**Bài tập tổng hợp về Công cụ Testing và Ôn tập**

**Phần 1: Lý thuyết**

**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 được sử dụng để hỗ trợ quá trình kiểm thử, bao gồm lập kế hoạch, thiết kế test case, thực thi kiểm thử, và báo cáo kết quả. Ví dụ: Selenium, JUnit, TestNG.

Automation giúp tiết kiệm thời gian bằng cách:

* **Tự động hóa các tác vụ lặp lại**: Chạy test case nhiều lần mà không cần can thiệp thủ công.
* **Tăng tốc độ thực thi**: Thực hiện hàng trăm test case trong thời gian ngắn, đặc biệt với các ứng dụng lớn.
* **Giảm lỗi con người**: Loại bỏ sai sót do kiểm thử thủ công.
* **Hỗ trợ kiểm thử liên tục**: Dễ dàng tích hợp vào quy trình CI/CD để kiểm tra thường xuyên.

**b. Ví dụ cụ thể về lợi ích của automation**

Trong kiểm thử regression cho một ứng dụng web (ví dụ: website thương mại điện tử), automation giúp:

* Tự động chạy hàng trăm test case để kiểm tra các chức năng cũ (như đăng nhập, tìm kiếm, giỏ hàng) sau mỗi lần cập nhật mã nguồn.
* Tiết kiệm thời gian so với kiểm thử thủ công, đặc biệt khi ứng dụng có nhiều phiên bản phát hành liên tục.
* Ví dụ: Selenium có thể tự động kiểm tra chức năng “thêm vào giỏ hàng” trên nhiều trình duyệt, phát hiện lỗi nhanh chóng mà không cần kiểm thử thủ công lặp đi lặp lại.

**Rủi ro của Automation**

**a. Liệt kê ít nhất 3 rủi ro và ảnh hưởng**

1. **Chi phí bảo trì cao**:
   * **Ảnh hưởng**: Script kiểm thử cần cập nhật thường xuyên khi giao diện hoặc chức năng ứng dụng thay đổi, tốn thời gian và công sức.
2. **Lỗi kịch bản kiểm thử**:
   * **Ảnh hưởng**: Script không chính xác có thể dẫn đến false positive/negative, làm giảm độ tin cậy của kết quả kiểm thử.
3. **Thiếu kiểm thử sáng tạo**:
   * **Ảnh hưởng**: Automation không thay thế được kiểm thử thủ công trong các trường hợp cần tư duy sáng tạo hoặc kiểm tra trải nghiệm người dùng.

**b. Cách giảm thiểu rủi ro**

1. **Sử dụng framework kiểm thử linh hoạt**:
   * Áp dụng các framework như Page Object Model (POM) trong Selenium để giảm chi phí bảo trì khi giao diện thay đổi.
2. **Kiểm tra và tối ưu hóa script thường xuyên**:
   * Thực hiện code review và chạy thử script để phát hiện lỗi sớm, đảm bảo script luôn chính xác và đáng tin cậy.

**Công cụ Automation phổ biến**

**a. Liệt kê 3 công cụ và ứng dụng thực tế**

1. **Selenium**:
   * **Ứng dụng**: Kiểm thử giao diện web trên nhiều trình duyệt (Chrome, Firefox). Phù hợp cho các dự án web lớn với yêu cầu kiểm thử đa nền tảng.
2. **JUnit**:
   * **Ứng dụng**: Kiểm thử đơn vị (unit testing) cho các ứng dụng Java. Thường được dùng trong các dự án phát triển phần mềm theo hướng TDD (Test-Driven Development).
3. **TestNG**:
   * **Ứng dụng**: Kiểm thử tự động cho các ứng dụng Java, hỗ trợ kiểm thử song song và quản lý test case phức tạp, phù hợp cho các dự án lớn.

**b. Lý do chọn công cụ (Selenium)**

Selenium phù hợp cho kiểm thử giao diện web vì:

* Hỗ trợ nhiều trình duyệt và nền tảng.
* Có cộng đồng lớn, tài liệu phong phú, và tích hợp dễ dàng với các công cụ CI/CD như Jenkins.
* Linh hoạt trong việc viết script bằng nhiều ngôn ngữ (Java, Python, C#), cho phép kiểm thử các chức năng như nhập liệu, nhấp chuột, và xác minh nội dung trang web.

**Ôn tập kiến thức Testing**

**a. Tóm tắt các khái niệm chính**

1. **Static Testing**: Kiểm tra tài liệu, mã nguồn mà không thực thi chương trình (ví dụ: code review, phân tích yêu cầu). Vai trò: Phát hiện lỗi sớm, giảm chi phí sửa lỗi.
2. **Black-box Testing**: Kiểm thử dựa trên yêu cầu mà không cần biết mã nguồn. Vai trò: Đảm bảo ứng dụng đáp ứng đúng chức năng người dùng.
3. **Regression Testing**: Kiểm thử để đảm bảo các chức năng cũ vẫn hoạt động sau khi có thay đổi. Vai trò: Ngăn chặn lỗi tái xuất hiện trong các phiên bản mới.
4. **Test Plan**: Tài liệu mô tả mục tiêu, phạm vi, tài nguyên, lịch trình, và tiêu chí kiểm thử. Vai trò: Hướng dẫn và tổ chức quá trình kiểm thử.

**b. Tầm quan trọng của ôn tập và làm bài kiểm tra thử**

* **Ôn tập**: Giúp củng cố kiến thức về các khái niệm testing, quy trình, và công cụ.
* **Làm bài kiểm tra thử**: Giúp làm quen với định dạng câu hỏi, cải thiện kỹ năng phân tích, và tăng tự tin trước kỳ thi thật.

**Phần 2: Bài tập thực hành**

**Phân tích lợi ích và rủi ro của Automation**

**a. Kịch bản kiểm thử và lợi ích của automation**

**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 automation áp dụng**:
  + Sử dụng Selenium để viết script tự động nhập từ khóa không hợp lệ vào ô tìm kiếm, nhấp nút tìm kiếm, và kiểm tra thông báo lỗi.
  + Script có thể chạy trên nhiều trình duyệt và lặp lại với nhiều từ khóa khác nhau, tiết kiệm thời gian so với nhập tay từng trường hợp.
  + **Lợi ích**: Tiết kiệm thời gian khi kiểm tra nhiều trường hợp tìm kiếm, đảm bảo tính nhất quán và độ chính xác.

**b. Rủi ro và cách khắc phục**

1. **Rủi ro: Script không xử lý được thay đổi giao diện** (ví dụ: ô tìm kiếm đổi ID hoặc class).
   * **Khắc phục**: Sử dụng Page Object Model để tách biệt logic kiểm thử và phần tử giao diện, giảm công sức cập nhật script.
2. **Rủi ro: Kết quả false positive** (ví dụ: script báo pass dù thông báo lỗi không hiển thị đúng).
   * **Khắc phục**: Thêm các bước kiểm tra bổ sung (assertions) để xác minh nội dung thông báo lỗi.

**Ứng dụng công cụ Automation**

**a. Cách sử dụng Selenium để kiểm thử “tìm kiếm sản phẩm”**

* **Bước 1: Viết script**:
  + Sử dụng Selenium WebDriver để xác định ô tìm kiếm, nhập từ khóa, và nhấp nút tìm kiếm.
  + Kiểm tra kết quả bằng assertions (ví dụ: kiểm tra thông báo lỗi hoặc danh sách sản phẩm).
* **Bước 2: Chạy test**:
  + Chạy script trên các trình duyệt như Chrome, Firefox bằng Selenium Grid hoặc local driver.
* **Bước 3: Báo cáo kết quả**:
  + Sử dụng TestNG để tạo báo cáo chi tiết, bao gồm số lượng test case pass/fail và lỗi cụ thể.

**b. Thiết kế 3 test case tự động**

1. **Test Case 1: Tìm kiếm với từ khóa hợp lệ**
   * **Mô tả**: Kiểm tra chức năng tìm kiếm với từ khóa hợp lệ (ví dụ: “laptop”).
   * **Dữ liệu đầu vào**: Từ khóa = “laptop”.
   * **Kết quả mong đợi**: Hiển thị danh sách sản phẩm liên quan đến “laptop”.
2. **Test Case 2: Tìm kiếm với từ khóa không hợp lệ**
   * **Mô tả**: Kiểm tra xử lý từ khóa không hợp lệ (ví dụ: “@#$%”).
   * **Dữ liệu đầu vào**: Từ khóa = “@#$%”.
   * **Kết quả mong đợi**: Hiển thị thông báo “Không tìm thấy sản phẩm”.
3. **Test Case 3: Tìm kiếm với từ khóa rỗng**
   * **Mô tả**: Kiểm tra hành vi khi để trống ô tìm kiếm và nhấn nút tìm kiếm.
   * **Dữ liệu đầu vào**: Từ khóa = “” (rỗng).
   * **Kết quả mong đợi**: Hiển thị thông báo “Vui lòng nhập từ khóa tìm kiếm”.

**Làm bài kiểm tra thử**

**Câu 1: Phân biệt black-box testing và white-box testing**

* **Black-box Testing**: Kiểm thử dựa trên yêu cầu, không cần biết mã nguồn. Ví dụ: Kiểm tra chức năng đăng nhập bằng cách nhập email/mật khẩu và kiểm tra kết quả.
* **White-box Testing**: Kiểm thử dựa trên mã nguồn, biết cấu trúc bên trong. Ví dụ: Kiểm tra hàm xử lý đăng nhập để đảm bảo mọi nhánh mã được thực thi.

**Câu 2: Mục đích của regression testing**

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

**Câu 3: Tiêu chí vào/ra trong test plan**

* **Tiêu chí vào**: Điều kiện để bắt đầu kiểm thử (ví dụ: yêu cầu phần mềm đã được phê duyệt).
* **Tiêu chí ra**: Điều kiện để kết thúc kiểm thử (ví dụ: 95% test case pass, không có lỗi nghiêm trọng).
* **Ví dụ**: Tiêu chí ra: Tất cả test case quan trọng pass, không có lỗi cấp cao còn lại.

**Câu 4: Khái niệm equivalence partitioning**

* **Khái niệm**: Chia dữ liệu đầu vào thành các nhóm tương đương, mỗi nhóm đại diện cho một hành vi của hệ thống.
* **Áp dụng**: Với trường nhập số lượng sản phẩm (1-100):
  + Nhóm 1: Số hợp lệ (1-100) → Mong đợi: Chấp nhận.
  + Nhóm 2: Số không hợp lệ (<1 hoặc >100) → Mong đợi: Thông báo lỗi.
  + Nhóm 3: Không phải số (ví dụ: “abc”) → Mong đợi: Thông báo lỗi.

**Câu 5: Tầm quan trọng của static testing**

* **Tầm quan trọng**: Phát hiện lỗi sớm trong tài liệu hoặc mã nguồn, giảm chi phí sửa lỗi so với giai đoạn sau.
* **Ví dụ lỗi**: Trong tài liệu yêu cầu, mô tả chức năng “tìm kiếm sản phẩm” thiếu trường hợp xử lý từ khóa rỗng, được phát hiện qua code review.

**Phần 3: Soạn Test Plan mẫu**

% Setting up the document class  
\documentclass[a4paper,12pt]{article}

% Including necessary packages  
\usepackage[utf8]{vietnam}

% Configuring the page layout  
\usepackage{geometry}  
\geometry{a4paper, margin=1in}

% Setting up fonts  
\usepackage{noto}

% Beginning the document  
\begin{document}

% Adding a title  
\title{Kế hoạch kiểm thử cho Tính năng Tìm kiếm Sản phẩm}  
\author{Đội kiểm thử}  
\date{Tháng 7, 2025}  
\maketitle

% Adding sections for the test plan  
\section{Mục tiêu kiểm thử}  
Đảm bảo tính năng “tìm kiếm sản phẩm” của ứng dụng thương mại điện tử hoạt động đúng theo yêu cầu, bao gồm:  
\begin{itemize}  
\item Tìm kiếm sản phẩm với từ khóa hợp lệ, không hợp lệ, và rỗng.  
\item Hiển thị kết quả chính xác, nhanh chóng, và xử lý lỗi phù hợp.  
\item Đảm bảo tính năng hoạt động trên các trình duyệt chính (Chrome, Firefox, Edge).  
\end{itemize}

\section{Phạm vi kiểm thử}  
\begin{itemize}  
\item Kiểm thử chức năng: Nhập từ khóa, hiển thị kết quả, xử lý lỗi.  
\item Kiểm thử giao diện: Đảm bảo ô tìm kiếm và nút tìm kiếm hoạt động đúng.  
\item Kiểm thử hiệu năng: Thời gian phản hồi khi tìm kiếm.  
\item Không bao gồm: Kiểm thử bảo mật hoặc kiểm thử tải trọng cao.  
\end{itemize}

\section{Tài nguyên}  
\begin{itemize}  
\item \textbf{Nhân sự}: 2 QA Engineer, 1 Automation Tester.  
\item \textbf{Công cụ}: Selenium WebDriver, TestNG, Jenkins (cho CI/CD).  
\item \textbf{Môi trường}: Chrome, Firefox, Edge; Hệ điều hành Windows/Linux.  
\end{itemize}

\section{Lịch trình kiểm thử}  
\begin{itemize}  
\item Tuần 1: Thiết kế test case và viết script automation.  
\item Tuần 2: Thực thi kiểm thử tự động và thủ công.  
\item Tuần 3: Phân tích kết quả, báo cáo lỗi, và kiểm thử lại.  
\end{itemize}

\section{Tiêu chí vào/ra}  
\begin{itemize}  
\item \textbf{Tiêu chí vào}:  
\begin{itemize}  
\item Yêu cầu phần mềm được phê duyệt.  
\item Môi trường kiểm thử sẵn sàng (máy chủ, trình duyệt).  
\end{itemize}  
\item \textbf{Tiêu chí ra}:  
\begin{itemize}  
\item 95% test case pass.  
\item Không còn lỗi nghiêm trọng (critical/high).  
\item Báo cáo kiểm thử được phê duyệt.  
\end{itemize}  
\end{itemize}

\section{Các chỉ số theo dõi tiến độ}  
\begin{itemize}  
\item Tỷ lệ test case pass: $\geq$ 95%.  
\item Số lượng lỗi phát hiện: Theo dõi lỗi critical, high, medium, low.  
\item Thời gian thực thi kiểm thử: Đảm bảo hoàn thành trong 3 tuần.  
\end{itemize}

% Ending the document  
\end{document}