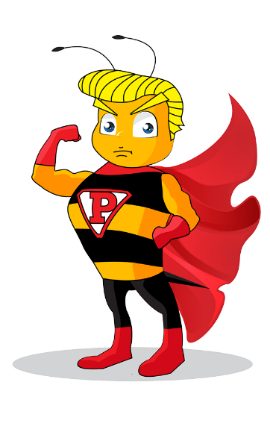
Graphical user interface, text

Description automatically generated

**Báo cáo SOF304**

MÔN HỌC : Kiểm thử nâng cao



Báo cáo kiểm thử

ASSIGNMENT

Họ và tên thành viên : Dương Tấn Lực PS17380

Nguyễn Phạm Hùng Phi PD04328

Lãnh Quốc Khánh PS13132

Nguyễn Công Tuấn PS18742

Lớp/Kỳ : IT16317/SPRING2022

***Giảng viên hướng dẫn : thầy Lại Minh Đăng***

Jan 2022

POLYTECHNIC

# 

Mục lục

[1. GIỚI THIỆU DỰ ÁN 5](#_Toc100957621)

[1.1 Giới Thiệu Công Ty Laptrinhcity 5](#_Toc100957622)

[1.2 Yêu Cầu Của Công Ty 5](#_Toc100957623)

[2 . PHÂN TÍCH YÊU CẦU KHÁCH HÀNG 6](#_Toc100957624)

[2.2 Đặc Tả Yêu Cầu Hệ Thống (SRS) 7](#_Toc100957625)

[2.2.1 Quản lý nhân viên 7](#_Toc100957626)

[2.2.2 Quản lý chuyên đề 7](#_Toc100957627)

[2.2.3 Quản lý khóa học 8](#_Toc100957628)

[2.2.4 Quản lý người học 9](#_Toc100957629)

[2.2.5 Quản lý học viên 10](#_Toc100957630)

[2.2.6 Tổng hợp và thống kê 11](#_Toc100957631)

[2.2.7 Đăng nhập 12](#_Toc100957632)

[2.2.8 Đổi mật khẩu 13](#_Toc100957633)

[2.3 Sơ Đồ Triển Khai Và Yêu Cầu Hệ Thống 14](#_Toc100957634)

[2.3.2 Yêu cầu hệ thống 14](#_Toc100957635)

[3.1 CHIẾN LƯỢC KIỂM THỬ 14](#_Toc100957636)

[3.2 MÔI TRƯỜNG VÀ YÊU CẦU KIỂM THỬ 18](#_Toc100957637)

[4.1 Thiết kế testcase 21](#_Toc100957638)

[4.2 Xây dựng testcase 24](#_Toc100957639)

[5.1 UI TEST SUITE 32](#_Toc100957640)

[33](#_Toc100957642)

[5.1.1 DangNhapJDialogTest 33](#_Toc100957643)

[5.1.2 DoiMatKhauJDialogTest 35](#_Toc100957644)

[5.1.3 HocVienJDialogTest 38](#_Toc100957645)

[5.2 DAO TEST SUITE 39](#_Toc100957646)

[5.2.1 NhanVienDAOTest 40](#_Toc100957647)

[5.2.2 HocVienDAOTest 43](#_Toc100957648)

[5.3 WEB TEST SUITE 47](#_Toc100957649)

[5.4 ALL-SUITE 50](#_Toc100957650)

[6. TEST REPORT VÀ DEFECT REPORT 52](#_Toc100957651)

[6.1 TEST REPORT 52](#_Toc100957652)

[6.1.1 Tổng quan 52](#_Toc100957653)

[6.1.2 Automation test result 54](#_Toc100957654)

[6.2 DEFECT REPORT 55](#_Toc100957655)

# 1. GIỚI THIỆU DỰ ÁN

# 1.1 Giới Thiệu Công Ty Laptrinhcity

Trung tâm đào tạo tin học ngắn hạn có tên là LapTrinhCity đang đào tạo các khóa tin học ngắn hạn theo chuyên đề như Photoshop, Java, Web…

Việc quản lý khóa học, học viên, bảng điểm và doanh thu hiện đang thực hiện thông qua excel.

Hiện tại Trung tâm đang gặp khó khăn khi số lượng người học ngày một nhiều nên việc quản lý excel gặp rất nhiều khó khăn.

## 1.2 Yêu Cầu Của Công Ty

LapTrinhCity mong muốn xây dựng một phần mềm để giải quyết khó khăn trên với yêu cầu như sau:

**✓ QUẢN LÝ**

• Các chuyên đề (Java, Photoshop…)

• Các khóa học (được tạo ra từ các chuyên đề)

• Học viên của các khóa học

• Người đăng ký học (người học)

**✓ THỐNG KÊ**

• Doanh thu, số lượng người học từng chuyên đề theo năm

• Số lượng người học đăng ký theo tháng

• Tổng hợp bảng điểm của các khóa học

**✓ BẢO MẬT**

• Phần mềm được viết cho 2 đối tượng sử dụng là trưởng phòng và nhân viên giáo vụ với yêu cầu bảo mật như sau:

• Tất cả mọi thành viên phải đăng nhập mới được sử dụng phần mềm

• Trưởng phòng đào tạo thực hiện được tất cả các chức năng

• Nhân viên đào tạo không được phép xóa và xem doanh thu

**✓CÔNG NGHỆ**

• Ứng dụng phải được thực hiện với công nghệ Swing và JDBC

• JDK: 1.8+

• Hệ quản trị CSDL: SQL Server 2008+

# 2 . PHÂN TÍCH YÊU CẦU KHÁCH HÀNG

Vẽ Sơ Đồ Use CaseDiagram

Description automatically generated

## 2.2 Đặc Tả Yêu Cầu Hệ Thống (SRS)

### 2.2.1 Quản lý nhân viên

**✓ Mô tả chức năng :**

Chức năng quản lý nhân viên được sủ dụng để quản lý thông tin nhân viên . Yêu cầu của chức năng này là:

• Liệt kê danh sách nhân viên

• Xem thông tin chi tiết của mỗi nhân viên

• Thêm nhân viên mới

• Cập nhật thông tin nhân viên

• Xóa nhân viên đã tồn tại

• Điều hướng: di chuyển đến dữ liệu của bản ghi chi tiết trước và sau

• Tìm kiếm: tìm kiếm theo điều kiện

• Kiểm lỗi hợp lý với dữ liệu đối với các thao tác thêm sửa xóa

**✓ Dữ liệu liên quan :** Thông tin mỗi nhân viên gồm

• Mã nhân viên

• Họ và tên nhân viên

• Mật khẩu đăng nhập

• Vai trò của nhân viên (Gồm trưởng phòng và nhân viên bình thường)

**✓** **Đối tượng sử dụng** : sau khi đăng nhập thì mọi nhân viên đều sử dụng được chức năng này , ngoại trừ xóa chỉ có trưởng phòng mới sử dụng được .

### 2.2.2 Quản lý chuyên đề

**✓ Mô tả chức năng :**

Chức năng quản lý chuyên đề được sủ dụng để quản lý các chuyên đề . Yêu cầu của chức năng này là:

• Liệt kê danh sách các chuyên đề

• Xem thông tin chi tiết của mỗi chuyên đề

• Tạo chuyên đề mới

• Cập nhật thông tin chuyên đề

• Xóa chuyên đề đã tồn tại

• Điều hướng: di chuyển đến dữ liệu của bản ghi chi tiết trước và sau

• Tìm kiếm: tìm kiếm theo điều kiện

• Kiểm lỗi hợp lý với dữ liệu đối với các thao tác thêm sửa xóa

**✓ Dữ liệu liên quan :** Thông tin mỗi chuyên đề gồm

• Mã chuyên đề

• Tên chuyên đề

• Học phí của chuyên đề

• Thời lượng chuyên đề

• Hình logo của chuyên đề

• Mô tả chuyên đề

✓ **Đối tượng sử dụng** : sau khi đăng nhập thì mọi nhân viên đều sử dụng được chức năng này , ngoại trừ xóa chỉ có trưởng phòng mới sử dụng được .

✓ **Lưu ý :** học phí và thời lượng của chuyên đề thay đổi không ảnh hưởng đến các khóa học đã được tạo .

### 2.2.3 Quản lý khóa học

**✓ Mô tả chức năng :**

Chức năng quản lý khóa học được sủ dụng để quản lý các khóa học của trung tâm . Yêu cầu của chức năng này là:

• Liệt kê danh sách các khóa học

• Xem thông tin chi tiết của mỗi khóa học

• Tạo khóa học mới

• Cập nhật thông tin khóa học

• Xóa khóa học đã tồn tại

• Điều hướng: di chuyển đến dữ liệu của bản ghi chi tiết trước và sau

• Tìm kiếm: tìm kiếm theo điều kiện

• Kiểm lỗi hợp lý với dữ liệu đối với các thao tác thêm sửa xóa

**✓ Dữ liệu liên quan :** Thông tin mỗi khóa học gồm

• Mã khóa học

• Mã chuyên đề

• Học phí của khóa học

• Thời lượng của khóa học

• Ngày khai giảng của khóa học

• Ghi chú của khóa học

✓ **Đối tượng sử dụng** : sau khi đăng nhập thì mọi nhân viên đều sử dụng được chức năng này , ngoại trừ xóa chỉ có trưởng phòng mới sử dụng được .

### 2.2.4 Quản lý người học

**✓ Mô tả chức năng :**

Chức năng quản lý người học được sủ dụng để quản lý các người học của trung tâm . Yêu cầu của chức năng này là:

• Liệt kê danh sách các người học

• Xem thông tin chi tiết của mỗi người học

• Tạo người học mới

• Cập nhật thông tin người học

• Xóa người học đã tồn tại

• Điều hướng: di chuyển đến dữ liệu của bản ghi chi tiết trước và sau

• Tìm kiếm: tìm kiếm theo điều kiện

• Kiểm lỗi hợp lý với dữ liệu đối với các thao tác thêm sửa xóa

**✓ Dữ liệu liên quan :** Thông tin mỗi người học gồm

• Mã người học

• Họ và tên người học

• Ngày sinh của người học

• Giới tính của người học

• Số điện thoại của người học

• Email của người học

• Ghi chú

✓ **Đối tượng sử dụng** : sau khi đăng nhập thì mọi nhân viên đều sử dụng được chức năng này , ngoại trừ xóa chỉ có trưởng phòng mới sử dụng được .

### 2.2.5 Quản lý học viên

**✓ Mô tả chức năng :**

Chức năng quản lý học viên được sủ dụng để quản lý các học viên các khóa học của trung tâm . Yêu cầu của chức năng này là:

• Liệt kê danh sách các học viên

• Xem thông tin chi tiết của mỗi học viên

• Thêm học viên mới của khóa học

• Cập nhật thông tin học viên

• Xóa học viên đã tồn tại

• Điều hướng: di chuyển đến dữ liệu của bản ghi chi tiết trước và sau

• Tìm kiếm: tìm kiếm theo điều kiện

• Kiểm lỗi hợp lý với dữ liệu đối với các thao tác thêm sửa xóa

**✓ Dữ liệu liên quan :** Thông tin mỗi người học gồm

• Mã học viên

• Mã khóa học

• Mã người học

• Điểm trung bình

✓ **Đối tượng sử dụng** : sau khi đăng nhập thì mọi nhân viên đều sử dụng được chức năng này , ngoại trừ xóa chỉ có trưởng phòng mới sử dụng được .

### 2.2.6 Tổng hợp và thống kê

**✓ Mô tả chức năng :**

Chức năng tổng hợp và thống kê được sủ dụng để thống kê và tổng hợp lại các số liệu cụ thể như doanh thu, số lượng học viên…của trung tâm . Yêu cầu của chức năng này là:

• Doanh thu theo năm .Thông tin tổng hợp cần thiết gồm

+ Chuyên đề

+ Số khóa

+ Số học viên

+ Doanh thu

+ Học phí thấp nhất

+ Học phí cao nhất

+ Học phí trung bình

• Xem thống kê số lượng người học từng chuyên đề theo năm . Thông tin tổng hợp cần thiết gồm

+ Năm

+ Tổng số học viên

+ Ngày đăng ký sớm nhất

+ Ngày đăng ký muộn nhất

• Xem tổng hợp điểm tất cả các chuyên đề . Thông tin tổng hợp gồm

+ Chuyên đề

+ Số học viên

+ Điểm cao nhất

+ Điểm thấp nhất

+ Điểm trung bình

• Xem tổng hợp bảng điểm của các khóa học . Thông tin tổng hợp gồm

+ Mã người học

+ Họ và tên

+ Điểm

+ Xếp loại

✓ **Đối tượng sử dụng** : sau khi đăng nhập thì chỉ có trưởng phòng mới sử dụng được sử dụng chức năng này .

### 2.2.7 Đăng nhập

**✓ Mô tả chức năng :**

Chức năng đăng nhập được chạy khi người sử dụng khởi động ứng dụng . Yêu cầu của chức năng này là:

• Cho phép người dùng đăng nhập để sử dụng các chức năng của phần mềm theo vai trò

• Lưu thông tin lần đăng nhập trước

• Kiểm tra thông tin đặng nhập của nhân viên (báo lỗi nếu sai thông tin)

**✓ Dữ liệu liên quan :** Thông tin đăng nhập gồm

• Mã nhân viên

• Mật khẩu đăng nhập

✓ **Đối tượng sử dụng** : Bất kỳ ai khi khởi động ứng dụng

✓ **Chú ý** : đây là yêu cầu bắt buộc để sử dụng các chức năng khác

### 2.2.8 Đổi mật khẩu

**✓ Mô tả chức năng :**

Chức năng đổi mật khẩu cho phép đổi mật khẩu của nhân viên hiện đang đăng nhập . Yêu cầu của chức năng này là:

• Cho phép nhân viên hiện đang đăng nhập đổi mật khẩu của họ

**✓ Dữ liệu liên quan :** Thông tin đổi mật khẩu gồm

• Mã nhân viên

• Mật khẩu đăng nhập

✓ **Đối tượng sử dụng** : sau khi đăng nhập thì mọi nhân viên đều có thể sử dụng được sử dụng chức năng này .

## 2.3 Sơ Đồ Triển Khai Và Yêu Cầu Hệ Thống

2.3.1 Sơ đồ triển khaiDiagram

Description automatically generated

### 2.3.2 Yêu cầu hệ thống

✓ Một máy cài SQL Server 2008+. Máy này cần hệ điều hành window xp trở lên

✓ Các máy nhân viên phòng đào tạo cài phần mềm edusys. Các máy nhân viên cần JDK 1.8+ với hệ điều hành bất ký

**3. KẾ HOẠCH KIỂM THỬ**

## 3.1 CHIẾN LƯỢC KIỂM THỬ

|  |  |  |
| --- | --- | --- |
| **Các chiến lược kiểm thử** | **Chi tiết chiến lược** | **Người thực hiện** |
| Kiểm thử đơn vị  Unit test | * Kiểm tra một đơn vị thiết kế nhỏ nhất – một mô đun của phần mềm. * Người tiến hành kiểm thử thông thường là người lập trình mô đun đó hoặc lập trình viên của nhóm. * Các mô đun thứ cấp của mô đun được kiểm thử nếu chưa được phát triển sẽ được thay bằng các chương trình tạm thời gọi là các stub. * Mô đun thượng cấp được thay bằng một trình điều khiển kiểm thử gọi là test driver. | Developer |
| Kiểm thử tích hợp  Intergrated Test | * Tích hợp các mô đun và kiểm thử chúng dưới một thể thống nhất. * Các đơn vị phần mềm (unit) được tích hợp dần thành các mô đun, hệ con, và cuối cùng là thành hệ thống hoàn chỉnh. * Một số lỗi giao diện (mô đun) điển hình:   + - Sử dụng sai giao diện     - Hiểu nhầm về giao diện     - Xung đột * **Các chiến lược kiểm thử tích hợp:** * **Kiểm thử dưới lên (bottom-up testing):**   + Là quá trình tích hợp và kiểm thử với các mô đun ở mức độ thấp trước.   + Tiến hành tích hợp và kiểm thử một số mô đun cấp trên trước * **Kiểm thử trên xuống (top-down testing):**   + Kiểm thử trên xuống tiến hành kiểm thử với các mô đun ở mức cao trước, các mô đun mức thấp được tạm thời phát triển với các chức năng hạn chế. * **Kiểm thử hồi qui (regression testing):**   + Là tiến hành lại các phép thử đã thành công mỗi khi tích hợp thêm mô đun hoặc khi cập nhật mã nguồn chương trình   + Khi chúng ta tích hợp thêm mô đun vào hệ thống hoặc khi tiến hành nâng cấp chương trình thì sẽ tạo ra một số tổ hợp trạng thái mới dẫn đến:   + Xuất hiện lỗi ở mô đun trước đây chưa gây lỗi   + Khắc phục một lỗi mới có thể sẽ làm ảnh hưởng tới một lỗi chúng ta đã sửa   + Sinh ra lỗi mới mà trước đây chưa có | Developer và Tester |
| Kiểm thử hệ thống  System test | * Kiểm thử khả năng hoạt động của hệ thống * Kiểm tra các vấn đề về hiệu năng của hệ thống, khả năng phục hồi khi gặp sự cố,… * **Một số các dạng kiểm thử hệ thống chính:** * **Kiểm thử phục hồi (recovery testing):** * Là các kiểm thử được tiến hành nhằm làm hệ thống ngừng hoạt động và đánh giá khả năng phục hồi sau đó * Với các hệ thống có khả năng phục hồi tự động, chúng ta cần đánh giá các công đoạn tái thiết lập thông số, khả năng khôi phục dữ liệu và tái khởi động * Với các trường hợp đòi hỏi khởi động lại thủ công, chúng ta cần đánh giá thời gian ngừng để sửa chữa (MTTR – Mean Time To Repair) và trong một số trường hợp đánh giá cả chi phí cho việc khôi phục. * **Kiểm thử gây áp lực (stress testing)** * Đây là loại (bước) kiểm thử được tiến hành khi đã có phiên bản làm việc, nhằm tìm hiểu hoạt động của hệ thống trong các trường hợp tải trọng lớn (dữ liệu lớn, số người sử dụng lớn, tài nguyên hạn chế...) * Mục đích của kiểm thử áp lực là:  1. Tìm hiểu giới hạn chịu tải của hệ thống 2. Tìm hiểu về đặc trưng của hệ thống khi đạt và vượt giới hạn chịu tải (khi bị sụp đổ) 3. Ngoài ra kiểm thử áp lực còn nhằm xác định các trạng thái đặc biệt như tổ hợp một số điều kiện dẫn đến sự sụp đổ của hệ thống; tính an toàn của dữ liệu, của dịch vụ khi hệ thống sụp đổ  * **Kiểm thử hiệu suất (performance testing):** * Kiểm thử hiệu suất (performance testing) được thiết kế để đánh giá hiệu suất hoạt động của phần mềm trong một ngữ cảnh cho trước, thông thường là trong một môi trường tích hợp các phần mềm và phần cứng cụ thể * Được tiến hành ở tất cả các công đoạn kiểm thử * Kiểm thử hiệu suất liên quan chặt chẽ đến ngữ cảnh sử dụng bao gồm cả các phần mềm khác (hệ điều hành, CSDL,…) và môi trường phần cứng (CPU, bộ nhớ, mạng) * Kiểm thử hiệu suất thường được tiến hành cùng với kiểm thử áp lực | Tester và User |
| Kiểm thử độ chấp nhận của user (User Acceptance Test) | * Là qui trình so sánh chương trình thực tế với các yêu cầu ban đầu của nó và với các nhu cầu hiện hành của người dùng đầu cuối. * Thường được thực hiện bởi khách hàng hay người dùng đầu cuối và thường không được coi như là 1 trách nhiệm của tổ chức phát triển phần mềm. * Trong trường hợp chương trình làm theo hợp đồng, bên đặt hàng thực hiện kiểm thử độ chấp nhận bằng cách so sánh hoạt động của chương trình với các điều khoản trong hợp đồng. * Trong trường hợp chương trình thương mại như HĐH, trình biên dịch, hệ quản trị CSDL, khách hàng nhạy cảm sẽ thực hiện kiểm thử độ chấp nhận để xác định xem sản phẩm có thỏa mãn các yêu cầu của họ không ? | User |

## 3.2 MÔI TRƯỜNG VÀ YÊU CẦU KIỂM THỬ

* **Yêu cầu về phần cứng:**

|  |  |  |
| --- | --- | --- |
| **YÊU CẦU PHẦN CỨNG** | | |
| **PHẦN CỨNG** | **YÊU CẦU TỔI THIỂU** | **YÊU CẦU ĐỀ XUẤT** |
| Ram | Dung lượng tối thiểu 4GB | Dung lượng trên 8GB |
| CPU | Cấu hình tối thiểu Intel Dual Core 3.0 GHz | Cấu hình đề xuất tối thiểu Intel Quara Core 4.5 GHz |
| Hardware | Yêu cầu dung lượng bộ nhớ trống tối thiểu 80GB | Yêu cầu dung lượng bộ nhớ trống tối thiểu 200GB |

* **Yêu cầu về phần mềm:**

|  |  |  |
| --- | --- | --- |
| **YÊU CẦU PHẦN MỀM** | | |
| **TÊN PHẦN MỀM** | **THÔNG TIN PHẦN MỀM** | **CHI TIẾT PHẦN MỀM** |
| Windows | Hệ điều hành | * **Windows 7:** * Professional 32bit – 64bit * Home Basic 32bit – 64bit * Home Premium 32bit – 64bit * Ultimate 32bit – 64bit * **Windows 8:** * Pro 32bit – 64bit * **Windows 10:** * Home 32bit – 64bit * Pro 32bit – 64bit |
| Linux | Hệ điều hành | * **Kali Linux:** * Kali linux 1.02 – 2.0 32bit – 64bit * **Ubuntu:** * Ubuntu  12.04 * Ubuntu  14.04 |
| MACOS | Hệ điều hành | * OS X v10.5 "Leopard" * OS X v10.6 "Snow Leopard" * OS X v10.7 "Lion" * OS X v10.8 "Mountain Lion |
| Fire Fox | Trình duyệt | Fire fox 3.5 – 3.6 – 52.0 |
| Chorme | Trình duyệt | Chorme 11.0 – 13.0 |
| CocCoc | Trình duyệt | CocCoc 1.0 – 3.0 – 3.1 |
| Apache Tomcat | Servlet | Tomcat 9.0+ |
| JDK | Công cụ phát triển | JDK 8.1 64bit – JDK 17+ 64bit |
| Eclipse | Phần mềm lập trình | Eclipse 2019+ |

* **Yêu cầu về chức năng:**

|  |  |
| --- | --- |
| **Các chức năng yêu cầu kiểm thử** | **Chi tiết chức năng yêu cầu kiểm thử** |
| Kiểm thử khả năng duyệt điều hướng các chức năng theo người dùng  (User Navigation Test) | * Khả năng người dùng login vào hệ thống với quyền hạn thích hợp. * Di chuyển qua các màn hình "thao tác" mong muốn 1 cách đúng đắn và logout khỏi phần mềm. * Điều hướng hướng người dùng hợp lý qua từng cửa sổ chức năng |
| Kiểm thử màn hình giao tác  (Transaction screen Test) | * Màn hình giao tác có các field nhập liệu, list chọn option, các options, các button chức năng (Add, Change, Delete, Submit, Cancel, OK...). * Thiết kế testcase để xác thực hoạt động của mỗi field dữ liệu, list, option và button * Đảm bảo tính hợp lý cũng như bố cục của từng layout |
| Kiểm thử luồng giao tác  (Transaction Flow Test) | * Kiểm tra kết quả tổng hợp của nhiều màn hình giao tác theo thứ tự duyệt đúng có hoàn thành hoạt động nghiệp vụ tương ứng không ? * Kiểu tra các lối tắt đến từng cửa sổ chức năng có hoạt động như mong muốn ? Việc đi tắt có tạo ra một lộ trình thao tác mới không ? |
| Kiểm thử hiển thị thông báo  (Message screen Test) | * Các thông báo và cảnh báo có được đưa ra một cách chính xác và thích hợp hay không ? Có dư thừa hay gây phiền phức cho người dùng không ? * Hình thức đưa ra thông báo hay cảnh báo có thích hợp trong từng trường hợp khác nhau hay không (file , dialog, window…. ) ? |
| Kiểm thử các hình thức báo cáo  (Export Report Test) | * Kiểm thử các khác biệt giữa kết quả hiển thị thông tin trên màn hình và các phương thức hiển thị các thông tin khác (như máy in, file,..). * Kiểm tra tính chính xác và hợp lí giữa các thông tin trên phần mềm và các thông tin được xuất ra theo các dạng khác nhau như word, excel ,file….? |
| Kiểm thử các thao tác dữ liệu trên cơ sở dữ liệu  (Database integrity test) | * Kiểm thử việc thiết kế, khởi tạo database ban đầu thông qua tiện ích bên ngoài phần mềm ứng dụng cần kiểm thử. * Kiểm tra khả năng truy cập và xử lý trên database * Kiểm thử việc phần mềm sử dụng database đã được thiết kế và khởi tạo đúng. * Kiểm tra việc thao tác cũng như truy vấn dữ liệu hoạt động đúng như thiết kế |

**4. THIẾT KẾ VÀ XÂY DỰNG TESTCASE**

## 4.1 Thiết kế testcase

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Chức năng** | **Mô tả** | **Tổng số testcase** | **Manual** | **Automation** |
| 1 | Quản lý nhân viên | Liệt kê danh sách nhân viên | 17 | 10 | 7 |
| Xem thông tin chi tiết của mỗi nhân viên |
| Thêm nhân viên mới |
| Cập nhật thông tin nhân viên |
| Xóa nhân viên đã tồn tại |
| Điều hướng: di chuyển đến dữ liệu của bản ghi chi tiết trước và sau |
| Tìm kiếm: tìm kiếm theo điều kiện |
| 2 | Quản lý chuyên đề | Liệt kê danh sách các chuyên đề | 10 | 10 | 0 |
| Xem thông tin chi tiết của mỗi chuyên đề |
| Tạo chuyên đề mới |
| Cập nhật thông tin chuyên đề |
| Xóa chuyên đề đã tồn tại |
| Điều hướng: di chuyển đến dữ liệu của bản ghi chi tiết trước và sau |
| Tìm kiếm: tìm kiếm theo điều kiện |
| 3 | Quản lý khóa học | Liệt kê danh sách các khóa học | 10 | 10 | 9 |
| Xem thông tin chi tiết của mỗi khóa học |
| Tạo khóa học mới |
| Cập nhật thông tin khóa học |
| Xóa khóa học đã tồn tại |
| Điều hướng: di chuyển đến dữ liệu của bản ghi chi tiết trước và sau |
| Tìm kiếm: tìm kiếm theo điều kiện |
| 4 | Quản lý người học | Liệt kê danh sách các người học | 10 | 10 | 0 |
| Xem thông tin chi tiết của mỗi người học |
| Tạo người học mới |
| Cập nhật thông tin người học |
| Xóa người học đã tồn tại |
| Điều hướng: di chuyển đến dữ liệu của bản ghi chi tiết trước và sau |
| Tìm kiếm: tìm kiếm theo điều kiện |
| 5 | Quản lý học viên | Liệt kê danh sách các học viên | 17 | 10 | 7 |
| Xem thông tin chi tiết của mỗi học viên |
| Thêm học viên mới của khóa học |
| Cập nhật thông tin học viên |
| Xóa học viên đã tồn tại |
| Điều hướng: di chuyển đến dữ liệu của bản ghi chi tiết trước và sau |
| Tìm kiếm: tìm kiếm theo điều kiện |
| 6 | Tổng hợp và thống kê | Doanh thu theo năm | 5 | 5 | 0 |
| Xem thống kê số lượng người học từng chuyên đề theo năm |
| Xem tổng hợp điểm tất cả các chuyên đề . |
| Xem tổng hợp bảng điểm của các khóa học . |
| 7 | Đăng nhập | Cho phép người dùng đăng nhập để sử dụng các chức năng của phần mềm theo vai trò | 6 | 3 | 3 |
| Lưu thông tin lần đăng nhập trước |
| Kiểm tra thông tin đặng nhập của nhân viên (báo lỗi nếu sai thông tin) |
| 8 | Đổi mật khẩu | Cho phép nhân viên hiện đang đăng nhập đổi mật khẩu của họ | 8 | 5 | 3 |

## 4.2 Xây dựng testcase

Tài liệu đính kèm TestCase.xlsx

**Một vài testcase(automation test)**

Folder: Test package

Class : DangNhapJDialogTest.java

Package: com.edusys.ui

Function: isLogin(String manv, String matKhau)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Test Case Description** | **Test Case Procedure** | **Test data** | **Expected Output** | **Inter-test case Dependence** |
| LOG\_FUNC\_1 | Kiểm tra phương thức đăng nhập khi tài khoản không tồn tại | Folder: Test package Class : DangNhapJDialogTest.java Package: com.edusys.ui Function: isLogin(String manv, String matKhau)  Phương thức được truyền vào đối tượng tài khoản nhân viên | Account: Id: 'admin123' Pw: '123' | Test Pass khi Phương thức quăng ra lỗi NullPointerException | Test Pass như kết quả mong muốn |
| LOG\_FUNC\_2 | Kiểm tra phương thức đăng nhập khi tài khoản sai mật khẩu | Folder: Test package Class : DangNhapJDialogTest.java Package: com.edusys.ui Function: isLogin(String manv, String matKhau)  Phương thức được truyền vào đối tượng tài khoản nhân viên | Account: Id: 'admin' Pw: '123' | Test Pass khi Phương thức trả về FALSE | Test Pass như kết quả mong muốn |
| LOG\_FUNC\_3 | Kiểm tra phương thức đăng nhập khi tài khoản hợp lệ | Folder: Test package Class : DangNhapJDialogTest.java Package: com.edusys.ui Function: isLogin(String manv, String matKhau)  Phương thức được truyền vào đối tượng tài khoản nhân viên | Account: Id: 'admin' Pw: '123' | Test Pass khi Phương thức trả về TRUE | Test Pass như kết quả mong muốn |

Folder: Test package  
Class : DoiMatKhauJDialogTest.java  
Package : com.edusys.ui  
Function : String changePass(String oldPw,String newPw,String cfPw)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Test Case Description** | **Test Case Procedure** | **Test data** | **Expected Output** | **Inter-test case Dependence** |
| LOG\_FUNC\_1 | Kiểm tra phương thức đổi mật khẩu khi tài khoản đang đăng nhập là null | Bước 1: Tạo hàm test Bước 2: Truyền dữ liệu  Bước 3: Gọi đến hàm đăng nhập Bước 4: Chạy hàm test | Account: Auth : null OldPw: '123' NewPw: '1234' CfPw: '1234' | Phương thức quăng ra lỗi NullPointerException | Test Pass như kết quả mong muốn |
| LOG\_FUNC\_2 | Kiểm tra phương thức đổi mật khẩu khi mật khẩu cũ không chính xác | Bước 1: Tạo hàm test Bước 2: Truyền dữ liệu  Bước 3: Gọi đến hàm đăng nhập Bước 4: Chạy hàm test | Account: Auth : 'admin' OldPw: '124' NewPw: '123' CfPw: '123' | Phương thức trả về 'Sai mật khẩu' | Test Pass như kết quả mong muốn |
| LOG\_FUNC\_3 | Kiểm tra phương thức đổi mật khẩu khi mật khẩu mới không đúng định dạng | Bước 1: Tạo hàm test Bước 2: Truyền dữ liệu  Bước 3: Gọi đến hàm đăng nhập Bước 4: Chạy hàm test | Account: Auth : 'admin' OldPw: '123456' NewPw: '12' CfPw: '12' | Test Pass khi Phương thức trả về 'Mật khẩu mới phải từ 3 kí tự trở lên' | Test Pass như kết quả mong muốn |
| LOG\_FUNC\_4 | Kiểm tra phương thức đổi mật khẩu khi nhập lại mật khẩu không khớp mật khẩu mới | Bước 1: Tạo hàm test Bước 2: Truyền dữ liệu  Bước 3: Gọi đến hàm đăng nhập Bước 4: Chạy hàm test | Account: Auth : 'admin' OldPw: '123456' NewPw: '123' CfPw: '124' | Phương thức trả về 'Đổi mật khẩu thành công' | Test Pass như kết quả mong muốn |
| LOG\_FUNC\_5 | Kiểm tra phương thức đổi mật khẩu khi thành công | Bước 1: Tạo hàm test Bước 2: Truyền dữ liệu  Bước 3: Gọi đến hàm đăng nhập Bước 4: Chạy hàm test | Account: Auth : 'admin' OldPw: '123456' NewPw: '123' CfPw: '123' | Phương thức trả về 'Sai mật khẩu' | Test Pass như kết quả mong muốn |

Folder: Test package  
Class : DoiMatKhauJDialogTest.java  
Package : com.edusys.ui  
Function : String changePass(String oldPw,String newPw,String cfPw)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Test Case Description** | **Test Case Procedure** | **Test data** | **Expected Output** | **Inter-test case Dependence** |
| QLHV\_FUNC\_1 | Kiểm tra phương thức getXepLoai khi điểm là -1 | Phương thức được truyền vào điểm tương ứng | Điểm : -1 | Phương thức trả về 'Không hợp lệ' | Test Pass như kết quả mong muốn |
| QLHV\_FUNC\_2 | Kiểm tra phương thức getXepLoai khi điểm là 4.3 | Phương thức được truyền vào điểm tương ứng | Điểm : 4.3 | Phương thức trả về 'Chưa đạt' | Test Pass như kết quả mong muốn |
| QLHV\_FUNC\_3 | Kiểm tra phương thức getXepLoai khi điểm là 6.1 | Phương thức được truyền vào điểm tương ứng | Điểm : 6.1 | Phương thức trả về 'Trung bình' | Test Pass như kết quả mong muốn |
| QLHV\_FUNC\_4 | Kiểm tra phương thức getXepLoai khi điểm là 7.9 | Phương thức được truyền vào điểm tương ứng | Điểm : 7.9 | Phương thức trả về 'Khá' | Test Pass như kết quả mong muốn |
| QLHV\_FUNC\_5 | Kiểm tra phương thức getXepLoai khi điểm là 8.7 | Phương thức được truyền vào điểm tương ứng | Điểm : 8.7 | Phương thức trả về 'Giỏi' | Test Pass như kết quả mong muốn |
| QLHV\_FUNC\_6 | Kiểm tra phương thức getXepLoai khi điểm là 9.2 | Phương thức được truyền vào điểm tương ứng | Điểm : 9.2 | Phương thức trả về 'Xuất sắc' | Test Pass như kết quả mong muốn |
| QLHV\_FUNC\_7 | Kiểm tra phương thức getXepLoai khi điểm là 11 | Phương thức được truyền vào điểm tương ứng | Điểm : 11 | Phương thức trả về 'Không hợp lệ' | Test Pass như kết quả mong muốn |

Folder: Test package  
Class : NhanVienDAOTestt.java  
Package : com.edusys.dao  
Function : testInsertUpdateDelete()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Test Case Description** | **Test Case Procedure** | **Test data** | **Expected Output** | **Inter-test case Dependence** |
| QLNV\_FUNC\_1 | Kiểm ra phương thức insert(Object) thành công | Truyền vào đối tượng nhân viên cần thêm | Nhân viên: Mã : "NV1" MK: "123" Tên: "NV1" Admin: true | Khi số bản ghi sau insert tăng lên 1 | Test Pass như kết quả mong muốn |
| QLNV \_FUNC\_2 | Kiểm ra phương thức update(Object) thành công | Truyền vào đối tượng nhân viên cần cập nhật | Nhân viên: Mã : "NV1" MK: "123" Tên: "NV2" Admin: true | Khi thông tin bản ghi thay đổi từ NV1 thành NV2 | Test Pass như kết quả mong muốn |
| QLNV \_FUNC\_3 | Kiểm ra phương thức delete(key) thành công | Truyền vào đối tượng nhân viên cần xóa | Mã: NV1 | Khi số bản ghi sau delete giảm 1 | Test Pass như kết quả mong muốn |
| QLNV \_FUNC\_4 | Kiểm tra phương thức SelectAll() | Gọi phương thức và so sánh với số bản ghi thực tế trong DB | Table NhanVien: record : 6 | Phương thức trả về danh sách mảng có số phần tử trùng với số bản ghi trên DB | Test Pass như kết quả mong muốn |
| QLNV \_FUNC\_5 | Kiểm tra phương thức selectByID() | Gọi phương thức và kiểm tra xem có đối tượng được trả về không | ID : "admin1" | Phương thức trả về một đối tượng | Test Pass như kết quả mong muốn |
| QLNV \_FUNC\_6 | Kiểm tra phương thức SelectBySql() | Gọi phương thức với câu query và so sánh kết quả trả về tương ứng | SQL query: "SELECT \* FROM NhanVien WHERE MaNV=?"  Key: "admin1" | Phương thức trả về một đối tượng | Test Pass như kết quả mong muốn |
| QLNV \_FUNC\_7 | Kiểm tra phương thứcSelectByKey() | Gọi phương thức với từ khóa và so sánh số bản ghi trả về với số bản ghi khớp với từ khóa trong DB | Table NhanVien: keyword: "admin" record : 2 | Phương thức trả về danh sách mảng có số phần tử trùng với số bản ghi trên DB ứng với từ khóa | Test Pass như kết quả mong muốn |

Folder: Test package  
Class : HocVienDAOTest.java  
Package : com.edusys.dao  
Function : testInsertUpdateDelete()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Test Case Description** | **Test Case Procedure** | **Test data** | **Expected Output** | **Inter-test case Dependence** |
| QLHV\_FUNC\_1 | Kiểm ra phương thức insert(Object) thành công | Truyền vào đối tượng nhân viên cần thêm | Học viên: Mã HV: auto Mã KH:3  Mã NH: PS17380  Điểm: 0 | Khi số bản ghi sau insert tăng lên 1 | Test Pass như kết quả mong muốn |
| QLHV\_FUNC\_2 | Kiểm ra phương thức update(Object) thành công | Truyền vào đối tượng nhân viên cần cập nhật | Học viên: Mã HV: auto Mã KH:3  Mã NH: PS17380  Điểm: 8 | Khi thông tin bản ghi thay đổi từ điểm 0 thành điểm 8 | Test Pass như kết quả mong muốn |
| QLHV\_FUNC\_3 | Kiểm ra phương thức delete(key) thành công | Truyền vào đối tượng nhân viên cần xóa | Mã KH:3  Mã NH: PS17380 | Khi số bản ghi sau delete giảm 1 | Test Pass như kết quả mong muốn |
| QLHV\_FUNC\_4 | Kiểm tra phương thức SelectAll() | Gọi phương thức và so sánh với số bản ghi thực tế trong DB | Table NhanVien: record : 230 | Phương thức trả về danh sách mảng có số phần tử trùng với số bản ghi trên DB | Test Pass như kết quả mong muốn |
| QLHV\_FUNC\_5 | Kiểm tra phương thức SelectBySql() | Gọi phương thức với câu query và so sánh kết quả trả về tương ứng | SQL query: "SELECT \* FROM HocVien “  Key: "admin1" | Phương thức trả về một đối tượng | Test Pass như kết quả mong muốn |
| QLHV\_FUNC\_6 | Kiểm tra phương thứcSelectByMaKHMaNH() | Truyền vào mã KH và mã NH | Mã KH:3  Mã NH: PS17380 | Phương thức trả về danh sách mảng có số phần tử trùng với số bản ghi trên DB ứng với từ khóa | Test Pass như kết quả mong muốn |

**5. QUÁ TRÌNH THỰC THI TESTCASE**

## 5.1 UI TEST SUITE

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >

<suite name=*"UIsuite"* parallel=*"false"*>

<test name=*"UIsuite-test"*>

<classes>

<class name=*"com.edusys.ui.DangNhapJDialog"*></class>

<class name=*"com.edusys.ui.DoiMatKhauJDialogTest"*></class>

<class name=*"com.edusys.ui.HocVienJDialogTest"*></class>

</classes>

</test>

### </suite>

### Graphical user interface, application Description automatically generated

### 5.1.1 DangNhapJDialogTest

**Phương thức cần kiểm tra**

//Kiểm tra trạng thái tài khoản

//1.Ném lỗi NullPointerException nếu tài khoản không tồn tại

//2. Trả về FALSE nếu tài mật khẩu không đúng

//3.Trả về TRUE nếu tài khoản xác nhận hợp lệ

**public** **boolean** **isLogin**(String manv, String matKhau) {

NhanVien nhanVien = dao.selectByID(manv);

**if** (nhanVien != **null**) {

String matKhau2 = nhanVien.getMatKhau();

**if** (matKhau.equals(matKhau2)) {

**return** **true**;

} **else** {

**return** **false**;

}

} **else** {

**throw** **new** NullPointerException();

}

}

**Test script**

**public** **class** DangNhapJDialogTest {

**public** **DangNhapJDialogTest**() {

}

//Test case xác nhận tài khoản đăng nhập đúng

@Test

**public** **void** **testIsLogin1**() {

String manv = "admin";

String matKhau = "123456";

System.***out***.println("Account :");

System.***out***.println("ID: " + manv);

System.***out***.println("PW: " + matKhau);

DangNhapJDialog instance = **new** DangNhapJDialog();

**boolean** expResult = **true**;

**boolean** result = instance.isLogin(manv, matKhau);

*assertEquals*(expResult, result);

}

//Test case xác nhận tài khoản đăng nhập sai

@Test

**public** **void** **testIsLogin2**() {

String manv = "admin";

String matKhau = "123";

System.***out***.println("Account :");

System.***out***.println("ID: " + manv);

System.***out***.println("PW: " + matKhau);

DangNhapJDialog instance = **new** DangNhapJDialog();

**boolean** expResult = **false**;

**boolean** result = instance.isLogin(manv, matKhau);

*assertEquals*(expResult, result);

}

//Test case tài khoản đăng nhập không tồn tại

@Test(expectedExceptions = NullPointerException.**class**)

**public** **void** **testIsLogin3**() {

String manv = "admin123";

String matKhau = "123";

System.***out***.println("Account :");

System.***out***.println("ID: " + manv);

System.***out***.println("PW: " + matKhau);

DangNhapJDialog instance = **new** DangNhapJDialog();

**boolean** expResult = **false**;

**boolean** result = instance.isLogin(manv, matKhau);

*assertEquals*(expResult, result);

}

}

**Kết quả thực thi**

**Graphical user interface, text, application

Description automatically generated**

### 5.1.2 DoiMatKhauJDialogTest

**Phương thức cần kiểm tra**

//Phương thức đổi mật khẩu tài khoản đang đăng nhập

**public** String **changePass**(String oldPw,String newPw,String cfPw){

**if**(Auth.*USER*==**null**){

**throw** **new** NullPointerException();

}

**if** (!oldPw.equals(Auth.*USER*.getMatKhau())) {

**return** "Sai mật khẩu";

} **else** **if** (!newPw.matches("[\\w]{3,}")) {

**return** "Mật khẩu mới phải từ 3 kí tự trở lên\n";

}**else** **if** (!newPw.equals(cfPw)) {

**return** "Xác nhận mật khẩu không đúng";

} **else** {

**try** {

Auth.*USER*.setMatKhau(newPw);

nvdao.update(Auth.*USER*);

**return** "Đổi mật khẩu thành công";

} **catch** (Exception ex) {

**return** "Đỏi mật khẩu thất bại";

}

}

}

**Test script**

**public** **class** DoiMatKhauJDialogTest {

**public** **DoiMatKhauJDialogTest**() {

}

//Khôi phục tài khoản admin1 sau mỗi testcase

//Cập nhật người dùng đang đăng nhập của hệ thống

@BeforeMethod

**public** **void** **setUp**() {

**new** NhanVienDAO().update(**new** NhanVien("admin1", "123456", "admin1", **true**));

Auth.*USER* = **new** NhanVienDAO().selectByID("admin1");

}

@AfterMethod

**public** **void** **tearDown**() {

**new** NhanVienDAO().update(**new** NhanVien("admin1", "123456", "admin1", **true**));

}

/\*\*

\* Test of changePass method, of class DoiMatKhauJDialog.

\*/

//Trường hợp kiểm tra khi không có tài khoản đăng nhập

@Test(expectedExceptions = NullPointerException.**class**)

**public** **void** **testChangePass1**() {

Auth.*USER* = **null**;

String oldPw = "123";

String newPw = "1234";

String cfPw = "1234";

DoiMatKhauJDialog instance = **new** DoiMatKhauJDialog(**new** JFrame(), **true**);

String expResult = "";

String result = instance.changePass(oldPw, newPw, cfPw);

*assertEquals*(expResult, result);

}

//Trường hợp kiểm tra khi mật khẩu sai

@Test

**public** **void** **testChangePass2**() {

String oldPw = "124";

String newPw = "123";

String cfPw = "123";

DoiMatKhauJDialog instance = **new** DoiMatKhauJDialog(**new** JFrame(), **true**);

String expResult = "Sai mật khẩu";

String result = instance.changePass(oldPw, newPw, cfPw);

*assertEquals*(expResult, result);

}

//Trường hợp kiểm tra khi mật khẩu không hợp lệ

@Test

**public** **void** **testChangePass3**() {

String oldPw = "123456";

String newPw = "12";

String cfPw = "124";

DoiMatKhauJDialog instance = **new** DoiMatKhauJDialog(**new** JFrame(), **true**);

String expResult = "Mật khẩu mới phải từ 3 kí tự trở lên\n";

String result = instance.changePass(oldPw, newPw, cfPw);

*assertEquals*(expResult, result);

}

//Trường hợp kiểm tra khi xác nhận mật khẩu không khớp với mật khẩu mới

@Test

**public** **void** **testChangePass4**() {

String oldPw = "123456";

String newPw = "123";

String cfPw = "124";

DoiMatKhauJDialog instance = **new** DoiMatKhauJDialog(**new** JFrame(), **true**);

String expResult = "Xác nhận mật khẩu không đúng";

String result = instance.changePass(oldPw, newPw, cfPw);

*assertEquals*(expResult, result);

}

//Trường hợp kiểm tra khi mật khẩu đổi mật khẩu thành công

@Test

**public** **void** **testChangePass5**() {

String oldPw = "123456";

String newPw = "123";

String cfPw = "123";

DoiMatKhauJDialog instance = **new** DoiMatKhauJDialog(**new** JFrame(), **true**);

String expResult = "Đổi mật khẩu thành công";

String result = instance.changePass(oldPw, newPw, cfPw);

*assertEquals*(expResult, result);

}

}

**Kết quả thực thi**

**Graphical user interface, text, application, email

Description automatically generated**

### 5.1.3 HocVienJDialogTest

**Phương thức cần kiểm tra**

//Phương thức trả về xếp loại học viên theo điểm

**public** String **getXepLoai**(**double** diem) {

**if**(diem<0){

**return** "Không hợp lệ";

}

**if** (diem < 5) {

**return** "Chưa đạt";

}

**if** (diem < 6.5) {

**return** "Trung bình";

}

**if** (diem < 8) {

**return** "Khá";

}

**if** (diem < 9) {

**return** "Giỏi";

}

**if** (diem <= 10) {

**return** "Xuất sắc";

}

**return** "Không hợp lệ";

}

**Test script**

**public** **class** HocVienJDialogTest {

HocVienJDialog instance;

//Danh sách các tham số truyền vào mỗi testcase

@DataProvider(name = "input")

**public** **static** Object[][] **input**() {

**return** **new** Object[][]{

{"-1", "Không hợp lệ"},

{"4.3", "Chưa đạt"},

{"6.1", "Trung bình"},

{"7.9", "Khá"},

{"8.7", "Giỏi"},

{"9.2", "Xuất sắc"},

{"11", "Không hợp lệ"}

};

}

/\*\*

\* Test of fillComboBoxChuyenDe method, of class HocVienJDialog.

\*/

@BeforeTest

**public** **void** **init**() {

instance = **new** HocVienJDialog(**new** Frame(), **true**);

}

//Testcase sẽ thực thi tướng ứng với mỗi phần tử trong danh sách tham số

@Test(dataProvider = "input")

**public** **void** **testGetXepLoai**(String diem,String expResult) {

System.***out***.println("Điểm :"+diem);

System.***out***.println("Kết quả mon muốn "+expResult);

String result = instance.getXepLoai(Double.*parseDouble*(diem));

*assertEquals*(expResult, result);

}

}

**Kết quả thực thi**

**Graphical user interface, text, application, email

Description automatically generated**

## 5.2 DAO TEST SUITE

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >

<suite name=*"DAOsuite"* parallel=*"false"*>

<test name=*"DAOsuite-test"*>

<classes>

<class name=*"com.edusys.dao.NhanVienDAOTest"*></class>

<class name=*"com.edusys.dao.HocVienDAOTest"*></class>

</classes>

</test>

</suite>

### 5.2.1 NhanVienDAOTest

**Lớp thực hiện kiểm thử**

**public** **class** NhanVienDAO **extends** DAO<NhanVien, String> {

@Override

**public** **void** **insert**(NhanVien entity) {

String sql = "INSERT INTO NhanVien (MaNV, MatKhau, HoTen, VaiTro) VALUES (?, ?, ?, ?)";

XJdbc.*update*(sql, entity.getMaNV(), entity.getMatKhau(), entity.getHoTen(), entity.isVaiTro());

}

@Override

**public** **void** **update**(NhanVien entity){

String sql = "UPDATE NhanVien SET MatKhau=?, HoTen=?, VaiTro=? WHERE MaNV=?";

XJdbc.*update*(sql, entity.getMatKhau(), entity.getHoTen(), entity.isVaiTro(), entity.getMaNV());

}

@Override

**public** **void** **delete**(String key) {

String sql = "DELETE FROM NhanVien WHERE MaNV=?";

XJdbc.*update*(sql, key);

}

@Override

**public** List<NhanVien> **selectAll**() {

String sql = "SELECT \* FROM NhanVien";

**return** selectBySql(sql);

}

@Override

**public** NhanVien **selectByID**(String keys) {

String sql = "SELECT \* FROM NhanVien WHERE MaNV=?";

**return** selectBySql(sql, keys).isEmpty() ? **null** : selectBySql(sql, keys).get(0);

}

@Override

**protected** List<NhanVien> **selectBySql**(String sql, Object... args) {

List<NhanVien> list = **new** ArrayList<NhanVien>();

**try** {

ResultSet rs = **null**;

**try** {

rs = XJdbc.*query*(sql, args);

**while** (rs.next()) {

NhanVien e = **new** NhanVien();

e.setMaNV(rs.getString(1));

e.setMatKhau(rs.getString(2));

e.setHoTen(rs.getString(3));

e.setVaiTro(rs.getBoolean(4));

list.add(e);

}

} **finally** {

rs.getStatement().getConnection().close();

}

} **catch** (Exception ex) {

**throw** **new** RuntimeException();

}

**return** list;

}

@Override

**public** List<NhanVien> **selectByKey**(String keys) {

String sql = "SELECT \* FROM NhanVien WHERE MaNV LIKE ? OR HoTen LIKE ?";

keys= "%"+keys+"%";

**return** selectBySql(sql, keys, keys);

}

}

**Test script**

**public** **class** NhanVienDAOTest {

**public** **NhanVienDAOTest**() {

}

//Testcase

//Thêm một bản ghi mới vào table

//Thõa mãn nếu số lượng bản ghi +1 sau insert

@Test(priority = 5)

**public** **void** **testInsert**() {

NhanVien entity = **new** NhanVien("NV1", "123", "NV1", **true**);

NhanVienDAO instance = **new** NhanVienDAO();

**int** expResult = instance.selectAll().size()+1;

instance.insert(entity);

**int** result = instance.selectAll().size();

*assertTrue*(expResult==result);

}

//Testcase

//Cập nhật một bản ghi mới vào table

//Thõa mãn nếu thông tin trong bản ghi khớp với thông tin cập nhật

@Test(priority = 6)

**public** **void** **testUpdate**() {

NhanVien entity = **new** NhanVien("NV1", "123", "NV1", **true**);

NhanVienDAO instance = **new** NhanVienDAO();

entity.setHoTen("NV2");

instance.update(entity);

String expResult1 = "NV2";

String result1 = instance.selectByID(entity.getMaNV()).getHoTen();

*assertEquals*(expResult1,result1);

}

//Testcase

//Xóa một bản ghi mới vào table

//Thõa mãn nếu số lượng bản ghi -1 sau delete

@Test(priority = 7)

**public** **void** **testDelete**() {

NhanVien entity = **new** NhanVien("NV1", "123", "NV1", **true**);

NhanVienDAO instance = **new** NhanVienDAO();

**int** expResult2 = instance.selectAll().size()-1;

instance.delete(entity.getMaNV());

**int** result2 = instance.selectAll().size();

*assertTrue*(expResult2==result2);

}

/\*\*

\* Test of selectAll method, of class NhanVienDAO.

\*/

//Testcase

//Truy vấn tất cả bản ghi trong bảng NhanVien

//Thõa mãn nếu số lượng bản ghi đúng với số lượng thực tế trong DB

@Test(priority = 2)

**public** **void** **testSelectAll**() {

System.***out***.println("selectAll");

NhanVienDAO instance = **new** NhanVienDAO();

**int** expResult = 6;

**int** result = instance.selectAll().size();

*assertEquals*(expResult, result);

}

/\*\*

\* Test of selectByID method, of class NhanVienDAO.

\*/

//Testcase

//Truy vấn bản ghi theo ID trong bảng NhanVien

//Thõa mãn nếu bản ghi trả về khác null

@Test(priority = 3)

**public** **void** **testSelectByID**() {

System.***out***.println("selectByID");

String keys = "admin1";

NhanVienDAO instance = **new** NhanVienDAO();

NhanVien result = instance.selectByID(keys);

*assertFalse*(result==**null**);

}

/\*\*

\* Test of selectBySql method, of class NhanVienDAO.

\*/

//Testcase

//Truy vấn bản ghi theo câu lệnh SQL trong bảng NhanVien

//Thõa mãn nếu bản ghi trả về khớp với kết quả mong muốn (1)

@Test(priority = 1)

**public** **void** **testSelectBySql**() {

System.***out***.println("selectBySql");

String sql = "SELECT \* FROM NhanVien WHERE MaNV=?";

Object[] args = **new** Object[]{"admin1"};

NhanVienDAO instance = **new** NhanVienDAO();

List<NhanVien> result = instance.selectBySql(sql, args);

*assertTrue*(result.size()==1);

}

/\*\*

\* Test of selectByKey method, of class NhanVienDAO.

\*/

//Testcase

//Truy vấn các bản ghi theo từ khóa trong bảng NhanVien

//Thõa mãn nếu bản ghi trả về bằng số lượng ước tính (2)

@Test(priority = 4)

**public** **void** **testSelectByKey**() {

System.***out***.println("selectByKey");

String keys = "admin";

NhanVienDAO instance = **new** NhanVienDAO();

**int** expResult = 2;

**int** result = instance.selectByKey(keys).size();

*assertTrue*(expResult== result);

}

}

**Graphical user interface, text, application, email

Description automatically generated**

### 5.2.2 HocVienDAOTest

**Lớp thực hiện kiểm thử**

**public** **class** NhanVienDAO **extends** DAO<NhanVien, String> {

@Override

**public** **void** **insert**(NhanVien entity) {

String sql = "INSERT INTO NhanVien (MaNV, MatKhau, HoTen, VaiTro) VALUES (?, ?, ?, ?)";

XJdbc.*update*(sql, entity.getMaNV(), entity.getMatKhau(), entity.getHoTen(), entity.isVaiTro());

}

@Override

**public** **void** **update**(NhanVien entity){

String sql = "UPDATE NhanVien SET MatKhau=?, HoTen=?, VaiTro=? WHERE MaNV=?";

XJdbc.*update*(sql, entity.getMatKhau(), entity.getHoTen(), entity.isVaiTro(), entity.getMaNV());

}

@Override

**public** **void** **delete**(String key) {

String sql = "DELETE FROM NhanVien WHERE MaNV=?";

XJdbc.*update*(sql, key);

}

@Override

**public** List<NhanVien> **selectAll**() {

String sql = "SELECT \* FROM NhanVien";

**return** selectBySql(sql);

}

@Override

**public** NhanVien **selectByID**(String keys) {

String sql = "SELECT \* FROM NhanVien WHERE MaNV=?";

**return** selectBySql(sql, keys).isEmpty() ? **null** : selectBySql(sql, keys).get(0);

}

@Override

**protected** List<NhanVien> **selectBySql**(String sql, Object... args) {

List<NhanVien> list = **new** ArrayList<NhanVien>();

**try** {

ResultSet rs = **null**;

**try** {

rs = XJdbc.*query*(sql, args);

**while** (rs.next()) {

NhanVien e = **new** NhanVien();

e.setMaNV(rs.getString(1));

e.setMatKhau(rs.getString(2));

e.setHoTen(rs.getString(3));

e.setVaiTro(rs.getBoolean(4));

list.add(e);

}

} **finally** {

rs.getStatement().getConnection().close();

}

} **catch** (Exception ex) {

**throw** **new** RuntimeException();

}

**return** list;

}

@Override

**public** List<NhanVien> **selectByKey**(String keys) {

String sql = "SELECT \* FROM NhanVien WHERE MaNV LIKE ? OR HoTen LIKE ?";

keys= "%"+keys+"%";

**return** selectBySql(sql, keys, keys);

}

}

**Test script**

**public** **class** HocVienDAOTest {

//Testcase

//Thêm một bản ghi mới vào table

//Thõa mãn nếu số lượng bản ghi +1 sau insert

@Test(priority = 4)

**public** **void** **testInsert**() {

HocVien entity = **new** HocVien(111,3,"PS17380",0);

HocVienDAO instance = **new** HocVienDAO();

**int** expResult = instance.selectAll().size()+1;

instance.insert(entity);

**int** result = instance.selectAll().size();

*assertTrue*(expResult==result);

}

//Testcase

//Cập nhật một bản ghi mới vào table

//Thõa mãn nếu thông tin trong bản ghi khớp với thông tin cập nhật

@Test(priority = 5)

**public** **void** **testUpdate**() {

HocVienDAO instance = **new** HocVienDAO();

HocVien entity = instance.selectByMaNHvaMaKH("PS17380", 3);

entity.setDiem(8);

instance.update(entity);

**double** expResult1 = 8;

**double** result1 = (instance.selectByMaNHvaMaKH("PS17380", 3)).getDiem();

*assertTrue*(expResult1==result1);

}

//Testcase

//Xóa một bản ghi mới vào table

//Thõa mãn nếu số lượng bản ghi -1 sau delete

@Test(priority = 6)

**public** **void** **testDelete**() {

HocVienDAO instance = **new** HocVienDAO();

HocVien entity = instance.selectByMaNHvaMaKH("PS17380", 3);

**int** expResult2 = instance.selectAll().size()-1;

instance.delete(entity.getMaHV());

**int** result2 = instance.selectAll().size();

*assertTrue*(expResult2==result2);

}

/\*\*

\* Test of selectAll method, of class HocVienDAO.

\*/

//Testcase

//Truy vấn tất cả bản ghi trong bảng HocVien

//Thõa mãn nếu số lượng bản ghi đúng với số lượng thực tế trong DB

@Test(priority = 2)

**public** **void** **testSelectAll**() {

System.***out***.println("selectAll");

HocVienDAO instance = **new** HocVienDAO();

**int** expResult = 230;

**int** result = instance.selectAll().size();

System.***out***.println(instance.selectAll().size());

*assertTrue*(expResult>0);

}

/\*\*

\* Test of selectBySql method, of class HocVienDAO.

\*/

//Testcase

//Truy vấn bản ghi theo câu lệnh SQL trong bảng NhanVien

//Thõa mãn nếu bản ghi trả về khớp với kết quả mong muốn (1)

@Test(priority = 1)

**public** **void** **testSelectBySql**() {

System.***out***.println("selectBySql");

String sql = "SELECT \* FROM HocVien WHERE MaHV=?";

Object[] args = **new** Object[]{"1"};

HocVienDAO instance = **new** HocVienDAO();

List<HocVien> result = instance.selectBySql(sql, args);

*assertTrue*(result.size()==1);

}

/\*\*

\* Test of selectByKey method, of class HocVienDAO.

\*/

//Testcase

//Truy vấn các bản ghi theo từ khóa trong bảng HocVien

//Thõa mãn nếu bản ghi trả về khác null

@Test(priority = 3)

**public** **void** **testSelectByNHKH**() {

System.***out***.println("selectByNHKH");

HocVienDAO instance = **new** HocVienDAO();

HocVien entity = instance.selectByMaNHvaMaKH("PS11478", 1);

*assertTrue*(entity!=**null**);

}

}

**Graphical user interface, text, application

Description automatically generated**

## 5.3 WEB TEST SUITE

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >

<suite name=*"UIsuite"* parallel=*"false"*>

<test name=*"UIsuite-test"*>

<classes>

<class name=*"com.edusys.ui.DangNhapJDialog"*></class>

<class name=*"com.edusys.ui.DoiMatKhauJDialogTest"*></class>

<class name=*"com.edusys.ui.HocVienJDialogTest"*></class>

</classes>

</test>

</suite>

**Testcripts**

**public** **class** DangNhapPageTest {

**public** WebDriver driver;

String user\_name = "lucdtps17380@fpt.edu.vn";

String pass\_word = "UT<uG4n;VR.<QG3";

Map<String, Object[]> NgResult;

**public** String url = "https://www.w3schools.com/";; // địa chỉ trang web cần kiểm tra

**public** String diverPath = "C:\\Users\\Admin\\Downloads\\Compressed\\chromedriver.exe"; // đường dẫn đến driver của google chrome trên máy

HSSFWorkbook workbook;

HSSFSheet sheet;

@BeforeClass(alwaysRun = **true**)

**public** **void** **suiteSetup**() {

workbook = **new** HSSFWorkbook();

sheet = workbook.createSheet("TestNG Result Summary");

NgResult = **new** LinkedHashMap<String, Object[]>();

NgResult.put("1", **new** Object[] { "Test Step No.", "Action", "Expected Output", "Actual Output" });

**try** {

System.*setProperty*("webdriver.chrome.driver", diverPath);// setProperty cho phép thiết lập driver để remote trình duyệt google chrome được sử dụng

driver = (WebDriver) **new** ChromeDriver();// Khởi tạo đối tượng driver

driver.manage().timeouts().**~~implicitlyWait~~**(5, TimeUnit.***MILLISECONDS***);

} **catch** (Exception e) {

**throw** **new** IllegalStateException("Can't start webdriver");

}

}

@Test(priority = 1)

**public** **void** **launchWebsite**() {

**try** {

driver.get(url);

driver.manage().window().maximize();

driver.findElement(By.*id*("w3loginbtn"))

.click();

NgResult.put("2", **new** Object[] { "1.", "Truy cập website", "Trang web đã được truy cập", "Pass" });

} **catch** (Exception e) {

// **TODO**: handle exception

NgResult.put("2", **new** Object[] { "1.", "Truy cập website", "Trang web đã được truy cập", "Fail" });

Assert.*assertTrue*(**false**);

}

}

@Test(priority = 2)

**public** **void** **FillLogin**() **throws** Exception {

NgResult.put("3", **new** Object[] { "2.", "Nhập dữ liệu để đăng nhập", "Biểu mẫu đăng nhập được điền", "Fail" });

WebElement username = driver.findElement(By.*name*("email"));

username.sendKeys(user\_name);

WebElement password = driver.findElement(By.*name*("current-password"));

password.sendKeys(pass\_word);

Thread.*sleep*(1000);

String us = username.getAttribute("value");

String ps = password.getAttribute("value");

*assertTrue*(us.contentEquals(user\_name) && ps.contentEquals(pass\_word));

NgResult.put("3", **new** Object[] { "2.", "Nhập dữ liệu để đăng nhập", "Biểu mẫu đăng nhập được điền", "Pass" });

}

@Test(priority = 3)

**public** **void** **DoLogin**() **throws** Exception {

NgResult.put("5",

**new** Object[] { "4.", "Bấm đăng nhập và điều hướng tới trang chủ", "Đăng nhập thành công", "Fail" });

Thread.*sleep*(1000);

driver.findElement(By.*name*("current-password")).sendKeys(Keys.***ENTER***);

Thread.*sleep*(10000);

System.***out***.println(driver.getTitle());

*assertTrue*(driver.getTitle().equals("My learning | W3Schools"));

NgResult.put("5",

**new** Object[] { "4.", "Bấm đăng nhập và điều hướng tới trang chủ", "Đăng nhập thành công", "Pass" });

}

@AfterClass

**public** **void** **suiteTearDown**() {

// Lấy bộ key của testcaseResult

Set<String> keyset = NgResult.keySet();

**int** rownumber = 0;

**for** (String key : keyset) {

Row row = sheet.createRow(rownumber++);// Tạo hàng

Object[] objArr = NgResult.get(key);

**int** cellnumber = 0;

// Tạo cột và ghi dữ liệu từ testcaseResult

**for** (Object obj : objArr) {

Cell cell = row.createCell(cellnumber++);

**if** (obj **instanceof** Date)

cell.setCellValue((java.util.Date) obj);

**if** (obj **instanceof** Boolean)

cell.setCellValue((Boolean) obj);

**if** (obj **instanceof** String)

cell.setCellValue((String) obj);

**if** (obj **instanceof** Double)

cell.setCellValue((Double) obj);

}

}

driver.close();

**try** {

FileOutputStream fos = **new** FileOutputStream(**new** File("TestCaseAssignment.xls"));// Tạo luồng mới

workbook.write(fos);// Ghi file

fos.close();

System.***out***.println("Successfully");

} **catch** (FileNotFoundException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

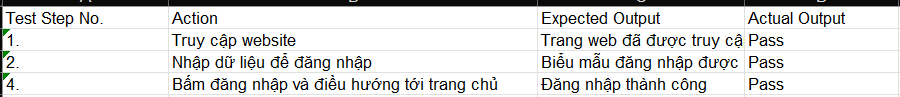
}

}

}

**Kết quả thực thi**

**Graphical user interface, text, application

Description automatically generated**

## 5.4 ALL-SUITE

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >

<suite name=*"main-suite"* parallel=*"false"*>

<suite-files>

<suite-file path=*"./DaoTestSuite.xml"*></suite-file>

<suite-file path=*"./UITestSuite.xml"*></suite-file>

<suite-file path=*"./LoginPageWebSuite.xml"*></suite-file>

</suite-files>

</suite>

**Kết quả thực thi**

Graphical user interface, text, application

Description automatically generatedA picture containing background pattern

Description automatically generated

# 6. TEST REPORT VÀ DEFECT REPORT

## TEST REPORT

### 6.1.1 Tổng quan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Functions** | **% TCS Executed** | **%TCS Passed** | **Number of defects** |
| 1 | Giao diện chính | 100% | 80% | 1 |
| 2 | Quản lý nhân viên | 100% | 100% | 0 |
| 3 | Quản lý chuyên đề | 100% | 100% | 0 |
| 4 | Quản lý khóa học | 100% | 100% | 0 |
| 5 | Quản lý người học | 100% | 100% | 0 |
| 6 | Quản lý học viên | 100% | 86% | 2 |
| 7 | Tổng hợp và thống kê | 100% | 100% | 0 |
| 8 | Đăng nhập | 100% | 100% | 0 |
| 9 | Đổi mật khẩu | 100% | 100% | 0 |

Chart

Description automatically generatedChart

Description automatically generated with low confidenceChart

Description automatically generated with low confidence

### 6.1.2 Automation test result

Table

Description automatically generated

Chart, treemap chart

Description automatically generated

## 6.2 DEFECT REPORT

**Defect Report: #Test1**

|  |  |
| --- | --- |
| Bug ID | #Test1 |
| Tester | Lãnh Quốc Khánh |
| Date | 25/03/2022 |
| Title | Đóng cửa sổ, ấn NO cửa sổ vẫn đóng |
| Defect Description | |
| Package | Com.edusys.ui |
| Summary | Khi đăng nhập vào ứng dụng, bấm dấu X trên góc màn hình, màn hình hiển thị thông báo “Có thoát hay không”, khi bấm No, thì chương trình vẫn thoát |
| Screenshot | Graphical user interface  Description automatically generated |
| Platform | Windows 10 |
| Class | MainJFrame |

Additional Notes

Step by Step:

1. Mở ứng dụng
2. Đăng nhập tài khoản admin với tên tài khoản “admin” mật khẩu “123456”
3. Chọn “OK” tại hộp thoại “Đăng nhập thành công”
4. Chọn X trên góc trái màn hình
5. Hiển thị hộp thoại “Bạn có muốn kết thúc phiên làm việc này” . Chọn “No”

Result –Should : Không kết thúc chương trình, ở lại màn hình trang chủ

Resut-Is: Kết thúc chương trình

**Defect Report: #Test2**

|  |  |
| --- | --- |
| Defect ID | #Test2 |
| Tester | Nguyễn Phạm Hùng Phi |
| Date | 25/03/2022 |
| Title | Truy vấn học viên trong chuyên đề |
| Defect Description | |
| Package | Com.edusys.ui |
| Summary | Chọn Chuyên đề không có học viên tham gia, nhưng vẫn trả về danh sách học viên của chuyên đề khác |
| Screenshot |  |
| Platform | Windows 10 |
| Class | MainJFrame |

Additional Notes

Step by step

* 1. Khởi động ứng dụng
  2. Đăng Nhập
  3. Chọn quản lý học viên
  4. Thay đổi lựa chọn chuyên đề không có học viên
  5. Hiển thị thông báo lỗi truy vấn

Result - should Khi thay đổi lựa chọn chuyên đề không có học viên thì không hiển thị

Result - Is Hiển thị danh sách học viên của chuyên đề khác

**Defect Report: #Test3**

|  |  |
| --- | --- |
| Defect ID | #Test3 |
| Tester | Lãnh Quốc Khánh |
| Date | 25/03/2022 |
| Title | Khi thêm mới học viên , cập nhật điểm sẽ thành công khi chưa nhập điểm của học viên vừa thêm vào |
| Defect Description | |
| Package | Com.edusys.ui |
| Summary | Khi thêm mới học viên , cập nhật điểm sẽ thành công khi chưa nhập điểm của học viên vừa thêm vào |
| Screenshot | Graphical user interface, application  Description automatically generated |
| Platform | Windows 10 |
| Class | MainJFrame |

Additional Notes

Step by Step:

1. Mở ứng dụng
2. Đăng nhập tài khoản admin với tên tài khoản “admin” mật khẩu “123456”
3. Chọn “OK” tại hộp thoại “Đăng nhập thành công”
4. Chọn người học
5. Chọn “Trần Nhật Duy” thêm vào khoá học
6. Qua tab Học Viên nhấn cập nhật

Result –Should : Hiển thị thông báo “Bạn phải nhập điểm từ 0 -10”

Resut-Is: Không hiện thông báo