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

# ỨNG DỤNG SELENIUM TRONG KIỂM THỬ TỰ ĐỘNG WEBSITE HRICK


Sinh viên thực hiện : **Bùi Thị Việt**

Lớp : **47K21.2**

Đơn vị thực tập : **IVS.DN - FPT Software**

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

# LỜI CẢM ƠN

Để hoàn thành đề tài "Ứng Dụng Selenium Trong Kiểm Thử Tự Động Website HRICK", em đã nhận được sự giúp đỡ và hỗ trợ nhiệt tình từ nhiều cá nhân và tổ chức. Chính vì thế em xin gửi lời cảm ơn chân thành đến tất cả những người đã đóng góp và ủng hộ em trong suốt quá trình thực hiện đề tài này.

Trước hết, em xin bày tỏ lòng biết ơn đến các thầy cô trong Khoa Thống kê - Tin học, Trường Đại học Kinh tế - Đại học Đà Nẵng, đã truyền đạt những kiến thức quý báu và tạo điều kiện thuận lợi cho em trong suốt quá trình học tập và thực hiện đề tài này. Đặc biệt, em xin gửi lời cảm ơn sâu sắc đến thầy Nguyễn Văn Chức, người đã tận tình hướng dẫn, hỗ trợ trợ em để hoàn thiện đề tài này trong thời gian 3 tháng qua.

Mình cũng xin chân thành cảm ơn đến các bạn bè, đồng nghiệp động viên và chia sẻ những khó khăn trong suốt quá trình thực hiện đề tài. Sự ủng hộ và khích lệ của mọi người là nguồn động lực to lớn giúp em vượt qua mọi thử thách và hoàn thành đề tài này như ngày hôm nay.

Cuối cùng, em xin gửi lời cảm ơn đến IVS - FPT Software đã hỗ trợ và tạo mọi điều kiện thuận lợi để em có thể hoàn thành tốt đề tài này và cũng giúp đỡ em rất nhiều trong thời gian em thực tập tại đây.

Trân trọng cảm ơn!

Việt

Bùi Thị Việt

# LỜI CAM ĐOAN

Em xin cam đoan rằng đề tài "Ứng Dụng Selenium Trong Kiểm Thử Tự Động Website HRICK" là kết quả nghiên cứu của riêng em, dưới sự hướng dẫn tận tình của thầy/cô [Tên Giáo Viên Hướng Dẫn]. Các số liệu và kết quả nghiên cứu trong báo cáo này đều trung thực và chưa từng được sử dụng để bảo vệ một học vị nào trước đây.

Trong quá trình thực hiện đề tài, nếu có sử dụng tài liệu, số liệu hoặc kết quả nghiên cứu của các tác giả khác, em đều đã trích dẫn và ghi rõ nguồn gốc theo đúng quy định. Em hoàn toàn chịu trách nhiệm về tính trung thực và chính xác của nội dung báo cáo này.

Em xin chân thành cảm ơn.

Trân trọng cảm ơn!

Việt

Bùi Thị Việt

# MỤC LỤC

# MỤC TIÊU CỦA ĐỀ TÀI

- Nghiên cứu về các kỹ năng, kiến thức cần có của tester
- Hiểu được tầm quan trọng của testing, mức lương, cơ hội nghề nghiệp
- Ứng dụng selenium với java để thực hiện tự động hóa quy trình kiểm thử

***Đối tượng và phạm vi nghiên cứu***

- Website: HRick
- Chức năng: Tìm kiếm người dùng trong màn hình Chat

***Kết cấu của đề tài***

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

- Mở đầu: Trình bày lý do chọn đề tài, mục tiêu nghiên cứu đồ án và bố cục đề án
- **Chương 1**: Tổng quan về FPT và lý thuyết về vị trí tester
- Giới thiệu tổng quan về IVS- FPT soft
- Các kỹ năng cần có để trở thành tester
- Cơ hội nghề nghiệp và mức lương cơ bản
- **Chương 2**: Cơ sở lý thuyết
- Tổng quan về kiểm thử phần mềm
- Quy trình phát triển phần mềm
- Giới thiệu về Mô hình Agile, Waterfall

**Chương 3**: Giới thiệu về kiểm thử tự động

- Sơ lược về kiểm thử tự động
- Tổng quan về java và selenium
- Cách cài đặt và thiết lập môi trường java
- **Chương 4**: Thực hành kiểm thử tự động trên chức năng search trong màn hình Chat với selenium
- Phân tích yêu cầu

- Test case, test script
- Chạy chương trình

***Kết luận và hướng phát triển***

Phần này đưa ra những kết luận kết quả đồ án làm được, những thiếu sót chưa thực hiện và hướng phát triển trong tương lai

# DANH MỤC HÌNH ẢNH



# DANH MỤC BẢNG BIỂU

# DANH MỤC TỪ VIẾT TẮT

| STT | KÝ HIỆU | CỤM TỪ ĐẦY ĐỦ | Ý NGHĨA |
|---|---|---|---|
| 1 | API | Application Programming Interface | Giao diện lập trình ứng dụng |
| | BA | Business Analysis | Người phân tích nghiệp vụ |
| | CSS | Cascading Style Sheets | Ngôn ngữ quy định cách hiển thị của các phần tử HTML |
| | DOM | Document Object Model | Mô hình đối tượng tài liệu |
| | ERP | Enterprise Resource Planning | Hệ thống hoạch định tài liệu doanh nghiệp |
| | HTML | HyperText Markup Language | Ngôn ngữ đánh dấu siêu văn bản |
| | PM | Project Manager | Người quản trị dự án |
| | ISTQB | International Software Testing Qualifications Board | Tổ chức cung cấp chứng chỉ kiểm thử phần mềm có giá trị toàn cầu |
| | IVS | Independent verification services | Công ty con của FPT software và cũng là công ty lớn nhất tại Việt Nam cung cấp dịch vụ xác thực độc lập. |
| | UI | User Interface | Giao diện người dùng |

# MỞ ĐẦU

Trong thời đại công nghệ 4.0, kiểm thử phần mềm (software testing) đóng một vai trò vô cùng quan trọng trong quá trình phát triển và duy trì chất lượng của các ứng dụng phần mềm. Kiểm thử phần mềm không chỉ giúp phát hiện ra các lỗi (bugs) mà còn đảm bảo rằng phần mềm hoạt động ổn định, đáp ứng được các yêu cầu nghiệp vụ và mang lại trải nghiệm tốt nhất cho người dùng.

Một tester chuyên nghiệp cần trang bị cho mình những kỹ năng và kiến thức cần thiết để có thể thực hiện công việc một cách hiệu quả.

Kiểm thử phần mềm không chỉ giúp phát hiện và sửa chữa lỗi trước khi phần mềm được phát hành mà còn giúp giảm thiểu rủi ro, tăng độ tin cậy của phần mềm và cải thiện trải nghiệm người dùng. Tầm quan trọng của kiểm thử phần mềm đã được nhiều tổ chức và doanh nghiệp nhận thức rõ, từ đó tạo ra nhiều cơ hội nghề nghiệp cho các chuyên viên kiểm thử (testers) với mức lương hấp dẫn.

Đề tài này sẽ đi sâu vào nghiên cứu các kỹ năng và kiến thức cần có của một tester, tầm quan trọng của kiểm thử phần mềm, mức lương và cơ hội nghề nghiệp trong lĩnh vực này. Bên cạnh đó, chúng ta sẽ cùng khám phá cách ứng dụng Selenium với Java để thực hiện tự động hóa quy trình kiểm thử, từ đó giúp các tester nâng cao hiệu quả công việc và đóng góp vào sự phát triển của phần mềm chất lượng cao.

# CHƯƠNG 1. TỔNG QUAN VỀ FPT VÀ LÝ THUYẾT VỀ TESTER

## 1.1. Giới thiệu tổng quát về doanh nghiệp thực tập

### 1.1.1. *Giới thiệu sơ lược về FPT và IVS*

#### a. FPT Software Đà Nẵng

FPT Software Đà Nẵng là công ty thành viên của Tập đoàn FPT, một trong những tập đoàn công nghệ hàng đầu Việt Nam. Công ty được thành lập vào năm 2005 và hiện là một trong những công ty phần mềm hàng đầu tại Việt Nam.

Công ty có trụ sở chính tại tòa nhà FPT Complex, đường Nam Kỳ Khởi Nghĩa, quận Ngũ Hành Sơn, thành phố Đà Nẵng. Công ty cũng có các chi nhánh tại các tỉnh thành khác của Việt Nam, bao gồm Hà Nội, Hồ Chí Minh, Hải Phòng, và Cần Thơ.

*Hình 1.1- 1 Logo tập đoàn FPT*

#### b. IVS Đà Nẵng

IVS - Independent verification services là một công ty con của FPT software và cũng là công ty lớn nhất tại Việt Nam cung cấp dịch vụ xác thực độc lập.

### 1.1.2. *Tầm nhìn – sứ mệnh*

**a. Tầm nhìn**

Trở thành người dẫn đầu trong thị trường chuyển đổi kỹ thuật số tại Việt Nam, góp phần nâng cao giá trị thương hiệu Việt Nam trên bản đồ công nghệ thế giới.

**b. Sứ mệnh**

FPT Software Đà Nẵng cam kết cung cấp các giải pháp công nghệ sáng tạo, mang lại giá trị cho khách hàng, đóng góp cho sự phát triển của xã hội và tạo ra môi trường làm việc tốt nhất cho nhân viên.

### 1.1.3. *Giá trị cốt lõi*

Học & Phát triển: Tạo điều kiện cho nhân viên học hỏi, phát triển bản thân, nâng cao năng lực chuyên môn và kỹ năng mềm.

Thành công Toàn cầu: Khát vọng của công ty trong việc vươn ra thế giới, mang lại giá trị cho khách hàng và cộng đồng.

Sứ mệnh đầy thử thách và có mục đích: FPT Software Đà Nẵng luôn hướng tới những mục tiêu cao cả, góp phần thay đổi cuộc sống con người bằng công nghệ.

Môi trường làm việc vui vẻ và thân thiện: Chú trọng xây dựng môi trường làm việc tích cực, năng động và sáng tạo.

## 1.2. Tổng quan về vị trí tester

### 1.2.1. *Các kỹ năng cần có để trở thành tester*

Để trở thành một tester chuyên nghiệp, cần phải có một loạt các kỹ năng kỹ thuật và phi kỹ thuật (kỹ năng mềm) . Dưới đây là một số kỹ năng quan trọng mà một tester cần có:

**a) Kỹ năng kỹ thuật:**

1. Kiến thức về kiểm thử phần mềm: Hiểu biết về các phương pháp kiểm thử phần mềm, quy trình kiểm thử và các kỹ thuật kiểm thử khác nhau như kiểm thử hộp đen, kiểm

thử hộp trắng, kiểm thử tích hợp, kiểm thử hệ thống, kiểm thử chấp nhận và kiểm thử tự động.

2. Kiến thức về phân tích yêu cầu: Khả năng phân tích tài liệu yêu cầu để xác định các yêu cầu chức năng và phi chức năng, và từ đó xác định các ca kiểm thử phù hợp.

3. Kỹ năng lập kế hoạch kiểm thử: Có khả năng lập kế hoạch cho các hoạt động kiểm thử, bao gồm việc xác định tài nguyên cần thiết, thời gian cần thiết và các bước thực hiện.

4. Kiến thức về kiểm thử tự động: Hiểu biết về công cụ và kỹ thuật kiểm thử tự động, có khả năng viết và thực thi các kịch bản kiểm thử tự động.

5. Kiến thức về môi trường và cơ sở hạ tầng kiểm thử: Hiểu biết về môi trường kiểm thử, bao gồm phần cứng, phần mềm và cơ sở hạ tầng cần thiết để thực hiện các hoạt động kiểm thử.

**b) Kỹ năng phi kỹ thuật:**

1. Kỹ năng giao tiếp: Có khả năng giao tiếp hiệu quả với các thành viên trong nhóm phát triển, những người quản lý dự án và khách hàng để làm rõ các yêu cầu và báo cáo các vấn đề phát hiện trong quá trình kiểm thử.

2. Kỹ năng phân tích và giải quyết vấn đề: Có khả năng phân tích các vấn đề gặp phải trong quá trình kiểm thử và đề xuất các giải pháp hiệu quả để khắc phục chúng.

3. Khả năng làm việc nhóm: Có khả năng làm việc hiệu quả trong một nhóm, hỗ trợ các thành viên khác và chia sẻ kiến thức và kỹ năng với họ.

4. Kiên nhẫn và sự tỉ mỉ: Phải có sự kiên nhẫn để kiểm thử kỹ lưỡng từng chi tiết của phần mềm và sự tỉ mỉ để bảo đảm rằng không có lỗi nào bị bỏ sót.

5. Khả năng học hỏi và cập nhật kiến thức: Công nghệ phần mềm luôn thay đổi, vì vậy một tester cần có khả năng học hỏi liên tục và cập nhật kiến thức để áp dụng các kỹ thuật và công nghệ mới nhất vào công việc của mình.

### 1.2.2. *Mức lương nghề nghiệp*

Mức lương nghề nghiệp của test manual và tester automation, giữa tester có nền tảng tiếng Anh/ tiếng Nhật có sự khác nhau và chênh lệch khá lớn.

Đối với Manual Tester: Theo thống kê từ Glassdoor, mức lương trung bình cho vị trí này từ 12.000.000 VNĐ - 23.000.000 VNĐ tùy vào năng lực, kinh nghiệm. Tuy nhiên đối với vị trí Intern có thể giao động từ 500,000-2,000,000 VND.

Đối với Automation Tester: Theo thống kê từ Glassdoor, mức lương ở vị trí này khá cao tùy thuộc vào vị trí, cấp bậc từ 20.000.000 VNĐ - 40.000.000 VNĐ. [1]

Đặc biệt trong lĩnh vực IT, với năng lực tiếng Anh hoặc tiếng Nhật cao thì mức lương có thể tăng khá cao.

### 1.2.3. *Con đường thăng tiến trong tương lai*

Nhìn chung có 3 con đường để các Tester có thể lựa chọn trong tương lai

**a) Trưởng nhóm kiểm thử (Test Leader)**

Trưởng nhóm kiểm thử đóng vai trò quan trọng trong việc dẫn dắt và quản lý nhóm kiểm thử. Họ chịu trách nhiệm lên kế hoạch, phân công công việc, và giám sát tiến độ kiểm thử để đảm bảo chất lượng sản phẩm.

Yêu cầu:

- Kinh nghiệm nhiều năm trong lĩnh vực kiểm thử phần mềm.

- Kỹ năng lãnh đạo và quản lý đội nhóm.

- Khả năng giao tiếp tốt để phối hợp với các bộ phận khác.

Cơ hội thăng tiến:

- Quản lý kiểm thử (Test Manager): Quản lý kiểm thử chịu trách nhiệm cao hơn, bao gồm lập chiến lược kiểm thử, quản lý ngân sách và tài nguyên, cũng như báo cáo tiến độ kiểm thử cho ban lãnh đạo.

- Quản lý kiểm thử cao cấp (Senior Test Manager): Đây là vị trí cấp cao nhất trong lĩnh vực kiểm thử, đòi hỏi kinh nghiệm dày dặn và khả năng điều hành các dự án kiểm thử lớn.

**b) Quản lý dự án (PM)**

Quản lý dự án là người chịu trách nhiệm quản lý toàn bộ dự án, từ lập kế hoạch, phân bổ tài nguyên, đến giám sát tiến độ và đảm bảo dự án hoàn thành đúng hạn.

- Kinh nghiệm quản lý dự án và kiến thức về quy trình phát triển phần mềm.

- Kỹ năng quản lý thời gian và nguồn lực.

- Khả năng giải quyết vấn đề và ra quyết định nhanh chóng.

**c) Nhà phân tích nghiệp vụ (BA)**

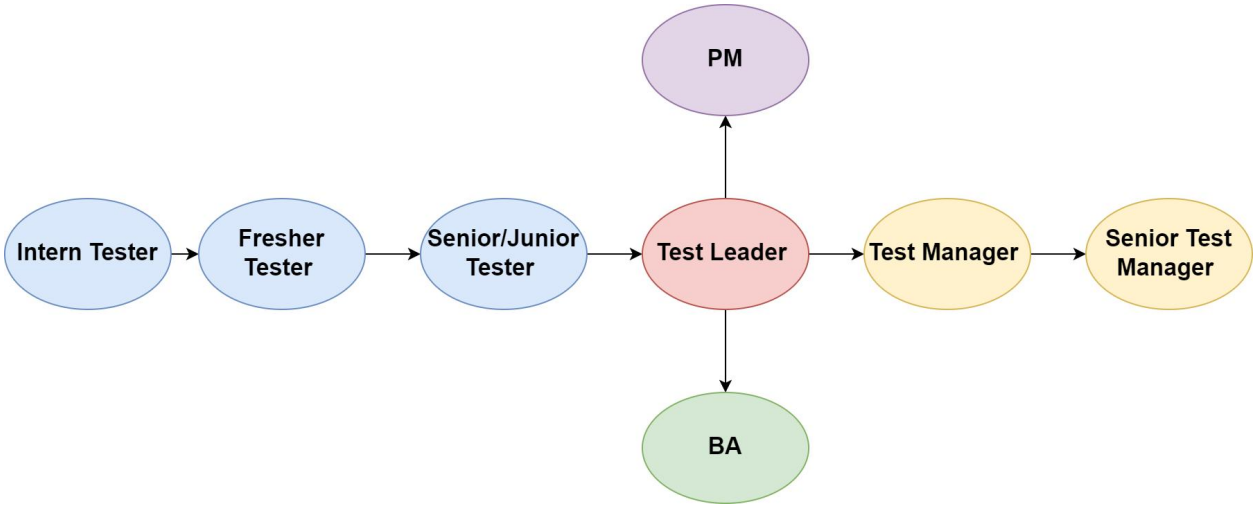
Nhà phân tích nghiệp vụ đóng vai trò trung gian giữa các bên liên quan và đội phát triển, giúp xác định yêu cầu và đảm bảo sản phẩm đáp ứng đúng nhu cầu của người dùng.

- Hiểu biết sâu rộng về lĩnh vực kinh doanh và kỹ thuật phần mềm.

- Kỹ năng phân tích và viết tài liệu yêu cầu.

- Khả năng giao tiếp tốt để làm việc với các bên liên quan khác nhau.

Mỗi con đường sự nghiệp trong lĩnh vực kiểm thử phần mềm đều có những yêu cầu và cơ hội thăng tiến riêng. Việc lựa chọn con đường nào phụ thuộc vào kỹ năng, sở thích và mục tiêu nghề nghiệp của từng cá nhân. Với sự phát triển không ngừng của công nghệ, cơ hội trong lĩnh vực này sẽ ngày càng rộng mở, mang lại nhiều triển vọng phát triển cho các tester.




*Hình 1.2- 1 Cơ hội thăng tiến trong công việc*

# CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VỀ KIỂM THỬ

## 2.1. Tổng quan về kiểm thử phần mềm

### 2.1.1. *Kiểm thử phần mềm là gì ?*

Kiểm thử phần mềm là quá trình đánh giá và xác minh rằng một phần mềm hoặc ứng dụng cụ thể đáp ứng các yêu cầu đã định trước và không có lỗi. Mục tiêu của kiểm thử là xác định các vấn đề trước khi phần mềm được phát hành đến người dùng cuối, từ đó đảm bảo chất lượng và độ tin cậy của sản phẩm.

### 2.1.2. *Loại kiểm thử phần mềm*

Kiểm thử phần mềm được chia thành nhiều loại hình khác nhau, mỗi loại có mục đích và phương pháp riêng. Dưới đây là một số loại hình kiểm thử phổ biến:

a. Kiểm thử chức năng (Functional Testing)

Kiểm thử chức năng tập trung vào việc kiểm tra các chức năng của phần mềm dựa trên các yêu cầu đã được xác định. Mục tiêu là đảm bảo rằng tất cả các chức năng hoạt động đúng như mong đợi.

b. Kiểm thử phi chức năng (Non-Functional Testing)

Kiểm thử phi chức năng kiểm tra các khía cạnh không liên quan đến chức năng cụ thể của phần mềm, chẳng hạn như hiệu suất, tính khả dụng, bảo mật và tính tương thích.

c. Kiểm thử đơn vị (Unit Testing)

Kiểm thử đơn vị tập trung vào kiểm tra các thành phần nhỏ nhất của phần mềm, chẳng hạn như các hàm hoặc module riêng lẻ, để đảm bảo chúng hoạt động chính xác.

d. Kiểm thử tích hợp (Integration Testing)

Kiểm thử tích hợp kiểm tra sự tương tác giữa các thành phần hoặc module của phần mềm để đảm bảo rằng chúng hoạt động cùng nhau một cách chính xác.

e. Kiểm thử hệ thống (System Testing)

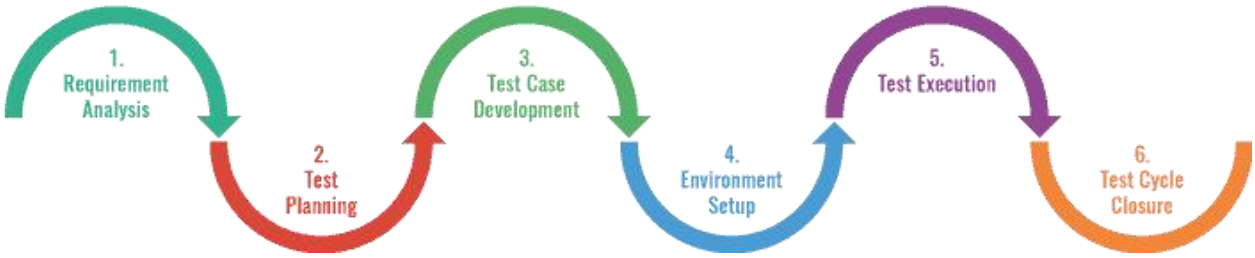
Kiểm thử hệ thống kiểm tra toàn bộ hệ thống phần mềm sau khi các thành phần đã được tích hợp, đảm bảo rằng hệ thống đáp ứng các yêu cầu đã định trước.

f. Kiểm thử chấp nhận (Acceptance Testing)

Kiểm thử chấp nhận là giai đoạn cuối cùng trước khi phần mềm được phát hành, nhằm xác định xem phần mềm có đáp ứng các yêu cầu và mong đợi của khách hàng hay không.

### 2.1.3. *Quy trình kiểm thử phần mềm (Software testing life cycle( STLC))*

Quy trình kiểm thử phần mềm (STLC) là một phương pháp tiếp cận có hệ thống để kiểm thử một ứng dụng phần mềm nhằm đảm bảo rằng ứng dụng đó đáp ứng các yêu cầu và không có lỗi. Đây là một quy trình tuân theo một loạt các bước hoặc giai đoạn, và mỗi giai đoạn có các mục tiêu và kết quả cụ thể. STLC được sử dụng để đảm bảo rằng phần mềm có chất lượng cao, đáng tin cậy và đáp ứng nhu cầu của người dùng cuối.




***Hình 2.1- 1 Hình ảnh quy trình kiểm thử phần mềm***

1. Phân tích yêu cầu : Phân tích yêu cầu là bước đầu tiên của Vòng đời kiểm tra phần mềm (STLC). Trong giai đoạn này, nhóm đảm bảo chất lượng hiểu được các yêu cầu cũng như những gì cần kiểm tra. Nếu có bất kỳ điều gì còn thiếu hoặc không thể hiểu được thì nhóm đảm bảo chất lượng sẽ xem xét các bên liên quan để hiểu rõ hơn về kiến thức chi tiết của các yêu cầu.

2. Test planning : Lên kế hoạch kiểm tra là giai đoạn hiệu quả nhất của phần mềm kiểm tra vòng đời, trong đó tất cả các kiểm tra kế hoạch đều được xác định. Trong giai đoạn này, người quản lý kiểm tra sẽ tính toán nỗ lực và chi phí ước tính cho công việc kiểm tra. Giai đoạn này bắt đầu sau khi giai đoạn thu thập yêu cầu hoàn thành.

3. Test Case Development : Giai đoạn phát triển giai đoạn kiểm tra hợp đồng trường được bắt đầu sau khi giai đoạn kiểm tra kế hoạch thiết lập giai đoạn hoàn tất. Trong giai đoạn này, nhóm kiểm thử lại các trường hợp kiểm thử chi tiết. kiểm thử nhóm cũng là kiểm thử dữ liệu cần thiết để kiểm thử. Khi các trường kiểm thử hợp lý được chuẩn bị thì chúng sẽ được nhóm đánh giá đảm bảo chất lượng.

4. Thiết lập môi trường kiểm thử :

Đây là một phần quan trọng của STLC. Về cơ bản, kiểm tra môi trường để xác định các điều kiện mà phần mềm được kiểm tra. Đây là hoạt động độc lập và có thể được bắt đầu cùng với kiểm thử phát triển môi trường việc làm. Trong quá trình này, kiểm thử nhóm không tham gia. nhà phát triển hoặc khách hàng tạo môi trường kiểm thử.

5. Thực hiện kiểm thử :

Sau khi thực hiện xong các giai đoạn trên, giai đoạn thực hiện kiểm tra sẽ bắt đầu. Trong giai đoạn này, nhóm kiểm tra bắt đầu thực hiện các kiểm tra dựa trên các tiêu chuẩn kiểm tra ca đã được thực hiện ở bước trước.

Chuẩn bị kiểm thử dữ liệu: kiểm thử dữ liệu đã được chuẩn hóa và tải vào hệ thống để thực hiện kiểm thử

Kiểm thử môi trường thiết lập: Cần phải thiết lập phần cứng cấu hình, phần mềm và mạng để thực hiện kiểm thử

Kiểm tra lại lỗi: Bất kỳ lỗi nào được xác định trong quá trình thực hiện kiểm thử đều được kiểm tra lại để đảm bảo rằng chúng đã được sửa chính xác.

Báo cáo kiểm thử: Kết quả kiểm thử được ghi lại và báo cáo cho các bên liên quan.

Điều quan trọng cần lưu ý là việc thực hiện kiểm thử là một quá trình lặp đi lặp lại và có thể cần phải lặp lại nhiều lần để cho đến khi tất cả các lỗi đã được xác định được giải quyết và phần mềm được coi là phù hợp để phát hành hành động.

6. Đóng kiểm thử : Đóng kiểm thử là giai đoạn cuối cùng của vòng đời kiểm tra phần mềm (STLC) trong đó tất cả các hoạt động liên quan đến kiểm thử đã được hoàn thành thành công và ghi lại. Mục tiêu chính của giai đoạn kiểm thử là đảm bảo rằng tất cả các hoạt động liên quan đến kiểm thử đã được hoàn thành thành công và phần mềm đã sẵn sàng để phát hành. [2]

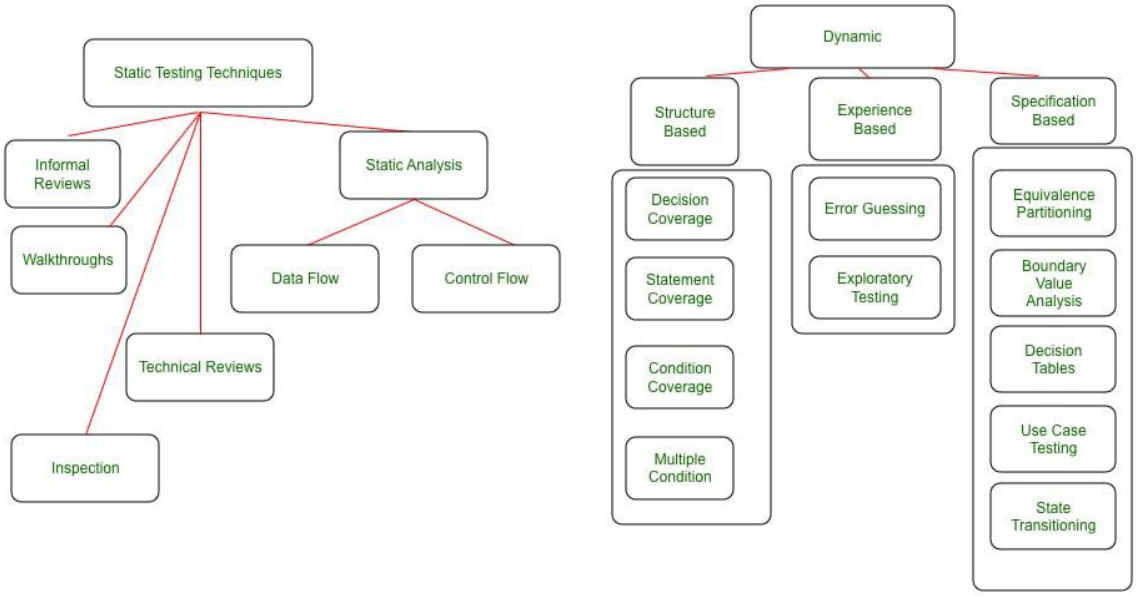
### 2.1.4. *Các kĩ thuật kiểm thử phần mềm*

Có hai loại kỹ thuật kiểm thử phần mềm chính:

Kỹ thuật kiểm thử tĩnh là các kỹ thuật kiểm thử được sử dụng để tìm ra lỗi trong ứng dụng đang được kiểm thử mà không thực thi mã. Kiểm tra tĩnh được thực hiện để tránh lỗi ở giai đoạn đầu của chu kỳ phát triển, do đó giảm chi phí sửa chúng.

Kỹ thuật kiểm tra động là các kỹ thuật kiểm tra được sử dụng để kiểm tra hành vi động của ứng dụng đang được kiểm tra, tức là bằng cách thực thi cơ sở mã. Mục đích chính của thử nghiệm động là kiểm tra ứng dụng với đầu vào động - một số trong đó có

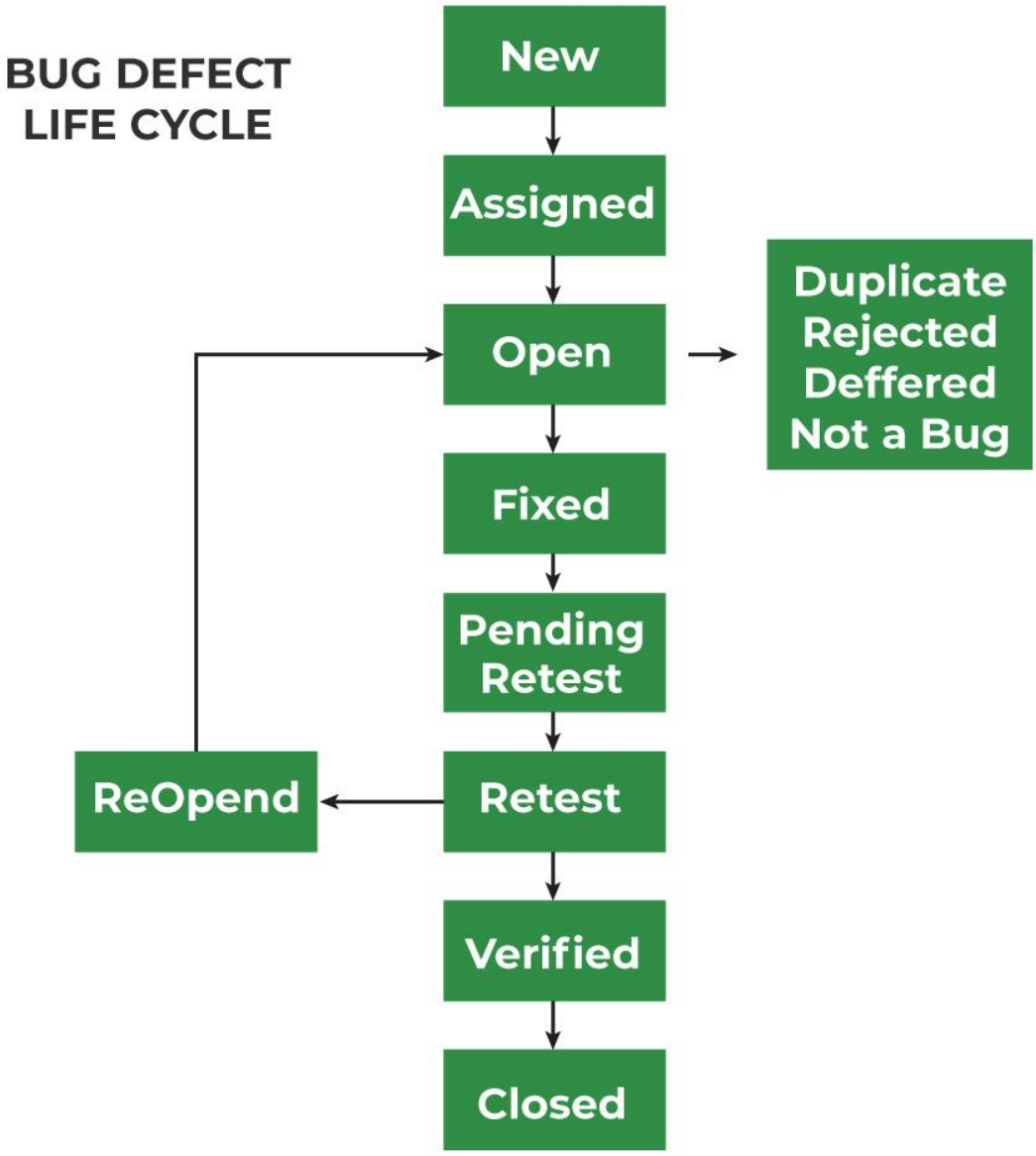
thể được cho phép theo yêu cầu (Thử nghiệm tích cực) và một số không được phép (Thử nghiệm tiêu cực) [3]




***Hình 2.1- 2 Hình ảnh các loại kĩ thuật kiểm thử phần mềm***

### 2.1.5. *Vòng đời của bug*

Vòng đời lỗi là một loạt các giai đoạn mà lỗi trải qua từ thời điểm phát hiện cho đến khi được giải quyết. Nó giúp các nhà kiểm thử phần mềm và nhà phát triển theo dõi và quản lý trạng thái và tiến trình của lỗi, cũng như giao tiếp hiệu quả với các bên liên quan. Một vòng đời lỗi được xác định rõ ràng có thể cải thiện chất lượng và hiệu quả của việc kiểm thử và phát triển phần mềm.




*Hình 2.1- 3 Hình ảnh vòng đời của bug [4]*

### 2.1.6. *Vai trò của tester trong kiểm thử phần mềm*

Người kiểm thử đóng vai trò quan trọng trong quá trình phát triển phần mềm. Công việc của người kiểm thử không chỉ là tìm kiếm và báo cáo lỗi mà còn đảm bảo phần mềm hoạt động theo mong đợi của người dùng. Người kiểm thử phải có kiến thức về quy trình phát triển phần mềm, ngôn ngữ lập trình và khả năng phân tích yêu cầu.

Vai trò của một người kiểm thử là đảm bảo chất lượng của phần mềm trước khi phát hành ra thị trường. Người kiểm thử phải có khả năng tìm kiếm lỗi, sử dụng các công cụ kiểm thử và tiến hành kiểm thử chi tiết và toàn diện. Kỹ năng ghi lại lỗi và báo cáo cũng rất quan trọng. Người kiểm thử không chỉ đánh giá chất lượng của phần mềm mà còn góp phần cải thiện quy trình phát triển phần mềm trong tương lai.

## 2.2. Quy trình phát triển phần mềm

Phát triển phần mềm có nhiều mô hình khác nhau, nhưng hai mô hình phổ biến nhất là mô hình Waterfall và Agile. Cả hai đều có những ưu và nhược điểm riêng, phù hợp với các loại dự án và yêu cầu khác nhau. Dưới đây là tổng quan về cả hai mô hình này.

### 2.2.1. *Mô hình Waterfall*

Mô hình Waterfall (thác nước) là một trong những mô hình phát triển phần mềm truyền thống nhất. Nó tuân theo một quy trình tuần tự, trong đó mỗi giai đoạn phải hoàn thành trước khi bước vào giai đoạn tiếp theo.

Các giai đoạn của mô hình Waterfall:

1. Yêu cầu (Requirements): Thu thập và phân tích yêu cầu từ khách hàng hoặc người dùng. Tạo tài liệu yêu cầu chi tiết.

2. Thiết kế hệ thống (System Design): Xây dựng kiến trúc hệ thống và thiết kế chi tiết dựa trên yêu cầu đã thu thập.

3. Triển khai (Implementation): Lập trình và phát triển phần mềm theo thiết kế đã định.

4. Kiểm thử (Testing): Thực hiện kiểm thử phần mềm để phát hiện và sửa lỗi.

5. Triển khai và bảo trì (Deployment & Maintenance): Triển khai phần mềm đến người dùng cuối và duy trì, cập nhật, và sửa lỗi khi cần.

**a) Ưu điểm của mô hình Waterfall:**

- Đơn giản và dễ hiểu: Mô hình tuần tự và rõ ràng, dễ dàng để quản lý và theo dõi tiến độ.

- Quản lý tài liệu tốt: Tài liệu chi tiết và đầy đủ trong mỗi giai đoạn, giúp duy trì kiểm soát tốt hơn.

- Thích hợp cho các dự án nhỏ: Với các yêu cầu rõ ràng và cố định.

**b) Nhược điểm của mô hình Waterfall:**

- Thiếu linh hoạt: Khó thay đổi yêu cầu sau khi giai đoạn đầu đã hoàn thành.

- Phát hiện lỗi muộn: Lỗi có thể chỉ được phát hiện ở giai đoạn kiểm thử, gây tốn kém thời gian và chi phí để sửa chữa.

- Không phù hợp với các dự án phức tạp: Khi yêu cầu có thể thay đổi trong quá trình phát triển.

### 2.2.2. *Mô hình Agile*

Mô hình Agile là một phương pháp phát triển phần mềm linh hoạt, nhấn mạnh sự cộng tác, cải tiến liên tục và phản hồi nhanh chóng từ người dùng. Agile phát triển phần mềm theo từng đợt nhỏ, gọi là "sprint," với mỗi sprint thường kéo dài từ 1-4 tuần.

Các nguyên tắc cơ bản của Agile:

1. Cộng tác với khách hàng: Sự tham gia liên tục của khách hàng trong suốt quá trình phát triển.

2. Thay đổi yêu cầu: Chấp nhận và linh hoạt với những thay đổi yêu cầu, ngay cả trong giai đoạn muộn của phát triển.

3. Phát hành sớm và thường xuyên: Phát hành các phiên bản nhỏ của phần mềm để nhận phản hồi nhanh chóng.

4. Làm việc nhóm: Các nhóm nhỏ, tự quản lý và cộng tác cao độ.

Các phương pháp Agile phổ biến:

- Scrum: Tập trung vào các sprint, với các cuộc họp hàng ngày (daily stand-up) và đánh giá sau mỗi sprint (sprint review).

- Kanban: Sử dụng bảng Kanban để trực quan hóa luồng công việc và tối ưu hóa quá trình.

**a) Ưu điểm của mô hình Agile:**

- Linh hoạt và phản hồi nhanh: Dễ dàng thích nghi với các thay đổi yêu cầu.

- Phát hiện và sửa lỗi sớm: Kiểm thử và phản hồi liên tục trong suốt quá trình phát triển.

- Tăng cường sự hài lòng của khách hàng: Sự tham gia liên tục của khách hàng đảm bảo sản phẩm cuối cùng đáp ứng yêu cầu.

**b) Nhược điểm của mô hình Agile:**

- Quản lý khó khăn hơn: Đòi hỏi sự quản lý và tổ chức tốt để duy trì luồng công việc.

- Tài liệu không đầy đủ: Có thể thiếu tài liệu chi tiết do tập trung vào phát triển nhanh chóng.

- Phụ thuộc vào nhóm: Thành công phụ thuộc lớn vào sự hợp tác và kỹ năng của nhóm phát triển.

**Kết luận**

Cả hai mô hình Waterfall và Agile đều có những ưu và nhược điểm riêng, và lựa chọn mô hình nào phụ thuộc vào yêu cầu cụ thể của dự án. Waterfall thích hợp cho các dự án nhỏ, cố định và có yêu cầu rõ ràng, trong khi Agile phù hợp với các dự án phức tạp, yêu cầu linh hoạt và sự tham gia liên tục của khách hàng. Việc hiểu rõ từng mô hình giúp các nhà quản lý dự án và nhóm phát triển lựa chọn phương pháp phù hợp nhất để đạt được mục tiêu dự án.

Trong thực tế, nhiều công ty thường áp dụng 2 mô hình này để thực hiện nhiều dự án để tận dụng được tối đa ưu điểm của các mô hình.

## 2.3. Test Case và Log bug

### 2.3.1. *Test Case là gì?*

Test case là "một tập hợp các thông số đầu vào kiểm thử, điều kiện thực thi, và kết quả mong đợi được phát triển cho một mục tiêu cụ thể, như thực hiện một chương trình cụ thể,kiểm tra sự tuân thủ với một yêu cầu cụ thể."

Vậy test case là gì? Test case (Kịch bản kiểm thử) hiểu đơn giản là tài liệu dùng để mô tả: Dữ liệu đầu vào (Input) – Hành động (Active) – Kết quả mong đợi (Expected response) để xác định một chức năng của ứng dụng phần mềm hoạt động đúng hay không.

Test case thường được Tester viết trên Excel hoặc Google Sheet. Một test case có thể có các phần đặc thù khác nhau như mã test case, tên test case, mục tiêu test, các điều kiện test, các yêu cầu data input, các bước thực hiện và các kết quả mong đợi. Mức chi tiết test case dựa vào ngữ cảnh của dự án và quy mô của công ty sản xuất phần mềm.

### 2.3.2. *Các thành phần của Test Case:*

**TC_ID:** Test Case ID

**Group Check:** Đây là các nhóm cần check, thông thường thường có UI, Validate, Business hoặc các group về Security…

**Item check:** Là cái item thực hiện check

**Pre-condition:** Đây là điều kiện ưu tiên trước khi thực hiện test

**Summary:** sơ lược việc mình thực hiện test ở đây là gì.

**Step by step**: Các bước thực hiện Test

**Input data:** dữ liệu test, thường dữ liệu này sẽ có ở requirement hoặc các tài liệu chuẩn bị trước để căn cứ vào đó

**Expected result:** đây là kết quả mong muốn khi test được thực hiện xong, nghĩa là test case đó được pass.

**Status:** Passed/Failed _ đây là hai trạng thái thường gặp khi mà kết quả thực tế và kết quả mong đợi được so sánh với nhau

**Who check:** Người thực hiện test case

**Date check:** Ngày thực hiện test

**No. Bug:** khi bug failed thì cần để No_Bug ở đây để dễ dàng trong việc đối chiếu.

### 2.3.3. *Log Bug là gì?*

Log bug có thể hiểu đơn giản việc báo cáo mô tả lỗi hay còn gọi là Bug report. Mục đích chính là để các bộ phận phát triển phần mềm có thể dễ dàng tìm kiếm nguyên nhân và sửa chữa kịp thời. Đây là công việc hàng ngày mà các Tester cần thực hiện.

### 2.3.4. *Các thành phần cần có của Log Bug*

**No_bug:** có thể được dùng để đánh dấu bug

**Title bug:** mô tả sơ lược về bug đó

**Description:** mô tả chi tiết về bug

**Step by step:** Bước thực hiện để xuất hiện bug

**Actual result:** kết quả thực tế khi thực hiện

**Expected result:** kết quả mong đợi

**No_TestCase:** So sánh với test case mình viết

Ngoài ra Log Bug còn có độ ưu tiên và độ nghiêm trọng

| **Độ ưu tiên** | **Mức độ nghiêm trọng** |
|---|---|
| Xác định thứ tự mà dev cần giải quyết | Xác định mức độ ảnh hưởng của lỗi đối với phần mềm |
| Liên quan đến việc lập kế hoạch | Liên quan đến tiêu chuẩn và các chức năng khác |
| Độ ưu tiên cho biết lỗi nên được sửa sớm ở mức nào | Mức độ nghiêm trọng cho thấy độ nghiêm trọng của lỗi trên chức năng sản phẩm |
| Độ ưu tiên được quyết định với sự tham vấn của quản lý / khách hàng | QA xác định mức độ nghiêm trọng của bug |
| Độ ưu tiên được xác định bởi nghiệp vụ | Mức độ nghiêm trọng được xác định bởi chức năng |

| Độ ưu tiên | Mức độ nghiêm trọng |
|---|---|
| Có thể thay đổi trong một khoảng thời gian tùy thuộc vào tình hình dự án | Ít có khả năng thay đổi |
| Khi UAT, team phát triển sẽ fix các lỗi dựa vào mức độ ưu tiên | Trong quá trình SIT, team phát triển fix lỗi dựa trên mức độ nghiêm trọng nhiều hơn và sau đó là mức độ ưu tiên |

***Bảng 2.3- 1 Bảng so sánh độ ưu tiên và độ nghiêm trọng của Bug***

- Việc lựa chọn bug nào cần được fix trước và sau còn phụ thuộc vào nhiều yếu tố và đánh giá của hệ thống, chính vì vậy không phải lúc nào cũng bắt buộc phải theo một đánh giá cụ thể nào được. Nó còn phụ thuộc vào các mức độ ảnh hưởng đến các hướng khác và luồng xử lý công việc của hệ thống.

### 2.3.5. *Trạng thái của Test Case trong IntelliJ IDEA*

1. **Passed**: Test case đã chạy thành công và tất cả các kiểm thử (assertions) trong test case đều đúng.
2. **Failed**: Test case đã chạy nhưng một hoặc nhiều kiểm thử (assertions) trong test case không đúng.
3. **Error**: Test case không thể chạy hoàn chỉnh do lỗi ngoài dự kiến, chẳng hạn như lỗi ngoại lệ (exception) không được xử lý.
4. **Skipped**: Test case không được chạy do bị bỏ qua. Điều này có thể do cấu hình hoặc điều kiện nào đó khiến test case không được thực thi.
5. **Running**: Test case đang trong quá trình chạy.
6. **Ignored**: Test case bị bỏ qua một cách có chủ đích (thường được đánh dấu bằng các chú thích như @Ignore trong JUnit hoặc @Test(enabled = false) trong TestNG).

- **Cách kiểm tra trạng thái của Test Case**

Khi chạy test trong IntelliJ IDEA, bạn có thể thấy các trạng thái này trong tab "Run" hoặc "Test" ở dưới cùng của IDE. Mỗi test case sẽ có một biểu tượng tương ứng với trạng thái của nó:

- **Passed**: Thường được biểu thị bằng một dấu kiểm màu xanh lá cây.
- **Failed**: Thường được biểu thị bằng một dấu X màu đỏ.
- **Error**: Thường được biểu thị bằng một dấu chấm than màu đỏ.
- **Skipped**: Thường được biểu thị bằng một biểu tượng màu vàng hoặc dấu gạch ngang.
- **Running**: Thường được biểu thị bằng một vòng tròn đang xoay.
- **Ignored**: Thường được biểu thị bằng một biểu tượng màu xám.

# CHƯƠNG 3. SƠ LƯỢC VỀ KIỂM THỬ TỰ ĐỘNG

## 3.1. Sơ lược về kiểm thử tự động:

### 3.1.1. *Khái niệm về kiểm thử tự động*

Kiểm thử tự động là việc sử dụng các công cụ để thực hiện các test case. Kiểm thử tự động cũng có thể nhập dữ liệu thử nghiệm vào hệ thống kiểm thử, so sánh kết quả mong đợi với kết quả thực tế và tạo ra các báo cáo kiểm thử chi tiết. [5]

### 3.1.2. *Sự khác nhau giữa kiểm thử tự động và kiểm thử thủ công*

| Parameter | Manual testing | Automation testing |
|---|---|---|
| Định nghĩa | Testcase được thực hiện thủ công bởi tester | Tester phải viết test script và lựa chọn công cụ để tự động hóa việc test |
| Thời gian xử lý | Cần nhiều thời gian và nhân lực | Thời gian kiểm thử nhanh hơn so với manual testing |

| | | |
|---|---|---|
| Exploratory Testing/ Kiểm thử khám phá | Exploratory Testing/ Kiểm thử khám phá được thực hiện | Không cho phép kiểm thử khám phá |
| Thay đổi UI | Sự thay đổi nhỏ như ID, Class hoặc 1 button nhưng không ảnh hưởng đến thực thi test | Chỉ 1 vài thay đổi nhỏ trong UI, người dùng phải update script để đảm bảo có kết quả như mong đợi |
| Độ tin cậy | Kết quả kiểm thử không đáng tin cậy vì có khả năng xảy ra lỗi do con người | Do được thực thi bằng tool và scripts nên kết quả đáng tin cậy hơn |
| Đầu tư | Cần nhiều nguồn nhân lực | Bắt buộc phải đầu tư tool để test và những kĩ sư auto |
| Báo cáo | Manual test thường lưu lại kết quả ở Excel, Word... | Tất cả stakeholders có thể đăng nhập vào hệ thống auto và kiểm tra lại kết quả test |
| Sự quan sát của con người | Cần có sự quan sát của con người để giúp cho hệ thống thân thiện với người dùng | Không có sự quan sát của con người |
| Kiểm thử hiệu năng/Performance Testing | Không thực hiện được Kiểm thử hiệu năng/Performance Testing | Kiểm thử hiệu năng/ Performance Testing phải được thực hiện bởi 1 tool phù hợp |

| | | |
|---|---|---|
| Kiến thức lập trình | Không cần có khả năng code | Phải có kiến thức về lập trình để tạo ra các test script |
| Cách tiếp cận tốt | Manual testing hữu ích khi chúng ta chạy lại bộ testcase 1 hoặc 2 lần | Auto test rất hữu ích khi ta chạy lại bộ script nhiều lần |
| Sử dụng khi nào? | Kiểm thử thủ công phù hợp cho Exploratory Testing/ test khám phá, Usability/ Khả năng sử dụng và Adhoc Testing/ Kiểm thử dựa vào thực tế | Test auto thích hợp cho kiểm thử hồi quy, hiệu năng hoặc các trường hợp có khả năng lặp lại nhiều lần |

***Bảng 3.1- 1 Bảng biểu sự khác nhau giữa kiểm thử tự động và kiểm thử manual***

### 3.1.3. *Ưu nhược điểm của kiểm thử tự động so với kiểm thử thủ công*

**a) Ưu điểm:**

- Quá trình kiểm thử diễn ra nhanh chóng và hiệu quả hơn.
- Kiểm tra tự động giúp chúng ta tìm thấy nhiều lỗi hơn so với con người
- Quá trình test được ghi lại => Cho phép sử dụng lại hàng loạt các hoạt động thử nghiệm.
- Kiểm thử tự động được thực hiện bằng cách sử dụng các công cụ phần mềm, do đó nó hoạt động không mệt mỏi, không giống như con người trong kiểm tra thủ công.
- Kiểm tra tự động hỗ trợ các ứng dụng khác nhau.
- Phạm vi kiểm tra có thể được tăng lên vì các tool không bao giờ quên kiểm tra ngay cả đơn vị nhỏ nhất.

**b) Nhược điểm:**

- Nếu không có yếu tố con người, rất khó để có được cái nhìn sâu sắc về các khía cạnh trực quan của giao diện người dùng của bạn như màu sắc, phông chữ, kích thước, độ tương phản hoặc kích thước nút.
- Các công cụ để chạy thử nghiệm tự động hóa có thể đắt tiền, có thể làm tăng chi phí của dự án.
- Công cụ chạy auto test vẫn chưa hoàn hảo.
- Bảo trì tốn kém.

## 3.2. Tổng quan về selenium

### 3.2.1. *Selenium là gì*

Selenium là một automation testing framework miễn phí (mã nguồn mở).
Nó được sử dụng để kiểm thử các ứng dụng web trên các trình duyệt (chrome, firefox, ms edge, …) và nền tảng khác nhau (Windows, Mac, Linux, …). Selenium hỗ trợ nhiều loại ngôn ngữ lập trình như: Java, C , Python, ... để tạo ra các bộ test script.

### 3.2.2. *Các thành phần của Selenium*

Selenium không chỉ là một công cụ đơn lẻ mà là một bộ gồm 4 công cụ, mỗi công cụ đáp ứng nhu cầu kiểm thử khác nhau.

Selenium IDE: Selenium Integrated Development Environment (IDE) là một plugin trên trình duyệt Chrome và Firefox. Ta có thể sử dụng chúng để ghi và phát lại(record and playback) các tương tác của người dùng theo một quy trình hay một test case nào đó.g mạnh mẽ hơn, mà hiện nay phần lớn các project Selenium đều sử dụng.

Selenium WebDriver: Selenium WebDriver là một automation testing tool dành riêng cho web, nó sẽ gửi lệnh khởi chạy và tương tác trực tiếp tới các trình duyệt.

Selenium Grid: Selenium Grid được sử dụng để khởi chạy nhiều kịch bản test song song cùng một lúc, và có thể chạy trên nhiều máy, nhiều hệ điều hành và nhiều trình duyệt khác nhau.

### 3.2.3. *Các tính năng nổi bật của Selenium*

- Tới thời điểm hiện tại, Selenium luôn nằm trong danh sách Top 10 Testing Automation Tools.
- Selenium là mã nguồn mở. Do đó, mọi người có thể download source code về sử dụng, và thay đổi tùy theo nhu cầu.
- Cộng đồng sử dụng rộng rãi. Thường xuyên được phát triển và cải tiến mạnh mẽ.

## 3.3. Cài đặt - thiết lập môi trường java và selenium trong intellij idea [6]

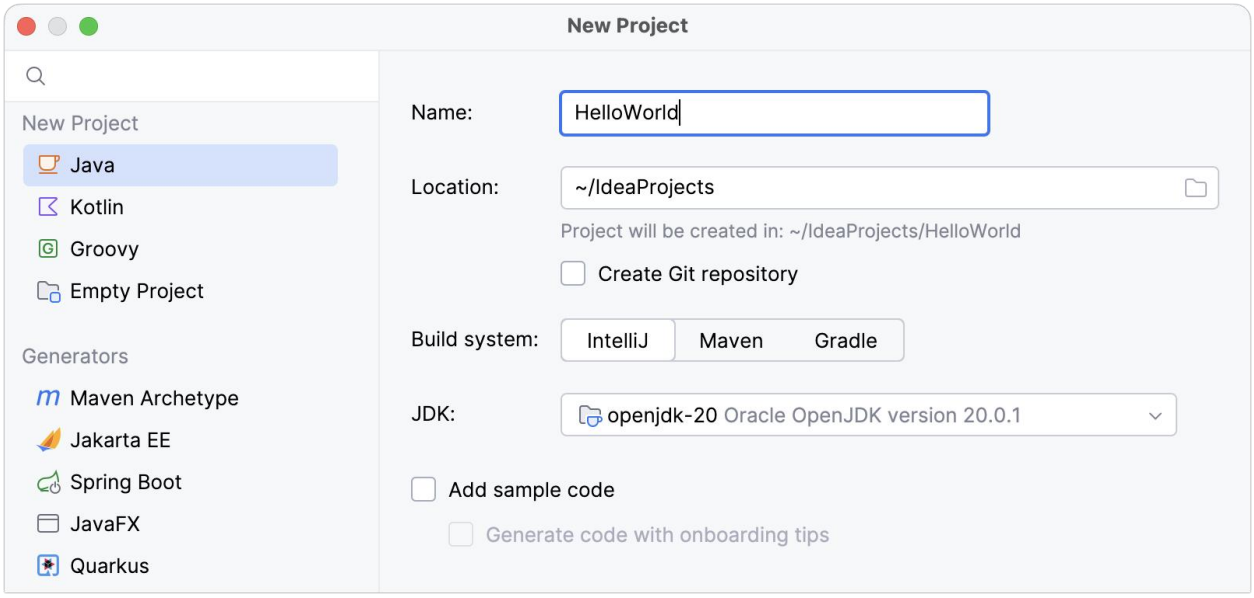
In IntelliJ IDEA, a project helps you organize your source code, tests, libraries that you use, build instructions, and your personal settings in a single unit.

In the New Project wizard, select Java from the list on the left.

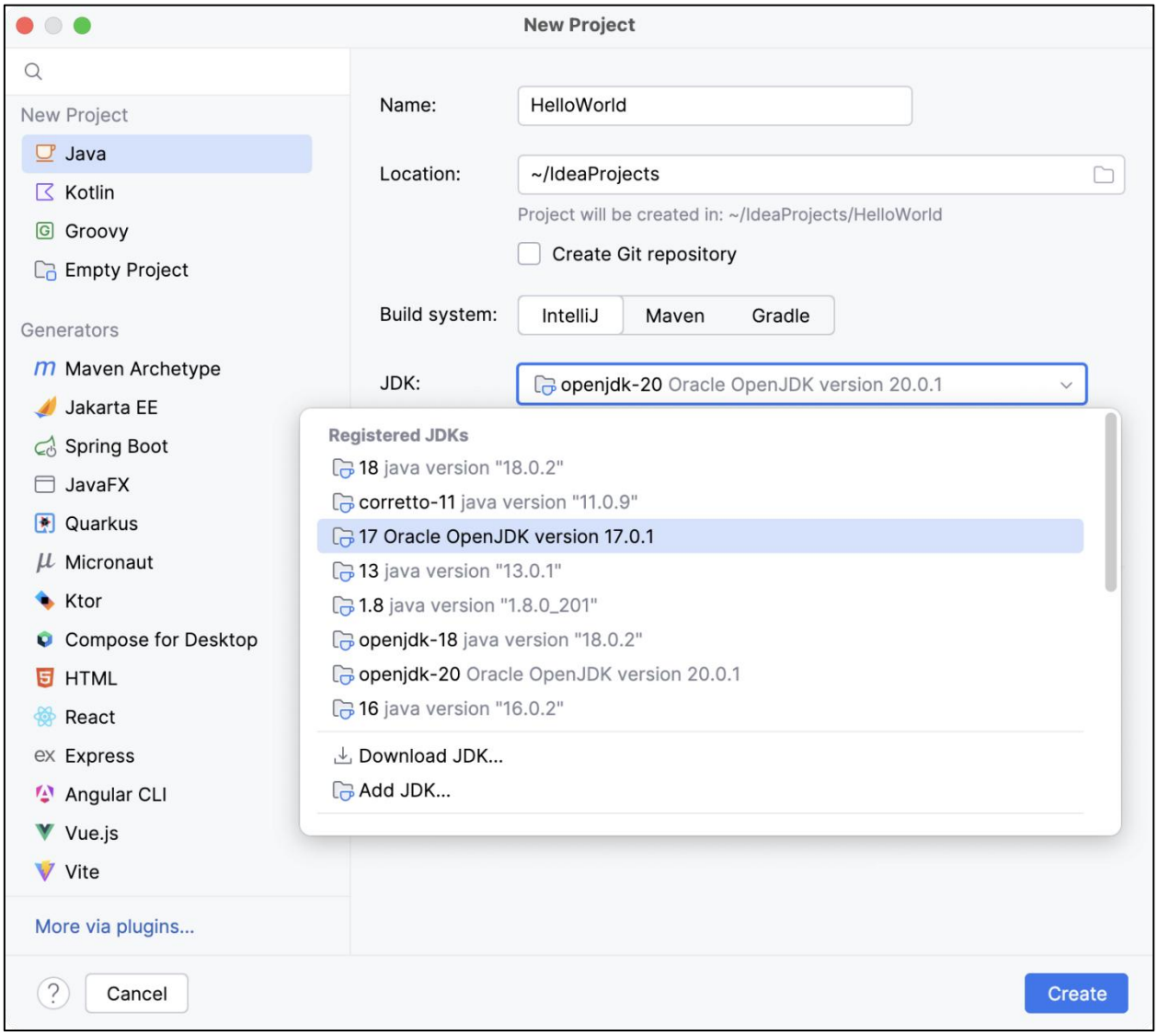
Name the project (for example HelloWorld) and change the default location if necessary.

We're not going to work with version control systems in this tutorial, so leave the Create Git repository option disabled.

Make sure that IntelliJ is selected in Build system.




***Hình 3.3- 1 Tạo project mới cho chương trình***




*Hình 3.3- 2 Chọn JDK (môi trường cho java)*


*Hình 3.3- 3 Đặt JDK đúng với vị trí được tải xuống*

# CHƯƠNG 4. THỰC HÀNH KIỂM THỬ TỰ ĐỘNG CHỨC NĂNG SEARCH TRONG MÀN HÌNH CHAT VỚI SELENIUM

**Requirement chức năng Search trong màn hình Chat**

***Search Functionality Requirement***

*Search Components:*

- Name: An input field with a maximum length of 100 characters, where users can enter a specific name, a part of a name, or the initial character of a name. The system will search for records containing the entered string.

- Recent Searches: When clicking on the search input field or pressing enter, the system will display up to the 10 most recent searches.

*Search Functionality:*

- No Search Criteria: If user input * are entered in the search fields, the system will display all available results.
- If not have result matched with input, system will inform: "Không có kết quả phù hợp"

- With Search Criteria: If one or more values are entered in the search fields, the system will display only the results that meet all entered conditions.

*Processing:*

- Search Request: When the user completes their input and clicks the search button, the system will send a search request with the entered information to the server or database.

- Server/Database Processing: The server or database will process the search request and return the appropriate results.

- Displaying Results: The results will be displayed on the search screen based on the data returned from the server or database.

***Pagination Search Requirements***

*Data Display*

- Display up to 10 items per page.

- Users can scroll down to load more data.

*Search Functionality*

- Search results must be paginated to show 10 items per page.

*Pagination Functionality*

- Automatically load more data when the user scrolls to the bottom of the page.

## 4.1. Requirement analysis

***Overview***

The search screen functionality needs to be implemented to allow users to search for records by name. The requirements include handling user input, displaying recent searches, and processing search requests to display results.

***Key Requirements***

- Input field for name with a maximum length of 100 characters.

- Display up to 10 most recent searches.

- Display all results if no search criteria are entered.

- Filter results based on entered search criteria.

- Send search requests to the server or database and display returned results.

## 4.2. Test Planning

Objective

To ensure the search screen functionality works as expected and meets the specified requirements.

Scope

- Verify the input field constraints.

- Validate recent searches display.

- Test search results display for various scenarios.

- Verify search request processing.

## 4.3. Test Case Development

### 4.3.1. *Decision table:*

Thực hiện vẽ bảng để cover hầu hết các test case có thể xảy ra

Việc tạo bảng decision table giúp ích cho quá trình check của manager khi đánh giá test case được phổ quát và rõ ràng hơn

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Input** | verify behavior in case click on Search button | | | | | | | | | | | | | | | | | | | | | | | |
| | | Tìm kiếm | Trường hợp nhập 101 kí tự | x | x | | | | | | | | | | | | | | | | | | | |
| | | | Trường hợp nhập 100 kí tự | | | x | x | | | | | | | | | | | | | | | | | |
| | | | Trường hợp nhập toàn bộ tên | | | | | x | x | | | | | | | | | | | | | | | |
| | | | Trường hợp nhập tên đúng 1 phần trước | | | | | | | x | x | | | | | | | | | | | | | |
| | | | Trường hợp nhập tên đúng 1 phần sau | | | | | | | | | x | x | | | | | | | | | | | |
| | | | Trường hợp nhập trên 1 kí tự đúng ở phần sau | | | | | | | | | | | x | x | | | | | | | | | |
| | | | Trường hợp nhập tên đúng 1 kí tự đầu tiên | | | | | | | | | | | | | x | x | | | | | | | |
| | | | Không nhập data | | | | | | | | | | | | | | | x | x | | | | | |
| | | Tồn tại data | | | | | | | | | | | | | | | | | | | | | | |
| | | | Thỏa mãn | x | | x | | x | | x | | x | | x | | x | | x | | x | | | | |
| | | | Không thỏa mãn | | x | | x | | x | | x | | x | | x | | x | | x | | x | | | |
| | Verify in case user click button remove | | | | | | | | | | | | | | | | | | | | | | | |
| | | Trường hợp đã nhập nhưng không có kết quả | | | | | | | | | | | | | | | | | | | | | x | |
| | | Trường hợp đã nhập và đã có kết quả | | | | | | | | | | | | | | | | | | | | | | x |
| | | Trường hợp đã nhập nhưng chưa tìm kiếm | | | | | | | | | | | | | | | | | | | | | | x |
| **Result** | Không hiển thị error message | | | | x | x | | | | | | | | | | | | | | | | | | |
| | Hiển thị error message "Nhập quá kí tự " | | x | x | | | | | | | | | | | | | | | | | | | | |
| | Hiển thị kết quả tương ứng với điều kiện search | | x | | x | | x | | x | | x | | x | | x | | x | | x | | x | | | |
| | Không hiển thị kết quả search | | | | | | | x | | x | | x | | x | | x | | x | | x | | | | |
| | Remove inputed data ở tất cả item | | | | | | | | | | | | | | | | | | | | | x | x | x |
| | Remove kết quả search | | | | | | | | | | | | | | | | | | | | | x | x | x |
| | Hiển thị màn hình giống như before starting search | | | | | | | | | | | | | | | | | | | | | x | x | x |

***Hình 4.3- 1 Decision table of Result search***

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Input | Số record hiển thị | | | | | | | | | | | | | |
| | | 1 record | x | | | | | | x | | | | | |
| | | 10 records | | x | | | | | | | | | | |
| | | 11 records | | | x | | | | | x | x | x | | |
| | | 20 records | | | | x | | | | | | | | |
| | | 21 records | | | | | x | | | | | | | |
| | Trang mà user đang đứng | | | | | | | | | | | | | |
| | | Trang 1 | | | | | | | x | x | | x | x | x |
| | | Trang 2 | | | | | | | | | x | | | |
| | | Trang 3 | | | | | | | | | | | | x |
| Result | | | | | | | | | | | | | | |
| | Số trang sẽ hiển thị | | | | | | | | | | | | | |
| | | Hiển thị 1 trang | x | x | | | | | | | | | | |
| | | Hiển thị 2 trang | | | x | x | | | | | | | | |
| | | Hiển thị 3 trang | | | | | x | x | | | | | | |
| | Trang mà user đang đứng | | | | | | | | | | | | | |
| | | Trang 1 | | | | | | | x | x | x | | x | |
| | | Trang 2 | | | | | | | | | | x | | x |
| | | Trang 3 | | | | | | | | | | | | x |
| | Status của last page button | | | | | | | | | | | | | |
| | | Enable | | | | | | | | x | x | | x | x |
| | | Disable | | | | | | | x | | | x | | x |
| | Status của first page button | | | | | | | | | | | | | |
| | | Enable | | | | | | | | | | x | | x |
| | | Disable | | | | | | | x | x | x | | x | |

***Hình 4.3- 2 Decision table of Phân trang Search***

### 4.3.2. *Test Case:*

Test Case for manual:

| TC-ID | Group check | Item check | Pre-condition | Summary | Step to perform | Input data | Expected result | Status | Who check | Date check | No. Bug |
|---|---|---|---|---|---|---|---|---|---|---|---|
| TC_Search_01 | UI | All items | At Search screen | Verify number of items | Open Search screen , verify behavior all items | | Item numbers are mapped to the design | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_02 | | | | Verify type of all items | Open Search screen , verify type of all items | | Item type is mapped to the design | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_03 | | | | Verify Format of UI | Open Searchscreen , verify format of UI | | Format of UI is mapped to the design | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_04 | | | | Verify Font size, font name, color of items | Open Search screen, verify Font size, font name, color of items | | Font size, font name, color of items are mapped to the design | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_05 | Validate | Name | At Chat screeen | Verify that the name input field accepts up to 100 characters. | 1. Input: 99 characters | Name: N*99 | Don't Show error message:"Nhập quá kí tự" | **FAILED** | VIETBT2 | 20/07/2024 | |
| TC_Search_06 | | | | Verify that the name input field accepts up to 100 characters. | 1. Input: 100 characters | Name: N*100 | Don't Show error message:"Nhập quá kí tự" | **FAILED** | VIETBT2 | 20/07/2024 | |
| TC_Search_07 | | | | Verify that the name input field no accepts up to 100 characters. (101 characters) | 1. Input: 101 characters | Name: N*101 | Show error message:"Nhập quá kí tự" | **FAILED** | VIETBT2 | 20/07/2024 | |
| TC_Search_08 | Business | Clear | At Chat screeen<br>No data | Verify that the clear button isn't appare | | | Don't show anythings | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_09 | | | At Chat screeen<br>Has data input | Verify that when user can click on clear, all result cleared | 1. Click on clear | Name: "A" | all data input cleared | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_10 | | | At Chat screeen<br>Has data input | Verify that the clear button clears all search fields and resets the search results. | 1. Press clear buton | | Show 10 recent user | **FAILED** | VIETBT2 | 20/07/2024 | |
| TC_Search_11 | | Search | Exist user has Ngoc | Verify that the search returns results that match all entered search criteria. | 1. Input "Ngoc" | Name: "Ngoc" | Show all item has Ngoc | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_12 | | | Don't exist user has alfha | Verify that an error message is displayed for invalid input. | 1. input "alfha" | Name"alfha" | Show error massage "Không tìm thấy" | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_13 | | | Exist user has Ngoc Kien | Verify that user can click on result | 1. Input user<br>2. Click on user inputted | Name: "Ngoc Kien" | User can click on result | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_14 | | | Has data | Verify that clicking on the search input field displays up to the 10 most recent searches. | 1. Input 10 users<br>2. Back to search | | Show 10 recent searches | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_15 | | | Has data | Verify that pressing enter in the search input field displays up to the 10 most recent searches. | 1. Input 11 users<br>2. Back to search | | Show 10 recent searches | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_16 | | | Has data | Verify that the search returns all results when user input * | 1. Input * | Name: * | Show all result | **FAILED** | VIETBT2 | 20/07/2024 | |
| TC_Search_17 | | | Has data | Verify behavior in case user input blank | 1. Input blank | Name: | no show | **FAILED** | VIETBT2 | 20/07/2024 | |
| TC_Search_18 | | | Exist user has B | Verify behavior in case user inputed 1 character in a first Name into Name field | 1. Input 1 character in a first Name into Name field<br>2. Enter Search | Name: B | Display all information relate to Name has inputed character | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_19 | | | Exist user has T | Verify behavior in case user inputed 1 character with up into Name field | 1. Input 1 character in a second Name into Name field<br>2. Enter Search | Name: T | Display all information relate to Name has inputed character | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_20 | | | Exist user has Việt | Verify behavior in case user inputed 1 part of Name into Name field | 1. Input 1 part of Name into Name field<br>2. Enter Search | Name: Việt | Display all information relate to Name has inputed word | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_21 | | | Exist user has Việt | Verify behavior in case user input a correct part of word of Name and a incorrect part of Name into Name field | 1. Input a part of Name into Name field<br>2. Enter Search | Name: kalsjlkfjd Việttttttttt | Display all information relate to Name has inputed word / Don't show anything | **FAILED** | VIETBT2 | 20/07/2024 | |
| TC_Search_22 | | | Exist user has Việt | Verify behavior in case user input a correct part of Name and a incorrect part of Name into Name field | 1. Input a part of Name into Name field<br>2. Enter Search | Name: kalsjlkfjd Việt | Display all information relate to Name has inputed word | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_23 | | | Exist user has Bùi Thị Việt | Verify behavior in case user input all part of Name into Name field | 1. Input all part of Name into Name field<br>2. Enter Search | Name: Bùi Thị Việt | Display all information relate to Name has inputed word | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_24 | | | Exist user has Bùi Thị Việt | Verify behavior in case user input a incorrect Name into Name field | 1. Input all part of Name into Name field<br>2. Enter Search | Name: kaglhsdfadsh | Don't show anything | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_25 | | | Exist 1 user have name is Bui Thi Viet | Verify that when has 1 like input data then system has 1 page | 1. Input Name<br>2. Enter | Name: Bui Thi Viet | Show 1 page | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_26 | | | Exist 10 users have fist name is Ngoc | Verify that when has 10 users like input data then system has 1 page | 1. Input Name<br>2. Enter | Name: Ngoc | Show 1 page | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_27 | | | Exist 11 users have fist name is Bui | Verify that when has 10 users like input data then system has 2 pages | 1. Input Name<br>2. Enter | Name: Bui | Show 2 pages | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_28 | | | Exist 20 users have fist name is Nguyen Thi | Verify that when has 20 users like input data then system has 2 pages | 1. Input Name<br>2. Enter | Name: Nguyen Thi | Show 2 pages | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_29 | | | Exist 21 users have fist name is Nguyen | Verify that when has 21 users like input data then system has 3 page | 1. Input Name<br>2. Enter | Name: Nguyen | Show 3 pages | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_30 | Phân trang | First page | At Search screen, user at antother page except first page | Verify behavior in case user click on first button when user at first page | 1. Enter search<br>2. click on first page | | First button disable | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_31 | | | At Search screen, user at antother page except first page | Verify balavior in case user click on first button when user at another page | 1. Enter search<br>2. click on first page | | Information at first page is displayed | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_32 | | Number page | At Search screen, user at last page | Verify balavior in case user click on number page when user at last page | 1. Enter search<br>2. click on number page | | The current page that the user is viewing will have a different status (for example, it can be highlighted or underscored). | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_33 | | | At Search screen, user at firts page | Verify balavior in case user click on number page when user at first page | 1. Enter search<br>2. click on number page | | The current page that the user is viewing will have a different status (for example, it can be highlighted or underscored). | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_34 | | | Exist 31 record<br>At Search screen, user at page 1 | Verify balavior in case user click on first button when user at page 1 | 1. Enter search<br>2. click on page 2 | | The current page that the user is viewing will have a different status (for example, it can be highlighted or underscored) and Last buton is disable , first is enable | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_35 | | | Exist 31 record<br>At Search screen, user at page 2 | Verify balavior in case user click on first button when user at page 2 | 1. Enter search<br>2. click on page 1 | | The current page that the user is viewing will have a different status (for example, it can be highlighted or underscored) and first buton is disable , last is enable | PASSED | VIETBT2 | 20/07/2024 | |
| TC_Search_36 | | | At Search screen, user at last page | Verify balavior in case user click on last page | 1. Enter search<br>2. click on last page | | First button disable | PASSED | VIETBT2 | 20/07/2024 | |

*Hình 4.3- 3 Test Case Manual*

Test Case for Automation:

+ Xác định vị trí Element của các iem test, dưới đây là một số cách bắt locator:

***By CSS ID***: find_element_by_id

***By CSS class name***: find_element_by_class_name

***By name attribute***: find_element_by_name

***By DOM structure or Xpath:*** find_element_by_xpath

***by tagName***: find_element_by_tag_name()

***By link text:*** find_element_by_link_text

***By partial link text:*** find_element_by_partial_link_text

***By HTML tag name:*** find_element_by_tag_name

+ Ở phần thêm pom.xml của maven thực hiện thêm các thư viện:

```
<dependencies>
    <!-- Selenium Java -->
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>3.141.59</version> <!-- Phiên bản tương thích với JDK 8 -->
    </dependency>
    <!-- WebDriverManager -->
    <dependency>
        <groupId>io.github.bonigarcia</groupId>
        <artifactId>webdrivermanager</artifactId>
        <version>5.9.1</version> <!-- Phiên bản mới nhất của WebDriverManager -->
```

```
        </dependency>
        <!-- TestNG -->
        <dependency>
            <groupId>org.testng</groupId>
            <artifactId>testng</artifactId>
            <version>7.3.0</version> <!-- Phiên bản mới nhất của TestNG -->
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.27</version>
        </dependency>
    </dependencies>
```

**Chú thích (Annotation) trong TestNG**

@Test: Đại diện cho một test case.

@BeforeSuite: Chú thích sẽ được chạy trước khi tất cả các kiểm tra trong bộ kiểm thử đã chạy.

@AfterSuite: Chú thích sẽ được chạy sau khi tất cả các kiểm tra trong bộ kiểm thử đã chạy.

@BeforeTest: Chú thích sẽ được chạy trước khi bất kỳ một @Test nào thuộc trong cùng một class được gọi để chạy.

@AfterTest: Chú thích sẽ được chạy sau khi tất cả các @Test thuộc cùng class đã chạy xong.

@BeforeGroups: Chạy trước các group trong các @Test.

Phương pháp này được đảm bảo chạy ngay trước @Test đầu tiên thuộc bất kỳ nhóm nào

được gọi ra.

@AfterGroups: Danh sách các nhóm mà phương pháp cấu hình này sẽ chạy sau.

Phương pháp này được đảm bảo chạy ngay sau khi phương pháp kiểm tra cuối cùng thuộc về bất kỳ nhóm nào được gọi ra.

@BeforeClass: Chú thích sẽ được chạy trước khi @Test đầu tiên trong lớp hiện tại được gọi. (sau @BeforeTest)

@AfterClass: Chú thích sẽ được chạy sau khi tất cả các @Test trong lớp hiện tại đã được chạy hết. (đóng trước @AfterTest)

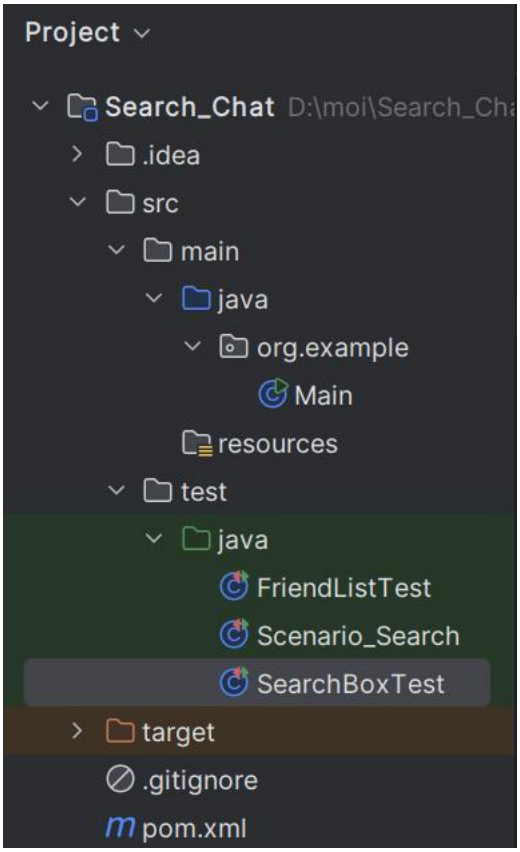
@BeforeMethod: Chú thích sẽ được chạy trước mỗi @Test. (dùng nhiều)

@AfterMethod: Chú thích sẽ được chạy sau mỗi @Test. (dùng nhiều)

Lợi ích của việc sử dụng chú thích

- Nó xác định các phương pháp nó quan tâm bằng cách tìm kiếm các chú thích. Do đó tên phương thức không bị hạn chế trong bất kỳ mẫu hoặc định dạng nào. (cùng một ghi chú có thể sử dụng cho nhiều hàm tên khác nhau)
- Chúng ta có thể truyền các thông số bổ sung vào trong các chú thích.
- Chú thích được đánh làm mốc mạnh mẽ rõ ràng, do đó trình biên dịch sẽ đánh dấu lỗi sai ngay lập tức nếu có.
- Các lớp kiểm tra thuận tiện trong quá trình sắp xếp chạy các test cases (trước, sau, logic)

- Test Case cho Chức năng search tìm kiếm người dùng trong chat được thực hiện như sau :




***Hình 4.3- 4 Tạo thư mục Maven cho test case trong itellij***

+ Ở thư mục test dùng để đưa vào các class của từng tính năng test

+ pom.xml phục vụ cho việc thêm các dependency vào maven, có thể thực hiện lấy từ đường link sau đây: https://mvnrepository.com/artifact/org.testng/testng

- Sau khi setup xong ta thực hiện viết test Case

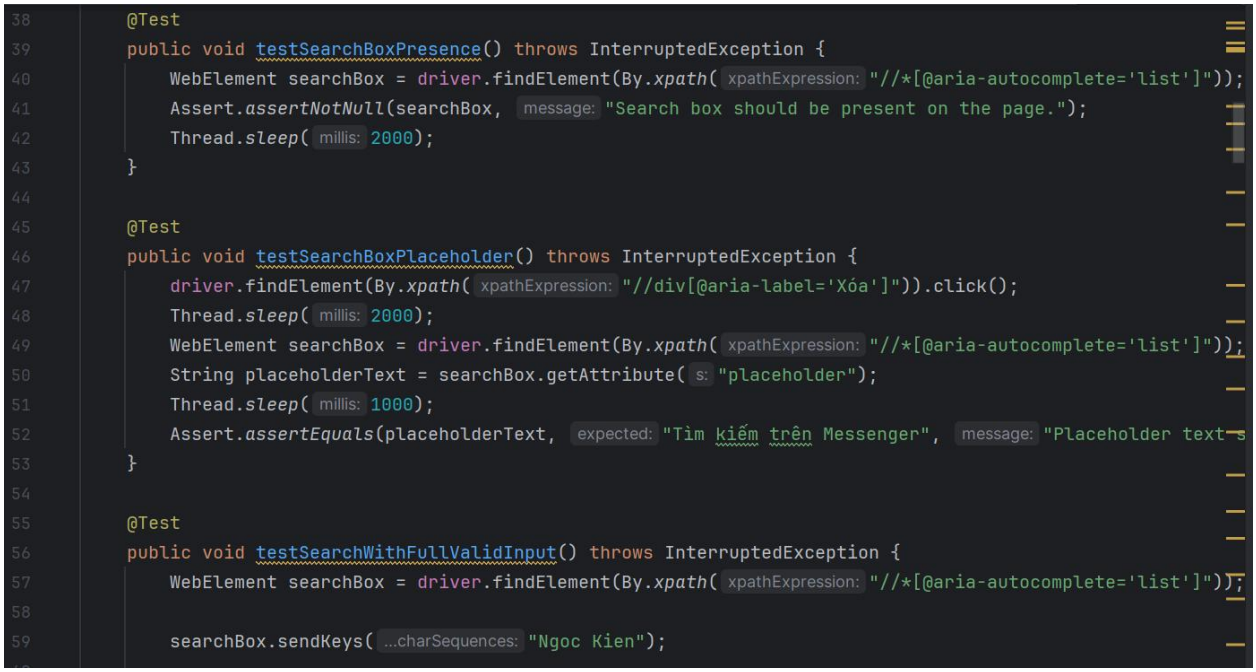
```
@BeforeTest
public void setup() throws InterruptedException {
    WebDriverManager.chromedriver().setup();
    driver = new ChromeDriver();
    Thread.sleep( millis: 2000);
    driver.get("https://https://dev-admin.kelick.io/");
    Thread.sleep( millis: 3000);
    WebElement email = driver.findElement(By.name("email"));
    email.sendKeys( ...charSequences: "admin@gmail.com");
    WebElement pass = driver.findElement(By.name("pass"));
    pass.sendKeys( ...charSequences: "123456");
    driver.findElement(By.name("login")).click();
    Thread.sleep( millis: 3000);
}
```

*Hình 4.3- 5 Tạo Before Test để thực thi trước khi bắt đầu các test phía sau*

```
    @After
    public void tearDown() {
        if (driver != null) {
            driver.quit();
        }
    }
}
```

*Hình 4.3- 6 Tạo After để thực hiện kết thúc sau khi chạy xong test*

Các lệnh cho before và after test được thực thi nhằm mục đích giúp cho test case không bị lặp lại liên tục cho các câu lệnh thường gặp, ví dụ như trường hợp phải thực hiện đăng nhập liên tục để thực hiện test case tiếp theo




*Hình 4.3- 7 Một số test case được viết*

Các phần mục test case sẽ được viết phía dưới before test và phía trên after.

**** Xem full Test case tại thư mục github sau:***

https://github.com/btviet01/TestCaseSearch.git

## 4.4. Environment Setup

- Môi trường Win 11

- Thiết bị: Laptop

- Browser: Chrome

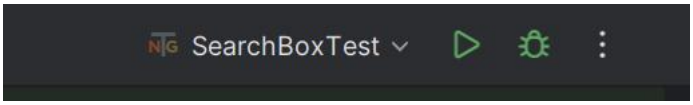
- Phần mềm: Intellij

## 4.5. Test Execution

https://www.browserstack.com/guide/locators-in-selenium

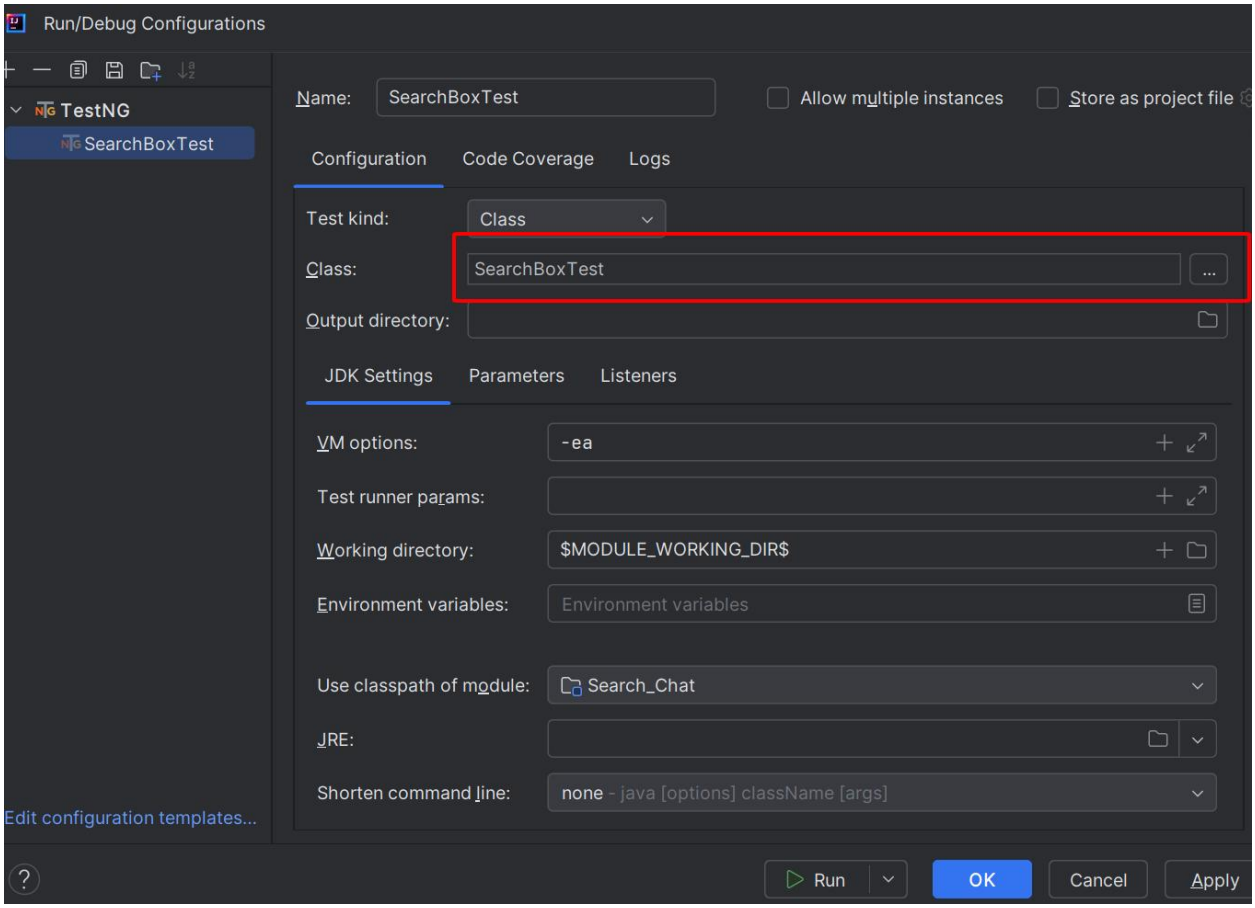
- Thực hiện dùng ngôn ngữ Java và Selenium thực hiện trên phần mềm Ideliij để execute test case.

- Quá trình thực hiện như sau:




*Hình 4.5- 1 Chọn execute*

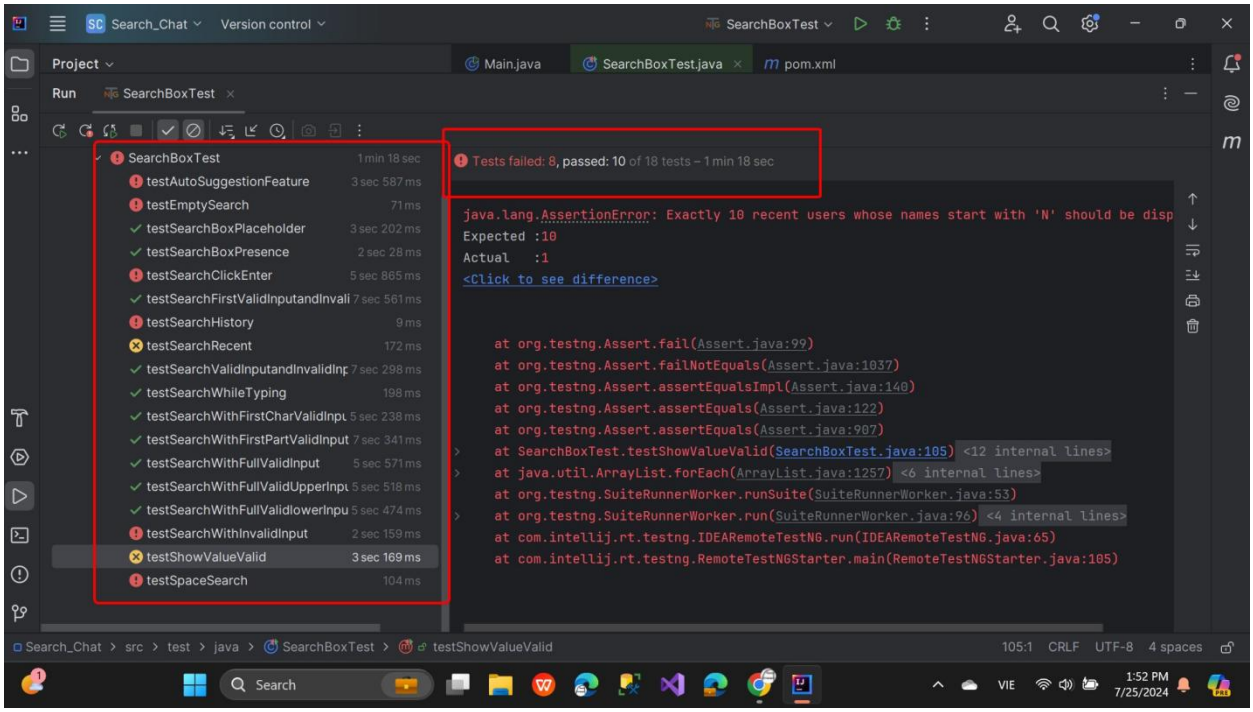
+ Chọn để chạy Test Case




*Hình 4.5- 2 Chọn class của test case mình cần thực thi*

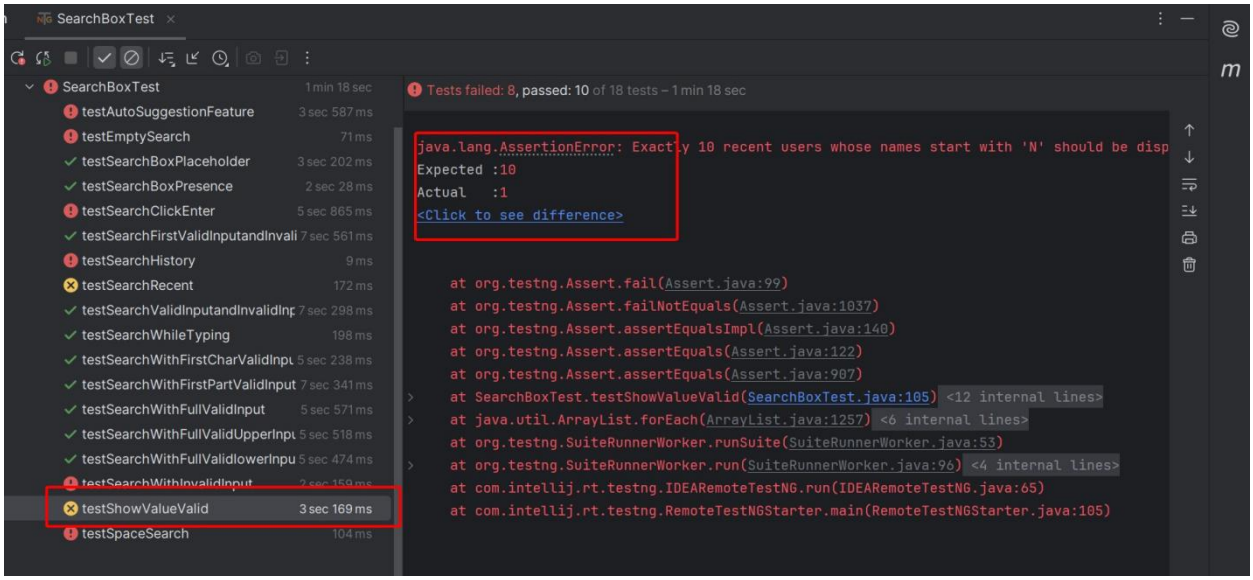
Sau khi hoàn tất thì bắt đầu execute Test Case.

- Lần execute cho các test case thử nghiệm cho ra kết quả như sau:



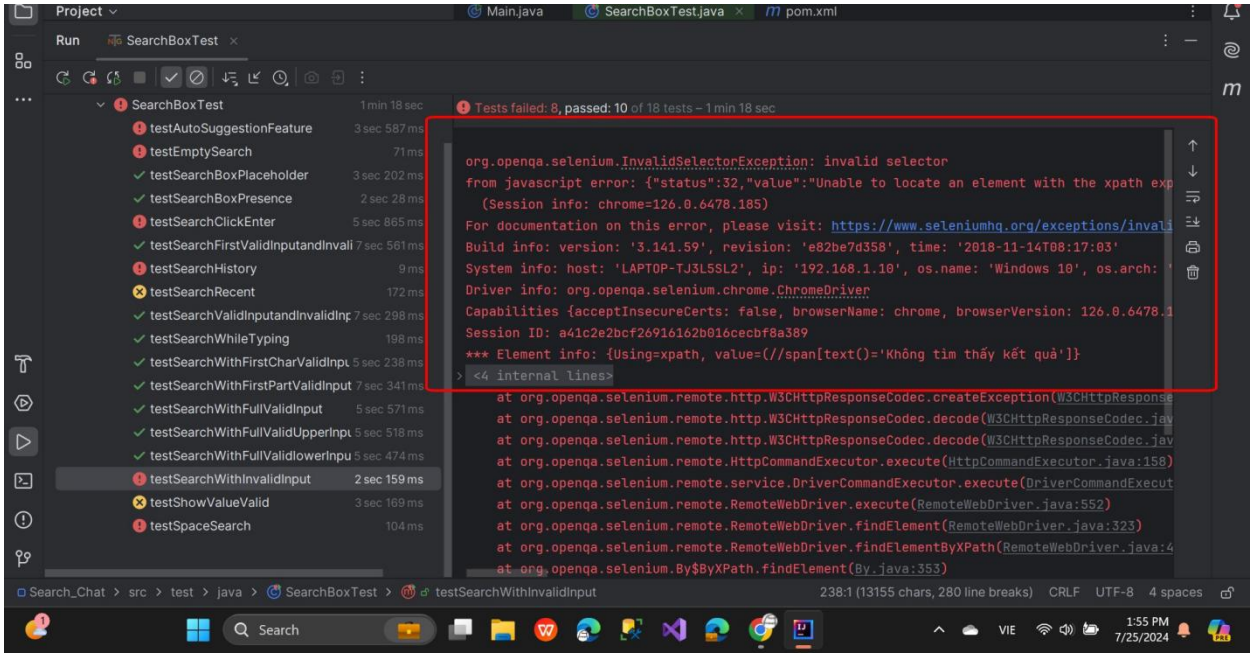

*Hình 4.5- 3 Chạy test case cho quá trình tìm kiếm theo tên*

Ở đây có thể thấy có 8 tests failed và 10 tests passed




*Hình 4.5- 4 Test Case kết quả mong đợi khác với kết quả thực tế (1)*

- Test fail ở selenium có vàng cho thấy kết quả thực tế và kết quả mong đợi không giống nhau




***Hình 4.5- 5 Test Case fail***

- Đối với test case fail này chỉ ra rằng không thể kích hoạt được locate của trang web, lỗi ở đây là do selector không tồn tại ở trang web

## 4.6. Test Cycle Closure

Review and Analysis

- Review test execution results.

- Analyze defects and ensure they are resolved.

* Report :

| Module Name | Date Check | Total | Passed | Failed | Pending |
|---|---|---|---|---|---|
| Search for manual | 20/07/2024 | 37 | 35 | 2 | 0 |
| Search for auto | 21/07/2024 | 25 | 10 | 8 | 7 |
| | | 62 | 45 | 10 | 7 |

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

- **Đạt được:**

- Đã học được các kĩ năng phân tích, giải quyết vấn đề. Các kiến thức về các mô hình Agile, Wanterfall, V-Model, Tạo Test Case, Viết Test Script Automation, học về API Testing, Performance Testing...
- Đã thực hiện viết được test case automation cho website HRick

- **Hạn chế:**

- Chưa thực sự năng nổ và làm quen với môi trường làm việc độc lập, tách biệt giữa đội ngũ Testing và các đội ngũ khác (do IVS là đơn vị kiểm thử riêng lẻ) …
- Chưa thực hiện log bug và retest

- **Hướng phát triển:**

- Học tiếng Anh để trao dồi kĩ năng giao tiếp với các bên của khách hàng.
- Trao dồi kiến thức chuyên sâu về Automation Test, bên cạnh đó bổ sung thêm về thực hành test API.

# TÀI LIỆU THAM KHẢO


[1] "Thống kê mức lương Tester," Glassdoor, Mỹ, 2021.

[2] H. Bui, "Test Process - Quy trình test," p. pirago, 11 8 2021.

[3] "https://www.geeksforgeeks.org/software-testing-techniques/".

[4] "https://www.geeksforgeeks.org/software-testing-life-cycle-stlc/".

[5] "https://katalon.com/resources-center/blog/what-is-automation-testing".

[6] "https://www.jetbrains.com/help/idea/creating-and-running-your-first-java-application.htmlget-started".


# CHECK LIST 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 (30-50 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 | | |