

**TRƯỜNG ĐẠI HỌC KINH TẾ
KHOA THỐNG KÊ – TIN HỌC**



BÁO CÁO THỰC TẬP NGHỀ NGHIỆP

NGÀNH HỆ THỐNG THÔNG TIN QUẢN LÝ

CHUYÊN NGÀNH QUẢN TRỊ HỆ THỐNG THÔNG TIN

Đề tài:

**ỨNG DỤNG MANUAL TESTING VÀ AUTOMATION TESTING
CHO TRANG WEBSITE MOLLEE**

Sinh viên thực hiện : Bùi Lê Khánh Vy
Lớp : 47K21.2
Đơn vị thực tập : FPT Software Đà Nẵng
Cán bộ hướng dẫn : Phạm Nguyễn Phong
Giảng viên hướng dẫn : ThS. Nguyễn Văn Chức

Đà Nẵng, 8/2023

NHẬN XÉT CỦA ĐƠN VỊ THỰC TẬP

Họ và tên sinh viên: Bùi Lê Khánh Vy

Lớp: 47K21.21

Khoa Thống kê – Tin học, Trường Đại học Kinh tế, Đại học Đà Nẵng

Thực tập từ ngày: 03/05/2023 đến ngày: 30/07/2023

Tên đơn vị thực tập: IVS-FPT Software

Địa chỉ:

Số điện thoại liên hệ:

Họ tên cán bộ hướng dẫn: Phạm Nguyễn Phong

Sau quá trình thực tập của sinh viên tại đơn vị, chúng tôi có một số đánh giá như sau:

STT	Nội dung đánh giá	Rất không tốt	Không tốt	Bình thường	Tốt	Rất tốt
1	Về thái độ, ý thức, đạo đức và việc tuân thủ các quy định và văn hóa đơn vị thực tập	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	Kiến thức chuyên môn	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	Khả năng hòa nhập, thích nghi và tác phong nghề nghiệp	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	Trách nhiệm, sáng tạo trong công việc	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

(Anh/chị vui lòng đánh dấu X vào ô tương ứng với năng lực của sinh viên)

Ý kiến nhận xét và đề xuất *(Nhằm nâng cao chất lượng đào tạo, Nhà trường rất mong muốn nhận thêm những ý kiến khác từ quý doanh nghiệp):*

Đà Nẵng, ngàythángnăm 2023

Xác nhận của đơn vị thực tập

LỜI CẢM ƠN

Đầu tiên, với lòng biết ơn sâu sắc, em xin được phép gửi lời cảm ơn chân thành đến tất cả các cá nhân và tổ chức đã tạo điều kiện hỗ trợ, giúp đỡ em trong suốt quá trình học tập và nghiên cứu đề tài này.

Em xin chân thành cảm ơn quý thầy, cô giáo trong khoa Thống Kê – Tin Học trường Đại học Kinh tế - Đại học Đà Nẵng vì đã tận tâm giảng dạy và truyền đạt những kiến thức, kinh nghiệm quý báu cho em. Đặc biệt, em xin gửi lời cảm ơn và lòng biết ơn tới thầy Nguyễn Văn Chúc – GVHD của em vì tất cả sự giúp đỡ tận tình của thầy trong suốt kỳ thực tập nghề nghiệp này.

Bên cạnh đó, em xin chân thành cảm ơn quý công ty FPT Software Đà Nẵng nói chung, và đơn vị thực tập Fsoft-IVS nói riêng vì đã tạo điều kiện thuận lợi để em được học tập, có cơ hội trải nghiệm thực tế và tích lũy được nhiều kinh nghiệm cho bản thân. Em cũng xin gửi lời cảm ơn đặc biệt đến anh Phạm Nguyễn Phong và các anh chị mentor hướng dẫn em trực tiếp tại công ty, tận tình chỉ bảo và cung cấp những tài liệu bổ ích, hỗ trợ em tận tình để em có thể hoàn thành tốt bài báo cáo này.

Trong khoảng thời gian thực tập tại công ty, em đã có cơ hội học tập và nghiên cứu, giúp tích lũy được nhiều kinh nghiệm hơn trong công việc nhưng vẫn không thể tránh khỏi các sai sót. Em rất mong sẽ được nhận những ý kiến đóng góp quý báu của quý Thầy cô cũng như của Ban lãnh đạo Công ty để em có thể hoàn thiện mình hơn trong tương lai.

Sau cùng, em xin kính chúc quý thầy cô, Ban lãnh đạo và tập thể anh chị trong công ty luôn dồi dào sức khỏe và thành công trong sự nghiệp của mình.

Em xin chân thành cảm ơn!

LỜI CAM ĐOAN

Em xin cam đoan thành quả báo cáo này là kết quả nghiên cứu do em tự thực hiện trong suốt quá trình thực tập tại công ty FPT Software Đà Nẵng, dưới sự hướng dẫn tận tình của các anh chị mentor và TS Nguyễn Văn Chức – GVHD. Ngoài ra không có bất kỳ sự sao chép nào khác.

Đề tài và nội dung sự án của em là trung thực, ngoài một số định nghĩa lý thuyết ra thì tất cả đều được thực hiện dựa trên các tài liệu nghiên cứu của công ty FPT Software Đà Nẵng. Những dữ liệu và kết quả đạt được trong báo cáo hoàn toàn là trung thực, không đạo nhái hay được sao chép từ bất kỳ ai. Em xin chịu hoàn toàn trách nhiệm, kỷ luật bộ môn và nhà trường đề ra nếu có vấn đề nào xảy ra.

MỤC LỤC

MỤC LỤC	
LỜI CẢM ƠN	2
LỜI CAM ĐOAN	3
MỤC LỤC	4
DANH MỤC HÌNH ẢNH	7
DANH MỤC BẢNG BIỂU	8
DANH MỤC CÁC TỪ VIẾT TẮT	9
LỜI MỞ ĐẦU	10
CHƯƠNG I: TỔNG QUAN VỀ DOANH NGHIỆP VÀ LÝ THUYẾT KIỂM THỬ PHẦN MỀM	11
1.1. Giới thiệu tổng quan về doanh nghiệp thực tập	11
1.1.1. Giới thiệu doanh nghiệp.....	11
1.1.2. Giới thiệu đơn vị thực tập IVS Đà Nẵng.....	11
1.1.3. Tầm nhìn và sứ mệnh	12
1.1.4. Giá trị cốt lõi	12
CHƯƠNG II: TỔNG QUAN VỀ LÝ THUYẾT KIỂM THỬ PHẦN MỀM	13
2.1. Tổng quan về kiểm thử phần mềm	13
2.1.1. Khái niệm.....	13
2.1.2. Mục đích của kiểm thử phần mềm.....	13
2.1.3. Bảy nguyên tắc kiểm thử phần mềm	13
2.1.4. Các giai đoạn của chu trình phát triển phần mềm	14
2.1.5. Quy trình kiểm thử.....	15
2.1.6. Các cấp độ kiểm thử.....	17
2.1.7. Các loại kiểm thử	18
2.2. Tổng quan về Test Case	20
2.2.1. Khái niệm	20
2.2.2. Các kỹ thuật thiết kế Test Case phổ biến	20
2.3. Tổng quan về Bug và Report Bug	21
2.3.1. Khái niệm Bug.....	21
2.3.2. Report Bug.....	22
2.4. Sự khác nhau giữa Manual Testing và Automation Testing	23
2.4.1. Khái niệm	23
2.4.2. Ưu nhược điểm	23

2.4.3. Bảng so sánh chi tiết	24
2.5. Tổng quan về vị trí việc làm	25
2.5.1. Khái niệm Tester	25
2.5.2. Các kỹ năng cần có của một Tester.....	25
2.5.3. Cơ hội nghề nghiệp	26
CHƯƠNG III: ĐẶC TẢ WEBSITE MOLLEE	27
3.1. Tổng quan về Website MOLLEE	27
3.2. Sơ đồ Use case tổng quát	27
3.2.1. Vai trò của từng tác nhân	27
3.2.2. Sơ đồ Use case	28
3.3. Workflow	29
3.4. Đặc tả yêu cầu, thiết kế hệ thống	30
3.4.1. Phân tích Use case Sign Up.....	30
□ Đặc tả yêu cầu chức năng Đăng ký (Login)	30
3.4.2. Phân tích Use case Login	33
□ Đặc tả yêu cầu chức năng Đăng nhập (Login).....	33
3.4.2. Phân tích Use case Quản lý giỏ hàng.....	35
□ Đặc tả yêu cầu chức năng Chỉnh sửa số lượng sản phẩm trong giỏ hàng	36
□ Đặc tả yêu cầu chức năng Xóa sản phẩm trong giỏ hàng	37
CHƯƠNG IV: TỔNG QUAN VỀ SELENIUM VÀ TRIỂN KHAI KIỂM THỬ WEBSITE MOLLEE.....	40
4.1. Tổng quan về tool kiểm thử tự động SELENIUM	40
4.1.1. Khái niệm.....	40
4.1.2. Cấu trúc của Selenium.....	40
4.1.2. Đánh giá chung.....	41
4.2. Triển khai kiểm thử Website MOLLEE.....	41
4.2.1. Môi trường kiểm thử	41
4.2. Trạng thái của Test case.....	41
4.3. Tiêu chí kiểm thử	41
4.3. Kiểm thử thủ công (Manual Test)	42
4.3.1. Thiết kế Test case	42
4.3.2. Cấu trúc Test case	42
4.3.3. Thiết kế Test case chức năng Đăng ký (Sign Up).....	42
4.3.4. Thiết kế Test case chức năng Đăng nhập (Log In).....	43

4.3.5. Thiết kế Test case chức năng Quản lý giỏ hàng (Cart)	43
4.4. Kiểm thử tự động (Automation Test).....	43
4.4.1. Môi trường kiểm thử	43
4.4.2. Thiết kế Test script chức năng Đăng ký (Sign Up)	43
4.4.3. Thiết kế Test script chức năng Đăng nhập (Login)	65
4.4.4. Thiết kế Test script chức năng Quản lý giỏ hàng (Cart)	72
4.4.5. Kết quả kiểm thử	83
4.5. Log Bug	85
4.5.1. Cấu trúc Report Bug	85
4.5.1. Log Bug Test case Manual	85
4.5.2. Log Bug Test case Automation	85
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	86
TÀI LIỆU THAM KHẢO	87
CHECKLIST CỦA BÁO CÁO.....	88
PHỤ LỤC.....	89

DANH MỤC HÌNH ẢNH

Hình 1. 1: Logo FPT-----	11
Hình 1. 2: Logo IVS -----	12
Hình 2. 1: 7 Nguyên tắc kiểm thử phần mềm -----	13
Hình 2. 2: Quy trình kiểm thử-----	16
Hình 2. 3: Các cấp độ kiểm thử -----	17
Hình 2. 4: Các loại kiểm thử-----	19
Hình 2. 5: Vòng đời của Bug-----	21
Hình 3. 1: Use Case hệ thống -----	28
Hình 3. 2: Workflow Hệ thống-----	29
Hình 3. 3: Màn hình Sign Up-----	30
Hình 3. 4: Màn hình LogIn -----	33
Hình 3. 5: Màn hình giỏ hàng -----	35
Hình 3. 6: Dialog “Are you sure?” -----	36
Hình 4. 1: Cấu trúc của Selenium-----	40
Hình 4. 2: Kết quả Automation test màn hình Sign Up-----	83
Hình 4. 3: Kết quả Automation test màn hình LogIn -----	84
Hình 4. 4: Kết quả Automation test màn hình Cart -----	84

DANH MỤC BẢNG BIỂU

Bảng 1: Bảng so sánh Automation và Manual Testing -----	25
Bảng 2: Đặc tả yêu cầu chức năng Đăng ký -----	31
Bảng 3: Yêu cầu và điều kiện của màn hình Sign Up -----	32
Bảng 4: Đặc tả yêu cầu chức năng LogIn -----	34
Bảng 5: Yêu cầu và điều kiện với màn hình LogIn -----	34
Bảng 6: Đặc tả yêu cầu chức năng thêm sản phẩm vào giỏ hàng -----	37
Bảng 7: Yêu cầu và điều kiện của màn hình Log In -----	38
Bảng 8: Bảng kết quả Manual màn hình Sign Up -----	43
Bảng 9: Bảng kết quả Manual màn hình LogIn -----	43
Bảng 10: Bảng kết quả Manual màn hình Cart -----	43
Bảng 11: Bảng kết quả Automatuion màn hình Sign Up -----	83
Bảng 12: Bảng kết quả Automatuion màn hình LogIn -----	84
Bảng 13: Bảng kết quả Automatuion màn hình Cart -----	85

DANH MỤC CÁC TỪ VIẾT TẮT

- **BU:** Viết tắt của từ Business Unit
- **N/A:** Viết tắt của từ Not Available

LỜI MỞ ĐẦU

1. Mục tiêu của đề tài

- Nắm vững và hiểu rõ về các kiến thức kiểm thử nói chung và các phương pháp kiểm thử thủ công và kiểm thử tự động nói riêng.
- Áp dụng được các kiến thức về kiểm thử cho website kinh doanh thời trang Mollee.
 - o Nắm được các kiến thức về kiểm thử.
 - o Nắm được đặc điểm của nghề Kiểm thử.
 - o Áp dụng các kiến thức và kỹ năng, công cụ kiểm thử phần mềm để kiểm thử.
 - o Thực hiện phân tích yêu cầu, viết testcase, xây dựng kịch bản kiểm thử, thực hiện kiểm thử, báo cáo bug.

2. Đối tượng và phạm vi nghiên cứu

- Đối tượng nghiên cứu:
 - o Các kỹ thuật kiểm thử thủ công.
 - o Websile kinh doanh thời trang Mollee.
- Phạm vi nghiên cứu: Kiểm thử thủ công kết hợp tự động hóa trên Selenium cho Websile Mollee
 - o Chức năng Login
 - o Chức năng Sign Up
 - o Chức năng Quản lý giỏ hàng với vai trò của Khách hàng

3. Kết cấu của đề tài

Đề tài được tổ chức gồm phần mở đầu, 4 chương nội dung và phần kết luận.

- Mở đầu
- Chương 1: Tổng quan về doanh nghiệp
- Chương 2: Tổng quan về lý thuyết kiểm thử phần mềm
- Chương 3: Tổng quan về Websile Mollee
- Chương 4: Tổng quan về Selenium và triển khai thực hiện
- Kết luận và hướng phát triển

CHƯƠNG I: TỔNG QUAN VỀ DOANH NGHIỆP VÀ LÝ THUYẾT KIỂM THỬ PHẦN MỀM

1.1. Giới thiệu tổng quan về doanh nghiệp thực tập

1.1.1. Giới thiệu doanh nghiệp

FPT Software là một thành viên quan trọng của tập đoàn FPT. Được thành lập vào năm 1999, FPT Software đã trở thành nhà cung cấp dịch vụ và phần mềm toàn cầu lớn nhất Việt Nam. Với hơn 800 khách hàng trên toàn thế giới và đội ngũ 28.000 nhân sự, FPT Software hiện diện tại 27 quốc gia và vùng lãnh thổ, bao gồm 59 văn phòng và 22 trung tâm phát triển phần mềm. Công ty không chỉ hướng tới đáp ứng nhu cầu phát triển CNTT trong nước mà còn tham vọng vươn ra toàn cầu, mang thương hiệu FPT đến với các công ty quốc tế, nâng cao năng suất lao động thông qua công nghệ.



Hình 1. 1: Logo FPT

FPT Software Đà Nẵng được thành lập vào năm 2005 và sau 19 năm phát triển, đã trở thành công ty công nghệ thông tin lớn nhất miền Trung. Trong ba năm gần đây, FPT Software Đà Nẵng đã đạt tốc độ tăng trưởng trung bình trên 70% mỗi năm, tốc độ cao nhất trong toàn tập đoàn. Sự tăng trưởng này mang đến nhiều cơ hội lớn cho các cá nhân tham gia vào quá trình phát triển của công ty. Các thế hệ nhân viên tại đây đều chia sẻ niềm tin rằng FPT Software sẽ tiếp tục tiến xa hơn nữa, hiện thực hóa ước mơ số của người Việt Nam trên toàn cầu.

Hiện tại, FPT Software Đà Nẵng có hai trụ sở chính: Tòa nhà FPT An Đồn tại Phường An Hải Bắc, Quận Sơn Trà và Tòa nhà FPT Complex tại đường Nam Kỳ Khởi Nghĩa, Phường Hòa Hải, Quận Ngũ Hành Sơn.

1.1.2. Giới thiệu đơn vị thực tập IVS Đà Nẵng



Hình 1. 2: Logo IVS

IVS (Independent Verification Services) là một đơn vị (Bu) kiểm thử phần mềm độc lập trực thuộc FPT Software, chuyên hoạt động trong lĩnh vực cung cấp các dịch vụ tư vấn giải pháp và kiểm thử phần mềm cho các ngành Ngân hàng, Tài chính, sản xuất Ô tô, Thiết bị thông minh, Y tế, Tự động hóa...

IVS được FPT Software cho ra mắt vào đầu năm 2019 là một thành viên của FPT Global Automative và trở thành đơn vị độc lập với tên gọi là IVS. Với mục tiêu là đảm nhận tất cả công việc liên quan đến dịch vụ kiểm thử của FPT Software với các đối tác và khách hàng. Với hàng trăm khách hàng là các doanh nghiệp lớn trên thế giới đến từ Nhật Bản, Hàn Quốc, Mỹ, Singapore, Canada,... cùng đội ngũ nguyên gia hàng chục năm kinh nghiệm trong lĩnh vực kiểm thử phần mềm sở hữu nhiều chứng chỉ đẳng cấp thế giới dành cho Tester như ISTQB, CAST. Ngày càng nhiều bạn trẻ với sự năng động và sáng tạo mới mẻ tạo nên môi trường làm việc với nhiều dấu ấn riêng biệt.

1.1.3. Tầm nhìn và sứ mệnh

- Tầm nhìn: Mục tiêu dài hạn của FPT Software là trở thành một doanh nghiệp số hàng đầu và lọt vào top 50 công ty hàng đầu thế giới về cung cấp dịch vụ, giải pháp chuyển đổi số toàn diện vào năm 2030.
- Sứ mệnh: FPT Software cam kết mang trí tuệ Việt Nam ra thế giới và thay đổi cuộc sống con người thông qua công nghệ.

1.1.4. Giá trị cốt lõi

Giá trị cốt lõi của FPT Software được tóm gọn trong sáu từ: "Tôn, Đổi, Đồng - Chí, Gương, Sáng". Đây là nền tảng tạo nên tinh thần và sức mạnh của FPT, thúc đẩy sự nỗ lực và sáng tạo của lãnh đạo, cán bộ nhân viên vì lợi ích chung của cộng đồng, khách hàng, cổ đông và các bên liên quan khác. Các giá trị cốt lõi gồm: Tôn trọng, Đổi mới, Đồng đội, Chí công, Gương mẫu, Sáng suốt

Những giá trị này không chỉ là kim chỉ nam mà còn là động lực để FPT Software duy trì và phát triển bền vững, hướng tới mục tiêu vươn ra biển lớn.

CHƯƠNG II: TỔNG QUAN VỀ LÝ THUYẾT KIỂM THỬ PHẦN MỀM

2.1. Tổng quan về kiểm thử phần mềm

2.1.1. Khái niệm

Kiểm thử phần mềm là quá trình thao tác thực hiện một chương trình hoặc ứng dụng với mục đích tìm ra những lỗi của phần mềm.

2.1.2. Mục đích của kiểm thử phần mềm

- Mục đích:
 - Phát hiện và sửa chữa các lỗi do lập trình viên trong khi phát triển phần mềm
 - Nâng cao chất lượng của sản phẩm đảm bảo sản phẩm hoạt động ổn định, hiệu quả
 - Giảm thiểu và ngăn ngừa rủi ro tiềm ẩn do lỗi phần mềm gây ra trong quá trình sử dụng
 - Đảm bảo kết quả cuối cùng sẽ đáp ứng được các yêu cầu kinh doanh và yêu cầu của khách hàng
 - Xây dựng và duy trì lòng tin của khách hàng thông qua việc cung cấp các sản phẩm chất lượng cao

2.1.3. Bảy nguyên tắc kiểm thử phần mềm



Hình 2. 1: 7 Nguyên tắc kiểm thử phần mềm

- **Kiểm thử chứng minh sự hiện diện của lỗi**

Kiểm thử không thể chứng minh rằng sản phẩm không có lỗi. Nghĩa là sản phẩm luôn có lỗi cho dù kiểm thử bao nhiêu

- **Kiểm thử toàn bộ là không thể:**

Các sản phẩm có sự đa dạng và phức tạp khác nhau, việc kiểm thử toàn bộ gần như là không thể. Việc phân tích rủi ro và dựa trên mức độ ưu tiên của chúng, ta có thể tập trung việc kiểm thử vào một số điểm cần thiết, có nguy cơ lỗi cao hơn.

- **Kiểm thử càng sớm càng tốt:**

Nguyên tắc này yêu cầu bắt đầu kiểm thử phần mềm trong giai đoạn đầu của vòng đời phát triển phần mềm. Các hoạt động kiểm thử ở giai đoạn này sẽ giúp phát hiện bug sớm hơn. Điều này cho phép bàn giao phần mềm đúng thời hạn và chất lượng như dự kiến, giảm thiểu nhiều chi phí sửa lỗi.

- **Lỗi thường được phân bố tập trung (8/2):**

Thông thường, phần lớn lỗi tập trung vào các module, thành phần chức năng của hệ thống. Tìm kiếm lỗi quanh khu vực được xác định là cách hiệu quả nhất để thực hiện kiểm tra.

- **Nghịch lý thuốc trừ sâu:**

Trong kiểm thử, việc dùng đi dùng lại một bộ test case thì khả năng kiểm tra lỗi sẽ rất thấp. Do khi hệ thống càng hoàn thiện, những lỗi được tìm thấy lúc trước đã được sửa xong trong khi những trường hợp kiểm thử đã cũ.

- **Kiểm thử phụ thuộc vào ngữ cảnh:**

Kiểm thử ứng dụng web và ứng dụng di động bằng cách sử dụng chiến lược kiểm thử giống nhau, thì điều đó là sai lầm. Chiến lược để kiểm thử nên khác nhau và phụ thuộc vào chính ứng dụng đó. Chiến lược cho test web application phải khác với ứng dụng android mobile.

- **Quan niệm sai lầm về việc “hết lỗi”:**

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

2.1.4. Các giai đoạn của chu trình phát triển phần mềm

- **Giai đoạn 1: Lên kế hoạch và phân tích yêu cầu – Analysis**

Đây là giai đoạn quan trọng nhất của quy trình, các nhà phát triển sẽ nghiên cứu nhu cầu, mong muốn của khách hàng và người dùng. Từ đó xác định được mục tiêu, phạm vi, ngân sách, thời gian và các ràng buộc trong dự án. Xây dựng các tài liệu yêu cầu làm cơ sở cho các giai đoạn tiếp theo.

- **Giai đoạn 2: Thiết kế phần mềm – Design**

Giai đoạn nhà phát triển dựa theo các tài liệu yêu cầu để thiết kế ra kiến trúc, giao diện và các thành phần của phần mềm. Thiết kế phần mềm sẽ giúp nhà phát triển hiểu được cách thức hoạt động, tính năng và chức năng của sản phẩm. Thiết kế phần mềm cũng sẽ giúp nhà phát triển ước lượng được nguồn lực, công cụ và công nghệ cần thiết cho việc coding. Kết quả của giai đoạn này là một tài liệu thiết kế phần mềm (SDD) hoặc một tài liệu đặc tả thiết kế (SDD).

- **Giai đoạn 3: Tiến hành coding – Development**

Giai đoạn này, nhà phát triển sẽ bắt tay vào viết mã nguồn cho các thành phần của phần mềm theo thiết kế đã có. Coding là giai đoạn chiếm nhiều thời gian và công sức nhất trong quy trình phát triển phần mềm. Coding cũng là giai đoạn có thể gặp nhiều lỗi và sai sót nhất. Do đó, nhà phát triển cần tuân thủ các chuẩn mực lập trình (coding standards), áp dụng các kỹ thuật lập trình tốt (best practices) và sử dụng các công cụ hỗ trợ (tools) để giảm thiểu lỗi và tăng hiệu quả.

- **Giai đoạn 4: Kiểm thử - Testing**

Giai đoạn kiểm thử, nhà phát triển sẽ kiểm tra chất lượng của phần mềm bằng cách thực hiện các loại kiểm thử khác nhau, như kiểm thử đơn vị (unit testing), kiểm thử tích hợp (integration testing), kiểm thử hệ thống (system testing), kiểm thử chấp nhận (acceptance testing), kiểm thử hiệu năng (performance testing), kiểm thử bảo mật (security testing) và kiểm thử khả năng sử dụng (usability testing).

- **Giai đoạn 5: Triển khai - Deployment**

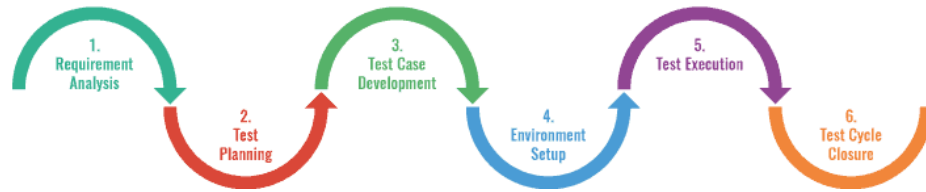
Giai đoạn này, nhà phát triển sẽ chuyển giao sản phẩm cho khách hàng và người dùng. Sản phẩm sẽ được cài đặt, cấu hình và vận hành trên môi trường thực tế. Nhà phát triển sẽ cung cấp các hướng dẫn, tài liệu và hỗ trợ kỹ thuật cho khách hàng và người dùng. Nhà phát triển cũng sẽ thu thập phản hồi và đánh giá của khách hàng và người dùng về sản phẩm.

- **Giai đoạn 6: Bảo trì - Maintenance**

Đây là giai đoạn cuối cùng và kéo dài nhất trong quy trình phát triển phần mềm. Ở giai đoạn này, nhà phát triển sẽ tiếp tục theo dõi, duy trì và cập nhật sản phẩm để đáp ứng nhu cầu thay đổi của khách hàng và người dùng. Nhà phát triển sẽ sửa chữa các lỗi, nâng cấp các tính năng, tăng cường bảo mật và hiệu suất của sản phẩm.

2.1.5. Quy trình kiểm thử

Software Testing Life Cycle (STLC)



Hình 2. 2: Quy trình kiểm thử

- **Bước 1: Requirement Analysis (Phân tích yêu cầu):**
 - Giai đoạn đầu tiên của quy trình kiểm thử phần mềm là Phân tích yêu cầu. Trong giai đoạn này, các tester sẽ phân tích tài liệu đặc tả yêu cầu để kiểm tra các yêu cầu do khách hàng đưa ra.
 - Yêu cầu được chia làm 2 dạng:
 - Functional (Chức năng): Mô tả tính năng
 - Non-Functional (Phi chức năng): Mô tả hiệu năng, tính bảo mật, tính hữu dụng của phần mềm.
- **Bước 2: Test Planning (Lập kế hoạch kiểm thử):**
 - Sau giai đoạn một, tester tiến hành Lập kế hoạch kiểm thử để kiểm tra xem phần mềm có đáp ứng các yêu cầu hay không. Kế hoạch kiểm thử là một tài liệu tổng quan về việc kiểm thử dự án bao gồm những thông tin sau:
 - Phạm vi kiểm thử, hướng tiếp cận, quy trình kiểm thử, tài nguyên và nhân lực test.
 - Các chức năng/module cần được kiểm tra; các công cụ và môi trường kiểm thử cần có.
 - Người thực hiện test, thời gian test
- **Bước 3: Test Case Development (Phát triển kịch bản kiểm thử):**
 - Ở bước này Tester bắt đầu xây dựng bộ Test Case dựa trên yêu cầu của phần mềm. Test Case cần mô tả được chi tiết dữ liệu đầu vào, hành động, kết quả mong đợi để xác định một chức năng của ứng dụng phần mềm có hoạt động đúng hay không.
- **Bước 4: Environment Setup (Thiết lập môi trường kiểm thử):**
 - Môi trường kiểm thử sẽ do developers tạo ra để deploy sản phẩm đã được hoàn thiện về phần lập trình.
 - Tester cần chuẩn bị smoke test case để kiểm tra môi trường cài đặt đã đáp ứng yêu cầu và sẵn sàng cho giai đoạn kiểm thử tiếp theo hay chưa.
- **Bước 5: Test Execution (Thực hiện kiểm thử):**
 - Tester sẽ thực thi dựa trên Test Case đã viết. Trong quá trình test, nếu phát hiện ra bug thì tester sẽ log lên các tool quản lý lỗi. Bug của lập trình viên

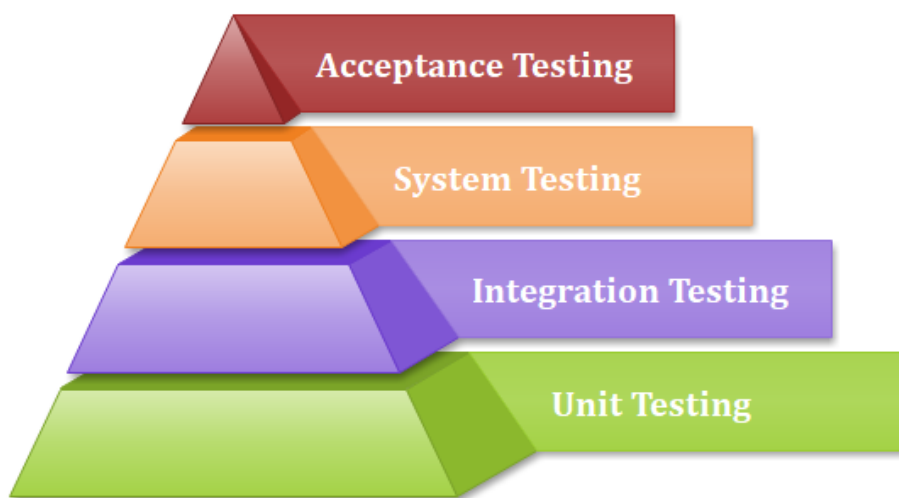
nào sẽ giao lại cho người đẩy xử lý. Khi nào developers fix bug xong, tester sẽ nhận lại và tiến hành kiểm thử lần 2.

- Trong cả quá trình kiểm thử phần mềm, tester ưu tiên kiểm tra chức năng chính trước, chức năng phụ và giao diện sẽ thực hiện test sau.

- **Bước 6: Test Cycle Closure (Kết thúc chu kỳ kiểm thử):**

- Ở giai đoạn cuối cùng, tester chuẩn bị báo cáo kết thúc kiểm thử, tổng hợp lại các chỉ số trong quá trình test. Cả team phát triển sẽ ngồi họp để đánh giá toàn bộ các tiêu chí xác định kiểm thử đã đủ hay chưa. Những tiêu chí này khác nhau tùy theo từng dự án, thông thường bao gồm:
 - Số lượng test case tối đa được thực thi Passed.
 - Tỷ lệ lỗi giảm xuống dưới mức nhất định.

2.1.6. Các cấp độ kiểm thử



Hình 2. 3: Các cấp độ kiểm thử

- **Unit Testing (Kiểm thử đơn vị):**

- Kiểm thử đơn vị là cấp độ kiểm thử cơ bản, thực hiện test từng module nhỏ trong hệ thống.
- Kiểm thử đơn vị có thể được thực hiện tách biệt với phần còn lại của hệ thống tùy thuộc vào mô hình vòng đời phát triển được chọn cho ứng dụng cụ thể đó.
- Mục đích: để xác nhận mỗi thành phần của phần mềm thực hiện đúng với thiết kế
- Kiểm thử đơn vị thường do lập trình viên thực hiện

- **Integration Testing (Kiểm thử tích hợp):**

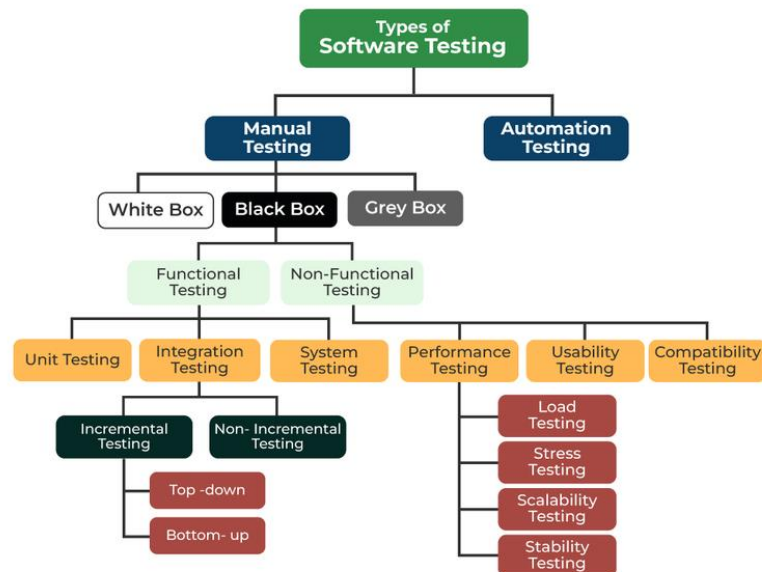
- kiểm thử tích hợp là tích hợp kiểm tra các module riêng lẻ với nhau thành một nhóm

- Tích hợp kiểm tra việc truyền dữ liệu giữa các module, tích hợp kiểm tra các hàm lại với nhau, các màn hình với nhau theo từng module hoặc theo chức năng
- Mục đích: để đảm bảo rằng hệ thống tích hợp đã sẵn sàng để thử nghiệm hệ thống
- Kiểm thử tích hợp được thực hiện sau khi kiểm tra đơn vị và trước khi kiểm tra hệ thống
- Integration testing được thực hiện bởi một người thử nghiệm cụ thể hoặc một nhóm kiểm thử
- Một số phương pháp kiểm thử tích hợp:
 - Phương pháp kiểm thử Bigbang
 - Phương pháp kiểm thử Topdown
 - Phương pháp kiểm thử Bottom up
 - Phương pháp kiểm thử Sandwich
- **System Testing (Kiểm thử hệ thống):**
 - System Testing là thực hiện kiểm thử một hệ thống đã được tích hợp hoàn chỉnh để xác minh rằng nó đúng yêu cầu của phần mềm.
 - Kiểm thử hệ thống nằm trong phạm vi kiểm thử hộp đen và do đó, không yêu cầu kiến thức về thiết kế bên trong của mã hoặc logic.
 - Kiểm thử hệ thống thường là thử nghiệm cuối cùng để xác minh rằng hệ thống được phân phối đáp ứng các đặc điểm kỹ thuật và mục đích của nó.
 - Kiểm thử hệ thống nên thực hiện kiểm thử chức năng và phi chức năng và được thực hiện bởi tester
- **Acceptance Testing (Kiểm thử chấp nhận):**
 - kiểm thử chấp nhận cũng khá giống kiểm thử hệ thống nhưng được thực hiện bởi khách hàng
 - Mục đích: đảm bảo phần mềm đáp ứng đúng yêu cầu của khách hàng. Sản phẩm nhận được sự chấp nhận từ khách hàng/ người dùng cuối.
 - Kiểm thử chấp nhận được chia thành 2 mức khác nhau:
 - Kiểm thử alpha: được thực hiện tại nơi phát triển phần mềm bởi những người trong tổ chức nhưng không tham gia phát triển phần mềm.
 - Kiểm thử beta: được thực hiện tại bởi khách hàng/ người dùng cuối tại địa điểm của người dùng cuối.

2.1.7. Các loại kiểm thử

- **Kiểm thử chức năng (Functional testing):**
 - Kiểm thử chức năng là một loại kiểm thử hộp đen (black box) và test case của nó được dựa trên đặc tả của ứng dụng phần mềm/ thành phần đang test. Các chức năng được test bằng cách nhập vào các giá trị và kiểm tra kết quả đầu ra, ít quan tâm đến cấu trúc bên trong của ứng dụng.
 - Các loại kiểm thử chức năng:

- Kiểm thử đơn vị (Unit Testing)
- Kiểm thử khói (Smoke Testing)
- Kiểm thử độ tinh tảo (Sanity Testing)
- Kiểm thử giao diện (Interface Testing)
- Kiểm thử tích hợp (Integration Testing)
- Kiểm thử hệ thống (Systems Testing)
- Kiểm thử hồi quy (Regression Testing)
- Kiểm thử chấp nhận (Acceptance testing)



Hình 2. 4: Các loại kiểm thử

- **Kiểm thử phi chức năng (Non-Functional testing):**

- Kiểm thử phi chức năng cùng giống kiểm thử chức năng ở chỗ là thực hiện được ở mọi cấp độ kiểm thử, Kiểm thử phi chức năng xem xét các hành vi bên ngoài của phần mềm. 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 khả năng sử dụng (Usability testing)
- Kiểm thử bảo trì (Maintainability testing)
- Kiểm thử độ tin cậy (Reliability testing)
- Kiểm thử tính tương thích (Portability testing)

- **Kiểm thử cấu trúc/kiến trúc phần mềm (Structural testing):**

- Kiểm thử cấu trúc có thể xảy ra ở bất kỳ mức độ kiểm thử nào, được áp dụng chủ yếu ở kiểm thử thành phần, tích hợp. Phương pháp kiểm thử cấu trúc cũng có thể áp dụng ở các mức độ như kiểm thử tích hợp hệ thống hoặc kiểm thử chấp nhận.
- **Kiểm thử xác nhận (Confirmation testing) và kiểm thử hồi quy (Regression testing):**
 - *Kiểm thử xác nhận:* Sau khi một lỗi được phát hiện và sửa chữa, phần mềm được kiểm thử lại để xác nhận lỗi ban đầu đã được khắc phục gọi là kiểm thử xác nhận. Khi thực hiện kiểm thử xác nhận phải đảm bảo rằng các thử nghiệm được thực hiện giống như lần đầu tiên sử dụng, sử dụng các inputs, dữ liệu và môi trường giống nhau.
 - *Kiểm thử hồi quy:* Mục đích của kiểm thử hồi quy là xác minh rằng sửa đổi trong phần mềm hoặc môi trường không gây ra các phản ứng phụ không mong muốn và hệ thống vẫn đáp ứng các yêu cầu.

2.2. Tổng quan về Test Case

2.2.1. Khái niệm

Test case là một trường hợp cần kiểm thử, nó bao gồm các thao tác/hành động trên hệ thống, điều kiện cần (tiền quyết), các giá trị đầu vào, và kết quả mong đợi. Một test case thì nên chỉ kiểm tra một trường hợp, một khía cạnh cụ thể nào đó chứ đừng lan man.

Thành phần của 1 Test Case thông thường bao gồm:

- Test case ID: Để dễ dàng xác định và phân biệt test case với nhau
- Group check: Nhóm các item cần test (UI, Validate, Business,...)
- Item check: Các Item test
- Pre-condition: Điều kiện tiên quyết, điều kiện cần để có của test case
- Description: Để mô tả tóm tắt trường hợp cần kiểm thử.
- Step to perform: Các bước thực hiện test case
- Expected result: Kết quả mong đợi, thường dựa vào tài liệu mô tả để xác định kết quả mong đợi này.
- Test data: Dữ liệu được chuẩn bị để thực hiện test case
- Priority: Độ ưu tiên để thực hiện test case
- Status: Ghi kết quả của test case (Passed, Failed,...)
- Who check: Người thực hiện test case
- Date check: Ngày test
- Device: Thiết bị thực hiện test case

2.2.2. Các kỹ thuật thiết kế Test Case phổ biến

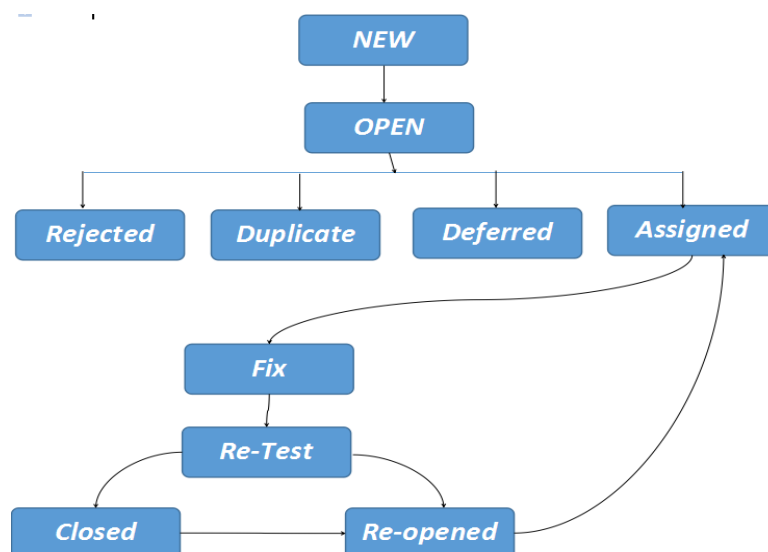
- **Phân vùng tương đương:**

- Khái niệm: Phân vùng tương đương là một kỹ thuật trong đó các đầu vào của hệ thống được chia thành các nhóm tương đương, sao cho hệ thống xử lý các đầu vào trong cùng một nhóm theo cùng một cách. Mỗi nhóm tương đương được coi là một phân vùng và chỉ cần kiểm thử một giá trị đại diện từ mỗi phân vùng.
- Mục đích: Giảm số lượng test case cần thiết để kiểm thử phần mềm một cách hiệu quả mà vẫn đảm bảo độ bao phủ kiểm thử cao.
- **Phân tích giá trị biên:**
 - Khái niệm: Là kỹ thuật thiết kế test case tập trung vào các giá trị ở ranh giới của các phân vùng tương đương. Ý tưởng là lỗi thường xảy ra ở các giá trị biên hơn là các giá trị trung tâm.
 - Mục đích: Tìm ra các lỗi xảy ra ở ranh giới của các phạm vi đầu vào
- **Bảng quyết định Decision Table:**
 - Khái niệm: Bảng quyết định là một kỹ thuật thiết kế test case sử dụng bảng để biểu diễn các kết hợp đầu vào và điều kiện, cũng như các hành động tương ứng. Mỗi cột trong bảng quyết định đại diện cho một quy tắc kinh doanh cụ thể, thể hiện một kịch bản kiểm thử.
 - Mục đích: Xác định tất cả các kết hợp có thể của điều kiện và hành động để đảm bảo kiểm thử đầy đủ.

2.3. Tổng quan về Bug và Report Bug

2.3.1. Khái niệm Bug

- Bug được định nghĩa là những lỗi phần mềm hoặc hệ thống chương trình máy tính làm cho kết quả trả về không chính xác hoặc hoạt động không như mong muốn.



Hình 2. 5: Vòng đời của Bug

- Các trạng thái của Bug:
 - NEW trạng thái là khi tester tìm ra bug. Sau khi tìm ra bug thì tester cần báo cho Teamlead hoặc developer bằng cách log bug
 - OPEN là trạng thái bug được log lên bởi tester. Team lead sẽ xác định xem đó có phải là bug hay không và đánh các trạng thái thông báo về Bug đó
 - REJECTED: Khi Team lead xác nhận rằng bug đó không hợp lệ thì sẽ được đánh dấu là Rejected. Còn nếu bug đó là hợp lệ thì sẽ xác định xem bug đó đã tồn tại hay chưa
 - DUPLICATE: Nếu bug đã tồn tại thì sẽ đánh dấu nó là DUPLICATE. Nếu bug chưa tồn tại thì sẽ xác định xem bug đó có trong phạm vi xử lý hay không (scope)
 - DEFERRED: Nếu bug không trong phạm vi xử lý thì sẽ được chuyển sang trạng thái Deferred
 - ASSIGNED: Nếu bug trong phạm vi xử lý thì sẽ được chuyển cho developer để tiến hành fix bug
 - FIX: Khi nhận được bug từ Team lead, developer sẽ thực hiện fix bug cho đúng với yêu cầu, và đẩy lại cho tester kiểm tra lại lỗi đó (Bug phát sinh từ Code của ai thì sẽ do người đó Fix)
 - RE-TESTING : Sau khi fix xong bug, thì bug sẽ được assign lại cho tester , tester sẽ test lại xem nó đã chạy đúng hay chưa.
 - CLOSED: Khi bug được xác minh lại là đã chạy đúng như yêu cầu thì tester sẽ đánh dấu nó là CLOSED.
 - RE-OPENED: Nếu như tester test lại mà bug đó vẫn xảy ra thì bug đó sẽ được gán là Re-Open và assign lại cho developer để fix lại

2.3.2. Report Bug

Là những mô tả lỗi phần mềm khi các lập trình viên thực thi test phần mềm đó. Các tester thường thực hiện bug report trên các phần mềm quản lý task như Redmine, Jira,...

Nội dung của 1 Report Bug bao gồm:

- Bug title: Mô tả ngắn gọn, xúc tích nhưng vẫn đảm bảo nội dung được đầy đủ về bug
- Description: Thể hiện được hiện tượng xảy ra của Bug một cách tổng quan nhất và chi tiết nhất.
- Pre-condition: Điều kiện tiên quyết, điều kiện cần để test case này chạy được
- Step to perform: Các bước thực hiện để tái hiện lỗi

- Actual Result: Kết quả thực tế
- Expected Result: Kết quả mong đợi
- Evidence: Bằng chứng
- Môi trường test cụ thể
- Ngày bắt đầu: Ngày thực hiện fix bởi developer
- Ngày hết hạn: Ngày sửa thực tế kết thúc

2.4. Sự khác nhau giữa Manual Testing và Automation Testing

2.4.1. Khái niệm

- Manual Testing: là việc thử nghiệm một phần mềm hoàn toàn được làm bằng tay bởi người tester. Nó được thực hiện nhằm phát hiện lỗi trong phần mềm đang được phát triển. Trong manual testing, tester sẽ thực hiện các trường hợp kiểm thử và tạo báo cáo kiểm thử hoàn toàn thủ công mà không có bất kỳ sự trợ giúp của công cụ tự động nào.
- Automation Testing: là phương pháp kiểm thử tự động. Người tester sẽ phải viết các kịch bản kiểm thử sau đó sử dụng các tool hỗ trợ để thực hiện kiểm thử, phương pháp này sẽ giúp việc kiểm thử hiệu quả và tốn ít thời gian hơn. Automation testing giúp chạy các kịch bản kiểm thử lặp lại nhiều lần và các task kiểm thử khác khó thực hiện bằng tay như performance testing và stress testing.

2.4.2. Ưu nhược điểm

- Manual Testing:
 - **Ưu điểm:**
 - Dễ dàng cho việc test giao diện, người tester sẽ có phản hồi nhanh và trực quan về giao diện ứng dụng
 - Mất ít chi phí cho các tool tự động và quy trình
 - Khi có thay đổi nhỏ manual testing manual testing không bị mất nhiều thời gian để thay đổi các trường hợp kiểm thử
 - **Nhược điểm:**
 - Kết quả kiểm thử ít tin cậy hơn vì có thể sai sót do yếu tố con người
 - Quá trình thực hiện các ca kiểm thử không được ghi lại, do vậy nó không có tính tái sử dụng
 - Với một số task khó thực hiện thủ công như performance testing và stress testing thì manual testing rất khó để thực hiện
- Automation Testing:

○ **Ưu điểm:**

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

○ **Nhược điểm:**

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

2.4.3. Bảng so sánh chi tiết

Parameter	Automation Testing	Manual Testing
Definition	Automation testing sử dụng các tool để thực hiện các trường hợp kiểm thử	Thực hiện kiểm thử hoàn toàn thủ công không có sự trợ giúp của bất kỳ công cụ tự động nào, được thực hiện bởi tester
Processing time	Thời gian kiểm thử rút ngắn hơn so với manual testing	Manual testing tốn nhiều thời gian và nguồn nhân lực
Exploratory Testing	Không cho phép kiểm thử khám phá	Có thể kiểm thử khám phá trong manual testing
Reliability	Kết quả kiểm thử đáng tin cậy vì nó được thực hiện bằng các tool và các kịch bản	Kết quả kiểm thử không đáng tin cậy vì có khả năng xảy ra lỗi của con người

Parameter	Automation Testing	Manual Testing
UI change	Cần chỉnh sửa kịch bản kể cả những thay đổi nhỏ trong giao diện	Những thay đổi nhỏ như thay đổi về id, class sẽ không cản trở quá trình kiểm thử
Investment	Cần phải đầu tư cho các công cụ kiểm thử	Cần đầu tư về nguồn nhân lực
Test Report Visibility	Tất cả các bên liên quan có thể Login vào hệ thống xem kết quả đã kiểm thử	Kết quả được lưu lại trong excel hoặc word
Performance Testing	Được thực hiện trong kiểm thử Load testing, stress testing	Không khả thi trong kiểm thử Load testing, stress testing
Parallel Execution	Có thể thực hiện song song trên các nền tảng vận hành khác nhau và giảm thời gian thực hiện kiểm thử	Kiểm thử song song trên các nền tảng khác nhau sẽ phải tăng nguồn nhân lực
Programming knowledge	Yêu cầu phải có kiến thức lập trình	Không cần có kiến thức lập trình vẫn có thể thực hiện
Ideal approach	Automation testing rất hữu ích khi thường xuyên thực hiện chạy lại một kịch bản nhiều nhiều lần	Manual testing hữu ích khi chạy bộ test case một hoặc hai lần

Bảng 1: Bảng so sánh Automation và Manual Testing

2.5. Tổng quan về vị trí việc làm

2.5.1. Khái niệm Tester

Tester là người chịu trách nhiệm kiểm tra, đánh giá và xác nhận chất lượng của phần mềm trước khi nó được phát hành cho người dùng cuối. Đảm bảo rằng sản phẩm, phần mềm đáp ứng được các tiêu chí về chất lượng, hoạt động đúng đắn và đáp ứng nhu cầu của người sử dụng.

Công việc của tester không chỉ dừng lại ở việc kiểm tra sản phẩm, mà còn giúp tạo ra các chiến lược để cải thiện quy trình phát triển phần mềm trong tương lai. Điều này đóng góp quan trọng vào việc đảm bảo rằng dự án được phát triển đáp ứng được mong đợi của khách hàng và có hiệu suất cao trong môi trường thực tế.

2.5.2. Các kỹ năng cần có của một Tester

- Kỹ năng phân tích: Kỹ năng phân tích sẽ giúp các Tester có thể chia một hệ thống phức tạp thành từng đơn vị nhỏ nhằm phân tích hiểu rõ hơn về từng yếu tố riêng lẻ.
- Kỹ năng học hỏi: Tester phải tìm hiểu học hỏi, phân tích các tình huống thông qua đồng nghiệp, các hội nhóm để tạo cơ hội phát triển cho bản thân mình.
- Kỹ năng giao tiếp: Công việc Tester mang tính chất làm việc theo nhóm và dự án, không hoạt động độc lập. Do đó kỹ năng giao tiếp sẽ giúp các tester giải quyết được các mâu thuẫn trong nhóm, giúp truyền đạt ý của mình tốt hơn.
- Kỹ năng làm việc nhóm: Kỹ năng làm việc nhóm sẽ dễ dàng giúp các bạn kết nối được với các thành viên trong nhóm với nhau, kết nối với các Developer. Tester sẽ là cầu nối giữa các Backend Developer, Frontend Developer với người sử dụng phần mềm.
- Ngoài những kỹ năng trên thì một Tester giỏi cần trau dồi kỹ năng tiếng anh, các kiến thức về ngôn ngữ lập trình và cơ sở dữ liệu. Bên cạnh đó còn cần rèn luyện tính cách cẩn thận, tỉ mỉ, nhạy bén.

2.5.3. Cơ hội nghề nghiệp

Với sự gia tăng của các ứng dụng di động, web và phần mềm doanh nghiệp, nhu cầu về đảm bảo chất lượng phần mềm ngày càng cao. Nghề Tester đang thu hút sự chú ý mạnh mẽ của giới trẻ bởi tiềm năng phát triển và mức thu nhập hấp dẫn. Điều này cũng khiến cho mức độ cạnh tranh giữa các Tester ngày càng cao, đòi hỏi các ứng viên không những phải có chuyên môn tốt mà còn yêu cầu ứng viên phải có khả năng giao tiếp bằng ngoại ngữ, khả năng học hỏi, có tư duy sáng tạo và nhiều kỹ năng mềm quan trọng khác.

Mức lương dành cho Tester phụ thuộc vào kinh nghiệm và năng lực của bản thân

- Mức lương giữa Manual Tester và Automation Tester sẽ có sự chênh lệch khá lớn. Automation Tester sẽ có mức lương cao hơn vì đây là vị trí yêu cầu các Tester phải có thêm kiến thức chuyên sâu về lập trình
- Chênh lệch mức lương cao giữa các cấp độ Tester là khá lớn, thấp nhất là Intern Tester và cao nhất là Senior Tester
- Tester có ngoại ngữ như tiếng Anh, tiếng Nhật,... là một lợi thế vì có thể tham gia vào nhiều loại dự án khác nhau, đặc biệt là dự án nước ngoài vì vậy mức lương có thể cao hơn.

CHƯƠNG III: ĐẶC TẢ WEBSITE MOLLEE

3.1. Tổng quan về Website MOLLEE

Website bán hàng thời trang MOLLEE ra đời nhằm mục đích cung cấp cho khách hàng một nền tảng mua sắm quần áo, phụ kiện tiện lợi và hiện đại. Trang web cung cấp cho người dùng trải nghiệm mua sắm dễ dàng và thuận tiện, cho phép người dùng khám phá thông tin sản phẩm, giá bán, hình ảnh thực tế một cách chi tiết. Với giao diện thân thiện và dễ dùng, MOLLEE giúp người dùng dễ dàng tìm kiếm, lựa chọn và đặt mua sản phẩm một cách nhanh chóng.

- Thành phần và chức năng của Website
 - Trưng bày hàng hóa
Hiện thị danh mục hàng hóa, hàng hóa theo loại, xem chi tiết và hàng cùng loại
 - Quản lý giỏ hàng
Quản lý các mặt hàng đã chọn (Thêm, sửa, xóa sản phẩm)
 - Quản lý Username
Sign Up, Login, quên Password...
 - Bảo mật ứng dụng
Chỉ được phép đặt hàng và quản lý thông tin riêng tư sau khi đã Login
 - Đặt hàng và quản lý đơn hàng
Tạo đơn hàng, liệt kê và xem chi tiết
- Đề tài chỉ thực hiện kiểm thử cho 3 chức năng đó là: Login, Sign Up và Quản lý giỏ hàng

3.2. Sơ đồ Use case tổng quát

3.2.1. Vai trò của từng tác nhân

Website MOLLEE có 2 tác nhân chính là Admin và Employee. Vai trò của từng tác nhân cụ thể như sau

- Quản trị viên (Admin):
 - Login
 - Đăng xuất
 - Quên Username
 - Quản lý sản phẩm
 - Quản lý đơn hàng
 - Quản lý vận chuyển
 - Quản lý Username khách hàng
 - Quản lý sự kiện, khuyến mãi
- Khách hàng (Employee):

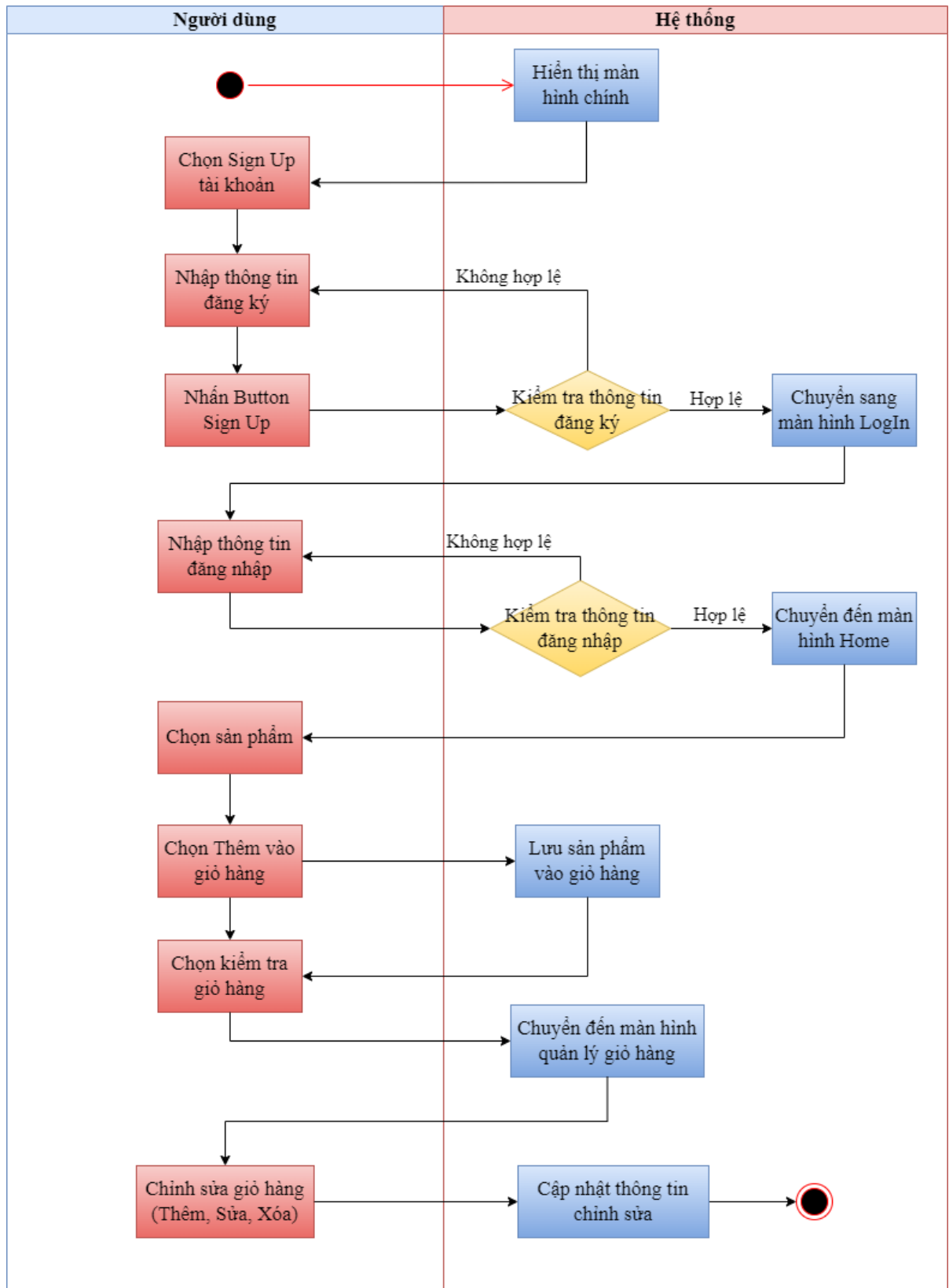
- Sign Up
- Login
- Đăng xuất
- Quên Username
- Xem sản phẩm
- Quản lý giỏ hàng
- Đặt hàng
- Quản lý thông tin cá nhân

3.2.2. Sơ đồ Use case



Hình 3. 1: Use Case hệ thống

3.3. Workflow

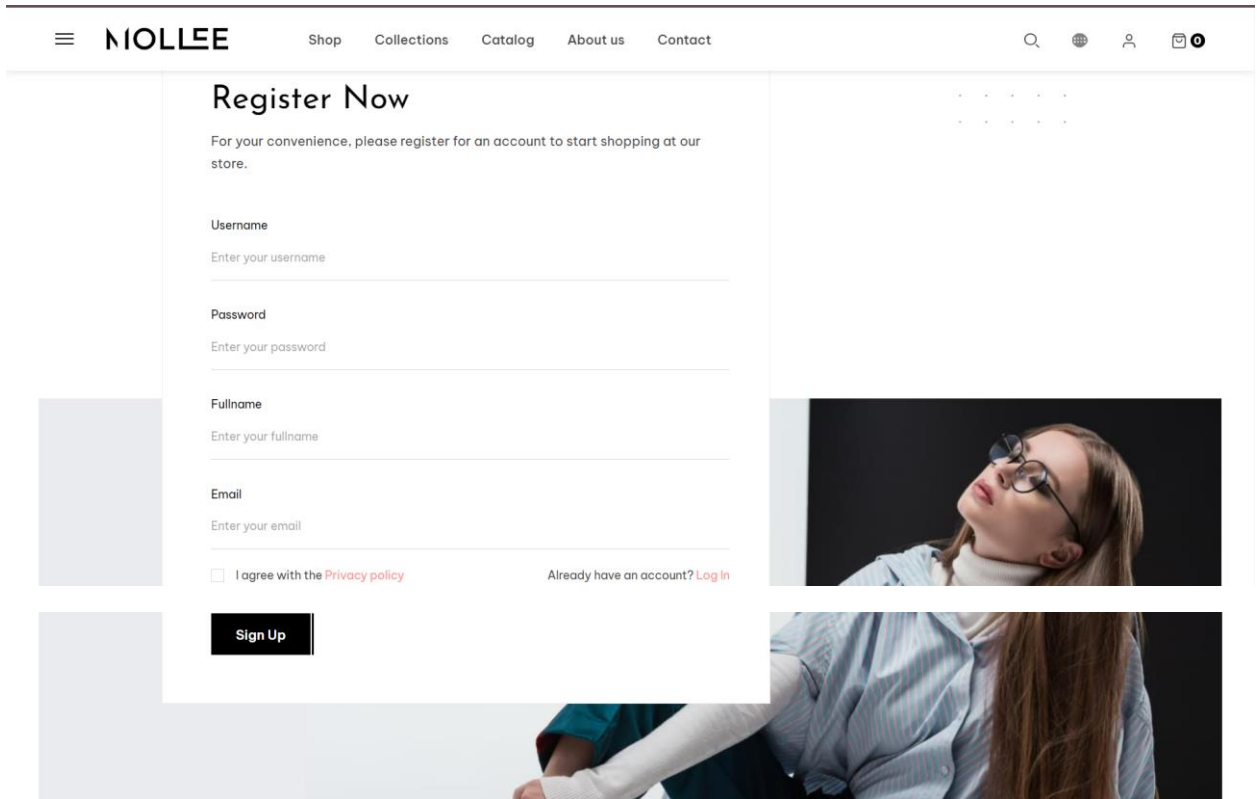


Hình 3. 2: Workflow Hệ thống

3.4. Đặc tả yêu cầu, thiết kế hệ thống

Đề tài này chỉ thực hiện kiểm thử cho 3 chức năng đó là: Login, Đăng kí và Quản lý giỏ hàng. Do vậy, phần phân tích dưới đây chỉ tập trung làm rõ yêu cầu của 3 chức năng này.

3.4.1. Phân tích Use case Sign Up



Hình 3. 3: Màn hình Sign Up

❖ Đặc tả yêu cầu chức năng Đăng ký (Login)

Use case name	Register
Description	Là một Khách hàng, tôi muốn Sign Up Username để có thể Login vào hệ thống
Actor	Khách hàng
Priority	High
Triggers	Người dùng truy cập vào website MOLLEE, tại màn hình Register
Pre-conditions	<ul style="list-style-type: none">- Người dùng chưa có Username Login- Người dùng truy cập vào màn hình Register
Post-conditions	<ul style="list-style-type: none">- Người dùng Sign Up thành công và chuyển đến màn hình Login- Hệ thống hiển thị các tính năng cho người dùng sử dụng

Main flow	<ol style="list-style-type: none"> 1. Người dùng truy cập vào chức năng “Register” 2. Nhập thông tin: Username vào textbox “Username”, Password vào textbox “Password”, Họ và tên vào textbox “Fullname”, email vào textbox “Email” 3. Người dùng nhấn vào Check box “Privacy policy” 4. Người dùng nhấn Enter trên bàn phím hoặc nhấn vào nút “Register” 5. Hệ thống kiểm tra tính hợp lệ của thông tin 6. Hệ thống sẽ chuyển người dùng đến “Home” 7. Người dùng có thể thực hiện các chức năng dành cho Khách hàng tại màn hình mới
Alternative flows	N/A
Exception flows	<p>2a. Nếu thông tin Sign Up không hợp lệ sẽ hiển thị thông báo lỗi</p> <p>2b. Nếu thông tin Sign Up bị trùng với dữ liệu lưu trữ trong hệ thống sẽ hiển thị thông báo lỗi</p> <p>5a. Nếu thông tin Sign Up được xác thực hợp lệ sẽ hiển thị thông báo Sign Up thành công</p>
Business rules	<ul style="list-style-type: none"> - Username nhập vào phải có định dạng thỏa mãn điều kiện không có dấu và khoảng trống trong Username - Password phải ít nhất 6 ký tự - Email nhập vào có định dạng: prefix@domain name. Phải chứa prefix và domain name, ký tự @ nằm giữa prefix và domain name, có tối thiểu 1 dấu chấm
Non-functional requirements	N/A

Bảng 2: Đặc tả yêu cầu chức năng Đăng ký

❖ Yêu cầu và điều kiện

No	Item name	Type of item	R/O	Min-max	Note
1	Username	Text	R	1-60	- Username có tối đa 60 ký tự
2	Password	Text	R	6-60	- Hiển thị Password theo định dạng dấu chấm
3	Họ và tên	Text	R		
4	Email	Text	R		<ul style="list-style-type: none"> - Email có định dạng prefix@domain name. - Phải chứa prefix và domain name, ký tự @ nằm giữa prefix và domain

					name, có tối thiểu 1 dấu chấm
5	Privacy policy	Check box	R		
		Hyperlink	O		Chuyển hướng đến trang tài liệu Privacy policy
6	Login	Hyperlink	O		Chuyển hướng đến trang Login
7	Sign Up	Button	R		Chuyển hướng đến trang Login

Bảng 3: Yêu cầu và điều kiện của màn hình Sign Up

- **Điều kiện:**

- Nếu Username khớp với dữ liệu sẵn có trong cơ sở dữ liệu, người dùng Sign Up không thành công, màn hình hiện thông báo lỗi ***"This username already exists."***
- Username là trường bắt buộc và phải có độ dài và định dạng hợp lệ, nếu không sẽ hiển thị thông báo lỗi:
 - Độ dài và định dạng không hợp lệ: ***"Invalid username!"***
 - Bỏ trống: ***"Please fill in this field."***
- Password phải được ẩn bằng dấu chấm, Password là trường bắt buộc và phải có độ dài hợp lệ, nếu không sẽ hiển thị thông báo lỗi:
 - Độ dài không hợp lệ: ***"Invalid Password!"***
 - Bỏ trống: ***"Please fill in this field"***
- Họ và tên là trường bắt buộc phải nhập, nếu không sẽ hiển thị thông báo lỗi ***"Please fill in this field."***
- Email là trường bắt buộc nhập, Email phải có định dạng prefix@domain name. Phải chứa prefix và domain name, ký tự @ nằm giữa prefix và domain name, có tối thiểu 1 dấu chấm
 - Định dạng không hợp lệ
 - Email thiếu prefix: ***"Please enter the part after '@'. 'Email' is incomplete."***
 - Email thiếu ký tự "@": ***"Please include '@' in email address. Email is missing '@'"***
 - Email thiếu domain name: ***"Please enter the part after '@'. Email is incomplete."***
 - Email chứa khoảng trống không hợp lệ: ***"The part before '@' cannot contain the ' ' symbol."***
 - Bỏ trống: ***"Please fill in this field."***
- Privacy policy là trường bắt buộc, nếu không click vào Check box, Sign Up sẽ không thành công
- Khi người dùng click vào nút Sign Up, thông tin Sign Up sẽ được gửi đến hệ thống để xác thực người dùng

- Sau khi hiển thị thông báo Sign Up thành công, hệ thống sẽ chuyển hướng đến màn hình Sign Up
- Chỉ Sign Up thành công khi người dùng click vào Check box “Privacy policy”. Nếu không sẽ hiển thị thông báo “*Account registration failed!*”
- Nút button phải được kích hoạt bằng cách nhấn phím Enter trên bàn phím hoặc click vào button “Sign Up”
- Nút button bị disable khi người dùng để trống trường dữ liệu hoặc nhập data không hợp lệ vào 1 trong các trường nhập dữ liệu.

3.4.2. Phân tích Use case Login

Hình 3. 4: Màn hình Login

❖ Đặc tả yêu cầu chức năng Đăng nhập (Login)

Use case name	Login
Description	Là một Khách hàng, tôi muốn Login vào hệ thống
Actor	Khách hàng
Priority	High
Triggers	Người dùng truy cập vào website MOLLEE, tại màn hình Login
Pre-conditions	<ul style="list-style-type: none"> - Người dùng phải có Username đã được Sign Up trước đó - Người dùng truy cập vào màn hình Login
Post-conditions	<ul style="list-style-type: none"> - Người dùng Login hệ thống thành công và chuyển đến màn hình “Home” - Hệ thống hiển thị các tính năng cho người dùng sử dụng

Main flow	<ol style="list-style-type: none"> 1. Người dùng truy cập vào chức năng “Login” 2. Nhập thông tin Username vào textbox “Username”, Password vào textbox “Password” 3. Người dùng nhấn vào Check box “Remember me?” 4. Người dùng nhấn Enter trên bàn phím hoặc nhấn vào nút “Login” 5. Hệ thống kiểm tra tính hợp lệ của thông tin 6. Hệ thống sẽ chuyển người dùng đến “Home” 7. Người dùng có thể thực hiện các chức năng dành cho Khách hàng
Alternative flows	N/A
Exception flows	<p>2a. Nếu thông tin Login không hợp lệ sẽ hiển thị thông báo lỗi:</p> <p>2b. Nếu người dùng quên Password, nhấn vào hyperlink “Fogot Password?”, hệ thống sẽ chuyển đến chức năng Fogot Password.</p> <p>2c. Nếu người dùng chưa có Tài khoản, nhấn vào hyperlink “Sign Up”, hệ thống sẽ chuyển đến chức năng Sign Up</p> <p>3a. Người dùng có thể lựa chọn không nhấn vào Check box “Remember me?”</p> <p>4a. Nếu “Username” sai và “Password” sai, hệ thống hiển thị thông báo: “Login không thành công. Username không tồn tại!”</p>
Business rules	<ul style="list-style-type: none"> - Username nhập vào phải có có định dạng thỏa mãn điều kiện không có dấu và khoảng trống. - Password phải ít nhất 6 ký tự
Non-functional requirements	N/A

Bảng 4: Đặc tả yêu cầu chức năng Login

❖ Yêu cầu và điều kiện

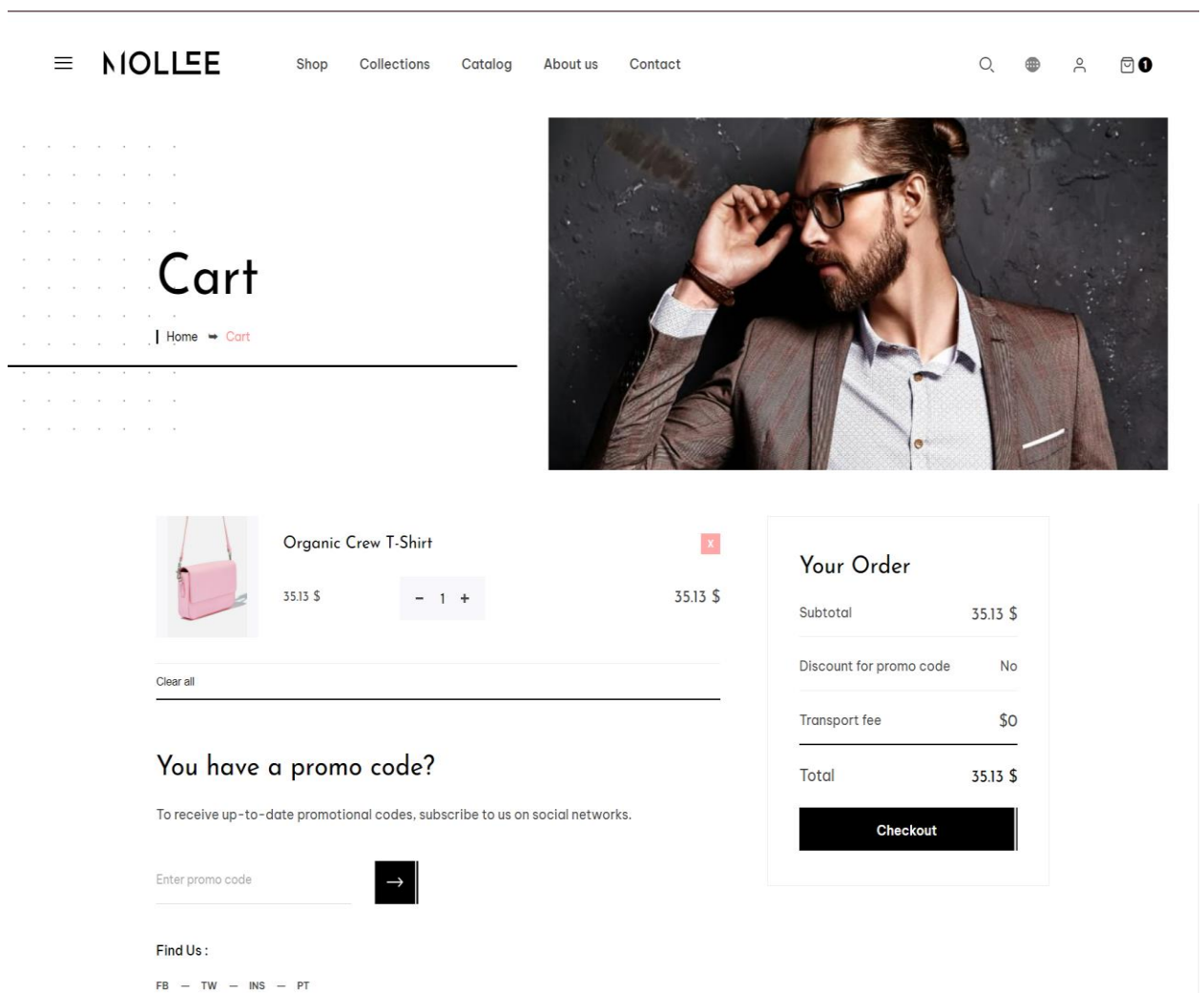
No	Item name	Type of item	R/O	Min-max	Note
1	Username	Text	R		
2	Password	Text	R		
3	Remember me?	Check box	O		Cho phép người dùng ghi nhớ thông tin Login trong lần Login tiếp theo
4	Login	Button	R		Xác nhận thông tin Login và xác thực người dùng
5	Fogot Password?	Hyperlink	O		Chuyển hướng đến trang đặt lại Password
6	Sign Up	Hyperlink	O		Chuyển hướng đến trang Sign Up

Bảng 5: Yêu cầu và điều kiện với màn hình Login

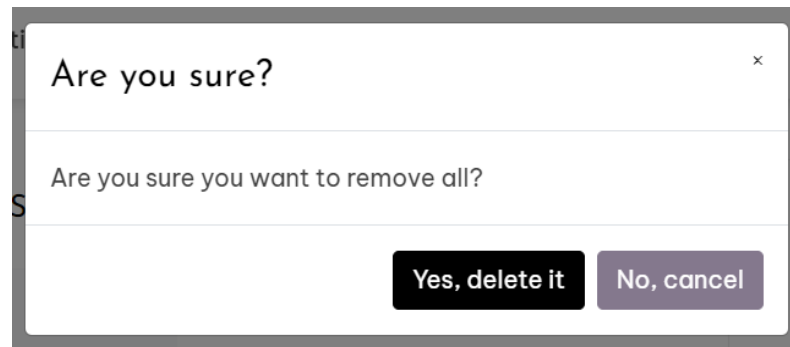
- **Điều kiện:**

- Nếu **Username** và **Password** khớp với cơ sở dữ liệu, người dùng sẽ Login thành công và di chuyển đến màn hình **Home**.
- Người dùng nhập **Username** và **Password** hợp lệ đã đăng ký
- Khi người dùng click vào nút **Login**, thông tin Login sẽ được gửi đến hệ thống để xác thực người dùng
 - Nếu xác thực không thành công, không tồn tại thông tin Tài khoản được nhập, hiển thị thông báo lỗi “*This username does not exist!*”. Người dùng vẫn ở màn hình **Login**
 - Nếu xác thực đúng, hệ thống sẽ chuyển hướng đến màn hình Home
 - Nút button **Log In** chỉ hoạt động khi người dùng điền đủ thông tin vào hai trường **Username** và **Password**. Nếu thiếu 1 trong 2 có giá trị Null sẽ hiển thị thông báo “*Please fill in this field!*”
 - Nút button phải được kích hoạt bằng cách nhấn phím Enter trên bàn phím hoặc click vào button “**Log In**”

3.4.2. Phân tích Use case Quản lý giỏ hàng



Hình 3. 5: Màn hình giỏ hàng



Hình 3. 6: Dialog “Are you sure?”

❖ **Đặc tả yêu cầu chức năng Chỉnh sửa số lượng sản phẩm trong giỏ hàng**

Use case name	Chỉnh sửa số lượng sản phẩm trong giỏ hàng
Description	Là một Khách hàng, tôi muốn cập nhật lại số lượng sản phẩm trong giỏ hàng, để có thể điều chỉnh số lượng các sản phẩm cần mua.
Actor	Khách hàng
Priority	High
Triggers	Người dùng truy cập vào giỏ hàng và chọn sửa số lượng sản phẩm trong giỏ hàng
Pre-conditions	Người dùng đã thêm ít nhất một sản phẩm vào giỏ hàng
Post-conditions	<ul style="list-style-type: none"> - Số lượng sản phẩm trong giỏ hàng được cập nhật - Tổng giá trị giỏ hàng được cập nhật dựa trên số lượng sản phẩm
Main flow	‘1. Người dùng truy cập vào giỏ hàng ‘2. Hệ thống hiển thị danh sách sản phẩm trong giỏ hàng ‘3. Người dùng nhập số lượng mới vào textbox “ Số lượng sản phẩm ” hoặc nhấn vào button quantity selector của sản phẩm muốn chỉnh sửa để điều chỉnh số lượng ‘ 4. Hệ thống tự động tính toán và cập nhật tổng giá trị giỏ hàng dựa trên số lượng mới
Alternative flows	N/A
Exception flows	3a. Nếu người dùng nhập số lượng không hợp lệ sẽ hiển thị thông báo lỗi

	3b. Nếu người dùng click quantity selector đến số lượng tối đa (10) và tối thiểu (1) nút button sẽ bị disable
Business rules	N/A
Non-functional requirements	N/A

Bảng 6: Đặc tả yêu cầu chức năng thêm sản phẩm vào giỏ hàng

❖ **Đặc tả yêu cầu chức năng Xóa sản phẩm trong giỏ hàng**

Use case name	Xóa sản phẩm trong giỏ hàng
Description	Là một Khách hàng, tôi muốn xóa sản phẩm trong giỏ hàng, để có thể điều chỉnh danh sách các sản phẩm cần mua.
Actor	Khách hàng
Priority	High
Triggers	Người dùng truy cập vào giỏ hàng và chọn xóa sản phẩm
Pre-conditions	Người dùng đã thêm ít nhất một sản phẩm vào giỏ hàng
Post-conditions	<ul style="list-style-type: none"> - Sản phẩm được xóa khỏi giỏ hàng của người dùng - Danh sách các sản phẩm trong giỏ hàng được cập nhật lại
Main flow	<p>‘1. Người dùng truy cập vào giỏ hàng</p> <p>‘2. Hệ thống hiển thị danh sách sản phẩm trong giỏ hàng</p> <p>‘3. Người dùng click vào icon “X” bên cạnh tên sản phẩm muốn xóa để xóa từng sản phẩm một, hoặc nhấn vào button “Clear all” để xóa hết toàn bộ sản phẩm</p> <p>‘4. Nếu người dùng chọn “Clear all” Hệ thống hiển thị pop up xác nhận xóa sản phẩm “Are you sure you want to remove all?”</p> <p>‘5. Người dùng xác nhận việc xóa sản phẩm bằng cách nhấn nút “Yes, delete it”</p> <p>‘6. Hệ thống xóa sản phẩm khỏi giỏ hàng.</p> <p>‘7. Hệ thống cập nhật lại giỏ hàng.</p>
Alternative flows	N/A

Exception flows	5a. Người dùng nhấn nút “ No, cancel ”, pop up sẽ biến mất, người dùng vẫn ở tại giỏ hàng
Business rules	N/A
Non-functional requirements	N/A

Bảng 1

❖ Yêu cầu và điều kiện

No	Item name	Type of item	R/O	Min-max	Note
1	Reduce quantity	Button	O		Khi sản phẩm hiển thị số lượng =1 button Reduce quantity bị disable
2	Increase quantity	Button	O		
3	Quantity	Text box	O		Chỉ chấp nhận ký tự số
4	X	Button	O		Sản phẩm bị xóa khỏi giỏ hàng
5	Clear all	Button	O		Pop Up hiển thị trên màn hình

Bảng 7: Yêu cầu và điều kiện của màn hình Log In

- Điều kiện:

- Nếu người dùng click button Reduce quantity, hệ thống sẽ cập nhật số lượng giảm đi bằng tổng số lượng click tính đến lần click cuối cùng
 - Khi sản phẩm hiển thị số lượng =1 button Reduce quantity bị disable
- Nếu người dùng click button Increase quantity, hệ thống sẽ cập nhật số lượng tăng lên bằng tổng số lượng click tính đến lần click cuối cùng
- Nếu người dùng nhập số lượng vào textbox Quantity, hiển thị thông báo lỗi
 - Không hiển thị ký tự khác số được nhập vào Text box Quantity
 - Không hiển thị ký tự đặc biệt, số âm được nhập vào Text box Quantity
 - Hiện hị thông báo lỗi khi người dùng để trống textbox Quantity: “Invalid quantity”
- Khi người dùng click button “Clear all”, Pop Up xác nhận sẽ hiển thị trên màn hình
 - Nếu người dùng click vào button “Yes, delete it”, toàn bộ sản phẩm sẽ bị xóa khỏi giỏ hàng. Màn hình Giỏ hàng sẽ hiển thị title “Giỏ hàng trống”
 - Nếu người dùng click vào button “No, cancel”, Pop Up biến mất, quay lại màn hình giỏ hàng ban đầu

- Khi số lượng sản phẩm được cập nhật, Tổng tiền trong giỏ hàng sẽ được cập nhật tương ứng. Nếu giỏ hàng không tồn tại sản phẩm nào, Total sẽ hiện giá trị = 0.

CHƯƠNG IV: TỔNG QUAN VỀ SELENIUM VÀ TRIỂN KHAI KIỂM THỬ WEBSITE MOLLEE

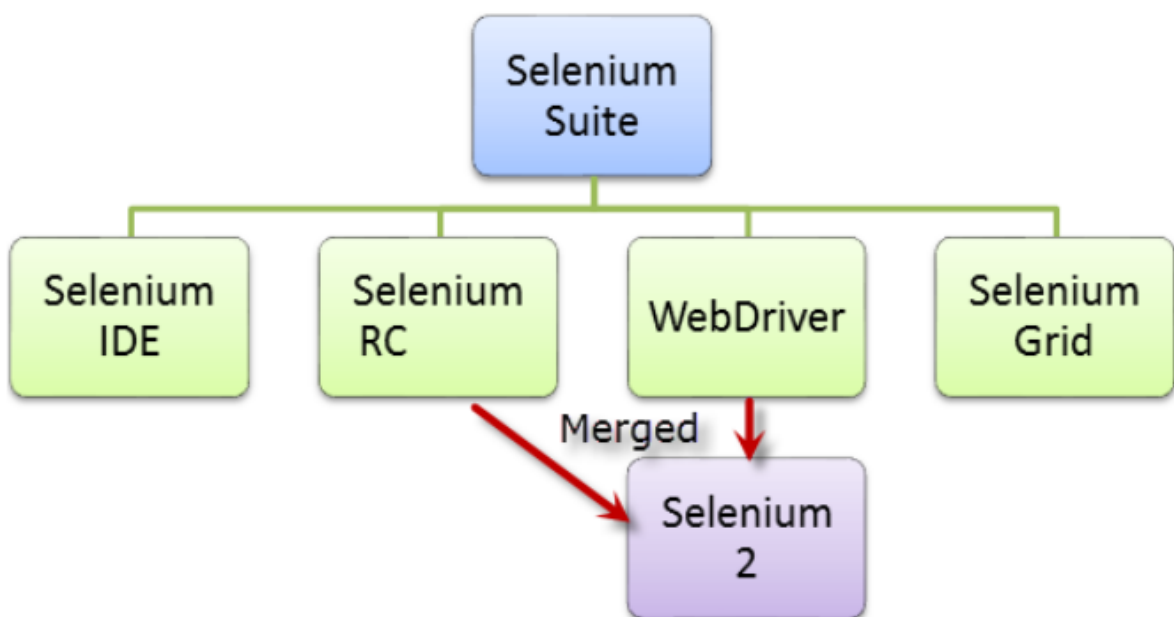
4.1. Tổng quan về tool kiểm thử tự động SELENIUM

4.1.1. Khái niệm

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

Selenium là một trong những công cụ kiểm thử phần mềm tự động mã nguồn mở (open source test automation tool) mạnh mẽ nhất hiện nay cho việc kiểm thử ứng dụng Web. Selenium script có thể chạy được trên hầu hết các trình duyệt như IE, Mozilla FireFox, Chrome, Safari, Opera và hầu hết các hệ điều hành như Windows, Mac, Linux.

4.1.2. Cấu trúc của Selenium



Hình 4. 1: Cấu trúc của Selenium

Cấu trúc của Selenium gồm có 4 phần chính:

- Selenium IDE: là một công cụ cho phép chúng ta Record/Playback một test script
- Selenium RC (Selenium 1 – Selenium Remote Control): là một thư viện cho phép chúng ta lập trình (scripting) test script trên các ngôn ngữ lập trình khác nhau như Python, Java, C#, Ruby.
- Selenium Grid: là một hệ thống hỗ trợ người dùng thực thi test script trên nhiều trình duyệt một cách song song mà không cần phải chỉnh sửa test script.

- Selenium WebDriver (Selenium 2): Tương tự Selenium RC

4.1.2. Đánh giá chung

- Selenium IDE gọn nhẹ và rất đơn giản trong việc cài đặt.
- Selenium IDE và Core đều có giao diện trực quan, và dễ sử dụng
- Selenium thực hiện tốt việc bắt các hành động, tuy nhiên không bắt được các thông báo được đưa dưới dạng alert
- Selenium IDE hiển thị rõ ràng các test đang chạy, chưa lưu , số lượng test bị sai . Chương trình sẽ ngừng và hiển thị bước bị lỗi tại các test không thực hiện được.

4.2. Triển khai kiểm thử Website MOLLEE

4.2.1. Môi trường kiểm thử

- Thiết bị: Laptop HP Pavilion 14
- Browser: Hệ điều hành Window. Trình duyệt web Chrome phiên bản 126.0.6478.127 (Phiên bản chính thức) (64 bit)
- Truy cập vào Google Sheet:
 - Link test case thủ công: [Testcase BuiLeKhanhVy 47K21.2](#)
 - Thực hiện kiểm thử tự động trên Selenium
 - Báo cáo, theo dõi Bug

4.2. Trạng thái của Test case

- Passed: Nhập “Passed” vào status nếu test case đã thực hiện và kết quả thực tế đúng với kết quả mong đợi
- Failed: Nhập Failed vào status nếu test case đã thực hiện và kết quả thực tế không đúng với kết quả mong đợi

4.3. Tiêu chí kiểm thử

Tiêu chí kiểm thử là các tiêu chuẩn và quy định để đánh giá xem một bài kiểm thử đã thành công hay thất bại. Trong phạm vi đề tài này các tiêu chí này bao gồm:

- Tiêu chí chấp nhận:
 - Cho phép người dùng đăng ký tài khoản cá nhân
 - Tài khoản đăng ký phải đăng nhập được vào trang Website.
 - Hệ thống phải thực hiện được các chức năng của giỏ hàng như thêm, sửa, xóa, cập nhật giá tiền
- Tiêu chí chức năng:
 - Tất cả chức năng phải hoạt động đúng theo đặc tả yêu cầu.

Hệ thống phải xử lý chính xác tất cả các thao tác theo yêu cầu của từng lần nhấn click chức năng.

4.3. Kiểm thử thủ công (Manual Test)

4.3.1. Thiết kế Test case

Test case được trình bày trong file Excel, bao gồm Test case của từng màn hình và file Log Bug.

Nội dung của Testcase sẽ tập trung vào kiểm tra giao diện người dùng (UI). Kiểm thử UI cần xem xét cách mà người dùng tương tác với ứng dụng. Đối với kiểm thử giao diện thì lựa chọn phương pháp kiểm thử thủ công sẽ là tối ưu nhất. Vì người kiểm thử thủ công có thể cảm nhận và đánh giá trải nghiệm người dùng theo cách mà tự động hóa khó có thể đạt được. Dễ dàng phát hiện ra các vấn đề về giao diện mà tự động hóa có thể bỏ sót, chẳng hạn như thiết kế không hấp dẫn, bố cục không hợp lý hoặc khó sử dụng.

4.3.2. Cấu trúc Test case

- **Testcase ID:** Đánh dấu vị trí của test case trong file, giúp việc tìm kiếm và theo dõi dễ dàng hơn.
- **Item check:** Cho biết phần tử nào trên màn hình chức năng đang được kiểm thử.
- **Sub-section:** Ghi chú chi tiết phần tử cụ thể được kiểm tra của chức năng cụ thể.
- **Pre_Condition:** Mô tả điều kiện tiên quyết cần thỏa mãn trước khi tiến hành test case.
- **Summary:** Cung cấp mô tả ngắn gọn về mục tiêu của test case, nêu rõ những gì sẽ được xác thực hoặc kiểm thử.
- **Step by perform:** Trình bày các bước cụ thể để thực hiện test case.
- **Input Data:** Dữ liệu cần chuẩn bị để thực hiện test, có thể có hoặc không tùy từng test case
- **Expected result:** Nêu rõ kết quả mong đợi sau khi thực hiện test case
- **Status:** Trạng thái của test case sau khi nó đã được thực hiện
- **Device:** Nền tảng thực hiện kiểm tra.
- **Who check:** Người thực hiện kiểm tra.
- **Date check:** Ngày thực hiện kiểm tra

4.3.3. Thiết kế Test case chức năng Đăng ký (Sign Up)

- Link Test case: [SignUp_Manual](#)
- Kết quả :

Module Name	Total TCs	Passed	Failed
Sign Up Screen	34	27	7

Bảng 8: Bảng kết quả Manual màn hình Sign Up

4.3.4. Thiết kế Test case chức năng Đăng nhập (Log In)

- Link Test case: [Login Manual](#)
- Chi tiết Test case ở phần Phụ lục bên dưới
- Kết quả :

Module Name	Total TCs	Passed	Failed
Sign Up Screen	16	15	1

Bảng 9: Bảng kết quả Manual màn hình LogIn

4.3.5. Thiết kế Test case chức năng Quản lý giỏ hàng (Cart)

- Link Test case: [Cart Manual](#)
- Chi tiết Test case ở phần Phụ lục bên dưới
- Kết quả :

Module Name	Total TCs	Passed	Failed
Sign Up Screen	21	20	1

Bảng 10: Bảng kết quả Manual màn hình Cart

4.4. Kiểm thử tự động (Automation Test)

4.4.1. Môi trường kiểm thử

- Công cụ sử dụng:
 - Selenium WebDriver: Điều khiển trình duyệt tự động.
 - TestNG: Framework tổ chức và chạy các test case.
 - WebDriverManager: Tự động tải và quản lý các driver cho trình duyệt.
- Công cụ hỗ trợ: sử dụng công cụ hỗ trợ Eclipse IDE
- Ngôn ngữ lập trình: Java
- Trình duyệt web: sử dụng trình duyệt Chrome
- Driver của trình duyệt: ChromeDriver

4.4.2. Thiết kế Test script chức năng Đăng ký (Sign Up)

- ❖ TC1: Xác minh rằng người dùng đăng ký không thành công khi để trống trường Username

Đoạn mã:

```
@Test
// Null 1 trường username
public void runTestcase1() {
    try {
        // Tìm phần tử 'username' và làm trống nó
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.clear();
        // Tìm phần tử 'password' và điền mật khẩu vào
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("qMe6Uq");
        sleep(3000);
        // Tìm phần tử 'fullname' và điền họ tên vào
        WebElement name = test.findElement(By.xpath("//input[@name='fullname']"));
        name.sendKeys("Khanh Vy");
        sleep(3000);
        // Tìm phần tử 'email' và điền địa chỉ email vào
        WebElement email = test.findElement(By.xpath("//input[@name='email']"));
        email.sendKeys("buvyka@gmail.com");
        sleep(3000);
        // Tìm và click vào checkbox đồng ý điều khoản
        WebElement privacy =
test.findElement(By.xpath("//label[@class='checkbox__label']"));
        privacy.click();
        sleep(3000);
        // Tìm và click vào nút đăng ký
        WebElement button =
test.findElement(By.xpath("//span[@class='button__text']"));
        button.click();
        sleep(4000);
        // Tìm phần tử hiển thị thông báo lỗi cho trường 'username'
        WebElement errorMessage = test.findElement(By.id("username"));
        String actualMessage = errorMessage.getText();
        // So sánh thông báo lỗi thực tế với thông báo mong muốn
        Assert.assertEquals(actualMessage, "Please fill in this field.");

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
    }
}
```

- ❖ TC2: Xác minh rằng người dùng đăng ký thành công khi nhập 1 ký tự vào trường Username

Đoạn mã:

```
@Test
// Username: 1 ký tự
public void runTestcase2() {
    try {
        // Tìm phần tử 'username' và nhập 1 ký tự
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.sendKeys("K");
        // Tìm phần tử 'password' và điền mật khẩu vào
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("abc123");
        sleep(3000);
        // Tìm phần tử 'fullname' và điền họ tên vào
        WebElement name = test.findElement(By.xpath("//input[@name='fullname']"));
        name.sendKeys("Khanh Vy");
        sleep(3000);
        // Tìm phần tử 'email' và điền địa chỉ email vào
        WebElement email = test.findElement(By.xpath("//input[@name='email']"));
        email.sendKeys("buvyka@gmail.com");
        sleep(3000);
        // Tìm và click vào checkbox đồng ý điều khoản
        WebElement privacy =
test.findElement(By.xpath("//label[@class='checkbox__label']"));
        privacy.click();
        sleep(3000);
        // Tìm và click vào nút đăng ký
        WebElement button = test.findElement(By.xpath("//span[@class='button__text']"));
        button.click();
        sleep(4000);
        // Kiểm tra xem có chuyển đến màn hình login hay không
        boolean isOnAddLoginScreen;
        try {
            WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(5));
            WebElement Login =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//h1[@class='inner-top__title']")));
            isOnAddLoginScreen = Login.isDisplayed();
        } catch (NoSuchElementException | TimeoutException e) {
            isOnAddLoginScreen = true;
        }

        // Xác nhận người dùng không chuyển đến màn hình login
        try {
            Assert.assertTrue(isOnAddLoginScreen, "User is not on the Login Screen");
        }
    }
}
```

```

    } catch (AssertionError e) {
        throw new RuntimeException("User is not on the Login Screen", e);
    }

    System.out.println("Testcase pass");
} catch (Exception e) {
    System.out.println("Testcase fail: " + e.getMessage());
    e.printStackTrace();
}
}

```

❖ TC3: Xác minh rằng người dùng đăng ký thành công khi nhập 60 ký tự vào trường Username

Đoạn mã:

```

@Test
// Username: 60 ký tự
public void runTestcase3() {
    try {
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.sendKeys(
            "NhaLanhDaoCongNgheDuaTheGioiDenMotTuongLaiTuoiDepVaPhatTrien");
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("abc123");
        sleep(3000);
        WebElement name = test.findElement(By.xpath("//input[@name='fullname']"));
        name.sendKeys("Khanh Vy");
        sleep(3000);
        WebElement email = test.findElement(By.xpath("//input[@name='email']"));
        email.sendKeys("buvyka@gmail.com");
        sleep(3000);
        WebElement privacy =
            test.findElement(By.xpath("//label[@class='checkbox__label']"));
        privacy.click();
        sleep(3000);
        WebElement button = test.findElement(By.xpath("//span[@class='button__text']"));
        button.click();
        sleep(4000);
        boolean isOnAddLoginScreen;
        try {
            WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(5));
            WebElement Login =
                wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//h1[@class='inner-top__title']")));
            isOnAddLoginScreen = Login.isDisplayed();
        }
    }
}

```

```

    } catch (NoSuchElementException | TimeoutException e) {
        isOnAddLoginScreen = true;
    }

    try {
        Assert.assertTrue(isOnAddLoginScreen, "User is not on the Login Screen");
    } catch (AssertionError e) {
        throw new RuntimeException("User is not on the Login Screen", e);
    }

    System.out.println("Testcase pass");
} catch (Exception e) {
    System.out.println("Testcase fail: " + e.getMessage());
    e.printStackTrace();
}
}

```

❖ TC4: Xác minh rằng người dùng đăng ký không thành công khi nhập 61 ký tự vào trường Username

Đoạn mã:

```

@Test
// Username: 61 ký tự
public void runTestcase4() {
    try {
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.sendKeys(
            "NhaLanhDaoCongNgheDuaTheGioiDenMotTuongLaiTuiDepVaPhatTien12");
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("qMe6Uq");
        sleep(3000);
        WebElement name = test.findElement(By.xpath("//input[@name='fullname']"));
        name.sendKeys("Khanh Vy");
        sleep(3000);
        WebElement email = test.findElement(By.xpath("//input[@name='email']"));
        email.sendKeys("buvyka@gmail.com");
        sleep(3000);
        WebElement privacy =
            test.findElement(By.xpath("//label[@class='checkbox__label']"));
        privacy.click();
        sleep(3000);
        WebElement button = test.findElement(By.xpath("//span[@class='button__text']"));
        button.click();
        sleep(4000);
        WebElement errorMessage = test.findElement(By.id("username"));
    }
}

```



```

String actualMessage = errorMessage.getText();
Assert.assertEquals(actualMessage, "Username không hợp lệ!");

System.out.println("Testcase pass");
} catch (Exception e) {
    System.out.println("Testcase fail: " + e.getMessage());
    e.printStackTrace();
}
}

```

- ❖ TC5: Xác minh rằng người dùng đăng ký không thành công để trống trường dữ liệu Password.

Đoạn mã:

```

@Test
// Null 1 trường password
public void runTestcase5() {
    try {
        WebElement user = test.findElement(By.xpath("//input[ @name='username']"));
        user.sendKeys("Khanh Vy");
        WebElement pass = test.findElement(By.xpath("//input[ @name='password']"));
        pass.clear();
        sleep(3000);
        WebElement name = test.findElement(By.xpath("//input[ @name='fullname']"));
        name.sendKeys("Khanh Vy");
        sleep(3000);
        WebElement email = test.findElement(By.xpath("//input[ @name='email']"));
        email.sendKeys("buvyka@gmail.com");
        sleep(3000);
        WebElement privacy =
test.findElement(By.xpath("//label[ @class='checkbox__label']"));
        privacy.click();
        sleep(3000);
        WebElement button =
test.findElement(By.xpath("//span[ @class='button__text']"));
        button.click();
        sleep(4000);
        WebElement errorMessage = test.findElement(By.id("Password"));
        String actualMessage = errorMessage.getText();
        Assert.assertEquals(actualMessage, "Please fill in this field.");

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
    }
}

```

```
}  
}
```

- ❖ TC6: Xác minh rằng người dùng đăng ký không thành công khi nhập ký tự khoảng trống vào trường Password.

Đoạn mã:

```
@Test  
// Password: ký tự khoảng trống  
public void runTestcase6() {  
    try {  
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));  
        user.sendKeys("KhanhVy");  
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));  
        pass.sendKeys(  
            " ");  
        sleep(3000);  
        WebElement name = test.findElement(By.xpath("//input[@name='fullname']"));  
        name.sendKeys("Khanh Vy");  
        sleep(3000);  
        WebElement email = test.findElement(By.xpath("//input[@name='email']"));  
        email.sendKeys("buvyka@gmail.com");  
        sleep(3000);  
        WebElement privacy =  
test.findElement(By.xpath("//label[@class='checkbox__label']"));  
        privacy.click();  
        sleep(3000);  
        WebElement button =  
test.findElement(By.xpath("//span[@class='button__text']"));  
        button.click();  
        sleep(4000);  
        WebElement testlogin =  
test.findElement(By.xpath("//html/body/div[2]/main/div[1]/div/div/div[1]/h1"));  
        if ("Login".equals(testlogin.getText())){  
            System.out.println("Testcase fail");  
            Assert.fail("Testcase fail");  
        }  
        else {  
            System.out.println("Testcase pass");  
        }  
    } catch (Exception e) {  
        System.out.println("Testcase fail: " + e.getMessage());  
        e.printStackTrace();  
    }  
}
```

- ❖ TC7: Xác minh rằng người dùng đăng ký không thành công khi nhập 5 ký tự vào trường Password.

Đoạn mã:

```
@Test
// Password: 5 ký tự
public void runTestcase7() {
    try {
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.sendKeys("KhanhVy");
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys(
            "abc12");
        sleep(3000);
        WebElement name = test.findElement(By.xpath("//input[@name='fullname']"));
        name.sendKeys("Khanh Vy");
        sleep(3000);
        WebElement email = test.findElement(By.xpath("//input[@name='email']"));
        email.sendKeys("buvyka@gmail.com");
        sleep(3000);
        WebElement privacy =
test.findElement(By.xpath("//label[@class='checkbox__label']"));
        privacy.click();
        sleep(3000);
        WebElement button =
test.findElement(By.xpath("//span[@class='button__text']"));
        button.click();
        sleep(4000);
        //html/body/div[2]/main/div[1]/div/div/div[1]/h1
        WebElement testlogin =
test.findElement(By.xpath("//html/body/div[2]/main/div[1]/div/div/div[1]/h1"));
        if ("Login".equals(testlogin.getText())){
            System.out.println("Testcase fail");
            Assert.fail("Testcase fail");
        }
        else {
            System.out.println("Testcase pass");}

    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
    }
}
```

- ❖ TC8: Xác minh rằng người dùng đăng ký thành công khi nhập 6 ký tự vào trường Password.

Đoạn mã:

```
@Test
// Passwor: 6 ký tự
public void runTestcase8() {
    try {
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.sendKeys("KhanhVy");
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("abc123");
        sleep(3000);
        WebElement name = test.findElement(By.xpath("//input[@name='fullname']"));
        name.sendKeys("Khanh Vy");
        sleep(3000);
        WebElement email = test.findElement(By.xpath("//input[@name='email']"));
        email.sendKeys("buvyka@gmail.com");
        sleep(3000);
        WebElement privacy =
test.findElement(By.xpath("//label[@class='checkbox__label']"));
        privacy.click();
        sleep(3000);
        WebElement button = test.findElement(By.xpath("//span[@class='button__text']"));
        button.click();
        sleep(4000);
        boolean isOnAddLoginScreen;
        try {
            WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(5));
            WebElement Login =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//h1[@class='inner
-top__title']")));
            isOnAddLoginScreen = Login.isDisplayed();
        } catch (NoSuchElementException | TimeoutException e) {
            isOnAddLoginScreen = true;
        }

        try {
            Assert.assertTrue(isOnAddLoginScreen, "User is not on the Login Screen");
        } catch (AssertionError e) {
            throw new RuntimeException("User is not on the Login Screen", e);
        }

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
    }
}
```

- ❖ TC9: Xác minh rằng người dùng đăng ký thành công khi nhập 60 ký tự vào trường Password.

Đoạn mã:

```
@Test
// Password: 60 ký tự
public void runTestcase9() {
    try {
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.sendKeys("KhanhVy");
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys(
            "NhaLanhDaoCongNgheDuaTheGioiDenMotTuongLaiTuoiDepVaPhatTrien");
        sleep(3000);
        WebElement name = test.findElement(By.xpath("//input[@name='fullname']"));
        name.sendKeys("Khanh Vy");
        sleep(3000);
        WebElement email = test.findElement(By.xpath("//input[@name='email']"));
        email.sendKeys("buvyka@gmail.com");
        sleep(3000);
        WebElement privacy =
            test.findElement(By.xpath("//label[@class='checkbox__label']"));
        privacy.click();
        sleep(3000);
        WebElement button = test.findElement(By.xpath("//span[@class='button__text']"));
        button.click();
        sleep(4000);
        boolean isOnAddLoginScreen;
        try {
            WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(5));
            WebElement Login =
                wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//h1[@class='inner-top__title']")));
            isOnAddLoginScreen = Login.isDisplayed();
        } catch (NoSuchElementException | TimeoutException e) {
            isOnAddLoginScreen = true;
        }

        try {
            Assert.assertTrue(isOnAddLoginScreen, "User is not on the Login Screen");
        } catch (AssertionError e) {
            throw new RuntimeException("User is not on the Login Screen", e);
        }

        System.out.println("Testcase pass");
    } catch (Exception e) {
```

```

        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
    }
}

```

- ❖ TC10: Xác minh rằng người dùng đăng ký không thành công khi nhập 61 ký tự vào trường Password.

Đoạn mã:

```

@Test
// Password 61 ký tự
public void runTestcase10() {
    try {
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.sendKeys(
            "NhaLanhDaoCongNgheDuaTheGioiDenMotTuongLaiTuiDepVaPhatTien12");
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("qMe6Uq");
        sleep(3000);
        WebElement name = test.findElement(By.xpath("//input[@name='fullname']"));
        name.sendKeys("Khanh Vy");
        sleep(3000);
        WebElement email = test.findElement(By.xpath("//input[@name='email']"));
        email.sendKeys("buvyka@gmail.com");
        sleep(3000);
        WebElement privacy =
            test.findElement(By.xpath("//label[@class='checkbox__label']"));
        privacy.click();
        sleep(3000);
        WebElement button = test.findElement(By.xpath("//span[@class='button__text']"));
        button.click();
        sleep(4000);
        WebElement errorMessage = test.findElement(By.id("username"));
        String actualMessage = errorMessage.getText();
        Assert.assertEquals(actualMessage, "Username không hợp lệ!");

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
    }
}

```

- ❖ TC11: Xác minh rằng người dùng đăng ký không thành công khi để trống trường dữ liệu Name.

Đoạn mã:

```
@Test
// Null 1 trường name
public void runTestcase11() {
    try {
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.sendKeys("KhanhVy");
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("qMe6Uq");
        sleep(3000);
        WebElement name = test.findElement(By.xpath("//input[@name='fullname']"));
        name.clear();
        sleep(3000);
        WebElement email = test.findElement(By.xpath("//input[@name='email']"));
        email.sendKeys("buvyka@gmail.com");
        sleep(3000);
        WebElement privacy =
test.findElement(By.xpath("//label[@class='checkbox__label']"));
        privacy.click();
        sleep(3000);
        WebElement button =
test.findElement(By.xpath("//span[@class='button__text']"));
        button.click();
        sleep(4000);
        WebElement errorMessage = test.findElement(By.id("Fullname"));
        String actualMessage = errorMessage.getText();
        Assert.assertEquals(actualMessage, "Please fill in this field.");

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
    }
}
```

- ❖ TC12: Xác minh rằng người dùng đăng ký không thành công khi để trống trường dữ liệu Email.

Đoạn mã:

```
@Test
// Null 1 trường email
public void runTestcase12() {
    try {
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.sendKeys("KhanhVy");
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("qMe6Uq");
    }
}
```

```

        pass.sendKeys("qMe6Uq");
        sleep(3000);
        WebElement name = test.findElement(By.xpath("//input[@name='fullname']"));
        name.sendKeys("Bùi Lê Khánh Vy");
        sleep(3000);
        WebElement email = test.findElement(By.xpath("//input[@name='email']"));
        email.clear();
        sleep(3000);
        WebElement privacy =
test.findElement(By.xpath("//label[@class='checkbox__label']"));
        privacy.click();
        sleep(3000);
        WebElement button =
test.findElement(By.xpath("//span[@class='button__text']"));
        button.click();
        sleep(4000);
        WebElement errorMessage = test.findElement(By.id("Email"));
        String actualMessage = errorMessage.getText();
        Assert.assertEquals(actualMessage, "Please fill in this field.");

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
    }
}

```

- ❖ TC13 Xác minh rằng người dùng đăng ký không thành công khi email được nhập thiếu ký tự “@”

Đoạn mã:

```

@Test
// Email thiếu ký tự '@'
public void runTestcase13() {
    try {
        // Tìm phần tử 'username' và nhập tên người dùng
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.sendKeys("KhanhVy");

        // Tìm phần tử 'password' và nhập mật khẩu
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("abc123");

        // Tìm phần tử 'fullname' và nhập họ tên
        WebElement name = test.findElement(By.xpath("//input[@name='fullname']"));
        name.sendKeys("Khanh Vy");
    }
}

```



```

// Tìm phần tử 'email' và nhập email thiếu ký tự '@'
WebElement email = test.findElement(By.xpath("//input[@name='email']"));
email.sendKeys("buvykagmail.com");

// Tìm và click vào checkbox đồng ý điều khoản
WebElement privacy =
test.findElement(By.xpath("//label[@class='checkbox__label']"));
privacy.click();

// Tìm và click vào nút đăng ký
WebElement button =
test.findElement(By.xpath("//span[@class='button__text']"));
button.click();

// Kiểm tra xem không chuyển đến màn hình đăng nhập
boolean isOnAddLoginScreen;
try {
    WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(5));
    WebElement Login =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/
main/section[1]/div[1]/h3")));
    isOnAddLoginScreen = Login.isDisplayed();
} catch (NoSuchElementException | TimeoutException e) {
    isOnAddLoginScreen = true;
}

try {
    Assert.assertTrue(isOnAddLoginScreen, "User is not on the Login Screen");
} catch (AssertionError e) {
    throw new RuntimeException("User is not on the Login Screen", e);
}

System.out.println("Testcase pass");
} catch (Exception e) {
    System.out.println("Testcase fail: " + e.getMessage());
    e.printStackTrace();
}
}

```

❖ TC14: Xác minh rằng người dùng đăng ký không thành công khi email được nhập thiếu prefix

Đoạn mã:

```

@Test
// Email thiếu prefix

```

```

public void runTestcase14() {
    try {
        // Tìm phần tử 'username' và nhập tên người dùng
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.sendKeys("KhanhVy");

        // Tìm phần tử 'password' và nhập mật khẩu
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("abc123");

        // Tìm phần tử 'fullname' và nhập họ tên
        WebElement name = test.findElement(By.xpath("//input[@name='fullname']"));
        name.sendKeys("Khanh Vy");

        // Tìm phần tử 'email' và nhập email thiếu ký tự '@'
        WebElement email = test.findElement(By.xpath("//input[@name='email']"));
        email.sendKeys("@gmail.com");

        // Tìm và click vào checkbox đồng ý điều khoản
        WebElement privacy =
test.findElement(By.xpath("//label[@class='checkbox__label']"));
        privacy.click();

        // Tìm và click vào nút đăng ký
        WebElement button =
test.findElement(By.xpath("//span[@class='button__text']"));
        button.click();

        // Kiểm tra xem không chuyển đến màn hình đăng nhập
        boolean isOnAddLoginScreen;
        try {
            WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(5));
            WebElement Login =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/
main/section[1]/div[1]/h3")));
            isOnAddLoginScreen = Login.isDisplayed();
        } catch (NoSuchElementException | TimeoutException e) {
            isOnAddLoginScreen = true;
        }

        try {
            Assert.assertTrue(isOnAddLoginScreen, "User is not on the Login Screen");
        } catch (AssertionError e) {
            throw new RuntimeException("User is not on the Login Screen", e);
        }

        System.out.println("Testcase pass");
    }
}

```

```

    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
    }
}

```

- ❖ TC15: Xác minh rằng người dùng đăng ký không thành công khi email được nhập thiếu domain name

Đoạn mã:

```

@Test
// Email thiếu domain name
public void runTestcase15() {
    try {
        // Tìm phần tử 'username' và nhập tên người dùng
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.sendKeys("KhanhVy");

        // Tìm phần tử 'password' và nhập mật khẩu
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("abc123");

        // Tìm phần tử 'fullname' và nhập họ tên
        WebElement name = test.findElement(By.xpath("//input[@name='fullname']"));
        name.sendKeys("Khanh Vy");

        // Tìm phần tử 'email' và nhập email thiếu ký tự '@'
        WebElement email = test.findElement(By.xpath("//input[@name='email']"));
        email.sendKeys("buvyka@");

        // Tìm và click vào checkbox đồng ý điều khoản
        WebElement privacy =
test.findElement(By.xpath("//label[@class='checkbox__label']"));
        privacy.click();

        // Tìm và click vào nút đăng ký
        WebElement button =
test.findElement(By.xpath("//span[@class='button__text']"));
        button.click();

        // Kiểm tra xem không chuyển đến màn hình đăng nhập
        boolean isOnAddLoginScreen;
        try {
            WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(5));
            WebElement Login =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/

```

```

main/section[1]/div[1]/h3"))));
    isOnAddLoginScreen = Login.isDisplayed();
} catch (NoSuchElementException | TimeoutException e) {
    isOnAddLoginScreen = true;
}

try {
    Assert.assertTrue(isOnAddLoginScreen, "User is not on the Login Screen");
} catch (AssertionError e) {
    throw new RuntimeException("User is not on the Login Screen", e);
}

System.out.println("Testcase pass");
} catch (Exception e) {
    System.out.println("Testcase fail: " + e.getMessage());
    e.printStackTrace();
}
}
}

```

- ❖ TC16: Xác minh rằng người dùng đăng ký không thành công khi email được nhập chứa khoảng trống

Đoạn mã:

```

@Test
// Email chứa khoảng trống
public void runTestcase16() {
    try {
        // Tìm phần tử 'username' và nhập tên người dùng
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.sendKeys("KhanhVy");

        // Tìm phần tử 'password' và nhập mật khẩu
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("abc123");

        // Tìm phần tử 'fullname' và nhập họ tên
        WebElement name = test.findElement(By.xpath("//input[@name='fullname']"));
        name.sendKeys("Khanh Vy");

        // Tìm phần tử 'email' và nhập email thiếu ký tự '@'
        WebElement email = test.findElement(By.xpath("//input[@name='email']"));
        email.sendKeys("buvy ka @gmail.com");

        // Tìm và click vào checkbox đồng ý điều khoản
        WebElement privacy =
test.findElement(By.xpath("//label[@class='checkbox__label']"));
    }
}

```

```

        privacy.click();

        // Tìm và click vào nút đăng ký
        WebElement button =
test.findElement(By.xpath("//span[@class='button__text']"));
        button.click();

        // Kiểm tra xem không chuyển đến màn hình đăng nhập
        boolean isOnAddLoginScreen;
        try {
            WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(5));
            WebElement Login =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/
main/section[1]/div[1]/h3")));
            isOnAddLoginScreen = Login.isDisplayed();
        } catch (NoSuchElementException | TimeoutException e) {
            isOnAddLoginScreen = true;
        }

        try {
            Assert.assertTrue(isOnAddLoginScreen, "User is not on the Login Screen");
        } catch (AssertionError e) {
            throw new RuntimeException("User is not on the Login Screen", e);
        }

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
    }
}

```

- ❖ TC17: Xác minh rằng màn hình chuyển hướng đến màn hình Login khi người dùng click vào hyperlink Login

Đoạn mã:

```

@Test
// Click hyperlink Login
public void runTestcase17() {
    try {
        WebElement Loginlink =
test.findElement(By.xpath("/html/body/div[2]/main/div[2]/div/div[2]/div/form/div/div[2]
/span/a"));
        ((JavascriptExecutor) test).executeScript("arguments[0].scrollIntoView(true);",
Loginlink);
        sleep(5000);
    }
}

```

```

        ((JavascriptExecutor) test).executeScript("arguments[0].click();", Loginlink);
        sleep(7000);
        boolean isOnAddLoginScreen;
        try {
            WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(5));
            WebElement Login =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/
main/div[2]/div/div[2]/form/div/div[1]/h2")));
            isOnAddLoginScreen = Login.isDisplayed();
        } catch (NoSuchElementException | TimeoutException e) {
            isOnAddLoginScreen = true;
        }

        try {
            Assert.assertTrue(isOnAddLoginScreen, "User is not on the Login Screen");
        } catch (AssertionError e) {
            throw new RuntimeException("User is not on the Login Screen", e);
        }

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
    }
}

```

❖ TC18: Xác minh rằng người dùng đăng ký thành công trong trường hợp click vào button Privacy

Đoạn mã:

```

@Test
// Đăng ký thành công ( click button privacy)
public void runTestcase18() {
    try {
        WebElement user = test.findElement(By.xpath("//input[ @name='username']"));
        user.sendKeys("KhanhVy");
        WebElement pass = test.findElement(By.xpath("//input[ @name='password']"));
        pass.sendKeys("qMe6Uq");
        sleep(3000);
        WebElement name = test.findElement(By.xpath("//input[ @name='fullname']"));
        name.sendKeys("Bùi Lê Khánh Vy");
        sleep(3000);
        WebElement email = test.findElement(By.xpath("//input[ @name='email']"));
        email.sendKeys("Buivyka@gmail.com");
        sleep(3000);
        WebElement privacy =

```

```

test.findElement(By.xpath("//label[@class='checkbox__label']"));
    privacy.click();
    sleep(3000);
    WebElement button =
test.findElement(By.xpath("//span[@class='button__text']"));
    button.click();
    sleep(4000);
    boolean isOnAddLoginScreen;
    try {
        WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(5));
        WebElement Login =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/
main/div[2]/div/div[2]/form/div/div[1]/h2")));
        isOnAddLoginScreen = Login.isDisplayed();
    } catch (NoSuchElementException | TimeoutException e) {
        isOnAddLoginScreen = true;
    }

    try {
        Assert.assertTrue(isOnAddLoginScreen, "User is not on the Login Screen");
    } catch (AssertionError e) {
        throw new RuntimeException("User is not on the Login Screen", e);
    }

    System.out.println("Testcase pass");
} catch (Exception e) {
    System.out.println("Testcase fail: " + e.getMessage());
    e.printStackTrace();
}
}

```

- ❖ TC19: Xác minh rằng người dùng có thể đăng nhập thành công với tài khoản vừa được tạo.

Đoạn mã:

```

@Test
// Test đăng nhập thành công tài khoản vừa tạo
public void runTestcase19() {
    try {
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.sendKeys("KhanhVy");
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("qMe6Uq");
        sleep(3000);
        WebElement name = test.findElement(By.xpath("//input[@name='fullname']"));
        name.sendKeys("Bùi Lê Khánh Vy");
    }
}

```

```

        sleep(3000);
        WebElement email = test.findElement(By.xpath("//input[@name='email']"));
        email.sendKeys("Buivvyka@gmail.com");
        sleep(3000);
        WebElement privacy =
test.findElement(By.xpath("//label[@class='checkbox__label']"));
        privacy.click();
        sleep(3000);
        WebElement button =
test.findElement(By.xpath("//span[@class='button__text']"));
        button.click();
        sleep(4000);
        WebElement username =
test.findElement(By.xpath("//input[@name='username']"));
        username.sendKeys("KhanhVy");
        sleep(3000);
        WebElement password =
test.findElement(By.xpath("//input[@name='password']"));
        password.sendKeys("qMe6Uq");
        sleep(3000);
        WebElement Loginbutton =
test.findElement(By.xpath("/html/body/div[2]/main/div[2]/div/div[2]/form/div/button/span"));
        Loginbutton.click();
        sleep(5000);
        boolean isOnAddLoginScreen;
        try {
            WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(5));
            WebElement Login =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/main/section[1]/div[1]/h3")));
            isOnAddLoginScreen = Login.isDisplayed();
        } catch (NoSuchElementException | TimeoutException e) {
            isOnAddLoginScreen = true;
        }

        try {
            Assert.assertTrue(isOnAddLoginScreen, "User is not on the Home Screen");
        } catch (AssertionError e) {
            throw new RuntimeException("User is not on the Home Screen", e);
        }

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
    }

```



```
}  
}
```

- ❖ TC20: Xác minh rằng người dùng đăng ký không thành công trong trường hợp không click vào button Privacy

```
5. @Test  
// Đăng ký thành công ( click button privacy)  
public void runTestcase20() {  
    try {  
        WebElement user =  
test.findElement(By.xpath("//input[@name='username']"));  
        user.sendKeys("KhanhVy");  
        WebElement pass =  
test.findElement(By.xpath("//input[@name='password']"));  
        pass.sendKeys("qMe6Uq");  
        sleep(3000);  
        WebElement name =  
test.findElement(By.xpath("//input[@name='fullname']"));  
        name.sendKeys("Bùi Lê Khánh Vy");  
        sleep(3000);  
        WebElement email =  
test.findElement(By.xpath("//input[@name='email']"));  
        email.sendKeys("Buivyka@gmail.com");  
        sleep(3000);  
        WebElement button =  
test.findElement(By.xpath("//span[@class='button__text']"));  
        button.click();  
        sleep(4000);  
        String currentUrl = test.getCurrentUrl();  
        String loginPageUrl = "http://localhost:8080/login"; // Thay đổi URL theo  
        trang Login của bạn  
  
        if (currentUrl.equals(loginPageUrl)) {  
            // Nếu URL hiện tại là màn hình Login, đánh dấu kiểm thử thất bại  
            Assert.fail("Màn hình đã chuyển hướng đến trang đăng nhập khi không  
chọn hộp kiểm privacy.");  
        }  
  
        System.out.println("Testcase pass");  
    } catch (Exception e) {  
        System.out.println("Testcase fail: " + e.getMessage());  
        e.printStackTrace();  
    }  
}
```

4.4.3. Thiết kế Test script chức năng Đăng nhập (Login)

- ❖ TC1: Xác minh rằng người dùng đăng nhập không thành công khi để trống trường Username

Đoạn mã:

```
@Test
// Null username
public void runTestcase1() {
    try {
        // Tìm phần tử 'username' và xóa nội dung
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.clear();
        // Tìm phần tử 'password' và nhập mật khẩu
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("qMe6Uq");
        // Chờ 3 giây
        sleep(3000);
        // Tìm và click vào nút đăng ký
        WebElement button =
test.findElement(By.xpath("//span[@class='button__text']"));
        button.click();
        // Chờ 5 giây
        sleep(5000);
        // Tìm phần tử hiển thị thông báo lỗi cho trường 'username'
        WebElement errorMessage = this.test.findElement(By.id("Username"));
        String actualMessage = errorMessage.getText();
        // Kiểm tra thông báo lỗi
        Assert.assertEquals(actualMessage, "Please fill in this field.");
        System.out.println("Testcase pass");
    } catch (Exception e) {
        // In ra thông báo lỗi nếu có ngoại lệ
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
    }
}
```

- ❖ TC2: Xác minh rằng người dùng đăng nhập không thành công khi để trống trường Password

Đoạn mã:

```
@Test
// Null password
public void runTestcase2() {
    try {
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("qMe6Uq");
        // Tìm và click vào nút đăng ký
        WebElement button =
test.findElement(By.xpath("//span[@class='button__text']"));
        button.click();
        // Chờ 5 giây
        sleep(5000);
        // Tìm phần tử hiển thị thông báo lỗi cho trường 'password'
        WebElement errorMessage = this.test.findElement(By.id("Password"));
        String actualMessage = errorMessage.getText();
        // Kiểm tra thông báo lỗi
        Assert.assertEquals(actualMessage, "Please fill in this field.");
        System.out.println("Testcase pass");
    } catch (Exception e) {
        // In ra thông báo lỗi nếu có ngoại lệ
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
    }
}
```

```

        user.sendKeys("vyahin6");
        WebElement pass = test.findElement(By.xpath("//input[ @name='password']"));
        pass.clear();
        sleep(3000);
        WebElement button =
test.findElement(By.xpath("//span[ @class='button__text']"));
        button.click();
        sleep(5000);
        WebElement errorMessage = this.test.findElement(By.id("Password"));
        String actualMessage = errorMessage.getText();
        // Kiểm tra thông báo lỗi
        Assert.assertEquals(actualMessage, "Please fill in this field.");
        System.out.println("Testcase pass");
    } catch (Exception e) {
        // In ra thông báo lỗi nếu có ngoại lệ
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
    }
}

```

- ❖ TC3: Xác minh rằng nút button Đăng nhập không hoạt động khi không có trường nào được nhập dữ liệu

Đoạn mã:

```

@Test
// Null all
public void runTestcase3() {
    try {
        WebElement user = test.findElement(By.xpath("//input[ @name='username']"));
        user.clear();
        WebElement pass = test.findElement(By.xpath("//input[ @name='password']"));
        pass.clear();
        sleep(3000);
        WebElement button =
test.findElement(By.xpath("//span[ @class='button__text']"));
        button.click();
        sleep(5000);
        WebElement errorMessage = this.test.findElement(By.id("Password"));
        String actualMessage = errorMessage.getText();
        // Kiểm tra thông báo lỗi
        Assert.assertEquals(actualMessage, "Please fill in this field.");
        System.out.println("Testcase pass");
    } catch (Exception e) {
        // In ra thông báo lỗi nếu có ngoại lệ
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
    }
}

```

```
}  
}
```

- ❖ TC4: Xác minh rằng hệ thống đăng nhập không thành công khi nhập 1 Username không tồn tại

Đoạn mã:

```
@Test  
// User không tồn tại  
public void runTestcase4() {  
    try {  
        WebElement user = test.findElement(By.xpath("//input[ @name='username']"));  
        user.sendKeys("vyahin");  
        sleep(3000);  
        WebElement pass = test.findElement(By.xpath("//input[ @name='password']"));  
        pass.sendKeys("qMe6Uq");  
        sleep(3000);  
        WebElement button =  
test.findElement(By.xpath("//span[ @class='button__text']"));  
        button.click();  
        sleep(3000);  
        WebDriverWait wait = new WebDriverWait(this.test, Duration.ofSeconds(10));  
        WebElement errorMessage =  
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("username-error")));  
        String actualMessage = errorMessage.getText();  
        Assert.assertEquals(actualMessage, "Username này không tồn tại!");  
        System.out.println("Testcase pass");  
    } catch (Exception e) {  
        System.out.println("Testcase fail: " + e.getMessage());  
        e.printStackTrace();  
    }  
}
```

- ❖ TC5: Xác minh rằng hệ thống đăng nhập thành công trong trường hợp không click button Privacy policy

Đoạn mã:

```
@Test  
// Login thành công, không click button Privacy policy  
public void runTestcase5() {  
    try {  
        WebElement user = test.findElement(By.xpath("//input[ @name='username']"));  
        user.sendKeys("vyahin6");  
        sleep(3000);
```

```

        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("qMe6Uq");
        sleep(3000);
        WebElement button =
test.findElement(By.xpath("//span[@class='button__text']"));
        button.click();
        sleep(5000);
        boolean isOnAddLoginScreen;
        try {
            WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(5));
            WebElement Login =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/
main/section[1]/div[1]/h3")));
            isOnAddLoginScreen = Login.isDisplayed();
        } catch (NoSuchElementException | TimeoutException e) {
            isOnAddLoginScreen = true;
        }

        try {
            Assert.assertTrue(isOnAddLoginScreen, "User is not on the Home Screen");
        } catch (AssertionError e) {
            throw new RuntimeException("User is not on the Home Screen", e);
        }

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
    }
}

```

- ❖ TC6: Xác minh rằng màn hình chuyển hướng đến màn hình Fogot Password khi người dùng click vào hyperlink Fogot Password

Đoạn mã:

```

@Test
// click hyperlink Fogot Password
public void runTestcase6() {
    try {
        WebElement user =
test.findElement(By.xpath("/html/body/div[2]/main/div[2]/div/div[2]/form/div/div[2]/div
[2]/span/a"));
        user.click();
        sleep(5000);
        boolean isOnAddLoginScreen;
        try {

```

```

        WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(5));
        WebElement Login =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/
main/div[2]/div/div[2]/form/div/button/span")));
        isOnAddLoginScreen = Login.isDisplayed();
    } catch (NoSuchElementException | TimeoutException e) {
        isOnAddLoginScreen = true;
    }

    try {
        Assert.assertTrue(isOnAddLoginScreen, "User is not on the Fogot Password
Screen");
    } catch (AssertionError e) {
        throw new RuntimeException("User is not on the Fogot Password Screen", e);
    }

    System.out.println("Testcase pass");
} catch (Exception e) {
    System.out.println("Testcase fail: " + e.getMessage());
    e.printStackTrace();
}
}

```

- ❖ TC7: Xác minh rằng màn hình chuyển hướng đến màn hình Register khi người dùng click vào hyperlink Register

Đoạn mã:

```

@Test
// click hyperlink Register
public void runTestcase7() {
    try {
        WebElement user =
test.findElement(By.xpath("/html/body/div[2]/main/div[2]/div/div[2]/form/div/div[3]/a"));
        ;
        user.click();
        sleep(5000);
        boolean isOnAddLoginScreen;
        try {
            WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(5));
            WebElement Login =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/
main/div[2]/div/div[2]/div/form/button/span")));
            isOnAddLoginScreen = Login.isDisplayed();
        } catch (NoSuchElementException | TimeoutException e) {
            isOnAddLoginScreen = true;
        }
    }
}

```

```

    try {
        Assert.assertTrue(isOnAddLoginScreen, "User is not on the Register Screen");
    } catch (AssertionError e) {
        throw new RuntimeException("User is not on the Register Screen", e);
    }

    System.out.println("Testcase pass");
} catch (Exception e) {
    System.out.println("Testcase fail: " + e.getMessage());
    e.printStackTrace();
}
}

```

- ❖ TC8: Xác minh rằng hệ thống đăng nhập thành công trong trường hợp click button Privacy policy.

Đoạn mã:

```

@Test
// Login thành công, có click button Privacy policy
public void runTestcase8() {
    try {
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.sendKeys("vyahin6");
        sleep(3000);
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("qMe6Uq");
        sleep(3000);
        WebElement remember =
test.findElement(By.xpath("/html/body/div[2]/main/div[2]/div/div[2]/form/div/div[2]/div
[1]/div/label/span[2]"));
        remember.click();
        sleep(3000);
        WebElement button =
test.findElement(By.xpath("/html/body/div[2]/main/div[2]/div/div[2]/form/div/button/spa
n"));
        button.click();
        sleep(5000);

        boolean isOnAddLoginScreen;
        try {
            WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(5));
            WebElement Login =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/
main/section[1]/div[1]/h3")));
            isOnAddLoginScreen = Login.isDisplayed();

```

```

    } catch (NoSuchElementException | TimeoutException e) {
        isOnAddLoginScreen = true;
    }

    try {
        Assert.assertTrue(isOnAddLoginScreen, "User is not on the Home Screen");
    } catch (AssertionError e) {
        throw new RuntimeException("User is not on the Home Screen", e);
    }

    System.out.println("Testcase pass");
} catch (Exception e) {
    System.out.println("Testcase fail: " + e.getMessage());
    e.printStackTrace();
}
}
}

```

- ❖ TC9: Xác minh rằng hệ thống hiển thị gợi ý tài khoản cho lần đăng nhập tiếp theo sau khi người dùng click vào button Remember me.

Đoạn mã:

```

@Test
// Kiểm tra chức năng ghi nhớ mật khẩu
public void runTestcase9() {
    try {
        // Nhập tên người dùng
        WebElement user = test.findElement(By.xpath("//input[@name='username']"));
        user.sendKeys("vyahin6");
        sleep(3000); // Chờ 3 giây để đảm bảo tên người dùng đã được nhập xong
        // Nhập mật khẩu
        WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
        pass.sendKeys("qMe6Uq");
        sleep(3000); // Chờ 3 giây để đảm bảo mật khẩu đã được nhập xong
        // Chọn checkbox "Ghi nhớ mật khẩu"
        WebElement remember =
test.findElement(By.xpath("/html/body/div[2]/main/div[2]/div/div[2]/form/div/div[2]/div
[1]/div/label/span[1]"));
        remember.click();
        sleep(3000); // Chờ 3 giây để đảm bảo checkbox được chọn
        // Nhấn nút Đăng nhập
        WebElement button =
test.findElement(By.xpath("//button[@type='submit']/span"));
        button.click();
        sleep(5000); // Chờ 5 giây để quá trình đăng nhập hoàn tất
        // Click vào tài khoản của tôi
        WebElement me =

```



```

test.findElement(By.xpath("/html/body/div[2]/header/div/div/div[2]/ul/li[3]/a/span"));
me.click();
sleep(3000); // Chờ 3 giây để trang thông tin cá nhân được mở
// Đăng xuất
WebElement logout =
test.findElement(By.xpath("/html/body/div[2]/header/div/div/div[2]/ul/li[3]/div/ul/li[3]/a"
));
logout.click();
sleep(5000); // Chờ 5 giây để quá trình đăng xuất hoàn tất
// Kiểm tra xem thông tin đăng nhập có được lưu lại hay không
WebElement usernameField = test.findElement(By.id("Username"));
WebElement passwordField = test.findElement(By.id("Password"));
try {
    // Kiểm tra xem trường tên người dùng và mật khẩu có được tự động điền
không
    Assert.assertEquals(usernameField.getAttribute("value"), "vyahin6", "Tên
người dùng không được tự động điền");
    Assert.assertEquals(passwordField.getAttribute("value"), "qMe6Uq", "Mật
khẩu không được tự động điền");
    System.out.println("Testcase pass");
} catch (AssertionError e) {
    System.out.println("Testcase fail: " + e.getMessage());
    Assert.fail("Testcase fail: " + e.getMessage());
}

} catch (Exception e) {
    System.out.println("Testcase fail: " + e.getMessage());
    Assert.fail("Testcase fail: " + e.getMessage());
}
}

```

4.4.4. Thiết kế Test script chức năng Quản lý giỏ hàng (Cart)

- ❖ TC1: Xác minh rằng người dùng có thể click vào sản phẩm và xem thông tin chi tiết của sản phẩm đó.

Đoạn mã:

```

@Test
// Kiểm tra người dùng có thể click vào xem sản phẩm
public void runTestcase1() {
    try {
        // Tìm phân tử sản phẩm và chờ cho đến khi có thể click
        WebElement SP =
waitCustom.findElementToBeClickableByXPath("//*[ @id=\"products-
1\"]/div/div/article[6]/div/h4/a");
        // Cuộn trang để phân tử sản phẩm có thể nhìn thấy

```

```

        ((JavascriptExecutor) test).executeScript("arguments[0].scrollIntoView(true);",
SP);
        sleep(5000);
        // Click vào phần tử sản phẩm
        ((JavascriptExecutor) test).executeScript("arguments[0].click();", SP);
        sleep(5000);
        // Cuộn trang để nhìn chi tiết sản phẩm
        WebElement XemSP =
waitCustom.findElementToBeClickableByXPath("/html/body/div[2]/main/div[1]/div/div
/div[1]/div/ul/li[2]");
        ((JavascriptExecutor) test).executeScript("arguments[0].scrollIntoView(true);",
XemSP);
        sleep(5000);
        // Kiểm tra nút "Add to cart" có xuất hiện sau khi click vào sản phẩm
        String buttonText =
waitCustom.findElementToBeClickableByXPath("/html/body/div[2]/main/section/div/di
v[2]/div[2]/div/a/span").getText();
        Assert.assertEquals(buttonText, "Add to cart");

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
        Assert.assertEquals(true, false);
    }
}

```

- ❖ TC2: Xác minh rằng người dùng có thể click vào sản phẩm và thêm sản phẩm đó vào giỏ hàng

Đoạn mã:

```

@Test()
// Click vào sản phẩm và thêm sản phẩm vào giỏ hàng
public void runTestcase2() {
    try {
        // Tìm phần tử sản phẩm và chờ cho đến khi có thể click
        WebElement SP =
waitCustom.findElementToBeClickableByXPath("//*[ @id=\"products-
1\"]/div/div/article[6]/div/h4/a");
        // Cuộn trang để phần tử sản phẩm có thể nhìn thấy
        ((JavascriptExecutor) test).executeScript("arguments[0].scrollIntoView(true);",
SP);
        sleep(5000);
        // Click vào phần tử sản phẩm
        ((JavascriptExecutor) test).executeScript("arguments[0].click();", SP);
        sleep(5000);
    }
}

```

```

        // Kiểm tra nút "Add to cart" có xuất hiện sau khi click vào sản phẩm
        String Test =
waitCustom.findElementToBeClickableByXPath("/html/body/div[2]/main/section/div/div[2]/div[2]/div/a/span").getText();
        Assert.assertEquals(Test, "Add to cart");
        // Tìm phần tử kích thước và cuộn trang để phần tử này có thể nhìn thấy
        WebElement Size =
waitCustom.findElementToBeClickableByXPath("/html/body/div[2]/main/section/div/div[2]/div[2]/div/a/span");
        ((JavascriptExecutor) test).executeScript("arguments[0].scrollIntoView(true);",
Size);
        sleep(5000);
        // Click vào phần tử kích thước
        ((JavascriptExecutor) test).executeScript("arguments[0].click();", Size);
        // Cuộn trang để nhìn chi tiết giỏ hàng
        WebElement XemSP =
waitCustom.findElementToBeClickableByXPath("/html/body/div[2]/main/div[1]/div/div/div[1]/div/ul/li[2]");
        ((JavascriptExecutor) test).executeScript("arguments[0].scrollIntoView(true);",
XemSP);
        sleep(5000);
        // Kiểm tra nút "Checkout" có xuất hiện sau khi thêm sản phẩm vào giỏ hàng
        String Test1 =
waitCustom.findElementToBeClickableByXPath("/html/body/div[2]/main/div[2]/div[1]/aside/section/a/span").getText();
        Assert.assertEquals(Test1, "Checkout");

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
        Assert.assertEquals(true, false);
    }
}
}

```

❖ TC3: Xác minh rằng số lượng sản phẩm và Total được cập nhật khi click vào button [+]

Đoạn mã:

```

@Test
// Click button [+] và kiểm tra cập nhật cột Total
public void runTestcase3() {
    try {
        addProductToCart();

        // Bấm vào nút dấu cộng để tăng số lượng sản phẩm
        WebElement plusButton = waitCustom.findElementToBeClickableByXPath(
            "/html/body/div[2]/main/div[2]/div[1]/div/div[1]/form/article/div/div[2]/div[2]/div[2]/div
            /button[2]");
        plusButton.click();
        sleep(3000);
        String oldTotalValue = "73.32 $";
        WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(10));

        wait.until(ExpectedConditions.not(ExpectedConditions.textToBePresentInElementLocate
            d(By.xpath(
                "/html/body/div[2]/main/div[2]/div[1]/aside/section/div/div[2]/span"),
            oldTotalValue)
        ));
        String expectedNewTotalValue = "146.64 $";

        // Kiểm tra giá trị mới của cột Total
        WebElement totalElement =
        wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/
            main/div[2]/div[1]/aside/section/div/div[2]/span")));
        String totalValue = totalElement.getText();
        Assert.assertEquals(totalValue, expectedNewTotalValue);

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
        Assert.assertEquals(true, false);
    }
}

```

- ❖ TC3: Xác minh rằng số lượng sản phẩm và Total được cập nhật khi click vào button [+]

Đoạn mã:

```
@Test
// Click button [+] và kiểm tra cập nhật cột Total
public void runTestcase3() {
    try {
        addProductToCart();

        // Bấm vào nút dấu cộng để tăng số lượng sản phẩm
        WebElement plusButton = waitCustom.findElementToBeClickableByXPath(
            "/html/body/div[2]/main/div[2]/div[1]/div/div[1]/form/article/div/div[2]/div[2]/div[2]/div/button[2]");
        plusButton.click();
        sleep(3000);
        String oldTotalValue = "73.32 $";
        WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(10));

        wait.until(ExpectedConditions.not(ExpectedConditions.textToBePresentInElementLocated(By.xpath(
            "/html/body/div[2]/main/div[2]/div[1]/aside/section/div/div[2]/span"),
            oldTotalValue)
        ));
        String expectedNewTotalValue = "146.64 $";

        // Kiểm tra giá trị mới của cột Total
        WebElement totalElement =
            wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/main/div[2]/div[1]/aside/section/div/div[2]/span")));
        String totalValue = totalElement.getText();
        Assert.assertEquals(totalValue, expectedNewTotalValue);

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
        Assert.assertEquals(true, false);
    }
}
```

- ❖ TC4: Xác minh button [-] không hoạt động khi số lượng sản phẩm hiển thị =1

Đoạn mã:

```

@Test
// Kiểm tra xác minh rằng nút giảm không hoạt động khi số lượng sản phẩm bằng 1
public void runTestcase4() {
    try {
        addProductToCart();
        // Kiểm tra giá trị hiện tại của ô nhập số lượng
        WebElement quantityInput =
waitCustom.findElementToBeClickableByXPath("//input[@value='1']");
        String quantityBeforeClick = quantityInput.getAttribute("value");
        Assert.assertEquals(quantityBeforeClick, "1");

        // Bấm vào nút giảm
        WebElement minusButton =
waitCustom.findElementToBeClickableByXPath("/html/body/div[2]/main/div[2]/div[1]/div/div[1]/form/article/div/div[2]/div[2]/div[2]/div/button[1]");
        minusButton.click();
        Thread.sleep(5000);

        // Kiểm tra lại giá trị của ô nhập số lượng
        String quantityAfterClick = quantityInput.getAttribute("value");
        Assert.assertEquals(quantityAfterClick, "1");

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
        Assert.assertEquals(true, false);
    }
}

```

- ❖ TC5: Xác minh rằng người dùng không thể input dữ liệu khác số vào Quantity Text box
Đoạn mã:

```

@Test
// Kiểm tra người dùng không thể nhập ký tự khác số
public void runTestcase5() {
    try {
        addProductToCart();

        // Thử nhập số <1
        appendToProductQuantity("abc");
        WebElement quantityInput =
waitCustom.findElementToBeClickableByXPath("//input[@value='1']");
        String quantity = quantityInput.getAttribute("value");
        Assert.assertEquals(quantity, "1");
    }
}

```

```

appendToProductQuantity("-");
quantity = quantityInput.getAttribute("value");
Assert.assertEquals(quantity, "1");

System.out.println("Testcase pass");
} catch (Exception e) {
    System.out.println("Testcase fail: " + e.getMessage());
    e.printStackTrace();
    Assert.assertEquals(true, false);
}
}

```

- ❖ TC6: Xác minh rằng hệ thống cập nhật số lượng và Total khi người dùng input dữ liệu vào Quantity Text box
Đoạn mã:

```

@Test
// Xóa số lượng hiện tại và điền số lượng mới
public void runTestcase6() {
    try {
        addProductToCart();
        updateProductQuantity("5");
        String oldTotalValue = "73.32 $";
        WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(10));
        wait.until(ExpectedConditions.not(
ExpectedConditions.textToBePresentInElementLocated(By.xpath("/html/body/div[2]/main/div[2]/div[1]/aside/section/div/div[2]/span"), oldTotalValue)
));

        String expectedNewTotalValue = "366.6 $";

        // Kiểm tra giá trị mới của cột Total
        WebElement totalElement =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/main/div[2]/div[1]/aside/section/div/div[2]/span")));
        String totalValue = totalElement.getText();
        Assert.assertEquals(totalValue, expectedNewTotalValue);

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
        Assert.assertEquals(true, false);
    }
}

```

- ❖ TC7: Xác minh rằng số lượng sản phẩm và Total được cập nhật khi click vào button [-]

Đoạn mã:

```
@Test
// Click button [-] và kiểm tra cập nhật cột Total
public void runTestcase7() {
    try {
        addProductToCart();
        updateProductQuantity("5");
        // Bấm vào nút dấu cộng để tăng số lượng sản phẩm
        WebElement giam =
waitCustom.findElementToBeClickableByXPath("/html/body/div[2]/main/div[2]/div[1]/div/div[1]/form/article/div/div[2]/div[2]/div[2]/div/button[1]");
        giam.click();
        sleep(3000);
        String oldTotalValue = "73.32 $";
        WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(15));

wait.until(ExpectedConditions.not(ExpectedConditions.textToBePresentInElementLocated(By.xpath(
        "/html/body/div[2]/main/div[2]/div[1]/aside/section/div/div[2]/span"),
oldTotalValue)
        ));
        String expectedNewTotalValue = "293.28 $";

        // Kiểm tra giá trị mới của cột Total
        WebElement totalElement =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/main/div[2]/div[1]/aside/section/div/div[2]/span")));
        String totalValue = totalElement.getText();
        Assert.assertEquals(totalValue, expectedNewTotalValue);

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
        Assert.assertEquals(true, false);
    }
}
```

- ❖ TC8: Xác minh rằng Dialog xác nhận xuất hiện khi người dùng click vào button “Clear all”

Đoạn mã:

```
@Test
// Xác nhận dialog xuất hiện
```



```

public void runTestcase8() {
    try {
        addProductToCart();
        WebElement dialog =
test.findElement(By.xpath("/html/body/div[2]/main/div[2]/div[1]/div/div[1]/div/button"));
;
        dialog.click();
        sleep(3000);
        WebElement dialogText =
dialog.findElement(By.xpath("//*[@id=\"deleteLabel\"]")); // Thay đổi XPath để khớp
với cấu trúc thực tế của hộp thoại
        String actualText = dialogText.getText();
        sleep(5000);
        String expectedText = "Are you sure?"; // Thay đổi văn bản dự kiến theo yêu cầu
kiểm thử
        Assert.assertEquals(actualText, expectedText, "Văn bản trong hộp thoại không
khớp");

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
        Assert.assertEquals(true, false);
    }
}

```

❖ TC9: Xác minh rằng Dialog xác nhận biến mất khi người dùng click vào button [X]

Đoạn mã:

```

@Test
// Click button X
public void runTestcase9() {
    try {
        addProductToCart();
        WebElement X =
test.findElement(By.xpath("/html/body/div[2]/main/div[2]/div[1]/div/div[1]/form/article/
div/div[2]/div[1]/div[2]/a"));
        X.click();
        sleep(7000);
        boolean isOnAddLoginScreen;
        try {
            WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(5));
            WebElement Login =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/
main/div[2]/div[1]/div/div[1]/div[1]/div/h2")));
            isOnAddLoginScreen = Login.isDisplayed();

```

```

    } catch (NoSuchElementException | TimeoutException e) {
        isOnAddLoginScreen = true;
    }

    try {
        Assert.assertTrue(isOnAddLoginScreen, "Cart is empty");
    } catch (AssertionError e) {
        throw new RuntimeException("Cart is not empty", e);
    }

    System.out.println("Testcase pass");
} catch (Exception e) {
    System.out.println("Testcase fail: " + e.getMessage());
    e.printStackTrace();
    Assert.assertEquals(true, false);
}
}

```

- ❖ TC10: Xác minh rằng Dialog xác nhận biến mất khi người dùng click vào button “No, cancel”

Đoạn mã:

```

@Test
// Chọn button "No, cancel"
public void runTestcase10() {
    try {
        addProductToCart();
        WebElement dialog =
test.findElement(By.xpath("/html/body/div[2]/main/div[2]/div[1]/div/div[1]/div/button"));
        ;
        dialog.click();
        sleep(Assert.assertEquals(actualText, expectedText, "Sản phẩm không tồn tại");

        System.out.println("Testcase pass");
    } catch (Exception e) {
        System.out.println("Testcase fail: " + e.getMessage());
        e.printStackTrace();
        Assert.assertEquals(true, false);
    }
}3000);
    WebElement no =
test.findElement(By.xpath("//*[ @id=\"delete\"]/div/div/div[3]/a[2]"));
    no.click();
    sleep(5000);
    WebElement Text =
dialog.findElement(By.xpath("/html/body/div[2]/main/div[2]/div[1]/div/div[1]/form/article/div/div[2]/div[1]/div[1]/h2/a")); // Thay đổi XPath để khớp với cấu trúc thực tế của

```

hộp thoại

```
String actualText = Text.getText();  
sleep(3000);  
String expectedText = "Mens Seamless Trunks"; // Thay đổi văn bản dự kiến theo  
yêu cầu kiểm thử
```

- ❖ TC11: Xác minh rằng Dialog xác nhận biến mất khi người dùng click vào button “Yes, Delete it”

Đoạn mã:

```
@Test  
// Chọn button "yes, delete it"  
public void runTestcase11() {  
    try {  
        addProductToCart();  
        WebElement dialog =  
test.findElement(By.xpath("/html/body/div[2]/main/div[2]/div[1]/div/div[1]/div/button"))  
        ;  
        dialog.click();  
        sleep(3000);  
        WebElement yes =  
test.findElement(By.xpath("//*[ @id=\"delete\"] /div/div/div[3]/a[1]"));  
        yes.click();  
        sleep(5000);  
        boolean isOnAddLoginScreen;  
        try {  
            WebDriverWait wait = new WebDriverWait(test, Duration.ofSeconds(5));  
            WebElement Login =  
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/  
main/div[2]/div[1]/div/div[1]/div[1]/div/h2")));  
            isOnAddLoginScreen = Login.isDisplayed();  
        } catch (NoSuchElementException | TimeoutException e) {  
            isOnAddLoginScreen = true;  
        }  
  
        try {  
            Assert.assertTrue(isOnAddLoginScreen, "Cart is empty");  
        } catch (AssertionError e) {  
            throw new RuntimeException("Cart is not empty", e);  
        }  
    } catch (Exception e) {  
        System.out.println("Testcase fail: " + e.getMessage());  
        e.printStackTrace();  
        Assert.assertEquals(true, false);  
    }  
}
```

```
}
}
```

4.4.5. Kết quả kiểm thử

❖ Chức năng Đăng ký (Sign Up)

- Kết quả :

▼	✖ Default Suite	5 min 31 sec
▼	✖ Testcase_BuiLeKhanhVy	5 min 31 sec
▼	✖ Register	5 min 31 sec
	✓ runTestcase1	16 sec 469 ms
	✓ runTestcase10	16 sec 470 ms
	✓ runTestcase11	16 sec 371 ms
	✓ runTestcase12	16 sec 419 ms
	✓ runTestcase13	5 sec 541 ms
	✓ runTestcase14	5 sec 534 ms
	✓ runTestcase15	5 sec 577 ms
	✓ runTestcase16	5 sec 521 ms
	✓ runTestcase17	12 sec 215 ms
	✓ runTestcase18	16 sec 496 ms
	✓ runTestcase19	27 sec 742 ms
	✓ runTestcase2	17 sec 35 ms
	✖ runTestcase20	13 sec 473 ms
	✓ runTestcase3	21 sec 709 ms
	✓ runTestcase4	16 sec 483 ms
	✓ runTestcase5	16 sec 411 ms
	✖ runTestcase6	16 sec 497 ms
	✖ runTestcase7	16 sec 503 ms
	✓ runTestcase8	16 sec 537 ms
	✓ runTestcase9	21 sec 680 ms

Hình 4. 2: Kết quả Automation test màn hình Sign Up

Module Name	Total TCs	Passed	Failed
Sign Up Screen	20	17	3

Bảng 11: Bảng kết quả Automatuion màn hình Sign Up

❖ Chức năng Đăng nhập (Login)

- Kết quả:

Default Suite	Sorting Options	57 sec
Testcase_BuiLeKhanhVy		1 min 57 sec
Login		1 min 57 sec
runTestcase1		8 sec 275 ms
runTestcase2		8 sec 207 ms
runTestcase3		8 sec 182 ms
runTestcase4		19 sec 723 ms
runTestcase5		11 sec 304 ms
runTestcase6		5 sec 199 ms
runTestcase7		5 sec 239 ms
runTestcase8		14 sec 354 ms
runTestcase9		22 sec 540 ms

Hình 4. 3: Kết quả Automation test màn hình Login

Module Name	Total TCs	Passed	Failed
Sign Up Screen	9	8	1

Bảng 12: Bảng kết quả Automatuion màn hình Login

❖ Chức năng Quản lý giỏ hàng (Cart)

- Kết quả:

Default Suite	3 min 33 sec
Testcase_BuiLeKhanhVy	3 min 33 sec
Cart	3 min 33 sec
runTestcase1	15 sec 397 ms
runTestcase10	11 sec 615 ms
runTestcase11	8 sec 636 ms
runTestcase2	20 sec 502 ms
runTestcase3	13 sec 497 ms
runTestcase4	5 sec 470 ms
runTestcase5	10 sec 479 ms
runTestcase6	5 sec 508 ms
runTestcase7	8 sec 585 ms
runTestcase8	8 sec 444 ms
runTestcase9	7 sec 507 ms

Hình 4. 4: Kết quả Automation test màn hình Cart

Module Name	Total TCs	Passed	Failed
Sign Up Screen	11	10	2

4.5. Log Bug

4.5.1. Cấu trúc Report Bug

- No: Sẽ đánh số thứ tự để phân biệt. Số thứ tự log bug này sẽ tương ứng với việc đánh số thứ tự ở cột No.bug trong file test case (cho biết log bug cho test case nào)
- Item: Item cụ thể xuất hiện bug
- Title: Cho biết tiêu đề của bug mình log là gì
- Description: Mô tả chi tiết bug đó
- Step to perform: Mô tả các bước để tái hiện bug đó
- Actual result: Kết quả thực tế trả về
- Expected result: Kết quả trả về mong đợi

4.5.1. Log Bug Test case Manual

Link file log bug cho 3 chức năng của Test case Manual: [Log Bug Manual](#)

4.5.2. Log Bug Test case Automation

Link file log bug cho 3 chức năng của Test case Automation: [Log Bug Automation](#)

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

❖ Đạt được:

Sau khoảng thời gian thực tập 3 tháng tại môi trường chuyên nghiệp của công ty FPT Software, đơn vị IVS.DN đã giúp em có cơ hội được học hỏi và tiếp thu nhiều kinh nghiệm quý báu trong nghề nghiệp:

- Học và nắm rõ kiến thức lý thuyết chuyên ngành một cách đầy đủ và chính xác
- Được thực hành viết Testcase, LogBug
- Học và thực hành test Automation bằng ngôn ngữ Java
- Được trải nghiệm chạy dự án thực tế, nâng cao kinh nghiệm

❖ Hạn chế:

- Ngoại ngữ chưa tốt trở thành trở ngại lớn trong việc tiếp thu kiến thức
- Còn nhiều khuyết điểm trong kỹ năng mềm như giao tiếp, đặt câu hỏi, thắc mắc cho mentor
- Chưa tối ưu hóa được code Automation Testing

❖ Hướng phát triển

- Nâng cao khả năng ngôn ngữ
- Tiếp tục học và nâng cao kiến thức về Kiểm thử
- Trau dồi kiến thức và tư duy kiểm thử tự động
- Tiếp tục cố gắng làm việc tại công ty FPT Software với vai trò là nhân viên OJT. Nỗ lực học hỏi nâng cao kỹ năng và tích lũy thêm nhiều kinh nghiệm cho bản thân.

TÀI LIỆU THAM KHẢO

1. [Mục tiêu của kiểm thử phần mềm và các mục đích Tester cần hướng đến](#)
2. [7 Nguyên tắc cơ bản của Kiểm Thử Phần Mềm](#)
3. [Tìm hiểu về các cấp độ kiểm thử - test levels](#)
4. [Sự khác nhau giữa Manual Testing và Automation Testing](#)
5. [TEST TOOL: SELENIUM IDE](#)
6. [Tester làm những công việc gì? Tiêu chí tuyển dụng Tester](#)
7. [Giới thiệu FPT Software Đà Nẵng](#)

CHECKLIST CỦA BÁO CÁO

STT	Nội dung công việc	Có	Không	Ghi chú
1	Báo cáo được trình bày (định dạng) đúng với yêu cầu.	X		
2	Báo cáo có số lượng trang đáp ứng đúng yêu cầu (40-60 trang)	X		
3	Báo cáo trình bày được phần mở đầu bao gồm: Mục tiêu, Phạm vi và đối tượng, kết cấu ...	X		
4	Báo cáo trình bày về công ty, vị trí việc làm (công việc đó làm gì, kiến thức và kỹ năng cần thiết là gì, con đường phát triển sự nghiệp (career path)), cơ sở lý thuyết phù hợp với nội dung của đề tài (Tối đa 10-12 trang)	X		
5	Báo cáo có sản phẩm cụ thể phù hợp với mục tiêu đặt ra của đề tài	X		
6	Báo cáo có phần kết luận và hướng phát triển của đề tài	X		

PHỤ LỤC

- Khởi tạo thư viện cho Test Automation chức năng Sign Up

```
import WebDriverWaitCustom.WebDriverWaitCustom;
import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Assert;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;
import java.time.Duration;

public class Register {
    WebDriver test;
    WebDriverWait waitCustom;

    @BeforeMethod
    public void setup() throws InterruptedException {
        WebDriverManager.chromedriver().setup();
        ChromeOptions options = new ChromeOptions();
        options.addArguments("--start-maximized");
        test = new ChromeDriver(options);
        test.get("http://localhost:8080/register?lang=en");

        waitCustom = new WebDriverWait(test, Duration.ofSeconds(10));
    }
}
```

- Khởi tạo thư viện cho Test Automation chức năng Log In

```
import WebDriverWaitCustom.WebDriverWaitCustom;
import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Assert;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;
```

```

import java.time.Duration;

public class Login {
    WebDriver test;
    WebDriverWaitCustom waitCustom;

    @BeforeMethod
    public void setup() throws InterruptedException {
        WebDriverManager.chromedriver().setup();
        ChromeOptions options = new ChromeOptions();
        options.addArguments("--start-maximized");
        test = new ChromeDriver(options);
        test.get("http://localhost:8080/login?lang=en");

        waitCustom = new WebDriverWaitCustom(test, Duration.ofSeconds(10));
    }
}

```

- Khởi tạo thư viện cho Test Automation chức năng Quản lý giỏ hàng (Cart)

```

import WebDriverWaitCustom.WebDriverWaitCustom;
import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Assert;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

import java.time.Duration;

public class Cart {
    WebDriver test;
    WebDriverWaitCustom waitCustom;
    private WebDriverWaitCustom wait;

    @BeforeMethod
    public void setup() throws InterruptedException {
        WebDriverManager.chromedriver().setup();
        ChromeOptions options = new ChromeOptions();
        options.addArguments("--start-maximized");
        test = new ChromeDriver(options);
        test.get("http://localhost:8080/login?lang=en");
    }
}

```

```

WebElement user = test.findElement(By.xpath("//input[@name='username']"));
user.sendKeys("vyahin6");
WebElement pass = test.findElement(By.xpath("//input[@name='password']"));
pass.sendKeys("qMe6Uq");
WebElement button = test.findElement(By.xpath("//span[@class='button__text']"));
button.click();
Thread.sleep(4000);

waitCustom = new WebDriverWaitCustom(test, Duration.ofSeconds(10));
}

```

```

public void addProductToCart() throws InterruptedException {
    WebElement SP =
waitCustom.findElementToBeClickableByXPath("//*[@id=\"products-1\"]/div/div/article[6]/div/h4/a");
    ((JavascriptExecutor) test).executeScript("arguments[0].scrollIntoView(true);", SP);
    ((JavascriptExecutor) test).executeScript("arguments[0].click();", SP);
    WebElement addToCartButton =
waitCustom.findElementToBeClickableByXPath("/html/body/div[2]/main/section/div/div[2]/div[2]/div/a/span");
    ((JavascriptExecutor) test).executeScript("arguments[0].scrollIntoView(true);",
addToCartButton);
    ((JavascriptExecutor) test).executeScript("arguments[0].click();", addToCartButton);
}

```

- Template cấu trúc file Testcase

TEST CASE											
TC-ID	Item check	Sub-section	Pre_Condition	Summary	Step by perform	Input Data	Expected result	Status	Device	who check	Date Check

- Template cấu trúc file Logbug

STT	Item	Title bug	Description	Step to perform	Data check	Expected result	Actual result	Evidence
-----	------	-----------	-------------	-----------------	------------	-----------------	---------------	----------

- Status kết quả

Status
Pass ▼
Fail ▼