### **Phần 1: Lý thuyết**

#### **Cấp độ kiểm thử**

**a. Mô tả 4 cấp độ kiểm thử chính:**

1. **Component Testing (Kiểm thử thành phần):**
   * **Mục đích**: Kiểm tra từng thành phần hoặc mô-đun riêng lẻ của phần mềm để đảm bảo chúng hoạt động đúng như thiết kế.
   * **Phạm vi**: Tập trung vào một đơn vị mã nguồn (ví dụ: hàm, lớp, hoặc mô-đun). Thường được thực hiện bởi nhà phát triển trong môi trường phát triển.
   * **Ví dụ**: Kiểm tra hàm tính giá phòng dựa trên số ngày lưu trú trong ứng dụng đặt phòng khách sạn.
2. **Integration Testing (Kiểm thử tích hợp):**
   * **Mục đích**: Kiểm tra sự tương tác giữa các thành phần hoặc mô-đun để đảm bảo chúng hoạt động đúng khi kết hợp với nhau.
   * **Phạm vi**: Tập trung vào giao diện và luồng dữ liệu giữa các thành phần. Có thể bao gồm tích hợp giữa các hệ thống con hoặc với hệ thống bên ngoài.
   * **Ví dụ**: Kiểm tra sự tích hợp giữa mô-đun tìm kiếm phòng trống và mô-đun cơ sở dữ liệu khách sạn để đảm bảo dữ liệu phòng được truy xuất chính xác.
3. **System Testing (Kiểm thử hệ thống):**
   * **Mục đích**: Kiểm tra toàn bộ hệ thống phần mềm như một khối thống nhất để đảm bảo nó đáp ứng các yêu cầu chức năng và phi chức năng.
   * **Phạm vi**: Bao gồm tất cả các thành phần tích hợp, kiểm tra các chức năng tổng thể, hiệu suất, bảo mật, và khả năng tương thích.
   * **Ví dụ**: Kiểm tra toàn bộ quy trình đặt phòng (từ tìm kiếm, chọn phòng, thanh toán đến xác nhận) trong ứng dụng đặt phòng khách sạn.
4. **Acceptance Testing (Kiểm thử chấp nhận):**
   * **Mục đích**: Xác minh rằng phần mềm đáp ứng các yêu cầu của khách hàng hoặc người dùng cuối và sẵn sàng đưa vào sử dụng.
   * **Phạm vi**: Thường được thực hiện trong môi trường giống sản xuất, tập trung vào trải nghiệm người dùng và các yêu cầu kinh doanh.
   * **Ví dụ**: Người dùng cuối kiểm tra ứng dụng đặt phòng khách sạn để đảm bảo quy trình đặt phòng dễ sử dụng và đáp ứng nhu cầu thực tế.

**b. Ví dụ cụ thể trong ứng dụng đặt phòng khách sạn:**

* **Component Testing**: Kiểm tra hàm kiểm tra tính hợp lệ của ngày đặt phòng (ví dụ: không cho phép chọn ngày trong quá khứ).
* **Integration Testing**: Kiểm tra tích hợp giữa mô-đun tìm kiếm phòng và mô-đun thanh toán để đảm bảo thông tin phòng được truyền đúng sang bước thanh toán.
* **System Testing**: Kiểm tra toàn bộ ứng dụng để đảm bảo người dùng có thể hoàn thành quy trình đặt phòng từ đầu đến cuối mà không gặp lỗi.
* **Acceptance Testing**: Khách hàng (chủ khách sạn hoặc người dùng) kiểm tra xem ứng dụng có đáp ứng các yêu cầu như giao diện thân thiện, tốc độ phản hồi nhanh, và chính xác trong việc hiển thị phòng trống.

#### **Vai trò của Regression Testing**

**a. Regression Testing là gì? Tầm quan trọng:**

* **Định nghĩa**: Regression Testing (kiểm thử hồi quy) là quá trình kiểm tra lại các chức năng đã được kiểm tra trước đó để đảm bảo rằng các thay đổi mới (sửa lỗi, cập nhật, hoặc thêm tính năng) không làm phát sinh lỗi mới hoặc ảnh hưởng đến các chức năng hiện có.
* **Tầm quan trọng**:
  + Đảm bảo rằng các sửa đổi không phá vỡ các chức năng cũ.
  + Giúp duy trì chất lượng phần mềm trong suốt vòng đời phát triển và bảo trì.
  + Giảm rủi ro lỗi không mong muốn khi triển khai các bản cập nhật.

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

* **Tình huống**: Trong ứng dụng đặt phòng khách sạn, một lỗi trong tính năng thanh toán (ví dụ: thanh toán thất bại khi sử dụng thẻ tín dụng) được sửa. Sau khi sửa lỗi, cần thực hiện regression testing để kiểm tra:
  + Tính năng thanh toán vẫn hoạt động tốt với tất cả các phương thức thanh toán (thẻ tín dụng, ví điện tử, chuyển khoản).
  + Các tính năng liên quan như xác nhận đặt phòng, gửi email xác nhận, hoặc cập nhật trạng thái phòng 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**: Đầu tiên, từng thành phần riêng lẻ được kiểm tra độc lập.
  + **Integration Testing**: Các thành phần được tích hợp và kiểm tra sự tương tác.
  + **System Testing**: Toàn bộ hệ thống được kiểm tra như một khối thống nhất.
  + **Acceptance Testing**: Cuối cùng, hệ thống được kiểm tra bởi người dùng cuối để đảm bảo đáp ứng yêu cầu.

**b. Giải thích vai trò sơ đồ**:

* Sơ đồ này giúp hình dung rõ ràng các giai đoạn kiểm thử trong quy trình phát triển phần mềm, từ cấp độ nhỏ nhất (thành phần) đến cấp độ cao nhất (chấp nhận).
* Nó thể hiện sự tiến triển tuần tự, đảm bảo rằng các lỗi được phát hiện và sửa chữa ở giai đoạn sớm trước khi chuyển sang giai đoạn tiếp theo.
* Giúp nhóm dự án lập kế hoạch kiểm thử hiệu quả, phân bổ nguồn lực hợp lý và đảm bảo chất lượng phần mềm.

#### **Testing trong bảo trì phần mềm**

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

* Khi bảo trì phần mềm, các thay đổi (sửa lỗi, cập nhật, hoặc thêm tính năng) cần được kiểm tra kỹ lưỡng để đảm bảo không ảnh hưởng đến các chức năng hiện có.
* Các loại kiểm thử thường được áp dụng:
  + **Regression Testing**: Kiểm tra lại các chức năng cũ để đảm bảo chúng không bị ảnh hưởng.
  + **Functional Testing**: Kiểm tra tính năng mới hoặc phần sửa lỗi hoạt động đúng.
  + **Integration Testing**: Kiểm tra tích hợp giữa tính năng mới và các thành phần hiện có.
  + **Performance Testing**: Đảm bảo hiệu suất hệ thống không bị suy giảm sau cập nhật.

**b. Ví dụ về cập nhật phần mềm:**

* **Tình huống**: Thêm tính năng “đề xuất khóa học” vào hệ thống quản lý học sinh dựa trên điểm số và sở thích.
* **Các loại kiểm thử cần thực hiện**:
  + **Functional Testing**: Kiểm tra xem tính năng đề xuất khóa học có đưa ra các gợi ý chính xác dựa trên dữ liệu đầu vào (điểm số, sở thích).
  + **Integration Testing**: Kiểm tra tích hợp giữa tính năng đề xuất và cơ sở dữ liệu học sinh để đảm bảo dữ liệu được truy xuất và xử lý đúng.
  + **Regression Testing**: Kiểm tra lại các chức năng cũ như quản lý hồ sơ học sinh, đăng ký khóa học, hoặc tính toán điểm trung bình để đảm bảo chúng không bị ảnh hưởng.
  + **Usability Testing**: Đảm bảo giao diện của tính năng mới dễ sử dụng và phù hợp với người dùng.

### **Phần 2: Bài tập thực hành**

#### **Phân tích cấp độ kiểm thử**

**Tình huống**: Tính năng “tìm kiếm phòng trống” trong ứng dụng đặt phòng khách sạn trả về kết quả không chính xác khi người dùng chọn ngày trùng với lịch bảo trì.

**a. Cấp độ kiểm thử phù hợp:**

* **Cấp độ**: Integration Testing.
* **Lý do**: Lỗi này liên quan đến sự tương tác giữa mô-đun “tìm kiếm phòng trống” và mô-đun quản lý lịch bảo trì khách sạn. Integration Testing sẽ giúp kiểm tra xem dữ liệu lịch bảo trì có được truyền và xử lý đúng trong quá trình tìm kiếm phòng hay không.

**b. Kịch bản kiểm thử:**

* **Mô tả kịch bản**: Kiểm tra chức năng tìm kiếm phòng trống khi người dùng chọn ngày trùng với lịch bảo trì khách sạn.
* **Dữ liệu đầu vào**:
  + Ngày check-in: 2025-07-15 (trùng với lịch bảo trì của khách sạn).
  + Ngày check-out: 2025-07-17.
  + Loại phòng: Deluxe.
* **Kết quả mong đợi**:
  + Hệ thống thông báo rằng không có phòng trống vào ngày 2025-07-15 do lịch bảo trì.
  + Hiển thị thông báo lỗi rõ ràng hoặc gợi ý ngày khác.

#### **Thiết kế kế hoạch Regression Testing**

**Tình huống**: Sửa lỗi trong tính năng “tìm kiếm phòng trống”.

**a. Kế hoạch Regression Testing:**

* **Các khu vực cần kiểm tra**:
  + **Tính năng tìm kiếm phòng trống**: Kiểm tra lại với các trường hợp khác nhau (ngày hợp lệ, ngày không hợp lệ, các loại phòng khác nhau).
  + **Tính năng thanh toán**: Đảm bảo thông tin phòng trống được truyền chính xác sang bước thanh toán.
  + **Tính năng xác nhận đặt phòng**: Kiểm tra email xác nhận và trạng thái đặt phòng.
  + **Tính năng quản lý lịch bảo trì**: Đảm bảo lịch bảo trì được cập nhật và hiển thị đúng.
* **Phương pháp thực hiện**:
  + **Thủ công**: Kiểm tra các trường hợp phức tạp, đặc biệt là giao diện người dùng và thông báo lỗi.
  + **Tự động**: Sử dụng các công cụ như Selenium để chạy các kịch bản kiểm thử tự động cho các chức năng lặp lại (ví dụ: tìm kiếm phòng với nhiều bộ dữ liệu).
* **Kết quả mong đợi**:
  + Tất cả các tính năng liên quan hoạt động bình thường, không có lỗi mới xuất hiện.
  + Tính năng tìm kiếm phòng trả về kết quả chính xác, bao gồm cả trường hợp ngày bảo trì.
  + Thanh toán và xác nhận đặt phòng hoàn tất mà không có lỗi.

**b. Lý do kế hoạch đảm bảo không có lỗi mới:**

* Kế hoạch kiểm tra toàn diện các tính năng liên quan và tích hợp, đảm bảo rằng sửa lỗi không gây ra vấn đề ở các phần khác của hệ thống.
* Sử dụng cả kiểm thử thủ công và tự động giúp phát hiện lỗi ở các khía cạnh khác nhau (chức năng, giao diện, hiệu suất).
* Kiểm tra lại các chức năng cũ đảm bảo tính ổn định của hệ thống sau khi cập nhật.

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

**Tình huống**: Thêm tính năng “đề xuất sách dựa trên lịch sử mượn” vào hệ thống quản lý thư viện.

**a. Các loại kiểm thử cần thực hiện:**

1. **Functional Testing**: Kiểm tra xem tính năng đề xuất sách có đưa ra các gợi ý chính xác dựa trên lịch sử mượn (ví dụ: đề xuất sách cùng thể loại với sách đã mượn).
2. **Integration Testing**: Kiểm tra tích hợp giữa tính năng đề xuất và cơ sở dữ liệu lịch sử mượn để đảm bảo dữ liệu được truy xuất đúng.
3. **Regression Testing**: Kiểm tra lại các chức năng cũ như mượn sách, trả sách, và tìm kiếm sách để đảm bảo chúng không bị ảnh hưởng bởi tính năng mới.

**b. Sử dụng Regression Testing:**

* **Cách thực hiện**:
  + Chạy lại các kịch bản kiểm thử cũ cho các chức năng như mượn sách, trả sách, và quản lý hồ sơ người dùng.
  + Sử dụng kiểm thử tự động để kiểm tra nhanh các trường hợp sử dụng phổ biến (ví dụ: tìm kiếm sách theo tiêu đề hoặc tác giả).
  + Kiểm tra thủ công các trường hợp phức tạp như xử lý lỗi khi mượn sách quá hạn.
* **Mục đích**: Đảm bảo rằng tính năng đề xuất sách mới không làm gián đoạn hoặc gây lỗi cho các chức năng hiện có, duy trì chất lượng tổng thể của hệ thống.