1. Unit Test (1)

**Unit Test (Kiểm thử đơn vị)** là một loại kiểm thử phần mềm trong đó thực hiện kiểm thử từng đơn vị hoặc thành phần riêng lẻ của phần mềm. Mục đích của việc Unit Test là để xác nhận rằng mỗi đơn vị hay mã code của phần mềm thực hiện chức năng của chúng đúng như mong đợi.Kiểm thử đơn vị được thực hiện trong quá trình phát triển (giai đoạn thực hiện code) của một ứng dụng và được thực hiện bởi các kỹ sư phần mềm.

1. Integration Test (2)

**Integration Test (Kiểm thử tích hợp)** là một giai đoạn trong quy trình kiểm thử phần mềm, trong đó các module riêng lẻ sau khi đã được kiểm thử đơn vị sẽ được kết hợp và kiểm tra chung. Quá trình này diễn ra sau kiểm thử đơn vị và trước kiểm thử hệ thống. Mục tiêu của kiểm thử tích hợp là đảm bảo rằng các module có thể hoạt động cùng nhau một cách chính xác. Các module đầu vào đã qua kiểm thử đơn vị sẽ được nhóm lại thành các cụm lớn hơn, sau đó áp dụng các kịch bản kiểm thử theo kế hoạch kiểm thử tích hợp. Kết quả của giai đoạn này giúp chuẩn bị hệ thống sẵn sàng cho kiểm thử hệ thống.

1. System Test (3)

**System Test (Kiểm thử hệ thống)** là quá trình đánh giá và theo dõi hoạt động của một sản phẩm hoặc hệ thống phần mềm đã được tích hợp đầy đủ, dựa trên các yêu cầu và đặc tả kỹ thuật đã được xác định trước. Mục tiêu của kiểm thử hệ thống là xác định xem hệ thống hoàn chỉnh có đáp ứng đúng các yêu cầu đặt ra hay không. Phương pháp này thuộc loại kiểm thử hộp đen, nghĩa là chỉ tập trung kiểm tra các chức năng bên ngoài của phần mềm mà không xem xét cách thức hoạt động bên trong của nó.

1. Acceptance Test (4)

**Acceptance Test (Kiểm thử chấp nhận)** là quá trình đánh giá hệ thống phần mềm để xác định liệu nó có đáp ứng đầy đủ các yêu cầu kỹ thuật và mong đợi của khách hàng hay không. Bằng cách kiểm tra hành vi của hệ thống với dữ liệu thực tế, kiểm thử chấp nhận giúp xác định hệ thống có thỏa mãn các tiêu chí đặt ra hay không. Một số kỹ thuật thường được sử dụng trong giai đoạn này bao gồm phân tích giá trị biên, phân vùng tương đương và bảng quyết định.

1. Stress / Load Test (5)

**Stress Test (Kiểm thử căng thẳng)** là một dạng kiểm thử hiệu suất nhằm đánh giá khả năng chịu tải của hệ thống hoặc ứng dụng phần mềm khi hoạt động vượt quá mức tải thông thường. Mục tiêu của kiểm thử này là xác định hệ thống có thể duy trì hoạt động ổn định trong điều kiện khắc nghiệt hay không. Bằng cách tạo ra áp lực cao hơn mức bình thường, kiểm thử căng thẳng giúp phát hiện giới hạn chịu tải của hệ thống và đảm bảo rằng nó vẫn có thể đáp ứng nhu cầu của người dùng trong những tình huống cực đoan.

**Load Test (Kiểm thử tải)** là quá trình đánh giá hiệu suất của hệ thống bằng cách tăng dần khối lượng công việc hoặc số lượng người dùng truy cập để xác định giới hạn chịu tải của hệ thống. Đây là một dạng kiểm thử hiệu suất nhằm đo lường cách hệ thống phản hồi và hoạt động dưới các mức tải khác nhau. Mục tiêu của kiểm thử tải là xác định hiệu suất tối đa mà hệ thống có thể xử lý ổn định trước khi hiệu suất bắt đầu suy giảm.

1. Performance Test (6)

**Kiểm thử hiệu năng (Performance Testing)** là quá trình đánh giá khả năng đáp ứng, độ tin cậy và khả năng mở rộng của hệ thống dưới các điều kiện tải trọng khác nhau. Mục đích là phát hiện các vấn đề hiệu suất, tối ưu hóa hệ thống và đảm bảo phần mềm đáp ứng các yêu cầu về hiệu năng. Các loại kiểm thử hiệu năng bao gồm: Load test, Stress test, Capacity test, Endurance test, Spike test, Volume test, Scalability test và Reliability test.

1. Install / Uninstall Test

**Kiểm thử cài đặt (Installation Testing)** là quá trình kiểm tra các quy trình cài đặt để xác nhận phần mềm có thể được thiết lập và vận hành đúng cách. Nó bao gồm kiểm tra cài đặt, nâng cấp, gỡ bỏ phần mềm, các tệp phụ thuộc, thư viện, cấu hình phần cứng, và khả năng kết nối. Thường được thực hiện trong giai đoạn Kiểm thử chấp nhận vận hành (Operational Acceptance Testing) bởi các kỹ sư kiểm thử và quản lý cấu hình.

**Kiểm thử gỡ cài đặt (Uninstallation Testing)** là quá trình kiểm tra nhằm đảm bảo tất cả thành phần của phần mềm, bao gồm tệp tin và cấu trúc thư mục, được xóa hoàn toàn khi thực hiện gỡ cài đặt. Sau khi gỡ cài đặt, hệ thống phải trở về trạng thái ổn định như trước khi phần mềm được cài đặt.

1. Recovery Test (1)

**Kiểm thử phục hồi (Recovery Testing)** là quá trình đánh giá khả năng khôi phục của hệ thống sau sự cố. Nó giúp đảm bảo rằng hệ thống có thể tiếp tục hoạt động bình thường sau sự cố, xác định thời gian phục hồi, tỷ lệ phục hồi thành công, khả năng khôi phục dữ liệu bị mất và khả năng kết nối lại của người dùng. Kiểm thử phục hồi rất quan trọng đối với các hệ thống yêu cầu khôi phục nhanh chóng. Việc kiểm thử thường được thực hiện trong môi trường thử nghiệm gần giống với hệ thống thực tế.

1. Compatibility Test (2)

K**iểm thử tương thích (Compatibility Testing)** kiểm tra khả năng tương thích của ứng dụng với các hệ điều hành, trình duyệt, phần cứng, phần mềm và môi trường mạng khác nhau. Mục đích là đảm bảo ứng dụng hoạt động chính xác và ổn định trong các môi trường khác nhau, đồng thời giảm thiểu các lỗi khi thay đổi hệ thống hoặc triển khai sản phẩm mới. Các loại kiểm thử tương thích bao gồm kiểm thử phần mềm, phần cứng, hệ điều hành, trình duyệt, mạng và thiết bị, cùng với kiểm thử phiên bản (tương thích ngược và tương thích chuyển tiếp).

1. Comparison Test

**Comparison Test** là một phương pháp kiểm thử phần mềm nhằm so sánh sản phẩm đang phát triển với các sản phẩm tương tự trên thị trường. Nhằm là đánh giá điểm mạnh, điểm yếu và cải thiện hiệu suất về các khía cạnh như giao diện, tốc độ, bảo mật và chức năng của phần mềm để giúp nó cạnh tranh tốt hơn. Kiểm thử so sánh không có giai đoạn cố định và có thể thực hiện ở bất kỳ giai đoạn nào của vòng đời phát triển phần mềm.

1. Conversion Test

**Conversion Test** là kỹ thuật kiểm thử đo lường tác động của việc thay đổi các thành phần trên trang web đối với tỷ lệ chuyển đổi, như thay đổi hình nền, vị trí nút bấm hoặc hình ảnh. Mục tiêu là xác định xem các thay đổi có giúp tăng tỷ lệ chuyển đổi (tỷ lệ người dùng thực hiện hành động mong muốn như mua hàng, tải tài liệu,... với hệ thống) hay không. Kỹ thuật này giúp tối ưu hóa giao diện và trải nghiệm người dùng. Đây là công cụ quan trọng để cải thiện hiệu suất trang web dựa trên dữ liệu thực tế.

1. Usability Test

**Usability Test** là quá trình đánh giá sản phẩm để xác định mức độ dễ sử dụng và tiện lợi cho người dùng. Kiểm thử này bao gồm ba điểm quan trọng: đánh giá sự dễ dàng và thoải mái khi người dùng hoàn thành nhiệm vụ, quan sát người dùng trong quá trình thao tác, và mục tiêu là kiểm tra sản phẩm, không phải người dùng. Quá trình này giúp phát hiện vấn đề về giao diện và khả năng tương tác. Từ đó, cải thiện trải nghiệm người dùng.

1. Configuration Test

​ **Configuration Testing** là kỹ thuật kiểm thử phần mềm nhằm đánh giá hiệu suất của ứng dụng trên nhiều cấu hình phần cứng và phần mềm khác nhau. Mục tiêu là xác định cấu hình tối ưu về sự tương thích giữa các phiên bản hệ điều hành, trình duyệt, driver, kích thước bộ nhớ, loại ổ đĩa và các loại CPU khác nhau và đảm bảo ứng dụng hoạt động ổn định trên các môi trường đa dạng. Thực hiện configuration testing giúp đảm bảo trải nghiệm người dùng nhất quán và giảm thiểu lỗi do sự không tương thích giữa các cấu hình.

1. Black box / functional testing

**Black box testing:** kiểm thử phần mềm mà không cần biết về cấu trúc bên trong. Tester tập trung vào đầu vào và đầu ra của hệ thống để xác minh chức năng. **Functional testing** là một dạng của black box testing, kiểm tra xem hệ thống có hoạt động theo yêu cầu không.

1. White box / structural testing

**White box testing** kiểm thử dựa trên cấu trúc và logic bên trong của mã nguồn. Tester cần hiểu rõ cách thức hoạt động của hệ thống và phải có kiến thức về lập trình.

**Structural testing** là cách tiếp cận dựa trên kiến trúc phần mềm, giúp tối ưu mã và phát hiện lỗi logic. Kiểm thử tập trung vào việc kiểm tra mã nguồn, đảm bảo tất cả được thực thi và hoạt động đúng

1. Sanity testing

**Sanity testing** là kiểm thử sơ bộ để xác định các chức năng chính của ứng dụng có hoạt động đúng sau khi cập nhật hay sửa lỗi không. Nó tập trung vào một phần cụ thể của hệ thống thay vì kiểm thử toàn bộ. Thường thực hiện nhanh chóng để quyết định có nên tiếp tục kiểm thử hay không.

1. Regression testing

**Regression testing** kiểm tra lại hệ thống sau khi có thay đổi để đảm bảo không ảnh hưởng đến các tính năng cũ. Nó giúp phát hiện lỗi phát sinh do sửa lỗi hoặc cập nhật tính năng mới.

1. **Incremental testing (3, 4)**

**Incremental testing** (kiểm thử tích hợp tăng tiến) là một trong các phương pháp thực hiện kiểm thử tích hợp, giúp giảm thiểu rủi ro và phát hiện lỗi sớm. Trong quá trình kiểm thử các thành phần của hệ thống sẽ được kiểm tra riêng lẻ trước, sau đó tích hợp, tổng hợp dần để kiểm thử một cách tổng thể. Có hai loại kiểm thử incremental testing là bottom up (kiểm từ dưới lên trên) và top down (kiểm từ trên xuống dưới).

1. **Thread testing**

**Thread testing** là một kỹ thuật kiểm thử phần mềm được sử dụng trong giai đoạn kiểm thử tích hợp để kiểm tra khả năng hoạt động của các chức năng chính theo từng luồng xử lý (thread). Loại kỹ thuật này rất hữu ích khi kiểm thử ứng dụng có sử dụng kiến trúc client server, đa luồng... Các tester thực hiện kiểm thử này để đảm bảo các luồng hoạt động một cách độc lập, phối hợp chính xác giữa các luồng với nhau.

1. **Alpha Test**

**Alpha Test** là loại kiểm thử phần mềm hoặc hệ thống được tiến hành bởi nội bộ của nhóm phát triển dự án trước khi phần mềm được phát hành ra bên ngoài. Mục đích của việc kiểm thử là cần xác định xem phần mềm còn xảy ra lỗi khi chạy thử hay không để từ đó đánh giá và xác định được mức độ đạt chuẩn dựa theo yêu cầu của khách hàng.

1. **Beta Test (5, 6)**

**Beta Test** là giai đoạn kiểm thử cuối cùng trước khi bàn giao và phát hành ứng dụng. Ở giai đoạn này, phần mềm sẽ được đưa cho nhóm người dùng bên ngoài nội bộ của dự án để có thể trải nghiệm và đánh giá, sau đó sẽ thu thập các phản hồi của nhóm người dùng về tính ổn định, hiệu suất trong quá trình sử dụng thực tế để xác định các lỗi còn chưa phát hiện trong quá trình Alpha Test, đảm bảo chất lượng trước khi phát hành.