Bài tập tuần

1. Stability test.
2. Khái niệm Stability test

Stability testing là khả năng duy trì hoạt động của sản phẩm xuyên suốt và vượt quá thời hạn sử dụng của nó, mà không hỏng hoặc xảy ra lỗi. Đây là 1 kỹ thuật non-functional, với mục đích đòi hỏi khả năng chịu tải của phần mềm tới mức tối đa. Trong quá trình xác định nó hoạt động tốt thế nào dưới tải ở mức chấp nhận được, mức đỉnh, các tải được tạo ra đột ngột, với số lượng dữ liệu lớn được xử lý… Stability testing được thực hiện để kiểm tra hiệu quả của 1 sản phẩm được phát triển vượt qua mức hoạt động bình thường, hay tới 1 điểm dừng. Có ý nghĩa quan trọng hơn là trong việc xử lý lỗi, độ tin cậy của phần mềm, khả năng chịu tải và khả năng mở rộng của 1 sản phẩm dưới tải lớn chứ không phải là kiểm tra cách hoạt động của hệ thống trong các hoàn cảnh bình thường.

## Chuyện gì sẽ xảy ra nếu 1 hệ thống thử nghiệm chưa trải qua Stability testing?

* Đối với 1 ứng dụng đang được thử nghiệm nơi mà 1 số lượng lớn user được giới thiệu và các ứng dụng phải chạy hàng tháng mà không cần khởi động lại, 1 số vấn đề có thể xảy ra:
* Các lỗi có thể phải đối mặt như: • Hệ thống chạy chậm lại • Hệ thống gặp các vấn đề về function • Hệ thống hiển thị wired behavior • Hệ thống bị treo
* Do đó, Stability testing thường liên quan đến việc thực thi của hệ thống với các heavy user (ảo) và đo các tham số hiệu năng dể xác minh liệu hệ thống có thể hỗ trợ tải dự tính hay không

## Tại sao phải thực hiện Stability testing?

Loại testing này giúp user hiểu được cách thức hệ thống sẽ làm việc trong các tình huống thực tế.

Do đó, Stability testing cho phép bạn kiểm tra:

* Ước lượng độ tin cậy trong sự ổn định của hệ thống của bạn đang thử nghiệm
* Đảm bảo rằng hệ thống của bạn có thể xử lý các chương trình lớn
* Theo dõi hiệu quả của hệ thống của bạn
* Kiểm tra hệ thống ổn định theo yêu cầu

Nó đóng 1 vai trò quan trọng trong phát triển sản phẩm vì nó được sử dụng để xác định những hạn chế của 1 sản phẩm phần mềm đang thử nghiệm trước khi nó được released hoặc những phạm vi cải tiến hơn trước khi sản phẩm goes live hay ở Production. 1 ví dụ rất phổ biến của kỹ thuật Stability testing là

Các cổng mua sắm trực tuyến: Stability testing sẽ kiểm tra cách các website sẽ hoạt động khi:

* Lượng dữ liệu nhập vào lớn ở giờ cao điểm
* Số lượt truy cập ở 1 thời điểm nhất định
* Vấn đề page load ở cùng 1 thời điểm
* Cách vận hành của hệ thống
* Phản ứng của hệ thống và nhiều hơn nữa buộc phải trải nghiệm dưới Stability testing

## Kết luận:

Stability testing là 1 quá trình non-functional của việc test hành vi của các ứng dụng bằng cách áp dụng tải tối đa. Nó được thực hiện để tìm ra khả năng mở rộng của hệ thống trong 1 môi trường nhất định

## Automation test

## Khái niệm.

Trong lĩnh vực kiểm thử phần mềm, thì kiểm thử tự động hay còn gọi là Automation testing đóng một vai trò quan trọng góp phần nâng cao năng suất kiểm thử, giảm thiểu lỗi cũng như sự nhàm chán với việc kiểm thử bằng tay trong một thời gian dài hoặc lặp đi lặp lại.

Kiểm thử tự động là một quá trình xử lý tự động các bước thực hiện một test case. Kiểm thử tự động được thực hiện bởi phần mềm kiểm thử tự động - hay còn gọi là Automation Testing Tool. Một số phần mềm kiểm thử tự động nổi tiếng hiện nay như:

* Quick Test Profressional - (HP)
* Selenium
* Test Architect - (LogiGear)
* Ranorex
* Visual Studio CodedUI Testing
* TestComplete (SmartBear)
* SOAPUI - Web Services Testing (SmartBear)

## Ưu, nhược điểm của automation test so với manual test

### Ưu điểm:

* Độ tin cậy cao: công cụ kiểm thử tự động có sự ổn định cao hơn so với con người, đặc biệt trong trường hợp nhiều test cases, nên độ tin cậy cao hơn so với kiểm thử thủ công.
* Khả năng lặp: công cụ kiểm thử tự động ra đời là để giúp cho các tester không phải lặp đi lặp lại các thao tác (ví dụ: nhập dữ liệu, click, check kết quả…) một cách nhàm chán với độ tin cậy và ổn định cao.
* Khả năng tái sử dụng: với một bộ kiểm thử tự động, người ta có thể sử dụng cho nhiều phiên bản ứng dụng khác nhau, đây được gọi là tính tái sử dụng.
* Tốc độ cao: do thực thi bởi máy nên tốc độ của kiểm thử tự động nhanh hơn nhiều so với tốc độ của con người. Nếu cần 5 phú để thực thi một test case một cách thủ công thì có thể người ta chỉ cần khoảng 30s để thực thi một cách tự động.
* Chi phí thấp: nếu áp dụng kiểm thử tự động đúng cách, người ta có thể tiết kiệm được nhiều chi phí, thời gian và nhân lực, do kiểm thử tự động nhanh hơn nhiều so với kiểm thử thủ công, đồng thời nhân lực cần để thực thi và bảo trì scripts không nhiều.

### Nhược điểm:

* Khó mở rộng, khó bảo trì: trong cùng một dự án, để mở rộng phạm vi cho kiểm thử tự động khó hơn nhiều so với kiểm thử thủ công vì cập nhật hay chỉnh sửa yêu cầu nhiều công việc như debug, thay đổi dữ liệu đầu vào và cập nhật code mới.
* Khả năng bao phủ thấp: do khó mở rộng và đòi hỏi nhiều kỹ năng lập trình nên độ bao phủ của kiểm thử tự động thấp xét trên góc nhìn toàn dự án.
* Vấn đề công cụ và nhân lực: hiện nay cũng có nhiều công cụ hỗ trợ kiểm thử tự động khá tốt nhưng chúng vẫn còn nhiều hạn chế. Ngoài ra nhân lực đạt yêu cầu (có thể sử dụng thành thạo các công cụ này) cũng không nhiều.

## Để trở thành Automation Tester thì bắt đầu từ đâu?

Một Automation Testing sẽ không design được đúng/đủ tất cả những cases mà mình cần nếu không nắm chắc những nội dung này, trong trường hợp bạn phải ôm xô cả vai trò của manual tester hoặc phải design testcase trước khi thực hiện scripting.

Để trở thành một Automation Tester, trước tiên, bạn cần có những kiến thức nằm lòng về testing:

* Types testing: Unit/Intergration/System/Sanity/Regression test/.... là gì?
* Testing Techniques: Phân tích giá trị biên/Phân vùng tương đương/Biểu đồ kết quả/Đoán lỗi/... là gì?
* Nắm rõ về HTML và XPath để nhận dạng đúng test objects/elements mà mình cần thao tác.
* Thành thạo ít nhất một ngôn ngữ lập trình thông dụng, hỗ trợ cho việc scripting trên test tools: Java/C#/Python/Ruby/JS/...
* Sử dụng thành thạo thư viện của Selenium WebDriver API bởi Selenium open source, dễ sử dụng, cộng đồng lớn.
* Sử dụng thành thạo ít nhất 1 framework testing: Junit/TestNG/NUnit/... Từ đây sẽ giúp bạn rất nhiều trong việc build framework, hỗ trợ trong việc phân nhóm, quản lý testscript, report, prepare data/environment/browsers.
* Tìm hiểu về software design pattern để build framework/common library
* Sử dụng/build framework thành thạo từ Page Object Model pattern.
* Kỹ năng về coding/IDE: Debug, coding convension, source version control (GIT, SVN,...), cách sử dụng IDE: Visual Studio, Eclipse, IntelliJ..., làm việc với database...
* Học hỏi công nghệ mới trong mảng automation testing: build tools: Maven, ANT..., CI/CD: Jenkins, TeamCity, CircleCI, TFS, Docker.., Clould: AWS, Saucelab, Browserstack, Testingbot..., big data: Hadoop, HBase, Hive..., mobile: Appinum...

## 4. Tầm quan trọng của automation testing trong thời kỳ công nghiệp 4.0

Cách mạng công nghiệp lần thứ tư đánh dấu kỷ nguyên vạn vật kết nối Internet. Nó xảy ra dựa trên sự hội tụ của nhiều công nghệ trong đó có công nghệ cốt lõi có công nghệ thông tin với sự phát triển không ngừng của công nghệ Internet từ thời kỳ kết nối nội dung như email đến mạng xã hội, Internet vạn vật, Internet kết nối thiết bị máy móc kết nối quá trình vận hành của các nhà máy. Ngoài công nghệ cốt lõi còn có sự hội tụ của công nghệ in 3D, công nghệ vật liệu tiên tiến, công nghệ lưu trữ…

Hiện nay, nền công nghiệp 4.0 đang phát triển mạnh mẽ, kéo theo sự phát triển của rất nhiều ngành nghề, đặc biệt là ngành công nghệ thông tin. Do đó, yêu cầu về nhân lực trong mảng này cũng đòi hỏi cac ứng viên cần phải có kỹ năng tốt hơn nữa về lập trình, technical, các kỹ năng về automation....

Nếu search trên các trang mạng tuyển dụng lớn như ITviec, Vietnamworks, LinkIn, Indeed, Dice, Monster, CareerBuilder … thì bạn sẽ thấy có vô vàn kết quả với các từ khóa tìm kiếm như:

* Test Automation Engineer
* Automation Developer
* Automation Testing/Automation Tester
* QA Automation Engineer
* Software Development Engineer in Test (SDET)

Từ đây, bạn có thể thấy được nhu cầu tuyển một QA có kiến thức và kinh nghiệm về automation test là rất lớn. Do đó, đủ để hiểu automation test sẽ là tiềm năng lớn cho tương lai của mảng Test nói riêng và công nghệ thông tin nói chung.

# Unit Test

## Định nghĩa

- Unit Test được thực hiện trong quá trình phát triển ứng dụng.

- Mục tiêu của Unit Test là cô lập từng phần code để xác minh tính chính xác của các phần đó

- Một unit có thể là một function, một class,..

- Unit Test thường được thực hiện bởi developer (nhà phát triển).

## Tại sao nên thực hiện Unit Test?

- Thực hiện unit test ngay trong quá trình phát triển sẽ giúp phát hiện lỗi sớm hơn, dẫn đến chi phí để sửa lỗi cũng ít hơn.

- Unit Test cho phép developer tái cấu trúc lại code sau này nhưng vẫn đảm bảo các module vẫn hoạt động chính xác.

## Thực hiện Unit Test

- Unit Test có thể thực hiện tự động hoặc thủ công, tuy nhiên thường sẽ tiếp cận theo hướng tự động hơn để có thể tái sử dụng trong tương lai (hoặc sử dụng trong các quá trình CI/CD).

- Người thực hiện unit test có thể sử dụng 1 framework tự động (JUnit, NUnit, PHPUnit,..)

- Cách tốt nhất để Unit Test có hiệu quả là khi xây dựng phần mềm theo quy trình TDD (Test-Driven Development).

- Kiểm thử chỉ 1 Unit Test tại 1 thời điểm, nếu một unit cần sử dụng những unit khác hay dữ liệu khác ở bên ngoài, hãy giả lập chúng.

- Một Unit Test đúng sẽ không bị hỏng khi tái cấu trúc hay thay đổi các dịch vụ bên ngoài (trừ trường hợp yêu cầu phần mềm thay đổi)

## Hạn chế của Unit Test

- Do bản chất của unit test là chỉ tập trung vào 1 đơn vị tại 1 thời điểm nên sẽ không thể phát hiện được lỗi tích hợp hay lỗi hệ thống.

## Một số kỹ thuật sử dụng khi viết Unit Test

- **Structural Techniques**: Đây là một kỹ thuật kiểm thử hộp trắng, dựa vào hiểu biết về cấu trúc của phần mềm để thiết kế test case. Bao gồm Statement Testing, Condition Testing, Branch Testing, Expression Testing, ..

- **Functional testing techniques:** Đây là kỹ thuật kiểm thử hộp đen dùng để thiết kế test case theo chức năng của ứng dụng

- **Error based Techniques:** Kỹ thuật này dùng để tìm kiếm lỗi dựa trên những lỗi đã biết trước đó hay kinh nghiệm của người viết unit test

# Acceptance Test

## Định nghĩa

- Đây là một kiểm thử liên quan đến nhu cầu của người sử dụng, để kiểm tra xem phần mềm làm ra đã thỏa mãn tất cả các yêu cầu của doanh nghiệp hay người dùng chưa?

- Acceptance Test được thực hiện trước khi sản phẩm được đưa vào hoạt động

- Acceptance Test được thực hiện bởi người dùng cuối hoặc khách hàng.

## Thực hiện Acceptance Test

### **Điều kiện:**

- BRD sẵn sàng

- Ứng dụng đã hoàn tất

- Kiểm thử hệ thống đã hoàn tất

- Các lỗi đã được sửa (chỉ lỗi về mặt thẩm mỹ mới được chấp nhận còn tồn đọng)

- Chuẩn bị sẵn môi trường cho Acceptance Test

### **Quá trình:**

1. **Phân tích Business Requirement**

2. **Lên kế hoạch**: Bao gồm tiêu chí thoát (exit criteria), hướng tiếp cận kịch bản test, test case hay thời gian thử nghiệm.

3. **Xác định kịch bản thử nghiệm, test case**: Dựa vào bản BR để xây dựng kịch bản thử nghiệm, các test case.

4. **Chuẩn bị dữ liệu**: Tốt nhất là chuẩn bị dữ liệu thật cho việc kiểm thử

5. **Thực hiện thử nghiêm và ghi chép lại kết quả:** Thông thường, Acceptance Test được thực hiện trong một phòng họp: người dùng, PM, đại diện nhóm QA ngồi chung với nhau trong một hoặc hai ngày và làm việc thông qua tất cả các test case,

## Những điểm quan trọng trong Acceptance Test

- Acceptance Test sẽ xem tất cả những chức năng chính đều hoạt động tốt. chứ không chú trọng đến các trang, các trường, các button,... Nếu người dùng tìm thấy bug ở những chức năng chính thì QA sẽ phải xem xét lại testcase, tìm hiểu nguyên nhân tại sao xảy ra bug đó.

- Kiểm thử chấp nhận được chia làm hai loại: thử nghiệm Alpha và Beta

- Hầu hết trong một dự án phát triển phần mềm thường thì UAT được thực hiện trong môi trường đảm bảo chất lượng nếu không có môi trường dàn dựng hoặc môi trường UAT