**Bài tập tổng hợp về Cấp độ và Loại Testing**

**Phần 1: Lý thuyết**

**Cấp độ kiểm thử**

**a. Mô tả 4 cấp độ kiểm thử chính**

1. **Component Testing**
   * **Mục đích**: Kiểm tra từng thành phần hoặc mô-đun riêng lẻ của phần mềm để đảm bảo nó hoạt động đúng như thiết kế.
   * **Phạm vi**: Tập trung vào một đơn vị mã nguồn (ví dụ: hàm, lớp, hoặc mô-đun) được kiểm thử độc lập, thường bởi các nhà phát triển.
2. **Integration Testing**
   * **Mục đích**: Kiểm tra sự tương tác giữa các thành phần hoặc mô-đun để đảm bảo chúng hoạt động tốt khi được tích hợp.
   * **Phạm vi**: Kiểm tra các giao diện, luồng dữ liệu, hoặc tương tác giữa các mô-đun, chẳng hạn như API hoặc cơ sở dữ liệu.
3. **System Testing**
   * **Mục đích**: Kiểm tra toàn bộ hệ thống đã tích hợp để đảm bảo nó đáp ứng các yêu cầu chức năng và phi chức năng.
   * **Phạm vi**: Toàn bộ ứng dụng được kiểm thử trong môi trường giống môi trường thực tế, bao gồm cả giao diện người dùng, cơ sở dữ liệu, và các tích hợp bên ngoài.
4. **Acceptance Testing**
   * **Mục đích**: Xác nhận rằng phần mềm đáp ứng các yêu cầu của khách hàng và sẵn sàng đưa vào sử dụng.
   * **Phạm vi**: Kiểm tra toàn bộ hệ thống từ góc độ người dùng cuối, thường liên quan đến các kịch bản sử dụng thực tế hoặc yêu cầu kinh doanh.

**b. Ví dụ áp dụng trong ứng dụng đặt phòng khách sạn**

* **Component Testing**: Kiểm tra hàm tính toán số phòng trống dựa trên ngày nhận và trả phòng. Ví dụ: Đảm bảo hàm trả về đúng số lượng phòng trống khi nhập ngày cụ thể.
* **Integration Testing**: Kiểm tra tích hợp giữa mô-đun tìm kiếm phòng và cơ sở dữ liệu khách sạn. Ví dụ: Xác minh rằng mô-đun tìm kiếm phòng gửi đúng yêu cầu SQL và nhận dữ liệu chính xác.
* **System Testing**: Kiểm tra toàn bộ quy trình đặt phòng từ tìm kiếm, chọn phòng, đến thanh toán. Ví dụ: Đảm bảo người dùng có thể hoàn thành việc đặt phòng mà không gặp lỗi.
* **Acceptance Testing**: Kiểm tra xem ứng dụng có đáp ứng yêu cầu của khách hàng không, chẳng hạn như khả năng đặt phòng trong vòng 3 phút với giao diện thân thiện.

**Vai trò của Regression Testing**

**a. Regression Testing là gì?**

Regression Testing là quá trình kiểm thử lại các phần mềm đã được kiểm thử trước đó để đảm bảo rằng các thay đổi hoặc cập nhật mới (sửa lỗi, thêm tính năng) không làm hỏng các chức năng hiện có.

* **Tầm quan trọng**:
  + Phát hiện lỗi mới do thay đổi mã nguồn.
  + Đảm bảo tính ổn định và chất lượng của phần mềm.
  + Giảm rủi ro khi triển khai bản cập nhật.

**b. Ví dụ tình huống cần Regression Testing**

Trong ứng dụng đặt phòng khách sạn, nếu sửa lỗi trong tính năng thanh toán (ví dụ: lỗi không xử lý đúng thanh toán bằng thẻ tín dụng), cần thực hiện regression testing để kiểm tra:

* Tính năng thanh toán vẫn hoạt động đúng với các loại thẻ khác (Visa, MasterCard).
* Các tính năng liên quan như xác nhận đặt phòng hoặc gửi email xác nhận không bị ảnh hưởng.

**Sơ đồ các cấp độ kiểm thử**

**a. Sơ đồ minh họa**

[Component Testing] --> [Integration Testing] --> [System Testing] --> [Acceptance Testing]

* **Mô tả sơ đồ**:
  + **Component Testing**: Kiểm tra các thành phần riêng lẻ (hình chữ nhật nhỏ).
  + **Integration Testing**: Kiểm tra sự kết hợp giữa các thành phần (mũi tên kết nối các chữ nhật).
  + **System Testing**: Kiểm tra toàn bộ hệ thống (hình chữ nhật lớn bao quanh các thành phần).
  + **Acceptance Testing**: Kiểm tra từ góc độ người dùng (hình người dùng ở cuối).

**b. Giải thích sơ đồ**

Sơ đồ này thể hiện quy trình kiểm thử tuần tự, từ kiểm tra chi tiết (component) đến kiểm tra toàn diện (acceptance). Nó giúp:

* Hình dung rõ các giai đoạn kiểm thử và mối quan hệ giữa chúng.
* Đảm bảo mỗi cấp độ được hoàn thành trước khi chuyển sang cấp độ tiếp theo, giảm thiểu lỗi.
* Hỗ trợ lập kế hoạch kiểm thử và phân bổ tài nguyên hiệu quả.

**Testing trong bảo trì phần mềm**

**a. Áp dụng testing khi bảo trì phần mềm**

Khi bảo trì phần mềm, testing được thực hiện để:

* Kiểm tra các thay đổi mới (tính năng, sửa lỗi) hoạt động đúng.
* Đảm bảo các chức năng hiện có không bị ảnh hưởng (regression testing).
* Xác minh tính tương thích với môi trường hoặc hệ thống hiện tại.  
  Các loại kiểm thử thường dùng: regression testing, functional testing, performance testing.

**b. Ví dụ: Thêm tính năng mới vào hệ thống quản lý học sinh**

* **Tính năng mới**: Thêm chức năng “xuất báo cáo điểm trung bình”.
* **Các loại kiểm thử cần thực hiện**:
  1. **Functional Testing**: Kiểm tra tính năng xuất báo cáo có tạo đúng định dạng (PDF, Excel) và dữ liệu chính xác.
  2. **Regression Testing**: Đảm bảo các chức năng cũ (nhập điểm, xem hồ sơ học sinh) không bị ảnh hưởng.
  3. **Integration Testing**: Kiểm tra tích hợp giữa tính năng mới và cơ sở dữ liệu học sinh.
  4. **Performance Testing**: Đảm bảo tính năng mới không làm chậm hệ thống khi xuất báo cáo cho số lượng lớn học sinh.

**Phần 2: Bài tập thực hành**

**Phân tích cấp độ kiểm thử**

**a. Xác định cấp độ kiểm thử**

* **Cấp độ phù hợp**: **System Testing**
* **Lý do**: Lỗi trong tính năng “tìm kiếm phòng trống” liên quan đến việc hệ thống trả về kết quả không chính xác khi chọn ngày trùng lịch bảo trì. Đây là lỗi xảy ra trong môi trường tích hợp đầy đủ (giao diện người dùng, logic xử lý, cơ sở dữ liệu), vì vậy cần kiểm tra toàn bộ hệ thống để phát hiện lỗi này.

**b. Kịch bản kiểm thử**

* **Mô tả kịch bản**: Kiểm tra tính năng tìm kiếm phòng trống khi người dùng chọn ngày trùng với lịch bảo trì khách sạn.
* **Dữ liệu đầu vào**:
  + Ngày nhận phòng: 2025-07-01
  + Ngày trả phòng: 2025-07-03
  + Lịch bảo trì khách sạn: 2025-07-01 đến 2025-07-02
* **Kết quả mong đợi**: Hệ thống hiển thị thông báo “Không có phòng trống trong khoảng thời gian này do lịch bảo trì” hoặc không hiển thị phòng nào trong kết quả tìm kiếm.

**Thiết kế kế hoạch Regression Testing**

**a. Kế hoạch regression testing**

* **Các khu vực cần kiểm tra**:
  + Tính năng tìm kiếm phòng trống (đã sửa lỗi).
  + Tính năng thanh toán (đảm bảo không bị ảnh hưởng).
  + Tính năng xác nhận đặt phòng (gửi email, thông báo).
  + Tính năng hiển thị chi tiết phòng (đảm bảo thông tin chính xác).
* **Phương pháp thực hiện**:
  + **Thủ công**: Kiểm tra các kịch bản người dùng thực tế (tìm kiếm phòng, đặt phòng, thanh toán).
  + **Tự động**: Sử dụng công cụ như Selenium để tự động hóa các kịch bản kiểm thử lặp lại, đặc biệt cho các chức năng thanh toán và xác nhận.
* **Kết quả mong đợi**:
  + Tính năng tìm kiếm phòng trống trả về kết quả chính xác.
  + Các tính năng liên quan (thanh toán, xác nhận) hoạt động bình thường, không có lỗi mới.

**b. Tầm quan trọng của kế hoạch**

Kế hoạch regression testing đảm bảo rằng sửa lỗi trong tính năng tìm kiếm không gây ra lỗi mới ở các chức năng liên quan. Nó giúp:

* Xác minh tính ổn định của toàn bộ ứng dụng sau cập nhật.
* Giảm rủi ro ảnh hưởng đến trải nghiệm người dùng.
* Đảm bảo chất lượng tổng thể của hệ thống.

**Áp dụng Testing trong bảo trì phần mềm**

**a. Các loại kiểm thử cho tính năng mới “đề xuất sách dựa trên lịch sử mượn”**

1. **Functional Testing**: Kiểm tra xem hệ thống có đề xuất sách chính xác dựa trên lịch sử mượn của người dùng không.
   * Ví dụ: Nếu người dùng mượn sách về khoa học, hệ thống phải đề xuất sách khoa học tương tự.
2. **Integration Testing**: Kiểm tra tích hợp giữa tính năng đề xuất sách và cơ sở dữ liệu lịch sử mượn.
   * Ví dụ: Đảm bảo dữ liệu lịch sử mượn được truy xuất chính xác để tạo đề xuất.
3. **Usability Testing**: Kiểm tra giao diện hiển thị đề xuất sách có thân thiện và dễ hiểu với người dùng không.

**b. Sử dụng Regression Testing**

* **Mục đích**: Đảm bảo các chức năng cũ (mượn sách, trả sách, tìm kiếm sách) không bị ảnh hưởng bởi tính năng mới.
* **Cách thực hiện**:
  + Kiểm tra lại các kịch bản mượn/trả sách (thủ công hoặc tự động).
  + Kiểm tra chức năng tìm kiếm sách để đảm bảo kết quả không bị ảnh hưởng bởi logic đề xuất mới.
* **Kết quả mong đợi**: Các chức năng cũ hoạt động bình thường, không xuất hiện lỗi mới.