Vai trò Tester trong dự án Selenium Login Test

## **1. Mô tả vai trò**

Tester chịu trách nhiệm đảm bảo chất lượng của form đăng ký người dùng mới (mô phỏng giao diện Facebook tại index.html) trong dự án selenium-login-test. Vai trò của Tester là thiết kế, thực thi, và báo cáo kết quả kiểm thử để xác nhận form hoạt động đúng theo yêu cầu nghiệp vụ và kỹ thuật, đồng thời không có lỗi nghiêm trọng ảnh hưởng đến trải nghiệm người dùng hoặc bảo mật. Tester sử dụng các công cụ tự động hóa (Selenium, TestNG, Allure) để tối ưu hóa quy trình kiểm thử.

## **2. Trách nhiệm chính**

### **2.1. Thiết kế test case**

* **Phân tích yêu cầu**:
  + Đọc BRD, FRS, và user story từ BA để hiểu chức năng form (ví dụ: kiểm tra dữ liệu hợp lệ, hiển thị thông báo).
  + Xác định các trường hợp kiểm thử, bao gồm trường hợp bình thường và biên (edge cases).
* **Viết test case**:
  + Tạo 100 test case (xem Test\_Cases.md), chia thành 4 nhóm:
    - **Functional** (TC01-TC50): Kiểm tra đăng ký với dữ liệu hợp lệ, sai, trống, hoặc không đúng định dạng.
    - **UI** (TC21-TC44, TC56-TC65, TC70, TC93-TC96): Kiểm tra giao diện, placeholder, màu sắc, font chữ.
    - **Performance** (TC36, TC81-TC82): Kiểm tra thời gian tải trang và phản hồi.
    - **Security** (TC83-TC90): Kiểm tra XSS, SQL injection, và độ dài tối đa.
  + Mỗi test case bao gồm: ID, mô tả, input, output mong đợi.
* **Tài liệu hóa test case**:
  + Lưu trong Test\_Cases.md, dễ dàng chia sẻ với BA và developer.
  + Đảm bảo test case bao quát toàn bộ yêu cầu (100% coverage).

### **2.2. Thực thi kiểm thử**

* **Kiểm thử tự động**:
  + Sử dụng Selenium WebDriver và TestNG để viết test script (SeleniumRegistrationTest.java).
  + Chạy 100 test case trên trình duyệt Chrome, sử dụng ChromeDriver tại C:\drivers\chromedriver.exe.
  + Tích hợp Page Object Model (LoginPage.java) để code dễ bảo trì.
* **Kiểm thử thủ công (nếu cần)**:
  + Kiểm tra giao diện trên các thiết bị di động thực tế (TC69).
  + Xác nhận UX (ví dụ: focus màu xanh #1877f2 khi nhập liệu).
* **Quản lý môi trường kiểm thử**:
  + Đảm bảo index.html được đặt trong src/main/resources.
  + Cấu hình Maven và testng.xml để chạy test.

### **2.3. Báo cáo kết quả kiểm thử**

* **Tạo báo cáo Allure**:
  + Sử dụng Allure Reports để ghi lại kết quả pass/fail, thời gian chạy, và screenshot (@Attachment trong SeleniumRegistrationTest.java).
  + Lưu kết quả trong target/allure-results, tạo báo cáo HTML bằng lệnh mvn allure:report.
  + In báo cáo HTML thành PDF để gửi developer (lưu trong reports/allure-report.pdf).
* **Phân tích kết quả**:
  + Báo cáo số lượng test case pass/fail (ví dụ: 90 pass, 10 fail).
  + Cung cấp stack trace và screenshot cho các test case fail.
* **Gửi báo cáo**:
  + Chia sẻ báo cáo HTML/PDF với developer qua email hoặc lưu trên server dự án.
  + Đề xuất lỗi cần sửa (ví dụ: "Thông báo lỗi không hiển thị khi email trống").

### **2.4. Hỗ trợ BA và developer**

* **Làm rõ yêu cầu**:
  + Trao đổi với BA để hiểu rõ các trường hợp đặc biệt (ví dụ: TC85 - XSS injection).
  + Xác nhận output mong đợi với BA (ví dụ: màu thông báo lỗi là rgb(221, 75, 57)).
* **Hỗ trợ sửa lỗi**:
  + Cung cấp chi tiết lỗi (log, screenshot) cho developer.
  + Kiểm tra lại sau khi lỗi được sửa (regression testing).
* **Đề xuất cải tiến**:
  + Gợi ý cải thiện UX (ví dụ: thêm tooltip cho trường mật khẩu).
  + Đề xuất kiểm tra bảo mật bổ sung (ví dụ: ngăn SQL injection).

### **2.5. Quản lý chất lượng**

* **Đảm bảo coverage**: Kiểm tra 100% yêu cầu chức năng và phi chức năng.
* **Kiểm tra bảo mật**: Phát hiện lỗ hổng như XSS (TC85) hoặc nhập liệu không an toàn (TC83).
* **Kiểm tra hiệu suất**: Đảm bảo thời gian tải trang dưới 2 giây (TC36) và phản hồi nút dưới 500ms (TC81).

## **3. Kỹ năng cần thiết**

* **Kỹ năng kiểm thử**:
  + Hiểu về kiểm thử tự động và thủ công.
  + Biết cách viết test case bao quát các trường hợp biên.
* **Lập trình**:
  + Thành thạo Java để viết test script Selenium.
  + Hiểu về Page Object Model và framework TestNG.
* **Công cụ**:
  + **Kiểm thử**: Selenium WebDriver, TestNG, Allure Reports.
  + **Quản lý dự án**: Jira, Confluence.
  + **Build**: Maven.
  + **Trình duyệt**: ChromeDriver, hiểu cách cấu hình.
* **Phân tích**:
  + Đọc hiểu BRD, FRS, và wireframe từ BA.
  + Xác định lỗi và nguyên nhân gốc rễ (root cause).
* **Báo cáo**:
  + Viết báo cáo rõ ràng, kèm bằng chứng (screenshot, log).
  + Trình bày kết quả kiểm thử cho BA và developer.
* **Kiến thức bảo mật**:
  + Hiểu về XSS, SQL injection, và các lỗ hổng web cơ bản.

## **4. Quy trình làm việc**

1. **Phân tích yêu cầu**:
   * Đọc tài liệu từ BA (BRD, FRS, wireframe).
   * Thảo luận với BA để làm rõ yêu cầu.
2. **Thiết kế test case**:
   * Viết 100 test case trong Test\_Cases.md.
   * Review test case với BA để đảm bảo coverage.
3. **Phát triển test script**:
   * Viết code trong SeleniumRegistrationTest.java.
   * Sử dụng LoginPage.java để quản lý phần tử.
4. **Thực thi kiểm thử**:
   * Chạy test bằng mvn test hoặc testng.xml.
   * Ghi lại kết quả và screenshot.
5. **Báo cáo và sửa lỗi**:
   * Tạo báo cáo Allure (mvn allure:report).
   * Gửi báo cáo cho developer, hỗ trợ sửa lỗi.
6. **Regression testing**:
   * Chạy lại test sau khi lỗi được sửa.
   * Cập nhật báo cáo nếu cần.

## **5. Đóng góp trong dự án**

* **Đảm bảo chất lượng**:
  + Kiểm tra toàn diện form đăng ký với 100 test case, phát hiện lỗi trước khi release.
  + Đảm bảo form đáp ứng yêu cầu chức năng, giao diện, hiệu suất, và bảo mật.
* **Tự động hóa**:
  + Tiết kiệm thời gian với test script Selenium, tích hợp Allure.
  + Giảm công sức kiểm thử thủ công, tăng độ chính xác.
* **Báo cáo chi tiết**:
  + Báo cáo Allure cung cấp thông tin pass/fail, screenshot, và stack trace.
  + Giúp developer nhanh chóng xác định và sửa lỗi.
* **Cải thiện UX và bảo mật**:
  + Phát hiện lỗi UI (ví dụ: placeholder sai, TC21-TC24).
  + Ngăn lỗ hổng bảo mật như XSS (TC85) hoặc nhập liệu không an toàn (TC83).
* **Hỗ trợ team**:
  + Làm rõ yêu cầu với BA, giảm hiểu lầm.
  + Cung cấp bằng chứng lỗi cho developer, tăng hiệu quả sửa lỗi.

## **6. Kết quả mong đợi**

* **Test case**:
  + 100 test case được thiết kế và thực thi, bao quát tất cả yêu cầu.
  + Tài liệu Test\_Cases.md đầy đủ, dễ hiểu.
* **Kết quả kiểm thử**:
  + Báo cáo Allure (HTML/PDF) chi tiết số lượng pass/fail, screenshot, và lỗi.
  + Form đăng ký không có lỗi nghiêm trọng về chức năng, UI, hoặc bảo mật.
* **Hỗ trợ sửa lỗi**:
  + Developer nhận báo cáo rõ ràng, sửa lỗi nhanh chóng.
  + Regression testing đảm bảo không phát sinh lỗi mới.
* **Chất lượng sản phẩm**:
  + Form đăng ký hoạt động đúng yêu cầu, giao diện giống Facebook.
  + Tăng sự hài lòng của khách hàng và người dùng cuối.

### **3. File danh sách 100 test case (Hoàn thiện)**

# Danh sách 100 Test Case cho Form Đăng ký (Selenium Login Test)

Danh sách này mô tả 100 test case kiểm tra form đăng ký người dùng mới, mô phỏng giao diện Facebook (index.html). Mỗi test case bao gồm ID, mô tả, input, và output mong đợi. Test case được chia thành 4 nhóm: Functional, UI, Performance, Security.