**Bài tập tổng hợp về Kiểm thử dựa trên Kinh nghiệm và Hợp tác**

**Phần 1: Lý thuyết**

**Error Guessing**

**a. Error Guessing là gì?**

Error Guessing là một kỹ thuật kiểm thử dựa trên kinh nghiệm, trong đó tester sử dụng kiến thức và kinh nghiệm thực tế để dự đoán các lỗi tiềm ẩn trong hệ thống. Thay vì dựa vào kịch bản kiểm thử cố định, tester phân tích hệ thống, các trường hợp sử dụng và các lỗi thường gặp trong các dự án tương tự để xác định các khu vực dễ xảy ra lỗi. Kỹ thuật này thường được sử dụng khi thời gian kiểm thử hạn chế hoặc tài liệu yêu cầu không đầy đủ.

**Cách sử dụng kinh nghiệm thực tế:**

* Dựa vào kinh nghiệm từ các dự án trước, tester xác định các khu vực thường xảy ra lỗi, ví dụ: các trường nhập liệu, tích hợp hệ thống, hoặc xử lý dữ liệu.
* Phân tích các lỗi phổ biến trong các ứng dụng tương tự (ví dụ: lỗi xác thực, lỗi giao diện, hoặc lỗi logic nghiệp vụ).
* Sử dụng các mẫu lỗi (error patterns) như nhập liệu không hợp lệ, trường hợp ranh giới, hoặc lỗi đồng bộ hóa.

**b. Ví dụ cụ thể**

Trong một ứng dụng thương mại điện tử, kiểm thử tính năng thanh toán:

* **Dự đoán lỗi**: Người dùng nhập sai định dạng số thẻ tín dụng (ví dụ: thiếu số hoặc ký tự không hợp lệ).
* **Kinh nghiệm áp dụng**: Tester biết rằng các trường nhập liệu liên quan đến thanh toán thường dễ bị lỗi do người dùng nhập sai hoặc hệ thống không kiểm tra kỹ định dạng.
* **Test case**: Nhập số thẻ tín dụng với 15 chữ số thay vì 16 chữ số, hệ thống phải hiển thị thông báo lỗi "Số thẻ không hợp lệ".

**Exploratory Testing**

**a. Exploratory Testing là gì?**

Exploratory Testing là một phương pháp kiểm thử trong đó tester khám phá hệ thống mà không cần kịch bản kiểm thử cố định. Tester đồng thời thiết kế và thực hiện các test case, dựa trên sự hiểu biết về hệ thống, phản hồi trực tiếp từ ứng dụng và khả năng sáng tạo. Phương pháp này hiệu quả vì:

* **Linh hoạt**: Tester có thể điều chỉnh hướng kiểm thử dựa trên các phát hiện trong quá trình thực hiện.
* **Khám phá lỗi bất ngờ**: Giúp phát hiện các lỗi không được dự đoán trước trong tài liệu yêu cầu.
* **Tận dụng kinh nghiệm**: Tester sử dụng trực giác và kinh nghiệm để tập trung vào các khu vực có khả năng xảy ra lỗi.

**b. Ví dụ tình huống**

Trong kiểm thử giao diện người dùng của một ứng dụng di động (ví dụ: ứng dụng đặt vé):

* **Tình huống**: Tester khám phá giao diện chọn ghế trên ứng dụng di động.
* **Hành động**: Tester thử thay đổi nhanh chóng giữa các ghế, kiểm tra khả năng phản hồi của giao diện, hoặc kiểm tra hiển thị trên các kích thước màn hình khác nhau.
* **Kết quả**: Phát hiện lỗi như giao diện hiển thị sai vị trí ghế hoặc nút chọn ghế không phản hồi khi nhấn nhanh.

**User Story và Tiêu chí chấp nhận**

**a. User Story là gì?**

User Story là một mô tả ngắn gọn, đơn giản về một tính năng từ góc nhìn của người dùng cuối, thường được viết theo định dạng: **Là một [vai trò], Tôi muốn [chức năng], Để [mục đích]**. Tiêu chí chấp nhận (Acceptance Criteria) là các điều kiện cụ thể mà tính năng phải đáp ứng để được coi là hoàn thành.

**Lợi ích của User Story và tiêu chí chấp nhận**:

* Tăng cường sự hiểu biết chung giữa các thành viên trong nhóm (developer, tester, product owner).
* Làm rõ yêu cầu, giúp tester thiết kế các test case chính xác.
* Hỗ trợ hợp tác đội nhóm bằng cách cung cấp mục tiêu rõ ràng và có thể kiểm tra.

**b. Ví dụ User Story**

**User Story**: Là một người dùng mới, Tôi muốn đăng ký tài khoản, Để có thể sử dụng các tính năng của ứng dụng.

**Tiêu chí chấp nhận**:

1. Người dùng phải nhập đầy đủ thông tin: email, mật khẩu, xác nhận mật khẩu.
2. Hệ thống phải hiển thị thông báo lỗi nếu email đã được sử dụng.
3. Hệ thống phải gửi email xác nhận sau khi đăng ký thành công.

**Acceptance Test-Driven Development (ATDD)**

**a. ATDD là gì?**

ATDD là một phương pháp phát triển phần mềm trong đó các tiêu chí chấp nhận được định nghĩa trước khi phát triển bắt đầu. Tester, developer và product owner hợp tác để viết các test case dựa trên tiêu chí chấp nhận, sau đó sử dụng chúng để hướng dẫn quá trình phát triển.

**Cải thiện hiệu quả làm việc nhóm**:

* **Hợp tác chặt chẽ**: Các bên liên quan (developer, tester, product owner) cùng làm việc để hiểu rõ yêu cầu.
* **Giảm lỗi**: Các test case được viết trước giúp đảm bảo mã đáp ứng yêu cầu ngay từ đầu.
* **Tăng chất lượng**: Tập trung vào các yêu cầu của người dùng cuối.

**b. Áp dụng ATDD trong dự án**

**Ví dụ**: Phát triển tính năng tìm kiếm sản phẩm trong một ứng dụng thương mại điện tử.

* **Quy trình**:
  1. Product owner định nghĩa yêu cầu: "Người dùng có thể tìm kiếm sản phẩm bằng từ khóa."
  2. Tester và developer cùng viết tiêu chí chấp nhận, ví dụ: "Hệ thống trả về danh sách sản phẩm phù hợp với từ khóa trong vòng 2 giây."
  3. Tester thiết kế test case trước khi coding: Nhập từ khóa "áo thun", hệ thống hiển thị các sản phẩm liên quan.
  4. Developer viết mã để đáp ứng test case.
  5. Tester chạy test case để xác nhận tính năng hoạt động đúng.
* **Lợi ích**: Đảm bảo tính năng đáp ứng yêu cầu, giảm thiểu lỗi do hiểu sai yêu cầu, và tăng cường giao tiếp trong nhóm.

**Phần 2: Bài tập thực hành**

**Áp dụng Error Guessing**

**Ứng dụng**: Ứng dụng đặt vé xe buýt trực tuyến, tính năng “chọn ghế ngồi”.

**a. Dự đoán lỗi tiềm ẩn**

1. **Lỗi trùng ghế**: Hệ thống cho phép nhiều người dùng chọn cùng một ghế do lỗi đồng bộ hóa.
   * **Lý do**: Các ứng dụng đặt vé thường gặp vấn đề về đồng bộ dữ liệu khi nhiều người dùng truy cập cùng lúc.
2. **Lỗi hiển thị trạng thái ghế**: Ghế đã được đặt nhưng vẫn hiển thị là “trống” trên giao diện.
   * **Lý do**: Dựa trên kinh nghiệm, lỗi giao diện thường xảy ra khi trạng thái không được cập nhật kịp thời.
3. **Lỗi chọn ghế không hợp lệ**: Người dùng có thể chọn ghế ngoài phạm vi (ví dụ: ghế không tồn tại).
   * **Lý do**: Các hệ thống thường thiếu kiểm tra ranh giới cho các lựa chọn.

**b. Thiết kế test case**

1. **Test Case 1: Kiểm tra trùng ghế**
   * **Mô tả**: Hai người dùng chọn cùng một ghế đồng thời.
   * **Dữ liệu đầu vào/Hành động**: Người dùng A và B đồng thời chọn ghế số 12 trên chuyến xe lúc 8:00 AM.
   * **Kết quả mong đợi**: Hệ thống chỉ cho phép một người dùng đặt ghế, hiển thị thông báo lỗi cho người còn lại: “Ghế đã được đặt.”
2. **Test Case 2: Kiểm tra hiển thị trạng thái ghế**
   * **Mô tả**: Kiểm tra trạng thái ghế sau khi đã được đặt.
   * **Dữ liệu đầu vào/Hành động**: Người dùng A đặt ghế số 10, sau đó người dùng B kiểm tra trạng thái ghế số 10.
   * **Kết quả mong đợi**: Ghế số 10 hiển thị trạng thái “Đã đặt” trên giao diện của người dùng B.
3. **Test Case 3: Kiểm tra chọn ghế không hợp lệ**
   * **Mô tả**: Kiểm tra khả năng chọn ghế ngoài phạm vi.
   * **Dữ liệu đầu vào/Hành động**: Người dùng cố chọn ghế số 50 (xe chỉ có 40 ghế).
   * **Kết quả mong đợi**: Hệ thống hiển thị thông báo lỗi: “Ghế không tồn tại.”

**Thực hiện Exploratory Testing**

**Tính năng**: Chọn ghế ngồi trong ứng dụng đặt vé xe buýt.

**a. Kế hoạch Exploratory Testing**

* **Mục tiêu kiểm thử**: Khám phá các lỗi liên quan đến giao diện và logic của tính năng chọn ghế.
* **Các khu vực tập trung**:
  + Giao diện hiển thị sơ đồ ghế (độ rõ ràng, trạng thái ghế, khả năng tương tác).
  + Phản hồi của hệ thống khi chọn ghế (đã đặt, trống, hoặc không hợp lệ).
  + Hiệu suất khi người dùng thao tác nhanh (chọn ghế liên tục hoặc chuyển đổi giữa các chuyến xe).
* **Thời gian dự kiến**: 30 phút.

**b. Dự đoán lỗi và cách ghi nhận**

1. **Lỗi 1: Giao diện hiển thị sai trạng thái ghế**
   * **Mô tả**: Ghế đã được đặt nhưng vẫn hiển thị là “trống” do lỗi đồng bộ hóa giao diện.
   * **Cách ghi nhận**: Ghi lại các bước thực hiện (chọn ghế, kiểm tra trạng thái), chụp ảnh màn hình giao diện, và báo cáo lỗi với chi tiết về chuyến xe và ghế.
2. **Lỗi 2: Hệ thống không phản hồi khi chọn ghế nhanh**
   * **Mô tả**: Khi người dùng nhấn nhanh vào nhiều ghế, hệ thống treo hoặc không phản hồi.
   * **Cách ghi nhận**: Ghi lại trình tự thao tác (nhấn nhanh vào 5 ghế trong 2 giây), quay video màn hình, và báo cáo lỗi với thông tin về thiết bị và phiên bản ứng dụng.

**Viết User Story và Áp dụng ATDD**

**Ứng dụng**: Ứng dụng quản lý công việc, tính năng “thêm công việc mới”.

**a. User Story**

**User Story**: Là một người dùng, Tôi muốn thêm một công việc mới vào danh sách công việc, Để theo dõi và quản lý các nhiệm vụ cần thực hiện.

**Tiêu chí chấp nhận**:

1. Người dùng phải nhập tiêu đề công việc (bắt buộc) và mô tả (không bắt buộc).
2. Hệ thống phải hiển thị thông báo lỗi nếu tiêu đề công việc để trống.
3. Công việc mới được thêm vào danh sách công việc với trạng thái “Chưa hoàn thành”.
4. Hệ thống phải cho phép chọn ngày hết hạn cho công việc (không bắt buộc).

**b. Thiết kế test case**

1. **Test Case 1: Thêm công việc với đầy đủ thông tin**
   * **Mô tả**: Kiểm tra thêm công việc với tiêu đề, mô tả và ngày hết hạn.
   * **Dữ liệu đầu vào**: Tiêu đề: “Hoàn thành báo cáo”, Mô tả: “Báo cáo quý 3”, Ngày hết hạn: 10/07/2025.
   * **Kết quả mong đợi**: Công việc được thêm vào danh sách với trạng thái “Chưa hoàn thành”.
2. **Test Case 2: Thêm công việc không có tiêu đề**
   * **Mô tả**: Kiểm tra trường hợp không nhập tiêu đề công việc.
   * **Dữ liệu đầu vào**: Tiêu đề: (trống), Mô tả: “Họp nhóm”, Ngày hết hạn: 10/07/2025.
   * **Kết quả mong đợi**: Hệ thống hiển thị thông báo lỗi: “Tiêu đề công việc không được để trống.”
3. **Test Case 3: Thêm công việc không có ngày hết hạn**
   * **Mô tả**: Kiểm tra thêm công việc không có ngày hết hạn.
   * **Dữ liệu đầu vào**: Tiêu đề: “Gọi khách hàng”, Mô tả: “Liên hệ khách hàng A”, Ngày hết hạn: (trống).
   * **Kết quả mong đợi**: Công việc được thêm vào danh sách với trạng thái “Chưa hoàn thành” và không có ngày hết hạn.

**c. Áp dụng ATDD**

**Quy trình áp dụng**:

1. **Product Owner**: Định nghĩa User Story và tiêu chí chấp nhận, làm việc với đội để đảm bảo yêu cầu rõ ràng.
2. **Tester**: Dựa trên tiêu chí chấp nhận, viết test case (như trên) và sử dụng công cụ như Cucumber hoặc FitNesse để tự động hóa kiểm thử.
3. **Developer**: Phát triển tính năng “thêm công việc mới” dựa trên các test case. Chạy test để đảm bảo mã đáp ứng tiêu chí.
4. **Hợp tác**: Nhóm họp định kỳ (ví dụ: trong sprint planning) để xem xét User Story, điều chỉnh tiêu chí chấp nhận nếu cần, và đảm bảo tất cả thành viên hiểu rõ yêu cầu.

**Vai trò của các thành viên**:

* **Product Owner**: Đảm bảo User Story phản ánh đúng nhu cầu người dùng.
* **Tester**: Xây dựng test case rõ ràng, kiểm tra tính năng và báo cáo lỗi.
* **Developer**: Viết mã đáp ứng test case và phối hợp với tester để sửa lỗi.

**Lợi ích**:

* Giảm thiểu hiểu lầm yêu cầu nhờ hợp tác chặt chẽ.
* Tăng chất lượng sản phẩm do kiểm thử được thực hiện từ đầu.
* Tiết kiệm thời gian sửa lỗi bằng cách phát hiện sớm các vấn đề.