**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**



**SEMINAR MÔN KIỂM CHỨNG PHẦN MỀM**

**Tên Đề Tài:**

**Functional Tests & Using Scripts to Test**

Giáo viên hướng dẫn: Th.S Hầu Nguyễn Thành Nam

Giáo viên thực hàng: Th.S Thái Hải Dương

Sinh viên thực hiện: Phạm Hoàng Hải Sơn – 13520708

Nguyễn Văn Quang – 13520675

Nguyễn Thị Hằng – 13520244

NHẬN XÉT CỦA GIẢNG VIÊN

………………………………………………………………………………………………....

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

………………………………………………………………………………………………….

Mục lục

[Chương I : Mục Tiêu Seminar 1](#_Toc470814675)

[Chương II: Tổng Quan Về Functional Tests 1](#_Toc470814676)

[2.1 Đặt vấn đề 1](#_Toc470814677)

[2.2 Các phương pháp 2](#_Toc470814678)

[2.2.1 Giới thiệu 2](#_Toc470814679)

[2.2.2 Quy trình thực hiện chung 2](#_Toc470814680)

[2.2.3 Một ví dụ về loại tính năng để chuẩn bị cho kiểm thử chức năng 3](#_Toc470814681)

[2.3 Các phương pháp kiểm thử chức năng 7](#_Toc470814682)

[2.3.1 Functional Acceptance Simple Tests(test đơn giản của chức năng nghiệm thu) 7](#_Toc470814683)

[2.3.2 Task-Oriented Functional Tests(kiểm chứng chức năng nhiệm vụ hướng) 9](#_Toc470814684)

[2.3.3 Forced-Error Tests 10](#_Toc470814685)

[2.3.4 Boundary Condition Tests and Equivalent Class Analysis( Kiểm tra Điều kiện Biên và Các Lớp Tương đương) 16](#_Toc470814686)

[2.3.5 Exploratory Testing 16](#_Toc470814687)

[2.3.6 Which Method Is It? 18](#_Toc470814688)

[2.4 Kết luận 21](#_Toc470814689)

[2.5 Hướng phát triển 21](#_Toc470814690)

[Chương III: Using Scripts to Test 22](#_Toc470814691)

[3.1 Đặt vấn đề 22](#_Toc470814692)

[3.2 Nội dung chính 22](#_Toc470814693)

[3.2.1 Giới thiệu 22](#_Toc470814694)

[3.2.2 Scripting Languages 22](#_Toc470814695)

[3.2.3 Scripting Good Practice 27](#_Toc470814696)

[3.3 Kết luận 28](#_Toc470814697)

[Chương IV: Demo 28](#_Toc470814698)

[4.1 Giới thiệu phần mềm 28](#_Toc470814699)

[4.2. Unit Test & Integration Test 29](#_Toc470814700)

[4.2.1 Unit Test 30](#_Toc470814701)

[4.2.2 Integration Test 34](#_Toc470814702)

[4.3 Auto Deploy & Check Unit Test Coverage 36](#_Toc470814703)

[4.4 Automation Test 38](#_Toc470814704)

[4.4.1 Chức năng lập phiếu nhập hàng 38](#_Toc470814705)

[4.4.2 Chức năng quản lý nhân viên 42](#_Toc470814706)

[4.4.3 Chức năng đăng nhập 47](#_Toc470814707)

[4.4.4 Chức năng quản lý giới tính 49](#_Toc470814708)

[4.5 Kết luận 54](#_Toc470814709)

# Chương I : Mục Tiêu Seminar

* Giúp các bạn hiểu được về kiểm thử chức năng là gì?
* Nắm bắt được các dạng kiểm thử cũng như các kỹ thuật được sử dụng trong kiểm thử chức năng và ứng dụng trong kiểm thử web.
* Ứng dụng scripts trong kiểm chứng
* Bạn sẽ hiểu được Scripting Languages là gì. Biết được tại sao lại sử dụng Scripting Languages trong quá trình test mà không sử dụng compiled language. Các công dụng của nó trong quá trình kiểm thử và làm thế nào để chuẩn bị được một bảng scripting tốt cho quá trình test
* Demo trực quan qua các ví dụ

# Chương II: Tổng Quan Về Functional Tests

## **2.1 Đặt vấn đề**

Mục đích của phần này là hiểu được kiểm thử chức năng là gì, chỉ ra một số loại test và các kỹ thuật được sử dụng trong test chức năng, cho dù chúng ta có thông số kỹ thuật hay không. Tiền đề của nghiên cứu này là để tìm những lỗi trong quá trình kiểm tra xem sản phẩm có hữu ích cho người dùng mà nó nhắm tới ko và nó có làm được những điều mà người dùng muốn ko. Phần này phân tích một loạt các hàm để test khả năng sử dụng của chúng .

## **2.2 Các phương pháp**

### **2.2.1 Giới thiệu**

* + Kiểm thử chức năng (Functional test) một dạng của kiểm thử hộp đen mà các test case của nó được thiết kế dựa trên các đặc tả của các thành phần chương trình cần kiểm thử.
  + Kiểm thử chức năng kiểm tra các chức năng bằng cách cung cấp các thông số đầu vào và kiểm tra các kết quả xuất ra.

### **2.2.2 Quy trình thực hiện chung**

* + Xác định chức năng cần kiểm thử
  + Tạo dữ liệu đầu vào dựa trên đặc tả của chức năng đó
  + Xác định dữ liệu đầu ra cũng dựa trên đặc tả chức năng
  + Tiến hành kiểm thử
  + So sánh kết quả thực tế với kết quả mong đợi

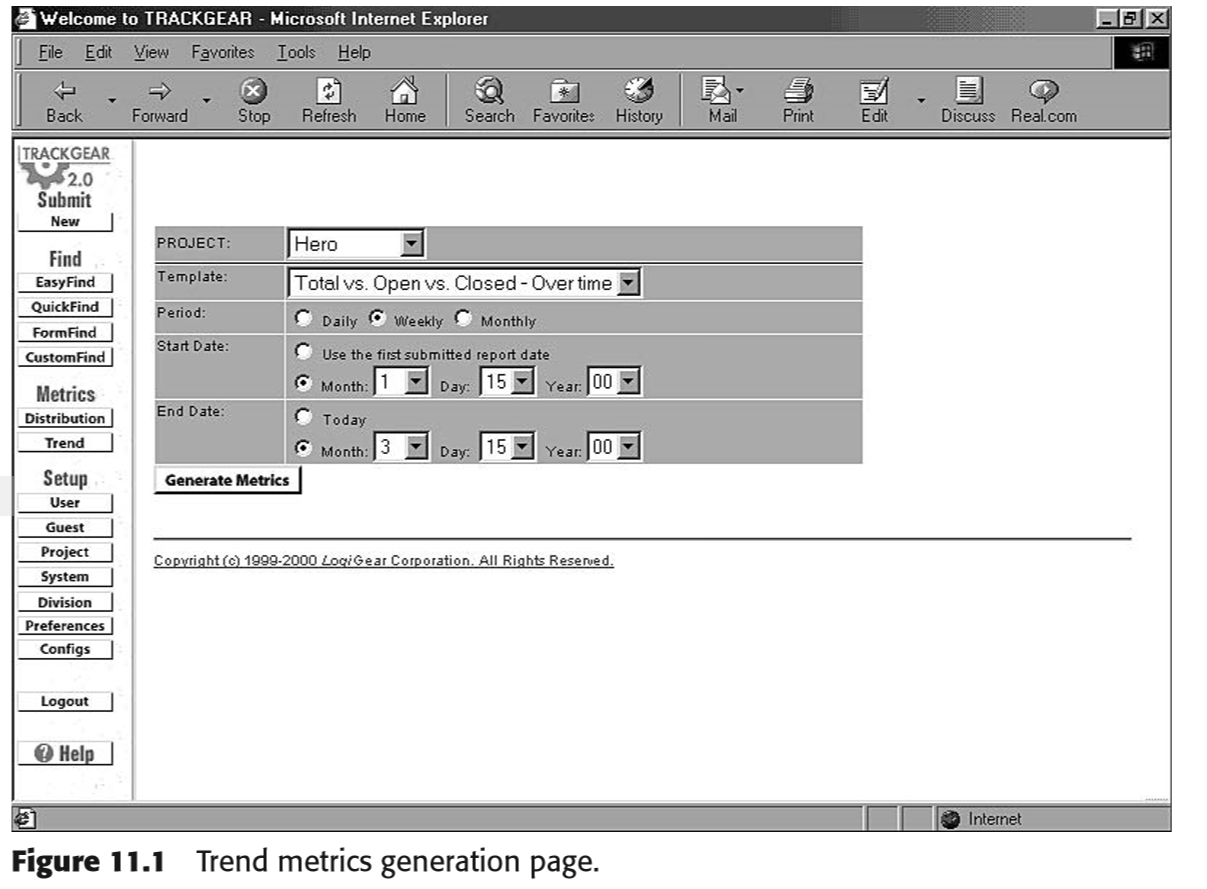
Để xác định rõ hơn phạm vi của kiểm thử chức năng, xem xét mức độ khác nhau về chức năng của một ứng dụng:

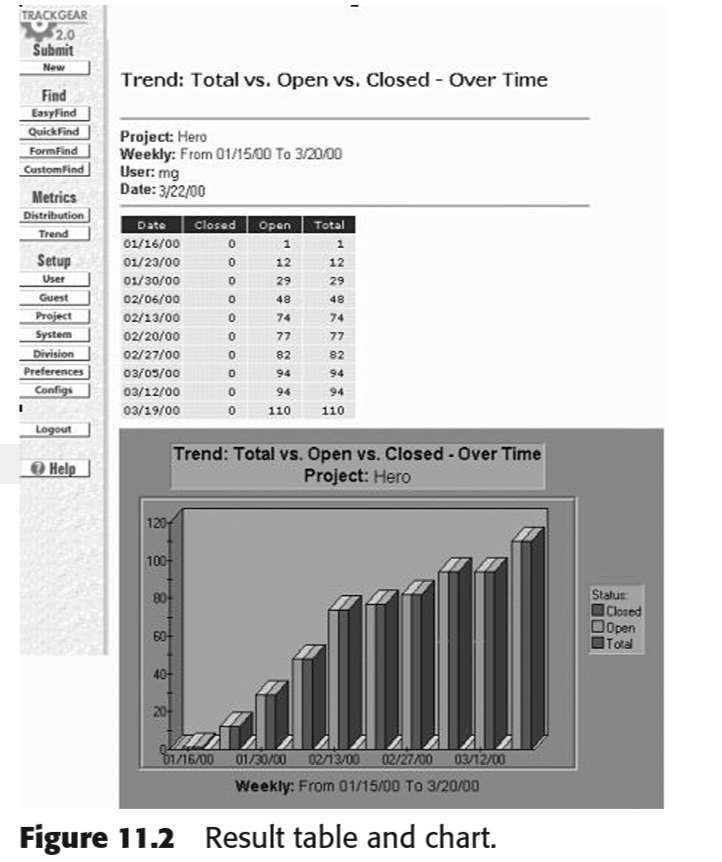
* FAST: Control input với control navigation có làm việc như mong đợi không?
* TOFT: Các ứng dụng có thể làm điều gì đó hữu ích như mong đợi?
* Boundary: Điều gì xảy ra bên cạnh các quy định sử dụng?
* Forced-error: Điều gì xảy ra khi một lỗi điều kiện/lỗi nào đó xảy ra?
* Exploratory: kinh nghiệm nói gì về những khu vực có vấn đề tiềm tàng trong các ứng dụng? Điều này liên quan đồng thời tới việc học tập, lập kế hoạch và kiểm tra thực hiện.
* Software attacks: Tại sao phần mềm lỗi? Làm thế nào để bạn chuyển bài học kinh nghiệm thành một loạt các cuộc tấn công để lộ ra lỗi của phần mềm?

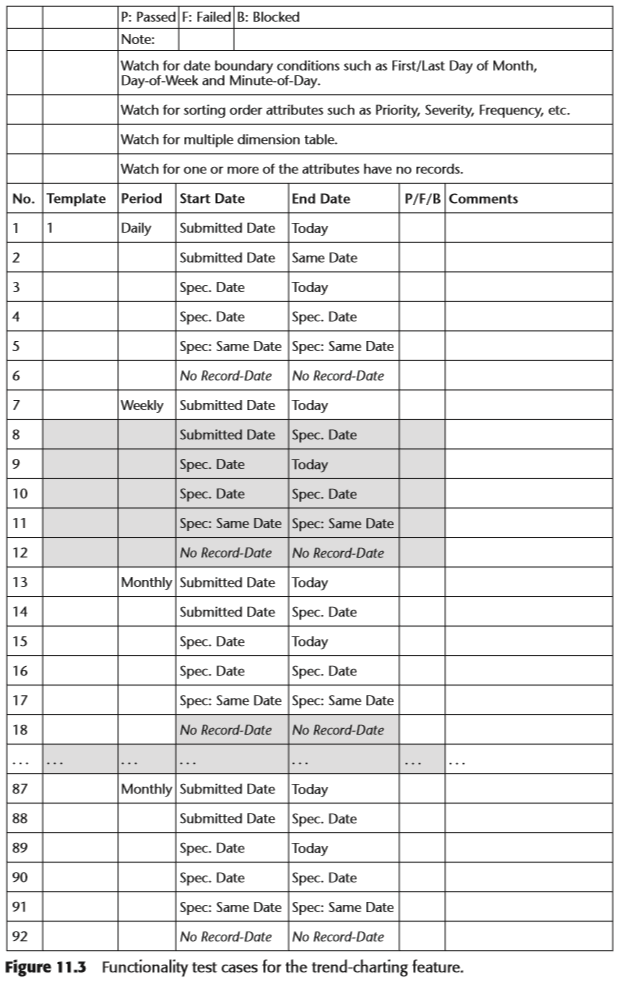
### **2.2.3 Một ví dụ về loại tính năng để chuẩn bị cho kiểm thử chức năng**

#### **Testing the Sample Application**

Chúng tôi sẽ sử dụng tính năng chart-generation của ứng dụng mẫu để trình bày một số trường hợp cụ thể về functional tests. Chương này mô tả tính năng này như một applet Java duy nhất chạy trên trình duyệt client để hiển thị các số liệu lỗi. Đối với mục đích của ví dụ này, các số liệu lỗi đo hoạt động báo cáo lỗi hoặc giải quyết tại một thời điểm (số liệu phân phối) hoặc theo thời gian (số liệu xu hướng) để hỗ trợ giám sát tình trạng project kiểm thử. Trong các ứng dụng mẫu, các số liệu có thể được hiển thị theo dạng bảng và biểu đồ. Tính năng này được kiểm thử cho phép người sử dụng để tạo ra các số liệu xu hướng dựa trên nhiều tiêu chí. Các chức năng hoạt động bằng cách lựa chọn (1) một mẫu để xác định loại dữ liệu hoặc so sánh với biểu đồ; (2) thời gian (hàng ngày, hàng tuần, hoặc hàng tháng); và (3) sự bắt đầu và ngày kết thúc. Hình 11.1 cho thấy các trang thế hệ các số liệu xu hướng từ project mẫu. Hình 11.2 minh họa bảng kết quả và biểu đồ từ các tiêu chí lựa chọn trong hình 11.1. Hình 11.3 cho thấy một ma trận mà các danh sách trường hợp kiểm thử để nhắm mục tiêu các chức năng của dụng tính năng chart-generation của ứng dụng mẫu.







## **2.3 Các phương pháp kiểm thử chức năng**

Một số phương pháp kiểm tra hộp đen phổ biến mà có thể được áp dụng để kiểm tra chức năng sẽ được thảo luận trong phần này.

### **2.3.1 Functional Acceptance Simple Tests(test đơn giản của chức năng nghiệm thu)**

Chức năng chấp thuận kiểm tra đơn giản (FASTs) đại diện cho mức độ thứ hai của sự chấp nhận kiểm thử. Thay vì chỉ bao gồm một mẫu nhỏ của các chức năng của chương trình (như trong RAT), FAST rộng bề ngang, nhưng nông bề sâu. FAST luyện tập mức thấp nhất của chức năng cho mỗi lệnh của một chương trình. Sự kết hợp của các chức năng, tuy nhiên, không có vấn đề làm thế nào tích hợp vững chắc, không được kiểm tra trong phạm vi của FAST. Những vấn đề này được xem xét trong kiểm thử chức năng nhiệm vụ theo định hướng (TOFT), được thảo luận sau trong chương này.

Một team test có thể từ chối một build sau khi nó đã được chấp nhận vào FAST. Tuy nhiên, điều này là rất hiếm. Sự từ chối thể hiện một quyết định của team rằng, tiếp tục kiểm thử của build này sẽ là một sự lãng phí thời gian, hoặc là bởi vì rất nhiều test bị chặn hoặc vì việc xây dựng chính nó là không hợp lệ. Thường xuyên hơn, team test vẫn tiếp tục kiểm thử một lỗi build nhưng chỉ định một số testers để làm việc khác nếu xét nghiệm của họ bị chặn bởi lỗi.

Một trong những mục tiêu của FAST là để kiểm tra các hành vi thích hợp của giao diện người dùng (UI) controls dựa trên thiết kế dự định (ví dụ, hộp văn bản, danh sách thả xuống, nút radio, vv). Điều này đòi hỏi phải kiểm tra sự tồn tại của UI controls trên mỗi trang, cửa sổ, hoặc hộp thoại; kiểm tra nếu các trạng thái mặc định (như enable,disable,Hightlighted, vv) là như dự định; kiểm tra nếu giá trị mặc định hoặc chế độ lựa chọn được như dự định; kiểm tra nếu trình tự tab được như dự định; và nếu hành vi của các phím tắt (ví dụ, Ctrl-X, Ctrl-V, vv) và các phím truy cập khác (ví dụ, Alto cho Open) là như dự định.

Trong một môi trường Web, điều cần kiểm tra trong FAST bao gồm:

* Kiểm tra Link: chẳng hạn như Link nội dung, Link hình ảnh thu nhỏ, Link bitmap, và các Link Image Map
* Kiểm tra Điều khiển cơ bản: chẳng hạn như backward và forward navigating, zoom-in và zoom-out, điều khiển giao diện người dùng khác, và kiểm tra content-refreshing
* Kiểm tra lệnh hành động, chẳng hạn như thêm, xóa, cập nhật, và các loại khác của dữ liệu; tạo hồ sơ người dùng hoặc tài khoản người dùng bao gồm cả các tài khoản e-mail, tài khoản cá nhân và tài khoản nhóm dựa trên; và kiểm tra dữ liệu nhập cảnh
* Kiểm tra Các tính năng khác như log in / log out, thông báo e-mail, tìm kiếm, xác nhận thẻ tín dụng và xử lý, hoặc xử lý quên mật khẩu

Một số lỗi đơn giản bạn có thể tìm thấy trong quá trình này bao gồm những điều sau đây:

* Liên kết bị hỏng
* Thiếu hình ảnh
* Liên kết sai
* Hình ảnh sai
* Liên kết chính xác không có hoặc nội dung cũ
* Lỗi trong tính toán đặt hàng / mua hàng
* Bỏ qua việc phân loại thẻ tín dụng
* Chấp nhận tín dụng hết hạn
* Chấp nhận số thẻ tín dụng hợp lệ
* Nội dung không chính xác hoặc trong bối cảnh email trả lời tự động
* Không có thông tin tình báo trong xác minh địa chỉ
* Máy chủ không đáp ứng (không có máy chủ cập nhật tin nhắn cho người sử dụng) (DNS) lỗi cho dịch vụ tên miền
* Không có khả năng xác nhận địa chỉ e-mail không hợp lệ của người dùng

### **2.3.2 Task-Oriented Functional Tests(kiểm chứng chức năng nhiệm vụ hướng)**

Kiếm chứng chức năng nhiệm vụ theo định hướng (TOFTs) Kiểm tra các tác vụ mà chức năng thực hiện, đảm bảo các tác vụ đó thực hiện chính xác. Chúng chứa các test case "tích cực", cái mà kiểm tra chức năng chương trình bằng cách so sánh các kết quả của nhiệm vụ thực hiện với thông số kỹ thuật sản phẩm và các tài liệu yêu cầu, nếu chúng tồn tại hoặc sự mong đợi của người sử dụng hợp lý. Tính toàn vẹn của mỗi công việc cá nhân thực hiện bởi chương trình được kiểm tra cho chính xác. Nếu hành vi hoặc đầu ra thay đổi từ những gì được quy định trong các yêu cầu sản phẩm, một báo cáo vấn đề được đệ trình.

Các TOFTs được cấu trúc xung quanh danh sách các tính năng để được kiểm tra. Để đến với một danh sách các tính năng cần đc test, các đặc điểm kỹ thuật sản phẩm cần được mổ xẻ một cách cẩn thận. Bản thân sản phẩm cũng phải được kiểm tra về những tính năng mà không được xác định rõ hoặc không có trong tất cả các đặc điểm kỹ thuật.

. Trong ngắn hạn, mọi chức năng và tính năng trở thành một mục trong danh sách các tính năng- được-kiểm tra. Xem xét cũng nên được trao cho những ảnh hưởng cạnh tranh và nhu cầu thị trường trong việc phát triển các chi tiết của danh sách. Ví dụ, nếu áp lực cạnh tranh đòi hỏi là một chức năng nhất định phải thực hiện trong vòng chưa đầy hai giây, sau đó yêu cầu đó nên được bổ sung vào danh sách các tính năng-để-được-kiểm tra.

Một khi bạn danh sách các tính năng-để-được-kiểm tra được hoàn tất, mỗi mục trên đó nên được sử dụng để xác định một trường hợp kiểm thử để kiểm tra xem các yêu cầu của tính năng đã được đáp ứng.

|  |
| --- |
| TESTING THE SAMPLE APPLICATION  Chức năng nghiệm thu kiểm thử ma trận đơn giản - Trend Metrics kiểm tra xem một biểu đồ được tạo ra khi các tester lựa chọn các tiêu chí cụ thể. FAST không nhất thiết phải kiểm tra xem các biểu đồ được tạo ra trình bày dữ liệu chính xác. Việc kiểm tra được thông qua ngay cả khi nội dung của một số liệu được tạo ra bởi một trong các trường hợp kiểm tra là không chính xác, mặc dù báo cáo vấn đề cần phải được viết lên để theo dõi vấn đề. The FAST, Trong trường hợp này, chỉ cần đảm bảo rằng một cái gì đó được rút ra, bất kể độ chính xác. Sự chính xác hoặc các đầu ra sẽ là trọng tâm của TOFT, được thảo luận trong phần tiếp theo. Đối với Chức năng nghiệm thu kiểm thử đơn giản - Trend Metrics, chúng tôi chọn trường hợp kiểm thử để thực hiện một sự kết hợp của sự lựa chọn chức năng. |

### **2.3.3 Forced-Error Tests**

FET là việc ép buộc chương trình vào các trường hợp lỗi (các lỗi này được lấy ra từ việc phân tích tài liệu thiết kế, kinh nghiệm của tester,…) mục đích là tìm ra các lỗi ẩn, kiểm tra các xử lý lỗi đó đã đúng và hiệu quả hay chưa.

Giả sử rằng bạn đang kiểm tra các lĩnh vực văn bản trong một form đăng ký trực tuyến và đặc điểm kỹ thuật của chương trình không cho phép các biểu tượng khác chữ cái trong trường Name. Một lỗi điều kiện sẽ được tạo ra nếu bạn nhập "123456" (hoặc bất kỳ ký tự khác ngoài chữ cái). Hãy nhớ rằng, đối với bất kỳ điều kiện hợp lệ, luôn luôn là một điều kiện không hợp lệ.

Một danh sách đầy đủ các lỗi điều kiện thường rất khó để tập hợp. Có một số cách biên dịch ra danh sách các lỗi điều kiện bao gồm những điều sau đây:

* Thu thập một danh sách các thông báo lỗi từ các nhà phát triển.
* Phỏng vấn các nhà phát triển.
* Kiểm tra các dữ liệu chuỗi trong một file resource.
* Thu thập thông tin từ các thông số kỹ thuật.
* Phân tích mọi khả năng có thể xảy ra đối với mỗi trường hợp lỗi
* Sử dụng kinh nghiệm của bạn.
* Sử dụng ma trận kiểm tra tiêu chuẩn đầu vào hợp lệ / không hợp lệ

|  |
| --- |
| TESTING THE SAMPLE APPLICATION  The Task-Oriented Functional Test Matrix— Trend Metrics sẽ bao gồm toàn bộ bộ kiểm tra chi tiết trong hình 11.3 để hoàn toàn thực hiện khía cạnh này của các số liệu xu hướng project mẫu của tính năng này. Lưu ý rằng nó sẽ mất rất nhiều hơn ma trận này đơn giản để hoàn toàn thực hiện điều này một phần của chương trình. Cẩn thận với những từ như "complete."  Những trường hợp kiểm thử kiểm tra xem dữ liệu đầu ra (các số liệu và biểu đồ xu hướng) phản ánh chính xác các thông số đầu vào (Trong trường hợp này, một tập hợp dữ liệu mà kết hợp hoạt động lỗi, theo dõi hàng ngày). Các tập dữ liệu mẫu bao gồm số lượng báo cáo mới đã mở ra so với số lượng báo cáo đóng cửa trong một thời gian nhất định. Các trường hợp kiểm tra TOFT kiểm tra xem các số liệu đầu ra phản ánh cụ thể của tập dữ liệu đầu vào. |

Một khi bạn có một danh sách đầy đủ các lỗi điều kiện, mỗi lỗi điều kiện nên được chạy thông qua quá trình kiểm thử sau đây:

1. Đưa các phần mềm vào các tình trạng lỗi được chỉ định trong trường hợp kiểm thử.
2. Kiểm tra logic phát hiện lỗi. Lỗi xử lý bắt đầu với việc phát hiện lỗi. Nếu một lỗi đi chưa được phát hiện, không có xử lý. Từ góc độ phát triển, phát hiện được thực hiện thông qua xác nhận. Ví dụ, trong các mã, giá trị đầu vào được xác nhận bằng cách sử dụng một số hạn chế. Nếu giá trị không đáp ứng các ràng buộc, sau đó làm một cái gì đó về nó (tức là, không cho phép đầu vào). Trường hợp kiểm tra nên được thiết kế để chọc vào các lỗ hổng của logic xác nhận.
3. Kiểm tra logic xử lý lỗi. Bây giờ các phát hiện được đặt ra, chúng ta cần phải kiểm tra mỗi điều kiện lỗi phát hiện được xử lý như thế nào từ các khía cạnh sau đây:

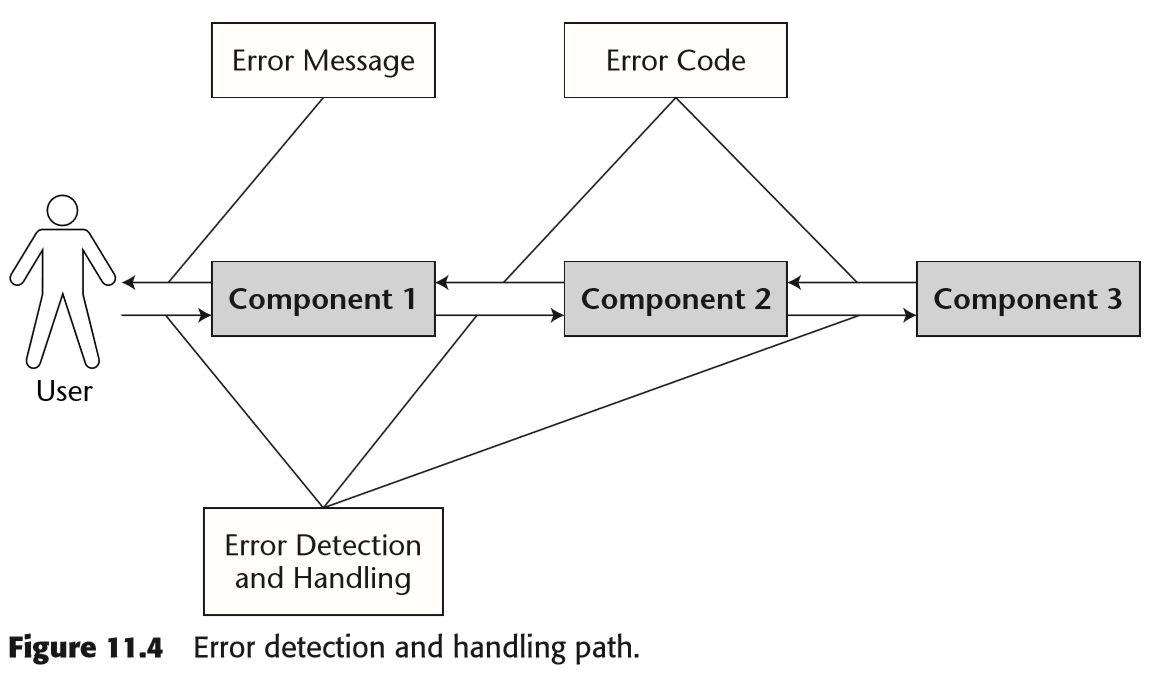
* Liệu các ứng dụng cung cấp đầy đủ việc bỏ qua lỗi và cho phép người sử dụng để phục hồi từ những sai lầm một cách duyên dáng? Ví dụ, nếu một trong các yếu tố đầu vào trong một hình thức 20 trường là không hợp lệ, không những ứng dụng cho phép người dùng nhập lại dữ liệu hợp lệ với một lĩnh vực, hay nó buộc người sử dụng để bắt đầu lại một lần nữa?
* Liệu bản thân ứng dụng xử lý các tình trạng lỗi một cách duyên dáng? Nếu chương trình chấm dứt bất thường (ví dụ, do một lỗi nghiêm trọng, thoát khỏi ứng dụng mà không cần phải thông qua các thói quen xuất cảnh bình thường), một loạt các hoạt động dọn dẹp có thể đã không được thực hiện. Ví dụ, một số tập tin có thể được mở lại, một số biến thể có dữ liệu không chính xác, hoặc các cơ sở dữ liệu là không ổn định.
* Có hệ thống phục hồi một cách duyên dáng? Liệu các hệ thống sụp đổ hoặc tiếp tục hoạt động không ổn định hoặc hạn chế các chức năng, hoặc bảo mật xâm nhập? Ví dụ, trong một hệ thống dựa trên Web, lỗi nghiêm trọng nên được dự đoán, kiểm tra, xử lý và do đó hệ thống treo có thể tránh được. Tuy nhiên, một hệ thống sụp đổ gây ra bởi một điều kiện lỗi có thể là một kết quả tốt hơn so với việc có hệ thống tiếp tục hoạt động nhưng tạo ra kết quả sai hoặc bị tổn hại an ninh.
* Hãy ghi nhớ rằng nếu một trong các máy chủ vật lý phải khởi động lại vì lý do nào, khi hệ thống được khởi động lại, nó có thể không phải tất cả các dịch vụ (ví dụ, SMTPservice, dịch vụ cơ sở dữ liệu, vv) sẽ khởi động lại thành công. Bạn nên kiểm tra để chắc chắn rằng các thường lệ khởi động lại những gì bạn mong đợi nó để làm.

1. Kiểm tra các thông báo lỗi. Làm như sau:

* Xác định xem một thông báo lỗi xuất hiện. Nếu một thông báo lỗi xuất hiện, sau đó phần này của bài kiểm tra được thông qua (tính chính xác của thông điệp là thứ yếu so với thực tế là một thông điệp đã được tạo ra).
* Phân tích tính chính xác của các thông báo lỗi. Liệu thông điệp cho người dùng những gì là sai? Người dùng có thể hiểu được nó? Do các hướng dẫn phù hợp với điều kiện lỗi? Người sử dụng hướng dẫn trợ giúp sẽ vượt thoát thành công các tình trạng lỗi? Bạn cũng cần phải tự hỏi mình nếu ứng dụng cung cấp các hướng dẫn đầy đủ để người dùng có thể tránh một điều kiện lỗi rõ ràng ở nơi đầu tiên.
* Lưu ý rằng các thông tin liên lạc không phải là vào một thông báo lỗi. Nó có thể là vào một phương tiện khác như một cue âm thanh hoặc hình ảnh. Sử dụng bản án của bạn để đánh giá đó vừa là thích hợp nhất, và nói lên mối quan tâm của bạn cho phù hợp.

1. Kiểm tra các vấn đề khác. Có phải bất kỳ vấn đề liên quan khác phát hiện trong việc tạo điều kiện lỗi này? Ví dụ, có thể có một rò rỉ bộ nhớ, một chồng tham nhũng, một cấu trúc dữ liệu hoặc bảng mục được cập nhật một phần, hoặc một con trỏ hoang dã. Acalculation có thể đã được hoàn thành chính xác. Bạn có thể sử dụng dữ liệu thu thập được từ các công cụ khai thác gỗ để phân tích tình trạng của các máy chủ, do đó phát hiện hàm ý liên quan đến thông báo lỗi.

Hình 11.4 bạn đi qua một ví dụ về một hệ thống Web để chứng minh quá trình xử lý lỗi. Đầu tiên, người sử dụng có hành động nào đó và đưa các hệ thống hoặc ứng dụng vào trạng thái lỗi. Điều này có thể được thực hiện theo nhiều cách khác nhau: đầu vào bàn phím không hợp lệ, ngắt kết nối từ một ISP, loại bỏ một cáp kết nối mạng, hoặc cố gắng theo đuổi một con đường không chính xác thông qua các ứng dụng.



Lỗi phải được xác định sau đó. Có thể là do code chứa một lỗi, nhưng nó sẽ vô dụng nếu như phần mềm không nhận ra được lỗi này. Như được đề cập đến trong ví dụ 11.4, lỗi này có thể được xác định ở bất cứ thành phần nào trong mắt xích giao tiếp mà có liên quan tới việc thực hiện yêu cầu. Liệu một thông điệp( tin nhắn) có được đưa tới người dùng không? Nếu không, có thể có một lỗi im lặng đúng nào đang xảy ra hay không? Có thể có lỗi im lặng sai nào đang xảy ra không? Phép kiểm tra nào phải được tạo ra để đưa ra kết quả về một lỗi phía sau người sử dụng mà người sử dụng không được khuyến cáo nhưng vẫn mắc phải?

Lỗi được xác định phải được xử lí sau đó. Liệu phần mềm sẽ tự sửa lỗi đó? Trong hầu hết các trường hợp, một số thành phần sẽ trả về một đoạn mã hay thông điệp lỗi. Các lỗi điều kiện thường được giao tiếp lại với người sử dụng trong một mẫu của thông điệp lỗi. Điều này có thể được giải quyết ngay từ phía người sử dụng ( nếu đó là thao tác được mong muốn/ chủ đích) hoặc từ phía máy chủ ( sau khi người dùng gửi yêu cầu tới máy chủ).

Ví dụ, người dùng nhập một yêu cầu và yêu cầu này được gửi tới máy chủ Web, nơi mà sau đó sẽ chuyển tới máy chủ cơ sở dữ liêu. Nếu cơ sở dữ liệu nhận ra một lỗi, sau đó nó sẽ gửi trả lại một đoạn mã lỗi. Máy chủ web không nhận ra được tin nhắn lỗi này được dựa trên đoạn mã lỗi hay nó chuyển tiếp từ đoạn mã lỗi trở lại người dùng. ( để biết thêm thông tin, xem chương 10: kiểm tra tương tác người dùng, ví dụ về việc xử lí tin nhắn lỗi từ người dùng và lỗi từ máy chủ ).

Trong kế hoạch xử lí lỗi này, nên cân nhắc các vấn đề sau:

Điều kiện xảy ra một lỗi có thể xảy ra bất cứ đâu trong mắt xích xử lí giao dịch.

Mỗi thành phần trong mắt xích có thể hỏng để từ đó xác định hoặc giải thích kiều kiện lỗi một cách chính xác

* Mỗi thành phần trong chuỗi thông tin liên lạc có thể thất bại để chuyển tiếp một mã lỗi cho phần tiếp theo.
* Một lỗi điều kiện (code) cần phải được dịch ra một thông điệp dễ hiểu để người dùng sẽ biết những gì đã xảy ra. Ví dụ, "Exception 80.065" sẽ không có nghĩa gì cho người sử dụng.
* Người dùng không biết hoặc quan tâm, nơi các lỗi xảy ra. Anh ta hoặc cô ta chỉ cần biết những lỗi là gì (từ quan điểm của con người) và làm thế nào để sửa chữa tình hình.
* Mục đích của orced-error tests là để đưa các ứng dụng vào một lỗi điều kiện. Mức đầu tiên của kiểm thử là để đảm bảo rằng một thông báo lỗi được nhận bởi các khách hàng. Mức độ thứ hai là để kiểm tra các thông báo lỗi là đúng; nghĩa là, nó cần truyền đạt các mô tả về lỗi và nói cho người dùng như thế nào để giải quyết lỗi.

Bước tiếp theo là kiểm tra xem tình trạng lỗi đã được xử lý đúng cách. Điều này có thể ngay lập tức rõ ràng, nhưng trong nhiều trường hợp, chúng ta có thể không biết điều gì đã dẫn đến từ các tình trạng lỗi. Một chìa khóa cho testers là ghi chép tốt của những hành động đã được thực hiện và theo thứ tự, để lỗi có thể được điều tra tốt hơn và tái tạo.

Như đã thảo luận trước đó, có rất nhiều cách để thu thập thông tin lỗi điều kiện. Cách có hệ thống nhất của việc tạo ra sự kiểm tra tình trạng lỗi sơ bộ là biên soạn một danh sách của tất cả các thông báo lỗi từ các nhà phát triển của chương trình. Lỗi điều kiện bổ sung có thể được phát hiện thông qua phỏng đoán dựa trên kinh nghiệm kiểm thử trong quá khứ. Bạn cũng có thể khám phá ra lỗi vô tình trong khi thực hiện các kiểu kiểm thử.

Một chu kỳ đầy đủ của FET có thể yêu cầu 2-3 builds để hoàn thành, bởi vì người kiểm tra phải thực hiện vô số các hoạt động lỗi phát sinh,để rằng, trong một thế giới lý tưởng, người dùng sẽ không bao giờ làm. Chỉ bằng cách tạo ra những tình huống không mong muốn (ví dụ, nhập các ký tự đặc biệt vào các lĩnh vực được thiết kế cho các chữ số và bằng cách yêu cầu thiết lập lề trang đó là không thể) bạn có thể tạo ra các lỗi điều kiện đó không liên quan đến các lỗi điều kiện được biết đến.

### **2.3.4 Boundary Condition Tests and Equivalent Class Analysis( Kiểm tra Điều kiện Biên và Các Lớp Tương đương)**

Kiểm tra điều kiện biên tương tự như kiểm thử FETs trong đó họ thử nghiệm ranh giới của từng biến

Ví dụ, khi kiểm thử một form đăng kí online, bạn cần kiểm tra một ô nhập (text field) chỉ được nhập từ 2 đến 7 kí tự, trên thực tế, hàm đã chạy như mong đợi. có ô nhập nhập được hai, ba, sáu, bảy kí tự? còn một kí tự, tám kí tự thì như thế nào? Ô nhập có thể được bỏ trống?

Kiểm tra điều kiện biên là một phần mở rộng của TOFTs và FETs, và chúng có những điểm tương tự nhau, hãy xem bảng 11.2 ở cuối chương này để có cái nhìn trực quan về sự khác nhau và giống nhau giữa TOFT, FET, kiểm tra điều kiện biên, và FAST

### **2.3.5 Exploratory Testing**

Thăm dò thử nghiệm\* là một quá trình của kiểm tra sản phẩm và quan sát cách thức của chúng, tốt nhất là đưa ra giả thuyết và cách thức của nó. Nó liên quan đến việc các test-case và tạo ra những cái mới, như những thông tin được thu thập từ kết quả thử nghiệm trước đây. Thực hiện kiểm tra bằng cách thêm vào môi trường, dữ liệu đầu vào để đưa vào chương trình, quan sát những kết quả từ chương trình, đánh giá những gì đã học được và sau đó bắt đầu các thử nghiệm tiếp theo.

Thử nghiệm thăm dò có nhiều tên gọi và có thể có nhiều hình thức. Nó cũng được gọi là *unstructured* hoặc *ad hoc testing* và là, trái với một số niềm tin, một chiến lược thử nghiệm phương pháp. Nó liên quan đến "suy nghĩ bên ngoài hộp," kiểm tra hành vi mà chúng ta có thể không mong đợi nhưng mà bất kỳ người dùng có thể, nhầm lẫn hay chủ ý, làm gì.

Bạn làm thử nghiệm thăm dò bằng cách đi thông qua các chương trình, tìm hiểu nó là gì, và thử nghiệm nó. Nó được gọi là thăm dò vì bạn khám phá. Theo James Bach, một tác giả và chuyên gia trong kiểm thử phần mềm, thử nghiệm thăm dò có nghĩa là học tập trong khi thử nghiệm. Nó là đối diện của prescripted thử nghiệm giống như chơi 20 câu hỏi. Nếu chúng ta phải xác định tất cả những câu hỏi của chúng tôi trước, các trò chơi sẽ là gần như không thể, bởi vì với mỗi câu hỏi, chúng tôi tìm hiểu thêm về những gì các thử nghiệm tiếp theo nên được. Các yếu tố của thử nghiệm thăm dò bao gồm:

* Thăm dò sản phẩm
* Kiểm tra thiết kế
* Thực hiện kiểm tra
* Heuristics
* Kết quả reviewable

Chú ý Để biết thêm về chủ đề này, đọc báo James Bach tại www.satisface.com. Cũng đi đến chỗ Brian Marick tại www.testingcraft.com/exploratory.html; và đọc báo Chris Agruss và Bob Johnson tại www.testingcraft.com/ ad\_hoc\_testing.pdf.

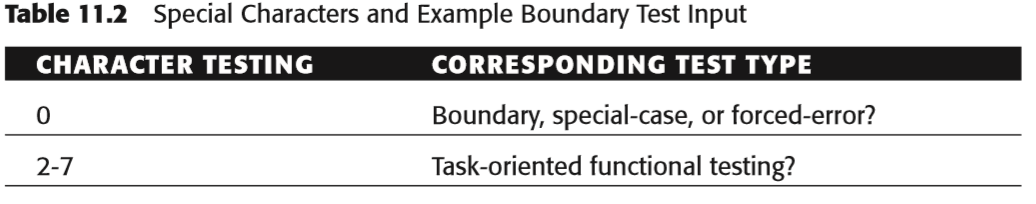
|  |
| --- |
| TESTING THE SAMPLE APPLICATION  Một ví dụ về một FET sử dụng project mẫu bao gồm việc yêu cầu một metric (hoặc biểu đồ) mà có ngày bắt đầu sau ngày kết thúc. Trong thực tế, kiểm tra trường hợp này yêu cầu những điều không thể, do đó, một biểu đồ như vậy có thể không tồn tại. Tuy nhiên, các project mẫu cho phép người sử dụng để yêu cầu một biểu đồ như vậy, do đó, các tình huống phải được kiểm tra.  Mục đích của kiểm tra trường hợp này không phải là để đảm bảo tính chính xác của đầu ra, nhưng để kiểm tra xem các chương trình đáp ứng các điều kiện lỗi. Có hệ thống sụp đổ? Sản phẩm sử dụng cho một cơ hội để sửa đổi theo yêu cầu của mình? Được một thông báo lỗi trình bày cho người sử dụng (không phụ thuộc vào độ chính xác của các thông báo lỗi)?  Biên soạn một danh sách của tất cả các thông báo lỗi là một phương tiện hiệu quả tạo ra các trường hợp kiểm thử cho FET. Hình 11.5 cho thấy một FET đơn giản với một ngày bắt đầu cho một biểu đồ tháng 31. Chúng ta đều biết "30 ngày có tháng Chín, do đó một đọc thông báo lỗi," Ngày bắt đầu Không tìm thấy hoặc không hợp lệ ", được trả về. |

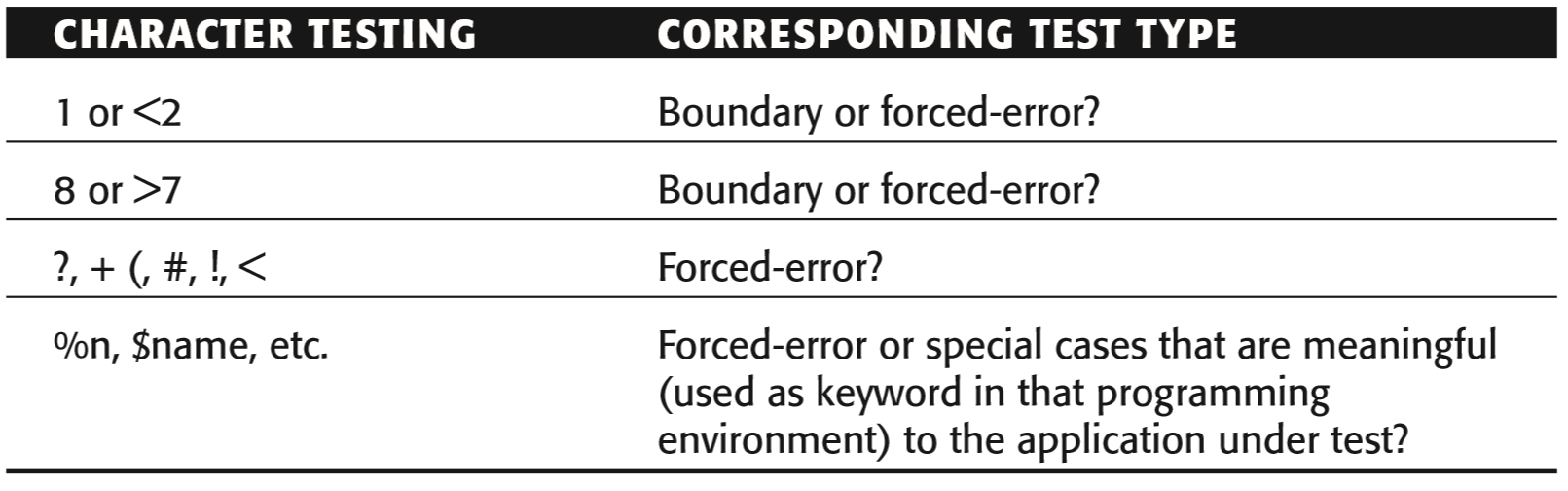
### **2.3.6 Which Method Is It?**

Định nghĩa của các loại bài kiểm tra hoặc các phương pháp, tự nó, không phải là quan trọng. Điều quan trọng là để đảm bảo rằng trường hợp kiểm thử liên quan (và xác suất của họ thất bại lộ) được sử dụng để thực hiện một tính năng tìm kiếm cho các lỗi.

|  |
| --- |
| TESTING THE SAMPLE APPLICATION  Sử dụng số liệu xu hướng mẫu project của tính năng như một ví dụ, kiểm tra ranh giới có thể được sử dụng để kiểm tra xem các chương trình đáp ứng một cách thích hợp để yêu cầu người dùng ở ranh giới của các tập dữ liệu. Ví dụ, chúng tôi thực hiện một tập hợp dữ liệu báo cáo dữ liệu bắt đầu vào ngày 1 tháng 4 năm 1999, và kết thúc vào ngày 30 tháng 4 năm 1999. Các giới hạn quy định của bộ số liệu này, do đó, 4/1/99 và 4/30/99. Bất kỳ yêu cầu người sử dụng mà xác định bắt đầu và ngày kết thúc giữa 4/1/99 và 4/30/99 không nên tạo ra lỗi. Bất kỳ yêu cầu người dùng mà chỉ rõ ngày bắt đầu trước khi 4/1/99 hoặc sau 4/30/99 nên tạo ra lỗi.  Trong ví dụ này, các bài kiểm tra ranh giới đòi hỏi ba giá trị cho mỗi đầu của các giới hạn: 3/31/99, 4/1/99, và 4/2/99 cho ngày bắt đầu; và 4/29/99, 4/30/99, và 5/1/99 cho ngày cuối cùng. Mỗi bài kiểm tra ranh giới phải xác nhận rằng dữ liệu xu hướng-metric chính xác được trả về cho những đề nghị chỉ định ngày bắt đầu và kết thúc giữa 4/1/99 và 4/30/99, và rằng các điều kiện lỗi được xử lý một cách duyên dáng cho những đề nghị chỉ rõ ngày bắt đầu và kết thúc trước khi 4/1/99 hoặc sau 4/30/99. Bảng 11.1 cho thấy sự kết hợp của các giá trị đó sẽ được kiểm thử trong kiểm tra ranh giới cho các số liệu xu hướng mẫu của project. |

Như minh họa trong Bảng 11.2, các đường ranh giới giữa các xét nghiệm, TOFTs, và FETs không phải luôn luôn rõ ràng. Sử dụng ví dụ trường văn bản được đưa ra trước đó, bạn có thể thực hiện bất kỳ của ba loại kiểm tra, tùy thuộc vào ký tự mà bạn nhập vào lĩnh vực này với hai đến giới hạn bảy ký tự. Nó không quan trọng mà phương pháp đó là. Điều quan trọng là một trường hợp kiểm thử sẽ có một xác suất cao của việc tìm kiếm lỗi.





## **Kết luận**

Kiểm thử phần mềm là một trong những hoạt động quan trọng trong quy trình phát triển phần mềm. Nó góp một phần rất lớn trong việc đánh giá chất lượng một phần mềm và là quy trình bắt buộc trong các dự án phần mềm trên thế giới cũng như trong nước. Tuy nhiên, hoạt động kiểm thử thường gặp nhiều khó khăn. Thứ nhất, kiểm thử các hệ thống phức tạp đòi hỏi rất nhiều nguồn tài nguyên và chi phí cao. Thứ hai, quy trình phát triển phần mềm luôn trải qua nhiều hoạt động biến đổi thông tin, sự mất mát thông tin trong quá trình biến đổi là yếu tố chính làm cho hoạt động kiểm thử khó khăn. Thứ ba, kiểm thử chưa được chú trọng trong đào tạo con người. Cuối cùng, không tồn tại kỹ thuật kiểm thử cho phép khẳng định một phần mềm hoàn toàn đúng đắn hay không còn lỗi.

Tóm lại, chương này tập trung tìm hiểu quy trình kiểm thử phần mềm với công cụ kiểm thử Functional Tests, các chức năng kiểm thử của công cụ này, ứng dụng để kiểm thử phần mềm. Kết quả thực hiện kiểm thử được thể hiện qua báo cáo kiểm thử và báo cáo ghi nhận lỗi phát sinh để người phát triển phân tích và chỉnh sửa các lỗi, đảm bảo được chất lượng của phần mềm. Kiểm thử tự động mang lại hiệu quả, sự tiện lợi trong kiểm thử và là hướng phát triển của kiểm thử phần mềm/webservice trong tương lai.

## **Hướng phát triển**

* Nghiên cứu sâu hơn kiểm thử hướng dịch vụ (SOA) và kiểm thử bảo mật.
* Tìm hiểu và xây dựng nhiều thư viện liên kết để việc kiểm thử đạt hiệu quả cao hơn, tiện dụng hơn.
* Nghiên cứu khả năng bắt lỗi chính xác và tự động sửa những lỗi đơn giản của các công cụ.

# Chương III: Using Scripts to Test

## **Đặt vấn đề**

Khi đọc xong phần này bạn sẽ hiểu được Scripting Languages là gì. Biết được tại sao lại sử dụng Scripting Languages trong quá trình test mà không sử dụng compiled language. Các công dụng của nó trong quá trình kiểm thử và làm thế nào để chuẩn bị được một bảng scripting tốt cho quá trình test.

## **Nội dung chính**

### **3.2.1 Giới thiệu**

* Test Script là một nhóm mã lệnh dạng đặc tả dạng Script dùng để tự động hóa một trình tự kiểm tra, giúp cho việc kiểm tra nhanh hơn, hoặc cho những trường hợp mà kiểm tra bằng tay sẽ rất khó khăn hoặc không khả thi.
* Test Script giúp chúng ta có thể bắt đầu test trước khi hoàn thành UI, hoặc truy cập vào trực tiếp vào máy chủ của ứng dụng một cánh đơn giản…

### **Scripting Languages**

#### **3.2.2.1. Scripting Languages là gì?**

- Là để chỉ các ngôn ngữ lập trình dùng để tạo các chương trình chạy "on-the-fly". Tức là để chạy các chương trình viết bằng scripting language thì chỉ cần có file source code của chương trình đó rồi dùng runner để thực thi cái source code đó. Nhờ có source code, người chạy chương trình viết bằng scripting language có thể chỉnh sửa mã nguồn theo ý của mình một cách đơn giản nhất, nên việc testing và chỉnh sửa nó tương đối đơn giản, được sử dụng khá tốt cho việc automation test.

- Scripting language: Gồm DOS batch của windows hay bash của Unix hay Perl(.pl) của Unix đều là scripting language.

- DOS batch files được thiết kế từ các chuỗi lệnh với nhau. Mặc dù nó có thể trở lên lớn và phức tạp hơn, nhưng nó lại giới hạn về khả năng thao tác dữ liệu và xử lý logic Vì thế ta cần có một ngôn ngữ Scrip mạnh mẽ hơn, cung cấp nhiều tính năng hơn, kiểm soát tốt hơn và linh hoạt hơn để thực thiện nhiệm vụ phức tạp hơn. Đến hôm nay thì có rất nhiều các ngôn ngữ kịch bản được phát triển và được sử dụng phổ biến như là: Perl, VBScript, Tcl, JavaScript, awk, sed, and Python, among others

#### **Why Not Just Use a Compiled Program Language?**

- Khác với scripting language, compiled language đòi hỏi source code phải được biên dịch thành mã máy. Do không có mã nguồn, nên người chạy chương trình viết bằng compiled language không thể tùy chỉnh chương trình theo ý của mình và việc automation test rất khó khăng

- Các ngôn ngữ C, C ++, Java, và Visual Basic, điều thực hiện nhiều nhiệm vụ nặng và thường có thể là quá mức cần thiết cho các công việc test

- Vd: Compiled language: C++,C biên dịch ra exe.

#### **Application of Scripting to Testing Tasks**

##### **Sử dụng chúng cho các nhiệm vụ quản trị hệ thống**

- Nó hay được dùng để bảo trì các máy tính hoặc các môi trường thử nghiệm. Việc học các dòng lệnh và viết các kịch bản đơn giản có thể giúp bạn tiết kiệm thời gian để làm tasks và có thể giúp bạn để đảm bảo các tasks được thực thi cùng một cách mỗi khi Scrip được chạy

- Một tình trạng hay gặp mà bạn sẽ phải đối mặt là việc thiết lập môi trường thử nghiệm trước khi test.

- Vd:

#!/bin/csh

# clean-up before testing product

cd ~tester/product

rm \*.log

rm product.rc

mv /test-cases/product.rc .

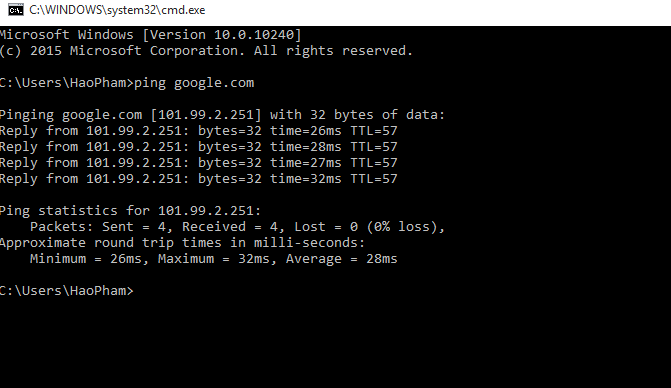
- Đối với ví dụ này, chúng ta sẽ viết một scrip bốn dòng để hoàn thành task. Trước khi một thử nghiệm được chạy, bạn có thể cần phải xóa các tập tin log cũ, sao chép các tập tin cấu hình để các địa điểm chính xác, và tải về phiên bản mới nhất. tất cả những nhiệm vụ này có thể được thực hiện với một kịch bản đơn giản, như ở trên.

##### **Khám phá thông tin về hệ thống**

- Có rất nhiều thông tin bạn có thể tìm hiểu về hệ thống mạng của bạn bằng cách sử dụng dòng lệnh. Từ phát hiện bao nhiêu bộ nhớ có sẵn để là người đăng nhập vào mạng tại bất kỳ thời điểm nào và đơn giản hơn để vào các hoạt động của mạng

- Ví dụ, bạn có thể thực hiện kiểm tra và chạy vào thông báo lỗi đọc "máy chủ không tìm thấy" hay "không thể xác định vị trí máy chủ." Các máy chủ có thể được kiểm tra được ẩn, được duy trì, hoặc nhận được cài đặt mới, hoặc bạn có thể có URL sai?

- Vd: Các lệnh ping có thể cho bạn biết tình trạng của một máy tính từ xa trên mạng:



- Trong trường hợp này, google.com, đang chạy và cho bạn biết mạng đang hoạt đông và một số thông tin.

##### **Kiểm tra các máy chủ trực tiếp**

- Testing Web applications trên dao diện client thì còn hạn chế nhưng đối với testing trên server-side thì có thể bổ sung cho việc testing trên client qua việc sử dụng các kịch bản, bằng cách sử dụng kịch bản phía máy chủ có thể truy cập trực tiếp

##### **Làm các yêu cầu hệ thống bỏ qua các giao diện người dùng**

- Một trong những vấn đề bắt đầu test ban đầu là chúng ta có thể phải chờ cho đến khi giao diện người dùng là đủ chức năng mới kiểm tra. Một cách giải quyết là để bỏ qua nó luôn

- Sử dụng ngôn ngữ kịch bản như VBScript hay Perl, chúng ta có thể mở rộng bộ công cụ thử nghiệm với khả năng mạnh mẽ, chẳng hạn như thực hiện cuộc gọi chức năng, do đó xây dựng các giao diện riêng của chúng tôi để phần mềm của máy chủ.

##### **Kiểm tra các bản ghi**

- Bản ghi được tạo bởi hệ thống và các ứng dụng thử là tuyệt vời nguồn tin cho việc phát hiện và phân tích lỗi. Hệ điều hành, máy chủ e-mail, các máy chủ cơ sở dữ liệu, và các máy chủ Web tất cả tạo ra các tập tin log.

- Nhưng lưu tất cả các thông tin này có thể làm cho các bản ghi rất dài và khó đọc. Thay vào đó, bạn có thể tạo ra các scrip để tìm kiếm các tập tin log cho các chuỗi văn bản cụ thể

- Vd:

**#!/bin/csh**

**# check product log for errors and warnings**

**if ( $#argv == 0 ) then**

**echo “usage: program-name logfile-name”**

**exit 1**

**endif**

**set file=$argv[1]**

**date**

**grep –ins ‘ error ‘ $file**

**grep –ins ‘ warning ‘ $file**

**grep –ins ‘ stack trace ‘ $file**

- Ở đây lệnh grep được sử dụng để trích xuất tất cả các lỗi, cảnh báo, và đống dấu vết các mục bản ghi từ nhật ký và đặt chúng vào một tập tin được phân tích.

##### **Dùng scrip để hiểu kết quả test**

- Việc xác định khi một thử nghiệm cho kết quả dự kiến có thể khó khăn. Thông thường, chạy thử sẽ tạo ra số lượng lớn dữ liệu được đọc, so sánh với các kết quả mong đợi, và hiểu bởi tester. Sử dụng kịch bản, chúng ta có thể trích xuất các dữ liệu có liên quan từ đầu ra của một thử nghiệm và làm ra báo cáo dễ hiểu.

##### **Dùng scrip để năng cao năng suất**

A Script to Test Many Files

set file = `find . -name ‘\*.html’ -print`

grep -ins ‘copyright’ $file A Set of Scripts That Run Many Times (tái sử dụng, hoặc dùng thêm vào dòng lặp)

Executing Tests That Cannot Be Run Manually

#!/bin/csh

@ counter = 0

while (counter <= 1000)

@ counter++

test\_001.csh

test\_002.csh

end

chk\_log.csh

### **Scripting Good Practice**

1. **Chuẩn hóa định dạng tiêu đề và tài liệu hướng dẫn cho tất cả các script**

**Chuẩn hóa theo:**

* + Author
  + Date
  + Purpose
  + Parameters
  + Valid input/expected output
  + Other data or files needed

1. **Thiết lập hướng dẫn sử lý lỗi**
2. **Giữ cho size script luôn luôn ở giá trị nhỏ nhất**
3. **Mô tả các chức năng cụ thể**

* Có danh sách rỏ ràng và mục tiêu nhất định
* Viết chức năng chung cho việc tái sử dụng

1. **Cấu trúc script phải luôn dể đọc, dễ hiểu và dễ bảo trì**
2. **Hãy chú ý đến định dạng kết quả. Đầu ra kịch bản hoặc các bản ghi có thể đọc được và có thể hiểu được**

## **Kết luận**

Như vậy việc sử dụng script trong quá trình test nó đạt được nhiều công dụng khá hữu ích bao gồm kiểm tra hệ thống, quản trị hệ thống, kiểm tra các bản ghi.., và mục tiêu quan trọng nhất là để thực thi hóa việc automation test, giúp cho việc test với số lượng lớn được thuận tiện hơn, qua đó góp phần giảm chi phí và thời gian và năng cao năng xuất.

# Chương IV: Demo

## **4.1 Giới thiệu phần mềm**

Phần mềm quản lý siêu thị điện thoại di động là phần mềm hỗ trợ nghiệp vụ mua bán hàng trong một cửa hàng vừa và nhỏ với quy mô khoảng chục nhân viên.

Phần mềm có các nghiệp vụ chức năng:

* Nghiệp vụ bán hàng: Giúp nhân viên bán hàng dễ dàng lập và in hóa đơn khi khách hàng có nhu càu mua hàng.
* Nghiệp vụ nhập hàng vào kho: Giúp thủ kho dễ dàng cập nhật mặt hàng cũng như số lượng mặt hàng vào kho.
* Nghiệp vụ kiểm kê hàng trong kho: Kiểm kê số lượng hàng tồn trong kho để từ đó hoạc định chiến lược kinh doanh cũng như nhập hàng mới
* Nghiệp vụ quản lý nhân viên, khách hàng, giới tính, ...
* Nghiệp vụ thống kê báo cáo: Báo cáo doanh thu, doanh số ….theo tháng, quý.

Các nghiệp vụ phi chức năng:

* Yêu cầu bảo mật: mỗi nhân viên sẽ được cấp tài khoản và mật khảu để đăng nhập vào hệ thống. Từ đó hạn chế các quyền mà mỗi nhân viên nhận được.
* Yêu cầu sao lưu: Khi người quản trị cần thiết thì có thể sao lưu hay hồi phục lại cơ sở dữ liệu đảm bảo hệ thống vận hành an toàn
* Danh sách chức năng để test:
* Chức năng đăng nhập.
* Chức năng quản lý nhân viên.
* Chức năng nhập hàng.
* Chức năng quản lý giới tính

## **4.2. Unit Test & Integration Test**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên Class** | **Số function** | **Số lượng Unit Test** | **Số lượng Integration Test** |
| 1 | MatHang (DTO) | 10 | 6 |  |
| 2 | PhieuNhap (DTO) | 3 | 9 |  |
| 3 | PhieuNhapDAO | 3 | 9 | 2 |
| 4 | Class NhanVien (DTO) | 13 | 9 |  |
| 5 | Class NhanVienBll | 5 | 8 | 8 |
| 6 | Class TTDangNhap (DTO) | 3 | 3 |  |
| 7 | Class DangNhapBll | 1 | 3 | 7 |
| 8 | Class GioiTinh (DTO) | 4 | 3 |  |
| 9 | Class GioiTinhBll | 5 | 8 | 6 |
|  | Tổng cộng | 47 | 58 | 14 |

### **4.2.1 Unit Test**

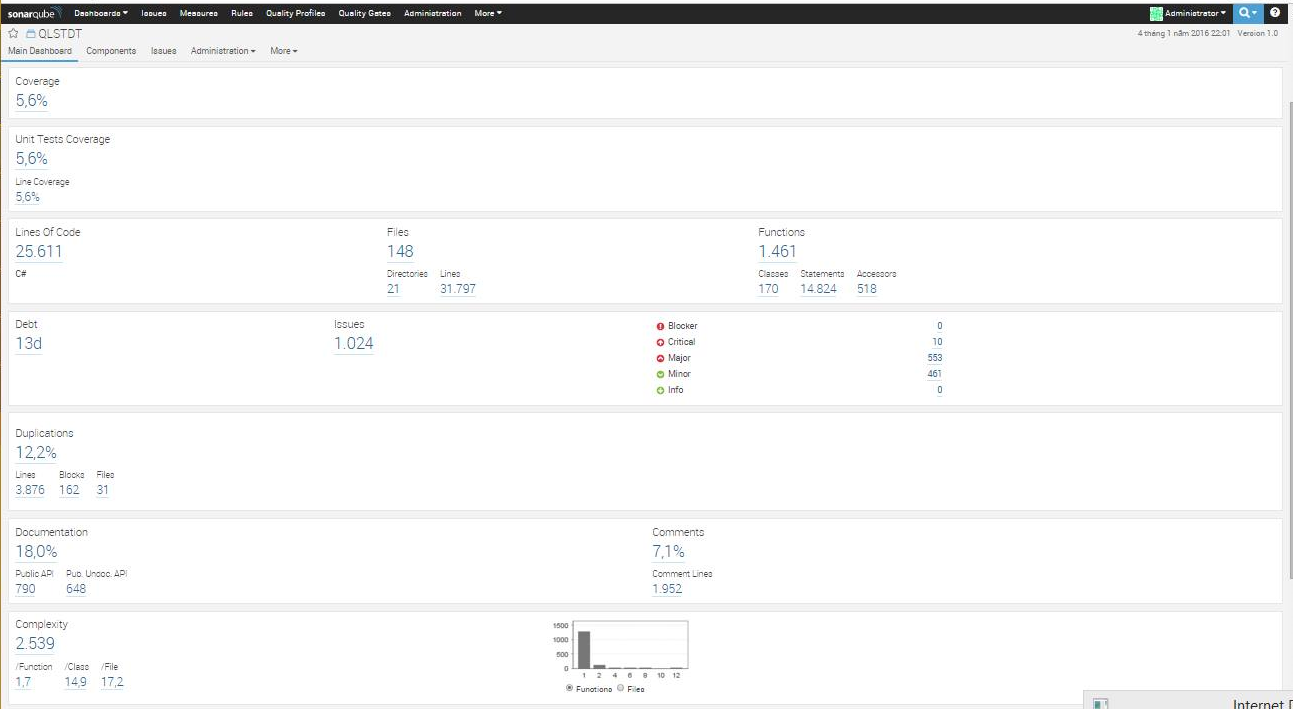
|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên Unit Test** | **Mô tả** | **Ghi chú** |
| 1 | Test\_MaMatHang\_1 | Test hàm get set mã mặt hàng | Class MatHangTest |
| 2 | Test\_MaMatHang\_2 | Test hàm get set mã mặt hàng | Class MatHangTest |
| 3 | Test\_MaMatHang\_3 | Test hàm get set mã mặt hàng | Class MatHangTest |
| 4 | Test\_TenMatHang\_1 | Test hàm get set tên mặt hàng | Class MatHangTest |
| 5 | Test\_TenMatHang\_2 | Test hàm get set tên mặt hàng | Class MatHangTest |
| 6 | Test\_TenMatHang\_3 | Test hàm get set tên mặt hàng | Class MatHangTest |
| 7 | Test\_maNhanVien\_1 | Test hàm get set mã nhân viên | Class PhieuNhapTest |
| 8 | Test\_maNhanVien\_2 | Test hàm get set mã nhân viên | Class PhieuNhapTest |
| 9 | Test\_maNhanVien\_3 | Test hàm get set mã nhân viên | Class PhieuNhapTest |
| 10 | Test\_maPhieuNhap\_1 | Test hàm get set mã phiếu nhập | Class PhieuNhapTest |
| 11 | Test\_maPhieuNhap\_2 | Test hàm get set mã phiếu nhập | Class PhieuNhapTest |
| 12 | Test\_maPhieuNhap\_3 | Test hàm get set mã phiếu nhập | Class PhieuNhapTest |
| 13 | Test\_ngayHoaDon\_1 | Test hàm get set ngày hoá đơn | Class PhieuNhapTest |
| 14 | Test\_ngayHoaDon\_2 | Test hàm get set ngày hoá đơn | Class PhieuNhapTest |
| 15 | Test\_ngayHoaDon\_3 | Test hàm get set ngày hoá đơn | Class PhieuNhapTest |
| 16 | Test\_ThemPN\_1 | Test hàm thêm phiếu nhập đúng | Class PhieuNhapDaoTest |
| 17 | Test\_ThemPN\_2 | Test hàm thêm phiếu nhập sai |  |
| 18 | Test\_ThemPN\_3 | Test hàm thêm phiếu nhập sai | Class PhieuNhapDaoTest |
| 19 | Test\_ThemCTPN\_1 | Test hàm thêm chi tiết phiếu nhập đúng | Class PhieuNhapDaoTest |
| 20 | Test\_ThemCTPN\_2 | Test hàm thêm chi tiết phiếu nhập sai | Class PhieuNhapDaoTest |
| 21 | Test\_ThemCTPN\_3 | Test hàm thêm chi tiết phiếu nhập sai | Class PhieuNhapDaoTest |
| 22 | Test\_LayMaPhieuNhapMax\_1 | Test hàm lấy mã phiếu nhập đúng | Class PhieuNhapDaoTest |
| 23 | Test\_LayMaPhieuNhapMax\_2 | Test hàm lấy mã phiếu nhập sai | Class PhieuNhapDaoTest |
| 24 | Test\_LayMaPhieuNhapMax\_3 | Test hàm lấy mã phiếu nhập sai | Class PhieuNhapDaoTest |
| 25 | Test\_maNhanVien | Test hàm get set mã nhân viên | Class EnployeeTest |
| 26 | Test\_tenNhanVien | Test hàm get set tên nhân viên | Class EnployeeTest |
| 27 | Test\_NgaySinh | Test hàm get set ngày sinh của nhân viên | Class EnployeeTest |
| 28 | Test\_gioiTinh | Test hàm get set giới tính của nhân viên | Class EnployeeTest |
| 29 | Test\_CMND | Test hàm get set CMND của nhân viên | Class EnployeeTest |
| 30 | Test\_DiaChi | Test hàm get set địa chỉ của nhân viên | Class EnployeeTest |
| 31 | Test\_SDT | Test hàm get set số điện thoại của nhân viên | Class EnployeeTest |
| 32 | Test\_phongBan | Test hàm get set phòng ban của nhân viên | Class EnployeeTest |
| 33 | Test\_tenDangNhap | Test hàm get set tên đăng nhập của nhân viên | Class EnployeeTest |
| 34 | Test\_SearchNhanVien | Test hàm search nhân viên | Class EmpoyeeTest |
| 35 | Test\_ThemNhanVien | Test hàm thêm nhân viên với các trường dữ liệu đúng | Class EnployeeTest |
| 36 | Test\_ThemNhanVienSai | Test hàm thêm nhân viên khi để trống các trường dữ liệu | Class EnployeeTest |
| 37 | Test\_ThemNhanVienSai2 | Test hàm thêm nhân viên khi trùng mã nhân viên | Class EnployeeTest |
| 38 | Test\_SuaNhanVien | Test hàm sửa nhân viên với các trường dữ liệu đúng | Class EnployeeTest |
| 39 | Test\_SuaNhanVienSai | Test hàm sửa nhân viên với các trường dữ liệu sai | Class EnployeeTest |
| 40 | Test\_SuaNhanVienSai | Test hàm sửa nhân viên với các trường dữ liệu để trống | Class EnployeeTest |
| 41 | Test\_XoaNhanVien | Test hàm xoá nhân viên thành công | Class EnployeeTest |
| 42 | Test\_MaNhanVien | Test hàm get set mã nhân viên | Class LoginTest |
| 43 | Test\_TenNhanVien | Test hàm get set tên nhân viên | Class LoginTest |
| 44 | Test\_PhongBan | Test hàm get set phòng ban | Class LoginTest |
| 45 | Test\_DangNhap | Test hàm đăng nhập với tài khoản và mật khẩu đúng | Class LoginTest |
| 46 | Test\_DangNhapSai | Test hàm đăng nhập khi nhập sai tài khoản hoặc mật khẩu | Class LoginTest |
| 47 | Test\_DangNhapSai2 | Test hàm đăng nhập khi để trống tài khoản và mật khẩu | Class LoginTest |
| 48 | Test\_MaGioiTinh | Test hàm get set mã giới tính | Class GenderTest |
| 49 | Test\_TenGioiTinh | Test hàm get set tên giới tính | Class GenderTest |
| 50 | Test\_HamKhoiTaoGioiTinh | Test hàm khởi tạo giới tính | Class GenderTest |
| 51 | Test\_ThemGioiTinh | Test hàm thêm giới tính với các trường dữ liệu nhập vào đúng | Class GenderTest |
| 52 | Test\_ThemGioiTinhSai | Test hàm thêm giới tính khi thêm trùng khoá chính | Class GenderTest |
| 53 | Test\_ThemGioiTinhSai2 | Test hàm thêm giới tính khi để trống trường tên giới tính | Class GenderTest |
| 54 | Test\_UpdateGioiTinh | Test hàm update giới tính với các trường dữ liệu đúng | Class GenderTest |
| 55 | Test\_UpdateGioiTinhSai | Test hàm update giới tính khi bỏ trống trường tên giới tính | Class GenderTest |
| 56 | Test\_DeleteGioiTinh | Test hàm xoá giới tính thành công | Class GenderTest |
| 57 | Test\_SelectAllGioiTinh | Test hàm lấy danh sách giới tính | Class GenderTest |
| 58 | Test\_SelectTenGioiTinh | Test hàm lấy danh sách tên giới tính | Class GenderTest |

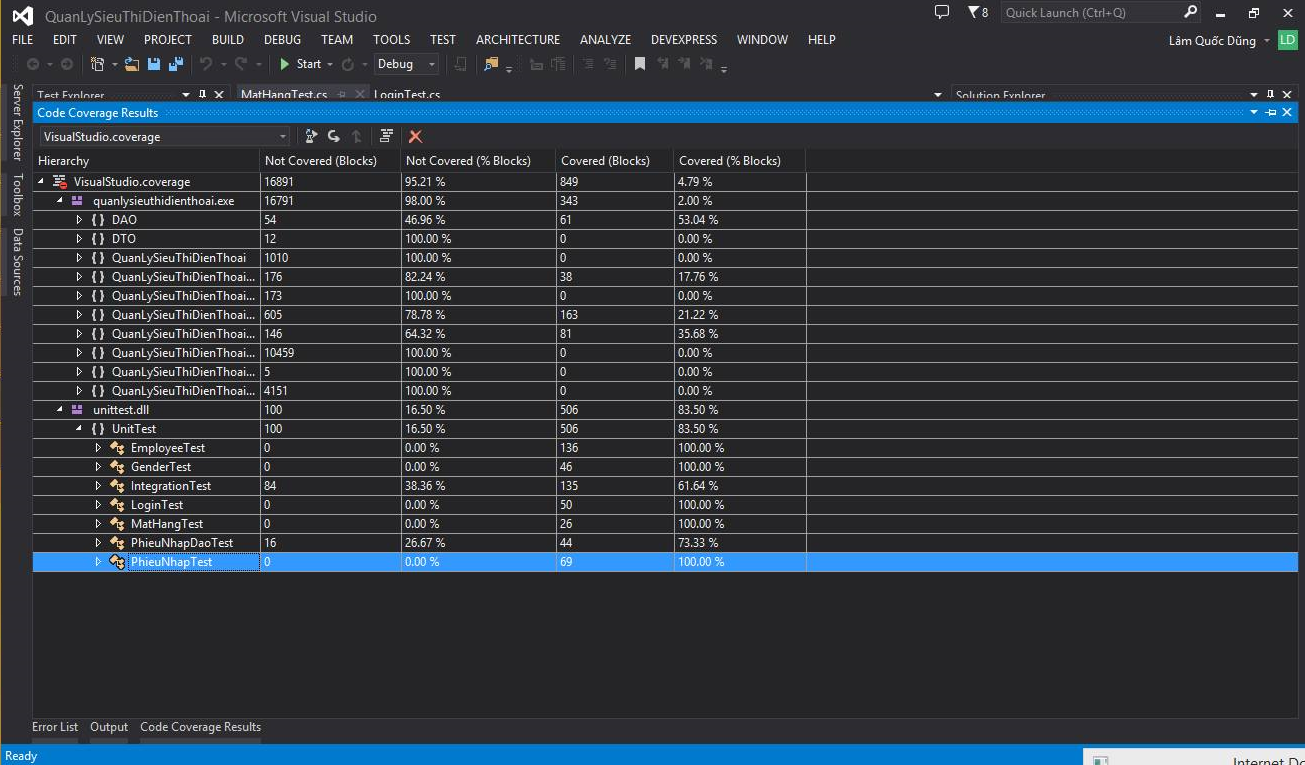
### **4.2.2 Integration Test**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên Unit Test** | **Mô tả** | **Ghi chú** |
| 1 | Test\_ThemSuaNhanVien | Test hàm thêm nhân viên xong sửa nhân viên | Class IntegrationTest |
| 2 | Test\_ThemXoaNhanVien | Test hàm thêm nhân viên xong xoá nhân viên | Class IntegrationTest |
| 3 | Test\_ThemNhanVienDangNhap | Test hàm thêm nhân viên xong đăng nhập | Class IntegrationTest |
| 4 | Test\_DangNhapThemNhanVien | Test hàm đăng nhập xong thêm nhân viên | Class IntegrationTest |
| 5 | Test\_DangNhapSuaNhanVien | Test hàm đăng nhập xong sửa nhân viên | Class IntegrationTest |
| 6 | Test\_DangNhapXoaNhanVien | Test hàm đăng nhập xong xoá nhân viên | Class IntegrationTest |
| 7 | Test\_DangNhapThemPhieuNhap | Test hàm đăng nhập xong thêm phiếu nhập | Class IntegrationTest |
| 8 | Test\_ThemNhanVienThemPhieuNhap | Test hàm thêm nhân viên xong thêm phiếu nhập | Class IntegrationTest |
| 9 | Test\_ThemSuaGioiTinh | Test hàm thêm giới tính xong sửa giới tính | Class IntegrationTest |
| 10 | Test\_ThemXoaGioiTinh | Test hàm thêm giới tính xong xoá giới tính | Class IntegrationTest |
| 11 | Test\_ThemGioiTinhThemNhanVien | Test hàm thêm giới tính xong thêm nhân viên | Class IntegrationTest |
| 12 | Test\_DangNhapThemGioiTinh | Test hàm đăng nhập xong thêm giới tính | Class IntegrationTest |
| 13 | Test\_DangNhapSuaGioiTinh | Test hàm đăng nhập xong sửa giới tính | Class IntegrationTest |
| 14 | Test\_DangNhapXoaGioiTinh | Test hàm đăng nhập xong xoá giới tính | Class IntegrationTest |

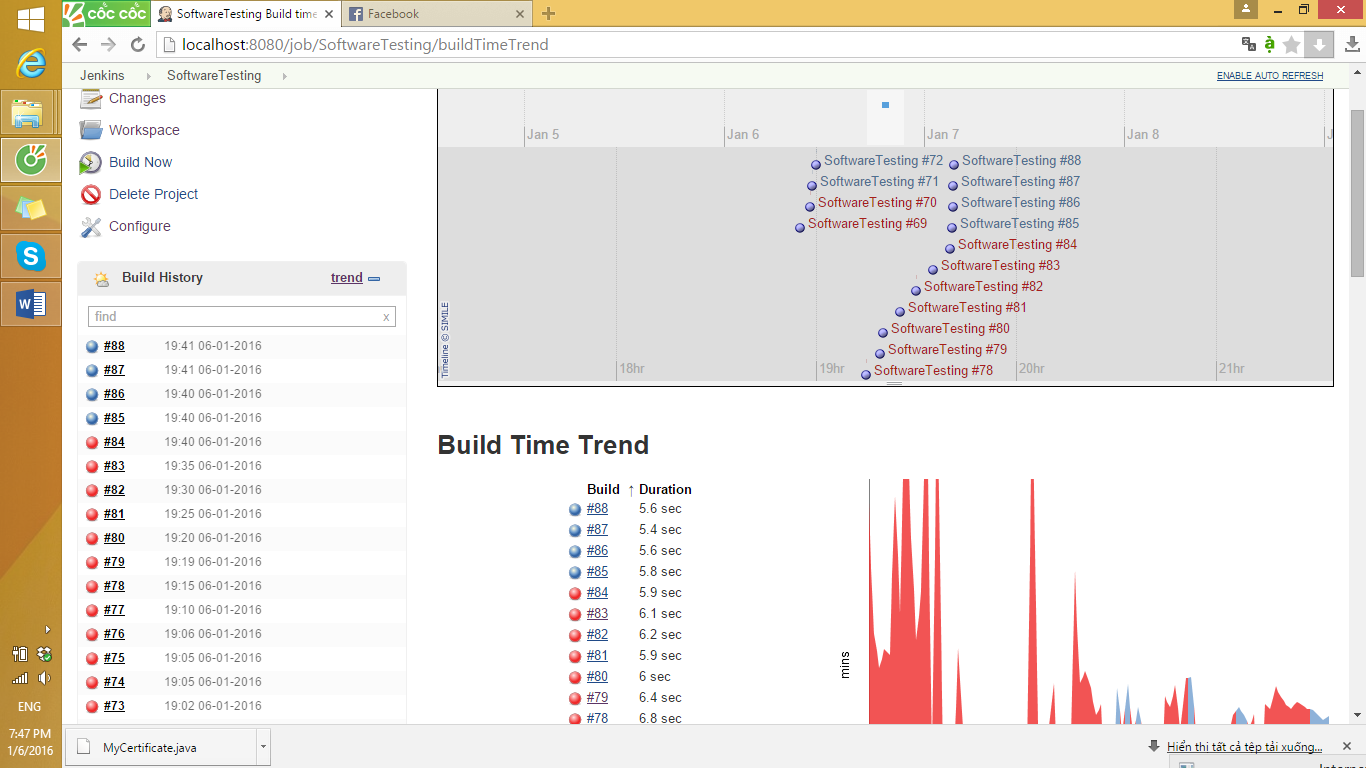
## **4.3 Auto Deploy & Check Unit Test Coverage**

* Màn hình check unit test coverage sử dụng tools sonarqube: (Xem phần hướng dẫn cài đặt để deploy)



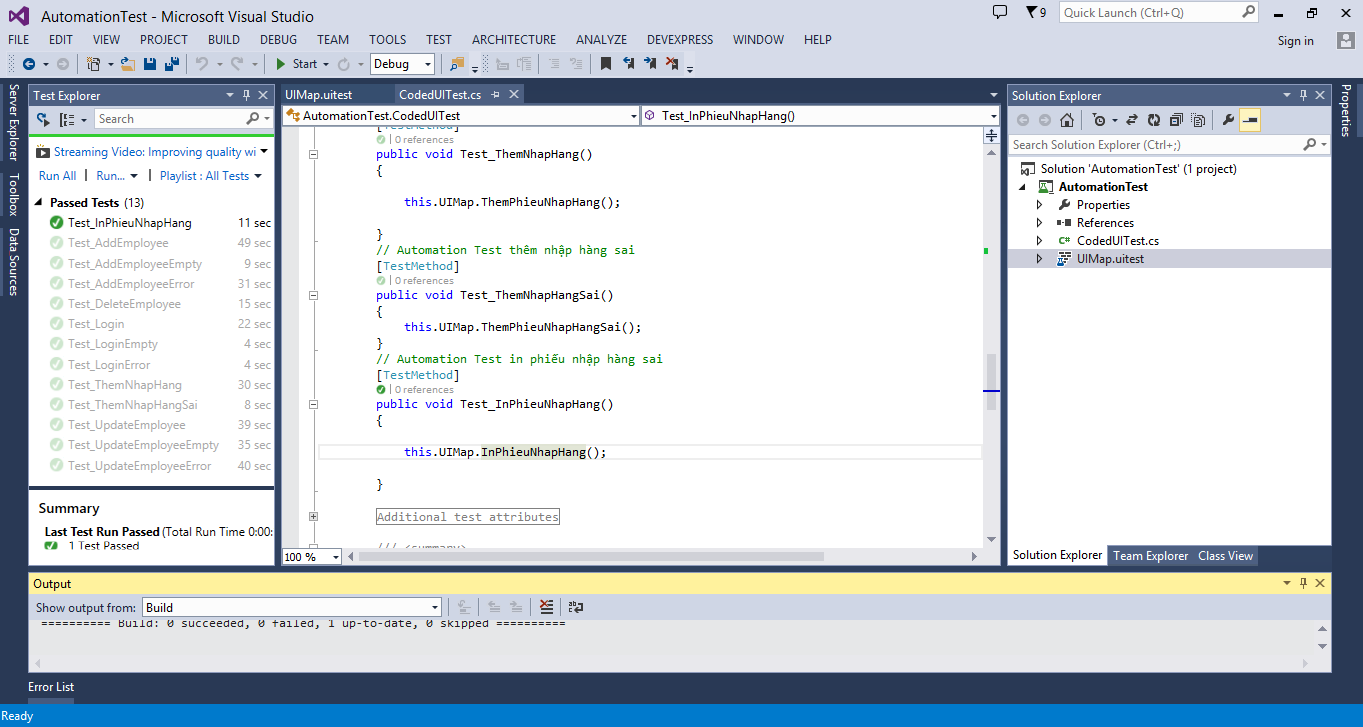


* Automation build with Jenkins: (Xem phần hướng dẫn cài đặt để deploy)



## **4.4 Automation Test**

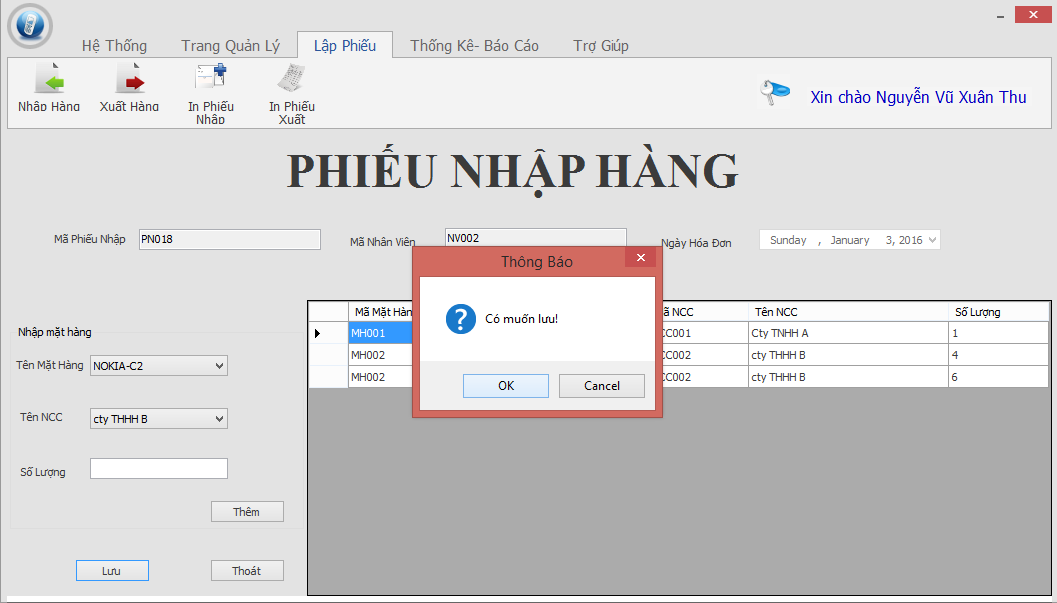
* Danh sách test case automation test:



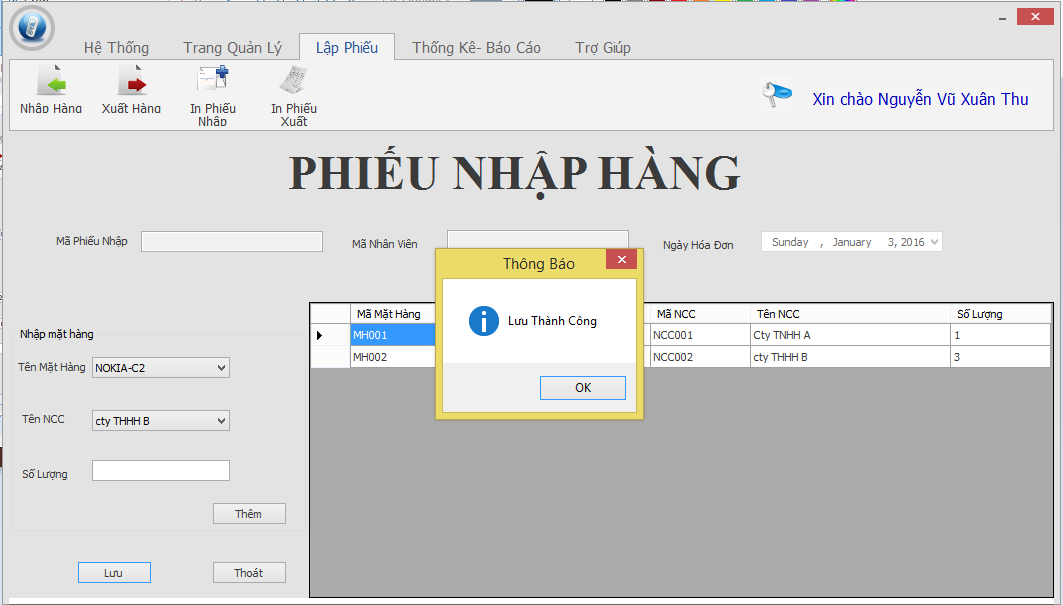
### **Chức năng lập phiếu nhập hàng**

#### **Thêm phiếu nhập hàng**

- Thêm phiếu nhập hàng đúng:

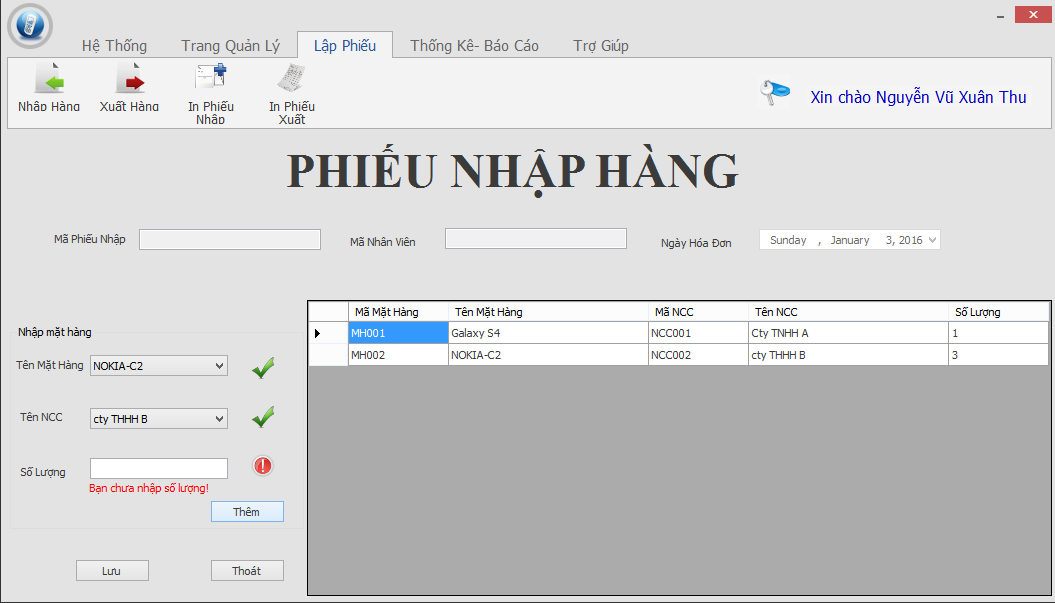


Xác nhận lưu phiếu nhập hàng



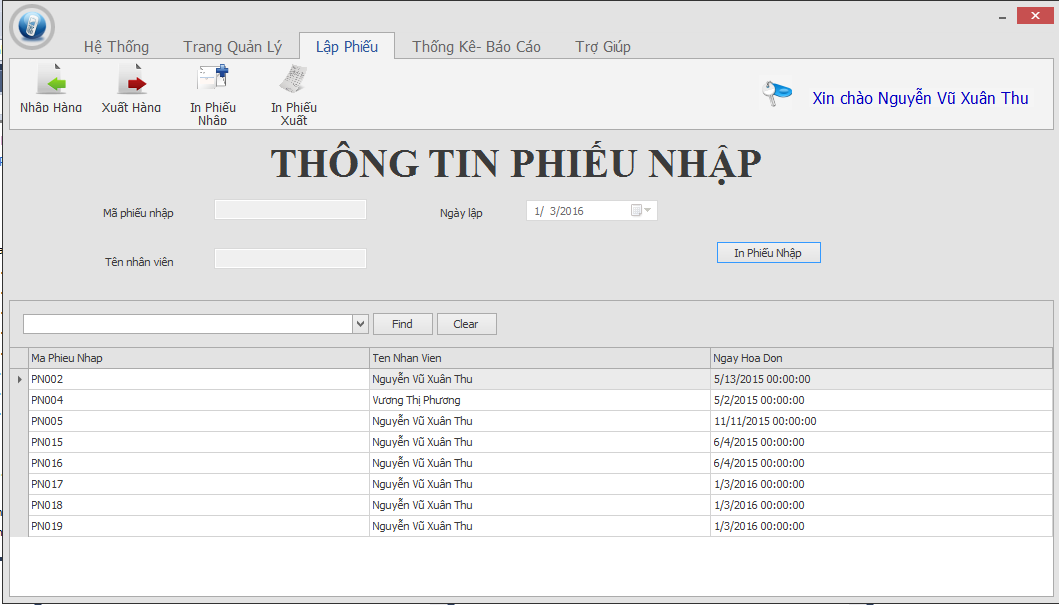
Lưu thành công

1. Thêm giới tính sai:

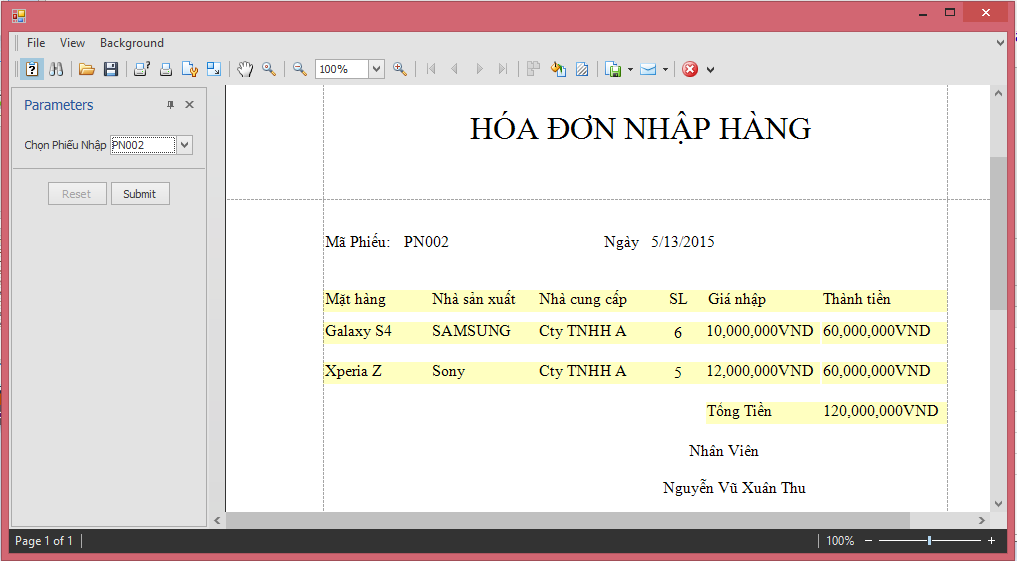


#### **4.4.1.2 In phiếu nhập hàng**

1. In phiếu nhập đúng:



Danh sách phiếu nhập hàng

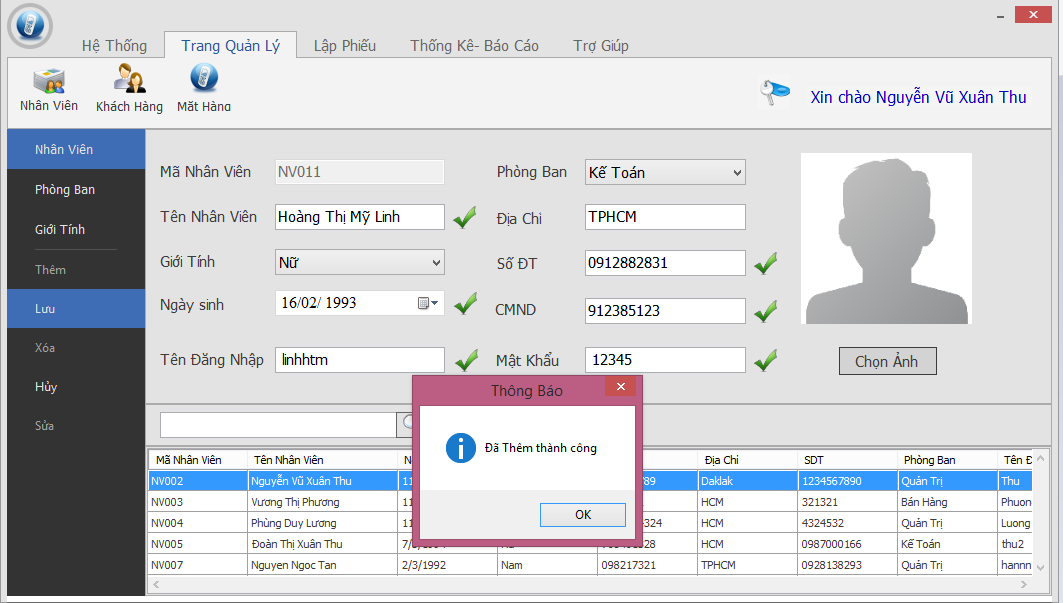


Màn hình xem phiếu in

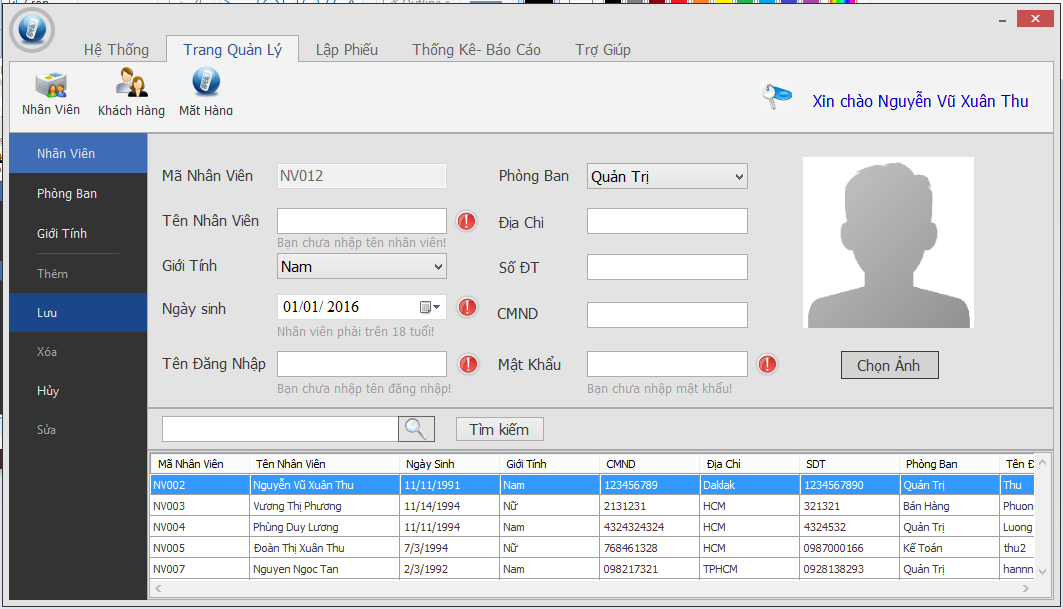
### **4.4.2 Chức năng quản lý nhân viên**

#### **4..4.2.1 Thêm nhân viên**

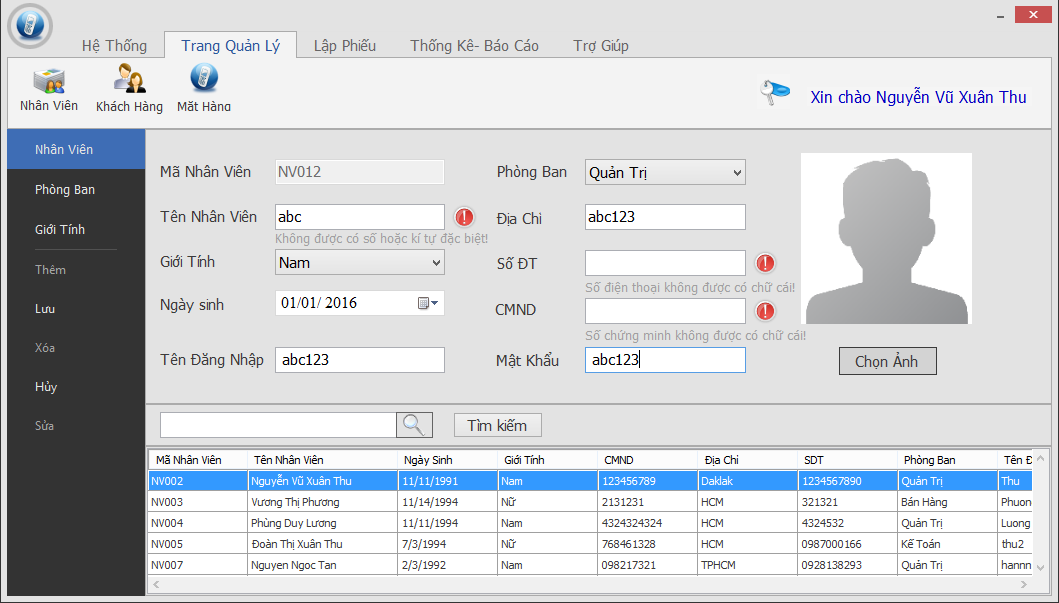
1. Thêm nhân viên đúng:



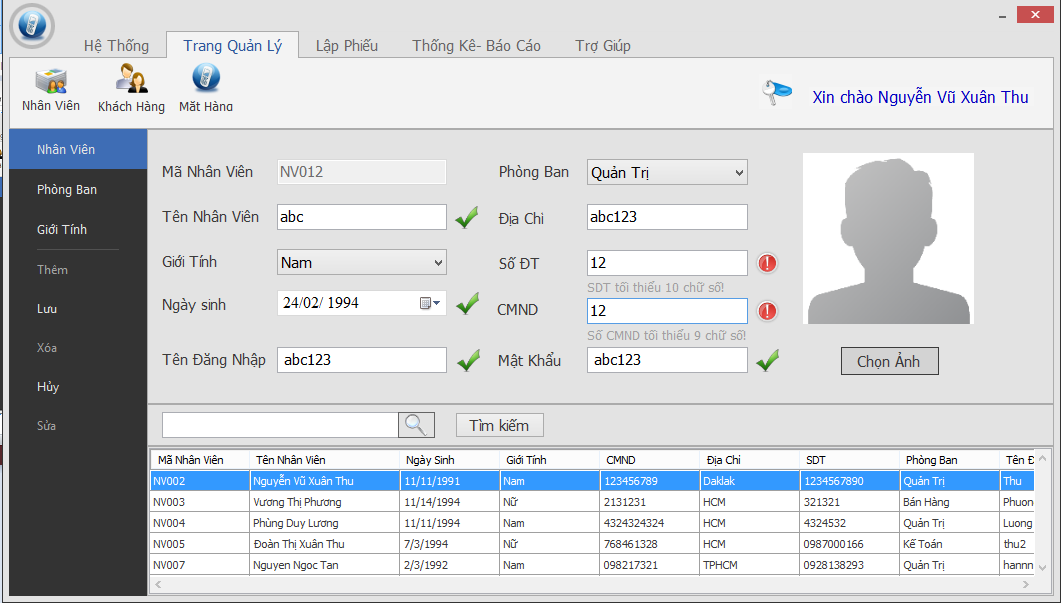
1. Thêm viên sai:



Lỗi bỏ trống trường dữ liệu



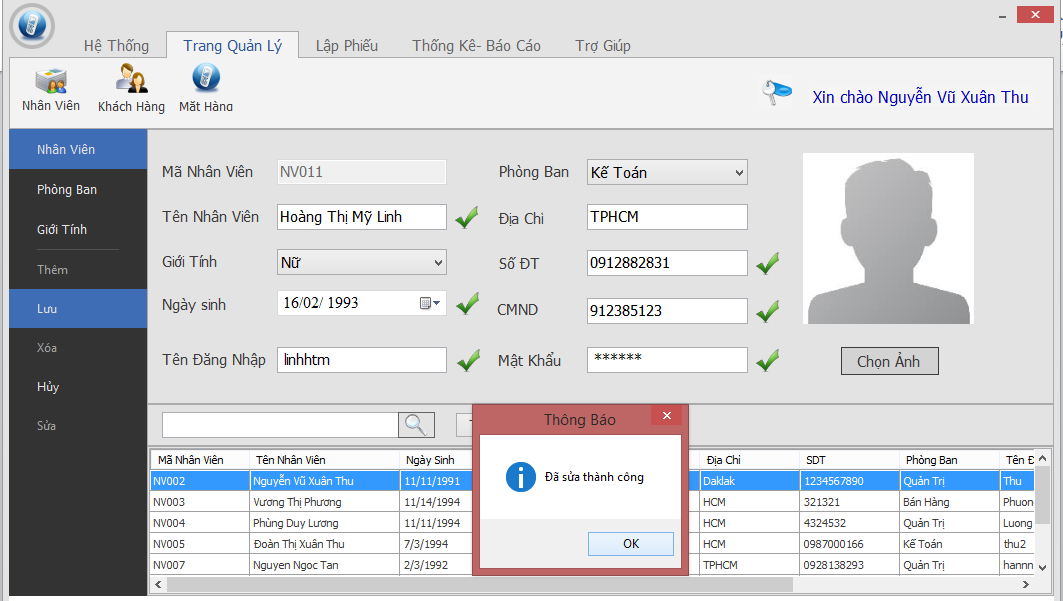
Lỗi nhập không đúng định dạng kiểu dữ liệu



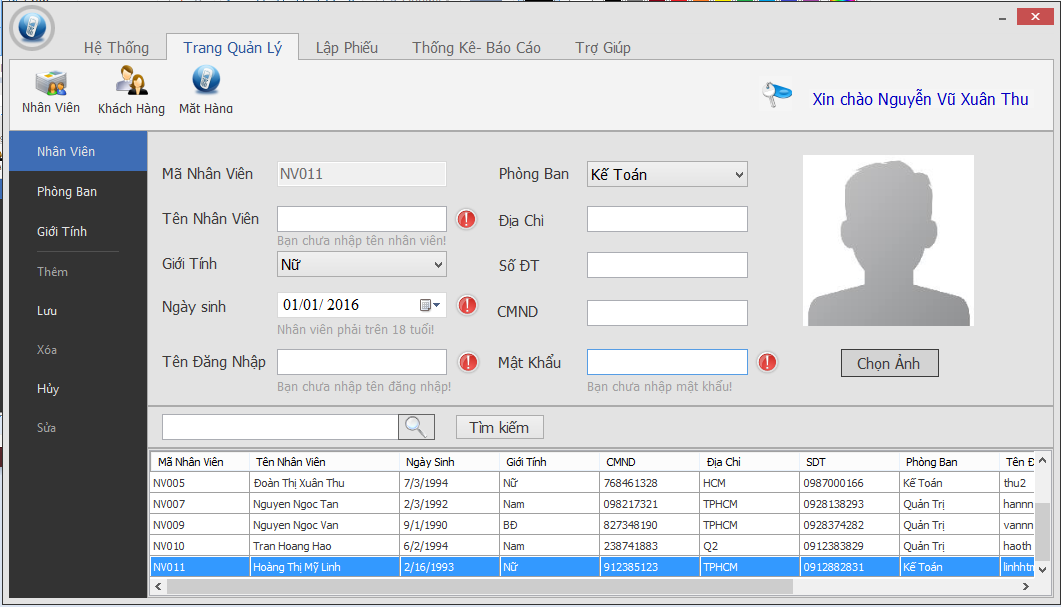
Lỗi nhập không đúng định dạng của số điện thoại và CMND

#### **4.4.2.2 Sửa nhân viên**

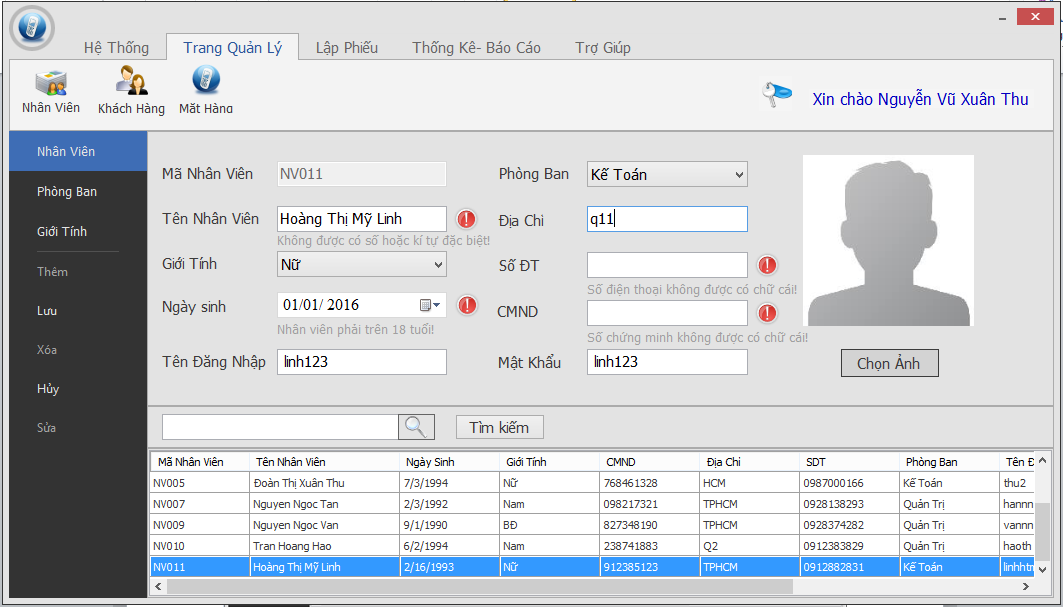
1. Sửa nhân viên đúng:



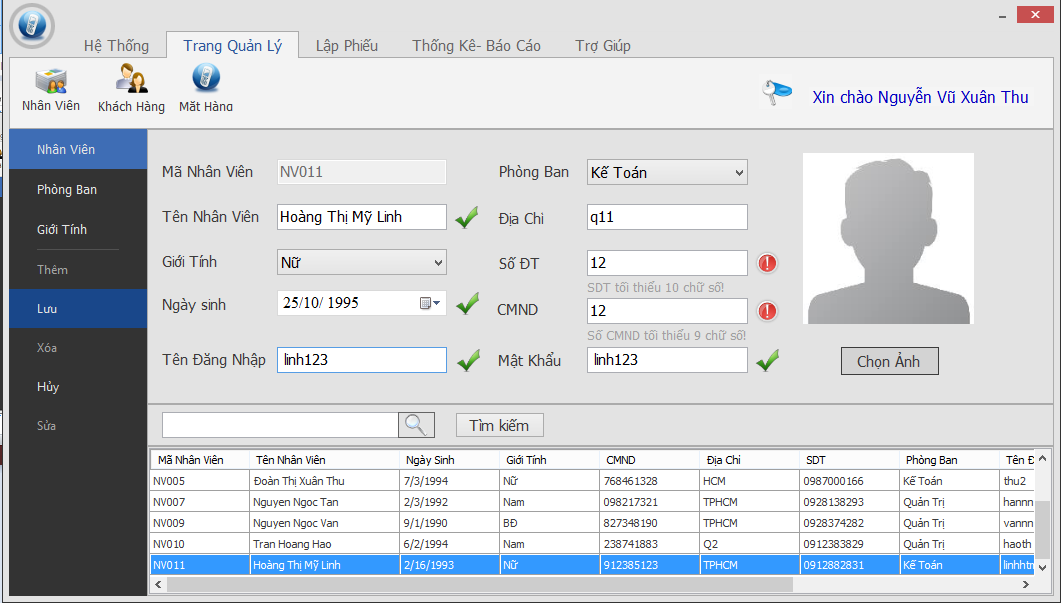
1. Sửa nhân viên sai:



Lỗi để trống các trường dữ liệu khi sửa



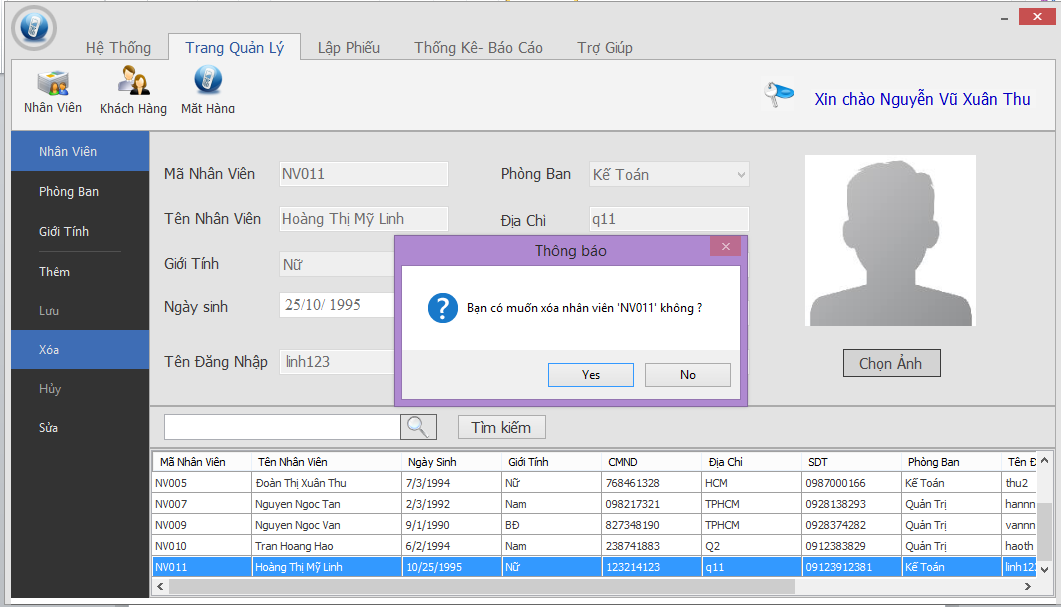
Lỗi nhập không đúng kiểu định dạng kiểu dữ liệu khi sửa



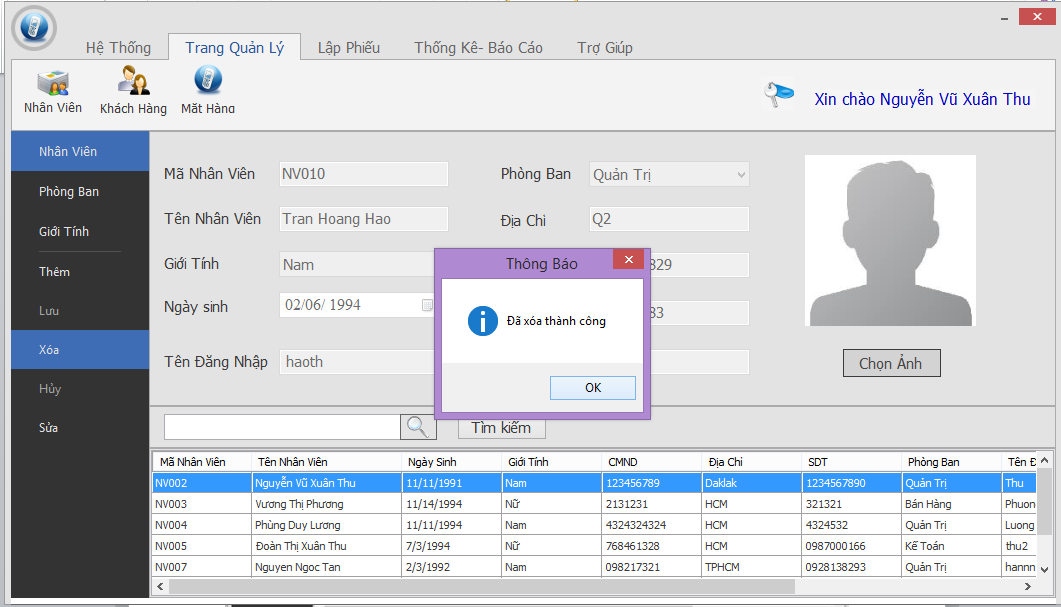
Lỗi nhập không đúng kiểu định dạng kiểu dữ liệu khi sửa

#### **4.4.2.3 Xoá nhân viên**

1. Xác nhận xoá:



1. Xoá thành công:

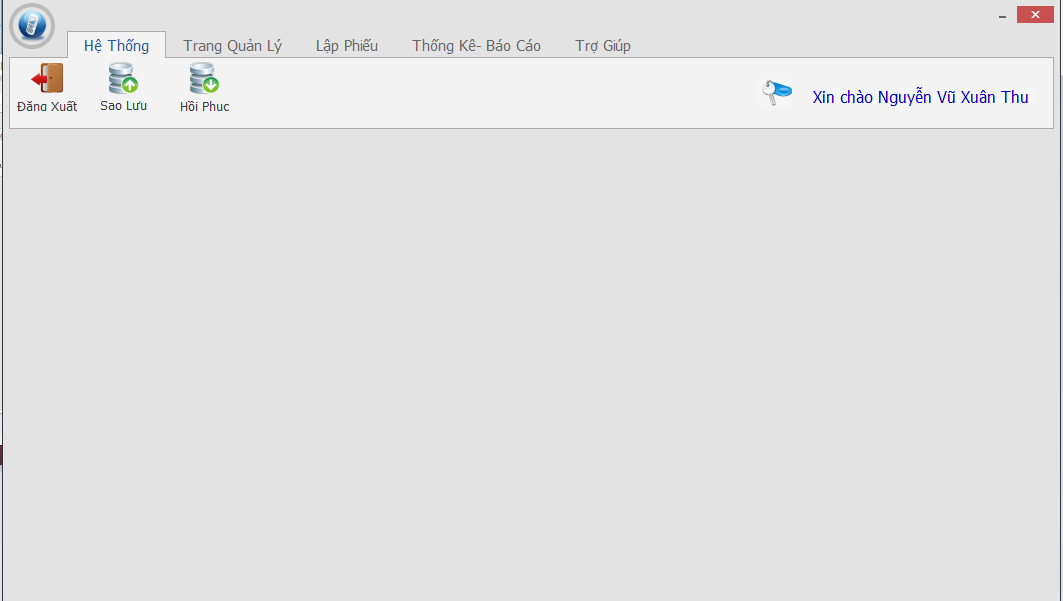


### **4.4.3 Chức năng đăng nhập**

* Đăng nhập đúng tài khoản:

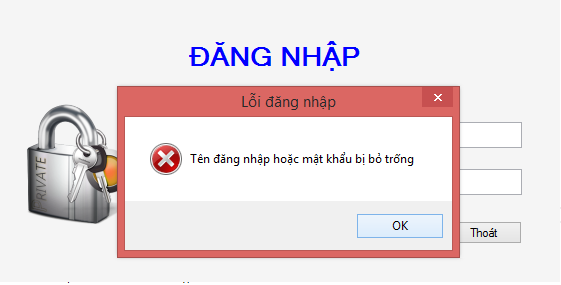


Nhập đúng tài khoản và mật khẩu

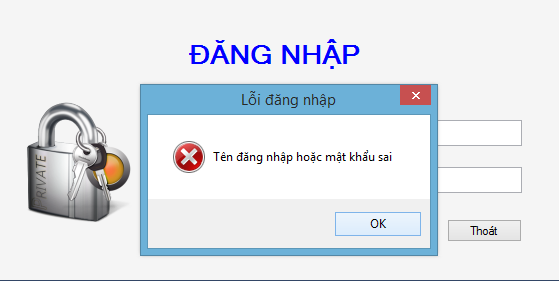


Hiển thị màn hình trang chủ khi bấm đăng nhập

* Đăng nhập bỏ trống:



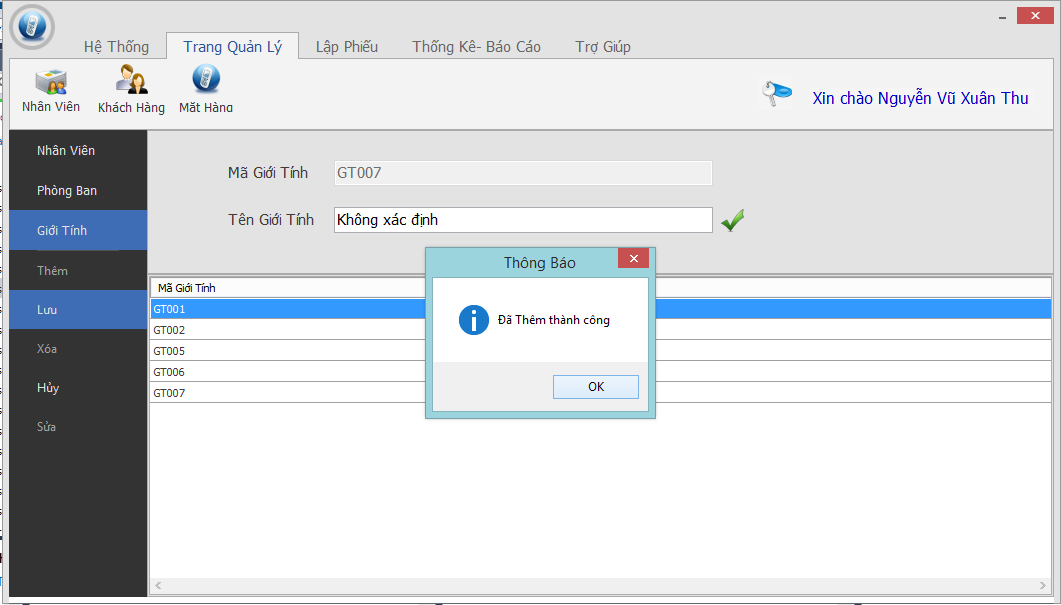
* Đăng nhập sai tài khoản:



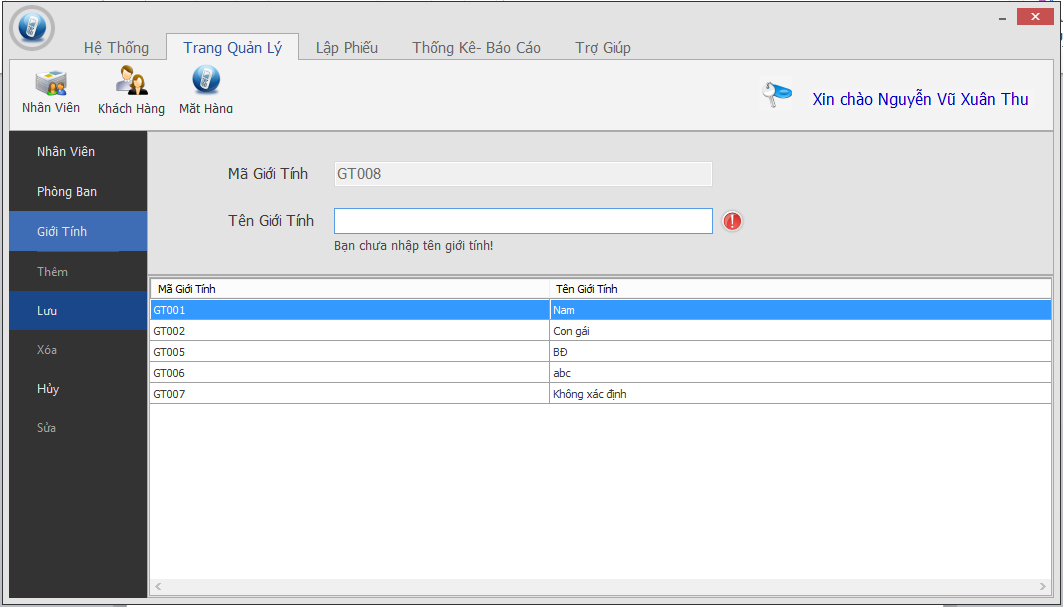
### **4.4.4 Chức năng quản lý giới tính**

#### **4.4.4.1 Thêm giới tính**

* Thêm giới tính đúng:

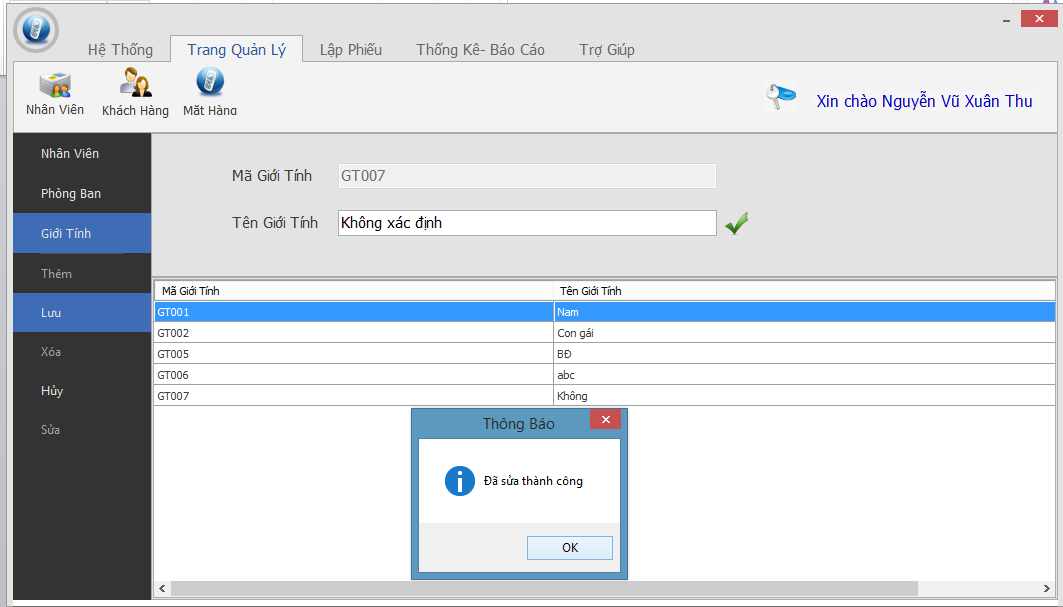


* Thêm giới tính sai:

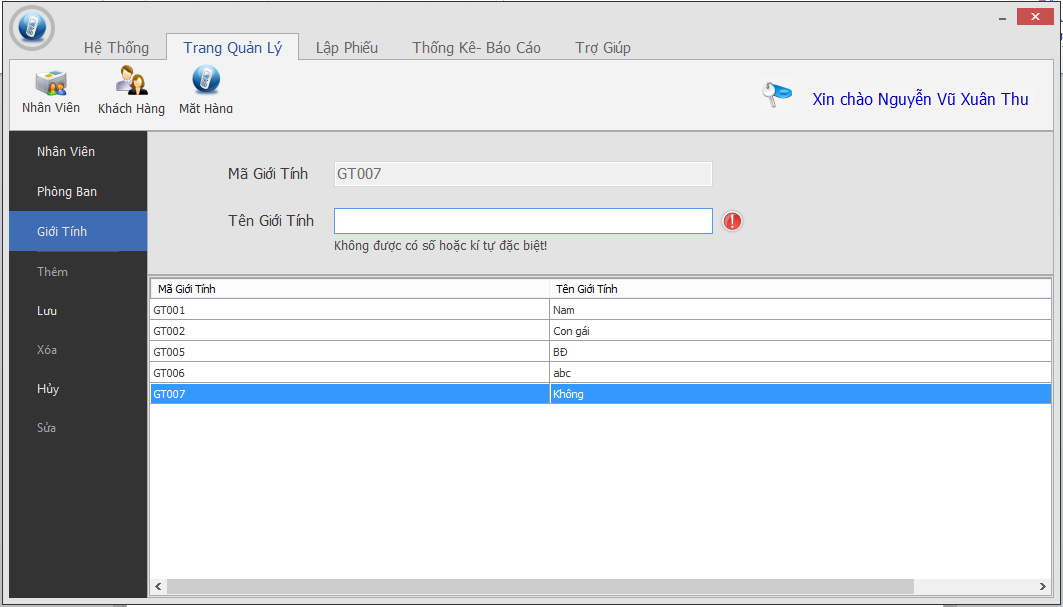


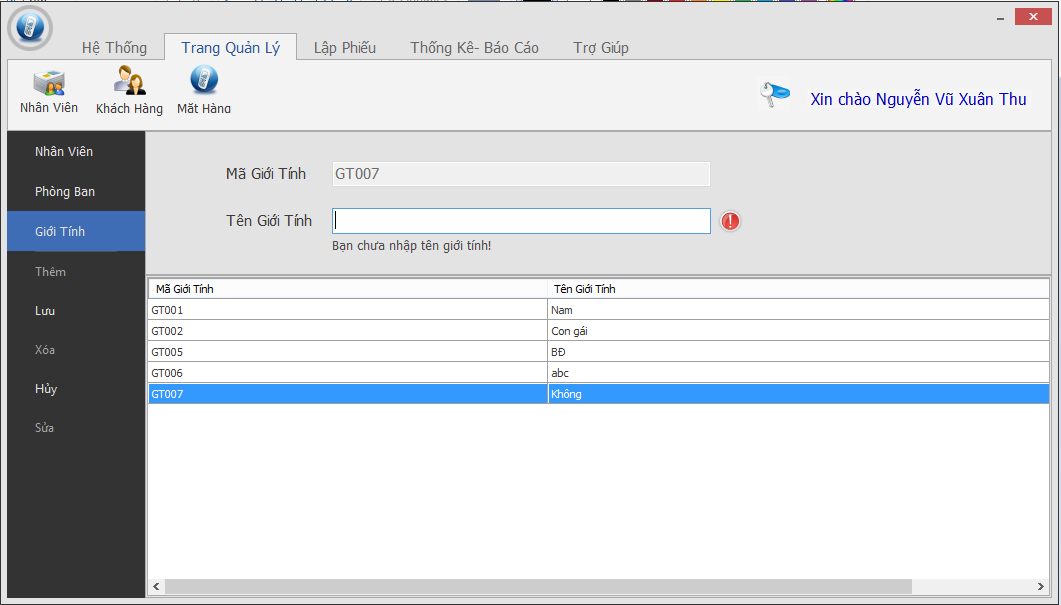
#### **4.4.4.2 Sửa giới tính**

* Sửa giới tính đúng:



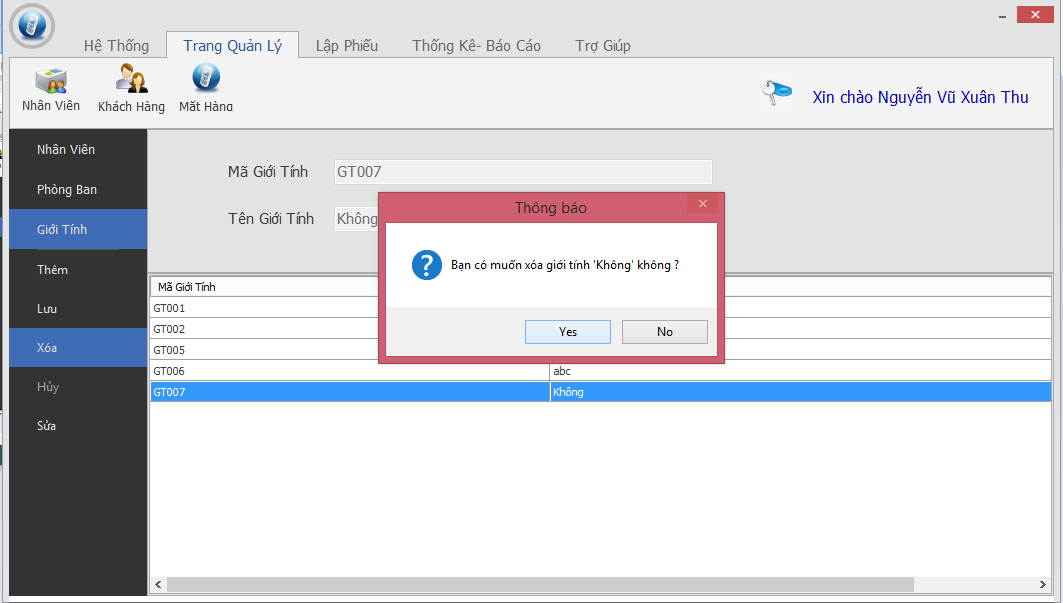
* Sửa giới tính sai:



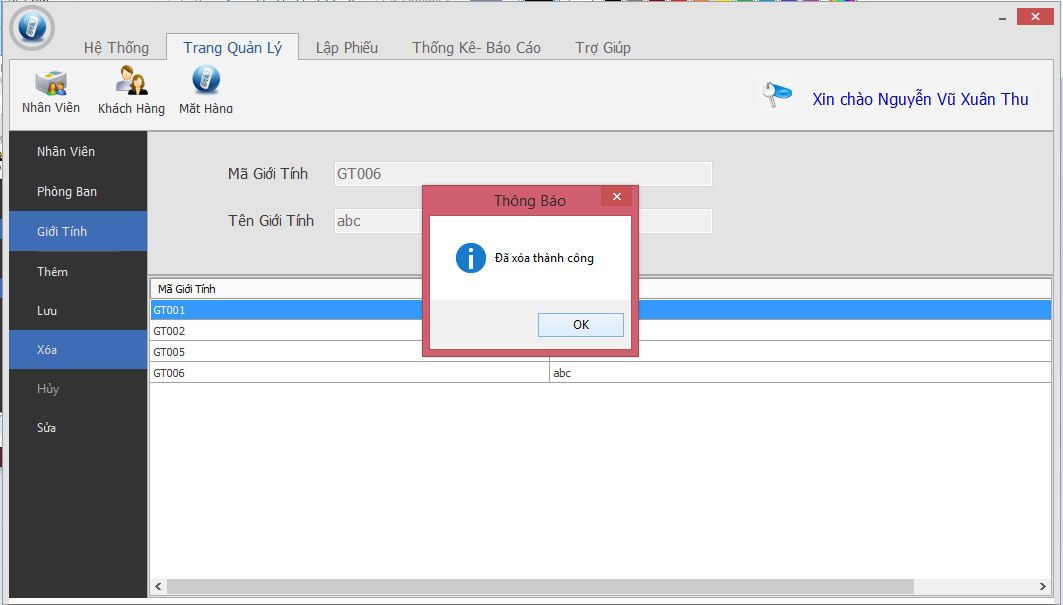


#### **4.4.4.3 Xoá giới tính**

* Xác nhận xoá:



* Xoá thành công:



## **4.5 Kết luận**

Sau khi sử dụng các kiến thức được học về các phương pháp kiểm thử phần mềm, các tool dùng để kiểm thử, nhóm đã vận dụng và viết được các unit test, integration test và automation test cho 3 chức năng chính của phần mềm nêu trên. Đã sử dụng tools sonar để auto deploy và check unit test coverage. Do chưa có nhiều kinh nghiệm và thời gian có hạn nên khi nhóm viết các unit test, integration test và automation test không tránh khỏi những lỗi nhỏ cũng như chưa chính xác hoàn toàn được.