**CÁCH LOG LỖI TRONG DỰ ÁN BẰNG JIRA**

Trong bài giảng này, chúng ta sẽ tìm hiểu cách log lỗi trong dự án thực tế.

Mỗi công ty sẽ sử dụng những tool để log lỗi khác nhau. Theo tài liệu chuẩn của môn Software Testing thì họ cũng có quy định những thông tin cần thiết của 1 lỗi.

Và cuối cùng là mỗi công ty lại log lỗi một kiểu. Thậm chí, trong cùng 1 công ty, các dự án có thể lại log lỗi theo các kiểu khác nhau.

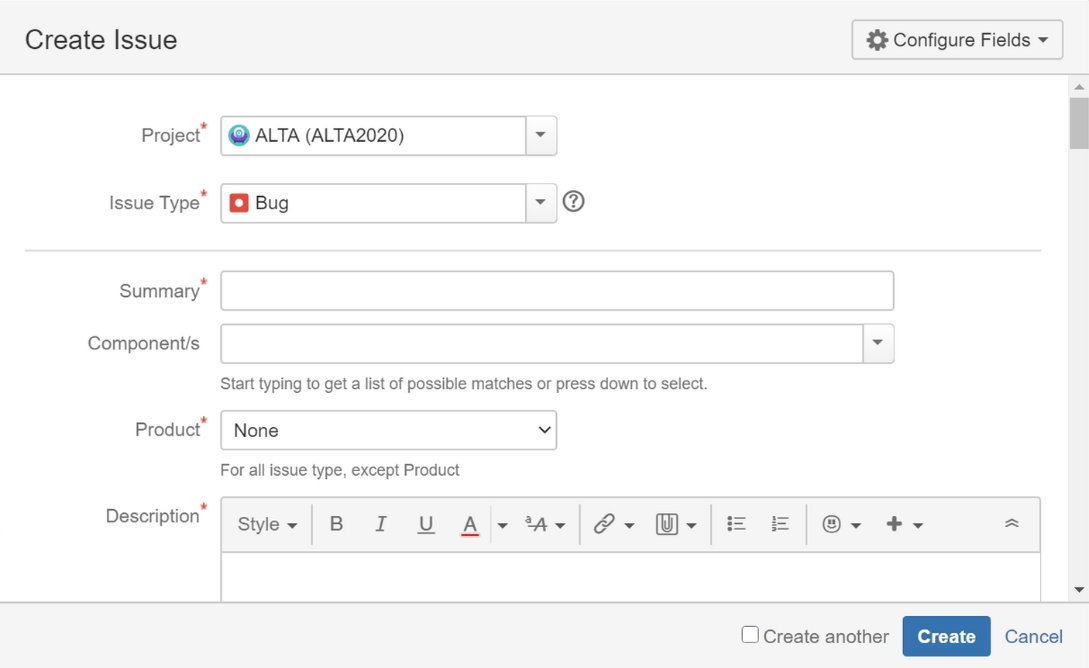
Cho nên ở bài giảng này, mình sẽ giới thiệu cho mọi người cách log lỗi trên 1 hệ thống khá phổ biến hiện tại, đó là JIRA. Các thông tin về lỗi cần phải được log mà mình sắp trình bày ở đây có tính chất tham khảo là chính. Sau này, khi vào công ty, làm dự án thực tế, các bạn hãy chú ý log lỗi theo quy định của công ty và dự án đó.

Trong quá trình mình hướng dẫn ở bài này, mình sẽ cố gắng chọn ra những thông tin cơ bản nhất của 1 lỗi, tức là những thông tin mà ở nhiều công ty và dự án khác nhau, họ thường sử dụng đến. Và mình cũng cố gắng chọn những thông tin quan trọng nhất, có mô tả trong các tài liệu giảng dạy môn Software Testing nữa. Hy vọng sau này các bạn đi làm thực tế, nó cũng không khác nhiều so với những cái này.

Còn vì sao mình chọn JIRA? Vì hiện tại JIRA đang được ưa chuộng. Nó vừa đáp ứng được cho các dự án đang sử dụng mô hình thác nước truyền thống, vừa đáp ứng được cả những dự án đang dùng quy trình Agile/Scrum thịnh hành.

Rồi, bây giờ bắt tay vào xem thử 1 lỗi trong hệ thống JIRA nó thường có các thông tin gì cần phải log.

Muốn tạo 1 bug trong JIRA, chúng ta hãy tạo 1 issue mới như sau:



**Project:** điền tên dự án mình đang làm. Hệ thống sẽ tạo sẵn danh sách dự án.

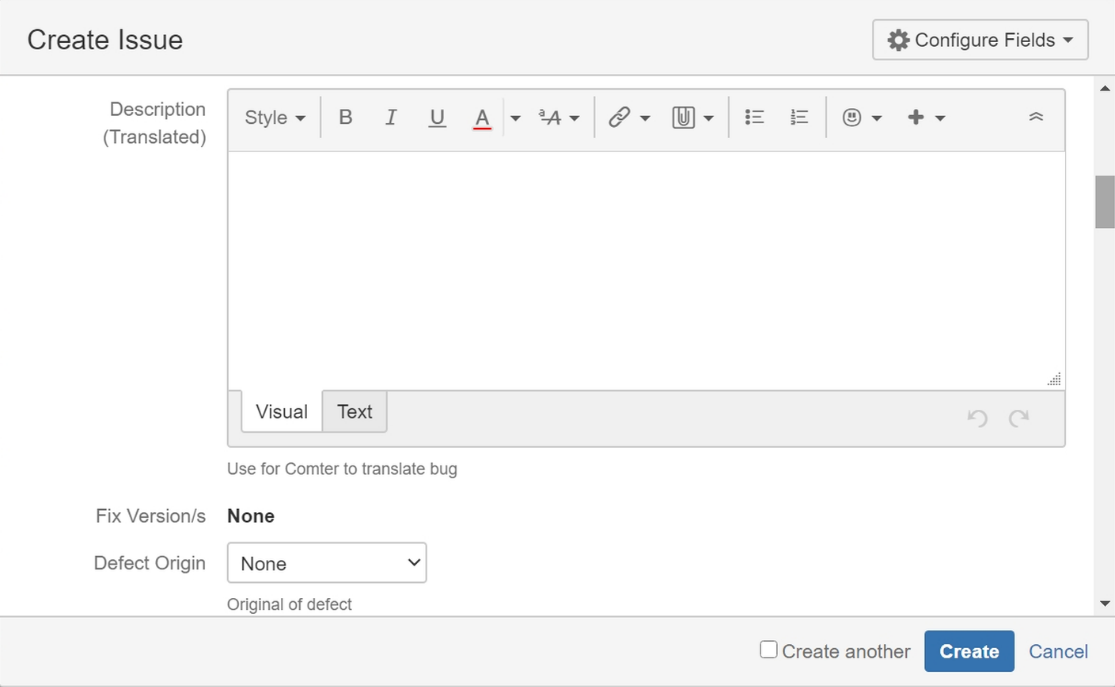
**Issue Type:** điền ”Bug”

**Summary:** điền mô tả tóm tắt về lỗi, *như những gì đã được thực hành testing package Feedback online One nhé. Xem lại file Defect List Sample.xlsx nếu bạn đã quên.*

**Component/s:** Không cần quan tâm.

**Product:** điền sản phẩm mình đang bắt lỗi. Cái này vào dự án sẽ được hướng dẫn chọn cái nào trong danh sách có sẵn.

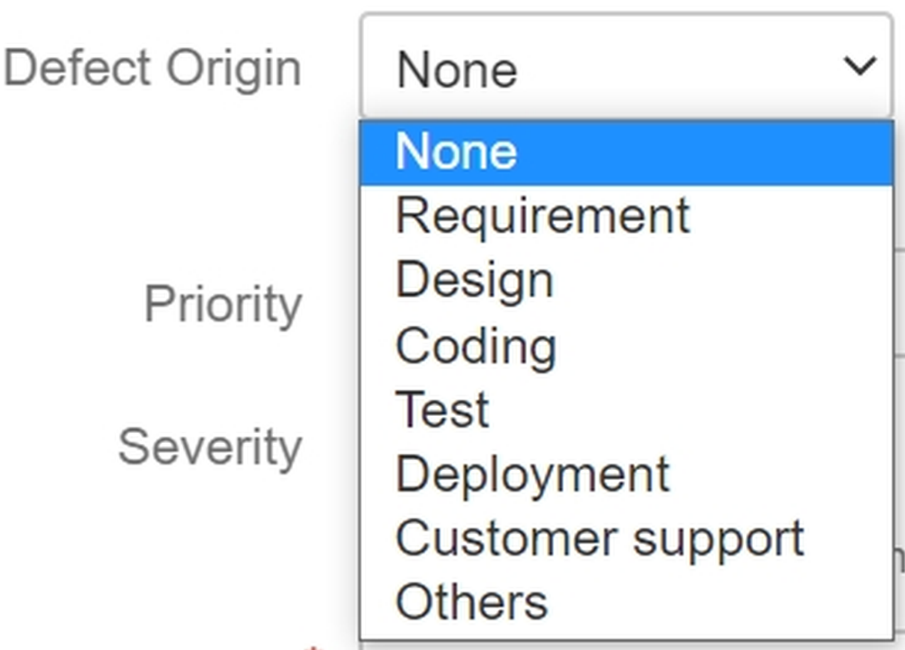
**Description:** điền mô tả cụ thể về lỗi, *như những gì đã được thực hành testing package Feedback online One nhé. Xem lại file Defect List Sample.xlsx nếu bạn đã quên.*



**Fix Version/s:** không cần quan tâm.

**Defect Origin:** điền giai đoạn mình đang bắt lỗi.

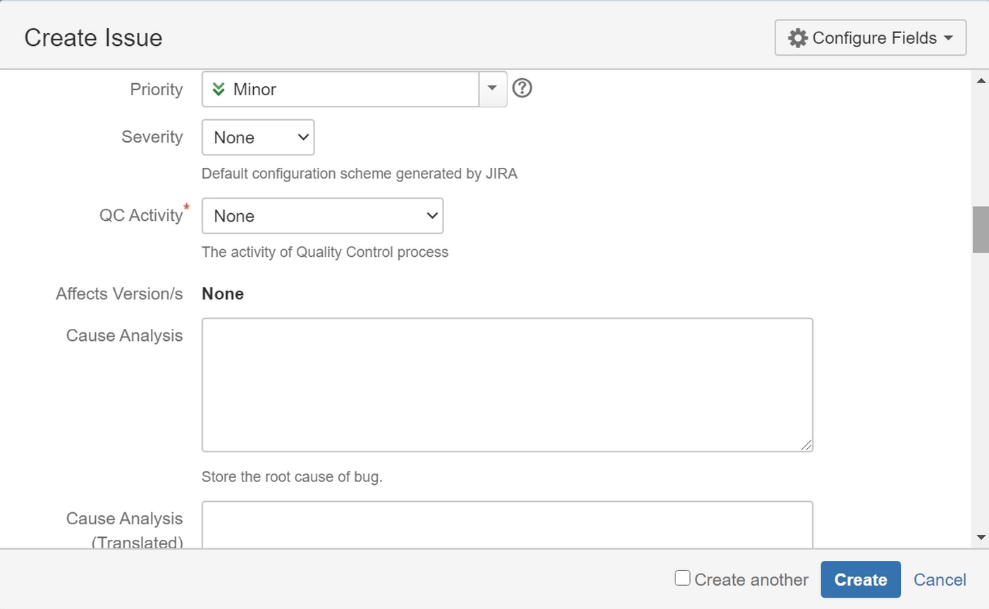
Ví dụ Defect Origin này sẽ có các giá trị là:



Nó chính là các giai đoạn trong mô hình thác nước truyền thống mà mình vừa học xong.

Nếu mình đang kiểm thử phần mềm thì mình sẽ chọn **Test**. Mình mà review và bắt lỗi cho SRS chẳng hạn, thì mình chọn **Requirement**…. Nếu mình thực hiện Unit Testing thì mình nên chọn là **Coding.**

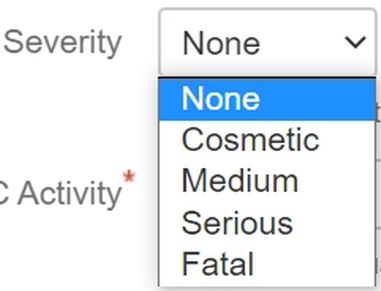
Còn bây giờ, mình đang học kiểm thử phần mềm, chúng ta cứ chọn chỗ **Test** là OK nhé.



**Priority:** là mức độ ưu tiên (của việc fix bug). Cái này là bắt buộc phải chọn. Chú ý là tùy vào thời điểm mà mức độ ưu tiên sẽ khác nhau nhé.

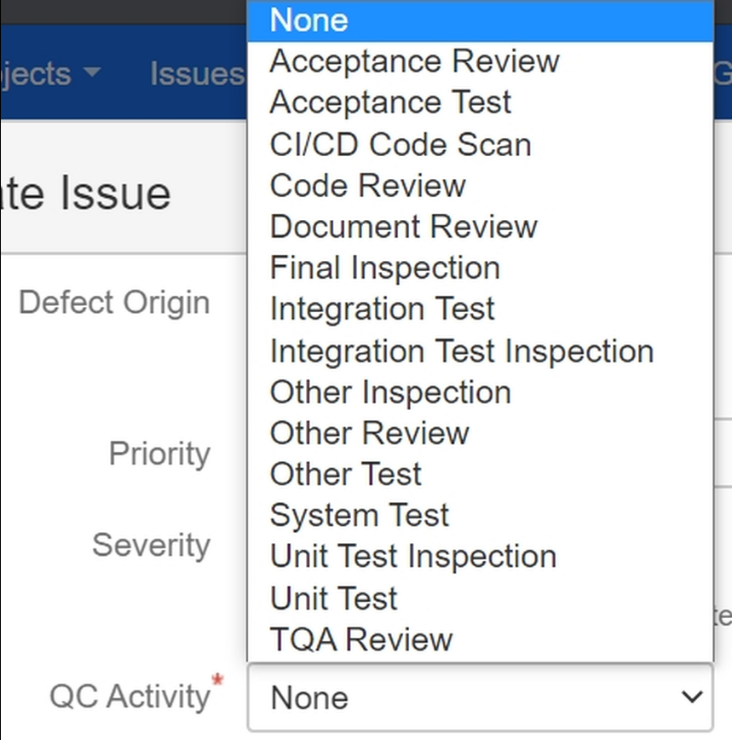
**High (Sửa ngay lập tức), Medium ( Sửa càng nhanh càng tốt) và Low (Lúc nào rảnh thì sửa)**.

**Severity:** điền mức độ nghiêm trọng của bug. Cái này là bắt buộc phải chọn.



* **Cosmetic:** nghĩa là lỗi nhỏ nhặt (Lỗi rất ít nghiêm trọng). Thường là các lỗi về GUI ta sẽ chọn cái này. Nói chung là những lỗi nào chỉ ảnh hưởng đến case đang test và không ảnh hưởng nhiều đến những chức năng khác của phần mềm thì mình chọn cái này.
* **Medium:** là lỗi ở mức độ trung bình (Lỗi hơi nghiêm trọng rồi). Thường là các lỗi liên quan đến chức năng (function) thì mình chọn cái này. Những lỗi medium thường chỉ ảnh hưởng đến 1 màn hình, 1 chức năng, hoặc 1 nhóm chức năng thôi.
* **Serious:** là lỗi ở mức độ nguy hiểm. Thường những lỗi liên quan đến chức năng làm cho cả website hay phần mềm bị treo, tạm thời không thể sử dụng tiếp được; hoặc lỗi sai về nghiệp vụ một cách rất nguy hiểm, có thể gây ảnh hưởng lớn đến nghiệp vụ chung của hệ thống… thì ta mới chọn mức độ này.
* **Fatal:** là ở cực kỳ nguy hiểm (cực kỳ nghiêm trọng). Thường những lỗi như hệ thống bị sập, hoặc buộc phải khởi động lại server… hoặc làm mất dữ liệu của người dùng (ví dụ như xóa sạch dữ liệu trong database hay xóa mất file dữ liệu của user...) thì nên chọn mức độ này.

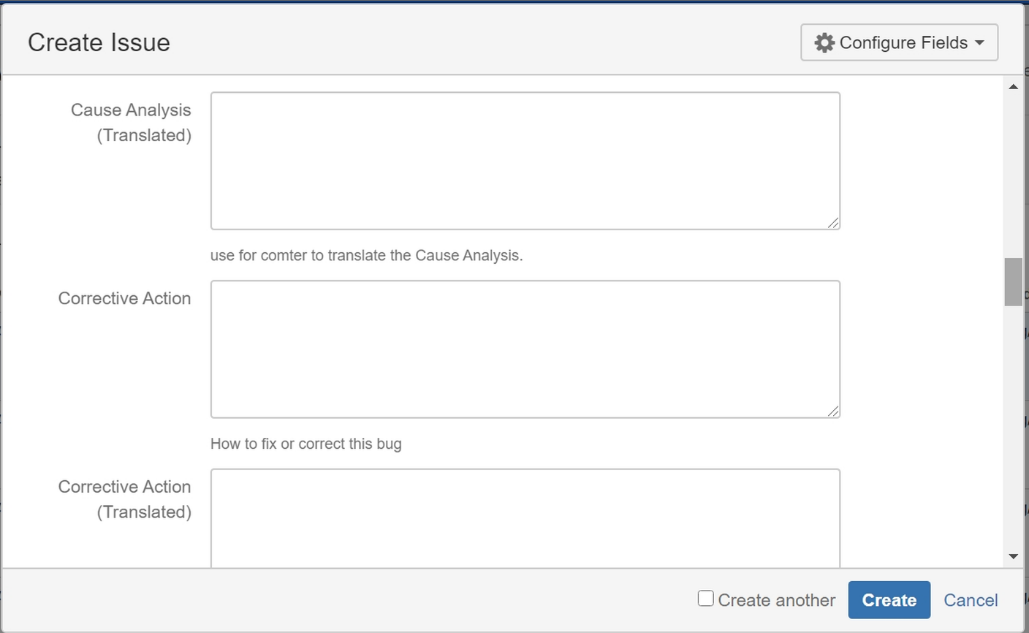
**QC Activity:** sẽ điền tên của loại hoạt động đảm bảo chất lượng mà mình đang thực hiện.



Chúng ta đang làm kiểm thử, cho nên chúng ta sẽ quan tâm đến các **hoạt động kết thúc bằng chữ Test** là chính.

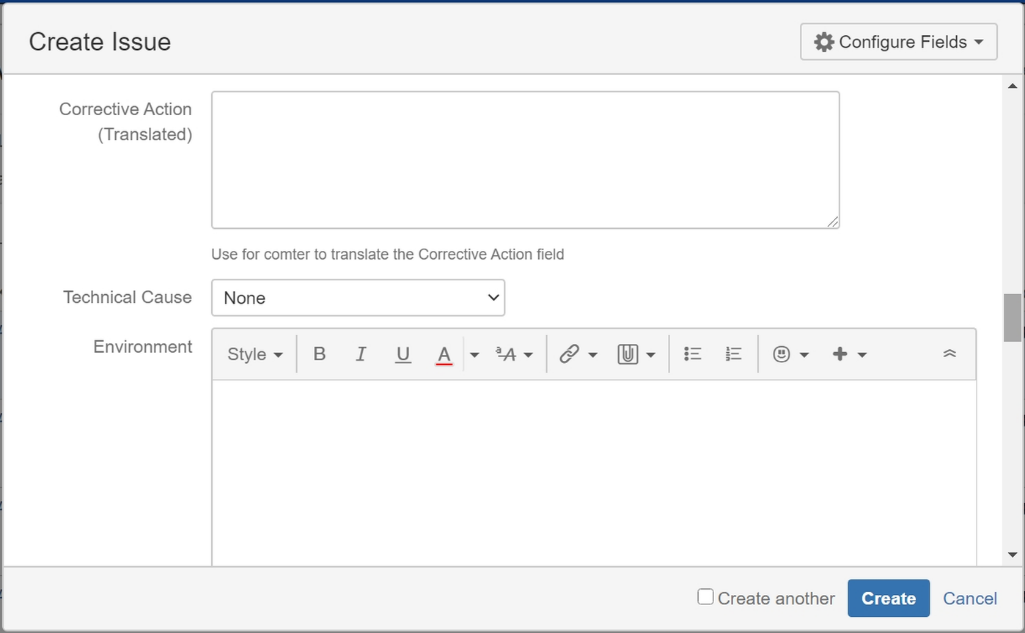
Ví dụ:

* Nếu tester đang làm Integration Test thì điền Integration Test
* Nếu dev đang làm unit test cho module của mình thì điền Unit Test
* Nếu tester hoặc dev hoặc BA đang review SRS thì điền Document Review
* Nếu mình log lỗi do khách hàng bắt được trong quá trình UAT thì mình điền Acceptance Test.



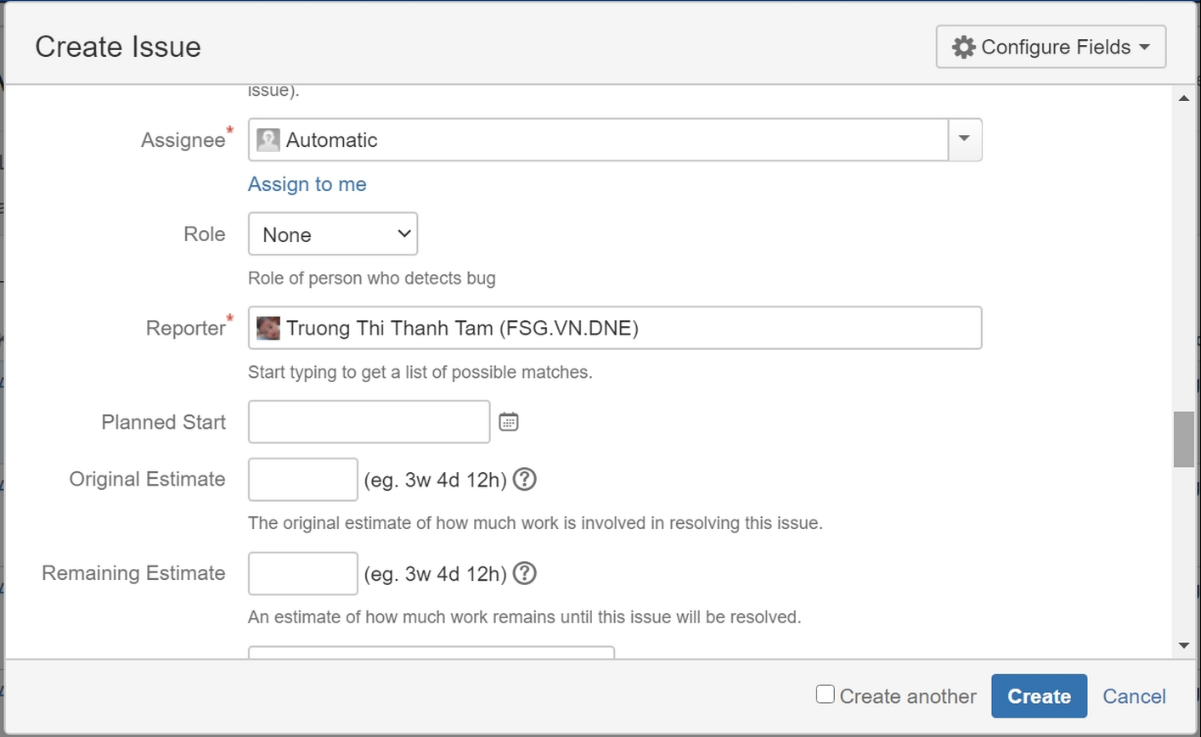
**Cause Analysis:** sẽ điền nguyên nhân sinh ra lỗi này. Cái này là người sửa lỗi sẽ điền vào, chứ người log lỗi thì họ không quan tâm.

**Corrective Action:** sẽ điền hành động (hay cách thức) để sửa lỗi. Cái này là người sửa lỗi sẽ điền vào, chứ người log lỗi thì họ cũng không quan tâm.



**Technical Cause:** sẽ điền nguyên nhân gây ra lỗi (nhìn về phương diện kỹ thuật, tức là nhìn theo quan điểm về lập trình). Tester không quan tâm đến cái này.

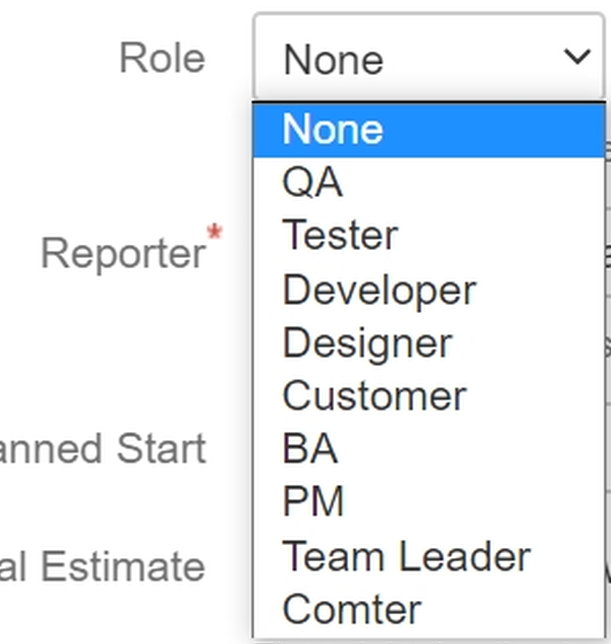
**Environment:** không cần quan tâm.



**Assignee:** sẽ điền account của người được yêu cầu sửa lỗi (tức là người sẽ phải sửa lỗi này).

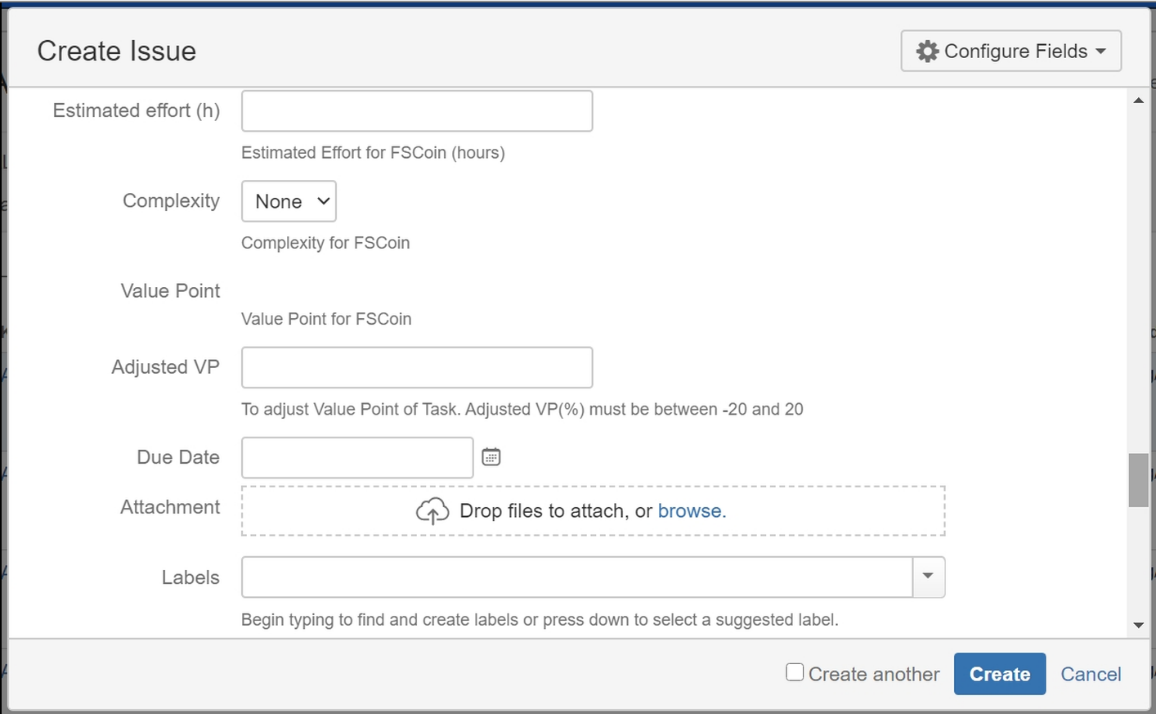
**Role:** sẽ điền vai trò của người log lỗi (không phải vai trò của người fix lỗi nha).

Ví dụ về danh sách role được cho sẵn như bên dưới:

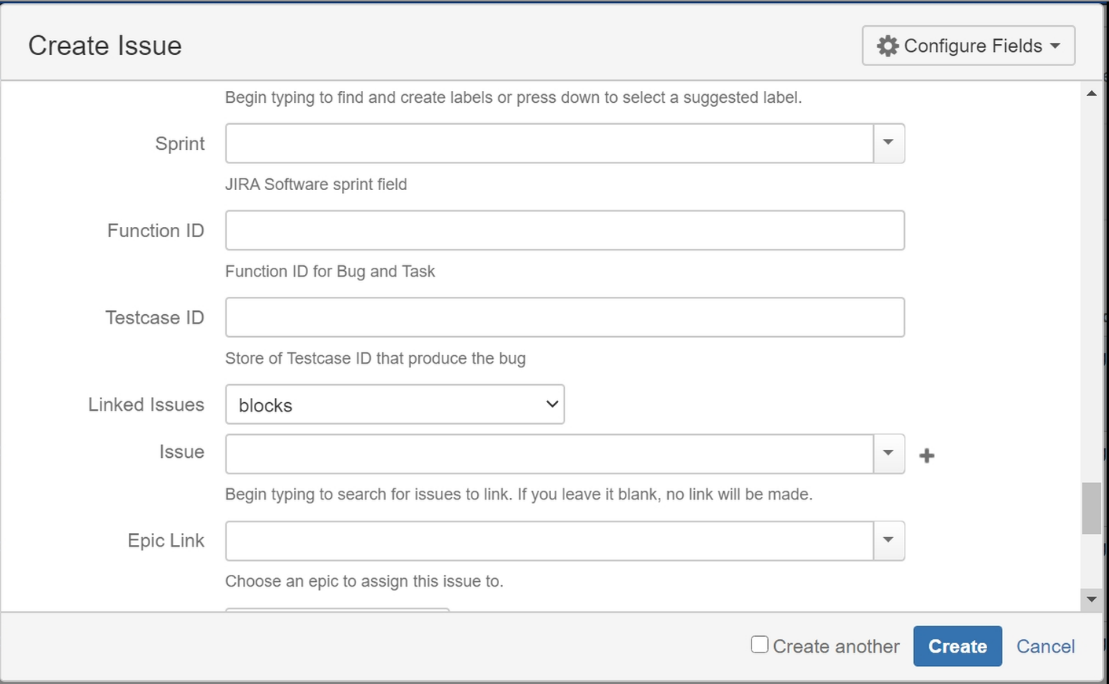


**Reporter:** sẽ điền account của người log lỗi này.

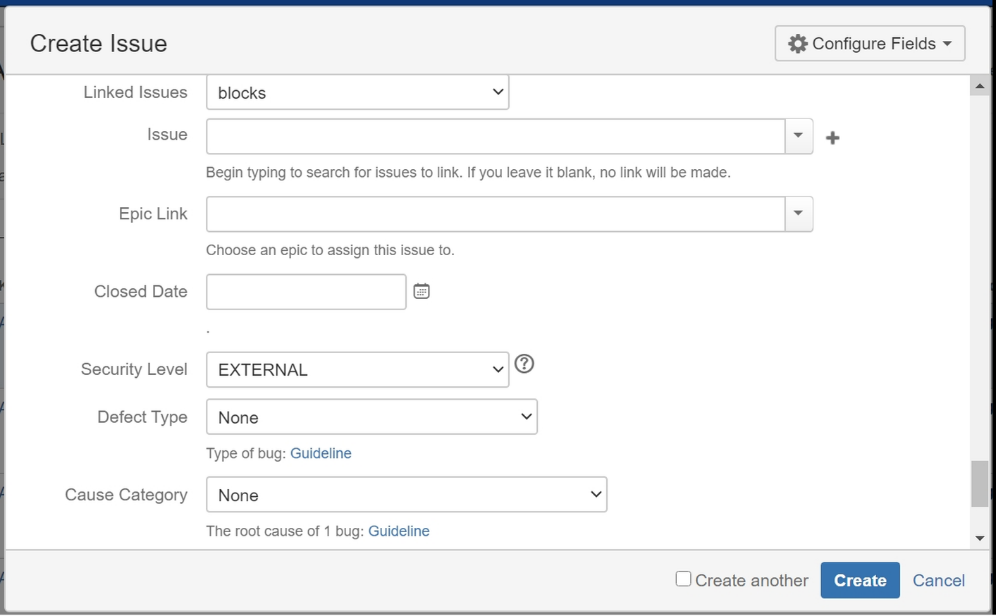
**Planned Start, Original Estimate, Remaining Estimate:** liên quan đến kế hoạch sửa lỗi. Cái này người log lỗi sẽ không quan tâm, cái này là chuyện của team leader hoặc PM dự án. Tester khi log lỗi không cần quan tâm đến mấy thông tin này.



Trong hình trên, chỉ có thông tin **Attachment**là chúng ta sẽ quan tâm, tester có thể attach (upload) các file liên quan đến lỗi ở chỗ này. Ví dụ: ta có thể upload file ảnh chụp màn hình lỗi, file chứa test data cần thiết cho defect,…



Hình ở trên chúng ta chỉ cần nhớ điền **Testcase ID** cho đúng, để biết lỗi hiện tại liên quan đến Testcase nào là OK nhé.

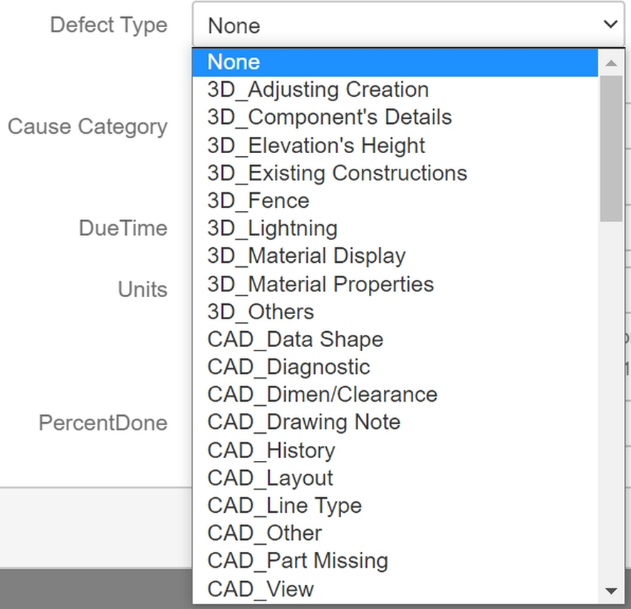


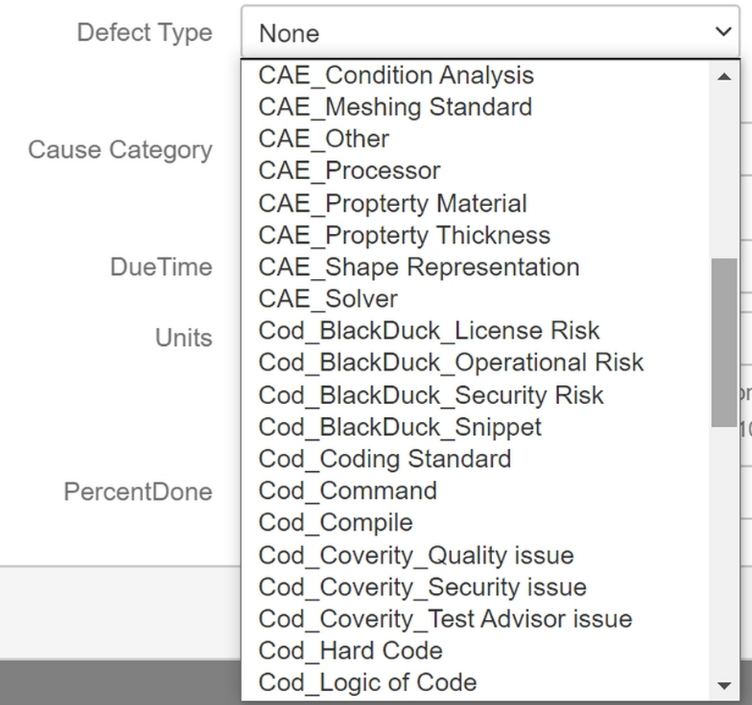
Hình trên thì chúng ta chỉ cần quan tâm đến:

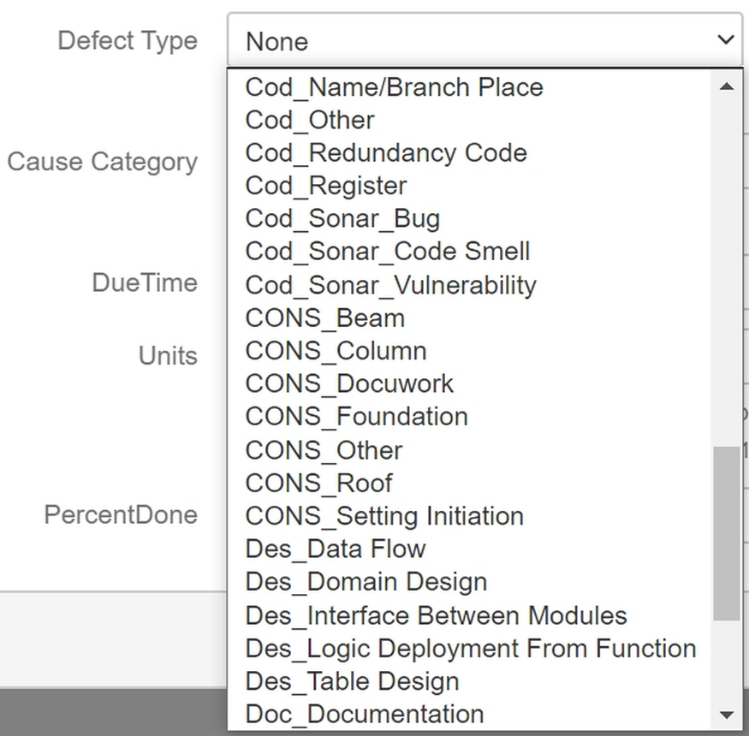
**Closed Date:** điền ngày close lỗi này. Close tức là sau khi ta log lỗi thì người khác sẽ fix lỗi. Sau khi người khác fix lỗi xong, ta kiểm thử lại mà thấy đã hết lỗi rồi thì ta close lỗi. Khi close lỗi ta sẽ điền thông tin này. Còn lúc mới log lỗi thì ta không thể điền thông tin này được.

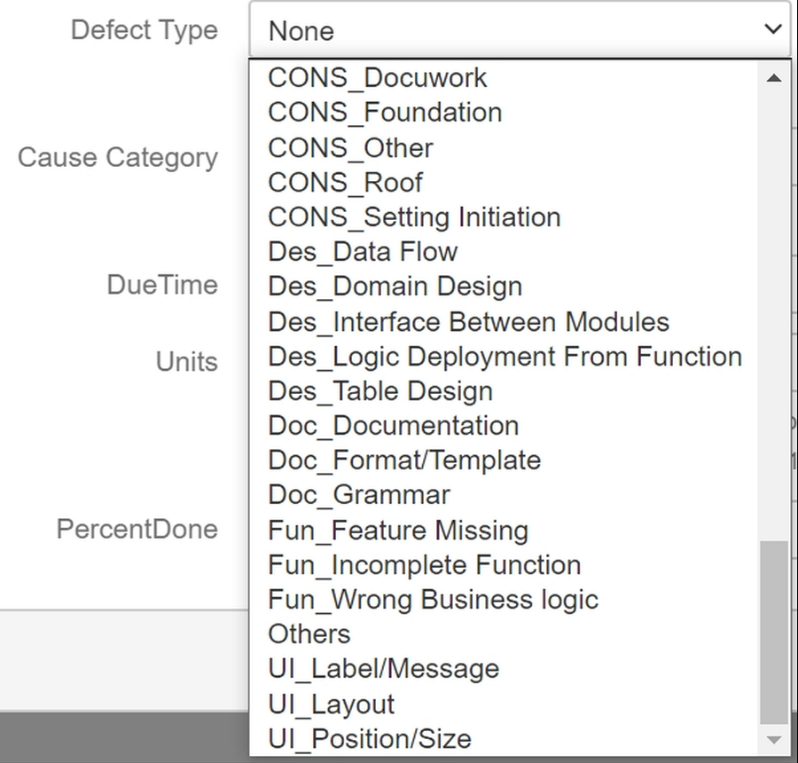
**Defect Type:** điền loại lỗi. Thường danh sách của Defect Type có rất nhiều item.

Ví dụ:



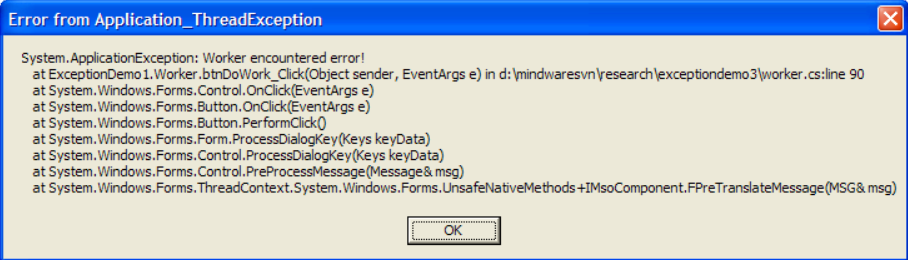


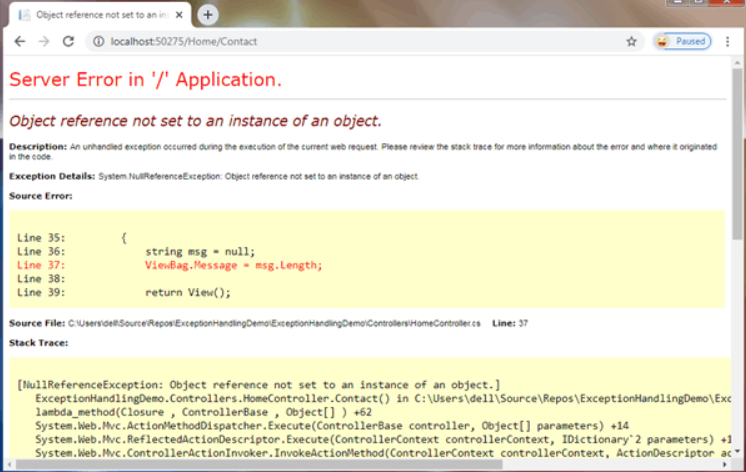




Rất nhiều cái để chọn lựa. Nhưng tester thường chỉ chọn các loại sau:

**Cod\_Logic of Code:** chọn cái này khi app hiển thị các message/page lỗi liên quan đến code làm cho chức năng (function) của phần mềm không thể hoàn thành (xem hình ví dụ bên dưới) hoặc khi app xử lý các field/chức năng không theo một logic nào cả. Nếu ta đoán là lập trình viên họ hiểu đúng nghiệp vụ nhưng mà code bị sai logic thì ta chọn loại lỗi này.





**Fun\_Feature Missing:** chọn cái này khi phần mềm thiếu hẳn 1 hoặc một số chức năng. Ví dụ chức năng quản lý Nhân Viên mà lập trình viên quên code màn hình Chỉnh Sửa Nhân Viên hoặc quên code cho chức năng của 1 button nào đó.

**Fun\_Incomplete Function:**chọn cái này nếu như chức năng của phần mềm chưa hoàn thiện (làm thiếu expected output của test case). Ví dụ case tạo mới Nhân Viên thành công có 4 expected output nhưng app chỉ mới thực hiện được 3 expected output.

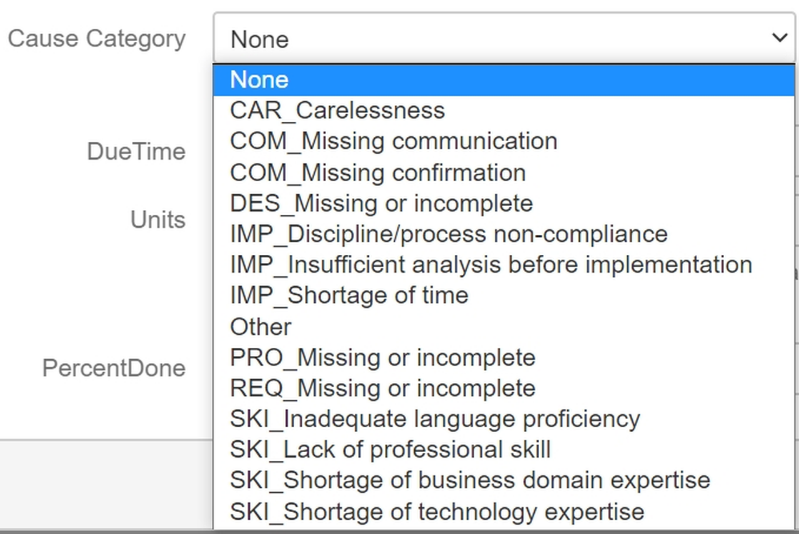
**Fun\_Wrong Business logic:**chọn cái này nếu như app đang chạy sai nghiệp vụ hay expected output trong test case nhưng không gây ra các thông báo lỗi như đã nêu ở loại Cod\_Logic of Code nhé.

Khi phân vân giữa 2 loại lỗi **Cod\_Logic of Code**và **Fun\_Wrong Business logic**thì nên đoán và chọn đại 1 cái thôi, chờ cho dev fix xong defect và điền Cause Analysis thì chúng ta sẽ căn cứ vào đó và update lại loại lỗi cho đúng. Chứ làm sao chúng ta biết là lập trình viên họ hiểu sai nghiệp vụ và code hay họ hiểu đúng nghiệp vụ và code sai được. Cái này chỉ có người làm ra lỗi họ mới biết thôi. Mà trong dự án thực tế, đa phần các lỗi được log có liên quan đến 2 loại lỗi này là nhiều nhất.

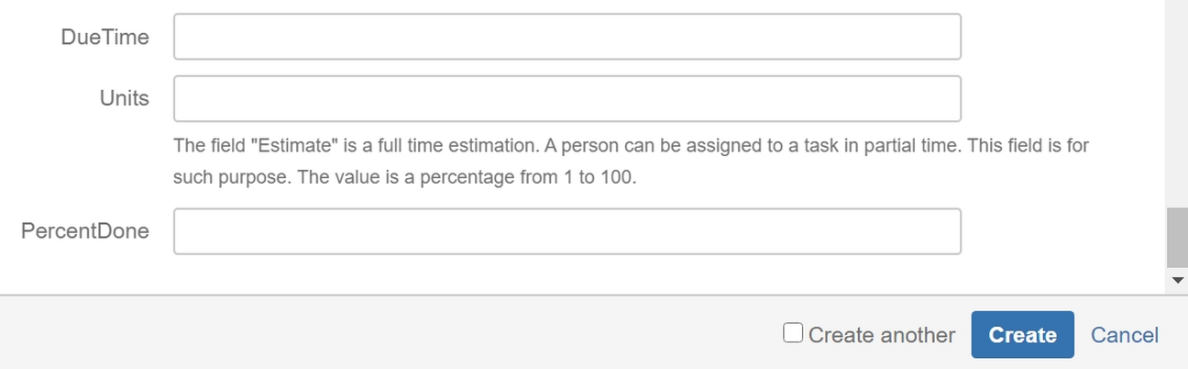
**UI\_XXX (3 loại lỗi liên quan đến giao diện):**chọn 1 trong 3 loại này khi ta log các lỗi liên quan đến giao diện (GUI).

**Cause Category:** điền nhóm nguyên nhân (loại nguyên nhân) chính của lỗi.

Ví dụ về danh sách Cause Category như bên dưới:



Thường tester bỏ qua, để lập trình viên họ sửa xong thì họ chọn thông tin này. Hoặc nếu bị ép phải chọn thì tester hay chọn REQ\_Missing or incomplete.



Mấy thông tin ở hình trên thì ta có thể không quan tâm.

Như vậy là đã tìm hiểu các thông tin trong quá trình log lỗi trong 1 dự án thực tế.

Dĩ nhiên, mỗi tool, mỗi dự án, mỗi công ty có thể thêm bớt những thông tin mà họ quan tâm.

Các bạn học viên có cảm giác như thế nào?

Vào dự án thực tế, đúng là có quá nhiều cái cần phải được hướng dẫn mới có thể làm được đúng không nào? Ngay cả việc log lỗi mà cũng thấy khá là phức tạp, rườm rà rồi.

Chúng ta không còn cách nào khác là phải làm quen với sự khó chịu, phải sống chung với lũ thôi.

Có 1 lưu ý nhỏ: ở các công ty phần mềm nhỏ, nhân sự rất ít người, thì họ thường bỏ qua các quy trình rườm rà này. Họ chỉ tập trung vào việc làm sản phẩm. Việc kiểm thử cũng không nặng nề về vấn đề phải làm theo đúng quy trình này nọ.

Tuy nhiên, nếu các bạn được làm trong các dự án lớn, có quy trình bài bản, bạn sẽ học được rất nhiều. Đặc biệt, khi đi thi các chứng chỉ quốc tế, bạn sẽ thấy được lợi ích của việc được tham gia các dự án có quy trình 1 cách bài bản. Cái gì nó cũng có cái hay cái dở riêng của nó đúng không nào?