**TRƯỜNG CAO ĐẲNG THỰC HÀNH**

**FPT POLYTECHNIC**

**KIỂM THỬ CƠ BẢN**

**WORKSHOP 2**

Sinh viên Thực hiện:

HCM-2022

Nguyễn Hoàng Duy

Nguyễn Đức Duy

Nguyễn Đào Danh Lợi

Dương Huy Phi

Ngô Hồng Linh

Vũ Trung Kiên

**MỤC LỤC**

# 1. Unit Test

Unit Test là một loại kiểm thử phần mềm trong đó các đơn vị hay thành phần riêng lẻ của phần mềm được kiểm thử. Kiểm thử đơn vị được thực hiện trong quá trình phát triển ứng dụng. Mục tiêu của Kiểm thử đơn vị là cô lập một phần code và xác minh tính chính xác của đơn vị đó.

**UT có 3 trạng thái cơ bản:**

* Fail (trạng thái lỗi)
* Ignore (tạm ngừng thực hiện)
* Pass (trạng thái làm việc)
* Toàn bộ UT được vận hành trong một hệ thống tách biệt. Có rất nhiều PM hỗ trợ thực thi UT với giao diện trực quan. Thông thường, trạng thái của UT được biểu hiện bằng các màu khác nhau: màu xanh (pass), màu vàng (ignore) và màu đỏ (fail)

**Mỗi UT đều được thiết kế theo trình tự sau:**

* Thiết lập các điều kiện cần thiết: khởi tạo các đối tượng, xác định tài nguyên cần thiết, xây dựng các dữ liệu giả…
* Triệu gọi các phương thức cần kiểm tra.
* Kiểm tra sự hoạt động đúng đắn của các phương thức.
* Dọn dẹp tài nguyên sau khi kết thúc kiểm tra.

## **Ứng dụng Unit test:**

* Kiểm tra mọi đơn vị nhỏ nhất là các thuộc tính, sự kiện, thủ tục và hàm.
* Kiểm tra các trạng thái và ràng buộc của đối tượng ở các mức sâu hơn mà thông thường chúng ta không thể truy cập được.
* Kiểm tra các quy trình (process) và mở rộng hơn là các khung làm việc(workflow – tập hợp của nhiều quy trình)

## **Lợi ích của việc áp dụng Unit test:**

Thời gian đầu, người ta thường do dự khi phải viết UT thay vì tập trung vào code cho các chức năng nghiệp vụ. Công việc viết Unit Test có thể mất nhiều thời gian hơn code rất nhiều nhưng lại có lợi ích sau:

* Tạo ra môi trường lý tưởng để kiểm tra bất kỳ đoạn code nào, có khả năng thăm dò và phát hiện lỗi chính xác, duy trì sự ổn định của toàn bộ PM và giúp tiết kiệm thời gian so với công việc gỡ rối truyền thống.
* Phát hiện các thuật toán thực thi không hiệu quả, các thủ tục chạy vượt quá giới hạn thời gian.
* Phát hiện các vấn đề về thiết kế, xử lý hệ thống, thậm chí các mô hình thiết kế.
* Phát hiện các lỗi nghiêm trọng có thể xảy ra trong những tình huống rất hẹp.
* Tạo hàng rào an toàn cho các khối mã: Bất kỳ sự thay đổi nào cũng có thể tác động đến hàng rào này và thông báo những nguy hiểm tiềm tàng.

Trong môi trường làm việc Unit Test còn có tác dụng rất lớn đến năng suất làm việc:

* Giải phóng chuyên viên QA khỏi các công việc kiểm tra phức tạp.
* Tăng sự tự tin khi hoàn thành một công việc. Chúng ta thường có cảm giác không chắc chắn về các đoạn mã của mình như liệu các lỗi có quay lại không, hoạt động của module hiện hành có bị tác động không, hoặc liệu công việc hiệu chỉnh mã có gây hư hỏng đâu đó…
* Là công cụ đánh giá năng lực của bạn. Số lượng các tình huống kiểm tra (test case) chuyển trạng thái “pass” sẽ thể hiện tốc độ làm việc, năng suất của bạn.

# 2. Integrated Test

1. Integration Testing là công việc kiểm thử tích hợp 1 nhóm các module riêng lẻ với nhau cùng với các Unit Test riêng lẻ trong từng module.
2. Một dự án phần mềm điển hình bao gồm nhiều module phần mềm, được code bởi nhiều người khác nhau. Tích hợp thử nghiệm tập trung vào kiểm tra truyền dữ liệu giữa các module.

**Tại sao Integration Testing là cần thiết**

Mặc dù mỗi module đều được unit test nhưng các lỗi vẫn còn tồn tại với các lý do khác nhau:

* Một Module nói chung được thiết kế bởi một lập trình viên có hiểu biết và logic lập trình có thể khác với các lập trình viên khác. Kiểm thử tích hợp là cần thiết để đảm bảo tính hợp nhất của phần mềm.
* Tại thời điểm phát triển module vẫn có thể có thay đổi trong spec của khách hàng, những thay đổi này có thể không được kiểm tra ở giai đoạn unit test trước đó.
* Giao diện và cơ sở dữ liệu của các module có thể chưa hoàn chỉnh khi được ghép lại
* Khi tích hợp hệ thống các module có thể không tương thích với cấu hình chung của hệ thống
* Thiếu các xử lý ngoại lệ có thể xảy ra

**Các phương pháp:**

#### **Phương pháp tiếp cận Big Bang**

Tại đây tất cả các thành phần được tích hợp cùng 1 lúc, sau đó sẽ tiến hành kiểm thử.

***Ưu điểm:***

Thuận tiện với các dự án nhỏ

***Nhược điểm:***

Khó khăn trong việc phát hiện bug.

Có thể bỏ qua các bug giao diện nhỏ trong quá trình tìm bug

Mất thời gian dành cho tích hợp hệ thống nên làm giảm thời gian dành cho test.

Vì các module được kiểm thử cùng 1 lúc nên các module có nguy cơ bị cô lập trong quá trình kiểm thử

#### **Phương pháp tiếp cận Incremental**

Trong phương pháp này, kiểm tra được thực hiện bằng cách kết hợp hai hay nhiều module có liên quan một cách hợp lý. Sau đó, các phân hệ liên quan khác được thêm vào và kiểm tra sự hoạt động đúng đắn. Quá trình tiếp tục cho đến khi tất cả các module được tham gia và thử nghiệm thành công.

Quá trình này được thực hiện bằng cách sử dụng các chương trình giả gọi là Stub and Driver. Sơ khai và trình điều khiển không thực hiện toàn bộ logic lập trình các module nhưng chỉ mô phỏng giao tiếp dữ liệu với các module được gọi.

Stub: Được gọi bởi Module dưới Test.

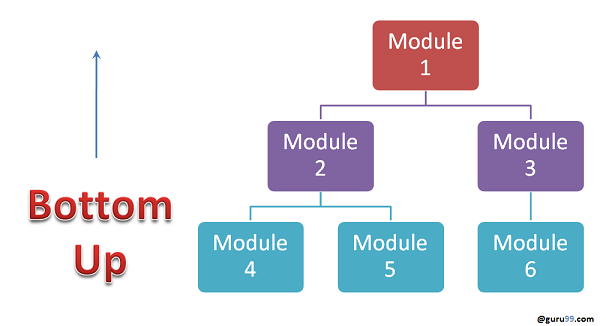
Driver: Gọi Module để được kiểm tra.

Phương pháp **Incremental** được thực hiện bởi hai phương pháp khác nhau:

**Bottom Up Top Down Bottom up Integration**

**Chiến lược Bottom Up**

Mỗi module ở mức thấp hơn được thử nghiệm với các module cao hơn cho đến khi tất cả các module đều được kiểm tra. Nó được sử dụng cho Driver testing. Thể hiện bằng biểu đồ dưới đây:



***Ưu điểm:***

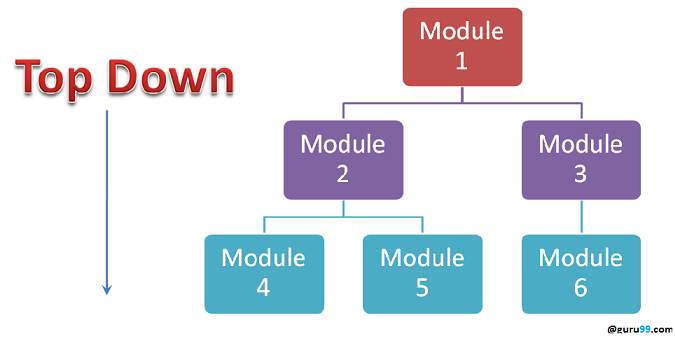
* Thu gọn phạm vi bug dễ dàng hơn
* Không mất thời gian chờ tất cả các module được tích hợp

***Nhược điểm:***

* Module quan trọng của hệ thống có thể dễ bị lỗi
* Không giữ được nguyên mẫu đầu tiên của hệ thống

#### **Top down Integration:**

Trong tiếp cận từ trên xuống , kiểm tra được thực hiện từ trên xuống dưới theo dõi dòng kiểm soát của hệ thống phần mềm. Nó được sử dụng cho Stub testing



***Ưu điểm:***

* Thu gọn phạm vi bug dễ dàng hơn
* Khả năng để có được một nguyên mẫu ban đầu.
* Modules quan trọng đang được thử nghiệm trên mức ưu tiên; lỗi trong thiết kế lớn có thể được tìm thấy và cố định đầu tiên.

***Nhược điểm:***

* Cần nhiều Stub.
* Module ở mức độ thấp hơn sẽ được kiểm tra không đầy đủ.

### **Các bước thực hiện test tích hợp**

* Chuẩn bị Integration Test Plan
* Thiết kế các kịch bản thử nghiệm, trường hợp, và Script (Test Scenarios, Cases, and Scripts ).
* Thực hiện kiểm tra theo test case đã viết
* Theo dõi & tái kiểm tra các lỗi ở trên.
* Bước 3 và 4 được lặp đi lặp lại cho đến khi hoàn thành Integration là thành công.

# 3. System Testing

1. **System Testing là gì:**

Kiểm thử hệ thống là một phương pháp theo dõi và đánh giá hành vi của sản phẩm hoặc hệ thống phần mềm hoàn chỉnh và đã được tích hợp đầy đủ, dựa vào đặc tả và các yêu cầu chức năng đã được xác định trước. Đó là giải pháp cho câu hỏi "Liệu hệ thống hoàn chỉnh có hoạt động đúng với yêu cầu hay không?"

System test được thử nghiệm trong hộp đen, tức là chỉ có các tính năng làm việc bên ngoài của phần mềm được đánh giá trong quá trình thử nghiệm này. Nó không đòi hỏi bất kỳ kiến thức nội bộ nào về coding, lập trình, thiết kế, v.v. và hoàn toàn dựa trên quan điểm của người dùng.

**Khi nào thì dùng System Testing:**

* Sau khi hoàn thành unit & integration testing.
* Trước khi bắt đầu acceptance testing
* Sau khi tích hợp hoàn toàn các mô-đun.
* Sau khi hoàn thành quy trình phát triển phần mềm, dựa trên đặc tả yêu cầu phần mềm (SRS).
* Sau khi môi trường thử nghiệm sẵn sàng.

**Điều kiện tiên quyết để thực hiện System Testing:**

* Phải đảm bảo phần mềm được thống nhất kiểm tra.
* Kiểm thử tích hợp đã được thực hiện trên sản phẩm.
* Phần mềm nên được phát triển hoàn chỉnh.
* Trước khi thực hiện quy trình kiểm tra hệ thống, phải đảm bảo rằng môi trường kiểm tra đã sẵn sàng.

**Lý do thực hiện System Testing:**

* Đảm bảo sản phẩm đáp ứng các tiêu chuẩn chất lượng.
* Xác minh hệ thống phần mềm đáp ứng các yêu cầu chức năng, kỹ thuật và kinh doanh theo yêu cầu của khách hàng.
* Thực hiện kiểm tra từ đầu đến cuối của sản phẩm phần mềm giúp ngăn ngừa lỗi hệ thống và sự cố trong quá trình thực hiện với môi trường thật.
* Được thực hiện trong một môi trường tương tự như môi trường production, cho phép các nhà phát triển cũng như các bên liên quan có được ý tưởng về phản ứng của người dùng đối với sản phẩm.
* Đóng một vai trò quan trọng trong việc cung cấp một sản phẩm chất lượng cho người dùng cuối.
* Chính trong giai đoạn này của vòng đời kiểm thử phần mềm (STLC), các Yêu cầu nghiệp vụ và Kiến trúc ứng dụng được kiểm tra.
* Đảm bảo rằng đầu vào được cung cấp đầu ra / kết quả như mong đợi.

**Lợi ích của System Testing:**

System testing là một phần không thể thiếu trong vòng đời kiểm thử phần mềm, được thực hiện khi quá trình phát triển phần mềm hoàn tất và sản phẩm đã trải qua kiểm thử đơn vị và tích hợp. Thử nghiệm này không giới hạn ở một khía cạnh hoặc thành phần của sản phẩm mà được sử dụng để kiểm tra toàn bộ hệ thống phần mềm, điều này làm cho nó trở thành một phần quan trọng của bất kỳ chu kỳ thử nghiệm nào và do đó nó được thực hiện bởi tester, những người mong muốn cung cấp trải nghiệm người dùng tốt nhất.

# 4. User Acceptance Test

## **User Acceptance Testing (UAT) là gì?**

User Acceptance Testing là quá trình xác nhận rằng phần mềm đã tạo ra có hoạt động phù hợp với người dùng cuối hay không.

## **Ai là người thực hiện UAT?**

* Người dùng cuối
* Khách hàng

## **Tại sao cần User Acceptance Testing?**

* Các nhà phát triển phần mềm dựa trên tài liệu yêu cầu và sự hiểu biết của họ để xây dựng các yêu cầu và có thể không thực sự là những gì khách hàng cần từ phần mềm.
* Các yêu cầu thay đổi trong quá trình làm dự án có thể không được truyền đạt hiệu quả và chính xác đến các nhà phát triển.

## **Các bước thực hiện UAT**

* Phân tích các yêu cầu nghiệp vụ của phần mềm
* Tạo kế hoạch kiểm tra UAT
* Xác định các kịch bản kiểm thử
* Tạo các trường hợp kiểm tra UAT
* Chuẩn bị data test (giống với data thật nhất)
* Thực hiện kiểm thử
* Ghi nhận kết quả
* Xác nhận các chức năng của sản phẩm

## **User Acceptance Testing (UAT) khác gì với Functional Testing?**

User Acceptance Tests bao gồm 1 bộ các test steps dùng để xác nhận xem các yêu cầu đặc tả đã đúng với nhu cầu của user không. Nếu khách hàng và nhà cung cấp đồng ý với sản phẩm thì phát triển phần mềm được bắt đầu.

Functional testing – kiểm tra các yêu cầu cụ thể và thông số kỹ thuật của phần mềm. Nó thiếu thành phần người dùng. Functional testing có thể đưa ra kết luận rằng phần mềm đáp ứng các thông số kỹ thuật của nó không. Tuy nhiên, không xác minh liệu phần mềm đó có thực sự phù hợp với nhu cầu người dùng không.

## **Các kiểu của User Acceptance Testing**

### **Alpha & Beta Testing**

Alpha testing thường diễn ra trong môi trường phát triển và thường được thực hiện bởi nhân viên nội bộ. Ngoài ra các nhóm người dùng tiềm năng cũng có thể tiến hành Alpha Tests. Dựa trên những phản hồi – được thu thập từ những người thử nghiệm alpha – nhóm phát triển sẽ khắc phục một số vấn đề cần thiết và cải thiện khả năng sử dụng của sản phẩm.

Beta Testing còn được gọi là “thử nghiệm lĩnh vực” của Wap, diễn ra trong môi trường của khách hàng và liên quan đến một số thử nghiệm rộng rãi của một nhóm khách hàng sử dụng hệ thống trong môi trường của họ. Những người thử nghiệm beta sau đó cung cấp thông tin phản hồi, từ đó dẫn đến những cải tiến của sản phẩm.

### **Usersnap**

Usersnap Classic là một giải pháp tuyệt vời để yêu cầu người kiểm tra alpha và beta phản hồi. Với Usersnap Classic, các nhóm UAT có thể dễ dàng thu thập và phân tích phản hồi định tính từ những người thử nghiệm. Và đối với những người thử nghiệm, nó cực kỳ dễ dàng để làm việc thông qua thử nghiệm alpha hoặc beta đầu tiên, vì họ có thể chỉ cần vẽ trên màn hình để cung cấp phản hồi.

### **Contract Acceptance Testing**

Contract Acceptance Testing (Kiểm tra chấp nhận hợp đồng) có nghĩa là một phần mềm phát triển được kiểm tra theo các tiêu chí và thông số kỹ thuật nhất định được xác định và thỏa thuận trong hợp đồng. Nhóm dự án xác định các tiêu chí và thông số kỹ thuật có liên quan để chấp nhận đồng thời khi nhóm đồng ý với hợp đồng.

### **Regulation Acceptance Testing**

Regulation Acceptance Testing (Kiểm tra chấp nhận quy định), còn được gọi là Compliance Acceptance Testing(Kiểm tra chấp nhận tuân thủ), kiểm tra xem phần mềm có tuân thủ các quy định hay không. Điều này bao gồm các quy định của chính phủ và pháp lý.

### **Operational acceptance testing**

Còn được gọi là Operational Readiness Testing (Thử nghiệm sẵn sàng hoạt động) hoặc Production Acceptance Testing (Thử nghiệm chấp nhận sản xuất). Các trường hợp thử nghiệm này đảm bảo các quy trình công việc để cho phép phần mềm hoặc hệ thống được sử dụng. Nó bao gồm các quy trình công việc cho các kế hoạch dự phòng, đào tạo người dùng, bảo trì và kiểm tra bảo mật khác nhau.

### **Black Box Testing**

Black Box Testing (Kiểm thử hộp đen) thường được phân loại là kiểm tra chức năng, nhưng trong một chừng mực nào đó có thể được xem là một loại Kiểm tra chấp nhận người dùng. Nó là một phương pháp kiểm thử phần mềm để phân tích các chức năng nhất định mà không cho phép người kiểm tra thấy cấu trúc code bên trong. Trong quá trình kiểm tra Hộp đen, người dùng không biết về bất kỳ cơ sở code nào, nhưng phải biết về các yêu cầu mà phần mềm phải đáp ứng.