[100+ KHÁI NIỆM TEST DÀNH CHO TESTER]

**1. Acceptance Testing**: Khách hàng thực hiện (hoặc ủy quyền cho một nhóm thứ ba thực hiện). Mục đích của Acceptance Test là để chứng minh PM thỏa mãn tất cả yêu cầu của khách hàng và khách hàng chấp nhận sản phẩm.

**2. Accessibility Testing**: Là một loại kiểm thử dành cho các hệ thống được thiết kế cho người khuyết tật (khiếm thính, khiếm thị, thiểu năng tâm thần...), xác định khả năng tiếp cận và sử dụng của họ với ứng dụng, ví dụ như các thiết lập Accessibility của smart phone: Text-to-Speech, "Magnification gestures"... Người khuyết tật thường sử dụng những tính năng hỗ trợ họ sử dụng các phần mềm. Loại test này có thể được thực hiện theo 2 cách là Manual và Automated.

**3. Active Testing**: Loại kiểm thử bao gồm việc đưa ra dữ liệu test và phân tích kết quả thực hiện. Nó thường được tiến hành bởi đội tester. Trong quá trình active testing, tester sẽ hình dung để dựng một mô hình của phần mềm để test, cái này sẽ được phát triển và làm hoàn thiện hơn các tương tác của nó với những phần mềm đang chạy.

**4. Agile Testing**: Kiểm thử Agile là việc kiểm thử phần mềm được thực hiện theo một số quy định của bản tuyên ngôn (manifesto) Agile, xem việc phát triển phần mềm như là khách hàng của việc kiểm thử. Kiểm thử Agile thực hiện kiểm thử theo quan điểm của khách hàng càng sớm càng tốt, thử nghiệm sớm và thường xuyên ngay khi code vừa xong và đủ ổn định để test từ level unit test.

**5. Age Testing:** Loại kiểm thử đánh giá khả năng hoạt động của hệ thống trong tương lai. Quá trình đánh giá được thực hiện bởi đội tester. Age Testing đo đạc sự hao hụt về performance của hệ thống khi hệ thống PM bị cũ đi.

**6. Ad-hoc Testing**: Việc thực hiện test mà không có plan, không có tài liệu, tester cố gắng phá vỡ hệ thống bằng cách thử ngẫu nhiên các chức năng của hệ thống. Việc này được thực hiện bởi đội tester.

**7. Alpha Testing**: Loại kiểm thử phần mềm hoặc hệ thống được tiến hành bởi người dùng cuối nhưng thực hiện ngay tại chỗ của đội developer.

**8. Assertion Testing**: Là việc xác minh xem các điều kiện có đáp ứng được các yêu cầu về logic hoặc business của sản phẩm hay không. Nó đảm bảo cho việc test và các mục tiêu test luôn liên quan đúng và rõ ràng.

**9. API Testing**:

* API (Application Programming Interface): cho phép kết nối và trao đổi dữ liệu giữa hai hệ thống phần mềm riêng biệt. Một hệ thống phần mềm có thể nhúng các API bao gồm các hàm/thủ tục con (functions/sub-routines) mà có thể chạy bởi một hệ thống phần mềm khác.
* Kiểm thử API khác hoàn toàn với kiểm thử GUI và các thành phần chủ yếu khác trong tầng business logic của kiến trúc phần mềm. Loại kiểm thử này không tập trung vào phần giao diện và thao tác giao diện của một ứng dụng. Thay vì sử dụng các đầu vào (bàn phím) và đầu ra tiêu chuẩn, trong kiểm thử API, bạn có thể sử dụng một phần mềm để gửi các yêu cầu đến API, nhận đầu ra và ghi lại phản hồi của hệ thống.

**10. All-pairs Testing**: All pair testing (Kiểm thử tất cả các cặp) hay còn gọi là pairwise testing, là một phương pháp test được thực hiện để kiểm thử các phần mềm sử dụng phương pháp tổ hợp. Đó là một phương pháp để kiểm tra tất cả các kết hợp rời rạc có thể có của các thông số liên quan, phương pháp test ít nhất sao cho chất lượng tốt nhất.

**11. Automated Testing**: Kiểm thử tự động sử dụng công cụ (tool) để kiểm soát môi trường thiết lập, thực hiện kiểm tra và báo cáo kết quả. Nó được thực hiện bởi máy tính và được sử dụng trong nội bộ team test.

**12. Basis Path Testing**: Thuộc phương pháp kiểm thử hộp trắng (White box test) là kĩ thuật kiểm thử mà phần mềm được chia thành các lộ trình, đảm bảo các lộ trình độc lập qua một module mã sẽ được kiểm thử đầy đủ. Thực hiện bởi developer.

**13. Backward Compatibility Testing**: Kiểm tra sự tương thích ngược: là việc kiểm tra xem các sản phẩm của ứng dụng cũ có tiếp tục làm việc tốt trên phiên bản mới của ứng dụng đó hay không.

**14. Benchmark Testing**: Là việc xác định perfomance hiện tại của hệ thống và thực hiện các thay đổi để cải thiện perfomance của sản phẩm cho tốt hơn. Như việc refactor code, tinh chỉnh cơ sở dữ liệu để người dùng có thể trải nghiệm những cải tiến hiệu suất. Được thực hiện bởi đội dev hoặc Database admin (DBAs).

**15. Beta Testing:** Là giai đoạn test cuối cùng trước khi bàn giao ứng dụng, ứng dụng hoặc phần mềm được phân bổ như một phiên bản thử nghiệm (sử dụng thử) để người dùng kiểm tra ứng dụng tại nơi làm việc của họ. Người sử dụng sẽ quan sát phần mềm, trong trường hợp nếu có bất kỳ lỗi xảy ra thì nó được báo cáo đến người phát triển.

**16. Big Bang Integration Testing:** Khi mà tất cả mọi thứ sẵn sàng thì kĩ thuật kiểm tra này thực hiện để tích hợp các module độc lập của chương trình.

**17. Binary Portability Testing**: Kĩ thuật test tính di động của phần mềm bằng cách chạy phần mềm trên các nền tảng và môi trường khác nhau. Nó được sử dụng cho cấu tạo của Application Binary Interface (ABI). Binary Portability testing nên được tiến hành trên các loại khác nhau của platform, như Windows(x86, X86-64), Linux, Mac OS, Java, Solaris, and Android. Nếu ứng dụng có tính di động cao thì người dùng có thể chạy ứng dụng trên bất kì nền tảng (platform) nào. Do đó để test Binary portability thì tức là khi xây dựng 1 phần mềm thì cần test việc nó thực thi được ứng dụng đó trên nhiều hệ điều hành khác nhau, nếu nó là website thì cần chạy được trên nhiều trình duyệt khác nhau để kiểm tra tính di động của nó. Được thực hiện bởi nhóm test.

**18. Boundary Value Testing**: Kĩ thuật kiểm thử phần mềm trong đó test case được thiết kế để bao gồm những giá trị giới hạn tiêu biểu. Được thực hiện bởi đội test.

**19. Bottom Up Integration Testing**: Trong kiểm thử tích hợp từ dưới lên, mô-đun ở mức thấp nhất được phát triển đầu tiên và các mô-đun khác đi theo hướng chương trình chính được tích hợp và kiểm thử cùng một lúc. Thực hiện bởi nhóm kiểm thử.

**20. Branch Testing**: Kỹ thuật kiểm thử, trong đó tất cả các nhánh trong mã nguồn của chương trình sẽ được kiểm tra ít nhất một lần. Thực hiện bởi các developer.

**21. Breadth Testing**: Là việc kiểm tra bằng các test suit thực hiện đầy đủ các chức năng của một sản phẩm nhưng không kiểm tra từng tính năng chi tiết. Người thực hiện: Các đội kiểm thử.

**22. Black box Testing**: Một phương pháp kiểm thử phần mềm xác minh các chức năng của một ứng dụng mà không có kiến thức cụ thể về mã / cấu trúc bên trong của ứng dụng. Các kiểm thử được dựa trên yêu cầu và chức năng. Thực hiện bởi team QA.

**23. Code-driven Testing**: Kỹ thuật kiểm thử sử dụng framework để kiểm thử (như xUnit) cho phép thực hiện các kiểm thử đơn vị (unit test) để xác định xem các phần khác nhau của mã đang hoạt động như mong đợi trong những tình huống khác nhau. Thực hiện bởi developers.

**24. Compatibility Testing**: Kỹ thuật kiểm thử xác nhận làm thế nào một phần mềm thực hiện tốt trong một phần cứng / phần mềm / hệ điều hành/ môi trường hệ thống / môi trường mạng. Thực hiện bởi các nhóm kiểm thử.

**25. Comparison Testing**: Kỹ thuật kiểm thử so sánh điểm mạnh và điểm yếu của sản phẩm với các phiên bản trước đó hoặc các sản phẩm tương tự khác. Thực hiện bởi nhân viên kiểm thử, các nhà phát triển, quản lý sản phẩm hoặc chủ sở hữu sản phẩm.

**26. Component Testing**: Kỹ thuật kiểm thử tương tự với kiểm thử đơn vị, nhưng với một mức độ kiểm thử tích hợp cao hơn được thực hiện kiểm thử các ứng dụng thay vì chỉ trực tiếp kiểm thử một phương thức cụ thể. Thực hiện bởi đội kiểm thử hoặc đội phát triển.**27. Configuration Testing**: Kỹ thuật kiểm thử quyết định cấu hình tối thiểu và tối ưu của phần cứng và phần mềm, hiệu quả của việc thêm hoặc sửa đổi các nguồn tài nguyên như bộ nhớ, ổ đĩa và CPU. Thực hiện bởi các kỹ sư kiểm thử hiệu năng.

**28. Condition Coverage Testing**: Loại kĩ thuật kiểm thử phần mềm mà mỗi điều kiện được gán 2 giá trị đúng và sai ít nhất 1 lần. Thực hiện bởi các nhóm kiểm thử tự động.

**29. Compliance Testing**: Loại kiểm thử mà kiểm tra xem hệ thống đã được xây dựng phù hợp với các tiêu chuẩn, thủ tục và hướng dẫn hay không. Người thực hiện: Các công ty bên ngoài, các công ty cung cấp thương hiệu "chứng nhận tuân thủ OGC ".

**30. Concurrency Testing**: Kiểm thử đa người dùng hướng tới việc xác định ảnh hưởng của việc tiếp cận cùng một mã ứng dụng, mô-đun hoặc cơ sở dữ liệu. Thực hiện bởi các kỹ sư hiệu năng.

**31. Conformance Testing**: Quá trình kiểm thử một sản phẩm hoặc hệ thống phù hợp với đặc điểm kỹ thuật của spec. Thực hiện bởi các đội kiểm thử.

**32. Context Driven Testing**: Một kỹ thuật kiểm tra Agile, khuyến khích các tester tự lựa chọn test techniques, test deliverable, test documentation và test objectives, theo chủ trương đánh giá liên tục và sáng tạo các cơ hội kiểm thử của những thông tin tiềm năng, các giá trị của thông tin để tổ chức kiểm thử tại một thời điểm cụ thể. Người thực hiện: Các đội kiểm thử Agile.

**33. Conversion Testing**: Được thực hiện khi chuyển đổi dữ liệu từ hệ thống hiện có sang hệ thống mới thay thế. Kiểm thử các chương trình hoặc các thủ tục được sử dụng để chuyển đổi dữ liệu từ hệ thống hiện có để sử dụng trong các hệ thống thay thế. Người thực hiện: Đội ngũ QA.

**34. Decision Coverage Testing**: Loại hình kiểm thử phần mềm mà mỗi quyết định được thực hiện ít nhất một lần đảm bảo rằng tất cả các code được thực hiện. Mỗi quyết định được thực hiện bằng cách đặt vào 2 giá trị đúng và sai. Thực hiện bởi dev hoặc các nhóm kiểm thử tự động hóa.

**35. Destructive Testing**: Loại kiểm thử trong đó các kiểm thử được tiến hành với các dữ liệu kiểm thử thất bại , để hiểu về hiệu suất ,hành vi quan trọng của một mẫu kiểm thử theo tải trọng khác nhau. Thực hiện bởi đội ngũ QA.

**36. Dependency Testing**: Loại kiểm thử trong đó xem xét các yêu cầu của một ứng dụng cho các phần mềm tiền đề, trạng thái ban đầu và cấu hình để duy trì chức năng thích hợp. Thực hiện bởi các đội kiểm thử.

**37. Dynamic testing**: Kiểm thử liên quan đến việc thực thi các thành phần hoặc toàn bộ hệ thống phần mềm. Thường được thực hiện bởi đội test.

**38. Domain testing**: Là loại kiểm thử kiểm tra những thông tin mà người dùng sử dụng để nhập vào trên các vùng dữ liệu, kiểm tra các kết quả nhận được và xem xét chúng có đúng không. Thường được thực hiện bởi đội phát triển phần mềm và team test automation.

**39. Error handling testing**: Loại kiểm thử phần mềm xác định trách nhiệm của hệ thống xử lý các giao dịch có lỗi một cách thích hợp. Thường được thực hiện bởi đội test.

**40. End - to- end testing**: Tương tự với system testing, liên quan đến việc kiểm thử trong môi trường tương tự với môi trường sử dụng thật, ví dụ tương tác với DB, sử dụng giao tiếp mạng, hoặc tương tác với các phần cứng, ứng dụng, hoặc hệ thống nếu phù hợp. Được thực hiện bởi đội QA.

**41. Endurance testing**: Loại test để check việc thiếu bộ nhớ hoặc các vấn đề có thể xảy ra khi ứng dụng được thực hiện trong thời gian dài. Thường được thực hiện bởi đội hiệu năng.

**42. Exploratory testing**: Kỹ thuật kiểm thử hộp đen thực hiện mà không cần plan và tài liệu. Thường được thực hiện bởi tester manual.

**43. Equivalence Partitioning testing**: Kỹ thuật kiểm thử phần mềm chia dữ liệu đầu vào thành các phân vùng nhỏ hơn. Thường được đội QA sử dụng.

**44. Fault injection testing**: Là kỹ thuật mà tester tập trung vào phương pháp mà hệ thống xử lý ngoại lệ thông qua việc truyền mã độc vào hệ thống. Được thực hiện bởi đội QA.

**45. Formal verification testing**: Kiểm định hình thức theo kiểu chứng minh định lý (Theorem Proving), kiểm tra logic hệ thống xem có phù hợp với spec không. Thường được thực hiện bởi đội QA.

**46. Functional testing**: Loại kiểm thử hộp đen dựa trên test case và spec. Được thực hiện bởi đội kiểm thử.

**47. Fuzz testing:** Kỹ thuật kiểm thử phần mềm dùng các dữ liệu input invalid, không mong muốn hoặc random - nó là trường hợp đặc biệt của mutation testing. Fuzz testing được thực hiện bởi đội test.

**48. Gorilla testing**: Là kỹ thuật mà tester thực hiện test lại phần đã test một vài lần trước đó để đảm bảo tính chắc chắn của hệ thống Được thực hiện bởi tester hoặc developer.

**49. Gray box testing**: Là sự kết hợp của kiểm thử hộp đen và kiểm thử hộp trắng. Test phần mềm dựa trên spec nhưng lại sử dụng cả các kiến thức bên trong về hoạt động của code. Được thực hiển bởi dev và test.

**50. Glass box testing**: Nó khá giống với kiểm thử hộp trắng. Nó cũng sử dụng kiến thức về xử lý code bên trong để test. Được đội dev thực hiện là chủ yếu.

**51. GUI software testing**: Test giao diện người dùng của sản phẩm để đảm bảo đáp ứng được các yêu cầu. Thường được thực hiện bởi đội test.

**52. Globalization testing**: Phương pháp kiểm thử kiểm tra các chức năng thích hợp của sản phẩm trong điều kiện thiết lập culture / locale sử dụng tất cả các loại đầu vào quốc tế có thể. Nó được thực hiện bởi đội test.

**53. Hybrid integration testing**: Kết hợp 2 kỹ thuật kiểm thử tích hợp từ trên xuống và tích hợp từ dưới lên để tận dụng những ưu điểm của cả 2 kỹ thuật đó. Thường được thực hiện bởi đội test.

**54. Intergration testing**: Integration Testing là công việc kiểm thử tích hợp 1 nhóm các module riêng lẻ với nhau. Thường được thực hiện bởi nhóm kiểm thử.

**55. Interface Testing**: Kiểm thử giao diện là việc kiểm tra các yếu tố liên quan đến giao diện người dùng. Được thực hiển bởi tester hoặc nhóm phát triển.

**56. Install/ Uninstall Testing:**

* Kiểm thử cài đặt: Được thực hiện để xác minh liệu rằng các phần mềm đã được cài đặt với tất cả các thành phần cần thiết và các ứng dụng đang làm việc như mong đợi.
* Kiểm thử gỡ bỏ cài đặt: Được thực hiện để xác minh liệu rằng tất cả các thành phần của ứng dụng có bị loại bỏ trong quá trình hay không. Tất cả các tập tin liên quan đến ứng dụng cùng cấu trúc thư mục của nó phải được gỡ bỏ sau khi quá trình gỡ bỏ thành công.
* Được thực hiện bởi các kỹ sư kiểm thử phần mềm.

**57. Internationalization Testing**: Quá trình đảm bảo chắc chắn các chức năng của sản phẩm không bị phá vỡ và tất cả các thông điệp được đưa ra ngoài chính xác khi được sử dụng trong các ngôn ngữ và miền địa phương khác nhau. Được thực hiện bởi đội kiểm thử.

**58. Inter-system Testing**: Kỹ thuật kiểm thử tập trung vào kiểm thử các ứng dụng nhằm đảm bảo các kết nối giữa chức năng ứng dụng luôn chính xác. Thường được thực hiện bởi các nhóm kiểm thử.

**59. Keyword driven testing**: Keyword Driven Testing (hay còn gọi là as table-driven testing hoặc action word based testing) là một kỹ thuật lập trình sử dụng các tập tin dữ liệu không chỉ chứa các dữ liệu kiểm thử và kết quả mong đợi, mà còn chứa các từ khóa liên quan đến ứng dụng đang được kiểm thử. Thực hiện bởi nhóm kiểm thử thủ công và tự động.

**60. Load Testing**: 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. Được thực hiện bởi các kỹ sư hiệu năng.

**61. Localization Testing**: Localization testing là quá trình kiểm thử nhằm mục đích kiểm tra các phiên bản địa phương hóa của sản phẩm đặc trưng cho một nền văn hóa hoặc một địa phương cụ thể. Được thực hiện bởi tester. Bạn đọc có thể tham khảo thêm để biết sự khác biệt giữa Globalization và Localization Testing tại link sau: [http://www.guru99.com/globalization-vs-localization-testing.html](https://l.facebook.com/l.php?u=http%3A%2F%2Fwww.guru99.com%2Fglobalization-vs-localization-testing.html%3Ffbclid%3DIwAR2VMR2ZHl7EOq1stw6nSukcIsuRQjWVjph3S2ukBL7riQ5iuMPOg2j9Rqs&h=AT3tpfFyEYXN45s9gTIKQjh6JOwB7tNdlY0oIgnP8NwPiCuvBY16CNtfZbLIbqHz7FBzGqRSKvO8qBil32b9S3mAa5-hvMbeFqfRxER4ZtSYrJXUc_2e0SMl0ZTGc3sPK5Ke&__tn__=-UK-y-y-R&c%5b0%5d=AT3M_cAwQKu3SmOrbLqE_PbAmbm1qsjUpGmxl_KyTDhmcmoESr8sVe090uTD2D14XQ-1Zi5hYanIzzzYmVGdf-t30e-qT97K89sZFngIVFuNWMHtp9NBbwYJuIDKu_lzDk-N75v_q7EvbTzhMINkWd84emDRM-vf58shrK1rvSKfBUbWt3UPiEVQnoBlfjy8q2tH90BsOho5T1_orbkOMOQIp-H0CDYnmz-z7WVQ)

**62. Loop Testing:**

* Kỹ thuật kiểm thử hộp trắng tạo ra một chu trình vòng lặp.
* Kiểm tra vòng lặp hoàn toàn tập trung vào tính hiệu lực của các cấu trúc vòng lặp. Nó là một trong những phần kiểm soát kiểm tra cơ cấu (kiểm tra đường dẫn, kiểm tra xác nhận dữ liệu, kiểm tra điều kiện).
* Được thực hiện bởi các kỹ sư phát triển.

**63. Manual Scripted Testing**: Kỹ thuật kiểm thử trong đó các TCs được thiết kể và xem xét bởi nhóm kiểm thử trước khi thực hiện chúng. Được hoàn thành bởi nhóm kiểm thử thủ công.

**64. Manual Support Testing**: Kiểm thử hỗ trợ thủ công là loại kiểm thử các chức năng hỗ trợ bằng tay. Nó đạt hiệu quả nhất trong giai đoạn cài đặt sản phẩm. Được thực hiện bởi nhóm kiểm thử.

**65. Model based Testing**: MBT là thế hệ tự động của quy trình kiểm thử phần mềm sử dụng mô hình hành vi và yêu cầu hệ thống. Được thực hiện bởi nhóm kiểm thử.

**66. Mutation Testing**: Kiểm thử hoán chuyển là phương pháp kiểm thử cấu trúc phần mềm nhằm mục đích đánh giá/cải thiện tính đầy đủ của điều kiện test và ước tính số lượng lỗi có thể phát sinh với hệ thống trong quá trình test. Được thực hiện bởi tester và developer.

**67. Modularity driven testing**: Kiểm thử mô đun là một framework kiểm thử tự động trong đó các mô đun nhỏ, độc lập của kịch bản tự động hóa được phát triển cho ứng dụng dưới quá trình kiểm thử. Được thực hiện bởi Tester.

**68. Non-functional Testing**: Kiểm thử phi chức năng đề cập đến các khía cạnh của phần mềm có thể không liên quan đến một chức năng cụ thể hoặc hành động người dùng, chẳng hạn như khả năng mở rộng và hiệu suất khác, hành vi dưới những hạn chế hoặc bảo mật nhất định. Có thể được thực hiện bởi các kỹ sư hiệu năng hoặc đội test.

**69. Negative Testing**: Để đảm bảo hệ thống chạy trơn tru và ổn định, chúng ta chỉ test những trường hợp bình thường và hợp lệ là chưa đủ. Vì vậy, để đảm bảo hệ thống chạy có thể xử lý được những trường hợp ngoại lệ ta cần test thêm những ngữ cảnh không hợp lệ. Việc test những ngữ cảnh không hợp lệ gọi là negative testing. Được thực hiện bởi nhóm kiểm thử thủ công hoặc tự động.

**70. Operational Testing**: Kiểm thử chấp nhận hoạt động (OAT) là một kỹ thuật kiểm thử được tiến hành nhằm xác nhận sự sẵn sàng hoạt động (trước khi release) của sản phẩm hoặc ứng dụng dưới quá trình test. Kỹ thuật này chủ yếu dựa trên sự sẵn sàng hoạt động của hệ thống sao cho gần giống với môi trường production. Được thực hiện bởi Tester.

**71. Orthogonal Array Testing**: Kiểm thử mảng trực giao là một loại kiểm thử hộp đen - kiểm tra các trường hợp tối ưu hóa kỹ thuật được sử dụng khi hệ thống được thử nghiệm có các dữ liệu đầu vào khổng lồ.

**72. Pair testing**: Kiểm thử theo cặp là cách tiếp cận một quá trình kiểm thử thiết kế bởi có hai tester kiểm tra được điều tương tự tại cùng một thời gian và địa điểm, liên tục trao đổi ý tưởng. Loại kiểm thử này có thể được kết hợp bởi Tester - Developer hoặc Tester - Business Analyst hoặc 2 testers.

**73. Passive Testing:** Kiểm tra kỹ thuật: bao gồm trong việc theo dõi kết quả của một hệ thống đang chạy mà không tác động vào hệ thống. Được thực hiện bởi tester.

**74. Performance Testing**: Kiểm thử hiệu năng là một phương tiện để đảm bảo chất lượng (QA), được thực hiện để xác định hệ thống thực hiện một khối lượng công việc cụ thể nhanh thế nào. Có thể dùng để xác nhận và xác minh những thuộc tính chất lượng khác của hệ thống như là khả năng mở rộng, độ tin cậy và sử dụng tài nguyên.

**75. Parallel Testing**: Kiểm thử song song là kiểm tra khả năng tương thích của hệ thống mới được phát triển với hệ thống cũ. Thực hiện bởi Tester.

**76. Path Testing**: Kiểm tra đường dẫn là một phương pháp kiểm tra cấu trúc liên quan đến việc sử dụng mã nguồn của một chương trình để tìm ra tất cả các đường dẫn thực thi tốt. Thực hiện bởi developer.

**77. Penetration Testing**: Kiểm thử thâm nhập là một loại kiểm tra an ninh dùng để kiểm tra các khu vực không an toàn của hệ thống hoặc ứng dụng. Phương pháp mà đánh giá sự an toàn của một hệ thống máy tính hoặc mạng bằng cách mô phỏng một cuộc tấn công từ một nguồn độc hại. Tạo ra bởi công ty kiểm thử thâm nhập chuyên ngành.

**78. Qualification Testing**: Là loại kiểm thử với mục đích kiểm tra lại các thông số kỹ thuật của phiên bản trước, thường được thực hiện bởi dev cho người tiêu dùng, để chứng minh rằng các phần mềm đáp ứng yêu cầu quy định của nó.

**79. Ramp Testing**: Loại thử nghiệm bao gồm việc nâng cao một tín hiệu đầu vào liên tục cho đến khi hệ thống bị phá vỡ. Có thể được thực hiện bởi tester hoặc dev.

**80. Regression Testing**: Kiểm thử hồi quy là loại kiểm thử phần mềm tìm cách để phát hiện ra các lỗi phần mềm sau khi thay đổi chương trình (ví dụ như sửa lỗi hoặc chức năng mới) đã được thực hiện, bởi việc kiểm tra lại chương trình.

*Các bạn có thể đọc đầy đủ bài viết tại đây:*[*https://viblo.asia/p/100-khai-niem-test-danh-cho-tester-roaerJOxeRM*](https://l.facebook.com/l.php?u=https%3A%2F%2Fviblo.asia%2Fp%2F100-khai-niem-test-danh-cho-tester-roaerJOxeRM%3Ffbclid%3DIwAR0C0s7hZzZyCBqMq2pj7sGP9wqdSaQb6TUmlwtv-qQjmSMYrwPYH1dWyio&h=AT0gyUrtDub3-96MVE6A0XPCaQALyYdbKWE3puVi_PZu3oXTBjUVZDP4_5uwnigo_BVNoRPSf5FwhAksaOw5WJzsNM85qdDcE9MmRJ1uOFRqqJIfnepbNSoIo0uYqG5nwE8T&__tn__=-U-UK-y-y-R&c%5b0%5d=AT3M_cAwQKu3SmOrbLqE_PbAmbm1qsjUpGmxl_KyTDhmcmoESr8sVe090uTD2D14XQ-1Zi5hYanIzzzYmVGdf-t30e-qT97K89sZFngIVFuNWMHtp9NBbwYJuIDKu_lzDk-N75v_q7EvbTzhMINkWd84emDRM-vf58shrK1rvSKfBUbWt3UPiEVQnoBlfjy8q2tH90BsOho5T1_orbkOMOQIp-H0CDYnmz-z7WVQ)