1. So sánh Black Box Testing và White Box Testing dựa tên ưu điểm

|  |  |  |
| --- | --- | --- |
|  | **Black Box Testing** | **White Box Testing** |
| **Ưu điểm** | - Thích hợp trong việc kiểm tra từng phân đoạn lớn các mã lệnh, chức năng lớn - Người thử nghiệm không cần hiểu biết về mã lệnh được viết trong chương trình - Tách biệt giữa quan điểm của người sử dụng và người phát triển phần mềm | * - Thích hợp trong việc tìm kiếm lỗi và các vấn đề trong mã lệnh - Biết được yêu cầu bên trong của phần mềm, kiểm tra sẽ sát hơn - Cho phép tìm kiếm các lỗi ẩn bên trong - Các lập trình viên có thể tự kiểm tra - Giúp tối ưu việc mã hoá - Do yêu cầu kiến thức cấu trúc bên trong của phần mềm, nên việc kiểm soát lỗi tối đa nhất |
| **Nhược điểm** | - Độ bao phủ hạn chế vì chỉ có một phần nhỏ trong số các kịch bản thử nghiệm được thực hiện - Kiểm tra không hiệu quả do người thử nghiệm không hiểu biết gì về cấu trúc bên trong phần mềm. - Tester có hạn chế về hiểu biết về ứng dụng | - Không thể tìm thấy tính năng chưa thực hiện hoặc bỏ sót - Đòi hỏi hiểu sâu về cấu trúc bên trong của phần mềm được thử nghiệm - Yêu cầu truy xuất mã lệnh bên trong chương trình |

1. **Phân biệt Black Box Testing và White Box testing**

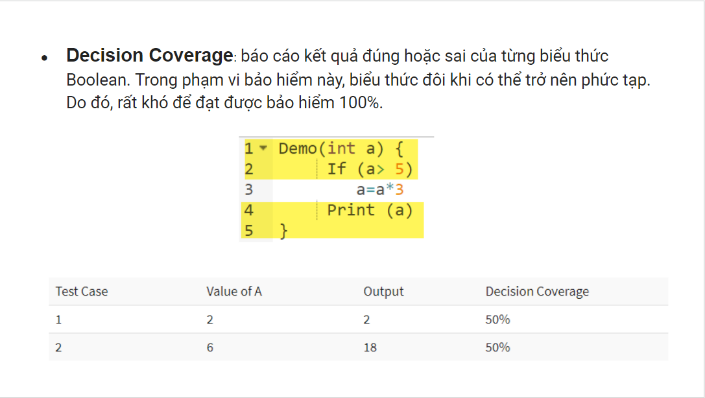
|  |  |  |
| --- | --- | --- |
| **Tiêu chuẩn** | **Black box test** | **White box test** |
| **1. Định nghĩa** | - Kiểm tra hộp đen là phương pháp thử nghiệm phần mềm được sử dụng để kiểm tra các phần mềm mà không quan tâm đến cấu trúc bên trong của chương trình. | - Kiểm tra hộp trắng là phương pháp kiểm thử phần mềm, sử dụng để kiểm tra phần mềm mà yêu cầu phải biết cấu trúc bên trong của chương trình. |
| **2. Trách nhiệm** | - Thử nghiệm được thực hiện bên ngoài, không liên quan đến nhà phát triển phần mềm. | - Thông thường, các thử nghiệm được thực hiện bởi nhà phát triển phần mềm. |
| **3. Cấp độ test sử dụng** | - Thử nghiệm áp dụng ở cấp độ cao như: kiểm tra hệ thống (System test), kiểm tra chấp nhận (Acceptance test) | - Thử nghiệm được áp dụng ở mức độ thấp hơn như thử nghiệm đơn vị (Unit Test), thử nghiệm hội nhập (Integration test) |
| **4. Biết lập trình** | - Không yêu cầu hiểu biết về Lập trình | - Yêu cầu hiểu biết nhất định về lập trình. |
| **5. Biết việc thực hiện chương trình** | - Không yêu cầu hiểu về cấu trúc bên trong chức năng, và không cẩn hiểu làm thế nào để có được chức năng đó | - Yêu cầu hiểu cấu trúc bên trong chức năng được thực hiện như nào. |
| **6. Cơ sở tạo Test Cases** | - Kiểm tra hộp đen được bắt đầu dựa trên tài liệu yêu cầu kỹ thuật | - Kiểm tra hộp trắng được bắt đầu dựa trên các tài liệu thiết kế chi tiết |

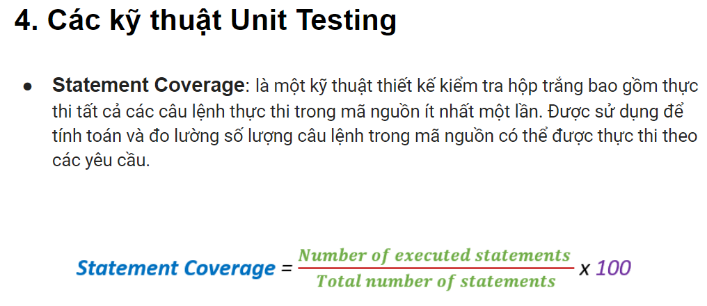
1. Phân biệt Functional Testing và Non Functional Testing

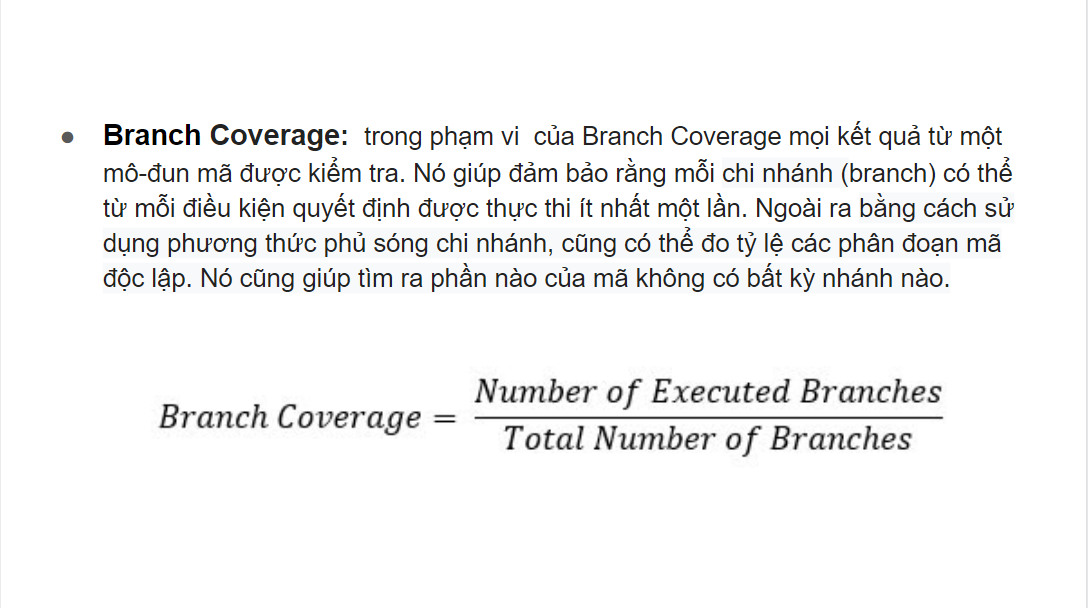
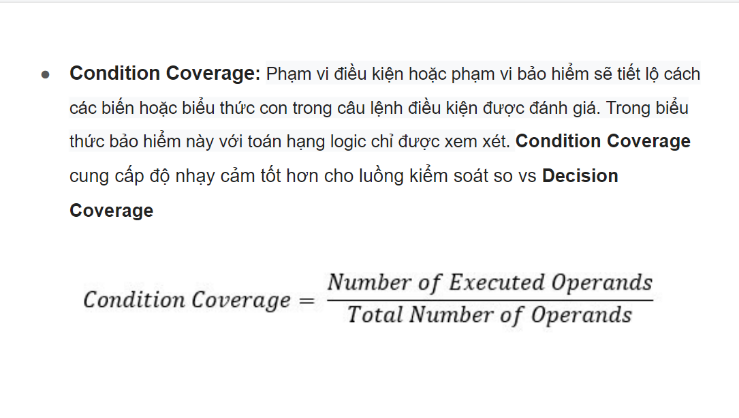
|  |  |  |
| --- | --- | --- |
| **Thông số** | **Functional Testing** | **Non-Functional Testing** |
| **Khái niệm** | Kiểm tra chức năng là một kiểu kiểm tra mà xác minh rằng mỗi chức năng của phần mềm hoạt động phù hợp với yêu cầu nêu ra.Việc kiểm tra này chủ yếu bao gồm kiểm tra hộp đen và nó không liên quan đến code của ứng dụng | kiểm tra phi chức năng là một kiểu kiểm tra để kiểm tra những mặt phi chức năng( sự trình bày, khả năng sử dụng, sự tin cậy,etc) của một ứng dụng phần mềm. Nó được thiết kế một cách rõ ràng để kiểm tra sự sẵn sàng của hệ thống theo các thông số phi chức năng khi mà mà không giải quyết được bằng kiểm tra chức năng |
| **Thực hiện** | Nó được thực hiện trước kiểm tra phi chức năng | Nó được thực hiện sau kiểm tra phi chức năng |
| **Vùng tập trung** | Nó đươc dựa trên yêu cầu của khách hàng | Nó dựa trên sự mong chờ của khách hàng |
| **Yêu cầu** | Dễ định nghĩa được chức năng yêu cầu | Khó định nghĩa được yêu cầu cho kiểm tra phi chức năng |
| **Sử dụng** | Giúp xác nhận cách ứng xử của ứng dụng | Giúp xác nhận sự thể hiện của ứng dụng |
| **Mục tiêu** | Thực hiện xác nhận hành động của phần mềm | xác nhận ( performance) của phần mềm |
| **Những yêu cầu** | Kiểm tra chức năng được xác nhận sử dụng chức năng | Loại kiểm tra này xác nhận các thông số kĩ thuật |
| **manual** | rất dễ dàng | khó khăn |
| **Ví dụ** | Kiểm tra chức năng đăng nhập | Bảng điều khiển( dashboard) nên tải trong 2 giây |

1. **So sánh Automation Testing và Manual Testing**

|  |  |  |
| --- | --- | --- |
| **Tiêu chuẩn** | **Automation Testing** | **Manual Testing** |
| **Định nghĩa** | Automation testing sử dụng các tool để thực hiện các trường hợp kiểm thử | Thực hiện kiểm thử hoàn toàn thủ công không có sự trợ giúp của bất kỳ công cụ tự động nào, được thực hiện bời tester |
| **Thời gian xử lý** | Thời gian kiểm thử rút ngắn hơn so với manual testing | Manual testing tốn nhiều thời gian và nguồn nhân lực |
| **Thử nghiệm thăm dò** | Không cho phép kiểm thử khám phá | Có thể kiểm thử khám phá trong manual testing |
| **độ tin cậy** | Kết quả kiểm thử đáng tin cậy vì nó được thực hiện bằng các tool và các kịch bản | Kết quả kiểm thử không đáng tin cậy vì có khả năng xảy ra lỗi của con người |
| **Thay đổi giao diện người dùng** | Chỉ là thay đổi nhỏ trong giao diện AUT nhưng các kịch bản kiểm thử tự động cần phải sửa đổi để hoạt động đúng như mong đợi | Những thay đổi nhỏ thư thay đổi về id, class sẽ không cản trở quá trình kiểm thử |
| **Đầu tư** | Cần phải đầu tư cho các công cụ kiểm thử | Cần đầu tư về nguồn nhân lực |
| **Kiểm tra báo cáo tầm nhìn** | Tất cả các bên liên quan có thể đăng nhập vào hệ thống xem kết quả đã kiểm thử | Kết quả được lưu lại trong excel hoặc word |
| **Kiểm tra năng suất** | Được thực hiện trong kiểm thử Load testing, stress testing | Không khả thi trong kiểm thử Load testing, stress testing |
| **Thi công song song** | Có thể thực hiện song song trên cấc nền tảng vận hành khác nhau và giảm thời gian thực hiện kiểm thử | Kiểm thử song song trên các nền tảng khác nhau sẽ phải tăng nguồn nhân lực |
| **Kiến thức lập trình** | Yêu cầu phải có kiến thức lập trình | Không cần có kiến thức lập trình vẫn có thể thực hiện |
| **Phương pháp lý tưởng** | Automation testing rất hữu ích khi thường xuyên thực hiện chạy lại một kịch bản nhiều nhiều lần | Manual testing hữu ích khi chạy bộ test case một hoặc hai lần |
| **Ưu điểm** | Sử dụng tool tự động giúp tìm kiếm được nhiều lỗi hơn  Automation testing nhanh và hiệu quả  Qúa trình kiểm thử được ghi lại, điều đó giúp chạy lại kịch bản kiểm thử nhiều lần và thực hiện trên nhiều nền tảng khác nhau  Automation testing được thực hiện bằng các công cụ phầm mềm, do đó nó hoạt động không mệt mỏi không giống như người kiểm thử tester  Automation testing năng suất và chính xác  Phạm vi kiểm thử rộng vì kiểm tra tự động không quên kiểm tra ngay cả đơn vị nhỏ nhất | Dễ dàng cho việc test giao diện, người tester sẽ có phản hồi nhanh và trực quan về giao diện ứng dụng  Mất ít chi phí cho các tool tự động và quy trình  Khi có thay đổi nhỏ manual testing manual testing không bị mất nhiều thời gian để thay đổi các trường hợp kiểm thử |
| **Nhược điểm** | Rất khó có cái nhìn đúng và trực quan về giao diện người dùng như màu sắc, font chữ, vị trí, kích thước các button nếu như không có yếu tố con người  Chi phí cho các tool kiểm thử có thể tốn kém, có thể làm tăng chi phí trong khâu kiểm thử của dự án  Nếu có một thay đổi nhỏ cũng sẽ mất thời gian để update kịch bản kiểm thử | Kết quả kiểm thử ít tin cậy hơn vì có thể sai xót do yếu tố con người  Qúa trình thực hiện các ca kiểm thử không được ghi lại, do vậy nó không có tính tái sử dụng  Với một số task khó thực hiện thủ công như performance testing và stress testing thì manual testing rất khó để thực hiện |
| **VD** |  |  |

1. **Unittest**: là mức độ kiểm thử nhỏ nhất trong quy trình kiểm thử phần mềm. Unit test kiểm thử các đơn vị nhỏ nhất trong mã nguồn như method, class, module...Do đó Unit test nhằm kiểm tra mã nguồn của các chương trình, các chức năng riêng rẽ hoạt động đúng hay không.





**Finite State Machine Coverage:** Đây chắc chắn là loại phương pháp bảo hiểm mã phức tạp nhất. Điều này là do nó hoạt động trên hành vi của thiết kế. Trong phương thức bảo hiểm này, cần tìm kiếm có bao nhiêu trạng thái cụ thể được truy cập, chuyển tiếp. Nó cũng kiểm tra có bao nhiêu chuỗi được bao gồm trong một máy trạng thái hữu hạn (finite state machine).

**Nhược điểm của Unit Test**

Kiểm thử đơn vị không thể phát hiện mọi lỗi trong một chương trình. Không thể đánh giá tất cả các đường dẫn thực hiện ngay cả trong các chương trình bình thường nhất

Kiểm thử đơn vị theo bản chất của nó tập trung vào một đơn vị mã. Do đó, nó không thể bắt lỗi tích hợp hoặc lỗi cấp hệ thống rộng.

1. **Integration testing:** Kiểm thử tích hợp (Integration testing) hay còn gọi là tích hợp và kiểm thử (integration and testing, viết tắt: I&T) là một giai đoạn trong kiểm thử phần mềm. Mỗi module phần mềm riêng biệt được kết hợp lại và kiểm thử theo nhóm.

Kiểm thử tích hợp xảy ra sau kiểm thử đơn vị (Unit Test) và trước kiểm thử hệ thống. Kiểm thử tích hợp nhận các module đầu vào đã được kiểm thử đơn vị, nhóm chúng vào các tập hợp lớn hơn, áp dụng các ca kiểm thử đã được định nghĩa trong kế hoạch kiểm thử tích hợp vào tập hợp đó, và cung cấp đầu ra cho hệ thống tích hợp

1. **System testing** là quá trình thử nghiệm một hệ thống phần cứng và phần mềm tích hợp với nhau một cách hoàn chỉnh để xác minh rằng hệ thống đáp ứng được các yêu cầu về chức năng và kỹ thuật (Sau khi Unit test và Intergration test)

VD: Toàn bộ chiếc xe cần được kiểm tra khả năng thỏa mãn theo các yêu cầu cụ thể như: vô lăng điều khiển có mượt mà, phanh xe, bánh xe và các chức năng khác có hoạt động bình thường?, xe liệu vẫn chạy bền bỉ sau 2500 dặm liên tục, màu xe có hài hòa, xe có thể lái được trên bất kỳ loại địa hình nào, từ bằng thẳng đến gập ghềnh hay không v.v… Toàn bộ quá trình kiểm tra này được gọi là kiểm thử hệ thống.

**Ưu Điểm:**

* Dễ dàng tự động hóa.
* Không cần truy cập mã nguồn, tester có kỹ năng vừa phải có thể kiểm tra mà không cần có nhiều kiến thức ngôn ngữ lập trình hoặc hệ điều hành.
* Phù hợp và hiệu quả khi số lượng các dòng lệnh của hệ thống lớn
* Có được sự đánh giá khách quan cao

**Nhược Điểm:**

* Bị giới hạn ở độ bao phủ các trường hợp kiểm thử.
* Sẽ không hiệu quả bởi các tester bị giới hạn kiến thức về hệ thống.

1. **Acceptance testing or Checklist (kiểm thử chấp nhận):** có thể được hiểu là một cách để kiểm tra xem "hợp đồng" đã được xác định trước đó giữa các nhà phát triển và khách hàng có đi đúng hướng hay không.

Liên quan đến nhu cầu, yêu cầu của người dùng và quy trình kinh doanh được tiến hành để xác định xem hệ thống có đáp ứng các tiêu chí chấp nhận hay không và cho phép người dùng, khách hàng hoặc tổ chức được ủy quyền khác xác định có chấp nhận hệ thống hay không.

VD: Facebook, Messenger cập nhật giao diện, thêm vào một số tính năng mới, họ sẽ để người dùng sử dụng ứng dụng dùng thử trước. Nếu người dùng cho nhận xét và feedback tích cực thì sẽ giữ đó, còn nếu feedback tiêu cực thì sẽ tiến hành cập nhật, sửa lỗi đến khi người dùng hài lòng.

1. **Alpha testing:** là một dạng của acceptance testing; Thực hiện để xác định tất cả các vấn đề/ lỗi có thể xảy ra trước khi phát hành sản phẩm đến tay người dùng. Trọng tâm của việc kiểm thử này là để mô phỏng người dùng thực – Real users bằng cách sử dụng các kỹ thuật Black box và white box. Mục đích là để thực hiện các nhiệm vụ mà một người sử dụng điển hình có thể thực hiện. Alpha testing được thực hiện trong môi trường lab và thường các tester là nhân viên nội bộ của tổ chức, công ty. Kiểu kiểm thử này được gọi là alpha vì nó được thực hiện sớm, gần cuối của sự phát triển của phần mềm, và trước khi thử nghiệm beta.

VD: 8/2 Đấu Trường Chân Lý Mobile đã chính thức ra mắt phiên bản Alpha Test đầu tiên, chỉ giới hạn một vài khu vực được trải nghiệm.

Đây là giai đoạn toàn bộ gamer trên thế giới sẽ trở thành tester.

Đội ngũ phát triển thông qua báo lỗi và phản hồi của gamer để tiếp tục hoàn thiện phần còn lại của game.

Thường thì nếu làm tốt 2 giai đoạn trên thì trong giai đoạn cuối này, đội ngũ phát triển sẽ ít gặp khó khăn.

**Ưu điểm:**

Cung cấp cách nhìn nhận tốt hơn về độ tin cậy của phần mềm trong giai đoạn đầu Giúp mô phỏng các hành vi thực của người dùng và mô phỏng môi trường.

Phát hiện nhiều lỗi nghiêm trọng

Có thể phát hiện sớm các lỗi về thiết kế và chức năng.

**Nhược điểm:**

Các chức năng không thể được kiểm tra vì phần mềm vẫn đang trong giai đoạn phát triển. Đôi khi developer và tester không hài lòng với kết quả của kiểm thử alpha.

1. **Beta test** một phần mềm được thực hiện bởi “người sử dụng thật – Real users” trong một “môi trường thực tế – Real environment” và có thể được coi là một hình thức acceptance testing bởi người dùng ngoài

VD: Tập đoàn Microsoft đã phát hành bản beta Windows 10 và dựa trên phản hồi từ hàng ngàn người dùng mà họ đã quản lý để phát hành một phiên bản hệ điều hành ổn định.

**Ưu điểm**

Thử nghiệm sản phẩm không thể kiểm soát được và người dùng có thể kiểm tra bất kỳ tính năng có sẵn nào theo bất kỳ cách nào - các khu vực góc được kiểm tra tốt trong trường hợp này

Giúp phát hiện ra các lỗi không được tìm thấy trong các hoạt động thử nghiệm trước đó (bao gồm cả alpha)

Quan điểm tốt hơn về cách sử dụng sản phẩm, độ tin cậy và bảo mật

Phân tích quan điểm và ý kiến của người dùng thực sự về sản phẩm

Phản hồi / đề xuất từ người dùng thực sự giúp ứng dụng sản phẩm trong tương lai

Giúp tăng sự hài lòng của khách hàng với sản phẩm

**Khuyết điểm**

Phạm vi được xác định có thể hoặc không được theo sau bởi người tham gia

Tài liệu tốn nhiều thời gian hơn - cần thiết để sử dụng công cụ ghi lỗi (nếu cần), sử dụng công cụ để thu thập phản hồi/ gợi ý, thủ tục kiểm tra (cài đặt/gỡ cài đặt, hướng dẫn sử dụng)

Không phải tất cả những người tham gia đều đảm bảo cung cấp thử nghiệm chất lượng

Không phải tất cả các phản hồi đều có hiệu lực - thời gian thực hiện để xem xét phản hồi cao

Quản lý kiểm tra khó

1. **A/B testing (hay còn được gọi là split testing) là** một quy trình mà trong đó hai phiên bản (A và B) sẽ được cùng so sánh trong một môi trường / tình huống được xác định và qua đó đánh giá xem phiên bản nào hiệu quả hơn. Phiên bản ở đây có thể là mọi thứ từ một hình banner, trang web, mẫu quảng cáo cho tới email và hiệu quả được đánh giá dựa trên mục tiêu của người làm test dành cho các phiên bản này.

VD: Chẳng hạn, một công ty công nghệ B2B có thể muốn cải thiện chất lượng và số lượng khách hàng tiềm năng từ các trang web của chiến dịch.

Để đạt được mục tiêu đó, một nhóm sẽ thử các thay đổi A/B Testing đối với các tiêu đề, hình ảnh trực quan, khung opt-in (biểu mẫu), CTA – kêu gọi hành động và bố cục tổng thể của trang.

Việc kiểm tra một thay đổi tại một thời điểm nhất định sẽ giúp họ xác định chính xác những thay đổi đó liệu có ảnh hưởng đến hành vi truy cập của người dùng hay có những thay đổi nào khác không.

Dần dần, dựa vào đó, họ có thể kết hợp hiệu ứng của nhiều thay đổi thành công từ các thử nghiệm trước để chứng minh sự cải thiện trải nghiệm mới so với trải nghiệm cũ.

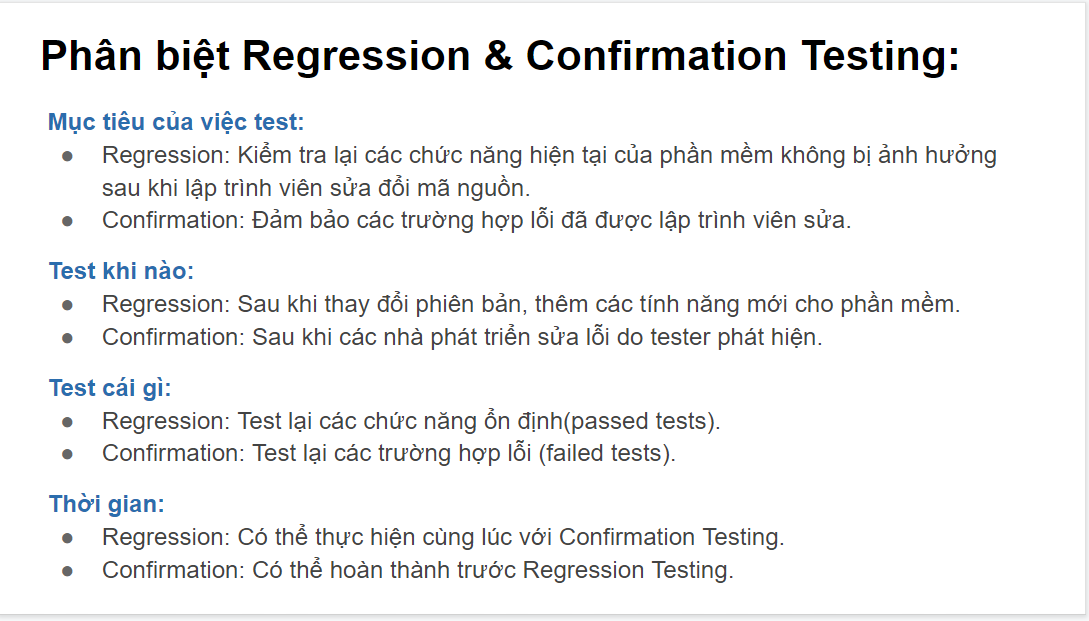
1. **Confirmation testing**  Sau khi một lỗi được phát hiện và sửa chữa, phần mềm được kiểm thử để xác nhận lỗi ban đầu đã được khắc phục chưa và sẽ kiểm tra lại để xem đã khám phá hết các lỗi chưa, quá trình này gọi là kiểm thử xác nhận (Confirmation testing).

VD: rong lúc làm đồ án môn học, sau khi viết xong 1 số chức năng hoặc xong chương trình, chúng ta phải test các chức năng của chương trình.

Giả sử có 100 test case để test, và trong 100 test case đó có 5 test case gây bug cho chương trình => Kết quả không như mong muốn => fix 5 test case gây bug.

Sau khi fix được các test case gây bug =>kiểm tra lại 5 test case gây lỗi. Công việc test lại 5 test case gây bug chính là Confirmation testing.

1. **Phân biệt Regression và Confirmation Testing**



1. **Phân biệt Re-testing và Regression Testing**

| **Regression Testing** | **Re-Testing** |
| --- | --- |
| Regresstion Testing được thực hiện nhằm xác nhận một chương trình hoặc một thay đổi mã gần đây không làm ảnh hưởng đến các chức năng hiện có | Re-testing được thực hiện nhằm đảm bảo các test case bị lỗi đã được pass trong bản build cuối cùng sau khi lỗi được fix |
| Mục đích của Regression Testing là những sự thay đổi mã không làm ảnh hưởng đến những chức năng đã tồn tại | Re-testing được thực hiện trên cơ sở các bản sửa lỗi |
| Xác minh lỗi không phải là một phần của Regression Testing | Xác minh lỗi là một phần của Re-testing |
| Dựa trên dự án và nguồn lực sẵn có, Regresstion Testing có thể thực hiện song song với Re-testing | Ưu tiên của Re-testing cao hơn regression testing, vì nó được thực hiện trước khi kiểm thử hồi quy |
| Bạn có thể thực hiện kiểm thử tự động trong Regression Testing, manual testing có thể tốn kém và tốn thời gian | Bạn không thể thực hiện kiểm thử tự động với Re-testing |
| Regression Testing là thử nghiệm chung | Re-testing là thử nghiệm có kế hoạch |
| Regression Testing thực hiện trên các test case đã passed | Re-testing thực hiện trên các test case failed |
| Regression Testing kiểm tra những ảnh hưởng không mong muốn | Re-testing đảm bảo rằng những lỗi ban đầu đã đúng |
| Regression Testing chỉ được thực hiện khi có bất kỳ sự sửa chữa hoặc thay đổi nào được thực hiện trong project hiện có | Re-testing thực thi một lỗi với dữ liệu và môi trường giống nhau với những đầu vào khác nhau với một bản build mới |
| Test case của Regression Testing có thể thu được từ spec, hướng dẫn sử dụng, và báo cáo lỗi liên quan đến những vấn đề đã sửa | Test case của Re-testing không thể được xác định trước khi bắt đầu test |

1. **Performance Testing:** là một loại kiểm thử nhằm xác định mức độ đáp ứng, băng thông, độ tin cậy, khả năng mở rộng của hệ thống dưới một khối lượng làm việc, truy cập nhất định. Đảm bảo cho các ứng dụng hoặc phần mềm có thể hoạt động hiệu quả trong khoảng chức năng dự kiến của ứng dụng.

VD: Các trang thương mại điện tử cần lưu trữ rất nhiều thông tin. Để đánh giá được hiệu năng của sản phẩm, lập trình viên quyết định tạo ra db với hàng triệu bản ghi, sau đó thực hiện query, cập nhật dữ liệu trên hệ thống để mô phỏng các hoạt động thực tế của trang thương mại điện tử. Từ đó đánh giá được hiệu năng thực của sản phẩm.

1. **GUI testing** là quá trình thử nghiệm Giao diện đồ họa người dùng trong các ứng dụng để đảm bảo đáp ứng được các yêu cầu và đánh giá khả năng sử dụng của nó. GUI testing bao gồm việc kiểm tra màn hình với các đối tượng như menu, button, icon, text box, … GUI là những gì người dùng nhìn thấy. Khi bạn ghé thăm một trang web, bạn sẽ thấy GUI của trang vì một người dùng bình thường sẽ không thể xem mã nguồn của trang.

**Đặc điểm**

GUI là một giao diện đồ họa có thứ bậc, phân cấp cho ứng dụng, chứa các đối tượng đồ họa với một tập các thuộc tính.

Trong quá trình thực hiện, các giá trị của các thuộc tính của từng đối tượng của một GUI xác định trạng thái GUI.

GUI có khả năng thực hiện các sự kiện GUI như nhấn phím / bấm chuột.

GUI có khả năng cung cấp đầu vào cho các đối tượng GUI.

GUI phụ thuộc mạnh vào công nghệ được sử dụng.

1. **Usability testing(Kiểm tra khả năng sử dụng)** Kiểm tra tính khả dụng là một kỹ thuật kiểm thử hộp đen để xác định sản phẩm của bạn có thân thiện với người dụng hay không.

Kiểm tra tính khả dụng để tạo ra một sản phẩm làm cho khách hàng mua ngay từ cái nhìn đầu tiên/lần sử dụng đầu tiên.

Ví dụ: Khách hàng mua hàng trực tuyến ở website

– Khi thực hiện ở hệ thống A phải trải qua các bước sau:

+ Step 1: Nhập thông tin mua hàng

+ Step 2: Nhập thông tin ship hàng

+ Step 3: Nhập thông tin thanh toán

+ Step 4: Xác nhận đơn hàng

– Khi thực hiện mua hàng ở hệ thống B trải qua các bước như sau:

+ Step 1: Nhập thông tin mua hàng, ship hàng, thanh toán

+ Step 2: Xác nhận đơn hơn

=> Như vậy, với hệ thống B người dùng sẽ tiết kiệm thời gian hơn vì việc setting các thông tin cần tiết cho đơn hàng được thực hiện trên cùng một step, một giao diện. Hệ thống B sẽ có tính khả dụng cao hơn hệ thống A.

Ưu điểm:

Tìm ra những lỗi quan trong và những lỗi của ứng dụng mà bên phát triển không nhìn ra được

Khắc phục tất cả các vấn đề mà người dùng sẽ gặp phải trước khi phát hành ứng dụng

Kiểm thử khả năng sử dụng có thể được sửa đổi tùy theo yêu cầu để hỗ trợ các loại thử nghiệm khác nhau như thử nghiệm chức năng, thử nghiệm tích hợp hệ thống, unit test...

Khuyết điểm:

Quá trình lập kế hoạch và thu thập dữ liệu tốn nhiều thời gian.

Kích thước nhỏ và đơn giản của nó làm cho nó không đáng tin cậy để rút ra kết luận về sở thích người dùng chủ quan.

Khó để tạo ra bối cảnh phù hợp. Không thể kiểm tra kinh nghiệm lâu dài. Các kết nối xã hội ngoài kế hoạch không thể được nhân rộng. Mọi người hành động theo một cách khác khi họ biết rằng họ đang được quan sát.

1. **Security Testing** là việc tìm kiếm tất cả các lỗ hổng có thể và điểm yếu của hệ thống mà có thể dẫn đến mất thông tin trong tay nhân viên hoặc người ngoài của tổ chức.

VD: HTTP là một giao thức văn bản rõ ràng, nếu các dữ liệu nhạy cảm như thông tin người dùng được truyền qua HTTP, thì đó là một mối đe dọa cho bảo mật ứng dụng. Thay vì HTTP, dữ liệu nhạy cảm sẽ được chuyển qua HTTPS (bảo mật thông qua SSL, đường hầm TLS). Tuy nhiên, HTTPS làm tăng bề mặt tấn công và do đó cần kiểm tra xem cấu hình máy chủ có phù hợp và hiệu lực chứng chỉ được đảm bảo hay không.

1. **Kiểm tra cài đặt** là kiểm tra xem ứng dụng phần mềm đã được cài đặt thành công chưa và nó có hoạt động như mong đợi sau khi cài đặt không.

Đây là giai đoạn thử nghiệm trước khi người dùng cuối trước tiên sẽ tương tác với ứng dụng thực tế.

Kiểm tra cài đặt cũng được gọi là Kiểm tra triển khai thực hiện.

Đây là bước quan trọng nhất cũng như thú vị nhất trong vòng đời thử nghiệm Phần mềm

Ví dụ: cài đặt phần mềm theo nhiều cách khác nhau và trên các loại hệ thống khác nhau mà phần mềm có thể được cài đặt. Kiểm tra những tập tin được thêm hoặc thay đổi trên đĩa. Phần mềm đã cài đặt có hoạt động không? Điều gì xảy ra khi bạn gỡ cài đặt ?

1. **Recovery Testing** : kiểm thử phục hồi là kiểm thử được thực hiện sau khi có các sự cố hệ thống dẫn đến chương trình lỗi, không hoạt động được. Kiểm thử phục hồi được thực hiện để đảm bảo chương trình sau khi khắc phục lỗi trên không xảy ra bug.

Kiểm tra khôi phục là khi bạn buộc lỗi phần mềm của mình để xác minh rằng việc khôi phục từ lỗi là có thể và được thực hiện đúng. Mục tiêu, rõ ràng, là để phục hồi nhanh chóng.

VD: Khi một ứng dụng đang nhận dữ liệu từ mạng, hãy rút cáp kết nối.

Sau một thời gian, cắm lại cáp và phân tích khả năng tiếp tục nhận dữ liệu của ứng dụng từ điểm kết nối mạng bị hỏng.

Khởi động lại hệ thống trong khi trình duyệt có số phiên nhất định mở và kiểm tra xem trình duyệt có thể khôi phục tất cả chúng hay không.

1. **Load testing** là một quá trình thêm nhu cầu vào một hệ thống hoặc thiết bị và đo lường phản ứng của nó  
   - Load testing được thực hiện để xác định ứng xử của hệ thống trong các điều kiện tải bình thường và cao hơn điều kiện tải dự kiến.  
   - Nó giúp xác định công suất vận hành tối đa của một ứng dụng như các điểm “thắt cổ chai” (bottleneck) và xác định phần tử nào là nguyên nhân gây ra điều đó.   
   - Load test được thực hiện vào cuối vòng đời phát triển phần mềm.

**Ưu điểm:**

Các vấn đề liên quan đến hiệu suất và tắc nghẽn có thể được xác định trước

Cải thiện khả năng mở rộng của hệ thống

Giảm thiểu rủi ro liên quan quan đến thời gian chết hệ thống

Giảm chi phí thất bại

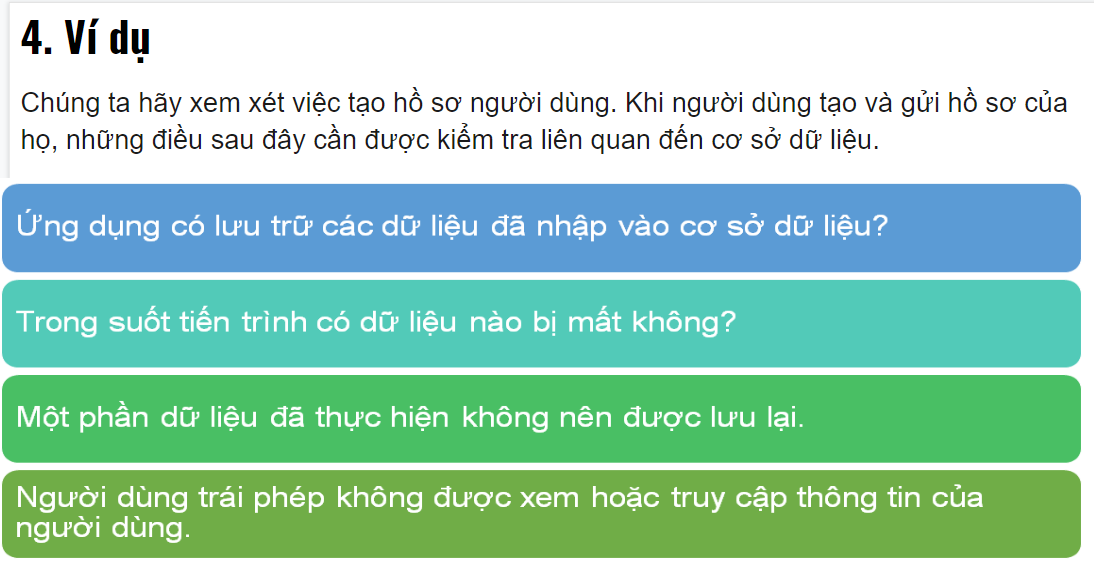
Tăng sự hài lòng của khách hàng

**Nhược điểm:**

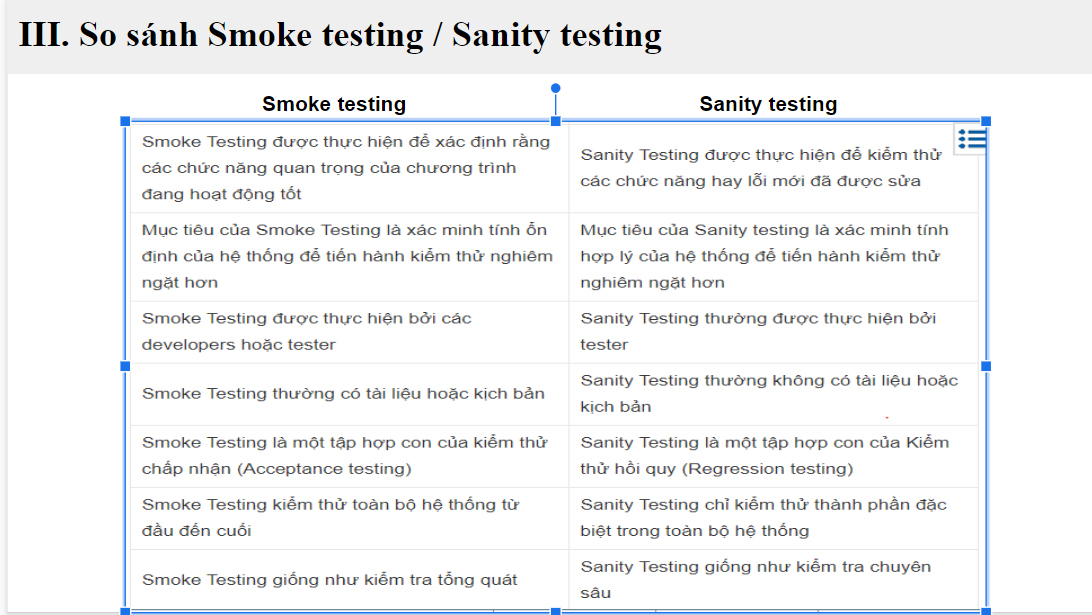
Cần kiến thức lập trình để sử dụng các công cụ kiểm tra tải

 Các công cụ có thể tốn kém vì giá cả phụ thuộc vào số lượng người dùng ảo được hỗ trợ

1. **Database Testing** là quá trình kiểm tra độ chính xác và tính toàn vẹn của cơ sở dữ liệu (CSDL). Cơ sở dữ liệu ở đây như:  lược đồ (schema), bảng biểu( tables), sự kích hoạt/ khởi động( triggers)... Đảm bảo rằng các dữ liệu là chính xác và duy nhất.



1. **Sanity testing** là một loại Kiểm thử phần mềm được thực hiện sau khi nhận bản build của phần mềm, với các thay đổi nhỏ trong code hoặc chức năng, để xác định rằng các lỗi đã được sửa và không có lỗi nào được đưa ra do những thay đổi này nữa.
2. **Smoke Testing** là một loại kiểm thử phần mềm được thực hiện sau khi xây dựng phần mềm để xác định rằng các chức năng quan trọng của chương trình đang hoạt động tốt.
3. So sánh Smoke testing và Sanity testing



VD Sanity testing: Trong ứng dụng ví dụ bên trên, ở bản build trước có phát hiện ra lỗi liên quan đến việc gửi hồ sơ do phân quyền sai nên hồ sơ được gửi đi nhưng lại gửi không đúng người nhận, ở bản build này bug này đã được sửa, sanity test ở đây sẽ kiểm tra nhanh việc chỉnh sửa chức năng này có liên quan đến các chức năng, vai trò khác hay không? Nhưng vẫn cần phải lưu ý rằng mình sẽ chỉ kiểm tra những phần có liên quan nhất, không sa đà chi tiết quá vào việc kiểm thử chức năng, giao diện gửi hồ sơ hay những chức năng đã ổn định khác, vì thời gian dành cho sanity test là không nhiều.

VD Smoke testing: Trong một ứng dụng web, có các chức năng như đăng nhập, trang chủ, tạo mới hồ sơ, gửi hồ sơ, quản lý hồ sơ,… Để có thể thực hiện được các thao tác trên hệ thống bạn cần phải đăng nhập vào ứng dụng thành công, tuy nhiên sau bản build thì một vài vấn đề xảy ra làm cho người dùng không thể login vào hệ thống được. Hay là đăng nhập được nhưng chức năng tạo mới hay gửi hồ sơ đi không thực hiện được. Đây là những tính năng quan trọng của hệ thống, vì vậy việc thực hiện smoke test trước khi bàn giao sẽ phát hiện sớm và fix sớm những vấn đề cơ bản này, thay vì phải bàn giao, thực hiện cài cắm cấu hình các thứ xong rồi bắt đầu kiểm thử thì mới phát hiện ra, gây lãng phí khá nhiều thời gian.

1. Giải thích ý nghĩa thuật ngữ Ramp-up time trong công cụ kiểm thử hiệu năng Jmeter.Nếu Number of thread = 1000(theard) và Ramp-up time = 10 giây điều đó có nghĩa là gì ?

Ramp-up time : là tổng thời gian thực hiện tất cả các yêu cầu(request) của bạn

Number of thread = 1000(theard) và Ramp-up time = 10 giây điều đó có nghĩa là có 1000 yêu cầu và cứ 10/1000 giây Jmeter sẽ gửi 1 yêu cầu

1. Mẹo Unittest

So sánh kq khi nhập giá trị vào và kq mong đợi

Assert.AreEqual( kq mong đợi , Hàm(gt nhập vào) ) => khác thì lỗi , giống thì ko lỗi ( ngược lại với Assert.AreNotEqual())