**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**



**BÀI TẬP LỚN**

**KIỂM THỬ PHẦN MỀM**

**Tên đề tài: KIỂM THỬ PHẦN MỀM QUẢN LÝ THƯ VIỆN ĐIỆN TỬ VỚI NUNIT**

Giáo viên hướng dẫn : TS. Nguyễn Hoàng Tú

Lớp : ĐH KTPM 3 - K11

Nhóm : 06

Sinh viên thực hiện : Phạm Quốc Cường

Phạm Hải Dương

Phạm Văn Hiểu

**Hà Nội - 2019**

**LỜI NÓI ĐẦU**

Trong những năm gần đây, công nghệ thông tin đã phát triển nhanh chóng, mạnh mẽ và được ứng dụng ở hầu hết các mặt của đời sống, kinh tế, văn hóa, xã hội. Công nghệ thông tin phát triển cũng đồng nghĩa với nó là ngành công nghệ phần mềm phát triển. Việc xây dựng các phần mềm, ứng dụng web ngày càng phổ biến. Các phần mềm, ứng dụng web ngày càng trở nên thân thiết với con người qua các hoạt động giời thiệu, quảng cáo, mua bán, quản lý... trên mạng. Cuộc sống càng lên cao nhu cầu về chất lượng của một sản phẩm cũng tăng lên. Một sản phẩm không chỉ dừng lại ở việc thân thiện dễ sử dụng mà nó còn phải luôn chính xác, kịp thời và đảm bảo tính bảo mật tốt. Lỗi của một phần mềm hay một ứng dụng web có thể gây ra những hậu quả nghiêm trọng. Do đó, hoạt động kiểm thử phần mềm là rất quan trọng. Trong khi đó hoạt động này lại tiêu tốn và chiếm tỉ trọng khá lớn công sức và thời gian trong một dự án. Chính vì lí do đó, việc đưa vào quy trình kiểm thử tự động là vô cùng cần thiết.

Qua thực tế cho thấy việc áp dụng kiểm thử tự động hợp lý sẽ đem lại hiệu quả thành công cho hoạt động kiểm thử phần mềm. Kiểm thử tự động giúp giảm bớt công sức thực hiện, tăng độ tin cậy, giảm sự nhàm chán và rèn luyện kĩ năng lập trình cho cán bộ kiểm thử.

Vì những nguyên nhân trên nên em quyết định chọn đề tài: “Kiểm thử phần mềm quản lý thư viện bằng Nunit”.

***Nhóm sinh viên thực hiện!***

**Bảng phân chia công việc**

|  |  |
| --- | --- |
| **Họ tên** | **Công việc thực hiện** |
| Phạm Quốc Cường | Kiểm thử Chức năng Sửa độc giả |
| Phạm Hải Dương | Kiểm thử Chức năng Sửa thông tin sách |
| Phạm Văn Hiểu | Kiểm thử Chức năng Thêm sách |

**MỤC LỤC**

[Chương 1. TÌM HIỂU CÔNG CỤ NUNIT 1](#_Toc8472719)

[1.1 Tóm tắt lịch sử 1](#_Toc8472720)

[1.2 Đặc điểm 1](#_Toc8472721)

[1.3 Cài đặt và sử dụng công cụ Nunit 1](#_Toc8472722)

[CHƯƠNG 2. GIỚI THIỆU PHẦN MỀM QUẢN LÝ THƯ VIỆN 6](#_Toc8472723)

[2.1 Yêu cầu đề tài 6](#_Toc8472724)

[2.2 Phân tích thiết kế hệ thống 7](#_Toc8472725)

[2.2.1 Đăc tả hệ thống 7](#_Toc8472726)

[2.2.2 Thiết kế hệ thống 9](#_Toc8472727)

[CHƯƠNG 3. KIỂM THỬ PHẦN MỀM QUẢN LÝ THƯ VIỆN ĐIỆN TỬ VỚI NUNIT 20](#_Toc8472728)

[3.1 Lập kế hoạch kiểm thử 20](#_Toc8472729)

[3.1.1 Mục tiêu lập kế hoạch kiểm thử 20](#_Toc8472730)

[3.1.2 Nhiệm vụ của lập kế hoạch kiểm thử 21](#_Toc8472731)

[3.2 Thực hiện kiểm thử 22](#_Toc8472732)

[3.2.1 Phạm Hải Dương - chức năng Sửa sách (v0.1) 22](#_Toc8472733)

[3.2.2 Phạm Hải Dương - chức năng Sửa sách (v0.2) 28](#_Toc8472734)

[3.2.3 Phạm Văn Hiểu - chức năng thêm sách (V1) 34](#_Toc8472735)

[3.2.4 Phạm Văn Hiểu - chức năng thêm sách (V2) 43](#_Toc8472736)

[3.2.5 Phạm Quốc Cường – Kiểm thử chức năng Sửa độc giả (Version 0.1) 52](#_Toc8472737)

[3.2.6 Phạm Quốc Cường – Kiểm thử chức năng Sửa độc giả (Version 0.2) 63](#_Toc8472738)

TÌM HIỂU CÔNG CỤ NUNIT

## Tóm tắt lịch sử

Nunit là một testing framework cho tất cả các ngôn ngữ .Net. Được hình thành và phát triển lên từ JUnit bởi 1 nhóm lập trình viên (Charlie Poole, Rob Prouse and Simone Busoli), tương tác trực tiếp với Visual Studio, đồng thời có thể chạy độc lập mà không phụ thuộc vào Visual Studio

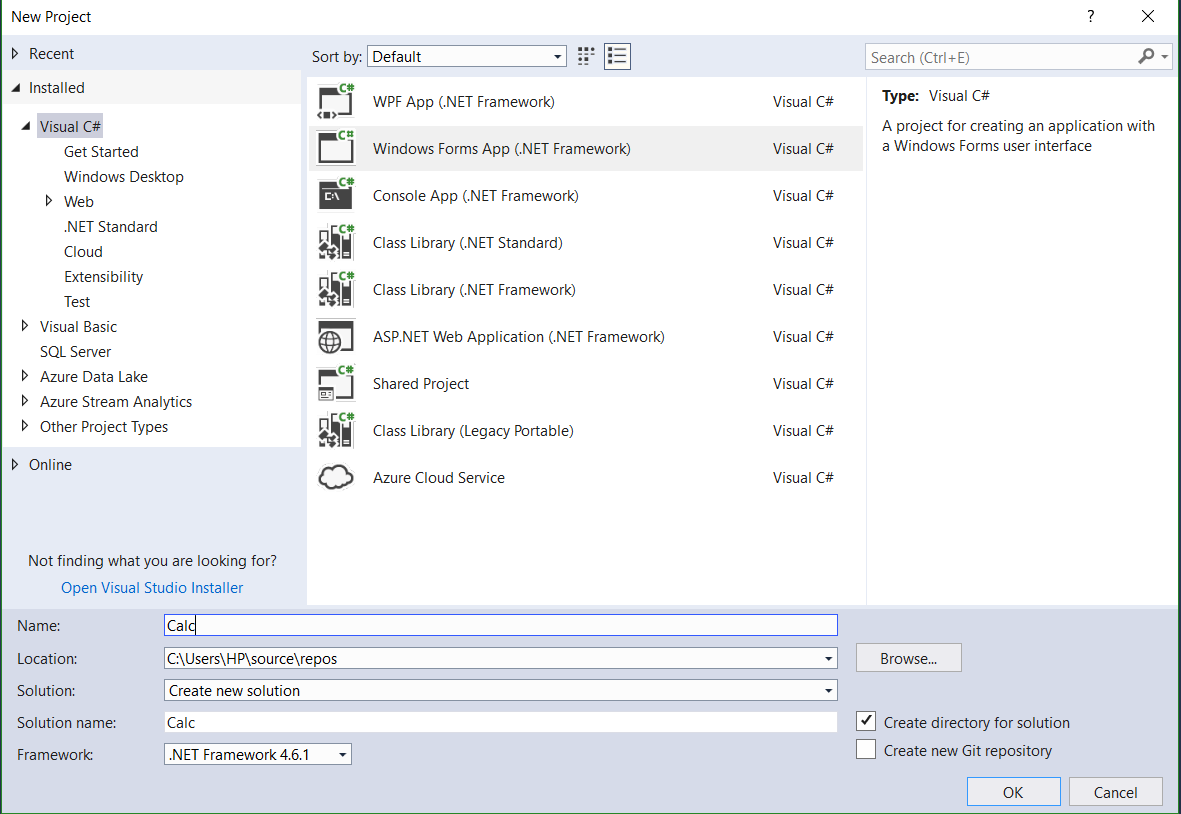
## Đặc điểm

Nunit là 1 công cụ mã nguồn mở, hỗ trợ kiểm thử đơn vị trên Visual Studio dành cho ngôn ngữ C#, được tích hợp dưới dạng 1 extension đây cũng là mặt hạn chế của công cụ Nunit.

## Cài đặt và sử dụng công cụ Nunit

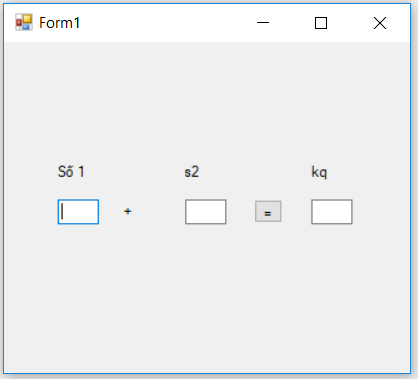
Để có thể cài đặt được NUnit cần phải có bộ cài của Nuint. Bạn đọc có thể download ở trang chủ của NUnit: <https://nunit.org/download/> . Sau đó giải nén file vừa được tải về.

* Cách sử dụng :
* Bước 1: Mở Visual Studio 2017, chọn New -> Project…



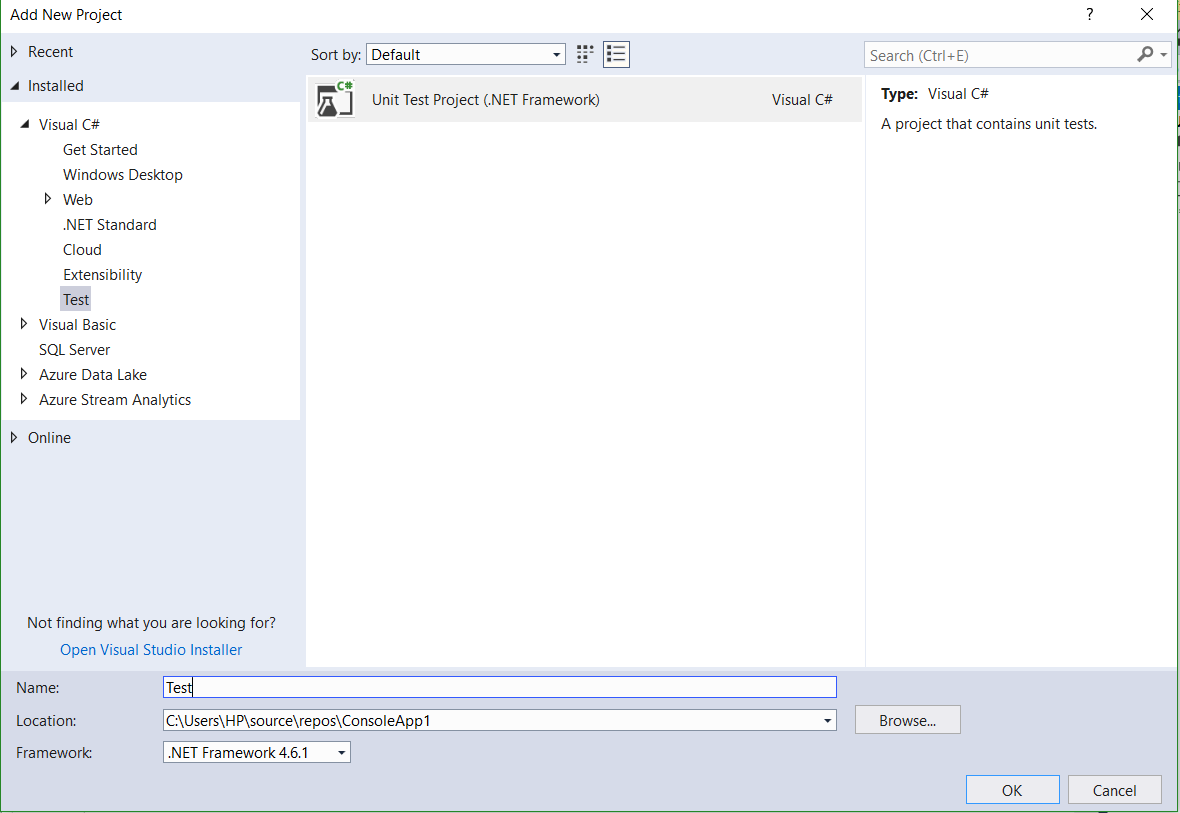
*Hình 1.1 Giao diện tạo project*

* Bước 2: Tạo form như hình mẫu:

**

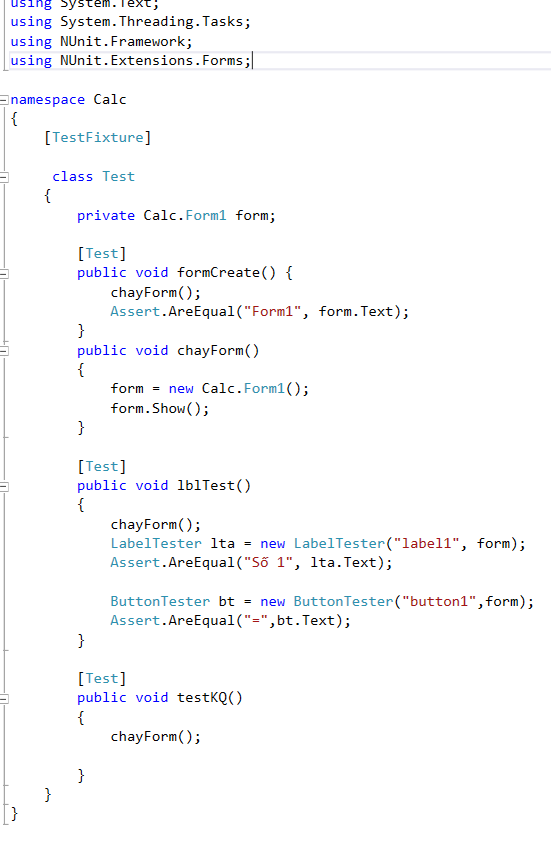
*Hình 1.2 Form mẫu*

* Bước 3: Thêm vào solution 1 project có tên ‘Test’ (Kích chuột phải vào Solution /Add/New Project /)

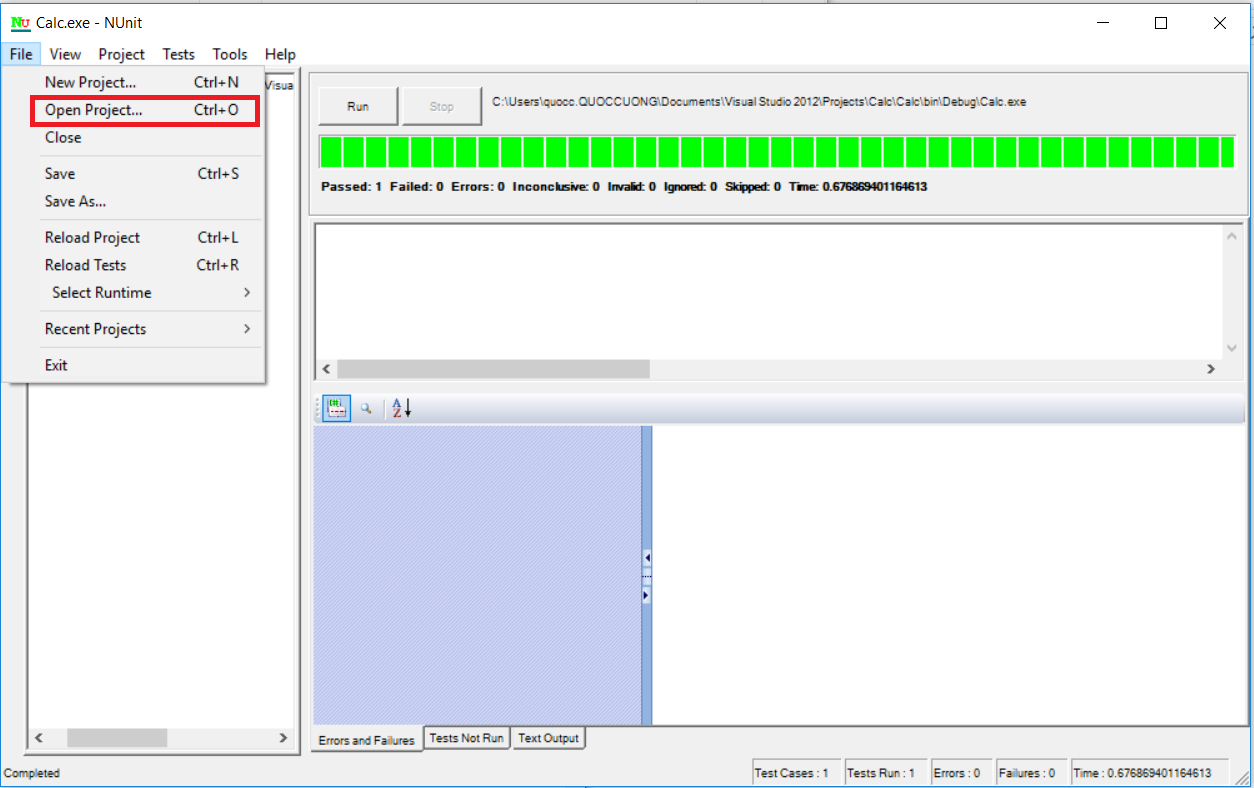


*Hình 1.3 Tạo Test Project*

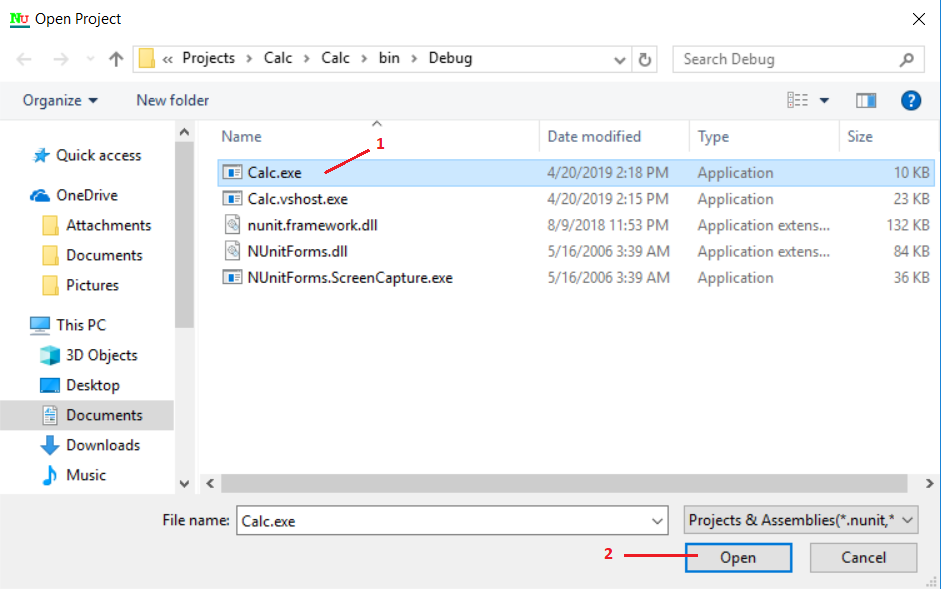
* Bước 4 : Thêm một tham chiếu tới các dự án đang được test. Select Project →Add Reference. Thêm một tham chiếu tới NUnit Framework: Add reference →Tab Project → Chọn dự án mà đang test →OK
* Bước 5: Viết code sử dụng để test trong class ‘Test’



* Bước 6 : vào thư mục đã giải nén lúc trước mở file nunit.exe, sau đó chọn file -> open chọn project mình muốn test(project ‘Calc’ vừa tạo)

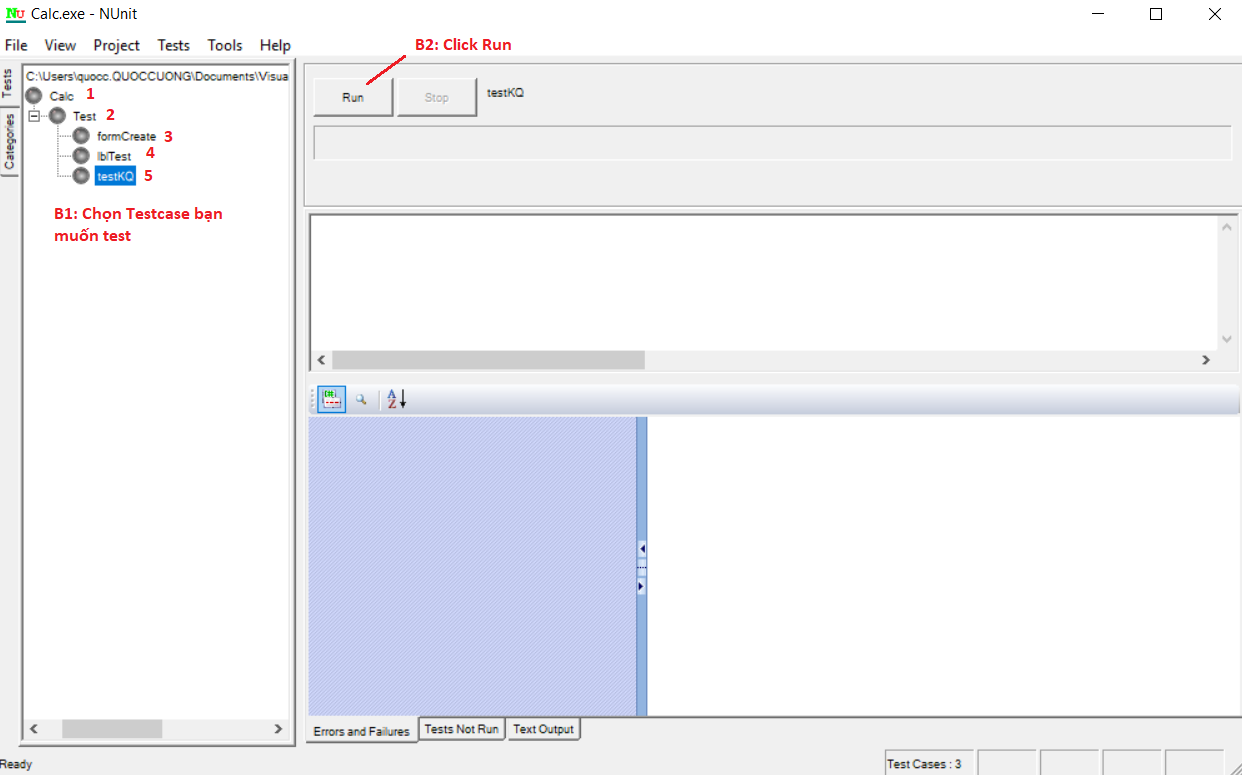


*Hình 1.4 Giao diện Nunit*



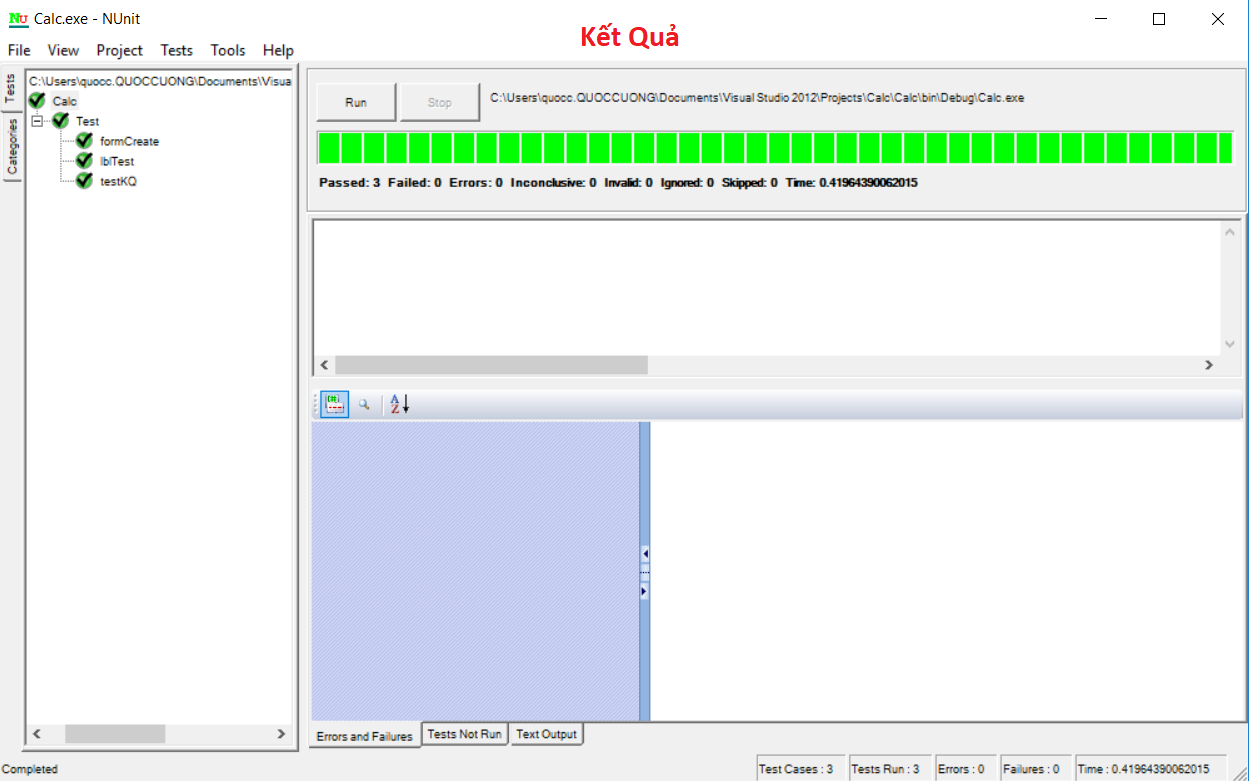
*Hình 1.5 Thêm file chạy test*

* Bước 7 : Chọn test case bạn muốn test, sau đó bấm Run



*Hình 1.6 Chạy chương trình*

* Kết quả



*Hình 1.7 Kết quả test*

# Giới thiệu phần mềm Quản lý thư viện

## Yêu cầu đề tài

1. Lý do chọn đề tài

Lĩnh vực mà nhóm chọn để thực hiện đề tài lần này chính là lĩnh vực quản lý với các thư viện truyển thống hiện nay. Đây là lĩnh vực rất phát triển, tiềm tàng nhiều cơ hội bởi vì còn có rất nhiều các thư viện đang sử dụng sức người, giấy tờ để quản lý thư viện. Chính vì vậy việc quản lý còn gặp rất nhiều khó khăn, dễ mắc sai lầm, cũng như tốn nhiều thời gian, do đó việc quản lý thư viện bằng công nghệ đang là xu hướng phát triển hiện nay.

Khảo sát trên địa bàn Hà Nội cho thấy nhiều thư viện từ lớn đến nhỏ đã áp dụng có hiệu quả các sản phẩm tin học, đều cho ra phản ưng tích cực về hiệu quả. Do vậy trong đợt báo cáo này em chọn đề tài: “Quản lý thư viện điện tử” để củng cố kiến thức đã học trong trường, nâng cao kỹ năng làm bài cũng như tiếp cận trực tiếp với thực tế phục vụ công tác sau này.

1. Mục đích nghiên cứu

Áp dụng được các kiến thức đã được học trong nhà trường, cũng như kiến thức thực tế. Nhằm hiểu rõ về quy trình xây dựng, phát triển phần mềm (khảo sát, phỏng vấn, lập kế hoạch, phân công công việc, xây dựng phần mềm, kiểm thử, bàn giao khách hàng), giúp cho chúng em có thêm kiến thức, kinh nghiệm cho việc phát triển phần mềm sau này.

1. Bài toán kiểm thử

Trong quá trình xây dựng và phát triển phần mềm chắc chắn không thể tránh được việc gặp những lỗi về hệ thống hay giao diện. Do vậy việc kiểm thử là vô cung cần thiết để đảm bảo chất lượng phần mềm bàn giao tới tay khách hàng. Bài toán kiểm thử đặt ra ở đây là làm sao để có thể khắc phục tối đa những lỗi mà chương trình gặp phải, muốn như thế thì người kiểm thử cần thu thập nhiều thông tin, hiểu biết về hệ thống và công cụ kiểm thử. Nên yêu cầu đặt ra là cần hiểu rõ phần mềm mà mình đang xây dựng, phát triển và phải có hiểu biết và ứng dụng công cụ kiểm thử mình định sử dụng, ngoài ra có thể tìm hiểu thêm một vài công cụ kiểm thử hiệu năng khác.

## Phân tích thiết kế hệ thống

### Đăc tả hệ thống

#### Khảo sát sơ bộ

* Tên hệ thống : Quản lý thư viện Vạn Hoa
* Thư viện Vạn Hoa nằm tại 264 Cổ Nhuế, Bắc Từ Liêm, Hà nội với tổng diện tích là 585 m2 , thư viện gồm có :
* 1 phòng đọc
* 1 phòng báo, tạp chí
* Phục vụ mượn về nhà :
* 1 phòng mượn
* 1 kho giáo trình
* Ngoài ra thư viện còn có phòng thảo luận nhóm với sức chứa 150 chỗ ngồi
* Trang thiết bị kĩ thuật được đầu tư gồm có :
* 60 máy client
* 2 máy server
* 4 máy in laser
* 2 máy scanner
* 2 máy quét mã vạch
* Mới được thành lập từ năm 2014 nên bộ máy quản lý còn đơn giản , mong muốn quản lý công tác tốt , đảm bảo sự phát triển bền vững lâu dài.
* Thư viện Vạn Hoa hiện tại có thủ thư và nhân viên thường trực là người trực tiếp quản lý hoạt động của thư viện.
* Nhiệm vụ chính là kiểm soát người đọc ,phiếu mượn sách và thống kê sách cụ thể:
* Sách được phân chia theo
* Tên tác giả : Nguyên Hồng , La Quán Trung , Ngô Thừa Ân,..
* Thể loại :truyện tranh , khoa học , viễn tưởng , khoa học , giáo dục ,…
* Độc giả mượn sách sẽ được lập phiếu mượn trong thời gian nhất định và cần cung cấp một số thông tin bắt buộc :
* Tên
* Địa chỉ
* Số điện thoại.
* Phiếu mượn sách gồm các thông tin cơ bản như :
  + - Mã phiếu
    - Mã độc giả
    - Mã sách
    - Số lượng sách mượn
    - Ngày mượn và ngày trả

#### Tài liệu đặc tả yêu cầu người dùng

* Phân tích yêu cầu người dùng :

|  |  |
| --- | --- |
| **Câu hỏi** | **Câu trả lời của khách hàng** |
| Anh/chị có thể cho chúng tôi biết mục địch anh/chị muốn có phần mềm không ạ ? | Thư viện chúng tôi muốn có phần mềm để dễ duyệt, quản lý các thông tin tài liệu, đọc giả, danh sách các phiếu trong thư viện, mục đích giảm thiểu các hồ sơ về giấy tờ cũng như đơn giản hóa trong công việc quản lý. |
| * Anh/chị có yêu cầu cơ bản nào về phần mềm quản lý này không ? | Tôi muốn phần mềm phải có các chức năng cơ bản như :thêm, sửa, xóa và tìm kiếm dữ liệu cho phần sách, độc giả và phiếu. |
| * Anh/ chị có muốn tài khoản để quản lý chứ ? | Vâng , tôi muốn có tài khoản với quyền quản trị nội dung. |
| Anh/chị có muốn Sách gồm những thông tin cụ thể nào ? | Có, tôi muốn thông tin về Sách phải có đủ : mã sách, tên sách, thể loại, số lượng, tác giả. |
| * Anh/chị có muốn Đọc giả gồm những thông tin cụ thể nào ? | Phía đọc giả tôi muốn có : mã độc giả, tên độc giả, giới tính, địa chỉ, số điện thoại. |
| * Anh/chị có yêu cầu gì về phần phiếu mượn trả ? | Không, khi cần chúng tôi sẽ liên lạc |
| * Anh/chị muốn chức năng tìm kiếm ra sao thì đáp ứng được yêu cầu ? | Chúng tôi muốn thanh tìm kiếm để khi nhập dữ liệu chúng tôi có thể tìm được các thông tin mong muốn. |
| * Anh/ chị có dặn giò thêm gì hay không nữa ạ ? | Hiện tại thì không có gì cần thêm, có gì chúng tôi sẽ liên lạc lại sau. |

* Yêu cầu chức năng
* Quản lý sách :
* Thêm sách : thêm sách mới bao gồm : mã sách, tên sách,thể loại, số lượng, tác giả
* Sửa sách : sửa lại thông tin của sách khi có sai sót
* Xóa sách : xóa 1 sách được chọn
* Tìm kiếm : tìm kiếm theo dữ liệu mong muốn
* Yêu cầu phi chức năng :
* Bảo mật về dữ liệu
* Tốc độ xử lý tốt
* Giao diện đẹp, màu sáng
* Dễ bảo trì , nâng cấp
* Tương thích với hệ điều hành Windows
* Mô tả hoạt động hệ thống :
* Người quản trị đăng nhập tài khoản và mật khẩu được cấp sẵn để đăng nhập vào hệ thống.
* Màn hình hiện lên với giao diện quản lý sách ở đây ta có thể thực hiện các chức năng : thêm sách mới, sửa sách, xóa sách và tìm kiếm
* Với giao diện quản lý độc giả: ta có thể xem các thông tin về độc giả, thực hiện thêm độc giả mới, sửa thông tin độc giả, xóa độc giả và chức năng tìm kiếm độc giả, lập phiếu mượn sách cho độc giả.
* Với giao diện quản lý phiếu: người quản lý có thể lập phiếu mượn, trả. Sửa phiếu nếu sai thông tin, xóa phiếu và tìm kiếm phiếu một cách rễ ràng hơn.

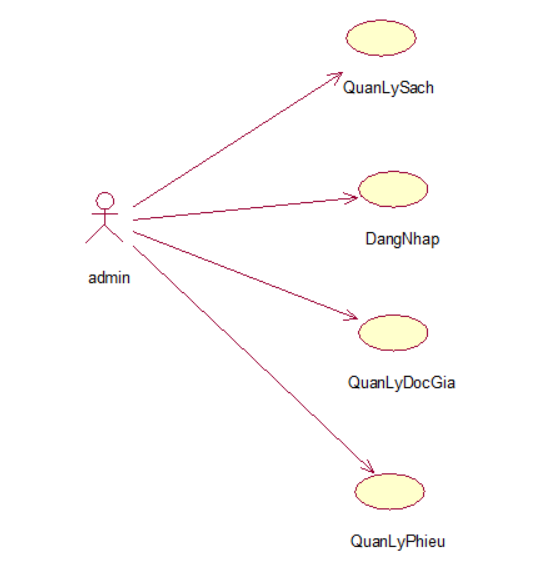
### Thiết kế hệ thống

#### Mô hình hóa chức năng của hệ thống

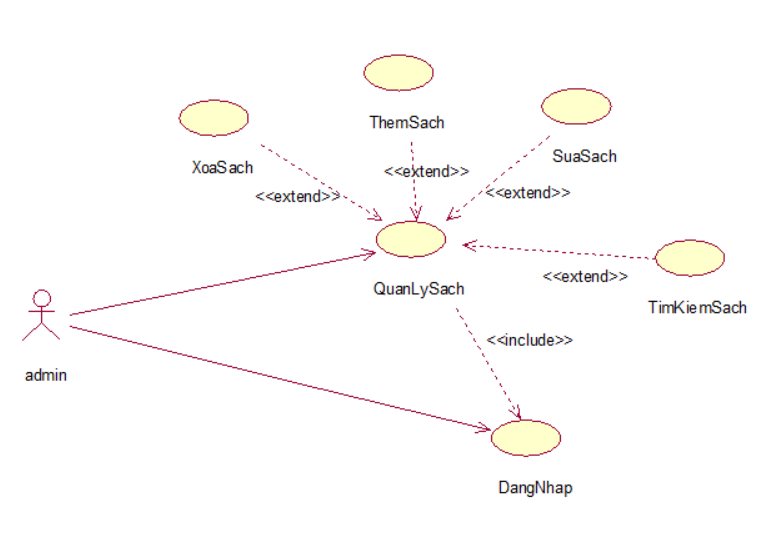
* **Actor :**

|  |  |
| --- | --- |
| **Tác nhân** | **Ca sử dụng** |
| Người quản lý (Admin) | * Đăng nhập. * Quản lý Sách (thêm, sửa, xóa, tìm kiếm). * Quản lý Độc giả (thêm, sửa, xóa). * Quản lý Phiế (Lập,sửa xóa,tìm kiếm). |

* **Biểu đồ UseCase :**



*Hình 2.1 Biều đồ usecase tổng quát của hệ thống*



*Hình 2.2 Biểu đồ usecase QuanLySach*



*Hình 2.3 Biểu đồ usecase quanLyDocGia*



*Hình 2.4 Biểu đồ usecase QuanLyPhieu*

* **Đặc tả UseCase**
* **Usecase DangNhap**

1. Mô tả : người quản lý đăng nhập vào hệ thống với 1 tên tài khoản và mật khẩu có sẵn
2. Các luồng sự kiện

* Luồng sự kiện chính
* Ca sử dụng này bắt đầu khi người dùng muốn đăng nhập vào hệ thống.
* Hệ thống yêu cầu người dùng nhập tài khoản và mật khẩu.
* Người dùng nhập tên đăng nhập và mật khẩu có sẵn.
* Hệ thống xác nhận tài khoản và mật khẩu có hợp lệ hay không, nếu không hợp lệ thì thực hiện luông A1.
* Hệ thống ghi lại quá trình đăng nhập.
* Các luồng rẽ nhánh
* Luồng A1 : Nhập sai tài khoản/mật khẩu đăng nhập
* Hệ thống thông báo lỗi.
* Người dùng có thể chọn đăng nhập lại hoặc thoát chương trình, khi đó ca sử dụng kết thúc.

1. Tiền điều kiện

Hệ thống chưa được đăng nhập

1. Hậu điều kiện

Nếu đăng nhập thành công, người sử dụng sẽ đăng nhập vào hệ thống.

* **Usecase QuanLySach**

1. Mô tả : người quản lý có thể quản lý sách thông qua giao diện , và các chức năng bao gồm, thêm, sửa, xóa và tìm kiếm sách.
2. Các luồng sự kiện

* Luồng sự kiện chính
* Sau khi đăng nhập màn hình hiển thị với giao diện quản lý sách
* Thêm sách : thêm sách mới
* Người dùng nhập dữ liệu vào các ô text sau đó bấm Thêm .
* Hệ thống kiểm tra xem thông tin sách có hợp lệ không. Nếu không thì sẽ thực hiện luồng A2.
* Thêm sách mới thành công, kêt thúc ca sử dụng.
* Sửa sách : sửa thông tin sách
* Người dùng chọn sách cần sửa.
* Người dùng nhập lại thông tin sách.
* Người dùng chọn Sửa để lưu lại thông tin.
* Hệ thống kiểm tra thông tin sách, nếu không hợp lệ thực hiện luồng A2.
* Sửa thông tin lớp thành công , kết thúc ca sử dụng.
* Xóa sách :
* Người dùng chọn lớp cần xóa.
* Người dùng bấm nút Xóa.
* Hệ thống sẽ hỏi lại có thật sự muốn xóa không.
* Người dùng xác nhận là muốn xóa. Nếu không xóa thực hiện luồng A3.
* Hệ thống kiểm tra ràng buộc, nếu có ràng buộc thì thực hiện luồng A4
* Lớp chọn sẽ bị xóa. Kết thúc ca sử dụng.
* Tìm sách :
* Người dùng nhập dữ liệu muốn tìm kiếm
* Nếu có hệ thống sẽ hiển thị kết quả tìm kiếm ngược lại thì sẽ không hiện gì .
* Luồng rẽ nhánh :
* Luồng A2 : thông tin sách mới không hợp lệ, hệ thống báo lỗi và yêu cầu nhập lại.
* Luồng A3 : xác nhận không muốn xóa lớp đã chọn, hệ thống sẽ không xóa lớp đã chọn và kết thúc ca sử dụng
* Luồng A4 : có ràng buộc, hệ thống thông báo không thể xóa lớp đã chọn và kết thúc ca sử dụng.

1. Tiền điều kiện

Người dùng cần phải đăng nhập

1. Hậu điều kiện

Sách có thể thêm, sửa, xóa tìm kiếm thông tin.

* **Usecase QuanLyDocGia**

1. Mô tả

* Tên ca sử dụng: Quản lý độc giả.
* Mục đích: mô tả cách người dùng muốn quản lý thông tin học sinh: thêm mới, sửa, xóa thông tin độc giả mượn sách.
* Tác nhân: Quản lý thư viện.

1. Các luồng sự kiện

* Luồng sự kiên chính:
  + Từ giao diện chính người dùng chọn quản lý thông tin độc giả.
  + Màn hình sẽ hiển thị giao diện quản lý độc giả.

1. Người dùng muốn thêm độc giả.

* Người dùng bấm nút “Độc Giả” trên giao diện.
* Hệ thống sẽ hiển thị ra màn hình giao diện để người dùng điền thông tin độc giả cần thêm.
* Người dùng điền thông tin độc giả cần thêm rồi bấm Thêm,nếu không muốn thêm nữa thì bấm Hủy.
* Hệ thống sẽ kiểm tra thông tin của độc giả mới có hợp lệ không. Nếu không hợp lệ thì thực hiện luồng A11.
* Hệ thống ghi lai quá trình đăng nhập,kết thúc ca sử dụng.

1. Sửa thông tin độc giả: người dùng muốn sửa thông tin độc giả.

* Người dùng nhập mã chọn độc giả cần sửa trên mục tìm kiếm .
* Người dùng nhập lại thông tin độc giả.
* Người dùng chọn Lưu, nếu không muốn sửa nữa chọn Hủy.
* Hệ thống kiểm tra thông tin độc giả, nếu không hợp lệ thì thực hiện luồng A11.
* Sửa thông tin độc giả thành công. Kết thúc ca sử dụng.

1. Xóa thông tin độc giả: người dùng muốn xóa độc giả

* Người dùng nhập mã tìm kiếm độc giả cần xóa.
* Người dùng bấm nút Xóa.
* Hệ thống sẽ hỏi lại có thật sự muốn xóa không.
* Người dùng xác nhận là muốn xóa. Nếu không thực hiện luồng A12.
* Hệ thống kiểm tra ràng buộc. nếu có ràng buộc thì thực hiện luồng A13.
* Độc giả được chọn sẽ bị xóa.kết thúc usecase.
* Luồng rẽ nhánh:
* Luồng A11: thông tin độc giả mới không hợp lệ.

Hệ thống sẽ gửi thông báo lỗi và yêu cầu nhập lại.

* Luồng A12: xác nhận không muốn xóa độc giả đã chọn.

Hệ thống sẽ không xóa độc giả đã chọn và usecase sẽ kết thúc.

* Luồng A13: có ràng buộc.

Hệ thống thông báo có ràng buộc, không thể xóa độc giả đã chọn. Kết thúc ca sử dụng.

1. Tiền điều kiện :

Người dùng phải đăng nhập được vào hệ thống quản lý của thư viện

1. Hậu điều kiện :

Độc giả phải có thêm , sửa, xóa, thông tin

* **Usecase QuanLyPhieu**

1. Mô tả : Người quản lý có thể quản Phiếu thông qua giao diện, và các chức năng bao gồm, thêm, sửa, xóa và tìm kiếm phiếu.
2. Các Luồng Sự kiện :

* Luồng Sự Kiện Chính :
* Sau khi người dùng Đăng Nhập hiện thị giao diện với quản lý phiếu.
* Người dùng lập thêm phiếu :
  + - Người dùng nhập dữ liệu vào các ô text sau đó bấm Thêm .
    - Hệ thống kiểm tra xem thông tin phiếu có hợp lệ không. Nếu không thì sẽ thực hiện luồng B2.
    - Thêm Phiếu mới thành công và kêt thúc.
* Sửa Phiếu :
  + - * + Người hiếu cần sửa.
        + Người dùng lại thông tin phiếu.
        + Người dùng chọn Sửa để lưu lại thông tin.
        + Hệ thống kiểm tra thông tin phiếu, nếu không hợp lệ thực hiện luồng B2.
        + Sửa thông tin lớp thành công và kết thúc.
* Xóa Phiếu :
  + Người dùng chọn Phiếu cần xóa và bấm Nút xóa để thực hiện.
  + Hệ thống sẽ hỏi lại có thật sự muốn xóa không.
  + Người dùng xác nhận là muốn xóa. Nếu không xóa thực hiện luồng B3.
  + Hệ thống kiểm tra ràng buộc, nếu có ràng buộc thì thực hiện luồng B4
  + Lớp chọn sẽ bị xóa và Kết thúc .
* Luồng rẽ nhánh :
* Luồng B2 : thông tin phiếu mới không hợp lệ, hệ thống báo lỗi và yêu cầu nhập lại.
* Luồng B3 : xác nhận không muốn xóa lớp đã chọn, hệ thống sẽ không xóa lớp đã chọn và kết thúc.
* Luồng B4 : có ràng buộc, hệ thống thông báo không thể xóa lớp đã chọn và kết thúc.

1. Tiền điều kiện :

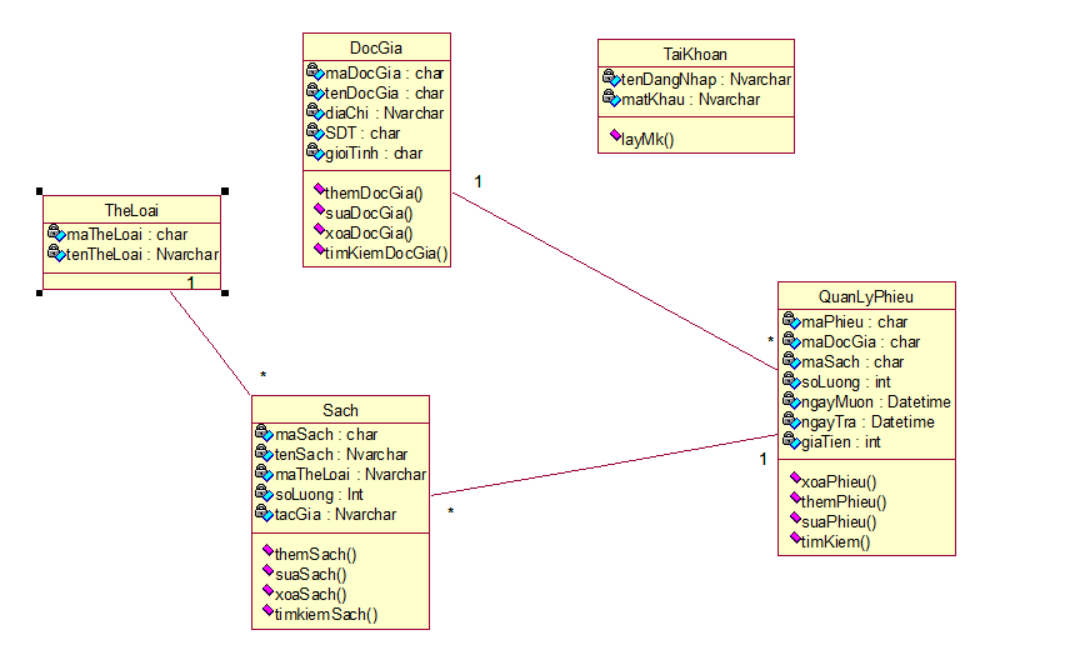
Người dùng phải đăng nhập.

1. Hậu điều kiện :

Thông tin phiếu có thể thêm ,xóa, sửa.

#### Biểu đồ mô hình hóa hệ thống

* Theo phần phân tích ở trên thì hệ thống có những chức năng sau:
* Đăng Nhập
* Quản Lý Sách
* Quản Lý Độc Giả
* Quản Lý Phiếu.
* Để phân tích tìm các lớp nhóm em sẽ thực hiện bằng phương pháp từ trên xuống . Hệ thống mà nhóm muốn xây dựng là hệ thống quản lý thư viện nên cần phải kiểm soát được: Người sử dụng hệ thống, quản lý về sách, người đọc cũng như việc cho mượn sách.
* Trong phần phân tích usecase tiền điều kiện của các chức năng đều yêu cầu phải được đăng nhập tài khoản dành cho người quản trị. Nên 1 lớp bắt buộc phải có đó là “TaiKhoan”.
* Ta thấy các chức năng cũng như yêu cầu của hệ thống đều liên quan đến “Sách”, vì vậy lớp cần phải có đó là lớp “Sach”.
* Lớp “Sach” có các thuộc tính cơ bản như sau: mã sách, tên sách, thể loại, tác giả, số lượng, giá tiền.
* Việc phân tích như trên ta có thể thấy việc trùng lặp thuộc tính thể loại của sách nên ta tạo thêm 1 bảng “TheLoai” để lưu trữ thông tin về thể loại sách.
* Một hệ thống thư viện cho mượn sách đều cần phải lưu trữ lại thông tin của đọc giả đến mượn sách nên ở đây ta cũng cần phải có 1 lớp để lưu trữ lại thông tin độc giả : “DocGia”.
* Lớp “DocGia” có các thuộc tính như sau: mã độc giả, tên độc giả, giới tính,số điện thoại, địa chỉ.
* Lớp tiếp theo đó là “Phieu” lớp rất quan trọng trong việc xử lý việc đọc giả mượn trả sách
* Ở đây ta có 2 loại phiếu tương ứng với việc mượn trả sách:phiếu mượn, phiếu trả
* Phiếu mượn: mã phiếu, mã đọc giả, mã sách, số lượng, giá tiền, ngày mượn.
* Phiếu trả: mã phiếu, mã độc giả, mã sách, số lượng, ngày trả.
* Ta thấy được 2 phiếu mượn và trả trùng nhau rất nhiều thuộc tính nên ta ghộp 2 loại phiếu lại để được các thuộc tính của lớp “Phieu” như sau: mã phiếu, mã độc giả, mã sách, số lượng, giá tiền, ngày mượn, ngày trả.
* **Biểu đồ lớp hệ thống :**

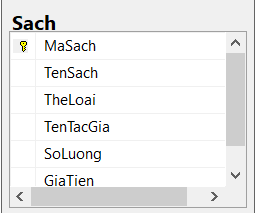


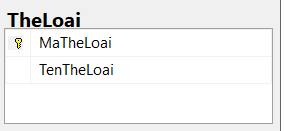
*Hình 2.5 Biểu đồ lớp hệ thống*

#### Thiết kế cơ sở dữ liệu

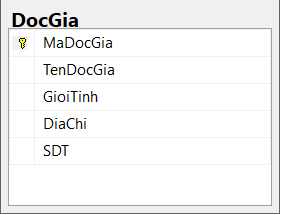
* Từ việc phân tích các lớp như trên thì ta xác định được các lớp có quan hệ với các lớp khác là: Sach, DocGia, TheLoai, Phieu.
* Còn 1 lớp khác là lớp ‘TaiKhoan’ thì lớp này sẽ đứng riêng lẻ vì nó chỉ đảm nhận việc cho phép người quản trị đăng nhập và sử dụng hệ thống.
  1. Bảng “Sach” :

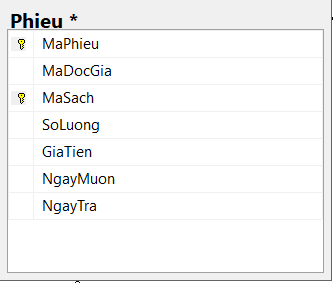
Thuộc tính:

* maSach: String
* tenSach: String
* maTheLoai: String
* tacGia: String
* soLuong: int
* giaTien: int
  1. Bảng “TheLoai”

 Thuộc tính:

* maTheLoai: String
* tenTheLoai: String
  1. Bảng “DocGia”

 Thuộc tính:

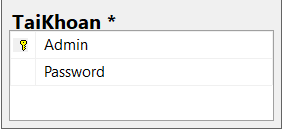
* maDocGia: String
* tenDocGia: String
* gioiTinh:String
* diaChi: String
* sdt: int
  1. Bảng “Phieu”

Thuộc tính:

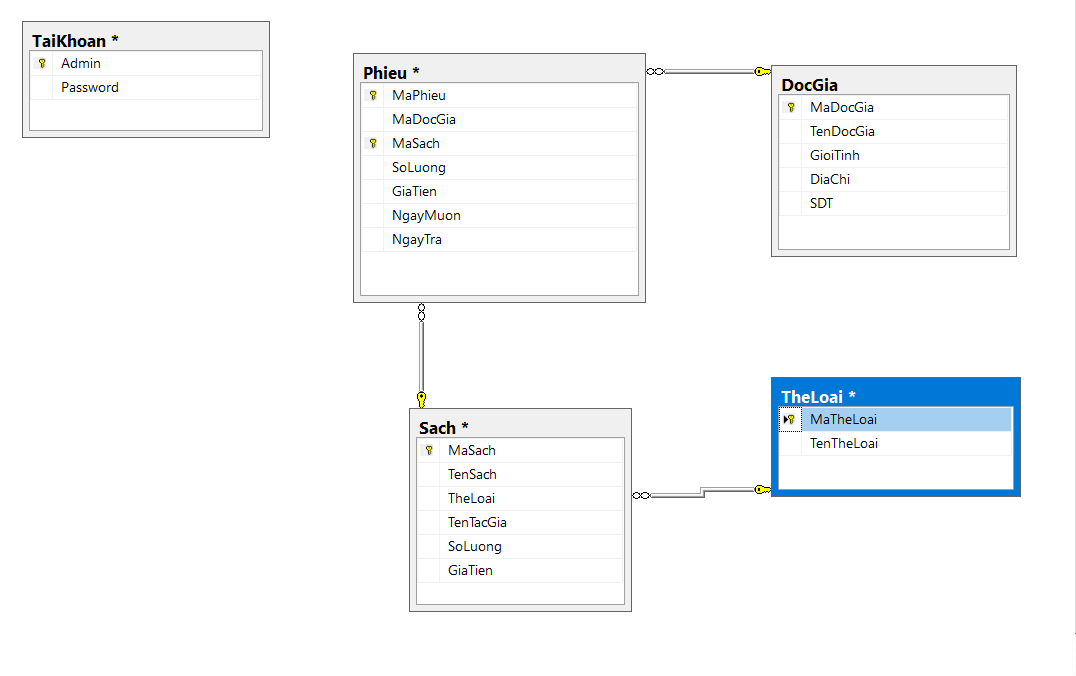
* maPhieu: String
* maDocGia: String
* maSach: String
* soLuong: int
* giaTien: int
* ngayMuon: datetime
* ngayTra: datetime
  1. Bang “TaiKhoan”

Thuộc tính:

* Admin : String
* Password: String



**Sơ đồ quan hệ các bảng :**



*Hình 2.6 Sơ đồ quan hệ giữa các bảng*

# KIỂM THỬ PHẦN MỀM QUẢN LÝ THƯ VIỆN ĐIỆN TỬ VỚI NUNIT

## Lập kế hoạch kiểm thử

Kế hoạch kiểm thử là tài liệu mô tả về phạm vi, cách tiếp cận, các nguồn lực và kế hoạch thực hiện kiểm thử.

* Mục tiêu của lập kế hoạch kiểm thử:
* Thiết lập được mục tiêu ngắn hạn và dài hạn của việc kiểm thử.
* Nhận biết được các rủi ro có thể xảy ra.
* Xác định được cách tiếp cận và kế hoạch cho việc kiểm thử.
* Nhiệm vụ chính của hoạt động lập kế hoạch kiểm thử:
* Xác định được phạm vi kiểm thử, các rủi ro có thể xảy ra, xác định được mục tiêu kiểm thử để đảm bảo đo đạc được chất lượng phần mềm.
* Xác định cách tiếp cận việc kiểm thử, các kĩ thuật được sử dụng, độ bao phủ kiểm thử cho phép.
* Thực thi theo chính sách và chiến lược kiểm thử.
* Xác định nguồn lực, chi phí kiểm thử cần thiết.
* Lên kế hoạch cho hoạt động phân tích, thiết kế, thực thi kiểm thử.
* Xác định tiêu chí kết thúc kiểm thử.
* Liệt kê những kết quả, tài kiệu có được sau khi thực hiện kiểm thử.

### Mục tiêu lập kế hoạch kiểm thử

* Mục tiêu ngắn hạn: Kiểm tra được ngay tính đúng đắn của chức năng trong hệ thống.
* Mục tiêu dài hạn: Đảm bảo chức năng hoạt động chuẩn xác, ổn định theo bản thiết kế với nhiều trường hợp khác nhau.
* Rủi ro có thể xảy ra: Đầu vào của chức năng có thể không phù hợp với tham số của chức năng. Hay đầu ra của chức năng không được như mong muốn tức là có vấn đề trong mã nguồn của chức năng.
* Cách tiếp cận và kế hoạch cho kiểm thử:
* Cách tiếp cận kiểm thử: Kiểm thử từ dưới lên (bottom-up). Vì kiểm thử từ dưới lên tích hợp một số module sau khi được kiểm thử đơn vị thành các thành phần và các thành phần này sẽ được kiểm thử cho đến khi chúng ổn định. Nên cách tiếp này là phù hợp với cách kiểm thử của các chức năng.
* Kế hoạch cho việc kiểm thử: Như đã trình bày ở cách tiếp cận kiểm thử thì việc kiểm thử sẽ thực hiện từ các module nhỏ bên trong chức năng. Sau khi kiểm thử các module nhỏ thì sẽ chuyển sang kiểm thử chức năng chính cho đến khi các chức năng hoạt động ổn định.
* Mục tiêu kiểm thử: Đảm bảo chức sửa thông tin sách hoạt động ổn định, đúng như bản thiết kế.

### Nhiệm vụ của lập kế hoạch kiểm thử

* Xác định phạm vi kiểm thử, các rủi ro có thể xảy ra, xác định được mục tiêu kiểm thử để đảm bảo đo đạc được chất lượng phần mềm.
* Xác định phạm vi kiểm thử: bao gồm việc kiểm tra tên sách,tên tác giả, số lượng, thực hiện trong môi trường và điều kiện khác nhau, chức năng sửa thông tin sách: nó có làm đúng nhiệm vụ của nó hay không, và nó có làm những gì cần phải làm hay không.
* Rủi ro có thể xảy ra: đầu vào không đúng như yêu cầu của chức năng dẫn tới chức năng không thể hoạt động, hay mã nguồn của chức năng không đúng như yêu cầu của bản thiết kế dẫn tới chức năng không hoạt động đúng chức năng của nó.
* Xác định cách tiếp cận việc kiểm thử, các kĩ thuật được sử dụng, độ bao phủ kiểm thử cho phép.
* Cách tiếp cận: Như đã trình bày ở trên thì phương pháp tiếp cận hợp lý nhất với chức năng sửa thông tin sách này là từ dưới lên.
* Các kĩ thuật sử dụng: kĩ thuật đánh giá rủi ro, phân tích rủi ro, xác định cách tiếp cận, xác định mục tiêu…
* Độ bao phủ kiểm thử cho phép: độ bao phủ kiểm thử phải đạt ít nhất 90%.
* Thực thi theo chính sách và chiến lược kiểm thử:
* Thực thi theo chính sách: Đảm bảo nguyên tắc kiểm thử hết với các giá trị biên, giá trị thỏa mãn, mỗi đường thực thi đêu được kiểm thử ít nhất một lần,... và hoàn thành tất cả các mục tiêu nêu trên.
* Chiến lược kiểm thử: Xác định chức năng sẽ được kiểm thử ở mức kiểm thử đơn vị, kiểm thử tích hợp.
* Xác định nguồn lức, chi phí kiểm thử cần thiết: Nguồn lực ở đây chính là người phát triển phần mềm nên không cần thêm bất kỳ chi phí nào.
* Xác định tiêu chí kết thúc kiểm thử: Chức năng hoạt động tốt, ổn định như bản thiết kế ban đầu và yêu cầu của khách hàng.
* **Bảng kế hoạch kiểm thử**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Người thực hiện** | **Công việc** | **Thời gian** | **Ngày** |
| 1 | Phạm Quốc Cường,  Phạm Hải Dương | Thu thập tài liệu, tìm hiểu unit test với NUnit và công cụ hỗ trợ NunitForms | 72h | 08/04/2019 – 11/04/2019 |
| 2 | Phạm Văn Hiểu, Phạm Hải Dương | Phân tích chức năng cần kiểm thử | 48h | 11/04/2019 – 13/04/2019 |
| 3 | Phạm Quốc Cường,  Phạm Văn Hiểu | Phân tích test case, viết test script | 72h | 13/04/2019 – 16/04/2019 |
| 4 | Phạm Văn Hiểu, Phạm Hải Dương | Phân tích kết quả test | 8h | 16/04/2019 – 17/04/2019 |
| 5 | Phạm Quốc Cường,  Phạm Hải Dương | Viết báo cáo | 10h | 17/04/2019 – 18/04/2019 |

## Thực hiện kiểm thử

### Phạm Hải Dương - chức năng Sửa sách (v0.1)

#### Phân tích thiết kế kiểm thử

* Giao diện chức năng:



Hình 3.1: Giao diện chức năng Sửa sách

* Mã nguồn chức năng:

private void btnSuaSach\_Click(object sender, EventArgs e)

{

1

if ( txtTenSach.Text.Equals(""))

2

MessageBox.Show("UNSUCCESS", "NOTE");

322

else if (txtTacGia.TextLength == 0)

422

MessageBox.Show("UNSUCCESS", "NOTE");

522

else if (txtSoLuong.TextLength == 0)

622

MessageBox.Show("UNSUCCESS", "NOTE");

else

{

try

{

int sl = int.Parse(txtSoLuong.Text);

7

Sach sach = new Sach(txtMaSach.Text, txtTenSach.Text, cmbTheLoai.SelectedValue.ToString(),sltxtTacGia.Text);

SachBUS.Instance.UpdateBook(sach);

QuanLySach\_Load(sender, e);

MessageBox.Show("SUCCESS", "NOTE");

}

catch (FormatException)

{

822

MessageBox.Show("UNSUCCESS", "NOTE");

}

}

}

* Đồ thị giải thuật:



Hình 3.2. Đồ thị lưu trình chức năng sửa sách

* Độ phức tạp của chu trình: C = (4 + 1) = (11-8+2) = 5
* 1) 1
* 2) 1 - 2 - 3
* 3) 1 - 2 - 3 - 4 - 5
* 4) 1 - 2 - 3 - 4 – 5 – 6 -7
* 5) 1 - 2 - 3 - 4 – 5 – 6 – 7 - 8
* Thiết kế testcase:

1. Mô tả điều kiện:

- Tất cả các trường phải được nhập đủ

- Tên sách: 1-100 kí tự

- Tác giả: 1 - 50 kí tự

- Số lượng: chỉ được nhập vào số nguyên dương

1. Thiết kế testcase

* Áp dụng phương pháp phân vùng tương đương và phân tích giá trị biên:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Đầu vào** | **Vùng hợp lệ** | **Đánh dấu** | **Vùng không hợp lệ** | **Đánh dấu** |
| Tên sách | [1-100] | V1 | <1 | I1 |
| >100 | I2 |
| Bỏ trống | I3 |
| Tác giả | [1-50] | V2 | <1 | I4 |
| >50 | I5 |
| Bỏ trống | I6 |
| Số lượng | [0-1000] | V3 | <0 | I7 |
| Nhập sai định dạng | I8 |
| Bỏ trống | I9 |
| >1000 | I10 |

Bảng 3.1. Bảng phân vùng tương đương chức năng sửa sách



Hình 3.3. Ca kiểm thử Sửa thông tin sách



Hình 3.4 Datatess sau khi phân tích testcase

#### Thực hiện kiểm thử

B1. Mở project-> tạo 1 class libraly tên Test

Chọn Add Reference và thêm các reference phù hợp

B2. Tạo 2 class với các chức năng tương ứng là đọc ghi file excel

và class chúa phương thức test trong Class libraly vừa tạo

+ Class để đọc ghi file excel

using System;

using System.Collections.Generic;

using Excel = Microsoft.Office.Interop.Excel;

namespace Test

{

//class để đọc ghi file excel

class DocGhi

{

//khởi tại 1 ứng dụng Excel

public static Excel.\_Application App;

//khởi tạo bảng tính excel

public static Excel.\_Workbook Workbook;

//khởi tạo các sheets trong workbook

public static Excel.\_Worksheet Worksheet;

//quản lý các thành phần trong sheets

public static Excel.Range Range;

//hàm để mở file excel

public static void open(String fileName)

{

//tạo đối tượng COM mở kết nối với các đối tượng cần dùng để đọc dữ liệu

App = new Excel.Application();

Workbook = App.Workbooks.Open(fileName);

//chọn sheets đầu tiên

Worksheet = Workbook.Sheets[1];

Range = Worksheet.UsedRange;

}

//phương thức tạo 1 danh sách mảng đối tượng 1

//chiều mỗi 1 mảng là 1 dòng trong bảng tính Excel

public static List<Object[]> getExcelFile(String fileName)

{

//mở file

open(fileName);

//đếm số dòng dữ liệu

int rowCount = Range.Rows.Count;

//khai báo số cột cần lấy giá trị

int colCount = 4;

//cột bắt đầu đọc dữ liệu từ cột 2 đến cột 4

int j = 2;

//khởi tạo 1 danh sách mảng đối tượng

List<Object[]> list = new List<Object[]>();

String ten = "";

int sl = 1;

String tg = "";

String mess = "";

//đọc từng thành phần trong excel

for (int i = 2; i <= 13; i++)

{

try

{

ten = Range.Cells[i, j++].Value2.ToString();

}

catch (Exception)

{

ten = "";

}

try

{

sl = Convert.ToInt32(Range.Cells[i, j++].Value2.ToString());

}

catch (Exception)

{

sl = 0;

}

try

{

tg = Range.Cells[i, j++].Value2.ToString();

}

catch (Exception)

{

tg = "";

}

try

{

mess = Range.Cells[i, j++].Value2.ToString();

}

catch (Exception)

{

mess = null;

}

if (j > colCount) { j = 2; }

//thêm dữ liệu vào mảng ta chỉ đọc đến cột số 4 nếu lớn hơn quay lại cột bạn đầu

Object[] obj = new Object[] { ten, tg, sl, mess };

list.Add(obj);

}

//đóng kết nối

Workbook.Close();

return list;

}

//phương thức dùng để ghi dữ liệu vào file excel

public static void setExcelFile(int i, int j, String str, String fileName)

{

open(fileName);

Range.Cells[i, j].value2 = str;

Workbook.Save();

Workbook.Close();

}

}

}

+Class chứa phương thức test

using QuanLyThuVien;

using System;

using NUnit.Extensions.Forms;

using NUnit.Framework;

using System.Collections.Generic;

namespace Test

{

[TestFixture]

public class TestThemSach

{

//dòng bắt đầu ghi dữ liệu

public int row = 2;

//hàm lấy datatest từ file excel

public static List<Object[]> data()

{

return DocGhi.getExcelFile(@"C:\Users\HP\Documents\TCs\_Sua1..xlsx");

}

//lấy dữ liệu đẩy vào testcase

[TestCaseSource("data"), Test]

public void SuaRound1(String ten, String tg, int sl, String mess)

{

//khởi tạo đối tượng của form QuanLySach

QuanLySach qls = new QuanLySach();

//Hiện form

qls.Show();

//khởi tạo 1 biến để lấy kết quả từ messagebox

string actmsg = "";

ModalFormTester msgBox = new ModalFormTester();

msgBox.ExpectModal("NOTE", delegate

{

MessageBoxTester ms = new MessageBoxTester("NOTE");

actmsg = ms.Text;

ms.ClickOk();

});

// set giá trị cho các trường sau khi lấy dữ liệu từ excel

qls.dataTest(ten, tg,sl);

//khởi tạo 1 button

ButtonTester btnSua = new ButtonTester("btnSuaSach");

//click vào button Sửa trên form

btnSua[0].Click();

//so sách kết quả lấy từ messagebox với kết quả mong đợi

if (mess.Equals(actmsg))

{

//ghi kết quả vào file excel

DocGhi.setExcelFile(row++, 6, "Pass", @"C:\Users\HP\Documents\TCs\_Sua1..xlsx");

Assert.AreEqual(actmsg, mess);

}

else

{

DocGhi.setExcelFile(row++, 6, "Pending", @"C:\Users\HP\Documents\TCs\_Sua1..xlsx");

Assert.AreEqual(actmsg, mess);

}

//đóng form sau khi thực hiện xong

qls.Close();

}

}

}

B3. Mở App Nunit chọn OpenProject và chọn đến file Test.dll, chọn Run để bắt đầu chạy

#### Báo cáo kết quả kiểm thử

Đã test : 100% (12 Testcase)

Thành công: 41.6% (5 Testcase)

Pending 58,4 % (7Testcase)

### Phạm Hải Dương - chức năng Sửa sách (v0.2)

#### Phân tích thiết kế kiểm thử

* Giao diện chức năng:



Hình 3.1b: Giao diện chức năng Sửa sách

* Mã nguồn chức năng:



--Vẫn chưa sửa được bắt lỗi ký tự đặc biệt

* Đồ thị giải thuật:



Hình 3.2b. Đồ thị lưu trình chức năng sửa sách

* Độ phức tạp của chu trình: C = (4 + 1) = (11-8+2) = 5
* 1) 1
* 2) 1 - 2 - 3
* 3) 1 - 2 - 3 - 4 - 5
* 4) 1 - 2 - 3 - 4 – 5 – 6 -7
* 5) 1 - 2 - 3 - 4 – 5 – 6 – 7 - 8
* Thiết kế testcase:

1. Mô tả điều kiện:

- Tất cả các trường phải được nhập đủ

- Tên sách: 1-100 kí tự

- Tác giả: 1 - 50 kí tự

- Số lượng: chỉ được nhập vào số nguyên dương

1. Thiết kế testcase

* Áp dụng phương pháp phân vùng tương đương và phân tích giá trị biên:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Đầu vào** | **Vùng hợp lệ** | **Đánh dấu** | **Vùng không hợp lệ** | **Đánh dấu** |
| Tên sách | [1-100] | V1 | <1 | I1 |
| >100 | I2 |
| Bỏ trống | I3 |
| Tác giả | [1-50] | V2 | <1 | I4 |
| >50 | I5 |
| Bỏ trống | I6 |
| Số lượng | [0-1000] | V3 | <0 | I7 |
| Nhập sai định dạng | I8 |
| Bỏ trống | I9 |
| >1000 | I10 |

Bảng 3.1. Bảng phân vùng tương đương chức năng sửa sách



Hình 3.3. Ca kiểm thử Sửa thông tin sách



Hình 3.4 Datatess sau khi phân tích testcase

#### Thực hiện kiểm thử

B1. Mở project-> tạo 1 class libraly tên Test

Chọn Add Reference và thêm các reference phù hợp

B2. Tạo 2 class với các chức năng tương ứng là đọc ghi file excel

và class chúa phương thức test trong Class libraly vừa tạo

+ Class để đọc ghi file excel

using System;

using System.Collections.Generic;

using Excel = Microsoft.Office.Interop.Excel;

namespace Test

{

//class để đọc ghi file excel

class DocGhi

{

//khởi tại 1 ứng dụng Excel

public static Excel.\_Application App;

//khởi tạo bảng tính excel

public static Excel.\_Workbook Workbook;

//khởi tạo các sheets trong workbook

public static Excel.\_Worksheet Worksheet;

//quản lý các thành phần trong sheets

public static Excel.Range Range;

//hàm để mở file excel

public static void open(String fileName)

{

//tạo đối tượng COM mở kết nối với các đối tượng cần dùng để đọc dữ liệu

App = new Excel.Application();

Workbook = App.Workbooks.Open(fileName);

//chọn sheets đầu tiên

Worksheet = Workbook.Sheets[1];

Range = Worksheet.UsedRange;

}

//phương thức tạo 1 danh sách mảng đối tượng 1

//chiều mỗi 1 mảng là 1 dòng trong bảng tính Excel

public static List<Object[]> getExcelFile(String fileName)

{

//mở file

open(fileName);

//đếm số dòng dữ liệu

int rowCount = Range.Rows.Count;

//khai báo số cột cần lấy giá trị

int colCount = 4;

//cột bắt đầu đọc dữ liệu từ cột 2 đến cột 4

int j = 2;

//khởi tạo 1 danh sách mảng đối tượng

List<Object[]> list = new List<Object[]>();

String ten = "";

int sl = 1;

String tg = "";

String mess = "";

//đọc từng thành phần trong excel

for (int i = 2; i <= 13; i++)

{

try

{

ten = Range.Cells[i, j++].Value2.ToString();

}

catch (Exception)

{

ten = "";

}

try

{

sl = Convert.ToInt32(Range.Cells[i, j++].Value2.ToString());

}

catch (Exception)

{

sl = 0;

}

try

{

tg = Range.Cells[i, j++].Value2.ToString();

}

catch (Exception)

{

tg = "";

}

try

{

mess = Range.Cells[i, j++].Value2.ToString();

}

catch (Exception)

{

mess = null;

}

if (j > colCount) { j = 2; }

//thêm dữ liệu vào mảng ta chỉ đọc đến cột số 4 nếu lớn hơn quay lại cột bạn đầu

Object[] obj = new Object[] { ten, tg, sl, mess };

list.Add(obj);

}

//đóng kết nối

Workbook.Close();

return list;

}

//phương thức dùng để ghi dữ liệu vào file excel

public static void setExcelFile(int i, int j, String str, String fileName)

{

open(fileName);

Range.Cells[i, j].value2 = str;

Workbook.Save();

Workbook.Close();

}

}

}

+Class chứa phương thức test

using QuanLyThuVien;

using System;

using NUnit.Extensions.Forms;

using NUnit.Framework;

using System.Collections.Generic;

namespace Test

{

[TestFixture]

public class TestThemSach

{

//dòng bắt đầu ghi dữ liệu

public int row = 2;

//hàm lấy datatest từ file excel

public static List<Object[]> data()

{

return DocGhi.getExcelFile(@"C:\Users\HP\Documents\TCs\_Sua1..xlsx");

}

//lấy dữ liệu đẩy vào testcase

[TestCaseSource("data"), Test]

public void SuaRound1(String ten, String tg, int sl, String mess)

{

//khởi tạo đối tượng của form QuanLySach

QuanLySach qls = new QuanLySach();

//Hiện form

qls.Show();

//khởi tạo 1 biến để lấy kết quả từ messagebox

string actmsg = "";

ModalFormTester msgBox = new ModalFormTester();

msgBox.ExpectModal("NOTE", delegate

{

MessageBoxTester ms = new MessageBoxTester("NOTE");

actmsg = ms.Text;

ms.ClickOk();

});

// set giá trị cho các trường sau khi lấy dữ liệu từ excel

qls.dataTest(ten, tg,sl);

//khởi tạo 1 button

ButtonTester btnSua = new ButtonTester("btnSuaSach");

//click vào button Sửa trên form

btnSua[0].Click();

//so sách kết quả lấy từ messagebox với kết quả mong đợi

if (mess.Equals(actmsg))

{

//ghi kết quả vào file excel

DocGhi.setExcelFile(row++, 6, "Pass", @"C:\Users\HP\Documents\TCs\_Sua1..xlsx");

Assert.AreEqual(actmsg, mess);

}

else

{

DocGhi.setExcelFile(row++, 6, "Pending", @"C:\Users\HP\Documents\TCs\_Sua1..xlsx");

Assert.AreEqual(actmsg, mess);

}

//đóng form sau khi thực hiện xong

qls.Close();

}

}

}

B3. Mở App Nunit chọn OpenProject và chọn đến file Test.dll, chọn Run để bắt đầu chạy

#### Báo cáo kết quả kiểm thử

Đã test : 100% (12 Testcase)

Thành công: 83.3% (10 Testcase)

Pending 16,7 % (2 Testcase)

### Phạm Văn Hiểu - chức năng thêm sách (V1)

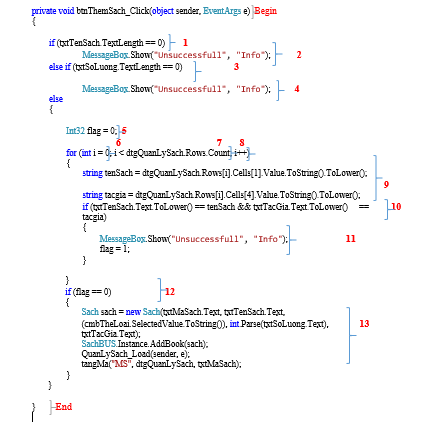
#### Phân tích thiết kế kiểm thử

* Giao diện chức năng :



*Hình 3.5 Chức năng Quản lý sách*

* Mã nguồn của chức năng



* Lưu đồ giải thuật hoặc sơ đồ đồ thị dòng
* Lưu đồ giải thuật



*Hình 3.6 Lưu đồ giải thuật*

* Độ phức tập của chu trình

C =5 + 1 = (17-13)+2=6

1. 1-2
2. 1-3-4
3. 1-3-5-6-7-8-9-11-7
4. 1-3-5-6-7-8-9-10
5. 1-3-5-6-7-12-13
6. 1-3-5-6-7-12

* Thiết kế test case :
  + Mô tả điều kiện :

**+** Trường tên sách

* Phải phải nhập
* 1 đến 255 kí tự

**+** Trường số lượng

* Phải phải nhập
* 1 đến 5 kí tự số

**+** Trường tác giả

* Phải phải nhập
* 1 đến 255 kí tự

**+** Trường thể loại

* Phải chọn
  + Xây dựng test case :
* Áp dụng phương pháp phân tích giá trị biên, phân vùng tương đương.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Đầu vào | Vùng hợp lệ | Ký hiệu đánh dấu | Vùng không hợp lệ | Đánh dấu |
| Tên sách | [1-50] | H1 | Trống | K1 |
| >50 | K2 |
| Chứa kí tự đặc biệt | K3 |
| Số lượng | [1-5] | H2 | Trống | K4 |
| >5 | K5 |
| Sai định dạng | K6 |
| Thể loại | selected | H3 | Trống | K7 |
| Tác giả | [1-50] | H4 | Trống | K8 |
| >50 | K9 |
| Chứa kí tự đặc biệt | K10 |

*Bảng 3.2 Bảng phân vùng tương đương.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TCs | Input | | | | Message | Cover |
| Tên sách | Số lượng | Thể loại | Tác giả |
| 0 | Buoc Duong Cung | 5 | Van Hoc | Nam Cao | Successfull | H1, H2, H3,H4 |
| 1 |  | 5 | Van Hoc | Nam Cao | Unsuccessfull | K1, H2, H3,H4 |
| 2 | So Do |  | Van Hoc | Vu Trong Phung | Unsuccessfull | H1, K3, H3,H4 |
| 3 | Truyen Kieu | 4 |  | Nguyen Du | Unsuccessfull | H1, H2, K5,H4 |
| 4 | Tuoi Tho Du Doi | 2 | Van Hoc |  | Unsuccessfull | H1, H2, K5,K6 |
| 5 | Hai Dua Tre | 123456 | Van Hoc | Victor Hugo | Unsuccessfull | H1, K4, H3,H4 |
| 6 | aaaaaaaaaaaa..aa | 3 | Van Hoc | Nguyen Trai | Unsuccessfull | K2, H2, H3,H4 |
| 7 | Tieng goi noi hoang da | 5 | Van Hoc | aaaaaaa..aa | Unsuccessfull | H1, H2, H3,K7 |
| 8 | Khong gia dinh | aaa | Van Hoc | Khong nho | Unsuccessfull | H1, K8, H3,H4 |
| 9 | Khong@ gia dinh | 5 | Van Hoc | Khong nho | Unsuccessfull | K3, H2, H3,H4 |
| 10 | Ki nguyen | 6 | Van Hoc | M@vel | Unsuccessfull | H1, H2, H3,K10 |

*Hình 3.7 Ca kiểm thử.*

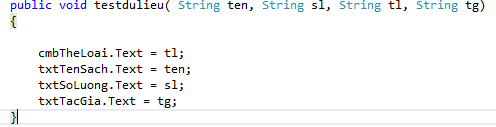
* Testcase thêm thông tin sách

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TCs | Tên test case | Input | | | | Message |
| Tên sách | Số lượng | Thể loại | Tác giả |
| 0 | Thêm sách | Buoc Duong Cung | 5 | Van Hoc | Nam Cao | Successfull |
| 1 | Thêm sách |  | 5 | Van Hoc | Nam Cao | Unsuccessfull |
| 2 | Thêm sách | So Do |  | Van Hoc | Vu Trong Phung | Unsuccessfull |
| 3 | Thêm sách | Truyen Kieu | 4 |  | Nguyen Du | Unsuccessfull |
| 4 | Thêm sách | Tuoi Tho Du Doi | 2 | Van Hoc |  | Unsuccessfull |
| 5 | Thêm sách | Hai Dua Tre | 123456 | Van Hoc | Victor Hugo | Unsuccessfull |
| 6 | Thêm sách | aaaaaaaaaaaa..aa | 3 | Van Hoc | Nguyen Trai | Unsuccessfull |
| 7 | Thêm sách | Tieng goi noi hoang da | 5 | Van Hoc | aaaaaaa..aa | Unsuccessfull |
| 8 | Thêm sách | Khong gia dinh | aaa | Van Hoc | Khong nho | Unsuccessfull |
| 9 | Thêm sách | Khong@ gia dinh | 5 | Van Hoc | Khong nho | Unsuccessfull |
| 10 | Thêm sách | Ki nguyen | 6 | Van Hoc | M@vel | Unsuccessfull |

*Hình 3.8 Datatest sau khi phân tích test case.*

#### Thực hiện kiểm thử

* Bước 1: Tạo một ClassLibrary với tên “PhamVanHieu” trong Solution “QuanLyThuVien”.
* Bước 2: Thêm các framework: Chuột phải “PhamVanHieu” 🡪 Add Reference 🡪 Chọn các framework sau:
  + NUnit.Extensions.Forms
  + NUnit.Framework
  + System.Windows.Forms
  + Microsoft.Office.Interop.Excel
  + System.Runtime.InteropServices
* Bước 3:
  + Tạo hàm testdulieu() trong file QuanLySach.cs để đưa dữ liệu các Testcases vào ô TextBox



* + Tạo class DocGhiFile.cs để đọc dữ liệu từ file Excel

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

using System.IO;

using System.Runtime.InteropServices;//thư viện hỗ trợ đọc ghi file excel

using Excel = Microsoft.Office.Interop.Excel;// Đối tượng thay thế cho tên thư viện

using System.Collections.ObjectModel;

using System.Collections;

using DTO;

namespace PhamVanHieu

{

class DocGhiFile

{

public static Excel.\_Application App;

public static Excel.\_Workbook Workbook; //Workbook cho file excel

public static Excel.\_Worksheet Worksheet;//Quản lý các sheet

public static Excel.Range Range;//Quản các thành phần trong sheet

public static void open(String fileName)

{

//tạo các đối tượng để đọc dữ liệu

App = new Excel.Application();

Workbook = App.Workbooks.Open(fileName);

Worksheet = Workbook.Sheets[1];

Range = Worksheet.UsedRange;

}

//Hàm lấy dữ liệu từ file excel

public static List<Object[]> getExcelFile(String fileName)

{

//Mở file excel

open(fileName);

int rowCount = Range.Rows.Count;

int colCount = 6;

//Đọc dữ liệu từ cột 2 đến cột 6

int j = 2;

List<Object[]> list = new List<Object[]>();

String ten = "1";

String sl = "1";

String tl = "1";

String tg = "1";

String mess = "1";

for (int i = 2; i <= rowCount; i++)

{

//Đọc thông tin từng ô trên 1 dòng

try

{

ten = Range.Cells[i, j++].Value2.ToString();

}

catch (Exception)

{

ten = null;

}

try

{

sl = Range.Cells[i, j++].Value2.ToString();

}

catch (Exception)

{

sl = null;

}

try

{

tl = Range.Cells[i, j++].Value2.ToString();

}

catch (Exception)

{

tl = null;

}

try

{

tg = Range.Cells[i, j++].Value2.ToString();

}

catch (Exception)

{

tg = null;

}

try

{

mess = Range.Cells[i, j++].Value2.ToString();

}

catch (Exception)

{

mess = null;

}

//xuống dòng

if (j > colCount) { j = 2; }

//tạo đối tượng lưu thông tin

Object[] obj = new Object[] {ten, sl, tl, tg, mess };

//thêm đối tượng vào list

list.Add(obj);

}

//Đóng file excel

Workbook.Close();

return list;

}

//Ghi dữ liệu vào một ô trong file excel

public static void setExcelFile(int i, int j, String str, String fileName)

{

open(fileName);

Range.Cells[i, j].value2 = str;

Workbook.Save();

Workbook.Close();

}

}

}

* + Tạo file TestThemSach.cs để thực hiện việc test tự động

using QuanLyThuVien;

using System;

using NUnit.Extensions.Forms; //Sử dụng để viết test, nhất thiết phải có

using NUnit.Framework;//Cần có để sử dụng các annotation trong NUnit

using System.Windows.Forms;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using DTO;

namespace PhamVanHieu

{

[TestFixture]

public class TestThemSach

{

//Khởi tạo dòng bắt đầu dọc trong file excel

public int row = 2;

public static List<Object[]> data()

{

//lấy dữ liệu từ file excel

return DocGhiFile.getExcelFile(@"C:\Users\phamv\Documents\HomeWork\Kiem thu phan mem\Kiem-Thu-Phan-Mem\Kiem-Thu-Phan-Mem\dulieutest.xlsx");

}

[TestCaseSource("data"), Test]//sử dụng dữ liệu từ hàm data

public void testRound1(String ten, String sl, String tl, String tg, String mess)

{

//Khởi tạo form để thêm sách

QuanLySach ds = new QuanLySach();

ds.Show();

//thông tin so sánh

String actmsg = "";

//Tạo messagebox hiển thị thông báo

ModalFormTester msgBox = new ModalFormTester();

msgBox.ExpectModal("Info", delegate

{

MessageBoxTester msg = new MessageBoxTester("Info");

actmsg = msg.Text;

msg.ClickOk();

});

//Hàm thêm dữ liệu test vào form

ds.testdulieu(ten, sl, tl, tg);

ButtonTester btnthem = new ButtonTester("btnThemSach");

btnthem[0].Click();//kích để thêm dữ liệu

if (mess.Equals(actmsg))

{

//Ghi dữ iệu vào file excel

DocGhiFile.setExcelFile(row++, 7, "Pass", @"C:\Users\phamv\Documents\HomeWork\Kiem thu phan mem\Kiem-Thu-Phan-Mem\Kiem-Thu-Phan-Mem\dulieutest.xlsx");

//Kiểm tra tescase đúng sai

Assert.AreEqual( mess,actmsg);

}

else

{

//Ghi dữ iệu vào file excel

DocGhiFile.setExcelFile(row++, 7, "False", @"C:\Users\phamv\Documents\HomeWork\Kiem thu phan mem\Kiem-Thu-Phan-Mem\Kiem-Thu-Phan-Mem\dulieutest.xlsx");

//Kiểm tra tescase đúng sai

Assert.AreEqual( mess,actmsg);

}

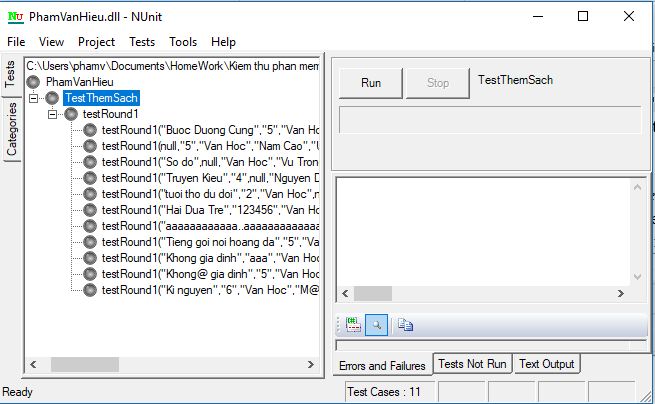
ds.Close();

}

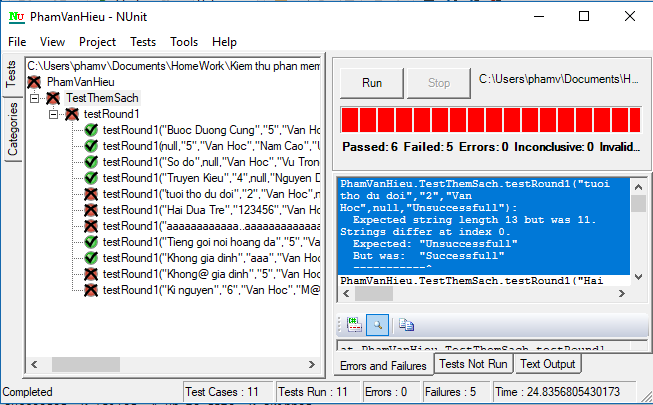
}

}

* Bước 4: Run test
  + Trên thanh công cụ chọn BUILD 🡪 Build Solution hoặc Rebuild Solution
  + Mở Nunit.exe
  + Chọn File 🡪 Open Project 🡪File Quản Lý Thư Viện 🡪 PhamVanHieu🡪 bin 🡪 Debug 🡪 PhamVanHieu.dll
  + Chọn thành công thi giao diện Nunit sẽ như sau:



* Click Run và chờ đợi kết quả.
* Kết quả Test đợt 1:



PASS: 6/11 (54.5%)

PENDING: 5/11 (45.5%)

FAIL: 0/11 (0%)

* Kết quả lưu trong file Excel:



### Phạm Văn Hiểu - chức năng thêm sách (V2)

#### Các lỗi đang chờ được sửa

1. Chưa kiểm tra số ký tự tối đa của tên sách
2. Chưa bắt lỗi tên sách chứa ký tự đặc biệt
3. Chưa bắt lỗi tên tác giả bỏ trống
4. Chưa kiểm tra số ký tự tối đa của tác giả
5. Chưa bắt lỗi tác giảchứa ký tự đặc biệt

#### Phân tích thiết kế kiểm thử

* Mã nguồn chức năng



* Lưu đồ giải thuật



* Độ phức tập của chu trình

C =5 + 1=(17-13)+2 =6

1. 1-2
2. 1-3-4
3. 1-3-5-6-7-8-9-11-7
4. 1-3-5-6-7-8-9-10
5. 1-3-5-6-7-12-13
6. 1-3-5-6-7-12

* Thiết kế test case :
  + Mô tả điều kiện :

**+** Trường tên sách

* Phải phải nhập
* 1 đến 50 kí tự
* Không chứa kí tự đặc biệt

**+** Trường số lượng

* Phải phải nhập
* 1 đến 5 kí tự số

**+** Trường tác giả

* Phải phải nhập
* 1 đến 50 kí tự
* Không chứa kí tự đặc biệt

**+** Trường thể loại

* Phải chọn
  + Xây dựng test case :
* Áp dụng phương pháp phân tích giá trị biên, phân vùng tương đương.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Đầu vào | Vùng hợp lệ | Ký hiệu đánh dấu | Vùng không hợp lệ | Đánh dấu |
| Tên sách | [1-50] | H1 | Trống | K1 |
| >50 | K2 |
| Chứa kí tự đặc biệt | K3 |
| Số lượng | [1-5] | H2 | Trống | K4 |
| >5 | K5 |
| Sai định dạng | K6 |
| Thể loại | selected | H3 | Trống | K7 |
| Tác giả | [1-50] | H4 | Trống | K8 |
| >50 | K9 |
| Chứa kí tự đặc biệt | K10 |

*Bảng 3.2 Bảng phân vùng tương đương.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TCs | Input | | | | Message | Cover |
| Tên sách | Số lượng | Thể loại | Tác giả |
| 0 | Buoc Duong Cung | 5 | Van Hoc | Nam Cao | Successfull | H1, H2, H3,H4 |
| 1 |  | 5 | Van Hoc | Nam Cao | Unsuccessfull | K1, H2, H3,H4 |
| 2 | So Do |  | Van Hoc | Vu Trong Phung | Unsuccessfull | H1, K3, H3,H4 |
| 3 | Truyen Kieu | 4 |  | Nguyen Du | Unsuccessfull | H1, H2, K5,H4 |
| 4 | Tuoi Tho Du Doi | 2 | Van Hoc |  | Unsuccessfull | H1, H2, K5,K6 |
| 5 | Hai Dua Tre | 123456 | Van Hoc | Victor Hugo | Unsuccessfull | H1, K4, H3,H4 |
| 6 | aaaaaaaaaaaa..aa | 3 | Van Hoc | Nguyen Trai | Unsuccessfull | K2, H2, H3,H4 |
| 7 | Tieng goi noi hoang da | 5 | Van Hoc | aaaaaaa..aa | Unsuccessfull | H1, H2, H3,K7 |
| 8 | Khong gia dinh | aaa | Van Hoc | Khong nho | Unsuccessfull | H1, K8, H3,H4 |
| 9 | Khong@ gia dinh | 5 | Van Hoc | Khong nho | Unsuccessfull | K3, H2, H3,H4 |
| 10 | Ki nguyen | 6 | Van Hoc | M@vel | Unsuccessfull | H1, H2, H3,K10 |

*Hình 3.7 Ca kiểm thử.*

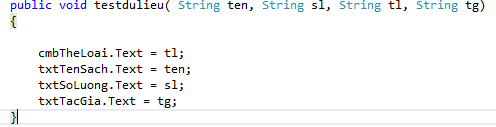
* Testcase thêm thông tin sách

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TCs | Tên test case | Input | | | | Message |
| Tên sách | Số lượng | Thể loại | Tác giả |
| 0 | Thêm sách | Buoc Duong Cung | 5 | Van Hoc | Nam Cao | Successfull |
| 1 | Thêm sách |  | 5 | Van Hoc | Nam Cao | Unsuccessfull |
| 2 | Thêm sách | So Do |  | Van Hoc | Vu Trong Phung | Unsuccessfull |
| 3 | Thêm sách | Truyen Kieu | 4 |  | Nguyen Du | Unsuccessfull |
| 4 | Thêm sách | Tuoi Tho Du Doi | 2 | Van Hoc |  | Unsuccessfull |
| 5 | Thêm sách | Hai Dua Tre | 123456 | Van Hoc | Victor Hugo | Unsuccessfull |
| 6 | Thêm sách | aaaaaaaaaaaa..aa | 3 | Van Hoc | Nguyen Trai | Unsuccessfull |
| 7 | Thêm sách | Tieng goi noi hoang da | 5 | Van Hoc | aaaaaaa..aa | Unsuccessfull |
| 8 | Thêm sách | Khong gia dinh | aaa | Van Hoc | Khong nho | Unsuccessfull |
| 9 | Thêm sách | Khong@ gia dinh | 5 | Van Hoc | Khong nho | Unsuccessfull |
| 10 | Thêm sách | Ki nguyen | 6 | Van Hoc | M@vel | Unsuccessfull |

*Hình 3.8 Datatest sau khi phân tích test case.*

#### Thực hiện kiểm thử

* Bước 1: Tạo một ClassLibrary với tên “PhamVanHieu” trong Solution “QuanLyThuVien”.
* Bước 2: Thêm các framework: Chuột phải “PhamVanHieu” 🡪 Add Reference 🡪 Chọn các framework sau:
  + NUnit.Extensions.Forms
  + NUnit.Framework
  + System.Windows.Forms
  + Microsoft.Office.Interop.Excel
  + System.Runtime.InteropServices
* Bước 3:
  + Tạo hàm testdulieu() trong file QuanLySach.cs để đưa dữ liệu các Testcases vào ô TextBox



* + Tạo class DocGhiFile.cs để đọc dữ liệu từ file Excel

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

using System.IO;

using System.Runtime.InteropServices;//thư viện hỗ trợ đọc ghi file excel

using Excel = Microsoft.Office.Interop.Excel;// Đối tượng thay thế cho tên thư viện

using System.Collections.ObjectModel;

using System.Collections;

using DTO;

namespace PhamVanHieu

{

class DocGhiFile

{

public static Excel.\_Application App;

public static Excel.\_Workbook Workbook; //Workbook cho file excel

public static Excel.\_Worksheet Worksheet;//Quản lý các sheet

public static Excel.Range Range;//Quản các thành phần trong sheet

public static void open(String fileName)

{

//tạo các đối tượng để đọc dữ liệu

App = new Excel.Application();

Workbook = App.Workbooks.Open(fileName);

Worksheet = Workbook.Sheets[1];

Range = Worksheet.UsedRange;

}

//Hàm lấy dữ liệu từ file excel

public static List<Object[]> getExcelFile(String fileName)

{

//Mở file excel

open(fileName);

int rowCount = Range.Rows.Count;

int colCount = 6;

//Đọc dữ liệu từ cột 2 đến cột 6

int j = 2;

List<Object[]> list = new List<Object[]>();

String ten = "1";

String sl = "1";

String tl = "1";

String tg = "1";

String mess = "1";

for (int i = 2; i <= rowCount; i++)

{

//Đọc thông tin từng ô trên 1 dòng

try

{

ten = Range.Cells[i, j++].Value2.ToString();

}

catch (Exception)

{

ten = null;

}

try

{

sl = Range.Cells[i, j++].Value2.ToString();

}

catch (Exception)

{

sl = null;

}

try

{

tl = Range.Cells[i, j++].Value2.ToString();

}

catch (Exception)

{

tl = null;

}

try

{

tg = Range.Cells[i, j++].Value2.ToString();

}

catch (Exception)

{

tg = null;

}

try

{

mess = Range.Cells[i, j++].Value2.ToString();

}

catch (Exception)

{

mess = null;

}

//xuống dòng

if (j > colCount) { j = 2; }

//tạo đối tượng lưu thông tin

Object[] obj = new Object[] {ten, sl, tl, tg, mess };

//thêm đối tượng vào list

list.Add(obj);

}

//Đóng file excel

Workbook.Close();

return list;

}

//Ghi dữ liệu vào một ô trong file excel

public static void setExcelFile(int i, int j, String str, String fileName)

{

open(fileName);

Range.Cells[i, j].value2 = str;

Workbook.Save();

Workbook.Close();

}

}

}

* + Tạo file TestThemSach.cs để thực hiện việc test tự động

using QuanLyThuVien;

using System;

using NUnit.Extensions.Forms; //Sử dụng để viết test, nhất thiết phải có

using NUnit.Framework;//Cần có để sử dụng các annotation trong NUnit

using System.Windows.Forms;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using DTO;

namespace PhamVanHieu

{

[TestFixture]

public class TestThemSach

{

//Khởi tạo dòng bắt đầu dọc trong file excel

public int row = 2;

public static List<Object[]> data()

{

//lấy dữ liệu từ file excel

return DocGhiFile.getExcelFile(@"C:\Users\phamv\Documents\HomeWork\Kiem thu phan mem\Kiem-Thu-Phan-Mem\Kiem-Thu-Phan-Mem\dulieutest.xlsx");

}

[TestCaseSource("data"), Test]//sử dụng dữ liệu từ hàm data

public void testRound1(String ten, String sl, String tl, String tg, String mess)

{

//Khởi tạo form để thêm sách

QuanLySach ds = new QuanLySach();

ds.Show();

//thông tin so sánh

String actmsg = "";

//Tạo messagebox hiển thị thông báo

ModalFormTester msgBox = new ModalFormTester();

msgBox.ExpectModal("Info", delegate

{

MessageBoxTester msg = new MessageBoxTester("Info");

actmsg = msg.Text;

msg.ClickOk();

});

//Hàm thêm dữ liệu test vào form

ds.testdulieu(ten, sl, tl, tg);

ButtonTester btnthem = new ButtonTester("btnThemSach");

btnthem[0].Click();//kích để thêm dữ liệu

if (mess.Equals(actmsg))

{

//Ghi dữ iệu vào file excel

DocGhiFile.setExcelFile(row++, 7, "Pass", @"C:\Users\phamv\Documents\HomeWork\Kiem thu phan mem\Kiem-Thu-Phan-Mem\Kiem-Thu-Phan-Mem\dulieutest.xlsx");

//Kiểm tra tescase đúng sai

Assert.AreEqual( mess,actmsg);

}

else

{

//Ghi dữ iệu vào file excel

DocGhiFile.setExcelFile(row++, 7, "False", @"C:\Users\phamv\Documents\HomeWork\Kiem thu phan mem\Kiem-Thu-Phan-Mem\Kiem-Thu-Phan-Mem\dulieutest.xlsx");

//Kiểm tra tescase đúng sai

Assert.AreEqual( mess,actmsg);

}

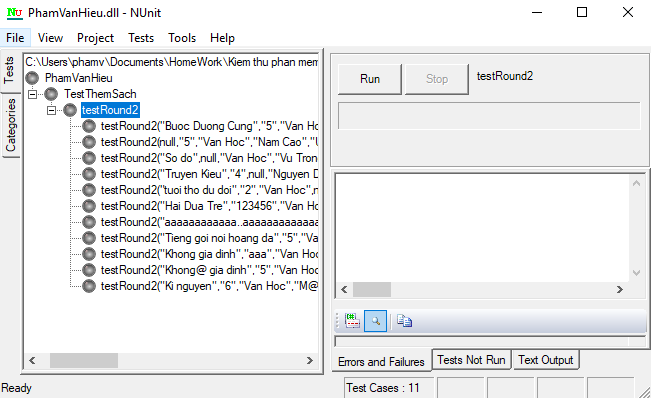
ds.Close();

}

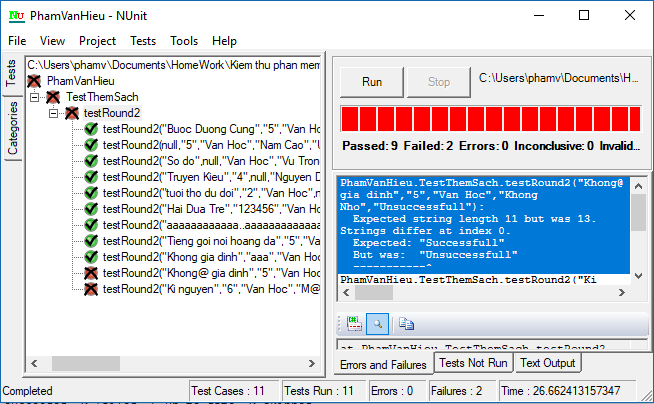
}

}

* Bước 4: Run test
  + Trên thanh công cụ chọn BUILD 🡪 Build Solution hoặc Rebuild Solution
  + Mở Nunit.exe
  + Chọn File 🡪 Open Project 🡪File Quản Lý Thư Viện 🡪 PhamVanHieu🡪 bin 🡪 Debug 🡪 PhamVanHieu.dll
  + Chọn thành công thi giao diện Nunit sẽ như sau:



* Click Run và chờ đợi kết quả.
* Kết quả Test đợt 2:



PASS: 9/11 (81.8%)

PENDING: 2/11 (18.2%)

FAIL: 0/11 (0%)

* Kết quả lưu trong file Excel:



#### Báo cáo kết quả kiểm thử

Đã test : 100% (11 Testcase)

Thành công: 81.8% (9 Testcase)

Pending 18.2% (2 Testcase)

### Phạm Quốc Cường – Kiểm thử chức năng Sửa độc giả (Version 0.1)

#### Phân tích thiết kế kiểm thử

* Giao diện chức năng



*Hinh 3.9 Chức năng Quản lý độc giả*

* Mã nguồn của chức năng

private void btnSuaDG\_Click(object sender, EventArgs e)

{

if (txtTenDG.Text == "" && rdoNam.Checked == false &&

**1**

rdoNu.Checked == false && txtDiaChi.Text == "" && txtSDT.Text == "")

MessageBox.Show("Edit Unsuccessfully"); **2**

else

{

String gioitinh;

if (rdoNam.Checked)

**3**

gioitinh = "Nam";

else gioitinh = "Nữ";

Double value;  **4**

if (Double.TryParse(txtSDT.Text, out value) == false)  **5**

MessageBox.Show("Edit Unsuccessfully "); **6**

else

if (txtSDT.TextLength < 10 || txtSDT.TextLength > 11) **7**

MessageBox.Show("Edit Unsuccessfully "); **8**

else

{

DocGia docgia = new DocGia(txtMaDG.Text, txtTenDG.Text, gioitinh, txtDiaChi.Text, txtSDT.Text);

**9**

DocGiaBUS.Instance.SuaDG(docgia);

QuanLySach\_Load(sender, e);

MessageBox.Show("Edit Successfully ");

}

}

* Lưu đồ giải thuật



*Hình 3.10 Lưu đồ giải thuật chức năng Sửa độc giả*

* Độ phức tạp chu trình :

C = ( 11 – 9 ) + 2 = (3 + 1) = 4

1. 1-2
2. 1-3-4-5-6
3. 1-3-4-5-7-8
4. 1-3-4-5-7-9

* Thiết kế test case :
  + Mô tả điều kiện
    - Bắt buộc phải nhập tất cả các trường trừ trường Tìm kiếm.
    - Mã độc giả được chọn từ danh sách độc giả.
    - Tên độc giả tối đa 30 ký tự.
    - Địa chỉ tối đa 100 ký tự.
    - Số điện thoại gồm 10 hoặc 11 số.
    - Giới tính được chọn từ RadioBox .
  + Xây dựng test case :
    - Áp dụng phương pháp phân vùng tương đương

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Đầu vào** | **Vùng hợp lệ** | **Ký hiệu đánh dấu** | **Vùng không hợp lệ** | **Dánh dấu** |
| Mã độc giả | Được chọn từ Datagridview | H1 | Không nằm trong Datagridview | K1 |
| Tên độc giả | [1 - 30] ký tự | H2 | >30 | K2 |
| Có ký tự đặc biệt | K3 |
| Bỏ trống | K4 |
| Địa chỉ | [1 - 100] ký tự | H3 | >100 | K5 |
| Có ký tự đặc biệt | K6 |
| Bỏ trống | K7 |
| Số điện thoại | [10 – 11] số | H4 | <10 | K8 |
| >11 | K9 |
| Không phải là số | K10 |
| Bỏ trống | K11 |
| Giới tính | Chọn RadioBox | H5 | Chưa chọn RadioBox | K12 |

*Bảng 3.3 Bảng phân vùng tương đương*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **TCs** | **Input** | | | | | **Output** | **Cover** |
| **Mã độc giả** | **Tên độc giả** | **Địa chỉ** | **Số điệnthoại** | **Giới tính** |
| TC1 | DG01 | James Bond | New York | 1235671520 | TRUE | Edit Successfully | H1,H2,H3,H4,H5 |
| TC2 | DG13 | James Bond | New York | 1235671520 | TRUE | Edit Unsuccessfully | K1 |
| TC3 | DG01 | AAAAABBBBBAAAAABBBBBAAAAABBBBBA | New York | 1235671520 | TRUE | Edit Unsuccessfully | K2 |
| TC4 | DG01 | James## Bond$$ | New York | 1235671520 | TRUE | Edit Unsuccessfully | K3 |
| TC5 | DG03 | Ellie Bamber | London | 1235671529 | FALSE | Edit Successfully | H1,H2,H3,H4,H5 |
| TC6 | DG01 |  | New York | 1235671520 | TRUE | Edit Unsuccessfully | K4 |
| TC7 | DG01 | James Bond | New..YorkNe..YorkNew..YorkNew..YorkNew..YorkNew..York | 1235671520 | TRUE | Edit Unsuccessfully | K5 |
| TC8 | DG02 | Emma Watson | Manchester City | 1235671520 | FALSE | Edit Successfully | H1,H2,H3,H4,H5 |
| TC9 | DG01 | James Bond | New£$York% | 1235671520 | TRUE | Edit Unsuccessfully | K6 |
| TC10 | DG01 | James Bond |  | 1235671520 | TRUE | Edit Unsuccessfully | K7 |
| TC11 | DG07 | Chinsu | Earth | 1235671521 | TRUE | Edit Successfully | H1,H2,H3,H4,H5 |
| TC12 | DG04 | Emilia Clark | England | 1235671520 | FALSE | Edit Successfully | H1,H2,H3,H4,H5 |
| TC13 | DG01 | James Bond | New York | 1235671 | TRUE | Edit Unsuccessfully | K8 |
| TC14 | DG01 | James Bond | New York | 123567123447 | TRUE | Edit Unsuccessfully | K9 |
| TC15 | DG06 | Anna Hathaway | Manhattan | 76543621987 | FALSE | Edit Successfully | H1,H2,H3,H4,H5 |
| TC16 | DG01 | James Bond | New York | abcdefgh | TRUE | Edit Unsuccessfully | K10 |
| TC17 | DG12 | Jason Bourne | Seattle |  | TRUE | Edit Unsuccessfully | K11 |
| TC18 | DG10 | Someone Else | Nowhere | 7896541235 | TRUE | Edit Successfully | H1,H2,H3,H4,H5 |
| TC19 | DG01 | James Bond | New York |  | TRUE | Edit Unsuccessfully | K11 |
| TC20 | DG01 | James Bond | New York | 123567124977 | TRUE | Edit Unsuccessfully | K9 |
| TC21 | DG08 | Jack Bauer | Utah | 1235671247 | TRUE | Edit Successfully | H1,H2,H3,H4,H5 |
| TC22 | DG08 | Jack Bauer | Utah | 12356712848 | TRUE | Edit Successfully | H1,H2,H3,H4,H6 |

*Hình 3.11 Ca kiểm thử*

* Testcase Sửa độc giả





*Hình 3.12 Datatest sau khi phân tích test case.*

#### Thực hiện kiểm thử

* Bước 1: Tạo một ClassLibrary với tên “PhamQuocCuong\_TestSuaDocGia” trong Solution “QuanLyThuVien”.
* Bước 2: Thêm các framework: Chuột phải “PhamQuocCuong\_TestSuaDocGia” 🡪 Add Reference 🡪 Chọn các framework sau:
  + NUnit.Extensions.Forms
  + NUnit.Framework
  + System.Windows.Forms
  + Microsoft.Office.Interop.Excel
  + System.Data.OleDb
  + System.Runtime.InteropServices
* Bước 3:
  + Tạo hàm setValue() trong file QuanLySach.cs để đưa dữ liệu các Testcases vào ô TextBox

public void setValue(string madg, string tendg, bool gt, string diachi, string sdt) //Function for Testing

{

txtMaDG.Text = madg;

txtTenDG.Text = tendg;

if (gt == true)

{

rdoNam.Checked = true;

}

else { rdoNu.Checked = true; }

txtDiaChi.Text = diachi;

txtSDT.Text = sdt.Trim();

}

* + Tạo class DocGhiFileExcel.cs để đọc dữ liệu từ file Excel

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.OleDb; //Thư viện hỗ trợ đọc ghi file excel

using System.Runtime.InteropServices; //Thư viện hỗ trợ đọc ghi file Excel, cái này cần add reference vào vì nó không có sẵn trong C#

using Excel = Microsoft.Office.Interop.Excel;

using System.Collections.ObjectModel;

using DTO;

using System.IO;

namespace PhamQuocCuong\_TestSuaDocGia

{

class DocGhiFileExcel

{

public static Excel.\_Application App;

public static Excel.\_Workbook Workbook; //Sử dụng Workbook cho file đuôi .xlsx và HWorkbook đối với file đuôi .slx

public static Excel.\_Worksheet Worksheet; //Quản lý các sheet trong excel

public static Excel.Range Range; //Quản lý các thành phần trong 1 sheet

public static void open(String fileName) {

//Tạo đối tượng COM và mở kết nối tới các đối tượng cần dùng để đọc dữ Liệu

App = new Excel.Application();

Workbook = App.Workbooks.Open(fileName);

Worksheet = Workbook.Sheets[1];

//Range = Worksheet.UsedRange;

Range = Worksheet.Range["D6:I26"]; //Vùng dữ liệu cần lấy

}

public static void close() {

//Đóng toàn bộ

GC.Collect();

GC.WaitForPendingFinalizers();

Marshal.ReleaseComObject(Range);

Marshal.ReleaseComObject(Worksheet);

//Đóng và thu hồi kết nối

Workbook.Save();

Workbook.Close();

Marshal.ReleaseComObject(Workbook);

//Thoát và thu hồi kết nối

App.Quit();

Marshal.ReleaseComObject(App);

}

//Phương thức này được dùng để tạo ra một danh sách các mảng đối tượng một Chiều, mỗi một mảng là một dòng trong bảng tính Excel.

public static List<Object[]> getExcelFile(String fileName) {

open(fileName);

int rowCount = Range.Rows.Count;

int colCount = 6;

int j = 1; //Đọc dữ liệu từ cột 1 đến cột 6

List<Object[]> list = new List<Object[]>();

string madg = "";

string tendg = "";

string diachi = "";

string sdt = "";

string gt = "";

string msg = "";

for (int i = 1; i <= rowCount; i++) {

//Đọc từng thành phần của dòng trong Excel

try {

madg = Range.Cells[i, j++].Value2.ToString();

} catch (Exception) {

madg = "null";

}

try {

tendg = Range.Cells[i, j++].Value2.ToString();

} catch (Exception) {

tendg = "null";

}

try {

diachi = Range.Cells[i, j++].Value2.ToString();

} catch (Exception) {

diachi = "null";

}

try {

sdt = Range.Cells[i, j++].Value2.ToString();

} catch (Exception) {

sdt = "null";

}

try {

gt = Range.Cells[i, j++].Value2.ToString();

} catch (Exception) {

gt = "null";

}

try {

msg = Range.Cells[i, j++].Value2.ToString();

} catch (Exception) {

msg = "null";

}

if (j > colCount) {

j = 1;

}

//Tạo mảng đối tượng lưu các giá trị vùa đọc được

Object[] obj = new Object[] { madg, tendg, diachi, sdt, gt, msg };

list.Add(obj); //Thêm đối tượng vào danh sách

}

close();//Đóng kết nối file Excel

return list;

}

//Phương thức này sử dụng để ghi dữ liệu vào file Excel theo từng thành phần

public static void setExcelFile(int i, int j, String str, String fileName) {

open(fileName);

Range.Cells[i, j].Value2 = str; //Set giá trị cho các Cells trong Excel

close();

}

}

}

* + Tạo file TestSuaDocGia.cs để thực hiện việc test tự động

using System;

using System.Windows.Forms;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using NUnit.Extensions.Forms; //Cần để viết test

using NUnit.Framework; //Cần để viết test

using System.Runtime.InteropServices;

using Microsoft.Office.Interop.Excel;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using QuanLyThuVien;

using DTO;

namespace PhamQuocCuong\_TestSuaDocGia

{

[TestFixture] //Sử dụng để thoogn báo chứa các Testcases ở đây

public class Test\_SuaDocGia

{

public int row = 1;

public static List<Object[]> data() { //Trả về các đối tượng đọc từ file Excel

return DocGhiFileExcel.getExcelFile(@"E:\Git\Kiem-Thu-Phan-Mem\Testcase\_SuaDocGia.xlsx");

}

[TestCaseSource("data"), Test]

public void Test(string madg, string tendg, string diachi, string sdt, string gt, string msg) {

//Tạo đối tượng của Form

QuanLySach qls = new QuanLySach();

qls.Show();

ButtonTester btnDocGia = new ButtonTester("btnDocGia");

btnDocGia[0].Click();

//Lấy thông tin từ MessageBox

string actmsg = "";

ModalFormTester msgBox = new ModalFormTester();

msgBox.ExpectModal("Info", delegate {

MessageBoxTester mess = new MessageBoxTester("Info");

actmsg = mess.Text;

mess.ClickOk(); //Click MessageBox

});

//Truyền giá trị vào hàm setValue() tạo bên QuanLySach.cs

qls.setValue(madg, tendg, Convert.ToBoolean(gt.ToLower()), diachi, sdt.Trim());

//Tạo Button Sửa độc giả rồi click

ButtonTester btnSua = new ButtonTester("btnSuaDG");

btnSua[0].Click();

//So sánh kết quả và ghi vào file Excel

try {

if (msg.Equals(actmsg)) {

DocGhiFileExcel.setExcelFile(row++, 7, "PASS", @"E:\Git\Kiem-Thu-Phan-Mem\Testcase\_SuaDocGia.xlsx");

Assert.AreEqual(actmsg, msg);

}

else {

Assert.Fail(); //Phát hiện các trường hợp Fail

}

} catch (Exception) {

DocGhiFileExcel.setExcelFile(row++, 7, "FAIL", @"E:\Git\Kiem-Thu-Phan-Mem\Testcase\_SuaDocGia.xlsx");

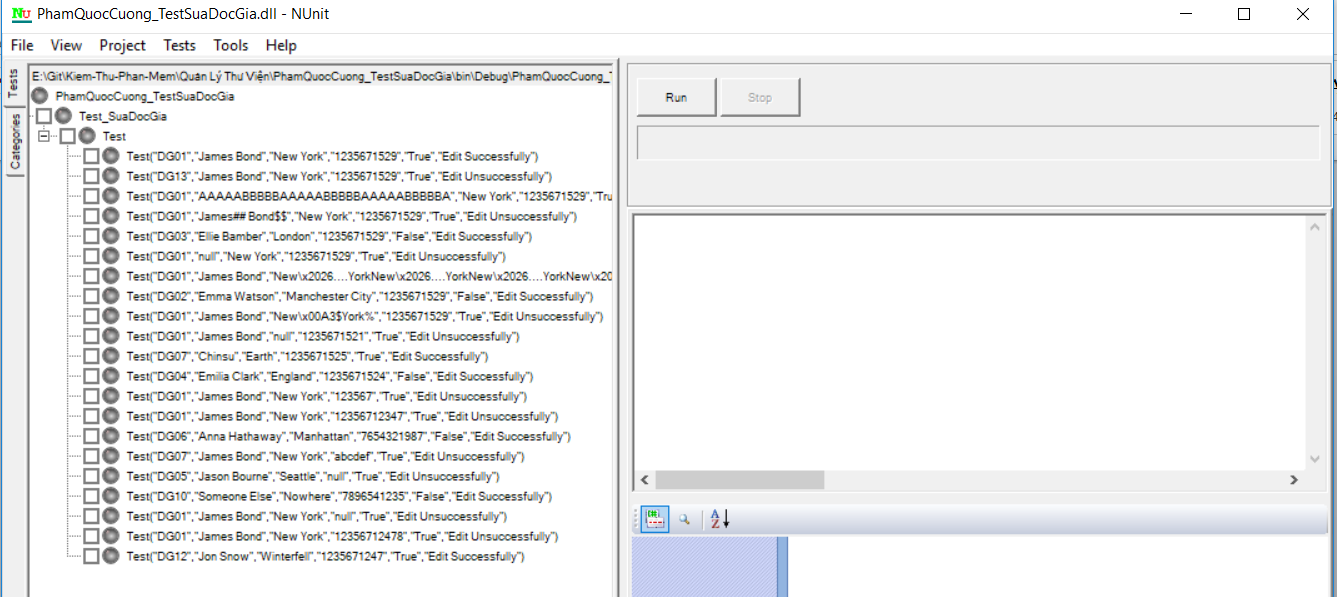
Assert.AreEqual(actmsg, msg); }

}

}

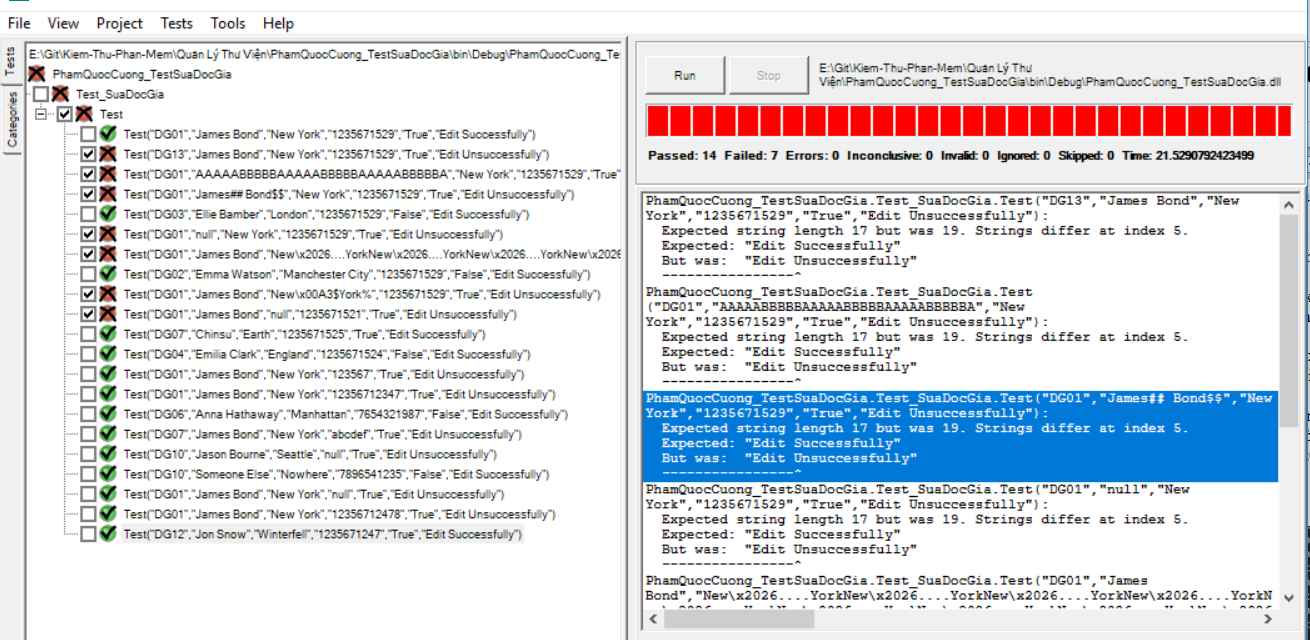
}

* Bước 4: Run test
  + Trên thanh công cụ chọn BUILD 🡪 Build Solution hoặc Rebuild Solution
  + Mở Nunit.exe
  + Chọn File 🡪 Open Project 🡪File Quản Lý Thư Viện 🡪 PhamQuocCuong\_TestSuaDocGia 🡪 bin 🡪 Debug 🡪 PhamQuocCuong\_TestSuaDocGia.dll
  + Chọn thành công thi giao diện Nunit sẽ như sau:



*Hình 3.13 Trước khi thực hiện test tự động*

* Click Run và chờ đợi kết quả.
* Kết quả Test đợt 1:



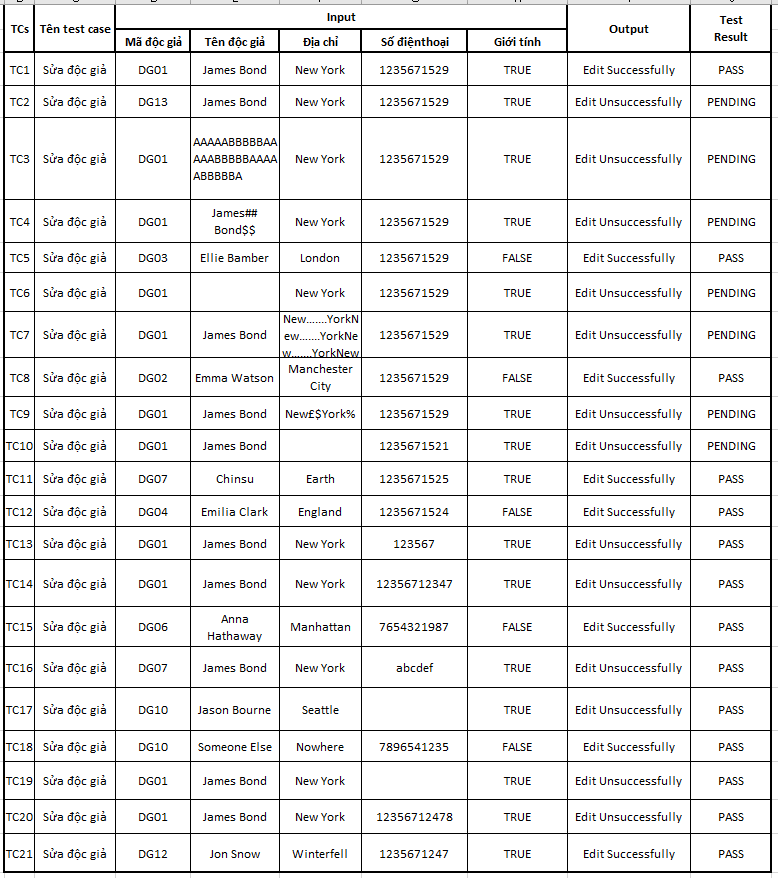
*Hình 3.14 Kết quả sau khi test tự động*

PASS: 14/21 (66.67%)

PENDING: 7/21 (33.33%)

FAIL: 0/21 (0%)

* Kết quả lưu trong file Excel:



*Hình 3.15 Kết quả test được lưu trong file Excel*

### Phạm Quốc Cường – Kiểm thử chức năng Sửa độc giả (Version 0.2)

#### Các lỗi đang chờ được sửa

1. Do chưa kiểm tra mã độc giả đã có trong cơ sở dữ liệu hay chưa

Giải pháp : tạo hàm KiemTraMaDG() để kiểm tra

Mã nguồn :

public bool KiemTraMaDG(DataGridView dtgDocGia) {

for (int i = 0; i < dtgDocGia.RowCount; i++) {

if (txtMaDG.Text.Equals(dtgDocGia.Rows[i].Cells[0].Value.ToString())) {

return true;

}

}

return false;

}

1. Chưa kiểm tra số ký tự tối đa của tên độc giả
2. Chưa bắt lỗi tên độc giả chứa ký tự đặc biệt
3. Chưa bắt lỗi tên độc giả bỏ trống
4. Chưa kiểm tra số ký tự tối đa của địa chỉ
5. Chưa bắt lỗi địa chỉ chứa ký tự đặc biệt
6. Chưa bắt lỗi địa chỉ bỏ trống

#### Phân tích thiết kế kiểm thử

* Mã nguồn chức năng sau khi chỉnh sửa:

private void btnSuaDG\_Click(object sender, EventArgs e)

{

if (txtTenDG.Text == "" || (rdoNam.Checked == false && rdoNu.Checked == false) || txtDiaChi.Text == "" || txtSDT.Text == "" || !KiemTraMaDG(dtgDocGia) || txtTenDG.Text.Length > 30 || txtDiaChi.Text.Length > 100) {

**1**

MessageBox.Show("Edit Unsuccessfully", "Info"); **2**

}

else {

String gioitinh;

if (rdoNam.Checked)

**3**

gioitinh = "Nam";

else gioitinh = "Nữ";

Double value; **4**

if (!Double.TryParse(txtSDT.Text, out value)) **5**

MessageBox.Show("Edit Unsuccessfully", "Info"); **6**

else if (txtSDT.Text.Length < 10 || txtSDT.Text.Length > 11) **7**

MessageBox.Show("Edit Unsuccessfully", "Info"); **8**

else

{

DocGia docgia = new DocGia(txtMaDG.Text, txtTenDG.Text, gioitinh, txtDiaChi.Text, txtSDT.Text);

**9**

DocGiaBUS.Instance.SuaDG(docgia);

QuanLySach\_Load(sender, e);

MessageBox.Show("Edit Successfully", "Info");

}

}

}

* Lưu đồ giải thuật



*Hình 3.16 Lưu đồ giải thuật chức năng Sửa độc giả*

* Độ phức tạp chu trình:

C = ( 11-9 ) +2 = (3 + 1) = 4

1. 1-2
2. 1-3-4-5-6
3. 1-3-4-5-7-8
4. 1-3-4-5-7-9

* Thiết kế test case :
  + Mô tả điều kiện
    - Bắt buộc phải nhập tất cả các trường trừ trường Tìm kiếm.
    - Mã độc giả được chọn từ danh sách độc giả.
    - Tên độc giả tối đa 30 ký tự.
    - Địa chỉ tối đa 100 ký tự.
    - Số điện thoại gồm 10 hoặc 11 số.
    - Giới tính được chọn từ RadioBox .
  + Xây dựng test case :
    - Áp dụng phương pháp phân vùng tương đương

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Đầu vào** | **Vùng hợp lệ** | **Ký hiệu đánh dấu** | **Vùng không hợp lệ** | **Dánh dấu** |
| Mã độc giả | Được chọn từ Datagridview | H1 | Không nằm trong Datagridview | K1 |
| Tên độc giả | [1 - 30] ký tự | H2 | >30 | K2 |
| Có ký tự đặc biệt | K3 |
| Bỏ trống | K4 |
| Địa chỉ | [1 - 100] ký tự | H3 | >100 | K5 |
| Có ký tự đặc biệt | K6 |
| Bỏ trống | K7 |
| Số điện thoại | [10 – 11] số | H4 | <10 | K8 |
| >11 | K9 |
| Không phải là số | K10 |
| Bỏ trống | K11 |
| Giới tính | Chọn RadioBox | H5 | Chưa chọn RadioBox | K12 |

*Bảng 3.4 Bảng phân vùng tương đương*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **TCs** | **Input** | | | | | **Output** | **Cover** |
| **Mã độc giả** | **Tên độc giả** | **Địa chỉ** | **Số điệnthoại** | **Giới tính** |
| TC1 | DG01 | James Bond | New York | 1235671520 | TRUE | Edit Successfully | H1,H2,H3,H4,H5 |
| TC2 | DG13 | James Bond | New York | 1235671520 | TRUE | Edit Unsuccessfully | K1 |
| TC3 | DG01 | AAAAABBBBBAAAAABBBBBAAAAABBBBBA | New York | 1235671520 | TRUE | Edit Unsuccessfully | K2 |
| TC4 | DG01 | James## Bond$$ | New York | 1235671520 | TRUE | Edit Unsuccessfully | K3 |
| TC5 | DG03 | Ellie Bamber | London | 1235671529 | FALSE | Edit Successfully | H1,H2,H3,H4,H5 |
| TC6 | DG01 |  | New York | 1235671520 | TRUE | Edit Unsuccessfully | K4 |
| TC7 | DG01 | James Bond | New..YorkNe..YorkNew..YorkNew..YorkNew..YorkNew..York | 1235671520 | TRUE | Edit Unsuccessfully | K5 |
| TC8 | DG02 | Emma Watson | Manchester City | 1235671520 | FALSE | Edit Successfully | H1,H2,H3,H4,H5 |
| TC9 | DG01 | James Bond | New£$York% | 1235671520 | TRUE | Edit Unsuccessfully | K6 |
| TC10 | DG01 | James Bond |  | 1235671520 | TRUE | Edit Unsuccessfully | K7 |
| TC11 | DG07 | Chinsu | Earth | 1235671521 | TRUE | Edit Successfully | H1,H2,H3,H4,H5 |
| TC12 | DG04 | Emilia Clark | England | 1235671520 | FALSE | Edit Successfully | H1,H2,H3,H4,H5 |
| TC13 | DG01 | James Bond | New York | 1235671 | TRUE | Edit Unsuccessfully | K8 |
| TC14 | DG01 | James Bond | New York | 123567123447 | TRUE | Edit Unsuccessfully | K9 |
| TC15 | DG06 | Anna Hathaway | Manhattan | 76543621987 | FALSE | Edit Successfully | H1,H2,H3,H4,H5 |
| TC16 | DG01 | James Bond | New York | abcdefgh | TRUE | Edit Unsuccessfully | K10 |
| TC17 | DG12 | Jason Bourne | Seattle |  | TRUE | Edit Unsuccessfully | K11 |
| TC18 | DG10 | Someone Else | Nowhere | 7896541235 | TRUE | Edit Successfully | H1,H2,H3,H4,H5 |
| TC19 | DG01 | James Bond | New York |  | TRUE | Edit Unsuccessfully | K11 |
| TC20 | DG01 | James Bond | New York | 123567124977 | TRUE | Edit Unsuccessfully | K9 |
| TC21 | DG08 | Jack Bauer | Utah | 1235671247 | TRUE | Edit Successfully | H1,H2,H3,H4,H5 |
| TC22 | DG08 | Jack Bauer | Utah | 12356712848 | TRUE | Edit Successfully | H1,H2,H3,H4,H6 |

*Hình 3.17 Ca kiểm thử*

* Testcase Sửa độc giả





*Hình 3.18 Datatest sau khi phân tích test case.*

#### Thực hiện kiểm thử

* Bước 1: Tạo một ClassLibrary với tên “PhamQuocCuong\_TestSuaDocGia” trong Solution “QuanLyThuVien”.
* Bước 2: Thêm các framework: Chuột phải “PhamQuocCuong\_TestSuaDocGia” 🡪 Add Reference 🡪 Chọn các framework sau:
  + NUnit.Extensions.Forms
  + NUnit.Framework
  + System.Windows.Forms
  + Microsoft.Office.Interop.Excel
  + System.Data.OleDb
  + System.Runtime.InteropServices
* Bước 3:
  + Tạo hàm setValue() trong file QuanLySach.cs để đưa dữ liệu các Testcases vào ô TextBox

public void setValue(string madg, string tendg, bool gt, string diachi, string sdt) //Function for Testing

{

txtMaDG.Text = madg;

txtTenDG.Text = tendg;

if (gt == true)

{

rdoNam.Checked = true;

}

else { rdoNu.Checked = true; }

txtDiaChi.Text = diachi;

txtSDT.Text = sdt.Trim();

}

* + Tạo class DocGhiFileExcel.cs để đọc dữ liệu từ file Excel

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.OleDb;

using System.Runtime.InteropServices;

using Excel = Microsoft.Office.Interop.Excel;

using System.Collections.ObjectModel;

using DTO;

using System.IO;

namespace PhamQuocCuong\_TestSuaDocGia

{

class DocGhiFileExcel

{

public static Excel.\_Application App;

public static Excel.\_Workbook Workbook; //Sử dụng Workbook cho file đuôi .xlsx và HWorkbook đối với file đuôi .slx

public static Excel.\_Worksheet Worksheet; //Quản lý các sheet trong excel

public static Excel.Range Range; //Quản lý các thành phần trong 1 sheet

public static void open(String fileName) {

//Tạo đối tượng COM và mở kết nối tới các đối tượng cần dùng để đọc dữ Liệu

App = new Excel.Application();

Workbook = App.Workbooks.Open(fileName);

Worksheet = Workbook.Sheets[1];

//Range = Worksheet.UsedRange;

Range = Worksheet.Range["D6:I27"]; //Vùng dữ liệu cần lấy

}

public static void close() {

//Đóng toàn bộ

GC.Collect();

GC.WaitForPendingFinalizers();

Marshal.ReleaseComObject(Range);

Marshal.ReleaseComObject(Worksheet);

//Đóng và thu hồi kết nối

Workbook.Save();

Workbook.Close();

Marshal.ReleaseComObject(Workbook);

//Thoát và thu hồi kết nối

App.Quit();

Marshal.ReleaseComObject(App);

}

//Phương thức này được dùng để tạo ra một danh sách các mảng đối tượng một Chiều, mỗi một mảng là một dòng trong bảng tính Excel.

public static List<Object[]> getExcelFile(String fileName) {

open(fileName);

int rowCount = Range.Rows.Count;

int colCount = 6;

int j = 1; //Đọc dữ liệu từ cột 1 đến cột 6

List<Object[]> list = new List<Object[]>();

string madg = "";

string tendg = "";

string diachi = "";

string sdt = "";

string gt = "";

string msg = "";

for (int i = 1; i <= rowCount; i++) {

//Đọc từng thành phần của dòng trong Excel

try {

madg = Range.Cells[i, j++].Value2.ToString();

} catch (Exception) {

madg = "null";

}

try {

tendg = Range.Cells[i, j++].Value2.ToString();

} catch (Exception) {

tendg = null;

}

try {

diachi = Range.Cells[i, j++].Value2.ToString();

} catch (Exception) {

diachi = null;

}

try {

sdt = Range.Cells[i, j++].Value2.ToString();

} catch (Exception) {

sdt = "null";

}

try {

gt = Range.Cells[i, j++].Value2.ToString();

} catch (Exception) {

gt = "null";

}

try {

msg = Range.Cells[i, j++].Value2.ToString();

} catch (Exception) {

msg = "null";

}

if (j > colCount) {

j = 1;

}

//Tạo mảng đối tượng lưu các giá trị vùa đọc được

Object[] obj = new Object[] { madg, tendg, diachi, sdt, gt, msg };

list.Add(obj); //Thêm đối tượng vào danh sách

}

close();//Đóng kết nối file Excel

return list;

}

//Phương thức này sử dụng để ghi dữ liệu vào file Excel theo từng thành phần

public static void setExcelFile(int i, int j, String str, String fileName) {

open(fileName);

Range.Cells[i, j].Value2 = str; //Set giá trị cho các Cells trong Excel

close();

}

}

}

* + Tạo file TestSuaDocGia.cs để thực hiện việc test tự động

using System;

using System.Windows.Forms;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using NUnit.Extensions.Forms; //Cần để viết test

using NUnit.Framework; //Cần để viết test

using System.Runtime.InteropServices;

using Microsoft.Office.Interop.Excel;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using QuanLyThuVien;

using DTO;

namespace PhamQuocCuong\_TestSuaDocGia

{

[TestFixture] //Sử dụng để thoogn báo chứa các Testcases ở đây

public class Test\_SuaDocGia

{

public int row = 1;

public static List<Object[]> data() { //Trả về các đối tượng đọc từ file Excel

return DocGhiFileExcel.getExcelFile(@"E:\Git\Kiem-Thu-Phan-Mem\Testcase\_SuaDocGia.xlsx");

}

[TestCaseSource("data"), Test]

public void Test(string madg, string tendg, string diachi, string sdt, string gt, string msg) {

//Tạo đối tượng của Form

QuanLySach qls = new QuanLySach();

qls.Show();

ButtonTester btnDocGia = new ButtonTester("btnDocGia");

btnDocGia[0].Click();

//Lấy thông tin từ MessageBox

string actmsg = "";

ModalFormTester msgBox = new ModalFormTester();

msgBox.ExpectModal("Info", delegate {

MessageBoxTester mess = new MessageBoxTester("Info");

actmsg = mess.Text;

mess.ClickOk(); //Click MessageBox

});

//Truyền giá trị vào hàm setValue() tạo bên QuanLySach.cs

qls.setValue(madg, tendg, Convert.ToBoolean(gt.ToLower()), diachi, sdt.Trim());

//Tạo Button Sửa độc giả rồi click

ButtonTester btnSua = new ButtonTester("btnSuaDG");

btnSua[0].Click();

//So sánh kết quả và ghi vào file Excel

try {

if (msg.Equals(actmsg)) {

DocGhiFileExcel.setExcelFile(row++, 7, "PASS", @"E:\Git\Kiem-Thu-Phan-Mem\Testcase\_SuaDocGia.xlsx");

Assert.AreEqual(actmsg, msg);

}

else {

Assert.Fail(); //Phát hiện các trường hợp Fail

}

} catch (Exception) {

DocGhiFileExcel.setExcelFile(row++, 7, "FAIL", @"E:\Git\Kiem-Thu-Phan-Mem\Testcase\_SuaDocGia.xlsx");

Assert.AreEqual(actmsg, msg);

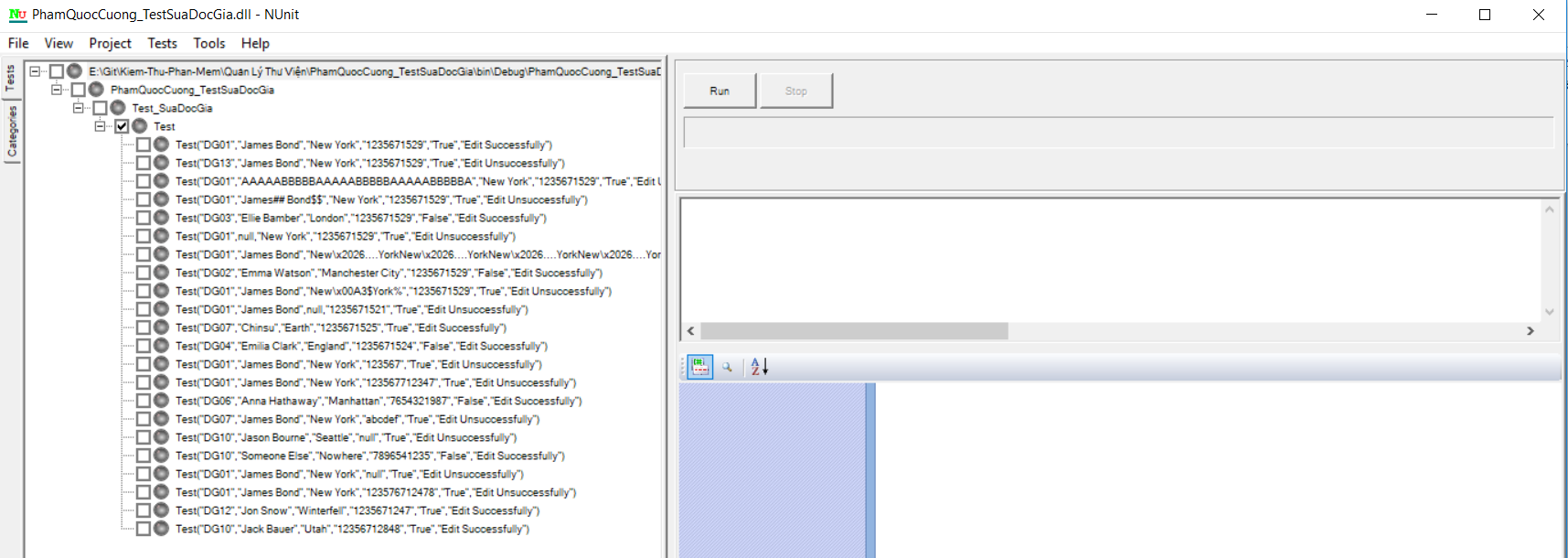
}

}

}

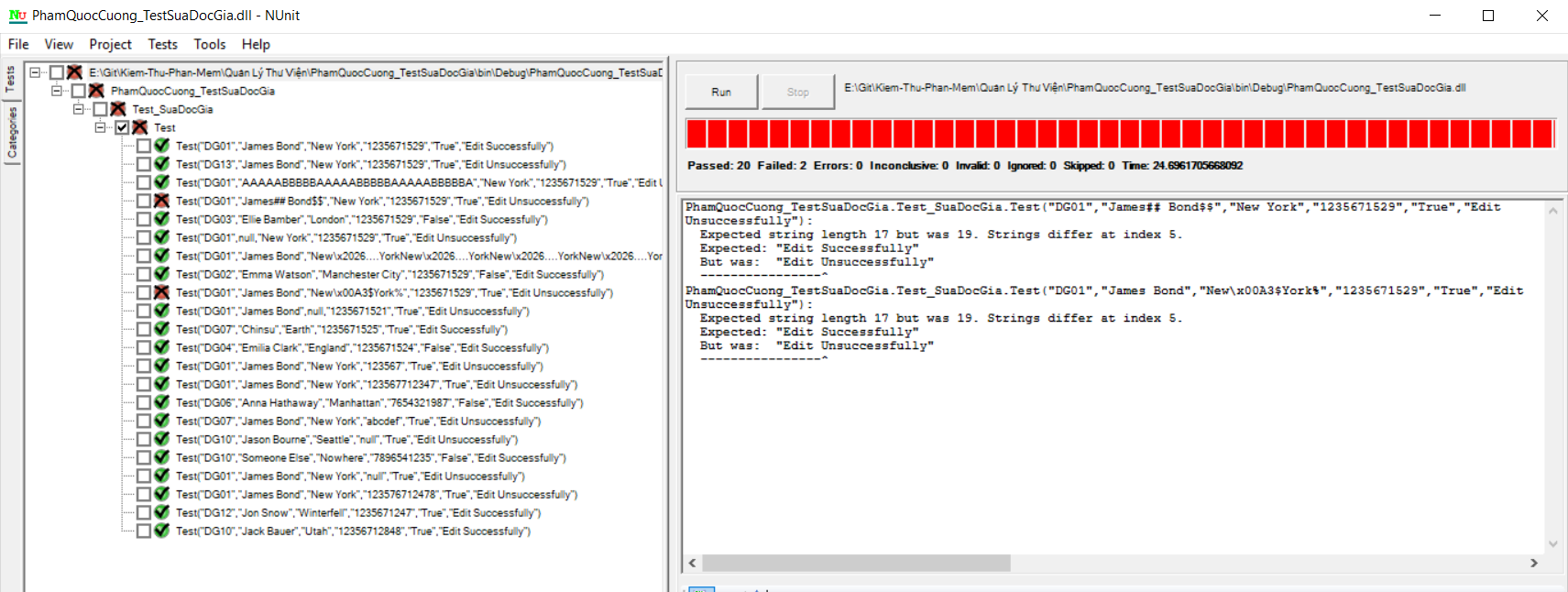
}

* Bước 4: Run test
  + Trên thanh công cụ chọn BUILD 🡪 Build Solution hoặc Rebuild Solution
  + Mở Nunit.exe
  + Chọn File 🡪 Open Project 🡪File Quản Lý Thư Viện 🡪 PhamQuocCuong\_TestSuaDocGia 🡪 bin 🡪 Debug 🡪 PhamQuocCuong\_TestSuaDocGia.dll
  + Chọn thành công thi giao diện Nunit sẽ như sau:



*Hình 3.19 Trước khi thực hiện test tự động*

* Click Run và chờ đợi kết quả.
* Kết quả Test đợt 2:



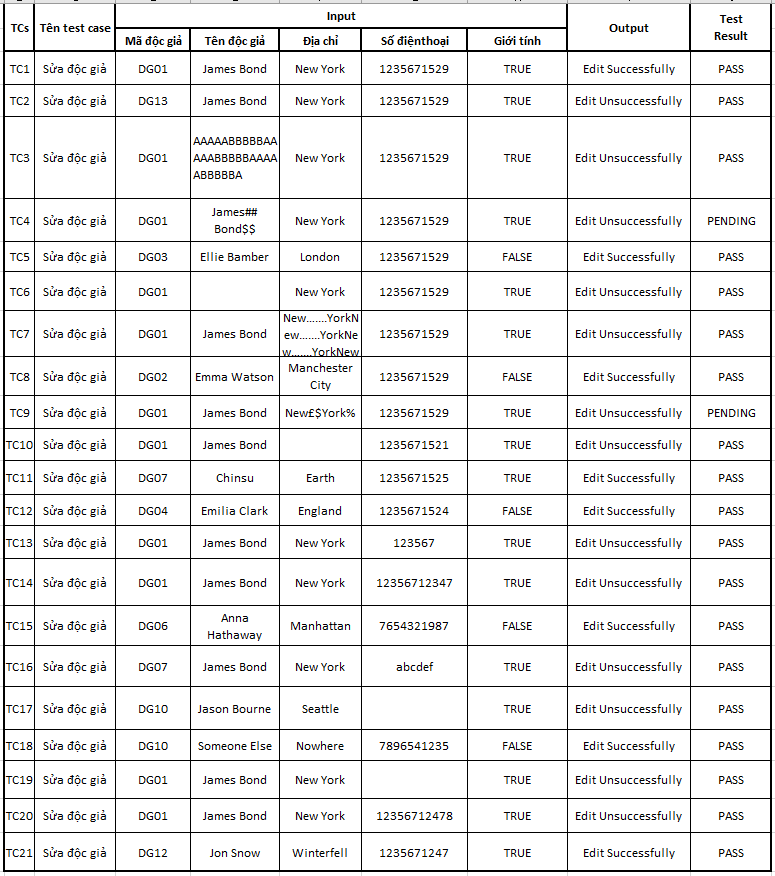
*Hình 3.20 Kết quả thực hiện test tự động*

PASS: 20/22 (90.91%)

PENDING: 2/22 (9.09%)

FAIL: 0/22 (0%)

* Lỗi đang trong hàng chờ:
  + Tên độc gỉả chứa ký tự đặc biệt.
  + Địa chỉ chứa ký tự đặc biệt.
* Kết quả lưu trong file Excel:



*Hình 3.21 Kết quả test được lưu trong file Excel*

#### Báo cáo kết quả kiểm thử Chức năng Sửa độc giả

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **ROUND1** | **Kết quả** | **ROUND2** | **Kết quả** |
| **Tổng số Testcase** | 21 | 100% | 22 | 100% |
| **PASS** | 14 | 66.67% | 20 | 90.91% |
| **PENDING** | 7 | 33.33% | 2 | 9.09% |
| **FAIL** | 0 | 0% | 0 | 0% |

Bảng 3.5 Kết quả kiểm thử chức năng Sủa độc giả