**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC KINH TẾ**

🙣 🕮 🙡

****

**BÁO CÁO LẦN 1**

**THỰC TẬP NGHỀ NGHIỆP**

**CÔNG TY TMA BÌNH ĐỊNH**

SVTH:   Trương Huyền Trang

Lớp:       44K14

GVHD:  Cao Thị Nhâm

***Đà Nẵng, tháng 7 năm 2021.***

**CƠ SỞ LÝ THUYẾT**

# Cơ hội nghề nghiệp

## Mô tả vị trí việc làm về Testing

* Xem xét yêu cầu phần mềm và chuẩn bị các kịch bản thử nghiệm
* Thực hiện các bài kiểm tra về khả năng sử dụng phần mềm
* Phân tích kết quả kiểm tra về tác động cơ sở dữ liệu hoặc lỗi và khả năng sử dụng
* Chuẩn bị báo cáo về tất cả các khía cạnh liên quan đến thử nghiệm phần mềm và gửi cho nhóm thiết kế
* Tương tác với khách hàng để hiểu các yêu cầu sản phẩm
* Tham gia đánh giá phần mềm và cung cấp đầu vào theo yêu cầu, thiết kế sản phẩm và phát hiện các vấn đề tiềm ẩn

## Các kỹ năng cần có

* Bằng cử nhân/ cao đẳng các chuyên ngành khoa học máy tính, công nghệ thông tin hoặc liên quan
* Kiến thức về thiết kế, kiểm thử phần mềm và thành thạo các phương pháp thử nghiệm
* Thành tạo các kỹ thuật kiểm thử phần mềm cũng như đánh giá khả năng tương thích với các chương trình phần mềm khác nhau
* Kiến thức về lập trình
* Giao tiếp tốt và kỹ năng tư duy phản biện
* Kỹ năng tổ chức tốt và tư duy định hướng chi tiết

## Mức lương khởi điểm dành cho fresher

* Mức lương trung bình phổ biến các của nhân viên Tester hiện nay dao động 15 triệu đồng/ tháng

# Software Testing Overview

## Introduction to Software Testing

## Objectives of Testing

* Kiểm thử phần mềm là bắt buộc để kiểm tra độ tin cậy của phần mềm
* Kiểm thử phần mềm để đảm bảo rằng hệ thống không có bất kỳ lỗi nào
* Kiểm thử phần mềm đảm bảo rằng sản phẩm phù hợp với yêu cầu của khách hàng
* Đảm bảo rằng sản phẩm cuối cùng thân thiện với người dùng

## Software Development Process

* Stage 1: Kế hoạch và phân tích yêu cầu
* Stage 2: Xác định yêu cầu
* Stage 3: Thiết kế kiến trúc sản phẩm
* Stage 4: Xây dựng hoặc phát triển sản phẩm
* Stage 5: Kiểm tra sản phẩm
* Stage 6: Triển khai trên thị trường và duy trì

## Error / Fault / Failure

* Error: Lỗi là hành động của con người tạo ra kết quả không chính xác dẫn đến lỗi
* Bug: Sự hiện diện của lỗi tại thời điểm thực thi phần mềm
* Fault: Trạng thái của phần mềm do lỗi
* Failure: Sự sai lệch của phần mềm so với kết quả mong đợi của nó

## Verification & Validation

## QA:

* + Đảm bảo chất lượng đảm bảo rằng team đang làm những điều đúng đắn
  + QA tập trung vào việc xây dựng chất lượng và do đó ngăn ngừa các khuyết tật
  + QA giải quyết quá trình
  + QA là cho toàn bộ vòng đời
  + QA là quá trình phòng ngừa
* **QC**
  + Kiểm soát chất lượng đảm bảo kết quả của những gì team đã làm là những gì chúng tôi mong đợi
  + QC tập trung vào việc kiểm tra chất lượng và do đó phát hiện ra các khuyết tật
  + QC giao dịch với sản phẩm
  + QC dành cho phần kiểm tra trong SDLC
  + QC là quá trình sửa chữa

# Software Testing Life Cycle (SDLC)

## SDLC Models: Water Fall

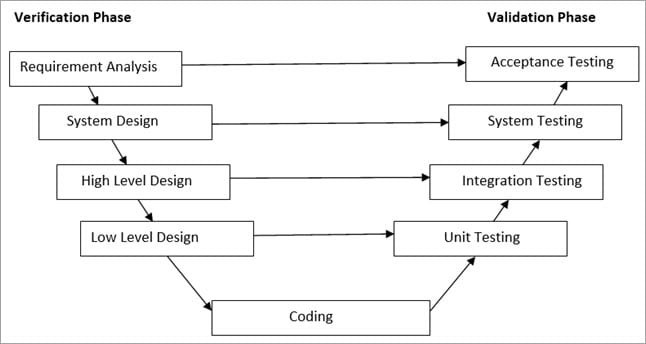
* Là mô hình tuần tự chia việc phát triển phần mềm thành các giai đoạn được xác định trước. Mỗi giai đoạn phải được hoàn thành trước khi giai đoạn tiếp theo có thể bắt đầu mà không có sự chồng chéo giữa các giai đoạn. Mỗi giai đoạn được thiết kế để thực hiện hoạt động cụ thể trong giai đoạn.
* Mô hình thác nước có thể sử dụng khi:
* Yêu cầu không thay đổi thường xuyên
* Ứng dụng không phức tạp và lớn
* Dự án ngắn
* Yêu cầu rõ ràng
* Môi trường ổn định
* Công nghệ và công cụ được sử dụng không năng động và ổn định
* Nguồn lực có sẵn và được đào tạo



*Hình 1: Mô hình waterfall*

## SDLC Models: V Model

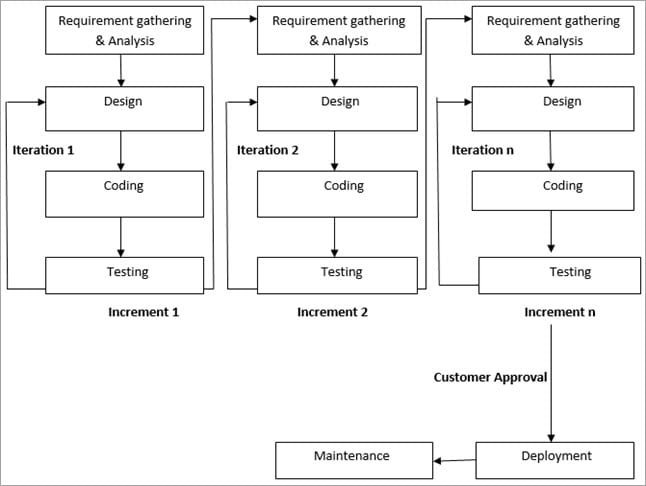
* Mô hình chữ V còn được gọi là mô hình xác minh và xác thực. Trong mô hình này, Xác minh và Xác thực song hành với nhau, tức là phát triển và kiểm tra song song với nhau.

****

*Hình2: Mô hình chữ V*

## SDLC Models: Agile Model

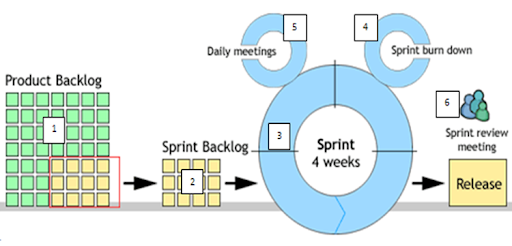
* Là một tập hợp các phương thức phát triển lặp và tăng dần trong đó các yêu cầu và giải pháp được phát triển thông qua sự liên kết cộng tác giữa các nhóm tự quản và liên chức năng.
* **Nguyên tắc áp dụng trong phương pháp Agile**
  + - * Kiểm thử giúp dự án nhanh chóng được bàn giao
      * Kiểm thử không chỉ là một giai đoạn của dự án
      * Cá nhân và sự tương hỗ quan trọng hơn quy trình
      * Rút ngắn vòng lặp phản hồi
      * Thỏa mãn mong muốn của khách hàng
      * Giữ code rõ ràng
      * Giản lược tài liệu kiểm thử
      * Chưa hoàn thành khi chưa qua giai đoạn kiểm thử
      * Test Last  và Test Driven



*Hình 3: Mô hình Agile*

## Scrum Methodology

* Scrum là một framework về quy trình và quản lý giúp giải quyết các vấn đề phức tạp, nhưng vẫn đảm bảo tính hiệu quả, sáng tạo và sản phẩm được tạo ra phải đạt được giá trị cao nhất. Nó khuyến khích các team học hỏi qua thực nghiệm, tự tổ chức khi giải quyết vấn đề, cũng như phản ánh những mặt đạt được và thiếu sót để từ đó từng bước phát triển
* Product owner (PO): là người phụ trách sản phẩm, có quyền hạn trong việc quyết định tính năng của sản phẩm, quyết định thứ tự, độ ưu tiên trong quá trình phát triển dự án.
* Scrum master: là người hỗ trợ cho Product owner và team. Scrum master không quản lý team mà giúp team giải quyết những vấn đề cản trở cho việc đạt mục tiêu của team.
* Scrum team: là toàn bộ thành viên còn lại của dự án.
* Các đối tượng, phương tiện của scrum:
* Product backlog: toàn bộ danh sách các yêu cầu, tính năng của sản phẩm. Danh sách này sẽ được cập nhật thường xuyên bởi PO và bất kỳ ai trong team Độ ueu tiên thứ tự của các tính năng sẽ được PO quyết định.
* Sprint: 1 giai đoạn của dự án với thời gian cố định. Độ dài của 1 sprint sẽ được team và PO quyết định. Thông thường là từ 1-4 tuần.
* Sprint backlog: các tính năng của sản phẩm sẽ được làm trong 1 sprint.
* Các nghi thức trong scrum:
* Sprint planing meeting: Làm khi bắt đầu 1 sprint, PO thường sẽ mô tả và đưa ra thứ tự của các item trong Product backlog. Scrum team có thể đặt câu hỏi cho PO để hiểu hơn về sản phẩm.
* Daily meeting: thực hiện hàng ngày và từng thành viên trong team nếu có vấn đề trả lời cho 3 câu hỏi:
  + - Hôm qua làm gì?
    - Hôm nay sẽ làm gì?
    - Có gặp vấn đề gì không?



Hình 4: Mô hình S

# Software Testing Type

## Manual testing

* Kiểm thử thủ công - là kiểm tra một phần mềm theo cách thủ công, tức là không sử dụng bất kỳ công cụ tự động nào hoặc bất kỳ tập lệnh nào**.**

## Automation Testing

* Kiểm thử tự động - là người kiểm thử viết các kịch bản và sử dụng một tool, phần mềm khác để kiểm tra sản phẩm.
* Kiểm thử tự động được sử dụng để chạy lại các kịch bản kiểm tra đã được thực hiện theo cách thủ công, nhanh chóng và lặp đi lặp lại

# Software Testing Methodologies

## White Box Testing

* **Khái niệm:**

Là phương pháp kiểm thử phần mềm mà người test biết về cấu trúc. Người kiểm tra chọn đầu vào để thực hiện các đường dẫn thông qua mã và xác định đầu ra thích hợp.

* **Đối tượng áp dụng:**

Một thành phần phần mềm có thể là một hàm chức năng, một module chức năng, một phân hệ chức năng...

* **Mức độ áp dụng:**

Thích hợp dùng để kiểm thử đơn vị, còn đối với thành phần phần mềm quá lớn thì không nên sử dụng vì phương thức này tốn nhiều thời gian, công suất và hiệu quả không cao

* **Ưu điểm:**
* Test có thể bắt đầu ở giai đoạn sớm hơn, không cần phải chờ đợi cho GUI để có thể test.
* Test kỹ càng hơn, có thể bao phủ hầu hết các đường dẫn.
* Thích hợp trong việc tìm kiếm lỗi và các vấn đề trong mã lệnh
* Cho phép tìm kiếm các lỗi ẩn bên trong.
* Các lập trình viên có thể tự kiểm tra.
* Giúp tối ưu việc mã hoá.
* Do yêu cầu kiến thức cấu trúc bên trong của phần mềm, nên việc kiểm soát lỗi tối đa nhất.
* **Nhược điểm:**
* Vì các bài kiểm tra rất phức tạp đòi hỏi phải có nguồn lực có tay nghề cao, với kiến thức sâu rộng về lập trình và thực hiện
* Không thể tìm thấy tính năng chưa thực hiện hoặc bỏ sót.
* Maintenance test script có thể là một gánh nặng nếu thể hiện thay đổi quá thường xuyên

## Black Box Testing

* **Khái niệm:**
* Là kiểm tra hành vi, kiểm tra hộp kín, kiểm tra dựa trên các thông số kỹ thuật hoặc kiểm tra bằng mắt.
* Là phương pháp kiểm tra phần mềm phân tích chức năng của phần mềm, ứng dụng mà không cần biết về cấu trúc bên trong
* **Mức độ áp dụng:**

Có thể áp dụng hầu như đến mọi cấp độ của kiểm thử phần mềm, tuy nhiên thích hợp được thực hiện trong System Test và Acceptance Test

* **Ưu điểm:**
* Thích hợp trong việc kiểm tra từng phân đoạn lớn các mã lệnh, chức năng lớn
* Người thử nghiệm không cần hiểu biết về mã lệnh được viết trong chương trình
* Tách biệt giữa quan điểm người sử dụng và người phát triển phần mềm
* **Nhược điểm:**
* Độ bao phủ hạn chế vì chỉ có một phần nhỏ trong số các kịch bản thử nghiệm được thực hiện
* Kiểm tra không hiệu quả do người thử nghiệm không hiểu biết gì về cấu trúc bên trong phần mềm
* Tester có hạn chế hiểu biết về ứng dụng

## Gray Box Testing

* **Khái niệm**
* Là sự kết hợp phương pháp kiểm thử hộp đen và hộp trắng
* Trong kiểm thử hộp xám cấu trúc bên trong sản phẩm chỉ được biết một phần
* **Mức độ áp dụng**

Có thể được sử dụng ở nhiều mức kiểm thử khác nhau. Tuy nhiên chủ yếu được ứng dụng trong Integration Test.

# Software Testing Levels

## Unit Testing

* Là một kiểu kiểm thử phần mềm trong đó các đơn vị hoặc thành phần riêng lẻ của một phần mềm được kiểm tra.
* Mục đích: để xác nhận rằng mỗi đơn vị của mã phần mềm hoạt động như mong đợi.
* Kiểm thử đơn vị được thực hiện bởi nhà phát triển.

## Integration Testing

* Là kiểm thử các mô đun phần mềm được tích hợp một cách logic và được kiểm tra như một nhóm.
* Mục đích: là để lộ ra các khiếm khuyết trong sự tương tác giữa các mô đun phần mềm này khi chúng được tích hợp.

## System Testing

* Là cấp độ kiểm tra xác nhận sản phẩm phần mềm hoàn chỉnh và được tích hợp đầy đủ.
* Mục đích: đánh giá các thông số kỹ thuật của hệ thống.
* Được thực hiện bởi những người phát triển và nhân viên kiểm thử.

## Acceptance Testing

* Là kiểm thử nhằm xác định hệ thống phần mềm có đạt yêu cầu kỹ thuật hay không. Bằng việc kiểm tra các hành vi của hệ thống qua dữ liệu thực tế, kiểm thử chấp nhận sẽ xác định có hay không việc hệ thống đáp ứng được các tiêu chí lẫn yêu cầu của khách hàng.
* Được thực hiện bởi người dùng cuối (khách hàng , người dùng và các bên liên quan) để kiểm tra hệ thống được xây dựng để phù hợp với yêu cầu kinh doanh của tổ chức.

# Test case Design Technique

## Test case là gì?

Là một tài liệu trong đó có một tập hợp dữ liệu thử nghiệm, điều kiện tiên quyết, kết quả mong đợi và điều kiện hậu kỳ, được phát triển cho một kịch bản thử nghiệm cụ thể để xác minh sự tuân thủ theo một yêu cầu cụ thể.

## Các thông số kỹ thuật của testcase

* ID trường hợp kiểm thử
* Kịch bản kiểm thử
* Mô tả trường hợp thử nghiệm
* Các bước kiểm tra
* Điều kiện tiên quyết
* Dữ liệu thử nghiệm
* Kết quả mong đợi
* Kết quả thực tế
* Thông tin môi trường
* Bình luận (ghi chú)

# Kỹ thuật thiết kế trường hợp thử nghiệm

## Static Testing

* Là phương pháp để cải thiện chất lượng và năng suất của quá trình phát triển phần mềm thông qua kiểm tra tài liệu và tự động phân tích cú pháp của code hoặc tài liệu dự án mà không cần chạy chương trình.
* Những work product có thể được kiểm tra bằng kiểm thử tĩnh:
  + - * Đặc tả yêu cầu: requirements, functional requirements, security requirements.
      * User story, acceptance criteria
      * Code
      * …...

### Informal Reviews

* Là một trong những loại đánh giá không tuân theo bất kỳ quy trình nào để tìm lỗi trong tài liệu. Chỉ cần xem lại tài liệu và đưa ra những nhận xét không chính thức về nó.
* Là một cuộc thảo luận ngắn, 2 người cũng có thể tham gia đánh giá và không cần ghi chép**.**

**Technical Reviews**

* Một nhóm bao gồm đồng nghiệp của bạn, xem xét đặc điểm kỹ thuật của sản phẩm phần mềm và kiểm tra xe nó có phù hợp với dự án hay không.
* Người dẫn dắt cuộc họp là người điều hành.

**Walkthrough**

* Tác giả của sản phẩm sẽ giải thích sản phẩm cho nhóm của mình. Người tham gia có thể đặt câu hỏi nếu có.
* Người dẫn dắt cuộc họp là tác giả, mọi người tham gia ghi chú các nhận xét đánh giá.

**Inspection**

* Đây là loại đánh giá chính thức, nó được tuân theo quy trình nghiêm ngặt để tìm ra các khiếm khuyết.
* Người đánh giá có danh sách kiểm tra để xem xét sản phẩm công việc, ghi lại những khiếm khuyết và thông báo cho những người tham gia để khắc phục những sai sót.
* Cuộc họp được dẫn dắt bởi người điều hành được đào tạo.

## Dynamic Testing

* Là thử nghiệm động được thực hiện trong môi trường thực thi chạy chương trình ứng dụng.
* Khi code được thực thi, đầu vào được truyền một giá trị, đầu ra của việc thực hiện được so sánh với kết quả dự kiến ban đầu đã đưa ra.

### Specification - based

**Phân tích giá trị ranh giới (BVA)**

* Phân tích giá trị ranh giới dựa trên thử nghiệm tại ranh giới giữa các phân vùng.
* Chúng ta có cả ranh giới hợp lệ (trong phân vùng hợp lệ) và ranh giới không hợp lệ (trong các phân vùng không hợp lệ)
* Một số quy tắc có thể xác định các ca kiểm thử:
  + Giá trị nhỏ nhất
  + Giá trị gần kề lớn hơn giá trị nhỏ nhất
  + Giá trị bình thường
  + Giá trị gần kề bé hơn giá trị lớn nhất
  + Giá trị biên lớn nhất

**Phân tích vùng tương đương (EP)**

* Chia dữ liệu kiểm tra đầu vào của ứng dụng đang được kiểm tra thành từng phân vùng ít nhất một lần dữ liệu tương đương mà từ đó các trường hợp kiểm thử có thể rút ra.
* Thiết kế test - case theo nguyên tắc:
  + Xác định số lớp tương đương hợp lệ
  + Xác định số lớp tương đương không hợp lệ

**Kiểm tra bảng quyết định**

* Đây là cách tốt nhất để giải quyết sự kết hợp của các yếu tố đầu vào, tạo ra các kết quả khác nhau. Nó giúp giảm nỗ lực thử nghiệm trong việc xác minh từng kết hợp dữ liệu thử nghiệm, đồng thời đảm bảo phạm vi toàn diện

**Sơ đồ chuyển đổi trạng thái**

* Được sử dụng khi một khía cạnh của hệ thống có thể được mô tả trong cái được gọi là máy trạng thái hữu hạn. Nó cho phép người kiểm tra xem phần mềm và trạng thái của nó, chuyển đổi giữa các trạng thái, đầu vào hoặc sự kiện kích hoạt thay đổi trạng thái (chuyển tiếp) và các hành động có thể xảy ra từ các chuyển đổi đó.

**Kiểm tra trường hợp sử dụng**

* Được sử dụng để xác định các trường hợp thử nghiệm toàn bộ hệ thống trên một giao dịch theo cơ sở giao dịch từ đầu tới cuối.
* Use case testing là một kỹ thuật xác định các test case mô tả từ đầu tới cuối hành vi của hệ thống từ góc nhìn của người sử dụng. Use case mô tả sự tương tác đặc trưng của người dùng bên ngoài và hệ thống. Mỗi usecase sẽ mô tả cách thức người dùng tương tác với hệ thống để đạt được mục tiêu nào đó.

### Structure - based

**Statement testing**

* Mọi câu lệnh trong cấu trúc code sẽ thực thi ít nhất một lần. Qua đó, tester có thể test được cách vận hành của toàn bộ source code (mã nguồn) của phần mềm. Tuy nhiên tester không thể kiểm thử điều kiện sai mà chỉ có thể thực thi các điều kiện đúng.

**Decision testing**

* Test những quyết định dựa trên decision result (kết quả quyết định). Để làm điều này, test case sẽ đi theo các control flow(kiểm soát dòng chảy) từ decision point (điểm quyết định). Decision testing giúp kiểm thử xem có câu lệnh không thể truy cập hay gây bất thường không.

**Condition testing**

* Được dùng để test các biểu thức Boolean có dạng True(đúng) hoặc False(sai). Mỗi biểu thức Boolean sẽ được thực thi ít nhất một lần bằng cả tham số True và False. Với kỹ thuật này, test case được thiết kế để những điều kiện Boolean có thể thực thi dễ dàng.

**Multiple condition testing**

* Mục đích của kỹ thuật này là kiểm thử mọi tổ hợp điều kiện có thể của quyết định. Công thức tính số tổ hợp này là 2 lũy thừa bậc N với N là số biến điều kiện. số lượng tổ hợp này cũng chính là số lượng test case mà tester phải dùng

**Path testing**

* Tester sẽ test từng câu lệnh có trong source code để tìm lỗi. Việc này giúp xác định lỗi tiềm ẩn trong một đoạn code. Tuy nhiên, tester không nên áp dụng kỹ thuật path testing khi kiểm thử các phần mềm phức tạp. Với cấu trúc code phức tạp, số test case hay câu lệnh mà bạn phải kiểm thử là rất nhiều.

### Experience - based

**Exploratory testing**

* Là sự khai phá, tìm tòi, học hỏi. Test case sẽ không được tạo ra trước mà thực hiện song song việc thiết kế và kiểm thử.
* Kiểm thử thăm dò được thực hiện như một kỹ thuật kiểm tra hộp đen, người kiểm tra học hỏi những điều cùng với kinh nghiệm và sự sáng tạo tạo ra các bài kiểm tra tốt mới để chạy

**Error guessing**

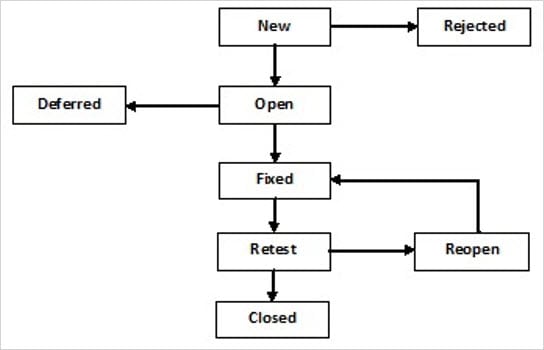
* Kỹ thuật kiểm tra sử dụng kỹ năng, trực giác và kinh nghiệm của người kiểm tra trong việc kiểm tra các ứng dụng tương tự để xác định các lỗi.
* Dự đoán các lỗi tiềm ẩn dựa trên kiến thức của Tester. Những kiến thức này thường sẽ về cách vận hành trước đây của phần mềm, các lỗi đã và có khả năng xuất hiện, những lỗi mà tester từng phát hiện.

# Bug Life Cycle

## Bug là gì?

* Là một lỗ hỏng hoặc một lỗi trong ứng dụng đang hạn chế luồng thông thường của ứng dụng bằng cách không khớp hành vi mong đợi của ứng dụng với ứng dụng thực tế.
* Lỗi xảy ra khi bất kỳ lỗi nào được thực hiện bởi nhà phát triển trong quá trình thiết kế hoặc xây dựng một ứng dụng

## Bug Life Cycle



*Hình : Vòng đời của Bug*

* **New :** là trạng thái đầu tiên của lỗi trong Vòng đời của lỗi. Khi bất kỳ khiếm khuyết mới nào được tìm thấy, nó sẽ ở trạng thái “Mới” và việc xác nhận và kiểm tra được thực hiện đối với khiếm khuyết này trong các giai đoạn sau của vòng đời khiếm khuyết.
* **Assigned :** Một lỗi mới được tạo ra được giao cho nhóm phát triển để khắc phục lỗi đó. Điều này được chỉ định bởi trưởng dự án hoặc người quản lý nhóm thử nghiệm cho một nhà phát triển.
* **Open :** Nhà phát triển bắt đầu quá trình phân tích lỗi và sửa chữa nó nếu được yêu cầu. Nhà phát triển cảm thấy rằng lỗi không phù hợp thì nó có thể được chuyển sang bất kỳ trạng thái nào trong bốn trạng thái (Trùng lặp, Trì hoãn, Bị từ chối hoặc không phải do lỗi)
* **Fixed :** Khi nhà phát triển hoàn thành nhiệm vụ sửa lỗi bằng cách thực hiện các thay đổi được yêu cầu thì anh ta có thể đánh dấu trạng thái của lỗi là “Đã sửa”
* **Pending Retest :** Sau khi sửa lỗi, nhà phát triển chỉ định lỗi cho người kiểm tra để kiểm tra lại lỗi ở cuối của họ và cho đến khi người kiểm tra làm việc để kiểm tra lại lỗi, trạng thái của lỗi vẫn ở trạng thái “kiểm tra lại đang chờ xử lý”
* **Retest :** Người kiểm tra bắt đầu nhiệm vụ làm việc kiểm tra lại lỗi để xác minh xem lỗi có được nhà phát triển sửa chính xác theo yêu cầu hay không
* **Reopen :** Nếu bất kỳ vấn đề nào vẫn tiếp diễn trong lỗi thì nó sẽ được chỉ định lại cho nhà phát triển để kiểm tra và trạng thái của lỗi sẽ được thay đổi thành Mở lại
* **Verified :** Nếu người kiểm tra không tìm thấy bất kỳ vấn đề nào trong lỗi sau khi được chỉ định cho nhà phát triển để kiểm tra lại và anh ta cảm thấy rằng lỗi đã được sửa chính xác thì trạng thái của lỗi sẽ được gán là “Đã xác minh”
* **Closed :** Khi lỗi không còn tồn tại nữa thì người kiểm tra sẽ thay đổi trạng thái của lỗi thành : Đã đóng.
* **Few more:**
  + **Rejected:** Nếu không được nhà phát triển coi là lỗi chính thì lỗi đó được nhà phát triển đánh dấu là “Bị từ chối’
  + **Duplicate:** Nếu nhà phát triển nhận thấy lỗi đó giống với bất kỳ lỗi nào khác hoặc nếu khái niệm về lỗi này khớp với bất kỳ lỗi nào khác thì trạng thái của lỗi sẽ được nhà phát triển thay đổi thành trùng lặp
  + **Deferred:** Nếu nhà phát triển cảm thấy rằng lỗi không phải là ưu tiên rất quan trọng và nó có thể khắc phục trong các version tiếp theo thì họ có thể thay đổi trạng thái của lỗi là :Bị hoãn
  + **Not a Bug:** Nếu lỗi không ảnh hưởng đến chức năng của ứng dụng thì trạng thái của lỗi sẽ được thay đổi thành : không phải lỗi.