**Phần mềm quản lý đào tạo trường phenikaa**

# **Phần 1: Tài liệu về tìm hiểu về Phần mềm/ hoặc Nội dung theo đề tài được lựa chọn**

Link: <https://qldt.phenikaa-uni.edu.vn/>

**Phần mềm quản lý đào tạo trường phenikaa**

Hệ thống quản lý đạo tạo của trường đại học Phenikaa. Đây là hệ thống do nhà cung cấp Thiên An phát triển. Mục đích chính là hỗi trợ sinh viên cũng như nhà trường dễ dàng thông báo và tiếp nhận, quản lý thông tin một cách nhanh và dễ dàng nhất.

**Các chức năng chính:**

* Đăng nhập:
* Lấy lại mật khẩu:
* Xem thông tin
* Xem thông tin cá nhân
* Xem thông tin học phần
* Xem điểm học tập
* Sửa thông tin cá nhân
* Đăng ký môn học
* Đóng học phí
* Đánh giá giảng viên
* Xem thông tin thực tập

# **Phần 2: Tài liệu phân tích yêu cầu Phần mềm, hoặc phân tích Nội dung, bao gồm cả phân tích về chức năng, các usecase, các sơ đồ diagrams (activities, sequences, state diagram)**

**1.Giới Thiệu**

Ngôn ngữ mô hình hóa thống nhất (tiếng Anh: Unified Modeling Language, viết tắt thành UML) là một ngôn ngữ mô hình gồm các ký hiệu đồ họa mà các phương pháp hướng đối tượng sử dụng để thiết kế các hệ thống thông tin một cách nhanh chóng.

Cách xây dựng các mô hình trong UML phù hợp mô tả các hệ thống thông tin cả về cấu trúc cũng như hoạt động. Cách tiếp cận theo mô hình của UML giúp ích rất nhiều cho những người thiết kế và thực hiện hệ thống thông tin cũng như những người sử dụng nó; tạo nên một cái nhìn bao quát và đầy đủ về hệ thống thông tin dự định xây dựng. Cách nhìn bao quát này giúp nắm bắt trọn vẹn các yêu cầu của người dùng; phục vụ từ giai đoạn phân tích đến việc thiết kế, thẩm định và kiểm tra sản phẩm ứng dụng công nghệ thông tin. Các mô hình hướng đối tượng được lập cũng là cơ sở cho việc ứng dụng các chương trình tự động sinh mã trong các ngôn ngữ lập trình hướng đối tượng, chẳng hạn như ngôn ngữ C++, Java,... Phương pháp mô hình này rất hữu dụng trong lập trình hướng đối tượng. Các mô hình được sử dụng bao gồm Mô hình đối tượng (mô hình tĩnh) và Mô hình động.

UML sử dụng một hệ thống ký hiệu thống nhất biểu diễn các Phần tử mô hình (model elements). Tập hợp các phần tử mô hình tạo thành các Sơ đồ UML (UML diagrams). Có các loại sơ đồ UML chủ yếu sau:

Sơ đồ lớp (Class Diagram)

Sơ đồ đối tượng (Object Diagram)

Sơ đồ tình huống sử dụng (Use Cases Diagram)

Sơ đồ trình tự (Sequence Diagram)

Sơ đồ cộng tác (Collaboration Diagram hay là Composite Structure Diagram)

Sơ đồ trạng thái (State Machine Diagram)

Sơ đồ thành phần (Component Diagram)

Sơ đồ hoạt động (Activity Diagram)

Sơ đồ triển khai (Deployment Diagram)

Sơ đồ gói (Package Diagram)

Sơ đồ liên lạc (Communication Diagram)

Sơ đồ tương tác (Interaction Overview Diagram - UML 2.0)

Sơ đồ phối hợp thời gian (Timing Diagram - UML 2.0)

**2.Một số dạng biểu đồ UML phổ biến**

**2.1.Biểu đồ Use case (Use Case Diagram)**

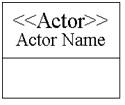
Một biểu đồ Use case chỉ ra một số lượng các tác nhân ngoại cảnh và mối liên kết của chúng đối với Use case mà hệ thống cung cấp. Một Use case là một lời miêu tả của một chức năng mà hệ thống cung cấp. Lời miêu tả Use case thường là một văn bản tài liệu, nhưng kèm theo đó cũng có thể là một biểu đồ hoạt động. Các Use case được miêu tả duy nhất theo hướng nhìn từ ngoài vào của các tác nhân (hành vi của hệ thống theo như sự mong đợi của người sử dụng), không miêu tả chức năng được cung cấp sẽ hoạt động nội bộ bên trong hệ thống ra sao. Các Use case định nghĩa các yêu cầu về mặt chức năng đối với hệ thống.

Hệ thống: Với vai trò là thành phần của biểu đồ use case, hệ thống biểu diễn ranh giới giữa bên trong và bên ngoài của một chủ thể trong phần mềm chúng ta xây dựng.Một hệ thống ở trong biểu đồ use case không nhất thiết là một hệ phần mềm; nó có thể là một chiếc máy,hoặc là một hệ thống thực như một doanh nghiệp, một trường đại học,…

Tác nhân(actor):là người dùng của hệ thống, một tác nhân có thể là một người dùng thực hoặc các hệ thống máy tính khác có vai trò nào đó trong hoạt động của hệ thống. Như vậy, tác nhân thực hiện các use case. Một tác nhân có thể thực hiện nhiều use case và ngược lại một use case cũng có thể được thực hiện bởi nhiều tác nhân

Tác nhân được kí hiệu:

hoặc



Các use case: Đây là thành phần cơ bản của biểu đồ use case. Các use case được biểu diễn bởi các hình elip.Tên các use case thể hiện một chức năng xác định của hệ thống.

Các Use case được kí hiệu bằng hình elips.

Mối quan hệ giữa các use case:

Association: thường được dùng để mô tả mối quan hệ giữa Actor và Use Case và giữa các Use Case với nhau

Association.jpg

Ví dụ quan hệ association:

Include: là quan hệ giữa các Use Case với nhau, nó mô tả việc một Use Case lớn được chia ra thành các Use Case nhỏ để dễ cài đặt (module hóa) hoặc thể hiện sự dùng lại.



Ví dụ quan hệ include:

Extent: Extend dùng để mô tả quan hệ giữa 2 Use Case. Quan hệ Extend được sử dụng khi có một Use Case được tạo ra để bổ sung chức năng cho một Use Case có sẵn và được sử dụng trong một điều kiện nhất định nào đó.

Generalization: được sử dụng để thể hiện quan hệ thừa kế giữa các Actor hoặc giữa các Use Case với nhau.

Ví dụ quan hệ Generalization:

**2.2.Biểu đồ lớp (Class Diagram)**

Một biểu đồ lớp chỉ ra cấu trúc tĩnh của các lớp trong hệ thống. Các lớp là đại diện cho các “đối tượng” được xử lý trong hệ thống. Các lớp có thể quan hệ với nhau trong nhiều dạng thức:

liên kết (associated - được nối kết với nhau),

phụ thuộc (dependent - một lớp này phụ thuộc vào lớp khác),

chuyên biệt hóa (specialized - một lớp này là một kết quả chuyên biệt hóa của lớp khác),

hay đóng gói ( packaged - hợp với nhau thành một đơn vị).

Tất cả các mối quan hệ đó đều được thể hiện trong biểu đồ lớp, đi kèm với cấu trúc bên trong của các lớp theo khái niệm thuộc tính (attribute) và thủ tục (operation). Biểu đồ được coi là biểu đồ tĩnh theo phương diện cấu trúc được miêu tả ở đây có hiệu lực tại bất kỳ thời điểm nào trong toàn bộ vòng đời hệ thống.

Một hệ thống thường sẽ có một loạt các biểu đồ lớp – không phải bao giờ tất cả các biểu đồ lớp này cũng được nhập vào một biểu đồ lớp tổng thể duy nhất – và một lớp có thể tham gia vào nhiều biểu đồ lớp.

**2.2.1.Một lớp có các thành phần sau**

Tên lớp

Các thuộc tính

Các phương thức

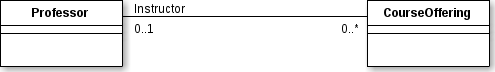
**2.2.2.Liên kết giữa các lớp**

Liên kết (Association)

Mối liên hệ ngữ nghĩa giữa hai hay nhiều lớp chỉ ra sự liên kết giữa các thể hiện của chúng

Mối quan hệ về mặt cấu trúc chỉ ra các đối tượng của lớp này có kết nối với các đối tượng của lớp khác.

Bội số quan hệ: là số lượng thể hiện của một lớp liên quan tới một thể hiện của lớp khác. Với mỗi liên kết, có hai bội số quan hệ cho hai đầu của liên kết.

Ví dụ:

Với mỗi đối tượng của Professor, có nhiều Course Offerings có thể được dạy. Với mỗi đối tượng của Course Offering, có thể có 1 hoặc 0 Professor giảng dạy.

Biểu diễn bội số quan hệ:

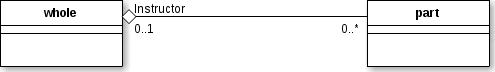
| **Biểu diễn** | **Ý nghĩa** |
| --- | --- |
|  | Unspecified |
| 1 | chính xác 1 |
| 0..\* | 0 hoặc nhiều |
| \* | 0 hoặc nhiều |
| 1..\* | 1 hoặc nhiều |
| 0..1 | 0 hoặc 1 |
| 2..4 | Specified Range |
| 2, 4..6 | Multiple, Disjoint Ranges |

Kết tập (Aggregation)

Là một dạng đặc biệt của liên kết mô hình hóa mối quan hệ toàn thể-bộ phận (whole-part) giữa đối tượng toàn thể và các bộ phận của nó.

Kết tập là mối quan hệ “là một phần” (“is a part-of”).

Bội số quan hệ được biểu diễn giống như các liên kết khác



Cấu thành (Composition) là :Một dạng của kết tập với quyền sở hữu mạnh và các vòng đời trùng khớp giữa hai lớp

▫ Whole sở hữu Part, tạo và hủy Part.

▫ Part bị bỏ đi khi Whole bị bỏ, Part không thể tồn tại nếu Whole không tồn tại.

Sự khác nhau giữa Association, Aggregation và Composition

Tổng quát hóa (Generalization)

Mối quan hệ giữa các lớp trong đó một lớp chia sẻ cấu trúc và/hoặc hành vi với một hoặc nhiều lớp khác

Xác định sự phân cấp về mức độ trừu tượng hóa trong đó lớp con kế thừa từ một hoặc nhiều lớp cha

▫ Đơn kế thừa (Single inheritance)

▫ Đa kế thừa (Multiple inheritance)

Là mối liên hệ “là một loại” (“is a kind of”)

Lớp trừu tượng và lớp cụ thể (Abstract and Concrete Class)

**2.2.3.Gói**

Nếu bạn đang mô hình hóa một hệ thống lớn hoặc một lĩnh vực nghiệp vụ lớn, thì không thể tránh khỏi, sẽ có nhiều phân loại khác nhau trong mô hình của bạn. Việc quản lý tất cả các lớp có thể là một nhiệm vụ khó khăn, do vậy UML cung cấp một phần tử tổ chức được gọi là gói. Các gói cho phép các nhà tạo mô hình tổ chức các phân loại của mô hình thành các vùng tên, là một kiểu giống như các thư mục trong một hệ thống tệp. Việc phân chia một hệ thống thành nhiều gói làm cho hệ thống trở nên dễ hiểu, đặc biệt là nếu từng gói đại diện cho một phần cụ thể của hệ thống

Có hai cách để vẽ các gói trên sơ đồ. Không có quy tắc để xác định xem ký pháp nào sẽ được sử dụng, ngoại trừ việc tuân theo phán xét riêng của bạn về việc ký pháp nào là dễ đọc các sơ đồ lớp mà bạn đang vẽ nhất. Cả hai cách sẽ bắt đầu bằng một hình chữ nhật lớn với một hình chữ nhật nhỏ hơn (phiếu) nằm ở phía trên cùng bên trái nó, như trong . Nhưng nhà tạo mô hình phải quyết định cách thể hiện các thành viên của gói như thế nào, ví dụ như sau:

Nếu nhà tạo mô hình quyết định hiển thị các thành viên của gói bên trong hình chữ nhật lớn, thì tất cả các thành viên4 sẽ phải được đặt trong hình chữ nhật đó. Cũng vậy, tên của gói cần được đặt trong hình chữ nhật nhỏ hơn của gói Nếu nhà tạo mô hình quyết định hiển thị các thành viên của gói bên ngoài hình chữ nhật lớn, thì tất cả các thành viên sẽ được hiển thị trên sơ đồ cần phải được đặt ở bên ngoài hình chữ nhật ấy. Để cho thấy phân loại nào thuộc về gói, thì một đường thẳng sẽ được vẽ từ từng phân loại đến một vòng tròn có dấu cộng (+) bên trong vòng tròn gắn liền với gói.

====> Như vậy trong phần này tôi đã giới thiệu với các bạn về biểu đồ UML và 2 dạng biểu đồ cơ bản hay được sử dụng trong các tài liệu thiết kế hệ thống.Ở phần tiếp theo tôi sẽ tiếp tục giới thiệu tới các bạn 3 dạng biểu đồ tiếp theo là:

Biểu đồ trạng thái (State Diagram)

Biểu đồ tuần tự (Sequence Diagram)

Biểu đồ hoạt động (Active Diagram)

**3.Công cụ vẽ biểu đồ UML**

Công cụ offline

Diagram download tại đây: <http://diagram-designer.en.softonic.com/>

Công cụ online: <https://cacoo.com/diagrams/>

<http://creately.com/Draw-UML-and-Class-Diagrams-Online>

**4.Tài liệu tham khảo**

Các bạn có thể xem các qui định chuẩn trong thiết kế biểu đồ UML tại đây:

<http://www.tutorialspoint.com/uml/uml_standard_diagrams.htm>

<https://www.smartdraw.com/uml-diagram/>

<http://www.holub.com/goodies/uml/>

# **Phần 3: Tài liệu test plan**

Test là một công việc quan trọng đòi hỏi người thực hiện rất nhiều kỹ thuật kinh nghiệm và cần phải thực hiện theo đúng quy trình đã được định trước.

**Test plan là gì**

TEST PLAN là một tài liệu chi tiết mô tả chiến lược kiểm thử, mục tiêu, lịch trình, ước tính và khả năng cung cấp và các nguồn lực cần thiết để kiểm thử. Test plan giúp tester xác định nỗ lực cần thiết để xác nhận chất lượng của ứng dụng đang được kiểm thử phần mềm.

Test plan đóng vai trò như một kế hoạch chi tiết để tiến hành các hoạt động kiểm thử phần mềm như một quy trình xác định, được giám sát và kiểm soát từng bước bởi người quản lý kiểm thử.

**Tầm quan trọng của test plan**

Lập test plan có nhiều lợi ích

Giúp những người ngoài nhóm kiểm thử như nhà phát triển, quản lý doanh nghiệp, khách hàng hiểu chi tiết về kiểm thử.

Test plan hướng dẫn suy nghĩ của chúng ta. Nó giống như một cuốn sách quy tắc, cần phải được tuân theo.

Các khía cạnh quan trọng như ước tính kiểm thử, phạm vi kiểm thử, chiến lược kiểm thử được ghi lại trong test plan. Do đó, nhóm quản lý có thể xem xét và sử dụng lại cho các dự án khác.

**Test Plan gồm những gì**

Bạn đã biết rằng lập test plan là nhiệm vụ quan trọng nhất của quy trình quản lý kiểm thử. Thực hiện theo bảy bước dưới đây để tạo một test plan theo IEEE 829

Phân tích sản phẩm

Thiết kế chiến lược kiểm thử

Xác định mục tiêu kiểm thử

Xác định tiêu chí kiểm thử

Hoạch định nguồn lực

Lên kế hoạch môi trường kiểm thử (Test Environment)

Lịch trình & Dự toán

Xác định sản phẩm kiểm thử

**Vai trò của Test Plan**

Test plan đóng vai trò như một kế hoạch chi tiết để tiến hành các hoạt động kiểm thử phần mềm, được giám sát và kiểm soát từng bước bởi người quản lý kiểm thử. Cụ thể, những vai trò của test plan là:

Giúp nhà phát triển, quản lý doanh nghiệp, khách hàng hiểu chi tiết về kiểm thử.

Test plan hướng dẫn những quy tắc và điều luật cần tuân theo để ó những hiệu quả tốt nhất

Nhóm quản lý có thể xem xét và sử dụng lại test plan cho các dự án khác nhau vì những khía cạnh quan trọng như ước tính kiểm thử, phạm vi kiểm thử, chiến lược kiểm thử được ghi lại trong test plan.

**8 Bước chuẩn chỉnh cho 1 test plan hoàn hảo**

* **Bước 1. Phân tích sản phẩm**

Đây là bước đầu tiên và có vai trò cực kỳ quan trọng đối với bất kỳ một quá trình kiểm thử nào. Nghiên cứu và phân tích sản phẩm càng sát sao bao nhiêu thì lại càng hạn chế được những sai lầm không đáng có bấy nhiêu.

* **Bước 2. Xây dựng chiến lược kiểm thử**

Chiến lược kiểm thử sẽ chi tiết hóa một số nội dung như:

Phạm vi test

Những phân đoạn sẽ được test và phần không được test

Những hình thức test cần sử dụng

Những rủi ro và vấn đề có thể phát sinh trong quá trình test

* **Bước 3. Xác định mục tiêu kiểm thử**

Mỗi quá trình test sẽ có một mục tiêu khác nhau, cần xác định rõ mục tiêu của quá trình đó là gì. Có thể là tìm ra lỗi của phần mềm, hay xác nhận phần mềm đã đủ tiêu chuẩn để đưa ra thị trường chưa. Do đó, việc xác định đúng mục tiêu sẽ giúp cho việc test sản phẩm diễn ra nhanh chóng và suôn sẻ hơn.

* **Bước 4. Xác định tiêu chí kiểm thử**

Tiêu chí kiểm thử (Test Criteria) bao gồm hai loại:

Tiêu chí tạm dừng (Suspension Criteria): Ở mục này, nếu không thỏa mãn các tiêu chí này thì quá trình test sẽ phải dừng lại. Chẳng hạn báo cáo cho thấy trường hợp test thất bại cao thì sẽ không tiếp tục nữa. Nhóm test sẽ chuyển kết quả sang cho nhóm phát triển để khắc phục kịp thời những lỗi hiện có.

Tiêu chí thoát (Exit Criteria): Đây là những điều kiện cần phải đạt được để có thể kết thúc quá trình test. Chẳng hạn như một phần mềm sẽ được coi là thành công nếu có 80% trường hợp test thành công.

* **Bước 5: Lập kế hoạch về nguồn lực**

Nguồn lực hay là các tài nguyên cho dự án test cũng là một vấn đề vô cùng quan trọng. Nguồn lực có thể bao gồm nhiều yếu tố, có thể là con người, các thiết bị và vật liệu cần thiết để hoàn thành dự án. Có được kế hoạch về nguồn lực sẽ giúp người quản lý quá trình test đưa ra lịch trình phù hợp và ước lượng được thời gian chính xác để chạy dự án.

* **Bước 6: Xác định môi trường kiểm thử**

Tất cả những phần cứng và phần mềm mà cả team sử dụng sẽ được tổng hợp trong môi trường kiểm thử. Kết quả của test plan có thành công hay không phụ thuộc rất lớn vào các yếu tố trong môi trường kiểm thử. Khi tester có thể giám sát mọi biến động của phần mềm trong điều kiện sử dụng thực tế chính là môi trường test lý tưởng.

* **Bước 7: Sắp xếp lịch hoạt động**

Ở bước này. bạn cần lên kế hoạch chi tiết cho dự án, có thể bằng cách chia cả quá trình thành những task nhỏ. Từ đó có thể dễ dàng phân bố thời gian và kiểm soát nhiệm vụ chi tiết cho mỗi task.

* **Bước 8: Kiểm soát sản phẩm thử nghiệm**

Trong suốt quá trình test, người thực hiện cần phải lập dữ liệu test, ghi lại chi tiết nhật ký test. Sau khi test xong, cần đưa ra được kết quả quá trình test cùng những báo cáo lỗi và ghi chú về việc phát hành.

# **Phần 4: Tài liệu test case**

**4.1. Định nghĩa về test case.**

- Test case (trường hợp kiểm thử) được định nghĩa là các đầu vào (input), điều kiện kiểm thử, quy trình kiểm thử và kết quả truy vấn (expected result) với một mục tiêu kiểm thử cụ thể, đó là kiểm tra các chức năng, tính năng của phần mềm trước khi tới với người dùng cuối cùng.

**4.2. Mục đích khi sử dụng TEST CASE.**

- Mục tiêu chính là đảm bảo các tính năng của một phần mềm đang hoạt động đúng như mong đợi. Nó giúp các tester xác định được liệu phần mềm đã hết các khuyết điểm hay đáp ứng đủ nhu cầu khi được tung ra thị trường hay chưa.

- Ngoài ra test case còn giúp ta:

+ Đảm bảo phạm vi được kiểm thử.

+ Giúp cải thiện chất lượng của phần mềm.

+ Giảm chi phí bảo hành và hỗ trợ phần mềm.

+ Giúp xác định phần mềm đáp ứng yêu cầu người dùng.

+ Có thể tái sử dụng trong tương lai và bất cứ ai cũng có thể tham khảo test case và tiến hành với chúng.

**4.3. Các bước dùng test case.**

Bước 1: Đặt tiêu đề gây ấn tượng mạnh

Bước 2: Trình bày tính năng nào của phần mềm sẽ được kiểm tra, có thể thêm một số thông tin khác như môi trường kiểm thử, dữ liệu kiểm thử.

Bước 3: Đặt ra các assumption hay điều kiện tiên quyết mà bạn cho là cần thiết để quá trình kiểm thử có thể diễn ra.

Bước 4: Trình bày các bước kiểm thử, lưu ý các bước phải chính xác rõ ràng.

Bước 5: Đưa ra kết quả truy vấn. Từ kết quả này tester có thể xác định kiểm thử là “pass” hay “fail”.

**4.5. Các loại TEST CASE phổ biến trên thị trường.**

- Các test case thường sẽ được chia làm hai nhóm: Kiểm thử chức năng và Kiểm thử phi chức năng.

+ **Kiểm thử chức năng** là kiểm thử các hành vi của phần mềm dựa theo yêu cầu của người dùng. Một số loại kiểm thử chức năng bao gồm: Kiểm thử tích hợp, Kiểm thử cơ sở dữ liệu, Kiểm thử giao diện người dùng, Kiểm thử chấp nhận người dùng.

+ **Kiểm thử phi chức năng** là kiểm tra các yếu tố không liên quan đến chức năng của phần mềm như hiệu suất. Một số loại kiểm thử phi chức năng như: Kiểm thử hiệu năng, Kiểm thử bảo mật, Kiểm thử khả năng sử dụng.

# **Phần 5: Test report**

**Test Report là gì**

Test Report có thể hiểu đơn giản là một bản tóm tắt, trong đó chứa mục tiêu kiểm thử, hoạt động kiểm thử và kết quả. Mục đích của test report là giúp những bộ phận liên quan như bộ phận dev, test, phân tích, v.v… đánh giá được chất lượng sản phẩm và liệu sản phẩm hay giải pháp đó đã có thể đưa vào vận hành được chưa.

Tuy nhiên, test report không chỉ dùng để đánh giá chất lượng, mà còn được sử dụng để những nhà phát triển hiểu rõ hơn về quy trình của bài test. Liệu bài test đã đi đúng hướng chưa? Test đã ổn định chưa? Liệu có thể phát hiện vấn đề sớm hơn không?

Do đó, Test Report không chỉ phục vụ cho mục đích kiểm tra chất lượng của sản phẩm, mà còn phục vụ cho việc tăng tốc quá trình ra đời của sản phẩm trong tương lai.

Tại sao Test Report lại cần thiết

Hãy lấy một ví dụ đơn giản như sau:  
Công ty bạn chuẩn bị bàn giao website A cho khách hàng. Sếp hỏi bạn liệu đã có thể bàn giao website cho khách chưa. Do team của bạn đã test website này nên bạn vô cùng tự tin đưa ra câu trả lời là có. Sếp của bạn hoàn toàn tin bạn nên đã bàn giao luôn website cho khách.

Sau đó khoảng 1-2 tháng, khách hàng liên tục phàn nàn về những trục trặc và trở ngại xuất hiện trên website. Tại sao hiện tượng này lại có thể xảy ra, team của bạn đã test rồi mà?

Lí do chính là bạn đã bỏ qua quá trình đánh giá và báo cáo việc quản lý kiểm thử. Không có Test Report, sếp của bạn không có đủ thông tin để đánh giá chất lượng sản phẩm và phải phụ thuộc hoàn toàn vào lời nói của bạn.

Test Report có 3 lợi ích chính:

Giúp đánh giá được tiến độ hiện tại cũng như chất lượng sản phẩm của team.

Giúp các bộ phận liên quan cùng đưa ra feedback kịp thời.

Test Report đóng vai trò như báo cáo cuối cùng để đánh giá xem sản phẩm đã sẵn sàng bàn giao hay chưa.

**Test Report nên bao gồm những phần nào?**

Một Test Report tối thiểu nên có 4 phần sau:

**Thông tin về dự án**: Bao gồm tên sản phẩm, tên dự án, mô tả dự án và một vài kết quả đáng nổi bật của dự án

**Mục tiêu Kiểm thử (Test Objectives)**: Bao gồm những thông tin như loại test nào được sử dụng, mục tiêu bài test hướng tới là gì

**Tóm tắt Kiểm thử (Test Summary)**: Phần này nên bao gồm những thông tin về sản phẩm vận hành trong bài test như thế nào, test nào pass và test nào failed, hay có test nào không được thực hiện hay không?

**Kết luận về thiếu sót (Defects)**: Đây có thể được coi là phần quan trọng nhất trong Test Report của bạn. Nội dung chính của phần này sẽ nói về trạng thái và ưu tiên cần phải làm. Trong đó, có thể nói về số bugs (lỗi) đã được fix, những việc cần làm, v.v… Để cho sinh động hơn, chúng ta có thể cân nhắc sử dụng biểu đồ, bảng được hoạt hoá, v.v.. cho dễ nhìn.

Test Report của bạn nên có đủ 4 đầu mục trên ở mức tối thiểu, tuy nhiên, những dự án phức tạp hơn sẽ đòi hỏi Test Report có nhiều hơn 4 phần cơ bản trên.

**3 tips để viết Test Report tốt hơn**

Test Report có thể được coi là một công cụ liên lạc giữa Quản lý Kiểm thử và các bộ phận liên quan. Do đó, nếu một Test Report không rõ ràng có thể khiến các bộ phận khác hiểu nhầm và không thể giải quyết vấn đề. Do đó, có một vài lưu ý sau trong quá trình viết test report:

**Chi tiết**: Test Report của bạn nên càng chi tiết càng tốt, chứ không nên chỉ là một tờ giấy và điền những con số vào đó. Một Test Report chi tiết nên miêu tả rõ quá trình test, bao gồm những loại test đã được thử, những test nào sản phẩm failed và tóm tắt lí do. Nếu chỉ có những con số, người đọc sẽ không thể hiểu chi tiết về những gì bạn đã test.

**Gọn gàng**, **Dễ hiểu**: Những thông tin bạn viết vào Test Report nên gọn gàng và dễ hiểu.

**Theo chuẩn chung**: Dù mỗi người có thể có một quy trình làm việc và kiểm thử khác nhau, nhưng team kiểm thử nên thống nhất một mẫu template để tiện theo dõi. Tránh trường hợp mỗi dự án lại có một mẫu Test Report khác nhau, gây mất cân đối trong quá trình kiểm thử giữa các dự án.