LỖI LÀ GÌ. QUẢN LÝ LỖI NHƯ THẾ NÀO ?

1. Thế nào là defect ?

* Defect/ Bugs là lỗi chương trình phần mềm khi sản phẩm phần mềm không thỏa mãn yêu cầu hoặc mong muốn của người dùng . Khi tester tìm thấy lỗi hoặc tìm thấy bất cứ vấn đề nào trong khi thực hiện test , nó sẽ được gọi là defect .
* Defect là độ lệch giữa kết quả mong muốn và kết quả thực tế . Trong quá trình phát triển phần mềm quá trình sửa lỗi được gọi là fix bug

1. Khi nào được báo cáo đây là lỗi ?

* Khi nhìn thấy một thông báo lỗi . Ex error 500 or 400 , server
* Khi dữ liệu hiển thị một trang trắng hoặc thiếu dữ liệu trên trang .
* Khi chương trình không hoạt động như mong đợi
* Khi chương trình bị treo , bị đóng băng ,không thể phản hồi hoặc phản hồi rất chậm
* Khi chương trình không thể trả lời các câu hỏi mà bạn cần.
* Khi chương trình không thể đáp ứng được yêu cầu từ hệ thống, khi không thể tìm ra cách nào đó để thực hiện chức năng mà cần đến
* Khi bạn cần thực hiện chức năng này trên hệ thống nhưng bạn phải đi vòng

=> Khi thấy hệ thống chạy sai so với mong muốn , sai so với yêu cầu thì đó sẽ là một defect .

1. Khi có defect thì phải làm gì ?

* Trong quá trình kiểm thử, bạn chạy theo test case hoặc test tự do để tìm ra lỗi , quá trình này sẽ lặp đi lặp lại cho đến khi kết thúc hoặc đến khi hệ thống đã thỏa mãn với kết quả mong muốn . Trong quá trình phát triển dự án, thực hiện test tích hợp , system test , nhiệm vụ của đội test là tìm ra càng nhiều lỗi càng tốt để giảm số lỗi khi bàn giao cho khách hàng nhằm nâng cao chất lượng dự án.
* Trong thực tế , lỗi có thể liên quan đến phần mềm nhưng cũng có thể nằm trong các sản phẩm trung gian của quá trình phát triển phần mềm (lỗi nằm trong tài liệu đặc tả yêu cầu , tài liệu thiết kế, test plan , test case , test script , tài liệu hướng dẫn sử dụng , tài liệu hướng dẫn cài đặt )
* Nếu bạn tìm thấy một lỗi thì cần báo cáo lỗi đó cho đội phát triển (log defect ) lên hệ quản lý lỗi để đội dev có thể sửa lỗi .
* Sau khi report lỗi xong , đội dev sẽ phân tích lỗi , chấp nhận hoặc từ chối lỗi
* Trong trường hợp đội dev từ chối lỗi thì đội tester sẽ test lại , nếu không phải là lỗi thì ấn hủy lỗi này. Nếu là lỗi thì tester cần cung cấp thêm thông tin( lý do lỗi , lỗi ở thời điểm nào , lỗi trên môi trường nào… ) để đội dev có thể phân tích lỗi và fix lỗi được .

1. Làm thế nào để có thể báo cáo lỗi ?

* Sử dụng một công cụ để quản lý lỗi (jira)
* Nếu dự án không sử dụng một công cụ quản lý lỗi nào thì có thể sử dụng file excel để quản lý, một lỗi thông thường bao gồm những content sau :
* Defect title : tiêu đề của lỗi là gì
* Defect description : mô tả của lỗi
* Severity : Trọng số
* Priority: mức độ ưu tiên
* Defect type : loại lỗi là gì
* Defect Origin : nguồn gốc của lỗi là ở đâu

1. Tìm hiểu chi tiết

Defect title

* Tiêu đề ngắn gọn , rõ ràng để chỉ ra lỗi
* Chúng ta có một cách cơ bản và rõ ràng để nghĩ cách viết tiêu đề lỗi đó là sử dụng cụm từ [P]roblem [A]ction [L]ocation ( trong đó P là vấn đề của hệ thống là gì , A là các hành động nào gây ra lỗi , L là lỗi xảy ra ở đâu)
* Ví dụ : Không nên Log lỗi “Wrong link ‘Open new page’ ” bởi vì lập trình viên không biết lỗi này ở đâu , sai như thế nào , sai về mặt chức năng hay giao diện . Cần phải viết rõ rằng “Link ‘Open new page’ does not work at Home screen” thì dev mới biết lỗi nằm ở đâu và lỗi như thế nào)

Defect description

* Một mô tả lỗi phải rõ ràng để dev có thể xác định chính xác đây là lỗi gì
* Ví dụ : Defect title : Một ứng dụng bị crash khi click vào nút Button trong lúc tạo user mới
* Các bước gây ra lỗi là gì :

1. Login vào hệ thống
2. Đến chức năng tạo người dùng
3. Nhập tất cả thông tin và các trường liên quan
4. Click chuột vào nút Save
5. Hiển thị màn hình lỗi “Insert values Error …”
6. Log file error để dev có thể hiểu rõ hơn.

* Cung cấp các bước để tái tạo lại lỗi
* Kết quả thực tế của dự án
* Kết quả mong đợi của hệ thống này ra làm sao
* Dữ liệu sử dụng trong test này là gì
* Màn hình lỗi như thế nào
* Log file
* Lỗi trong database như thế nào
* Ngoài ra, cần phải cung cấp thêm một số phiên bản chạy code : version phiên bản bao nhiêu , browser , server , test devices ..

Defect Severity: Trọng số của lỗi

* Dựa vào mức độ ảnh hưởng của lỗi, mức độ ảnh hưởng càng lớn thì trọng số càng cao

| Severity | Weight | Description |
| --- | --- | --- |
| Critical/Fatal(nguy hiểm/ chết) | 10 | Lỗi này cực kỳ nghiêm trọng có thể là bị treo, bị chết , không nhận được dữ liệu. Khiến hệ thống bị mất dữ liệu … |
| Serious (lỗi nghiêm trọng) | 5 | Khi các chức năng chính không hoạt động hoặc hoạt động bị sai, các lỗi này ảnh hưởng đến tính năng chính của hệ thống  Ex: Người dùng có thể sử dụng một số hoạt động mà họ không có quyền - lỗi bảo mật thông tin và phân quyền bị sai. |
| Medium | 3 | Khi chức năng hoạt động không đúng hoặc chức năng phụ của hệ thống hoạt động bị sai .  Ex: Hệ thống trả về giá trị sai với dữ liệu không hợp lệ , với các giá trị biên , hoặc hệ thống xử lý chậm |
| Cosmetic | 1 | Lỗi có trọng số là cosmetic là các lỗi liên quan đến giao diện , không ảnh hưởng đến chức năng của hệ thống  Ex : Lỗi sai về layout , các thứ tự phím tab , … |

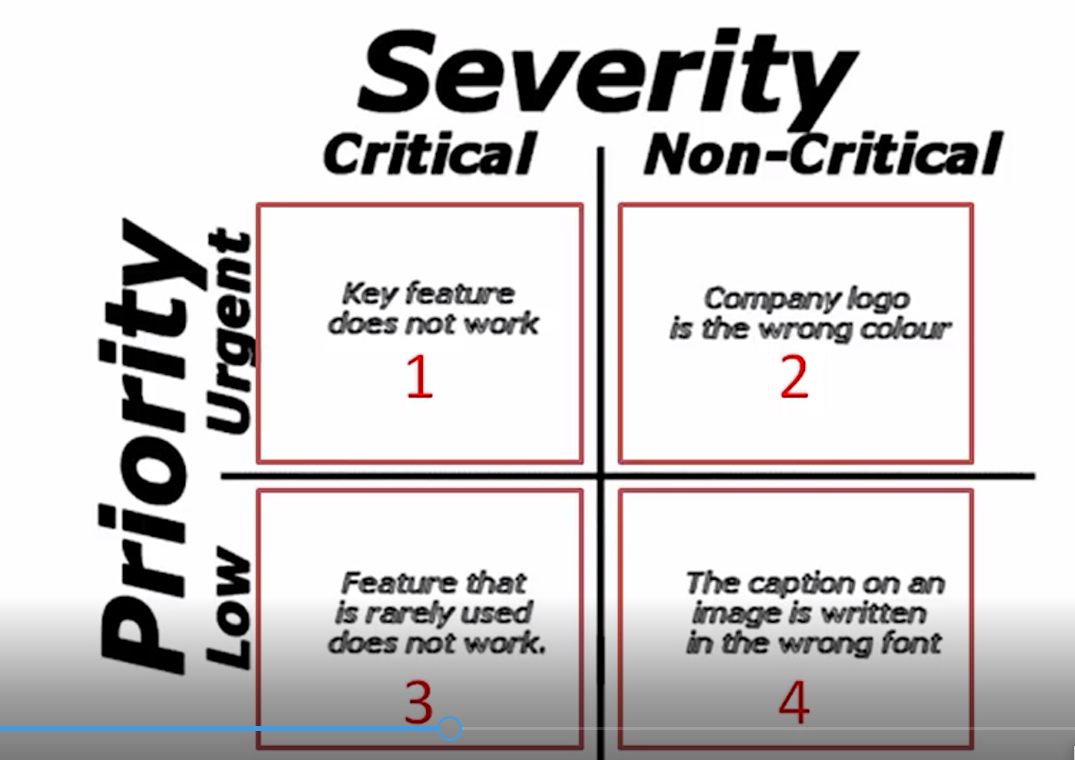
Defect Priority

* Là trả lời cho câu hỏi lỗi cần sửa nhanh hay chậm , mức độ ưu tiên để sửa các lỗi như thế nào

| Defect Priority | Description |
| --- | --- |
| Immediately | Lỗi cần sửa luôn và ngay , có thể là trong 1 ngày làm việc |
| High | Lỗi phải được sửa càng sớm càng tốt |
| Normal | Lỗi có thể được xem xét và sửa trong 4-5 ngày sau khi log |
| Low | Lỗi này có mức độ ưu tiên thấp ( sau khi sửa hết tất cả các lỗi trên thì sửa low cuối cùng ) |

Phân biệt Defect Severity and Priority

* Trọng số và mức độ ưu tiên không phải lúc nào cũng đi đôi với nhau , không phải có lỗi giao diện thì có trọng số cosmetic phải được fix sau cùng



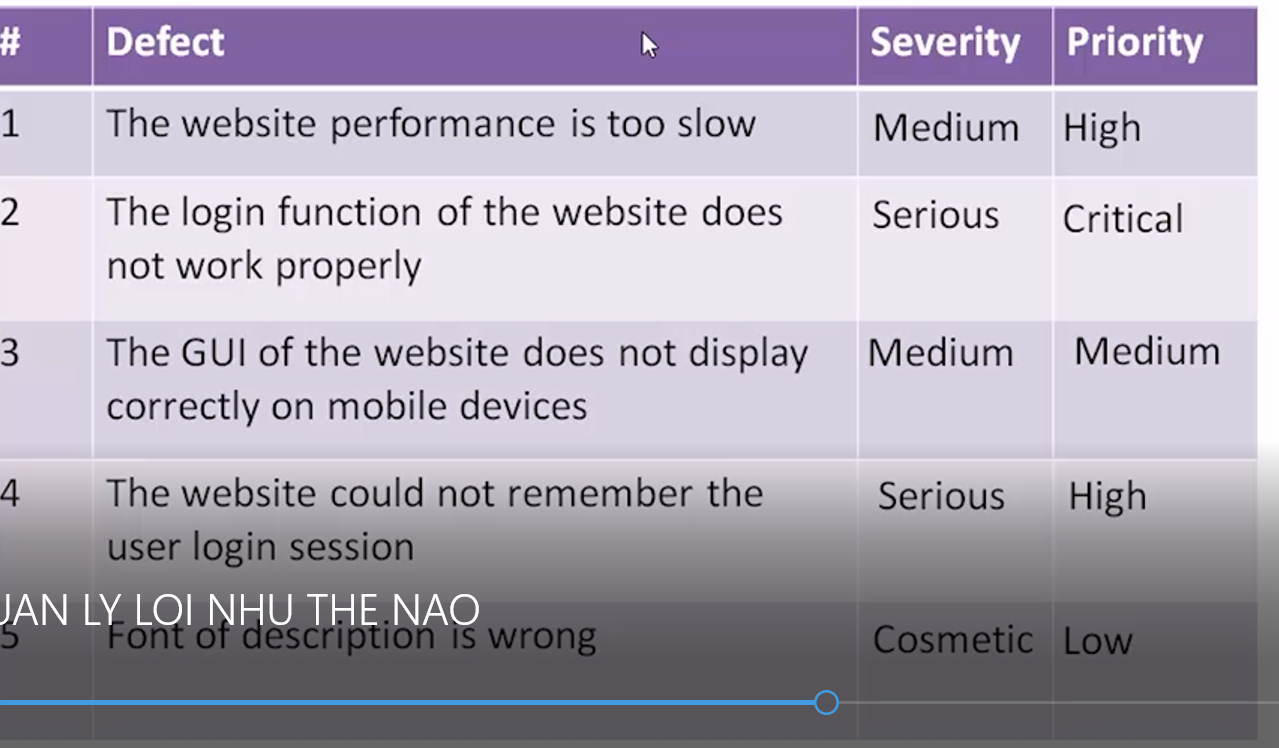
Defect Type

* Việc phân loại lỗi này làm cho việc phân tích nguyên nhân và có các hành động khắc phục và cải tiến quy trình phát triển sản phẩm cũng như cải tiến quy trình kiểm thử dễ dàng và hiệu quả hơn .

| Defect type | Description |
| --- | --- |
| Functional defects | * Sai nghiệp vụ , sai về coding |
| Security-Access control (bảo mật -phân quyền truy cập ) |  |
| Feature missing |  |
| Data-Database integrity( dữ liệu-tính toàn vẹn của dữ liệu) |  |
| Performance -hiệu năng | Thời gian phản hồi của hệ thống chậm |
| Requirement misunderstand- yêu cầu hiểu lầm | Hiểu sai yêu cầu , yêu cầu không đúng , |
| Ambiguous/ Incorrect(không đầy đủ/yêu cầu không rõ ràng  Requirements: | yêu cầu sai hoặc không chính xác hoặc không rõ ràng |
| User interface | Lỗi font chữ , image , color |
| Document | SAS sai , design sai |

Defect origin-Lỗi được xuất phát từ đâu

* requirement
* design
* coding
* testing
* deployment
* Ex

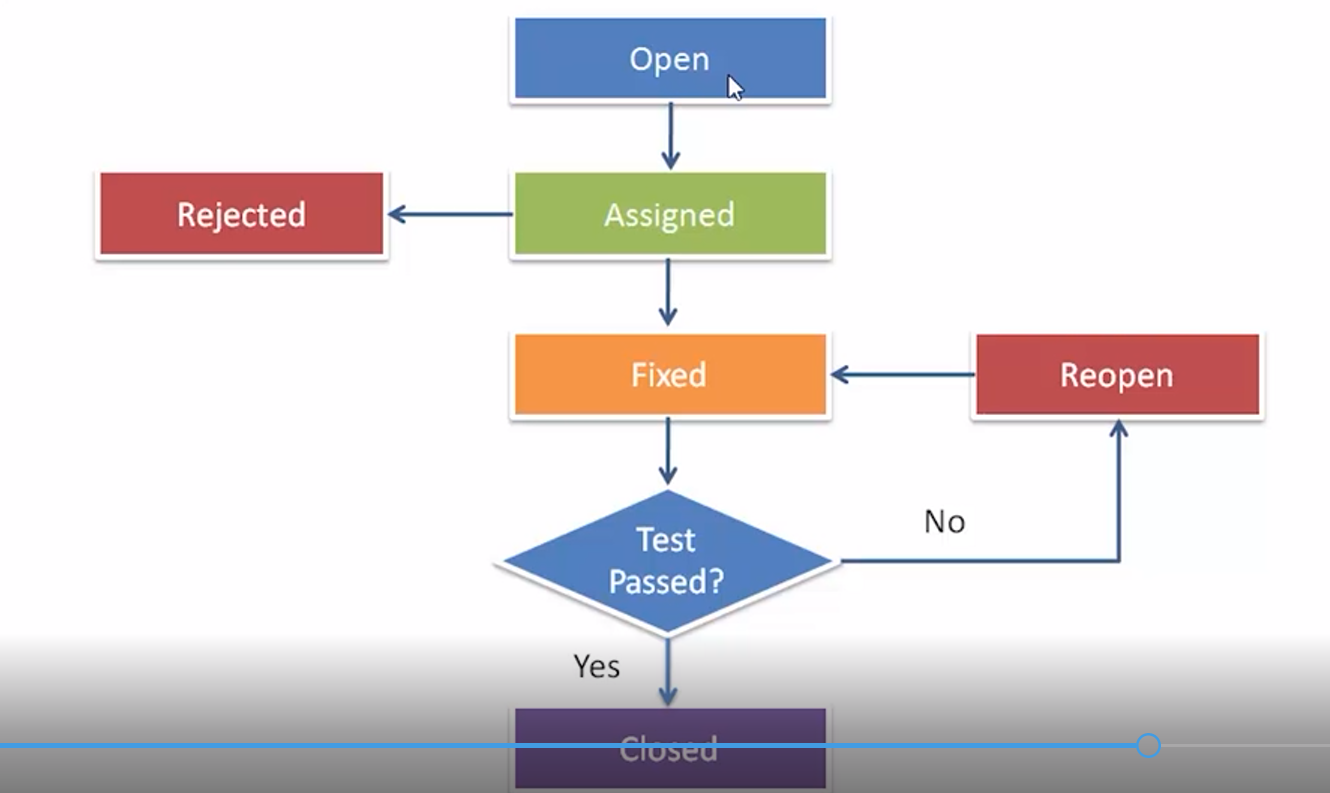


Common defects- Một số lỗi thông thường trên hệ thống

* Đội dev không xem xét kiểm tra đến dữ liệu đầu vào (Xử lý giá trị biên không đầy đủ, chưa xử lý các thẻ , ký tự trắng , ký tự nháy đơn …)
* Over design (Ex nhập ký tự rất dài nhưng không có ký tự trắng nào xem giữa khi đó sẽ làm vỡ trang )
* Data format (Sai format của ngày/tháng , sai format của ký tự kiểu số…)
* Calculator : Công thức tính toán bị sai , chia cho 0…
* Lỗi liên quan đến layout ( dự án có thích ứng trên nhiều độ phân giải khác nhau hay không )
* Ứng dụng của bạn có hỗ trợ trên nhiều thiết bị di động hay không
* Lỗi liên quan đến việc thực hiện phím Enter
* Không setup nút back ở trên trình duyệt
* Xử lý multi languages -hỗ trợ dịch nhiều ngôn ngữ.
* Thông báo tới người dùng không rõ ràng hoặc không có ý nghĩa.
* Hardcode : Một số giá trị nào trên hệ thống mà đội dev đã fix cứng mà các giá trị này phải lấy trong file database (dữ liệu tĩnh)
* Các lỗi liên quan đến giá trị biên , vòng lặp , datetime…
* Kiểm tra các giá trị null, empty , space values

=> Defect Management : Quản lý lỗi là quá trình ghi nhận và xử lý lỗi nhanh nhất có thể, để đưa ứng dụng , hoạt động nghiệp vụ trở lại đúng và bình thường

* Defect life cycle:



* Một số công cụ để log lỗi : jira, BugZilla, Mantis, excel...

=> Việc log lỗi là vô cùng quan trọng và đóng góp rất lớn vào chất lượng của sản phẩm . Do đó , lỗi cần phải log rõ ràng , dễ hiểu , để giảm thời gian phân tích lỗi , giảm thời gian trao đổi giữa dev và tester, tránh các lỗi trùng lặp gây hiểu nhầm hoặc sai lệch , nhóm các lỗi cùng với nhau (cùng nguyên nhân , cùng màn hình ) để đội dev có thể fix triệt để .