### **Phần 1: Lý thuyết**

#### **1. Lập kế hoạch kiểm thử**

**a. Tiêu chí vào/ra (entry/exit criteria) trong kế hoạch kiểm thử là gì? Giải thích tại sao chúng quan trọng.**

* **Tiêu chí vào (Entry Criteria):** Là các điều kiện cần thỏa mãn để bắt đầu quá trình kiểm thử, ví dụ: tài liệu yêu cầu hoàn thiện, môi trường kiểm thử sẵn sàng, hoặc mã nguồn đã được biên dịch thành công.
* **Tiêu chí ra (Exit Criteria):** Là các điều kiện để kết thúc kiểm thử, ví dụ: 95% test case pass, không còn lỗi nghiêm trọng (critical) chưa giải quyết, hoặc đạt mức độ bao phủ mã (code coverage) yêu cầu.
* **Tầm quan trọng:**
  + **Tiêu chí vào** đảm bảo rằng kiểm thử bắt đầu trong điều kiện phù hợp, tránh lãng phí thời gian và tài nguyên khi hệ thống chưa sẵn sàng.
  + **Tiêu chí ra** giúp xác định khi nào kiểm thử hoàn tất, đảm bảo chất lượng sản phẩm đạt yêu cầu trước khi triển khai, đồng thời cung cấp cơ sở để đánh giá tiến độ và hiệu quả kiểm thử.

**b. Cách ưu tiên test case dựa trên mức độ rủi ro và ví dụ cụ thể.**

* **Cách ưu tiên:**
  + Đánh giá rủi ro dựa trên: **tác động (impact)** (mức độ ảnh hưởng đến người dùng hoặc hệ thống) và **xác suất xảy ra (likelihood)**.
  + Ưu tiên test case cho các tính năng: có tác động lớn (ví dụ: ảnh hưởng đến tài chính, trải nghiệm người user), được sử dụng thường xuyên, hoặc có lịch sử lỗi cao.
  + Sử dụng ma trận rủi ro (Risk Matrix) để xếp hạng mức độ rủi ro (cao, trung bình, thấp).
* **Ví dụ: Kiểm thử tính năng thanh toán trong ứng dụng thương mại điện tử.**
  + **Test case ưu tiên cao:** Kiểm tra việc xử lý thanh toán khi người dùng sử dụng thẻ tín dụng.
  + **Lý do ưu tiên:**
    - **Tác động cao:** Lỗi trong tính năng thanh toán có thể dẫn đến mất doanh thu, mất niềm tin của khách hàng.
    - **Xác suất xảy ra:** Thanh toán là tính năng phức tạp, tích hợp với nhiều hệ thống bên ngoài (ngân hàng, cổng thanh toán), dễ xảy ra lỗi.
    - **Tần suất sử dụng:** Tính năng thanh toán được sử dụng thường xuyên bởi hầu hết người dùng.

#### **2. Viết báo cáo defect**

**a. Một báo cáo defect cần bao gồm những yếu tố nào? Liệt kê ít nhất 5 yếu tố và giải thích tầm quan trọng.**

1. **Tiêu đề defect:** Tóm tắt ngắn gọn lỗi, giúp đội phát triển nhanh chóng hiểu vấn đề.
   * *Tầm quan trọng:* Giúp phân loại và tìm kiếm defect dễ dàng, tiết kiệm thời gian.
2. **Mô tả defect:** Chi tiết về lỗi, bao gồm ngữ cảnh và hành vi bất thường.
   * *Tầm quan trọng:* Cung cấp thông tin đầy đủ để đội phát triển hiểu rõ vấn đề.
3. **Bước tái hiện:** Các bước cụ thể để tái hiện lỗi.
   * *Tầm quan trọng:* Giúp đội phát triển tái tạo lỗi, xác định nguyên nhân và kiểm tra sau khi sửa.
4. **Mức độ nghiêm trọng (Severity):** Đánh giá mức độ ảnh hưởng của lỗi (critical, major, minor).
   * *Tầm quan trọng:* Giúp đội phát triển ưu tiên sửa lỗi theo mức độ quan trọng.
5. **Kết quả thực tế và kết quả mong đợi:** Mô tả hành vi hiện tại của hệ thống và hành vi mong muốn.
   * *Tầm quan trọng:* Làm rõ sự khác biệt giữa thực tế và yêu cầu, hỗ trợ xác định mục tiêu sửa lỗi.

**b. Ví dụ báo cáo defect và cách hỗ trợ đội phát triển.**

* **Ví dụ: Lỗi hiển thị sai giá sản phẩm.**
  + **Tiêu đề defect:** Giá sản phẩm hiển thị sai trên trang chi tiết sản phẩm.
  + **Mô tả defect:** Khi xem chi tiết sản phẩm "Áo thun nam", giá hiển thị là 0 VNĐ thay vì giá thực tế.
  + **Bước tái hiện:**
    1. Truy cập trang web thương mại điện tử.
    2. Tìm kiếm sản phẩm "Áo thun nam".
    3. Nhấp vào sản phẩm để xem chi tiết.
    4. Quan sát giá hiển thị trên trang.
  + **Mức độ nghiêm trọng:** Major (ảnh hưởng đến trải nghiệm người dùng và quyết định mua hàng).
  + **Kết quả thực tế:** Giá hiển thị là 0 VNĐ.
  + **Kết quả mong đợi:** Giá hiển thị đúng là 200.000 VNĐ (theo thông tin trong cơ sở dữ liệu).
* **Cách hỗ trợ đội phát triển:**
  + Báo cáo cung cấp thông tin chi tiết, rõ ràng về lỗi, giúp lập trình viên xác định chính xác vị trí lỗi (có thể liên quan đến truy xuất dữ liệu giá).
  + Bước tái hiện cụ thể giúp tái tạo lỗi nhanh chóng, tiết kiệm thời gian phân tích.
  + Mức độ nghiêm trọng giúp đội ưu tiên sửa lỗi này trước các lỗi ít quan trọng hơn.

#### **3. Phân tích rủi ro sản phẩm và dự án**

**a. Phân biệt rủi ro sản phẩm và rủi ro dự án trong testing. Đưa ra ví dụ.**

* **Rủi ro sản phẩm:** Liên quan đến chất lượng hoặc chức năng của phần mềm, có thể ảnh hưởng đến người dùng cuối.
  + **Ví dụ:** Lỗi trong tính năng xác thực người dùng có thể cho phép truy cập trái phép, gây rủi ro bảo mật.
* **Rủi ro dự án:** Liên quan đến quá trình phát triển, chẳng hạn như lịch trình, tài nguyên, hoặc quản lý dự án.
  + **Ví dụ:** Thiếu nhân sự kiểm thử dẫn đến không đủ thời gian kiểm tra toàn bộ tính năng.
* **Sự khác biệt:** Rủi ro sản phẩm tập trung vào sản phẩm cuối cùng, trong khi rủi ro dự án liên quan đến quy trình phát triển và quản lý dự án.

**b. Cách phân tích rủi ro để tập trung nỗ lực kiểm thử.**

1. **Xác định rủi ro:** Liệt kê các rủi ro tiềm ẩn dựa trên yêu cầu, tính năng, và lịch sử dự án.
2. **Đánh giá rủi ro:** Sử dụng ma trận rủi ro để xếp hạng dựa trên tác động và xác suất xảy ra.
3. **Ưu tiên kiểm thử:** Tập trung kiểm thử vào các khu vực có rủi ro cao (ví dụ: tính năng phức tạp, tích hợp bên ngoài, hoặc tính năng quan trọng với người dùng).
4. **Lập kế hoạch giảm thiểu:** Xây dựng các test case chi tiết, tăng cường kiểm thử tự động, hoặc phân bổ thêm tài nguyên cho các khu vực rủi ro cao.
5. **Theo dõi và cập nhật:** Liên tục đánh giá lại rủi ro trong suốt dự án để điều chỉnh nỗ lực kiểm thử.

#### **4. Test Plan và theo dõi tiến độ kiểm thử**

**a. Một test plan cơ bản bao gồm những thành phần nào? Liệt kê ít nhất 5 thành phần chính.**

1. **Mục tiêu kiểm thử:** Xác định mục đích kiểm thử (ví dụ: đảm bảo tính năng hoạt động đúng, đạt độ tin cậy 99%).
2. **Phạm vi kiểm thử:** Xác định những gì sẽ kiểm thử (tính năng, module) và không kiểm thử.
3. **Tài nguyên:** Liệt kê nhân sự, công cụ (ví dụ: Selenium, JIRA), và môi trường kiểm thử.
4. **Lịch trình kiểm thử:** Thời gian bắt đầu, kết thúc, và các cột mốc quan trọng.
5. **Chiến lược kiểm thử:** Mô tả loại kiểm thử (unit, integration, system) và cách tiếp cận (thủ công hay tự động).

**b. Cách sử dụng các chỉ số để theo dõi tiến độ kiểm thử và đánh giá chất lượng.**

* **Tỷ lệ test case hoàn thành:** Đo lường phần trăm test case đã thực thi so với tổng số test case.
  + *Ứng dụng:* Giúp đánh giá tiến độ kiểm thử, xác định xem có kịp lịch trình hay không.
* **Số lượng defect:** Theo dõi số lượng lỗi được phát hiện, phân loại theo mức độ nghiêm trọng.
  + *Ứng dụng:* Đánh giá chất lượng sản phẩm; số lượng lỗi nghiêm trọng cao cho thấy cần tập trung sửa chữa.
* **Tỷ lệ defect được giải quyết:** Đo lường số lỗi đã được sửa so với tổng số lỗi.
  + *Ứng dụng:* Đánh giá hiệu quả của đội phát triển và tiến độ khắc phục lỗi.
* **Độ bao phủ kiểm thử (Test Coverage):** Đo lường tỷ lệ mã nguồn hoặc yêu cầu được kiểm thử.
  + *Ứng dụng:* Đảm bảo kiểm thử bao quát các tính năng quan trọng, giảm rủi ro bỏ sót lỗi.

### **Phần 2: Bài tập thực hành**

#### **1. Lập kế hoạch kiểm thử (Ứng dụng đặt lịch khám bác sĩ trực tuyến)**

**a. Đề xuất tiêu chí vào/ra cho quá trình kiểm thử.**

* **Tiêu chí vào:**
  + Tài liệu yêu cầu (SRS) đã được phê duyệt.
  + Môi trường kiểm thử (server, cơ sở dữ liệu) đã được thiết lập và ổn định.
  + Mã nguồn đã được triển khai và biên dịch thành công.
  + Đội kiểm thử đã được phân công và đào tạo về ứng dụng.
* **Tiêu chí ra:**
  + 95% test case pass (không có lỗi critical hoặc major chưa giải quyết).
  + Độ bao phủ kiểm thử đạt ít nhất 90% cho các tính năng chính (đặt lịch, hủy lịch, xác nhận lịch).
  + Hệ thống đáp ứng thời gian phản hồi dưới 2 giây cho các giao dịch chính.
  + Tất cả lỗi nghiêm trọng (critical) đã được sửa và kiểm tra lại.

**b. Xác định ít nhất 3 test case ưu tiên cao dựa trên rủi ro.**

1. **Test case: Kiểm tra xác nhận lịch hẹn.**
   * **Mô tả:** Kiểm tra xem hệ thống có gửi thông báo xác nhận lịch hẹn đến bệnh nhân và bác sĩ sau khi đặt lịch thành công hay không.
   * **Lý do ưu tiên:**
     + **Tác động cao:** Lỗi xác nhận lịch có thể gây nhầm lẫn, dẫn đến bỏ lỡ lịch hẹn.
     + **Tần suất sử dụng:** Là tính năng cốt lõi, được sử dụng bởi mọi người dùng.
2. **Test case: Kiểm tra tính hợp lệ của thời gian đặt lịch.**
   * **Mô tả:** Kiểm tra xem hệ thống có ngăn chặn đặt lịch ngoài giờ làm việc của bác sĩ (ví dụ: 12:00-13:00 là giờ nghỉ trưa).
   * **Lý do ưu tiên:**
     + **Tác động cao:** Đặt lịch vào giờ không hợp lệ gây xung đột lịch trình, ảnh hưởng đến bác sĩ và bệnh nhân.
     + **Xác suất xảy ra:** Lỗi logic xử lý thời gian dễ xảy ra nếu không kiểm tra kỹ.
3. **Test case: Kiểm tra tích hợp với hệ thống thanh toán (nếu có).**
   * **Mô tả:** Kiểm tra xem hệ thống xử lý thanh toán phí đặt lịch đúng cách (ví dụ: trừ tiền từ ví điện tử).
   * **Lý do ưu tiên:**
     + **Tác động cao:** Lỗi thanh toán có thể gây mất tiền hoặc không hoàn thành giao dịch, ảnh hưởng uy tín.
     + **Phức tạp:** Tích hợp với hệ thống thanh toán bên ngoài thường dễ xảy ra lỗi.

#### **2. Viết báo cáo defect**

**a. Báo cáo defect: Hệ thống cho phép đặt lịch vào giờ nghỉ trưa của bác sĩ.**

* **Tiêu đề defect:** Hệ thống cho phép đặt lịch vào giờ nghỉ trưa của bác sĩ (12:00-13:00).
* **Mô tả defect:** Hệ thống không chặn việc đặt lịch khám trong khung giờ nghỉ trưa (12:00-13:00), dẫn đến lịch hẹn không hợp lệ được tạo.
* **Bước tái hiện:**
  1. Đăng nhập vào ứng dụng với tài khoản bệnh nhân.
  2. Chọn bác sĩ và khung giờ 12:30 để đặt lịch.
  3. Nhập thông tin cần thiết và nhấn "Xác nhận đặt lịch".
  4. Quan sát hệ thống cho phép đặt lịch thành công và gửi thông báo xác nhận.
* **Mức độ nghiêm trọng:** Major (gây xung đột lịch trình, ảnh hưởng đến bác sĩ và bệnh nhân).
* **Kết quả thực tế:** Hệ thống cho phép đặt lịch vào 12:30 và gửi thông báo xác nhận.
* **Kết quả mong đợi:** Hệ thống hiển thị thông báo lỗi: "Khung giờ 12:00-13:00 không khả dụng để đặt lịch."

**b. Cách báo cáo này giúp đội phát triển sửa lỗi nhanh chóng.**

* **Thông tin chi tiết:** Mô tả và bước tái hiện giúp lập trình viên xác định chính xác lỗi nằm ở logic kiểm tra khung giờ.
* **Mức độ nghiêm trọng:** Xác định lỗi là "Major" giúp đội ưu tiên sửa lỗi này trước các lỗi ít quan trọng hơn.
* **Kết quả mong đợi:** Làm rõ hành vi đúng của hệ thống, giúp lập trình viên biết cần điều chỉnh gì (thêm điều kiện chặn khung giờ 12:00-13:00).
* **Tái hiện dễ dàng:** Bước tái hiện cụ thể giúp đội phát triển tái tạo lỗi nhanh chóng, xác định nguyên nhân và kiểm tra sau khi sửa.

#### **3. Phân tích rủi ro và soạn Test Plan (Ứng dụng quản lý kho hàng)**

**a. Phân tích rủi ro.**

* **Rủi ro sản phẩm:**
  1. **Lỗi tính toán số lượng tồn kho:** Số lượng tồn kho hiển thị sai do lỗi trong logic tính toán (ví dụ: không cập nhật khi nhập/xuất kho).
     + **Tác động:** Gây nhầm lẫn trong quản lý, dẫn đến thiếu hụt hoặc dư thừa hàng hóa.
  2. **Lỗi tích hợp với hệ thống bên thứ ba:** Tích hợp với hệ thống vận chuyển hoặc nhà cung cấp bị lỗi, gây gián đoạn quy trình.
     + **Tác động:** Làm chậm quá trình giao hàng, ảnh hưởng đến khách hàng.
  3. **Lỗi bảo mật:** Dữ liệu tồn kho nhạy cảm (ví dụ: số lượng hàng hóa giá trị cao) bị truy cập trái phép.
     + **Tác động:** Gây mất dữ liệu hoặc thiệt hại tài chính.
* **Rủi ro dự án:**
  1. **Thiếu tài nguyên kiểm thử:** Đội kiểm thử không đủ nhân sự hoặc công cụ để kiểm tra toàn diện.
     + **Tác động:** Bỏ sót lỗi quan trọng, làm giảm chất lượng sản phẩm.
  2. **Lịch trình bị trì hoãn:** Giai đoạn phát triển kéo dài, làm rút ngắn thời gian kiểm thử.
     + **Tác động:** Không đủ thời gian để kiểm tra tất cả tính năng, tăng rủi ro lỗi trong sản phẩm cuối.

**b. Test Plan cơ bản.**

* **Mục tiêu kiểm thử:**
  + Đảm bảo hệ thống quản lý kho hàng hoạt động đúng, tính toán tồn kho chính xác, tích hợp ổn định với hệ thống bên thứ ba.
  + Đạt độ bao phủ kiểm thử 90% cho các tính năng chính (nhập kho, xuất kho, báo cáo tồn kho).
  + Phát hiện và sửa tất cả lỗi critical và major trước khi triển khai.
* **Phạm vi kiểm thử:**
  + **Kiểm thử:** Tính năng nhập kho, xuất kho, báo cáo tồn kho, tích hợp với hệ thống vận chuyển, bảo mật dữ liệu.
  + **Không kiểm thử:** Các tính năng chưa phát triển (ví dụ: dự báo nhu cầu kho).
* **Tài nguyên:**
  + **Nhân sự:** 1 QA Lead, 3 QA Tester, 1 chuyên viên kiểm thử tự động.
  + **Công cụ:** JIRA (quản lý defect), Selenium (kiểm thử tự động), Postman (kiểm thử API).
  + **Môi trường kiểm thử:** Server staging với cơ sở dữ liệu mẫu.
* **Lịch trình kiểm thử:**
  + Tuần 1: Thiết kế test case, thiết lập môi trường kiểm thử.
  + Tuần 2-3: Thực hiện kiểm thử thủ công và tự động, báo cáo defect.
  + Tuần 4: Kiểm thử hồi quy, xác nhận sửa lỗi, hoàn thiện báo cáo kiểm thử.
* **Các chỉ số theo dõi tiến độ:**
  + Tỷ lệ test case hoàn thành: Nhắm mục tiêu 100% test case được thực thi.
  + Số lượng defect: Theo dõi số lỗi phát hiện mỗi tuần, phân loại theo mức độ nghiêm trọng.
  + Tỷ lệ defect được giải quyết: Đảm bảo 100% lỗi critical và major được sửa trước khi kết thúc kiểm thử.
  + Độ bao phủ kiểm thử: Đạt ít nhất 90% cho các tính năng chính.