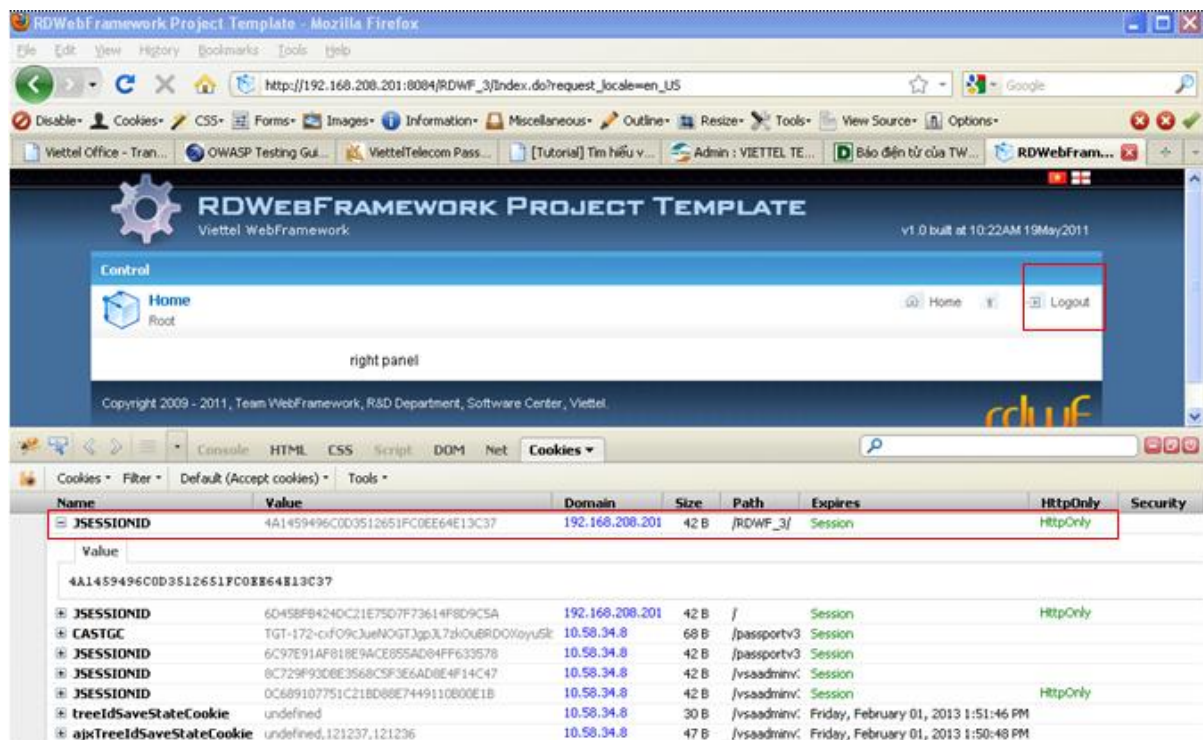
- Hướng dẫn kiểm tra lỗi:
 - Cài và bật Add-ons FireBug cho trình duyệt Firefox
 - Đăng nhập vào ứng dụng, và kiểm tra Cookie trước và sau khi đăng nhập
 - So sánh Cookie trước và sau đăng nhập. Nếu giống nhau thì bị lỗi.
 - Ví dụ:



Hình 2.14: Kiểm tra lỗi Session fixation

- Lỗi HTTP Only Cookie

- Cài và chạy Add-ons Firecookie và FireBug cho FireFox
- Đăng nhập vào ứng dụng đồng thời kiểm tra HTTP Only trong FireBug.
- Ví dụ:



Hình 2.15: Lỗi HTTP Only Cookie

- **Kiểm tra lỗi lỗ hổng cho phép dò đoán mật khẩu để thu thập danh sách, thông tin người dùng**

- Mô tả: 1 số ứng dụng không có cơ chế bảo vệ đăng nhập như đăng nhập sai quá số lần quy định cần yêu cầu người dùng nhập các kí tự từ 1 bức ảnh (captcha), hoặc tạm thời khóa account trong 1 khoảng thời gian nhất định. Khi đó, kẻ tấn công có thể tiến hành tấn công bằng cách thử các mật khẩu có trong 1 từ điển (tấn công dạng từ điển), hoặc thử tất cả các khả năng của mật khẩu (bruteforce), với các mật khẩu đơn giản có thể dễ dàng bị tìm ra. Ngoài ra, với các ứng dụng có cung cấp các tính năng tìm kiếm, tra cứu thông tin người dùng mà không có biện pháp kiểm soát thì kẻ tấn công có thể sử dụng các script để tự động hóa quá trình truy vấn và thu thập được thông tin của người dùng hoặc ứng dụng.

- Hướng dẫn kiểm tra lỗi: Các ứng dụng không có cơ chế bảo vệ sử dụng captcha, lock account khi đăng nhập sai quá số lần quy định đều bị mắc lỗi này.

2.6.4. Kiểm thử chấp nhận

- Mục đích: Kiểm thử chấp nhận còn gọi là kiểm thử nghiệm thu nhằm mục đích chứng minh phần mềm thỏa mãn tất cả yêu cầu của khách hàng và khách hàng đã chấp nhận sản phẩm.
- Người thực hiện: Khách hàng.
- Có 2 phương pháp kiểm thử chấp nhận: Kiểm thử alpha và kiểm thử beta.

2.6.4.1. Kiểm thử alpha

- Người dùng kiểm thử phần mềm ngay tại nơi phát triển phần mềm dưới sự hỗ trợ của nhân viên kiểm thử, nhân viên kiểm thử sẽ ghi nhận các lỗi hoặc phản hồi của khách hàng và báo lại với đơn vị phát triển phần mềm để lên kế hoạch sửa chữa.

2.6.4.2. Kiểm thử Beta

- Phần mềm sẽ được gửi tới cho người dùng để kiểm thử trong môi trường thực, lỗi hoặc phản hồi cũng sẽ gửi lại cho đơn vị phát triển phần mềm để lên kế hoạch sửa chữa.

2.6.5. Kiểm thử hồi qui

Kiểm thử hồi qui là một hoạt động cần thiết để chỉ ra rằng việc thay đổi mã nguồn không gây ra những ảnh hưởng bất lợi đến hệ thống nói chung.

Mục đích kiểm thử:	<ul style="list-style-type: none"> - Kiểm thử hồi qui dùng để kiểm tra các phần được sửa chữa và các phần liên quan đến các phần sửa chữa trong phần mềm, để đảm bảo rằng những sự thay đổi đó không gây ra lỗi trong những phần khác
Cách thực hiện:	<ul style="list-style-type: none"> - Tái sử dụng các kịch bản kiểm thử từ những phần kiểm thử trước để kiểm thử các mô-đun đã được sửa chữa. - Sử dụng công cụ kiểm thử tự động: Tạo một số test script về chức năng. - Xây dựng một chương trình phân tích sơ sở hạ tầng. Chúng ta dựng một cơ sở hạ tầng có thể mở rộng được để thực hiện và đánh giá chương trình phân tích. Dựa vào kết quả phân tích chúng ta xác định phạm vi cần kiểm thử hồi qui.
Điều kiện hoàn thành:	<ul style="list-style-type: none"> - Toàn bộ các trường hợp kiểm thử đã chọn được thực hiện và đạt yêu cầu.

Bảng 2.5: Kiểm thử hồi qui

2.7. Kiểm thử tự động**2.7.1. Kiểm thử tự động là gì? Quy trình kiểm thử tự động**

- Kiểm thử tự động là quá trình xử lý một cách tự động các bước thực hiện các test case. Kiểm thử tự động bằng một công cụ nhằm rút ngắn thời gian kiểm thử.
- Quy trình kiểm thử tự động gồm 4 bước:
 - Bước 1: Viết kịch bản kiểm thử, dùng công cụ kiểm thử để ghi lại các thao tác lên phần mềm cần kiểm tra và tự động sinh ra test script.
 - Bước 2: Chỉnh sửa để kịch bản kiểm thử thực hiện kiểm tra theo đúng yêu cầu đặt ra, làm theo trường hợp kiểm thử cần thực hiện.
 - Bước 3: Chạy kịch bản kiểm thử, giám sát hoạt động kiểm tra phần mềm của kịch bản kiểm thử.
 - Bước 4: Kiểm tra kết quả thông báo sau khi thực hiện kiểm thử tự động. Sau đó bổ sung, chỉnh sửa những sai sót.

2.7.2. Ưu điểm và nhược điểm của kiểm thử tự động

- Các ưu điểm có thể kể đến của kiểm thử tự động là:
 - Kiểm thử chính xác và có thể bao quát thông tin

- Theo dõi được chính xác kết quả từng giai đoạn và các báo cáo tổng hợp
 - Cần ít nhân lực trong quá trình kiểm thử
 - Chu kỳ kiểm thử diễn ra trong thời gian ngắn
 - Hiệu năng của kiểm thử các lớp vượt xa tầm với của kiểm thử thủ công
- Tuy nhiên không thể không kể đến các nhược điểm của kiểm thử tự động:
- Chi phí cao cho việc chuyển giao công nghệ và đào tạo nhân viên
 - Tốn chi phí đầu tư lớn cho việc phát triển công cụ kiểm thử tự động
 - Tốn chi phí và thời gian cho việc tạo các kịch bản kiểm thử và bảo trì các kịch bản kiểm thử
 - Giai đoạn chuẩn bị kiểm thử yêu cầu nhiều nhân lực
 - Khu vực kiểm thử tự động có thể không bao quát đầy đủ, không áp dụng được trong việc tìm lỗi mới của phần mềm.

2.7.3. Các trường hợp nên áp dụng kiểm thử tự động

Không phải lúc nào cũng nên áp dụng kiểm thử tự động trong việc kiểm thử phần mềm, vì nhiều khi chi phí và thời gian cho việc kiểm thử tự động còn lớn hơn nhiều so với kiểm thử thủ công. Dưới đây là một số trường hợp nên áp dụng phương pháp kiểm thử tự động để đạt được hiệu quả cao về thời gian, chi phí cũng như chất lượng.

- **Trường hợp không đủ tài nguyên:** Là khi số lượng trường hợp kiểm thử lặp lại quá nhiều trên nhiều môi trường kiểm thử khác nhau, không có đủ nguồn nhân lực để kiểm thử thủ công trong một giới hạn thời gian nào đó.
- **Trường hợp kiểm thử hồi qui:** Trong quá trình phát triển phần mềm, nhóm lập trình thường đưa ra nhiều phiên bản phần mềm liên tiếp để kiểm thử. Thực tế cho thấy việc đưa ra các phiên bản phần mềm có thể là hàng ngày, mỗi phiên bản bao gồm những tính năng mới, hoặc tính năng cũ được sửa lỗi hay nâng cấp. Việc bổ sung hoặc sửa lỗi mã chương trình cho những tính năng ở phiên bản mới có thể làm cho những tính năng khác đã kiểm tra tốt chạy sai mặc dù phần mã chương trình của nó không hề chỉnh sửa. Để khắc phục điều này, đối với từng phiên bản, kiểm thử viên không chỉ kiểm tra chức năng mới hoặc được sửa, mà phải kiểm tra lại tất cả những tính năng đã kiểm tra tốt trước đó. Điều này khó khả thi về mặt thời gian nếu kiểm thử thủ công.
- **Trường hợp kiểm thử khả năng vận hành phần mềm trong môi trường đặc biệt:** Đây là kiểm thử nhằm đánh giá xem vận hành của phần mềm có thỏa mãn yêu cầu đặt ra hay không. Thông qua đó kiểm thử viên có thể xác định được các yếu tố về phần cứng, phần mềm ảnh hưởng đến khả năng vận hành của hệ thống. Có thể liệt kê một số tình huống kiểm tra tiêu biểu thuộc loại này như sau:
 - Đo tốc độ trung bình xử lý một yêu cầu của web server.
 - Thiết lập 1000 yêu cầu, đồng thời gửi đến web server để kiểm tra tình huống 1000 người dùng truy xuất web cùng lúc.
 - Xác định số yêu cầu tối đa được xử lý bởi web server hoặc xác định cấu hình máy thấp nhất mà tốc độ xử lý của phần mềm vẫn có thể hoạt động ở mức cho phép.

2.8. Tổng kết chương 2

Chương 2 của đồ án đã trình bày được các vấn đề cơ bản về kiểm thử phần mềm. Các vấn đề chính được trình bày bao gồm:

- Một số định nghĩa của kiểm thử phần mềm
- Mục tiêu của kiểm thử phần mềm
- Các nguyên tắc cơ bản của kiểm thử phần mềm
- Các bước cơ bản của một quy trình kiểm thử phần mềm
- Các kỹ thuật kiểm thử phần mềm cơ bản
- Các giai đoạn thực hiện kiểm thử phần mềm
- Giới thiệu sơ lược về kiểm thử tự động

CHƯƠNG 3: CÔNG CỤ KIỂM THỬ TỰ ĐỘNG SELENIUM

3.1. Tổng quan về Selenium

3.1.1. Selenium là gì?

Selenium (thường được viết tắt là SE) là một phần mềm mã nguồn mở, được phát triển bởi Jason Huggins, sau đó được tiếp tục phát triển bởi nhóm ThoughtWorks vào năm 2004.

Selenium là một bộ các công cụ hỗ trợ kiểm thử tự động các tính năng của ứng dụng web, bao gồm 4 phần: Selenium IDE, Selenium Remote Control (RC), Selenium Core và Selenium Grid.

Selenium hỗ trợ kiểm thử trên hầu hết các trình duyệt web phổ biến hiện nay như Firefox, Internet Explorer, Googlechrome và hỗ trợ trên rất nhiều ngôn ngữ lập trình phổ biến như C#, Java, Python, PHP. Không những vậy, Selenium còn có thể kết hợp với một số công cụ kiểm thử khác như Junit, Bromien, Nunit.

3.1.2. 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 để 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 hầu 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.

Đồ án trình bày cụ thể về hai thành phần của bộ công cụ Selenium là Selenium IDE và Selenium RC. Các hướng dẫn cụ thể về Selenium IDE và Selenium RC sẽ được trình bày chi tiết ở phần sau của đồ án.

3.2. Selenium IDE

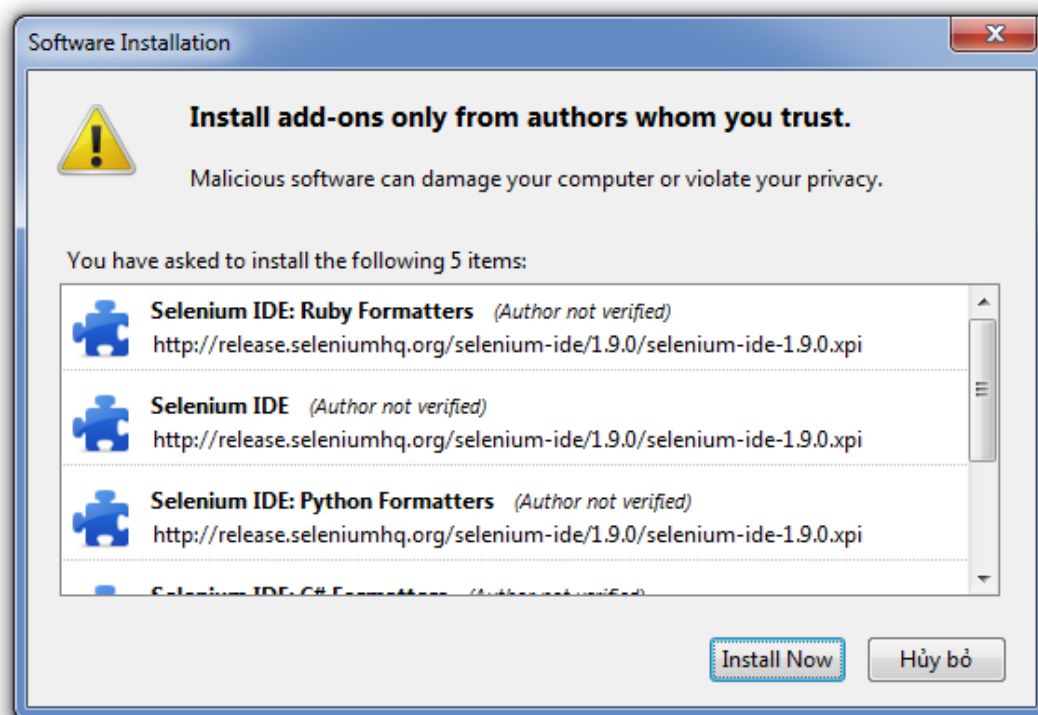
Selenium IDE là một add-on của Mozilla Firefox phiên bản 2.0 trở lên, ban đầu được phát triển bởi Shinya Kasatani theo hướng sử dụng Selenium Core mà không cần cài đặt Selenium vào máy chủ ứng dụng. Nó được xây dựng sử dụng JavaScript do vậy mà nó có thể tương tác với DOM (Document Object Model), sử dụng được những cách gọi JavaScript.

Selenium cho phép ghi lại những hành động trong luồng công việc cần kiểm tra bằng các chức năng Record và Playback.

Selenium IDE cũng chứa một menu ngữ cảnh cho phép lựa chọn yếu tố giao diện người dùng từ các trình duyệt đang hiển thị trang và sau đó chọn từ một danh sách các lệnh Selenium và các thông số được xác định theo ngữ cảnh của phần giao diện người dùng lựa chọn.

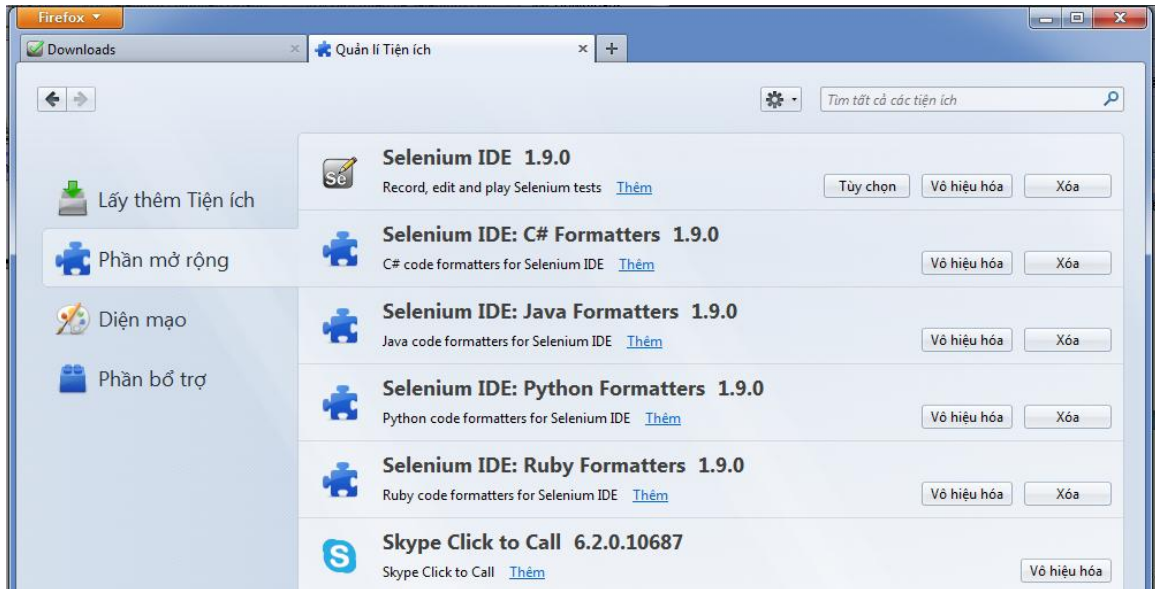
3.2.1. Cài đặt Selenium IDE

- Bước 1: Vào trang <http://seleniumhq.org/download> để download Selenium IDE
- Bước 2: Click vào link download cho Selenium IDE. Bạn sẽ nhận được tin nhắn "Firefox prevented this site (seleniumhq.org) from asking you to install software on your computer" (Firefox đã chặn phần mềm từ trang web (seleniumhq.org), bạn có chắc chắn muốn cài đặt trên máy tính của bạn không). Nếu thực hiện, click nút Allow.
- Bước 3: Một pop up xuất hiện như hình:



Hình 3.1: Pop up cài đặt Selenium

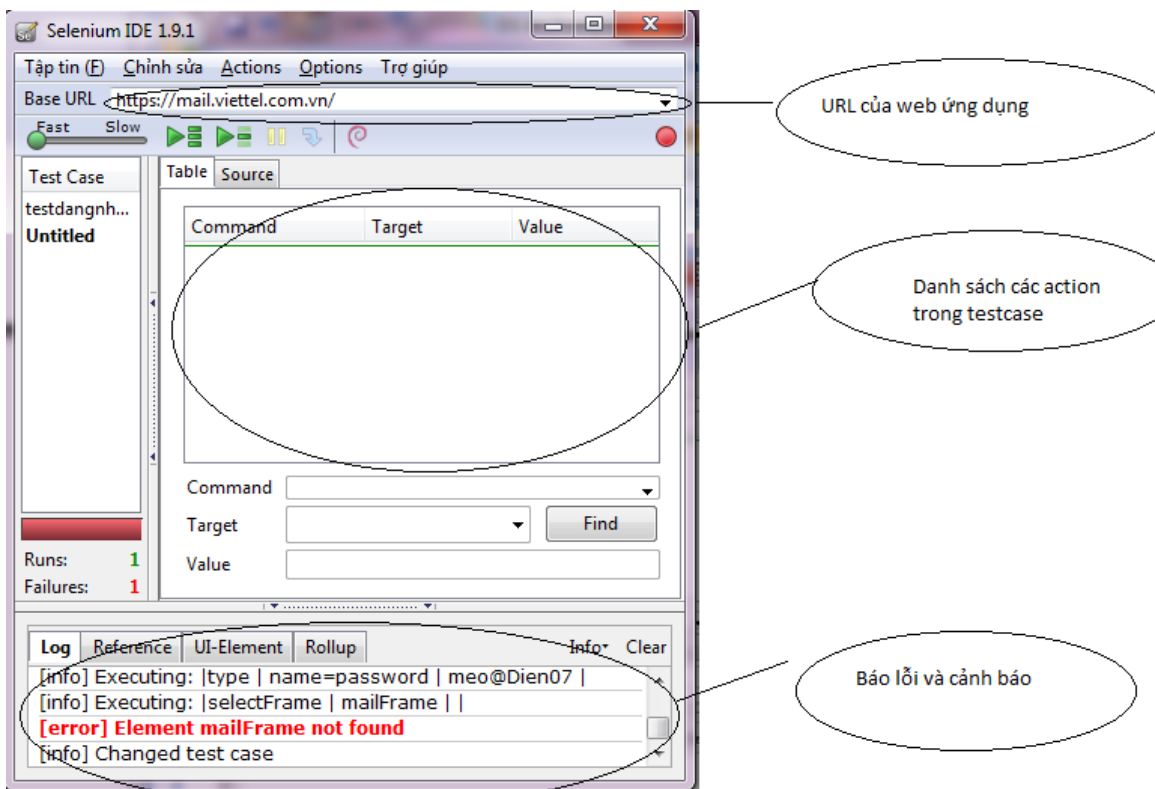
- Bước 4: Firefox thực hiện đếm ngược, nút Cài đặt chuyển sang trạng thái active, có thể click được. Selenium sẽ bắt đầu được cài đặt trong máy tính giống như 1 add-on của firefox.
- Bước 5: Tiến trình cài đặt hoàn thành, hệ thống hỏi bạn có muốn khởi động lại firefox không. Click vào nút Restart. Firefox sẽ đóng và mở lại.
- Bước 6: Kiểm tra lại phần add-on của firefox xem đã có Selenium chưa. Hiển thị như hình thì việc cài Selenium đã thành công.



Hình 3.2: Kiểm tra cài đặt Selenium thành công

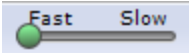





3.2.2. Các icon của Selenium IDE

Phần này giải thích một số ký hiệu và thành phần của Selenium IDE.



Hình 3.3: Các icon của Selenium IDE

Giải thích một số ký hiệu:

- Base URL: Đây là nơi điền URL của ứng dụng web được tiến hành kiểm thử.
- Thanh trượt : Đây là thanh trượt nằm dưới nhãn trên màn hình. Dùng để điều chỉnh tốc độ nhanh/chậm khi chạy test case.
- Nút : Chạy tất cả các test case.
- Nút : Chỉ chạy test case được chọn.
- Nút : Tạm dừng một test case đang chạy
- Nút : Bỏ qua một test case khi nó đã bị tạm dừng
- Nút : Nút thu được sử dụng để thu các test case qua những thao tác bạn tác động đến trang web cần kiểm thử.
- Textbox Command: Dòng lệnh
- Text box Target: Kết quả mong đợi của dòng lệnh
- Text box Value: Giá trị đầu vào của dòng lệnh

Bảng Selenium sẽ lưu lại các lệnh, kết quả mong đợi và giá trị đầu vào của các lệnh. Nếu click vào tab Source, ta có thể thấy Selenium IDE lưu trữ các test case có dạng HTML:

```
<tr>
  <td>open</td>
  <td>/chapter1</td>
  <td></td>
</tr>
```

Hình 3.4: Test case Selenium IDE

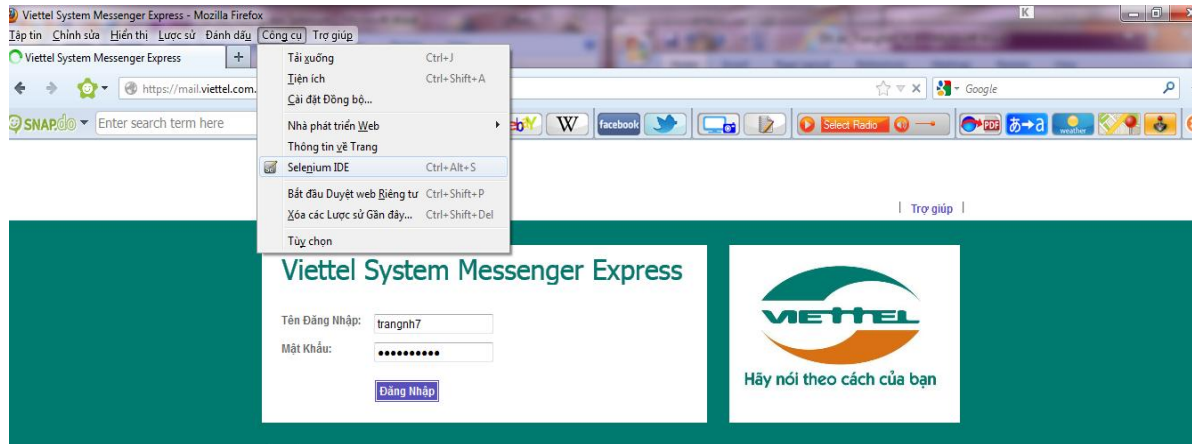
- Khu vực phía dưới textbox Value sẽ hiển thị các log của Selenium trong khi các test case chạy. Nếu có một test case bị thất bại Selenium IDE sẽ log một lỗi.
- Log: Hiển thị thông báo lỗi và các bước được thực thi trong quá trình chạy một test case tự động. Ngay cả khi ta không chọn tab log, các thông tin này vẫn hiển thị. Các thông tin này giúp ích cho nhân viên kiểm thử cũng như nhân viên lập trình trong quá trình tìm ra nguyên nhân lỗi đã phát hiện trong test case (nếu có).
- Reference: Thẻ tham chiếu
- UI-Element và Rollup: Tính năng nâng cao của Selenium IDE
- Lưu ý:
 - Các test case luôn luôn có điểm bắt đầu. Trong ngữ cảnh của Selenium, điều này có nghĩa là mở một trang nào đó để bắt đầu luồng công việc.
 - Các test case có thể không cần dựa trên những test case khác để chạy.
 - Một test case chỉ nên dùng để kiểm thử một chức năng nhỏ xác định trong một thời gian xác định.

3.2.3. Các thao tác thực hiện kiểm thử tự động với Selenium

3.2.3.1. Recording_Thực hiện thu một kịch bản với Selenium IDE

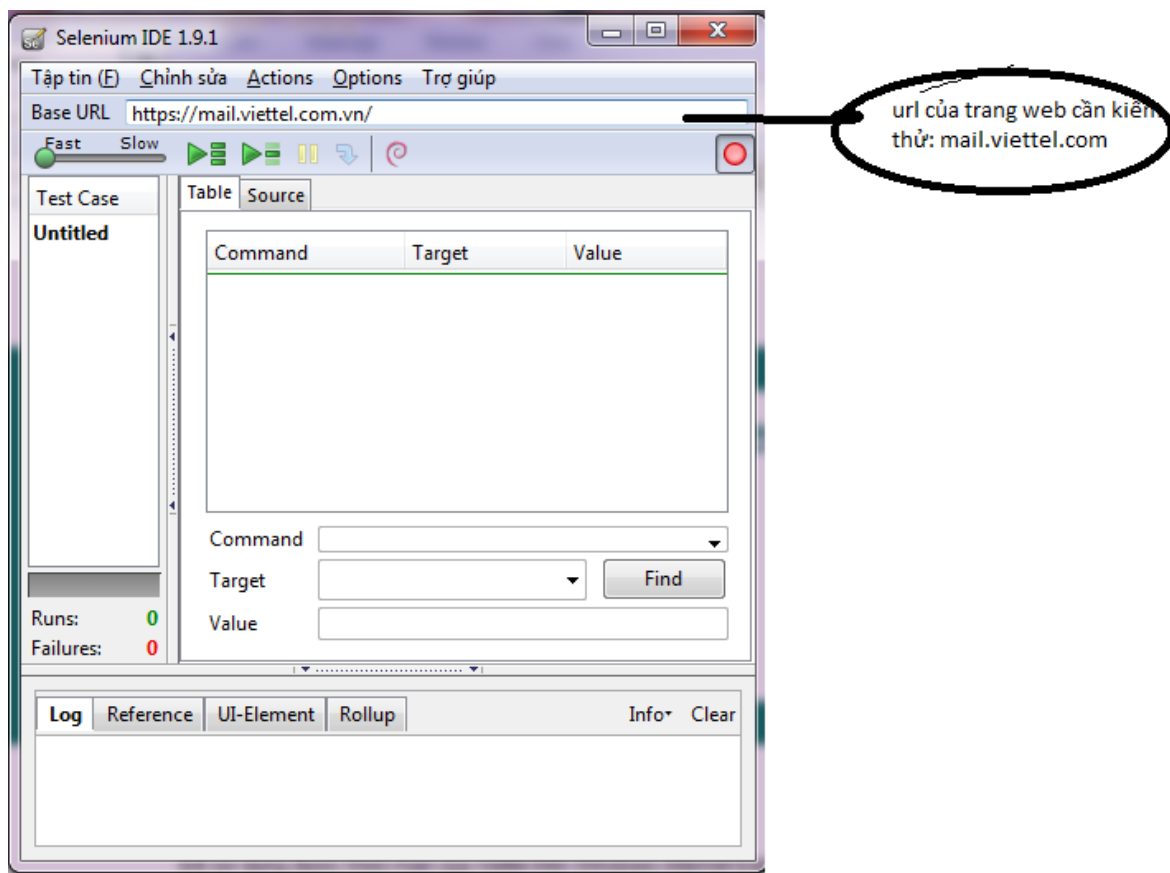
Các bước để bắt đầu thu lại test case:

- **Bước 1:** Vào Firefox/công cụ/chọn Selenium IDE hoặc nhấn tổ hợp phím Ctrl+Alt+s



Hình 3.5: Thực hiện thu các trường hợp kiểm thử_1

- **Bước 2:** Thay đổi mục Based URL thành URL của ứng dụng cần kiểm thử. Lấy ví dụ ứng dụng web cần kiểm thử có url là: <https://mail.viettel.com.vn/>
 - Nút thu mặc định ở trạng thái "now recording, click to stop recording".



Hình 3.6: Thực hiện thu các trường hợp kiểm thử_2

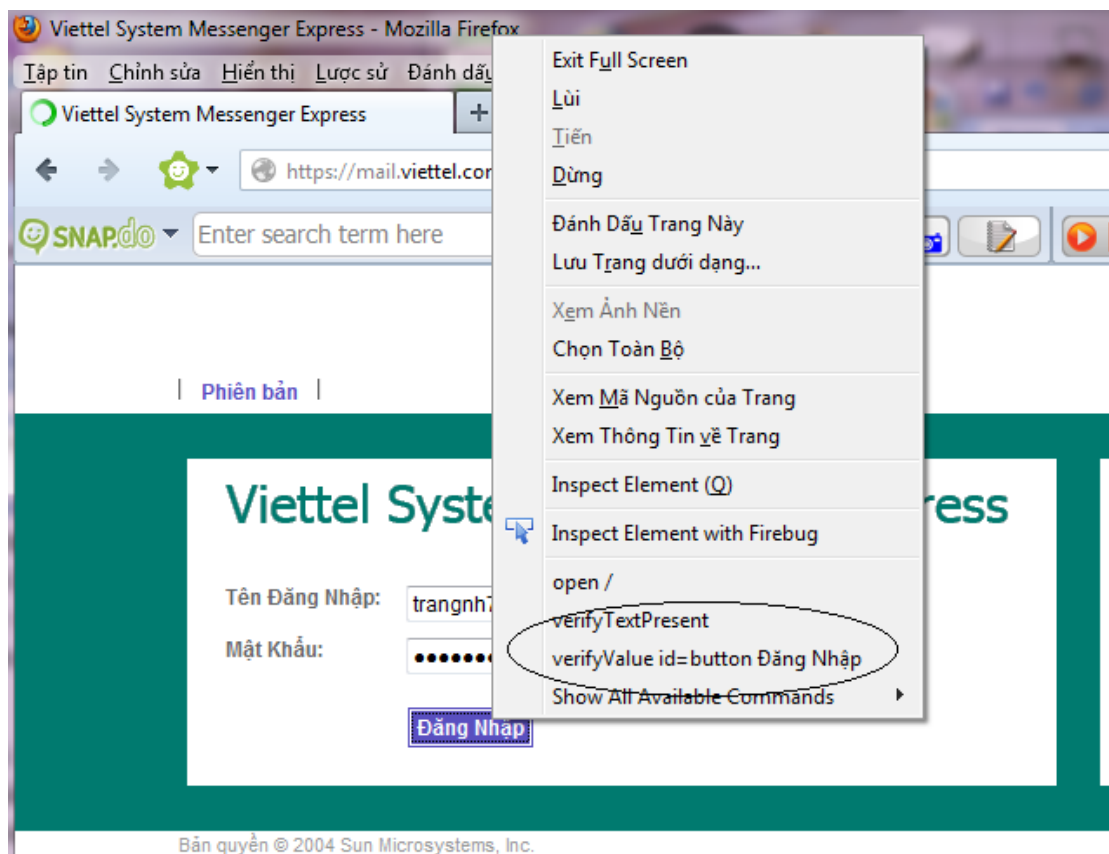
- **Bước 3:** Tiến hành các thao tác cần kiểm thử trên link
 - Ví dụ: Ta thực hiện kiểm thử tự động trường hợp đăng nhập vào trang web thành công với username/pass word hợp lệ.

- Trong quá trình thu, Selenium IDE sẽ tự động chèn thêm các lệnh vào test case dựa trên hành động của người thực hiện. Các command được tự động thêm phổ biến:
 - Click a link- *click* or *clickAndWait* commands
 - Nhập các giá trị- *type* command
 - Chọn các giá trị từ một select box - *select* command
 - Click vào các checkboxe hoặc các radio button - *click* command
- **Bước 4:** Click vào nút thu. Nút thu ở trạng thái "Click to record".
- **Bước 5:** Save as test case
- **Một số lưu ý:** Sau một liên kết thường ghi lại một lệnh nhấp chuột, phải thay đổi tốc độ chạy của test case để đảm bảo test case tạm dừng cho đến khi trang mới được tải xong. Nếu không, test case sẽ tiếp tục chạy trước khi các trang đã được nạp tất cả các yếu tố của nó. Điều này sẽ gây ra test case bị thất bại.

3.2.3.2. Thêm các lệnh khẳng định và xác nhận với menu ngữ cảnh

Các trường hợp kiểm kiểm thử các thuộc tính của một trang web sẽ đòi hỏi các lệnh xác minh (verify) và khẳng định (assert) các yếu tố trên giao diện. Phần dưới đây sẽ trình bày cách thêm các lệnh này vào test case của chúng ta.

Khi thu một test case với Selenium IDE, vào trình duyệt hiển thị website ta muốn thực hiện kiểm thử, trỏ chuột phải vào bất cứ vị trí nào trên trang, ta sẽ thấy các lệnh xác minh và khẳng định như hình dưới. Để sử dụng các lệnh này ta chỉ việc chọn lệnh xác minh hoặc khẳng định mong muốn. Các lệnh này sẽ tự động hiển thị trong test case. Selenium sẽ dự đoán các lệnh, các thông số cần có trên giao diện để bổ xung các lệnh xác minh. Khi chọn thông báo Show All Available Commands, sẽ có nhiều lệnh xác minh được gợi ý hơn.



Hình 3.7: Lệnh xác minh (verify) một yếu tố trên trang web

3.2.3.3. Các thao tác chỉnh sửa

- **Chèn lệnh:**

- Chèn vào bảng: Trong ô test case, click chuột trái tại vị trí muốn chèn lệnh. Chuột phải và chọn Insert command. Selenium IDE sẽ thêm một dòng trống phía trước dòng được chọn. Nhập lệnh vào ô command, kết quả mong muốn vào ô target, giá trị đầu vào vào ô value.
- Chèn vào mã nguồn: Chọn vị trí trong test case mà bạn muốn chèn lệnh. Trong ô test case, chuột trái vào vị trí muốn chèn lệnh. Vào tag HTML, cần tạo 3 dòng chứa lệnh bao gồm tham số đầu tiên (nếu lệnh yêu cầu có tham số), tham số thứ hai (nếu có). Lưu test case trước khi chọn lại table view.

- **Chèn comment:** Các comment có thể được thêm vào cho test case để hiểu hơn. Những comment được bỏ qua khi chạy test case. Comment có thể được sử dụng để thêm vào các khoảng trống dọc (một hoặc nhiều dòng trống) vào các test case của chúng ta, khi chúng ta tạo ra các comment trống. Một lệnh trống sẽ tạo ra 1 lỗi khi thực thi còn một comment trống thì không tạo ra lỗi khi thực thi.

- Chèn vào bảng: Chọn vị trí trong test case muốn comment. Click chuột phải và chọn Insert Comment. Sử dụng trường Command để nhập comment.
- Chèn vào mã nguồn: Chọn vị trí trong test case muốn chèn comment. Thêm một comment có dạng HTML. Ví dụ: `<!-- Enter your comment here -->`

- **Chỉnh sửa comment hay lệnh:**

- Chỉnh sửa qua giao diện: Chọn dòng cần chỉnh sửa và chỉnh sửa nó bằng các trường Command, Target, và Value.
- Chỉnh sửa qua mã nguồn: Vào mã nguồn, chỉnh sửa trực tiếp vào dòng comment hay lệnh muốn chỉnh sửa.

3.2.3.4. Mở và lưu lại một test case

- Chọn tập tin/ Open hoặc Save. Tuy nhiên Selenium có sự khác biệt giữa các test case và test suite. Để lưu lại các bước kiểm thử trên Selenium-IDE sau khi sử dụng, bạn có thể lưu lại một test case riêng lẻ, hay lưu nhiều test case dưới dạng một test suite. Nếu các test case của test suite không được lưu. Chương trình sẽ nhắc nhở ta lưu chúng trước khi lưu một test suite. Khi mở một test case hoặc một test suite đã có, Selenium-IDE hiển thị các câu lệnh trong ô test case.

3.2.3.5. Chạy các test case

Selenium IDE có nhiều lựa chọn để chạy test case. Bạn có thể chạy một test case, dừng và chạy tiếp, chạy một dòng lệnh riêng lẻ, hay chạy một test suite.

- Chạy một test case: Chọn một test case sau đó click vào nút Run để chạy một test case.
- Stop and Start: Nút Pause được dùng để tạm dừng một test case khi nó đang chạy. Để tiếp tục chạy test case bị tạm dừng, click nút Resume.
- Tạm dừng ở giữa: Bạn có thể chọn một điểm ở giữa test case để tạm dừng nó tại một câu lệnh đặc biệt. Điều này có ích trong việc gỡ lỗi trong test case. Để chọn một điểm dừng cho test case, chọn câu lệnh, chuột phải, chọn Set/Clear Start Point.
- Bắt đầu từ giữa: Chúng ta cũng có thể bắt đầu chạy một test case từ một điểm xác định ở giữa test case, thao tác này cũng được sử dụng trong gỡ lỗi. Để gán điểm bắt đầu, ta chọn câu lệnh làm điểm bắt đầu, chuột phải, chọn Set/Clear Start Point.

- Chạy một câu lệnh đơn lẻ bất kỳ: Double-Click câu lệnh muốn chạy. Việc này có ích khi viết một câu lệnh đơn lẻ.

3.2.4. Selenese

Tập lệnh Selenium gọi là Selenese là một tập các lệnh để chạy kịch bản kiểm thử. Một chuỗi các lệnh được gọi là một kịch bản kiểm thử. Phần dưới của đồ án sẽ trình bày chi tiết các lệnh thường được sử dụng trong Selenium.

Selenium cung cấp một tập đầy đủ các lệnh để kiểm thử các ứng dụng web. Trong selenese có thể kiểm thử tình trạng của các yếu tố giao diện người dùng dựa trên các thẻ HTML, kiểm thử nội dung xác định, kiểm thử các link hỏng, lỗi, các trường đầu vào, lựa chọn danh sách.

Một lệnh mô tả thao tác phải làm. Lệnh Selenium bao gồm ba yếu tố: Actions, accessors, assertion.

- Action: là các thao tác chung trên ứng dụng, ví dụ: “Click this link”, “Select that option”. Nếu như thao tác thất bại sẽ có 1 lỗi, việc thực thi kiểm thử sẽ bị tạm dừng. Một vài hành động sử dụng hậu tố “AndWait”, ví dụ: “ClickAndWait”. Selenium sử dụng hậu tố này trong trường hợp chờ một trang web được tải.
- Accessor: Kiểm tra trạng thái của ứng dụng và lưu trữ kết quả vào các biến. Ví dụ: “storeTitle”. Chúng có thể được sử dụng để sinh tự động các Assertion.
- Assertion: Giống như những Accessor, nhưng nó xác định trạng thái của ứng dụng thích nghi với kết quả mong đợi.

Assertion của Selenium có thể được chia thành 3 dạng: “assert”, “verify”, “waitFor”. Ví dụ: “assertText”, “verifyText”, “waitForText”. Khi “assert” thất bại, việc kiểm thử sẽ dừng lại. Khi “verify” thất bại, việc kiểm thử vẫn tiếp tục nhưng sẽ hiển thị một lỗi. Lệnh “waitFor” chờ một vài điều kiện được thực thi (có ích khi kiểm thử các ứng dụng Ajax), nó sẽ thành công nếu điều kiện đúng nhưng sẽ thất bại và tạm dừng việc kiểm thử nếu các điều kiện không đúng.

3.2.4.1. Cú pháp Script

Các lệnh Selenium rất đơn giản, nó bao gồm lệnh và 2 tham số.

Các tham số không nhất thiết phải có trong mọi trường hợp, nó phụ thuộc vào câu lệnh, trong một số trường hợp câu lệnh yêu cầu cả hai tham số, một số chỉ yêu cầu một tham số, và cũng có những câu lệnh không cần có tham số.

Ví dụ:

Câu lệnh	Tham số thứ nhất	Tham số thứ hai
goBackAndWait		
VerifyTextPresent		Wellcome!
Type	Id=name	Trangnh7
Type	Id= password	meo@Dien07

Bảng 3.1: Cú pháp câu lệnh Selenese

Phân loại tham số:

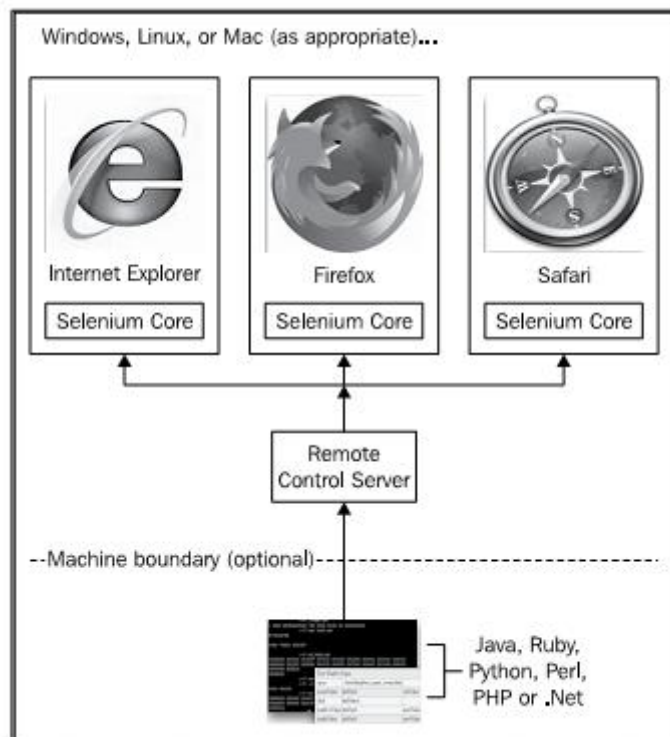
- Locator: Tham số xác minh các yếu tố trên giao diện người dùng.
- Text pattern: Tham số xác minh nội dung mong đợi của ứng dụng web.
- Selenium variable: Nhập văn bản trong một trường đầu vào để lựa chọn từ danh sách lựa chọn.

3.2.4.2. Một số lệnh thường sử dụng trong Selenium

- Open: Mở một ứng dụng web sử dụng URL.
- Click/clickAndWait: Thực thi click và đợi tải 1 trang web mới.
- VerifyTitle/assertTitle: xác nhận một tiêu đề trang được mong đợi.
- VerifyTextPresent: Xác nhận văn bản được mong đợi tại một vị trí nào đó trên trang.
- VerifyElementPresent: Xác nhận một yếu tố được mong đợi trên giao diện người sử dụng, được định nghĩa bởi thẻ HTML.
- VerifyText: Xác nhận văn bản được mong đợi và các thẻ HTML tương ứng.
- VerifyTable: Xác nhận các nội dung được mong đợi của 1 bảng.
- waitForPageToLoad: Tạm dừng thực thi lệnh cho đến khi trang web mong đợi được tải thành công, được gọi tự động khi sử dụng lệnh clickAndWait.
- waitForElementPresent: Tạm dừng thực thi lệnh cho tới khi một yếu tố giao diện người dùng xuất hiện trên trang web (được định nghĩa bởi các thẻ HTML).

3.3. Selenium Remote Control (Selenium RC)

Selenium RC ban đầu được phát triển bởi Patrick Lightbody theo hướng kiểm tra các ứng dụng web trên các trình duyệt khác nhau mà không cần cài đặt Selenium Core trên Server. Nó được phát triển để tương tác như một giao tiếp giữa ứng dụng cần kiểm tra và kịch bản kiểm thử. Selenium Core được tích hợp với Selenium RC thay cho việc cài đặt trên máy chủ.



Hình 3.8: Vai trò của Remote Control Server

Selenium RC là công cụ phục vụ cho các công việc kiểm thử đòi hỏi nhiều hơn việc thao tác với website trên giao diện. Selenium RC sử dụng ngôn ngữ lập trình để kiểm thử các trường hợp phức tạp hơn mà Selenium IDE không hỗ trợ.

3.3.1. Các thành phần của Selenium Remote Control

2 thành phần chính của Selenium RC là:

- **Máy chủ Selenium:** Thực hiện phân tích và chạy các lệnh được gửi đến từ ứng dụng cần kiểm thử và các thao tác như HTTP proxy, phân tích và xác minh các thông điệp HTTP giữa trình duyệt và ứng dụng cần kiểm tra.
- **Các thư viện máy khách:** Cung cấp giao tiếp giữa ngôn ngữ lập trình và máy chủ Selenium RC.

3.3.1.1. Máy chủ Selenium

Máy chủ Selenium nhận lệnh từ chương trình kiểm thử Selenium, thực hiện biên dịch nó và gửi lại thông báo kết quả của việc chạy các test case.

Máy chủ Selenium được tích hợp Selenium Core và tự động đưa nó vào trình duyệt. Điều này xảy ra khi chương trình được kiểm thử mở trên trình duyệt (sử dụng một chức năng thư viện máy khách API). Selenium-Core là một chương trình JavaScript, thực tế là một tập các chức năng JavaScript dùng để biên dịch và thực thi các lệnh Selenese sử dụng trình duyệt trong thông dịch JavaScript.

Máy chủ nhận các lệnh Selenese từ chương trình được kiểm thử sử dụng các đề nghị HTTP GET/POST đơn giản. Điều này có nghĩa là chúng ta có thể sử dụng mọi ngôn ngữ lập trình có khả năng gửi yêu cầu HTTP tới các kịch bản kiểm thử tự động trên trình duyệt.

3.3.1.2. Các thư viện máy khách

Các thư viện máy khách cung cấp giao diện hỗ trợ lập trình, cho phép chạy lệnh Selenium từ chương trình của chúng ta. Các thư viện máy khách hỗ trợ cho các ngôn ngữ lập trình khác nhau thì khác nhau. Giao diện lập trình (API) là một tập các chức năng chạy lệnh Selenium từ chương trình của chúng ta, trong mỗi giao diện có một chức năng lập trình hỗ trợ lệnh Selenium.

Thư viện máy khách sử dụng lệnh Selenese và chuyển tới máy chủ Selenium để xử lý các hoạt động cụ thể và kiểm tra ngược lại với các ứng dụng cần kiểm tra. Thư viện máy khách cũng nhận kết quả của lệnh và chuyển trở lại chương trình của chúng ta. Chương trình có thể nhận kết quả và lưu nó vào một biến chương trình và thông báo trở lại thành công hay thất bại hoặc có thể thực thi các hành động trực tiếp nếu nó là các lỗi không được mong đợi.

Để thực hiện kiểm thử một chương trình, ta cần viết một chương trình chạy một tập lệnh Selenese sử dụng thư viện khách API và nếu đã có một kịch bản kiểm thử được tạo bởi Selenium IDE, ta có thể sử dụng chức năng “Export the Selenium RC Code”. Selenium-IDE có thể biên dịch các lệnh Selenium của nó một các hàm gọi API của Drive khách.

3.3.2. Cài đặt Selenium Remote Control

Download Selenium Remote Control từ link: <http://seleniumhq.org/download/>

Sau khi download file zip Selenium RC từ trang download và giải nén ta sẽ thấy có một vài thư mục con. Những thư mục này có tất cả các thành phần cần sử dụng Selenium RC với ngôn ngữ lập trình ta chọn.

Khi đã chọn ngôn ngữ lập trình ta thực hiện:

- Cài đặt máy chủ Selenium RC
- Thiết lập một dự án sử dụng một ngôn ngữ cụ thể với driver máy khách cụ thể

3.3.2.1. Cài đặt máy chủ Selenium

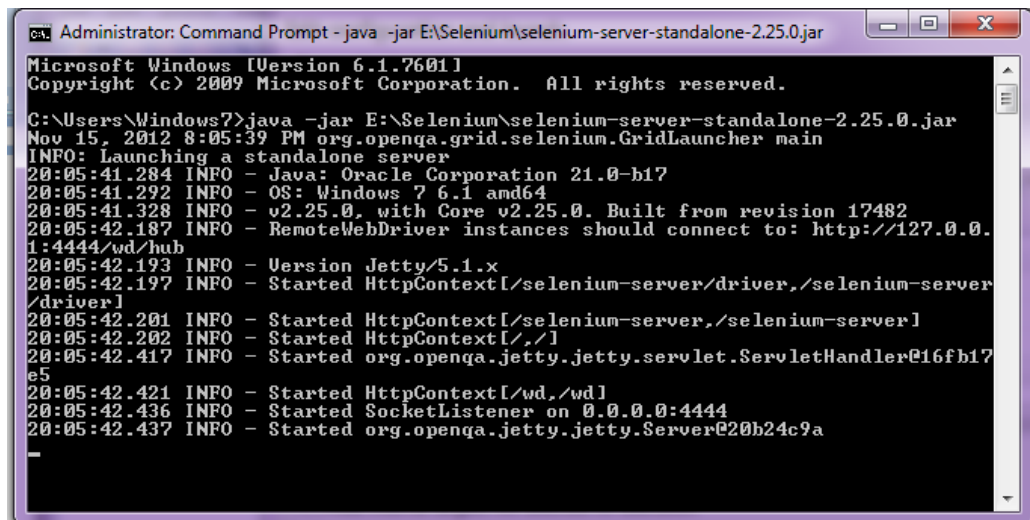
Máy chủ Selenium RC đơn giản là một file Jar của Java (Selenium-server-standalone-2.0.25.jar), nó không yêu cầu việc cài đặt đặc biệt nào. Ta chỉ cần tải về file zip và giải nén phần server trong thư mục mong muốn.

3.3.2.2. Chạy Selenium Server

Trước khi bắt đầu kiểm thử, ta phải khởi động Server bằng cách đi đến thư mục chứa máy chủ Selenium RC, chạy file từ bảng điều khiển dòng lệnh (Command-line console): Java-jar selenium-server-standalone-2.0.25.jar.

Để server có thể chạy, máy tính cần cài Java và biến môi trường PATH được cấu hình đúng để chạy từ bảng điều khiển. Chúng ta có thể check xem đã cài Java đúng các hay chưa bằng cách chạy lệnh java-version trên bảng điều khiển. Nếu nhận được một số phiên bản tức là đã cài java đúng cách.

Nếu chạy máy chủ Selenium thành công, Command Prompt sẽ như hình dưới:



```
Administrator: Command Prompt - java -jar E:\Selenium\selenium-server-standalone-2.25.0.jar
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Windows7>java -jar E:\Selenium\selenium-server-standalone-2.25.0.jar
Nov 15, 2012 8:05:39 PM org.openqa.grid.selenium.GridLauncher main
INFO: Launching a standalone server
20:05:41.284 INFO - Java: Oracle Corporation 21.0-b17
20:05:41.292 INFO - OS: Windows 7 6.1 amd64
20:05:41.328 INFO - v2.25.0, with Core v2.25.0. Built from revision 17482
20:05:42.187 INFO - RemoteWebDriver instances should connect to: http://127.0.0.1:4444/wd/hub
20:05:42.193 INFO - Version Jetty/5.1.x
20:05:42.197 INFO - Started HttpContext[/selenium-server/driver,/selenium-server/driver]
20:05:42.201 INFO - Started HttpContext[/selenium-server,/selenium-server]
20:05:42.202 INFO - Started HttpContext[/,/]
20:05:42.417 INFO - Started org.openqa.jetty.jetty.servlet.ServletHandler@16fb17e5
20:05:42.421 INFO - Started HttpContext[/wd,/wd]
20:05:42.436 INFO - Started SocketListener on 0.0.0.0:4444
20:05:42.437 INFO - Started org.openqa.jetty.jetty.Server@20b24c9a
```

Hình 3.9: Chạy máy chủ Selenium thành công

3.3.2.3. Sử dụng Java Client Driver

- Download Selenium RC từ trang <http://seleniumhq.org/download/>
- Giải nén file selenium-java-client-driver.jar
- Mở Java IDE (Eclipse, NetBeans, IntelliJ, Netweaver, v.v..)
- Tạo một project mới
- Thêm file selenium-java-client-driver.jar vào project
- Thêm file selenium-java-client-driver.jar vào classpath của project
- Từ Selenium-IDE xuất ra một kịch bản cho một file Java và đặt nó trong project Java, hoặc viết kịch bản kiểm thử Selenium bằng Java sử dụng API selenium-java-client. API sẽ được nói rõ ở phần sau của đồ án. Ta có thể dùng JUnit, hoặc TestNg để chạy kịch bản kiểm thử, hoặc cũng có thể viết thành một hàm main().
- Chạy Selenium sever từ bảng điều khiển
- Thực hiện việc kiểm thử từ Java IDE hoặc từ dòng lệnh

3.3.2.4. Sử dụng Python Client Driver

- Download Selenium RC từ trang <http://seleniumhq.org/download/>
- Giải nén file selenium.py
- Viết test case Selenium trong Python hoặc xuất một test case từ Selenium-IDE thành một file python (chức năng có trong Selenium IDE)
- Thêm file selenium.py vào đường dẫn của kịch bản kiểm thử
- Chạy Selenium server từ bảng điều khiển
- Thực thi kịch bản kiểm thử từ một bảng điều khiển hoặc từ Python IDE

3.3.2.5. Sử dụng .NET Client Driver

- Download Selenium RC từ trang <http://seleniumhq.org/download/>
- Giải nén thư mục
- Tải và cài đặt NUnit (ta có thể sử dụng NUnit như một phương tiện kiểm thử. Nếu chưa quen với NUnit, ta có thể viết một hàm main() đơn giản để chạy các kịch bản kiểm thử, dù vậy NUnit thực sự rất có ích trong vai trò một phương tiện kiểm thử)
- Mở .NET IDE (Visual Studio, SharpDevelop, MonoDevelop)
- Tạo một thư viện class (.dll)
- Add các truy vấn vào các DLL sau: nmock.dll, nunit.core.dll, nunit.framework.dll, ThoughtWorks.Selenium.Core.dll, ThoughtWorks.Selenium.IntegrationTests.dll và ThoughtWorks.Selenium.UnitTests.dll
- Viết kịch bản kiểm thử Selenium bằng một ngôn ngữ .NET (C# hoặc VB.Net), hoặc xuất một kịch bản từ Selenium-IDE thành một file C# và chép mã nguồn vào lớp vừa tạo
- Viết hàm main() hoặc có thể bao gồm NUnit trong project để chạy kịch bản kiểm thử
- Chạy máy chủ Selenium từ bảng điều khiển
- Chạy kịch bản kiểm thử từ IDE hoặc NUnit GUI hoặc từ dòng lệnh

3.3.2.6. Sử dụng Ruby Client Driver

- Nếu chưa có RubyGems thì phải thực hiện cài đặt từ RubyForge
- Chạy cài đặt gem từ selenium-client
- Thêm “selenium/client” vào phần trên cùng của kịch bản kiểm thử
- Viết kịch bản kiểm thử sử dụng bất kỳ dụng cụ kiểm thử Ruby nào (Ví dụ: Test::Unit, Mini::Test hoặc RSpec)
- Chạy Selenium server từ bảng điều khiển
- Thực thi kịch bản kiểm thử giống như chạy bất kỳ kịch bản Ruby nào khác

3.3.3. Các thao tác với Selenium RC**3.3.3.1. Chạy các kịch bản kiểm thử Selenium IDE với Selenium Remote Control**

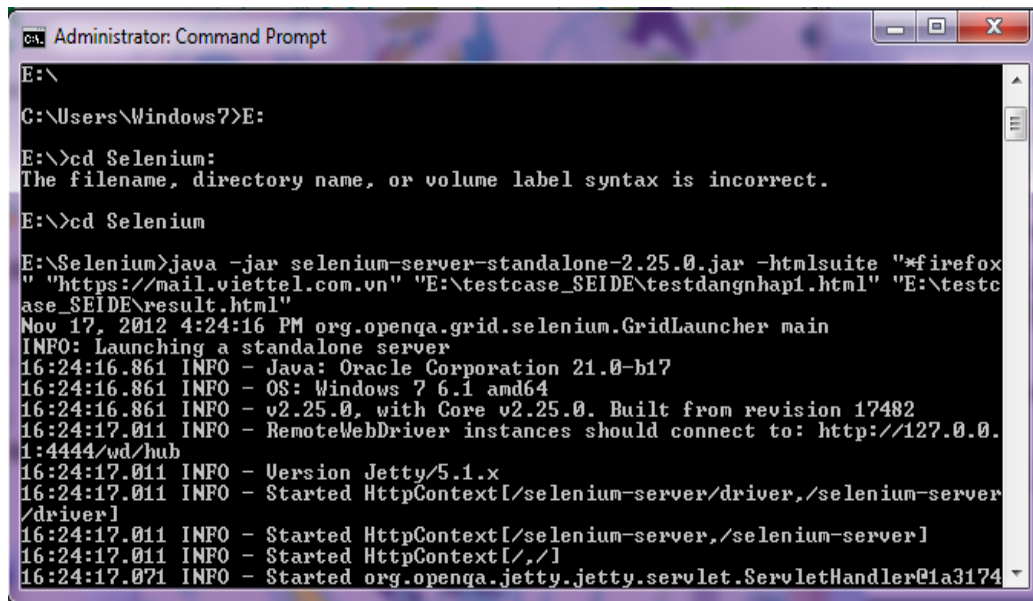
Để chạy các kịch bản kiểm thử Selenium trên Selenium RC chúng ta phải sử dụng biến –htmlsuite. Biến này gọi Selenium để mở Test Suite mà chúng ta đã tạo ra. Sau đó chúng ta cần xác định vị trí của Test Suite và vị trí lưu kết quả. Các lệnh chạy trong Command Prompt sẽ tương tự như dưới:

```
Java -jar selenium-server-standalone.jar -htmlsuite “*firefox” http://mail.viettel.com.vn
“E:\Testcase_SEIDE\testdangnhap1.html” “E:\Testcase_SEIDE\result.html”
```

Hình 3.10: Câu lệnh chạy testcase Selenium IDE bằng Selenium RC

Các bước thực hiện:

- Bước 1: Mở Command Prompt.
- Bước 2: Chạy câu lệnh sau:



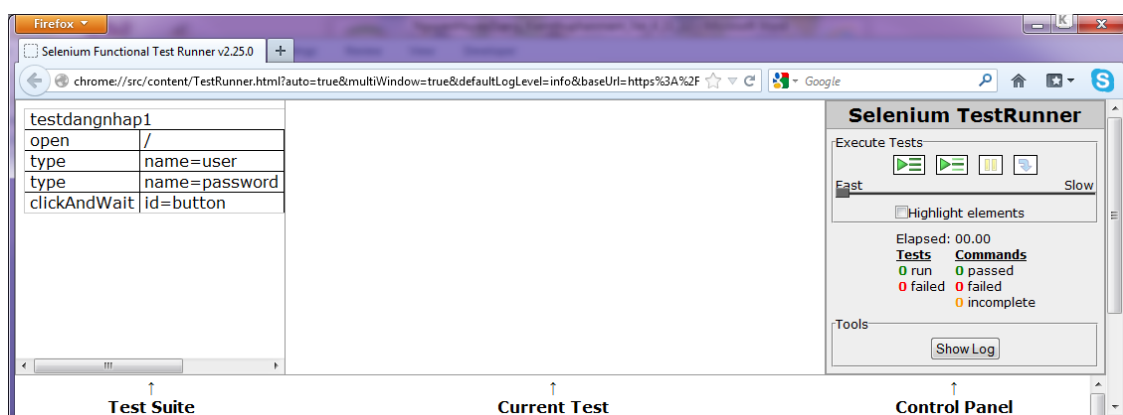
Hình 3.11: Chạy kịch bản kiểm thử Selenium IDE trên Selenium RC

Trong ví dụ, chúng ta đặt test case tại thư mục E:\Testcase_SEIDE.

Lưu ý :

- Khi chạy câu lệnh chúng ta nên gõ trực tiếp câu lệnh vào command prompt mà không nên gõ ra word và copy paste vì đôi khi làm như vậy sẽ xảy ra lỗi vì các bộ gõ không giống nhau.
- Khi chạy lệnh trên không cần phải chạy Selenium-server trước đó, vì lệnh này chứa cả phần khởi động Selenium-server.

Khi việc kiểm thử bắt đầu tiến hành, nó sẽ đưa đến hai cửa sổ trình duyệt. Cửa sổ đầu tiên giữ Selenium Core Framework với Test Suite ở bên phía tay trái, các bước kiểm thử ở phía trung tâm, và kết quả ở bên tay phải, tương tự như hình dưới:



Hình 3.12: Chạy kịch bản kiểm thử Selenium IDE trên Selenium RC

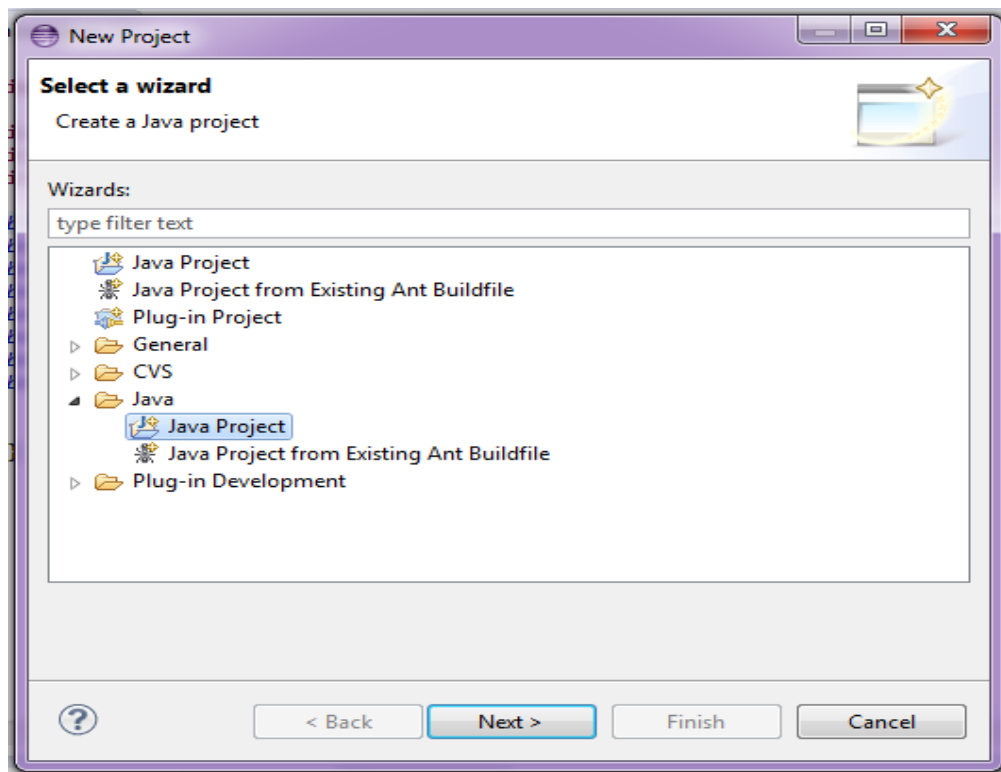
Sử dụng biến `-htmlsuite`, chúng ta điều khiển chạy các kịch bản kiểm thử Selenium IDE bằng Selenium Remote Control. Trong trường hợp này, chúng ta sử dụng Firefox để chạy các kịch bản kiểm thử Selenium IDE. Lệnh đã khởi động Firefox và tải URL của ứng dụng cần kiểm tra. Nó sẽ tải Test Suite của chúng ta và biết được nơi để lưu kết quả khi việc kiểm thử kết thúc.

- Lưu ý: Để chạy các kịch bản kiểm thử Selenium IDE trên các trình duyệt khác, chúng ta chỉ cần chỉnh sửa câu lệnh trên bằng việc thay *firefox bằng mã của trình duyệt mà chúng ta muốn thực thi kiểm thử trên đó và thực hiện các bước giống như trong Mozilla Firefox. Một số trình duyệt phổ biến:
 - *chrome
 - *ieexploreproxy
 - *ieexplore
 - *firefox3
 - *safariproxy
 - *googlechrome
 - *konqueror
 - *firefox2
 - *safari
 - *piexplore
 - *firefoxchrome
 - *opera
 - *iehta
 - *custom

3.3.3.2. Tạo một kịch bản kiểm thử mới với ngôn ngữ lập trình Java và Eclipse

Phần này của đồ án trình bày các bước để tạo một kịch bản kiểm thử Selenium RC chạy bằng hàm main với ngôn ngữ lập trình Java và bằng IDE Eclipse:

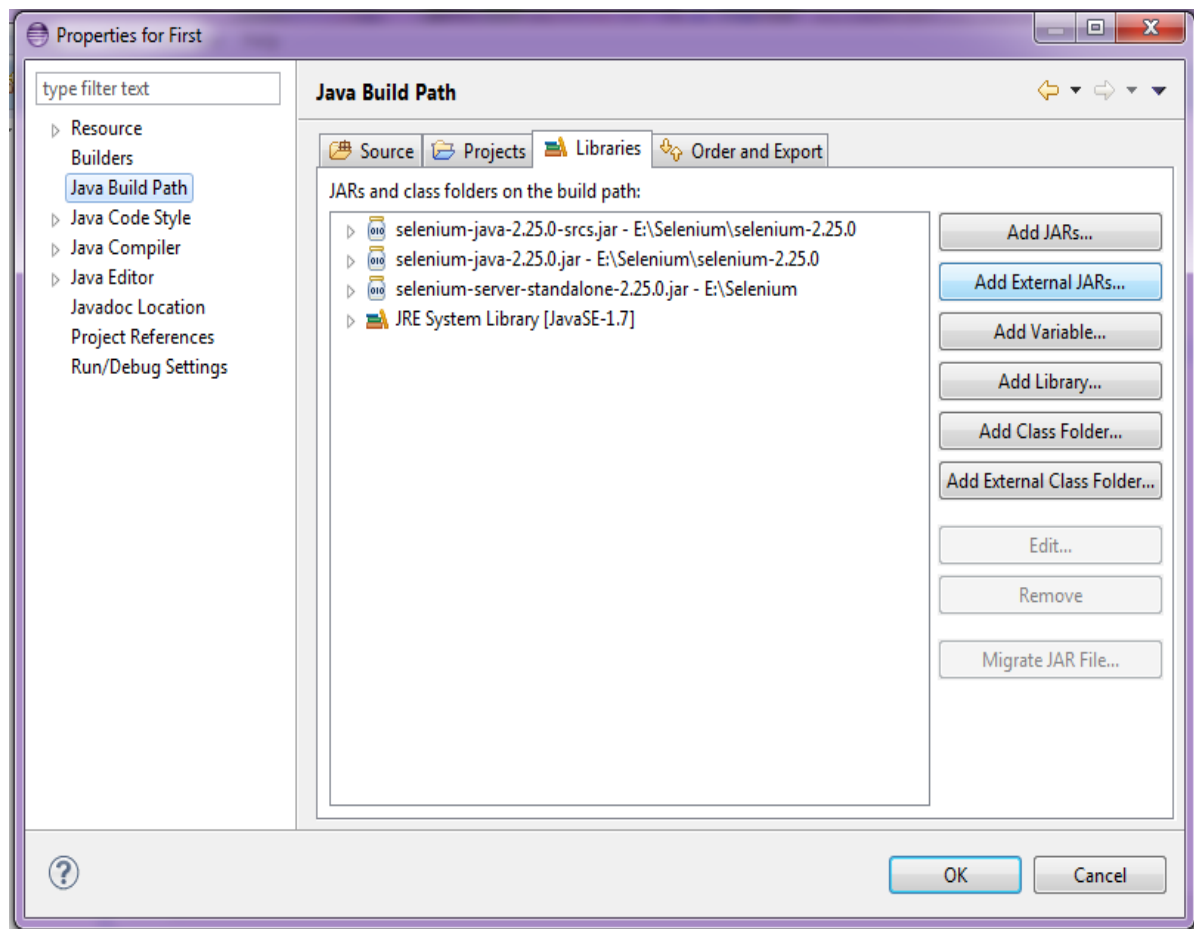
- Bước 1: Khởi động Eclipse và tạo một project mới. Việc này được thực hiện bằng cách vào File -> New -> Project -> Java -> Java Project -> Next -> Điền Project name -> Finish



Hình 3.13: Tạo project Java

Ta đặt tên project mới là First.

- Bước 2: Nhập các thư viện Selenium vào Project, nhờ đó chúng ta có thể thực thi các lệnh Selenium trên Eclipse. Các bước để làm việc này:
 - Click chuột phải vào folder “First”, chọn Properties
 - Click vào mục Java Build Path
 - Chọn tab Libraries
 - Chọn Add External JARs
 - Tìm và chọn file selenium-server-standalone-2.25.0.jar và selenium-java.jar
 - Click Open-> OK



Hình 3.14 : Thêm các file jar vào thư viện

- Bước 3: Tạo một lớp mới trong Project. Các bước thực hiện bước này: Chọn First -> New -> Class -> OK

Đặt tên lớp của là MySelenium.

Ví dụ với test case kiểm thử việc đăng nhập vào trang email của Viettel, có url là <https://mail.viettel.com.vn>, bằng trình duyệt firefox, dữ liệu đầu vào username trangnh7, password meo@Dien07, test case Selenium RC được viết bằng ngôn ngữ Java sẽ như sau:

```

import com.thoughtworks.selenium.Selenium;
import com.thoughtworks.selenium.DefaultSelenium;
// Import Selenium
public class MySelenium {
    static Selenium browser;
    public static void main(String agr[])
    { // Khai báo một biến Selenium có tên Browser
        browser=new
DefaultSelenium("localhost",4444,"*iexplore","http://mail.viettel.com.vn");
        browser.start(); // Lệnh bắt đầu Selenium
        // Các lệnh Selenese
        browser.open("/");
        browser.type("user","trangnh7");
        browser.type("password","meo@Dien07");
        browser.click("button");
        browser.waitForPageToLoad("30000");
        browser.close(); // Lệnh đóng
    }
}

```

Hình 3.15: Mẫu test case Selenium RC

Các tham số được yêu cầu với một biến có dạng DefaultSelenium bao gồm:

- Host: Thông thường giống với máy tính chạy máy khách, trong trường hợp đó localhost được thông qua. Với một số máy khách thì đây là một tham số tùy chọn.
- Port: Xác định socket TCP/IP tại điểm máy chủ nghe và đợi máy khách thiết lập kết nối. Tham số này cũng tùy chọn với một số driver của máy khách.
- Browser: Trình duyệt được sử dụng để chạy kịch bản kiểm thử.
- url: URL của ứng dụng được kiểm thử.
- Bước 4: Chạy kịch bản kiểm thử vừa tạo.
 - Để chạy kịch bản kiểm thử vừa tạo, trước tiên ta phải chạy Server của Selenium như đã nói ở phần trên.
 - Sau khi chạy Server, ta tiến hành chạy file kịch bản kiểm thử vừa tạo bằng cách chuột phải vào lớp -> Run As -> Java Application.
- Bước 5: Xem kết quả. Sau khi chạy kịch bản kiểm thử. Trang web sẽ được tải và thực thi các bước như trong kịch bản kiểm thử.

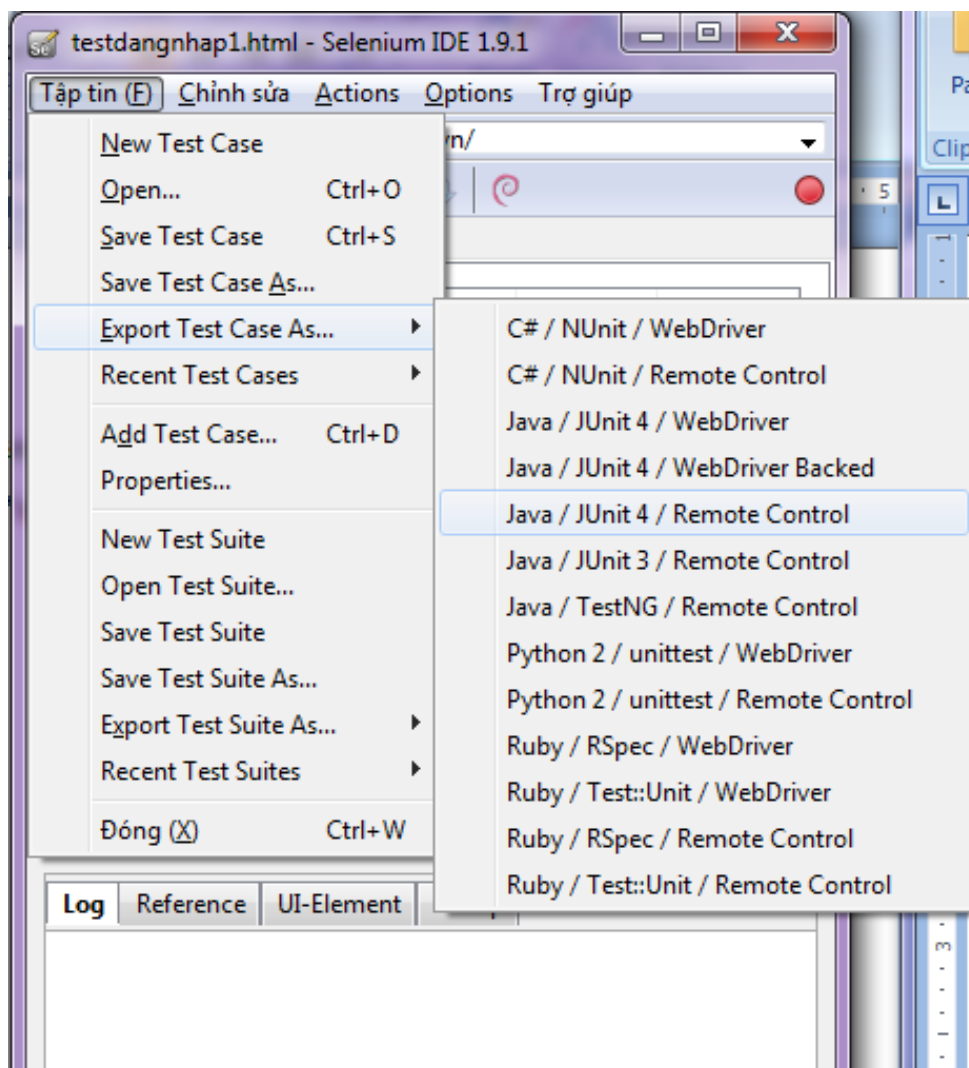
3.3.3.3. Dịch một kịch bản kiểm thử Selenium IDE thành kịch bản kiểm thử Selenium RC

Phần này của đồ án trình bày cách dịch một test case Selenium IDE sang một test case Selenium RC dưới dạng một lớp Java và dựa trên JUnit framework.

Trong phạm vi đồ án, chúng ta sẽ không đi sâu vào JUnit mà chỉ giới thiệu sơ qua về công cụ này. JUnit là một framework dùng cho kiểm thử đơn vị tự động trong Java, được phát triển bởi Erich Gamma và Kent Beck. Việc cài đặt JUnit cũng hết sức đơn giản. Chúng ta chỉ cần download JUnit từ trang <http://www.junit.org>, trong đồ án này chúng ta sẽ sử dụng JUnit 4. Sau khi download file jar về, ta add nó vào project của chúng ta giống như phần hướng dẫn add Selenium đã nêu chi tiết ở phần 3.2.

Để dịch một kịch bản kiểm thử Selenium IDE thành một kịch bản kiểm thử Selenium RC ta cần thực hiện một số bước như sau:

- Bước 1: Mở test case Selenium IDE muốn dịch. Ví dụ ta muốn dịch test case có tên testdangnhap1.html đã được tạo trước đó sang testcase Selenium RC.
Bật trình duyệt firefox -> Mở Selenium IDE -> Chọn Tập tin -> Open -> Chọn tập tin muốn dịch sang Selenium RC-> Open
- Bước 2: Export test case Selenium IDE sang test case Selenium RC dưới dạng ngôn ngữ lập trình Java và dựa trên framework JUnit 4
Chọn Tập tin -> Export Test Case As -> Java/JUnit 4/ Remote Control -> Save As lưu tên tập tin là dangnhapJUnit4.java



Hình 3.16: Export Test Case Selenium IDE sang Test Case Selenium Remote Control

Test Case được dịch có dạng như mã nguồn dưới đây:

```
import com.thoughtworks.selenium.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import java.util.regex.Pattern;

public class dangnhapJunit4 {

    public Selenium selenium;

    @Before

    public void setUp() throws Exception {

        selenium = new DefaultSelenium("localhost", 4444,
"*googlechrome", "https://mail.viettel.com.vn/");

        selenium.start();

    }

    @Test

    public void testDangnhapjunit4() throws Exception {

        selenium.open("/");

        selenium.type("name=user", "trangnh7");

        selenium.type("name=password", "meo@Dien07");

        selenium.click("id=button");

        selenium.waitForPageToLoad("30000");

    }

    @After

    public void tearDown() throws Exception {

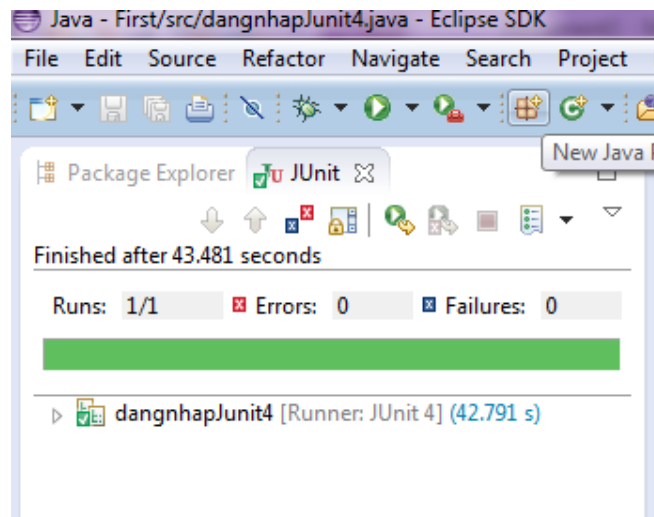
        selenium.stop();

    }

}
```

Hình 3.17: Test Case Selenium Remote Control được export từ test case Selenium IDE

- Bước 3: Thêm tập tin vào Project
- Bước 4: Chạy test case.
 - Chuột phải -> Run As -> JUnit Test
 - Test case sẽ được chạy vào kết quả chạy được hiển thị trên Junit:



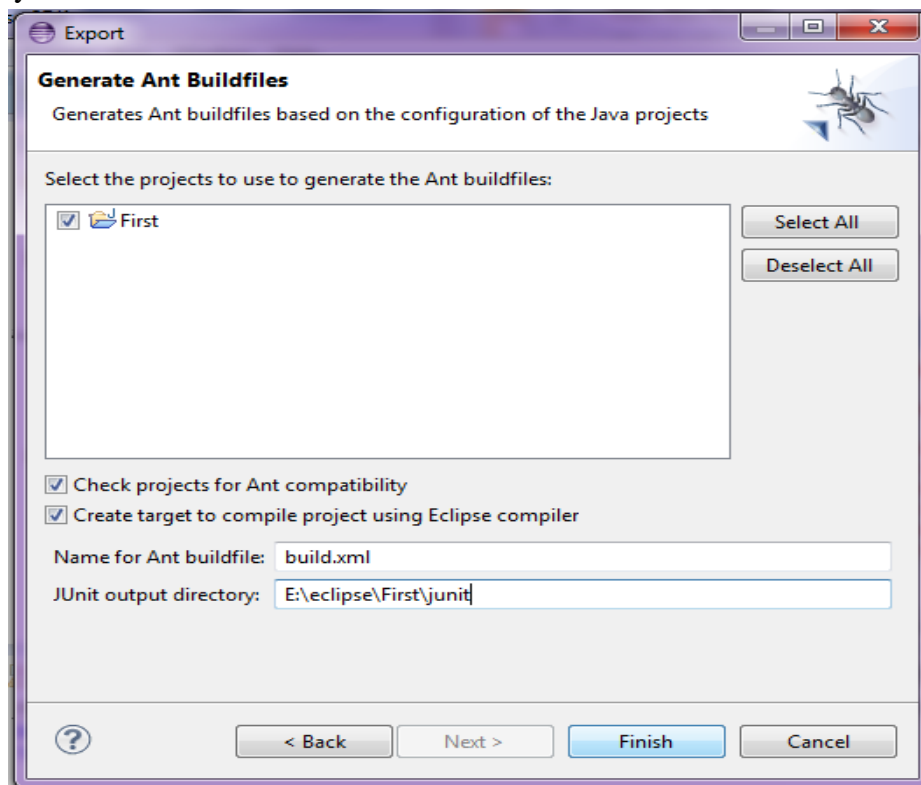
Hình 3.18 : Kết quả chạy test case trên Junit 4

3.3.3.4. Báo cáo kết quả kiểm thử

Selenium RC không có cơ chế tự báo cáo kết quả kiểm thử. Nhưng nó cho phép xây dựng báo cáo theo các đặc điểm của ngôn ngữ lập trình. Chúng ta có thể sử dụng các framework kiểm thử của các ngôn ngữ lập trình để xuất báo cáo. Trong Java có hai framework thường được sử dụng là Junit và TestNG.

Phần này của đồ án trình bày các bước để xuất một báo kiểm thử trong Eclipse dựa trên framework Junit. Dưới đây là các bước cần thực hiện:

- Bước 1: Chuột phải vào Project -> Click Export -> Chọn folder General -> Chọn Ant Buildfiles.
- Bước 2: Chọn Project, điền thông tin thư mục lưu báo cáo vào textfield JUnit output directory -> Finish

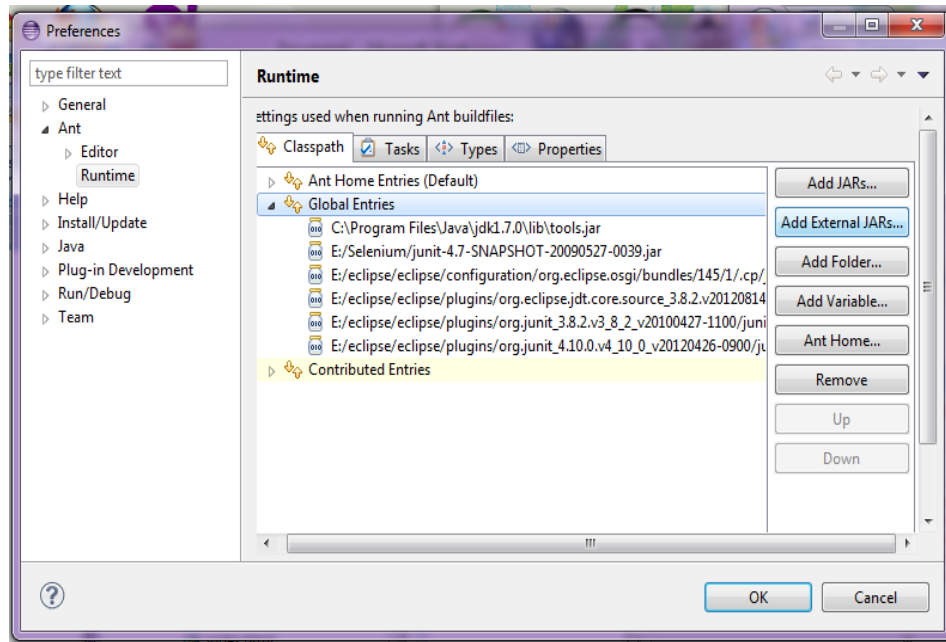


Hình 3.19: Tạo file build.xml

Sau khi click finish, file build.xml sẽ được tạo trong Project First.

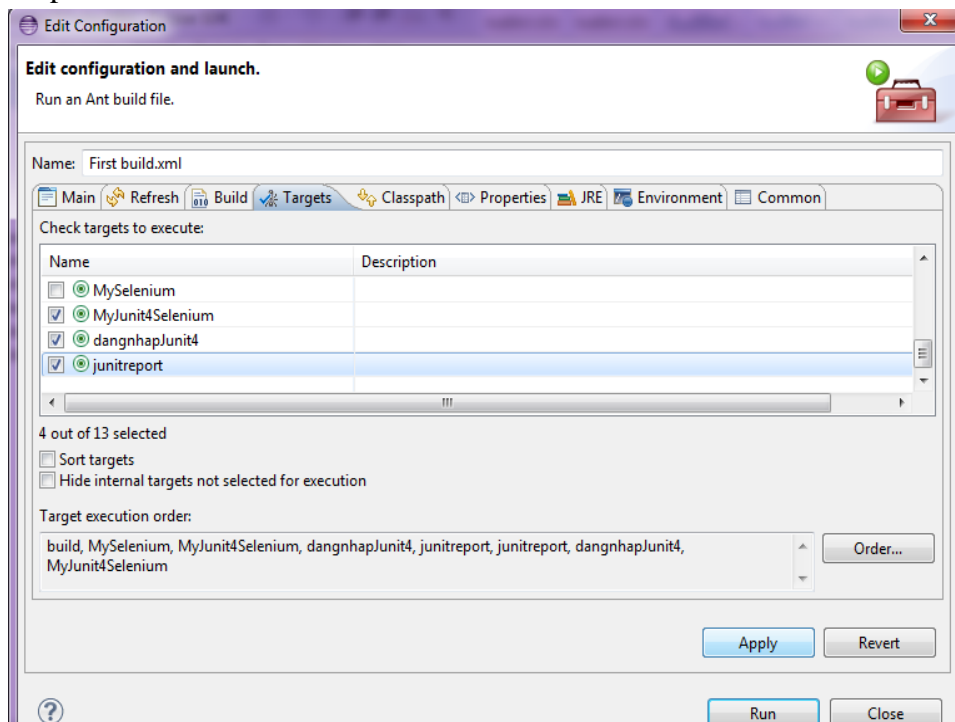
- Bước 3: Thêm file junit.jar vào Global Entries của Ant:

Click Window -> Chọn Preferences -> Expand Ant -> Click Runtime -> Chọn tab Classpath -> Expand Global Entries -> Click Add External JARs -> Vào thư mục cài đặt Eclipse E:\eclipse\ eclipse\plugins\org.junit_3.8.2.v3_8_2_v20100427-1100 chọn junit.jar -> Open



Hình 3.20: Thêm file junit.jar vào Global Entries của Ant

- Bước 4: Chuột phải vào file build.html -> Chọn Run As -> Ant Build... -> Hiện thị cửa sổ Edit Configuration and launch -> Check vào các checkbox build, các testcase muốn chạy và junitreport -> Click nút Run.



Hình 3.21: Lựa chọn các mục tiêu để thực thi file build.xml

Ở bước này, lưu ý việc sắp xếp sao cho Eclipse thực thi junitreport cuối cùng để tránh báo cáo thiếu trường hợp.

Sau khi click nút Run, chương trình sẽ thực hiện chạy các test case được chọn, và xuất báo cáo vào thư mục được nhập vào tại bước tạo file build.xml.

- Bước 5: Xem báo cáo kiểm thử. Trước khi xem báo cáo kiểm thử, ta cần refresh lại project. Vào thư mục được chọn để lưu báo cáo, mở file index. Báo cáo sẽ có dạng như hình:

Unit Test Results.

Designed for use with [JUnit](#) and [Ant](#).

Summary

Tests	Failures	Errors	Success rate	Time
2	0	1	50.00%	59.850

Note: failures are anticipated and checked for with assertions while errors are unanticipated.

Packages

Name	Tests	Errors	Failures	Time(s)	Time Stamp	Host
<none>	2	1	0	59.850	2012-11-15T08:44:14	w-PC

Hình 3.22: Mẫu báo cáo kết quả kiểm thử Selenium dựa trên JUnit

Ngoài ra báo cáo còn hỗ trợ tìm ra nguyên nhân test case bị thất bại bằng cách click vào lớp của test case.

3.4. Tổng kết chương 3

Chương 3 của đồ án đã giới thiệu được về công cụ kiểm thử phần mềm Selenium và đã nêu được những đặc điểm và cách sử dụng của hai bộ công cụ cơ bản và phổ biến là Selenium IDE và Selenium RC. Các nội dung cụ thể đã được làm rõ trong chương 3 bao gồm:

- Tổng quan về Selenium: Giới thiệu những nét chính về nguồn gốc, quá trình phát triển và các thành phần cơ bản của Selenium.
- Selenium IDE: Trình bày được phạm vi ứng dụng, cách cài đặt, cách sử dụng của Selenium IDE.
- Selenium Remote Control: Trình bày về các thành phần của Selenium RC, cách cài đặt và cách sử dụng một số chức năng của Selenium RC.

Từ những kiến thức tìm hiểu được trong nội dung chương, em rút ra được một số đánh giá với Selenium như sau:

- Những ưu điểm chung Selenium:
 - Selenium là bộ công cụ mã nguồn mở, do vậy mà nó hoàn toàn miễn phí.
 - Selenium hỗ trợ nhiều ngôn ngữ lập trình như Java, C#, Python... và kết hợp được với nhiều framework kiểm thử như JUnit, NUnit, TestNG...
 - Selenium hỗ trợ kiểm thử trên rất nhiều trình duyệt web như Firefox, Googlechrome, Internet Explore...

- Hỗ trợ gỡ lỗi
- Những nhược điểm chung của Selenium:
 - Nhược điểm lớn nhất của Selenium là nó chỉ tích hợp với các hệ thống phát triển dựa trên các nền tảng web, vì lý do đó mà nó không thể sử dụng để kiểm thử các phần mềm ứng dụng khác.
 - Selenium không thể thực hiện kiểm thử nếu bản thân nó không nhận biết được đối tượng.
 - Những hỗ trợ được cung cấp cho Selenium khá ít như việc Selenium không hỗ trợ việc xuất báo cáo kiểm thử mà ta phải làm điều đó dựa vào các framework kiểm thử khác.
 - Khó chuẩn đoán những lỗi mới phát sinh.
 - Những khó khăn trong việc cài đặt và cấu hình với người mới sử dụng.

CHƯƠNG 4: THỬ NGHIỆM

4.1. Bài toán thử nghiệm

- Vấn đề đặt ra là kiểm thử hai chức năng cơ bản cho ứng dụng web email của tập đoàn viễn thông quân đội Viettel là chức năng đăng nhập và chức năng soạn thảo và gửi email.
- Link ứng dụng: <https://mail.viettel.com.vn>
- Ứng dụng được kiểm thử trên 3 trình duyệt: Mozilla Firefox, Internet Explore, Googlechrome.
- Chức năng đăng nhập: Chức năng này là một chức năng đăng nhập thuần túy vào các ứng dụng web thông thường giống như các ứng dụng khác như yahoo, google, các forum. Các yếu tố cần kiểm tra:
 - Nếu đăng nhập đúng tên và mật khẩu thì tải đến trang chủ của ứng dụng email.
 - Nếu đăng nhập sai tên hoặc mật khẩu thì đưa ra thông báo: “Tên đăng nhập hoặc mật khẩu đăng nhập không đúng”.
 - Nếu nhập thiếu tên đăng nhập thì đưa ra thông báo: “Bạn phải nhập tên đăng nhập”.
 - Nếu nhập thiếu mật khẩu thì đưa ra thông báo: “Bạn phải nhập mật khẩu”.
- Chức năng gửi email: Ứng dụng email của Viettel có hai đặc điểm đặc trưng tạo nên sự khác biệt khi viết kịch bản kiểm thử chức năng này so với các ứng dụng email khác là:
 - Email Viettel là email nội bộ, chỉ cho phép gửi và nhận email giữa các địa chỉ nội bộ, nghĩa là các địa chỉ có phần mở rộng là @viettel.com.vn, ví dụ: trangnh7@viettel.com.vn. Các địa chỉ email có phần mở rộng khác như địa chỉ email của google hay yahoo thì ứng dụng sẽ không thể gửi email đến cũng như không thể nhận email từ các địa chỉ đó.
 - Email Viettel cũng cho phép ghi địa chỉ email mà không điền phần mở rộng, ví dụ muốn gửi thư đến địa chỉ email trangnh7@viettel.com.vn chúng ta cũng có thể cần điền trangnh7 vào địa chỉ gửi đi, ứng dụng vẫn hiểu được địa chỉ email mà người gửi muốn gửi email đến là trangnh7@viettel.com.vn.Ứng dụng email của Viettel cũng có những đặc điểm chung giống như các ứng dụng email khác, và các yếu tố chính cần kiểm tra là:
 - Kiểm tra gửi email tới một hoặc nhiều địa chỉ hợp lệ thành công
 - Kiểm tra hoạt động của các chức năng gửi Cc/ Bcc
 - Kiểm tra gửi email báo lỗi khi gửi email đến một địa chỉ không phải địa chỉ email nội bộ, đồng thời kiểm tra tại địa chỉ email nhận cũng không nhận được email đã gửi
 - Kiểm tra chức năng attach file

4.2. Sự khác nhau giữa kịch bản kiểm thử tự động và kịch bản kiểm thử thủ công

Trước khi thực hiện kiểm thử ứng dụng, cần phải nói thêm về sự khác nhau giữa một kịch bản kiểm thử thủ công và một kịch bản kiểm thử tự động.

Với kiểm thử thủ công, kịch bản kiểm thử chức năng thông thường được chia thành ba phần chính:

- Phần giao diện
- Phần chức năng
- Phần an toàn thông tin

Với kiểm thử tự động, có hai phần chính mà ta cần quan tâm là test case và dữ liệu kiểm thử. Trong đó:

- Test case: Có thể là một lớp hoặc một hàm hoặc một lớp ghi lại một chuỗi sự kiện mà ta thao tác với ứng dụng cần kiểm thử. Khác với khái niệm test case khi thực hiện kiểm thử thủ công là cứ mỗi giá trị đầu vào khác nhau thì sẽ tạo thành một testcase.
- Dữ liệu kiểm thử: Là dữ liệu nhập vào để kiểm thử.

4.3. Kịch bản kiểm thử thủ công

4.3.1. Chức năng đăng nhập

- Ở chức năng đăng nhập, ba phần chính cần kiểm tra là:
 - Giao diện: Kiểm thử các yếu tố giao diện chung như kiểm tra giao diện theo thiết kế, kiểm tra khi ấn tab, shift-tab, kiểm tra việc bị vỡ giao diện hay không, các giá trị mặc định của textbox.
 - Chức năng: Có bốn trường hợp chức năng cần chính cần kiểm thử:
 - Kiểm tra đăng nhập thành công với Tên đăng nhập/ Mật khẩu hợp lệ
 - Kiểm tra đăng nhập không thành công khi sử dụng sai Tên đăng nhập/ Mật khẩu
 - Kiểm tra thông báo khi không nhập Tên đăng nhập
 - Kiểm tra thông báo khi không nhập mật khẩu
 - Kiểm thử an toàn bảo mật: Vì chức năng đăng nhập không nhập số liệu vào cơ sở dữ liệu do vậy ta có thể bỏ qua không kiểm tra một số lỗi an toàn thông tin và chỉ cần kiểm tra một số lỗi sau:
 - Lỗi SQL Injection
 - Lỗi User Enumeration
 - Kiểm tra lỗ hổng cho phép dò đoán mật khẩu
- Kịch bản kiểm thử cụ thể được trình bày ở định dạng excel được đính kèm trong phụ lục của đồ án.

4.2.2. Chức năng soạn thảo và gửi email

Trong luồng công việc diễn ra từ khi bắt đầu soạn thảo đến lúc thực hiện gửi email thành công, chúng ta có thể phải thực hiện thao tác đính kèm file (attack file), do vậy attack file được coi là một chức năng nhỏ trong chức năng soạn thảo và gửi email và chúng ta cũng phải tiến hành kiểm thử chức năng này.

- Chức năng Attack file:
 - Giao diện: Kiểm tra giao diện chung
 - Chức năng: Chúng ta phải kiểm tra được các trường hợp chính:
 - Attack file có dung lượng hợp lệ < 10240 kb thành công
 - Kiểm tra thông báo lỗi khi Attack file có dung lượng > 10240 kb
 - Kiểm tra thực hiện chức năng của button Chọn file, Add, Cancel, Attack, Help
- Chức năng chính soạn thảo và gửi email:
 - Giao diện: Kiểm tra giao diện chung, kiểm tra các combo-box
 - Chức năng: Kịch bản kiểm thử phải đáp ứng bao quát được một số chức năng:
 - Kiểm tra soạn thảo và gửi email thành công không có file attack
 - Kiểm tra soạn thảo và gửi email thành công khi có file attack
 - Kiểm tra lựa chọn receipt request (Thông báo nhận tin, đọc tin)

- Kiểm tra việc gửi email cho nhiều người một lúc
- Kiểm tra lựa chọn gửi Cc/Bcc
- Kiểm tra được điều kiện chỉ cho phép gửi email nội bộ (các địa chỉ email của Viettel)
- Kiểm tra nhận email thông báo không gửi được email khi gửi email đến các địa chỉ email không phải email của Viettel
- Kiểm tra cảnh báo khi nhập vào địa chỉ email Viettel không có thật
- An toàn thông tin:
 - Việc kiểm tra an toàn thông tin tương tự như với chức năng đăng nhập, bổ xung thêm việc kiểm tra lỗi Session fixation và lỗi xác thực phân quyền.
- Kịch bản kiểm thử cụ thể được trình bày ở định dạng excel được đính kèm trong phụ lục của đồ án.

4.4. Kịch bản kiểm thử tự động

Do những hạn chế về kinh nghiệm và thời gian tìm hiểu tool và sự phức tạp của ứng dụng email, đồ án trình bày demo một số case cơ bản của chức năng đăng nhập bằng hai công cụ là Selenium IDE và Selenium RC và thực hiện báo cáo kết quả dựa trên framework kiểm thử JUnit.

- Đăng nhập thành công bằng firefox.

```
import com.thoughtworks.selenium.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class DangNhapFirefox {
    public Selenium selenium;
    @Before
    public void setUp() throws Exception {
        selenium = new DefaultSelenium("localhost", 4444,
        "*firefox", "https://mail.viettel.com.vn/");
        selenium.start();
    }
    @Test
    public void testDangnhapjunit4() throws Exception {
        selenium.open("/");
        selenium.type("name=user", "trangnh7");
        selenium.type("name=password", "meo@Dien07");
        selenium.click("id=button");
        selenium.waitForPageToLoad("30000");
    }
    @After
    public void tearDown() throws Exception {
        selenium.stop();
    }
}
```

Hình 4.1: Test case đăng nhập bằng Firefox

- Đăng nhập thành công bằng Internet Explore:

```
import com.thoughtworks.selenium.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class DangNhapIE {
    public Selenium selenium;

    @Before
    public void setUp() throws Exception {
        selenium = new DefaultSelenium("localhost", 4444,
        "**ie:explore", "https://mail.viettel.com.vn/");
        selenium.start();
    }

    @Test
    public void testDangnhapjunit4() throws Exception {
        selenium.open("/");
        selenium.type("name=user", "trangnh7");
        selenium.type("name=password", "meo@Dien07");
        selenium.click("id=button");
        selenium.waitForPageToLoad("30000");
    }

    @After
    public void tearDown() throws Exception {
        selenium.stop();
    }
}
```

Hình 4.2: Test case đăng nhập bằng Internet Explore

- Đăng nhập thành công Googlechrome:

```
import com.thoughtworks.selenium.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class DangNhapGC {
    public Selenium selenium;

    @Before
    public void setUp() throws Exception {
        selenium = new DefaultSelenium("localhost", 4444,
        "**googlechrome", "https://mail.viettel.com.vn/");
        selenium.start();
    }

    @Test
    public void testDangnhapjunit4() throws Exception {
        selenium.open("/");
        selenium.type("name=user", "trangnh7");
        selenium.type("name=password", "meo@Dien07");
        selenium.click("id=button");
        selenium.waitForPageToLoad("30000");
    }

    @After
    public void tearDown() throws Exception {
        selenium.stop();
    }
}
```

Hình 4.3: Test case đăng nhập bằng Googlechrome

- Báo cáo kiểm thử:

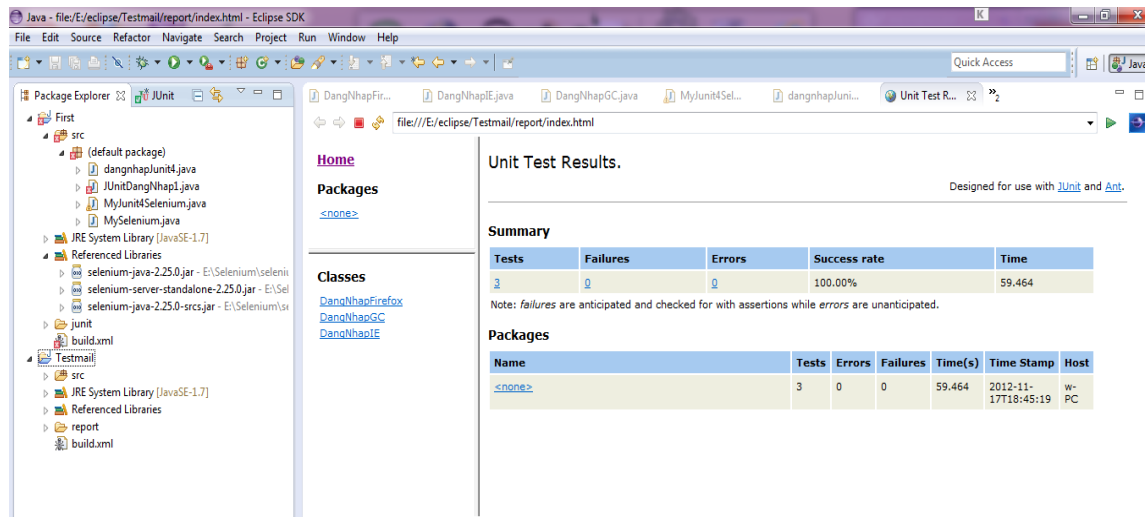
Mã nguồn sinh báo cáo:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- WARNING: Eclipse auto-generated file.
      Any modifications will be overwritten.
      To include a user specific buildfile here, simply create one in the
same
      directory with the processing instruction <?eclipse.ant.import?>
      as the first entry and export the buildfile again. -->
<project basedir="." default="build" name="Testemail">
  <property environment="env"/>
  <property name="ECLIPSE_HOME" value="../eclipse"/>
  <property
                                name="junit.output.dir"
value="E:\eclipse\Testemail\report"/>
  <property name="debuglevel" value="source,lines,vars"/>
  <property name="target" value="1.7"/>
  <property name="source" value="1.7"/>
  <path id="Testemail.classpath">
    <pathelement location="bin"/>
    <pathelement    location="../../Selenium/selenium-2.25.0/selenium-
java-2.25.0.jar"/>
    <pathelement    location="../../Selenium/selenium-server-standalone-
2.25.0.jar"/>
    <pathelement    location="../../Selenium/junit-4.7-SNAPSHOT-
20090527-0039.jar"/>
    <pathelement    location="../../Selenium/selenium-2.25.0/selenium-
java-2.25.0-srsc.jar"/>
  </path>
  <target name="init">
    <mkdir dir="bin"/>
    <copy includeemptydirs="false" todir="bin">
      <fileset dir="src">
        <exclude name="**/*.java"/>
      </fileset>
    </copy>
  </target>
  <target name="clean">
```

```
<delete dir="bin"/>
</target>
<target depends="clean" name="cleanall"/>
<target depends="build-subprojects,build-project" name="build"/>
<target name="build-subprojects"/>
<target depends="init" name="build-project">
  <echo message="${ant.project.name} : ${ant.file}"/>
  <javac debug="true" debuglevel="${debuglevel}" destdir="bin"
includeantruntime="false" source="${source}" target="${target}">
    <src path="src"/>
    <classpath refid="Testemail.classpath"/>
  </javac>
</target>
<target description="Build all projects which reference this project.
Useful to propagate changes." name="build-refprojects"/>
<target description="copy Eclipse compiler jars to ant lib directory"
name="init-eclipse-compiler">
  <copy todir="${ant.library.dir}">
    <fileset dir="${ECLIPSE_HOME}/plugins"
includes="org.eclipse.jdt.core_*.jar"/>
  </copy>
  <unzip dest="${ant.library.dir}">
    <patternset includes="jdtCompilerAdapter.jar"/>
    <fileset dir="${ECLIPSE_HOME}/plugins"
includes="org.eclipse.jdt.core_*.jar"/>
  </unzip>
</target>
<target description="compile project with Eclipse compiler"
name="build-eclipse-compiler">
  <property name="build.compiler"
value="org.eclipse.jdt.core.JDTCompilerAdapter"/>
  <antcall target="build"/>
</target>
<target name="DangNhapFirefox">
  <mkdir dir="${junit.output.dir}"/>
  <junit fork="yes" printsummary="withOutAndErr">
    <formatter type="xml"/>
    <test name="DangNhapFirefox" todir="${junit.output.dir}"/>
  </junit>
</target>
```

```
<classpath refid="Testmail.classpath"/>
</junit>
</target>
<target name="DangNhapGC">
  <mkdir dir="${junit.output.dir}"/>
  <junit fork="yes" printsummary="withOutAndErr">
    <formatter type="xml"/>
    <test name="DangNhapGC" todir="${junit.output.dir}"/>
    <classpath refid="Testmail.classpath"/>
  </junit>
</target>
<target name="DangNhapIE">
  <mkdir dir="${junit.output.dir}"/>
  <junit fork="yes" printsummary="withOutAndErr">
    <formatter type="xml"/>
    <test name="DangNhapIE" todir="${junit.output.dir}"/>
    <classpath refid="Testmail.classpath"/>
  </junit>
</target>
<target name="junitreport">
  <junitreport todir="${junit.output.dir}">
    <fileset dir="${junit.output.dir}">
      <include name="TEST-*.xml"/>
    </fileset>
    <report format="frames" todir="${junit.output.dir}"/>
  </junitreport>
</target>
</project>
```

Hình 4.4: File build.xml



Hình 4.5: Báo cáo kết quả kiểm thử

4.5. Kết quả thử nghiệm

Sau khi thực thi kiểm thử với kịch bản kiểm thử đã được lập phía trên, tác giả xin trình bày một số nhận định về kết quả thử nghiệm với 2 chức năng bao gồm chức năng đăng nhập và chức năng gửi email của ứng dụng web <https://mail.viettel.com.vn>.

4.5.1. Chức năng đăng nhập

- Tổng số trường hợp kiểm thử: 12
- Số trường hợp kiểm thử thành công: 10
- Số trường hợp kiểm thử không thành công: 2

Nhận xét:

- Xét về tổng thể giao diện:
 - Giao diện dễ hiểu, dễ sử dụng, các chức năng của phím tab, shift+tab, enter hoạt động tốt.
 - Khi phóng to, thu nhỏ không bị vỡ giao diện.
 - Tuy nhiên vẫn có lỗi câu thông báo khi thực hiện chức năng ở các trường hợp:
- Đăng nhập không nhập tên đăng nhập: Chương trình không đưa ra câu cảnh báo mà reset hai trường tên đăng nhập và mật khẩu đăng nhập về null gây khó khăn cho người sử dụng mới. Trường hợp kiểm thử này đề xuất đưa ra câu cảnh báo: "Bạn chưa nhập Tên đăng nhập".
- Đăng nhập không nhập mật khẩu đăng nhập: Chương trình không đưa ra câu cảnh báo mà reset hai trường tên đăng nhập và mật khẩu đăng nhập về null gây khó khăn cho người sử dụng mới. Trường hợp này đề xuất đưa ra câu cảnh báo: "Bạn chưa nhập mật khẩu".
- Giao diện đăng nhập bằng tiếng Việt nhưng giao diện chính của chương trình cũng như các cảnh báo đều bằng tiếng Anh, gây ra sự không nhất quán cũng như ảnh hưởng tới thẩm mỹ của chương trình. Do vậy đề xuất sửa chương trình thống nhất về ngôn ngữ.
- Về chức năng: Chức năng này thực hiện đúng và đủ các trường hợp đăng nhập.
- Về an toàn thông tin: Chức năng này đảm bảo về an toàn thông tin.

4.5.2. Chức năng gửi e mail

- Tổng số trường hợp kiểm thử: 52
- Số trường hợp kiểm thử thành công: 47
- Số trường hợp kiểm thử không thành công: 5

Nhận xét:

- Xét về tổng thể giao diện:
 - Giao diện dễ hiểu, dễ sử dụng, các chức năng của phím tab, shift+tab, enter hoạt động tốt.
 - Khi phóng to, thu nhỏ không bị vỡ giao diện.
 - Giao diện đáp ứng được thiết kế.
 - Tuy nhiên vẫn có nhược điểm là không nhất quán về ngôn ngữ, trên giao diện có cả tiếng Anh và Tiếng Việt.
- Về chức năng: Chức năng này chứa một chức năng con là chức năng attack file.
 - Chức năng attack file:
 - Các nút add, chọn tập tin, attack, cancel, help hoạt động tốt.
 - Nút remove không hoạt động do vậy đề xuất là chỉnh sửa hoạt động của nút remove
 - Chức năng soạn thảo và gửi email:
 - Các yếu tố trên màn hình hoạt động tốt.
 - Gửi email đến các địa chỉ hợp lệ thành công.
 - Các chế độ gửi email: To, Cc, Bcc hoạt động tốt.
 - Thiếu sót là không chặn việc gửi email đến các địa chỉ mail không phải là mail nội bộ. Đây là một lỗi lớn cần được khắc phục.
- Về an toàn thông tin: Hệ thống đáp ứng được vấn đề an toàn thông tin.

4.6. Tổng kết chương 4

Chương 4 của đồ án đã hoàn thành nhiệm vụ ứng dụng các kiến thức đã nghiên cứu về kiểm thử và công cụ kiểm thử tự động Selenium để kiểm thử cho hai chức năng của ứng dụng web <https://mail.viettel.com.vn> là: chức năng đăng nhập và chức năng soạn thảo và gửi email.

KẾT LUẬN

Kiểm thử phần mềm hiện nay vẫn là vấn đề hết sức quan trọng với các tổ chức phát triển phần mềm. Trong khuôn khổ đồ án của mình do thời gian và kinh nghiệm còn hạn chế nên có những phần của đồ án chưa được đào sâu nghiên cứu.

Sau một thời gian thực hiện đồ án dưới sự hướng dẫn của Thạc sĩ Đỗ Mạnh Hùng, đồ án của em đã thực hiện tốt được các mục tiêu đề ra và đạt được những kết quả như sau:

- **Kết quả đạt được:**

Trình bày đầy đủ và chính xác các vấn đề tổng quan về phần mềm, công nghệ phần mềm, lỗi phần mềm, và các vấn đề liên quan đến kiểm thử phần mềm.

- Giới thiệu công cụ kiểm thử phần mềm Selenium
- Giới thiệu Selenium IDE và Selenium Remote Control, các thao tác cơ bản để sử dụng hai công cụ này.
- Áp dụng các kiến thức đã nghiên cứu thực hiện kiểm thử hai chức năng của ứng dụng web <https://mail.viettel.com.vn> là chức năng đăng nhập và chức năng soạn thảo và gửi mail.
- Đồ án là một tài liệu xúc tích tổng hợp được các vấn đề chính của kiểm thử phần mềm, và có thể được coi là một tài liệu hướng dẫn sử dụng Selenium IDE và Selenium RC ngắn gọn rõ ràng bằng tiếng Việt để tham khảo.

- **Hạn chế:**

Mặc dù đã cố gắng hết sức trong thời gian thực hiện đề tài nhưng với kinh nghiệm còn hạn chế nên đồ án không tránh khỏi những thiếu sót:

- Chỉ đi vào nghiên cứu 2 trong 4 công cụ của bộ Selenium. Còn 2 bộ công cụ là Selenium Core và Selenium Grid chỉ giới thiệu sơ qua.
- Chưa nghiên cứu phần lập trình nâng cao với Selenium.
- Chỉ áp dụng kiểm thử được hai chức năng của ứng dụng email của tập đoàn viễn thông quân đội Viettel.

- **Hướng phát triển đề tài:**

Trong thời gian tới em sẽ tiếp tục nghiên cứu sâu hơn về các vấn đề của kiểm thử phần mềm, và đặc biệt là bộ công cụ kiểm thử ứng dụng web Selenium, để có thể vận dụng vào kiểm thử các ứng dụng lớn hơn trong thực tế công việc trong tương lai nhằm góp một phần nhỏ bé vào công cuộc chuyên nghiệp hóa kiểm thử phần mềm ở Việt Nam.

DANH MỤC TÀI LIỆU THAM KHẢO

Tiếng Anh:

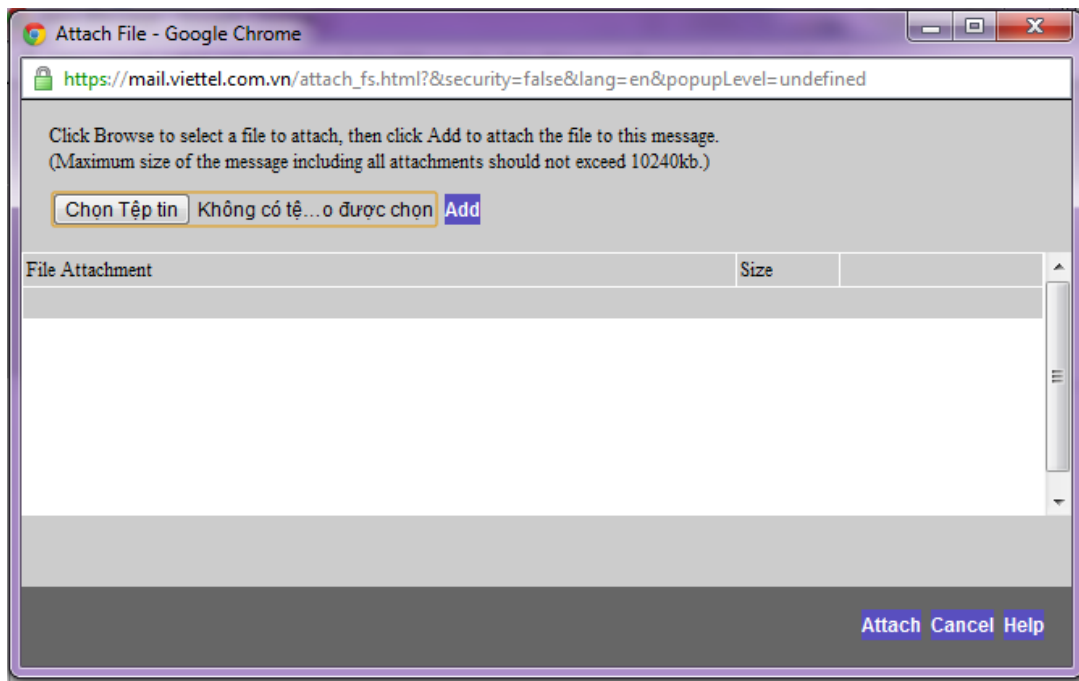
- [1] D. Burns, Selenium 1.0 Testing Tools Beginner's Guide, Birmingham-Mumbai: [PACKT] publishing.
- [2] D. Galin, Software Quality Assurance_From theory to implementation, PEARSON Education, 2004.
- [3] D. Graham, E. v. Veenendaal, R. Black and I. Evans, Foundations of software testing.
- [4] SeleniumHQ.org, "Selenium Documentation".

Website:

- [1] "<http://www.testingvn.com>," [Online].
- [2] <http://www.vietnamesetestingboard.org>. [Online].

PHỤ LỤC: KỊCH BẢN KIỂM THỬ THỦ CÔNG CHO ỨNG DỤNG THỬ NGHIỆM

Giao diện attack file



Giao diện soạn thảo email

