|  |
| --- |
| **BỘ GIÁO DỤC VÀ ĐÀO TẠO**  NGUYỄN QUỐC BẢO <TÊN ĐỀ TÀI> 2020  **TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HƯNG YÊN**  **NGUYỄN QUỐC BẢO**  **XÂY DỰNG FRAMEWORK KIỂM THỬ HƯỚNG LAI ÁP DỤNG KIỂM THỬ WEBSITE BÁN SÁCH FAIRY TAIL**  **ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**  **HƯNG YÊN - 2021** |

|  |
| --- |
| **BỘ GIÁO DỤC VÀ ĐÀO TẠO**  **TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HƯNG YÊN**  **NGUYỄN QUỐC BẢO**  **XÂY DỰNG FRAMEWORK KIỂM THỬ HƯỚNG LAI ÁP DỤNG KIỂM THỬ WEBSITE BÁN SÁCH FAIRY TAIL**  NGÀNH: CÔNG NGHỆ THÔNG TIN  CHUYÊN NGÀNH: KỸ THUẬT PHẦN MỀM  **ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**  **NGƯỜI HƯỚNG DẪN**  **ĐỖ THỊ THU TRANG**  **HƯNG YÊN - 2021** |

**NHẬN XÉT**

**Nhận xét của giảng viên hướng dẫn:**

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

**GIẢNG VIÊN HƯỚNG DẪN**

*(Ký và ghi rõ họ tên)*

**LỜI CAM ĐOAN**

Em xin cam đoan đồ án tốt nghiệp “Xây dựng framework kiểm thử hướng lai áp dụng kiểm thử website bán sách Fairy Tail” là công trình nghiên cứu của bản thân. Những phần sử dụng tài liệu tham khảo trong đồ án đã được nêu rõ trong phần tài liệu tham khảo. Các số liệu, kết quả trình bày trong đồ án là hoàn toàn trung thực, nếu sai em xin chịu hoàn toàn trách nhiệm và chịu mọi kỷ luật của Bộ môn và Nhà trường đề ra.

*Hưng Yên, ngày … tháng … năm…..*

Sinh viên

…………………………………..

MỤC LỤC

[MỤC LỤC 5](#_Toc65246139)

[DANH SÁCH CÁC THUẬT NGỮ 7](#_Toc65246140)

[DANH SÁCH BẢNG BIỂU 8](#_Toc65246141)

[DANH SÁCH HÌNH VẼ 9](#_Toc65246142)

[CHƯƠNG 1: MỞ ĐẦU 10](#_Toc65246143)

[1.1 Lý do chọn đồ án 10](#_Toc65246144)

[1.2 Mục tiêu của đồ án 10](#_Toc65246145)

[1.2.1 Mục tiêu tổng quát 10](#_Toc65246146)

[1.2.2 Mục tiêu cụ thể 10](#_Toc65246147)

[1.3 Giới hạn và phạm vi của đồ án 10](#_Toc65246148)

[1.3.1 Đối tượng nghiên cứu 10](#_Toc65246149)

[1.3.2 Phạm vi nghiên cứu 11](#_Toc65246150)

[1.4 Nội dung thực hiện 11](#_Toc65246151)

[1.5 Phương pháp tiếp cận 11](#_Toc65246152)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 12](#_Toc65246153)

[2.1 Tổng quan về kiểm thử phần mềm 12](#_Toc65246154)

[2.2 Kiểm thử tự động phần mềm 12](#_Toc65246155)

[2.3 Kiểm thử chức năng/Kiểm thử hiệu năng/Kiểm thử bảo mật… 12](#_Toc65246156)

[2.4 <Hướng kiểm thử tự động>/Kiểm thử API/… 12](#_Toc65246157)

[2.5 <Công nghệ/công cụ áp dụng> 12](#_Toc65246158)

[CHƯƠNG 3: ĐẶC TẢ HỆ THỐNG PHẦN MỀM 13](#_Toc65246159)

[3.1 Giới thiệu về hệ thống phần mềm 13](#_Toc65246160)

[3.2 Các yêu cầu chức năng 13](#_Toc65246161)

[3.3 Các yêu cầu phi chức năng 13](#_Toc65246162)

[CHƯƠNG 4: TRIỂN KHAI KIỂM THỬ TỰ ĐỘNG 14](#_Toc65246163)

[4.1 Thiết kế các yêu cầu kiểm thử 14](#_Toc65246164)

[4.2 Xây dựng ca kiểm thử 14](#_Toc65246165)

[4.3 Xây dựng dữ liệu kiểm thử 14](#_Toc65246166)

[4.4 Xây dựng kịch bản kiểm thử (Test scripts)/ Phương pháp xây dựng Framework 14](#_Toc65246167)

[4.5 Thực thi và báo cáo kiểm thử 14](#_Toc65246168)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 15](#_Toc65246169)

[Kết quả đạt được 15](#_Toc65246170)

[Hạn chế của đề tài 15](#_Toc65246171)

[Hướng phát triển của đề tài 15](#_Toc65246172)

[TÀI LIỆU THAM KHẢO 16](#_Toc65246173)

[PHỤ LỤC 17](#_Toc65246174)

DANH SÁCH CÁC THUẬT NGỮ

|  |  |  |
| --- | --- | --- |
| Từ viết tắt | Từ đầy đủ | Giải thích |
| SRS | Software Requirement Specification | Đặc tả yêu cầu phần mềm |
| ………… | ……………………………… | ………………………………… |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

DANH SÁCH BẢNG BIỂU

[Bảng 1‑1 Danh mục ví dụ phạm vi nghiên cứu 11](#_Toc65180383)

DANH SÁCH HÌNH VẼ

[Hình 3‑1 Hình ví dụ 1 13](file:///E:\Bo%20mon\Bieu%20mau%20do%20an%20mon%20hoc\Bieu%20mau%20do%20an%20tot%20nghiep\ISO-IT13-M09.Mau%20Cuon%20do%20an%20tot%20nghiep%20Web24-2-2021.docx#_Toc65180467)

# MỞ ĐẦU

1.1 Lý do chọn đồ án

Chúng ta đã và đang chứng kiến sự tăng trưởng đáng kinh ngạc của nghành công nghiệp phần mềm trong vài thập kỉ qua. Nếu như trước đây, phần mềm máy tính chỉ được sử dụng để tính toán khoa học kỹ thuật và xử lý dữ liệu thì ngày nay nó đã được ứng dụng vào mọi mặt đời sống con người. Có thể nói, công nghiệp phần mềm đã len lỏi đến từng ngóc nghách nhỏ nhất của tất cả các hoạt động xã hội, đỏi hỏi chất lượng phần mềm ngày một nâng cao hơn. Đồng nghĩa với việc cần phải kiểm thử phần mềm chặt chẽ để có thể đảm bảo chất lượng của phần mềm.

Kiểm thử phần mềm là khâu sống còn của sản phẩm trước khi đưa vào sử dụng, góp phần quyết định sự thành công của dự án phần mềm. Tuy nhiên, kiểm thử là một công việc tiêu tốn rất nhiều thời gian, tiền bạc, công sức. Nhất là đối với các phần mềm lớn, chi phí này càng tăng lên gấp bội mỗi khi có sự thay đổi, nâng cấp các chức năng của phần mềm. Khi có sự thay đổi của phần mềm, đồng nghĩa với việc ngoài việc kiểm thử chức năng mới, các chức năng cũ cũng cần được kiểm tra lại kỹ càng để đảm bảo chúng vẫn hoạt động tốt khi có sự thay đổi. Đó chính là hoạt động kiểm thử hồi quy. Việc kiểm thử hồi quy cần được tự động hóa nhằm giảm thời gian, công sức và chi phí của kiểm thử viên cũng như của dự án phần mềm. Do đó, đề tài nghiên cứu mong muốn đưa ra giải pháp tự động hóa kiểm thử nhằm giảm thiểu chi phí kiểm thử cả về thời gian, tiền bạc, công sức của con người và làm giảm sự nhàm chán cho kiểm thử viên nhưng vẫn đảm bảo được chất lượng của sản phẩm phần mềm. Với sự phát triển mạnh mẽ của các phần mềm cũng như kiểm thử phần mềm hiện nay, có rất nhiều công cụ hỗ trợ cho việc kiểm thử tự động, mỗi công cụ đều có một hoặc một số phương pháp luận khác nhau. Nhưng điều đó không đồng nghĩa với việc lựa chọn bất kỳ công cụ nào cũng tốt, hay cứ áp dụng kiểm thử tự động là có thể tiết kiệm chi phí và đảm bảo dự án sẽ thành công.

Vì vậy, việc xây dựng được 1 framework để áp dụng vào kiểm thử tự động là cần thiết, phù hợp với điều kiện thực tế.

1.2 Mục tiêu của đồ án

1.2.1 Mục tiêu tổng quát

Xây dựng thành công framework kiểm thử tự động hướng lai và áp dụng vào kiểm thử cho website bán sách Fairy Tail

1.2.2 Mục tiêu cụ thể

Nghiên cứu framework kiểm thử tự động phần mềm trong ngành công nghiệp phần mềm: Tập trung nghiên cứu kiểm thử hướng hybrid

Xây dựng bộ thư viện, tài liệu kiểm thử hỗ trợ việc tự động hóa kiểm thử phần mềm

1.3 Giới hạn và phạm vi của đồ án

1.3.1 Đối tượng nghiên cứu

- Framework kiểm thử tự động hướng lai, kết hợp giữa hướng từ khóa và hướng dữ liệu

- Xây dựng framework dựa trên selenium webdriver, ngôn ngữ java

- Khách thể nghiên cứu: Đề tài sử dụng website bán sách Fairy Tail để làm đối tượng thực hiện kiểm thử tự động

1.3.2 Phạm vi nghiên cứu

Phạm vi không gian: tại trường Đại học Sư phạm Kỹ thuật Hưng Yên

Phạm vi thời gian: 01/2021 -05/2021

Bảng 1‑ Danh mục ví dụ phạm vi nghiên cứu

|  |  |
| --- | --- |
| Tiêu đề 1 | Mô tả |
| Phạm vi không gian | Trường Đại học Sư phạm Kỹ thuật Hưng Yên |
| Phạm vi thời gian | 01/2021 -05/2021 |
|  |  |

1.4 Nội dung thực hiện

* Xây dựng đặc tả của trang web, test plan, test design, testcase.
* Xây dựng framework.
* Áp dụng framework vào kiểm thử tự động.

1.5 Phương pháp tiếp cận

* Ngôn ngữ lập trình java.
* Nghiên cứu các công cụ như: Selenium, Eclipse, MS Office…..
* Phương pháp phân tích tài liệu
* Phương pháp phân tích mẫu
* Phương pháp thực nghiệm

# CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về kiểm thử phần mềm

Kiểm thử là một quá trình đánh giá một hệ thống hay là các thành phần của nó với mục đích là xác định xem nó có thỏa mãn những yêu cầu được đưa ra hay không. Hiểu một cách đơn giản, kiểm thử - test là chạy một chương trình để xác nhận bất kì lỗ hổng, lỗi sai hay những yêu cầu bị bỏ quên, những yêu cầu không đúng so với yêu cầu thực tế đề ra.

Theo tiêu chuẩn ANSI/IEEE 1059, kiểm thử - test có thể được định nghĩa như - một quá trình của việc phân tích các thành phần của phần mềm để dò tìm sự khác biệt giữa phần mềm thực tế đang tồn tại và những điều kiện được yêu cầu – requirement (đó là thiếu sót – defect, sai sót - error, lỗi - bug). Từ đó đánh giá được chất lượng của sản phẩm phần mềm.



Ai sẽ là người test?

Điều này phụ thuộc vào quy trình và các bên liên quan đến dự án. Trong ngành công nghiệp phần mềm, những công ty lớn có một team chuyên chịu trách nhiệm việc đánh giá phần mềm đã phát triển với những yêu cầu được chỉ định từ trước. Hơn thế nữa, nhân viên phát triển phần mềm – deverloper cũng tiến hành kiểm thử - test sản phẩm, và việc đó được gọi là unit testing. Trong hầu hết các trường hợp, người kiểm thử - tester có thể là:

Software Tester – Nhân viên kiểm thử phần mềm

Software Deverloper – Nhân viên phát triển phần mềm

Leader hoặc Manager của dự án

User - Người sử dụng cuối cùng

Các công ty khác nhau sẽ có các quy định về tên gọi của nhân viên kiểm thử dựa trên kinh nghiệm và kiến thức về kiểm thử phần mềm, như là: software tester – nhân viên kiểm thử phần mềm, software quality assurance engineer - kĩ sư đảm bảo chất lượng phần, QA analyst – nhân viên phân tích chất lượng phần mềm ...

Thời điểm bắt đầu test

Việc kiểm thử - test sớm sẽ giúp giảm chi phí và thời gian để xây dựng lại và sửa lỗi để bàn giao sản phẩm phần mềm. Tuy nhiên trong một vòng đời của phần mêm, việc kiểm thử - test nên được bắt đầu từ khi có những yêu cầu từ phía khách hàng và được kéo dài đến cho đến khi triển khai phầm mềm. Thời điểm bắt đầu kiểm thử còn phụ thuộc vào mô hình phát triển phần mềm đang được sử dụng. Ví dụ như: trong mô hình thách nước – waterfall model, kiểm thử - test chính thức được thực hiện ở giai đoạn kiểm thử - testing phase. Nhưng ở trong mô hình gia tăng – incremental model, kiểm thử được thực hiện ở cuối mỗi chu kỳ con và kiểm thử cho toàn bộ sản phẩm được thực hiện ở giai đoạn cuối khi hoàn thiện sản phẩm.

Việc kiểm thử - test được thể hiện theo nhiều dạng công việc khác nhau ở các giai đoạn khác trong suốt vòng đời phát triển phần mềm:

Trong quá trình tập hơp yêu cầu – requirement gathering phase, phân tích và xác minh yêu cầu cũng được coi là kiểm thử - test requirement.

Trong giai đoạn thiết kế - design phase, kiểm tra lại thiết kế với mục đích cải thiện thiết kế cũng được tính là kiểm thử.

Trong giai đoạn phát triển phần mềm – Implement phase, kiểm thử được thực hiện bởi lập trình viên – unit testing cũng được tính là kiểm thử.

Khi nào thì dừng việc test

Rất khó để xác định khi nào thì dừng kiểm thử - test, vì test là quá trình ko bao giờ kết thúc và không ai có thể yêu cầu phần mềm được test 100%. Các khía cạnh dưới đây có thể được cân nhắc cho việc dừng lai quy trình test:

Hạn chót phát triển phần mềm – deadline.

Đã thực hiện hết các testcase được đề ra.

Hoàn thiện các chức năng và toàn bộ code đã đảm bảo được các yêu cầu đề ra.

Tỷ lệ bug phải trong giới hạn mong muốn.

Không có những bug nghiêm trọng

Quyết định của quản lý dự án

Verification & Validation

Đây là hai khái niệm làm khó hiểu cho nhiều người trong việc sử dụng chúng

Bảng dưới đây sẽ chỉ ra sự khác nhau giữa Verification & Validation: **VERIFICATION** là qui trình đánh giá một hệ thống hoặc một thành phần để xác định xem liệu các sản phẩm của một giai đoạn phát triển nhất định có đáp ứng được những yêu cầu được định tại thời điểm bắt đầu của giai đoạn đó không.

**VALIDATION** là qui trình đánh giá một hệ thống hoặc một thành phần trong suốt quá trình phát triển hoặc lúc kết thúc của quá trình phát triển để xác định xem liệu nó có được làm ra đúng như những yêu cầu cụ thể như những yêu cầu lúc đầu đưa ra không.



2.2 Kiểm thử tự động phần mềm

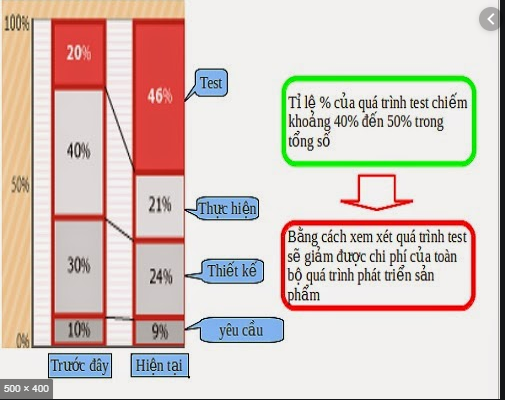
* + 1. Giới thiệu về kiểm thử tự động
* Kiểm thử tự động là gì?

Là việc sử dụng phần mềm đặc biệt (tách biệt với phần mềm đang được kiểm thử) để kiểm soát việc thực hiện các bài kiểm tra và so kết quả thực tế với kết quả dự đoán.

Kiểm thử tự động có thể tự động hóa một số nhiệm vụ lặp đi lặp lại nhưng cần thiết trong một quá trình thử nghiệm đã được chính thức hóa, hay là các kiểm thử bổ sung nhưng sẽ khó thực hiện thủ công.

Kiểm thử tự động là rất quan trọng cho phân phối liên tục và kiểm thử liên tục.

* Tại sao cần sử dụng kiểm thử tự động ?



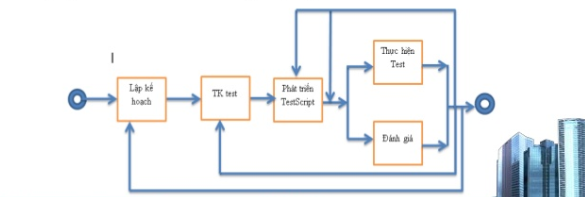
Hình 2.1 Tại sao cần sử dụng kiểm thử tự động

Kiểm thử tự động được thực hiện khi:

Các trường hợp kiểm thử được thực hiện lặp đi lặp lại để đảm bảo tính năng của phần mềm/ sản phẩm.

Thực hiện ở các trường hợp mà kiểm thử thủ công khó thực hiện. Các trường hợp kiểm thử cần tốn nhiều thời gian.

* Ưu điểm của kiểm thử tự động:
* Giúp tiết kiệm tiền bạc và thời gian: nhất là trong giai đoạn bảo trì của các dự án lớn. Mỗi tuần chúng ta phải thực hiện regression test từ 1 đến 2 lần với số lượng test case rất lớn trong 1 đến 2 ngày. Điều này gần như không thể thực hiện bằng cách thủ công, trong khi với kiểm thử tự động chúng ta hoàn toàn có thể với nguồn nhân lực vô cùng khiêm tốn.
* Chính xác hơn: Nhờ độ ổn định cao, kiểm thử tự động có thể thực thi các test case với độ chính xác cao hơn.
* Độ bao phủ cao: Như đã nói ở trên, khi sử dụng kiểm thử tự động, chúng ta có thể thực thi số lượng lớn test case trong một thời gian ngắn. Nên độ bao phủ của nó rất cao. Điều này giúp chúng ta tăng độ bao phủ trong giai đoạn regression test.
* Hoàn thành các công việc mà con người không thể làm được: Nếu chúng ta muốn thực thi load test, performance test, thì kiểm thử tự động là cách duy nhất.
  + 1. Quy trình kiểm thử tự động



Hình 2.2 Quy trình kiểm thử tự động

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

Để kiểm thử tự động thì công cụ là thành phần không thể thiếu trong tiến trình này, việc kiểm thử viên thành thạo các công cụ kiểm thử đảm bảo cho quy trình kiểm thử tự động được hiệu quả.

* Một công cụ kiểm thử phần mềm tự động yêu cầu phải làm được những công việc sau:
* Hiểu các mã assembly được kiểm tra một cách tự động.
* Tiến hành các nhiệm vụ đơn giản và lặp đi lặp lại một cách tự động.
* Tạo ra các kịch bản và chạy các kịch bản trong những lô lệnh theo một lịch trình đã vạch ra.
* Kiểm tra giao diện của các đối tượng COM và các thành phần phần mềm khác với tập dữ liệu được thiết lập sẵn.
* Truy cập vào dữ liệu để xác minh lại các kết quả.
* Truy cập vào Regestry để xác minh lại các kết quả.
  + 1. Một số công cụ hỗ trợ kiểm thử tự động
* Selenium

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

* QTP (HP UFT )

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

* Rational Function Tester

Là 1 công cụ kiểm tra tự động hướng đối tượng có khả năng tự động kiểm tra dữ liệu, kiểm tra giao diện, và kiểm thử hồi quy (Regression Testing).

* Watir

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

* SilkTest

Silk Test được thiết kế để thực hiện kiểm tra chức năng (Functional Testing) và hồi quy (Regression Testing). Nó là một ngôn ngữ hướng đối tượng giống như C ++. Nó sử dụng các khái niệm về đối tượng, các class và sự kế thừa.

2.3 Các loại kiểm thử phần mềm

* + 1. Testing of function ( Functional testing)

Kiểm thử chức năng là kiểm tra xem hệ thống có hoạt động theo đúng theo các yêu cầu nghiệp vụ không? Kiểm thử chức năng được thực hiện ở tất cả các mức kiểm thử.

Kiểm thử chức năng có thể thực hiện theo 2 quan điểm: requirements-based và business - process - based

**Requirements - based:** Sử dụng các đặc tả yêu cầu của hệ thống làm cơ sở để design test. Một cách tốt để bắt đầu là sử dụng bảng nội dung của đặc tả yêu cầu như một danh sách các mục kiểm thử và không kiểm thử. Chúng ta nên xét độ ưu tiên của yêu cầu dựa trên các tiêu chí rủi ro và sử dụng độ ưu tiên để kiểm thử. Điều này sẽ đảm bảo những phần quan trọng nhất sẽ được kiểm thử.

**Business - process - based:** sử dụng các kiến thức về quy trình nghiệp vụ. Quy trình nghiệp vụ mô tả các kịch bản liên quan đến nghiệp vụ hằng ngày của hệ thống.

**Ví dụ:** Một hệ thống quản lý nhân sự và bảng lương có thể có quy trình nghiệp vụ như sau: người nào đó gia nhập vào công ty, họ được trả lương thường xuyên và cuối cùng họ rời khỏi công ty. Các use case bắt nguồn từ hướng đối tượng, nhưng ngày nay phổ biến nhiều là dựa trên vòng đời phát triển phần mềm. Chúng ta có thể lấy quy trình nghiệp vụ là điểm bắt đầu. Các use case là một cơ sở rất hữu ích tạo ra các trường hợp kiểm thử từ góc nhìn về nghiệp vụ.

Kiểm thử chức năng bao gồm 5 bước:

* Xác định các chức năng mà phần mềm mong muốn sẽ thực hiện.
* Tạo các dữ liệu đầu vào dựa trên các tài liệu đặc tả kỹ thuật của các chức năng.
* Xác định các kết quả đầu ra dựa trên các tài liệu đặc tả kỹ thuật của các chức năng.
* Thực hiện các trường hợp kiêm thử.
* So sánh kết quả thực tế và kết quả mong muốn.

Các loại kiểm thử chức năng bao gồm:

* Kiểm thử đơn vị (Unit Testing)
* Smoke Testing
* Sanity Testing
* Kiểm thử giao diện (Interface Testing)
* Kiểm thử tích hợp (Integration Testing)
* Kiểm thử hệ thống (System Testing)
* Kiểm thử hồi quy (Regression Testing)
* Kiểm thử chấp nhận (Acceptance Testing)
  + 1. Testing of software product characteristics (Non - Functional testing)

Kiểm thử phi chức năng là các đặc tính chất lượng của hệt thống sẽ được kiểm tra. Chúng ta quan tâm đến việc mọi thứ hoạt động tốt không? Hay nhanh như thế nào? Chúng ta sẽ kiểm tra những thứ cần phải đo như thời gian phản hồi, hay bao nhiêu người có thể đăng nhập cùng một lúc? Kiểm thử phi chức năng cũng giống như kiểm thử chức năng được thực hiện ở tất cả các cấp độ kiểm thử.

Kiểm thử phi chức năng bao gồm:

* Kiểm thử hiệu năng (Performance testing)
* Kiểm thử khả năng chịu tải (Load testing)
* Kiểm thử áp lực (Stress testing)
* Kiểm thử tính khả dụng (Usability testing)
* Kiểm thử bảo trì (Maintainability testing)
* Kiểm thử độ tin cậy (Reliability testing)
* Kiểm thử tính tương thích (Portability testing)

Các đặc điểm và các đặc điểm phụ tương ứng:

**Chức năng** (Functionality) gồm 5 đặc điểm phụ: sự phù hợp, chính xác, bảo mật, khả năng tương tác và tuân thủ.

**Độ tin cậy** (Reliability) gồm 4 đặc điểm phụ: độ bền, khả năng chịu lỗi, khả năng phục hồi và tuân thủ.

**Khả năng sử dụng** (Usability) gồm 5 đặc điểm phụ: dễ hiểu, khả năng học hỏi, khả năng hoạt động, sự thu hút và tính tuân thủ.

**Tính hiệu quả** (Efficiency) gồm 3 đặc điểm phụ: thời gia (hiệu suất), sử dụng tài nguyên và tuân thủ.

**Khả năng bảo trì** (Maintainability) gồm 5 đặc điểm phụ: khả năng phân tích, khả năng thay đổi, tính ổn định, khả năng kiểm tra và tuân thủ.

**Tính tương thích** (Portability) gồm 5 đặc điểm phụ: khả năng thích ứng, khả năng cài đặt, cùng tồn tại, khả thăng thay thế và tuân thủ.

* + 1. Testing of software structure/architecture (Structural testing)

Kiểm thử cấu trúc thường được gọi là "hộp trắng" hoặc "hộp thủy tinh" vì chúng quan tâm đến những gì đang xảy ra bên trong hộp.

Kiểm thử cấu trúc thường được sử dụng như một cách đo lường của kiểm thử thông qua độ bao phủ của một tập hợp các yếu tố cấu trúc hoặc các mục bao phủ.

Nó có thể xảy ra ở bất kỳ mức độ kiểm thử nào chủ yếu ở kiểm thử thành phần, tích hợp.

Ở cấp độ thành phần, và mức thấp hơn trong kiểm thử tích hợp thành phần có hỗ trợ công cụ tốt để đo mức độ bao phủ của mã. Các công cụ đo lường độ bao phủ đánh giá tỉ lệ phần trăm thực thi đã được thực hiện bởi một bộ kiểm thử. Nếu độ bao phủ không phải là 100% thì các kiểm thử bổ sung có thể cần phải được viết và chạy để bao phủ những phần chưa được thực hiện.

Các kỹ thuật được sử dụng để kiểm tra cấu trúc là kỹ thuật kiểm thử hộp trắng, các mô hình luồng điều khiển thường sử dụng để hỗ trợ kiểm thử cấu trúc.

* + 1. Testing related to changes (Confirmation and regression testing)

Confirmation testing (Kiểm thử xác nhận)

Khi kiểm thử bị lỗi, và chúng ta xác định nguyên nhân lỗi là do lỗi phần mềm, lỗi được báo cáo, khi một phiên bản mới của phần mềm đã sửa lỗi. Trong trường hợp này chúng ta cần thực hiện kiểm tra một lần nữa để xác định rằng lỗi thực sự đã được sửa.

Khi thực hiện kiểm tra xác nhận điều quan trọng là phải đảm bảo rằng thử nghiệm được thực hiện chính xác giống như lần đầu tiên, sử dụng cùng một đầu vào, dữ liệu và môi trường. Nếu bây giờ đúng có nghĩa là phần mềm chính xác. Chúng ta biết rằng ít nhất một phần của phần mềm là chính xác, nhưng điều đó là không đủ. Sửa lỗi có thể gây ra một lỗi khác trong phần mềm. Cách phát hiện các bất lợi ngoài ý muốn của việc sửa lỗi là thực hiện kiểm thử hồi quy.

Regression testing (Kiểm thử hồi quy)

Giống như kiểm thử xác nhận kiểm thử hồi quy liên quan đến việc thực hiện các trường hợp kiểm thử đã được thực hiện trước đó. Sự khác biệt là đối với kiểm thử hồi quy, các trường hợp kiểm thử có thể đúng ở lần cuối cùng chúng được thực thi.

Mục đích của kiểm thử hồi quy để xác minh rằng các sửa đổi trong phần mềm hoặc môi trường không gây ra bất lợi ngoài ý muốn và hệ thống vẫn đáp ứng các yêu cầu của nó.

Bộ kiểm thử hồi quy hoặc gói kiểm tra hồi quy là một tập hợp các trường hợp kiểm thử được sử dụng đặc biệt để kiểm tra hồi quy. Chúng được thiết kế để thực hiện hầu hết các chức năng trong một hệ thống nhưng không chi tiết bất kỳ chức năng nào. Tất cả các trường hợp trong bộ kiểm thử hồi quy sẽ được thực thi mỗi khi một phiên bản mới của phần mềm được phát hành và điều này làm cho chúng trở lên lý tưởng cho tự động hóa.

Kiểm thử hồi quy được thực hiện khi phần mềm thay đổi, do sửa lỗi, chức năng mới. Nó cũng là một ý tưởng tốt để thực thi chúng khi một vài khía cạnh của môi trường thay đổi.

2.4 Các hướng kiểm thử tự động

Những Mô hình phổ biến hiện nay

* Tuyến tính
* Hướng module
* Hướng dữ liệu
* Hướng từ khóa
* Hỗn hợp
  + 1. Tuyến tính – Linear Framework

Đặc điểm cơ bản

* Mọi thứ liên quan đến mã đều được định nghĩa bên trong phương thức kiểm thử
* Không quan tâm đến việc lãng phí và trùng lặp các câu lệnh
* Việc record/playback thường xuyên sinh ra mã tuyến tính
* Dễ dàng để bắt đầu
* Khó khăn trong việc chỉnh sửa

Mô hình này có thể được dùng trong dự án nhỏ, nơi mà không có quá nhiều màn hình giao diện cũng như chức năng. Mặc khác, chúng ta cũng hay dùng Mô hình này khi sử dụng một công cụ kiểm thử tự động lần đâu tiên. Thông qua việc record/playback và phát sinh mã, chúng ta có thể học được cách công cụ tương tác với ứng dụng như thế nào. Ngoài hai lý do trên, Mô hình này không được khuyến khích sử dụng trong kiểm thử tự động.

* + 1. Hướng modul – Modularity

Đặc điểm cơ bản

* Các đối tượng được định nghĩa một lần và tái sử dụng trong các phương thức kiểm thử.
* Các phương thức nhỏ và có mục đích được tạo ra cho những chức năng riêng biệt
* Kịch bản kiểm thử tự động là một tập hợp các phương thức nhỏ và các đối tượng được định nghĩa từ trước
* Cho phép chúng ta dễ dàng viết các mã dễ dàng được chỉnh sửa

Mục đích chính của Mô hình này là việc chỉnh sửa dễ dàng. Nếu có bất kỳ thay đổi nào trên giao diện, chúng ta chỉ cần chỉnh sửa trong các phương thức và đối tượng. Mã kiểm thử chính của chúng ta vẫn hoạt động chính xác. Mô hình POM (Page Object Model) – thường được dùng với [Selenium](https://vntesters.com/tu-hoc-selenium/) – là một dạng của việc ứng dụng Mô hình hướng modul. Toàn bộ trang web sẽ được chia nhỏ thành các trang. Các đối tượng UI của từng trang được định nghĩa bên trong từng lớp của trang đó. Nếu có bất kỳ thay đổi nào trên ứng dụng web, chúng ta chỉ cần chỉnh sửa lớp của trang đó, những lớp của trang khác vẫn giữ nguyên. Kết quả cuối cùng chúng ta sẽ có những đoạn mã được bảo trì tốt hơn và dễ đọc hơn.

Điểm yếu của mô hình này là nó yêu cầu một mức độ kỹ năng lập trình và hiểu sâu về hướng đối tượng. Nếu bạn có nó, khuôn mẫu này được khuyến khích sử dụng.

* + 1. [Hướng dữ liệu](https://en.wikipedia.org/wiki/Data-driven_testing) – Data driven

Đặc điểm cơ bản

* Dữ liệu kiểm thử (giá trị đầu vào và đầu ra) được tách khỏi mã nguồn và lưu trong một tập tin bên ngoài. Nó có thể là một tập tin CSV, một bảng Excel hay một cơ sở dữ liệu.
* Khi mã kiểm thử thực thi, các giá trị này được lấy ra từ tập tin, chứa vào biến và thay thế các giá trị cứng (nếu có) trong mã nguồn.
* Thực sự hữu ích khi mà cùng một kịch bản kiểm thử cần thực thi với nhiều dữ liệu đầu vào khác nhau.

Có vài ưu điểm khi áp dụng mô hình này. Tất cả các giá trị kiểm thử được lưu bên ngoài mã nguồn, do đó, bất kỳ thay đổi nào xảy ra trong quá trình phát triển ứng dụng, chúng ta chỉ cần thay đổi dữ liệu trong tập tin bên ngoài, và mã kiểm thử tự động của chúng ta vẫn được giữ nguyên.

Một ưu điểm khác là, khả năng sử dụng một kịch bản kiểm thử cho nhiều dữ liệu khác nhau. Ví dụ như, bạn đang làm một kịch bản đăng nhập hệ thống với 100 user. Bạn có thể viết 1 đoạn mã và một tập tin lưu trữ thông tin của 100 user. Sau đó, bạn chỉ cần thực thi 1 lần, và đi qua cả 100 bộ dữ liệu. Bạn dễ dàng phát hiện, với kiểu dữ liệu nào thì đoạn mã Fail. Đây cũng là một thế mạnh khi bạn đang làm kiểm thử phủ định – Negative Test.

* + 1. [Hướng từ khóa](https://en.wikipedia.org/wiki/Keyword-driven_testing) – Keyword driven

Đặc điểm cơ bản

* Cả dữ liệu và chức năng được định nghĩa bên ngoài mã nguồn
* Cần phát triển các từ khóa cho nhiều chức năng khác nhau
* Mã kiểm thử tự động đôi khi được lưu trữ ở một tập tin bên ngoài mã nguồn giống như mô hình hướng dữ liệu. Các bước của kịch bản kiểm thử được viết từng bước với định dạng bảng, nơi mà sử dụng các từ khóa và dữ liệu kiểm thử
* Mã nguồn chính sẽ đọc các bước trong định dạng bảng và thực thi các chức năng tương ứng
* Cho phép các kỹ sư kiểm thử thủ công, nhưng người không biết về lập trình, có thể là một phần, ở một mức độ, của nhóm kiểm thử tự động

Ưu điểm của mô hình hướng từ khóa

Mô hình này rất hữu dụng trong những trường hợp mà kịch bản kiểm thử có quá nhiều thay đổi. Nếu bất kỳ bước nào trong kịch bản kiểm thử bị thay đổi, chúng ta không cần phải chỉnh sửa mã nguồn. Chúng ta chỉ cần chỉnh sửa tập tin bên ngoài và như vậy, kịch bản kiểm thử tự động sẽ được chỉnh sửa theo.

Chúng ta định nghĩa toàn bộ kịch bản ở tập tin và đưa cho kỷ sư kiểm thử thủ công, họ sẽ thêm các đoạn văn bản (text) hoặc chỉnh sửa cái có sẵn. Bằng cách này, kỹ sư kiểm thử tự động cũng có thể trở thành một phần của nhóm kiểm thử tự động bởi vì họ không cần phải lập trình gì cả. Họ chỉ cần chỉnh sửa tập tin ở những vị trí cần thiết và kịch bản kiềm thử tự động sẽ được chỉnh sửa một cách tự động.

Một lợi ích khác của mô hình này là, kịch bản kiểm thử của bạn trở thành một công cụ độc lập. Bạn chỉ cần bảo trì kịch bản kiểm thử trong một tập tin và nếu bạn cần thay đổi công cụ kiềm thử tự động ở điểm nào đó, bạn có thể dễ dàng chỉnh sửa bằng cách viết lại cách đọc và thực thi tập tin với công cụ mới.

Mặc khác, khuyết điểm của mô hình này là, bạn cần phát triển các từ khóa cho các chức năng khác nhau. Trong một dự án lớn, có thể có rất nhiều từ khóa mà bạn cấn phải nhớ và tổ chức nó hợp lý. Bản thân việc này có thể sẽ làm một công việc nặng nhọc cho quá trình phát triển kiềm thử tự động.

Ở vài trường hợp phức tạp, khi mà các đối tượng UI không thể được xác định dễ dàng, chúng ta phải sử dụng nhiều kỹ thuật khác nhau để xử lý, mô hình này không hữu dụng cho lắm.

Mô hình hướng từ khóa là một mô hình ưa thích của nhiều kỹ sư kiểm thử tự động. [Robot Framework](http://robotframework.org/) – công cụ kiểm thử tự động được phát triển bởi Google – là một công cụ phổ biến đi theo hướng từ khóa. Những công cụ đi theo hướng từ khóa này còn có [Test Architect](http://www.testarchitect.com/) hay [Katalon Studio](https://vntesters.com/katalon-studio-tool-from-viet-nam/)

* + 1. Hướng hỗn hợp – Hybrid

Đặc điểm cơ bản

* Là sự kết hợp của hai hoặc nhiều kỹ thuật ở trên, kế thửa thế mạnh và loại bỏ những điểm yếu của các mô hình khác.
* Mô hình này sử dụng cách tiếp cận theo hướng modul, kết hợp với hướng dữ liệu hoặc hướng từ khóa
* Mô hình này có thể dùng mã nguồn để xủ lý những công việc đặc biệt mà quá khó để tạo ra với cách làm từ khóa

Một cách đơn giản, mô hình này sử dụng kết hợp nhiều kỹ thuật với nhau. Chúng ta có thể sử dụng hướng dữ liệu đồng thời với hướng modul. Trong vài trường hợp, chúng ta có thể dùng từ khóa song song với modul. Cơ bản, khi nào chúng ta sử dụng nhiều hơn một mô hình, đó là lúc chúng ta sử dụng hỗn hợp – Hybrid

2.5 Công nghệ/công cụ áp dụng

* + 1. Selenium WebDriver

Selenium Webdriver là gì?

Selenium WebDriver là một khuôn khổ tự động hóa web cho phép bạn thực hiện các kiểm thử của mình trên các trình duyệt khác nhau. Nó nằm trong bộ kiểm thử tự động Selenium.

Tại sao sử dụng Selenium WebDriver?



Người dùng có thể dùng miền phí.

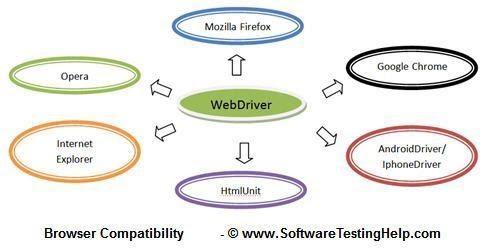
Kiến trúc đơn giản: Điều khiển trình duyệt hệ điều hành.

Ngôn ngữ lập trình hỗ trợ: WebDriver hỗ trợ bạn viết kịch bản kiểm thử bằng các ngôn ngữ khác nhau như Java, .NET, PHP, Python, Perl, Ruby và bạn có thể sử dụng các điều kiện if else, các vòng lặp để tăng tính chính xác cho kịch bản kiểm thử.

Tốc độ: Khi so sánh với các công cụ khác của bộ Selenium, WebDriver là công cụ nhanh nhất trong số tất cả do tương tác trực tiếp từ hệ điều hành tới trình duyệt.

Sử dụng Selenium WebDriver ở đâu?

Web Driver được hỗ trợ trên các trình duyệt: Firefox, Google Chrome, Internet Explorer, Opera browser, Sarafi,…



Hình 2.3 Các trình duyệt Selenium hỗ trợ

* + 1. TestNG

TestNG là một framework thử nghiệm lấy cảm hứng từ JUnit và NUnit nhưng đem đến một số tính năng mới làm cho nó mạnh hơn và dễ sử dụng hơn.

Đây là một framework thử nghiệm mã nguồn mở tự động; trong đó NG của TestNG có nghĩa là Next Generation - Thế hệ tiếp theo. TestNG tương tự như JUnit nhưng mạnh hơn Junit, nhưng nó vẫn được lấy cảm hứng từ JUnit. Nó được thiết kế tốt hơn so với JUnit, đặc biệt là khi kiểm tra các lớp tích hợp.

TestNG loại bỏ hầu hết các hạn chế của khuôn khổ cũ hơn và cho phép nhà phát triển khả năng viết các bài kiểm tra linh hoạt và mạnh mẽ hơn với sự trợ giúp của các chú thích đơn giản, nhóm, sắp xếp và parametrizing.

Lợi ích của TestNG

Có nhiều lợi ích nhưng theo quan điểm của Selenium, ưu điểm chính của TestNG là:

* Nó cho phép tạo ra các bản báo cáo HTML của tiến trình thực thi
* Các chú thích giúp việc kiểm thử dễ dàng hơn
* Các trường hợp kiểm thử có thể được nhóm lại và được ưu tiên dễ dàng hơn
* Có thể kiểm thử song song
* Tạo ra các bản ghi
* Có thể tham số hóa dữ liệu
  + 1. ****Eclipse****

Eclipse là 1 công cụ hỗ trợ lập trình mã nguồn mở được phát triển bởi IBM.

Eclipse như một môi trường phát triển Java tích hợp (IDE), với Eclipse chúng ta có thể mở rộng hơn mã nguồn bằng cách chèn thêm các plugins cho project (PDE- Plug-in Development Environment). Mặc dù Eclipse được viết bằng ngôn ngữ lập trình Java, nhưng việc sử dụng nó không hạn chế chỉ cho ngôn ngữ Java. Ví dụ, Eclipse hỗ trợ sẵn hoặc có thể cài thêm các plugins để hỗ trợ cho các ngôn ngữ lập trình như C/C + + và COBOL. Ngoài ra, còn rất nhiều ngôn ngữ khác như PHP, Groovy, ...

Eclipse còn hỗ trợ cho lập trình viên code theo các mô hình phát triển như MVC, tạo thêm các lib hỗ trợ phát triển phần mềm.

Ưu điểm

* Tạo thuận lợi cho tích hợp liền mạch các công cụ bên trong mỗi một và xuyên qua nhiều kiểu nội dung và các nhà cung cấp công cụ khác nhau.
* Hỗ trợ việc xây dựng nhiều công cụ.
* Hỗ trợ một tập hợp không hạn chế các nhà cung cấp công cụ, bao gồm cả các nhà cung cấp phần mềm độc lập (ISV).
* Hỗ trợ các công cụ thao tác các kiểu nội dung bất kỳ (bao gồm cả HTML, Java, C, JSP, EJB, XML, và GIF).
* Hỗ trợ cả môi trường phát triển ứng dụng GUI lẫn không dựa trên GUI.
* Chạy trên nhiều hệ điều hành, bao gồm cả Windows và Linux.
* Lợi dụng tính phổ biến của ngôn ngữ Java để viết các công cụ.
* Do sử dụng SWT/JFace nên có lẽ load nhanh hơn

Nhược điểm

* Cài đặt phức tạp
* Tốn phần cứng máy Eclipse cũng khá tốn bộ nhớ máy để cài đặt nên hiện nay eclipse đã khắc phục bằng phiên bản Eclipse Luna, không cần cái đặt
* Tốn bộ nhớ máy: chạy eclipse cũng khá tốn bộ nhớ máy tính nên để sử dụng nó máy của bạn cũng cần có cấu hình tương đối cao
* Thiếu tính nhất quán vì có nhiều plugins quá

# ĐẶC TẢ HỆ THỐNG PHẦN MỀM

* 1. Giới thiệu về hệ thống phần mềm

Trong những năm gần đây với sự phát triển vượt trội của khoa học kỹ thuật đặt biệt là công nghệ thông tin, với những ứng dụng của công nghệ thông tin vào các lĩnh vực đã đóng góp phần to lớn cho sự nghiệp phát triển của con người. Trong các lĩnh vực đó thì lĩnh vực quản lý là thật sự giúp ích được rất nhiều cho con người, việc áp dụng quản lý và mua bán bằng máy tính thay cho quản lý và mua bán bằng thủ công ở các doanh nghiệp, công ty, cá nhân… là rất cần thiết và thật sự cần thiết. Do đó, việc “quản lý mua bán hàng qua mạng ” là không thể thiếu được trong mọi doanh nghiệp, cửa hàng vừa và nhỏ .

Năm 2008 là năm đầu tiên Việt Nam có sự phát triển lớn mạnh về lĩnh vực thương mại khi chính thức trở thành thành viên thứ 150 của tổ chức thương mại thế giới WTO.

Với sự phát triển mạnh mẽ đó không thể phủ nhận sự đóng góp của thương mại điện tử, một lĩnh vực nóng bỏng hiện nay! Một đất nước đang phát triển mạnh mẽ, cuộc sống con người càng ngày nâng cao, mức tiêu thụ sản phẩm ngày càng tăng…Tóm lại nhu cầu con người ngày càng cao. Vì thế việc trao đổi mua bán cũng như quản lý hàng hóa cần phải có sự thay đổi từ thủ công sang máy móc.

Như chúng ta cũng thấy trong thị trường hiện nay thì việc cạnh tranh về kinh doanh ngày càng trở nên quyết liệt và hầu hết những nhà kinh doanh, những công ty lớn đều rất chú tâm đến việc làm thỏa mãn khách hàng một cách tốt nhất. So với kinh doanh truyền thống thì thương mại điện tử không những giúp giảm thiểu chi phí mà còn đạt hiệu quả cao hơn.

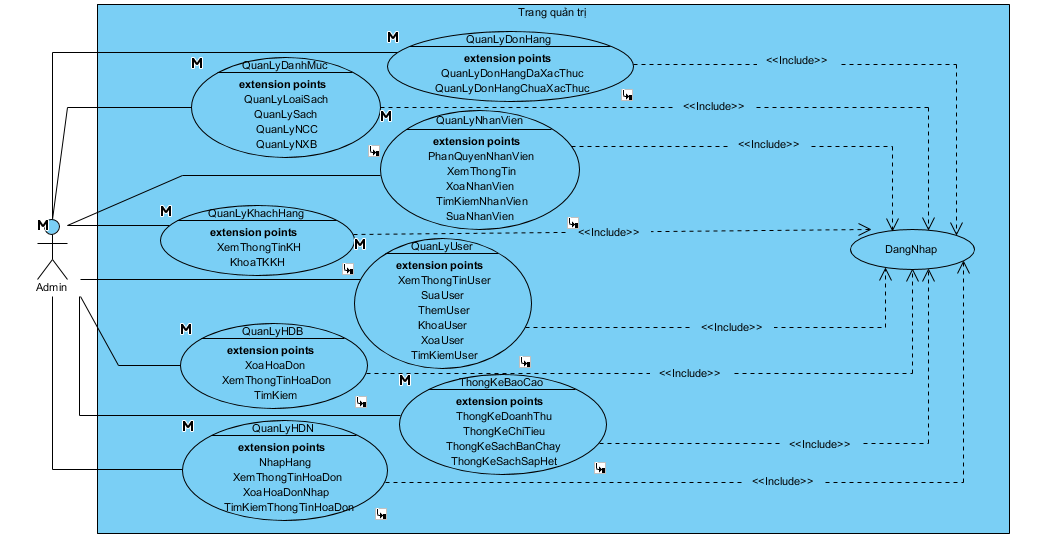
Hơn thế nữa, với lợi thế của công nghệ Internet nên việc truyền tải thông tin về sản phẩm nhanh chóng, thuận tiện. Kết hợp với bộ phận giao hàng tận nơi, là thông qua bưu điện và ngân hàng để thanh toán tiền, càng tăng thêm thuận lợi để loại hình này phát triển.

**Và đối tượng kinh doanh mà em đề cập ở đây là sách - một tên gọi cực kỳ quan trọng trong cuộc sống của chúng ta và quả thực nó là phần không thể thiếu cho sự phát triển toàn diện của mỗi người.** Trong sách có chứa đựng rất nhiều thứ: Đó là kiến thức, kinh nghiệm sống, kinh nghiệm kinh doanh, cảm xúc của con người và đó là nơi ghi chép lại những thành tựu của con người, là viên gạch giúp con người xây dựng lên thành của bản thân.

Nắm bắt được tình hình hiện tại cũng như nhu cầu của con người và sự phát triển của công nghệ, dịch vụ kinh doanh online hiện nay nên em đã chọn đề tài: “Xây dựng website bán sách FairyTail” giúp cho người bán và người mua dễ dàng tiếp cận với nhau một cách nhanh chóng, thuận tiện.

* 1. Các yêu cầu chức năng

Các chức năng chính của hệ thống được mô tả như sơ đồ dưới đây.



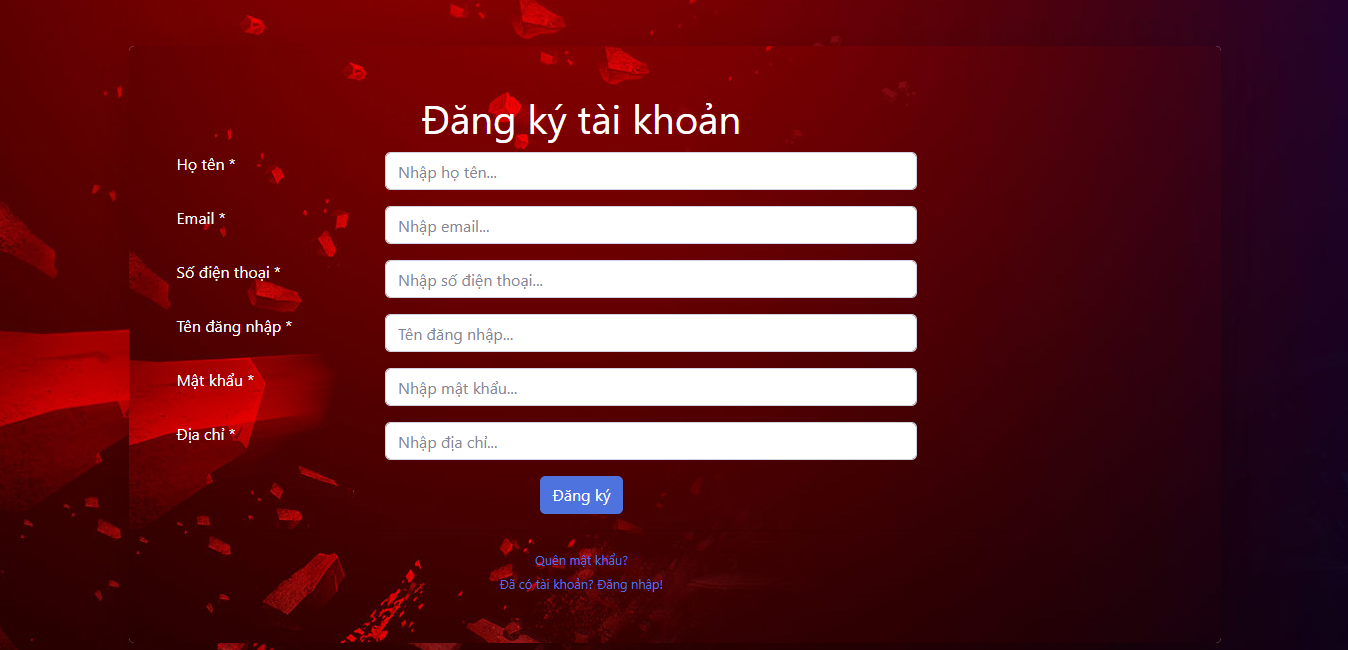
Hình 3.1 Biểu đồ Use case tổng quát

Use case được mô tả như bảng dưới:

Bảng 2.1

|  |  |  |
| --- | --- | --- |
| STT | Chức năng | Mô tả |
| 1 | Quản lý loại sách | Quản lý thông tin loại sách bao gồm các chức năng: thêm, sửa, xóa, tìm kiếm. |
| 2 | Quản lý sách | Người quản trị có thể quản lý thông tin tất cả sách bao gồm cả thêm, sửa, xóa, tìm kiếm các sản phẩm trên website đang bán. |
| 4 | Quản lý giá | Người quản trị có thể quản lý thông tin về giá bao gồm thêm, sửa giá. |
| 5 | Quản lý tác giả | Người quản trị có thể quản lý thông tin về tác giả bao gồm thêm, sửa, tìm kiếm tác giả. |
| 6 | Quản ý nhà xuất bản | Người quản trị có thể quản lý thông tin về nhà xuất bản bao gồm thêm, sửa, xóa, tìm kiếm nhà xuất bản. |
| 7 | Quản lý đơn hàng chưa xác thực | Người quản trị quản lý các đơn hàng chưa xác thực: xác thực đơn hàng, hủy đơn hàng chưa xác thực. |
| 8 | Quản lý đơn hàng đã xác thực | Người quản trị quản lý các đơn hàng đã xác thực |
| 9 | Quản lý hóa đơn bán | Người quản trị quản lý các đơn hàng đã giao: ghi nhận thanh toán, ghi nhận giao hàng không thành công. |
| 10 | Quản lý nhà cung cấp | Khi nhập sách về bán, quản trị viên sẽ thêm thông tin nhà cung cấp sách vào hệ thống, khi thông tin của nhà cung cấp thay đổi thì sẽ phải cập nhật lại, và khi không còn nhập sách từ đó nhà cung cấp đó nữa thì xóa khỏi hệ thống |
| 11 | Quản lý nhập hàng | Khi có hàng mới nhập thêm về , quản trị viên sẽ tiến hành nhập thông tin hóa đơn nhập hàng, và nếu nội dung hóa đơn nhập có sai sót sẽ tiến hành cập nhật lại thông tin hóa đơn nhập hàng. Khi không còn muốn lưu trữ thông tin nhập hàng sẽ xóa khỏi hệ thống |
| 12 | Quản lý tin tức | Quản trị viên sẽ quản lý thông tin các tin tức liên quan đến sách của cửa hàng bao gồm: mã tin tức, tiêu đề, tên tin tức, nội dung, ảnh, ngày đăng, người đăng. Thêm tin tức mới, cập nhật tin, xóa tin. |
| 13 | Quản lý nhân viên | Người quản trị có thể quản lý thông tin về nhân viên bao gồm thêm, sửa, xóa, hiển thị nhân viên. |
| 14 | Quản lý khách hàng | Người quản trị có thể quản lý thông tin về khách hàng |
| 15 | Quản lý user | Người quản trị có thể quản lý thông tin về user |

* + 1. Chức năng Đăng ký



Hình 3.2 Màn hình đăng ký tài khoản

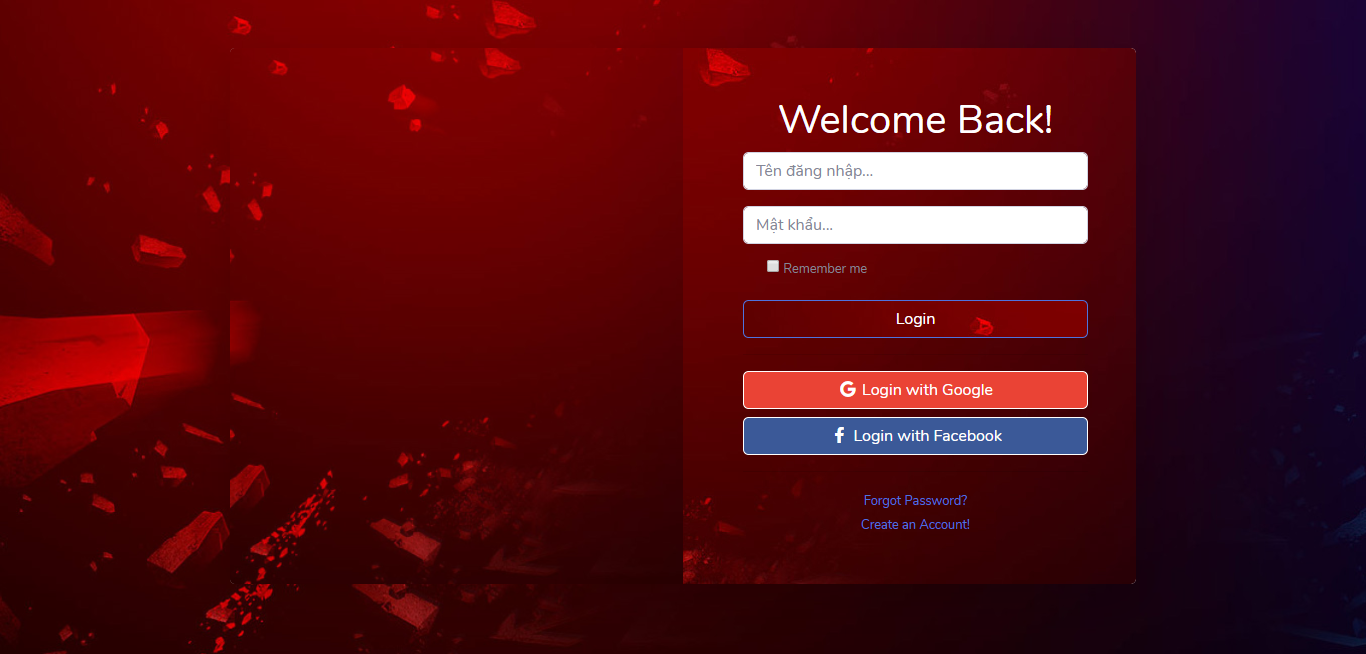
Chức năng **[Đăng ký]:** cho phép đăng ký tài khoản người dùng (tài khoản khách hàng) để có thể đăng nhập vào hệ thống.

Validate:

* Tên đăng nhập: length(50), khác rỗng
* Mật khẩu: kiểu mật khẩu(“\*\*\*\*\*”), length(32), khác rỗng
* Tên người dùng: length(50), khác rỗng
* Số điện thoại: length(15), kiểu number, khác rỗng
* Địa chỉ: length(500), khác rỗng
* Email: length(50), kiểu email (“xyz@gmail.com”)

Chức năng: Người dùng nhập thông tin vào các ô nhập liệu và nhấn nút [Đăng ký]

* Nếu có trường không hợp lệ về validate: hiện label màu đỏ ngay dưới ô nhập liệu không hợp lệ.
* Nếu [Email] đã có người sử sụng để đăng ký: hiện label màu đỏ dưới nút [Đăng ký] với nội dung: “Email đã có người đăng ký!”.
* Nếu tên tài khoản đã có người sử dụng: hiện label màu đỏ dưới nút [Đăng ký] với nội dung: “Tên tài khoản đã tồn tại”.
* Khi đăng ký thành công: hệ thống sẽ chuyển người dùng tới trang đăng nhập, tại đây người dùng có thể đăng nhập bằng tài khoản mình vừa đăng ký.
* Mật khẩu sau khi đăng kí sẽ được mã hóa bằng MD5 (là 1 hàm mã hóa 1 chiều của C#) và lưu mật khẩu sau khi mã hóa vào CSDL.
* Chức năng: [Đã có tài khoản?Đăng nhập!]
* Khi nhấn vào nút [Đã có tài khoản?Đăng nhập!] hệ thống sẽ chuyển về trang đăng nhập, tại đây người dùng có thể đăng nhập vào hệ thống với quyền quản trị viên, nhân viên hoặc quyền khách hàng.
  + 1. Chức năng Đăng nhập



Hình 3.3 Màn hình đăng nhập

Chức năng [Đăng nhập]: cho phép đăng nhập tài khoản vào hệ thống tùy theo quyền hạn của tài khoản.

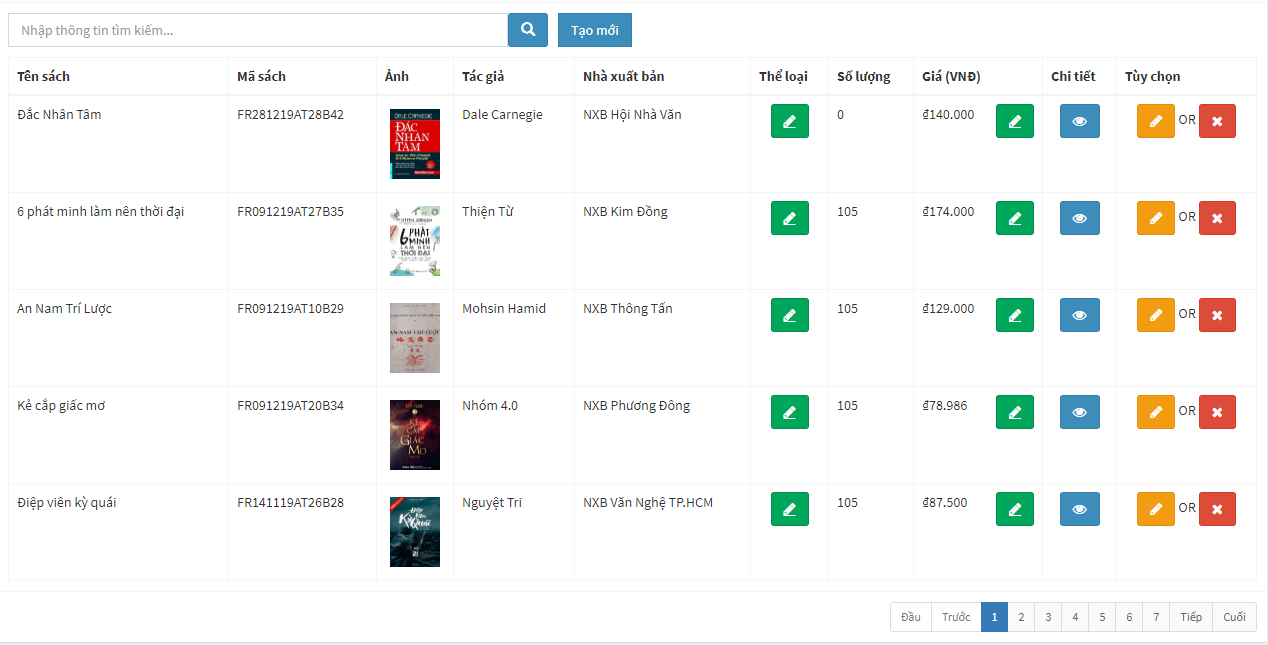
Validate:

* Tên đăng nhập: length(50), khác rỗng
* Mật khẩu: kiểu mật khẩu(“\*\*\*\*\*”), length(32), khác rỗng
* Chức năng: Người dùng nhập thông tin vào các ô nhập liệu và nhấn nút [Đăng nhập]
* Nếu có trường không hợp lệ về validate: hiện label màu đỏ ngay dưới ô nhập liệu không hợp lệ.
* Nếu [Tên đăng nhập] không tồn tại: hiện label màu đỏ dưới nút [Đăng ký] với nội dung: “Tên đăng nhập không tồn tại!”.
* Nếu [Tên đăng nhập] tồn tại nhưng mật khẩu sai, hiện label màu đỏ dưới nút [Đăng nhập] với nội dung: “Mật khẩu không khớp!”
* Nếu [Đăng nhập] thành công, chuyển người dùng đến trang tương ứng với quyền hạn của tài khoản(vd: với tài khoản có quyền hạn là admin hoặc nhân viên thì sẽ chuyển tới trang quản trị và hiển thị tên người đăng nhập, quyền hạn là khách hàng, sẽ chuyển tới trang khách hàng, tên khách được hiển thị).
* Khi đăng nhập, người dùng sẽ nhập mật khẩu và phần mật khẩu sẽ được mã hóa (MD5) rồi kiểm tra với CSDL.

Chức năng: [Tạo tài khoản]

Khi nhấn vào nút [Tạo tài khoản!] hệ thống sẽ chuyển về trang đăng ký, tại đây người dùng có thể đăng ký tài khoản.

* + 1. Chức năng Quản lý sách



Hình 3.4 Màn hình quản lý sách

Tiền điều kiện: đã đăng nhập với tài khoản admin hoặc nhân viên và đang ở View Quản lý sách

Chức năng **[Tìm kiếm]**: cho phép người dùng tìm kiếm chính xác theo: mã, tìm kiếm gần đúng theo: tên sách, tên tác giả, tên nhà xuất bản



Hình 3.5 Ảnh minh họa thanh tìm kiếm

Validate:

* Chuỗi tìm kiếm: minlenght(0), maxlength(250)
* Chức năng: Người dùng nhập thông tin vào ô tìm kiếm và nhấn nút **[Tìm kiếm]**
* Nếu ô tìm kiếm không hợp lệ về validate: hiện label màu đỏ ngay dưới ô tìm kiếm.
* Nếu chuỗi tìm kiếm thỏa mãn tìm kiếm gần đúng của 1 trong 3 thuộc tính: tên sách, tên tác giả, tên nhà xuất bản hoặc tìm kiếm chính xác: mã sách, thì hiển thị các đầu sách tương ứng dưới dạng table, kết quả có thể có nhiều bản ghi, nếu trên 5 bản ghi sẽ hiển thị phân trang.
* Nếu chuỗi tìm kiếm không thỏa mãn điều kiện tìm kiếm nào, hiển thị kết quả dưới dạng 1 table rỗng.

Kết hợp kiểm thử chức năng tìm kiếm với phân trang.

Chức năng: **[Phân trang]** (kết hợp kiểm tra với chức năng tìm kiếm): cho phép hiển thị danh sách đầu sách dưới dạng các trang khi số bản ghi vượt quá 5 (mỗi trang sẽ hiển thị 5 bản ghi, nếu vượt quá 5 sẽ phân trang dữ liệu).



Hình 3.6 Ảnh minh họa thanh phân trang

TH1: số bản ghi < 5 bản ghi(tức có 1 trang): chỉ hiển thị số trang: 1 dưới thanh phân trang, các nút “Đầu”, “Trước”, “Tiếp”, “Cuối” bị vô hiệu hóa

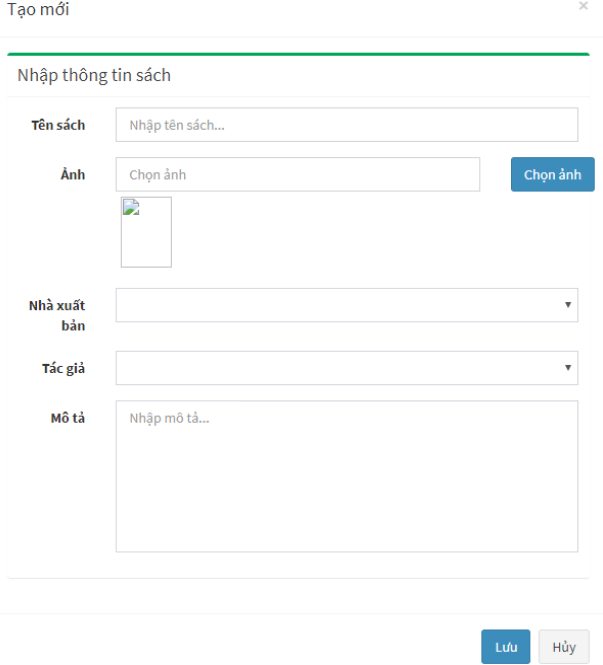
TH2: số bản ghi > 5 bản ghi(tức có nhiều trang): số trang tối đa sẽ hiển thị là: (số bản ghi/số bản ghi 1 trang) lấy phần nguyên + 1.

Tại trang 1: nút “Đầu”, “Trước” bị vô hiệu hóa, các nút khác trên thanh phân trang sử dụng bình thường, khi chọn “Tiếp” sẽ chuyển tới trang 2, khi chọn “Cuối” sẽ chuyển tới trang cuối cùng(tức số trang tối đa).

Tại trang n > 1: tất cả các nút tên thanh phân trang sử dụng bình thường, khi nhấn nút “Đầu” sẽ chuyển về trang 1, khi chọn “trước” sẽ chuyển về trang n-1, khi chọn “tiếp” sẽ chuyển tới trang n+1, khi chọn “cuối” sẽ chuyển tới trang cuối cùng.

Tại trang cuối cùng: nút “tiếp” và “cuối” bị vô hiệu hóa, khi chọn “trước” sẽ về trang n-1(tức trang trước trang cuối), khi chọn “đầu” sẽ về trang đầu tiên.

Chức năng **[Tạo mới]:** cho phép thêm sách



Hình 3.7 Ảnh minh hoa thêm sách

Validate:

* Tên sách: not null, maxlenght(250)
* Ảnh: not null, maxlenght(250)
* Nhà xuất bản: combobox, notnull
* Tác giả: combobox, notnull
* Mô tả: nvarchar(max), notnull

Chức năng:

Người dùng chọn chức năng **[Tạo mới]**, một popup hiện ra như trên hình 3.6, người dùng nhập/chọn thông tin vào các ô nhập liệu và nhấn nút **[Lưu]**

Nếu có trường không hợp lệ về validate: focus vào ô nhập liệu và hiển thị border màu đỏ tại ô nhập liệu đó.

Nếu thêm mới thất bại, hiển thị 1 alert: Thêm sách thất bại

Nếu thêm mới thành công:

Lưu sách vào cơ sở dữ liệu, mã sách tự tăng, ngày tạo: ngày hiện tại, người tạo: lấy thông tin của tài khoản đang đăng nhập, metatile được tạo tự động (vd: tên sách là “Ảnh hậu tái sinh” thì metatile là: “anh-hau-tai-sinh”), giá sách = 0, khuyến mãi = 0%

1 giá mới được tạo ra và lưu kèm với sách.

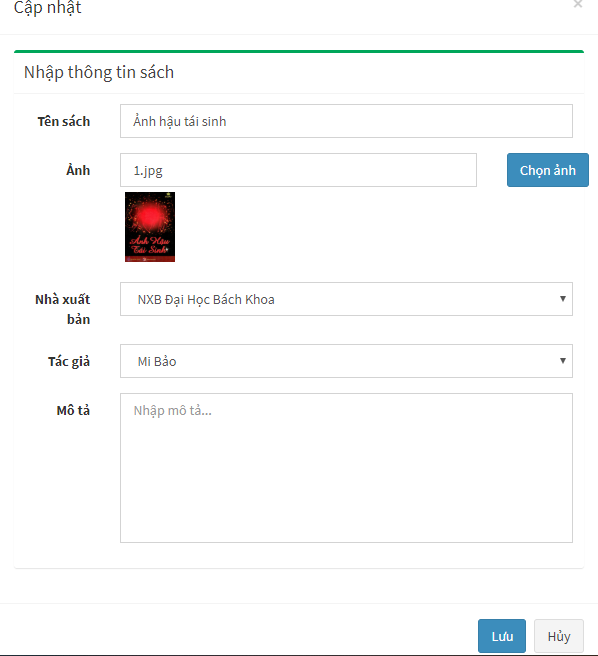
Mã sách được tạo theo công thức: “FR”+ ngày + tháng + năm(tạo)+ “AT” + mã tác giả + “B” + mã sách.

Sách mới tạo sẽ không thuộc thể loại nào.

Hiển thị alert: Thêm sách thành công và load lại body của table(không load lại cả trang)

Khi người dùng nhấn dấu x, nút [Hủy] hoặc nhấn ra ngoài popup, popup biến mất, không có thay đổi trong CSDL.

Chức năng **[Sửa sách]:** cho phép sửa sách



Hình 3.8 Ảnh minh họa chức năng sửa sách

Validate:

* Tên sách: not null, maxlenght(250)
* Ảnh: not null, maxlenght(250)
* Nhà xuất bản: combobox, notnull
* Tác giả: combobox, notnull
* Mô tả: nvarchar(max), notnull

Chức năng:

Người dùng chọn chức năng **[Sửa sách]**, một popup hiện ra như trên hình 3.5, thông tin về sách được hiển thị trên các trường, người dùng nhập/chọn thông tin vào các ô nhập liệu và nhấn nút **[Lưu]**

Nếu có trường không hợp lệ về validate: focus vào ô nhập liệu và hiển thị border màu đỏ tại ô nhập liệu đó.

Nếu thêm mới thất bại, hiển thị 1 alert: Sửa sách thất bại

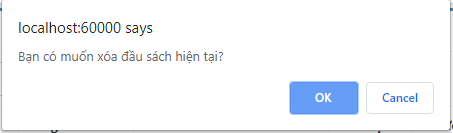
Nếu thêm mới thành công:

Lưu sách vào cơ sở dữ liệu, ngày sửa: ngày hiện tại, người sửa: lấy thông tin của tài khoản đang đăng nhập, metatile được tạo tự động (vd: tên sách là “Ảnh hậu tái sinh” thì metatile là: “anh-hau-tai-sinh”)

Hiển thị alert: Sửa sách thành công và load lại body của table(không load lại cả trang)

Khi người dùng nhấn dấu x, nút [Hủy] hoặc nhấn ra ngoài popup, popup biến mất, không có thay đổi trong CSDL.

Chức năng **[Xóa sách]:** cho phép xóa sách



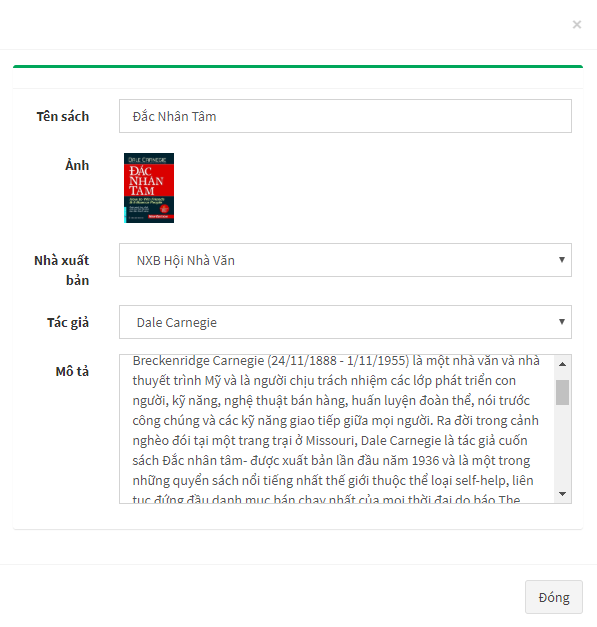
Hình 3.9 Ảnh minh họa chức năng xóa sách

Chức năng: người dùng chọn đầu sách muốn xóa và nhấn vào dấu x, 1 alert hiện ra với nội dung: Bạn có muốn xóa đầu sách hiện tại?

Người dùng chọn [OK], alert biến mất, hệ thống xóa đầu sách khỏi CSDL và load lại body của table

Người dùng chọn [Cancel], alert biến mất, không có đầu sách nào bị xóa khỏi CSDL

Chức năng **[Xem chi tiết]:** cho phép xem chi tiết đầu sách

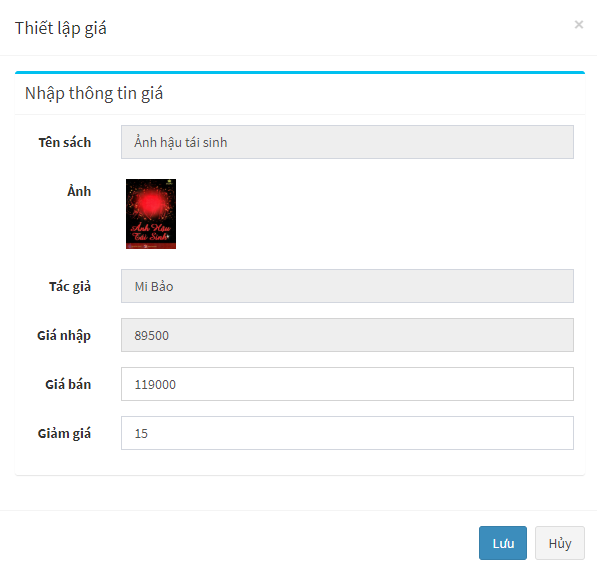


Hình 3.10 Ảnh minh họa chức năng xem chi tiết sách

Người dùng chọn đầu sách muốn xóa và nhấn vào xem chi tiết, 1 popup hiện ra với thông tin sách đã chọn, các trường đều bị vô hiệu hóa.

Người dùng chọn [Đóng], popup biến mất, không có thay đổi gì trong CSDL

Chức năng **[Cập nhật giá]:** cho phép cập nhật giá



Hình 3.11 Ảnh minh họa chức năng cập nhật giá

Validate:

* Giá bán: double
* Giảm giá: int, min: 0; max: 100

Chức năng:

Người dùng chọn chức năng **[Cập nhật giá]**, một popup hiện ra như trên hình 3.11, thông tin về sách được hiển thị trên các trường, các trường tên sách, tác giả, giá nhập bị vô hiệu hóa, người dùng nhập/chọn thông tin vào các ô nhập liệu và nhấn nút **[Lưu]**

Nếu có trường không hợp lệ về validate: focus vào ô nhập liệu và hiển thị border màu đỏ tại ô nhập liệu đó.

Nếu cập nhật thất bại, hiển thị 1 alert: Cập nhật giá thất bại

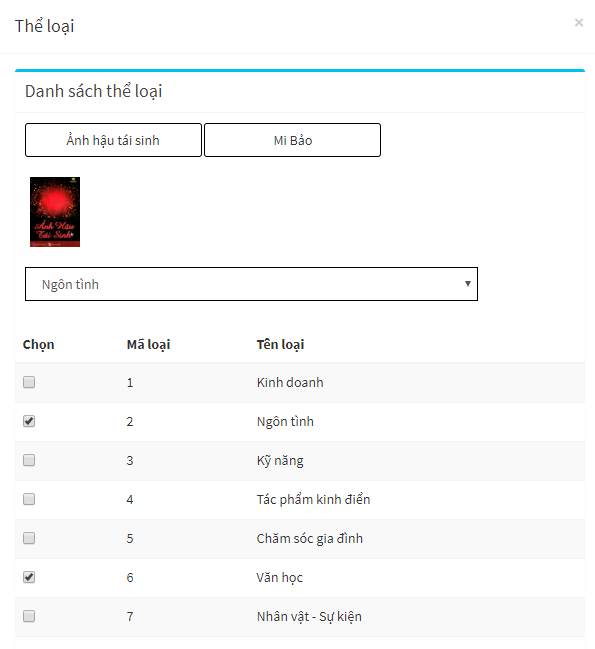
Nếu thêm mới thành công:

Cập nhật sách vào cơ sở dữ liệu, ngày sửa: ngày hiện tại, người sửa: lấy thông tin của tài khoản đang đăng nhập, giảm giá = trường giảm giá đã nhập vào.

Hiển thị alert: “Cập nhật giá thành công” và load lại body của table(không load lại cả trang)

Khi người dùng nhấn dấu x, nút [Hủy] hoặc nhấn ra ngoài popup, popup biến mất, không có thay đổi trong CSDL.

Chức năng **[Cập nhật thể loại]:** cho phép cập nhật thể loại cho mỗi đầu sách



Hình 3.12 Ảnh minh họa chức năng cập nhật thể loại

Người dùng chọn chức năng **[Cập nhật thể loại]**, một popup hiện ra như trên hình 3.8, thông tin về sách được hiển thị trên các trường, các thể loại của sách được tích vào các dòng tương ứng, combobox chứa các thể loại của đầu sách đó, các trường tên sách, tác giả bị vô hiệu hóa

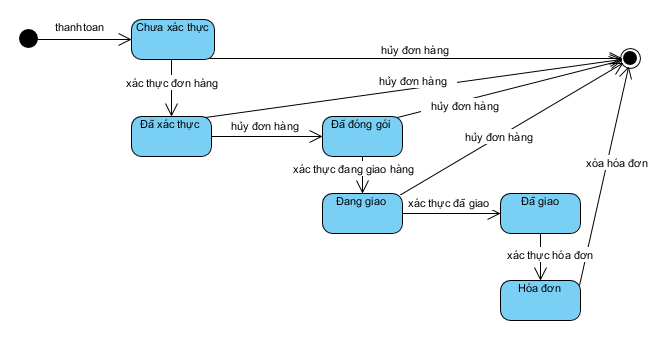
Người dùng tích vào trường muốn chọn, thông tin về thể loại đó được thêm vào combobox phía trên.

Người dùng bỏ tích vào 1 trường, thông tin về trường đó bị xóa khỏi combobox.

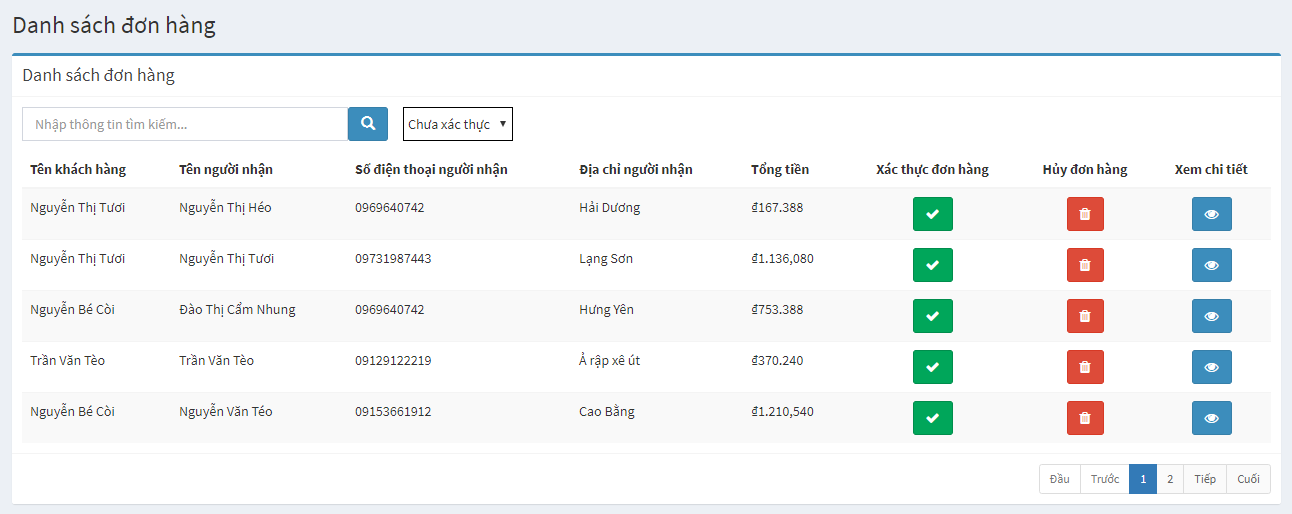
Người dùng nhấn [Lưu], lưu thể loại vào bảng sách thể loại, nếu thành công, hiển thị alert: Cập nhật thể loại thành công và load lại body của table, ngược lại hiển thị alert: Cập nhật thể loại thất bại và load lại trang đồng thời đóng popup.

Nếu người dùng dấu x, nút [Hủy] hoặc nhấn ra ngoài popup, popup biến mất, không có gì thay đổi trong CSDL.

* + 1. Chức năng quản lý đơn hàng



Hình 3.13 Biểu đồ trạng thái của đơn hàng



Hình 3.14 Màn hình quản lý đơn hàng

Các chức năng: [Tìm kiếm], [Xác thực đơn hàng], [Xác thực đã đóng gói], [Xác thực đang giao], [Xác thực đã giao], [Xác thực hóa đơn], [Hủy đơn hàng], [Xem chi tiết], [Xóa hóa đơn], [Chức năng phân trang], [Xem đơn hàng theo trạng thái].

Tiền điều kiện : đã đăng nhập với tài khoản admin hoặc nhân viên và đang ở View Quản lý đơn hàng

Chức năng **[Tìm kiếm]**: cho phép người dùng tìm kiếm gần đúng theo: tên khách hàng, tên người nhận, địa chỉ người nhận; tìm kiếm chính xác theo: số điện thoại người nhận



Hình 3.15 Ảnh minh họa thanh tìm kiếm

Validate:

* Chuỗi tìm kiếm: minlenght(0), maxlength(250)
* Chức năng: Người dùng nhập thông tin vào ô tìm kiếm và nhấn nút **[Tìm kiếm]**
* Nếu ô tìm kiếm không hợp lệ về validate: hiện label màu đỏ ngay dưới ô tìm kiếm.
* Nếu chuỗi tìm kiếm thỏa mãn tìm kiếm gần đúng của 1 trong 3 thuộc tính: tên sách, tên tác giả, tên nhà xuất bản hoặc tìm kiếm chính xác: số điện thoại, thì hiển thị các đầu sách tương ứng dưới dạng table, kết quả có thể có nhiều bản ghi, nếu trên 5 bản ghi sẽ hiển thị phân trang.
* Nếu chuỗi tìm kiếm không thỏa mãn điều kiện tìm kiếm nào, hiển thị kết quả dưới dạng 1 table rỗng.
* Kết hợp kiểm thử chức năng tìm kiếm với phân trang dữ liệu.

Chức năng: **[Phân trang]** (kết hợp kiểm tra với chức năng tìm kiếm): cho phép hiển thị danh sách đầu sách dưới dạng các trang khi số bản ghi vượt quá 5 (mỗi trang sẽ hiển thị 5 bản ghi, nếu vượt quá 5 sẽ phân trang dữ liệu).



Hình 3.16 Ảnh minh họa thanh phân trang

TH1: số bản ghi < 5 bản ghi(tức có 1 trang):

chỉ hiển thị số trang: 1 dưới thanh phân trang, các nút “Đầu”, “Trước”, “Tiếp”, “Cuối” bị vô hiệu hóa

TH2: số bản ghi > 5 bản ghi(tức có nhiều trang):

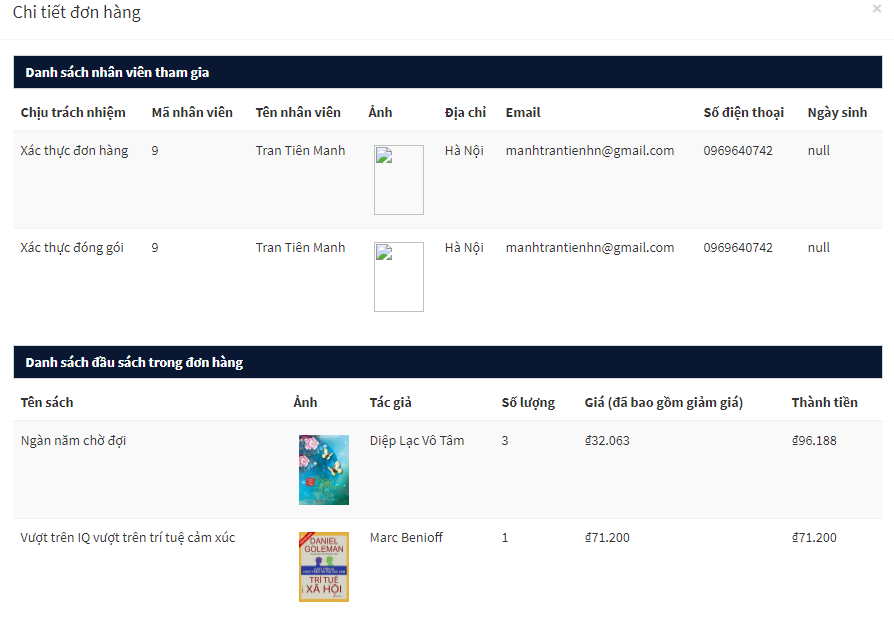
số trang tối đa sẽ hiển thị là: (số bản ghi/số bản ghi 1 trang) lấy phần nguyên + 1.

Tại trang 1: nút “Đầu”, “Trước” bị vô hiệu hóa, các nút khác trên thanh phân trang sử dụng bình thường, khi chọn “Tiếp” sẽ chuyển tới trang 2, khi chọn “Cuối” sẽ chuyển tới trang cuối cùng(tức số trang tối đa).

Tại trang n > 1: tất cả các nút tên thanh phân trang sử dụng bình thường, khi nhấn nút “Đầu” sẽ chuyển về trang 1, khi chọn “trước” sẽ chuyển về trang n-1, khi chọn “tiếp” sẽ chuyển tới trang n+1, khi chọn “cuối” sẽ chuyển tới trang cuối cùng.

Tại trang cuối cùng: nút “tiếp” và “cuối” bị vô hiệu hóa, khi chọn “trước” sẽ về trang n-1(tức trang trước trang cuối), khi chọn “đầu” sẽ về trang đầu tiên.

Chức năng [Xem chi tiết]:

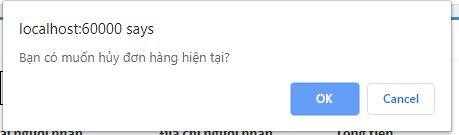


Hình 3.17 Màn hình xem chi tiết đơn hàng

Người dùng chọn đơn hàng/hóa đơn muốn xem và nhấn vào xem chi tiết, 1 popup hiện ra với thông tin chi tiết về đơn hàng đã chọn gồm 2 table, table 1: thông tin các nhân viên liên quan, table 2: thông tin các đầu sách đã mua của khách hàng.

Người dùng chọn [Đóng], bấm dấu x hoặc nhấn ra ngoài table, popup biến mất, không có thay đổi gì trong CSDL

Chức năng [Hủy đơn hàng], [Xóa hóa đơn]



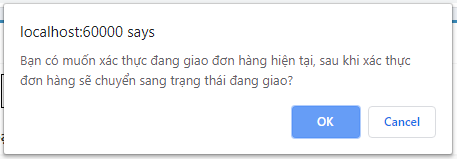
Hình 3.18 Ảnh minh họa chức năng hủy đơn hàng và xóa hóa đơn

Chức năng: người dùng chọn đơn hàng/hóa đơn muốn hủy/xóa và nhấn vào dấu biểu tượng thùng rác, 1 alert hiện ra với nội dung: Bạn có hủy/xóa đơn hàng/hóa đơn hiện tại?

Người dùng chọn [OK], alert biến mất, hệ thống xóa đơn hàng/hóa đơn khỏi CSDL và load lại trang

Người dùng chọn [Cancel], alert biến mất, không có đơn hàng/hóa đơn nào bị xóa khỏi CSDL

Chức năng [Xác thực]: đơn hàng, đóng gói, đang giao, đã giao, hóa đơn



Hình 3.19 Ảnh minh họa chức năng xác thực

Chức năng: người dùng chọn đơn hàng xác thực và nhấn vào biểu tượng chữ v, 1 alert hiện ra với tùy vào trạng thái của đơn hàng:

Đơn hàng chưa xác thực: “Bạn có muốn xác thực đơn hàng hiện tại, sau khi xác thực đơn hàng sẽ chuyển sang trạng thái đã xác thực?”

Đơn hàng đã xác thực: “Bạn có muốn xác thực đã đóng gói đơn hàng hiện tại, sau khi xác thực đơn hàng sẽ chuyển sang trạng thái đã đóng gói?

Đơn hàng đã đóng gói: “Bạn có muốn xác thực đang giao đơn hàng hiện tại, sau khi xác thực đơn hàng sẽ chuyển sang trạng thái đang giao?”

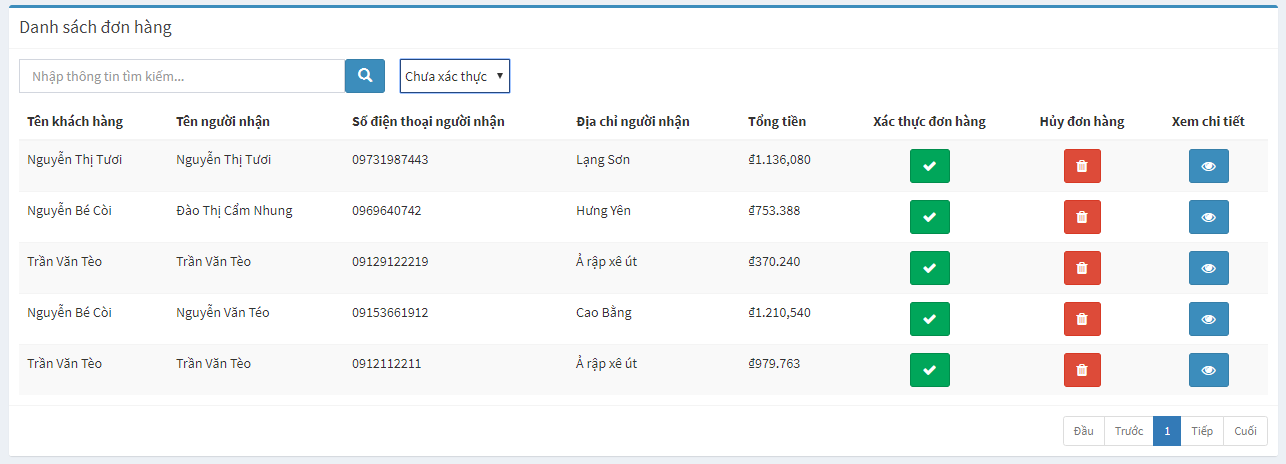
Đơn hàng đang giao: “Bạn có muốn xác thực đã giao đơn hàng hiện tại, sau khi xác thực đơn hàng sẽ chuyển sang trạng thái đã đã giao?”

Đơn hàng đã giao: “Bạn có muốn xác thực đã nhận tiền đơn hàng hiện tại, sau khi xác thực đơn hàng sẽ chuyển sang trạng thái hóa đơn?”

Người dùng chọn [OK], alert biến mất, đơn hàng được cập nhật vào CSDL: chưa xác thực -> đã xác thực ->đã đóng gói -> đang giao -> đã giao -> hóa đơn.

Người dùng chọn [Cancel], alert biến mất, không có gì thay đổi trong CSDL.

Chức năng [Xem theo trạng thái]: cho phép xem đơn hàng theo trạng thái



Hình 3.20 Ảnh minh họa chức năng xem theo trạng thái

Chức năng:

Người dùng chọn trạng thái cần xem:

Chưa xác thực: tất cả đơn hàng chưa xác thực sẽ hiển thị, cột tiêu đề xác thực là “Xác thực đơn hàng”

Đã xác thực: tất cả đơn hàng đã xác thực sẽ hiển thị, cột tiêu đề xác thực là “Xác thực đóng gói”

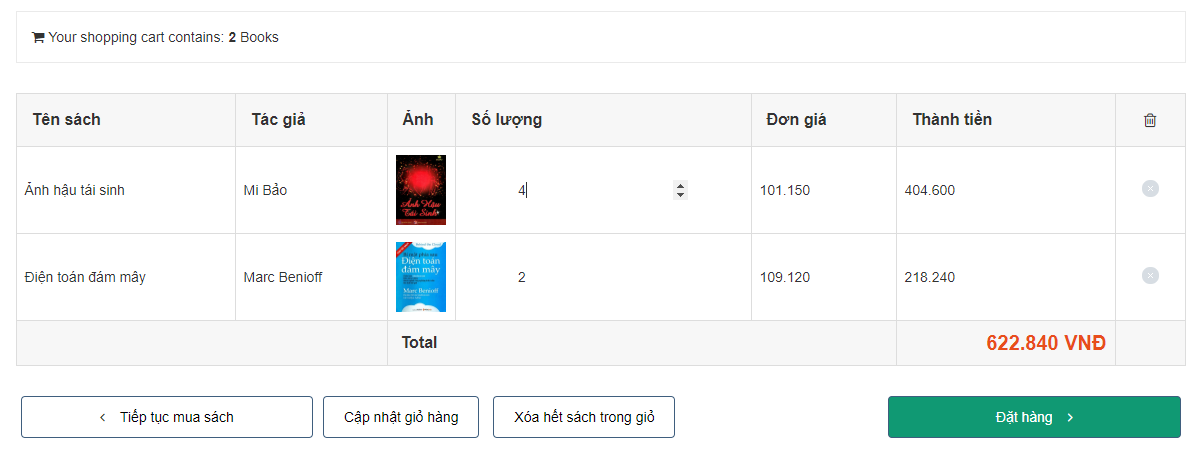
Đã đóng gói: tất cả đơn hàng đã đóng gói sẽ hiển thị, cột tiêu đề xác thực là “Xác thực đang giao hàng”

Đang giao: tất cả đơn hàng đang giao sẽ hiển thị, cột tiêu đề xác thực là “Xác thực đã giao hàng”

Đã giao: tất cả đơn hàng đã giao sẽ hiển thị, cột tiêu đề xác thực là “Xác thực hóa đơn”

Kết hợp kiểm thử xem theo trạng thái với phân trang dữ liệu

* + 1. Chức năng đặt hàng



Hình 3.21 Minh họa chức năng đặt hàng

Tiền điều kiện : đã đăng nhập với tài khoản khách hàng

Người dùng chọn đầu sách cần mua và nhấn thêm vào giỏ:

* Nếu trong giỏ chưa có sách: đầu sách đã được thêm vào giỏ với số lượng là 1
* Nếu trong giỏ đã có sách:
* Nếu đầu sách đã chọn trùng với đầu sách đã có trong giỏ: số lượng đầu sách trong giỏ sẽ tăng thêm 1
* Ngược lại: thêm 1 đầu sách vào giỏ với số lượng là 1

Tại giỏ hàng: người dùng có thể: cập nhật giỏ hàng, xóa hết sách trong giỏ hàng, xóa 1 đầu sách chỉ định trong giỏ, tiếp tục mua sách, đặt hàng

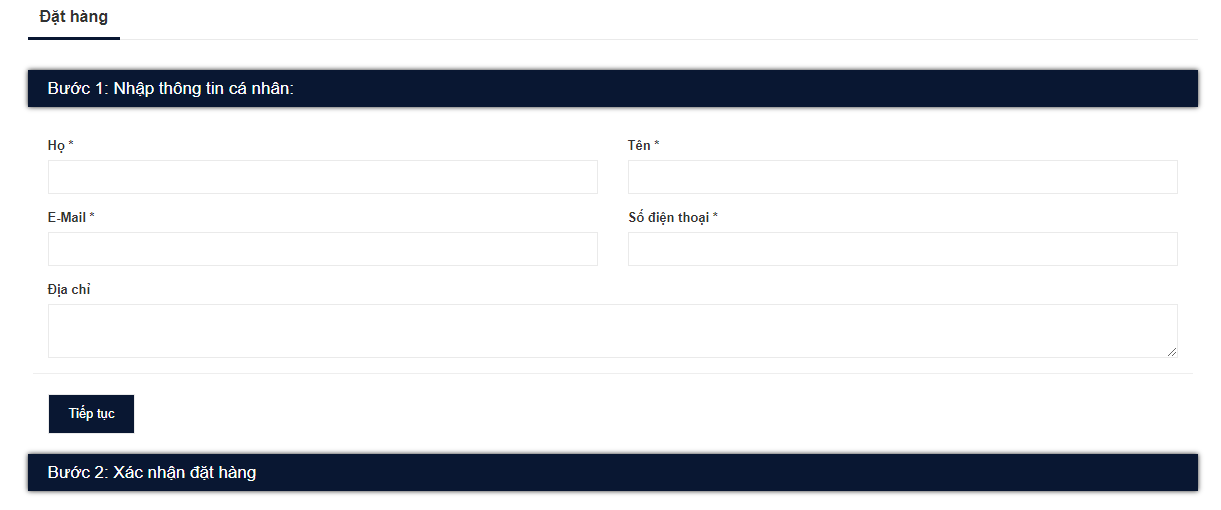
Cập nhật giỏ hàng: người dùng có thể nhập số lượng sách thay đổi cho mỗi đầu sách vào các ô text (kiểu number -> dùng cho validate) và bấm vào cập nhật giỏ hàng, số lượng đầu sách trong giỏ sẽ thay đổi tương ứng.

Xóa hết sách trong giỏ hàng: khi chọn xóa hết sách trong giỏ thành công hiện 1 label chưa có đầu sách nào trong giỏ.

Xóa chỉ định 1 đầu sách trong giỏ: người dùng chọn đầu sách muốn xóa khỏi giỏ và bấm vào dấu x, đầu sách đã bị xóa khỏi giỏ

Tiếp tục mua hàng: sau khi bấm vào, màn hình trở về trang chủ

Đặt hàng: sau khi chọn thanh toán, màn hình sẽ chuyển tới trang thanh toán



Hình 3.22 Hình minh họa chức năng thanh toán

Chức năng thanh toán:

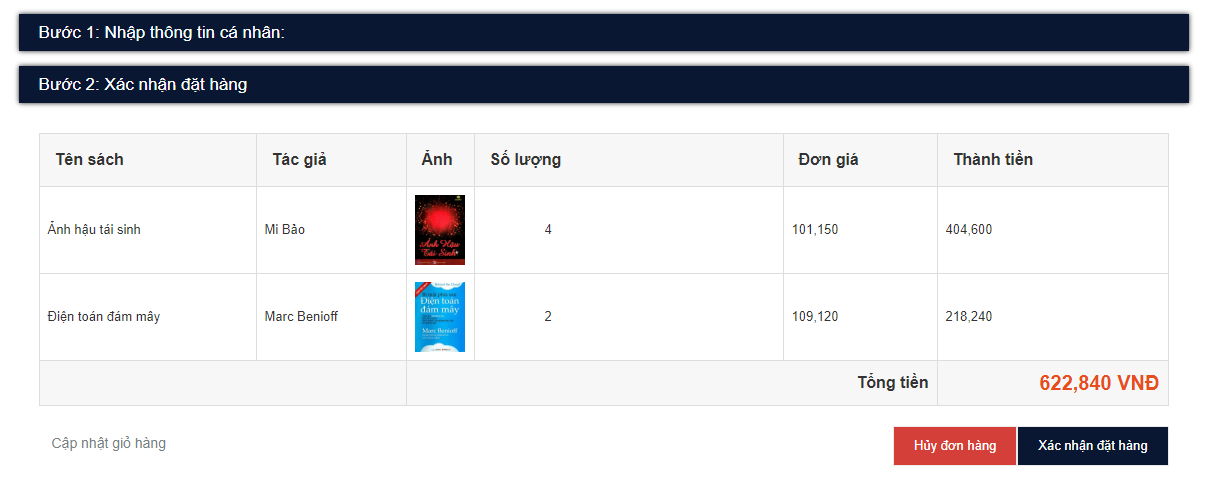
Validate:

* Họ, tên: length(50), khác rỗng
* Số điện thoại: length(15), kiểu number, khác rỗng
* Địa chỉ: length(500), khác rỗng
* Email: length(50), kiểu email (“xyz@gmail.com”)

Chức năng: Người dùng nhập các thông tin vào các trường nhập liệu

Nếu có trường không hợp lệ về validate, focus tại trường đó và viền đỏ xuất hiện

Ngược lại, người dùng bấm tiếp tục, sẽ chuyển xuống bảng thanh toán



Hình 3.23 Ảnh minh họa bảng thanh toán

Tại đây, người dùng có thể: cập nhật lại giỏ hàng, hủy đơn hàng, xác nhận thanh toán

Hủy đơn hàng: người dùng bấm hủy đơn hàng, màn hình trở về trang chủ, các đầu sách trog giỏ bị xóa hết

Cập nhật giỏ hàng: người dùng có thể nhập số lượng sách thay đổi cho mỗi đầu sách vào các ô text (kiểu number -> dùng cho validate) và bấm vào cập nhật giỏ hàng, số lượng đầu sách trong giỏ sẽ thay đổi tương ứng.

Xác nhận thanh toán: sau khi bấm xác nhận thanh toán, người dùng sẽ nhận được 1 email về danh sách các đầu sách trong đơn hàng của mình và hiện 1 label: Thanh toán thành công, nếu thất bại sẽ hiện 1 label: thanh toán thất bại

* 1. Các yêu cầu phi chức năng
     1. Tính ứng dụng

Hệ thống cần đáp ứng được cho 10 đến 20 user cùng sử dụng.

Thiết kế website cần đơn giản, phù hợp với các chuẩn màn hình thông dụng và thói quen sử dụng.

Có thể chạy trên các trình duyệt google chrome, internet explorer, microsoft edge

* + 1. Tính toàn vẹn dữ liệu/ perfomcance
* Tính toàn vẹn dữ liệu

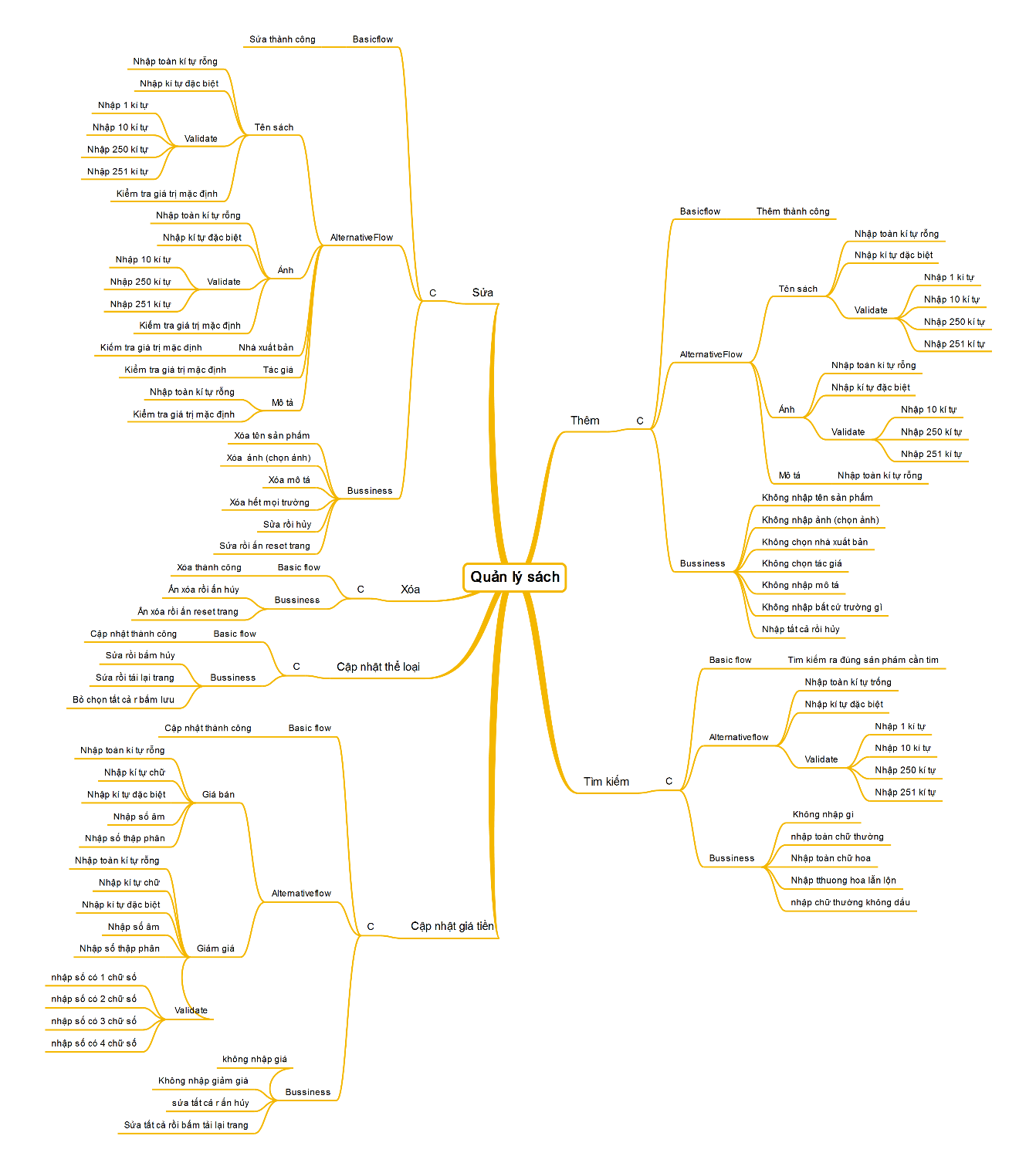
Khi có nhiều hơn 1 user cùng thay đổi 1 data trong DB thì chỉ user đầu tiên thực hiện được bình thường. Đối với các user khác sẽ không lưu lại thay đổi mà hiển thị thông báo và yêu cầu thực hiện lại xử lý.

* Performcance

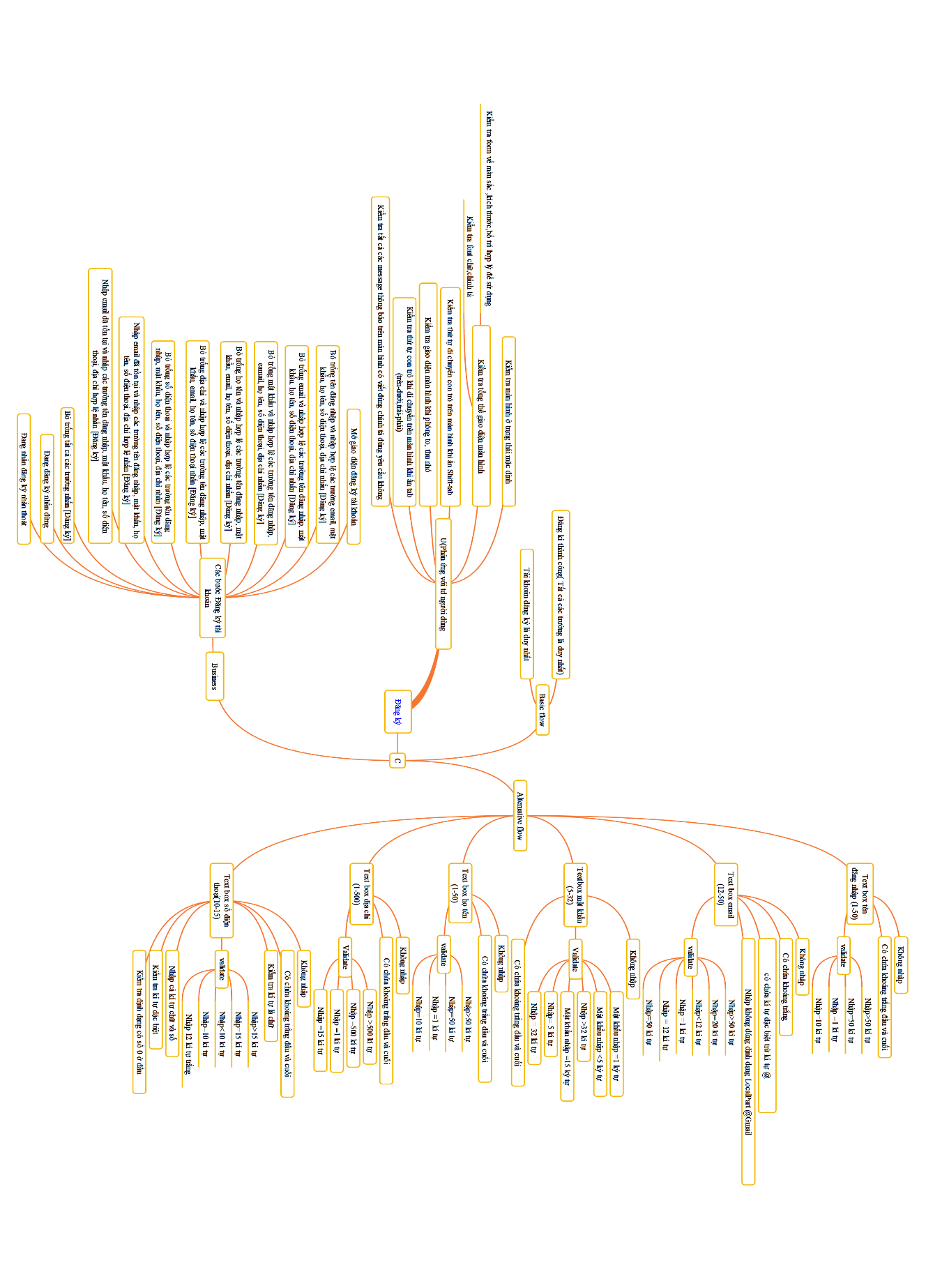
Yêu cầu thời gian mở website con bất kỳ không được chậm hơn 3 giây, và xử lý lưu thông tin không được chậm hơn 4 giây.

# TRIỂN KHAI KIỂM THỬ TỰ ĐỘNG

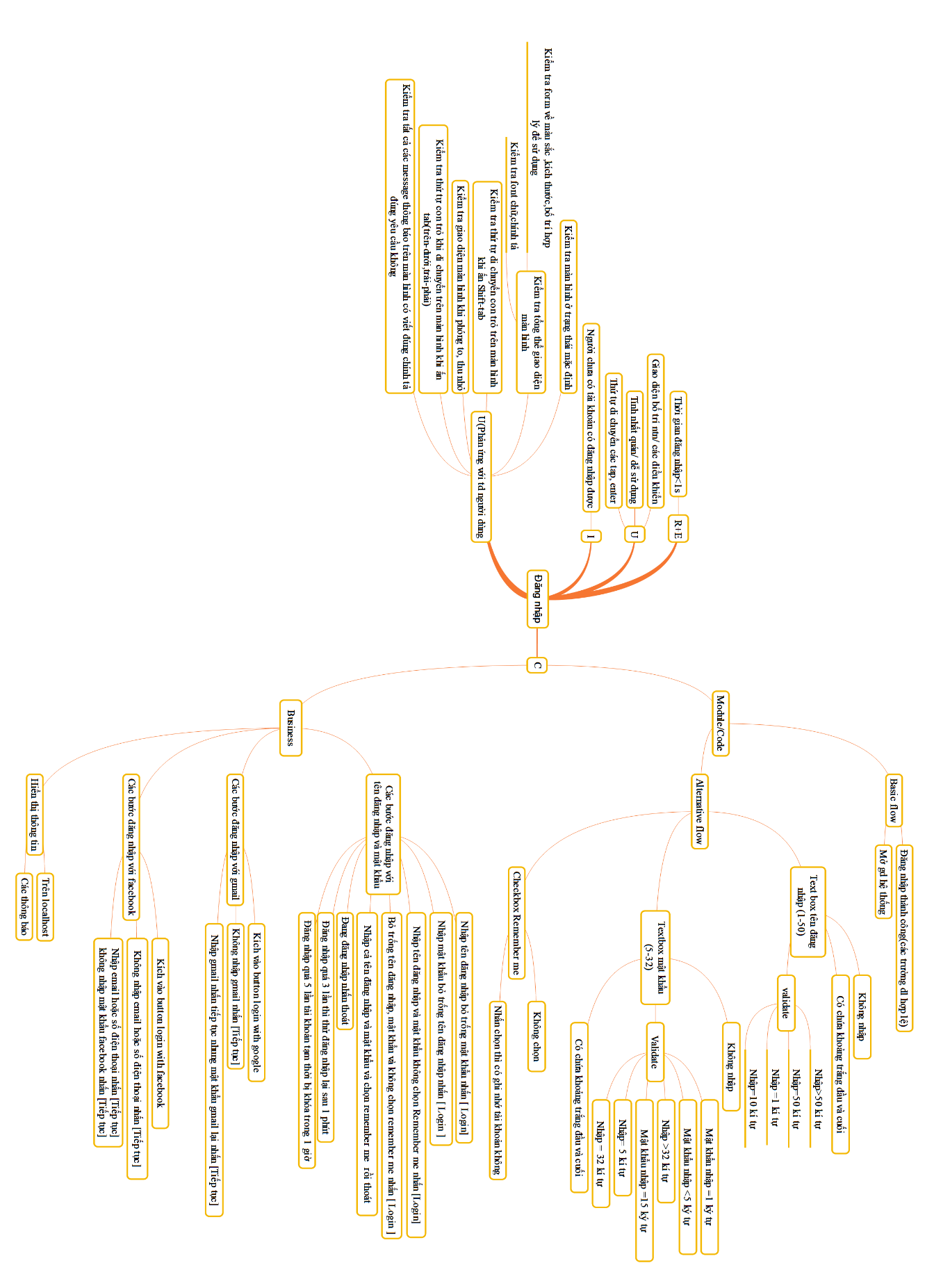
* 1. Thiết kế các yêu cầu kiểm thử



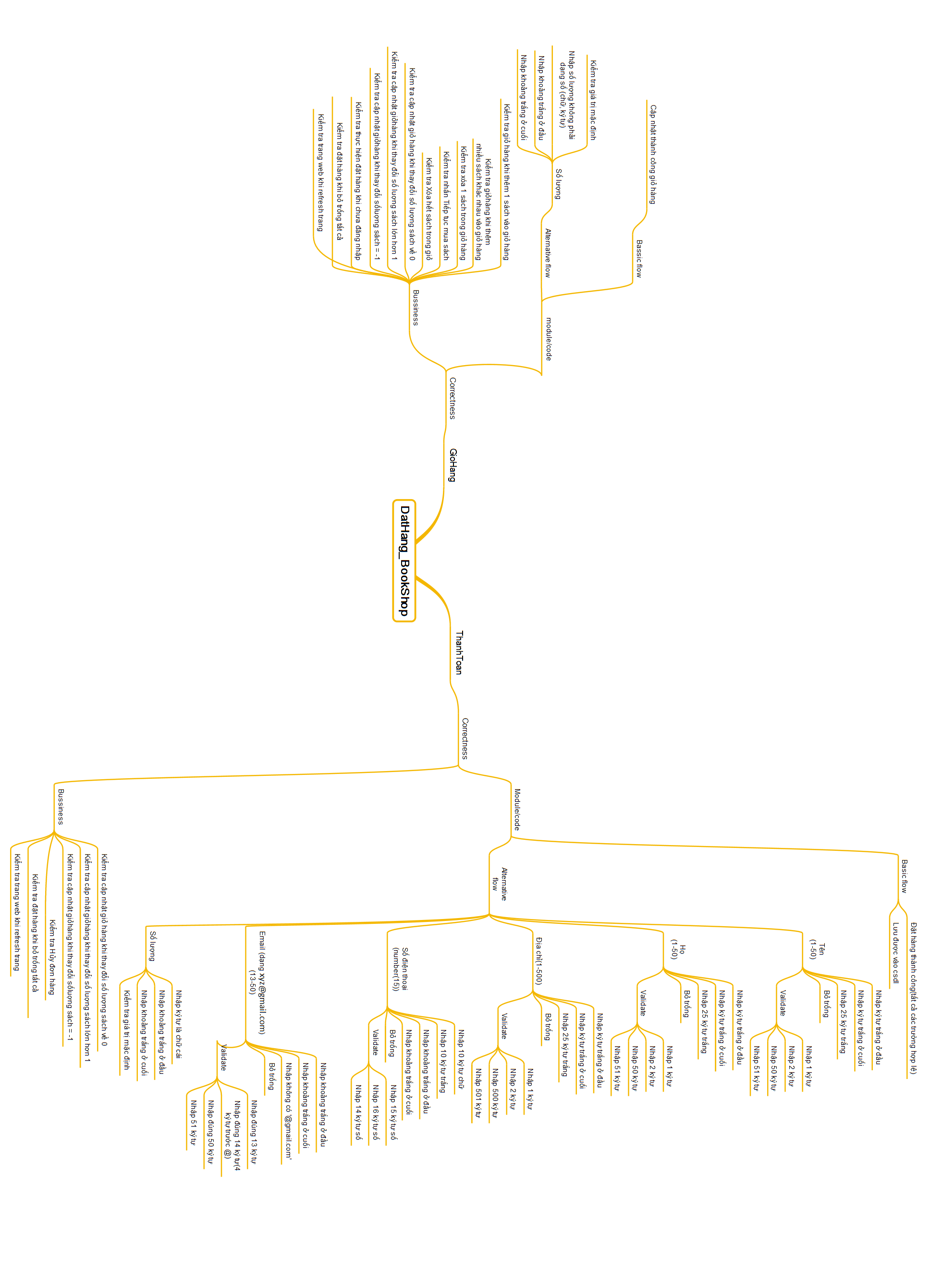
Hình 4.1 Test design Quản lý sách



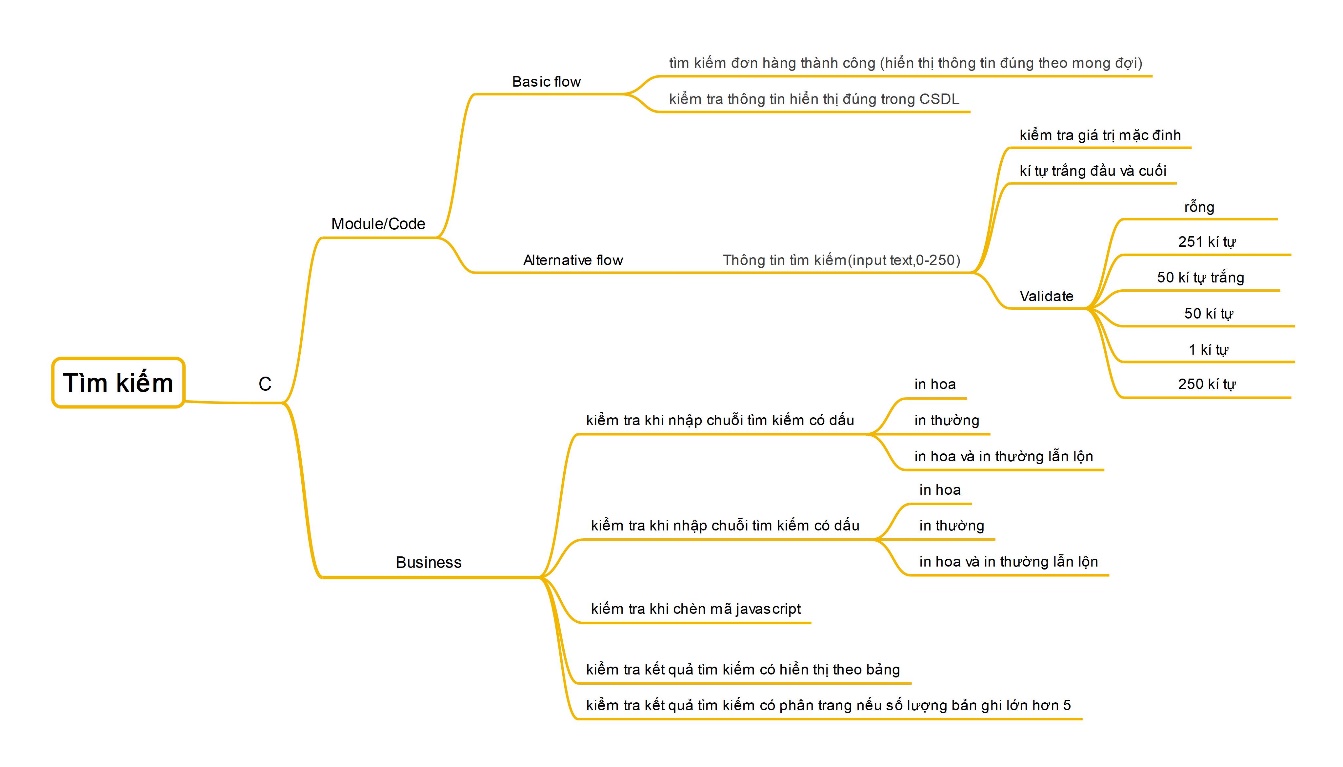
Hình 4.2 Test design Đăng ký



Hình 4.3 Test design Đăng nhập

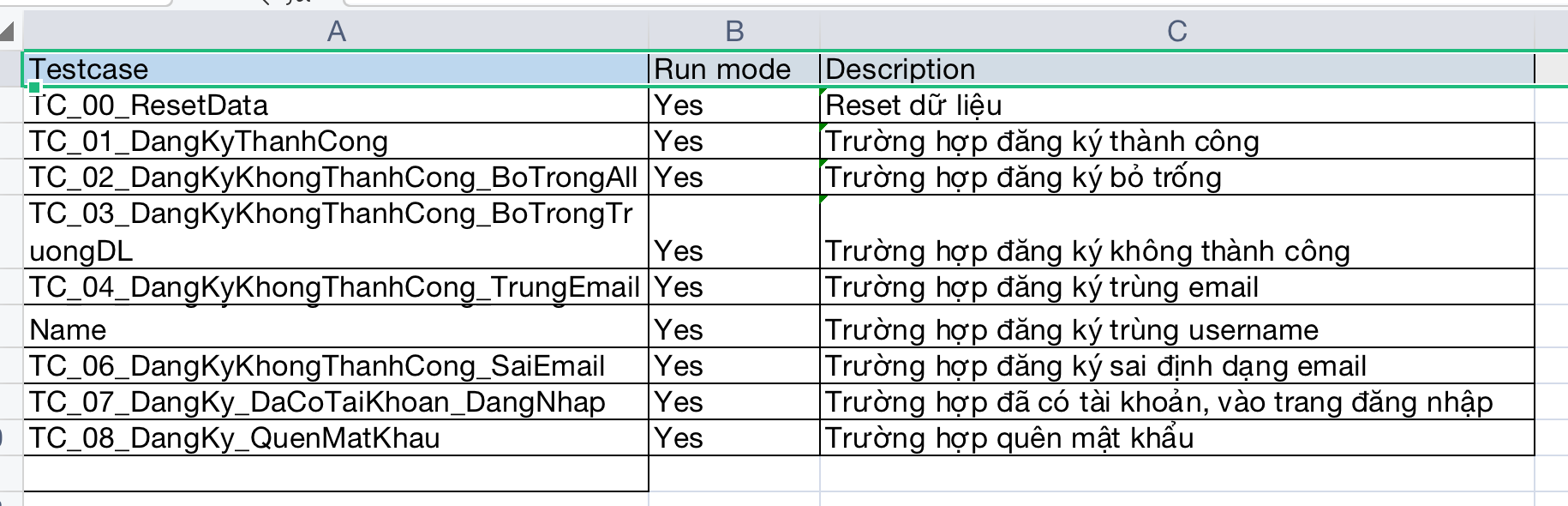


Hình 4.4 Test design Đặt hàng



Hình 4.5 Test design Tìm kiếm

* 1. Xây dựng ca kiểm thử















* 1. Xây dựng dữ liệu kiểm thử

Bảng : Dữ liệu Testcase đăng nhập



Bảng : Dữ liệu Testcase đăng ký

|  |  |
| --- | --- |
| Name | Value |
| browser | chorme |
| url | http://113.160.133.144:20201/ |
| TC01\_fullName | Nguyen Quoc Bao |
| TC01\_email | nguyenquocbao.1999443@gmail.com |
| TC01\_numberPhone | 0334218638 |
| TC01\_userName | baobao |
| TC01\_password | baobao |
| TC01\_address | Văn Giang, Hưng Yên |
| TC01\_titleHomePage | Home |
| TC02\_popUpFail | Vui lòng nhập đầy đủ thông tin |
| TC03\_fullName | Nguyen Quoc Bao |
| TC03\_email | nguyenquocbao.1999443@gmail.com |
| TC03\_numberPhone | 0334218638 |
| TC03\_userName | baobao |
| TC03\_password | baobao |
| TC03\_address | Văn Giang, Hưng Yên |
| TC03\_popUpFail | Vui lòng nhập đầy đủ thông tin |
| TC04\_fullName | Nguyen Quoc Bao |
| TC04\_email | nguyenquocbao.1999443@gmail.com |
| TC04\_numberPhone | 0334218638 |
| TC04\_userName | baobao |
| TC04\_password | baobao |
| TC04\_address | Văn Giang, Hưng Yên |
| TC04\_popUpFail | Email đã được sử dụng |
| TC05\_fullName | Nguyen Quoc Bao |
| TC05\_email | nguyenquocbao.1999443@gmail.com |
| TC05\_numberPhone | 0334218638 |
| TC05\_userName | baobao |
| TC05\_password | baobao |
| TC05\_address | Văn Giang, Hưng Yên |
| TC05\_popUpFail | Tên tài khoản đã được sử dụng |
| TC06\_fullName | Nguyen Quoc Bao |
| TC06\_email | nguyenquocbao.1999443@gmail.com |
| TC06\_numberPhone | 0334218638 |
| TC06\_userName | baobao |
| TC06\_password | baobao |
| TC06\_address | Văn Giang, Hưng Yên |
| TC06\_popUpFail | Email không đúng định dạng |
| TC07\_fullName | Nguyen Quoc Bao |
| TC07\_email | nguyenquocbao.1999443@gmail.com |
| TC07\_numberPhone | 0334218638 |
| TC07\_userName | baobao |
| TC07\_password | baobao |
| TC07\_address | Văn Giang, Hưng Yên |
| TC07\_titleLogin | Login |
| TC08\_fullName | Nguyen Quoc Bao |
| TC08\_email | nguyenquocbao.1999443@gmail.com |
| TC08\_numberPhone | 0334218638 |
| TC08\_userName | baobao |
| TC08\_password | baobao |
| TC08\_address | Văn Giang, Hưng Yên |
| TC08\_titleLogin | Login |

* 1. Xây dựng kịch bản kiểm thử (Test scripts)/ Phương pháp xây dựng Framework
     1. Scripst Keywords

|  |
| --- |
| **public** **class** CommonBase {  **public** WebDriver driver;  **protected** **int** DEFAULT\_TIMEOUT = 60000;  **protected** **int** WAIT\_INTERVAL = 1000;  **public** **int** loopCount = 0;  **public** **final** **int** ACTION\_REPEAT = 5;  **public** WebDriver openBrowser(String BrowserName, String url) {  **try** {  **if** (BrowserName == "FireFox") {  // System.setProperty("webdriver.gecko.driver", System.getProperty("user.dir") +  // "/DRIVER/geckodriver.exe");  WebDriverManager.*firefoxdriver*().setup();  driver = **new** FirefoxDriver();  driver.get(url);  } **else** **if** (BrowserName == "Chrome") {  // System.setProperty("webdriver.chrome.driver",  // System.getProperty("user.dir") + "/DRIVER/chromedriver.exe");  WebDriverManager.*chromedriver*().setup();  driver = **new** ChromeDriver();  driver.get(url);  } **else** **if** (BrowserName == "IE") {  WebDriverManager.*iedriver*().setup();  // System.setProperty("webdriver.ie.driver", System.getProperty("user.dir") +  // "/DRIVER/IEDriverServer.exe");  driver = **new** InternetExplorerDriver();  driver.get(url);  } **else** **if** (BrowserName == "Opera") {  WebDriverManager.*operadriver*().setup();  driver = **new** InternetExplorerDriver();  driver.get(url);  } **else** {  WebDriverManager.*chromedriver*().setup();  driver = **new** ChromeDriver();  driver.get(url);  }  } **catch** (Exception e) {  // **TODO**: handle exception  e.printStackTrace();  }  driver.manage().window().maximize();  **return** driver;  }  // close browser  **public** **void** closeBrowser() {  driver.close();  }  /\*\*  \* quit driver if driver existed  \*  \* **@param** dr  \*/  **public** **void** quitDriver(WebDriver dr) {  **if** (dr.toString().contains("null")) {  System.***out***.print("All Browser windows are closed ");  } **else** {  driver.manage().deleteAllCookies();  dr.quit();  }  }  **public** **void** waitForPageLoaded(WebDriver driver) {  ExpectedCondition<Boolean> expectation = **new** ExpectedCondition<Boolean>() {  **public** Boolean apply(WebDriver driver) {  **return** ((JavascriptExecutor) driver).executeScript("return document.readyState").toString()  .equals("complete");  }  };  **try** {  Thread.*sleep*(1000);  WebDriverWait wait = **new** WebDriverWait(driver, 30);  wait.until(expectation);  } **catch** (Throwable error) {  error.getCause().toString();  }  }  **public** **void** waitPageLoaded(String value) {  **try** {  Thread.*sleep*(Long.*parseLong*(value));  } **catch** (Exception e) {  }  }  /\*\*  \* Open page  \*  \* **@param** pageUrl  \* **@param** driver  \*/  **public** **void** navigateURL(String pageUrl, WebDriver driver) {  driver.get(pageUrl);  pause(1000);  }  **public** **void** pause(**long** timeInMillis) {  **try** {  Thread.*sleep*(timeInMillis);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  }  **public** WebElement getElement(Object locator) {  By by = locator **instanceof** By ? (By) locator : By.*xpath*(locator.toString());  WebElement elem = **null**;  **try** {  elem = driver.findElement(by);  } **catch** (NoSuchElementException e) {  checkCycling(e, 10);  pause(WAIT\_INTERVAL);  getElement(locator);  } **catch** (StaleElementReferenceException e) {  checkCycling(e, 10);  pause(WAIT\_INTERVAL);  getElement(locator);  }  **return** elem;  }  **public** **void** checkCycling(Exception e, **int** loopCountAllowed) {  System.***out***.println("Co exception xay ra: " + e.getClass().getName());  **if** (loopCount > loopCountAllowed) {  Assert.*fail*("Qua thoi gian nhung khong thay hoac thay doi tuong " + e.getMessage());  }  System.***out***.println("Lap lai lan thu " + loopCount);  loopCount++;  }  **private** **static** **void** info(String string) {  System.***out***.print(string);  }  **public** WebElement getElementPresent(Object locator, **int**... opParams) {  WebElement elem = **null**;  **int** timeout = opParams.length > 0 ? opParams[0] : DEFAULT\_TIMEOUT;  **int** isAssert = opParams.length > 1 ? opParams[1] : 1;  **int** notDisplayE = opParams.length > 2 ? opParams[2] : 0;  **for** (**int** tick = 0; tick < timeout / WAIT\_INTERVAL; tick++) {  **if** (notDisplayE == 2) {  elem = getElement(locator);  } **else** {  elem = getDisplayedElement(locator);  }  **if** (**null** != elem)  **return** elem;  pause(WAIT\_INTERVAL);  }  **if** (isAssert == 1) {  *info*("date");  **assert** **false** : ("Qua thoi gian " + timeout + "ma khong tim thay doi tuong " + locator);  quitDriver(driver);  }  **return** **null**;  }  **public** **void** check(Object locator, **int**... opParams) {  **int** notDisplayE = opParams.length > 0 ? opParams[0] : 0;  Actions actions = **new** Actions(driver);  **try** {  WebElement element = getElementPresent(locator, DEFAULT\_TIMEOUT, 1, notDisplayE);  **boolean** a = element.getAttribute("class").contains("ui-state-active");  **if** (!element.isSelected() && !a) {  actions.click(element).perform();  } **else** {  *info*("Element " + locator + " is already checked.");  }  } **catch** (StaleElementReferenceException e) {  checkCycling(e, DEFAULT\_TIMEOUT / WAIT\_INTERVAL);  pause(WAIT\_INTERVAL);  check(locator);  } **finally** {  loopCount = 0;  }  }  /\*\*  \*  \* **@param** locator  \* **@param** opParams  \*/  **public** **void** uncheck(Object locator, **int**... opParams) {  **int** notDisplayE = opParams.length > 0 ? opParams[0] : 0;  Actions actions = **new** Actions(driver);  **try** {  WebElement element = getElementPresent(locator, DEFAULT\_TIMEOUT, 1, notDisplayE);  **if** (element.isSelected()) {  actions.click(element).perform();  } **else** {  *info*("Element " + locator + " is already unchecked.");  }  } **catch** (StaleElementReferenceException e) {  checkCycling(e, 5);  pause(1000);  uncheck(locator);  } **finally** {  loopCount = 0;  }  }  **public** **void** doubleClickOnElement(Object locator) {  Actions actions = **new** Actions(driver);  **try** {  WebElement element = getElementPresent(locator);  actions.doubleClick(element).perform();  } **catch** (StaleElementReferenceException e) {  checkCycling(e, 5);  pause(1000);  doubleClickOnElement(locator);  } **finally** {  loopCount = 0;  }  }  /\*\*  \* switch to a frame  \*  \* **@param** locator  \* **@param** opParams  \*/  **public** **void** switchToFrame(Object locator, Object... opParams) {  *info*("Switch to frame " + locator);  **int** notDisplay = (Integer) (opParams.length > 0 ? opParams[0] : 0);  **try** {  driver.switchTo().frame(getElementPresent(locator, DEFAULT\_TIMEOUT, 1, notDisplay));  } **catch** (Exception e) {  switchToFrame(locator, notDisplay);  }  }  **public** String getAttribute(Object locator, String att, **int**... opParams) {  **try** {  **return** getElementPresent(locator, opParams).getAttribute(att);  } **catch** (StaleElementReferenceException e) {  checkCycling(e, DEFAULT\_TIMEOUT / WAIT\_INTERVAL);  pause(WAIT\_INTERVAL);  **return** getValue(locator);  } **finally** {  loopCount = 0;  }  }  **public** **void** back() {  pause(1000);  driver.navigate().back();  }  **public** **void** verifyEmpty(Object object) {  WebElement element = getElement(object);  WebDriverWait wait = **new** WebDriverWait(driver, 10);  wait.until(ExpectedConditions.*visibilityOf*(element));  Assert.*assertEquals*(element.getAttribute("value").toString().trim(),"");  }  **public** **void** clearText(Object object) {  WebElement element = getElement(object);  WebDriverWait wait = **new** WebDriverWait(driver, 10);  **try** {  // WebElement element = getElementPresent(locator, 10000, 0);  **if** (element != **null**) {  wait.until(ExpectedConditions.*visibilityOf*(element));  element.clear();  }  } **catch** (StaleElementReferenceException e) {  checkCycling(e, DEFAULT\_TIMEOUT / WAIT\_INTERVAL);  pause(WAIT\_INTERVAL);  clearText(element);  } **catch** (NoSuchElementException e) {  checkCycling(e, DEFAULT\_TIMEOUT / WAIT\_INTERVAL);  pause(WAIT\_INTERVAL);  clearText(element);  } **catch** (ElementNotVisibleException e) {  checkCycling(e, DEFAULT\_TIMEOUT / WAIT\_INTERVAL);  pause(WAIT\_INTERVAL);  clearText(element);  } **finally** {  loopCount = 0;  }  }  /\*\*  \* input data to element  \*  \* **@param** locator  \* **@param** value  \* **@param** validate  \*/  **public** **void** setText(Object object, String value) {  WebElement element = getElement(object);  WebDriverWait wait = **new** WebDriverWait(driver, 10);  **try** {  // WebElement element = getElementPresent(locator, 10000, 0);  **if** (element != **null**) {  wait.until(ExpectedConditions.*visibilityOf*(element));  element.clear();  element.sendKeys(value);  } **else** {  wait.until(ExpectedConditions.*visibilityOf*(element));  element.sendKeys(value);  }  } **catch** (StaleElementReferenceException e) {  checkCycling(e, DEFAULT\_TIMEOUT / WAIT\_INTERVAL);  pause(WAIT\_INTERVAL);  setText(element, value);  } **catch** (NoSuchElementException e) {  checkCycling(e, DEFAULT\_TIMEOUT / WAIT\_INTERVAL);  pause(WAIT\_INTERVAL);  setText(element, value);  } **catch** (ElementNotVisibleException e) {  checkCycling(e, DEFAULT\_TIMEOUT / WAIT\_INTERVAL);  pause(WAIT\_INTERVAL);  setText(element, value);  } **finally** {  loopCount = 0;  }  }  /\*\*  \* get value of element in web page  \*  \* **@param** locator  \* **@return**  \*/  **public** String getValue(Object locator, Object... opParams) {  **int** notDisplay = (Integer) (opParams.length > 0 ? opParams[0] : 0);  **try** {  **return** getElementPresent(locator, DEFAULT\_TIMEOUT, 1, notDisplay).getAttribute("value");  } **catch** (StaleElementReferenceException e) {  checkCycling(e, DEFAULT\_TIMEOUT / WAIT\_INTERVAL);  pause(WAIT\_INTERVAL);  **return** getValue(locator);  } **finally** {  loopCount = 0;  }  }  /\*\*  \* get a display element in web page  \*  \* **@param** locator  \* **@return**  \*/  **public** WebElement getDisplayedElement(Object locator) {  By by = locator **instanceof** By ? (By) locator : By.*xpath*(locator.toString());// co phai 1 kieu dl nao do k  WebElement e = **null**;  **try** {  **if** (by != **null**)  e = driver.findElement(by);  **if** (e != **null**) {  **if** (isDisplay(by))  **return** e;  }  } **catch** (NoSuchElementException ex) {  checkCycling(ex, DEFAULT\_TIMEOUT / WAIT\_INTERVAL);  pause(WAIT\_INTERVAL);  getDisplayedElement(locator);  } **catch** (StaleElementReferenceException ex) {  checkCycling(ex, 10);  pause(WAIT\_INTERVAL);  getDisplayedElement(locator);  } **finally** {  loopCount = 0;  }  **return** **null**;  }  /\*\*  \* checking an element is displayed in web page  \*  \* **@param** locator  \* **@return**  \*/  **public** **boolean** isDisplay(Object locator) {  **boolean** bool = **false**;  WebElement e = getElement(locator);  **try** {  **if** (e != **null**)  bool = e.isDisplayed();  } **catch** (StaleElementReferenceException ex) {  checkCycling(ex, 10);  pause(WAIT\_INTERVAL);  isDisplay(locator);  } **finally** {  loopCount = 0;  }  **return** bool;  }  /\*\*  \* click on an element  \*  \* **@param** locator  \* **@param** opParams  \*/  **public** **void** click(Object locator, Object... opParams) {  **int** notDisplay = (Integer) (opParams.length > 0 ? opParams[0] : 0);  WebElement element = getElement(locator);  Actions actions = **new** Actions(driver);  **try** {  **if** (element.isEnabled()) {  actions.click(element).perform();  } **else** {  *info*("Element is not enabled");  }  } **catch** (StaleElementReferenceException e) {  checkCycling(e, DEFAULT\_TIMEOUT / WAIT\_INTERVAL);  pause(WAIT\_INTERVAL);  click(element, notDisplay);  } **catch** (ElementNotVisibleException e) {  checkCycling(e, DEFAULT\_TIMEOUT / WAIT\_INTERVAL);  pause(WAIT\_INTERVAL);  click(element, notDisplay);  } **catch** (NoSuchElementException e) {  checkCycling(e, DEFAULT\_TIMEOUT / WAIT\_INTERVAL);  pause(WAIT\_INTERVAL);  click(element, notDisplay);  } **finally** {  loopCount = 0;  }  }  /\*\*  \* get text of element  \*  \* **@param** locator  \* **@return**  \*/  **public** String getText(Object locator) {  WebElement element = **null**;  **try** {  element = getElementPresent(locator);  **return** element.getText();  } **catch** (StaleElementReferenceException e) {  checkCycling(e, DEFAULT\_TIMEOUT / WAIT\_INTERVAL);  pause(WAIT\_INTERVAL);  **return** getText(locator);  } **finally** {  loopCount = 0;  }  }  /\*\*  \* get text of element  \*  \* **@param** locator  \* **@return**  \*/  **public** String getText(WebElement element) {  **try** {  **return** element.getText();  } **catch** (StaleElementReferenceException e) {  checkCycling(e, DEFAULT\_TIMEOUT / WAIT\_INTERVAL);  pause(WAIT\_INTERVAL);  **return** getText(element);  } **finally** {  loopCount = 0;  }  }  /\*\*  \* compare 2 string  \*  \* **@param** s1  \* **@param** s2  \*/  **public** **void** verifyCompare(String s1, String s2) {  **if** (s1 != "" && s1 != **null** && s2 != **null** && s2 != "") {  Assert.*assertFalse*(!s1.equalsIgnoreCase(s2), "So sanh khong bang nhau: " + s1 + " va " + s2);  } **else** **if** ((s1 == "" || s1 == **null**) && (s2 == "" || s2 == **null**)) {  *info*("2 truong du lieu can so sanh deu null");  } **else** {  Assert.*fail*("Du lieu so sanh co 1 truong bi null");  }  }  /\*\*  \* verifyElementText  \*  \* **@param** s1  \* **@param** s2  \*/  **public** Boolean verifyElementText(WebElement element, String text) {  Boolean check = **false**;  **try** {  **if** (element.getText() == text) {  check = **true**;  }  } **catch** (StaleElementReferenceException e) {  checkCycling(e, DEFAULT\_TIMEOUT / WAIT\_INTERVAL);  pause(WAIT\_INTERVAL);  **return** **false**;  } **finally** {  loopCount = 0;  }  **return** check;  }  **public** **void** switchNewTab(**int**... index) {  **int** tab = index.length > 0 ? index[0] : 1;  List<String> tabs2 = **new** ArrayList<String>(driver.getWindowHandles());  driver.switchTo().window(tabs2.get(tab));  }  **public** **void** verifyCompareNotEqual(String s1, String s2) {  **if** (s1 != "" && s1 != **null** && s2 != **null** && s2 != "") {  Assert.*assertTrue*(!s1.equalsIgnoreCase(s2),  "Pass neu 2 truong so sanh khong bang nhau: " + s1 + " va " + s2);  } **else** **if** ((s1 == "" || s1 == **null**) && (s2 == "" || s2 == **null**)) {  *info*("2 truong du lieu can so sanh deu null");  } **else** {  Assert.*fail*("Du lieu so sanh co 1 truong bi null");  }  }  **public** **void** enter(Object locator) {  **if** (locator != **null**) {  WebElement e = getElementPresent(locator);  e.sendKeys(Keys.***ENTER***);  }  }  **public** **void** waitForElementDisappear(Object locator) {  By by = locator **instanceof** By ? (By) locator : By.*xpath*(locator.toString());  **int** i = 0;  **while** (i < 60) {  **try** {  **if** (driver.findElement(by).isDisplayed()) {  pause(1000);  i++;  }  } **catch** (NoSuchElementException ex) {  **break**;  } **catch** (StaleElementReferenceException ex) {  **break**;  }  }  **if** (i == 60) {  Assert.*fail*("Qua thoi gian doi tuong van display");  }  }  **public** **void** selectValueFromDropdown(Object locator, String value) {  Select dropDown = **new** Select(getElement(locator));  dropDown.selectByVisibleText(value);  }  /\*\*  \*  \* **@param** locator  \* **@return**  \*/  **public** List<WebElement> getListElement(Object locator) {  By by = locator **instanceof** By ? (By) locator : By.*xpath*(locator.toString());  List<WebElement> elementOptions;  **try** {  elementOptions = driver.findElements(by);  **return** elementOptions;  } **catch** (NoSuchElementException ex) {  checkCycling(ex, 10);  pause(WAIT\_INTERVAL);  getListElement(locator);  } **catch** (StaleElementReferenceException ex) {  checkCycling(ex, 10);  pause(WAIT\_INTERVAL);  getListElement(locator);  } **finally** {  loopCount = 0;  }  **return** **null**;  }  /\*\*  \*  \* **@param** urlText  \* **@param** addParam  \* **@param** column  \* **@return**  \*/  **public** **static** String getRestService(String urlText, String addParam) {  String output = "";  String urlT = "";  **if** (urlText != "" && !urlText.contains("http")) {  urlT = System.*getProperty*("serviceLink") + urlText;  } **else** {  urlT = urlText;  }  **try** {  URL url = **new** URL(urlT);  HttpURLConnection con = (HttpURLConnection) url.openConnection();  con.setRequestMethod("GET");  con.setRequestProperty("Accept", "application/json");  **if** (addParam != "") {  String[] param = addParam.split(",");  **if** (param.length > 0) {  **for** (**int** i = 0; i < param.length; i++) {  **if** (param[i] != "") {  String[] a = param[i].split(":");  **if** (a.length > 0) {  con.setRequestProperty(a[0], a[1]);  }  }  }  }  }  **int** code = con.getResponseCode();  *info*("Result Execute Services is: " + code + " " + con.getResponseMessage());  **if** (code == 200) {  InputStream in = con.getInputStream();  output = *read*(in);  } **else** {  InputStream in = con.getErrorStream();  output = *read*(in);  }  *info*("Output of service is: " + output);  } **catch** (IOException | RuntimeException e) {  e.printStackTrace();  }  **return** output;  }  **public** **static** String read(InputStream input) **throws** IOException {  **try** (BufferedReader buffer = **new** BufferedReader(**new** InputStreamReader(input))) {  **return** buffer.lines().collect(Collectors.*joining*("\n"));  }  }  **public** Response getAPI(String URL, String... value) **throws** UnsupportedEncodingException {  Response res = **null**;  **if** (value.length == 1) {  String token = value[0];  res = *given*().header("Authorization", "Bearer " + token).when().get(URL).then()  .contentType(ContentType.***JSON***).log().all().extract().response();  }  **if** (value.length == 2) {  String token = value[0];  **int** status = 0;  **try** {  status = Integer.*parseInt*(value[1]);  } **catch** (Exception e) {  e.printStackTrace();  }  res = *given*().header("Authorization", "Bearer " + token).when().get(URL).then()  .contentType(ContentType.***JSON***).statusCode(status).log().all().extract().response();  } **else** {  res = *given*().when().get(URL).then().contentType(ContentType.***JSON***).log().all().extract().response();  }  **return** res;  }  **public** String getJSONPath(Response res) {  String JSONPath = "";  **try** {  res.jsonPath().toString();  } **catch** (Exception e) {  // **TODO**: handle exception  }  **return** JSONPath;  }  } |

* + 1. Script Convert Excel File

|  |
| --- |
| **public** **class** ExcelUtils {  **private** **static** XSSFWorkbook *workBook*;  **private** **static** XSSFSheet *workSheet*;  **private** **static** XSSFCell *cell*;  // private static XSSFRow row;    **private** String filepath;    **public** ExcelUtils(String path) {  filepath=path;  }    //1. WORKING WITH EXCEL FILE  **public** **static** **void** setExcelFile(String path, String SheetName) **throws** Exception{  File file=**new** File(path);  **if**(file.canRead()) {  FileInputStream ExcelFile=**new** FileInputStream(file);  *workBook* =**new** XSSFWorkbook(ExcelFile);  *workSheet*=*workBook*.getSheet(SheetName);  ExcelFile.close();  }  }    **public** **static** XSSFSheet getSheet(String SheetName) {  XSSFSheet sheet=*workBook*.getSheet(SheetName);  **if**(!*isSheet*(SheetName)) {  sheet=*workBook*.createSheet(SheetName);  }  **return** sheet;  }    **public** **void** save()**throws** IOException{  FileOutputStream streamOut=**new** FileOutputStream(filepath);  *workBook*.write(streamOut);  streamOut.flush();  streamOut.close();  }    **public** **static** **void** saveAs(String path)**throws** IOException{  FileOutputStream streamOut=**new** FileOutputStream(path);  *workBook*.write(streamOut);  streamOut.flush();  streamOut.close();  }    **public** **static** **boolean** isSheet(String SheetName) {  **return** *workBook*.getSheetIndex(SheetName)>=0;  }    **public** **static** **void** addSheet(String SheetName) {  **if**(!*isSheet*(SheetName)) {  *workBook*.createSheet(SheetName);  }  }    **public** **static** **void** removeSheet(**int** SheetIndex) {  *workBook*.removeSheetAt(SheetIndex);  }    **public** **static** **void** removeSheet(String SheetName) {  **int** index=*workBook*.getSheetIndex(SheetName);  *removeSheet*(index);  }  //2. WORKING WITH COLUMN IN EXCEL  **public** **static** **void** addColumn(String SheetName,String ColName) {  XSSFSheet sheet=*getSheet*(SheetName);  *addColumn*(sheet, ColName);  }    **public** **static** **void** addColumn(XSSFSheet sheet, String ColName) {  XSSFCellStyle style=*workBook*.createCellStyle();  style.setFillForegroundColor(HSSFColorPredefined.***RED***.getIndex());  style.setFillPattern(FillPatternType.***SOLID\_FOREGROUND***);  XSSFRow row=sheet.getRow(0);  **if**(row==**null**)  row=sheet.createRow(0);    XSSFCell cell;  **if**(row.getLastCellNum()==-1)  cell=row.createCell(0);  **else**  cell=row.createCell(row.getLastCellNum());  cell.setCellValue(ColName);  //cell.setCellStyle(style);  }    **public** **static** **void** removeColumn(String SheetName, **int** colNum) {  XSSFSheet sheet=*getSheet*(SheetName);  **int** rowCount=sheet.getLastRowNum()+1;    XSSFRow row;  **for**(**int** i=0;i<rowCount;i++) {  row=sheet.getRow(i);  **if**(row!=**null**) {  XSSFCell cell =row.getCell(colNum);  **if**(cell!=**null**) {  row.removeCell(cell);  }  }  }  }    **public** **static** **int** convertColNameToColNum(XSSFSheet sheet, String colName) {  XSSFRow row=sheet.getRow(0);  **int** cellRowNumber=row.getLastCellNum();  **int** colNum=-1;    **for**(**int** i=0;i<cellRowNumber;i++) {  **if**(row.getCell(i).getStringCellValue().trim().equals(colName)) {  colNum=i;  }  }  **return** colNum;  }    //3. WORKING WITH ROW IN EXCEL  **public** **static** XSSFRow getRow(XSSFSheet sheet, **int** rowIndex) {  XSSFRow row=sheet.getRow(rowIndex);  **if**(row==**null**) {  row=sheet.createRow(rowIndex);  }  **return** row;  }    // This method is to get the row count used of the excel sheet  **public** **static** **int** getRowCount(String SheetName) {  *workSheet* = *workBook*.getSheet(SheetName);  **int** number = *workSheet*.getLastRowNum() + 1;  **return** number;  }  // This method is to get the Row number of the test case  // This methods takes three arguments(Test Case name , Column Number & Sheet  // name)  **public** **static** **int** getRowContains(String SheetName,String sTestCaseName, **int** colNum ) **throws** Exception {  **int** i;  *workSheet* = *workBook*.getSheet(SheetName);  **int** rowCount = ExcelUtils.*getRowCount*(SheetName);  **for** (i = 0; i < rowCount; i++) {  **if** (ExcelUtils.*getCellValue*(SheetName, i, colNum).equalsIgnoreCase(sTestCaseName)) {  **break**;  }  }  **return** i;  }    **public** **static** String getCellValue(**int** RowNum, **int** ColNum) **throws** Exception {  **try** {  *cell* = *workSheet*.getRow(RowNum).getCell(ColNum);  String cellValue = *cell*.getStringCellValue();  **return** cellValue;  } **catch** (Exception e) {  **return** "";  }  }    **public** **static** String getCellValue(String sheetName, **int** rowIndex, **int** colIndex) {  XSSFSheet sheet = *getSheet*(sheetName);  XSSFRow row=*getRow*(sheet, rowIndex);  XSSFCell cell=*getCell*(row, colIndex);  **return** cell.getStringCellValue();  }      **public** **static** String getCellValue(XSSFCell cell) {  FormulaEvaluator evaluator = *workBook*.getCreationHelper().createFormulaEvaluator();  DataFormatter df = **new** DataFormatter();    **if**(cell!=**null**) {  **switch**(evaluator.evaluateInCell(cell).getCellType()) {    **case** ***STRING***:  **return** cell.getStringCellValue();    **case** ***NUMERIC***:  // return String.valueOf(cell.getNumericCellValue());  **return** df.formatCellValue(cell);    **case** ***BOOLEAN***:  **return** String.*valueOf*(cell.getBooleanCellValue());    **case** ***BLANK***:  **return** "";    **case** ***ERROR***:  **return** cell.getErrorCellString();    **case** ***FORMULA***:  **return** cell.getCellFormula();  **default**:  **break**;  }  }  **return** **null**;  }    **public** **static** XSSFCell getCell(XSSFRow row, **int** colIndex) {  XSSFCell cell=row.getCell(colIndex-1);  **if**(cell==**null**) {  cell=row.createCell(colIndex-1);  }  **return** cell;  }    **public** **static** **void** setCell(XSSFSheet sheet, **int** rowIndex, **int** colIndex, String value) {  XSSFRow row=*getRow*(sheet, rowIndex);  XSSFCell cell=*getCell*(row,colIndex);  cell.setCellValue(value);  }  **public** **static** **void** setCell(String sheetName,**int** rowIndex,**int** colIndex, String value) {  XSSFSheet sheet=*getSheet*(sheetName);  *setCell*(sheet, rowIndex, colIndex, value);  }    **public** **static** **void** setCell(String sheetName,String colName,**int** rowIndex, String value) {  XSSFSheet sheet=*getSheet*(sheetName);  **int** colIndex=*convertColNameToColNum*(sheet,colName);  *setCell*(sheet, rowIndex, colIndex, value);  }    **public** **static** List<List<XSSFCell>> createList(String sheetName) {  // Create an ArrayList to store the data read from excel sheet.  List<List<XSSFCell>> myList = **new** ArrayList<List<XSSFCell>>();    // When we have a sheet object in hand we can iterator on  // each sheet's rows and on each row's cells. We store the  // data read on an ArrayList so that we can printed the  // content of the excel to the console.  **try** {  XSSFSheet sheet = *getSheet*(sheetName);  Iterator<Row> rows = sheet.rowIterator();  **while** (rows.hasNext()) {  XSSFRow row = (XSSFRow) rows.next();  Iterator<Cell> cells = row.cellIterator();  List<XSSFCell> data = **new** ArrayList<XSSFCell>();  **while** (cells.hasNext()) {  XSSFCell cell = (XSSFCell) cells.next();  data.add(cell);  }  myList.add(data);  }  } **catch** (Exception e) {  e.getMessage();  }  **return** myList;  }  }  **public** **class** TestCase  {  **private** String name;  **private** List<TestDetail> steps;    **public** TestCase() {  }    **public** TestCase(**final** String name, **final** List<TestDetail> steps) {  **this**.name = name;  **this**.steps = steps;  }    **public** String getName() {  **return** **this**.name;  }    **public** TestCase setName(**final** String name) {  **this**.name = name;  **return** **this**;  }    **public** List<TestDetail> getSteps() {  **return** **this**.steps;  }    **public** TestCase setSteps(**final** List<TestDetail> steps) {  **this**.steps = steps;  **return** **this**;  }    @Override  **public** String toString() {  **return** "automation.TestCase{name='" + **this**.name + '\'' + ", steps=" + **this**.steps + '}';  }  }  **package** common;  **public** **class** TestDetail  {  **private** String testCase;  **private** String script;  **private** String object;  **private** String input;  **private** String output;    **public** TestDetail(String testCase, String script, String object, String input, String output) {  **this**.testCase=testCase;  **this**.script=script;  **this**.object=object;  **this**.input=input;  **this**.output=output;    }    **public** String getScript() {  **return** **this**.script;  }    **public** TestDetail setScript(**final** String script) {  **this**.script = script;  **return** **this**;  }    **public** String getObject() {  **return** **this**.object;  }    **public** TestDetail setObject(**final** String object) {  **this**.object = object;  **return** **this**;  }    **public** String getInput() {  **return** **this**.input;  }    **public** TestDetail setInput(**final** String input) {  **this**.input = input;  **return** **this**;  }    **public** String getTestCase() {  **return** **this**.testCase;  }    **public** TestDetail setTestCase(**final** String testCase) {  **this**.testCase = testCase;  **return** **this**;  }    **public** String getOutput() {  **return** **this**.output;  }    **public** TestDetail setOutput(**final** String output) {  **this**.output = output;  **return** **this**;  }    }  **public** **class** TestFunction  {  **private** String name;  **private** List<String> functions;    **public** TestFunction(**final** String name, **final** List<String> functions) {  **this**.functions = **new** ArrayList<String>();  **this**.name = name;  **this**.functions = functions;  }    **public** **boolean** contains(**final** String function) {  **return** **this**.functions.contains(function);  }    **public** String getName() {  **return** **this**.name;  }    **public** TestFunction setName(**final** String name) {  **this**.name = name;  **return** **this**;  }    **public** List<String> getFunctions() {  **return** **this**.functions;  }    **public** TestFunction setFunctions(**final** List<String> functions) {  **this**.functions = functions;  **return** **this**;  }  }  **public** **class** TestObject  {  **private** String name;  **private** String value;  **private** String type;    **public** TestObject(**final** String name, **final** String value, **final** String type) {  **this**.name = name;  **this**.value = value;  **this**.type = type;  }    **public** String getName() {  **return** **this**.name;  }    **public** TestObject setName(**final** String name) {  **this**.name = name;  **return** **this**;  }    **public** String getValue() {  **return** **this**.value;  }    **public** TestObject setValue(**final** String value) {  **this**.value = value;  **return** **this**;  }    **public** String getType() {  **return** **this**.type;  }    **public** TestObject setType(**final** String type) {  **this**.type = type;  **return** **this**;  }  }  **public** **class** TestParam  {  **private** String name;  **private** String value;    **public** TestParam(**final** String name, **final** String value) {  **this**.name = name;  **this**.value = value;  }    **public** String getName() {  **return** **this**.name;  }    **public** TestParam setName(**final** String name) {  **this**.name = name;  **return** **this**;  }    **public** String getValue() {  **return** **this**.value;  }    **public** TestParam setValue(**final** String value) {  **this**.value = value;  **return** **this**;  }  } |

* 1. Thực thi và báo cáo kiểm thử

< Phần này trình bày kết quả thực thi kiểm thử tự động và báo cáo kết quả kiểm thử chương trình >

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kết quả đạt được

<Phần này trình bày kết quả đạt được của đề tài>

Hạn chế của đề tài

<Phần này trình bày hạn chế của đề tài>

Hướng phát triển của đề tài

<Phần này trình bày hướng phát triển tiếp theo của đề tài>

TÀI LIỆU THAM KHẢO

x

|  |  |
| --- | --- |
| [1] | Tên tác giả (năm XB), Tên sách, NXB, Nơi XB |
| [2] | ……….. |

PHỤ LỤC

1. <Tiêu đề phụ lục 1 (nếu có) >
2. <Tiêu đề phụ lục 1 (nếu có) >

…..