# **Day 10: Bài tập tổng hợp về Công cụ Testing và Ôn tập**

## **Phần 1: Lý thuyết**

### **1. Công cụ hỗ trợ kiểm thử và Automation**

**a. Công cụ hỗ trợ kiểm thử là gì? Giải thích cách automation giúp tiết kiệm thời gian trong kiểm thử phần mềm.**

Công cụ hỗ trợ kiểm thử là các phần mềm hoặc chương trình được sử dụng để hỗ trợ quá trình kiểm thử, bao gồm việc lập kế hoạch, thực thi, theo dõi và báo cáo kết quả kiểm thử. Chúng giúp đơn giản hóa và tối ưu hóa quy trình kiểm thử, giảm thiểu công việc thủ công.

Automation giúp tiết kiệm thời gian bằng cách:

* **Tự động hóa các tác vụ lặp lại**: Thay vì thực hiện thủ công các kịch bản kiểm thử giống nhau (ví dụ: kiểm tra đăng nhập), automation chạy nhanh chóng và chính xác.
* **Thực hiện kiểm thử hàng loạt**: Automation cho phép chạy nhiều test case đồng thời trên nhiều môi trường, giảm thời gian kiểm thử.
* **Tái sử dụng kịch bản**: Một kịch bản kiểm thử tự động có thể được sử dụng lại nhiều lần mà không cần viết lại, tiết kiệm thời gian cho các chu kỳ kiểm thử tiếp theo.
* **Phát hiện lỗi nhanh chóng**: Automation phát hiện lỗi ngay lập tức và báo cáo chi tiết, giảm thời gian phân tích.

**b. Ví dụ cụ thể về tình huống automation mang lại lợi ích rõ rệt.**

Trong kiểm thử regression cho một ứng dụng web như trang thương mại điện tử, mỗi khi có bản cập nhật mới (thêm tính năng hoặc sửa lỗi), cần kiểm tra lại toàn bộ các tính năng cũ (đăng nhập, tìm kiếm, giỏ hàng). Nếu thực hiện thủ công, việc này mất nhiều thời gian và dễ xảy ra lỗi do con người. Automation (sử dụng Selenium) cho phép chạy toàn bộ test suite trong vài phút, đảm bảo các tính năng cũ vẫn hoạt động đúng mà không cần can thiệp thủ công, tiết kiệm hàng giờ làm việc.

### **2. Rủi ro của Automation**

**a. Liệt kê ít nhất 3 rủi ro của automation testing và giải thích cách chúng ảnh hưởng.**

1. **Chi phí bảo trì cao**: Kịch bản kiểm thử tự động cần được cập nhật khi giao diện hoặc logic ứng dụng thay đổi (ví dụ: thay đổi tên nút trên giao diện web). Điều này làm tăng chi phí bảo trì, đặc biệt trong các dự án thay đổi thường xuyên.
2. **Lỗi kịch bản kiểm thử**: Kịch bản kiểm thử có thể được viết sai hoặc không bao quát hết các trường hợp, dẫn đến kết quả kiểm thử không chính xác (false positive/negative). Điều này làm giảm độ tin cậy của kiểm thử.
3. **Chi phí đầu tư ban đầu cao**: Xây dựng framework kiểm thử tự động và đào tạo đội ngũ sử dụng công cụ (như Selenium) đòi hỏi thời gian và chi phí lớn, ảnh hưởng đến ngân sách dự án.

**b. Đề xuất ít nhất 2 cách để giảm thiểu rủi ro.**

1. **Sử dụng thiết kế mô-đun**: Tạo framework kiểm thử với các module tái sử dụng (ví dụ: Page Object Model trong Selenium) để dễ dàng cập nhật khi ứng dụng thay đổi, giảm chi phí bảo trì.
2. **Kiểm tra và bảo trì định kỳ kịch bản**: Thực hiện đánh giá kịch bản kiểm thử thường xuyên để phát hiện và sửa lỗi, đồng thời đảm bảo kịch bản bao quát các trường hợp cần thiết.

### **3. Công cụ Automation phổ biến**

**a. Liệt kê 3 công cụ automation testing phổ biến và mô tả ứng dụng thực tế.**

1. **Selenium**: Công cụ mã nguồn mở, hỗ trợ kiểm thử giao diện web trên nhiều trình duyệt (Chrome, Firefox). Ứng dụng: Kiểm thử giao diện người dùng của ứng dụng web như tìm kiếm, đăng nhập, giỏ hàng.
2. **JUnit**: Framework kiểm thử đơn vị cho các ứng dụng Java. Ứng dụng: Kiểm thử các hàm hoặc lớp trong mã nguồn Java để đảm bảo logic hoạt động đúng.
3. **TestNG**: Framework kiểm thử nâng cao, hỗ trợ kiểm thử đơn vị, tích hợp và hệ thống. Ứng dụng: Chạy các test case song song hoặc kiểm thử tích hợp trong các dự án phức tạp.

**b. Lý do Selenium phù hợp cho kiểm thử giao diện web.**

Selenium phù hợp cho kiểm thử giao diện web vì:

* Hỗ trợ nhiều trình duyệt và nền tảng (Windows, macOS, Linux).
* Tích hợp với các ngôn ngữ lập trình phổ biến (Java, Python, C#).
* Cho phép ghi lại và phát lại hành động người dùng (Selenium IDE) hoặc viết script phức tạp (Selenium WebDriver).
* Ví dụ: Kiểm thử tính năng “tìm kiếm sản phẩm” trên web bằng cách tự động nhập từ khóa, kiểm tra kết quả hiển thị và xác minh giao diện.

### **4. Ôn tập kiến thức Testing**

**a. Tóm tắt các khái niệm chính trong testing và vai trò.**

1. **Static Testing**: Kiểm tra tài liệu, mã nguồn hoặc yêu cầu mà không thực thi chương trình. Vai trò: Phát hiện lỗi sớm (ví dụ: lỗi cú pháp trong mã nguồn), giảm chi phí sửa lỗi.
2. **Black-box Testing**: Kiểm thử dựa trên yêu cầu mà không cần biết cấu trúc mã nguồn. Vai trò: Đảm bảo ứng dụng đáp ứng yêu cầu người dùng (ví dụ: kiểm tra giao diện).
3. **Regression Testing**: Kiểm thử để đảm bảo các tính năng cũ vẫn hoạt động sau khi có thay đổi. Vai trò: Ngăn chặn lỗi tái phát sinh khi cập nhật phần mềm.
4. **Test Plan**: Tài liệu mô tả mục tiêu, phạm vi, tài nguyên và lịch trình kiểm thử. Vai trò: Định hướng và tổ chức quá trình kiểm thử, đảm bảo kiểm thử hiệu quả.

**b. Tại sao ôn tập và làm bài kiểm tra thử quan trọng?**

Ôn tập và làm bài kiểm tra thử giúp:

* Củng cố kiến thức về các khái niệm testing (black-box, regression, v.v.).
* Làm quen với định dạng câu hỏi thi, cải thiện kỹ năng trả lời chính xác và nhanh chóng.
* Phát hiện lỗ hổng kiến thức để cải thiện trước kỳ thi.

## **Phần 2: Bài tập thực hành**

### **1. Phân tích lợi ích và rủi ro của Automation**

**a. Đề xuất kịch bản kiểm thử và cách automation tiết kiệm thời gian.**

**Kịch bản kiểm thử**: Kiểm thử tính năng “tìm kiếm sản phẩm” với từ khóa không hợp lệ (ví dụ: ký tự đặc biệt như “@#$%”).

* **Cách thực hiện thủ công**: Nhập từ khóa “@#$%” vào ô tìm kiếm, nhấn nút tìm kiếm, kiểm tra thông báo lỗi hiển thị đúng (ví dụ: “Không tìm thấy sản phẩm”). Lặp lại cho nhiều trường hợp (rỗng, ký tự đặc biệt, số).
* **Cách automation tiết kiệm thời gian**: Sử dụng Selenium để viết script tự động nhập nhiều từ khóa không hợp lệ, kiểm tra thông báo lỗi và lưu kết quả. Script có thể chạy trong vài giây, thay vì hàng phút nếu làm thủ công, đặc biệt khi kiểm thử trên nhiều trình duyệt.

**b. Rủi ro và cách khắc phục.**

1. **Rủi ro**: Kịch bản kiểm thử không bao quát hết các trường hợp (ví dụ: bỏ sót từ khóa rỗng).
   * **Khắc phục**: Thiết kế test case dựa trên kỹ thuật equivalence partitioning và boundary value analysis để đảm bảo bao quát.
2. **Rủi ro**: Thay đổi giao diện (ví dụ: đổi ID của ô tìm kiếm) làm hỏng script.
   * **Khắc phục**: Sử dụng Page Object Model để tách biệt logic kiểm thử và yếu tố giao diện, dễ dàng cập nhật khi giao diện thay đổi.

### **2. Ứng dụng công cụ Automation**

**a. Mô tả cách sử dụng Selenium để kiểm thử tính năng “tìm kiếm sản phẩm”.**

1. **Viết script**: Sử dụng Selenium WebDriver (Python) để xác định ô tìm kiếm (bằng ID/class), nhập từ khóa, nhấn nút tìm kiếm và kiểm tra kết quả.
2. **Chạy test**: Cấu hình môi trường (trình duyệt Chrome, Firefox) và chạy script bằng framework như Pytest để kiểm tra nhiều test case.
3. **Báo cáo kết quả**: Sử dụng công cụ như Allure hoặc HTML reports để ghi lại kết quả kiểm thử (pass/fail) và lỗi chi tiết.

**b. Thiết kế 3 test case tự động.**

1. **Test Case 1: Tìm kiếm với từ khóa hợp lệ**
   * **Mô tả**: Nhập từ khóa “áo sơ mi” và kiểm tra kết quả hiển thị.
   * **Dữ liệu đầu vào**: Từ khóa = “áo sơ mi”.
   * **Kết quả mong đợi**: Danh sách sản phẩm chứa “áo sơ mi” được hiển thị.
2. **Test Case 2: Tìm kiếm với từ khóa không hợp lệ**
   * **Mô tả**: Nhập từ khóa “@#$%” và kiểm tra thông báo lỗi.
   * **Dữ liệu đầu vào**: Từ khóa = “@#$%”.
   * **Kết quả mong đợi**: Hiển thị thông báo “Không tìm thấy sản phẩm”.
3. **Test Case 3: Tìm kiếm với ô trống**
   * **Mô tả**: Nhấn nút tìm kiếm khi ô tìm kiếm rỗng.
   * **Dữ liệu đầu vào**: Từ khóa = “” (rỗng).
   * **Kết quả mong đợi**: Hiển thị thông báo “Vui lòng nhập từ khóa tìm kiếm”.

### **3. Làm bài kiểm tra thử**

**Câu 1: Phân biệt black-box testing và white-box testing. Đưa ra ví dụ.**

* **Black-box Testing**: Kiểm thử dựa trên yêu cầu, không cần biết mã nguồn. Ví dụ: Kiểm tra ô tìm kiếm hiển thị đúng sản phẩm khi nhập “áo sơ mi”.
* **White-box Testing**: Kiểm thử dựa trên cấu trúc mã nguồn. Ví dụ: Kiểm tra hàm tìm kiếm trong mã nguồn xử lý đúng từ khóa “áo sơ mi”.

**Câu 2: Mục đích của regression testing và khi nào thực hiện?**

* **Mục đích**: Đảm bảo các tính năng cũ vẫn hoạt động đúng sau khi có thay đổi (thêm tính năng, sửa lỗi).
* **Khi thực hiện**: Sau mỗi lần cập nhật phần mềm hoặc trước khi phát hành phiên bản mới.

**Câu 3: Tiêu chí vào/ra trong test plan là gì? Ví dụ.**

* **Tiêu chí vào (entry criteria)**: Điều kiện cần thỏa mãn để bắt đầu kiểm thử. Ví dụ: Môi trường kiểm thử đã được thiết lập, tài liệu yêu cầu hoàn thiện.
* **Tiêu chí ra (exit criteria)**: Điều kiện để kết thúc kiểm thử. Ví dụ: 95% test case pass, không còn lỗi nghiêm trọng.

**Câu 4: Khái niệm equivalence partitioning và cách áp dụng.**

* **Equivalence Partitioning**: Chia dữ liệu đầu vào thành các nhóm tương đương, mỗi nhóm cho kết quả tương tự.
* **Áp dụng**: Với trường nhập số lượng sản phẩm (1-100):
  + Nhóm hợp lệ: 1-100 (kiểm tra số 50).
  + Nhóm không hợp lệ: <1 (kiểm tra -1), >100 (kiểm tra 101).

**Câu 5: Tại sao static testing quan trọng? Ví dụ lỗi.**

* **Tầm quan trọng**: Phát hiện lỗi sớm, giảm chi phí sửa lỗi trước khi thực thi.
* **Ví dụ lỗi**: Lỗi cú pháp trong tài liệu yêu cầu (thiếu trường hợp xử lý từ khóa rỗng trong tính năng tìm kiếm).

## **Phần 3: Soạn Test Plan mẫu**

**Test Plan cho tính năng “Tìm kiếm sản phẩm”**

### **1. Mục tiêu kiểm thử**

* Đảm bảo tính năng “tìm kiếm sản phẩm” hoạt động đúng theo yêu cầu, bao gồm tìm kiếm hợp lệ, không hợp lệ và các trường hợp đặc biệt.
* Đảm bảo tính ổn định và hiệu suất trên các trình duyệt (Chrome, Firefox).

### **2. Phạm vi kiểm thử**

* Kiểm thử giao diện người dùng (ô tìm kiếm, nút tìm kiếm, kết quả hiển thị).
* Kiểm thử chức năng (tìm kiếm với từ khóa hợp lệ, không hợp lệ, rỗng).
* Không bao gồm kiểm thử hiệu năng hoặc bảo mật.

### **3. Tài nguyên**

* **Nhân sự**: 2 QA Engineers, 1 Automation Engineer.
* **Công cụ**: Selenium WebDriver (Python), Pytest, Allure (báo cáo).
* **Môi trường**: Chrome, Firefox, Windows 10.

### **4. Lịch trình kiểm thử**

* Tuần 1: Phân tích yêu cầu, viết test case.
* Tuần 2: Phát triển script automation, chạy kiểm thử.
* Tuần 3: Báo cáo kết quả, sửa lỗi (nếu có).

### **5. Tiêu chí vào/ra**

* **Tiêu chí vào**: Yêu cầu tính năng hoàn thiện, môi trường kiểm thử sẵn sàng.
* **Tiêu chí ra**: 95% test case pass, không còn lỗi nghiêm trọng.

### **6. Các chỉ số theo dõi tiến độ**

* Tỷ lệ test case pass/fail.
* Số lượng lỗi phát hiện và sửa chữa.
* Thời gian hoàn thành mỗi giai đoạn kiểm thử.