**LỜI NÓI ĐẦU**

Với mong muốn có cái nhìn xác thực, rõ ràng hơn về kiểm thử phần mềm và tiếp cận được với công cụ kiểm thử tự động Unified Functional Testing (UFT). Chúng em lựa chọn để tài “Kiểm thử website bằng công cụ Unified Functional Testing (UFT)” làm đề tài cho bài tập lớn môn Kiểm Thử Phần Mềm của mình. Sau một thời gian chúng em đã hoàn thành báo cáo bài tập lớn của mình.

Chúng em xin cảm ơn giảng viên Ths. Nguyễn Hoàng Tú đã giảng dạy và chỉ bảo tận tình trong quá trình học tập, nghiên cứu và hoàn thành báo cáo bài tập lớn này.

Trong khuôn khổ bài tập lớn, do thời gian và kinh nghiệm thực tế còn hạn chế nên có những phần thực hiện chưa được tốt, chúng em rất mong nhận được sự góp ý của thầy và các bạn.

Chúng em xin chân thành cảm ơn !

**MỤC LỤC**

[**LỜI NÓI ĐẦU** 1](#_Toc483320451)

[**MỤC LỤC** 2](#_Toc483320452)

[**Chương 1: TỔNG QUAN VỀ ĐỀ TÀI** 5](#_Toc483320453)

[**1.1** **Lý do chọn đề tài** 5](#_Toc483320454)

[**1.2** **Mục tiêu của đề tài** 6](#_Toc483320456)

[**1.3** **Giới hạn và phạm vi của đề tài** 6](#_Toc483320460)

[**1.4** **Nội dung thực hiện** 6](#_Toc483320463)

[**1.5** **Phương pháp tiếp cận** 6](#_Toc483320468)

[**CHƯƠNG 2: TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM** 7](#_Toc483320473)

[**2.1** **Phần mềm là gì ?** 7](#_Toc483320474)

[**2.1.1** **Khái niệm** 7](#_Toc483320475)

[**2.1.2** **Đặc trưng** 7](#_Toc483320476)

[**2.1.3** **Lỗi phần mềm** 7](#_Toc483320477)

[**2.1.4** **Sai sót của phần mềm** 7](#_Toc483320478)

[**2.1.5** **Các nguyên nhân gây ra lỗi phần mềm** 7](#_Toc483320479)

[**2.1.6** **Các yếu tố ảnh hưởng tới chất lượng phần mềm** 7](#_Toc483320480)

[**2.2** **Tổng quan về kiểm thử** 8](#_Toc483320481)

[**2.2.1** **Khái niệm kiểm thử** 8](#_Toc483320482)

[**2.2.2** **Mục tiêu kiểm thử** 8](#_Toc483320483)

[**2.2.3** **Tầm quan trọng của kiểm thử** 8](#_Toc483320484)

[**2.2.4** **Các nguyên tắc trong kiểm thử** 8](#_Toc483320485)

[**2.2.5** **Phân loại kiểm thử** 9](#_Toc483320486)

[**2.2.5.1** **Dựa vào mục đích kiểm thử** 9](#_Toc483320487)

[**2.2.5.2** **Dựa vào chiến lược kiểm thử** 12](#_Toc483320488)

[**2.2.5.3** **Dựa vào phương pháp kiểm thử** 12](#_Toc483320489)

[**2.2.5.4** **Dựa vào kỹ thuật kiểm thử** 12](#_Toc483320490)

[**2.2.6** **Quy trình kiểm thử phần mềm** 16](#_Toc483320492)

[**Chương 3: Kiểm thử website bằng công cụ kiểm thử tự động** 18](#_Toc483320493)

[**3.1** **Kiểm thử website** 18](#_Toc483320494)

[**3.1.1** **Khái quát** 18](#_Toc483320495)

[**3.1.2** **Đặc điểm về chất lượng của ứng dụng Website** 18](#_Toc483320496)

[**3.2** **Công việc khi kiểm thử một Website** 19](#_Toc483320497)

[**3.2.1** **Kiểm tra chức năng(hồi quy, tích hợp, kiểm tra khói)** 19](#_Toc483320498)

[**3.2.2** **Kiểm tra sự tương thích với trình duyệt** 20](#_Toc483320503)

[**3.3** **Thử nghiệm tính năng** 21](#_Toc483320507)

[**3.4** **Kiểm tra bảo mật** 21](#_Toc483320510)

[**3.5** **Giám sát sản xuất** 22](#_Toc483320511)

[**3.6** **Kiểm tra khả năng sử dụng** 22](#_Toc483320512)

[**3.7** **Khái quát về kiểm thử tự động** 23](#_Toc483320515)

[**3.8** **Công cụ test bảo mật Web** 23](#_Toc483320516)

[**3.9** **Công cụ test hiệu năng** 23](#_Toc483320517)

[**3.10** **Công cụ test chức năng** 23](#_Toc483320518)

[**Chương 4 : Khảo sát các công cụ kiểm thử tự động** 25](#_Toc483320519)

[**4.1. Công cụ Quick Test Pro(QTP).** 25](#_Toc483320520)

[**4.1.1.     Loại phần mềm hỗ trợ** 25](#_Toc483320521)

[**4.1.2.     Đặc điểm.** 25](#_Toc483320522)

[**4.1.3.     Các thành phần quan trọng QTP.** 26](#_Toc483320523)

[**4.1.4.     Ngôn ngữ sử dụng viết script** 26](#_Toc483320524)

[**4.2. Công Cụ Unified Functional Testing (UFT).** 26](#_Toc483320526)

[**4.3. Công Cụ Load Runner .** 28](#_Toc483320529)

[**4.3.1. Những thành phần của LoadRunner.** 28](#_Toc483320530)

[**4.3.2.Thuật ngữ dừng trong LoadRunner.** 28](#_Toc483320531)

[**4.3.3. Phương phát Load Test. 29**](#_Toc483320532)

[**4.4. Công Cụ JMeter.** 29](#_Toc483320533)

[**4.5 . Công Cụ JNnit.** 30](#_Toc483320534)

[**4.5.1. Giới thiệu: 30**](#_Toc483320535)

[**4.5.2. Những điều kiện tiên quyết: 30**](#_Toc483320536)

[**4.5.3. Tổng quan: 30**](#_Toc483320537)

[**4.5.3.Tính năng:** 31](#_Toc483320538)

[**4.5.4**. **Unit Test Case là gì?** 32](#_Toc483320539)

[**Chương 5 : Unified Functional Testing (UFT)**  33](#_Toc483320540)

[**5.1 Giới thiệu về Unified Functional Testing (UFT)** 33](#_Toc483320541)

[**5.2. Tìm hiểu Unified Functional Testing (UFT)**  34](#_Toc483320542)

[**Chương 6 : Ứng dụng Unified Functional Testing (UFT) vào kiểm thử website** 39](#_Toc483320544)

[**6.1 Lập kế hoạch kiểm thử.** 39](#_Toc483320545)

[**6.1.1 Các usecase của ứng dụng.** 39](#_Toc483320546)

[**6.1.2 mô tả ca sử dụng đăng nhập.** 41](#_Toc483320547)

[**6.2 thiết kế testcase** 43](#_Toc483320548)

[**6.2.1: kĩ thuật kiểm thử hộp trắng**. 43](#_Toc483320549)

[**6.2.2: kĩ năng kiểm thử hộp đen**. 46](#_Toc483320550)

[**6.3 Thực thi kiểm thử:** 48](#_Toc483320551)

[**6.4 Tổng hợp kết quả test:** 49](#_Toc483320552)

[**Kết luận** 50](#_Toc483320553)

[**Tài liệu tham khảo** 51](#_Toc483320554)

**Chương 1: TỔNG QUAN VỀ ĐỀ TÀI**

1. **Lý do chọn đề tài**

Trong giai đoạn phát triển của công nghệ thông tin, ngành công nghệ phần mềm đang ngày một chiếm vị trí quan trọng trong xu hướng phát triển kinh tế công nghiệp hóa, hiện đại hóa của đất nước ta. Cùng với sự phát triển của công nghệ phần mềm, lỗi phần mềm và chất lượng phần mềm luôn là thách thức lớn để phần mềm khi thực tế đã chứng minh, kiểm thử phần mềm là giai đoạn chiếm đến hơn 30%-70% thời gian, kinh phí và nguồn nhân lực phát triển dự án phần mềm (tùy theo loại phần mềm và lĩnh vực). Tuy nhiên ở Việt Nam hiện nay, việc kiểm thử phần mềm vẫn chưa thực sự được nhìn nhận đúng với tầm quan trọng của nó. Điều này thể hiện ở tỷ lệ kỹ sư kiểm thử phần mềm ở Việt Nam còn khá thấp, cứ 5 lập trình viên thì mới có 1 kỹ sư kiểm thử , trong khi tỷ lệ này theo chuẩn quốc tế là 3:1. Thêm vào đó, mức độ đáp ứng của kỹ sư kiểm thử phần mềm ở Việt Nam chưa cao. Nguyên nhân của việc này đến từ sự thiếu hụt các đơn vị đào tạo chuyên sâu về kiểm thử và nguyên nhân sâu xa vẫn là vấn đề kiểm thử phần mềm ở Việt Nam vẫn chưa được chuyên nghiệp hóa và đầu tư đúng mức. Ngày nay, tự động hóa đang được nghiên cứu và ứng dụng trong nhiều lĩnh vực trong đó công nghệ phần mềm nói chung và kiểm thử phần mềm nói riêng cũng không ngoại lệ. Khi mà kiểm thử phần mềm vẫn tiêu tốn một lượng lớn thời gian, kinh phí và nhân lực trong một dự án phần mềm thì song song với kiểm thử truyền thống thủ công, sự ra đời của các công cụ hỗ trợ kiểm thử tự động như Quick Test Professional, Nunit, Junit, Load Runner ... là tất yếu. Unified Functional Testing (UFT) là một công cụ kiểm thử các ứng dụng web có khá nhiều ưu điểm như có thể kiểm thử trên nhiều trình duyệt, hỗ trợ nhiều ngôn ngữ lập trình, giao tiếp được với các công cụ kiểm thử khác như Junit, TestNG (với Java) hay Nunit (với C#), và ưu điểm đặc biệt của công cụ này là nó là một bộ mã nguồn mở, do đó các tổ chức sẽ không tốn kinh phí mua bản quyền. Tuy chưa được ứng dụng nhiều trong các tổ chức ở Việt Nam, song với Tìm hiểu về kiểm thử tự động và ứng dụng kiểm thử website sử dụng công cụ kiểm thử tự động Unified Functional Testing (UFT). Những ưu điểm trên, Unified Functional Testing (UFT) hứa hẹn sẽ ngày càng phát triển và trở lên thông dụng hơn trong các tổ chức phát triển phần mềm ở nước ta.

1. **Mục tiêu của đề tài**

* Nắm được lý thuyết kiểm thử phần mềm, kiểm thử tự động phần mềm.
* Nắm được lý thuyết về công cụ kiểm thử tự động Unified Functional Testing (UFT).
* Ứng dụng được công cụ Unified Functional Testing (UFT) vào kiểm thử website.

1. **Giới hạn và phạm vi của đề tài**

* Tập trung vào lý thuyết kiểm thử, kiểm thử tự động, công cụ Unified Functional Testing (UFT).
* Ứng dụng được công cụ Unified Functional Testing (UFT) vào kiểm thử website.

1. **Nội dung thực hiện**

* Trình bày được lý thuyết kiểm thử tự động phần mềm, kiểm thử website.
* Trình bày được lý thuyết về công cụ kiểm thử tự động Unified Functional Testing (UFT).
* Ứng dụng được bộ công cụ kiểm thử tự động Unified Functional Testing (UFT) vào kiểm thử website.
* Tìm hiểu về kiểm thử tự động và ứng dụng kiểm thử tự động website sử dụng công cụ kiểm thử Unified Functional Testing (UFT).

1. **Phương pháp tiếp cận**

Sử dụng các phương pháp nghiên cứu:

* Phương pháp đọc tài liệu.
* Phương pháp phân tích mẫu.
* Phương pháp thực nghiệm (làm giảm bớt thời gian kiểm thử sau này.)

# **CHƯƠNG 2:TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM**

* 1. **Phần mềm là gì ?**

1. **Khái niệm**

Phần mềm máy tính hay còn được gọi tắt là phần mềm (software) là một tập hợp các câu lệnh hoặc chỉ thị được viết bằng 1 hoặc nhiều ngôn ngữ lập trình theo một trận tự xác định nhằm tạo ra một nhiệm vụ hay chức năng năng hoặc một vấn đề cụ thể nào đó.

Phần mềm máy tính hoạt động bằng cách gửi trực tiếp các chỉ thị đến phần cứng hoặc bằng cách cung cấp dữ liệu để phục vụ các chương trình phần mềm khác thực hiện nhiệm vụ của mình.

Phần mềm có thể hiểu một cách trừu tượng là những thứ không thể cầm, nắm như phần cứng và phần mềm hoạt động phụ thuộc vào phần cứng.

* + 1. **Đặc trưng**

- Được thiết kế ,chế tạo như sản phẩm công nghiệp khác nhưng không được định hình trước

- Các phần mềm chỉ thực sự được tìm ra lỗi trong pha phát triển

- Có tính phức tạp cao và luôn thay đổi.

* + 1. **Lỗi phần mềm**

- Do con người gây ra

- Có thể do lỗi cú pháp hoặc logic

* + 1. **Sai sót của phần mềm**

- Không phải lúc nào cũng do lỗi phần mềm

- Thiếu,thừa,bỏ sót yêu cầu

* + 1. **Các nguyên nhân gây ra lỗi phần mềm**

1. Hiểu sai yêu cầu
2. Thất bại trong giao tiếp giữa người phát triển và khách hàng
3. Tạo độ lệch cố ý trong yêu cầu khách hàng
4. Lỗi logic trong thiết kế
5. Lỗi mã hóa
6. Không tuân theo tài liệu và cấu trúc code
7. Rút ngắn quá trình kiểm thứ
8. Lỗi thủ tục
9. Lỗi tài liệu
   * 1. **Các yếu tố ảnh hưởng tới chất lượng phần mềm**

- Con người

- Quy trình

* 1. **Tổng quan về kiểm thử**
     1. **Khái niệm kiểm thử**

Kiểm thử phần mềm là việc kiểm tra kết quả thực hiện của chương trình máy tính xem có đúng với mục tiêu đã đặt ra với nó không thông qua việc thực hiện ở một số mẫu thử.

Kiểm thử phần mềm là việc tìm ra lỗi trong bản thân phần mềm, việc kiểm thử này trong phần mềm sẽ biểu thị ra những thiếu sót mà ta có thể nhận thấy trong hành vi của phần mềm, và tìm ra những phần mềm không tuân theo quy định, đi lệch ra khỏi những yêu cầu của phần mềm.

Theo Glenford Myes : Kiểm thử là quá trình vận hành chương trình để tìm ra lỗi

Theo IEEE:

* Là quá trình vận hành hệ thống hoặc thành phần dưới những điều kiện xác định,quan sát hoặc ghi nhận kết quả và đưa ra đánh giá về hệ thống hoặc thành phần đó
* Là quá trình phân tích phần mềm để tìm ra sự khác biệt giữa điều kiện thưc tế và điều kiện yêu cầu và dựa vào điều khác biệt đó để đánh giá tính năng phần mềm
  + 1. **Mục tiêu kiểm thử**
* Tìm ra lỗi
* Chứng minh phần mềm hợp với các đặc tả
* Xác thực chất lượng kiểm thử đã dùng chi phí và nỗ lực tối thiểu
* Hậu quả,tiết kiệm thời gian,công sức
  + 1. **Tầm quan trọng của kiểm thử**
* Chứng minh chương trình không có lỗi
* Chỉ ra rằng các chương trình thực hiện đúng các chức năng
  + 1. **Các nguyên tắc trong kiểm thử**

Có 7 nguyên tắc cơ bản

1. Kiểm thử chỉ ra sự hiện diện của lỗi trong phần mềm
2. Kiểm thử tốt các trường hợp là điều không thể
3. Nên thực hiện kiểm thử càng sớm càng tốt
4. Sự phân cụm của các lỗi
5. Nghịch lý thuốc trừ sâu
6. Kiểm thử theo các ngữ cảnh độc lập
7. Sư sai lầm về việc không có lỗi
   * 1. **Phân loại kiểm thử**
        1. **Dựa vào mục đích kiểm thử**

**Kiểm thử đơn vị**

Unit Test thường do lập trình viên thực hiện. Công đoạn này cần được thực hiện càng sớm càng tốt trong giai đoạn viết code và xuyên suốt chu kỳ PTPM. Thông thường, Unit Test đòi hỏi kiểm tra viên có kiến thức về thiết kế và code của chương trình. Mục đích của Unit Test là bảo đảm thông tin được xử lý và xuất (khỏi Unit) là chính xác, trong mối tương quan với dữ liệu nhập và chức năng của Unit. Điều này thường đòi hỏi tất cả các nhánh bên trong Unit đều phải được kiểm tra để phát hiện nhánh phát sinh lỗi. Một nhánh thường là một chuỗi các lệnh được thực thi trong một Unit, ví dụ: chuỗi các lệnh sau điều kiện If và nằm giữa then … else là một nhánh. Thực tế việc chọn lựa các nhánh để đơn giản hóa việc kiểm tra và quét hết Unit đòi hỏi phải có kỹ thuật, đôi khi phải dùng thuật toán để chọn lựa.

Cũng như các mức kiểm tra khác, Unit Test cũng đòi hỏi phải chuẩn bị trước các tình huống (test case) hoặc kịch bản (script), trong đó chỉ định rõ dữ liệu vào, các bước thực hiện và dữ liệu mong chờ sẽ xuất ra. Các test case và script này nên được giữ lại để tái sử dụng.

**Kiểm thử cấu hình**

Kiểm tra nhằm bảo đảm các thành phần bên trong của một chương trình chạy đúng, chú trọng đến hoạt động của các thành phần cấu trúc nội tại của chương trình chẳng hạn các lệnh và nhánh bên trong.

**Kiểm thử sơ lược**

Kiểm tra sơ lược về thiết kế,quy trình và ,các kịch bản và dữ liệu.

**Kiểm thử chức năng (functional)**

Kiểm tra chỉ chú trọng đến chức năng của chương trình, không quan tâm đến cấu trúc bên trong, chỉ khảo sát chức năng của chương trình theo yêu cầu kỹ thuật.

**Kiểm thử tích hợp (Integration test)**

Integration test kết hợp các thành phần của một ứng dụng và kiểm tra như một ứng dụng đã hoàn thành. Trong khi Unit Test kiểm tra các thành phần và Unit riêng lẻ thì Intgration Test kết hợp chúng lại với nhau và kiểm tra sự giao tiếp giữa chúng.  
  
Integration Test có 2 mục tiêu chính:  
  
- Phát hiện lỗi giao tiếp xảy ra giữa các Unit.  
  
- Tích hợp các Unit đơn lẻ thành các hệ thống nhỏ (subsystem) và cuối cùng là nguyên hệ thống hoàn chỉnh (system) chuẩn bị cho kiểm tra ở mức hệ thống (System Test).

Trong Unit Test, lập trình viên cố gắng phát hiện lỗi liên quan đến chức năng và cấu trúc nội tại của Unit. Có một số phép kiểm tra đơn giản trên giao tiếp giữa Unit với các thành phần liên quan khác, tuy nhiên mọi giao tiếp liên quan đến Unit thật sự được kiểm tra đầy đủ khi các Unit tích hợp với nhau trong khi thực hiện Integration Test.

Trừ một số ít ngoại lệ, Integration Test chỉ nên thực hiện trên những Unit đã được kiểm tra cẩn thận trước đó bằng Unit Test, và tất cả các lỗi mức Unit đã được sửa chữa. Một số người hiểu sai rằng Unit một khi đã qua giai đoạn Unit Test với các giao tiếp giả lập thì không cần phải thực hiện Integration Test nữa. Thực tế việc tích hợp giữa các Unit dẫn đến những tình huống hoàn toàn khác.

Một chiến lược cần quan tâm trong Integration Test là nên tích hợp dần từng Unit. Một Unit tại một thời điểm được tích hợp vào một nhóm các Unit khác đã tích hợp trước đó và đã hoàn tất (passed) các đợt Integration Test trước đó. Lúc này, ta chỉ cần kiểm tra giao tiếp của Unit mới thêm vào với hệ thống các Unit đã tích hợp trước đó, điều này làm cho số lượng kiểm tra sẽ giảm đi rất nhiều, sai sót sẽ giảm đáng kể.

**Kiểm thử hồi quy (Regression Test)**

Regression Test không phải là một mức kiểm tra, như các mức khác đã nói ở trên. Nó đơn thuần kiểm tra lại PM sau khi có một sự thay đổi xảy ra, để bảo đảm phiên bản PM mới thực hiện tốt các chức năng như phiên bản cũ và sự thay đổi không gây ra lỗi mới trên những chức năng vốn đã làm việc tốt. Regression test có thể thực hiện tại mọi mức kiểm tra.

Mặc dù không là một mức kiểm tra, thực tế lại cho thấy Regression Test là một trong những loại kiểm tra tốn nhiều thời gian và công sức nhất. Tuy thế, việc bỏ qua Regression Test là “không được phép” vì có thể dẫn đến tình trạng phát sinh hoặc tái xuất hiện những lỗi nghiêm trọng, mặc dù ta “tưởng rằng” những lỗi đó hoặc không có hoặc đã được kiểm tra và sửa chữa rồi!

**Kiểm thử hệ thống ( System Test)**

Mục đích System Test là kiểm tra thiết kế và toàn bộ hệ thống (sau khi tích hợp) có thỏa mãn yêu cầu đặt ra hay không.

System Test bắt đầu khi tất cả các bộ phận của PM đã được tích hợp thành công. Thông thường loại kiểm tra này tốn rất nhiều công sức và thời gian. Trong nhiều trường hợp, việc kiểm tra đòi hỏi một số thiết bị phụ trợ, phần mềm hoặc phần cứng đặc thù, đặc biệt là các ứng dụng thời gian thực, hệ thống phân bố, hoặc hệ thống nhúng. Ở mức độ hệ thống, người kiểm tra cũng tìm kiếm các lỗi, nhưng trọng tâm là đánh giá về hoạt động, thao tác, sự tin cậy và các yêu cầu khác liên quan đến chất lượng của toàn hệ thống.

Điểm khác nhau then chốt giữa Integration Test và System Test là System Test chú trọng các hành vi và lỗi trên toàn hệ thống, còn Integration Test chú trọng sự giao tiếp giữa các đơn thể hoặc đối tượng khi chúng làm việc cùng nhau. Thông thường ta phải thực hiện Unit Test và Integration Test để bảo đảm mọi Unit và sự tương tác giữa chúng hoạt động chính xác trước khi thực hiện System Test.

Sau khi hoàn thành Integration Test, một hệ thống PM đã được hình thành cùng với các thành phần đã được kiểm tra đầy đủ. Tại thời điểm này, lập trình viên hoặc kiểm tra viên (tester) bắt đầu kiểm tra PM như một hệ thống hoàn chỉnh. Việc lập kế hoạch cho System Test nên bắt đầu từ giai đoạn hình thành và phân tích các yêu cầu. Phần sau ta sẽ nói rõ hơn về một quy trình System Test cơ bản và điển hình.

System Test kiểm tra cả các hành vi chức năng của phần mềm lẫn các yêu cầu về chất lượng như độ tin cậy, tính tiện lợi khi sử dụng, hiệu năng và bảo mật. Mức kiểm tra này đặc biệt thích hợp cho việc phát hiện lỗi giao tiếp với PM hoặc phần cứng bên ngoài, chẳng hạn các lỗi “tắc nghẽn” (deadlock) hoặc chiếm dụng bộ nhớ. Sau giai đoạn System Test, PM thường đã sẵn sàng cho khách hàng hoặc người dùng cuối cùng kiểm tra để chấp nhận (Acceptance Test) hoặc dùng thử (Alpha/Beta Test).

**Kiểm thử tải trọng (Stress Test hay Load Test)**

Bảo đảm hệ thống vận hành đúng dưới áp lực cao (ví dụ nhiều người truy xuất cùng lúc). Stress Test tập trung vào các trạng thái tới hạn, các “điểm chết”, các tình huống bất thường…

**Kiểm thử hiệu suất (performance)**

Kiểm tra việc vận hành của hệ thống.

**Kiểm thử chấp nhận ( Acceptance Test)**

Thông thường, sau giai đoạn System Test là Acceptance Test, được khách hàng thực hiện (hoặc ủy quyền cho một nhóm thứ ba thực hiện). Mục đích của Acceptance Test là để chứng minh PM thỏa mãn tất cả yêu cầu của khách hàng và khách hàng chấp nhận sản phẩm (và trả tiền thanh toán hợp đồng).

Acceptance Test có ý nghĩa hết sức quan trọng, mặc dù trong hầu hết mọi trường hợp, các phép kiểm tra của System Test và Accepatnce Test gần như tương tự, nhưng bản chất và cách thức thực hiện lại rất khác biệt.

Đối với những sản phẩm dành rộng rãi trên thị trường cho nhiều người sử dụng, thông thường sẽ thông qua hai loại kiểm tra gọi là Alpha Test và Beta Test. Với Alpha Test, người sử dụng (tiềm năng) kiểm tra PM ngay tại nơi PTPM, lập trình viên sẽ ghi nhận các lỗi hoặc phản hồi, và lên kế hoạch sửa chữa. Với Beta Test, PM sẽ được gửi tới cho người sử dụng (tiềm năng) để kiểm tra ngay trong môi trường thực, lỗi hoặc phản hồi cũng sẽ gửi ngược lại cho lập trình viên để sửa chữa.

Thực tế cho thấy, nếu khách hàng không quan tâm và không tham gia vào quá  
trình PTPM thì kết quả Acceptance Test sẽ sai lệch rất lớn, mặc dù PM đã trải qua  
tất cả các kiểm tra trước đó. Sự sai lệch này liên quan đến việc hiểu sai yêu cầu  
cũng như sự mong chờ của khách hàng. Ví dụ đôi khi một PM xuất sắc vượt qua  
các phép kiểm tra về chức năng thực hiện bởi nhóm thực hiện dự án, nhưng khách  
hàng khi kiểm tra sau cùng vẫn thất vọng vì bố cục màn hình nghèo nàn, thao tác không tự nhiên, không theo tập quán sử dụng của khách hàng v.v…

Gắn liền với giai đoạn Acceptance Test thường là một nhóm những dịch vụ và tài liệu đi kèm, phổ biến như hướng dẫn cài đặt, sử dụng v.v… Tất cả tài liệu đi kèm phải được cập nhật và kiểm tra chặt chẽ.

* + - 1. **Dựa vào chiến lược kiểm thử**
* **Kiểm thử thủ công**

Bằng tay,viết test case đến thực hiện test

* **Tự động**

Thực hiện tự động các bước trong kịch bản kiểm thử

Tiết kiệm thời gian

* + - 1. **Dựa vào phương pháp kiểm thử**

**Kiểm thử tĩnh (Static Testing)**

Là phương pháp kiểm thử phần mềm đòi hỏi phải duyệt lại các yêu cầu và các đặc tả bằng tay, thông qua việc sử dụng giấy, bút để kiểm tra logic, lần từng chi tiết mà không cần chạy chương trình. Kiểu kiểm thử này thường được sử dụng bởi chuyên viên thiết kế người mà viết mã lệnh một mình.

Kiểm thử tĩnh cũng có thể được tự động hóa. Nó sẽ thực hiện kiểm tra toàn bộ bao gồm các chương trình được phân tích bởi một trình thông dịch hoặc biên dịch mà xác nhận tính hợp lệ về cú pháp của chương trình.

**Kiểm thử động (Dynamic Testing)**

Là phương pháp kiểm thử phần mềm thông qua việc dùng máy chạy chương trình để điều tra trạng thái tác động của chương trình. Đó là ki ểm thử dựa trên các ca kiểm thử xác định bằng sự thực hiện của đối tượng kiểm thử hay chạy các chương trình. Kiểm thử động kiểm tra cách thức hoạt động của mã lệnh, tức là kiểm tra sự phản ứng vật lý từ hệ thống tới các biến luôn thay đổi theo thời gian.

Trong kiểm thử động, phần mềm phải thực sự được biên dịch và chạy. Kiểm thử động thực sự bao gồm làm việc với phần mềm, nhập các giá trị đầu vào và kiểm tra xem liệu đầu ra có như mong muốn hay không. Các phương pháp kiểm thử động gồm có kiểm thử Unit – Unit Tests, Kiểm thử tích hợp – Intergration Tests, Kiểm thử hệ thống –System Tests, và Kiểm thử chấp nhận sản phẩm – Acceptance Tests.

* + - 1. **Dựa vào kỹ thuật kiểm thử**

**Kiểm thử hộp trắng**

Kiểm thử hộp trắng hay kiểm thử hướng logic cho phép khảo sát cấu trúc bên trong của chương trình. Kiểm thử viên sẽ truy cập vào cấu trúc dữ liệu và giải thuật bên trong chương trình (và cả mã lệnh thực hiện chúng).

* Các phương pháp kiểm thử hộp trắng:

- Kiểm  thử giao  diện  lập  trình ứng  dụng - API  testing  (application programming interface): là phương pháp kiểm thử của ứng dụng sử dụng các API công khai và riêng tư.

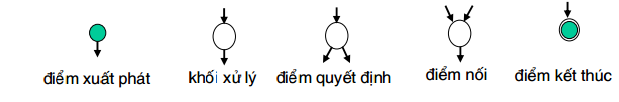
- Các phương pháp gán lỗi – Fault injection.

- Các phương pháp kiểm thử hoán chuyển – Mutation testing methods.

- Kiểm thử tĩnh – Static testing: kiểm thử hộp trắng bao gồm mọi kiểm thử tĩnh.Phương pháp kiểm thử hộp trắng cũng có thể được sử dụng để đánh giá sự hoàn thành của một bộ kiểm thử mà được tạo cùng với các phương pháp kiểm thử hộp đen.

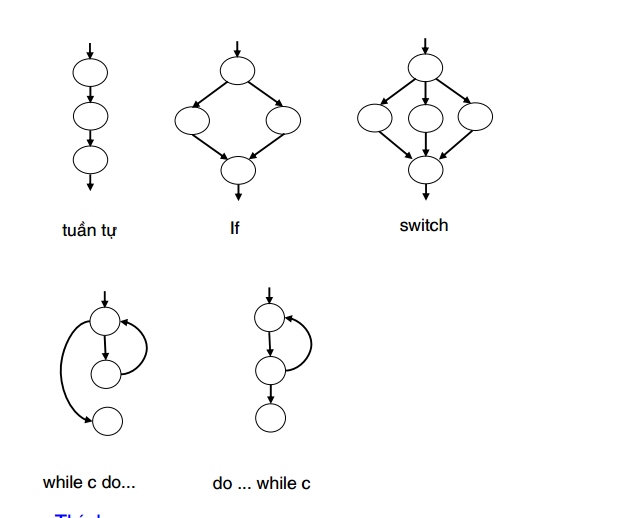
- Bao phủ mã lệnh – Code coverage: tạo các kiểm tra để đáp ứng một số tiêu chuẩn về bao phủ mã lệnh.

- Phủ kiểm thử (Coverage): Là tỉ lệ các thành phần thực sự được kiểm thử so với tổng thể sau khi đã kiểm thử các test case được chọn. Phủ càng lớn thì độ tin cậy càng cao. Thành phần liên quan có thể là lệnh thực thi, điểm quyết định, điều kiện con hay là sự kết hợp của chúng.

* **Phủ cấp 0**: Kiểm thử những gì có thể kiểm thử được, phần còn lại để người dùng phát hiện và báo lại sau. Đây là mức độ kiểm thử không thực sự có trách nhiệm.
* **Phủ cấp 1**: Kiểm thử sao cho mỗi lệnh được thực thi ít nhất 1 lần.
* **Phủ cấp 2**: Kiểm thử sao cho mỗi điểm quyết định luận lý đều được thực hiện ít nhất 1 lần cho trường hợp TRUE lẫn FALSE. Ta gọi mức kiểm thử này là phủ các nhánh (Branch coverage). Phủ các nhánh đảm bảo phủ các lệnh.
* **Phủ cấp 3:** Kiểm thử sao cho mỗi điều kiện luận lý con (subcondition) của từng điểm quyết định đều được thực hiện ít nhất 1 lần cho trường hợp TRUE lẫn FALSE. Ta gọi mức kiểm thử này là phủ các điều kiện con (subcondition coverage). Phủ các điều kiện con chưa chắc đảm bảo phủ các nhánh và ngược lại.
* **Phủ cấp 4**: Kiểm thử sao cho mỗi điều kiện luận lý con (subcondition) của từng điểm quyết định đều được thực hiện ít nhất 1 lần cho trường hợp TRUE lẫn FALSE và điểm quyết định cũng được kiểm thử cho cả 2 nhánh TRUE lẫn FALSE. Ta gọi mức kiểm thử này là phủ các nhánh và các điều kiện con (branch and subcondition coverage). Đây là mức độ phủ kiểm thử tốt nhất trong thực tế. Phần còn lại của chương này sẽ giới thiệu qui trình kỹ thuật để định nghĩa các test case sao cho nếu kiểm thử hết các testcase được định nghĩa này, ta sẽ đạt phủ kiểm thử cấp 4.
* **Đồ thị dòng điều khiển**: Là một trong nhiều phương pháp miêu tả thuật giải. Đây là phương pháp trực quan cho chúng ta thấy dễ dàng các thành phần của thuật giải và mối quan hệ trong việc thực hiện các thành phần này. Gồm 2 loại thành phần : Các nút và các cung nối kết giữa chúng. Các loại nút trong đồ thị dòng điều khiển :

**Hình 2.1 Các loại nút trong đồ thị dòng điều khiển**

* Miêu tả các cấu trúc điều khiển phổ dụng :



**Hình 2.2 Các cấu trúc điều khiển phổ dụng**

* **Độ phức tạp Cyclomatic C:**

Độ phức tạp Cyclomatic C = V(G) của đồ thị dòng điều khiển được tính bởi 1 trong các công thức sau :

V(G) = E - N + 2, trong Đó E là số cung, N là số nút của đồ thị.

V(G) = P + 1, nếu là đồ thị dòng điều khiển nhị phân (chỉ chứa các nút quyết định luận lý - chỉ có 2 cung xuất True/False) và P số nút quyết định. Độ phức tạp Cyclomatic C chính là số đường thi hành tuyến tính độc lập của TPPM cần kiểm thử. Nếu chúng ta chọn lựa được đúng C đường thi hành tuyến tính độc lập của TPPM cần kiểm thử và kiểm thử tất cả các đường thi hành này thì sẽ đạt được phủ kiểm thử cấp 3 như đã trình bày.

* **Quy trình kiểm thử hộp trắng:**

Tom McCabe đề nghị qui trình kiểm thử TPPM gồm các bước công việc sau:

1. Từ TPPM cần kiểm thử, xây dựng đồ thị dòng điều khiển tương ứng, rồi chuyển thành đồ thị dòng điều khiển nhị phân, rồi chuyển thành đồ thị dòng điều khiển cơ bản.

2. Tính độ phức tạp Cyclomatic của đồ thị (C = P +1).

3. Xác định C đường thi hành tuyến tính độc lập cơ bản cần kiểm thử.

4. Tạo từng test case cho từng đường thi hành tuyến tính độc lập cơ bản.

5. Thực hiện kiểm thử trên từng test case.

6. So sánh kết quả có được với kết quả được kỳ vọng.

7. Lập báo cáo kết quả để phản hồi cho những người có liên quan.

- Các phương pháp gán lỗi – *Fault injection.*

- Các phương pháp kiểm thử hoán chuyển – *Mutation testing methods.*

Kiểm thử tĩnh - *Static testing:* kiểm thử hộp trắng bao gồm mọi kiểm thử tĩnh.

**Kiểm thử hộp đen**

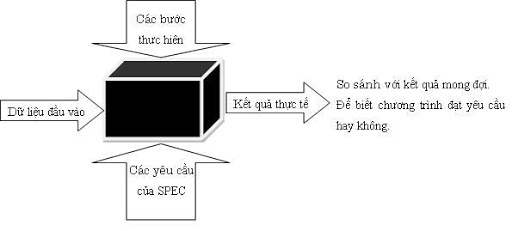
Một trong những chiến lược kiểm thử quan trọng là kiểm thử hộp đen, hướng dữ liệu, hay hướng vào/ra. Kiểm thử hộp đen xem chương trình như là một “hộp đen”. Mục đích của kiểm thử viên là hoàn toàn không quan tâm về cách cư xử và cấu trúc bên trong của chương trình. Thay vào đó, tập trung vào tìm các trường hợp mà chương trình không thực hiện theo các đặc tả của nó.

Theo hướng tiếp cận này, dữ liệu kiểm tra được lấy chỉ từ các đặc tả.

### Các phương pháp kiểm thử hộp đen:

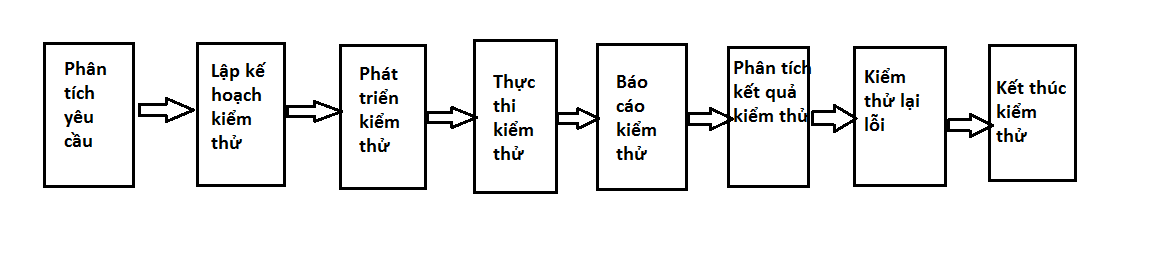
* Phân lớp tương đương *– Equivalencepartitioning.*
* Phân tích giá trị biên *– Boundary valueanalysis.*
* Kiểm thử mọi cặp *– All-pairstesting.*
* Kiểm thử fuzz *– Fuzztesting.*
* Kiểm thử dựa trên mô hình *– Model-basedtesting.*
* Ma trận dấu vết *– Traceabilitymatrix.*
* Kiểm thử thăm dò *– Exploratorytesting.*
* Kiểm thử dựa trên đặc tả *– Specification-basetesting.*

Kiểm thử dựa trên đặc tả tập trung vào kiểm tra tính thiết thực của phần mềm theo những yêu cầu thích hợp. Do đó, kiểm thử viên nhập dữ liệu vào, vàchỉ thấy dữ liệu ra từ đối tượng kiểm thử. Mức kiểm thử này thường yêu cầu các kiểm thử viên xác minh là đối với dữ liệu đầu vào đã cho, giá trị đầu ra (haycách thức hoạt động) có giống với giá trị mong muốn trong ca kiểm thử đó hay không. Kiểm thử dựa trên đặc tả là cần thiết, nhưng không đủ để để ngăn chặn những rủi ro chắcchắn.



**Hình 2.3 Quy trình kiểm thử hộp đen**

* **Ưu điểm và nhược điểm**: Các tester kiểm thử theo phương pháp black box không có “mối ràng buộc” nào với code, và nhận thức của một tester rất đơn giản: một source code có nhiều lỗi. Sử dụng nguyên tắc, "Hỏi và bạn sẽ nhận" các tester black box tìm được nhiều bug ở nơi mà các DEV không tìm thấy. Mặt khác, việc kiểm thử black box được xem như "là bước đi trong mê cung tối đen mà không mang đèn pin” bởi vì tester không biết phần mềm đang test đã được xây dựng như thế nào.
* *Vì vậy, black box testing có ưu điểm là "an unaffiliated opinion" (một quan điểm độc lập), mặt khác nó có nhược điểm là ‘‘blind exploring ’’(khám phá mù)*
  + 1. **Quy trình kiểm thử phần mềm**

**Hình 2.4 Quy trình kiểm thử phần mềm**

Mô tả các bước trong quy trình kiểm thử phần mềm:

1. Phân tích yêu cầu: Xác định phạm vi test.

2. Lập kết hoạch kiểm thử: Chiến lược kiểm thử (test strategy), test plan.

3. Phát triển kiểm thử: Viết test procedure, test scenario, test case, test data và test script.

4. Thực thi kiểm thử: Tester thực thi phần mềm dựa trên test plan và test case.

5. Báo cáo kiểm thử: Tester điền kết quả test vào test case và tạo báo cáo kết quả test.

6. Phân tích kết quả kiểm thử: Hoặc còn gọi là phân tích lỗi để quyết định lỗi nào sẽ được sửa và lỗi nào sẽ không sửa.

7.Kiểm thử lại lỗi: Sau khi một lỗi (defect) được DEV sửa xong, chuyển phần mềm cho tester test lại.

8.Kết thúc kiểm thử: Khi test đã đáp ứng được điều kiện dừng. Từ đó rút ra các bài học kinh nghiệm.

# **Chương 3 : Unified Functional Testing (UFT)**

## **I Giới thiệu về Unified Functional Testing (UFT)**

1. **Giới thiệu**

* **a Tổng quan về Micro Focus Unified Functional Testing (UFT)**
  + Khái niệm UFT
    - Micro Focus Unified Functional Testing (UFT) là một công cụ kiểm thử tự động toàn diện giúp đơn giản hóa và đẩy nhanh quá trình kiểm thử chức năng end-to-end cho các ứng dụng doanh nghiệp trên thiết bị di động, web và desktop. UFT cung cấp giải pháp thông minh để xây dựng và tự động hóa các bài kiểm thử, giúp đảm bảo chất lượng phần mềm cao hơn và tiết kiệm thời gian, chi phí cho doanh nghiệp.
  + Đối tượng sử dụng UFT
    - UFT phù hợp với các nhà thử nghiệm chức năng và QA cần kiểm tra toàn diện các ứng dụng.
  + Phạm vi sử dụng UFT
    - Tự động hóa việc thực thi các bài kiểm thử chức năng cho các ứng dụng web, di động và desktop.
    - Hỗ trợ kiểm thử end-to-end cho toàn bộ quy trình làm việc của ứng dụng.
    - Phát hiện lỗi hồi quy và đảm bảo tính ổn định của ứng dụng.
    - Kiểm tra các ứng dụng doanh nghiệp phức tạp trên nhiều nền tảng.
    - Tích hợp trí tuệ nhân tạo (AI) để tối ưu hóa hiệu quả kiểm thử.
  + Lợi ích sử dụng UFT
    - Tiết kiệm thời gian và công sức cho các nhà thử nghiệm.
    - Nâng cao độ chính xác và hiệu quả của quá trình kiểm thử.
    - Phát hiện lỗi sớm và giảm thiểu chi phí sửa chữa.
    - Đảm bảo chất lượng phần mềm tốt hơn.
    - Tăng tốc độ phát triển và triển khai ứng dụng.
  + Điểm nổi bật của UFT
    - Tích hợp trí tuệ nhân tạo (AI)
    - Hỗ trợ hầu hết trên các hệ điều hành như Windows, Linux, Mac,….
  + Lịch sử phát triển của UFT
    - UFT ban đầu được viết bởi Mercury Interactive và được gọi là QuickTest Professional (QTP) Sau đó, Mercury Interactive đã được Hewlett-Packard (HP) mua lại vào năm 2006. Năm 2012, lúc bản QTP 11 được phát hành, nó đã được đổi tên thành Unified Functional Testing (UFT).
  + Phân loại UFT

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**/\*\*\*\***

**1.2 Khái niệm**

* **UFT One:** Là công cụ kiểm thử tự động toàn diện giúp kiểm tra chức năng end-to-end cho các ứng dụng doanh nghiệp.
* **UFT Developer:** Là công cụ kiểm thử tự động tập trung vào kiểm thử do nhà phát triển thực hiện, giúp các nhà phát triển phần mềm tự động hóa các bài kiểm thử đơn vị và tích hợp.

**2. Phân loại theo chức năng**

**2.1 Kiểm thử chức năng end-to-end**

* UFT One hỗ trợ kiểm tra toàn bộ quy trình làm việc của ứng dụng, từ đầu đến cuối, đảm bảo rằng tất cả các chức năng hoạt động chính xác và trơn tru.
* UFT Developer không hỗ trợ kiểm thử end-to-end một cách trực tiếp.

**2.2 Kiểm thử do nhà phát triển thực hiện**

* UFT One không hỗ trợ kiểm thử do nhà phát triển thực hiện một cách trực tiếp.
* UFT Developer hỗ trợ kiểm thử đơn vị và kiểm thử tích hợp, giúp các nhà phát triển phần mềm tự động hóa các bài kiểm thử của họ.

**2.3 Phát triển theo hướng dẫn (TDD)**

* UFT One không hỗ trợ phát triển theo hướng dẫn (TDD).
* UFT Developer hỗ trợ phát triển theo hướng dẫn (TDD), giúp các nhà phát triển phần mềm viết mã hiệu quả và dễ kiểm thử hơn.

**2.4 Tự động hóa kiểm thử giao diện người dùng (GUI)**

* Cả UFT One và UFT Developer đều hỗ trợ tự động hóa kiểm thử giao diện người dùng (GUI).

**2.5 Kiểm thử hiệu suất**

* Cả UFT One và UFT Developer đều hỗ trợ kiểm thử hiệu suất (tùy chọn).

**2.6 Kiểm thử hiệu quả**

* Cả UFT One và UFT Developer đều hỗ trợ kiểm thử hiệu quả (tùy chọn).

**2.7 Tích hợp với các công cụ DevOps khác**

* Cả UFT One và UFT Developer đều hỗ trợ tích hợp với các công cụ DevOps khác.

**3. Phân loại theo ngôn ngữ lập trình**

* UFT One hỗ trợ nhiều ngôn ngữ lập trình bao gồm VBScript, C#, Python và Java.
* UFT Developer chỉ hỗ trợ ngôn ngữ lập trình C#.

**4. Phân loại theo đối tượng sử dụng**

* UFT One phù hợp với các nhà thử nghiệm chức năng và QA cần kiểm tra toàn diện các ứng dụng.
* UFT Developer phù hợp với các nhà phát triển phần mềm muốn tự động hóa các bài kiểm thử đơn vị và tích hợp, cũng như thực hành TDD.

**5. Phân loại theo giá cả**

* UFT One có giá cao hơn UFT Developer.

**6. Gói UFT One và UFT Developer kết hợp**

Micro Focus cũng cung cấp các gói UFT One và UFT Developer kết hợp để đáp ứng nhu cầu đa dạng của người dùng. Gói kết hợp này cung cấp đầy đủ các tính năng của cả hai công cụ, cho phép bạn kiểm thử end-to-end toàn diện và tự động hóa các bài kiểm thử do nhà phát triển thực hiện.

\*\*\*/

Ở đây chúng em sẽ sử dụng UFT One.

**II. Tìm hiểu UFT One**

## a. Các bước tải UFT One:

**1. Truy cập trang web của Micro Focus:**

* Mở trình duyệt web của bạn và truy cập trang web của Micro Focus: <https://www.microfocus.com/en-us/home>
* Nhấp vào tab **Sản phẩm** (Products) ở menu trên cùng.
* Tìm kiếm **UFT One** trong thanh tìm kiếm và chọn **UFT One** từ kết quả tìm kiếm.

**2. Chọn phiên bản UFT One bạn muốn tải xuống:**

* Micro Focus cung cấp hai phiên bản UFT One: **Miễn phí** và **Trả phí**.
* Phiên bản **Miễn phí** có chức năng hạn chế, nhưng nó là một lựa chọn tốt để đánh giá phần mềm.
* Phiên bản **Trả phí** có đầy đủ chức năng và được hỗ trợ bởi Micro Focus.
* Chọn phiên bản phù hợp với nhu cầu của bạn.

**3. Tạo tài khoản Micro Focus (nếu bạn chưa có):**

* Nếu bạn chưa có tài khoản Micro Focus, bạn cần tạo tài khoản mới.
* Nhấp vào nút **Đăng ký** (Sign up) và làm theo hướng dẫn trên màn hình.
* Sau khi tạo tài khoản, bạn sẽ có thể đăng nhập và tải xuống UFT One.

**4. Tải xuống UFT One:**

* Nhấp vào nút **Tải xuống** (Download) bên cạnh phiên bản UFT One bạn muốn tải xuống.
* Chọn tệp cài đặt phù hợp với hệ điều hành của bạn.
* Lưu tệp cài đặt vào máy tính của bạn.

**5. Cài đặt UFT One:**

* Chạy tệp cài đặt đã tải xuống.
* Làm theo hướng dẫn trên màn hình để hoàn tất quá trình cài đặt.

**6. Kích hoạt UFT One (nếu bạn đang sử dụng phiên bản trả phí):**

* Sau khi cài đặt UFT One, bạn cần kích hoạt nó bằng khóa cấp phép của mình.
* Khóa cấp phép sẽ được cung cấp cho bạn khi bạn mua phiên bản trả phí của UFT One.
* Làm theo hướng dẫn trên màn hình để kích hoạt UFT One.

**Lưu ý:**

* Bạn cần có kết nối internet hoạt động để tải xuống và cài đặt UFT One.
* Bạn cần có quyền quản trị viên trên máy tính của mình để cài đặt UFT One.
* Nếu bạn gặp bất kỳ vấn đề nào khi tải xuống hoặc cài đặt UFT One, bạn có thể tham khảo tài liệu trợ giúp của Micro Focus hoặc liên hệ với bộ phận hỗ trợ khách hàng của Micro Focus.

**Ngoài ra, bạn có thể tham khảo các video hướng dẫn sau để tải xuống và cài đặt UFT One:**

* <https://m.youtube.com/watch?v=h65Ax60bAnQ>
* <https://m.youtube.com/watch?v=h65Ax60bAnQ>

Chúc bạn thành công!

## b. Các chức năng chính của UFT One

* **Kiểm thử chức năng end-to-end:** UFT One hỗ trợ kiểm tra toàn bộ quy trình làm việc của ứng dụng, từ đầu đến cuối, đảm bảo rằng tất cả các chức năng hoạt động chính xác và trơn tru.
* **Kiểm thử web:** UFT One hỗ trợ kiểm tra các ứng dụng web trên nhiều trình duyệt và hệ điều hành.
* **Kiểm thử di động:** UFT One hỗ trợ kiểm tra các ứng dụng di động trên nhiều thiết bị và nền tảng.
* **Kiểm thử desktop:** UFT One hỗ trợ kiểm tra các ứng dụng desktop trên nhiều hệ điều hành.
* **Kiểm thử API:** UFT One hỗ trợ kiểm tra các API REST và SOAP.
* **Tự động hóa kiểm thử giao diện người dùng (GUI):** UFT One hỗ trợ tự động hóa kiểm thử giao diện người dùng (GUI) bằng cách ghi lại các hành động của người dùng.
* **Kiểm thử hiệu suất:** UFT One hỗ trợ kiểm thử hiệu suất để đánh giá khả năng chịu tải và hiệu suất của ứng dụng.
* **Kiểm thử hiệu quả:** UFT One hỗ trợ kiểm thử hiệu quả để đánh giá mức độ sử dụng và hiệu quả của ứng dụng.
* **Tích hợp với các công cụ DevOps khác:** UFT One hỗ trợ tích hợp với các công cụ DevOps khác để tự động hóa quy trình phát triển và triển khai phần mềm

## c. Các thành phần quan trọng của UFT One

**1. Môi trường phát triển tích hợp (IDE):**

* IDE là giao diện người dùng đồ họa cho phép người dùng viết, chỉnh sửa và chạy các kịch bản kiểm thử UFT.
* IDE bao gồm nhiều công cụ và tính năng giúp người dùng phát triển các kịch bản kiểm thử hiệu quả, chẳng hạn như:
  + Trình chỉnh sửa mã
  + Trình gỡ lỗi
  + Trình theo dõi biến
  + Trình quản lý đối tượng
  + Trình ghi kịch bản
  + Trình trình bày báo cáo

**2. Kịch bản kiểm thử:**

* Kịch bản kiểm thử là một tập hợp các hướng dẫn được viết bằng ngôn ngữ lập trình Visual Basic .NET hoặc C# để kiểm tra chức năng của một ứng dụng.
* Kịch bản kiểm thử có thể bao gồm các bước sau:
  + Khởi chạy ứng dụng
  + Nhập dữ liệu
  + Thực hiện các hành động
  + Kiểm tra kết quả

**3. Đối tượng:**

* Đối tượng là các thành phần của ứng dụng, chẳng hạn như nút, trường văn bản, hộp kiểm và menu.
* UFT sử dụng các đối tượng để tương tác với ứng dụng trong các kịch bản kiểm thử.

**4. Kho dữ liệu:**

* Kho dữ liệu là nơi lưu trữ dữ liệu được sử dụng trong các kịch bản kiểm thử.
* Dữ liệu có thể bao gồm các giá trị văn bản, số, ngày tháng và hình ảnh.
* UFT có thể truy cập dữ liệu từ nhiều nguồn khác nhau, chẳng hạn như bảng tính Excel, cơ sở dữ liệu và tệp CSV.

**5. Báo cáo:**

* Báo cáo là kết quả của quá trình thực thi kịch bản kiểm thử.
* Báo cáo cung cấp thông tin về các bước đã thực hiện, kết quả đã đạt được và các lỗi đã xảy ra.
* UFT có thể tạo nhiều loại báo cáo khác nhau, chẳng hạn như báo cáo chi tiết, báo cáo tóm tắt và báo cáo đồ họa.

**6. Add-in:**

* Add-in là các chương trình bổ sung mở rộng chức năng của UFT.
* Có nhiều loại add-in khác nhau dành cho UFT, chẳng hạn như add-in để kiểm tra hiệu suất, add-in để kiểm tra bảo mật và add-in để kiểm tra di động.

Ngoài ra, UFT còn bao gồm nhiều thành phần khác như trình quản lý dự án, trình quản lý cấu hình và trình quản lý kho dữ liệu.

# **Chương 6 : Ứng dụng Unified Functional Testing (UFT) vào kiểm thử website**

## **6.1 Phân tích thiết kế kiểm thử.**

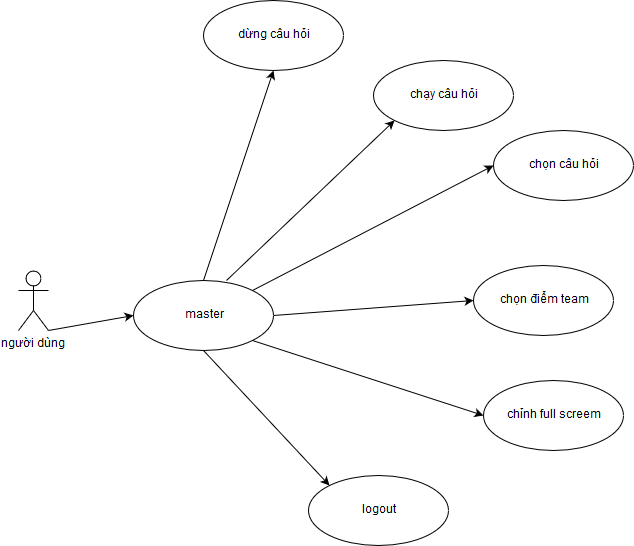
Kế hoạch kiểm thử:

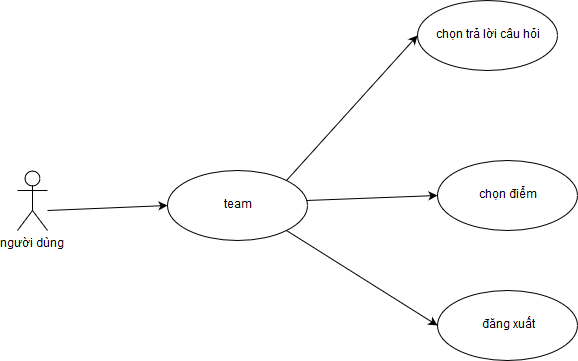
|  |  |  |  |
| --- | --- | --- | --- |
| Công việc | Số ngày làm | Ngày bắt đầu | Ngày kết thúc |
| Đọc tài liệu đặc tả | 2 | 14/05/2017 | 15/05/2017 |
| Lập kế hoạch kiểm thử | 2 | 17/05/2017 | 18/05/2017 |
| Kiểm thử đơn vị | 4 | 20/05/2017 | 23/05/2017 |
| Phân tích thiết kế kiểm thử | 3 | 20/05/2017 | 21/05/2017 |
| Thực hiện kiểm thử đơn vị | 1 | 23/05/2017 | 23/05/2017 |
| Kiểm thử hệ thống | 7 | 24/05/2017 | 30/05/2017 |
| Phân tích thiết kế các ca kiểm thử hệ thống | 5 | 24/05/2017 | 28/05/2017 |
| Thực hiện kiểm thử | 2 | 29/05/2017 | 30/05/2017 |
| Tổng hợp và đánh giá kết quả | 1 | 31/05/2017 | 31/05/2017 |
| Báo cáo kết quả kiểm thử | 1 | 01/06/2017 | 01/06/2017 |

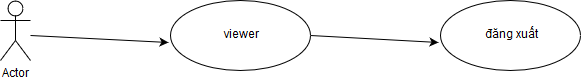
**Hình 6.1: kế hoạch kiểm thử**

### 6.1.1 Các usecase của ứng dụng.

**Hình 6.2: usecase chính**

**Hình 6.3 : usecase master.**



**Hình 6.3 : usecase team.**

**Hình 6.4 : usecase viewer**

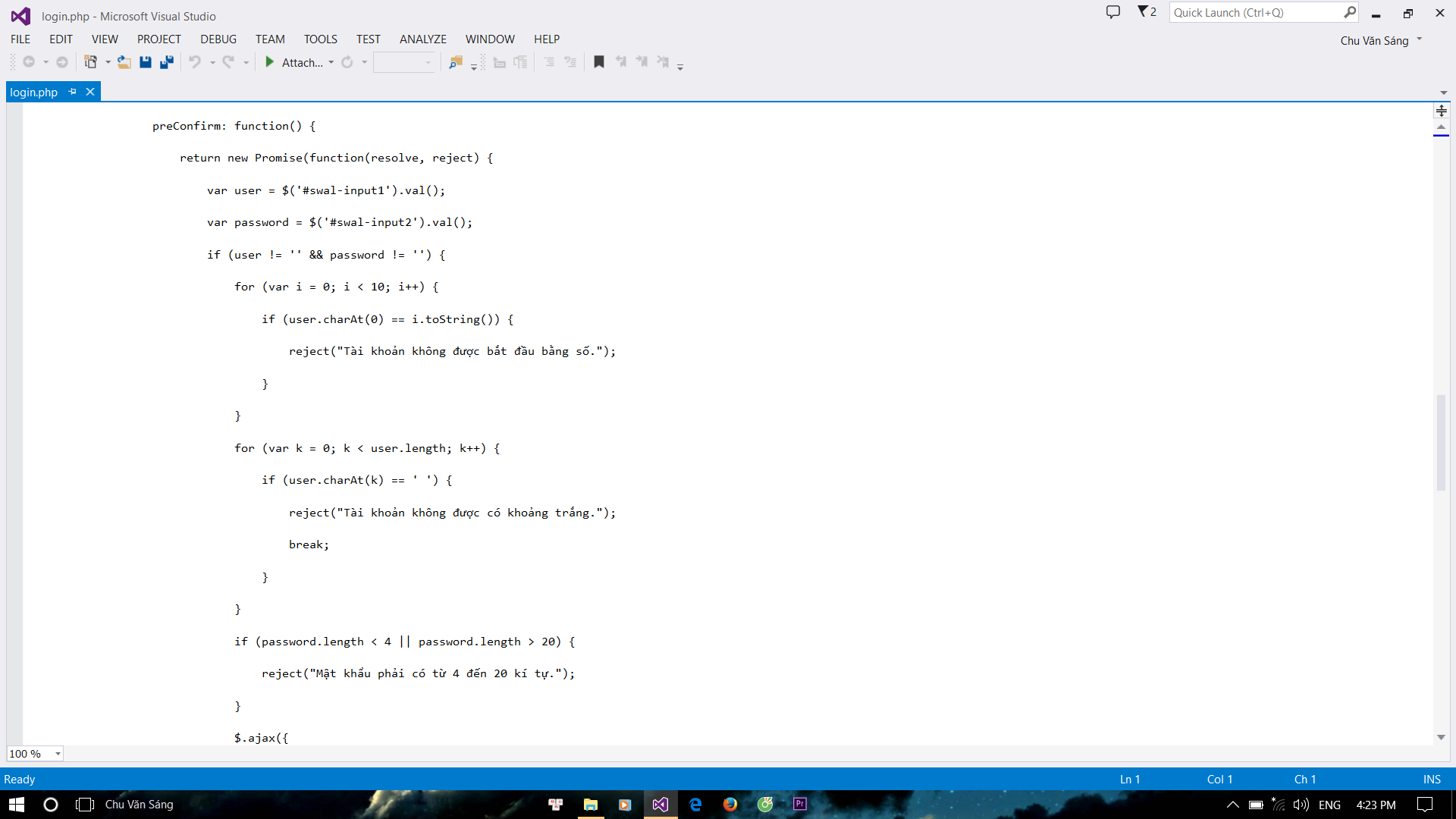
### 6.1.2 mô tả ca sử dụng đăng nhập.

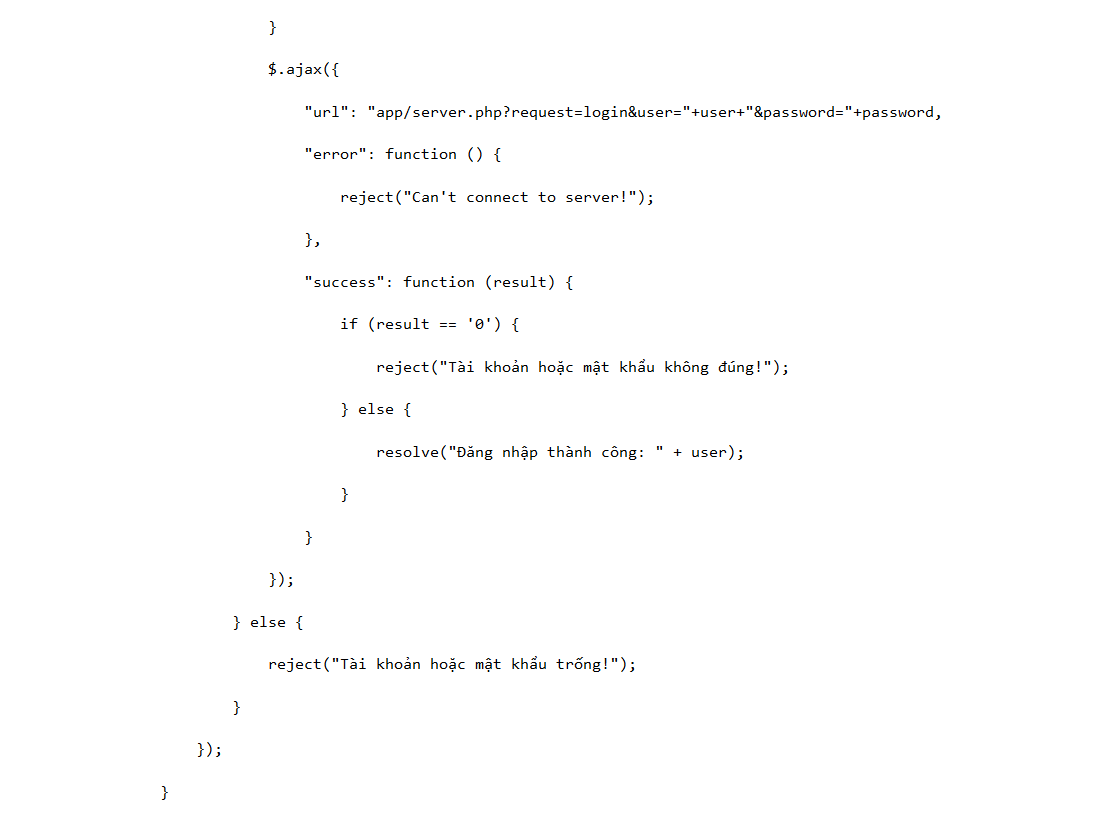
**Đối tượng:** người dùng, quản trị viên.

**Yêu cầu:** đối tượng phải truy cập vào website và chưa có phiên làm việc nào khác trước đó.

**Luồng chính:** đối tượng sẽ nhập vào tài khoản và mật khẩu. Đối tượng bấm nút "Đăng nhập" để đăng nhập hệ thống, chương trình chạy luồng phụ E1. Ca sử dụng kết thúc khi người dùng không còn ở trong trang đăng nhập nữa.

**Luồng phụ E1**: bắt đầu khi người dùng ấn nút "Đăng nhập". Hệ thống kiểm tra tài khoản có trống hay ko. Nếu có thì thông báo "tài khoản hoặc mật khẩu trống", kết thúc luồng phụ E1. Nếu không thì hệ thống kiểm tra mật khẩu có trống hay ko. Nếu có thì thông báo "tài khoản hoặc mật khẩu trống", kết thúc luồng phụ E1. Nếu không thì hệ thống kiểm tra tài khoản có bắt đầu bằng số hay không. Nếu có thì thông báo "tài khoản không được bắt đầu bằng số", kết thúc luồng E1. Nếu không thì hệ thống kiểm tra tài khoản có khoảng trắng hay không. Nếu có thì thông báo "tài khoản không được có khoảng trắng", kết thúc luồng E1. Nếu không thì hệ thống kiểm tra mật khẩu có ít hơn 4 kí tự hay không. Nếu có thì thông báo "mật khẩu phải có từ 4 đến 20 kí tự", kết thúc luồng E1. Nếu không thì hệ thống kiểm tra mật khẩu có nhiều hơn 20 kí tự hay không. Nếu có thì thông báo "mật khẩu phải có tự 4 đến 20 kí tự", kết thúc luồng E1. Nếu không thì hệ thống kiểm tra tài khoản và mật khẩu có trùng trong cơ sở dữ liệu hay không. Nếu không thì thông báo "tài khoản hoặc mật khẩu không đúng", kết thúc luồng phụ E1. Nếu có thì hiện bảng thông báo đăng nhập thành công. Người dùng chọn "OK" thì chuyển đến trang tương ứng với tài khoản trong hệ thống, kết thúc luồng phụ E1.

Mã nguồn của chức năng đăng nhập:

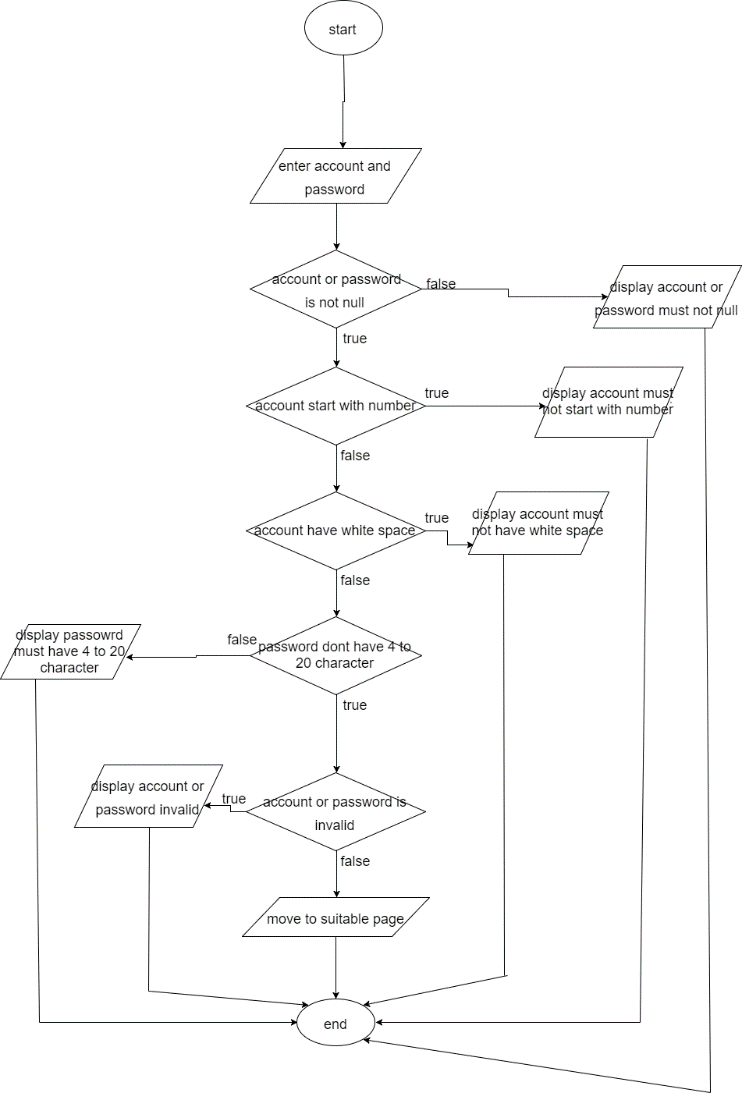
**Hình 6.5: code chức năng đăng nhập (1).**

**Hình 6.6: code chức năng đăng nhập (2).**

## **6.2 thiết kế testcase**

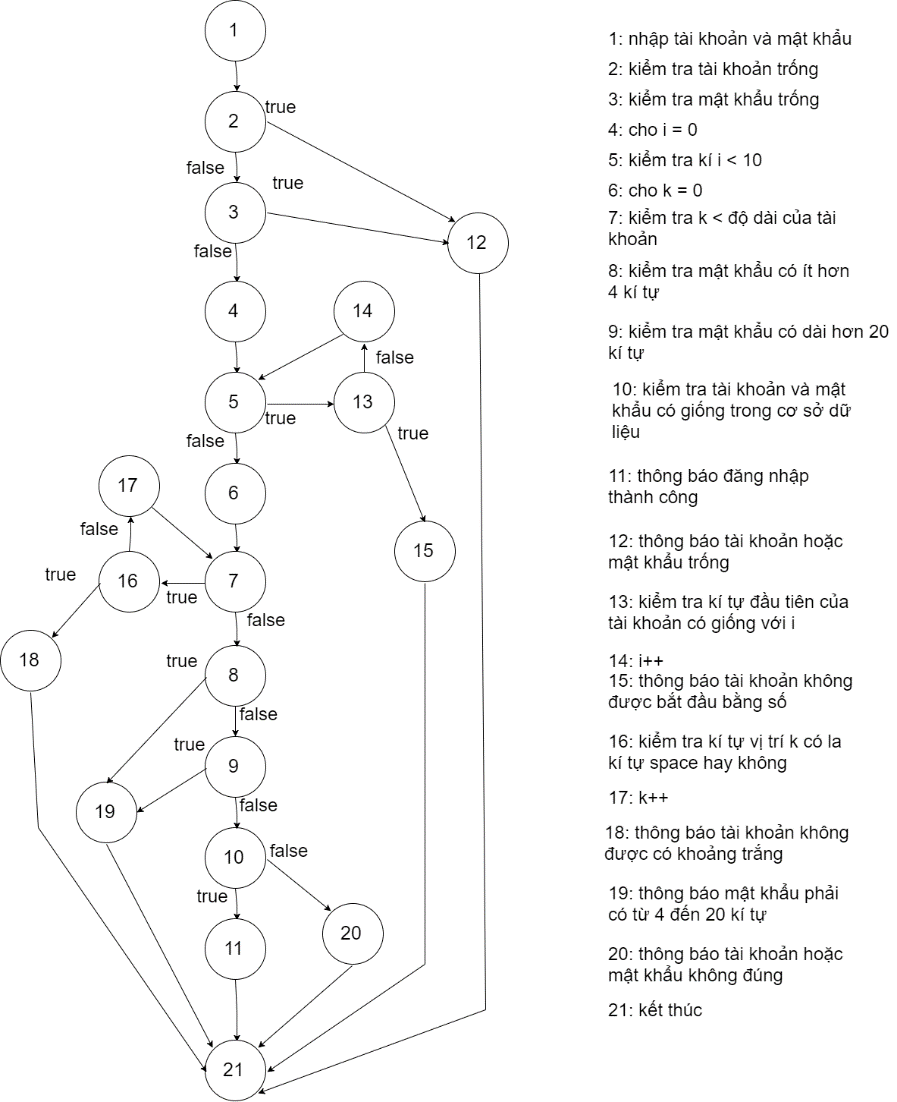
Tiến hành kiểm thử ở mức đơn vị.

### 6.2.1: kĩ thuật kiểm thử hộp trắng.



**Hình 6.7: Sơ đồ của chức năng đăng nhập.**

Dựa vào code, ta có đồ thị dòng:



**Hình 6.8: đồ thị dòng chức năng đăng nhập.**

Số nút quyết định: 9 ; Số nút : 20; số cung: 28.

Độ phức tạp: 9 + 1 = 10; 29 – 21 + 2 = 10.

Đường thi hành độc lập:

1-2-3-4-5-6-7-8-9-10-11-21

1-2-3-4-5-6-7-8-9-10-20-21

1-2-3-4-5-6-7-8-9-19-21

1-2-3-4-5-6-7-8-19-21

1-2-3-4-5-6-7-16-18-21

1-2-3-4-5-6-7-16-17-7

1-2-3-4-5-13-15-21

1-2-3-4-5-13-14-5

1-2-3-12-21

1-2-12-21

**Thiết kế testcase**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| stt | testcase | Trường hợp | Miêu tả | Kết quả mong muốn |
| 1 | Không thể thiết kế test case | 1-2-3-4-5-6-7-8-9-10-11-21 | Không xảy ra trường hợp này |  |
| 2 | Tài khoản: tung  Mật khẩu: sang | 1-2-3-4-5-6-7-8-9-10-20-21 | Tài khoản trống | Thông báo tài khoản hoặc mật khẩu trống |
| 3 | Tài khoản: master  Mật khẩu: 123456789012345678901 | 1-2-3-4-5-6-7-8-9-19-21 | Mật khẩu trống | Thông báo tài khoản hoặc mật khẩu trống |
| 4 | Tài khoản: tung  Mật khẩu: 12 | 1-2-3-4-5-6-7-8-19-21 | Tài khoản hoặc mật khẩu không chính xác | Thông báo tài khoản hoặc mật khẩu không đúng |
| 5 | Tài khoản: master  Mật khẩu: master | 1-2-3-4-5-6-7-16-18-21 | Tài khoản chứa khoảng trắng | Thông báo tài khoản không được chứa khoảng trắng |
| 6 | Tài khoản: master  Mật khẩu: master | 1-2-3-4-5-6-7-16-17-7 | Tài khoản không chứa khoảng trắng | Thông báo đăng nhập thành công |
| 7 | Tài khoản: 1master  Mật khẩu: master | 1-2-3-4-5-13-15-21 | Tài khoản bắt đầu bằng số | Thông báo tài khoản không được bắt đầu bằng số |
| 8 | Tài khoản: master  Mật khẩu: master | 1-2-3-4-5-13-14-5 | Tài khoản không bắt đầu bằng số | Thông báo đăng nhập thành công |
| 9 | Tài khoản: master  Mật khẩu: | 1-2-3-12-21 | Mật khẩu trống | Thông báo tài khoản hoặc mật khẩu trống |
| 10 | Tài khoản:  Mật khẩu: master | 1-2-12-21 | Tài khoản trống | Thông báo tài khoản hoặc mật khẩu trống |

**Hình 6.9: testcase kiểm thử hộp trắng cho chức năng login.**

Nếu chỉ dùng kiểm thử hộp trắng thì trường hợp tài khoản có khoảng trắng và mật khẩu ít hơn 4 kí tự sẽ không được kiểm tra. Trường hợp tài khoản có khoảng trắng và bắt đầu bằng số cũng không được kiểm tra, suy ra thiếu trường hợp. Do đó ta phối hợp thêm phương pháp kiểm thử hộp đen.

### 6.2.2: kĩ năng kiểm thử hộp đen.

Sử dụng phương pháp bảng hỗ trợ quyết định.

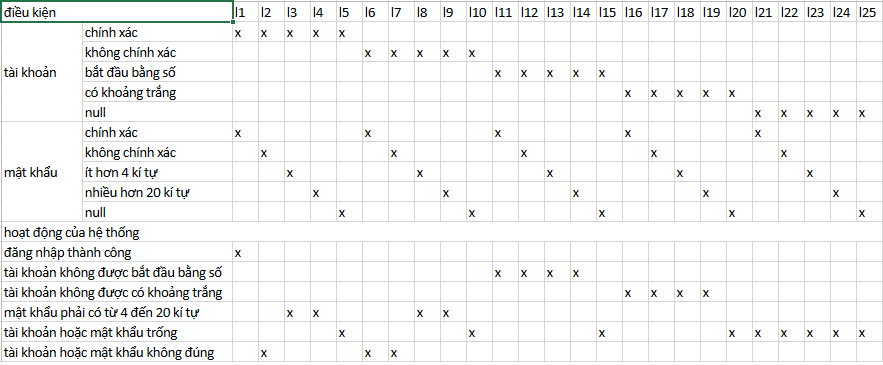
Dựa theo chức năng của hệ thống, ta có điều kiện chức năng hệ thống

Tài khoản: chính xác, không chính xác, có khoảng trắng, bắt đầu bằng số, null.

Mật khẩu: chính xác, không chính xác, ít hơn 4 kí tự, nhiều hơn 20 kí tự, null.

Thông báo: tài khoản hoặc mật khẩu không được trống; tài khoản hoặc mật khẩu không đúng; tài khoản không được bắt đầu bằng số; tài khoản không được có khoảng trắng; mật khẩu phải có tự 4 đến 20 kí tự; đăng nhập thành công.

Ta lập được bảng quyết định sau:



**Hình 6.10: bảng quyết định cho chức năng đăng nhập**

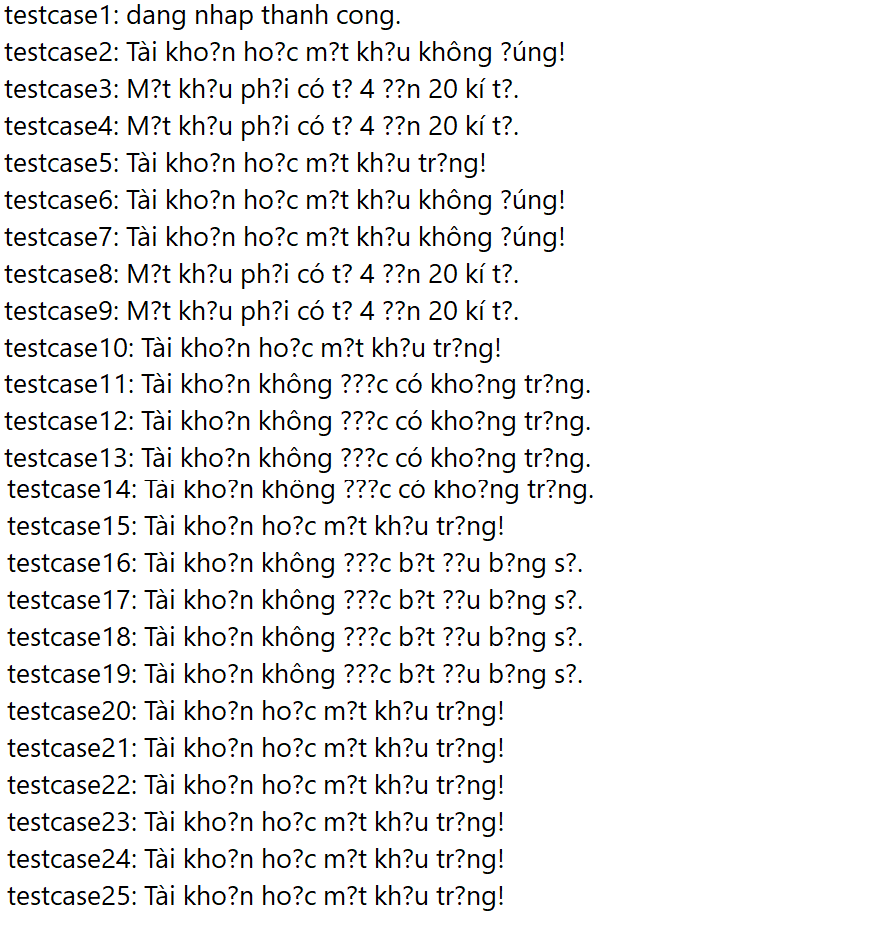
Dựa vào bảng quyết định và các test case ở kiểm thử hộp trắng, ta thiết kê được danh sách testcase sau:

|  |  |  |  |
| --- | --- | --- | --- |
| stt | Testcase | Miêu tả | Kết quả mong muốn |
| 1 | Tài khoản: master  Mật khẩu: master | Tài khoản và mật khẩu chính xác. | Thông báo đăng nhập thành công. |
| 2 | Tài khoản: master  Mật khẩu: sang | Tài khoản chính xác, mật khẩu không chính xác. | Thông báo tài khoản hoặc mật khẩu không đúng. |
| 3 | Tài khoản: master  Mật khẩu: ma | Tài khoản chính xác, mật khẩu ít hơn 4 kí tự. | Thông báo mật khẩu có từ 4 đến 20 kí tự |
| 4 | Tài khoản: master  Mật khẩu: 123456  789012345678901 | Tài khoản chính xác, mật khẩu chính nhiều hơn 20 kí tự | Thông báo mật khẩu có từ 4 đến 20 kí tự |
| 5 | Tài khoản: master  Mật khẩu: | Tài khoản chính xác, mật khẩu trống | Thông báo tài khoản hoặc mật khẩu trống |
| 6 | Tài khoản: tung  Mật khẩu: master | Tài khoản không chính xác, mật khẩu chính xác | Thông báo tài khoản hoặc mật khẩu không đúng |
| 7 | Tài khoản: tung  Mật khẩu: sang | Tài khoản không chính xác, mật khẩu không chính xác | Thông báo tài khoản hoặc mật khẩu không chính xác |
| 8 | Tài khoản: tung  Mật khẩu: ma | Tài khoản không chính xác, mật khẩu ít hơn 4 kí tự | Thông báo mật khẩu phải có tự 4 đến 20 kí tự |
| 9 | Tài khoản: tung  Mật khẩu: 123456789012345678901 | Tài khoản không chính xác, mật khẩu nhiều hơn 20 kí tự | Thông báo mật khẩu phải có từ 4 đến 20 kí tự |
| 10 | Tài khoản: tung  Mật khẩu: | Tài khoản không chính xác, mật khẩu trống | Thông báo tài khoản hoặc mật khẩu trống |
| 11 | Tài khoản: mas ter  Mật khẩu: master | Tài khoản có khoảng trắng, mật khẩu chính xác | Thông báo tài khoản không được có khoảng trắng |
| 12 | Tài khoản: mas ter  Mật khẩu: sang | Tài khoản có khoảng trắng, mật khẩu không chính xác | Thông báo tài khoản không được có khoảng trắng |
| 13 | Tài khoản: mas ter  Mật khẩu: ma | Tài khoản có khoảng trắng, mật khẩu ít hơn 4 kí tự | Thông báo tài khoản không được có khoảng trắng |
| 14 | Tài khoản: mas ter  Mật khẩu: 123456789012345678901 | Tài khoản có khoảng trắng, mật khẩu nhiều hơn 20 kí tự | Thông báo tài khoản không được có khoảng trắng |
| 15 | Tài khoản: mas ter  Mật khẩu: | Tài khoản có khoảng trắng, mật khẩu trống | Thông báo tài khoản hoặc mật khẩu trống |
| 16 | Tài khoản: 1master  Mật khẩu: master | Tài khoản bắt đầu bằng số, mật khẩu chính xác | Thông báo tài khoản không được bắt đầu bằng số |
| 17 | Tài khoản: 1master  Mật khẩu: sang | Tài khoản bắt đầu bằng số, mật khẩu không chính xác | Thông báo tài khoản không được bắt đầu bằng số |
| 18 | Tài khoản: 1master  Mật khẩu: ma | Tài khoản bắt đầu bằng số, mật khẩu ít hơn 4 kí tự | Thông báo tài khoản không được bắt đầu bằng số |
| 19 | Tài khoản: 1master  Mật khẩu: 123456789012345678901 | Tài khoản bắt đầu bằng số, mật khẩu nhiều hơn 20 kí tự | Thông báo tài khoản không được bắt đầu bằng số |
| 20 | Tại khoản: 1master  Mật khẩu: | Tài khoản bắt đầu bằng số, mật khẩu trống | Thông báo tài khoản hoặc mật khẩu trống |
| 21 | Tài khoản:  Mật khẩu: master | Tài khoản trống, mật khẩu chính xác | Thông báo tài khoản hoặc mật khẩu trống |
| 22 | Tài khoản:  Mật khẩu: sang | Tài khoản trống, mật khẩu không chính xác | Thông báo tài khoản hoặc mật khẩu trống |
| 23 | Tài khoản:  Mật khẩu: ma | Tài khoản trống, mật khẩu ít hơn 4 kí tự | Thông báo tài khoản hoặc mật khẩu trống |
| 24 | Tài khoản:  Mật khẩu: 123456789012345678901 | Tài khoản trống, mật khẩu nhiều hơn 20 kí tự | Thông báo tài khoản hoặc mật khẩu trống |
| 25 | Tài khoản:  Mật khẩu: | Tài khoản trống, mật khẩu trống | Thông báo tài khoản hoặc mật khẩu trống |

**Hình 6.10: danh sách usecase kiểm thử chức năng đăng nhập.**

## **6.3 Thực thi kiểm thử:**

**Hình 6.11: dữ liệu test case(1)**



**Hình 6.12: kết quả test.**

## **6.4 Tổng hợp kết quả test:**

|  |  |  |
| --- | --- | --- |
| Testcase pass | 25 | 100% |
| Testcase false | 0 | 0% |
| Tổng testcase đã test | 25 | 100% |
| Tổng testcase chưa test | 0 | 0% |
| Tổng testcase | 25 |  |

# 

# **Kết luận**

Kiểm thử phần mềm là một hoạt động quan trọng nhằm đảm bảo chất lượng phần mềm.

Việc nghiên cứu lựa chọn các kỹ thuật và chiến lược kiểm thử phần mềm phù hợp giúp cho việc kiểm thử có hiệu quả, giảm chi phí và thời gian. Việc xây dựng tài liệu kiểm thử phần mềm hợp lý sẽ giúp cho việc tổ chức, quản lý và thực hiện kiểm thử có hiệu quả.

Những vấn đề đã đạt được

- Nắm được tổng quan về kiểm thử phần mềm: các phương pháp, kỹ thuật kiểm thử phần mềm và các vấn đề liên quan…..

- Tìm hiểu và nắm được phương pháp thiết kế test case trong kiểm thử phần mềm và áp dụng phương pháp đó với bài toán cụ thể.

- Nghiên cứu những chức năng và cách thức hoạt động kiểm thử công cụ mã nguồn mở Unified Functional Testing (UFT).

- Giúp các bạn có cái nhìn rõ hơn về kiểm thử phần mềm.

Hướng phát triển của đề tài trong tương lai

Khi nghiên cứu về kiểm thử phần mềm nói chung và công cụ Unified Functional Testing (UFT) nói riêng, chúng em đã hiểu được kiểm thử là rất quan trọng trong quy trình sản xuất phần mềm, đảm bảo chất lượng phần mềm. Sự áp dụng với kiến thức tìm hiểu được mới chỉ dừng lại ở một bài toán nhỏ. Hướng phát triển của báo cáo là:

- Thực hiện kiểm thử trên mô hình bài toán chi tiết hơn hơn.

- Tìm hiểu và nghiên cứu thêm về các công cụ kiểm thử tự động, kiểm thử website, khác.

# **Tài liệu tham khảo**

<https://www.microfocus.com/documentation/silk-central/200/en/silkcentral-help-en/GUID-531809BA-688F-41D5-BDB2-FCE786A284CE.html>

<https://www.google.com.vn>

[www.testervn.com](http://www.testervn.com)

[www.testingvn.com](http://www.testingvn.com)