**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC KINH TẾ**

🙣 🕮 🙡

****

**BÁO CÁO THỰC TẬP**

**ỨNG DỤNG CÔNG CỤ SELENIUM TRONG KIỂM THỬ WEBSITE**

**CHẤM ĐIỂM RÈN LUYỆN**

SVTH: Phạm Văn Huy

Lớp: 42K21

GVHD: ThS. Cao Thị Nhâm

***Đà Nẵng, 10/2021.***

NHẬN XÉT CỦA DOANH NGHIỆP THỰC TẬP

1. ***Về ý thức và tinh thần trách nhiệm của sinh viên khi thực tập tại doanh nghiệp:***

1. ***Về năng lực và khả năng thích nghi của sinh viên với môi trường doanh nghiệp:***

1. ***Kiến nghị của Doanh nghiệp đối với Nhà trường để cải thiện chất lượng đào tạo:***

LỜI CẢM ƠN

Canh dòng: Paragraph\Line spacing 1.5)

**....**

**Họ và tên sinh viên**

LỜI CAM ĐOAN

*Tôi xin cam đoan :*

1. *Nội dung trong luận văn này là do tôi thực hiện dưới sự hướng dẫn trực tiếp của Thầy/Cô XXX.*
2. *Các tham khảo dùng trong luận văn đều được trích dẫn rõ ràng tên tác giả, tên công trình, thời gian, địa điểm công bố.*
3. *Nếu có những sao chép không hợp lệ, vi phạm, tôi xin chịu hoàn toàn trách nhiệm.*

*Sinh viên thực hiện*

**Họ và Tên**

MỤC LỤC

>> Chú ý: Trong mục lục chỉ nên liệt kê Heading 1, 2, 3

[CHƯƠNG 1: Cơ sở lý thuyết 9](#_Toc84095363)

[1.1. Khái niệm về kiểm thử phần mềm 10](#_Toc84095364)

[~~1.2. Các mô hình ứng dụng 10~~](#_Toc84095365)

[~~1.3. Vòng đời phát triển phần mềm 10~~](#_Toc84095366)

[~~1.4. Software Requirements Specifications – SRS 10~~](#_Toc84095367)

[~~1.5. Tài liệu thiết kế hệ thống 10~~](#_Toc84095368)

[~~1.6. Tại sao cần kiểm thử và kiểm thử để làm gì 10~~](#_Toc84095369)

[1.7. Quy trình kiểm thử cơ bản 10](#_Toc84095370)

[1.7.1. Giới thiệu về quy trình kiểm thử 10](#_Toc84095371)

[1.7.2. Mô hình phát triển phần mềm 10](#_Toc84095372)

[1.7.3. Các mức kiểm thử 10](#_Toc84095373)

[1.7.4. Các loại hình kiểm thử 10](#_Toc84095374)

[1.7.5. Chiến lược kiểm thử - Testing Strategy 10](#_Toc84095375)

[1.7.6. Kế hoạch kiểm thử - Test Plan 10](#_Toc84095376)

[1.7.7. Kịch bản kiểm thử - Test Scenario 10](#_Toc84095377)

[1.7.8. Trường hợp kiểm thử - Test Case 10](#_Toc84095378)

[1.7.9. Kiểm thử theo mô hình Agile 10](#_Toc84095379)

[1.8. Kỹ thuật thiết kế Test 10](#_Toc84095380)

[1.8.1. Giới thiệu về thiết kế Test 10](#_Toc84095381)

[1.8.2. Quy trình phát triển 10](#_Toc84095382)

[1.8.3. Các kỹ thuật thiết kế 10](#_Toc84095383)

[1.8.4. Kỹ thuật Black Box và White Box 10](#_Toc84095384)

[1.8.5. Chọn kỹ thuật để thiết kế Test 10](#_Toc84095385)

[1.9. Thực hiện kiểm thử và báo cáo 10](#_Toc84095386)

[1.9.1. Xây dựng dữ liệu kiểm thử - Test Data 10](#_Toc84095387)

[1.9.2. Thực hiện kiểm thử - Test Execution 10](#_Toc84095388)

[1.9.3. Báo cáo và ghi lại lịch sử kiểm thử - Test Reports/Test Logs 10](#_Toc84095389)

[1.9.4. Quy trình báo cáo lỗi – Bug and Defect Reporting Process 10](#_Toc84095390)

[1.9.5. Báo cáo, quản lý và giám sát lỗi 10](#_Toc84095391)

[1.9.6. Phân tích lỗi và rút ra bài học kinh nghiệm 11](#_Toc84095392)

[1.10. Kiểm thử trên các loại ứng dụng khác nhau 11](#_Toc84095393)

[1.10.1. Ứng dụng Web 11](#_Toc84095394)

[1.10.2. Ứng dụng Desktop 11](#_Toc84095395)

[1.10.3. Ứng dụng di động 11](#_Toc84095396)

[1.10.4. Web Services 11](#_Toc84095397)

[1.11. Công tác quản lý 11](#_Toc84095398)

[1.11.1. Kiểm soát rủi ro trong kiểm thử 11](#_Toc84095399)

[1.11.2. Công tác tổ chức 11](#_Toc84095400)

[1.11.3. Chiến lược kiểm thử 11](#_Toc84095401)

[1.11.4. Ước lượng và lập kế hoạch 11](#_Toc84095402)

[1.11.5. Giám sát và kiểm soát quá trình test 11](#_Toc84095403)

[1.12. Các công cụ hỗ trợ 11](#_Toc84095404)

[1.12.1. Công cụ quản lý chất lượng 11](#_Toc84095405)

[1.12.2. Công cụ quản lý Test Cases – Test Link 11](#_Toc84095406)

[1.12.3. Công cụ quản lý Bug – Jira Bug Tracker 11](#_Toc84095407)

[1.13. KẾT CHƯƠNG 11](#_Toc84095408)

[CHƯƠNG 2: Kiểm thử tự động 12](#_Toc84095409)

[2.1. Cơ bản về kiểm thử tự động 12](#_Toc84095410)

[2.1.1. Giới thiệu về kiểm thử tự động 12](#_Toc84095411)

[2.1.2. Tại sao cần kiểm thử tự động 12](#_Toc84095412)

[2.1.3. Quy trình kiểm thử tự động 12](#_Toc84095413)

[2.1.4. Giới thiệu về Selenium 12](#_Toc84095414)

[2.2. Java trong Kiểm thử tự động 12](#_Toc84095415)

[2.2.1. Giới thiệu về Java 12](#_Toc84095416)

[2.2.2. Kiểm soát lỗi trong Java 12](#_Toc84095417)

[2.3. Sử dụng Selenium Webdriver để kiểm thử tự động 12](#_Toc84095418)

[2.3.1. Giới thiệu về Selenium Webdriver 12](#_Toc84095419)

[2.3.2. Tạo ra Selenium Webdriver Script 12](#_Toc84095420)

[2.3.3. Kiểm thử tham số - Parameterize Test 12](#_Toc84095421)

[2.3.4. Selenium Method 12](#_Toc84095422)

[2.3.5. Verification Point 12](#_Toc84095423)

[2.3.6. Sử dụng Custom Function 12](#_Toc84095424)

[2.3.7. Làm việc với Data Driven Test 12](#_Toc84095425)

[2.3.8. Debug Scripts 12](#_Toc84095426)

[2.3.9. Kiểm soát lỗi ngoại lệ - Exception Handling 12](#_Toc84095427)

[2.3.10. Thực hiện báo cáo 12](#_Toc84095428)

[2.3.11. Thực hiện Test và kết hợp với Jenkins 12](#_Toc84095429)

[2.3.12. Automation Test Framework 12](#_Toc84095430)

[2.4. KẾT CHƯƠNG 12](#_Toc84095431)

[CHƯƠNG 3: Thực hiện dự án 13](#_Toc84095432)

[3.1. Loading 13](#_Toc84095433)

[CHƯƠNG 4: Kết luận và hướng phát triển 13](#_Toc84095434)

DANH SÁCH HÌNH ẢNH

DANH SÁCH TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **Từ viết tắt** | **Diễn giải** |
|  |  |

MỞ ĐẦU

**1.**  **Tổng quan về đề tài**

**2. Mục đích và ý nghĩa của đề tài**

***2.1. Mục đích***

***2.2. Ý nghĩa***

**3. Phương pháp thực hiện**

**3. Bố cục**

# Chương 1

## Giới thiệu về kiểm thử phần mềm

### Khái niệm kiểm thử phần mềm

Kiểm thử phần mềm là quá trình thực thi một chương trình với mục đích là tìm ra lỗi, đảm bảo những lỗi này được tìm ra sớm nhất có thể và đảm bảo được sửa chữa.

Vậy, kiểm thử phần mềm làm gì?

* Kiểm tra hoạt động của sản phẩm.
* Kiểm tra những yêu cầu của sản phẩm được đáp ứng chưa.
* Xem thử những chức năng được kiểm tra đã hoàn thành hay chưa, có hoạt động giống như người dùng và phía khách hàng mong muốn chưa.
* Tìm và ngăn chặn defects (những khiếm khuyết) và lỗ hổng của sản phẩm.
* Cung cấp đủ những thông tin cần thiết để những người có quyền quyết định có thể dựa vào để đưa ra quyết định.
* Giảm thiểu rủi ro chất lượng sản phẩm không đảm bảo.

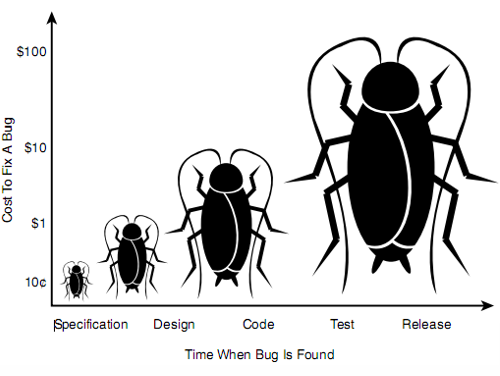
### Kiểm thử và Debug

Đầu tiên, Bug là kết quả thực hiện chức năng cụ thể không đúng như mong đợi.

Bug xảy ra vì nhiều lý do khác nhau:

* Vì tài liệu mô tả sản phẩm: chưa được viết, chưa mô tả rõ ràng, bị thay đổi giữa chừng, hay là chưa trao đổi rõ ràng với Dev. Team.
* Vì phía thiết kế sản phẩm: bị hối thúc, chưa rõ ràng chi tiết, chưa được trao đổi rõ ràng với nhau.
* Vì bên lập trình: áp lực về thời gian thực hiện sản phẩm, độ phức tạp mà sản phẩm yêu cầu, tài liệu nghèo nàn thông tin hoặc chưa rõ ràng, hay những lỗi cơ bản nhỏ nhặt nhất.
* Ngoài ra còn những lý do khác xung quanh gây ảnh hưởng đến nhân sự, cơ sở vật chất cũng gây ra bug cho sản phẩm.

Chi phí để fix bug tỉ lệ thuận với thời gian phát hiện ra bug trên tổng tiến trình phát triển sản phẩm. Cụ thể là, phát hiện bug càng sớm, chi phí fix bug càng thấp và ngược lại, càng về sau, chi phí để fix bug càng tăng cao.

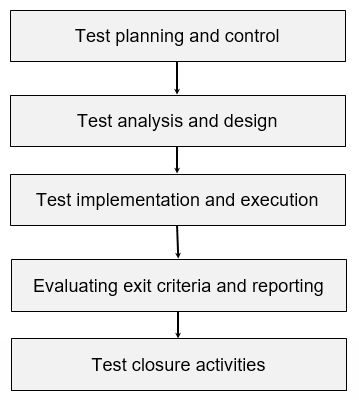


Hình 1. Mối liên hệ giữa chi phí fix bug và thời gian phát hiện ra bug

Vậy thì, để fix bug ta cần phải kiểm thử để phát hiện ra bug. Khi phát hiện được bug sẽ tiến hành debug – cố gắng fix được bug đã phát hiện ra.

## Quy trình kiểm thử cơ bản

### Giới thiệu về quy trình kiểm thử



Hình 2. Quy trình kiểm thử

**Test planning and control – Lập kế hoạch và giám sát việc kiểm thử**. Bước này tiến hành lập kế hoạch để xác định đối tượng kiểm thử và kiểm thử như thế nào. Lên sơ đồ cụ thể các công việc cần thực hiện và chi tiết cách thực hiện như thế nào. Đồng thời xác định tiêu chuẩn cụ thể cho việc hoàn thành kiểm thử.

**Test analysis and design – Phân tích và thiết kế kiểm thử**. Bước này chuyển đổi các mục tiêu tổng quát thành các điều kiện kiểm thử và thiết kế kiểm thử cụ thể. Đây là bước kết nối giữa bước lên kế hoạch kiểm thử và bước thực hiện kiểm thử. Phân tích cụ thể thời gian biểu cho việc kiểm thử, nhân lực, đối tượng được kiểm thử, kết quả kiểm thử mong đợi ra sao và điều kiện môi trường cho việc kiểm thử cụ thể như thế nào.

**Test implementation and execution – Thực hiện và thực thi kiểm thử**. Bước này bao gồm các công việc thực hiện các trường hợp kiểm thử, thu thập kết quả kiểm thử, ghi nhận và báo cáo sự khác biệt giữa kết quả mong đợi và kết quả thực tế khi kiểm thử hoàn tất.

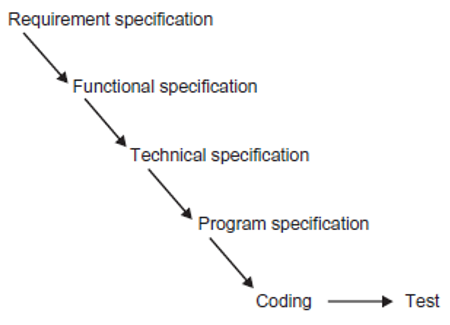
**Evaluating exit criteria and reporting - Đánh giá tiêu chuẩn kết thúc và báo cáo**. Tiêu chuẩn kết thúc được xác định ở bước lập kế hoạch và giám sát việc kiểm thử. Ở bước này sẽ tiến hành kiểm tra xem đã đáp ứng được các tiêu chí đề ra hay chưa, có cần thực hiện kiểm thử bổ sung hay không. Và viêt báo cáo kiểm thử.

**Test closure activities – Các hoạt động kết thúc kiểm thử**. Bước này tập trung vào việc đảm bảo hoàn tất các công việc, khách hàng nhận được sản phẩm như dự kiến, các sự cố đã được giải quyết, bàn giao sản phẩm cho bộ phận bảo trì và hỗ trợ khách hàng, đánh giá cách kiểm thử và rút ra kinh nghiệm, bài học cho các dự án về sau.

### Mô hình phát triển phần mềm

Ngành công nghiệp phát triển phần mềm phát triển lớn mạnh trên toàn thế giới, đồng thời, các mô hình phát triển phần mềm cũng được phát triển theo. Có nhiều mô hình để doanh nghiệp có thể áp dụng để phát triển sản phẩm của mình. Trong đó có thể kể đến ba mô hình khá phổ biến là: Waterfall, chữ V, Agile-Scrum.

**Mô hình Waterfall**



Hình 3. Mô hình Waterfall

Đây được coi như là mô hình phát triển phần mêm đầu tiên được sử dụng. Mô hình áp dụng tuần tự các giai đoạn. Đầu ra của giai đoạn trước là đầu vào của giai đoạn sau, không có sự tương tác qua lại. Thường được áp dụng cho dự án có thời gian ngắn.

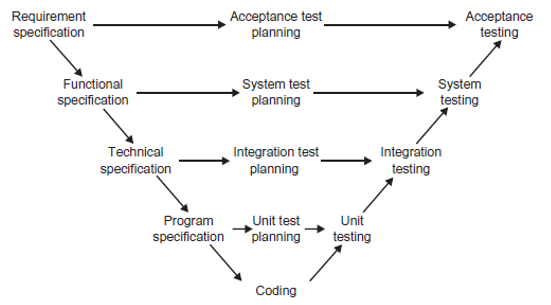
**Điểm mạnh:**

* Dễ hiểu, dễ sử dụng, dễ quản lý
* Có thể áp dụng với thành viên không có nhiều kinh nghiệm
* Các cột mốc của dự án cụ thể rõ ràng
* Yêu cầu của sản phẩm được xác định chi tiết rõ ràng ngay từ đầu
* Tập trung vào chất lượng sản phẩm hơn là các yếu tố khác.

**Điểm yếu:**

* Các yêu cầu của sản phẩm cần phải xác định chi tiết rõ ràng ngay từ ban đầu
* Đầu ra của các giai đoạn nhất định cố định, thiếu tính linh động
* Các đơn vị được kết nối với nhau cùng một lúc ở giai đoạn cuối cùng
* Rất ít hoặc hầu như không có cơ hội cho khách hàng có thể theo dõi sản phẩm cho đến giai đoạn cuối cùng

**Mô hình chữ V**



Hình 4. Mô hình chữ V

Mô hình chữ V có thể coi như là mô hình mở rộng của mô hình Waterfall, dựa trên sự kết hợp với kiểm thử cho từng giai đoạn phát triển tương ứng. Đây là một mô hình có tính kỷ luật cao và giai đoạn tiếp theo chỉ bắt đầu sau khi hoàn thành giai đoạn trước. Với mô hình chữ V thì công việc test được tham gia ngay từ đầu. Thường được áp dụng cho dự án nhỏ, đã nắm rõ ràng các yêu cầu ngay từ ban đầu.

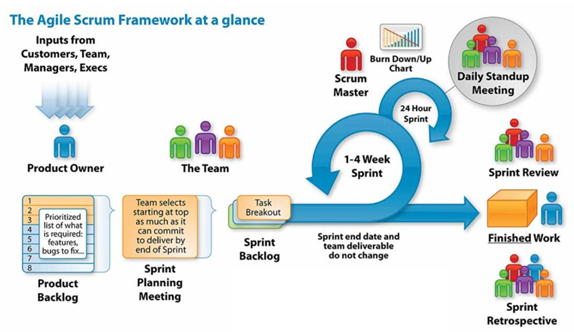
**Điểm mạnh:**

* Định hình được sản phẩm rõ ràng ngay từ ban đầu
* Dễ sử dụng, dễ quản lý
* PM có thể theo dõi tiến độ dự án dễ dàng theo các cột mốc
* Đầu ra mỗi giai đoạn đều khả dụng để kiểm thử

**Điểm yếu:**

* Không dễ dàng trong việc kiểm soát các sự kiện đồng thời xảy ra
* Không có giới hạn về rủi ro, cho nên không thật sự có trọng tâm giữa các tầng
* Khó khăn trong việc xử lý khi có thay đổi bất ngờ trong yêu cầu về sản phẩm

**Mô hình Agile – Scrum**



Hình 5. Mô hình Agile – Scrum

Đây là mô hình được sử dụng rộng rãi trên thế giới. Là mô hình có quy trình nhẹ và gọn cho mục đích phát triển nhanh sản phẩm. Thường được sử dụng để quản lý phát triển phần mềm và sản phẩm phức tạp, sử dụng phương pháp có tính lặp lại và tăng dần. Tăng năng suất và giảm thời gian so với mô hình Waterfall cổ điển. Mô hình này được áp dụng cho những sản phẩm có các yêu cầu thay đổi liên tục, tạo ra sản phẩm đáp ứng được các yêu cầu liên quan đến mục tiêu kinh doanh. Có thể tiến hành kiểm thử sớm.

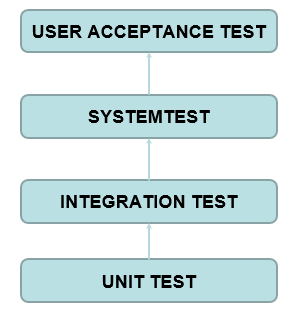
**Điểm mạnh:**

* Dễ quản lý, dễ sử dụng
* Phát hiện ra lỗi sớm
* Có thể kiểm soát được sản phẩm có yêu cầu chưa rõ ràng hoặc chưa đầy đủ từ đầu
* Một người có thể thực hiện các vai trò các nhau như dev., test,…
* Tăng cường tinh thần làm việc nhóm của team

**Điểm yếu**

* Yêu cầu nhân lực có kinh nghiệm
* Chuyển giao công nghệ cho các thành viên trong nhóm có thể gặp khó khăn vì thiếu tài liệu từ đầu
* Phụ thuộc nhiều vào sự tương tác với khách hàng, thời gian thực hiện có thể kéo dài so với dự kiến
* Không cố định trong việc xác định ngân sách và thời gian thực hiện dự án

### Các mức kiểm thử - Test Levels



Hình 6. Các mức kiểm thử từ dưới lên trên

**Unit Test:** Là kiểm thử đơn vị ở mức độ nhỏ nhất. Mục đích là kiểm thử từng đơn vị, từng thành phần cấu tạo nên phần mềm, hệ thống có hoạt động đúng như mong muốn hay không. Ví dụ như kiểm thử xem lệnh vừa code có chạy đúng hay không. Thường thì Unit Test sẽ do dev. thực hiện luôn đồng thời trong quá trình code.

**Integration Test**: Ở mức độ kiểm thử tích hợp này, các đơn vị, các thành phần cấu tạo được kết hợp với nhau để tiến hành kiểm thử. Mục đích là để phát hiện ra lỗi khi kết hợp các đơn vị, các thành phần với nhau.

**System Test**: Với mức độ kiểm thử hệ thống, một sản phẩm được tích hợp hoàn chỉnh sẽ được kiểm thử. Mục đích là để đánh giá sự tương thích của hệ thống với các yêu cầu cụ thể.

**User Acceptance Test**: Kiểm thử khách hàng nghiệm thu. Mức độ này tiến hành kiểm thử trải nghiệm sử dụng dưới góc độ khách hàng thực tế. Mục đích là đánh giá sự tương thích của hệ thống với các yêu cầu kinh doanh và tiến hành đánh giá xem sản phẩm đã phù hợp để bàn giao cho khách hàng hay chưa.

### Các loại hình kiểm thử - Test Types

Có thể phân biệt bốn loại hình kiểm thử, bao gồm:

**Functional Testing – Kiểm thử chức năng**: Loại hình này sẽ kiểm thử những chức năng của sản phẩm như các luồng quy trình, các mô hình chuyển đổi trạng thái, các mô hình nguy cơ bảo mật, các thông số kỹ thuật thuần ngôn ngữ,…

Ví dụ: Tìm kiếm chuyến bay trên website, kiểm tra độ bảo mật của hệ thống, kiểm tra tương tác của hệ thống với những thành phần cụ thể,…

**Non-functional Testing – Kiểm thử phi chức năng:** Loại hình này tiến hành kiểm thử hiệu suất, khả năng sử dụng sản phẩm, độ chịu tải và khả năng xử lý của sản phẩm (Load test và Stress test), kiểm thử mức độ sử dụng có dễ dàng hay không.

Ví dụ: Giao diện có dễ nhìn hay không, cùng lúc website có thể tải được bao nhiêu luồng truy cập, có thể xử lý được bao nhiêu lệnh cùng lúc,…

**Structual Testing – Kiểm thử cấu trúc:** Loại hình kiểm thử này với mục đích đo lường bao nhiêu dòng code cụ thể đã được kiểm thử, những control follow, những giải thuật có hoạt động ổn hay chưa.

**Testing related to changes – Kiểm thử liên quan đến những thay đổi:** Sau khi phát hiện ra những lỗ hổng, bug, khiếm khuyết của sản phẩm và tiến hành sửa chữa, sản phẩm cần được **tái kiểm thử** (retest) lại để đảm bảo rằng những vấn đề ấy đã được xử lý hoàn tất. Ngoài ra, đối với sản phẩm có sự thay đổi trong chức năng, sản phẩm cũng cần được tái kiểm thử lại, đây được gọi là **kiểm thử hồi quy**, kiểm thử hồi quy cũng cần thực hiện khi thay đổi môi trường sử dụng sản phẩm bị thay đổi.

## Kỹ thuật thiết kế Test

### Giới thiệu về thiết kế Test

Thiết kế test bao gồm ba bước chính:

Bước 1: Xác định Test condition – Điều kiện test: là một trong những đặc trưng cụ thể của phần mềm mà ta có thể kiểm tra với một hoặc nhiều tests.

Bước 2: Chỉ ra rõ Test cases - Trường hợp kiểm thử: bao gồm các điều kiện tiên quyết (nếu có), đầu vào, hành động kiểm thử, kết quả mong muốn và kết quả thực tế dựa vào Test condition.

Bước 3: Chỉ ra rõ Test procedures – Thủ tục kiểm thử: là chuỗi các Test cases theo trình tự thực hiện, bao gồm cả các hành động được yêu cầu để thiết lập điều kiện tiên quyết ban đầu và bất kỳ hoạt động kết thúc nào sau khi thực hiện tests.

### Trường hợp kiểm thử - Test case

Test case là tập hợp các điều kiện và các biến mà theo đó tester sẽ xác định xem một phần mềm, ứng dụng hay một trong những chức năng của nó có hoạt động đúng như thiết lập hay không.

Có hai loại Test cases, bao gồm:

* Positive test cases – Test to pass: được thiết kế để trả về kết quả đúng như yêu cầu.
* Negative test cases – Test to fail: được thiết kế để trả về kết quả không đúng như yêu cầu.

Cấu trúc cơ bản của một Test case bao gồm:

* Summary: Tổng quan
* Preconditions: Điều kiện tiên quyết
* Steps: Các bước thực thi
* Expected result: Kết quả mong muốn

Ngoài ra, một Test case có thể bao gồm những thông tin khác như: related requirements, author, priority, notes,…

Ví dụ một Test case đối với trường hợp gửi gmail mà không điền địa chỉ gmail người nhận:

* Summary: Verify that user cannot send an email without recipient
* Precondition: N/A
* Steps:

1. Login Gmail with valid username/password
2. Click Compose button
3. Leave Recipient blank
4. Enter email subject, body massage
5. Click Send button

* Expected result: “You cannot send an email without entering recipient.” message displays.

### Các kỹ thuật thiết kế test

**Kỹ thuật Black box**: Dựa trên phân tích của tài liệu cơ bản để thực hiện kiểm thử, bao gồm khía cạnh chức năng và khía cạnh phi chức năng.

**Kỹ thuật White box**: Dựa trên kiểm thử trực tiếp cấu trúc của chức năng, hệ thống.

**Kỹ thuật Ad-hoc**: dựa trên cơ sở kinh nghiệm: Dựa trên kinh nghiệm kiểm thử của tester với hệ thống tương tự hoặc kinh nghiệm kiểm thử tổng quan.

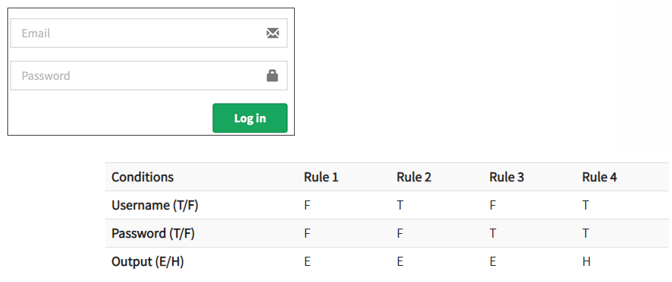
### Kỹ thuật Black Box

Bao gồm năm kỹ thuật cơ bản.

**Phân vùng tương đương**: Tận dụng các thuộc tính của phân vùng tương đương để giảm thiểu số lượng Test cases cần thực hiện. Vì tất cả các giá trị trong cùng một phân vùng có nguyên lý hoạt động giống nhau nên ta chỉ cần lấy một giá trị làm đại diện.

**Phân tích giá trị biên**: Từ lỗi mà các lập trình viên hay mắc phải, lỗi có xu hướng thường xảy ra ở các vùng biên. Nên ta cần phân tích giá trị biên để không bị thiếu sót Test cases.

**Kiểm thử bảng quyết định**: là bảng gồm tổ hợp các đầu vào khác nhau, được sử dụng để kiểm tra hành vi của hệ thống. Thường là tổ hợp các quy tắc, điều kiện hoặc là các trường hợp.



Hình 7. Ví dụ bảng quyết định của một form đăng nhập

**Kiểm thử chuyển đổi trạng thái**: liên quan đến việc phân tích các mối quan hệ giữa trạng thái và các sự kiện, hành động xảy ra tạo ra sự chuyển đổi từ trạng thái này sang trạng thái khác. Là kỹ thuật mà đầu vào tạo ra sự chuyển đổi trạng thái, thường được dùng để phân tích hành vi thể hiện của sản phẩm khi thay đổi một chuỗi các giá trị đầu vào

**Kiểm thử trường hợp sử dụng – Use case testing**: Thường được sử dụng với sản phẩm có giá trị đầu vào hữu hạn. Khi mà tester thực hiện kiểm thử một chuỗi sự kiện xảy ra với sản phẩm, tức là cho phép tester thực hiện kiểm thử với một chuỗi giá trị đầu vào hữu hạn. Ngoài ra, kỹ thuật còn được sử dụng khi hệ thống phụ thuộc vào các sự kiện, các giá trị trong quá khứ.

“Use case” là một cách để chỉ định chức năng dưới dạng các kịch bản kinh doanh hoặc các luồng quy trình. Họ nắm bắt các tương tác riêng lẻ giữa các "tác nhân" và hệ thống. Tác nhân là một người dùng hoặc một hệ thống khác sử dụng các chức năng của hệ thống hiện tại.

“Use case” mô tả về việc sử dụng hệ thống cụ thể của một tác nhân hoặc người dùng. Nó được sử dụng rộng rãi trong việc phát triển các thử nghiệm ở mức kiểm thử hệ thống hoặc mức độ UAT.

Use Case Testing được định nghĩa là một kỹ thuật kiểm thử phần mềm, giúp xác định các trường hợp kiểm thử bao quát toàn bộ hệ thống, trên cơ sở theo từng giao dịch từ đầu đến cuối.

### Kỹ thuật White Box

Bao gồm ba kỹ thuật:

**Statement testing and coverage:** Là kiểm thử mục đich kiểm tra các câu lệnh lập trình.

**Paths through a program:** Giúp tester xác định số lượng Test cases dựa vào các luồng thông tin, lối đi logic có thể thực hiện xuyên suốt chương trình.

**Decision testing and coverage:** Nhằm đảm bảo rằng các quyết định trong chương trình đều được thực hiện đầy đủ. Ta cần thực hiện kiểm thử khi điều kiện đúng và điều kiện sai, đảm bảo rằng không bị sót.

### Ad-hoc testing

Không có thông số cụ thể nào để quyết định. Dựa trên kinh nghiệm cá nhân tester, kỹ thuật này gồm:

* Dự đoán lỗi
* Test khám phá (test tự do)

## Thực hiện kiểm thử và báo cáo

### Thực hiện kiểm thử - Test Execution

Thực hiện các test case như thiết kế và mức độ ưu tiên đã đưa ra trên môi trường đã xác định.

So sánh với kết quả mong đợi sau báo cáo các bug xảy ra và theo dõi trạng thái của lỗi đến khi được sửa thành công.

Thực hiện re-test để verify các bug đã được fix và regression test khi có sự thay đổi liên quan.

Trong quá trình thực hiện kiểm thử, tester cũng có thể hỗ trợ, đề xuất cho cả đội dự án để có giải pháp hợp lý và kết hợp công việc hiệu quả.

Đo và phân tích tiến độ: kiểm thử viên cũng cần kiểm soát chặt chẽ tiến độ công việc bằng cách so sánh tiến độ thực tế với kế hoạch, nếu chậm cần phải điều chỉnh sao cho kịp tiến độ dự án, nếu nhanh cũng cần điều chỉnh vì có thể test lead lên kế hoạch chưa sát với thực tế dự án. Từ đó có thể sửa chữa test plan cần điều chỉnh để phù hợp với tiến độ dự án đưa ra.

Report thường xuyên cho PM và khách hàng về tình hình thực hiện dự án: Cung cấp thông tin trong quá trình kiểm thử đã làm được những chức năng nào, còn chức năng nào, hoàn thành được bao nhiều phần trăm công việc, báo cáo các trường hợp phát sinh sớm, tránh ảnh hưởng tiến độ công việc của cả ngày.

### Báo cáo và quản lý lỗi

Khi tiến hành thực hiện các Test cases, nếu như phát hiện ra error, tester cần thực hiện các công việc sau:

**Tái hiện lại lỗi**. Lỗi thường sẽ dễ dàng có thể tái hiện lại, tuy nhiên, một số lỗi rất hiếm khi xảy ra, tester đôi khi cũng gặp khó khăn trong việc tái hiện lại lỗi vì một số lý do khác nhau như: lỗi phần cứng, lỗi phần mềm, cài đặt, thời gian, vân vân. Tuy nhiên, cần phải cố gắng tái hiện lại lỗi hết sức có thể.

**Phân tích lỗi**. Đầu tiên cần đảm bảo rằng lỗi có thể được tái hiện lại.

* Mô tả trạng thái chương trình, sản phẩm, phần mềm một cách rõ ràng cụ thể nhất.
* Xác định cụ thể từng bước một để khiến cho lỗi xảy ra
* Kiểm tra lại các bước đã xác định xem có tái hiện lại lỗi một cách chính xác hay không

**Báo cáo lỗi.** Một báo cáo lỗi là hữu ích khi báo cáo trình bày rõ ràng, dễ hiểu, đánh số cụ thể, có thể tái hiện lại và không bao gồm các quan điểm cá nhân.

**Nội dung báo cáo lỗi bao gồm:**

* Tổng quan
* Mô tả lỗi
* Các bước để tái hiện lại lỗi

Ngoài ra, một báo cáo lỗi có thể bao gồm các thông tin khác như: tần số xuất hiện, độ ưu tiên, phân loại độ nghiêm trọng,…

## Kiểm thử trên các loại ứng dụng khác nhau

### Ứng dụng Web

Kiểm thử Web là phương pháp kiểm thử phần mềm dùng kiểm thử website và ứng dụng web để tìm ra các lỗi tiềm ẩn.

Ứng dụng nền tảng Web cần phải kiểm thử toàn bộ trước khi đưa lên môi trường trực tuyến.

Thực hiện kiểm thử Web, tổ chức có thể đảm bảo rằng hệ thống nền tảng web hoạt động bình thường và được người dùng theo thời gian thực chấp nhận.

Thiết kế giao diện và Chức năng chính là mấu chốt của kiểm thử website.

Kiểm thử Web thực hiện các công việc cụ thể sau:

**Kiểm thử chức năng:** Tiến hành kiểm thử tất cả các đường liên kết trên các trang web, kiểm tra liên kết cơ sở dữ liệu, biểu mẫu thu thập thông tin người dùng, bộ nhớ đệm,…

* Các liên kết nội bộ, các liên kết ngoại vi, liên kết dịch vụ mail và link hỏng.
* Biểu mẫu thu thập thông tin người dùng có lỗi, thiếu thông tin yêu cầu, chính tả, quy định nhập liệu,…
* Toàn vẹn cơ sở dữ liệu

**Kiểm thử trải nghiệm người dùng:** Đo lường tương tác giữa con người – hệ thống

* Dễ dàng sử dụng hay không
* Tính điều hướng
* Mức độ hài lòng chủ quan người dùng
* Cái nhìn tổng quan

**Kiểm thử giao diện người dùng:**

* Máy chủ web và giao diện máy chủ ứng dụng
* Máy chủ ứng dụng và giao diện máy chủ Cơ sở dữ liệu.

**Kiểm thử mức độ tương thích:**

* Tương thích trình duyệt
* Tương thích hệ điều hành
* Tương thích duyệt web trên di động
* Tương thích với máy in

**Kiểm thử mức độ đáp ứng**

* Load test
* Stress test

**Kiểm thử mức độ bảo mật**

* Tính xác thưc
* Tính ủy quyền
* Mật khẩu
* Nâng cấp, cập nhật hệ thống
* Bảo mật mạng

### Ứng dụng di động

### Web Services

Dịch vụ web là một phương tiện được chuẩn hóa để truyền thông tin liên lạc giữa các ứng dụng máy khách và máy chủ trên World Wide Web.

Dịch vụ web là một mô-đun phần mềm được thiết kế để thực hiện một số tác vụ nhất định.

Dịch vụ web còn gọi là API Web, chúng hỗ trợ cho ứng dụng di động hoặc SOA (Kiến trúc hướng phần mềm).

Dịch vụ Web có thể giao tiếp với hệ thống khác bằng ngôn ngữ XML hoặc JSON

Vì dịch vụ Web không có một giao diện cụ thể, ta cần sử dụng công cụ Postman để tiến hành kiểm thử dịch vụ Web.

# TỔNG QUAN VỀ CÔNG CỤ SELENIUM

## Cơ bản về kiểm thử tự động

### Giới thiệu về kiểm thử tự động

Kiểm thử tự động là xử lý một cách tự động các bước thực hiện các testcase, kiểm thử tự động bằng một công cụ nhằm rút ngắn thời gian kiểm thử.

Kiểm thử tự động là một kỹ thuật tự động trong đó người kiểm thử tự viết các tập lệnh và sử dụng phần mềm phù hợp để kiểm thử phần mềm. Nó về cơ bản là một quá trình tự động hóa của một quy trình kiểm thử thủ công. Giống như kiểm thử hồi quy, kiểm thử tự động cũng được sử dụng để kiểm thử ứng dụng theo quan điểm tải, hiệu năng và ứng suất.

Kiểm thử tự động với các mục đích:

* Giảm bớt công sức và thời gian thực hiện quá trình kiểm thử
* Tăng độ tin cậy.
* Giảm sự nhàm chán cho con người
* Rèn luyện kỹ năng lập trình cho kiểm thử viên
* Giảm chi phí cho tổng quá trình kiểm thử

Khi nào cần kiểm thử tự động:

* Không đủ tài nguyên: Khi số lượng TestCase quá nhiều mà kiểm thử viên không thể hoàn tất trong thời gian cụ thể.
* Kiểm tra hồi quy: Nâng cấp phần mềm, kiểm tra lại các tính năng đã chạy tốt và những tính năng đã sửa. Tuy nhiên, việc này khó đảm bảo về mặt thời gian.
* Kiểm tra khả năng vận hành phần mềm trong môi trường đặc biệt (Đo tốc độ trung bình xử lý một yêu cầu của Web server, xác định cấu hình máy thấp nhất mà phần mềm vẫn có thể hoạt động tốt).

### Sự cần thiết của kiểm thử tự động

Kiểm thử tự động giúp giảm chi phí kiểm thử bằng cách hỗ trợ quá trình kiểm thử thông qua các công cụ phần mềm.

Kiểm thử tự động hay sử dụng phần mềm để kiểm thử với các ưu điểm:

* Có thể thực hiện các kiểm thử một cách liên tục, lặp lại và giảm chi phí cho nhân lực kiểm thử.
* Luôn đảm bảo hoạt động theo một kịch bản duy nhất – không bị ảnh hưởng như với kiểm thử viên.

### Quy trình kiểm thử tự động

Quy trình kiểm thử tự động bao gồm: tester sử dụng các kịch bản tự động (automation scripts) và thực thi các script để chạy ứng dụng với sự giúp sức của các automation tool. Một khi script đã sẵn sàng thì việc thực thi kiểm thử có thể diễn ra nhanh chóng và hiệu quả.

Các hoạt động của kiểm thử tự động:

* Phân tích yêu cầu/Xác định môi trường/công cụ
* Xác định tiêu chí đầu ra
* Lên kế hoạch và kiểm soát
* Thiết lập môi trường kiểm thử
* Triển khai thiết kế kiểm thử
* Thực thi kiểm thử
* Phân tích, báo cáo

### Giới thiệu về Selenium

Selenium (thường được viết tắt là SE) là một công cụ kiểm thử phần mềm tự động, được phát triển bởi ThoughtWorks từ năm 2004 với tên ban đầu là JavaScriptTestRunner. Đến năm 2007, tác giả Jason Huggins rời ThoughtWorks và gia nhập Selenium team, một phần của Google và phát triển thành Selenium như hiện nay.

Selenium là một công cụ hỗ trợ kiểm tra tự động cho các ứng dụng chạy trên nền web. Selenium hỗ trợ kiểm tra hầu hết trên các trình duyệt phổ biến hiện nay như Firefox, Internet Explorer, Safari,…cũng như các hệ điều hành chủ yếu như Windows, Linux, Mac,…

Selenium hỗ trợ một số lớn các ngôn ngữ lập trình như C#, Java, Perl, PHP, Python, Ruby,…

Selenium có thể kết hợp thêm một số công cụ khác như Bromien, Junit nhưng với người dùng thông thường chỉ cần chạy tự động mà không cần cài thêm các công cụ hỗ trợ.

Ưu điểm:

* Mã nguồn mở: Đây là điểm mạnh nhất của Selenium khi so sánh với các test tool khác. Selenium không mất phí bản quyền hay thời hạn sử dụng.
* Cộng đồng hỗ trợ khá mạnh mẽ, đặc biệt là nhà phát triển là Google. Đơn giản, dễ cài đặt, dễ làm việc.
* Selenium hỗ trợ nhiều ngôn ngữ lập trình và hỗ trợ chạy trên nhiều OS khác nhau. Dễ dàng điều chỉnh thông qua các plugin.
* Hỗ trợ các tệp tin selenium user-extensions.js.
* Tự động hoàn chỉnh cho tất cả các lệnh Selenium thường gặp.

Hạn chế :

* Selenium chỉ hỗ trợ những ứng dụng web
* Các ứng dụng trên moblie không thể sử dụng Selenium
* Selenium là một tool miễn phí, do đó không có hỗ trợ từ nhà cung cấp mặc dù có thể tìm thấy giúp đỡ từ đông đảo cộng đồng sử dụng Selenium
* Người sử dụng cần có kiến thức về ngôn ngữ lập trình

## Java trong Kiểm thử tự động

### Giới thiệu về Java

### Kiểm soát lỗi trong Java

## Sử dụng Selenium Webdriver để kiểm thử tự động

### Giới thiệu về Selenium Webdriver

Selenium WebDriver được thiết kế để mang đến một giao diện lập trình đơn giản và giải quyết một số hạn chế trong Selenium- RC API (Một phần khác của Selenium có thể coi là phiên bản tiền thân của selenium WebDriver).

Selenium WebDriver cho phép sử dụng một trong số các ngôn ngữ lập trình như HTML, Java, .Net, Perl, Ruby… để tạo kịch bản test (Test Case) kết hợp với sử dụng các điều kiện, vòng lặp… khiến cho test script trở nên chính xác hơn.

Selenium WebDriver được phát triển tốt hơn để hỗ trợ cho các trang web động (Những trang web mà phần tử trong nó có thể thay đổi ngay cả khi trang đó không được tải lại).

Mục đích của WebDriver là hỗ trợ cho các vấn đề về kiểm thử web-app hiện nay.

### Tạo ra Selenium Webdriver Script

Ví dụ cụ thể, tạo Script cho website Gmail.com

Scenario/Kịch bản:

* Khởi chạy trình duyệt và mở “Gmail.com”.
* Xác nhận tiêu đề của trang và in ra kết quả.
* Nhập tên người dùng và mật khẩu.
* Nhấp vào nút Đăng nhập.
* Đóng trình duyệt web.

Bước 1: Tạo 1 java class mới trong project tương ứng.

Bước 2: Viết Script

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

public class Gmail\_Login

{

/\*\*

\* @param args

\*/

public static void main(String[] args)

{

// khai báo biến và object (đối tượng)

WebDriver driver = new FirefoxDriver();

// tạo mới object "driver" là 1 driver của Firefox

String appUrl = "https://accounts.google.com";

// gắn value cho biến appUrl

// khởi chạy trình duyệt firefox và mở ứng dụng tương ứng với appUrl, ở đây là accounts.google.com

driver.get(appUrl);

// maximize (mở rộng tối đa) cửa sổ trình duyệt

driver.manage().window().maximize();

// khai báo và khởi tạo biến để lưu trữ tiêu đề dự kiến của trang web

String expectedTitle = " Sign in - Google Accounts ";

// fetch (tìm) tiêu đề thực của trang web và lưu nó vào biến actualTitle

String actualTitle = driver.getTitle();

// so sánh tiêu đề dự kiến (expected) với tiêu đề thực tế của trang và in kết quả

if (expectedTitle.equals(actualTitle))

{

System.out.println("Verification Successful - Hiển thị đúng tiêu đề trang.");

}

else

{

System.out.println("Verification Failed - Hiển thị sai tiêu đề trang");

}

// nhập giá trị username hợp lệ trong email textbox

WebElement username = driver.findElement(By.id("Email"));

username.clear();

username.sendKeys("TestSelenium");

// nhập giá trị password hợp lệ trong password textbox

WebElement password = driver.findElement(By.id("Passwd"));

password.clear();

password.sendKeys("password123");

// click button Đăng nhập

WebElement SignInButton = driver.findElement(By.id("signIn"));

SignInButton.click();

// đóng trình duyệt

driver.close();

System.out.println("Test script executed successfully.");

// kết thúc chương trình

System.exit(0);

}

}

### Kiểm thử tham số - Parameterize Test

Một parameterized test là một phương thức test bình thường, được thực hiện lặp đi lặp lại bằng cách sử dụng các tham số test khác nhau. Nó giúp chúng ta tiết kiệm thời gian trong việc thực hiện nhiều lần cùng một phương thức test với các loại đầu vào khác nhau để kiểm tra các kết quả khác nhau của chức năng.

Chúng ta có thể truyền đối số vào các phương thức unit test thông qua các cách sau:

* Truyền thông qua đối số của constructor (Constructor Injection).
* Đánh dấu annotation @Parameter trên các filed (Fields injection).

### Selenium Method

* Browser Methods perform actions on a browser.
* WebElement Methods perform actions on WebElements.
* Navigation Methods load a web page, refresh a web page, or move backwards and forwards in our browser’s history.
* Wait Methods pause between execution statements.
* Switch Methods switch to alerts, windows, and frames. An alert is also known as a pop-up.

### Verification Point

A verification point is a specialized step that compares two values and reports the result. A verification point compares the actuals from the test run, with the expected results in the test case.

You use a checkpoint to:

* Verify the state of an object
* Confirm that an application performs as expected

A verification point checks whether an application responds appropriately when a user performs tasks correctly while testing application. A verification point ensures that a user is barred from performing certain tasks and confirms that invalid or incomplete data are flagged with appropriate messages.

# THỰC HIỆN KIỂM THỬ WEBSITE CHẤM ĐIỂM RÈN LUYỆN

## Giới thiệu về Website chấm điểm rèn luyện

Mỗi năm học, mỗi học kỳ trôi qua, cứ đến cuối học kỳ sinh viên trường DUE lại rục rịch chấm điểm rèn luyện để đánh giá hoạt động đóng góp cá nhân cho các hoạt động của trường.

Về cơ bản, điểm rèn luyện được chấm theo từng cá nhân trên từng tờ giấy A4, sau đó gộp lại nộp về lớp trưởng. Tiếp theo, lớp trưởng tiến hành đánh giá và sửa điểm của các thành viên trong lớp rồi nộp lại cho GVCN. Ở bước cuối này, GVCN sẽ đánh giá lại lần cuối và sửa điểm rèn luyện của sinh viên trong lớp nếu có sai sót.

Nhìn chung, quy trình chấm điểm rèn luyện hiện tại còn nhiều điểm cần khắc phục như:

* Tốn nhiều thời gian
* Tốn giấy A4 để in ấn – tiêu tốn tài nguyên
* Dễ xảy ra thất lạc, sai sót khiến sinh viên cần phải thực hiện lại cả quá trình chấm điểm

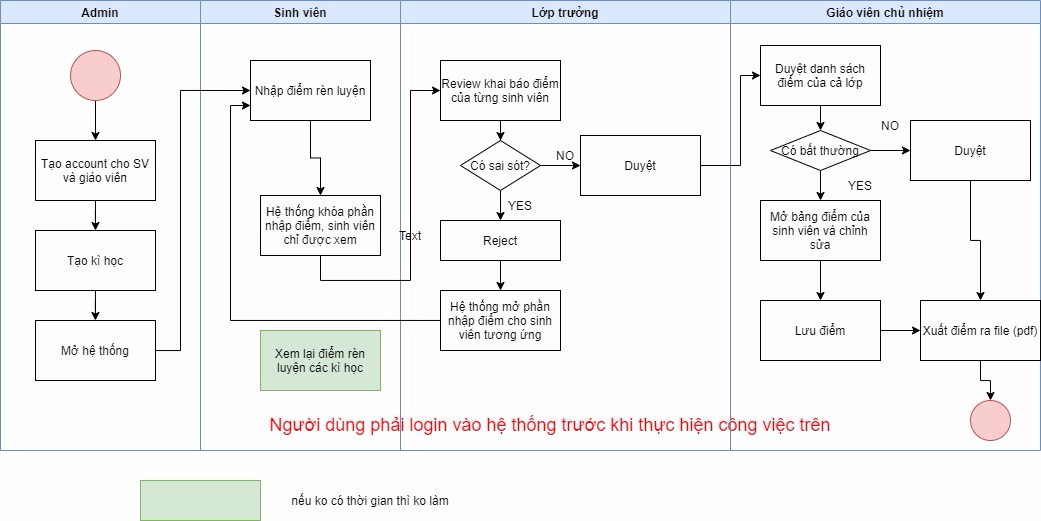
Nhằm khắc phục những điểm trên, kết hợp với cơ sở công nghệ của thời đại 4.0, dự án “Website quản lý điểm rèn luyện” được lên ý tưởng để giải quyết nhanh, gọn những điểm được nêu ở trên.

Trước hết, “Website quản lý điểm rèn luyện” sẽ có những ưu điểm cụ thể như sau:

* Nhỏ, gọn, dễ dàng truy cập và xử lý, giúp ích trong quá trình quản lý điểm rèn luyện
* Tiết kiệm rất nhiều thời gian cho cả sinh viên và GVCN
* Tiết kiệm được giấy A4 – tiết kiệm tài nguyên
* Đảm bảo điểm được chấm và quản lý đầy đủ, không bị thất lạc hay sai sót không thể sửa chữa kịp thời

## Phân tích Website chấm điểm rèn luyện

Cơ bản về ý tưởng hoạt động của website được mô tả như sau:



Hình 8. Sơ đồ hoạt động của Website chấm điểm rèn luyện

***Đặc tả yêu cầu của dự án****: Website sẽ phân quyền tương ứng cho từng vai trò cụ thể của user. Mỗi user có quyền và chức năng riêng biệt. Cụ thể:*

**Admin:**

* Khởi tạo database lưu điểm rèn luyện
* Tạo tài khoản và mật khẩu đăng nhập cho sinh viên và GVCN
* Tạo năm học và kỳ học tương ứng
* Tạo form để sinh viên thực hiện chấm điểm cá nhân
* Mở hệ thống chấm điểm cho phép sinh viên và GVCN truy cập để thực hiện việc chấm điểm
* Khóa hệ thống khi kết thúc thời gian chấm điểm
* Khóa hệ thống để thực hiện bảo trì và nâng cấp nếu cần

**Sinh viên thường**

* Đăng nhập hệ thống bằng tài khoản được cấp bởi admin
* Sinh viên có thể tự đổi mật khẩu tài khoản cá nhân
* Giao diện trang sinh viên hiện rõ ràng tên sinh viên, lớp sinh hoạt, mã sinh viên, vai trò của tài khoản
* Chỉ có thể truy cập link chấm điểm khi admin mở hệ thống chấm điểm
* Có thể tùy ý xem điểm rèn luyện cá nhân trong các học kỳ trước đó
* Khi sinh viên hoàn thành chấm điểm, hệ thống sẽ hỏi xác nhận trước khi lưu điểm chuyển tiếp đến lớp trưởng
* Có thông báo trạng thái điểm được duyệt hoàn tất hay chưa
* Có thể thực hiện chấm lại khi điểm bị từ chối từ tài khoản sinh viên lớp trưởng hoặc tài khoản GVCN

**Sinh viên là lớp trưởng**

* Tương tự như sinh viên thường, sinh viên là lớp trưởng sẽ có thêm một số chức năng
* Tài khoản được phân quyền lớp trưởng
* Theo dõi trực quan tiến độ chấm điểm của các sinh viên trong lớp
* Phê duyệt điểm của các sinh viên đã tự chấm điểm xong
* Điểm được duyệt sẽ lưu lại để chuyển tiếp đến GVCN
* Điểm bị từ chối sẽ trả về sinh viên để thực hiện tự chấm lại

**GVCN**

* Đăng nhập hệ thống bằng tài khoản được cấp bởi admin
* Tài khoản được phân quyền GVCN
* Hiện danh sách lớp mà GV đang chủ nhiệm
* Hiện danh sách sinh viên trong lớp tương ứng và trạng thái điểm của cả lớp (lớp trưởng đã chuyển cho GVCN hay còn đang chấm)

Tiến hành duyệt điểm:

* Hiển thị trực quan tên sinh viên, mã sinh viên và điểm đã chấm
* GVCN đánh giá điểm đã đạt hay chưa. Nếu đạt thì lưu lại. Nếu chưa đạt thì mở chi tiết điểm đã chấm để tiến hành sửa, sau đó lưu lại.
* Điểm sau khi được lưu lại hết, GVCN vẫn có thể xem và sửa lại nếu cần thiết.
* Có thể xuất file điểm .pdf

## Kế hoạch kiểm thử

## Thiết kế test case

## Thực hiện kiểm thử

## Kết quả kiểm thử

# Kết luận và hướng phát triển

## KẾT QUẢ ĐẠT ĐƯỢC

Về mặt lý thuyết:

Về mặt thực tiễn:

Kết quả được thể hiện.

Tuy nhiên, còn tồn tại các vấn đề:

## KIẾN NGHỊ VÀ HƯỚNG PHÁT TRIỂN

TÀI LIỆU THAM KHẢO

**Tiếng Việt**

**Tiếng Anh**

**Internet**

PHỤ LỤC